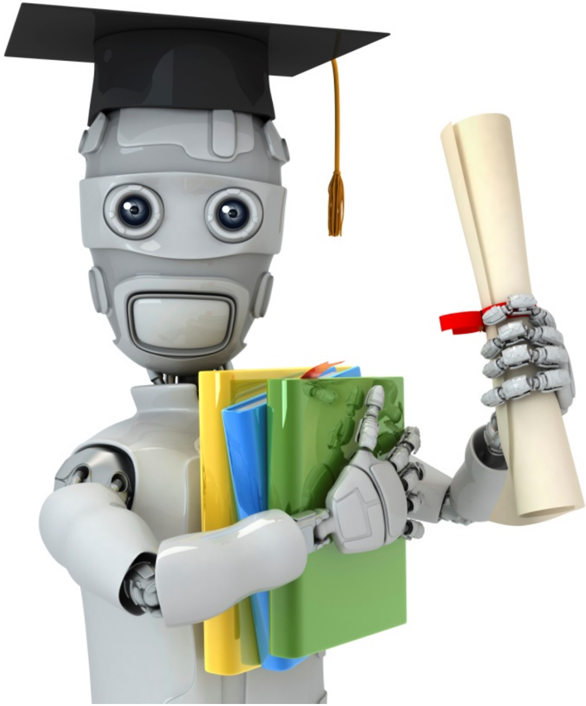


Machine Learning

Introduction

Welcome

Apple - iPhoto - New full-s
www.apple.com/ilife/iphoto/
Store
Mac
iPod
iPhone
iPad
iTunes
Support
iLife '11
iPhoto
iMovie
GarageBand
Video Showcase
Resources
Upgrade Now
iPhoto '11
From your Facebook Wall to your coffee table to your best friend's inbox (or mailbox). Do more with your photos than you ever thought possible. And do it all in one place. iPhoto.
Watch the iPhoto video
MacBook Pro
What's New in iPhoto
What is iPhoto?


SPAM

Machine Learning

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

Examples:

- Database mining
  Large datasets from growth of automation/web.
  E.g., Web click data, medical records, biology, engineering
- Applications can't program by hand.
  E.g., Autonomous helicopter, handwriting recognition, most of Natural Language Processing (NLP), Computer Vision.

Machine Learning

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

Examples:

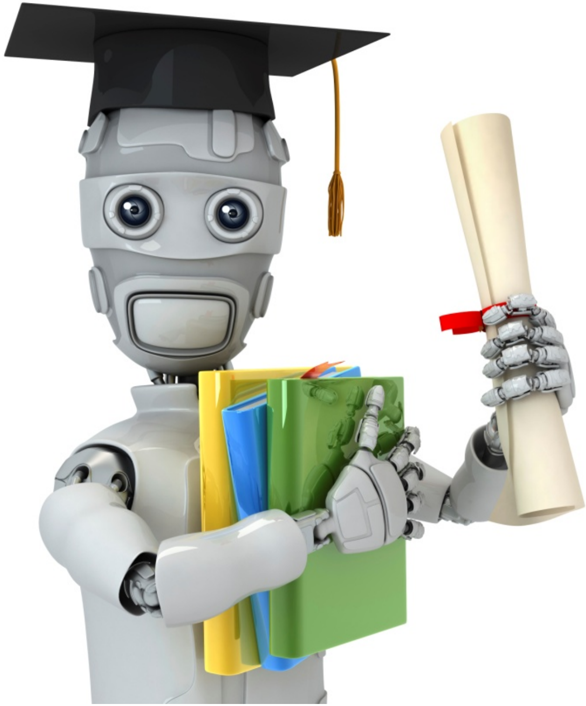
- Database mining

  Large datasets from growth of automation/web.

  E.g., Web click data, medical records, biology, engineering
- Applications can't program by hand.

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

  E.g., Amazon, Netflix product recommendations
- Understanding human learning (brain, real AI).



Machine Learning

Introduction

What is machine learning

Machine Learning definition

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

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

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

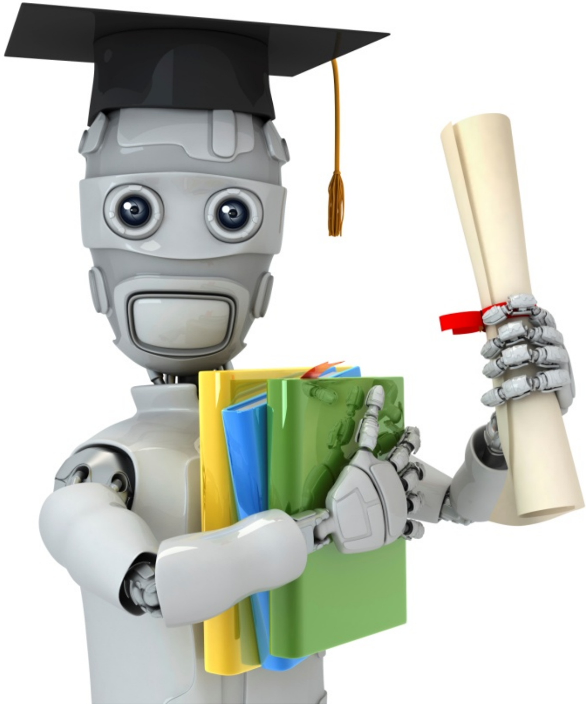
- ◯ Classifying emails as spam or not spam. T ←
- ◯ Watching you label emails as spam or not spam. E ←
- ◯ The number (or fraction) of emails correctly classified as spam/not spam.
- ◯ None of the above—this is not a machine learning problem. P ←

Machine learning algorithms:

- Supervised learning
- Unsupervised learning

Others: Reinforcement learning, recommender systems.

Also talk about: Practical advice for applying learning algorithms.

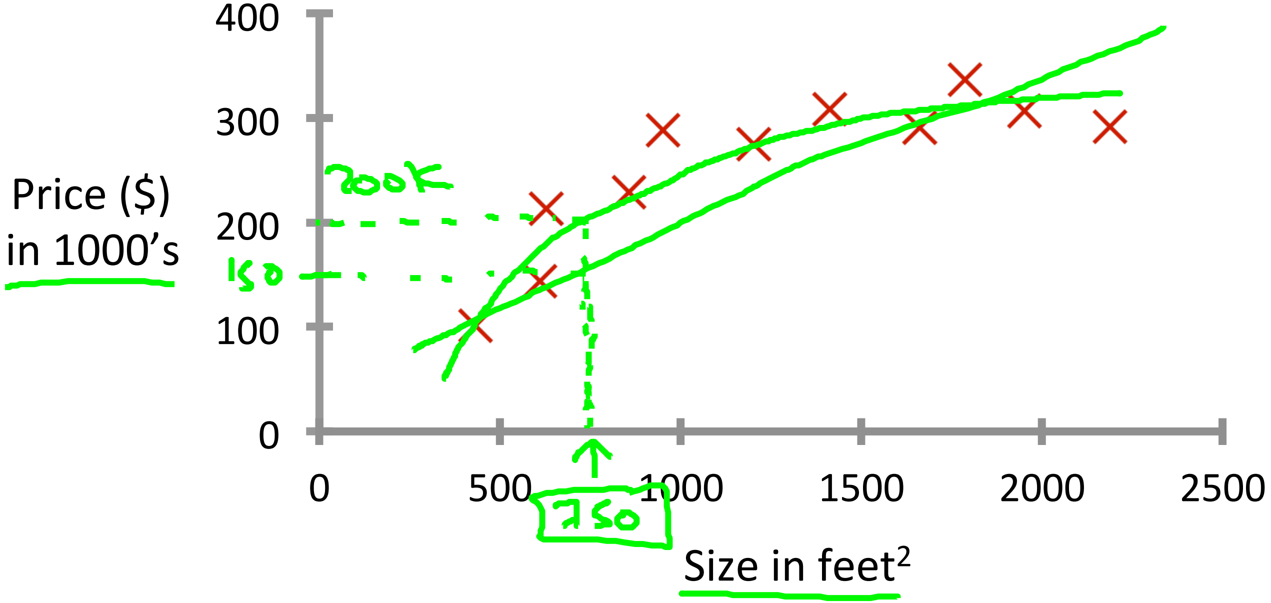


Machine Learning

Introduction

Supervised Learning

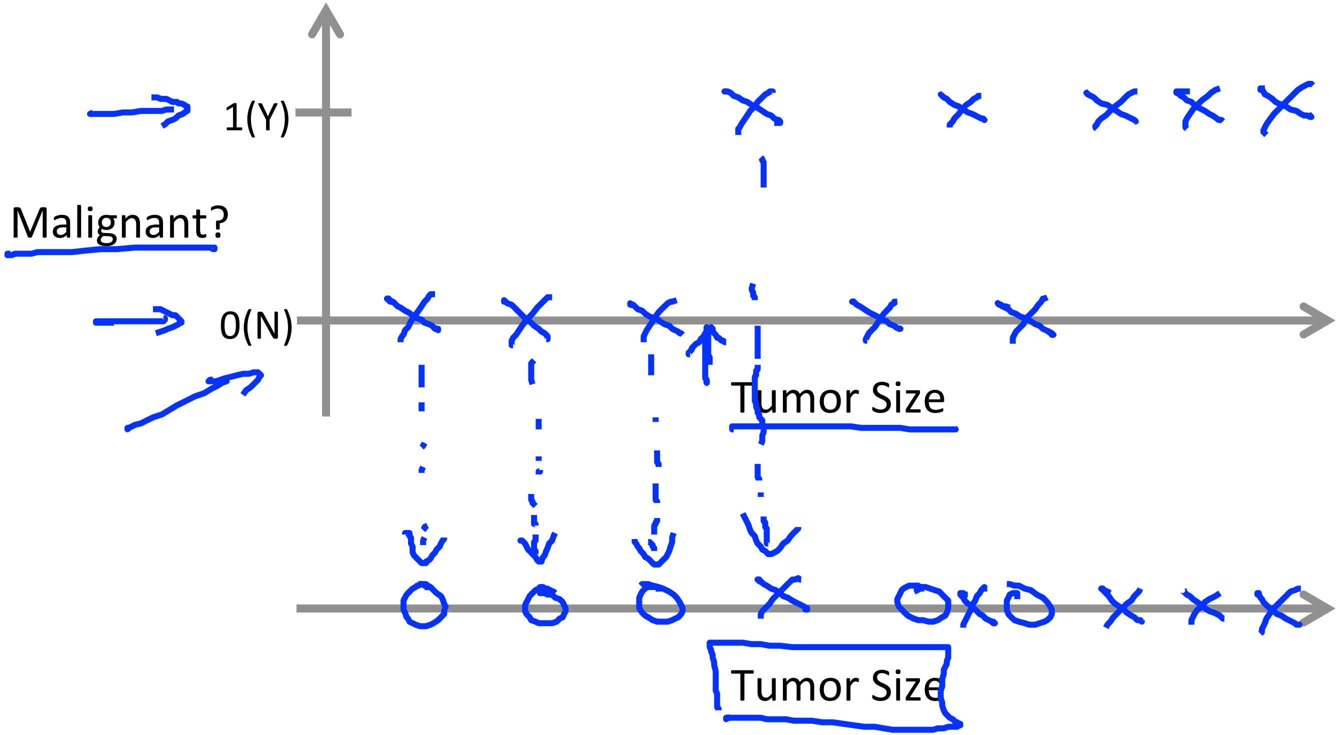
Housing price prediction.



Supervised Learning
'right answers' given

Regression: Predict continuous valued output (price)

Breast cancer (malignant, benign)



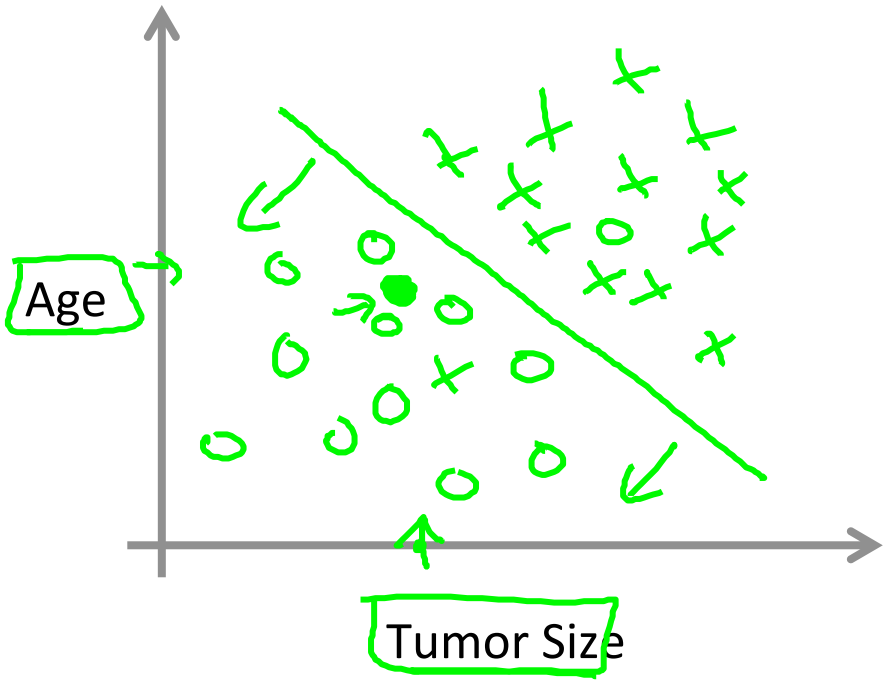
Classification

Discrete valued output (0 or 1)

0, 1, 2, 3

benign

type 1 cancer



- 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.

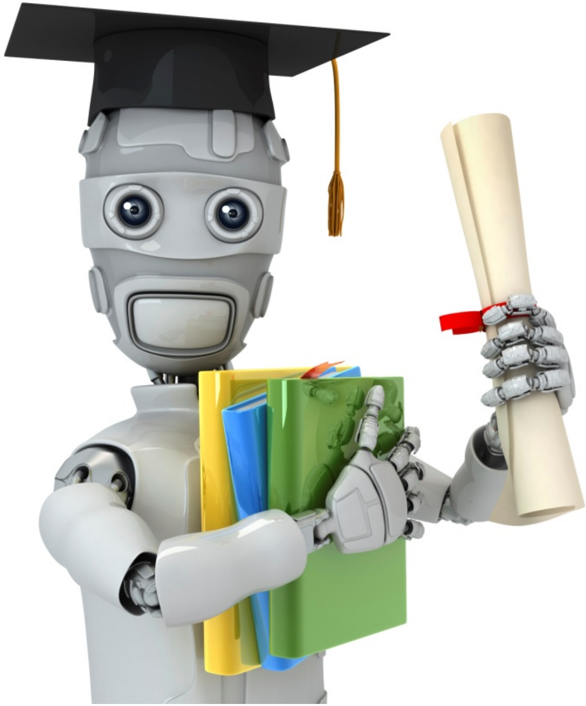
Problem 2: You'd like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised.

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.

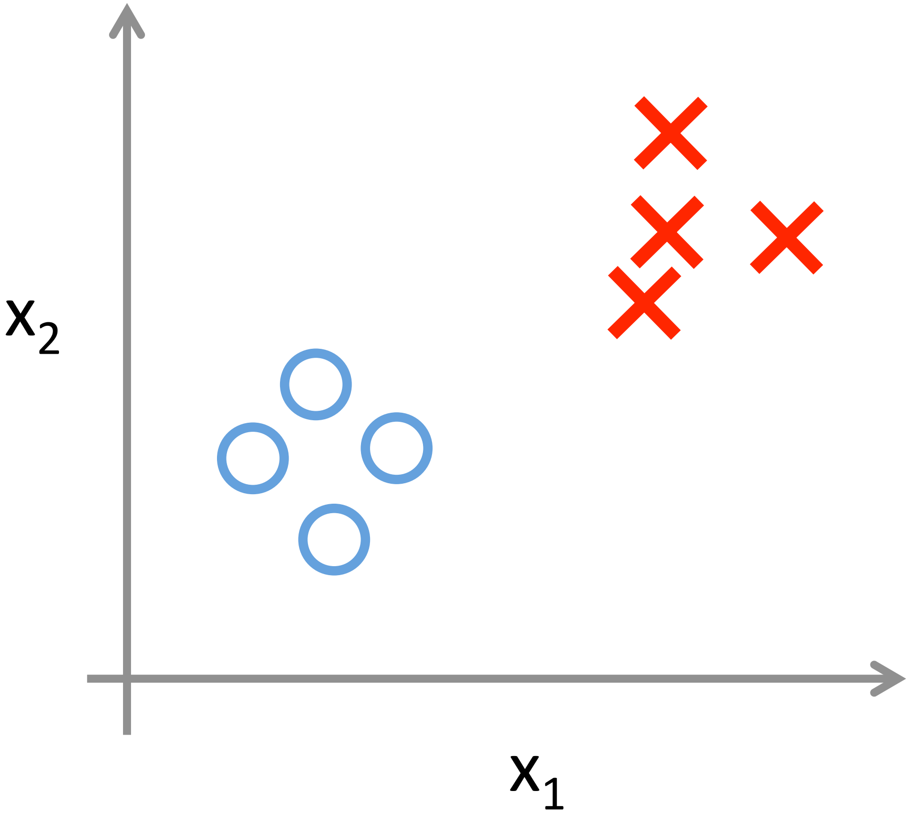


Machine Learning

Introduction

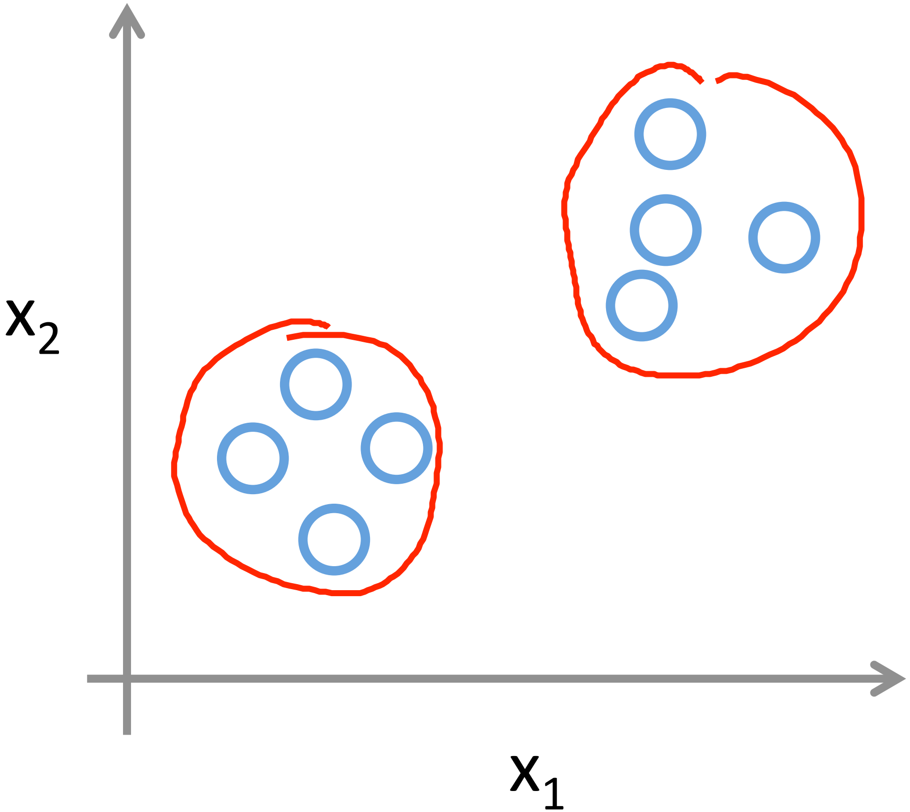
Unsupervised Learning

Supervised Learning

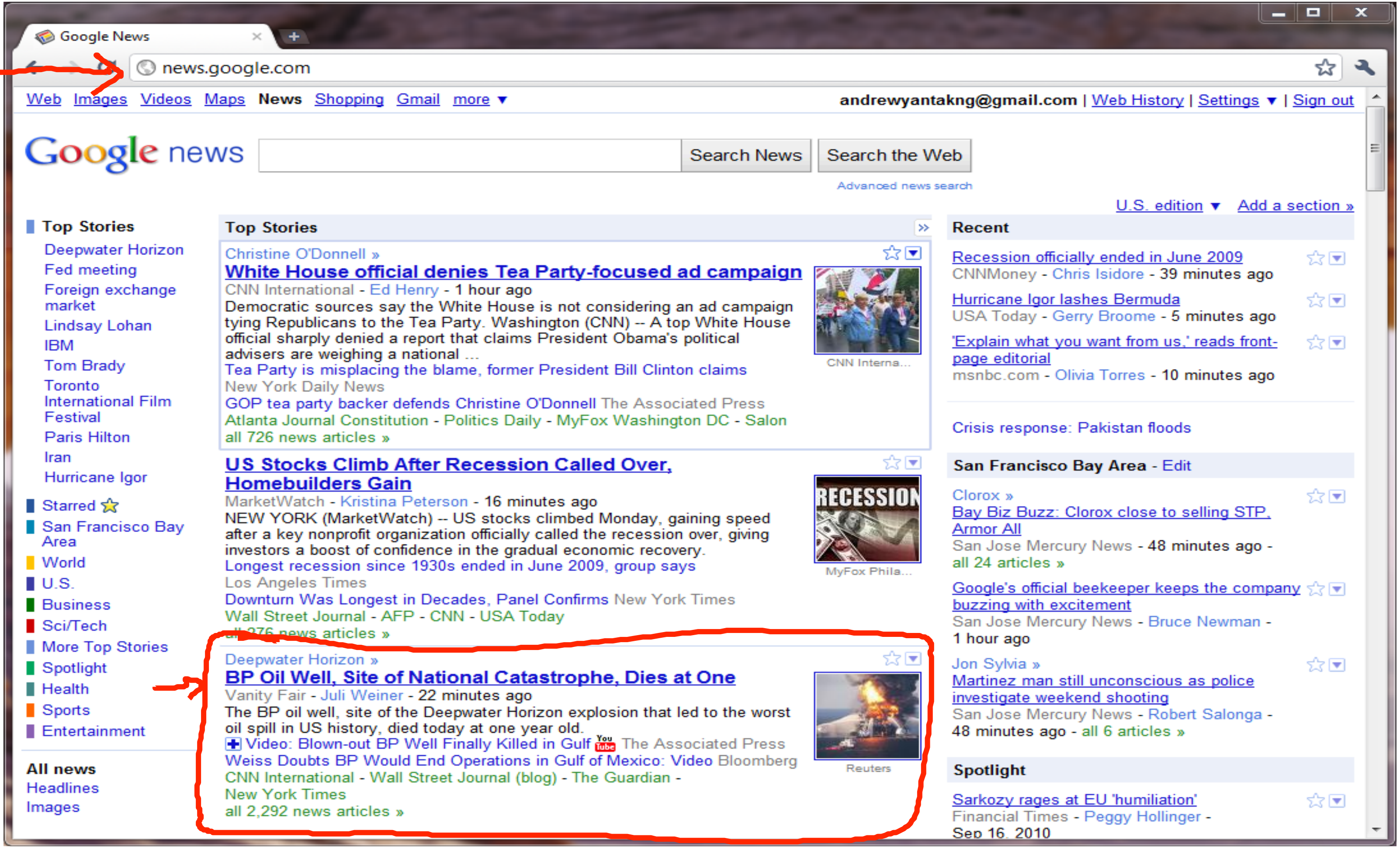


Andrew Ng

Unsupervised Learning



Andrew Ng


Google News
news.google.com
Web Images Videos Maps News Shopping Gmail more
andrewyantakng@gmail.com | Web History | Settings | Sign out
Google news
Search News
Search the Web
Advanced news search
U.S. edition
Add a section »
Top Stories
Deepwater Horizon
Fed meeting
Foreign exchange market
Lindsay Lohan
IBM
Tom Brady
Toronto International Film Festival
Paris Hilton
Iran
Hurricane Igor
Starred
San Francisco Bay Area
World
U.S.
Business
Sci/Tech
More Top Stories
Spotlight
Health
Sports
Entertainment
All news
Headlines
Images
Top Stories
Christine O'Donnell »
White House official denies Tea Party-focused ad campaign
CNN International - Ed Henry - 1 hour ago
Democratic sources say the White House is not considering an ad campaign tying Republicans to the Tea Party. Washington (CNN) -- A top White House official sharply denied a report that claims President Obama's political advisers are weighing a national ...
Tea Party is misplacing the blame, former President Bill Clinton claims
New York Daily News
GOP tea party backer defends Christine O'Donnell The Associated Press
Atlanta Journal Constitution - Politics Daily - MyFox Washington DC - Salon
all 726 news articles »
CNN Interna...
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 »
RECESSION
MyFox Phila...
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 »
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


BP Kills Macondo, But Its ...

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 | All Things Digital. | FINS | SmartMoney | More | SEARCH

THE WALL STREET JOURNAL.

Log In • Register For Free • Subscribe Now, Get 2 Weeks Free

« PREVIOUS

THE SOURCE

NEXT »

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

Search The Source | SEARCH

Article | Comments (2)

THE SOURCE HOME PAGE »

Email | Print | Permalink | Like 2 | + More | Text

By James Herron

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

The most immediate worry for BP and its shareholders is how the authorities will apportion blame for the spill. BP's own investigation spread responsibility across

Associated Press

Fire boat response crews battled the blazing remnants of the off shore oil rig Deepwater Horizon on April 21, 2010.

About The Source

Follow Us:

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

Most Recent

Articles | Comments

1. Who Needs Plaza II Anyway
2. Will Banks Be Forced to Split Retail And Banking Arms?
3. Timing of Ratings Award Intriguing
4. BP Kills Macondo, But Its Legacy Lives On
5. We Already Need a Sequel to Basel III

Allen: Well is dead, but m... ×

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

EDITION: INTERNATIONAL | U.S. | MÉXICO | ARABIC

Set edition preference

CNN

Sign up | Log in

SEARCH

POWERED BY Google

Home | Video | World | U.S. | Africa | Asia | Europe | Latin America | Middle East | Business | World Sport | Entertainment | Tech | Travel | iReport

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

By **the CNN Wire Staff**

September 20, 2010 -- Updated 1317 GMT (2117 HKT)

Mixx | Twitter | Share | Email | Save | Print

Recommend | Be the first of your friends to recommend this.

What next for Gulf oil spill?

Most Popular »

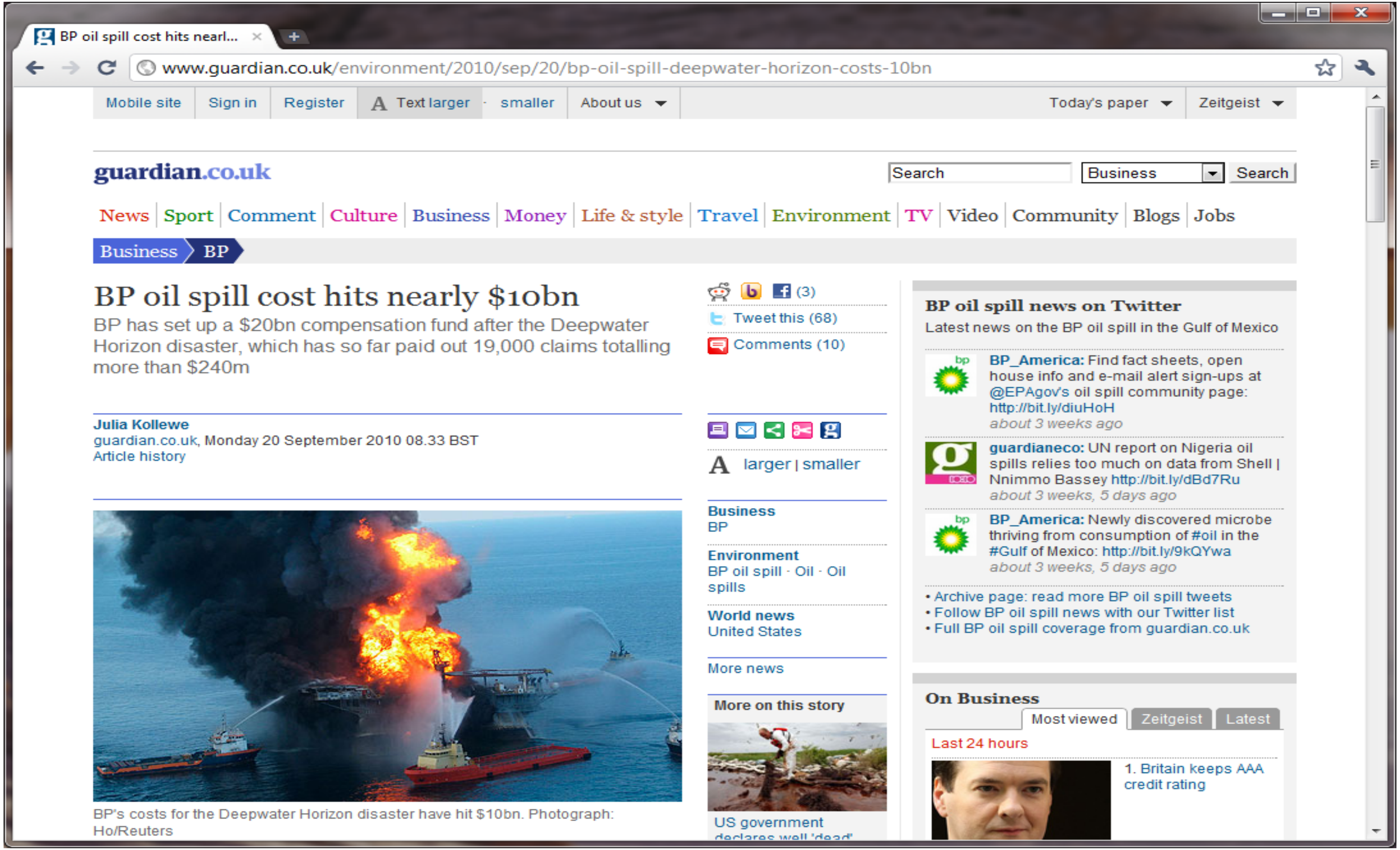
Today's five most popular stories

- White House official denies Tea Party-focused ad campaign
- Swedish far-right leader: Success due to immigration backlash
- Lady Gaga to lead rally against military's 'don't ask, don't tell'
- Why Facebook is blue -- six facts about Mark Zuckerberg
- China, Japan relations sour in dispute over fishing captain

More

STORY HIGHLIGHTS

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



BP oil spill cost hits nearl...

www.guardian.co.uk/environment/2010/sep/20/bp-oil-spill-deepwater-horizon-costs-10bn

Mobile site | Sign in | Register | A Text larger · smaller | About us | Today's paper | Zeitgeist

guardian.co.uk

Search | Business | Search

News | Sport | Comment | Culture | Business | Money | Life & style | Travel | Environment | TV | Video | Community | Blogs | Jobs

Business › BP

BP oil spill cost hits nearly $10bn

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

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

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

(3)
Tweet this (68)
Comments (10)

A larger | smaller

Business
BP

Environment
BP oil spill · Oil · Oil spills

World news
United States

More news

More on this story

US government declares well 'dead'

BP oil spill news on Twitter

Latest news on the BP oil spill in the Gulf of Mexico

BP_America: Find fact sheets, open house info and e-mail alert sign-ups at @EPAgov's oil spill community page: http://bit.ly/diuHoH
about 3 weeks ago

guardianeco: UN report on Nigeria oil spills relies too much on data from Shell | Nnimmo Bassey http://bit.ly/dBd7Ru
about 3 weeks, 5 days ago

BP_America: Newly discovered microbe thriving from consumption of #oil in the #Gulf of Mexico: http://bit.ly/9kQYwa
about 3 weeks, 5 days ago

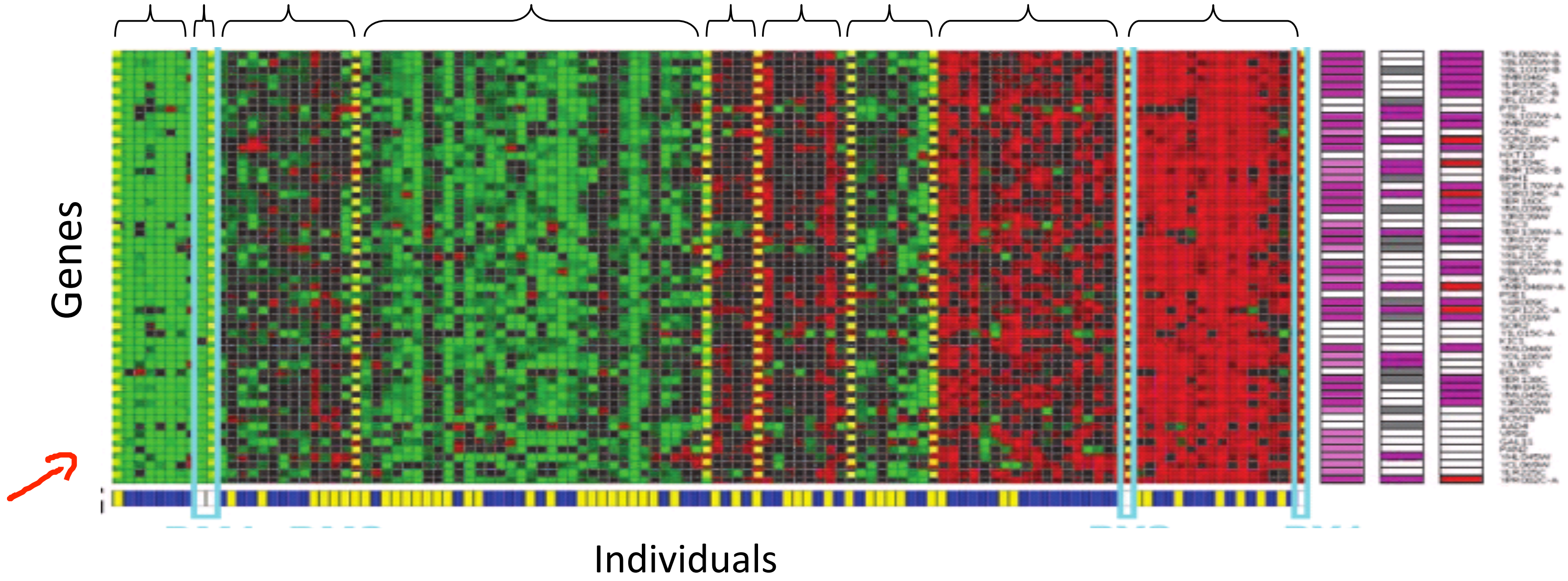
- Archive page: read more BP oil spill tweets
- Follow BP oil spill news with our Twitter list
- Full BP oil spill coverage from guardian.co.uk

On Business

Most viewed | Zeitgeist | Latest

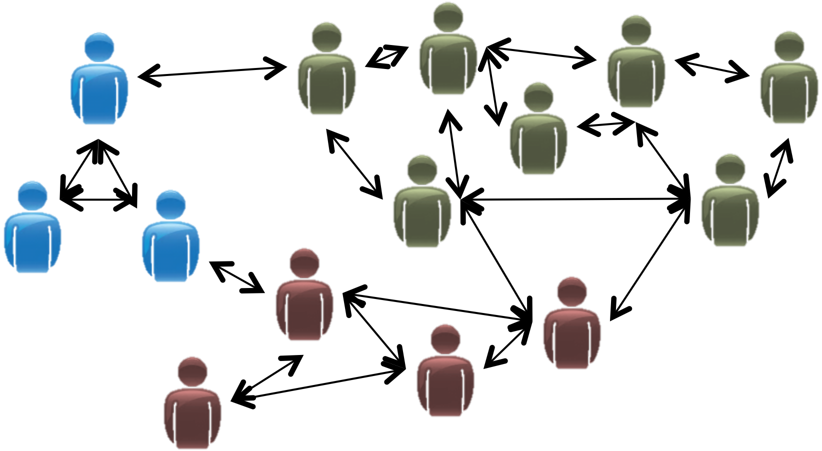
Last 24 hours

1. Britain keeps AAA credit rating

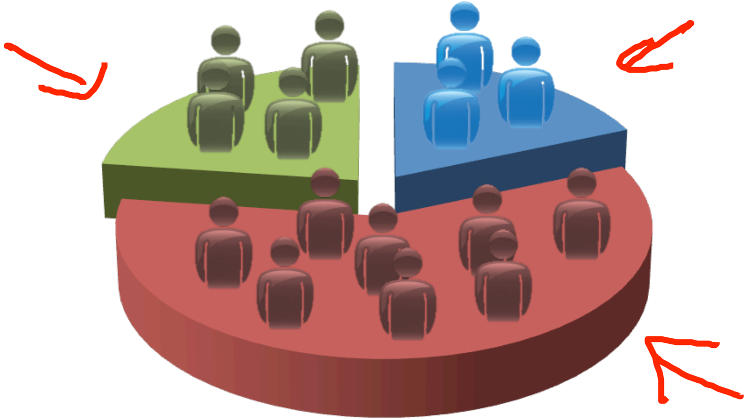


[Source: Su-In Lee, Dana Pe’er, Aimee Dudley, George Church, Daphne Koller]

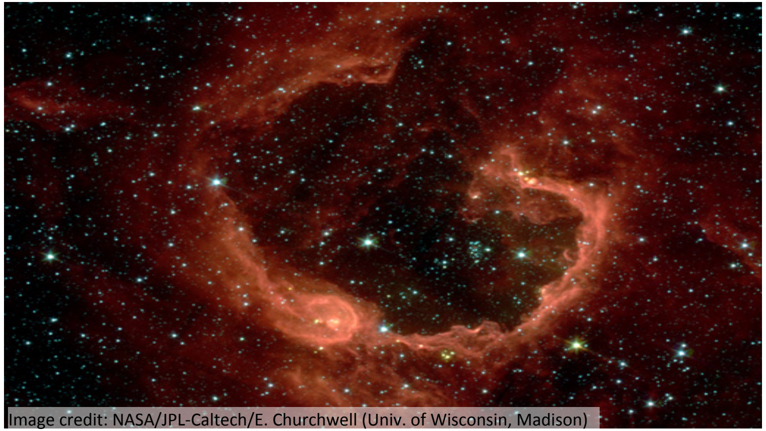
Organize computing clusters



Social network analysis



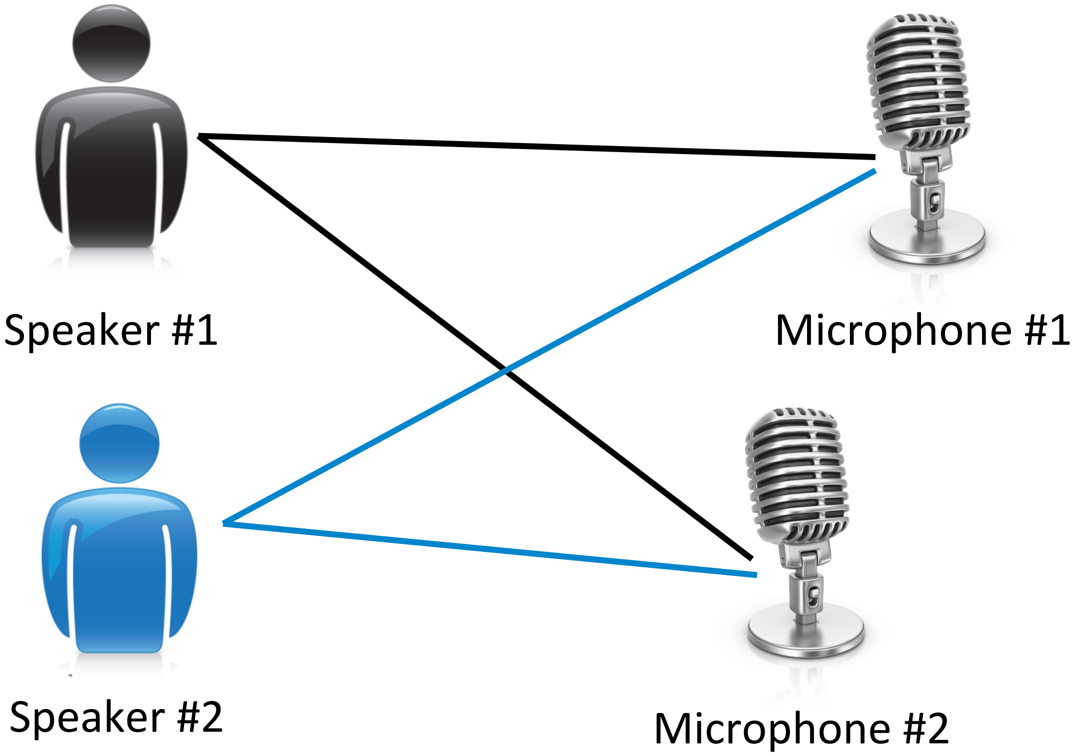
Market segmentation



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.]

Cocktail party problem algorithm

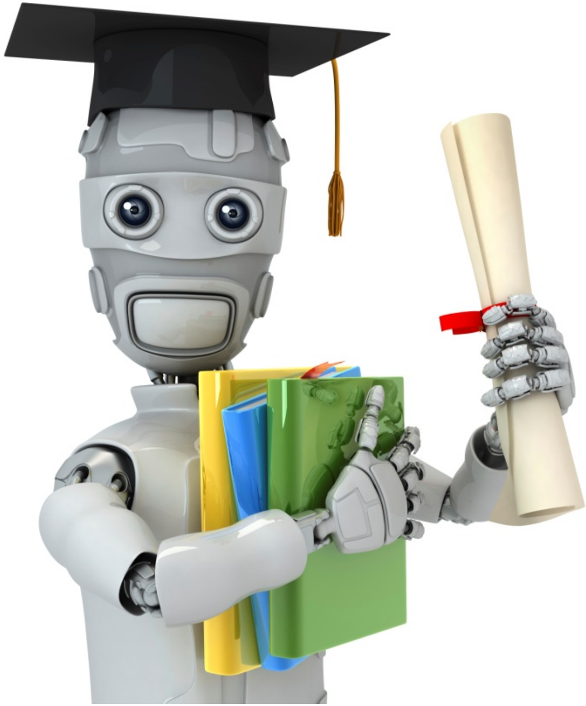
```
[W,s,v] = svd((repmat(sum(x.*x,1),size(x,1),1).*x)*x');
```

[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.

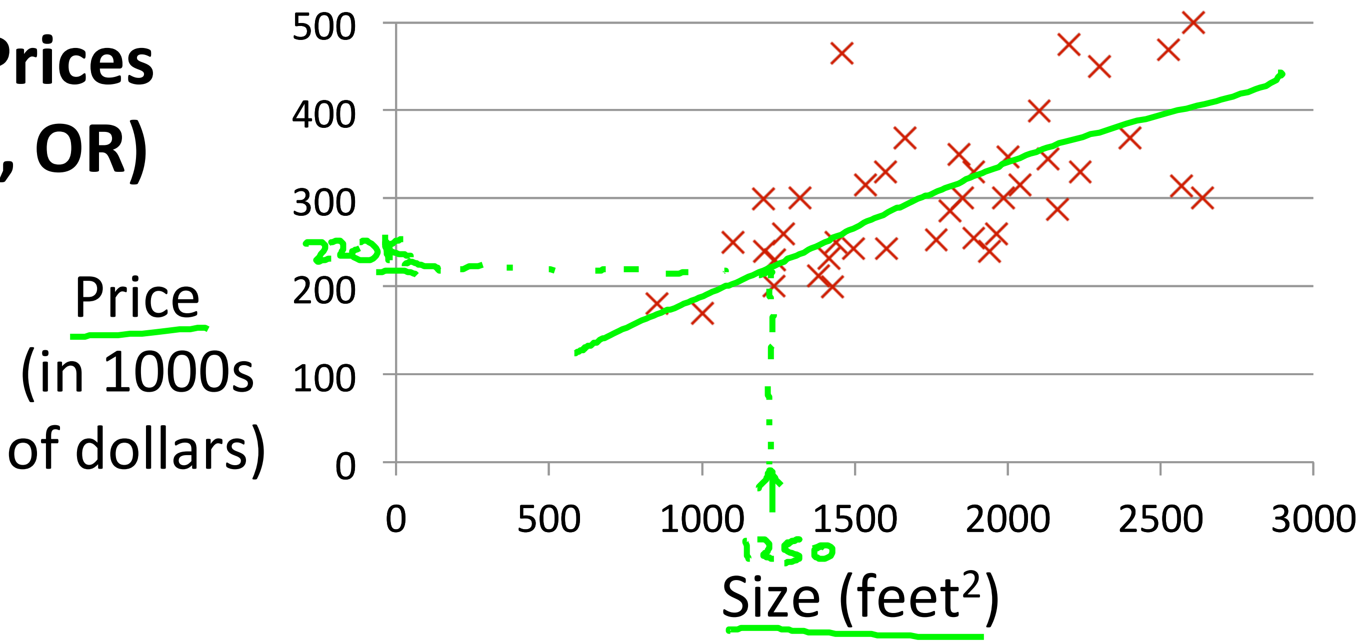


Machine Learning

Linear regression with one variable

Model representation

Housing Prices (Portland, OR)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Classification: Discrete-valued output

Andrew Ng

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
…	…

$m = 47$

Notation:

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

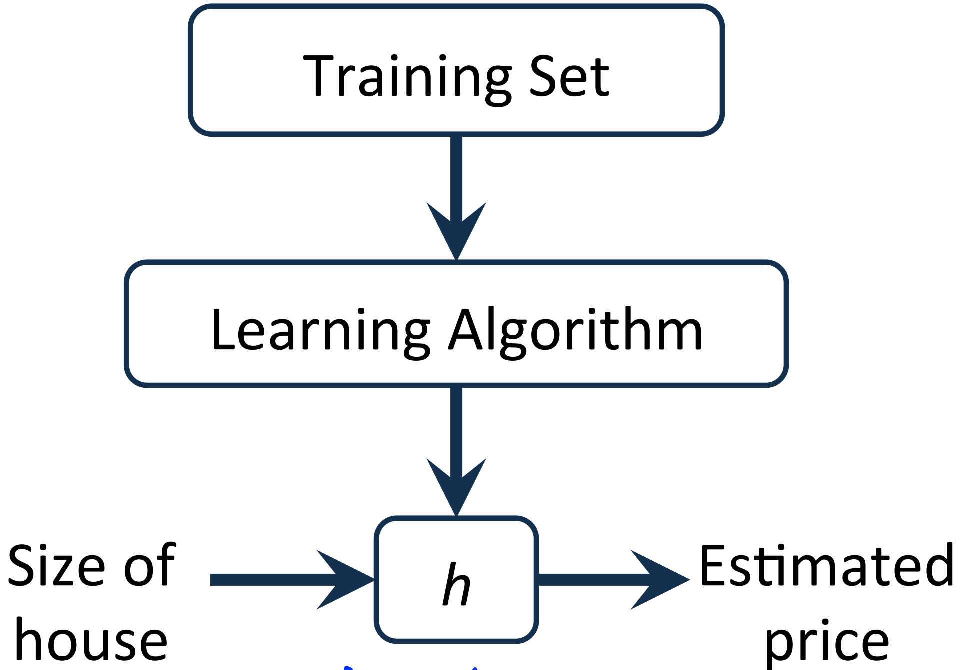
(x, y) - one training example

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

$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$



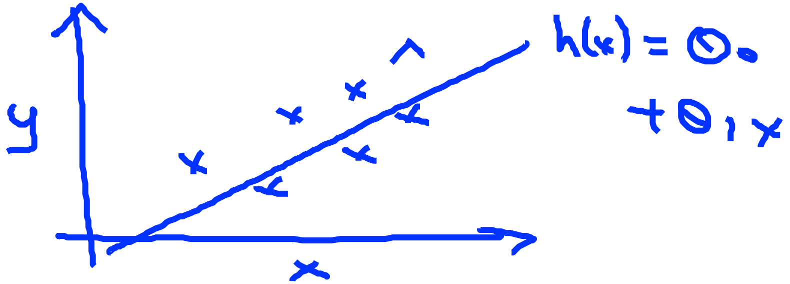
x hypothesis (estimated value of y)

h maps from x's to y's.

How do we represent *h* ?

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

Shorthand: $h(x)$

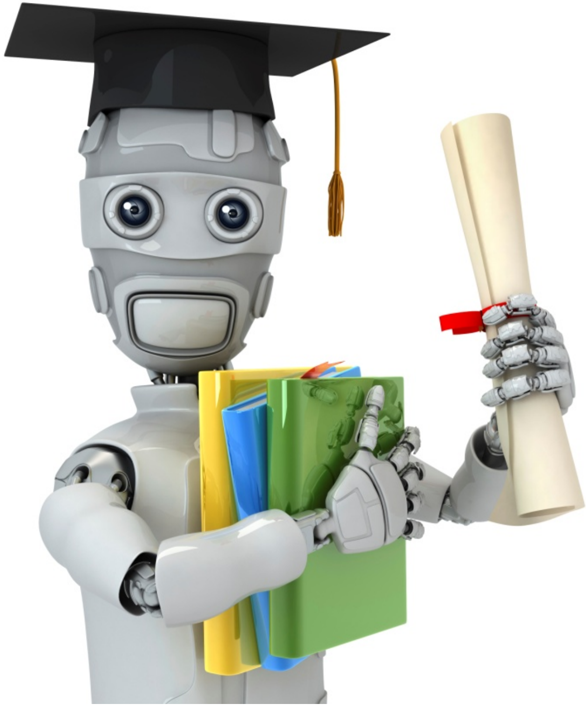


Linear regression with one variable. (x)

Univariate linear regression.

one variable

Andrew Ng



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
…	…

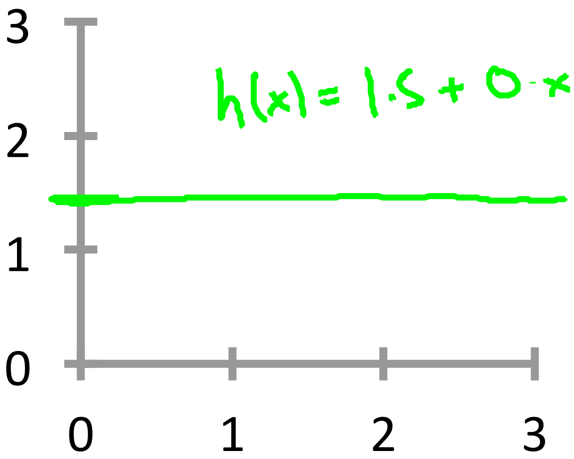
$m = 47$

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

θ_i's: Parameters

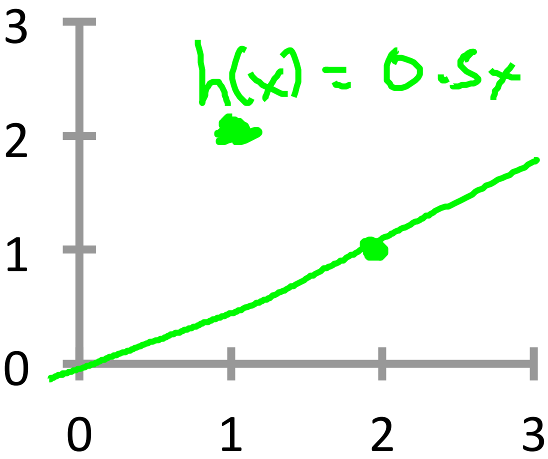
How to choose θ_i's ?

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



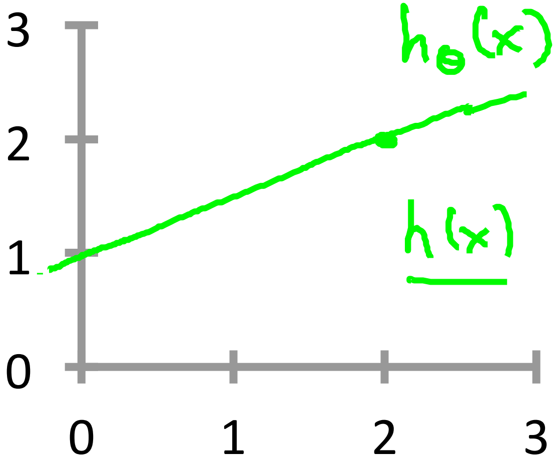
$\theta_0 = 1.5$

$\theta_1 = 0$



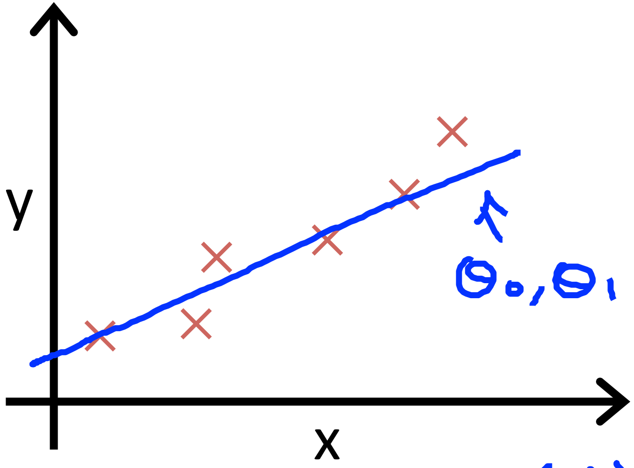
$\theta_0 = 0$

$\theta_1 = 0.5$



$\theta_0 = 1$

$\theta_1 = 0.5$



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

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

x, y

\# training examples

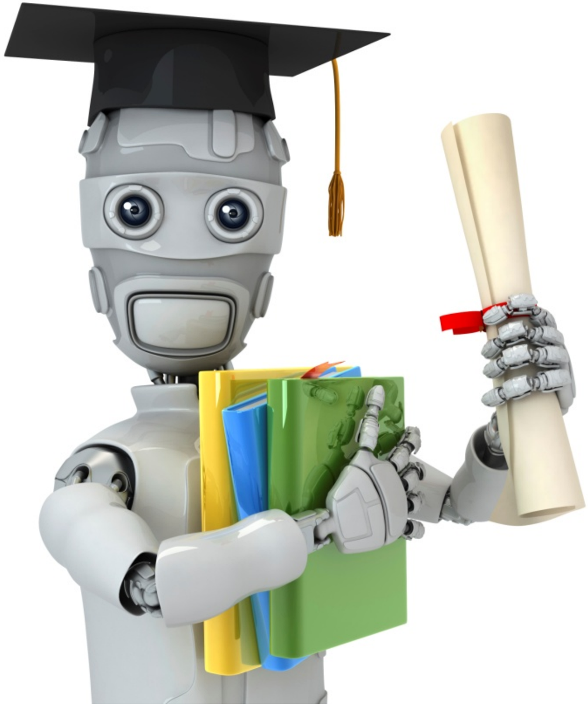
$$\boxed{\underset{\theta_0, \theta_1}{\text{minimize}}} \; \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

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

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

$$\underset{\theta_0, \theta_1}{\text{minimize}} \; \underbrace{J(\theta_0, \theta_1)}_{\text{Cost function}}$$

Squared error function



Machine Learning

Linear regression with one variable

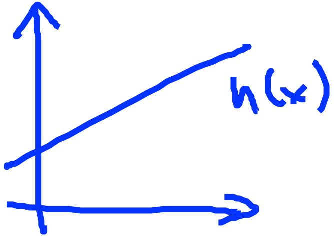
Cost function intuition I

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$



Cost Function:

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

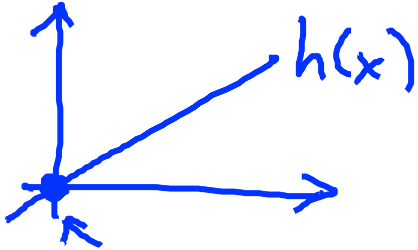
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Simplified

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

$\theta_0 = 0$

$$\theta_1$$



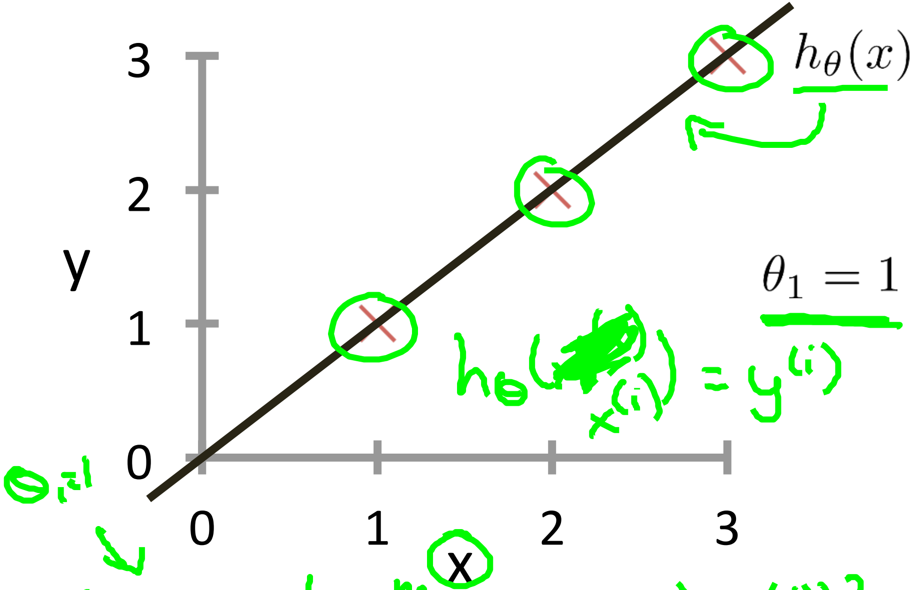
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

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

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

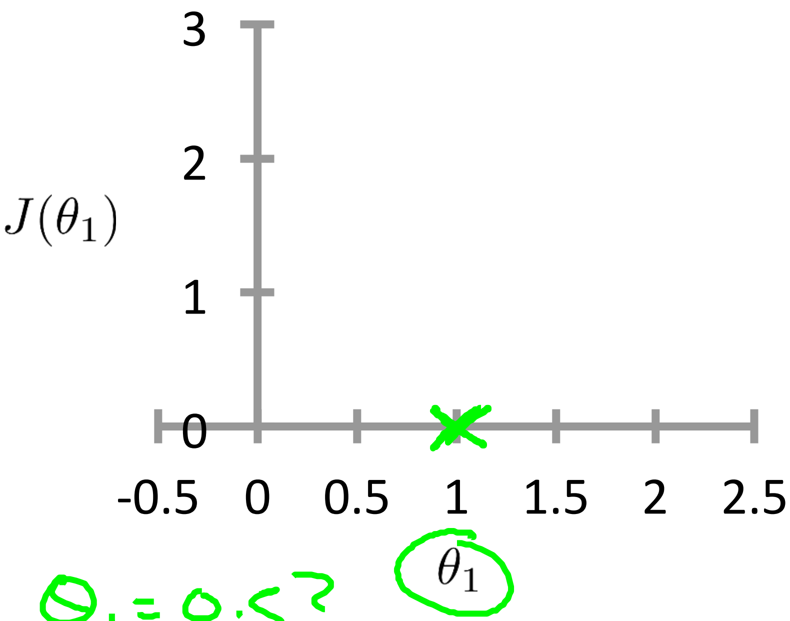
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



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

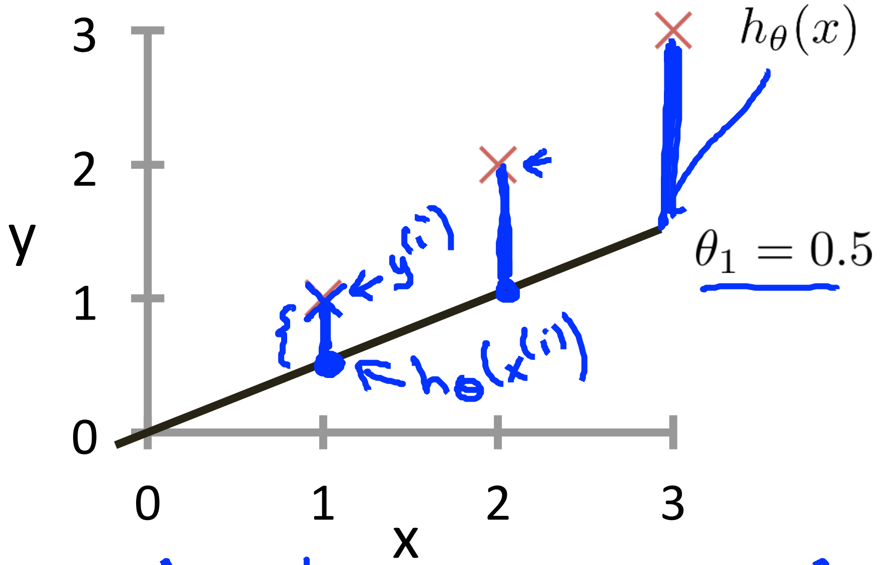
$$= \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_1 x^{(i)} - y^{(i)}\right)^2 = \frac{1}{2m}(0^2 + 0^2 + 0^2) = 0^2$$

$\theta_1 = 0.5?$

$J(1) = 0$

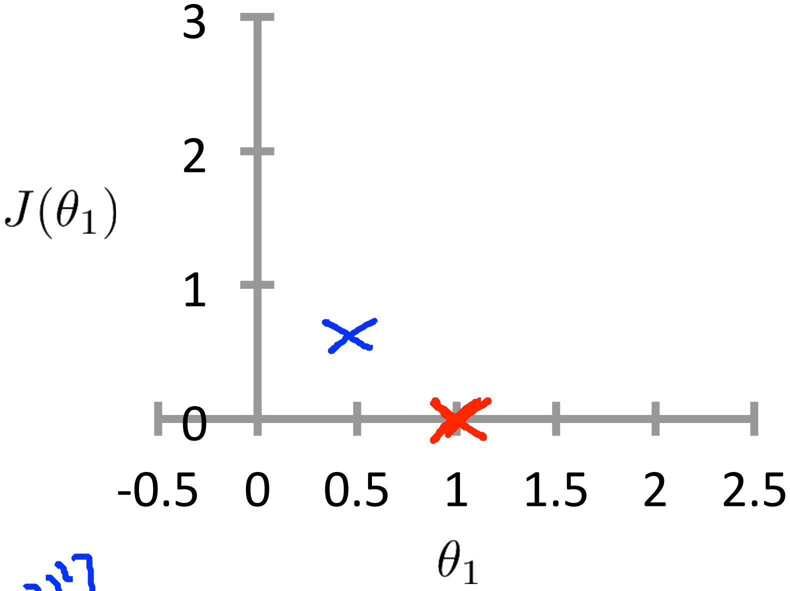
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

$$= \frac{1}{2\times 3}(3.5) = \frac{3.5}{6} \approx 0.58$$

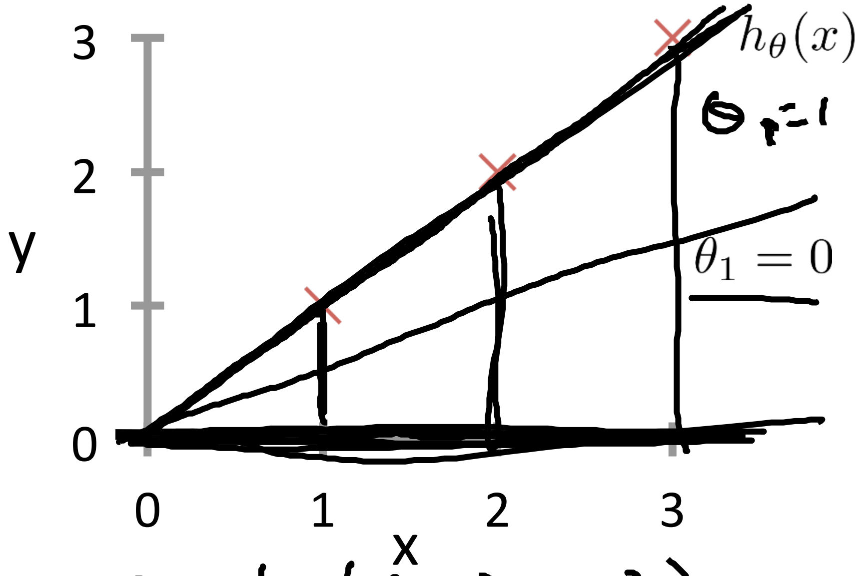
$\theta_1 = 0?$

$J(0) = ?$

Andrew Ng

$h_\theta(x)$

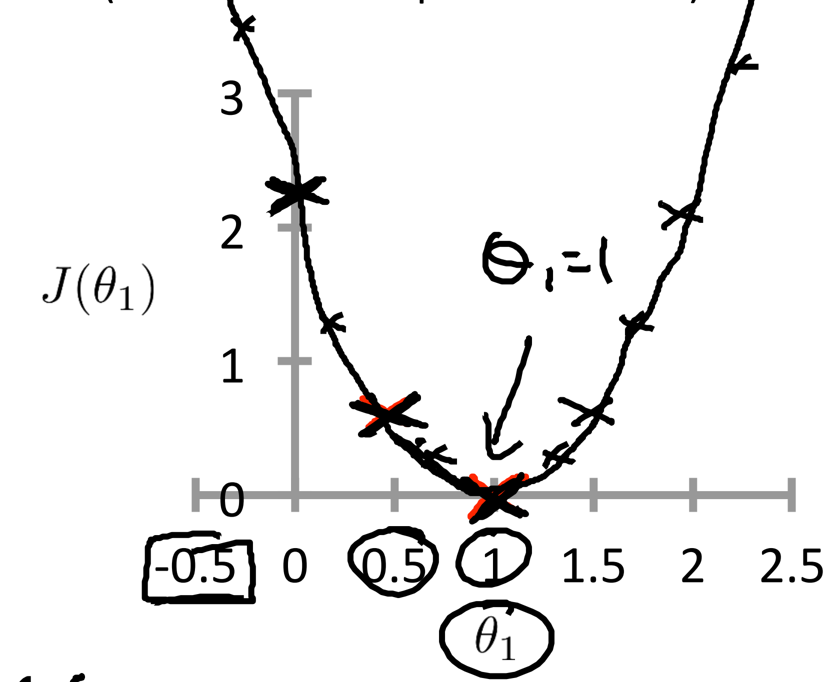
(for fixed θ_1, this is a function of x)



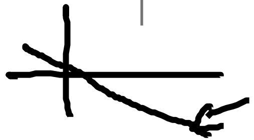
$$J(0) = \frac{1}{2m}(1^2 + 2^2 + 3^2)$$
$$= \frac{1}{6} \cdot 14 \approx 2.3$$

$J(\theta_1)$

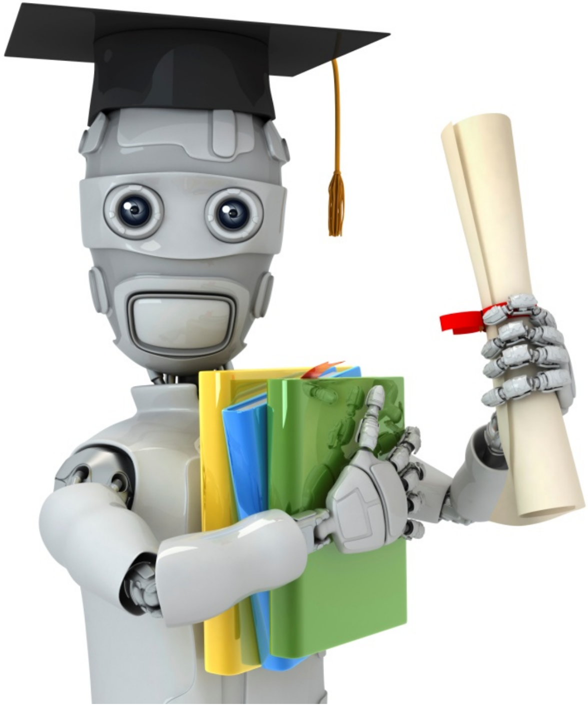
(function of the parameter θ_1)



$h(x) = -0.5x$



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



Machine Learning

Linear regression with one variable

Cost function intuition II

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

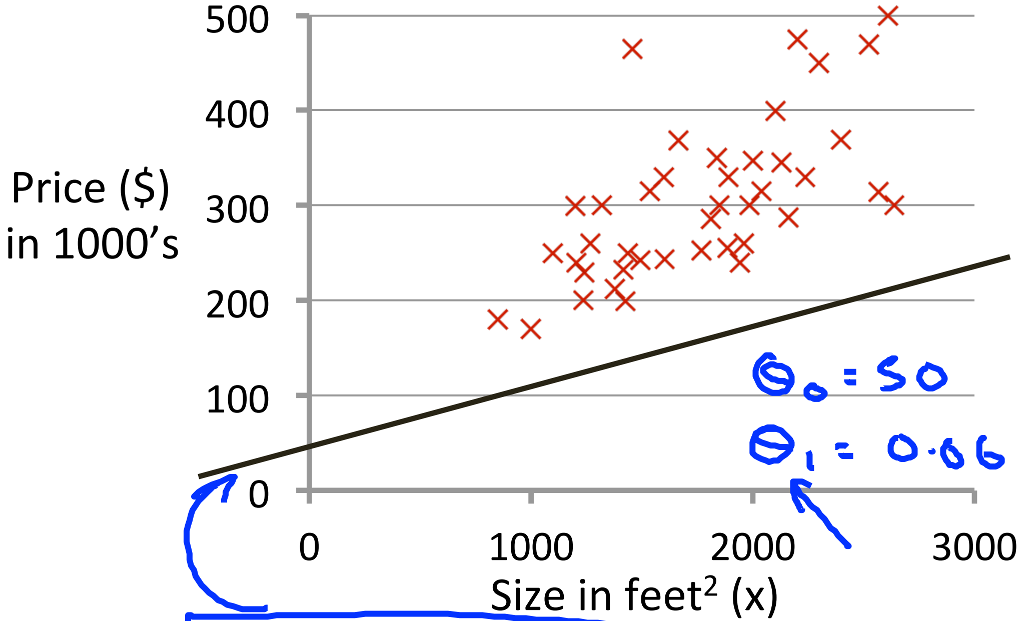
Parameters: θ_0, θ_1

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

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

$h_\theta(x)$

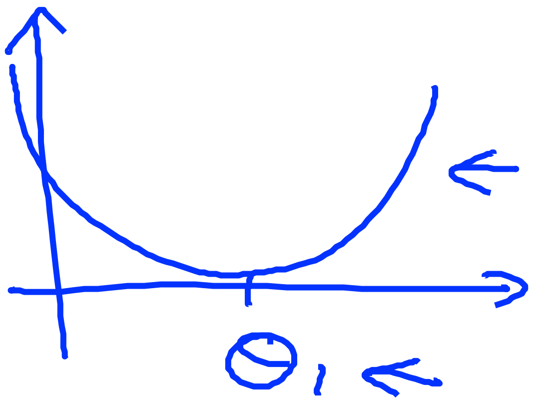
(for fixed θ_0, θ_1, this is a function of x)



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

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

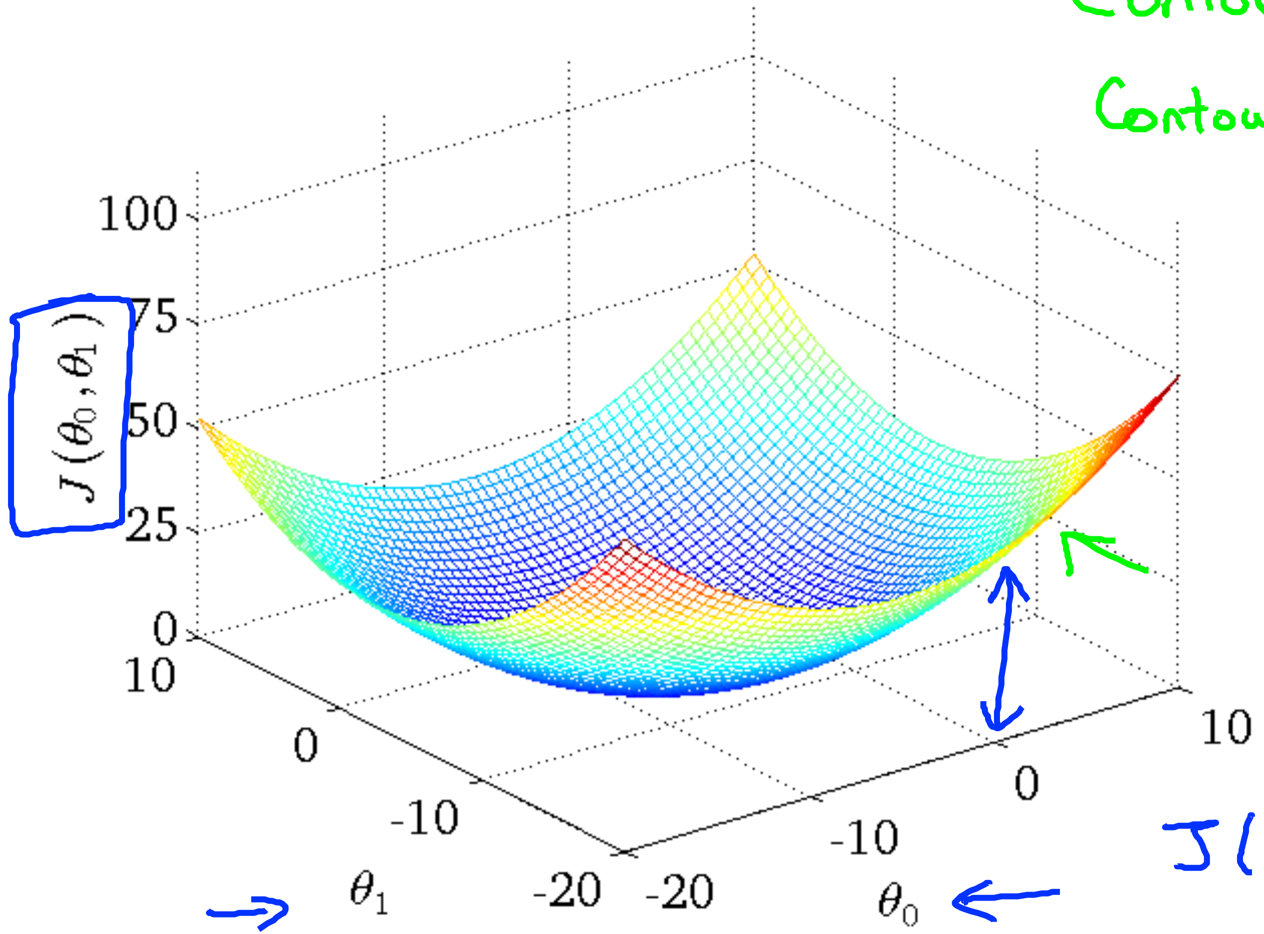


$\boxed{\theta_0, \theta_1}$

Andrew Ng

Contour plots

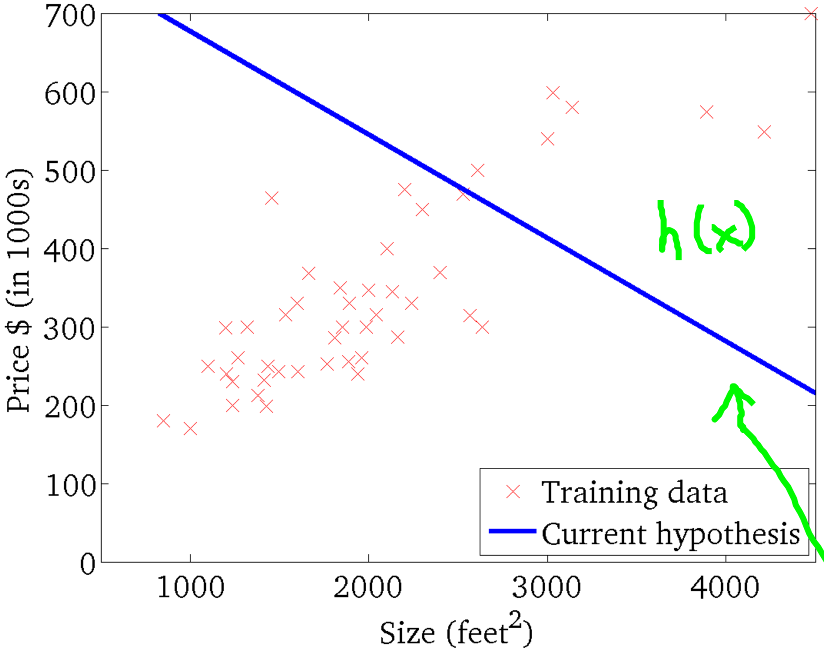
Contour figures -



$J(\theta_0, \theta_1)$

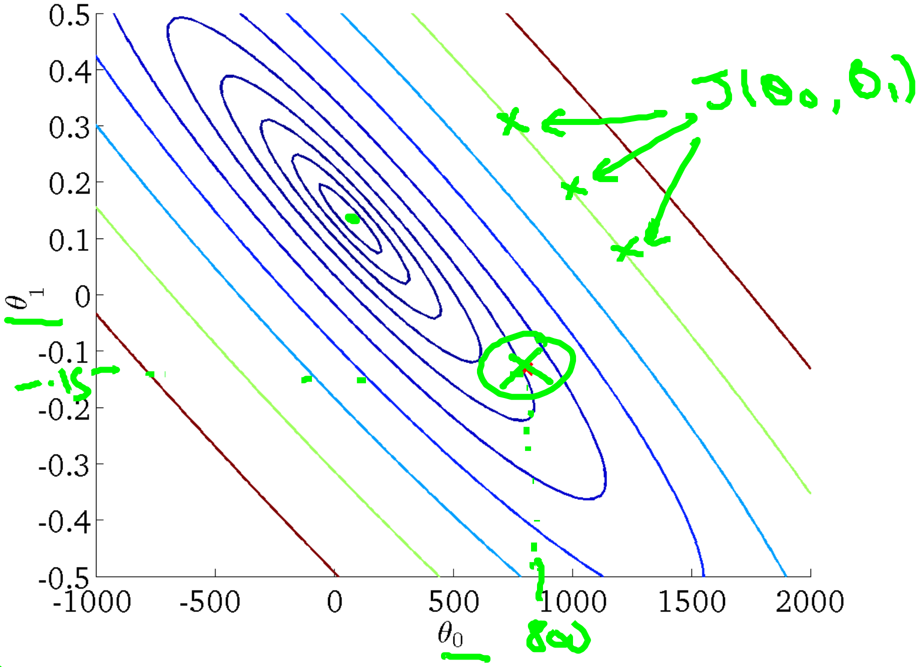
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

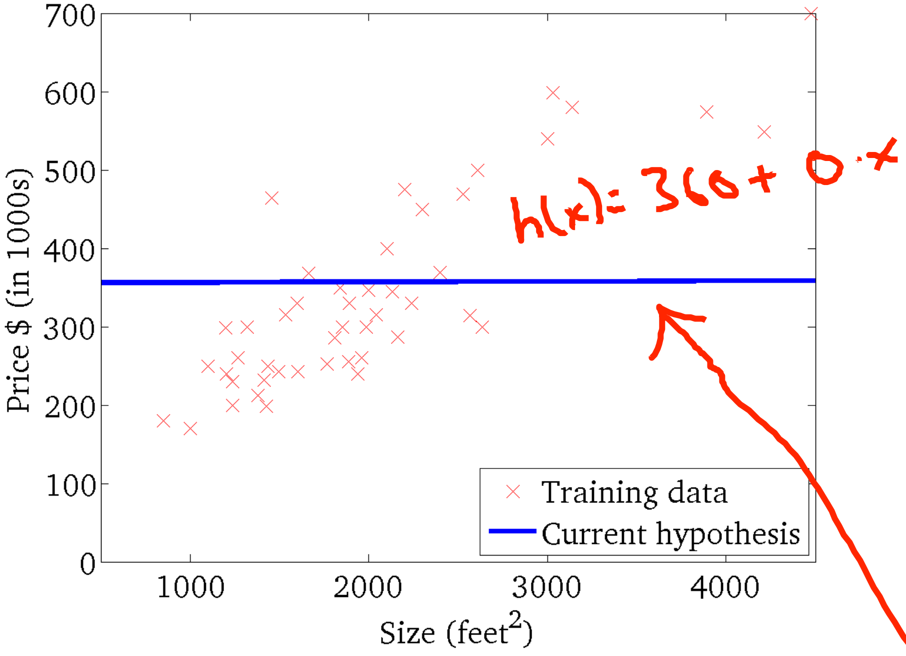
(function of the parameters θ_0, θ_1)



θ_0, θ_1

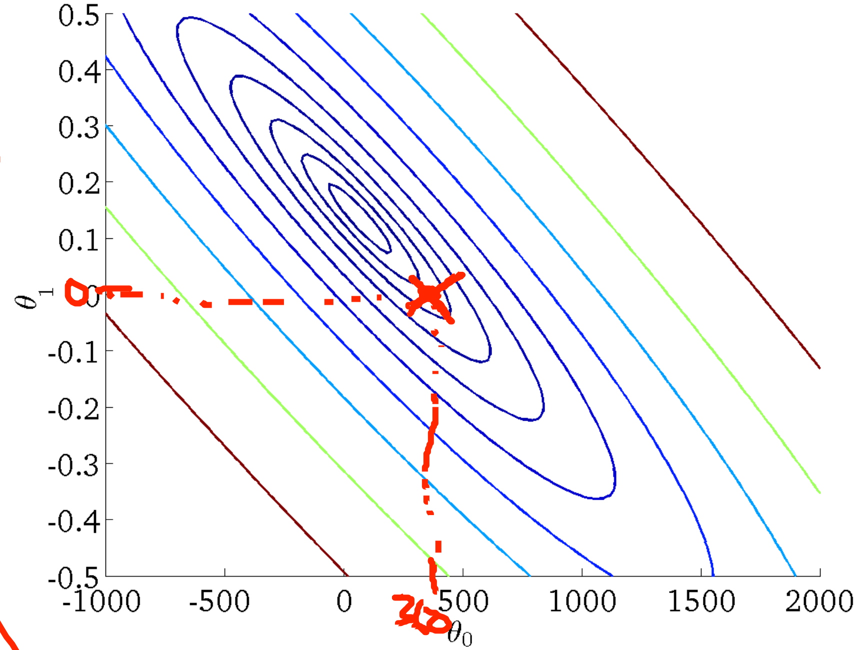
$h_\theta(x)$

(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)

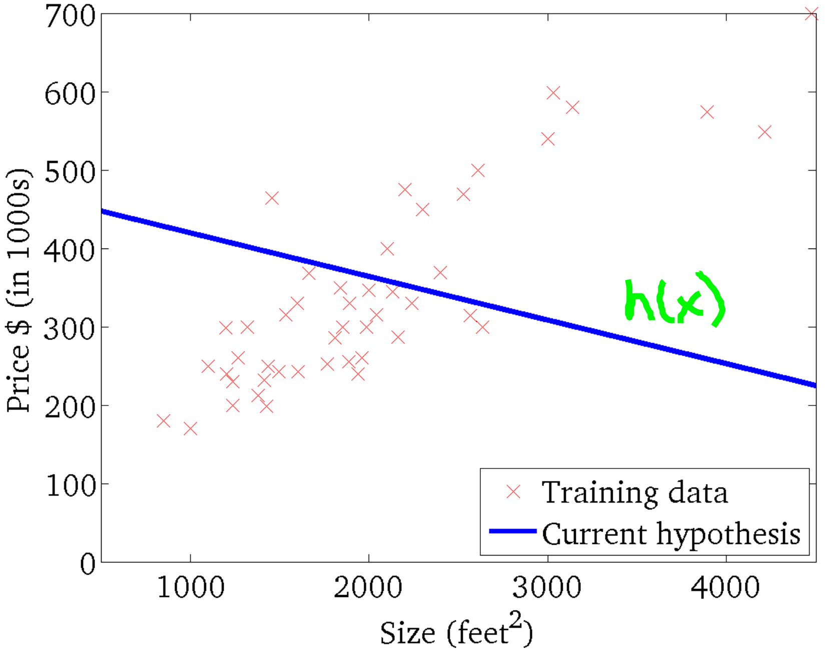


$\theta_0 = 360$

$\theta_1 = 0$

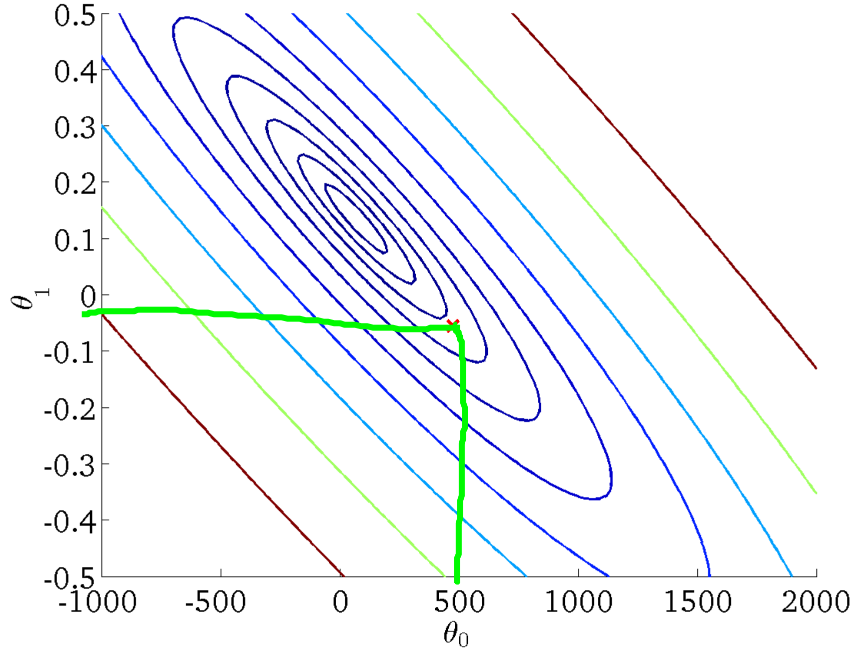
$h_\theta(x)$

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



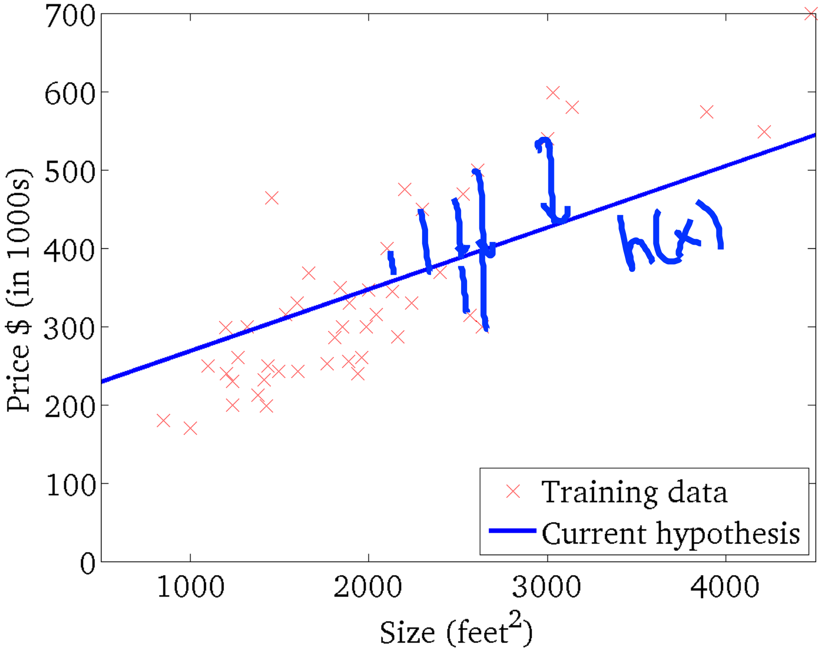
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



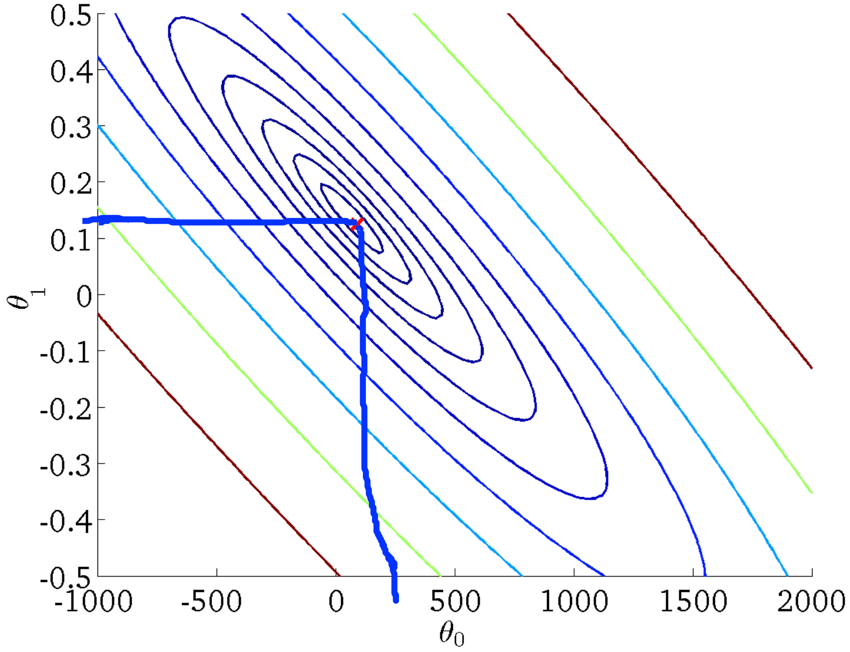
$h_\theta(x)$

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

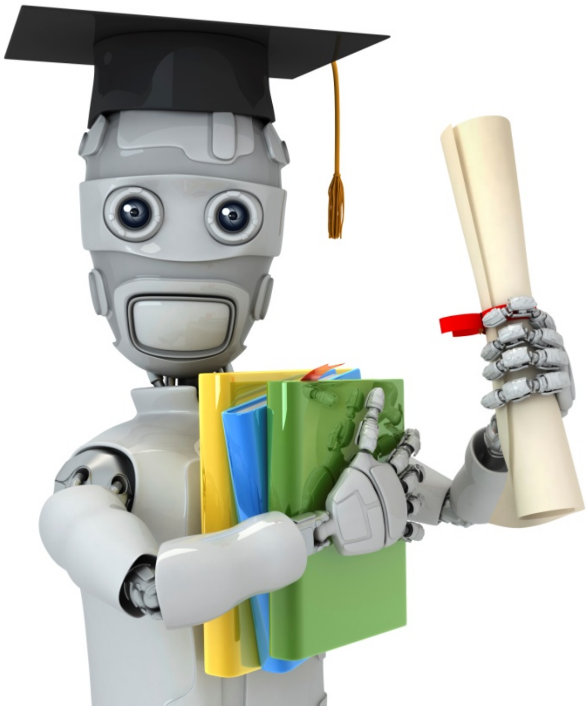


$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Andrew Ng



Machine Learning

Linear regression with one variable

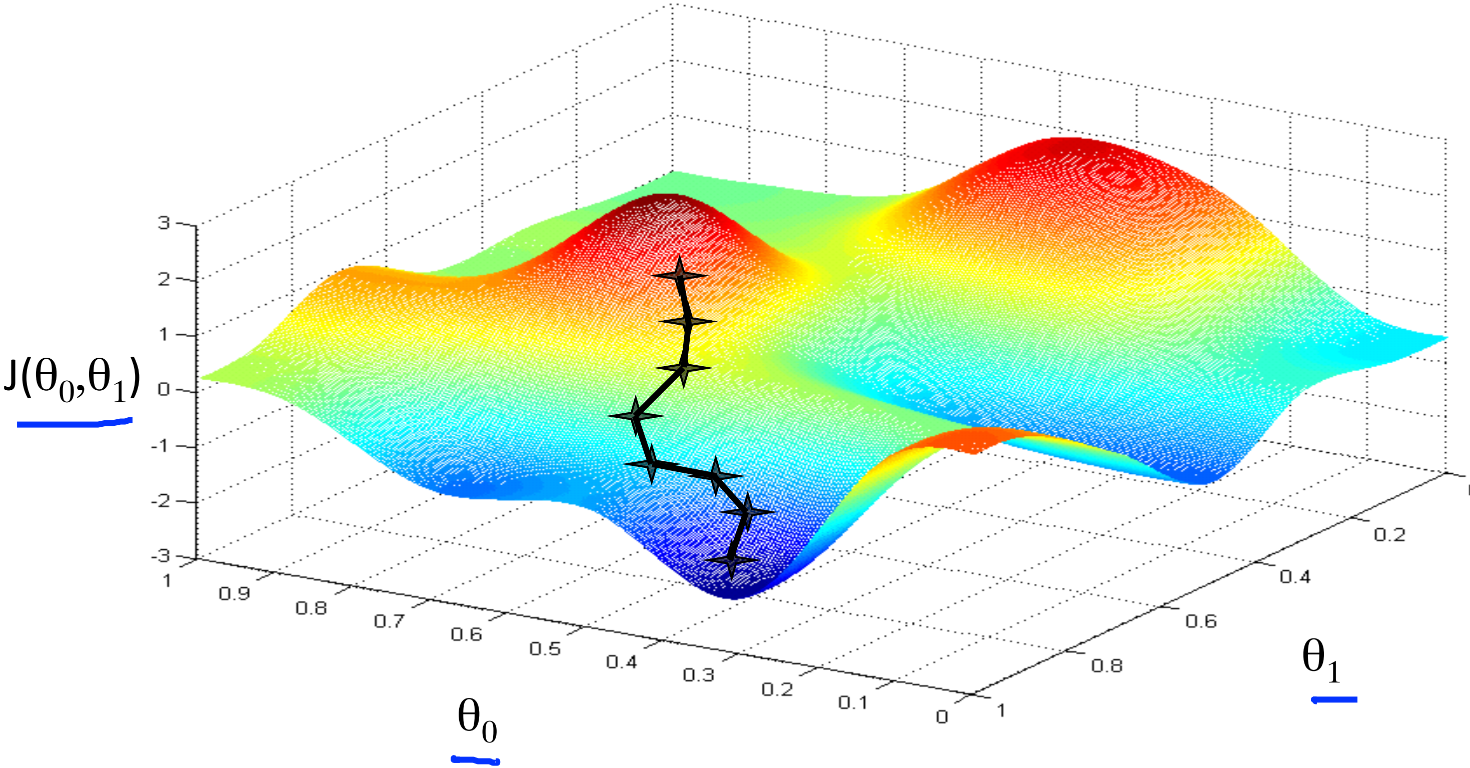
Gradient descent

Have some function $J(\theta_0, \theta_1)$ $J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$

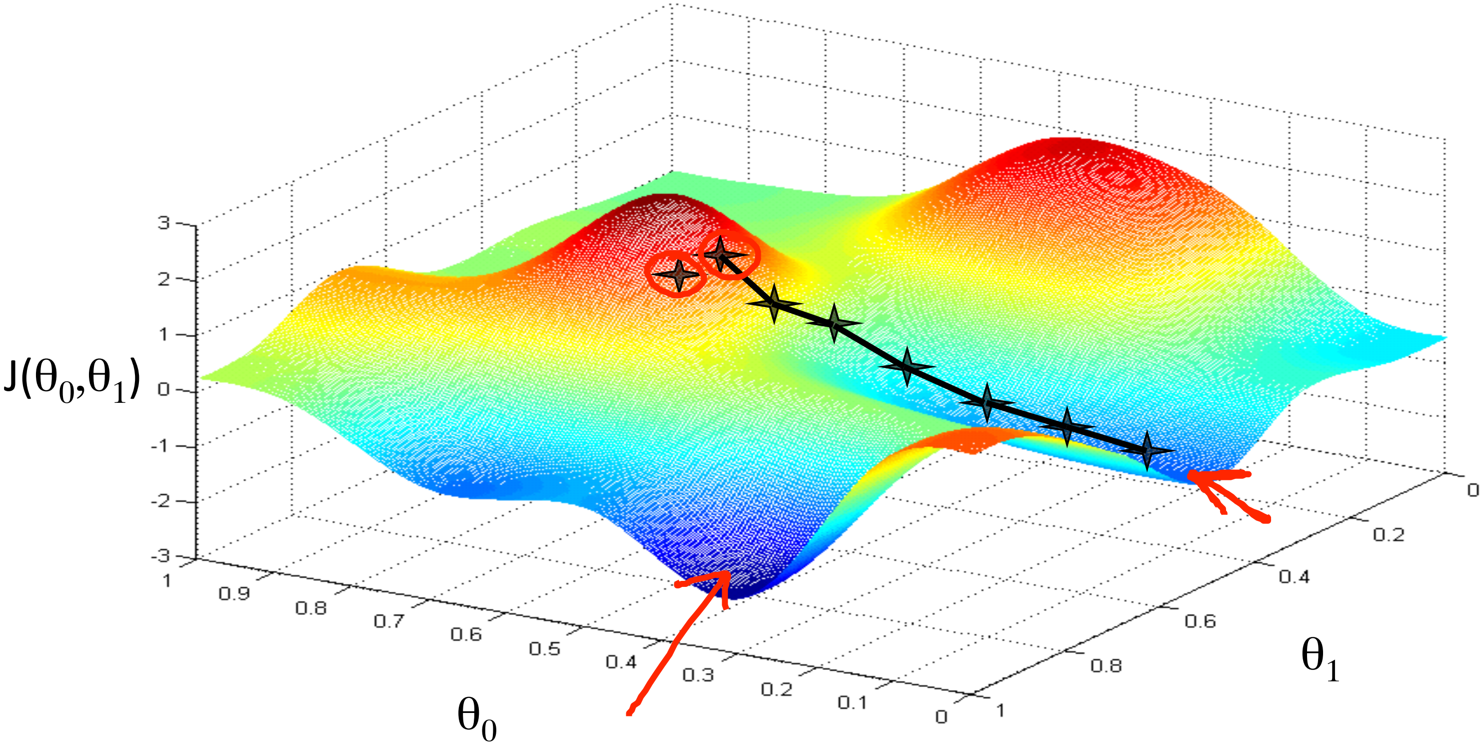
Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ $\min_{\theta_0, \ldots, \theta_n} J(\theta_0, \ldots, \theta_n)$

Outline:

- Start with some θ_0, θ_1 (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum



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



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

Gradient descent algorithm

Assignment | Truth assertion
$a := b$ | $a = b$
$a := a+1$ | $a = a+1$ X

θ_0, θ_1

repeat until convergence {

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

}

learning rate

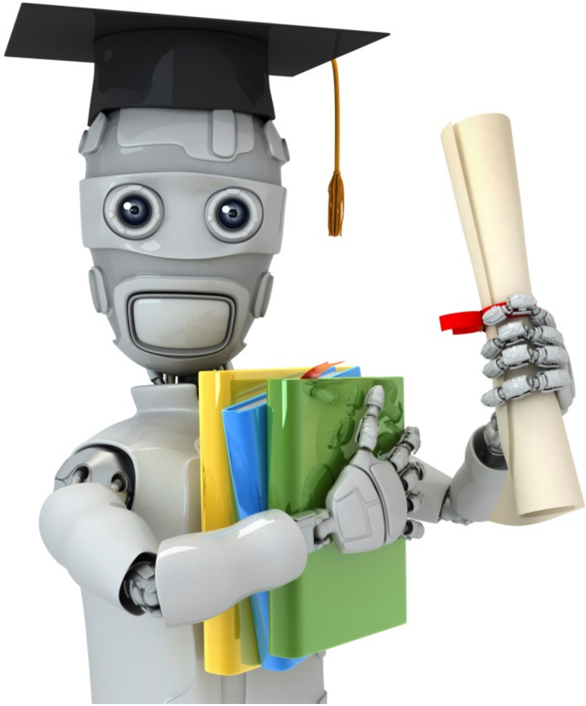
Simultaneously update θ_0 and θ_1

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 {

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

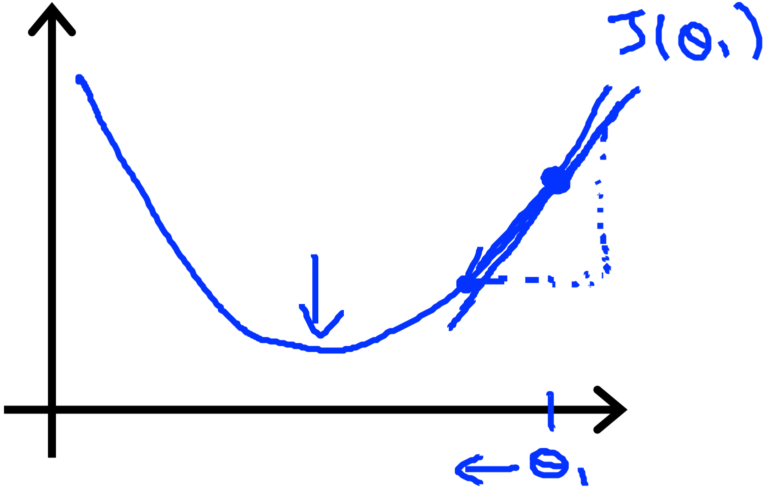
}

learning rate

derivative

$\min_{\theta_1} J(\theta_1)$

$\theta_1 \in \mathbb{R}$.

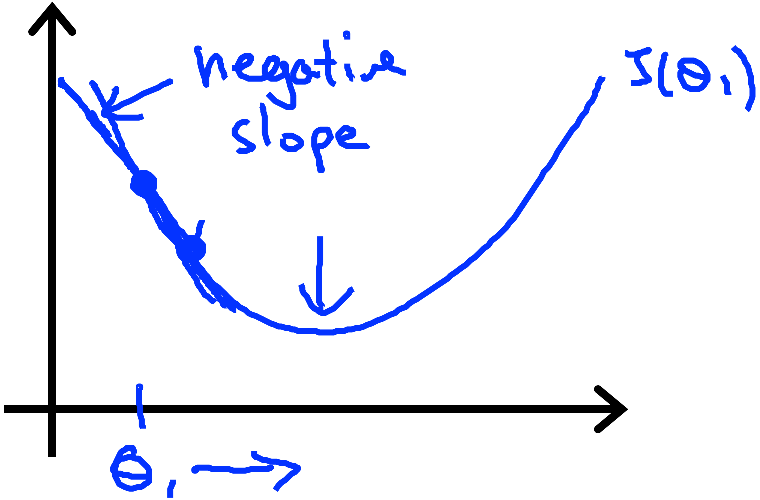


$(\theta_1 \in \mathbb{R})$

$$\frac{\partial}{\partial \theta_1}$$

$$\theta_1 := \theta_1 - \alpha \underbrace{\boxed{\frac{d}{d\theta_1} J(\theta_1)}}_{\geq 0}$$

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



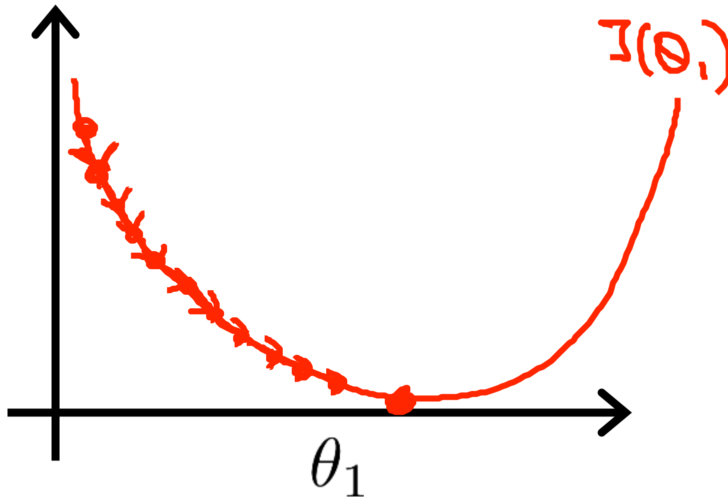
$$\underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\leq 0}$$

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

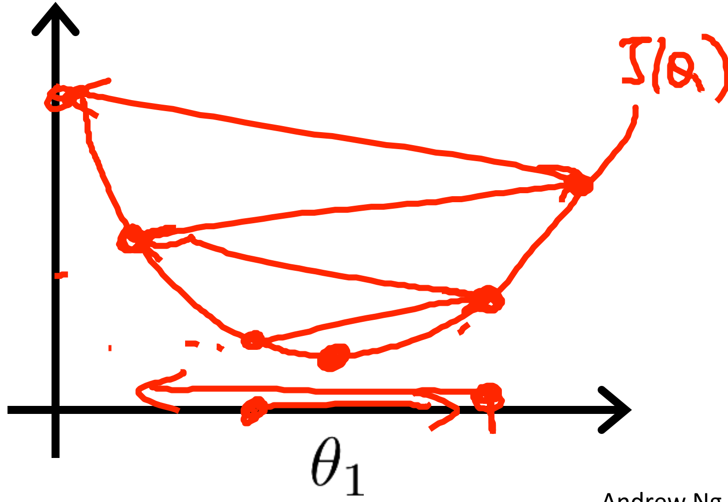
Andrew Ng

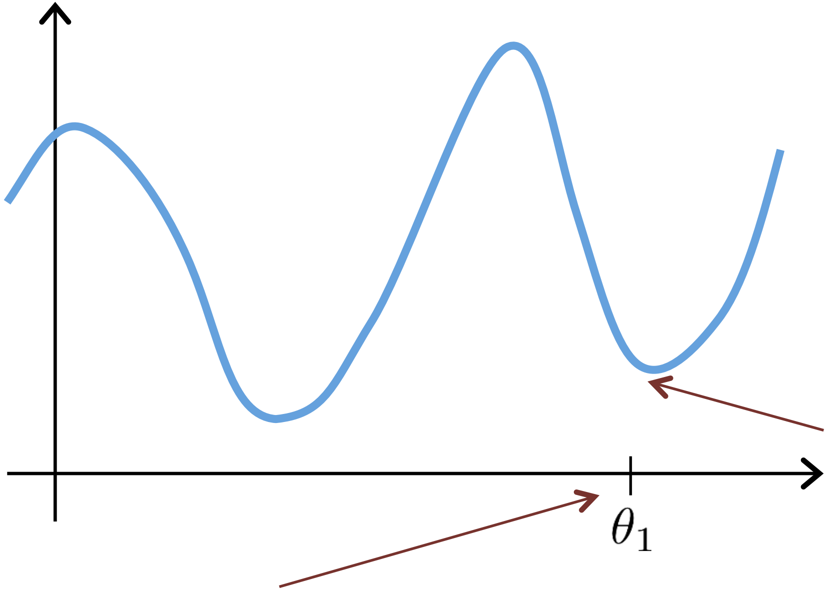
$$\boxed{\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)}$$

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.



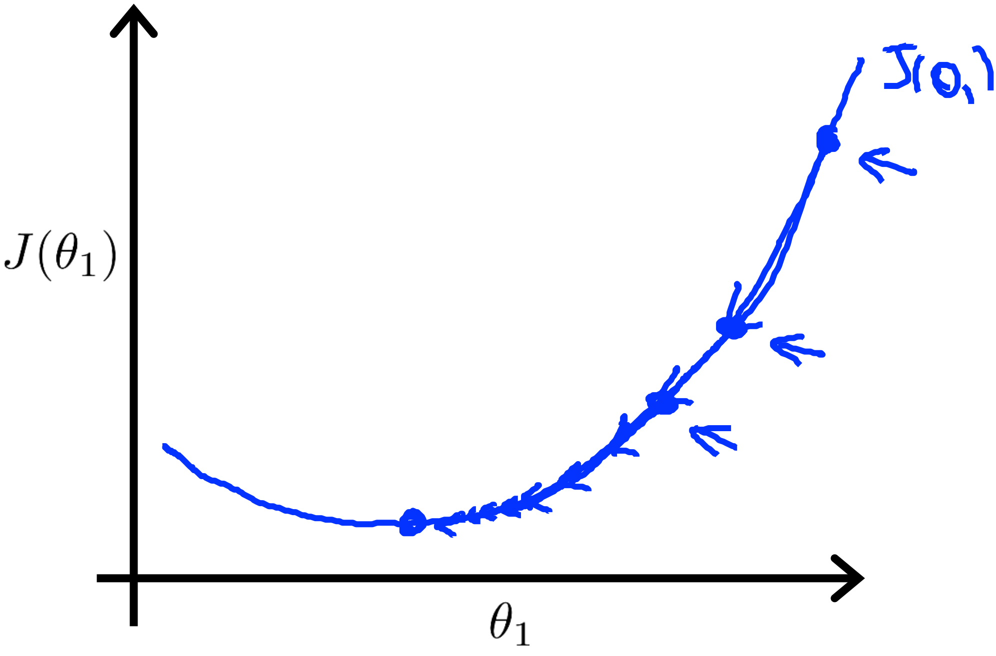


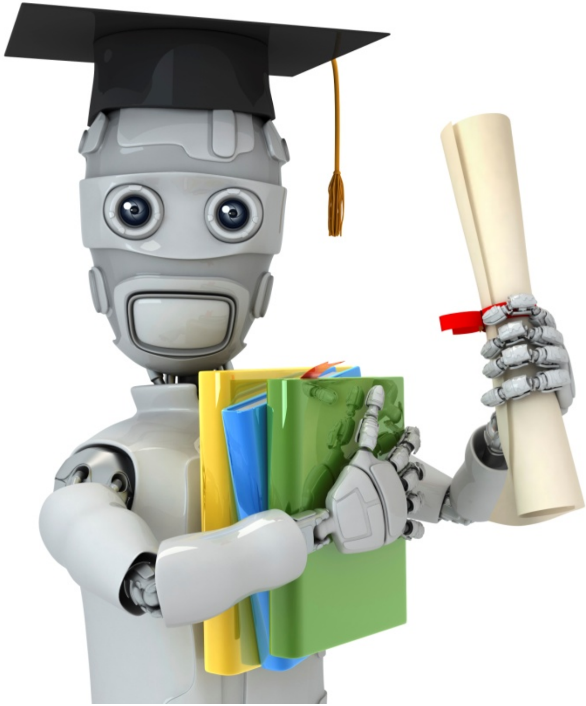
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

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

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

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

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

Linear Regression Model

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

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

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

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

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

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

Gradient descent algorithm

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

repeat until convergence {

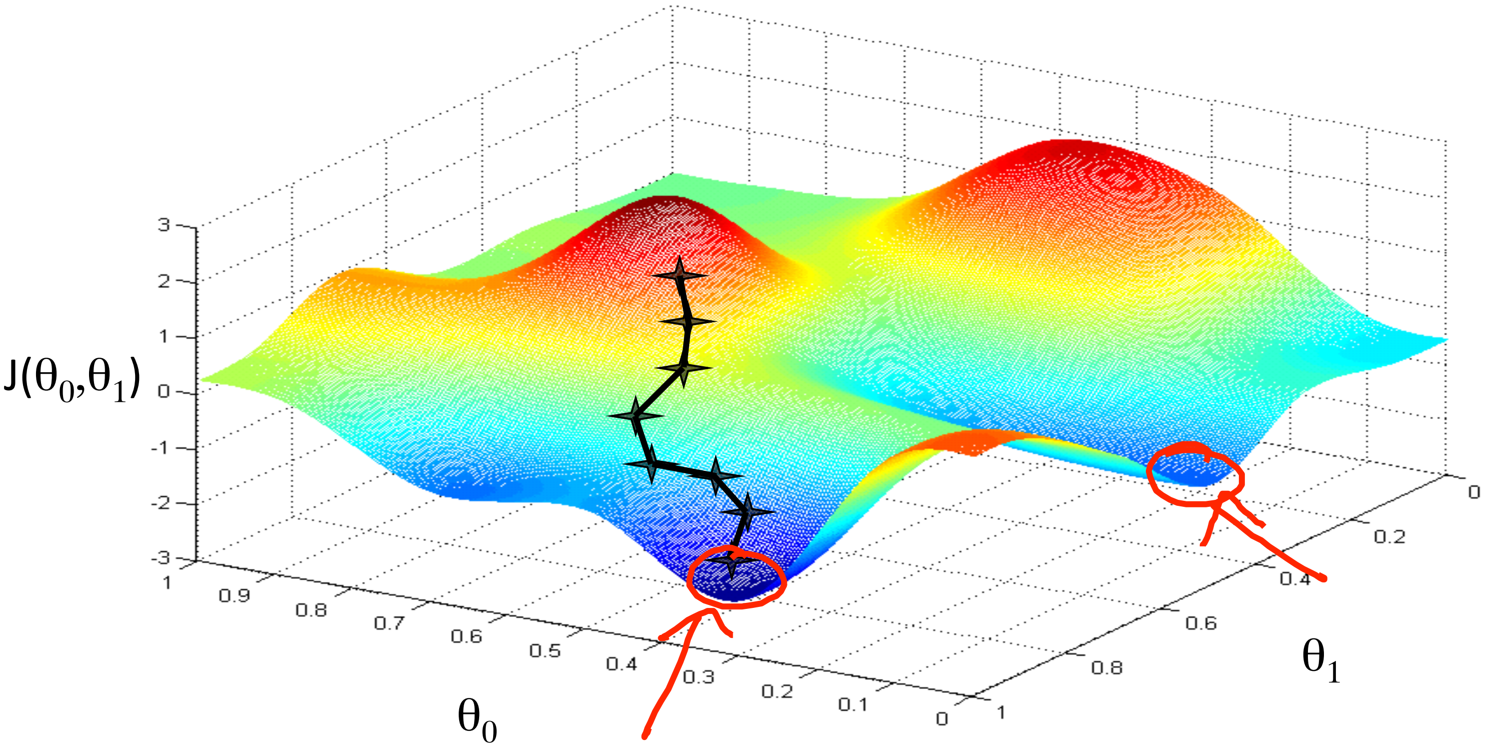
$$\theta_0 := \theta_0 - \alpha \boxed{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)}$$

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

}

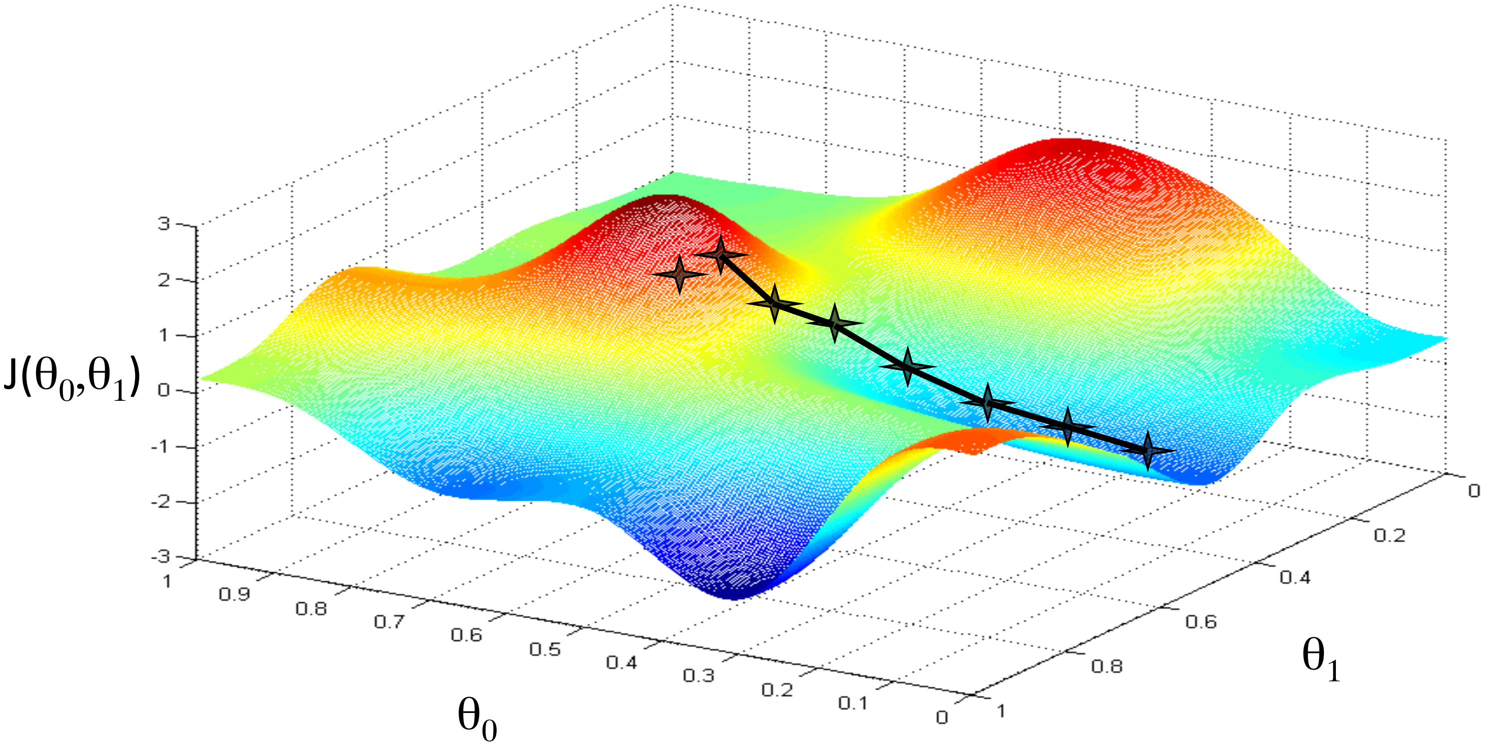
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

update θ_0 and θ_1 simultaneously



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

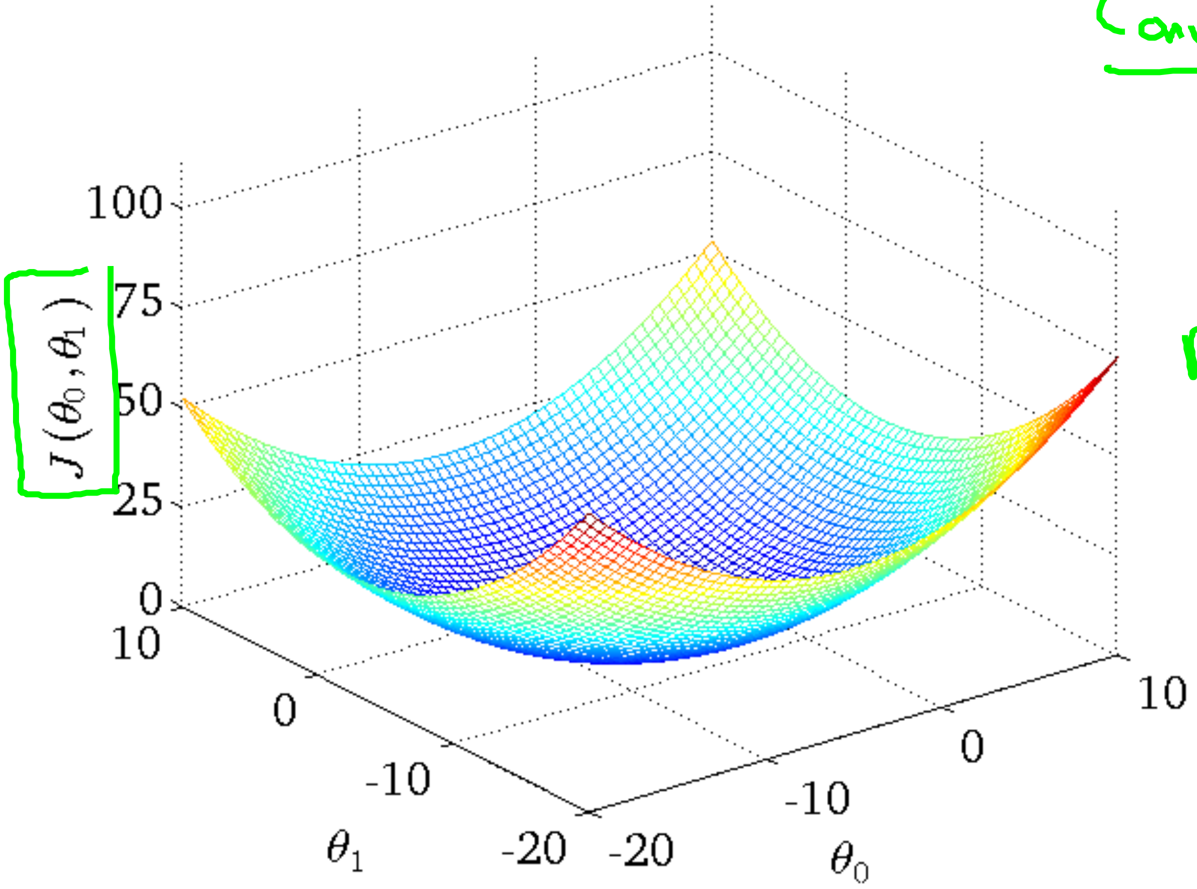
Andrew Ng



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

Andrew Ng

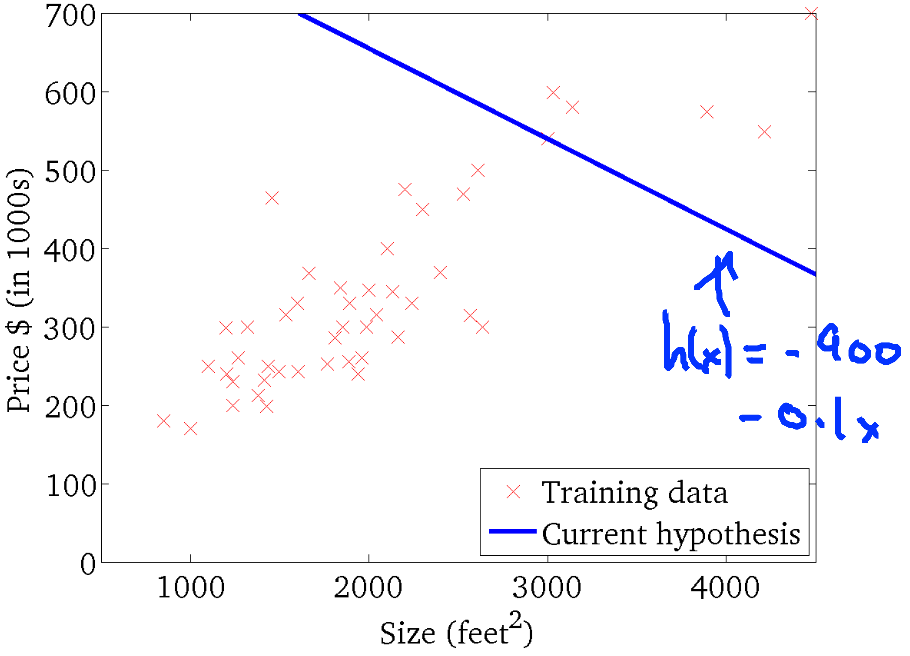
"Convex function"



Bowl-shaped

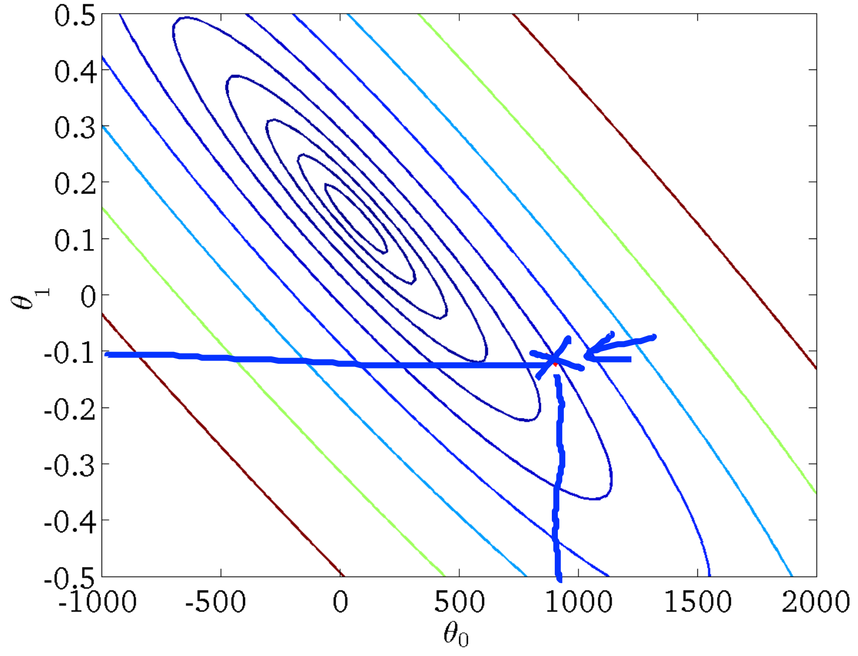
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

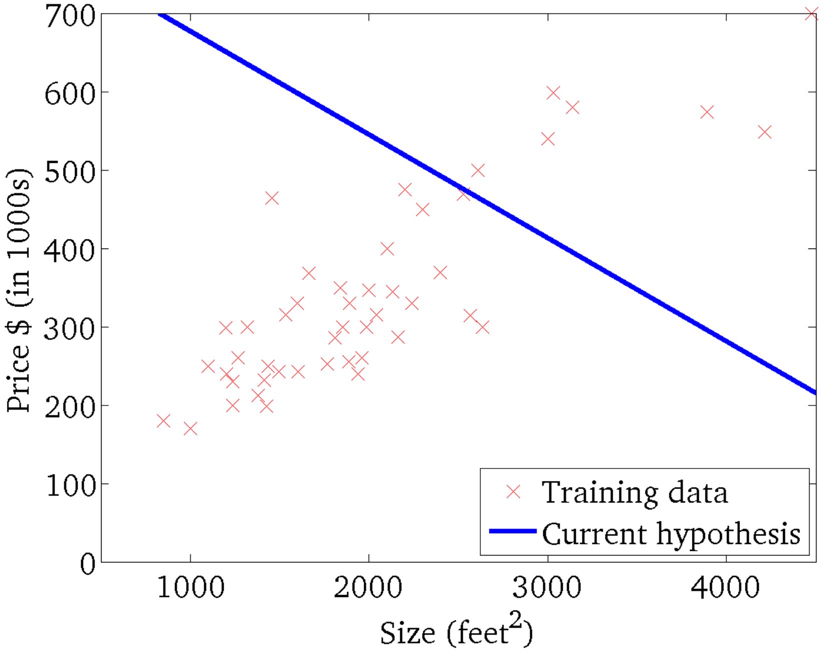
(function of the parameters θ_0, θ_1)



Andrew Ng

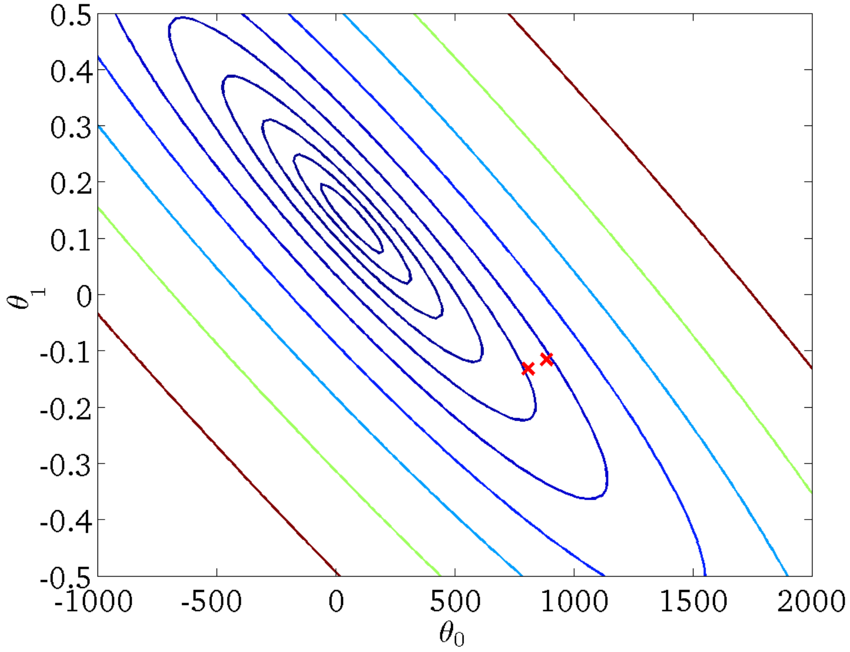
$$h_\theta(x)$$

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



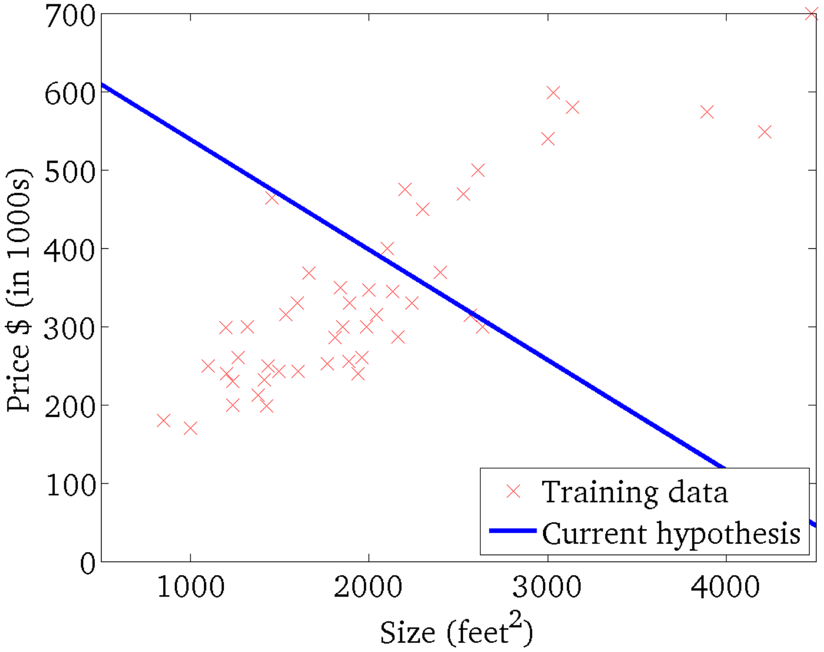
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



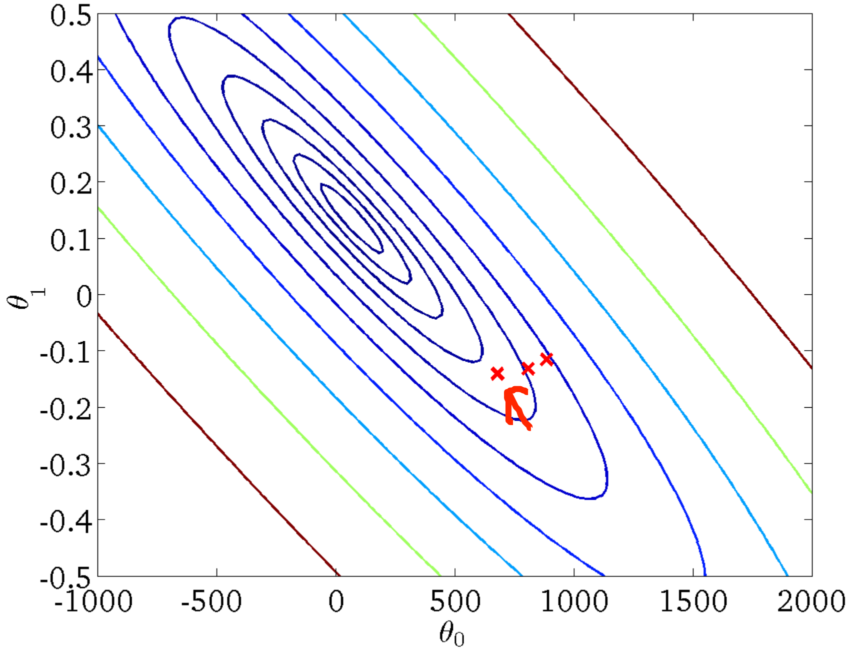
$$h_\theta(x)$$

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



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

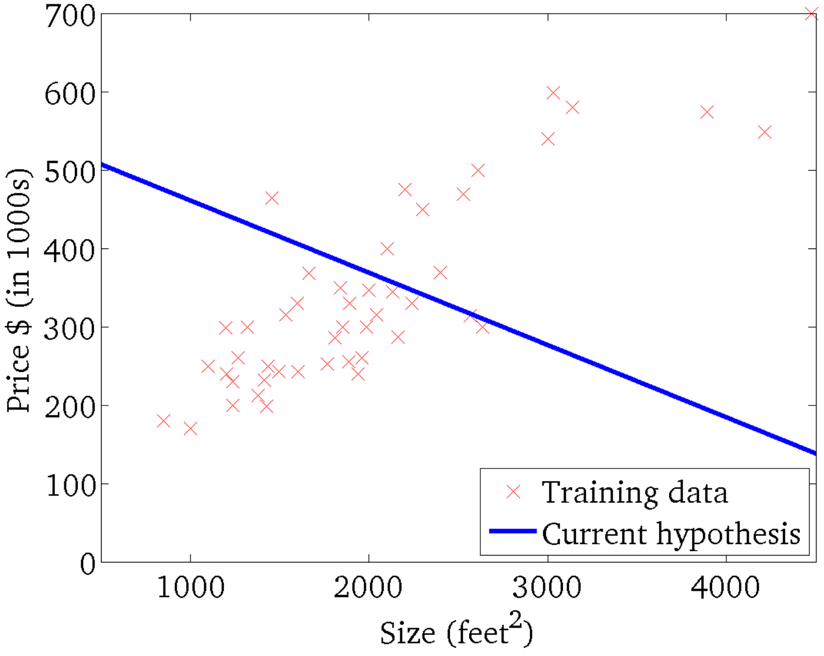
(function of the parameters θ_0, θ_1)



Andrew Ng

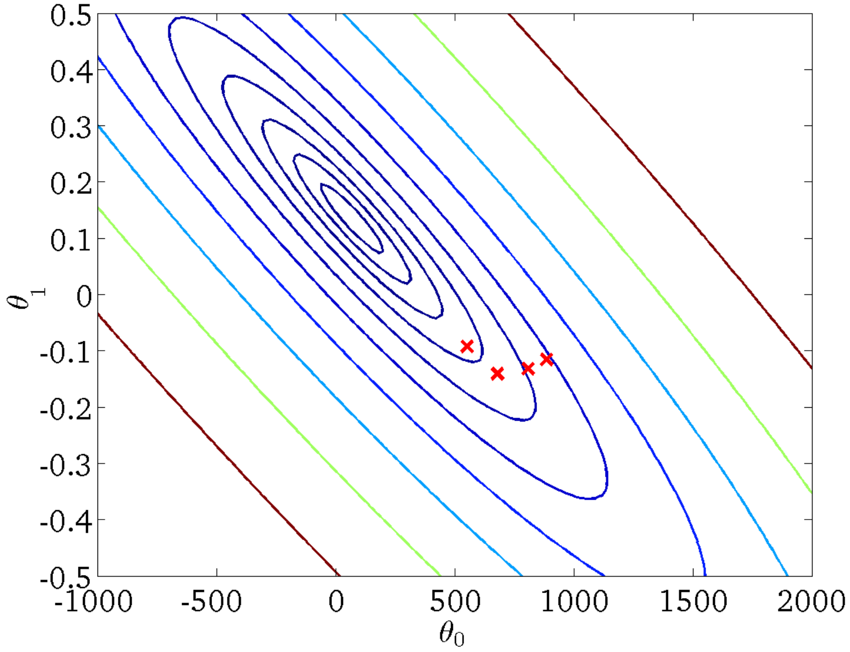
$$h_\theta(x)$$

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



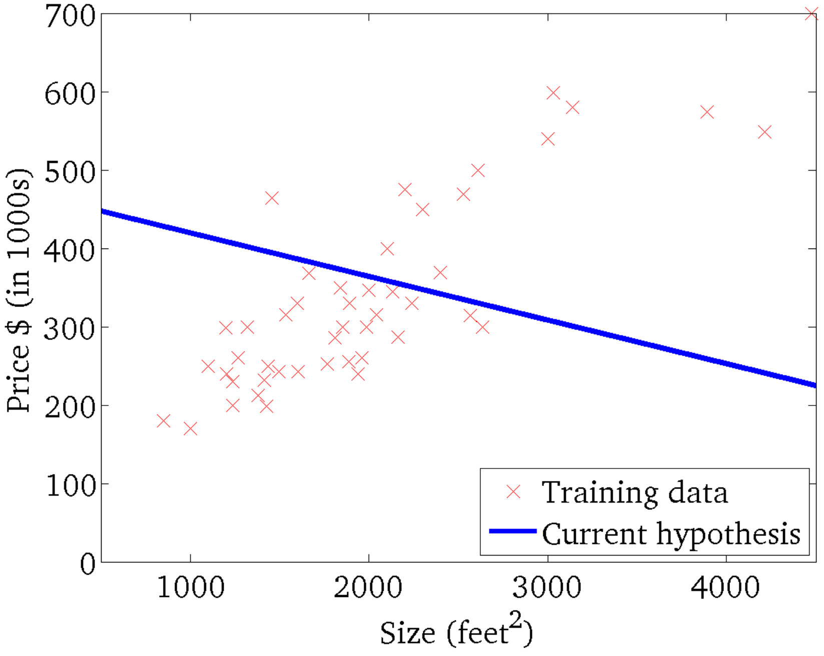
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



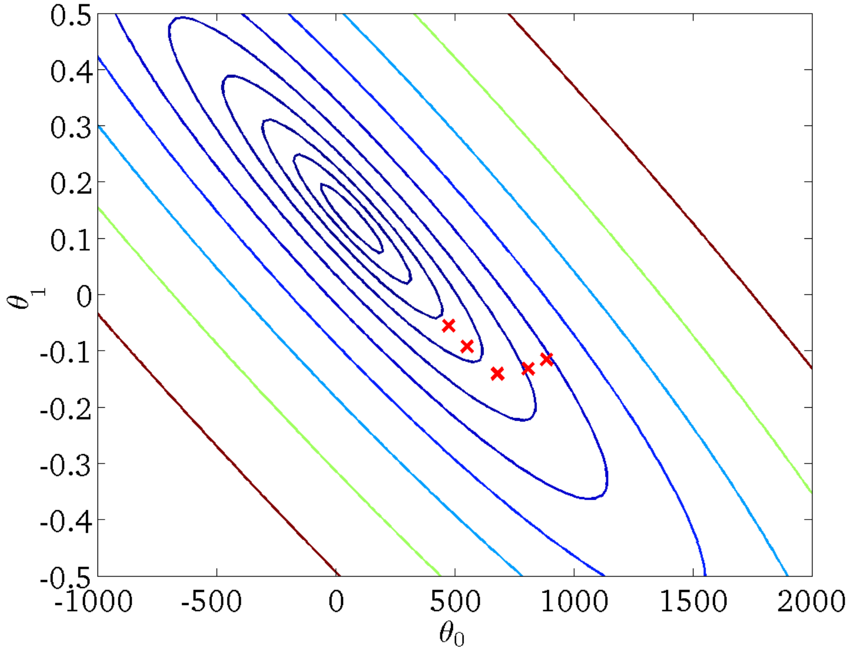
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

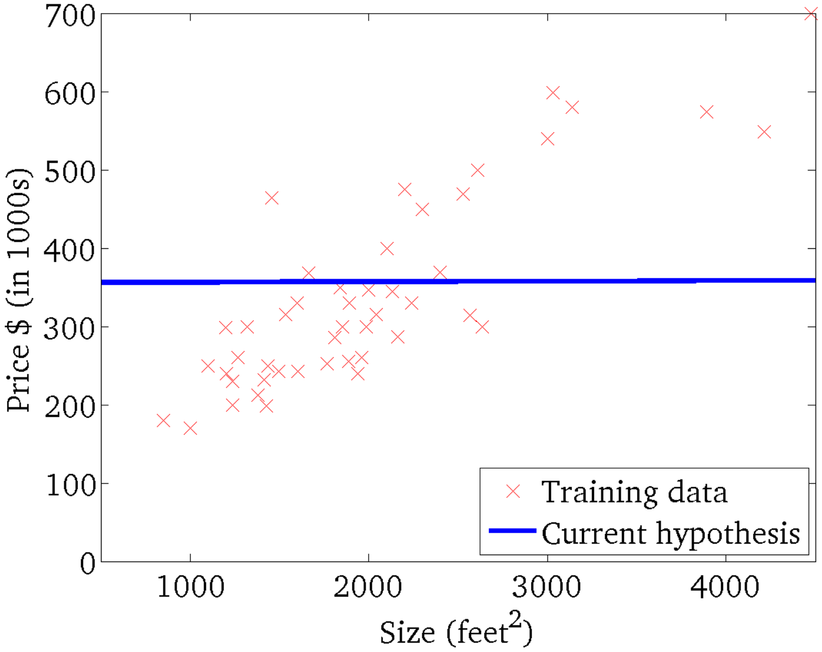
(function of the parameters θ_0, θ_1)



Andrew Ng

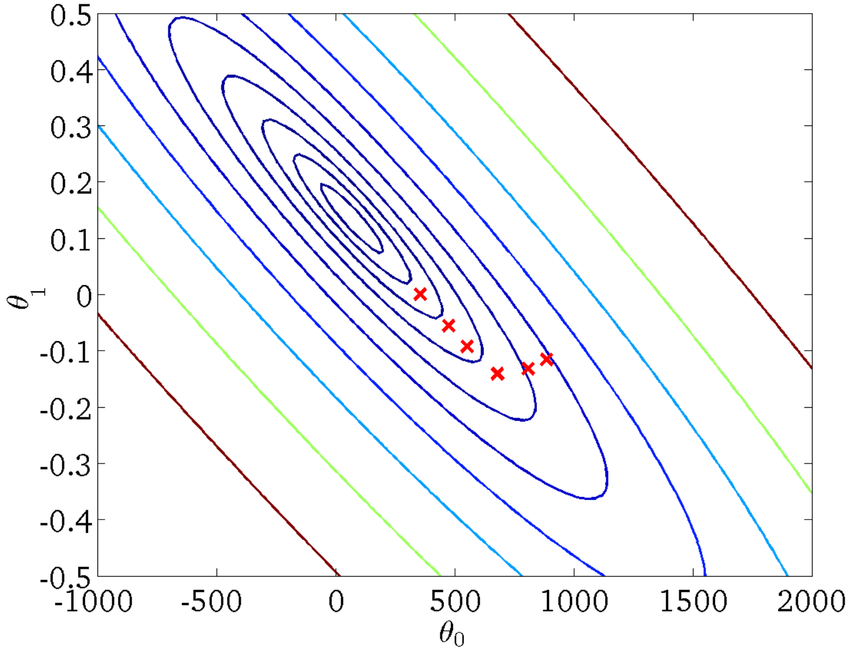
$$h_\theta(x)$$

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



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

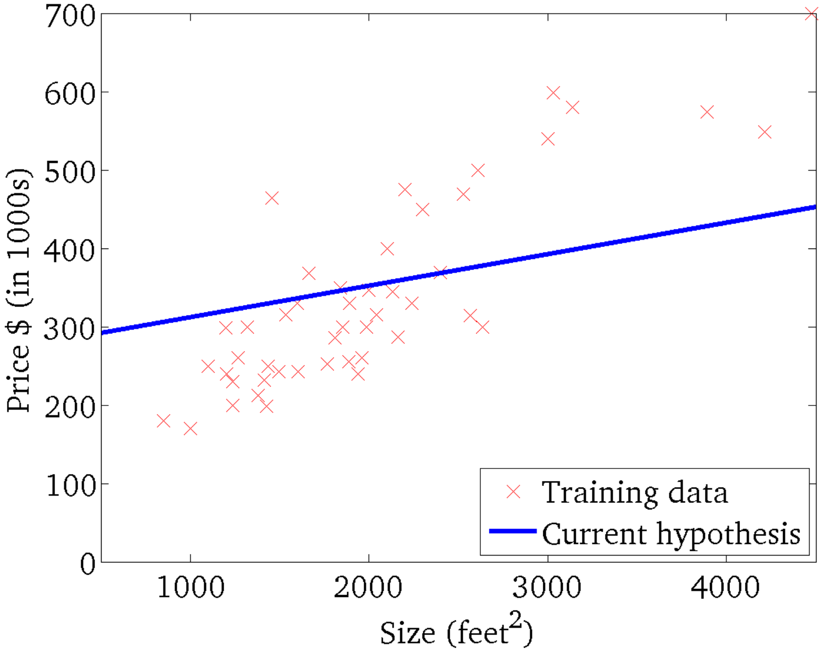
(function of the parameters θ_0, θ_1)



Andrew Ng

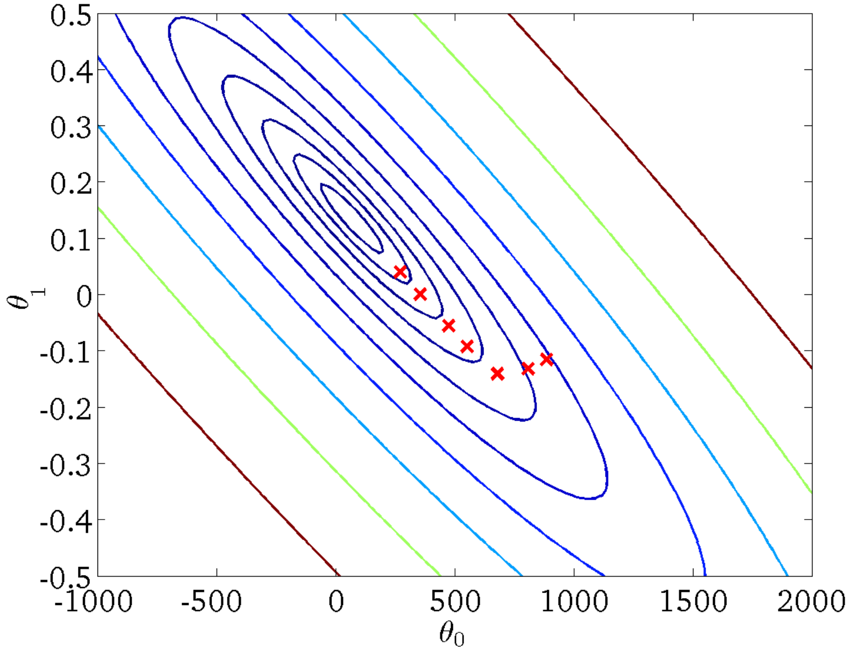
$$h_\theta(x)$$

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



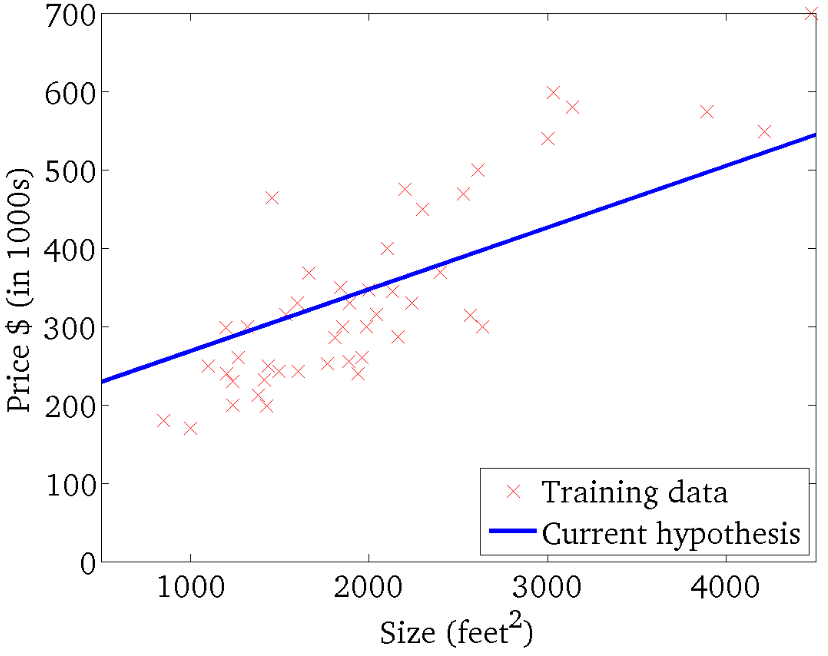
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



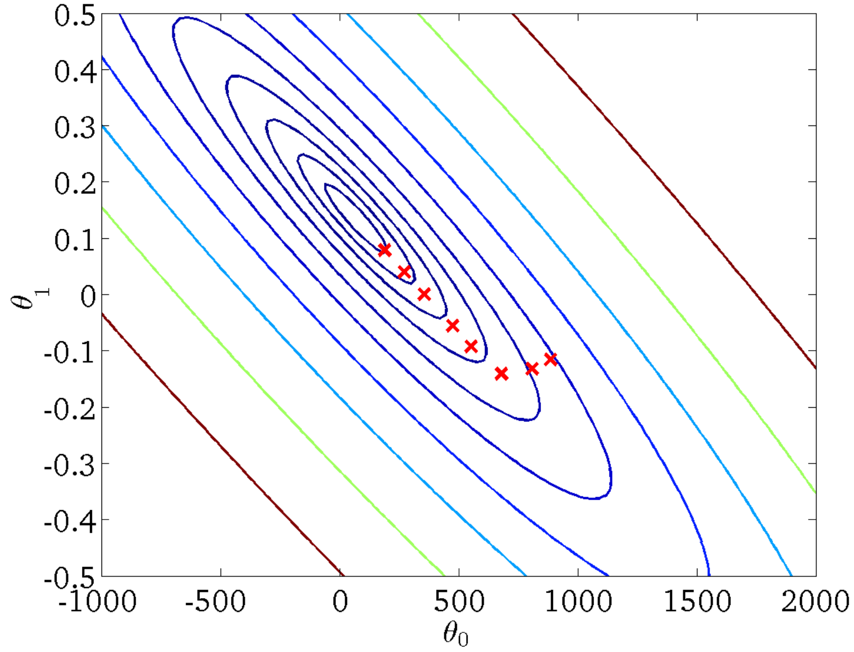
$$h_\theta(x)$$

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



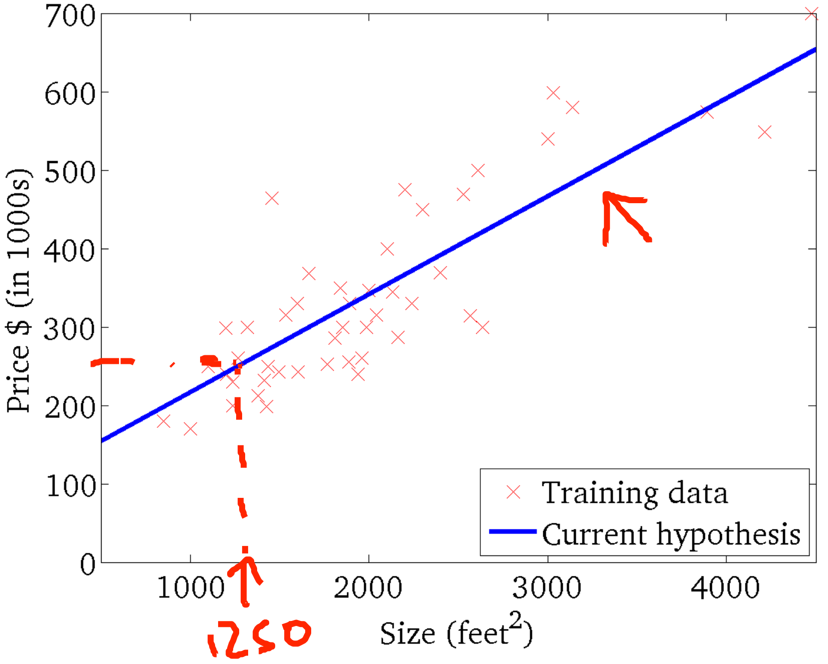
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



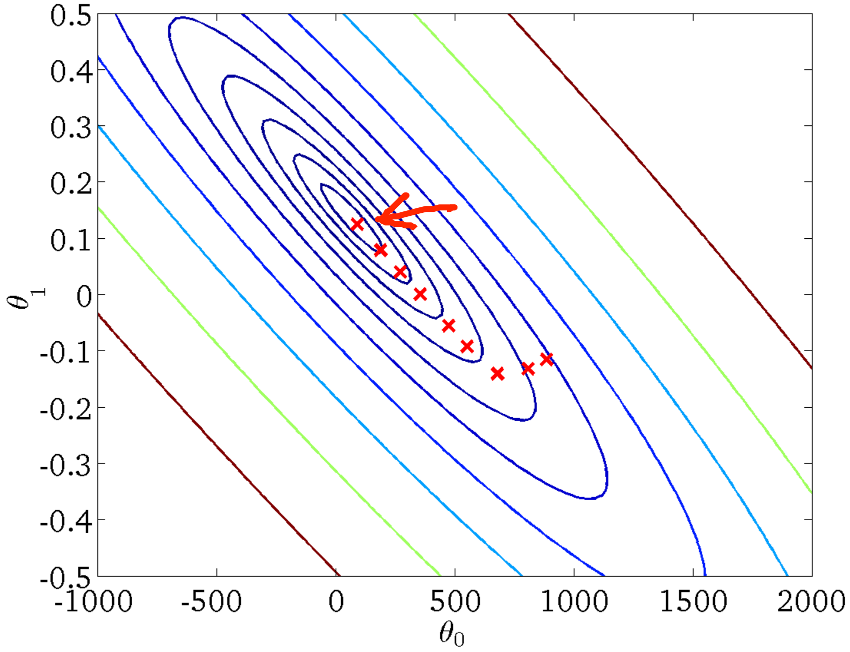
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

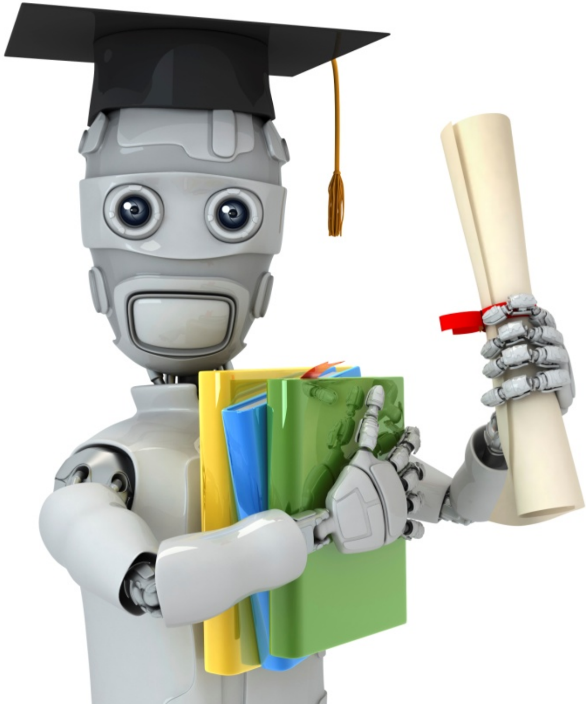


Andrew Ng

"Batch" Gradient Descent

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

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



Machine Learning

Linear Algebra review (optional)

Matrices and vectors

Matrix: Rectangular array of numbers:

$$\begin{bmatrix} 1402 & 191 \\ 1371 & 821 \\ 949 & 1437 \\ 147 & 1448 \end{bmatrix} \qquad 2 \; \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$3$$

4×2 matrix $\qquad$ 2×3 matrix

$\boxed{\mathbb{R}^{4\times 2}}$ $\qquad$ $\boxed{\mathbb{R}^{2\times 3}}$

Dimension of matrix: number of rows x number of columns

Andrew Ng

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.

$A_{11} = 1402$

$A_{12} = 191$

$A_{32} = 1437$

$A_{41} = 147$

$\cancel{A_{43}}$ = undefined (error)

Vector: An n x 1 matrix.

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \leftarrow \text{4-dimensional vector.} \quad \mathbb{R}^4$$

$n = 4$

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

$y_i = i^{th}$ element

$y_1 = 460$

$y_2 = 232$

$y_3 = 315$

$\boxed{A, B, C, X}$ $\boxed{a, b, x, y}$

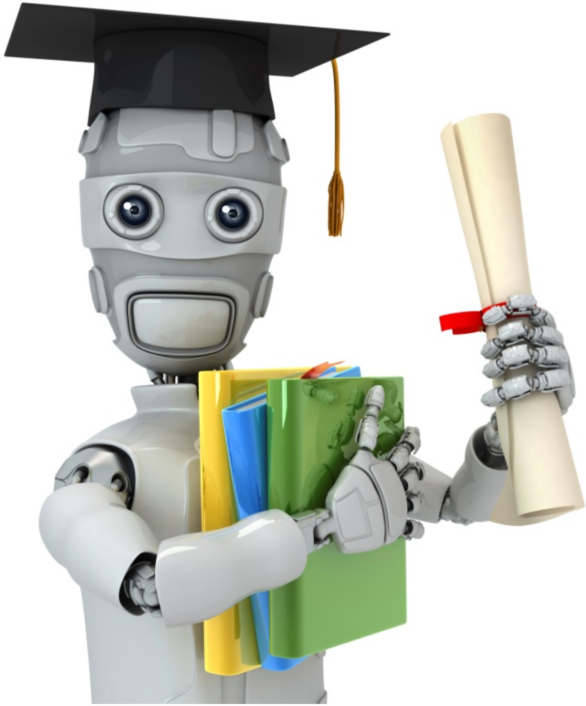
1-indexed vs 0-indexed:

$$y[1] \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

1-indexed

$$y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad y[0]$$

0-indexed



Machine Learning

Linear Algebra review (optional)

Addition and scalar multiplication

Matrix Addition

$$\underbrace{\begin{bmatrix}1 & 0\\ 2 & 5\\ 3 & 1\end{bmatrix}}_{\substack{3\times 2\\ \text{matrix}}} + \underbrace{\begin{bmatrix}4 & 0.5\\ 2 & 5\\ 0 & 1\end{bmatrix}}_{3\times 2} = \underbrace{\begin{bmatrix}5 & 0.5\\ 4 & 10\\ 3 & 2\end{bmatrix}}_{3\times 2}$$

$$\underbrace{\begin{bmatrix}1 & 0\\ 2 & 5\\ 3 & 1\end{bmatrix}}_{3\times 2} + \underbrace{\begin{bmatrix}4 & 0.5\\ 2 & 5\end{bmatrix}}_{2\times 2} = \text{error}$$

Scalar Multiplication

real number

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

3×2 3×2

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

Combination of Operands

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

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

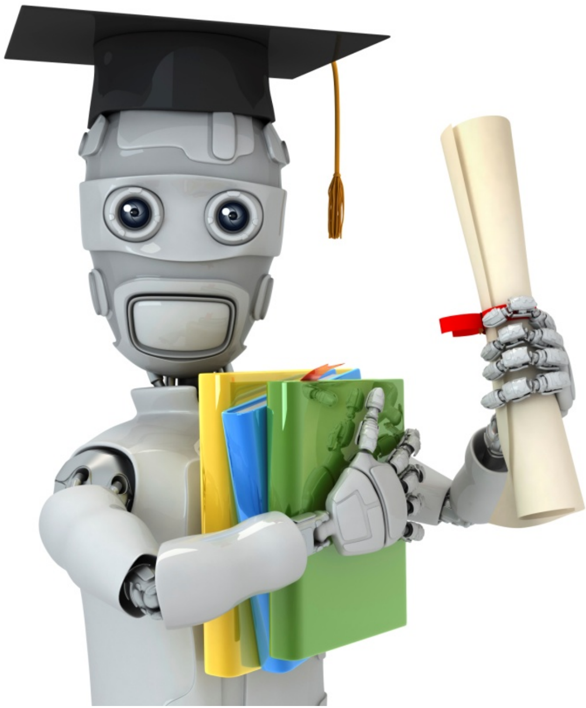
matrix addition / vector addition

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

3x1 matrix

3-dimensional vector

Andrew Ng



Machine Learning

Linear Algebra review (optional)

Matrix-vector multiplication

Example

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

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

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

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

Details:

$$A \times x = y$$

$$\begin{bmatrix} \; & \; & \; \\ \; & \; & \; \\ \; & \; & \; \end{bmatrix} \times \begin{bmatrix} \; \\ \; \\ \; \end{bmatrix} = \begin{bmatrix} \; \\ \; \\ \; \end{bmatrix}$$

m x n matrix (m rows, n columns) | n x 1 matrix (n-dimensional vector) | m-dimensional vector

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

Example

$$\begin{bmatrix} 1 & 2 & 1 & 5 \\ 0 & 3 & 0 & 4 \\ -1 & -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} \boxed{14} \\ \boxed{13} \\ \boxed{-7} \end{bmatrix} = \begin{bmatrix} 14 \\ 13 \\ -7 \end{bmatrix}$$

$\boxed{3} \times 4 \qquad 4 \times 1 \qquad 3 \times 1$

$$\left.\begin{array}{l} 1 \times 1 + 2 \times 3 + 1 \times 2 + 5 \times 1 = 14 \\ 0 \times 1 + 3 \times 3 + 0 \times 2 + 4 \times 1 = 13 \\ -1 \times 1 + (-2) \times 3 + 0 \times 2 + 0 \times 1 = -7 \end{array}\right]$$

Andrew Ng

House sizes:

→ 2104
→ 1416
→ 1534
→ 852

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

$h_\theta(x)$

4x2 Matrix

2x1 Vector

4x1 matrix

$h_\theta(2104)$

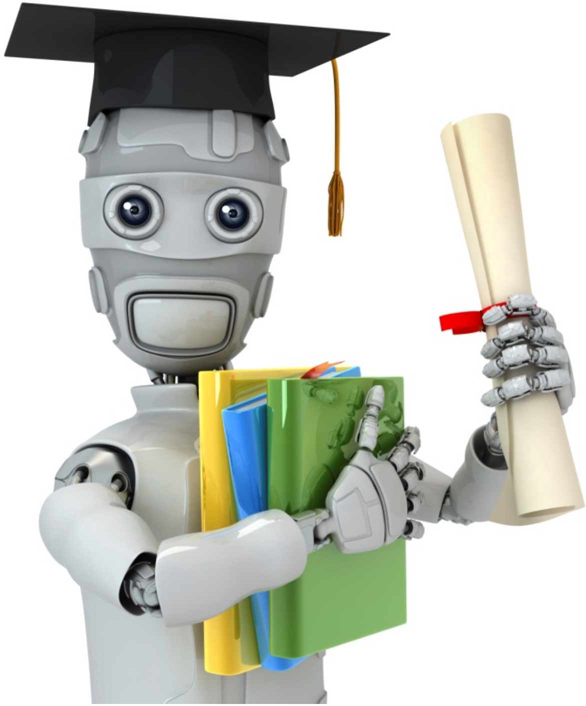
$h_\theta(1416)$

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 \\ 0.25 \end{bmatrix} = \begin{bmatrix} -40\times 1 + 0.25\times 2104 \\ -40\times 1 + 0.25\times 1416 \\ \\ \\ \end{bmatrix}$$

$$\underbrace{\text{prediction}}_{4\times 1} = \text{DataMatrix} * \text{parameters}$$

for i = 1:1,000,
prediction(i) = ...

Andrew Ng



Machine Learning

Linear Algebra review (optional)

Matrix-matrix multiplication

Example

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

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

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

Details:

$$A \times B = C$$

m x n matrix
(m rows,
n columns)

n x o matrix
(n rows,
o columns)

m x o
matrix

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

Example

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

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

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

House sizes:

$$\begin{Bmatrix} 2104 \\ 1416 \\ 1534 \\ 852 \end{Bmatrix}$$

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

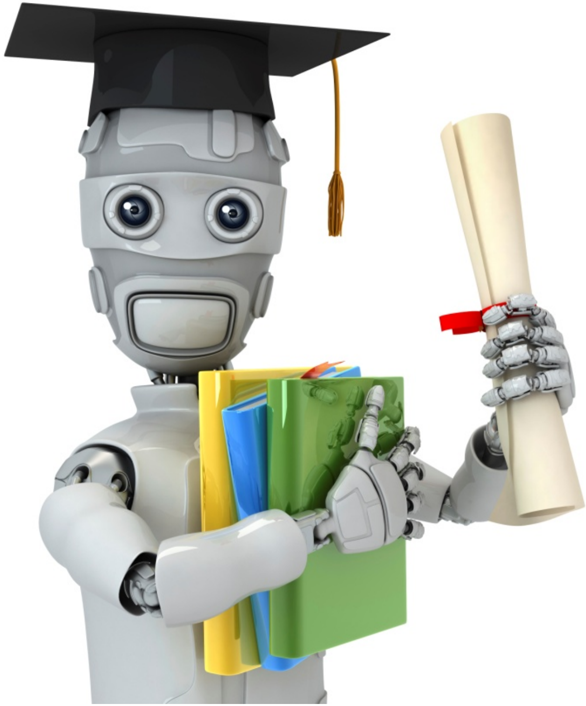
Matrix

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix} = \begin{bmatrix} 486 & 410 & 692 \\ 314 & 342 & 416 \\ 344 & 353 & 464 \\ 173 & 285 & 191 \end{bmatrix}$$

Prediction of first h_θ

Predictions of 2nd h_θ

Andrew Ng



Machine Learning

Linear Algebra review (optional)

Matrix multiplication properties

$3 \times 5 = 5 \times 3$ "Commutative"

Let A and B be matrices. Then in general,

$A \times B \neq B \times A$. (not commutative.)

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

$\neq$

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

$\underset{m \times n}{A} \times \underset{n \times m}{B}$

$A \times B$ is $m \times m$

$B \times A$ is $n \times n$

$3 \times 5 \times 2$

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

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

$\boxed{\text{"Associative"}}$

$A \times (B \times C)$

$(A \times B) \times C$

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

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

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

$A \times (B \times C)$

$(A \times B) \times C$

Same answer.

Identity Matrix

1 is identity

$\boxed{1 \times z = z \times 1 = z}$

for any z

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

Examples of identity matrices:

$[1]$

1×1

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

2 x 2   3 x 3   4 x 4

Informally:

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

For any matrix A,

$$\underset{m \times n}{A} \cdot \underset{n \times n}{\boxed{I}} = \underset{m \times m}{\boxed{I}} \cdot \underset{m \times n}{A} = \underset{m \times n}{A}$$

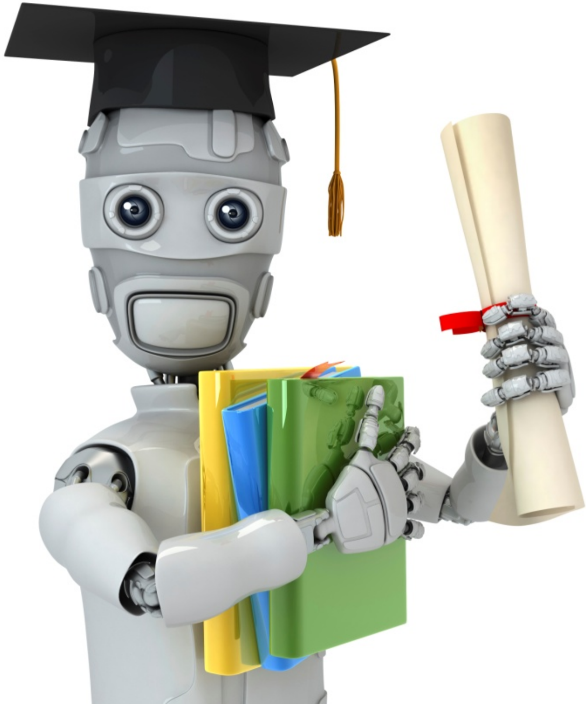
$I_{n \times n}$

Note:

$AB \neq BA$ in general

$AI = \cancel{}\ IA$ ✓

Andrew Ng



Machine Learning

Linear Algebra review (optional)

Inverse and transpose

1 = "identity"

$3 \boxed{(3^{-1})} = 1$ $\quad \frac{1}{3}$

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

$0 \, (0^{-1})$ undefined

Not all numbers have an inverse.

Matrix inverse:

square matrix (#rows = #columns)

A^{-1}

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

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

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

e.g.

$$\underbrace{\begin{bmatrix} 3 & 4 \\ 2 & 16 \end{bmatrix}}_{\substack{2\times 2 \\ A}} \underbrace{\begin{bmatrix} 0.4 & -0.1 \\ -0.05 & 0.075 \end{bmatrix}}_{A^{-1}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_{2\times 2}$$

$A^{-1}A$

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 x 3

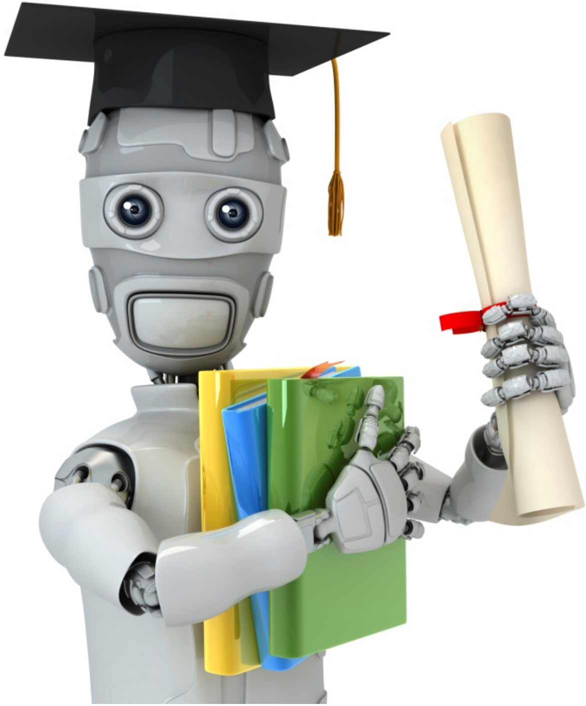
$B = \underline{A^T} = \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 0 & 9 \end{bmatrix}$ 3 x 2

Let A be an m x n matrix, and let $B = A^T$.
Then B is an n x m matrix, and

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

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

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



Machine Learning

Linear Regression with multiple variables

Multiple features

Multiple features (variables).

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

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

Multiple features (variables).

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

$m = 47$

Notation:

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

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

$$x_3^{(2)} = 2$$

Andrew Ng

Hypothesis:

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

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

E.g. $\underline{h_\theta(x)} = \underline{80} + \underline{0.1} x_1 + \underline{0.01} x_2 + 3 x_3 - 2 x_4$

age

Andrew Ng

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

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

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\underbrace{\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix}}_{\theta^T} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$(n+1) \times 1$ matrix $\qquad x$

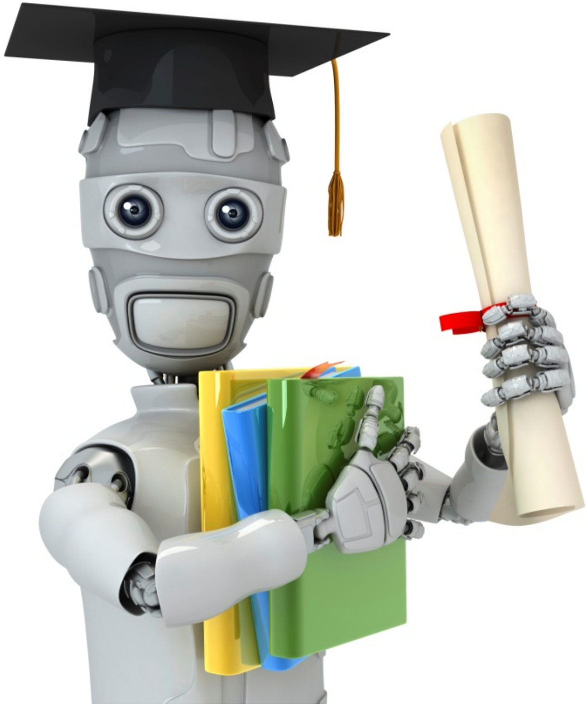
$\theta^T x$

$$h_\theta(x) = \theta_0 \underbrace{x_0}_{=1} + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$= \boxed{\theta^T x.}$$

Multivariate linear regression.

Andrew Ng



Machine Learning

Linear Regression with multiple variables

Gradient descent for multiple variables

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

$x_0 = 1$

Parameters: $\cancel{\theta_0, \theta_1, \ldots, \theta_n}$ θ n+1 - dimensional vector

Cost function:

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

$J(\theta)$

Gradient descent:

Repeat $\{$

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

$\}$

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

Gradient Descent

Previously (n=1):

Repeat {

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

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

$x^{(i)}$

(simultaneously update θ_0, θ_1)

}

New algorithm $(n \geq 1)$:

Repeat {

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

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

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

}

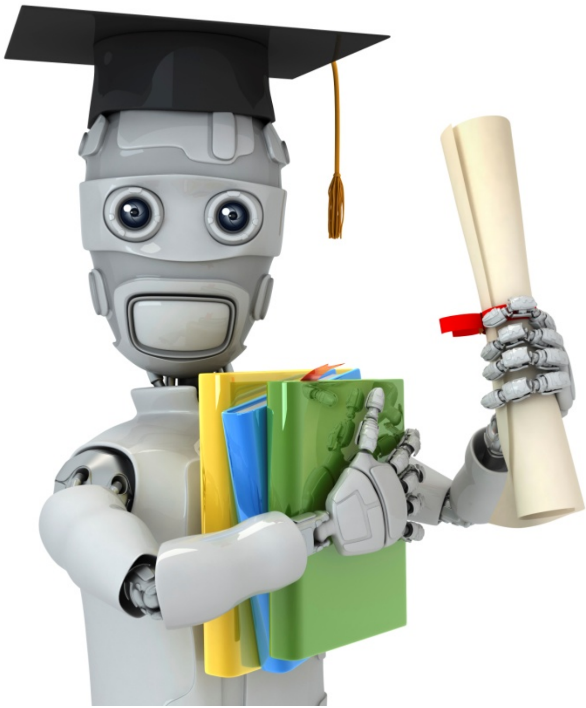
$x_0^{(i)} = 1$

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

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

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

...



Machine Learning

Linear Regression with multiple variables

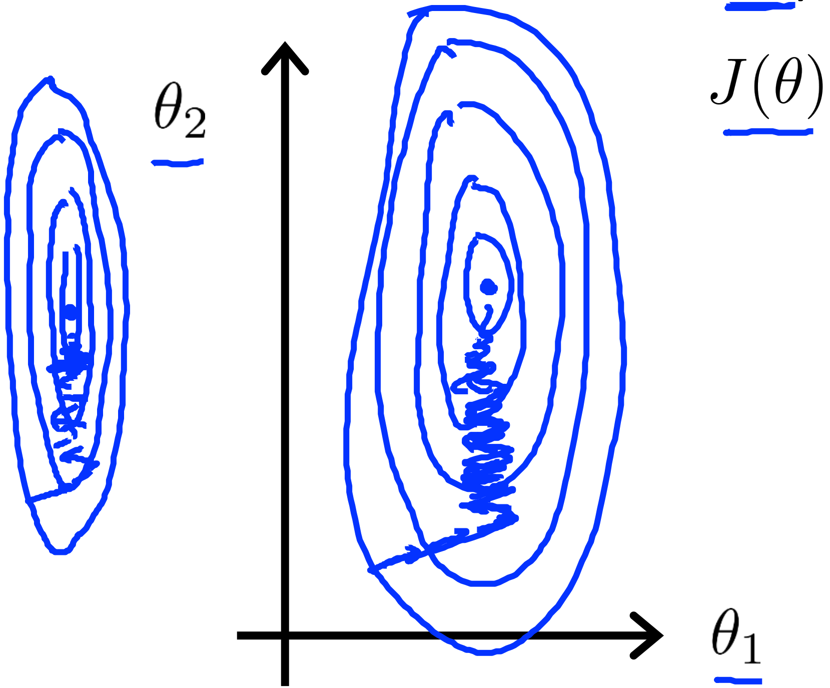
Gradient descent in practice I: Feature Scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

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

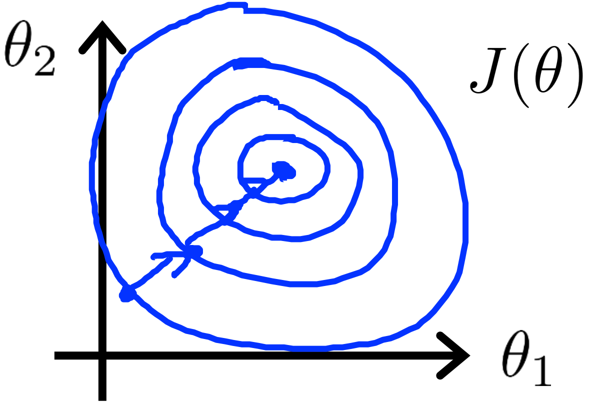
x_2 = number of bedrooms (1-5)



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

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

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



Feature Scaling

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

$x_0 = 1$

$0 \leq x_1 \leq 3$ ✓

$-2 \leq x_2 \leq 0.5$ ✓

$-\underline{100} \leq x_3 \leq \boxed{100}$ ✗

$-0.0001 \leq x_4 \leq \boxed{0.0001}$ ✗

-3 to 3 ✓

$-\frac{1}{3}$ to $\frac{1}{3}$ ✓

Mean normalization

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

E.g. $\boxed{x_1 = \frac{size - 1000}{2000}}$ Average size = 100

$\boxed{x_2 = \frac{\#bedrooms - 2}{5 \;\; 4}}$ 1-5 bedrooms

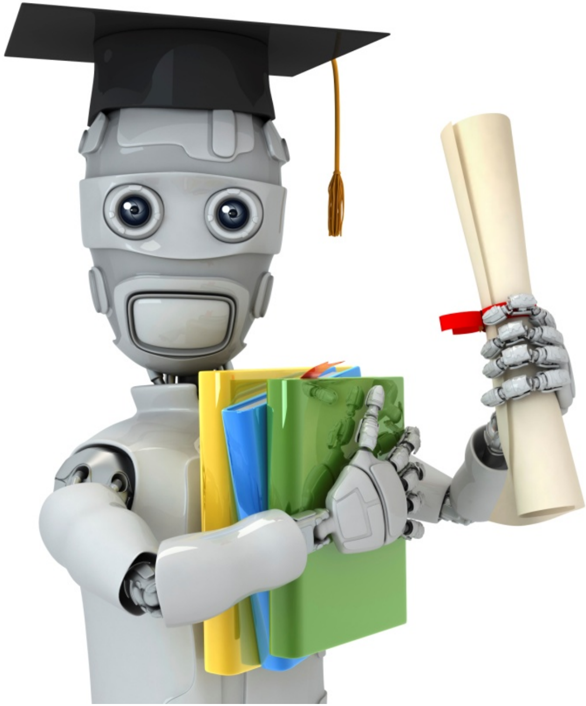
$\boxed{-0.5 \le x_1 \le 0.5}$, $\boxed{-0.5 \le x_2 \le 0.5}$

$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1}$ avg value of x_1 in training set

s_1: range (max - min) (or standard deviation)

$x_2 \leftarrow \frac{x_2 - \mu_1}{s_2}$

Andrew Ng



Machine Learning

Linear Regression with multiple variables

Gradient descent in practice II: Learning rate

Gradient descent

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

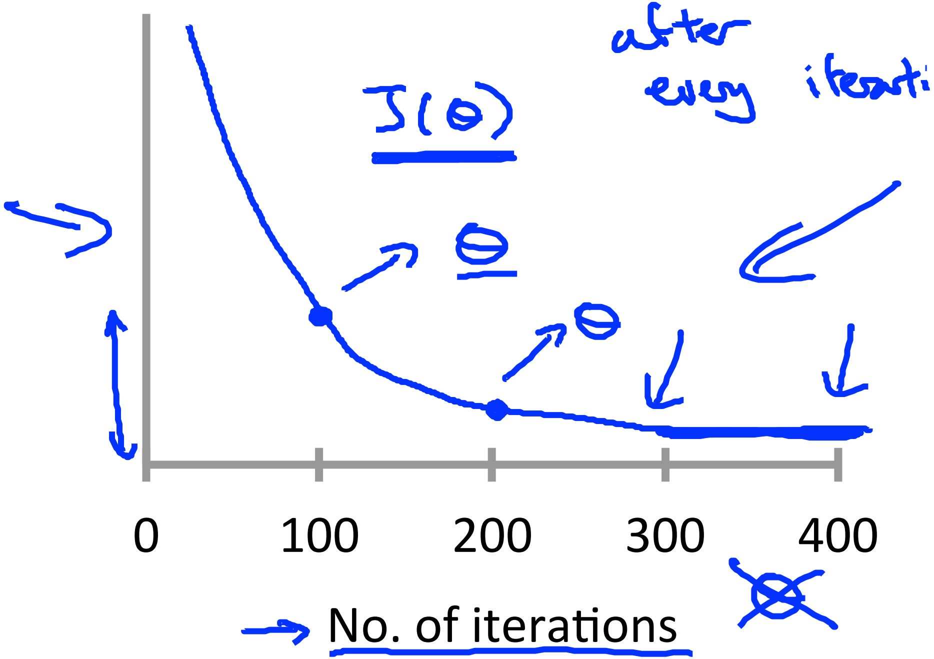
- "Debugging": How to make sure gradient descent is working correctly.

- How to choose learning rate $\boxed{\alpha.}$

Making sure gradient descent is working correctly.

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

J(θ) should decrease after every iteration



30, 3000, 3,000,000

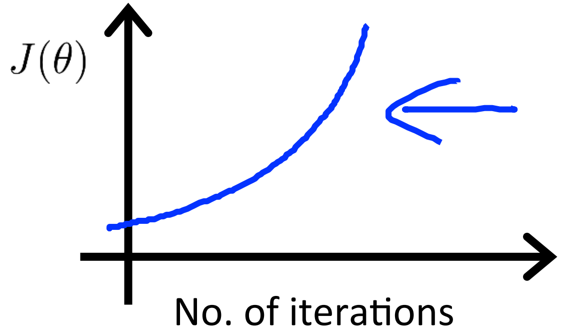
→ Example automatic convergence test:

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

ε

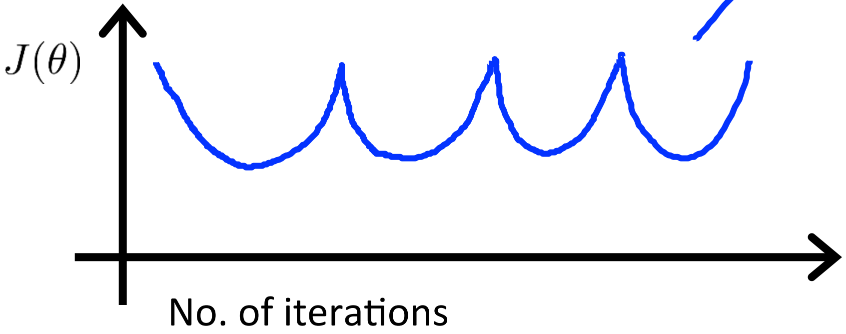
Andrew Ng

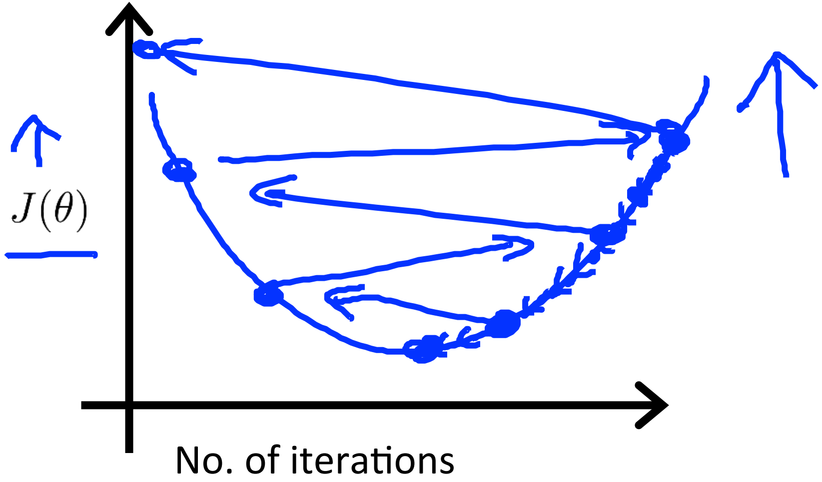
Making sure gradient descent is working correctly.



Gradient descent not working.

Use smaller α.





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

Andrew Ng

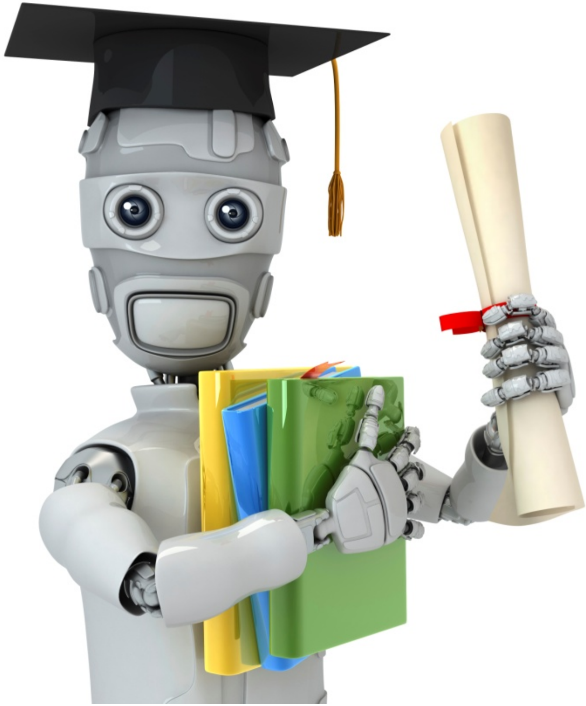
Summary:

- 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

$$\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \ldots$$

3x ≈3x 3x ≈3x



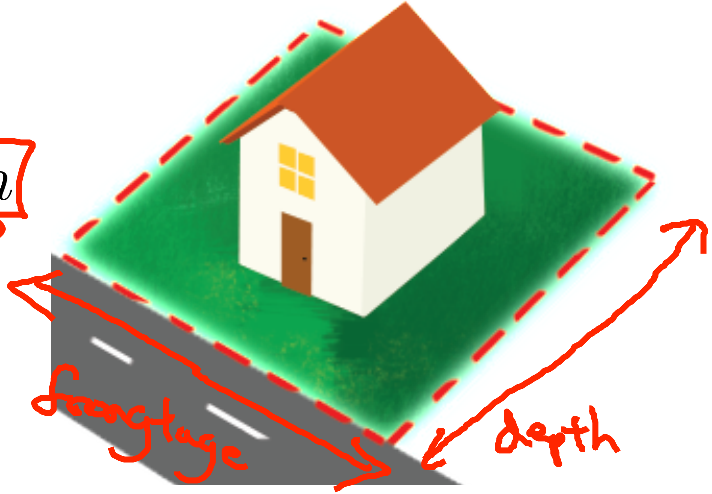
Machine Learning

Linear Regression with multiple variables

Features and polynomial regression

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\boxed{frontage}}_{x_1} + \theta_2 \times \underbrace{\boxed{depth}}_{x_2}$$



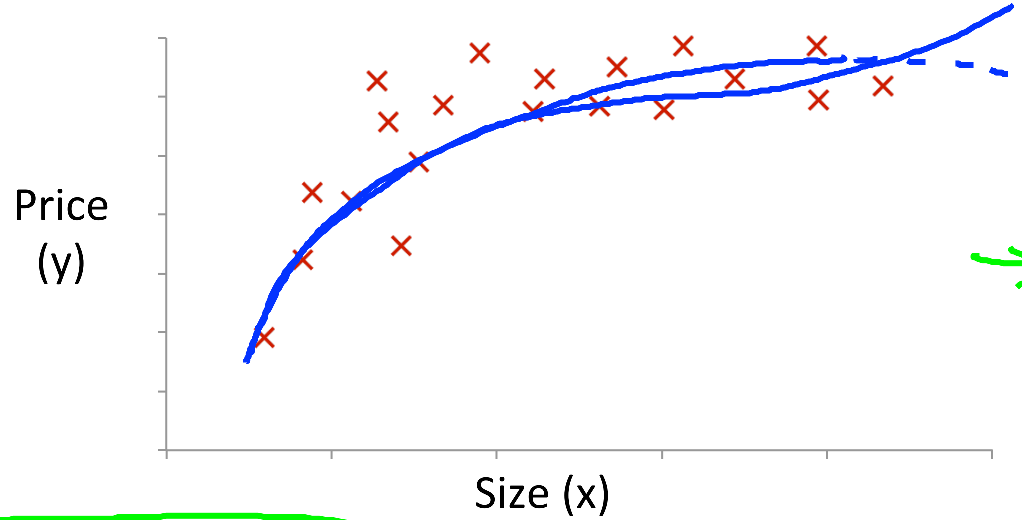
Area

$x = \underline{frontage * depth}$

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

land area

Polynomial regression



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

$$\boxed{\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3}$$

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

$$\left.\begin{aligned} x_1 &= (size) \\ x_2 &= (size)^2 \\ x_3 &= (size)^3 \end{aligned}\right\}$$

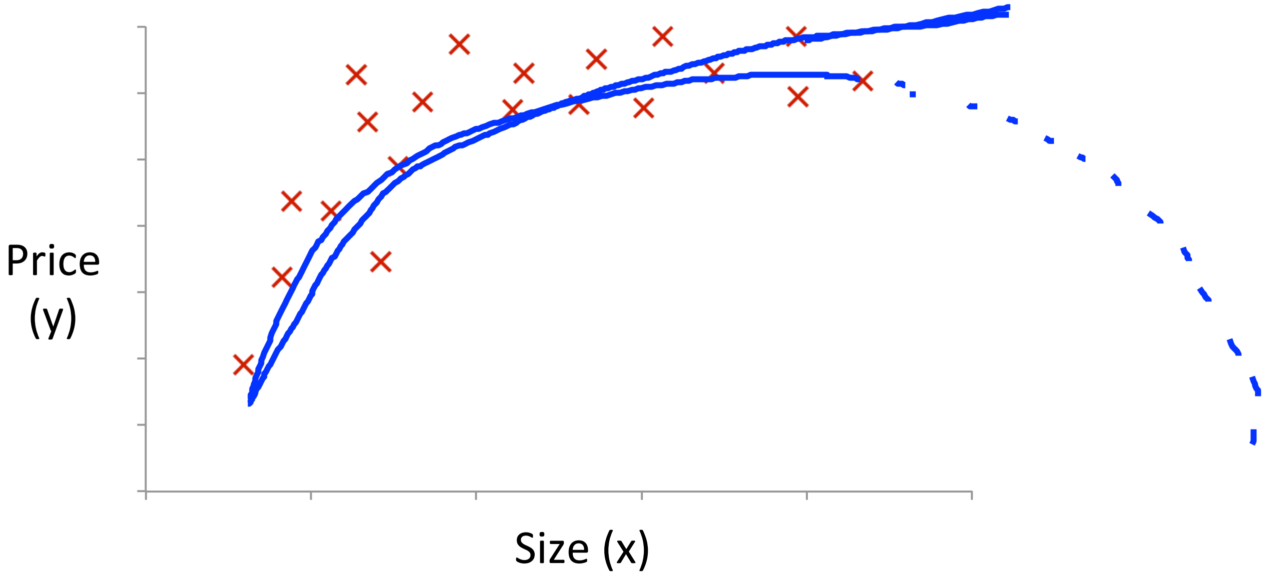
Size: 1 - 1000

$size^2$: 1 - 1000,000

$size^3$: 1 - 10^9

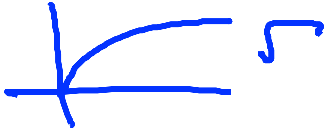
Andrew Ng

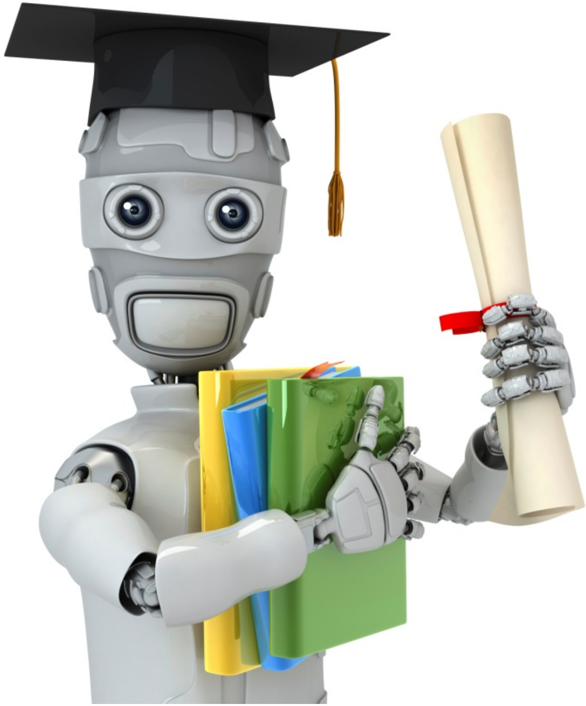
Choice of features



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

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



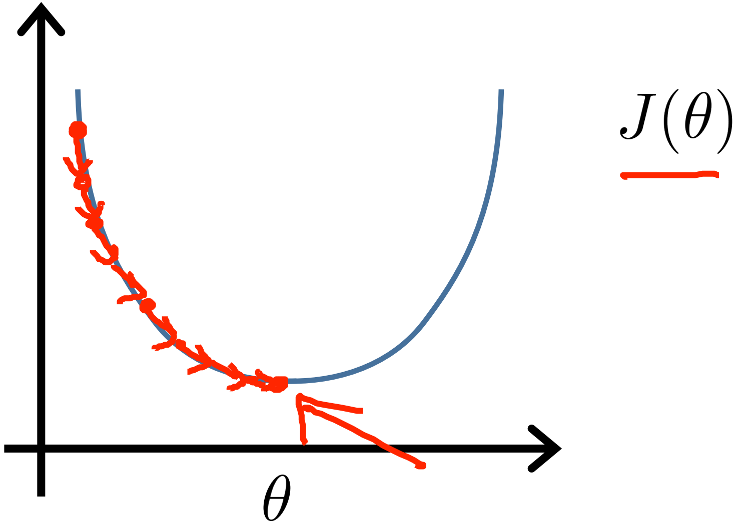


Machine Learning

Linear Regression with multiple variables

Normal equation

Gradient Descent



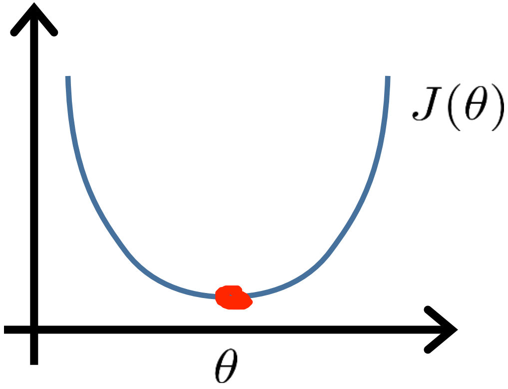
Normal equation: Method to solve for θ analytically.

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

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

$$\frac{\partial}{\partial \theta} J(\theta) = \cdots \overset{\text{set}}{=} 0$$

Solve for θ



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

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

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

Examples: $m = 4.$

	Size (feet²)	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

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

m × (n+1)

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

m-dimensional vector

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

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

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

$$X = \begin{bmatrix} \text{---} & (x^{(1)})^T & \text{---} \\ \text{---} & (x^{(2)})^T & \text{---} \\ & \vdots & \\ \text{---} & (x^{(m)})^T & \text{---} \end{bmatrix}$$

(design matrix)

$m \times (n+1)$

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

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_2^{(1)} \\ \vdots & \vdots \\ 1 & x_m^{(1)} \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$m \times 2$

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

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

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

Set $\underbrace{A}$ $=$ $\underbrace{X^T}\underbrace{X}$

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

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

pinv(X^T*X) * X^T * y

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

$\min_\Theta J(\Theta)$

X' X^T

~~Feature Scaling~~

$0 \le x_1 \le 1$

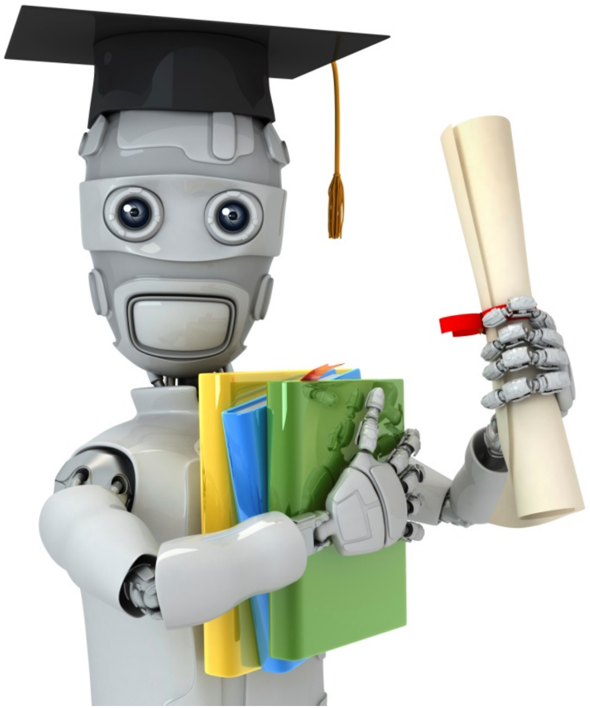
$0 \le x_2 \le 1000$

$0 \le x_3 \le 10^{-5}$ ✓

m training examples, n features.

Gradient Descent	Normal Equation
• Need to choose α.	• No need to choose α.
• Needs many iterations.	• Don't need to iterate.
• Works well even when n is large.	• Need to compute $\boxed{(\boxed{X^T X})^{-1}}$ $n \times n$ $O(n^3)$
$n = 10^6$	• Slow if n is very large.
	$n = 100$
	$n = 1000$
	$n = 10000$

Andrew Ng



Machine Learning

Linear Regression with multiple variables

Normal equation and non-invertibility (optional)

Normal equation

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

$X^T X$

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

$\boxed{\text{pinv}}$

inv

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

- Redundant features (linearly dependent).
  E.g. x_1 = size in feet^2
  ~~x_2 = size in m^2~~

  $x_1 = (3.28)^2 x_2$

1m = 3.28 feet

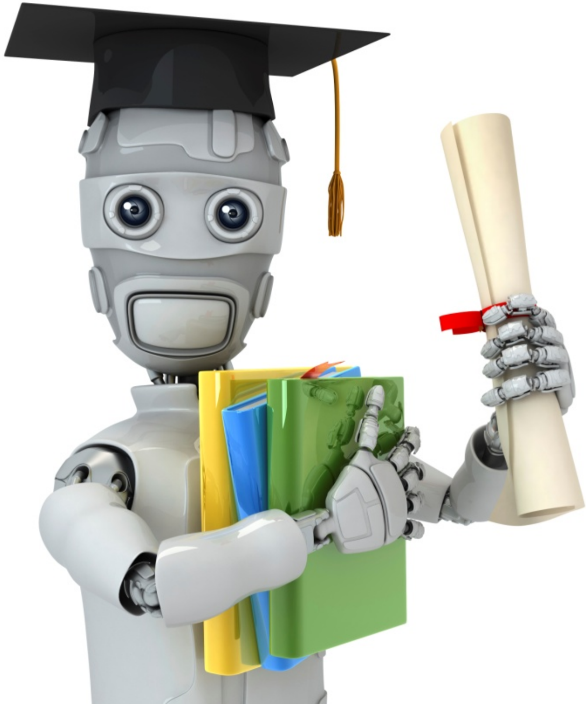
→ $m = 10$ ←

→ $n = 100$ ←

$\theta \in \mathbb{R}^{101}$

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

↓ later



Machine Learning

Logistic Regression

Classification

Classification

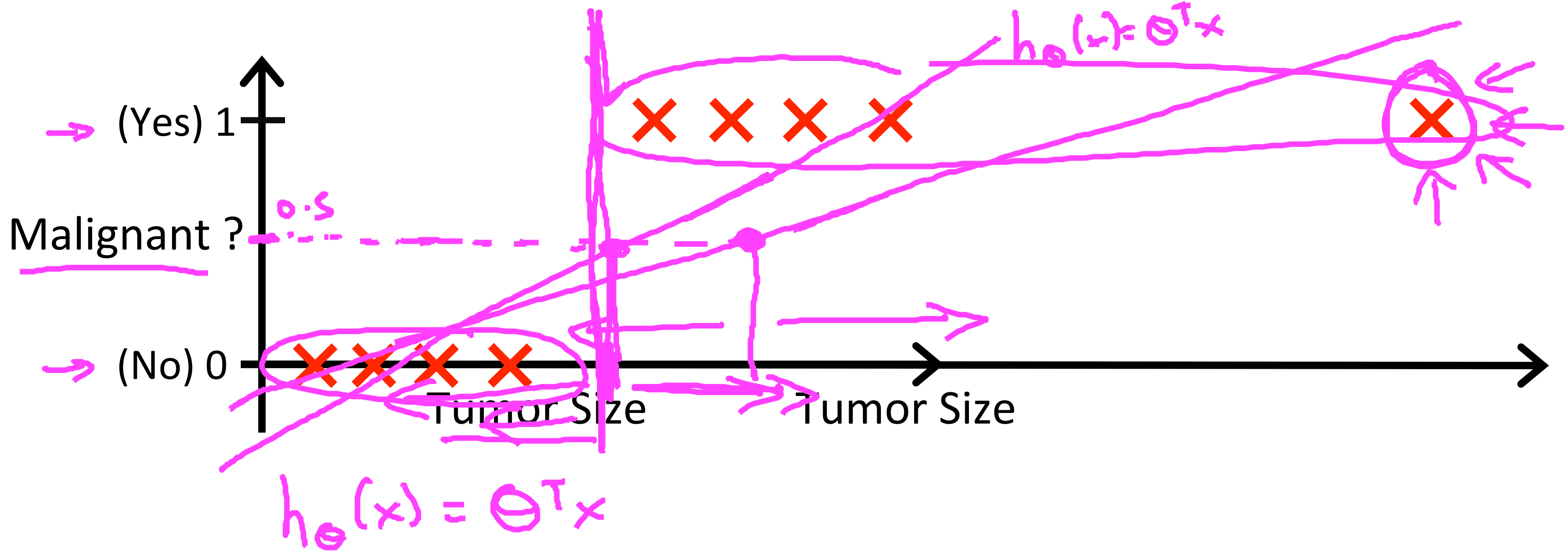
- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign ?

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

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"

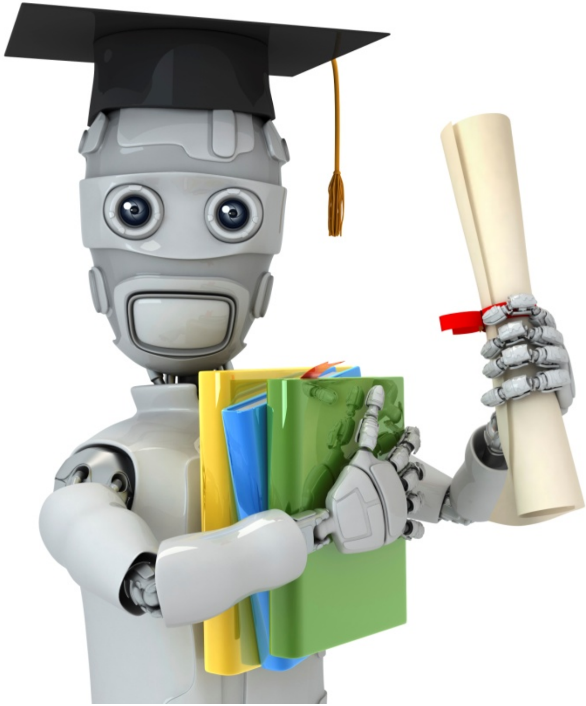
If $h_\theta(x) < 0.5$, predict "y = 0"

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

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

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

Classification



Machine Learning

Logistic Regression

Hypothesis Representation

Logistic Regression Model

Want $\boxed{0 \leq h_\theta(x) \leq 1}$

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

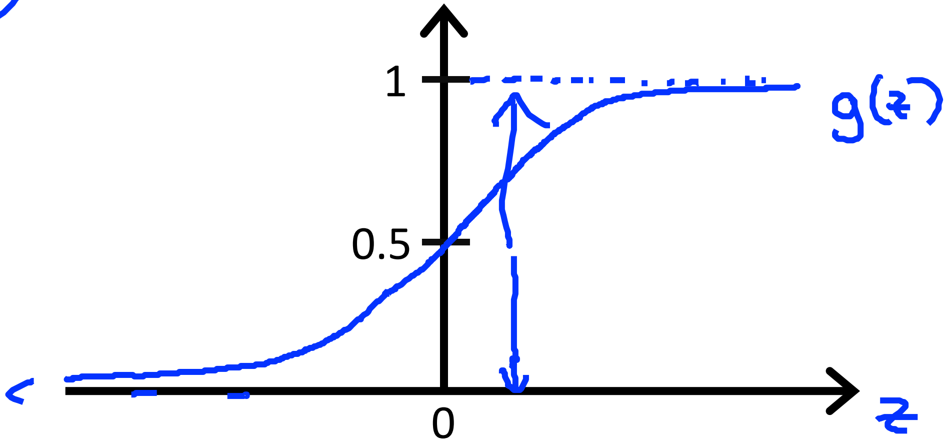
$$g(z) = \frac{1}{1+e^{-z}}$$

$\theta^T x$

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

Sigmoid function

Logistic function



Parameters θ.

Interpretation of Hypothesis Output

$h_\theta(x)$

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

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

$$h_\theta(x) = 0.7 \qquad y=1$$

Tell patient that 70% chance of tumor being malignant

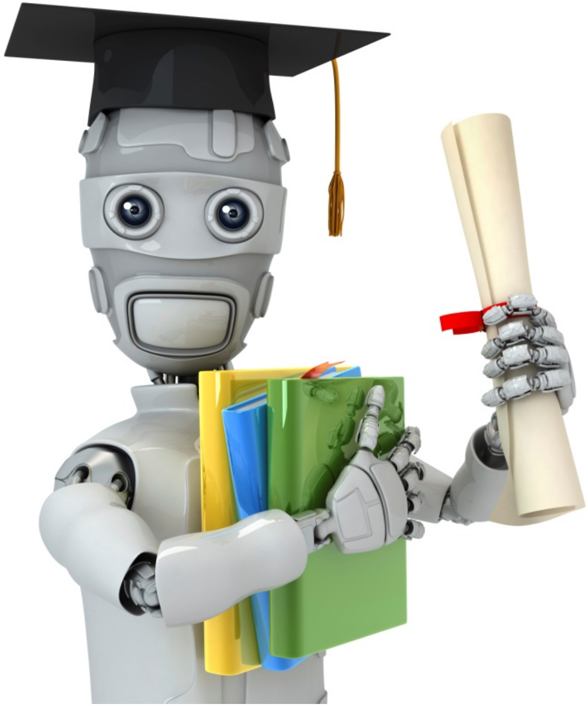
$h_\theta(x) = P(y=1|x;\theta)$

$y = 0$ or 1

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

$$P(y=0|x;\theta) + \boxed{P(y=1|x;\theta)} = 1$$

$$P(y=0|x;\theta) = 1 - \boxed{P(y=1|x;\theta)}$$



Machine Learning

Logistic Regression

Cost function

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

m examples

$$x \in \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}}_{}\ \mathbb{R}^{n+1} \qquad x_0 = 1, y \in \{0, 1\}$$

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

How to choose parameters θ ?

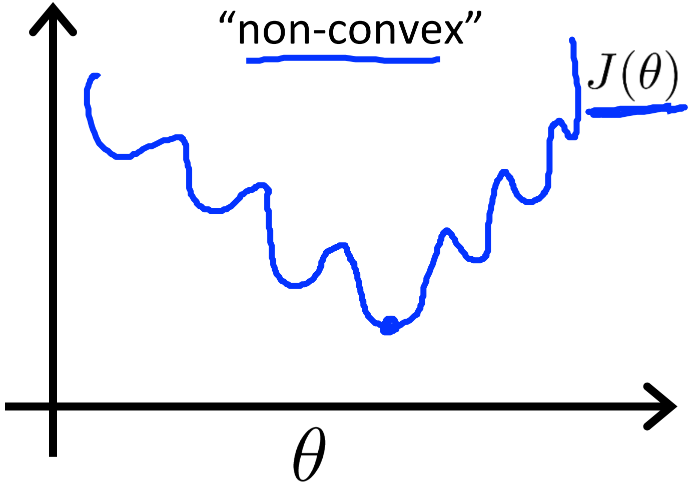
Cost function

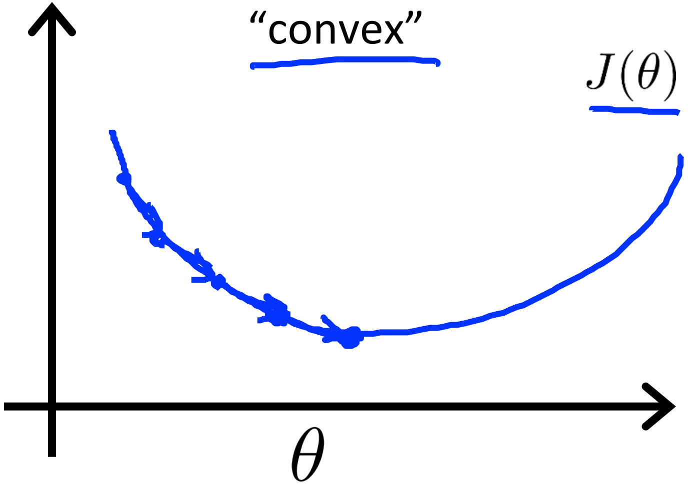
→ ~~Linear~~ regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \cancel{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}$

Logistic

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

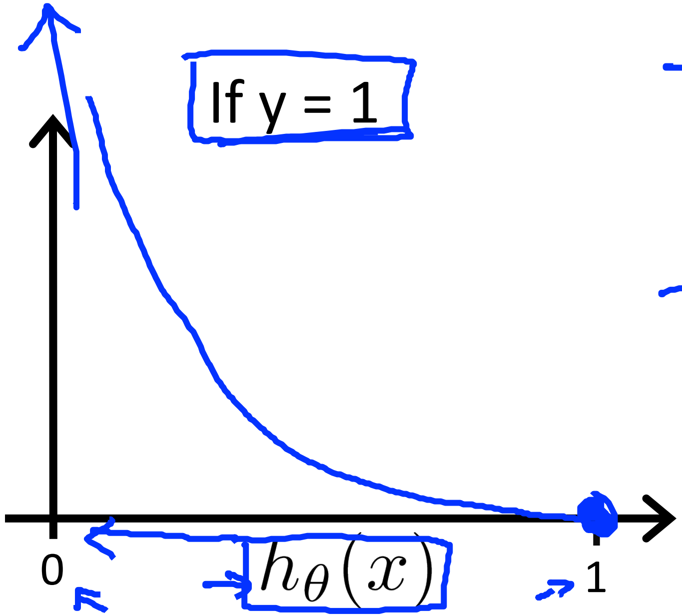
$\text{Cost}(h_\theta(x), y) = \frac{1}{2}\left(h_\theta(x) - y\right)^2$ ← $\frac{1}{1+e^{-\theta^T x}}$





Logistic regression cost function

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



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

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

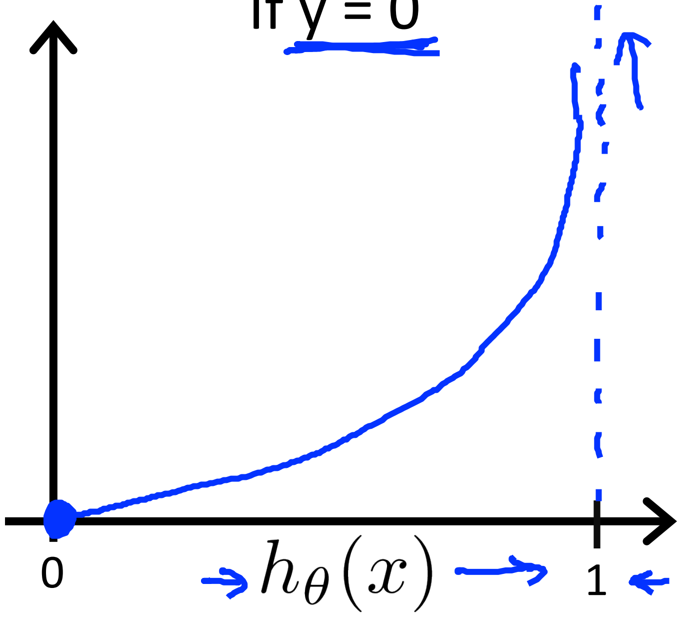
$Cost \to \infty$

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

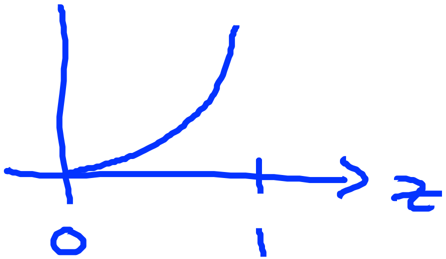
Logistic regression cost function

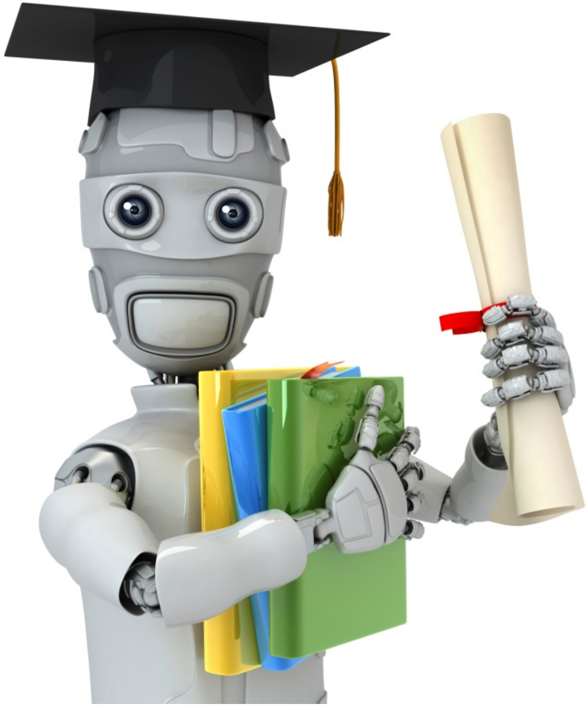
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ \boxed{-\log(1 - h_\theta(x))} & \text{if } y = 0 \end{cases}$$

If y = 0



$-\log(1-z)$





Machine Learning

Logistic Regression

Simplified cost function and gradient descent

Logistic regression cost function

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

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

Note: $y = \boxed{0}$ or $\boxed{1}$ always

$$\rightarrow \mathrm{Cost}(h_\theta(x), y) = -\underbrace{y \log(h_\theta(x))}_{=0} - \underbrace{(1-y)}_{=0} \underbrace{\log(1 - h_\theta(x))}_{=1}$$

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

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

Logistic regression cost function

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

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

To fit parameters θ :

$$\boxed{\min_\theta J(\theta) \quad \text{Get } \theta}$$

To make a prediction given new x:

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

$p(y=1 \mid x;\theta)$

Gradient Descent

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

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

Repeat $\{$

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

(simultaneously update all θ_j)

$\}$

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

Gradient Descent

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

for $i = 0$ to n

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

Repeat $\{$

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

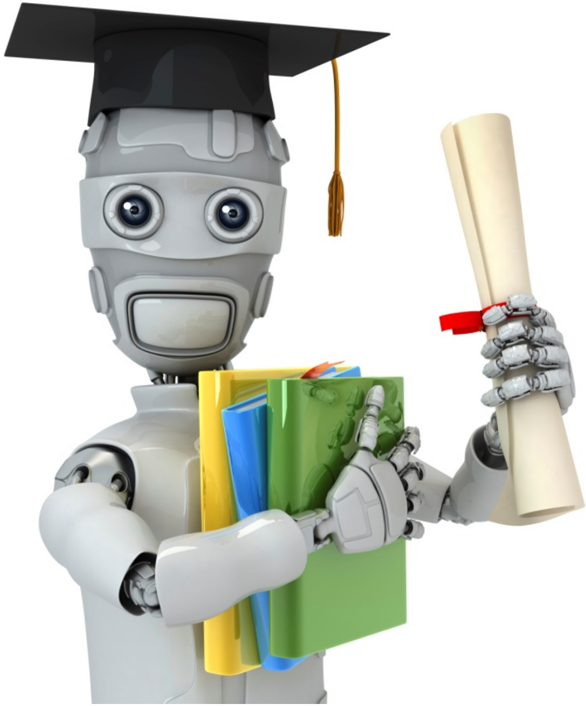
$h_\theta(x) = \theta^T x$

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

(simultaneously update all θ_j)

$\}$

Algorithm looks identical to linear regression!



Machine Learning

Logistic Regression

Advanced optimization

Optimization algorithm

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

Given θ, we have code that can compute

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

Gradient descent:

Repeat $\{$

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

$\}$

Andrew Ng

Optimization algorithm

Given θ, we have code that can compute

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

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

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

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

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

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

```
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);
```

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

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

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

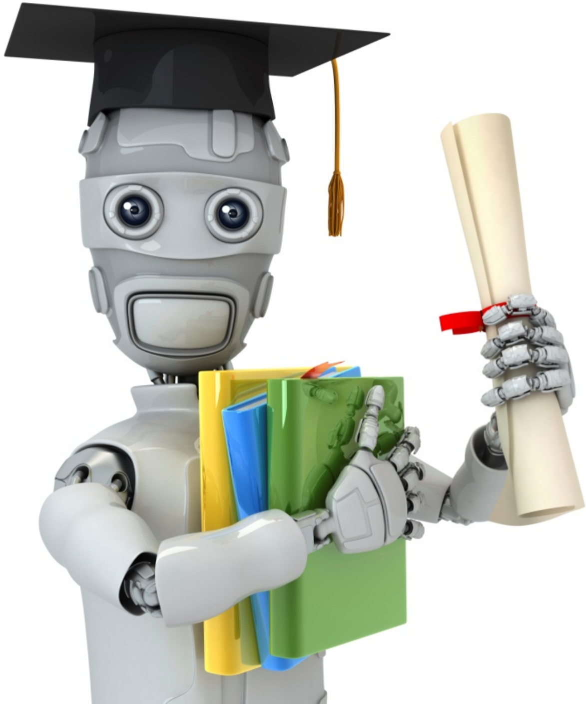
```
function [jVal, gradient] = costFunction(theta)

    jVal = [code to compute J(θ)];

    gradient(1) = [code to compute ∂/∂θ0 J(θ)];

    gradient(2) = [code to compute ∂/∂θ1 J(θ)];
      ⋮
    gradient(n+1) = [code to compute ∂/∂θn J(θ)];
```

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

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

Medical diagrams: Not ill, Cold, Flu

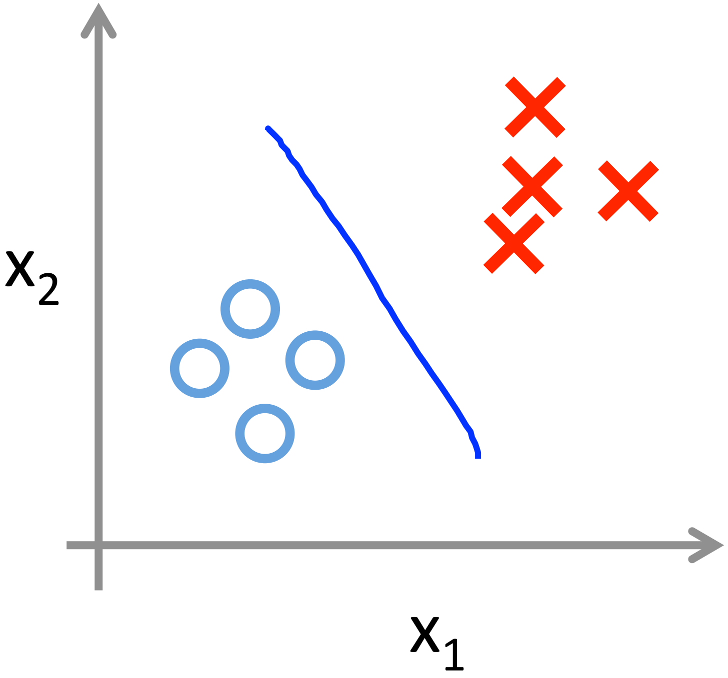
$y=1$ $\quad 2$ $\quad 3$

Weather: Sunny, Cloudy, Rain, Snow

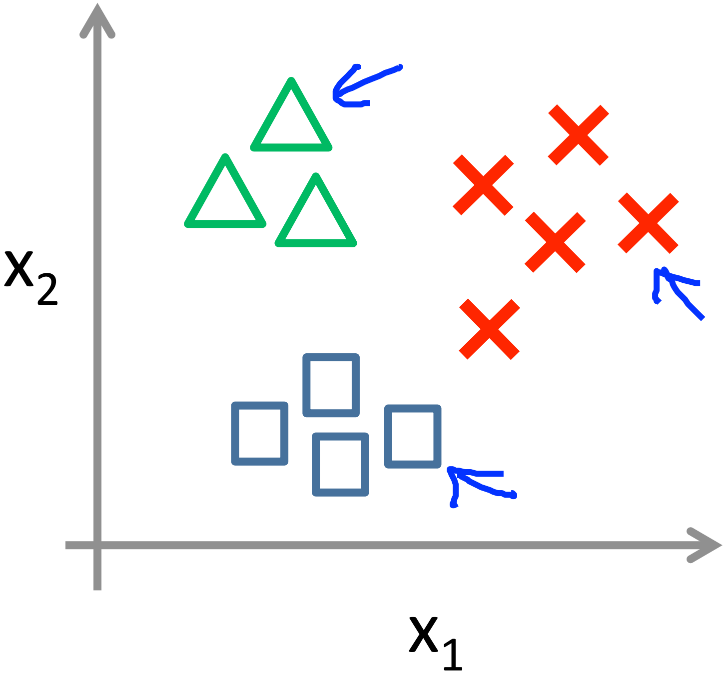
$y=1$ $\quad 2$ $\quad 3$ $\quad 4$

0 $\quad 1$ $\quad 2$ $\quad 3$

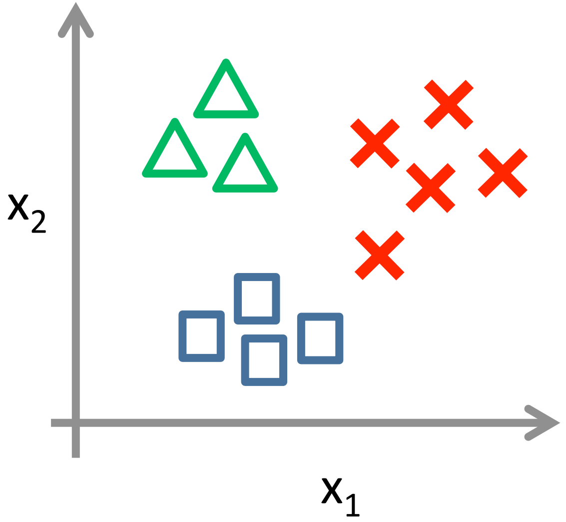
Binary classification:

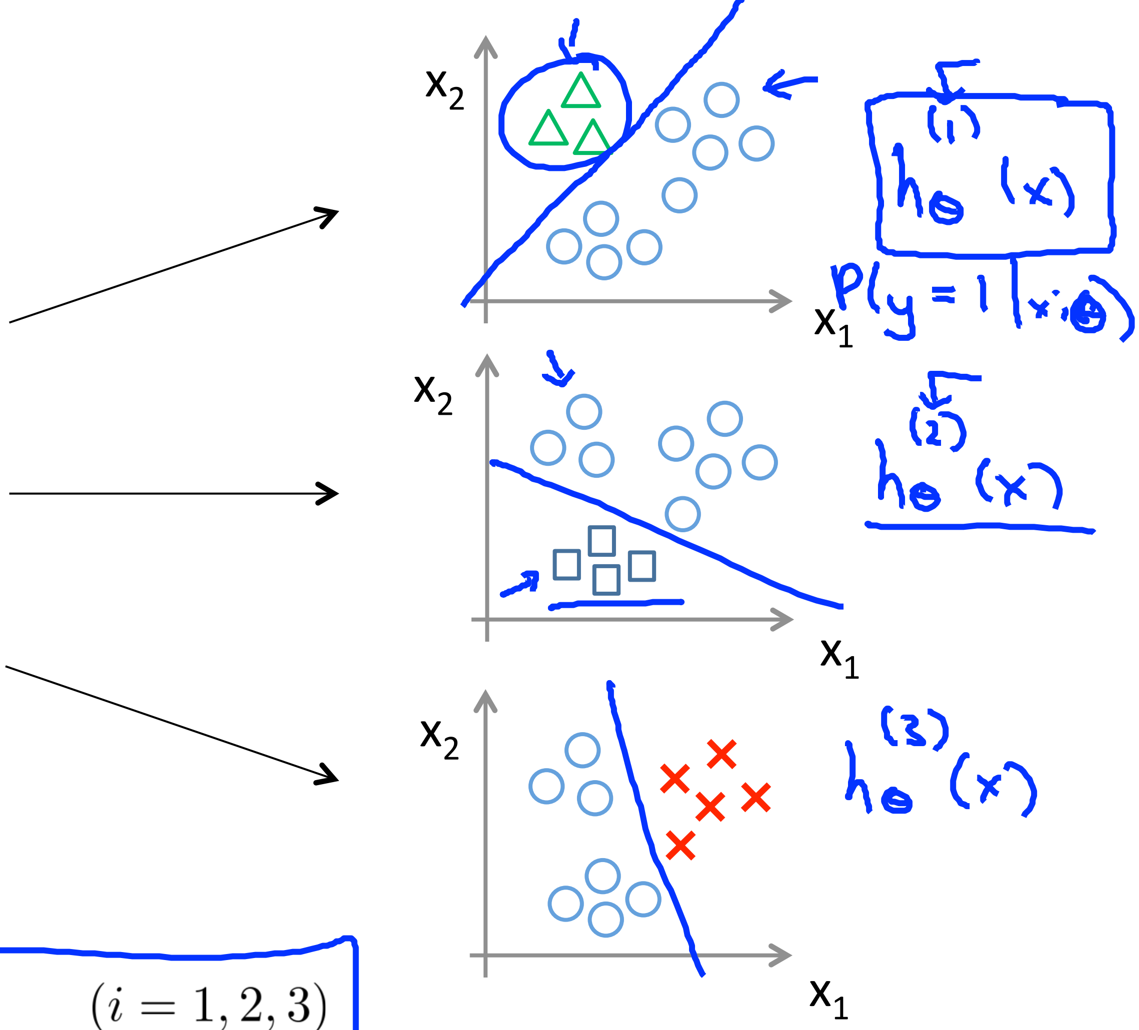


Multi-class classification:



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





Class 1: △ ←

Class 2: □ ←

Class 3: ✕ ←

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

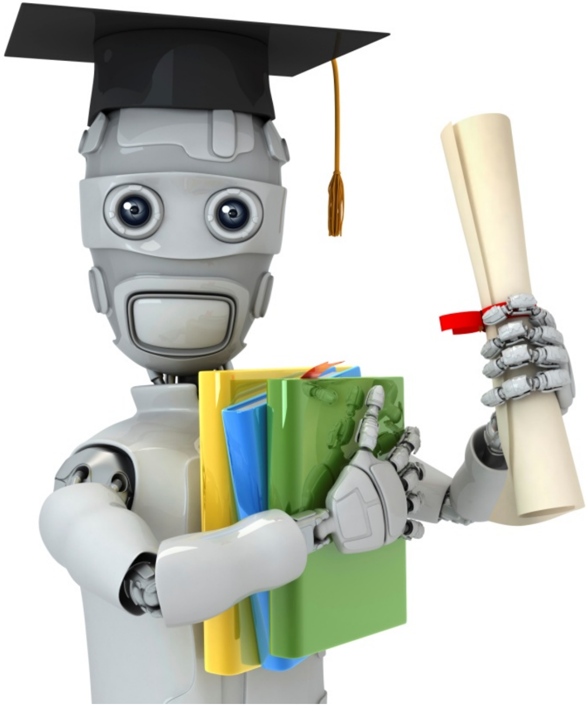
Andrew Ng

One-vs-all

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

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

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

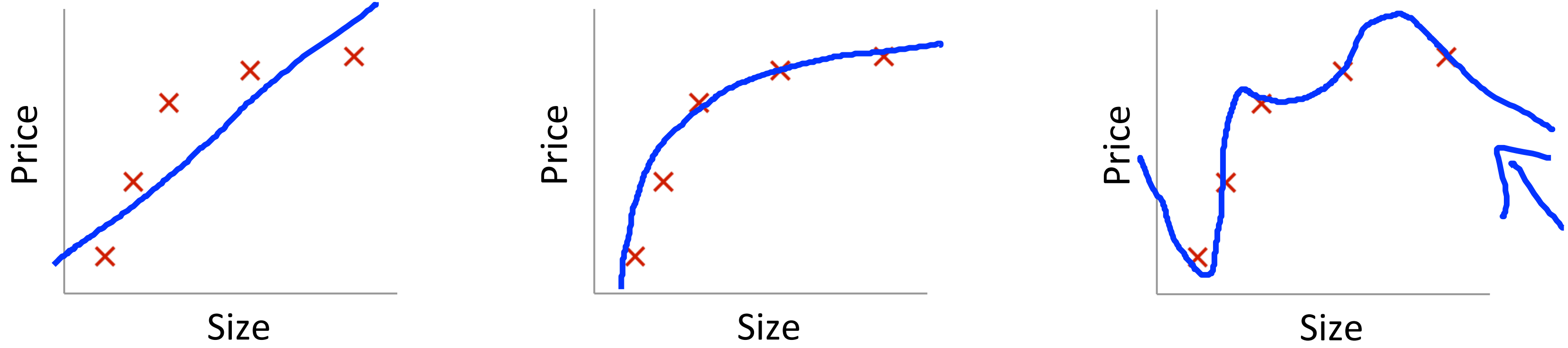


Machine Learning

Regularization

The problem of overfitting

Example: Linear regression (housing prices)



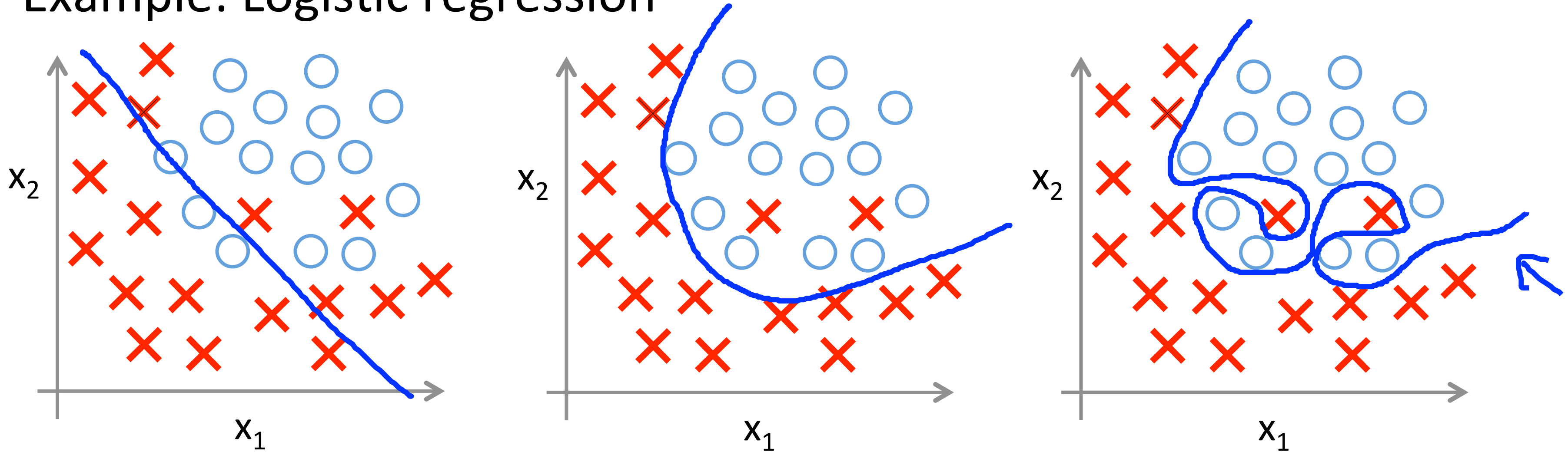
$\theta_0 + \theta_1 x$ "Underfit" "High bias"

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

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



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

(g = sigmoid function)

"Underfit"

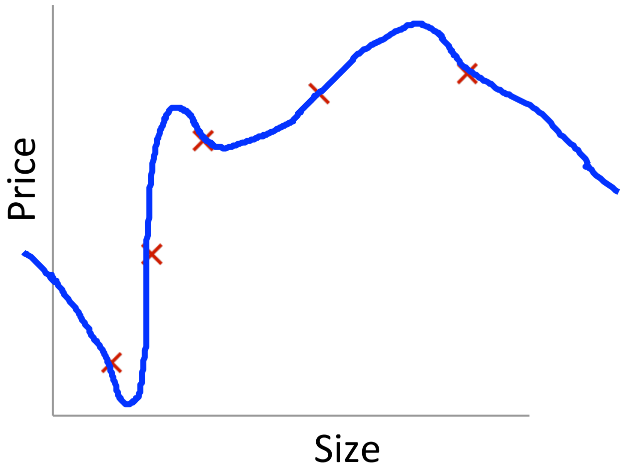
$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$

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

"Overfit"

Addressing overfitting:

$$\begin{bmatrix} x_1 = \text{size of house} \\ x_2 = \text{no. of bedrooms} \\ x_3 = \text{no. of floors} \\ x_4 = \text{age of house} \\ x_5 = \text{average income in neighborhood} \\ x_6 = \text{kitchen size} \\ \vdots \\ x_{100} \end{bmatrix}$$

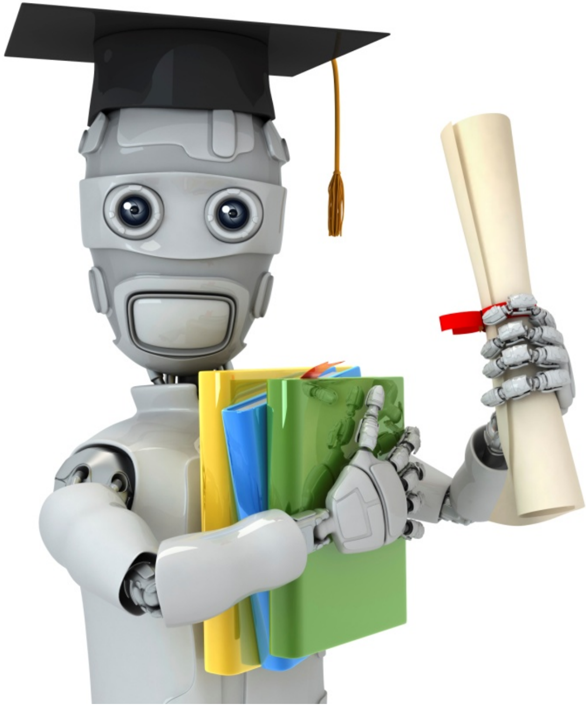


Andrew Ng

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.

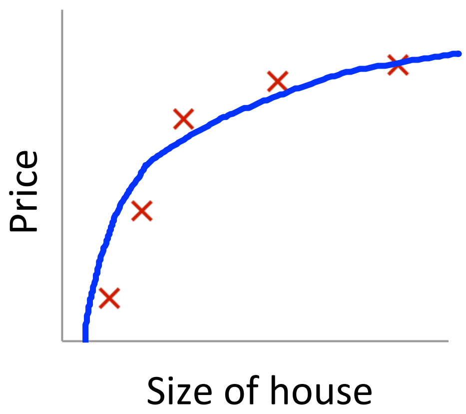


Regularization

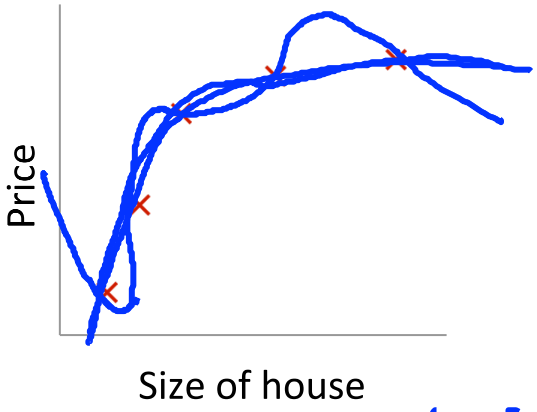
Cost function

Machine Learning

Intuition



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



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

Suppose we penalize and make θ_3, θ_4 really small.

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

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

Regularization.

Small values for parameters $\boxed{\theta_0, \theta_1, \ldots, \theta_n}$

- "Simpler" hypothesis
- Less prone to overfitting

$\boxed{\theta_3, \theta_4}$ ≈ 0

Housing:

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

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

$\cancel{\theta_0}$ $\theta_1, \theta_2, \theta_3, \ldots, \theta_{100}$ $\cancel{\theta_0}$

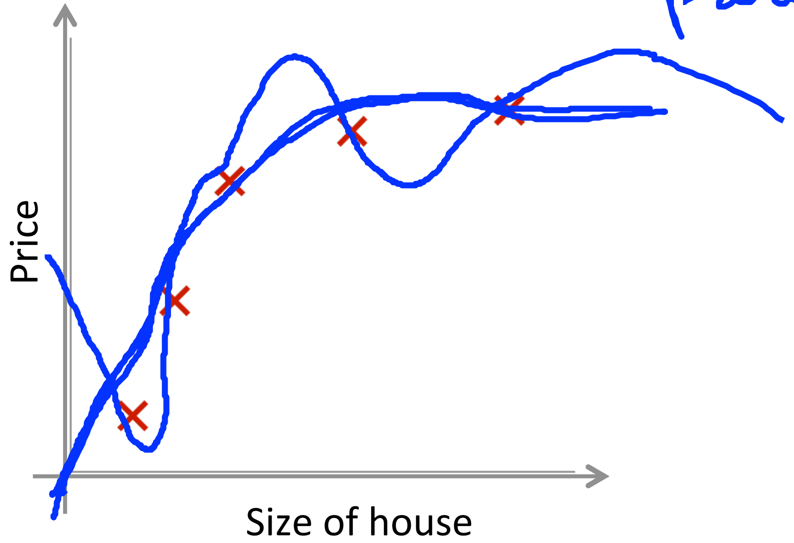
Andrew Ng

Regularization.

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

regularization parameter

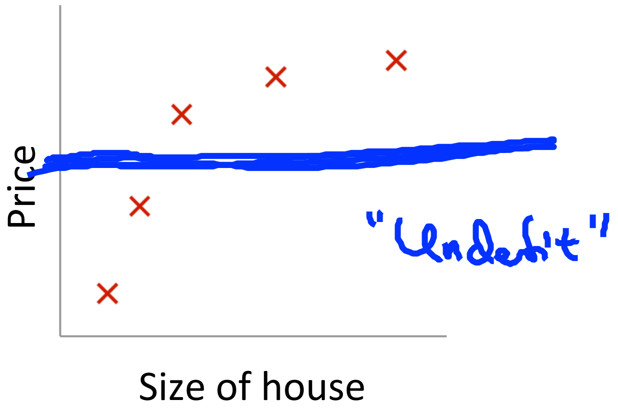
$$\min_\theta J(\theta)$$



In regularized linear regression, we choose θ to minimize

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

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



$h_\theta(x)$

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

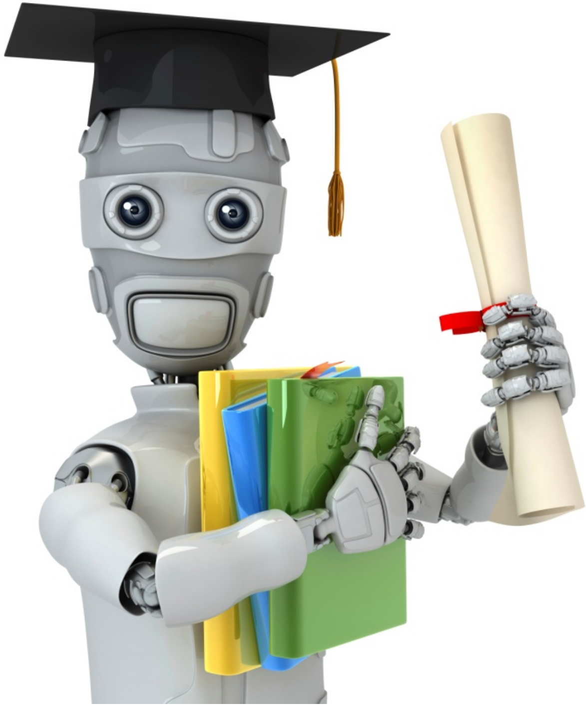
$\theta_1, \theta_2, \theta_3, \theta_4$

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

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

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

Andrew Ng



Machine Learning

Regularization

Regularized linear regression

Regularized linear regression

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

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

Gradient descent

$\theta_0 \quad \theta_1, \theta_2, \ldots, \theta_n$

Repeat {

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

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

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

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

}

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

$\rightarrow J(\theta)$

$1 - \alpha \frac{\lambda}{m} < 1$ $\quad 0.99 \quad \theta_j \times 0.99 \quad \boxed{\theta_j}$

Andrew Ng

Normal equation

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

$m \times (n+1)$

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

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

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

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

Andrew Ng

Non-invertibility (optional/advanced).

Suppose $m \leq n$,

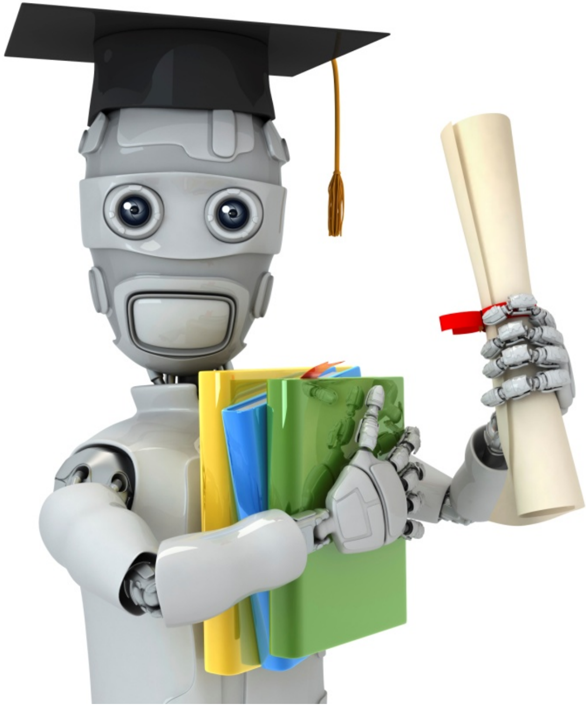
(#examples) (#features)

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

pinv inv

If $\lambda > 0$,

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

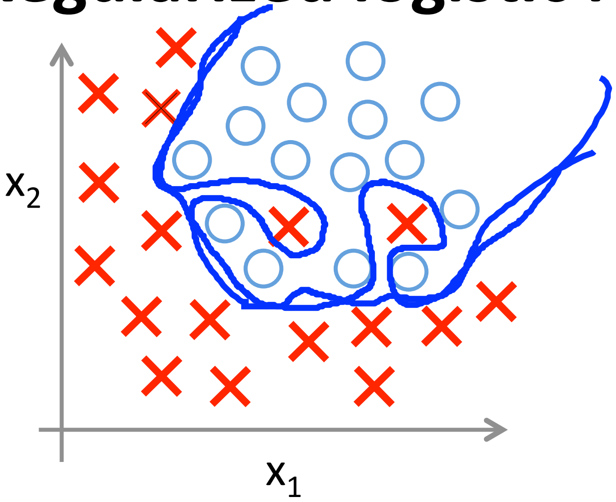


Machine Learning

Regularization

Regularized logistic regression

Regularized logistic regression.



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

Cost function:

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

$$+ \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

$$\boxed{\theta_1, \theta_2, \dots, \theta_n}$$

Andrew Ng

Gradient descent

Repeat $\{$

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

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

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

$\theta_1, \ldots, \theta_n$

$\}$

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

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

Advanced optimization

fminunc(@costFunction, ...) $\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ theta(1) theta(2) theta(n+1)

```
function [jVal, gradient] = costFunction(theta)
    jVal = [code to compute J(θ)];
```

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

```
    gradient(1) = [code to compute ∂/∂θ0 J(θ)];
```

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

```
    gradient(2) = [code to compute ∂/∂θ1 J(θ)];
```

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

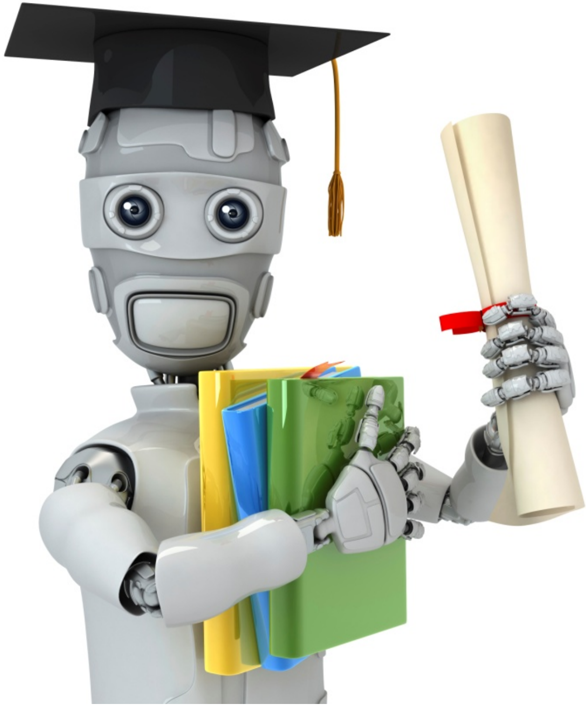
```
    gradient(3) = [code to compute ∂/∂θ2 J(θ)];
```

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

⋮

```
    gradient(n+1) = [code to compute ∂/∂θn J(θ)];
```

$J(\theta)$

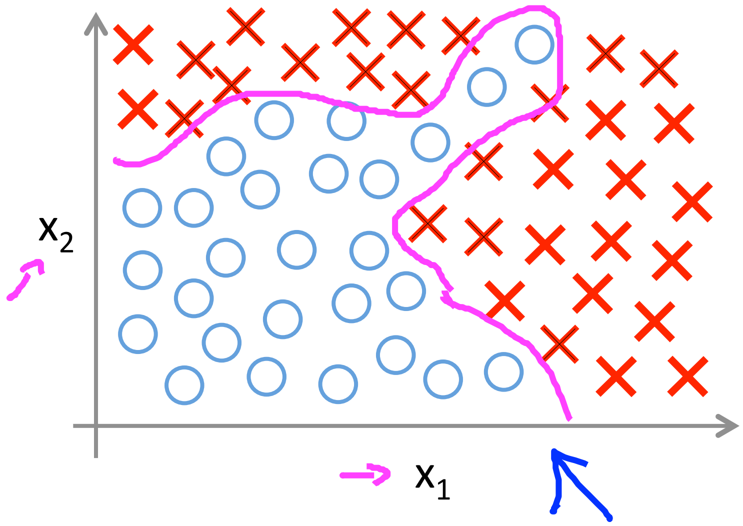


Machine Learning

Neural Networks: Representation

Non-linear hypotheses

Non-linear Classification



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

$x_1^2,\ x_1x_2,\ x_1x_3,\ x_1x_4 \dots x_1x_{100}$

$x_2^2,\ x_2x_3 \dots$

≈ 5000 features $\quad O(n^2)$ $\quad \approx \frac{n^2}{2}$

$x_1^2,\ x_2^2,\ x_3^2,\ \dots,\ x_{100}^2$

$x_1x_2x_3,\ x_1^2x_2,\ x_{10}x_{11}x_{17},\ \dots$

$O(n^3)$

170,000

x_1 = size

x_2 = # bedrooms

x_3 = # floors

x_4 = age

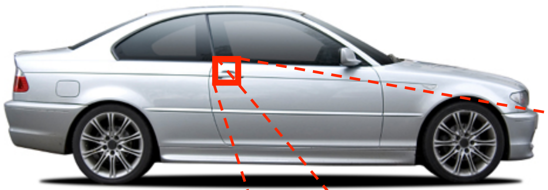
...

x_{100}

$n = 100$

What is this?

You see this:

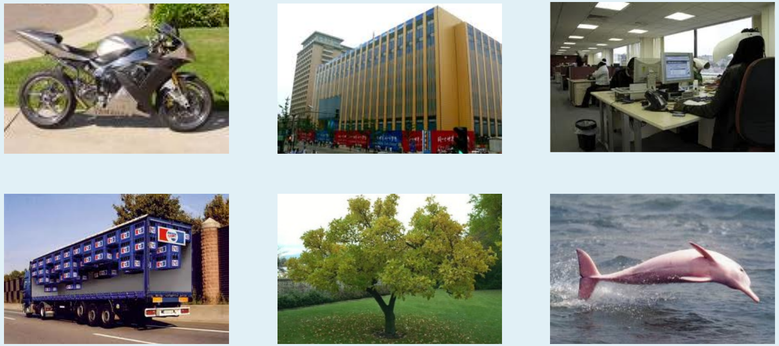


But the camera sees this:

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

Computer Vision: Car detection

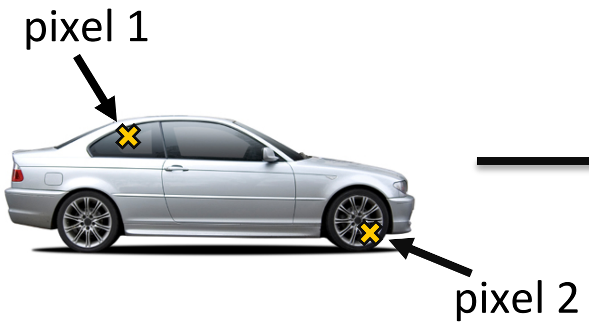
Cars



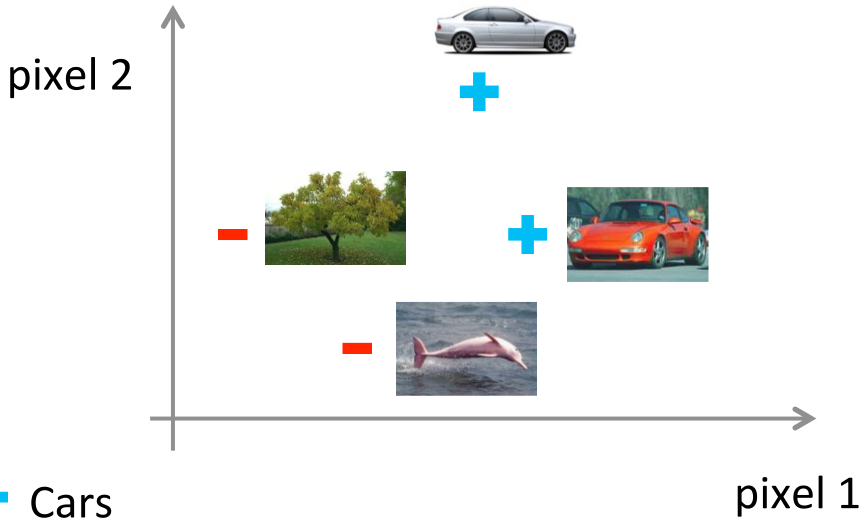
Not a car

Testing:

What is this?

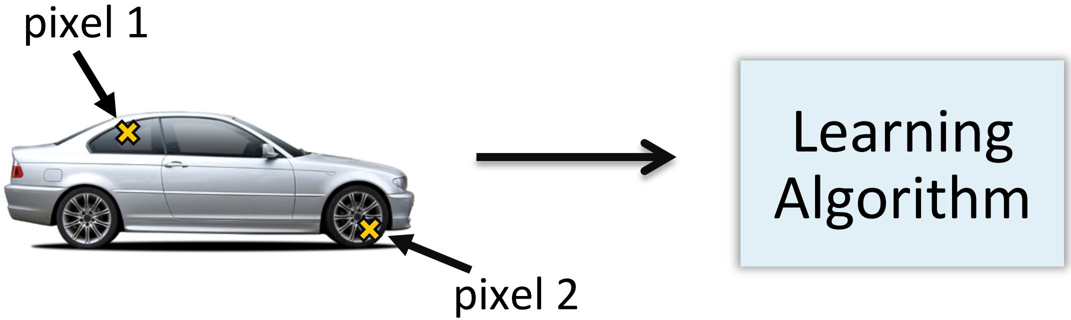


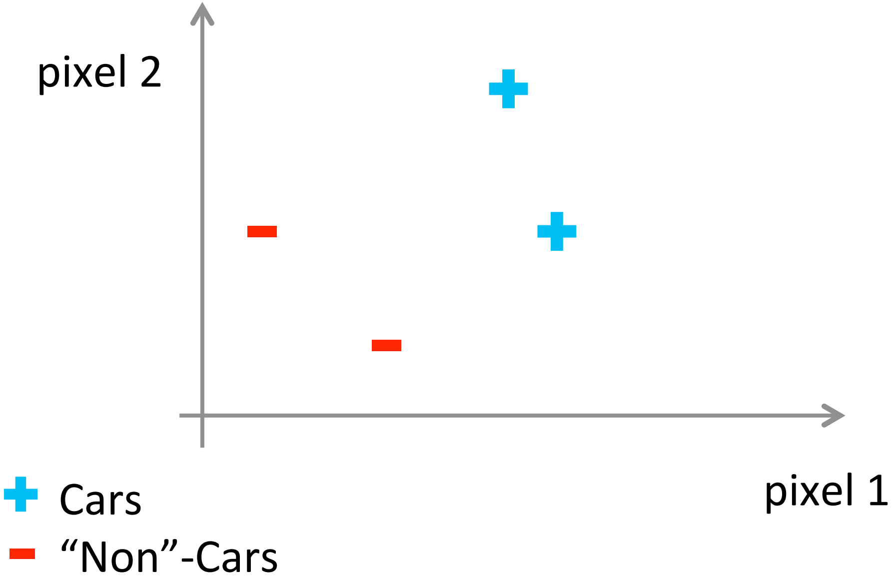
Learning Algorithm

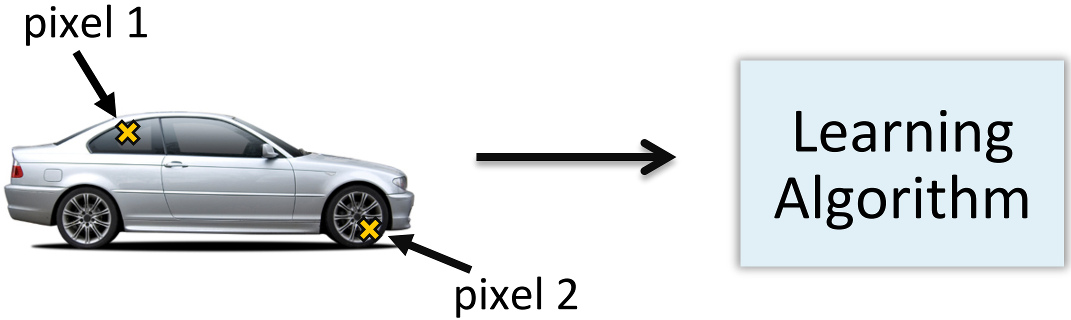


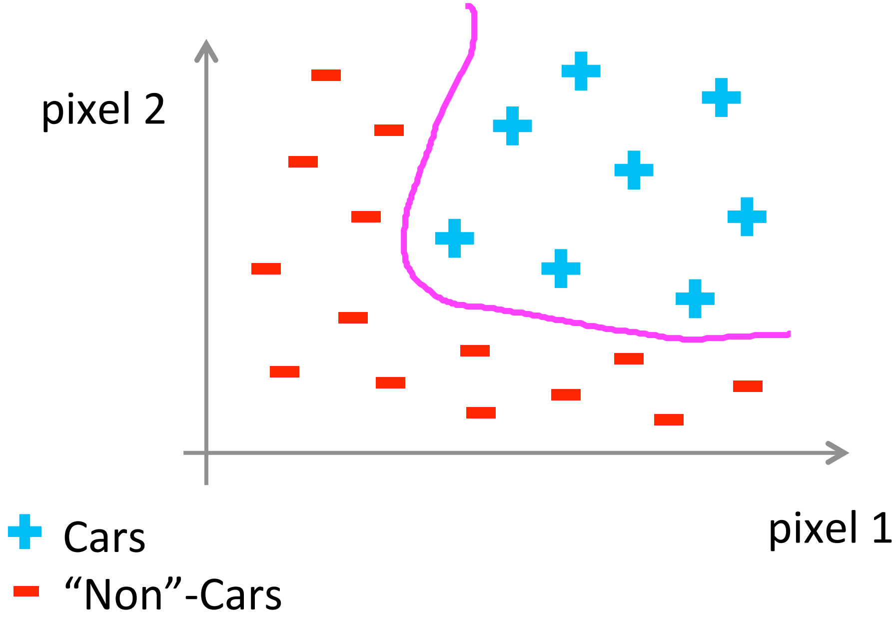
+ Cars

– “Non”-Cars


pixel 1
pixel 2
Learning Algorithm


pixel 2
pixel 1
Cars
“Non”-Cars





50 x 50 pixel images → 2500 pixels

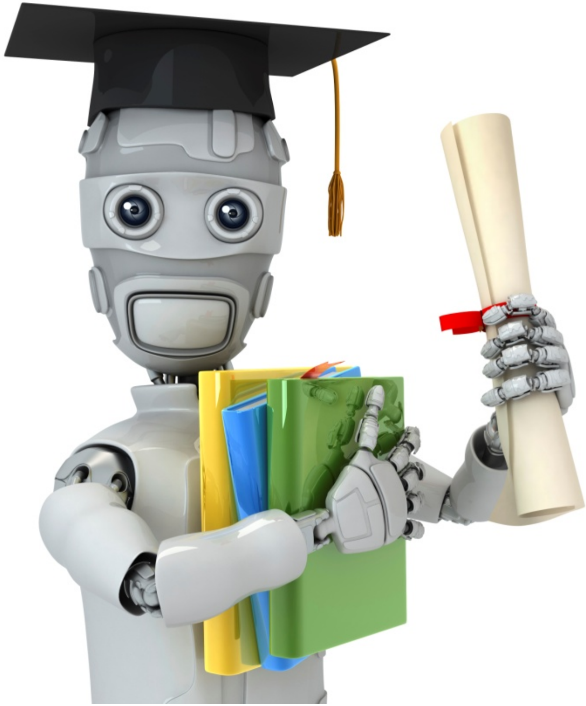
$\boxed{n = 2500}$ (7500 if RGB)

0-255

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

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

Andrew Ng



Machine Learning

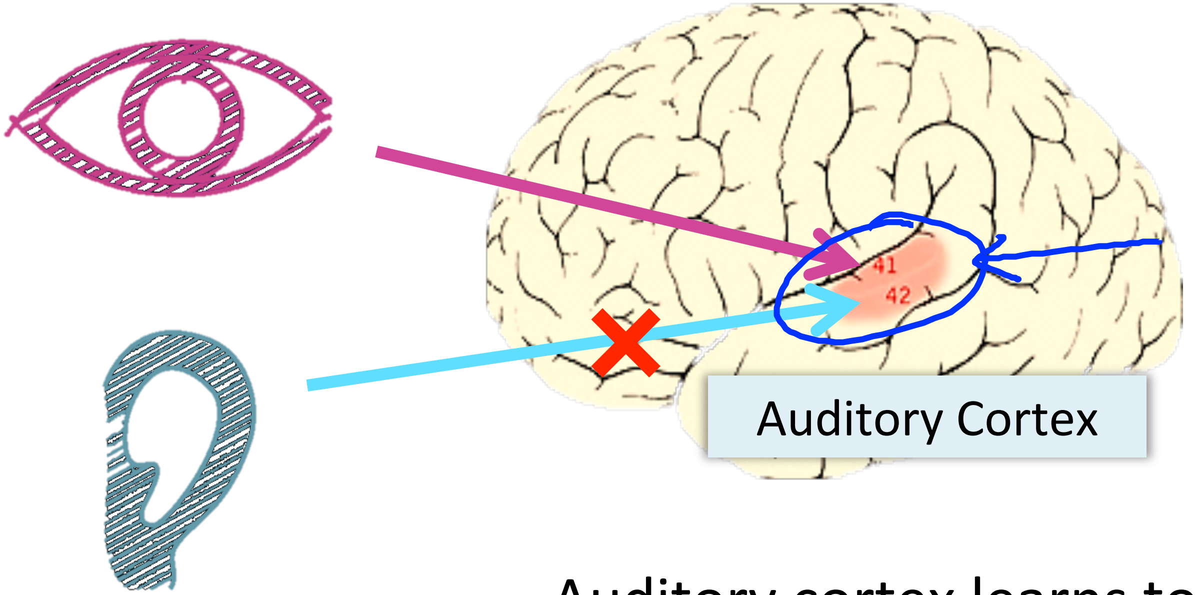
Neural Networks: Representation

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

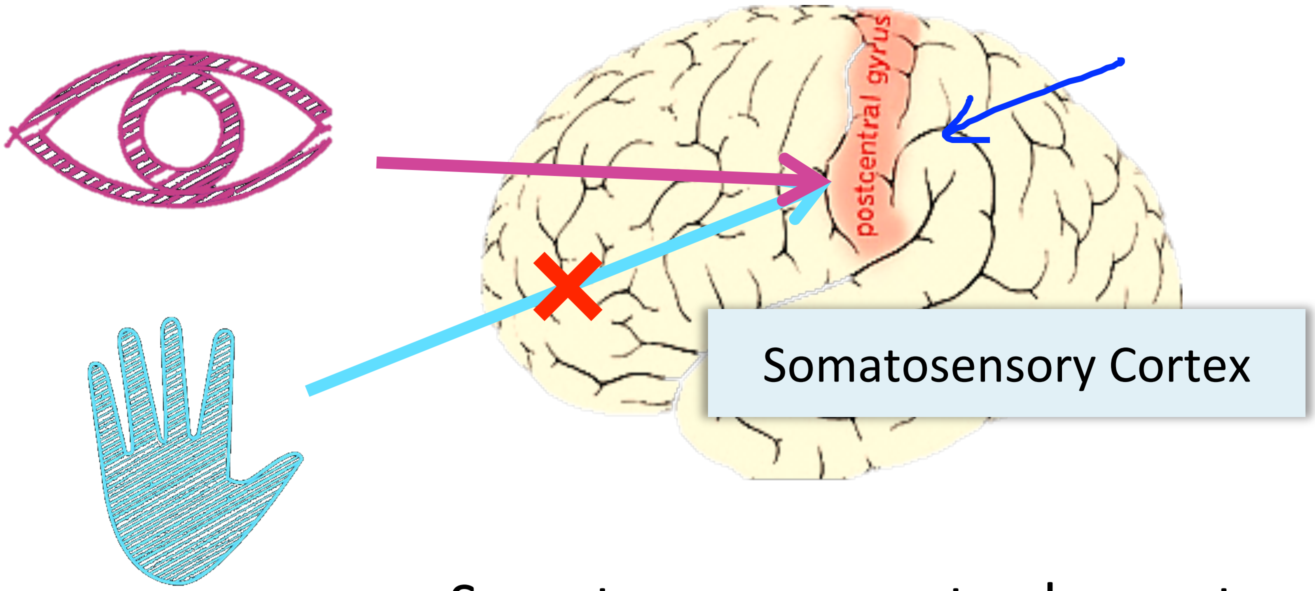
The "one learning algorithm" hypothesis



Auditory cortex learns to see

[Roe et al., 1992]

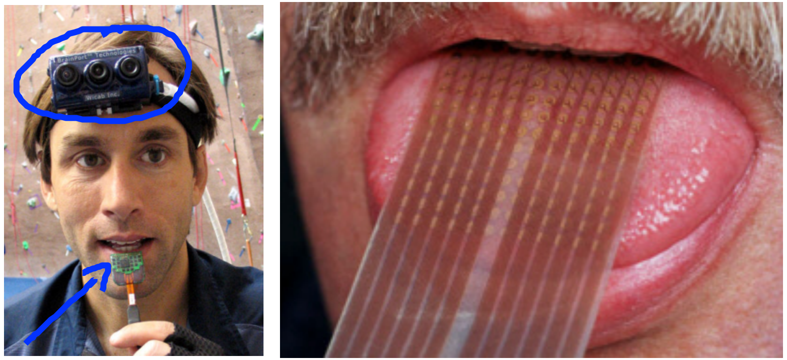
The "one learning algorithm" hypothesis



Somatosensory cortex learns to see

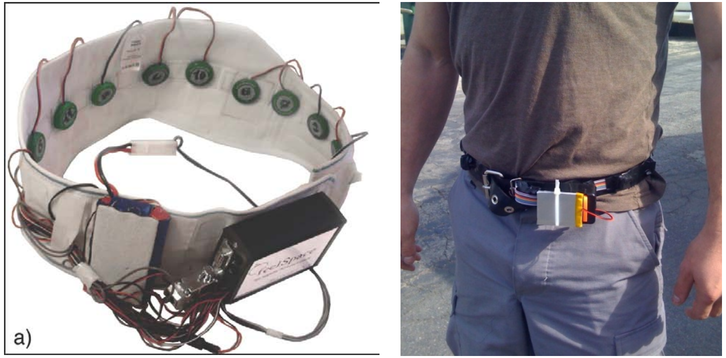
[Metin & Frost, 1989]

Sensor representations in the brain

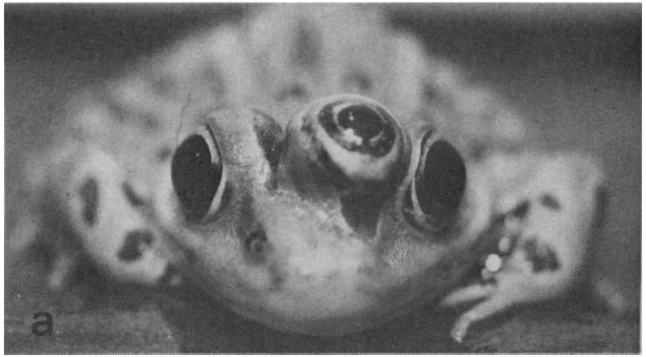


Seeing with your tongue

Human echolocation (sonar)

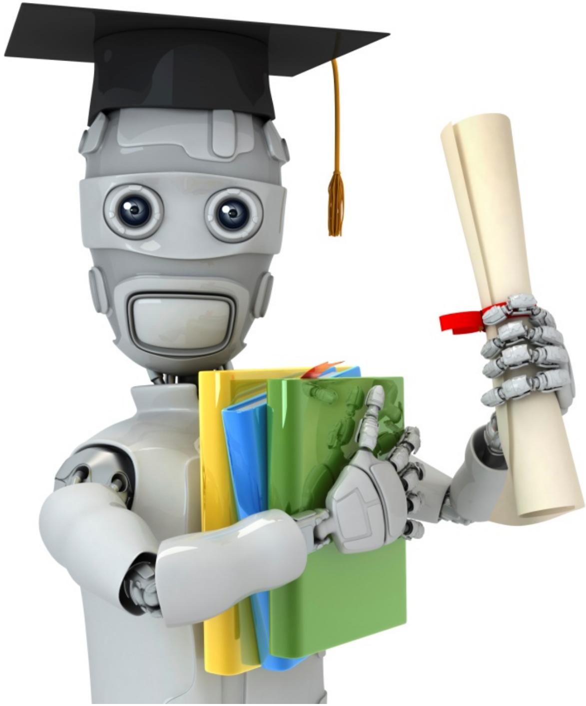


Haptic belt: Direction sense



Implanting a 3rd eye

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

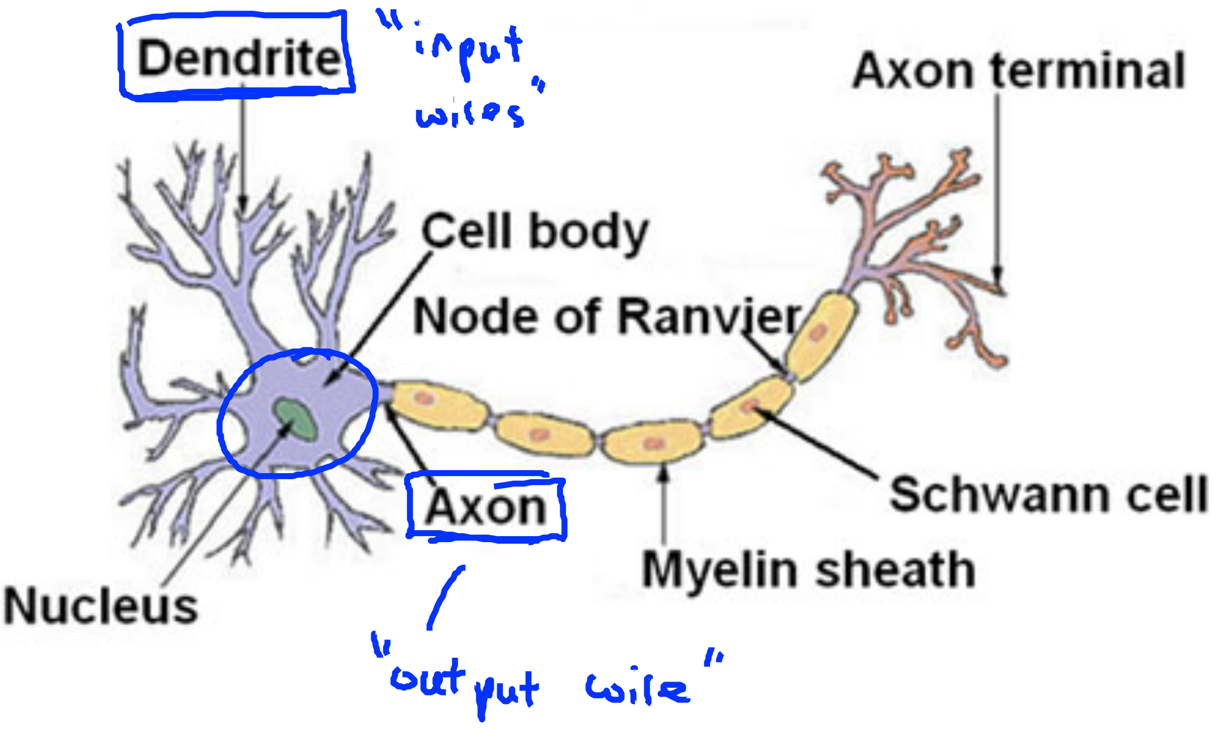


Machine Learning

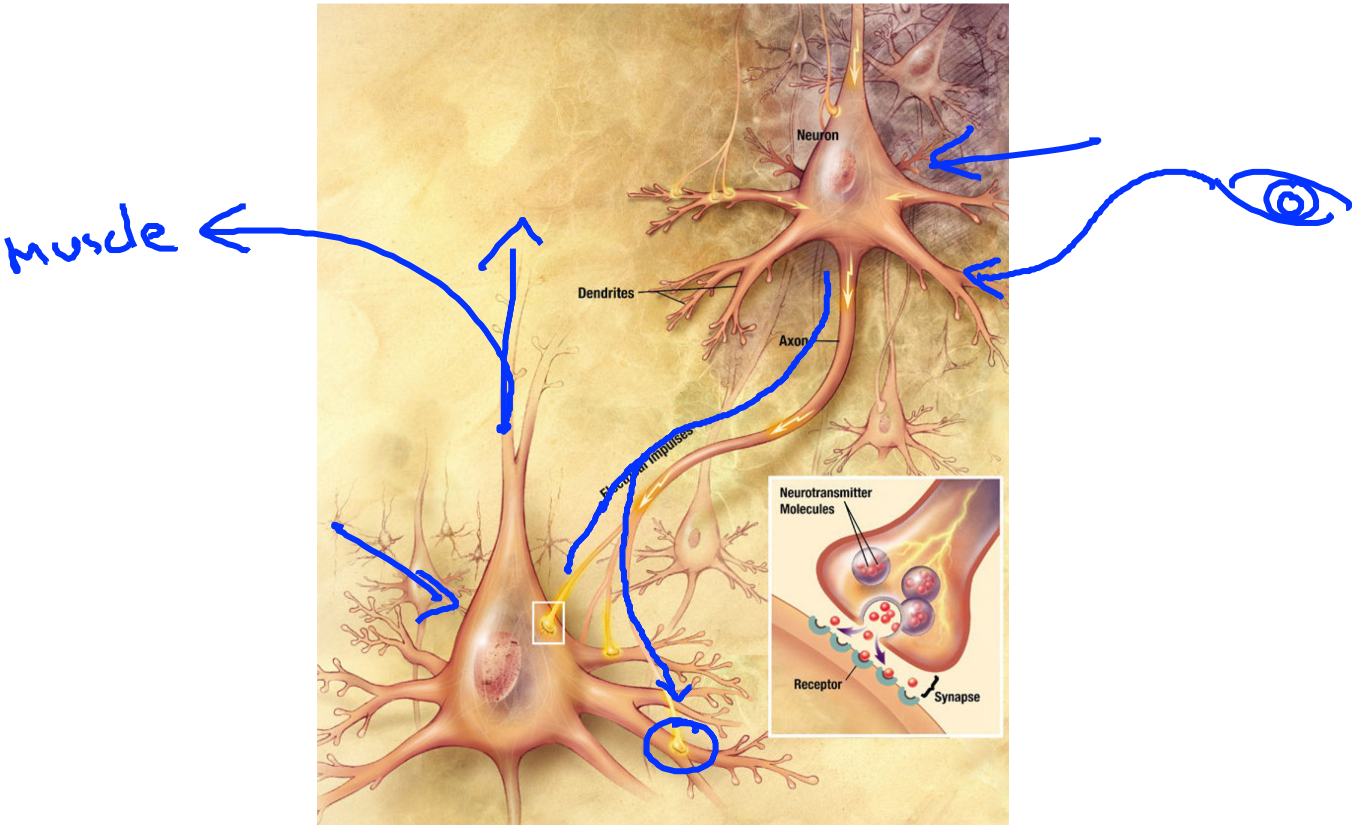
Neural Networks: Representation

Model representation I

Neuron in the brain

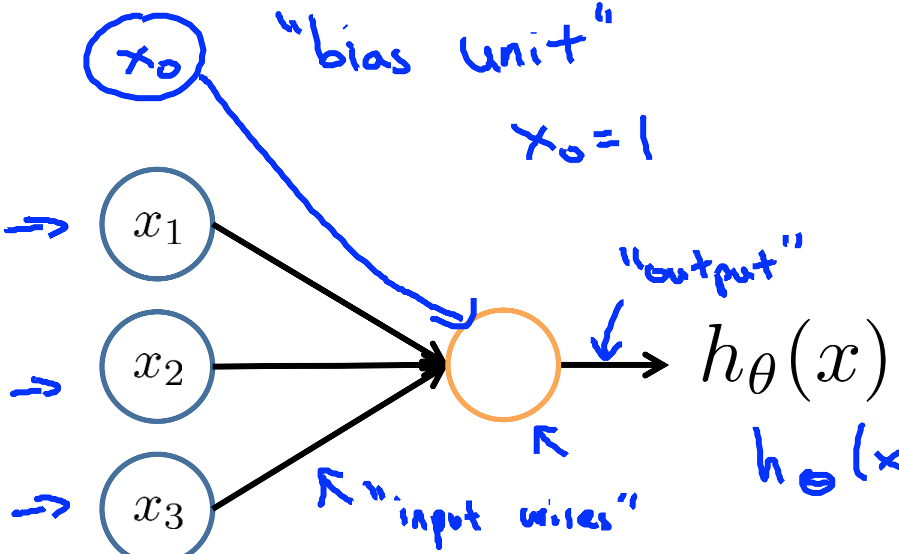


Neurons in the brain



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

Neuron model: Logistic unit



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

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

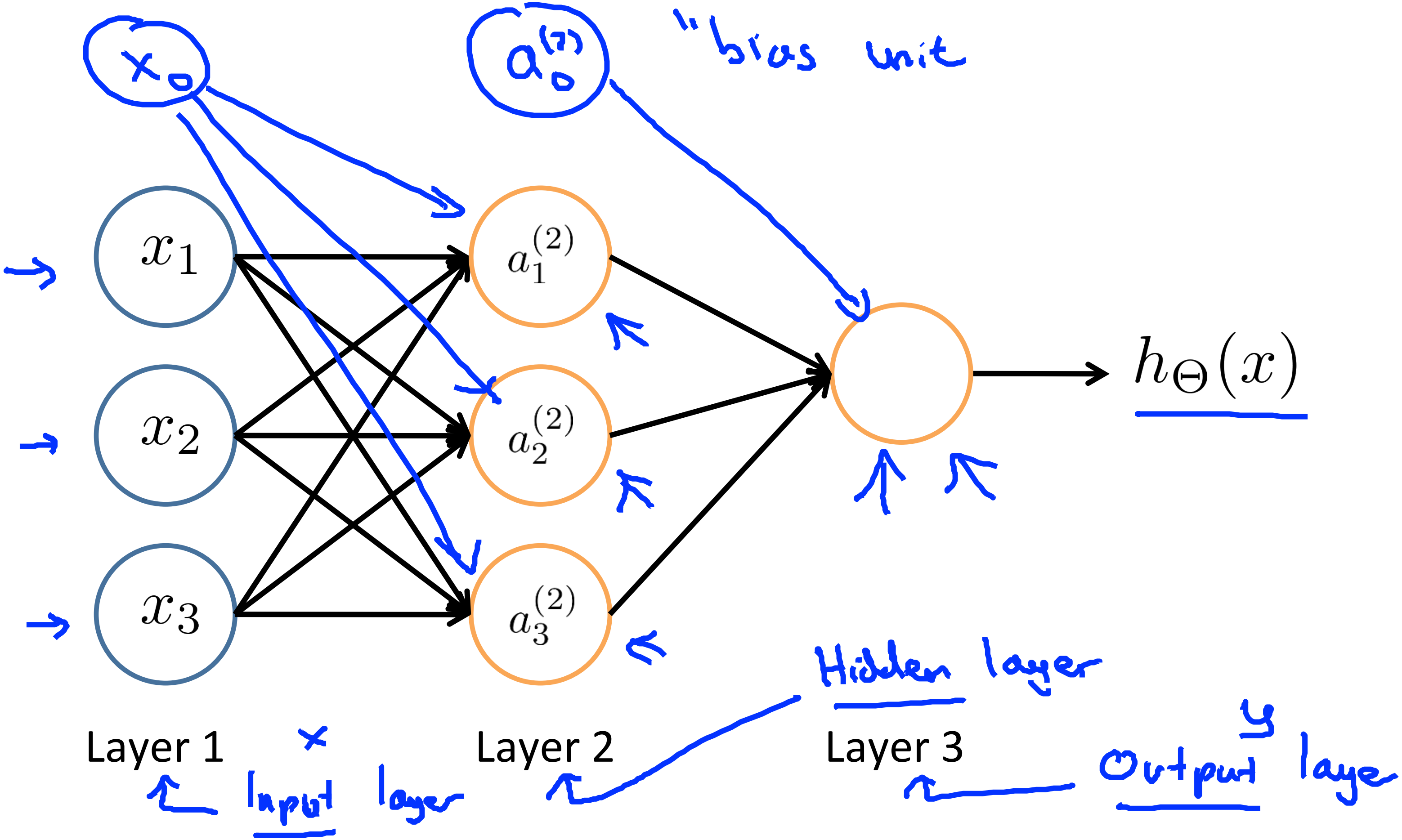
"weights"

(parameters

Sigmoid (logistic) activation function.

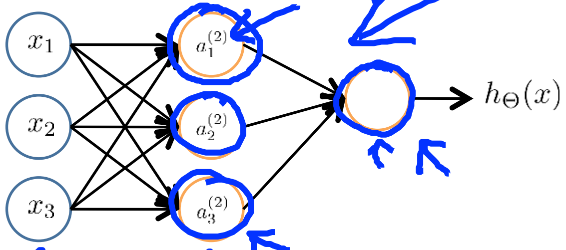
$$g(z) = \frac{1}{1+e^{-z}}$$

Neural Network



Andrew Ng

Neural Network



3 units 3 hidden units

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

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

$\Theta^{(1)} \in \mathbb{R}^{3\times 4}$ $h_\Theta(x)$

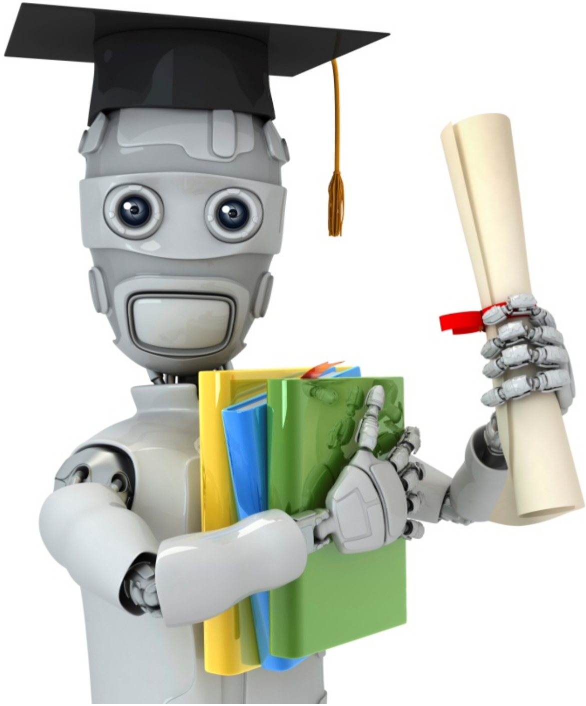
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$\Theta^{(2)}$

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

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

$S_{j+1} \times (S_j + 1)$

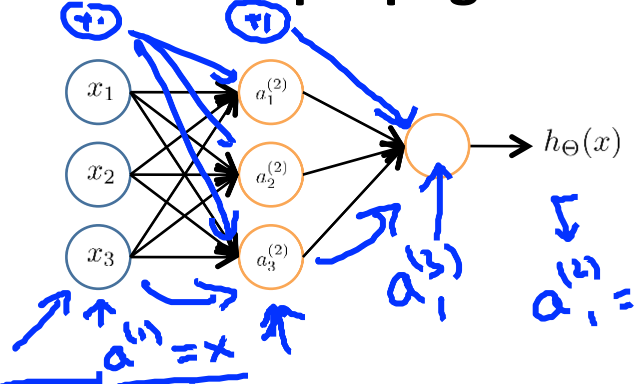


Machine Learning

Neural Networks: Representation

Model representation II

Forward propagation: Vectorized implementation



$a_1^{(2)} = g(z_1^{(2)})$

$z_1^{(2)}$

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

$z_2^{(2)}$

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

$z_3^{(2)}$

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

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

$\Theta^{(1)} x$

$a_3^{(2)} = g(z_3^{(2)})$ $\qquad z^{(3)}$

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

$a_1^{(2)}$, $a_2^{(2)}$, $a_3^{(2)}$

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

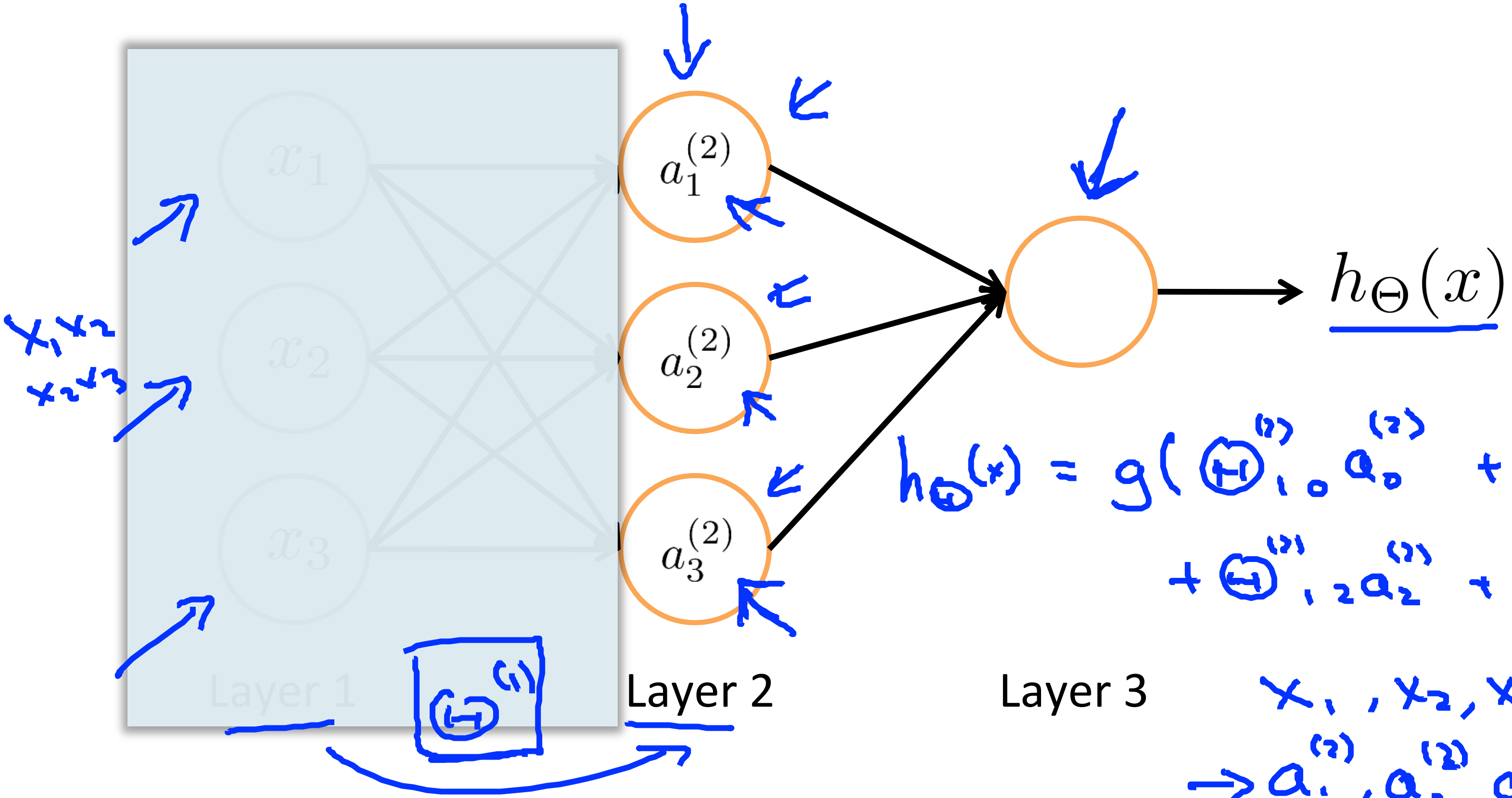
$$\underbrace{\boxed{a^{(2)}}}_{\mathbb{R}^3} = g(\underbrace{z^{(2)}}_{\mathbb{R}^3})$$

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

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

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

Neural Network learning its own features

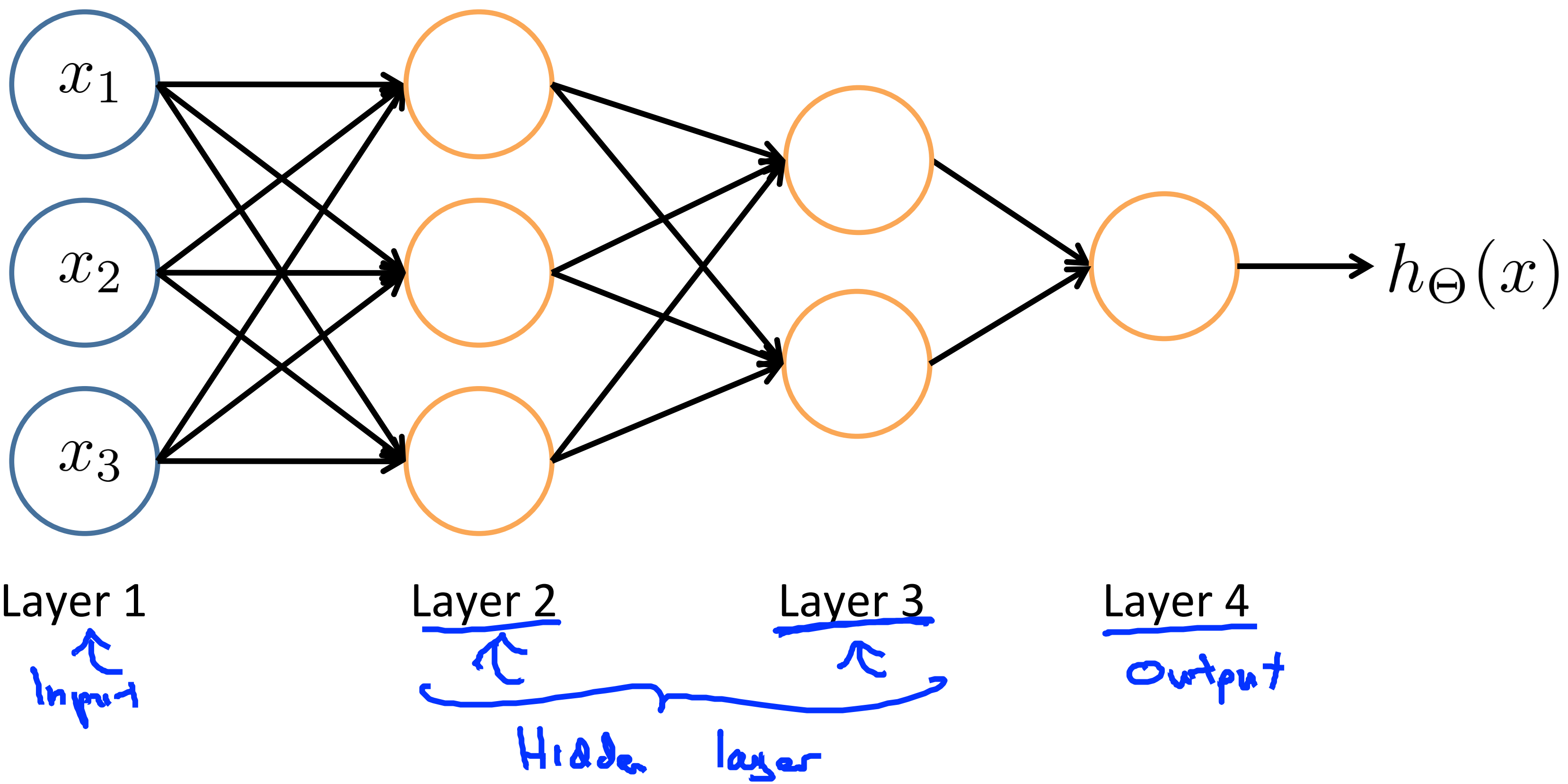


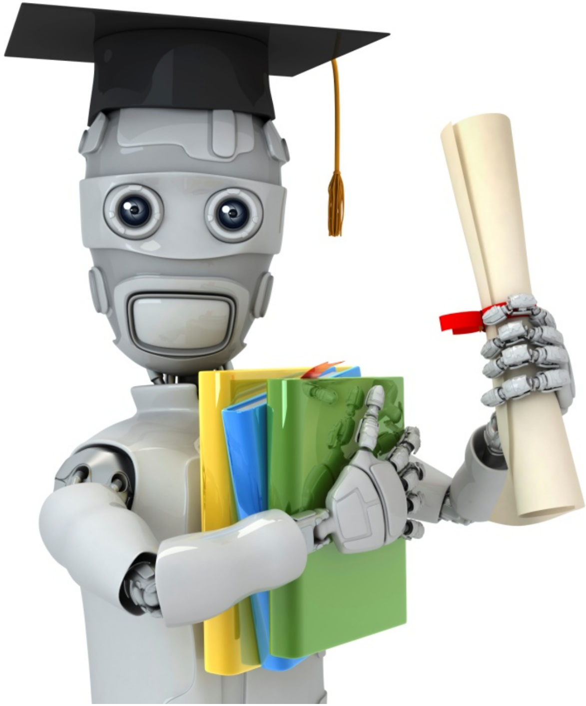
$$h_\Theta(x) = g(\Theta^{(2)}_{10} a^{(2)}_0 + \Theta^{(1)}_{11} a^{(2)}_1 + \Theta^{(2)}_{12} a^{(2)}_2 + \Theta^{(1)}_{13} a^{(2)}_3)$$

x_1, x_2, x_3

$\rightarrow a^{(2)}_1, a^{(2)}_2, a^{(2)}_3$

Other network architectures





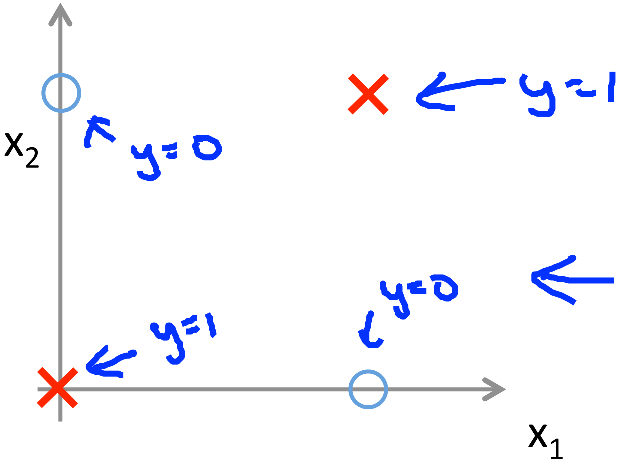
Machine Learning

Neural Networks: Representation

Examples and intuitions I

Non-linear classification example: XOR/XNOR

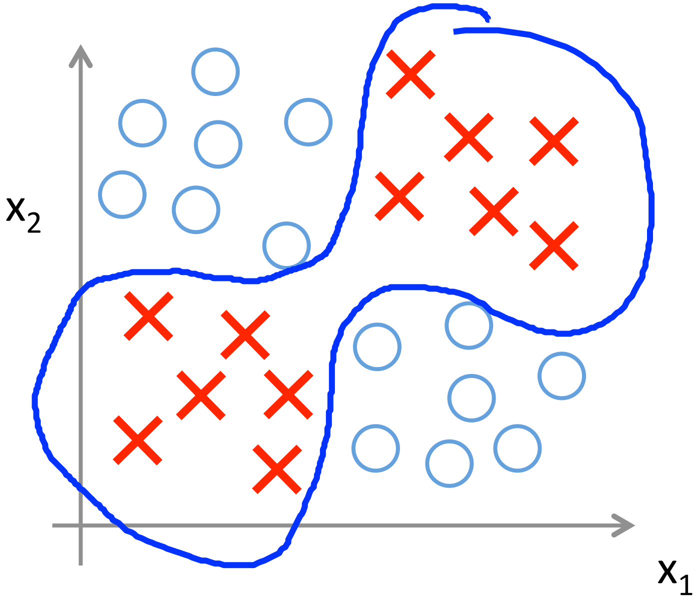
→ x_1, x_2 are binary (0 or 1).



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

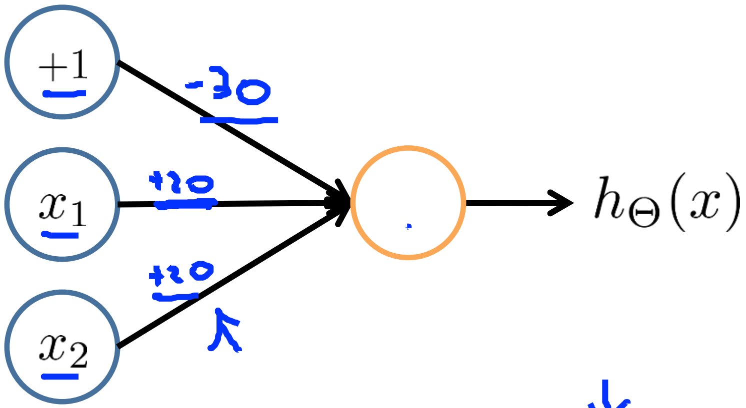
$$\underline{x_1 \text{ XNOR } x_2}$$

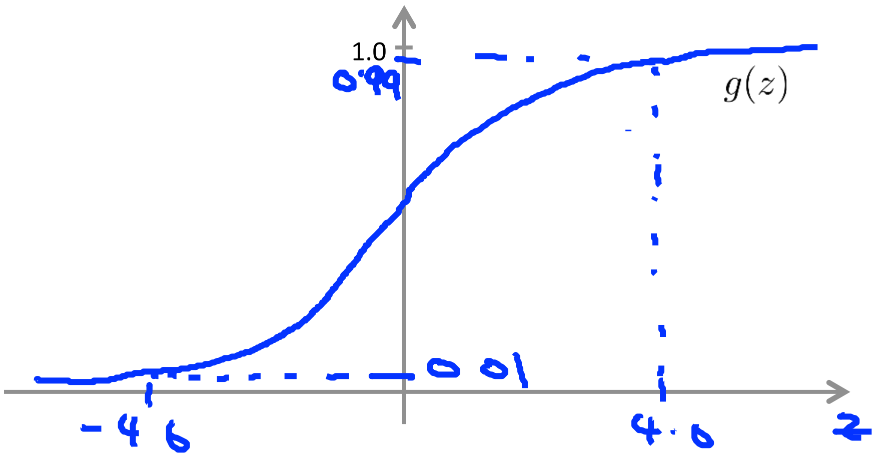
$$\underline{\text{NOT } (x_1 \text{ XOR } x_2)}$$



Simple example: AND

- $x_1, x_2 \in \{0, 1\}$
- $y = x_1 \text{ AND } x_2$



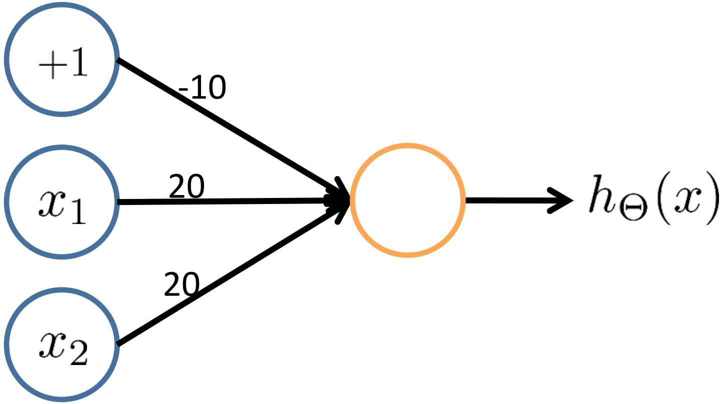


$$h_\Theta(x) = g(\underbrace{-30}_{\Theta^{(1)}_{10}} + \underbrace{20}_{\Theta^{(1)}_{11}} x_1 + \underbrace{20}_{\Theta^{(1)}_{12}} x_2)$$

x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

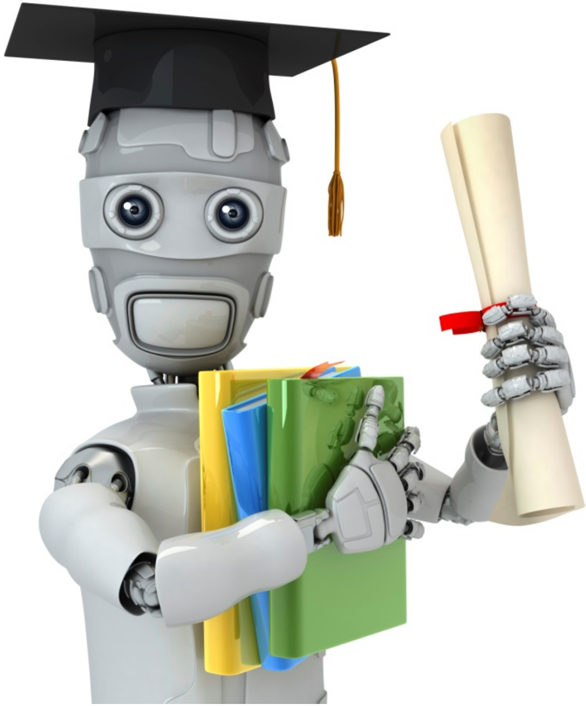
$h_\Theta(x) \approx x_1 \text{ AND } x_2$

Example: OR function



$g(-10 + 20x_1 + 20x_2)$

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



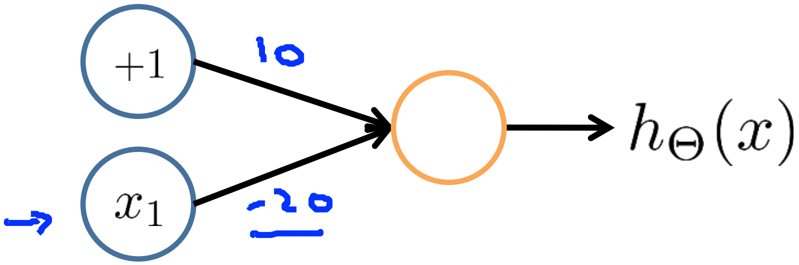
Machine Learning

Neural Networks: Representation

Examples and intuitions II

→ x_1 AND x_2 → x_1 OR x_2 $\{0,1\}$.

Negation: NOT x_1



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

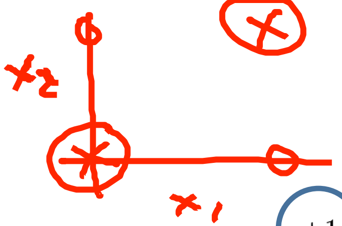
$$h_\Theta(x) = g(10 - 20x_1)$$

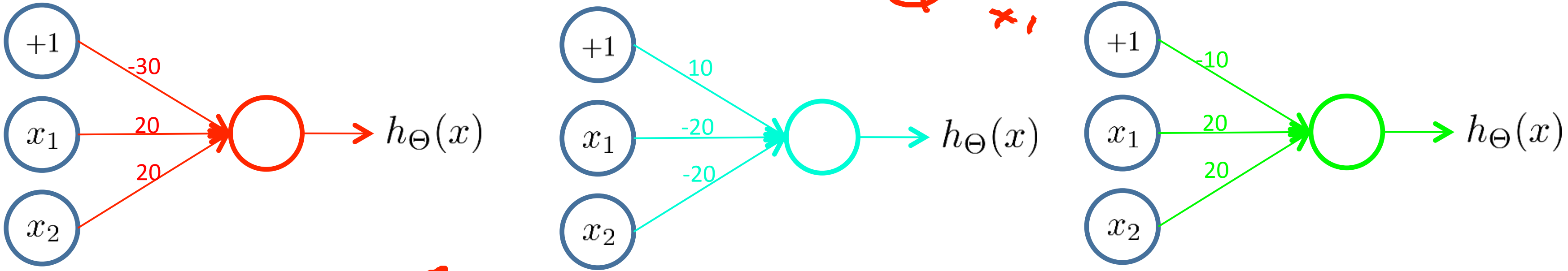
→ (NOT x_1) AND (NOT x_2)

($=1$ if and only if

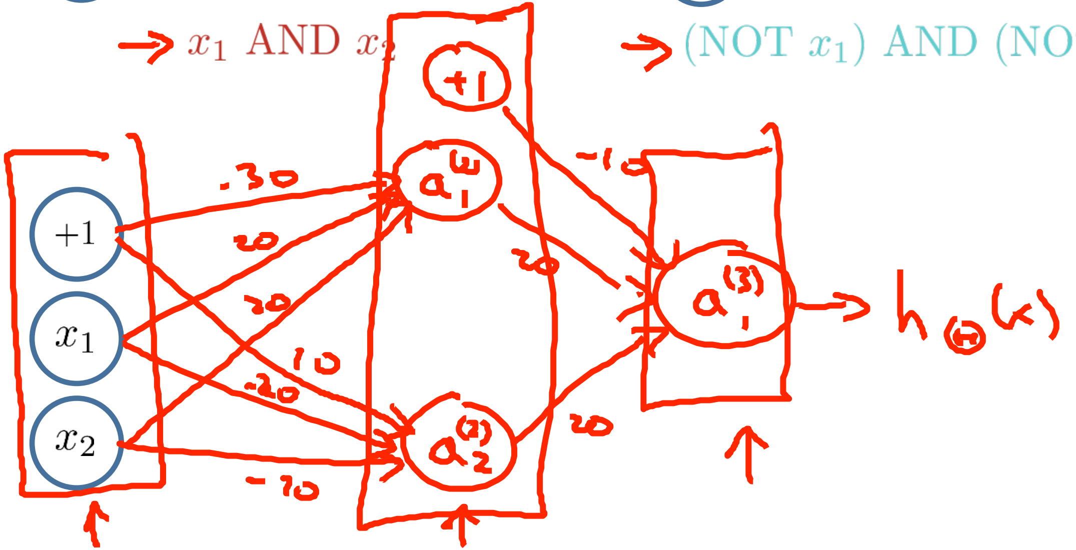
→ $x_1 = x_2 = 0$

Putting it together: x_1 XNOR x_2



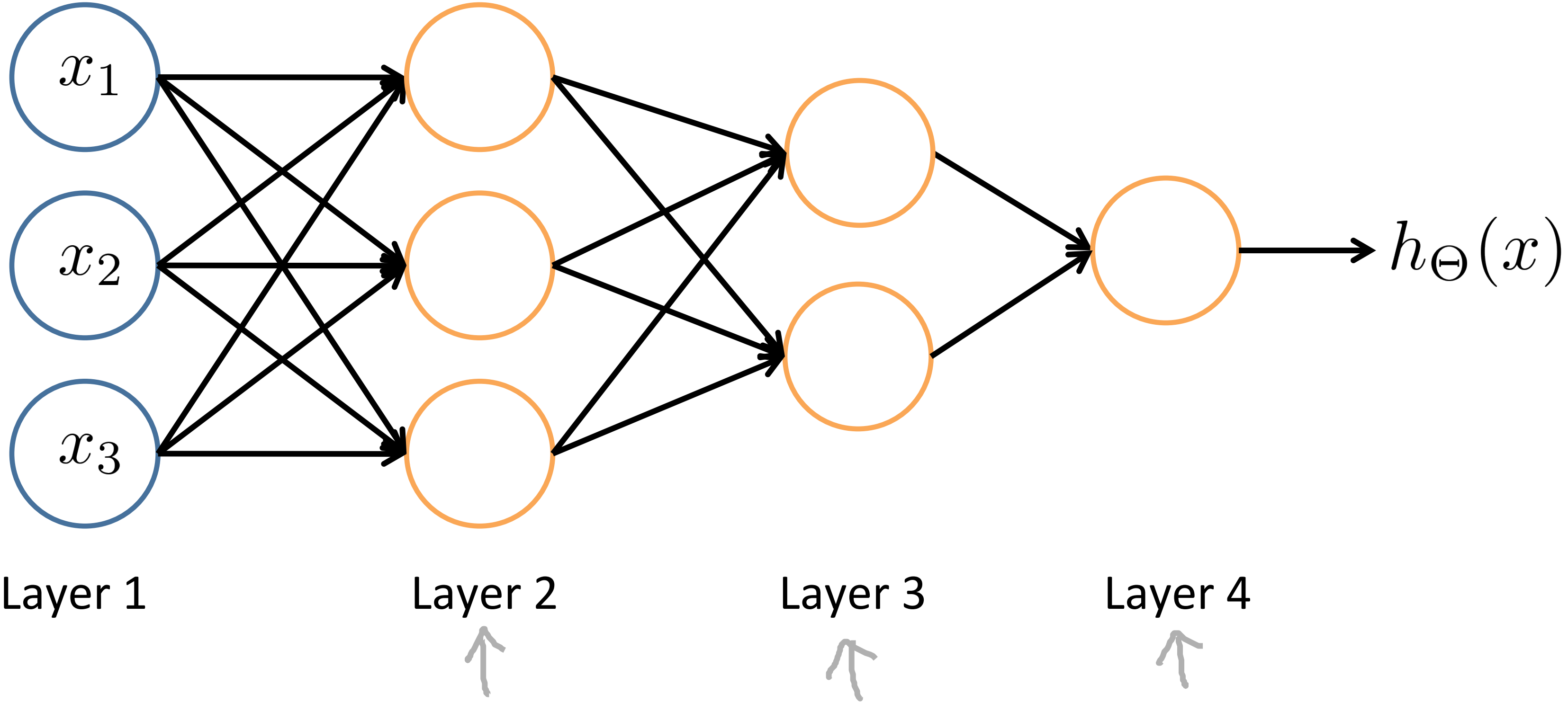


→ x_1 AND x_2 → (NOT x_1) AND (NOT x_2) → x_1 OR x_2

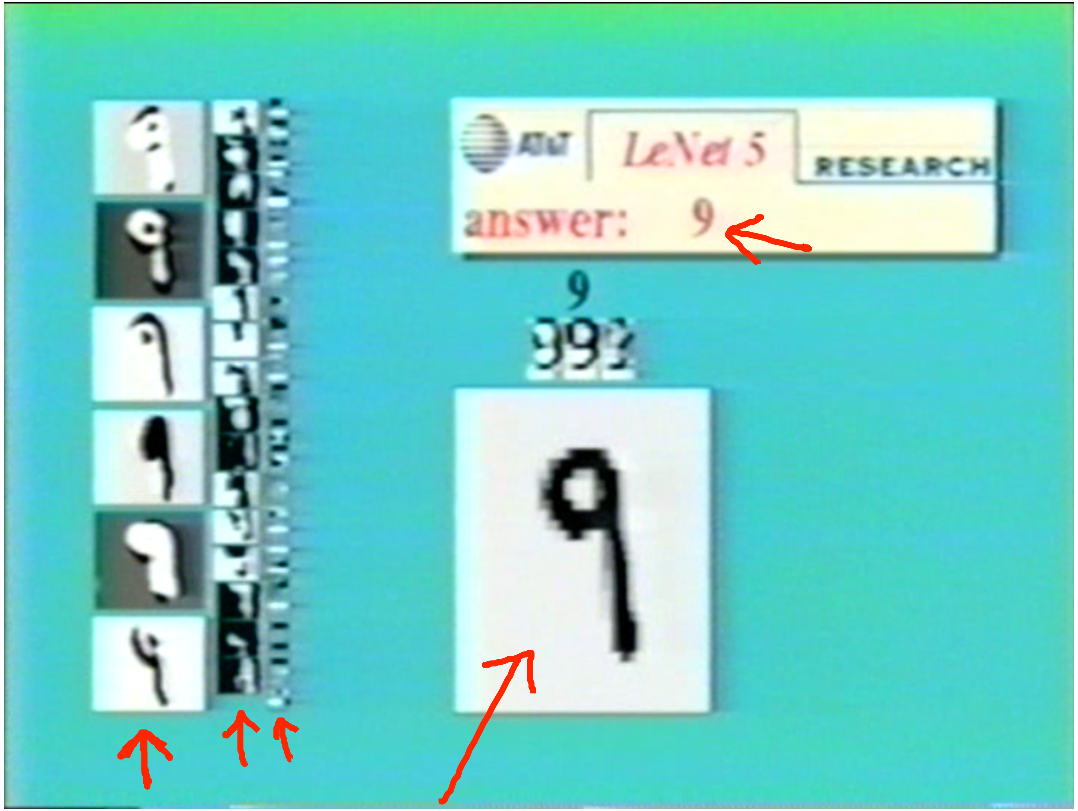


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Neural Network intuition

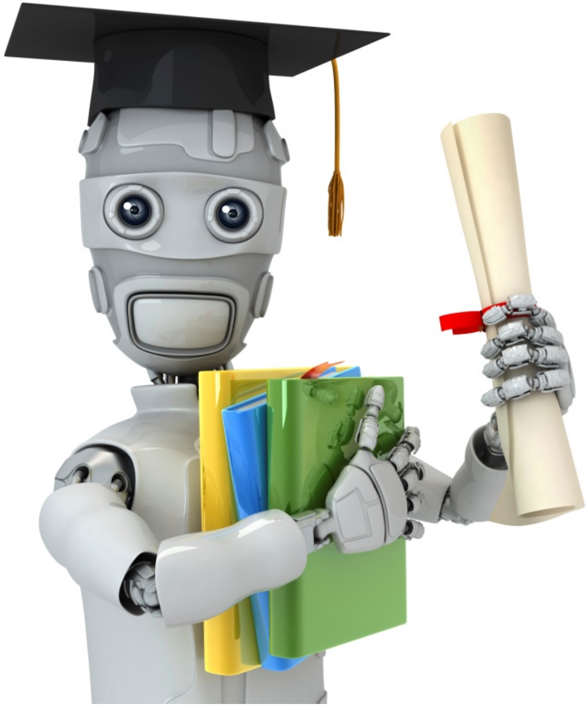


Handwritten digit classification



[Courtesy of Yann LeCun]

Andrew Ng



Machine Learning

Neural Networks: Representation

Multi-class classification

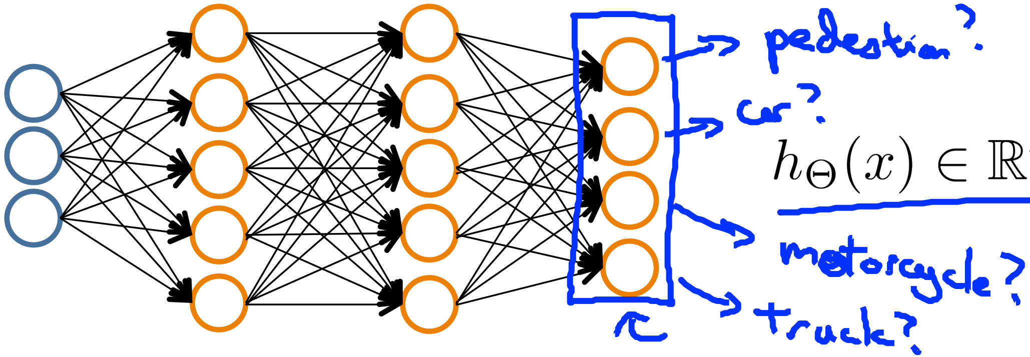
Multiple output units: One-vs-all.

Pedestrian

Car

Motorcycle

Truck

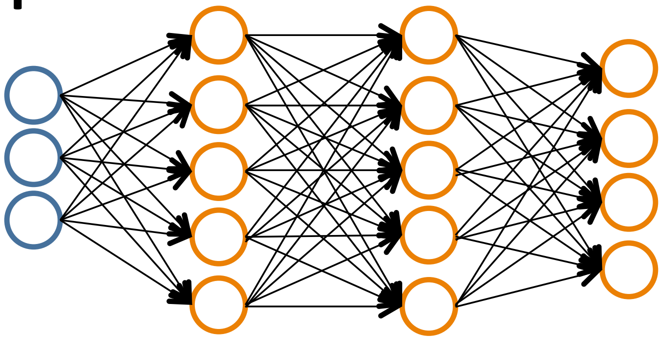


$$h_\Theta(x) \in \mathbb{R}^4$$

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

when pedestrian   when car   when motorcycle

Multiple output units: One-vs-all.



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

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

when pedestrian   when car   when motorcycle

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

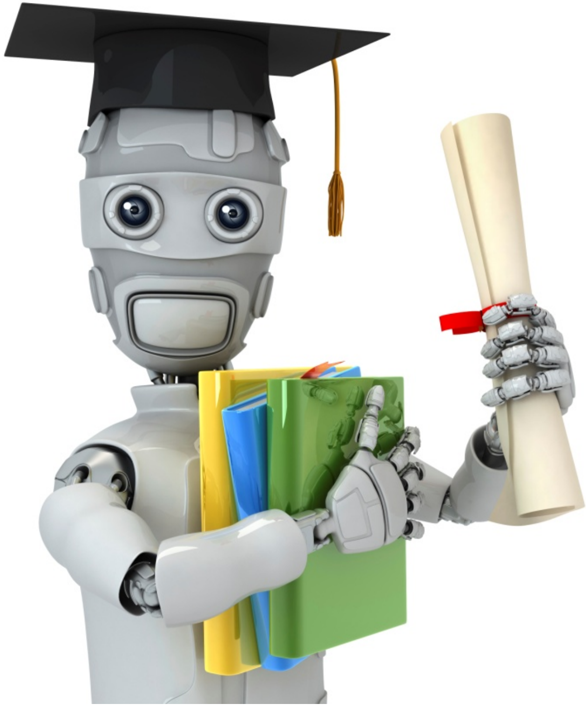
$y^{(i)}$ one of $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian   car   motorcycle   truck

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

~~Previously $y \in \{1, 2, 3, 4\}$~~

$\underbrace{h_\Theta(x^{(i)})}_{\mathbb{R}^4} \approx y^{(i)}$

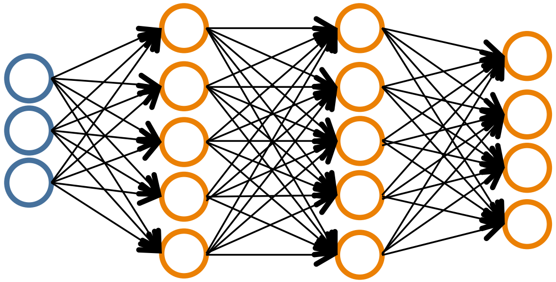


Machine Learning

Neural Networks: Learning

Cost function

Neural Network (Classification)



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

$L =$ total no. of layers in network

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

<u>Binary classification</u>

$y = 0 \text{ or } 1$

1 output unit

<u>Multi-class classification</u> (K classes)

$y \in \mathbb{R}^K$ 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:

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

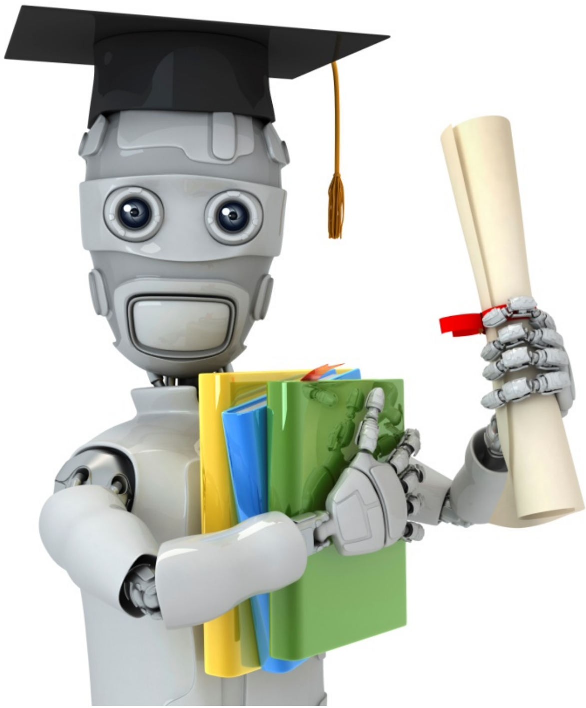
Neural network:

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

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log(1-(h_\Theta(x^{(i)}))_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

Andrew Ng



Machine Learning

Neural Networks: Learning

Backpropagation algorithm

Gradient computation

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

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

Need code to compute:

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

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

Gradient computation

Given one training example (x, y):

Forward propagation:

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

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

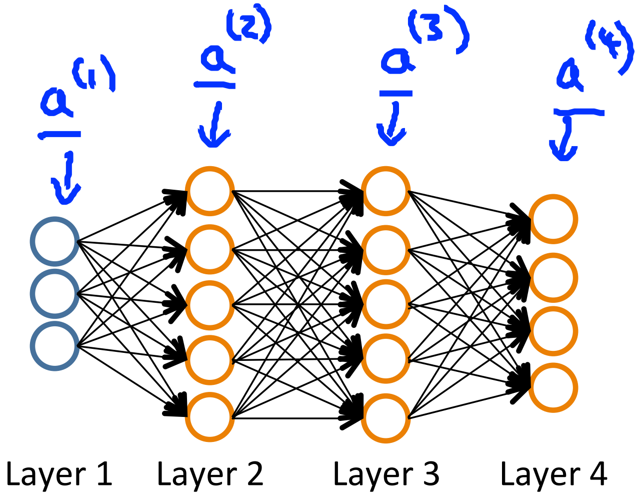
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$

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

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

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

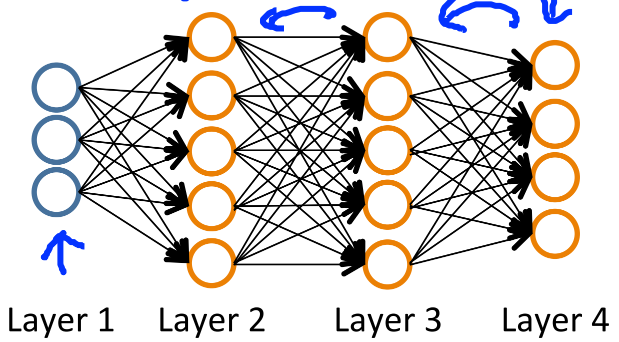
$$a^{(4)} = h_\Theta(x) = g(z^{(4)})$$



Gradient computation: Backpropagation algorithm

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

$\delta^{(2)}$ $a_j^{(l)}$ $\delta^{(3)}$ $\delta^{(4)}$



For each output unit (layer L = 4)

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

$(h_\Theta(x))_j$ $\delta^{(4)} = a^{(4)} - y$

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

$a^{(3)} .* (1 - a^{(3)})$

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

$a^{(2)} .* (1 - a^{(2)})$

(No $\delta^{(1)}$)

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$$

(ignoring λ; if $\lambda = 0$)

Backpropagation algorithm

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

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

(use to compute $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$)

For $i = 1$ to m

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

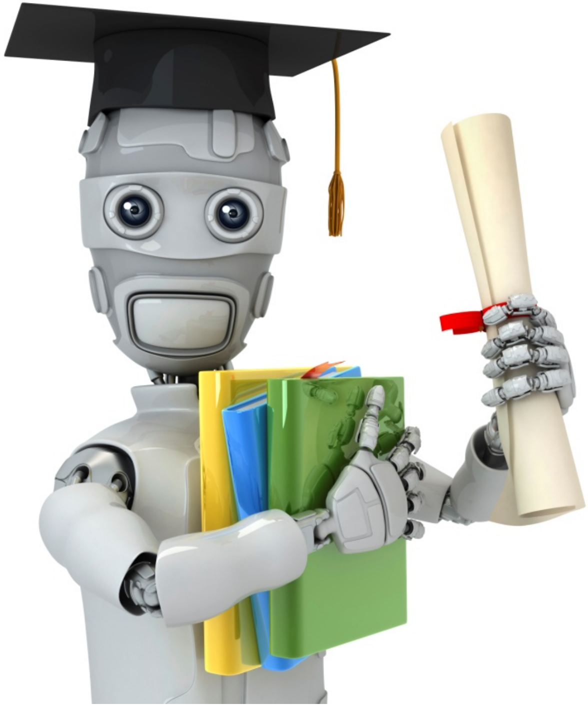
- Set $a^{(1)} = x^{(i)}$
- Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$
- Using $y^{(i)}$, compute $\delta^{(L)} = \boxed{a^{(L)}} - \boxed{y^{(i)}}$
- Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$ $\cancel{\delta^{(1)}}$
- $\boxed{\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}}$

$\triangle^{(l)} := \triangle^{(l)} + \delta^{(l+1)} (a^{(l)})^T$

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

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

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



Machine Learning

Neural Networks: Learning

Implementation note: Unrolling parameters

Advanced optimization

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

$\mathbb{R}^{n+1}$ $\mathbb{R}^{n+1}$ (vectors)

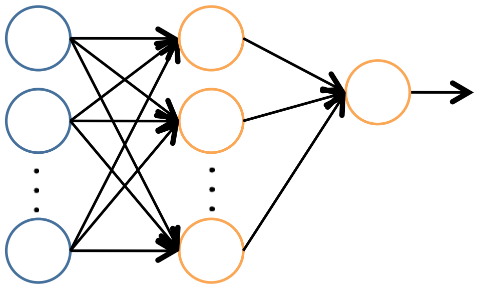
Neural Network (L=4):

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

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

"Unroll" into vectors

Example



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

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

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

$\Theta^{(1)}$ $\Theta^{(2)}$ $\Theta^{(3)}$

```
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);
```

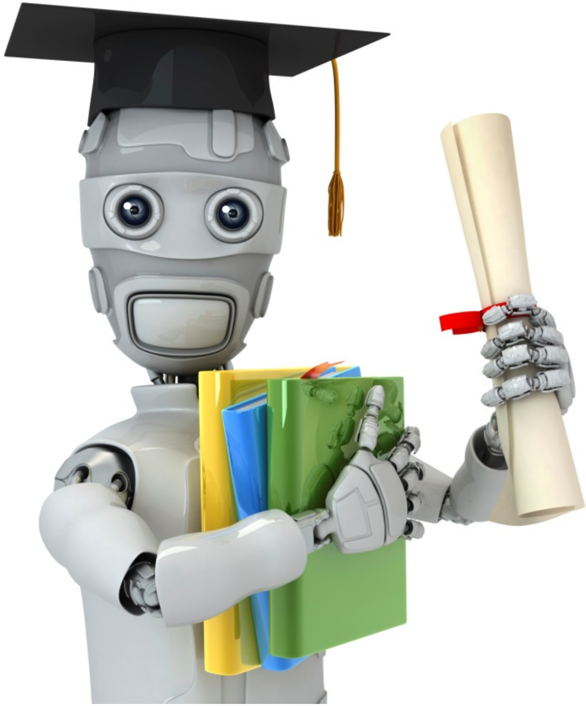
Learning Algorithm

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

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

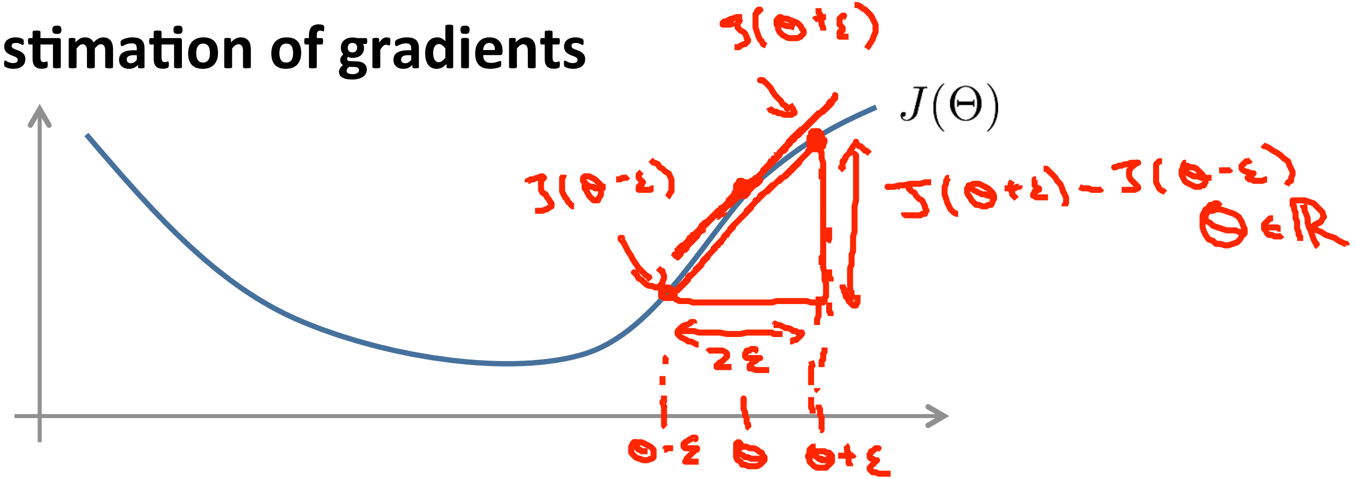


Machine Learning

Neural Networks: Learning

Gradient checking

Numerical estimation of gradients



$$\frac{d}{d\theta} J(\theta) \approx \underbrace{\frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon}}$$

$\varepsilon = 10^{-4}$

$$\cancel{\frac{J(\theta+\varepsilon) - J(\theta)}{\varepsilon}}$$

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

Andrew Ng

Parameter vector θ

- $\theta \in \mathbb{R}^n$ (E.g. θ is "unrolled" version of $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$)
- $\theta = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$
- $$\frac{\partial}{\partial \theta_1} J(\theta) \approx \frac{J(\boxed{\theta_1 + \epsilon}, \theta_2, \theta_3, \dots, \theta_n) - J(\boxed{\theta_1 - \epsilon}, \theta_2, \theta_3, \dots, \theta_n)}{2\epsilon}$$
- $$\frac{\partial}{\partial \theta_2} J(\theta) \approx \frac{J(\theta_1, \boxed{\theta_2 + \epsilon}, \theta_3, \dots, \theta_n) - J(\theta_1, \boxed{\theta_2 - \epsilon}, \theta_3, \dots, \theta_n)}{2\epsilon}$$

$\vdots$

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

```
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;
```

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

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

Check that $\boxed{\texttt{gradApprox}} \approx \boxed{\texttt{DVec}}$

From backprop.

Implementation Note:

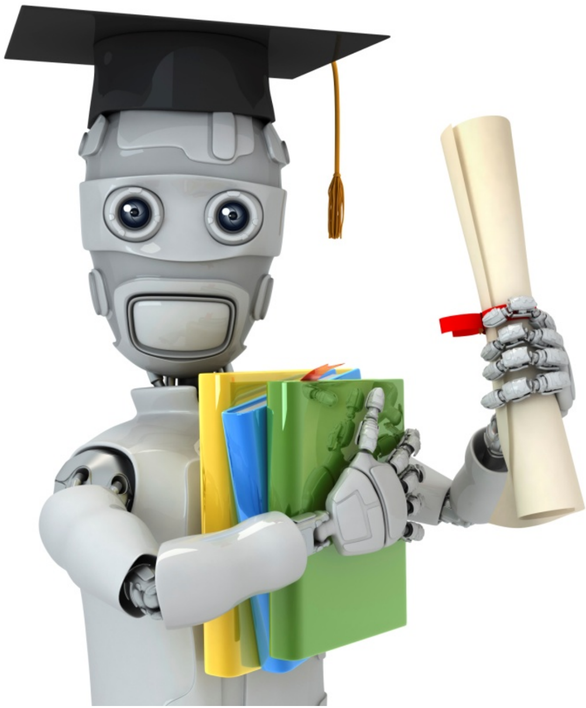
- Implement backprop to compute `DVec` (unrolled $D^{(1)}, D^{(2)}, D^{(3)}$).
- Implement numerical gradient check to compute $\boxed{\texttt{gradApprox}}$.
- Make sure they give similar values.
- Turn off gradient checking. Using backprop code for learning.

DVec

$\delta^{(4)}, \delta^{(3)}, \delta^{(2)}$

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 <u>very</u> slow.



Machine Learning

Neural Networks: Learning

Random initialization

Initial value of Θ

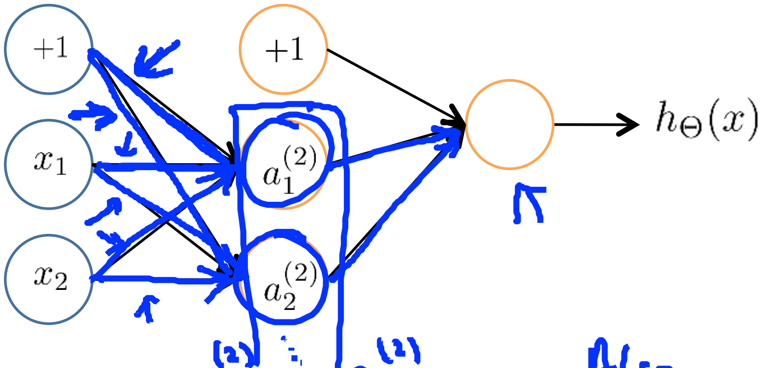
For gradient descent and advanced optimization method, need initial value for Θ.

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

Consider gradient descent

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

Zero initialization



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

$a_1^{(2)} = a_2^{(2)}$. Also $\delta_1^{(2)} = \delta_2^{(2)}$.

$$\frac{\partial}{\partial \Theta_{01}^{(1)}} J(\Theta) = \frac{\partial}{\partial \Theta_{02}^{(1)}} J(\Theta)$$

$$\Theta_{01}^{(1)} = \Theta_{02}^{(1)}$$

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

$a_1^{(2)} = a_2^{(2)}$

Random initialization: Symmetry breaking

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

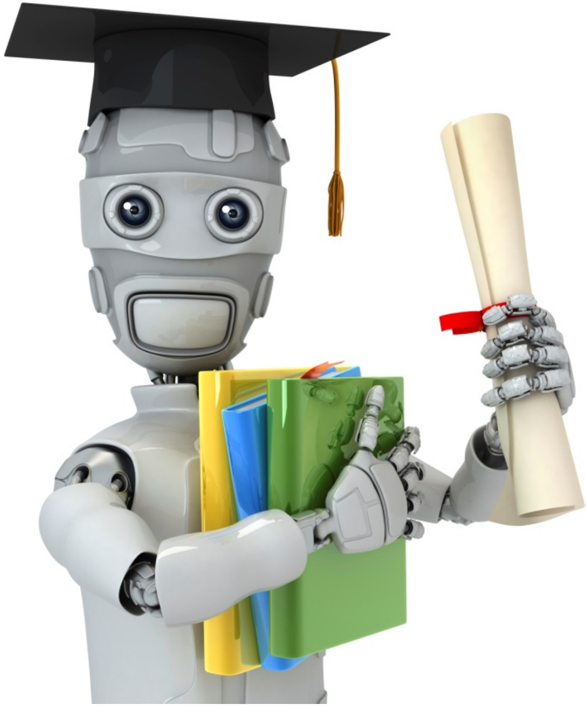
E.g.

random 10×11 matrix (betw. 0 and 1)

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

$[-\epsilon, \epsilon]$

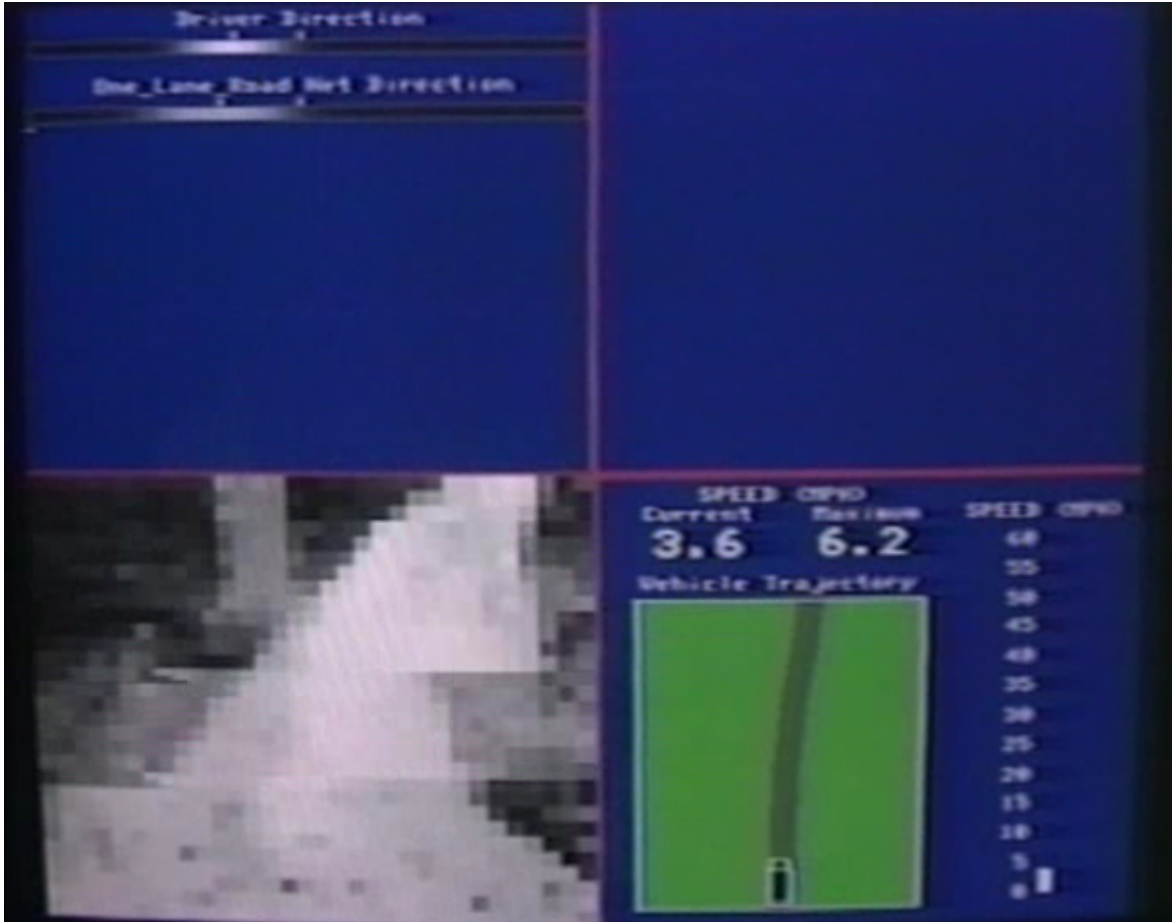
```
Theta2 =  rand(1,11)*(2*INIT_EPSILON)
          - INIT_EPSILON;
```



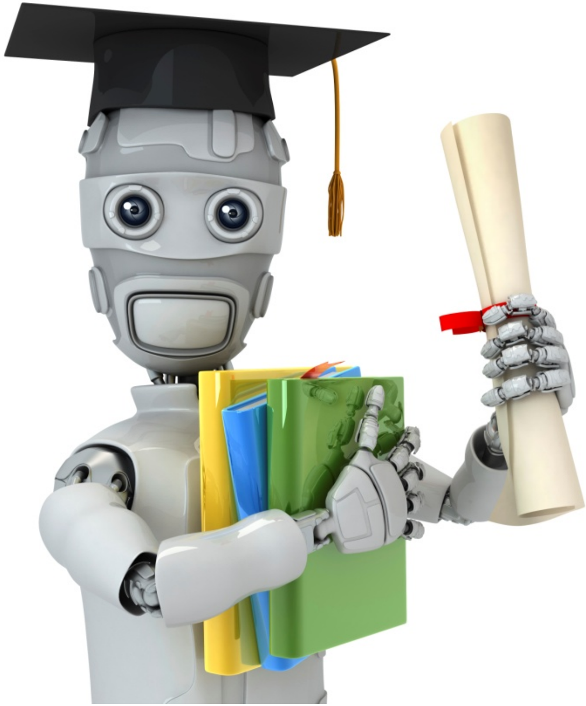
Machine Learning

Neural Networks: Learning

Backpropagation example: Autonomous driving (optional)



[Courtesy of Dean Pomerleau]



Machine Learning

Advice for applying machine learning

Deciding what to try next

Debugging a learning algorithm:

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

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

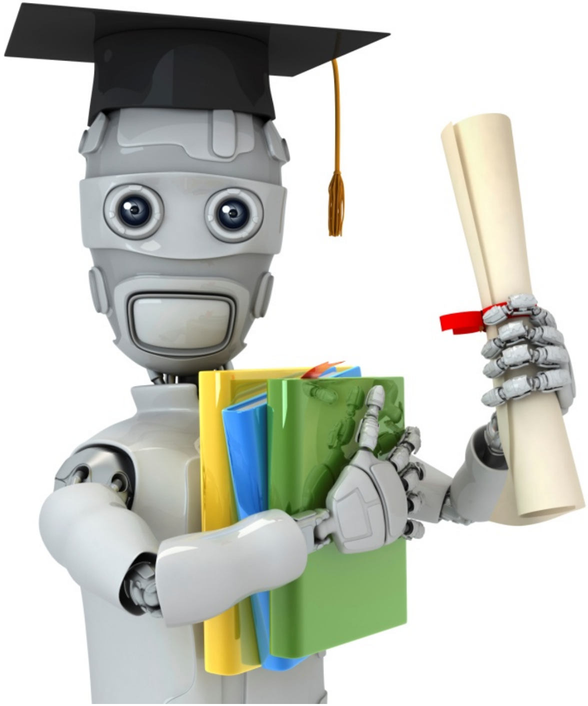
However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- Get more training examples
- Try smaller sets of features $x_1, x_2, x_3, \ldots, x_{100}$
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, \text{etc.})$
- Try decreasing λ
- Try increasing λ

Machine learning diagnostic:

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

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

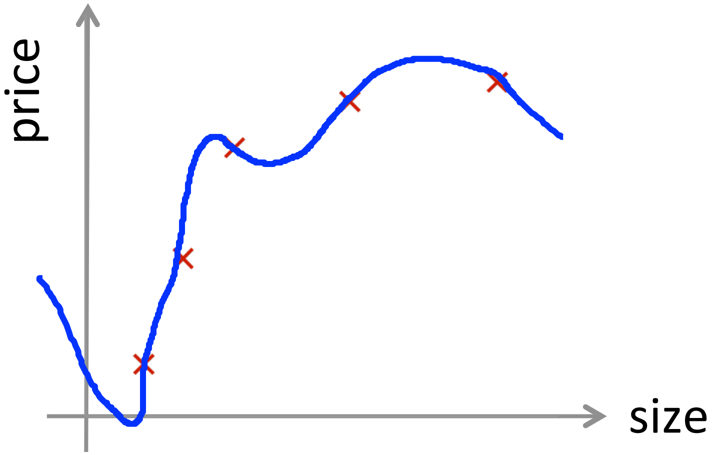


Machine Learning

Advice for applying machine learning

Evaluating a hypothesis

Evaluating your hypothesis



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

Fails to generalize to new examples not in training set.

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

Evaluating your hypothesis

Dataset:

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

70%

Training set

30%

Test set

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

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

m_{test} = no. of test example

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

Andrew Ng

Training/testing procedure for linear regression

- Learn parameter θ from training data (minimizing training error $J(\theta)$) 70%

- Compute test set error:

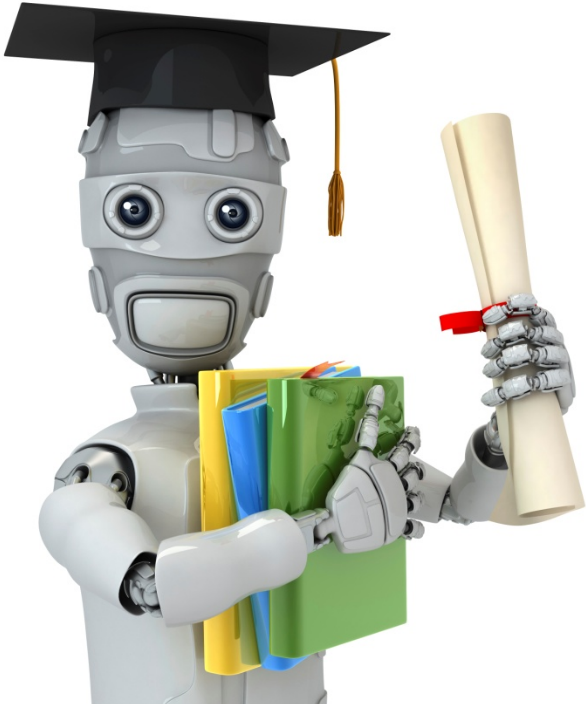
$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right)^2$$

Training/testing procedure for logistic regression

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

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

- Misclassification error (0/1 misclassification error):

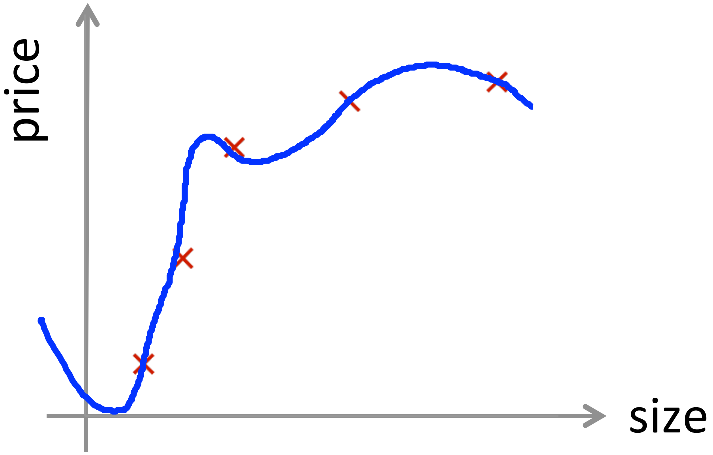


Machine Learning

Advice for applying machine learning

Model selection and training/validation/test sets

Overfitting example



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

Once parameters $\theta_0, \theta_1, \ldots, \theta_4$ 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

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

$d=1$ 1. $h_\theta(x) = \theta_0 + \theta_1 x \longrightarrow \Theta^{(1)} \longrightarrow J_{test}(\Theta^{(1)})$

$d=2$ 2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \longrightarrow \Theta^{(2)} \longrightarrow J_{test}(\Theta^{(2)})$

$d=3$ 3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3 \longrightarrow \Theta^{(3)} \longrightarrow J_{test}(\Theta^{(3)})$

$\vdots$ $\qquad \vdots$

$d=10$ 10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10} \longrightarrow \Theta^{(10)} \longrightarrow J_{test}(\Theta^{(10)})$

Choose $\boxed{\theta_0 + \ldots \theta_5 x^5}$

$\boxed{\Theta_0, \Theta_1, \ldots}$

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

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

Evaluating your hypothesis

Dataset:

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

60% Training set

20% Cross validation set (CV)

20% test set

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

$$(x_{cv}^{(1)}, y_{cv}^{(1)})$$
$$(x_{cv}^{(2)}, y_{cv}^{(2)})$$
$$\vdots$$
$$(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$$

m_{cv} = no. of cv example

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

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

m_{test}

Train/validation/test error

Training error:

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

$J(\theta)$

Cross Validation error:

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

Test error:

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

Model selection

$d=1$ 1. $h_\theta(x) = \theta_0 + \theta_1 x \longrightarrow \min_\theta J(\theta) \rightarrow \Theta^{(1)} \longrightarrow J_{cv}(\Theta^{(1)})$

$d=2$ 2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \longrightarrow \Theta^{(2)} \longrightarrow J_{cv}(\Theta^{(2)})$

$d=3$ 3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3 \longrightarrow \Theta^{(3)}$

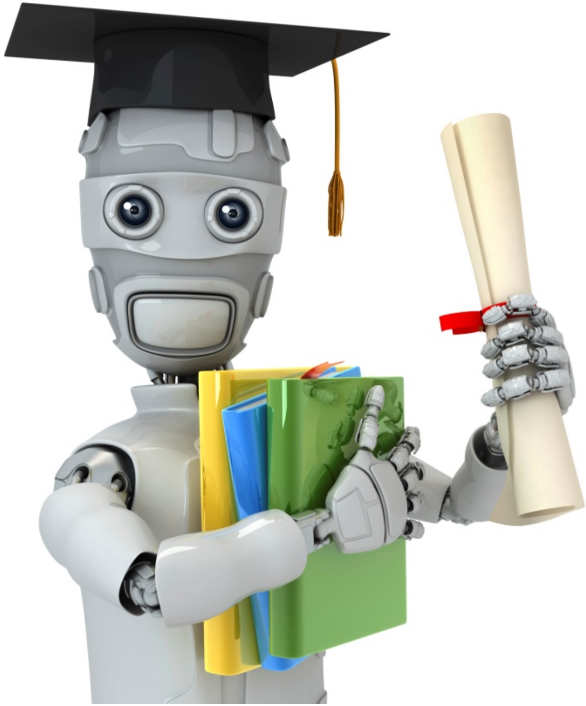
$\vdots \qquad \vdots \qquad \vdots \qquad J_{cv}(\Theta^{(4)})$

$d=10$ 10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10} \longrightarrow \Theta^{(10)} \longrightarrow J_{cv}(\Theta^{(10)})$

$\underline{d=4}$

Pick $\theta_0 + \theta_1 x_1 + \cdots + \theta_4 x^4 \leftarrow$

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

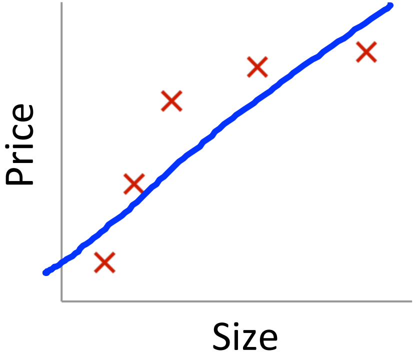


Machine Learning

Advice for applying machine learning

Diagnosing bias vs. variance

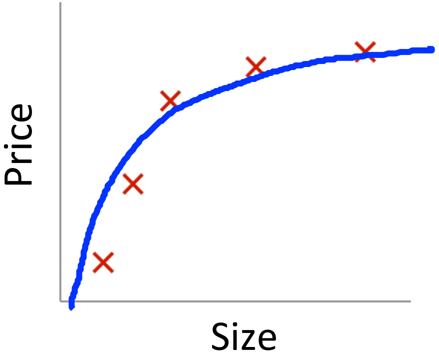
Bias/variance



$$\theta_0 + \theta_1 x$$

High bias
(underfit)

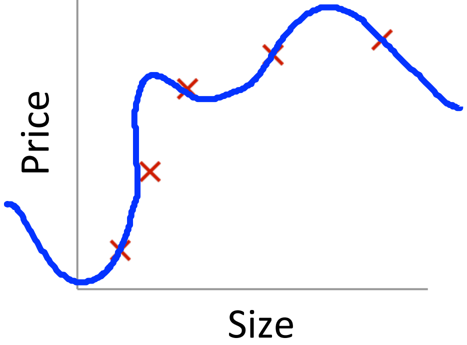
$d=1$



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

"Just right"

$d=2$



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

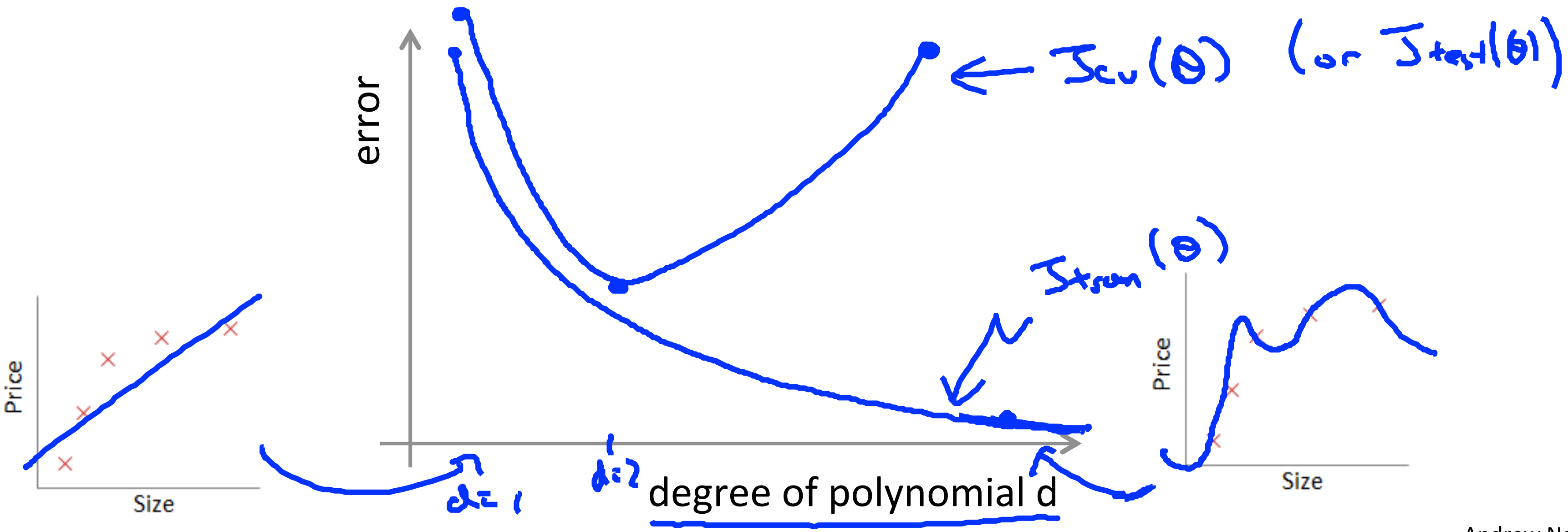
High variance
(overfit)

$d=4$

Bias/variance

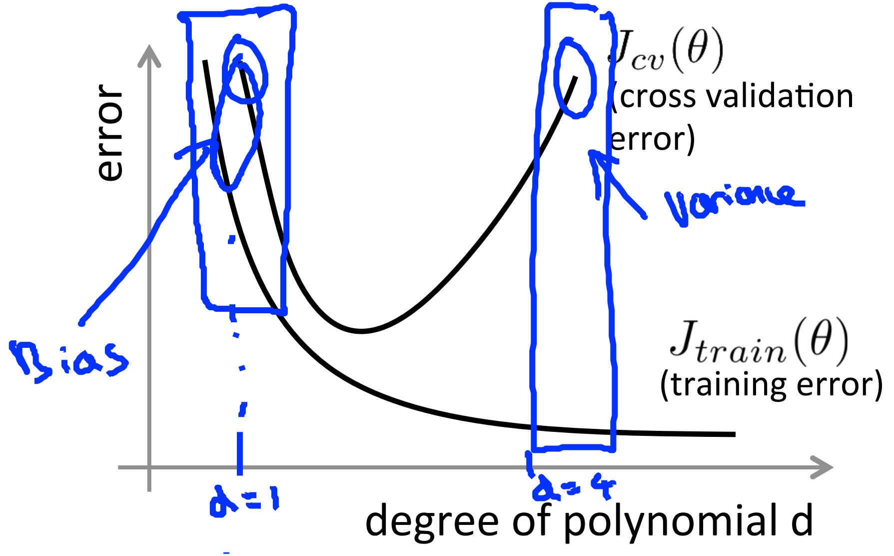
Training error: $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

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

→ $J_{train}(\theta)$ will be high

$J_{cv}(\theta) \approx J_{train}(\theta)$

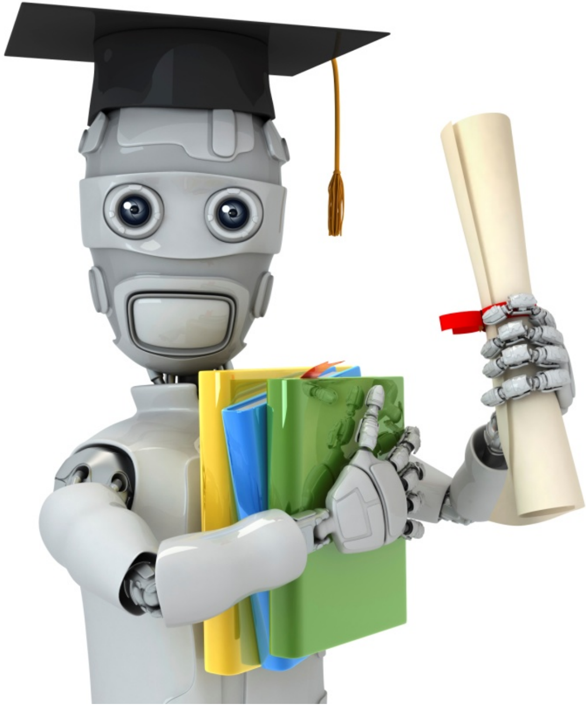
Variance (overfit):

→ $J_{train}(\theta)$ will be low

$J_{cv}(\theta) \gg J_{train}(\theta)$

$\gg$

Andrew Ng



Machine Learning

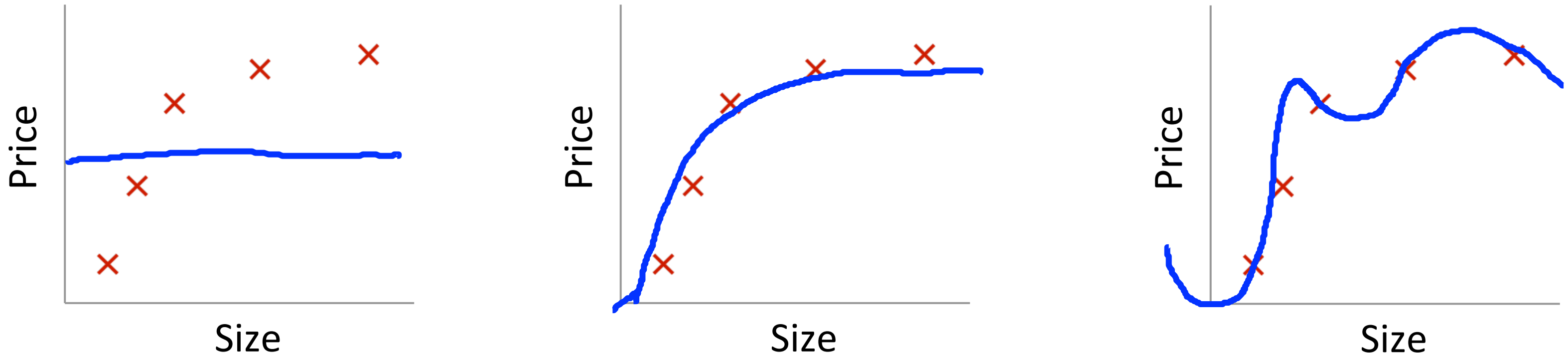
Advice for applying machine learning

Regularization and bias/variance

Linear regression with regularization

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

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



Large λ	Intermediate λ	Small λ
High bias (underfit)	"Just right"	High variance (overfit)

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

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

$\lambda = 0$

Andrew Ng

Choosing the regularization parameter λ

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

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

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

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

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

$J(\theta)$

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 $\lambda = 0$ $\longrightarrow \min_\theta J(\theta) \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$
2. Try $\underline{\lambda = 0.01}$ $\longrightarrow \min_\theta J(\theta) \rightarrow \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$
3. Try $\lambda = \underline{0.02}$ $\longrightarrow \theta^{(3)} \rightarrow J_{cv}(\theta^{(3)})$
4. Try $\lambda = \underline{0.04}$

   $\vdots$
5. Try $\lambda = 0.08$ $\longrightarrow \theta^{(5)}$ $\quad J_{cv}(\theta^{(5)})$

   $\vdots$

12. Try $\underline{\lambda = 10}$ $\longrightarrow \theta^{(12)} \rightarrow J_{cv}(\theta^{(12)})$

$\underline{10.24}$

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

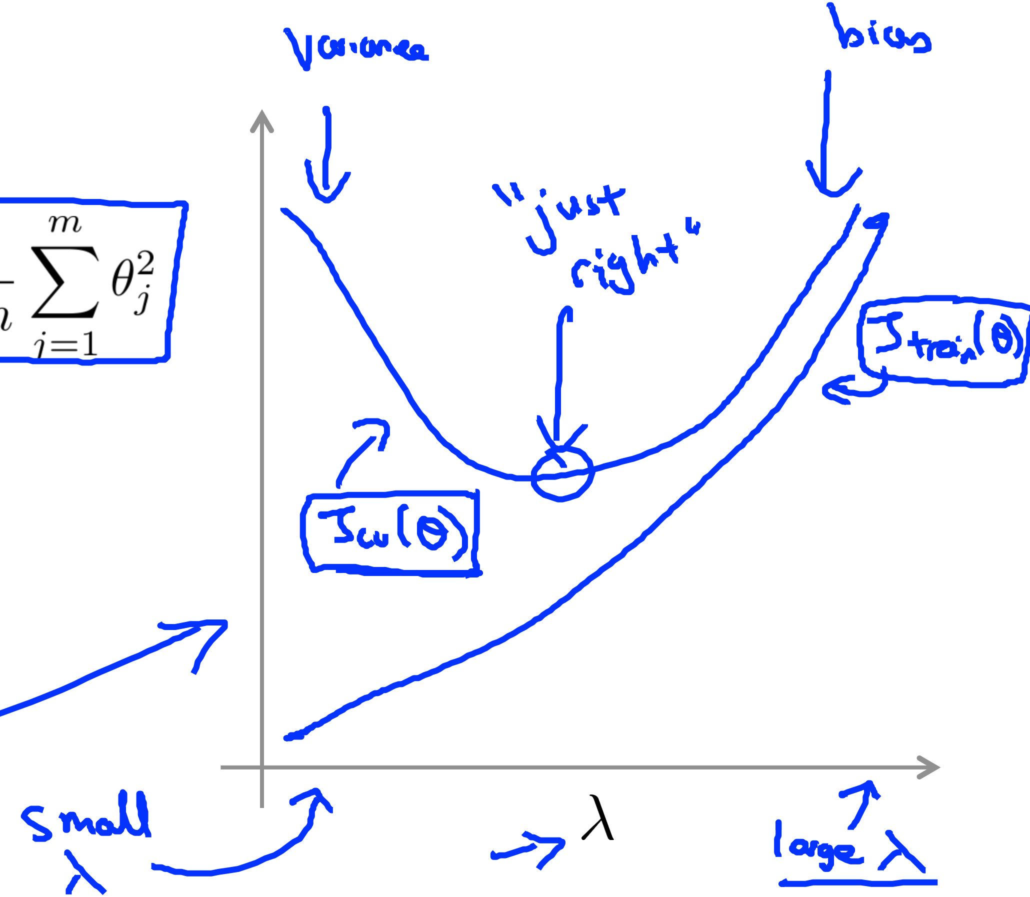
Andrew Ng

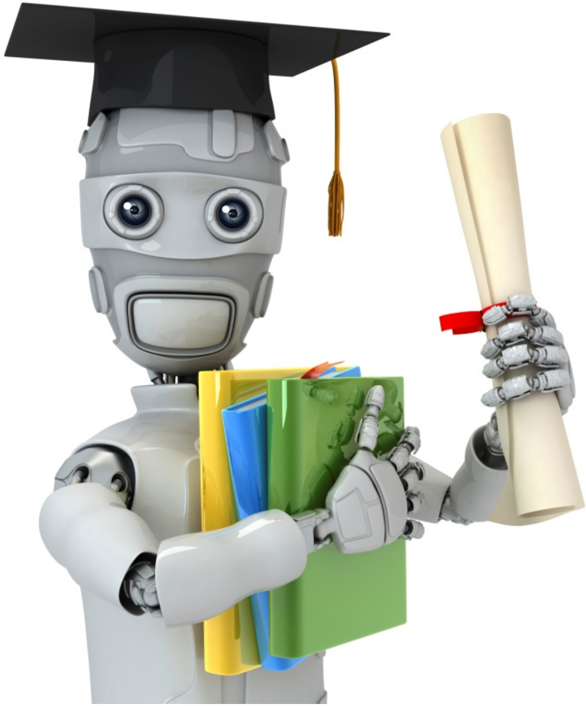
Bias/variance as a function of the regularization parameter λ

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

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

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





Machine Learning

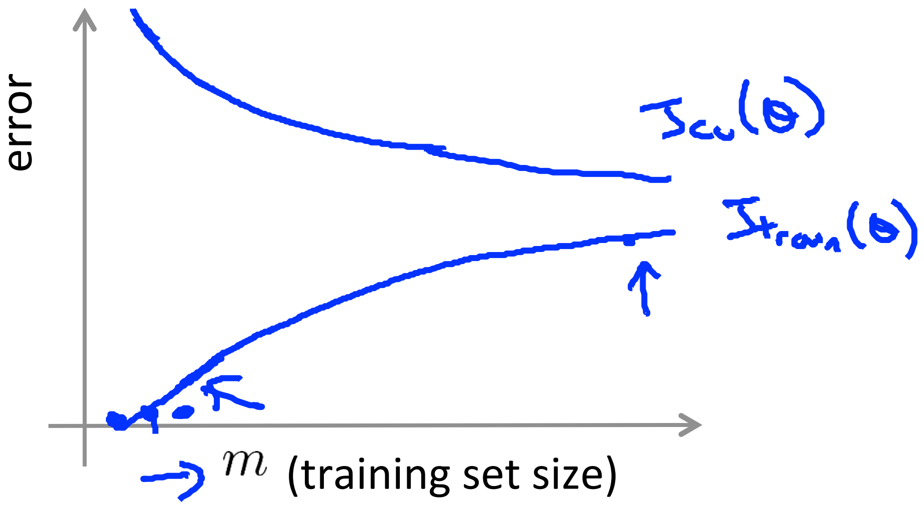
Advice for applying machine learning

Learning curves

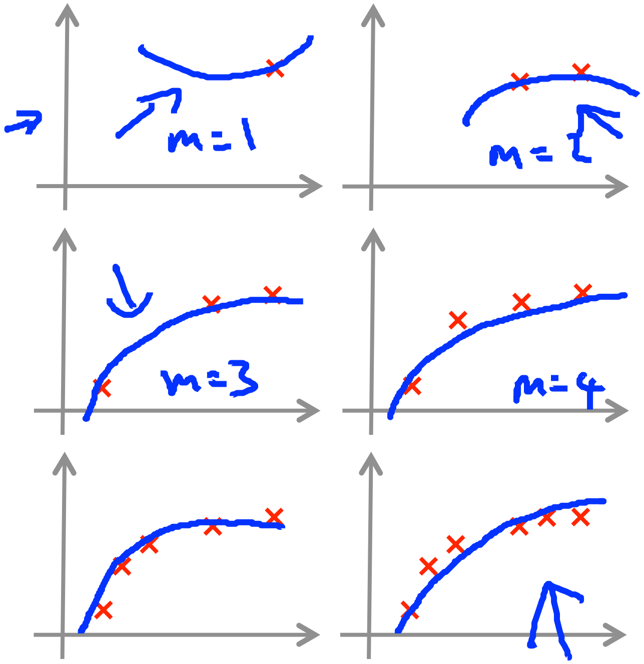
Learning curves

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

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

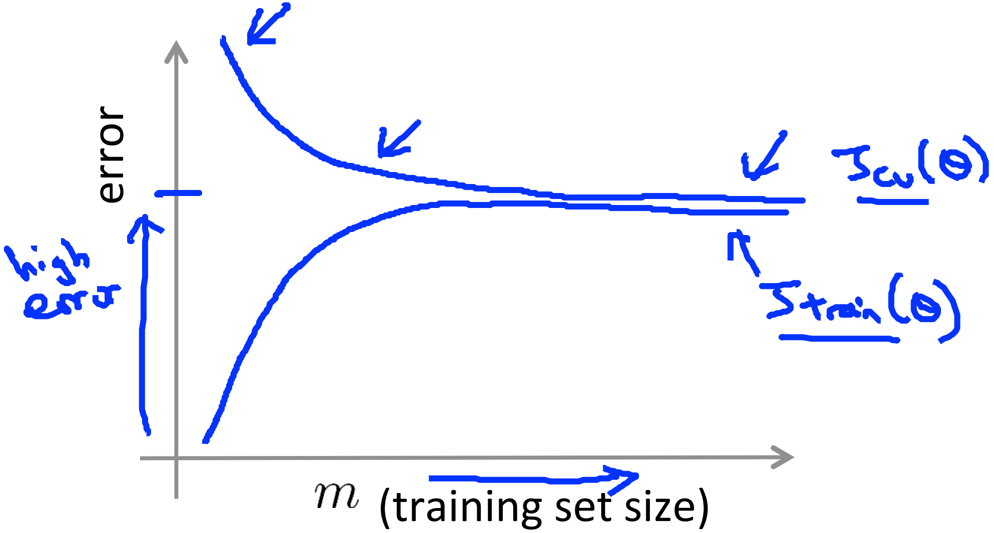


$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$



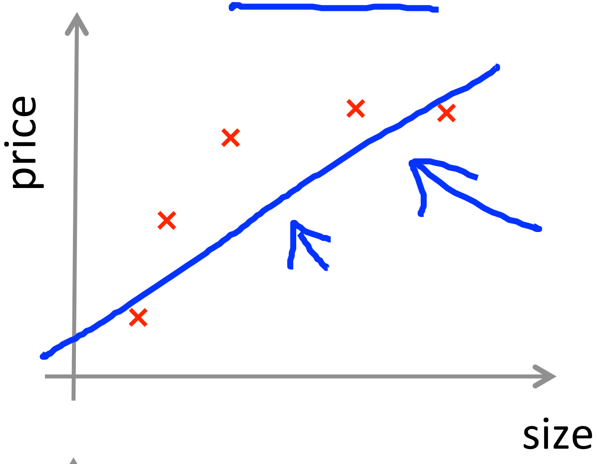
Andrew Ng

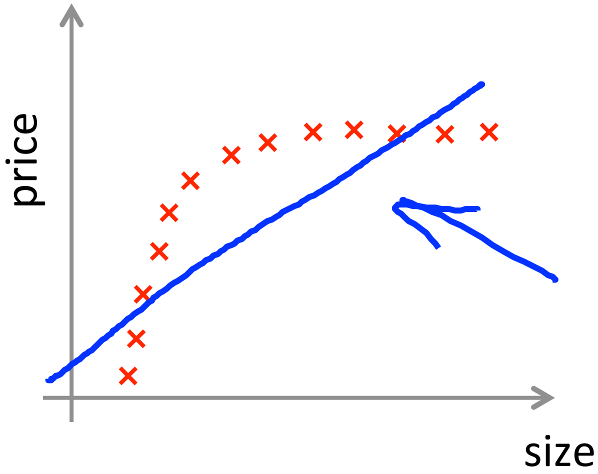
High bias



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

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



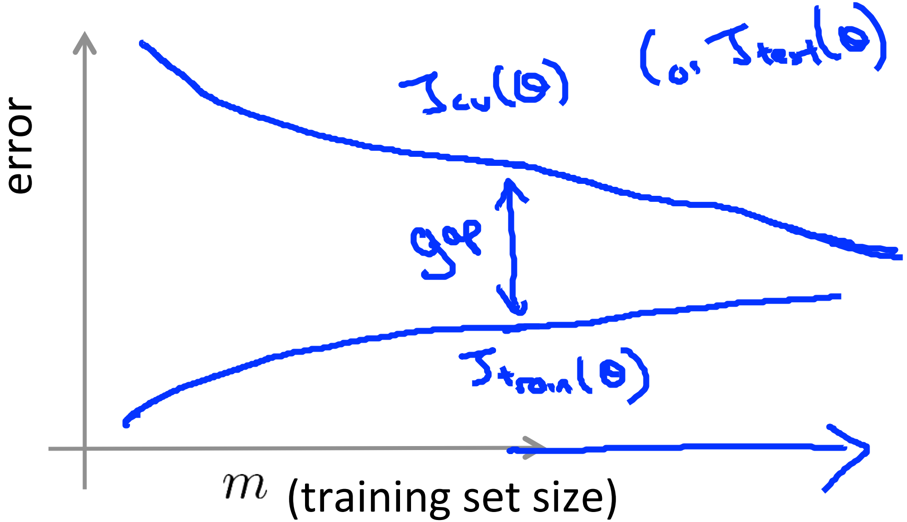


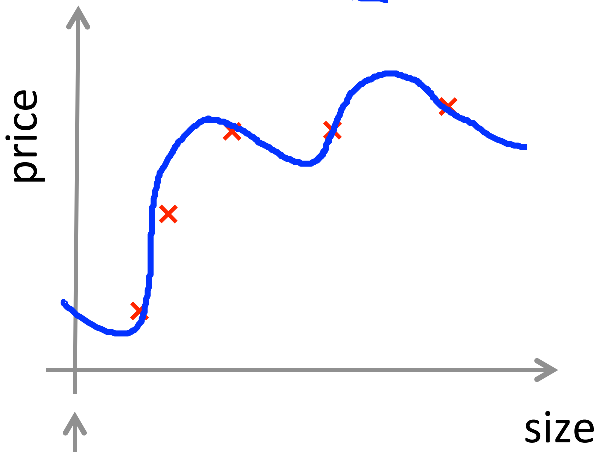
Andrew Ng

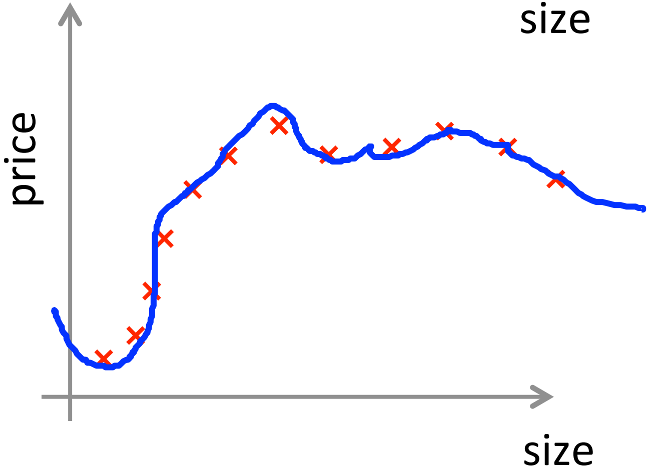
High variance

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

(and small λ)

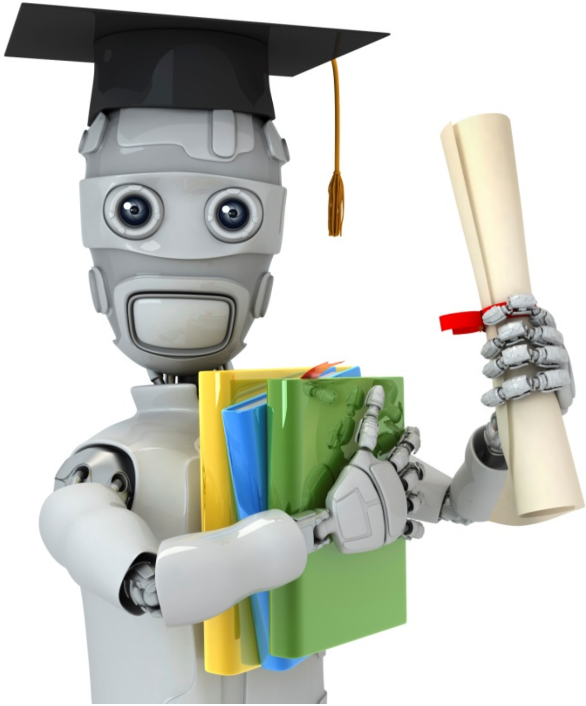






If a learning algorithm is suffering from high variance, getting more training data is likely to help.

Andrew Ng



Machine Learning

Advice for applying machine learning

Deciding what to try next (revisited)

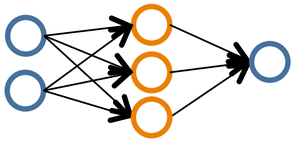
Debugging a learning algorithm:

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 $(x_1^2, x_2^2, x_1 x_2, \text{etc})$ → fixes high bias.
- Try decreasing λ → fixes high bias
- Try increasing λ → fixes high variance

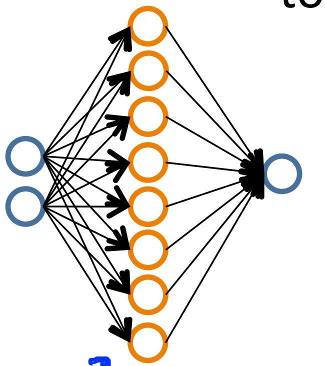
Neural networks and overfitting

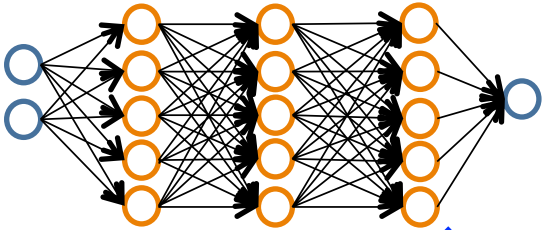
"Small" neural network
(fewer parameters; more prone to underfitting)



Computationally cheaper

"Large" neural network
(more parameters; more prone to overfitting)



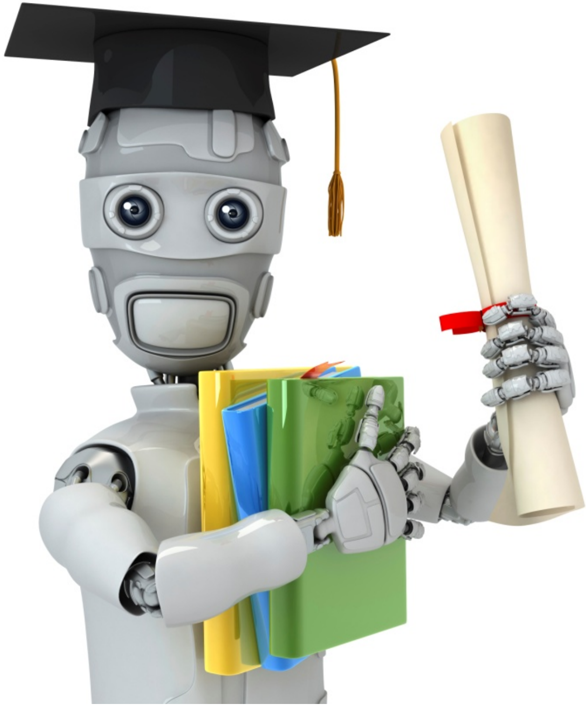


Computationally more expensive.

Use regularization (λ) to address overfitting.

$J_{cv}(\Theta)$

Andrew Ng



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, . . .

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

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

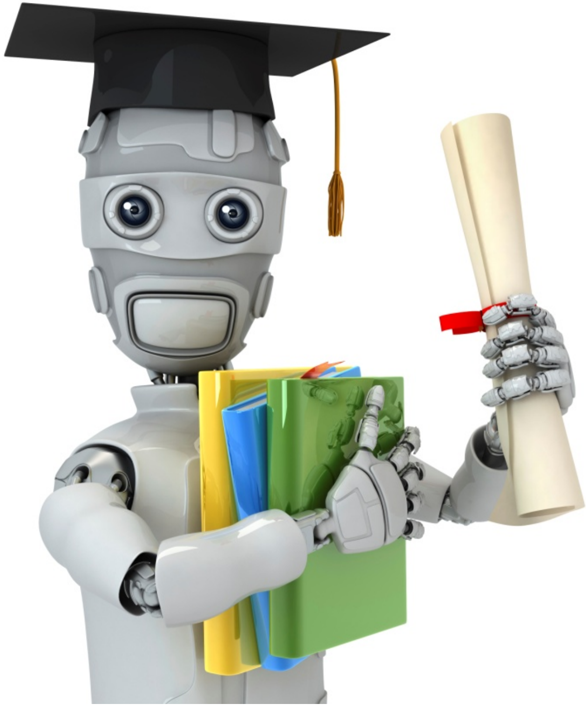
```
From: cheapsales@buystufffromme.com
To: ang@cs.stanford.edu
Subject: Buy now!

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")

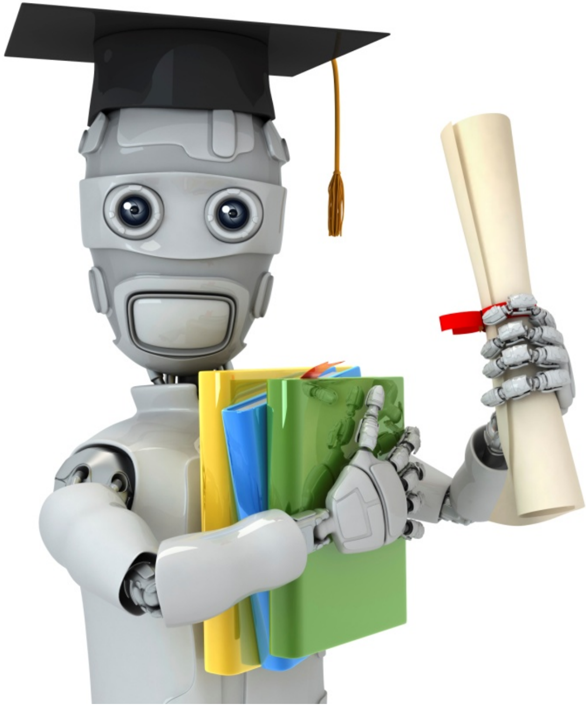
universe/university.

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 $\boxed{\text{1\% error}}$ 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% accuracy (0.8% error)

→ 99.5% accuracy (0.5% error)

Precision/Recall

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

Actual class

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

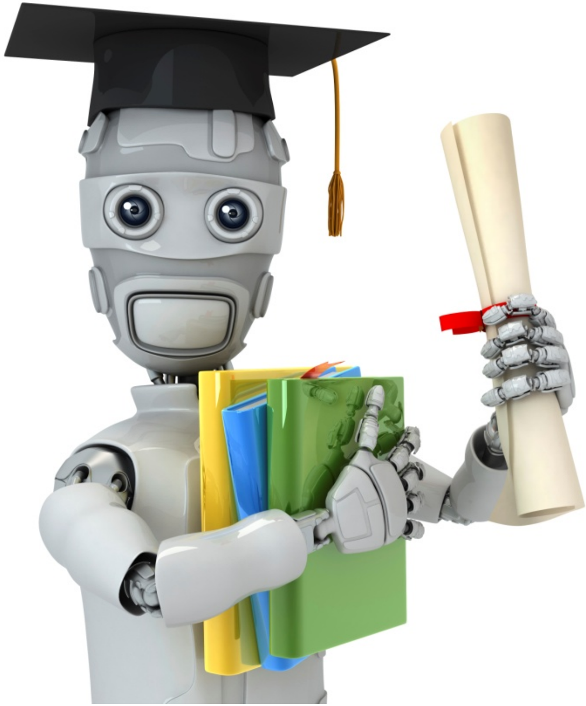
Precision
(Of all patients where we predicted $y = 1$, what fraction actually has cancer?)

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

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

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

$y = 0$
recall $= 0$



Machine Learning

Machine learning system design

Trading off precision and recall

Trading off precision and recall

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

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

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

  Predict 1 if $h_\theta(x) \geq \cancel{0.5}$ $\cancel{0.7}$ $\cancel{0.9}$ 0.3

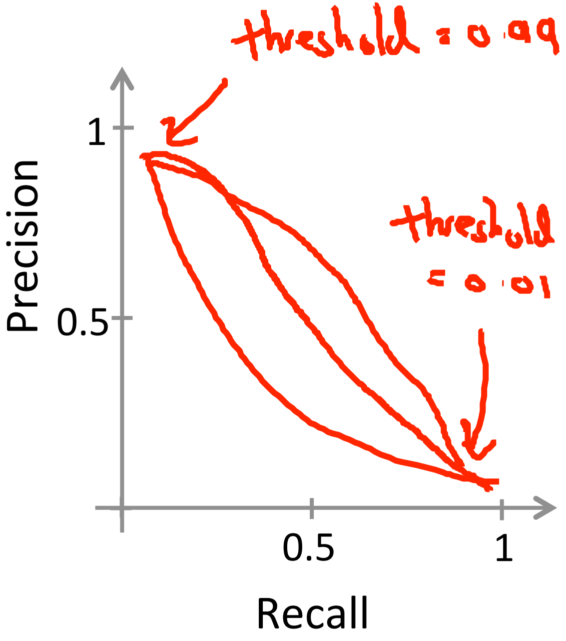
  Predict 0 if $h_\theta(x) < \cancel{0.5}$ $\cancel{0.7}$ $\cancel{0.9}$ 0.3

- 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 $\boxed{h_\theta(x) \geq \text{threshold}}$

Andrew Ng

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

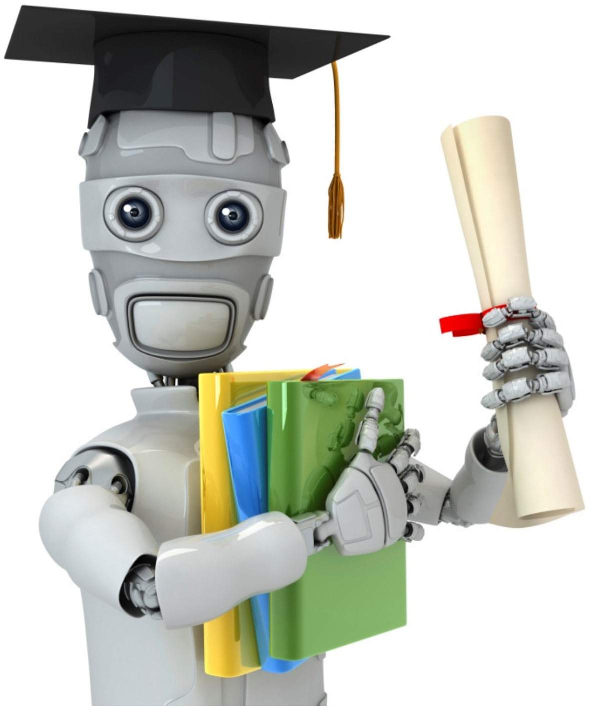
Predict y=1 all the time

Average: $\cancel{\dfrac{P+R}{2}}$

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

P=0 or R=0 ⇒ F-score = 0

P=1 and R=1 ⇒ F-score = 1



Machine Learning

Machine learning system design

Data for machine learning

Designing a high accuracy learning system

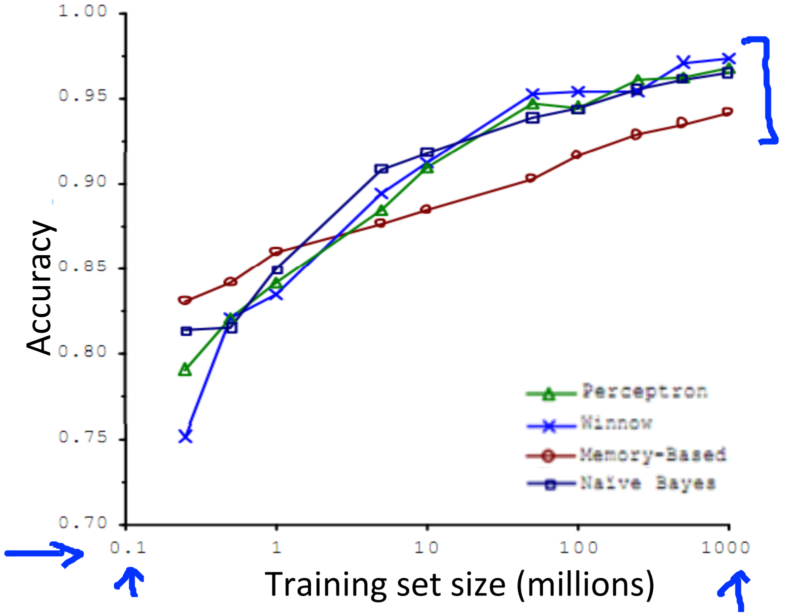
E.g. Classify between confusable words.

{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.

~~to~~, ~~too~~

Example: For breakfast I ate two ______ eggs.

Counterexample: Predict housing price from only size (feet^2) and no other features.

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

Large data rationale

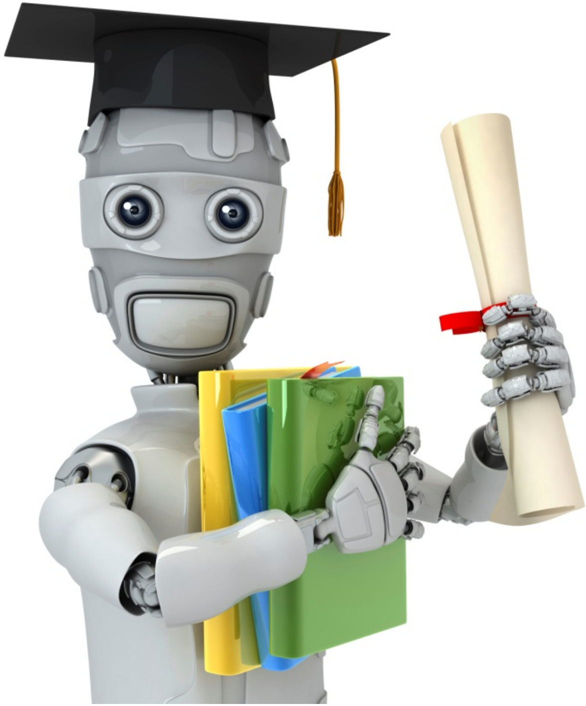
→ Use a learning algorithm with many parameters (e.g. logistic regression/linear regression with many features; neural network with many hidden units). low bias algorithms. ←

→ $J_{train}(\theta)$ will be small.

Use a very large training set (unlikely to overfit) low variance ↙

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

→ $J_{test}(\theta)$ will be small



Machine Learning

Machine learning system design

Artificial data synthesis

Artificial data synthesis for photo OCR

[http://www.publicdomainpictures.net/view-image.php?image=5745&picture=times-square]

Artificial data synthesis for photo OCR

Real data

Abcdefg
Abcdefg
Abcdefg
Abcdefg
Abcdefg

[Adam Coates and Tao Wang]

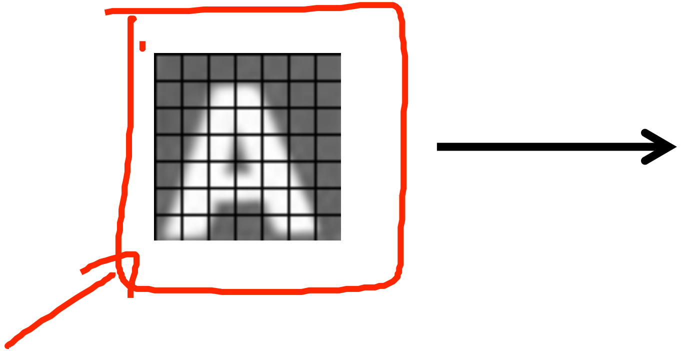
Artificial data synthesis for photo OCR

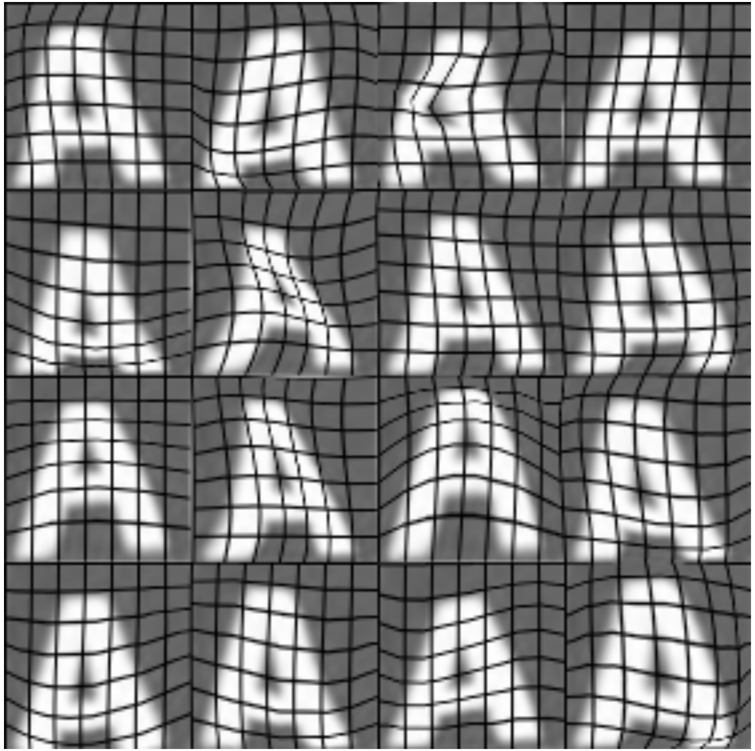
Real data

Synthetic data

[Adam Coates and Tao Wang]

Synthesizing data by introducing distortions





Synthesizing data by introducing distortions: Speech recognition

Original audio (counting from zero to five)

Noisy background: Machinery

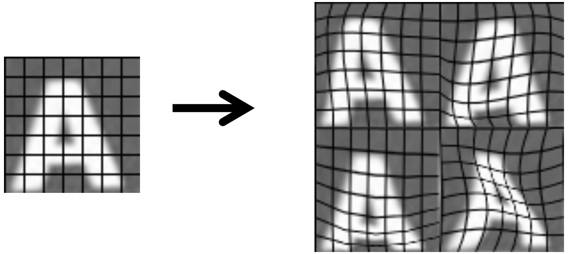
Noisy background: Crowd

Audio on bad cellphone connection

[www.pdsounds.org]

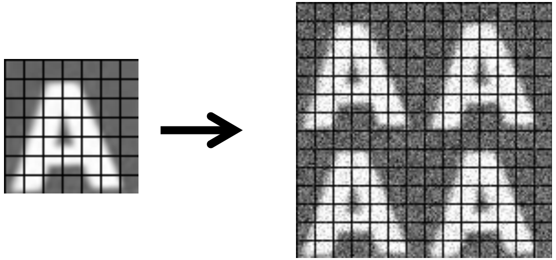
Synthesizing data by introducing distortions

Distortion introduced should be representation of the type of noise/distortions in the test set.



Audio:
Background noise,
bad cellphone connection

Usually does not help to add purely random/meaningless noise to your data.

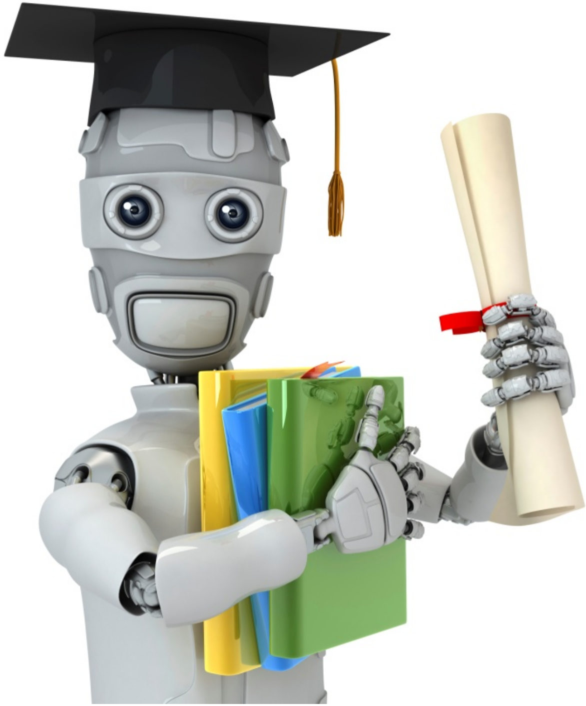


$x_i =$ intensity (brightness) of pixel i

$x_i \leftarrow x_i +$ random noise

[Adam Coates and Tao Wang]

Andrew Ng



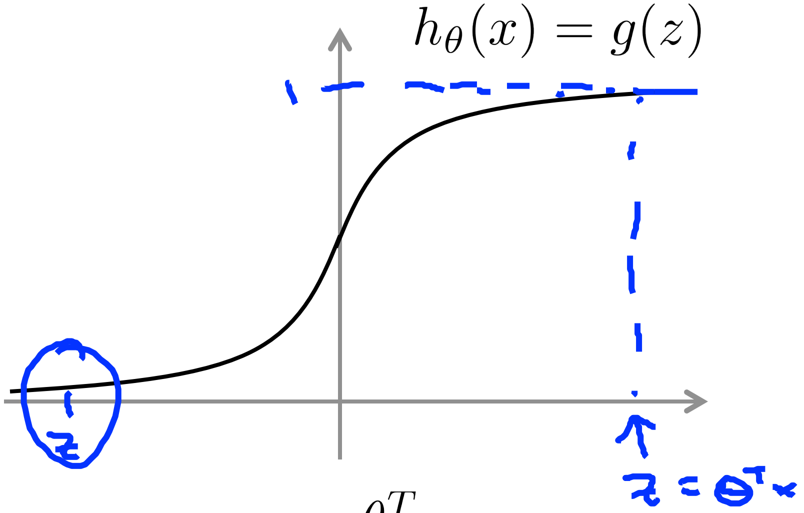
Machine Learning

Support Vector Machines

Optimization objective

Alternative view of logistic regression

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



$z = \theta^T x$

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

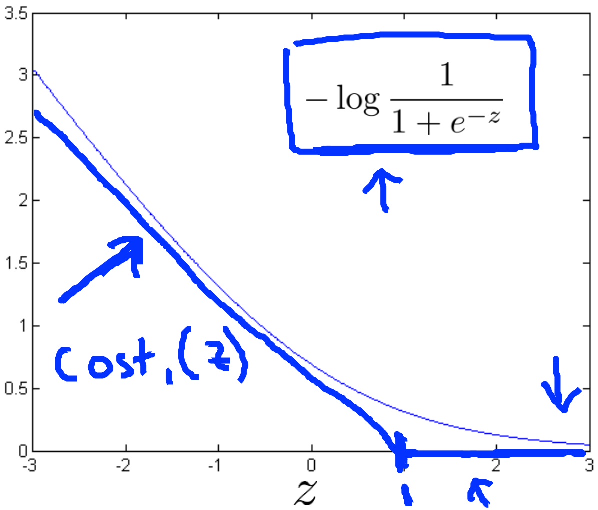
(x, y)

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

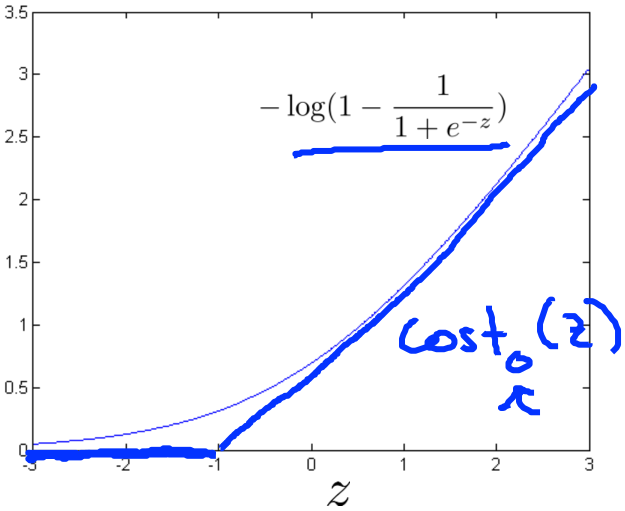
$$= \boxed{-y \log \frac{1}{1+e^{-\theta^T x}}} - \boxed{(1-y)\log\left(1 - \frac{1}{1+e^{-\theta^T x}}\right)}$$

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

$z = \theta^T x$



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



Support vector machine

Logistic regression:

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

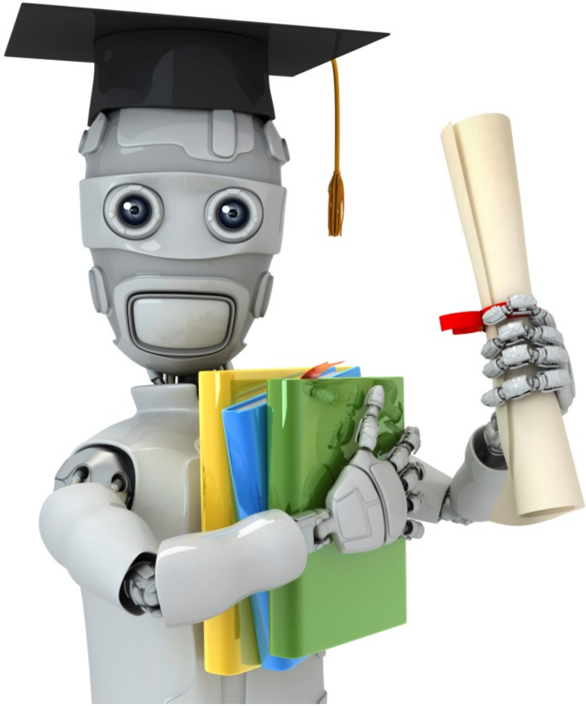
Support vector machine:

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

SVM hypothesis

$$\min_\theta C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

Hypothesis:



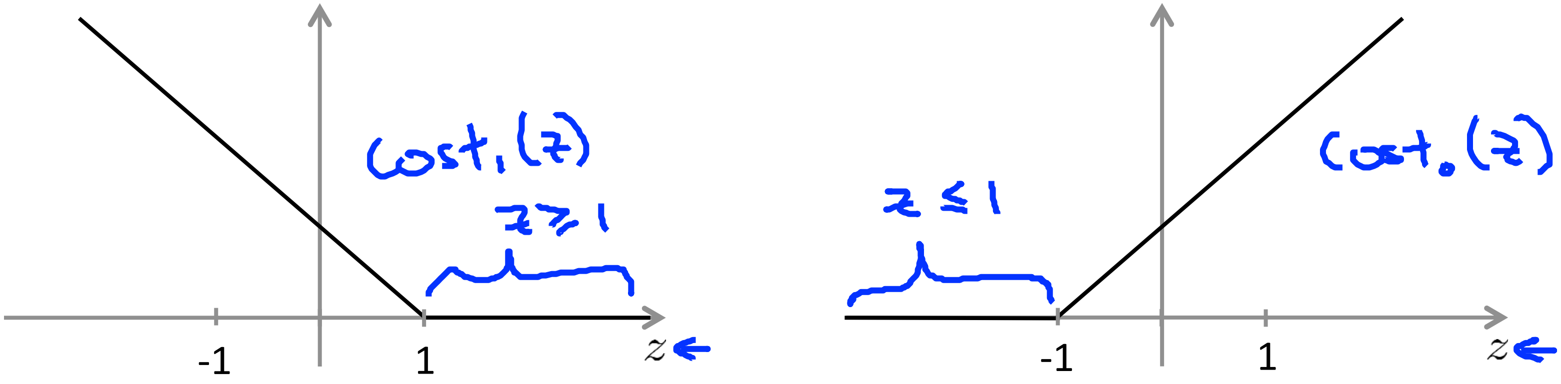
Machine Learning

Support Vector Machines

Large Margin Intuition

Support Vector Machine

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} \underline{cost_1(\theta^T x^{(i)})} + (1 - y^{(i)}) \underline{cost_0(\theta^T x^{(i)})} \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$



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

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

$C = 100{,}000$

SVM Decision Boundary

$$\min_{\theta} C \underbrace{\boxed{\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right]}}_{=0} + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

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

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

$$\min_{\theta} \cancel{C \times 0} + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

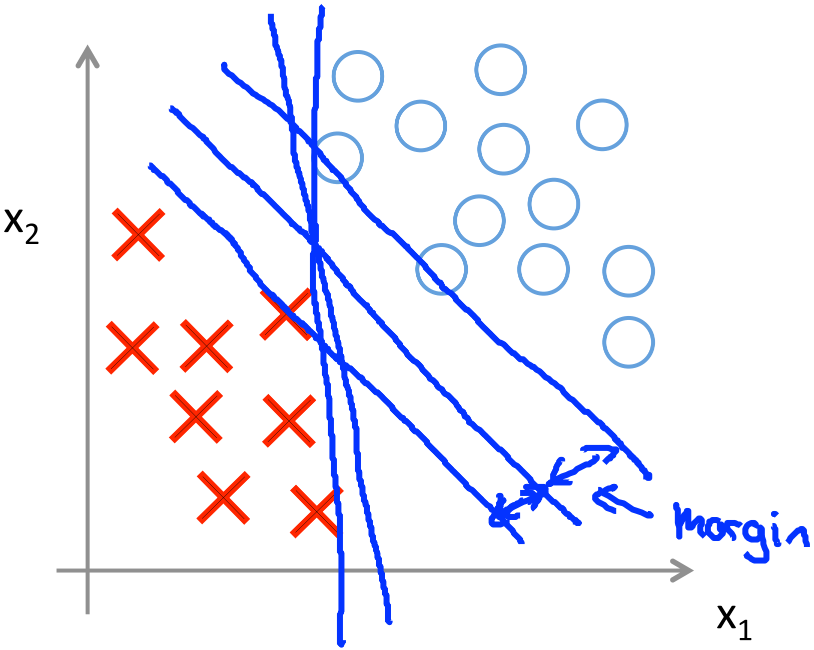
s.t. $\theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$

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

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

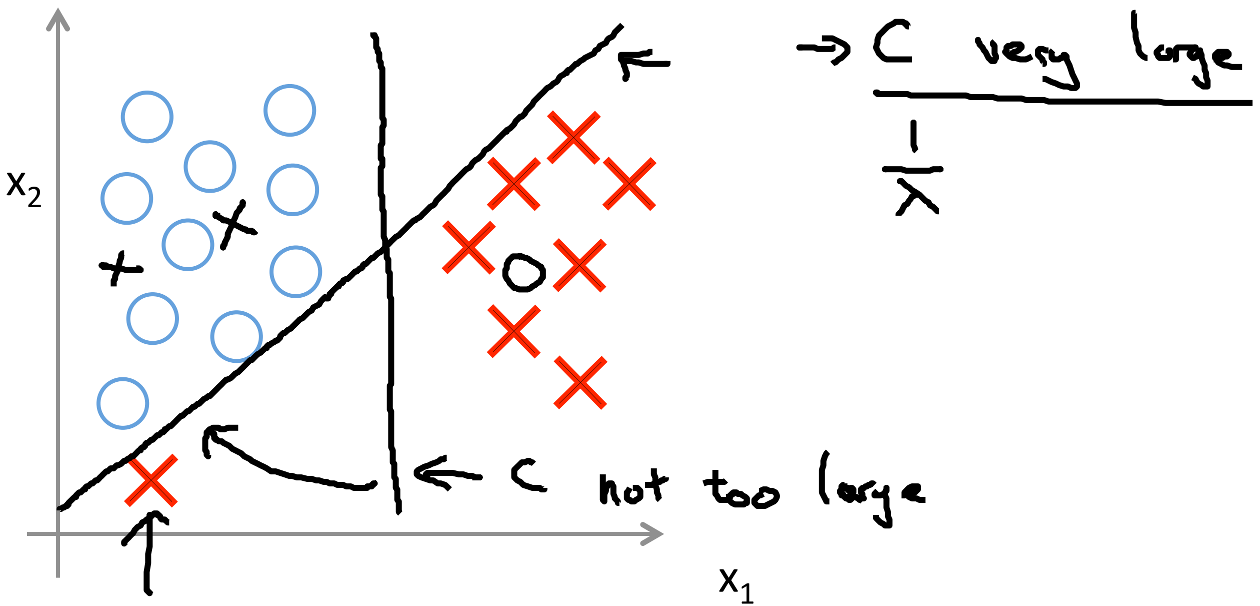
$\theta^T x^{(i)} \leq -1$

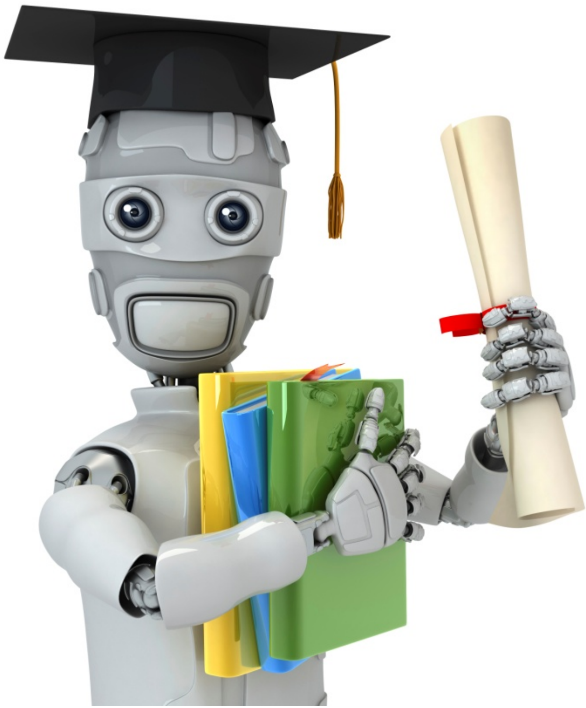
SVM Decision Boundary: Linearly separable case



Large margin classifier

Large margin classifier in presence of outliers



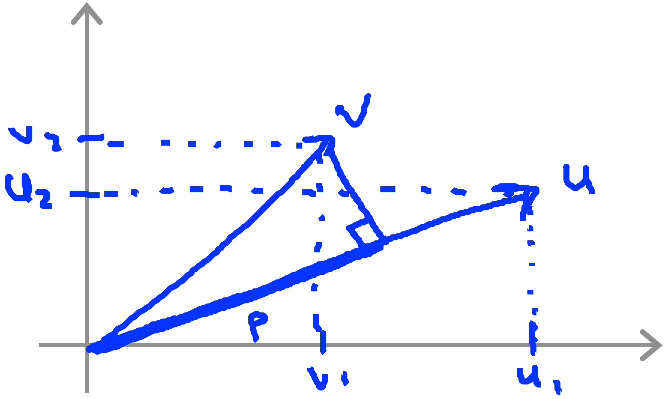


Machine Learning

Support Vector Machines

The mathematics behind large margin classification (optional)

Vector Inner Product



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

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

$\|u\|$ = length of vector u

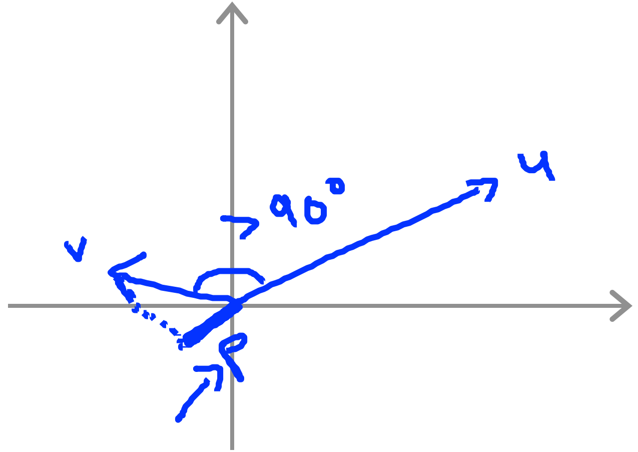
$= \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$

p = length of projection of v onto u.

Signed

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

$= u_1 v_1 + u_2 v_2$ $\qquad p \in \mathbb{R}$



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

$p < 0$

SVM Decision Boundary

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

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

$$\sqrt{\theta_1^2+\theta_2^2} = \|\theta\|$$

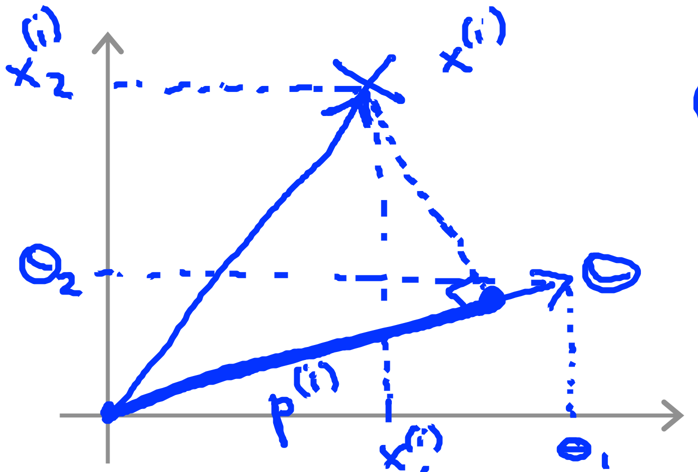
s.t. $\boxed{\theta^T x^{(i)} \geq 1}$ if $y^{(i)} = 1$

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

Simplification: $\theta_0 = 0$. $n=2$

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

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



$$\theta^T x^{(i)} = \boxed{p^{(i)} \cdot \|\theta\|}$$

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

SVM Decision Boundary

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

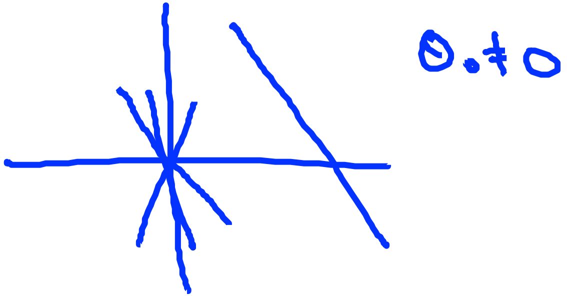
$$\text{s.t.} \quad \boxed{p^{(i)} \cdot \|\theta\| \geq 1} \quad \text{if } y^{(i)} = 1$$

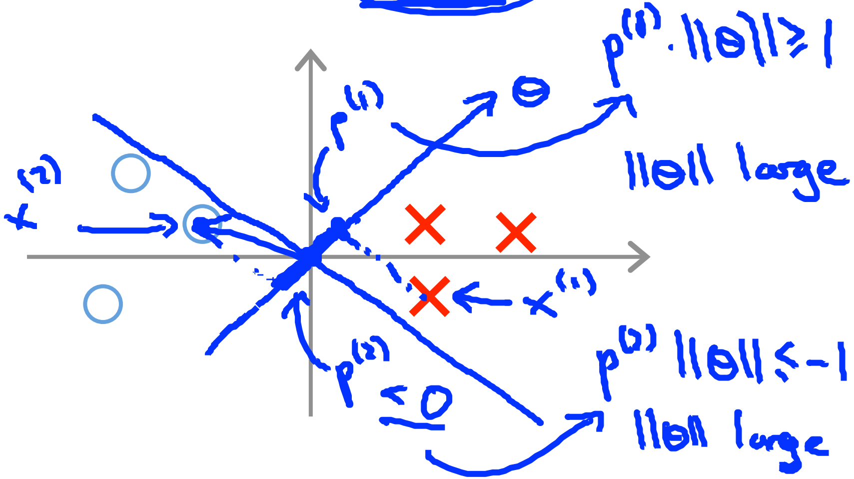
$$p^{(i)} \cdot \|\theta\| \leq -1 \quad \text{if } y^{(i)} = 1$$

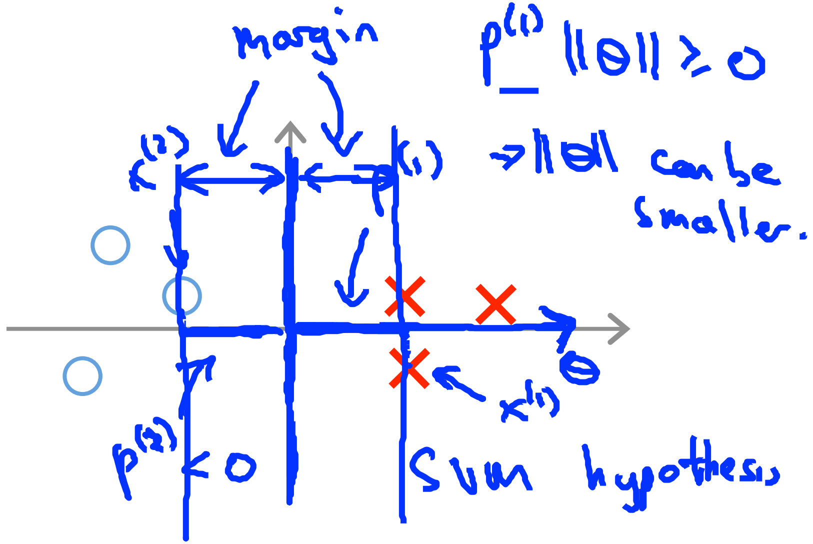
C very large

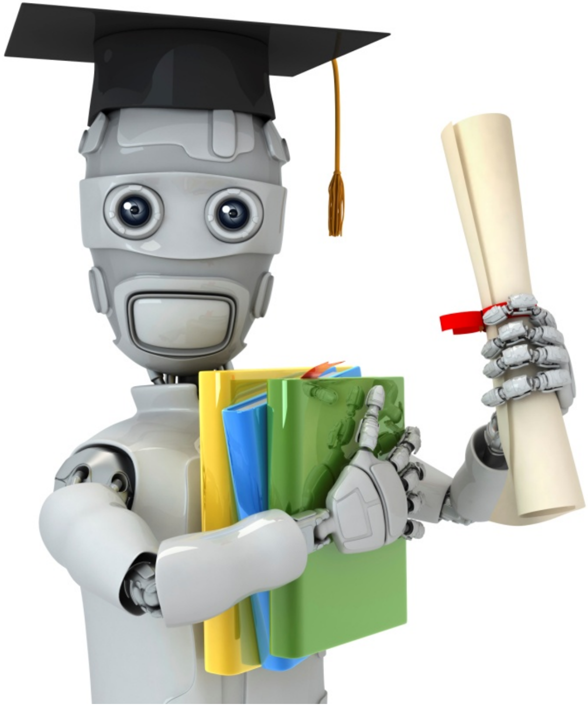
where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplification: $\theta_0 = 0$







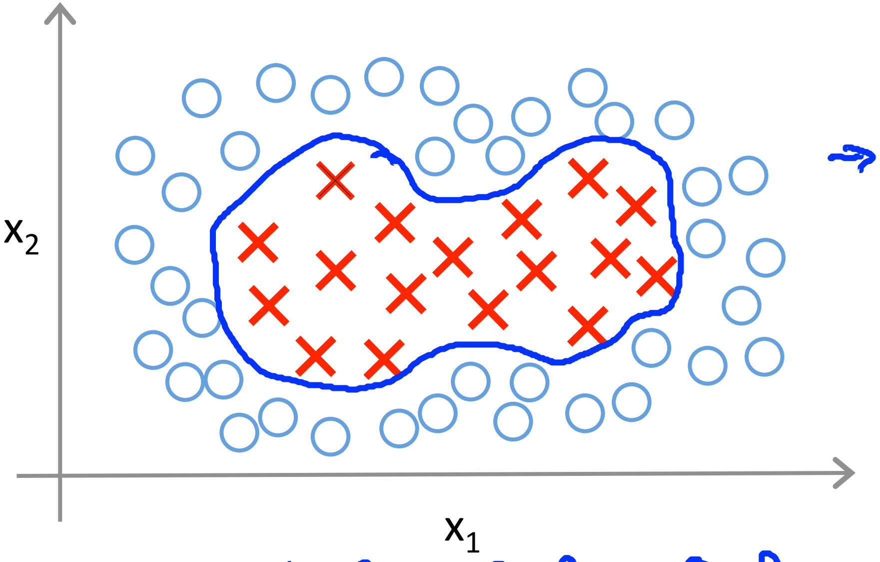


Machine Learning

Support Vector Machines

Kernels I

Non-linear Decision Boundary



Predict $y = 1$ if

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

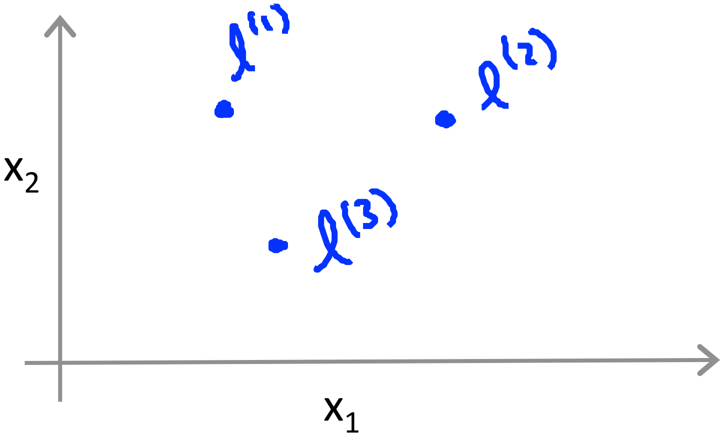
$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \cdots \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \cdots$$

$f_1 = x_1$, $f_2 = x_2$, $f_3 = x_1 x_2$, $f_4 = x_1^2$, $f_5 = x_2^2$, ...

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

Kernel



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

Given x:

$\|w\|$

$\|x - l^{(1)}\|^2$

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

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

$f_3 = \text{similarity}(x, l^{(3)}) = \exp(\dots)$

Kernel (Gaussian kernels)

$k(x, l^{(i)})$

Andrew Ng

Kernels and Similarity

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

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

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

$l^{(1)} \to f_1$

$l^{(2)} \to f_2$

$l^{(3)} \to f_3$.

x

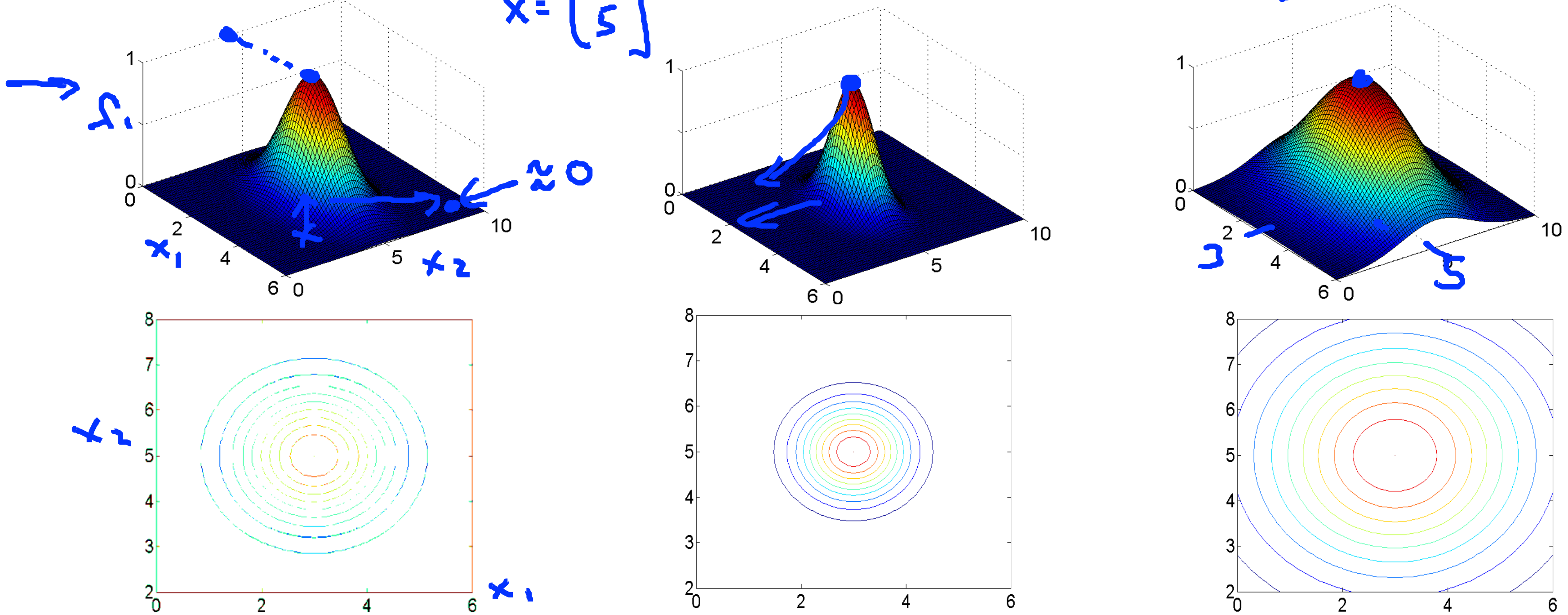
If x if far from $l^{(1)}$:

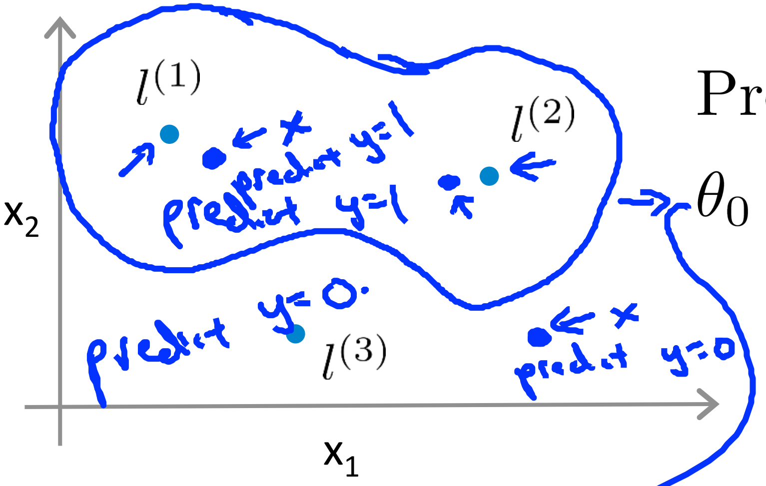
$$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0.$$

Example:

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

$\sigma^2 = 1$ $\qquad$ $x = \begin{bmatrix}3\\5\end{bmatrix}$ $\qquad$ $\sigma^2 = 0.5$ $\qquad$ $\sigma^2 = 3$





Predict "1" when

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

$\theta_0 = -0.5, \quad \theta_1 = 1, \quad \theta_2 = 1, \quad \theta_3 = 0$

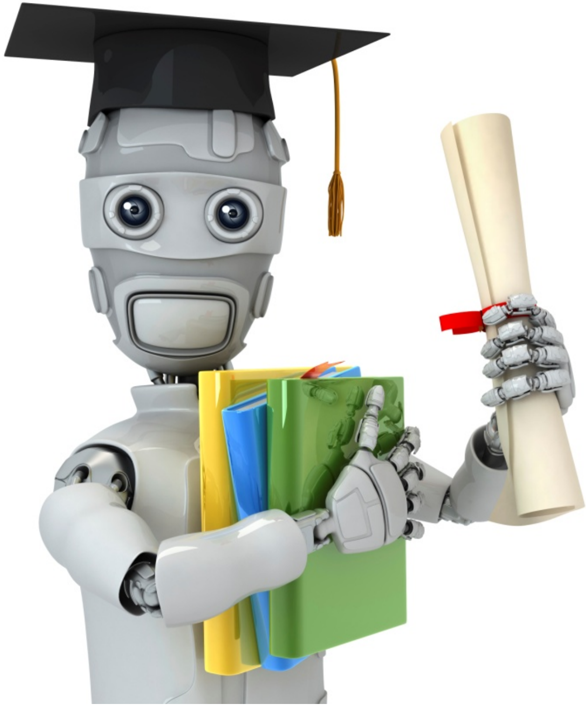
$f_1 \approx 1, \quad f_2 \approx 0, \quad f_3 \approx 0.$

$\rightarrow \theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0$

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

$f_1, f_2, f_3 \approx 0$

$\rightarrow \theta_0 + \theta_1 f_1 + \ldots \approx -0.5 < 0$

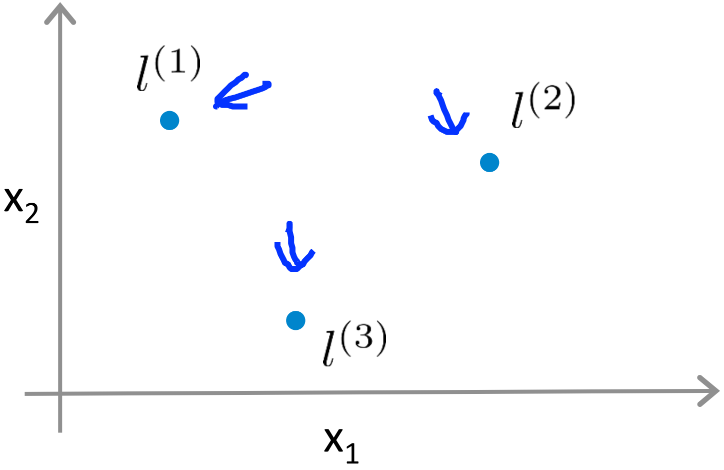


Machine Learning

Support Vector Machines

Kernels II

Choosing the landmarks

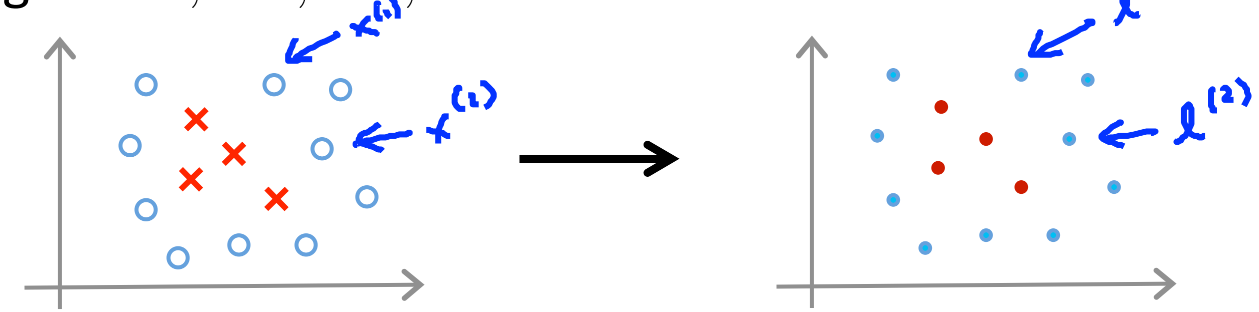


Given x:

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

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

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



$l^{(1)}$
$l^{(2)}$
$\vdots$
$l^{(m)}$

SVM with Kernels

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

Given example x:

$$f_1 = \text{similarity}(x, l^{(1)}) \quad \leftarrow x^{(1)}$$
$$f_2 = \text{similarity}(x, l^{(2)})$$
$$\ldots$$

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

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

$$x^{(i)} \rightarrow \begin{bmatrix} f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)}) \\ f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)}) \\ \vdots \\ f_m^{(i)} \quad \text{sim}(x^{(i)}, l^{(m)}) \end{bmatrix}$$

$$f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = \exp\left(-\frac{0}{2\sigma^2}\right) = 1 \quad \leftarrow x^{(i)}$$

$x^{(i)} \in \mathbb{R}^{n+1}$ (or $\mathbb{R}^n$)

$$f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$$

$f_0^{(i)} = 1$

SVM with Kernels

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

→ Predict "y=1" if $\theta^T f \geq 0$ $\qquad \theta_0 f_0 + \theta_1 f_1 + \cdots + \theta_m f_m$

Training:

$n = m$

$$\min_{\theta} C \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) + \boxed{\frac{1}{2} \sum_{j=1}^{\cancel{n}\, m} \theta_j^2}$$

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

$\rightarrow \theta_0$

$$\sum_j \theta_j^2 = \theta^T \theta \leftarrow \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} \quad \text{(ignore } \theta_0\text{)}$$

$\|\theta\|^2$

$\theta^T M \theta$

$m = 10{,}000$

SVM parameters:

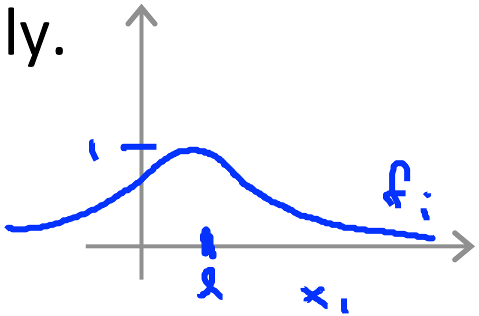
C ($= \frac{1}{\lambda}$). Large C: Lower bias, high variance. (small λ)

Small C: Higher bias, low variance. (large λ)

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

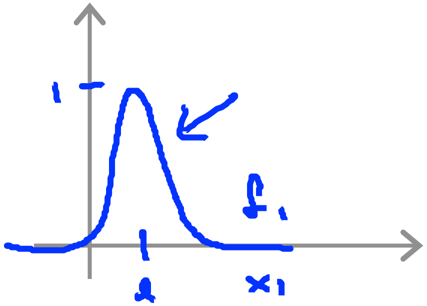
Higher bias, lower variance.

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

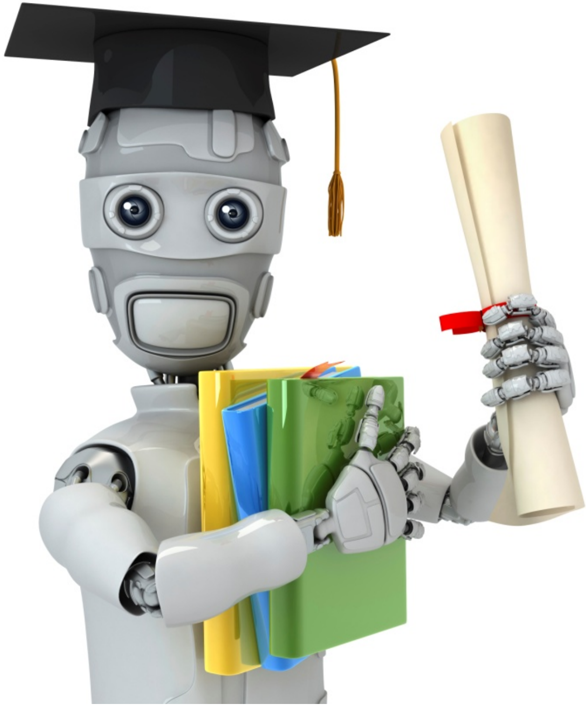


Small σ^2: Features f_i vary less smoothly.

Lower bias, higher variance.



Andrew Ng



Machine Learning

Support Vector Machines

Using an SVM

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

Need to specify:

→ Choice of parameter C.

Choice of kernel (similarity function):

E.g. No kernel ("linear kernel")

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

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

→ 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 σ^2.

$x \in \mathbb{R}^n$, n small

and/or m large

Kernel (similarity) functions:

$x^{(i)}$ $\quad l^{(j)} = x^{(j)}$

f_i

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

$$f = \exp\left(-\frac{\| \mathbf{x1} - \mathbf{x2} \|^2}{2\sigma^2}\right)$$

```
return
```

$x \rightarrow f_1, f_2, \ldots, f_m$

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

$x \in \mathbb{R}^n$

$\boxed{\|x - l\|^2}$

$v = x - l$

$\|v\|^2 = v_1^2 + v_2^2 + \cdots + v_n^2$

$= (x_1 - l_1)^2 + (x_2 - l_2)^2 + \cdots + (x_n - l_n)^2$

1000 feet^2 $\quad$ 1-5 bedrooms

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:

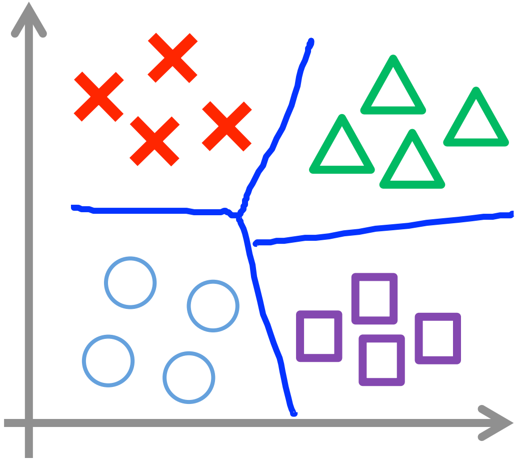
- Polynomial kernel: $k(x, l) = (x^T l)^2$, $(x^T l)^3$, $(x^T l + 1)^3$, $(x^T l + 5)^4$

  $(x^T l + \text{constant})^{\text{degree}}$

- More esoteric: String kernel, chi-square kernel, histogram intersection kernel, …

  $\text{sim}(x, l)$

Multi-class classification



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

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 $i = 1, 2, \ldots, K$), get $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$

y=1 y=2 .... θ ← K

Pick class i with largest $(\theta^{(i)})^T x$

Logistic regression vs. SVMs

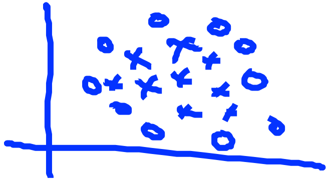
$n =$ number of features ($x \in \mathbb{R}^{n+1}$), $m =$ number of training examples

If n is large (relative to m): (e.g. $n \geq m$, $n = 10{,}000$, $m = 10 \cdots 1000$)

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

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

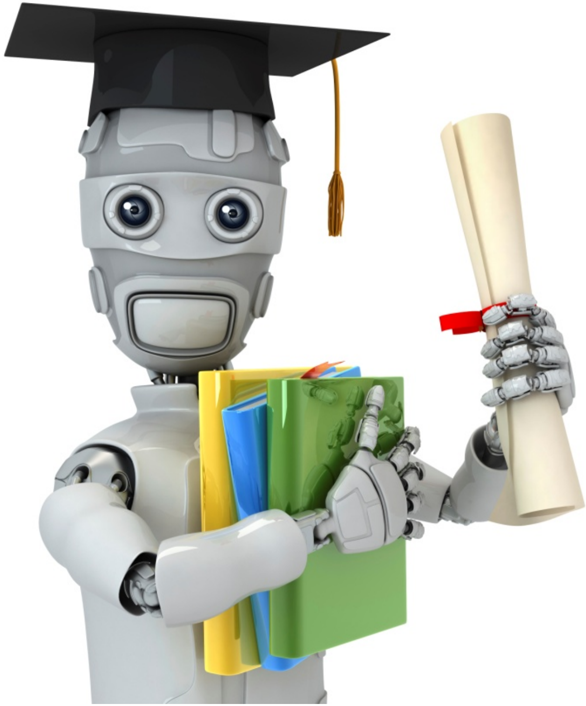
Use SVM with Gaussian kernel



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

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.

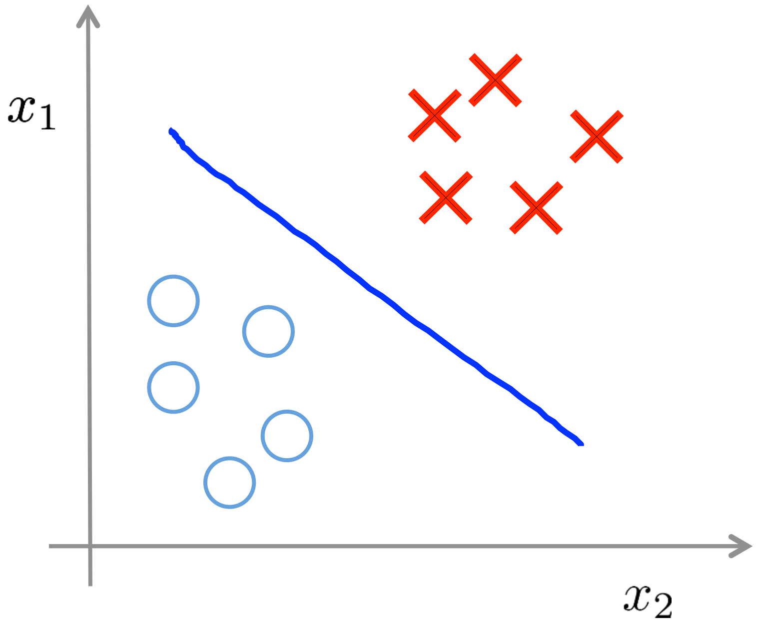


Machine Learning

Clustering

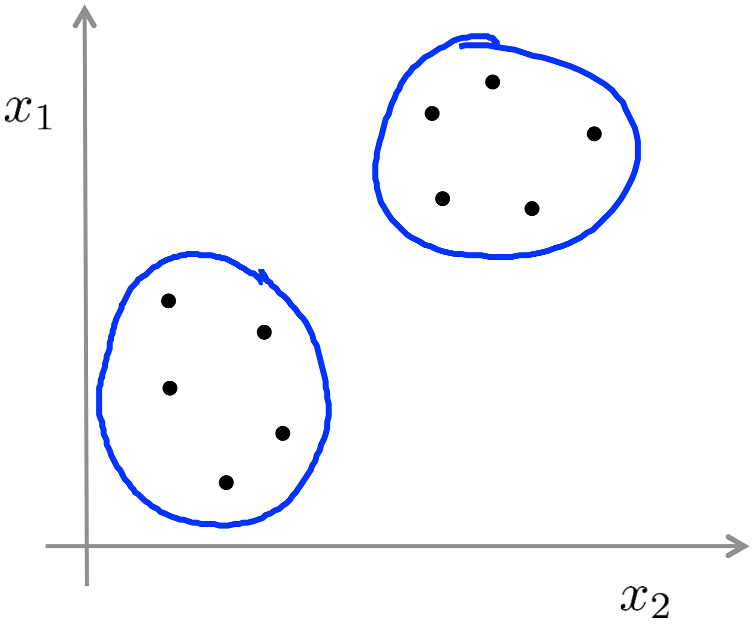
Unsupervised learning introduction

Supervised learning



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

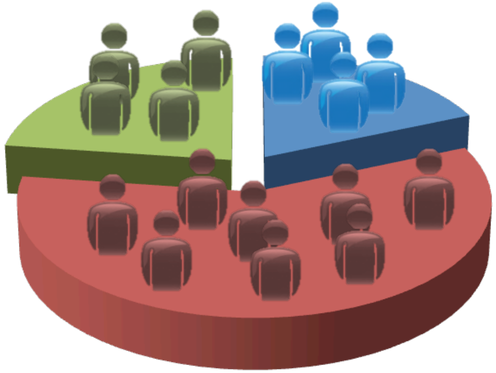
Unsupervised learning



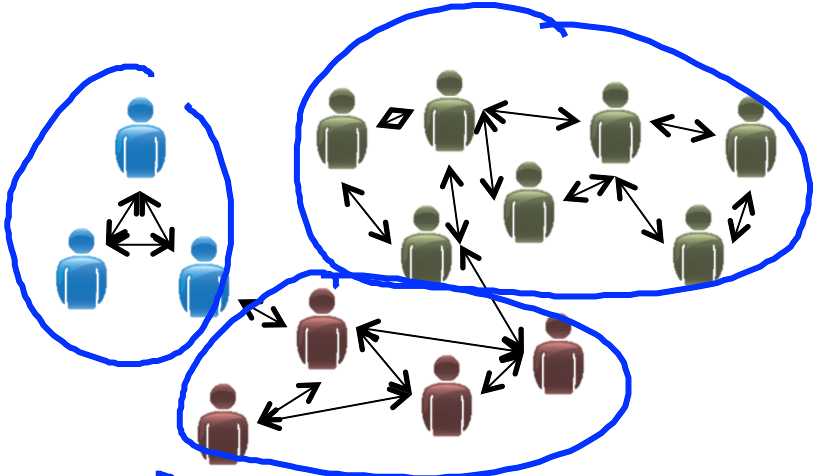
Clustering algorithm

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

Applications of clustering



Market segmentation

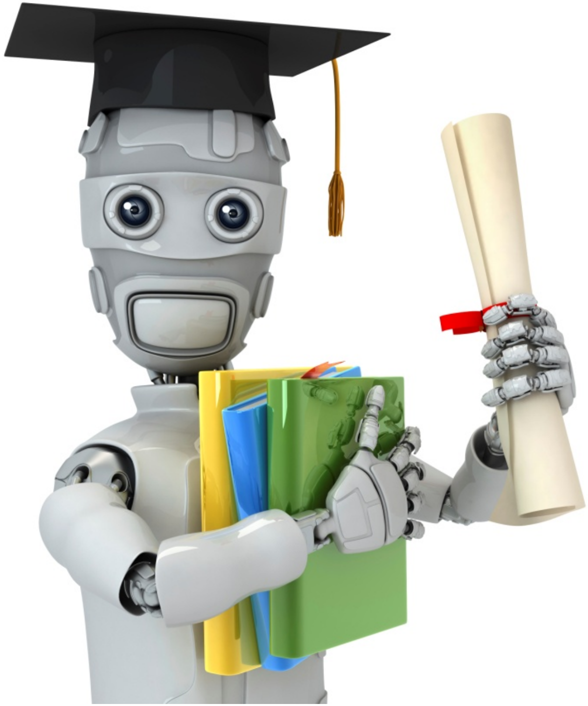


Social network analysis

Organize computing clusters

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

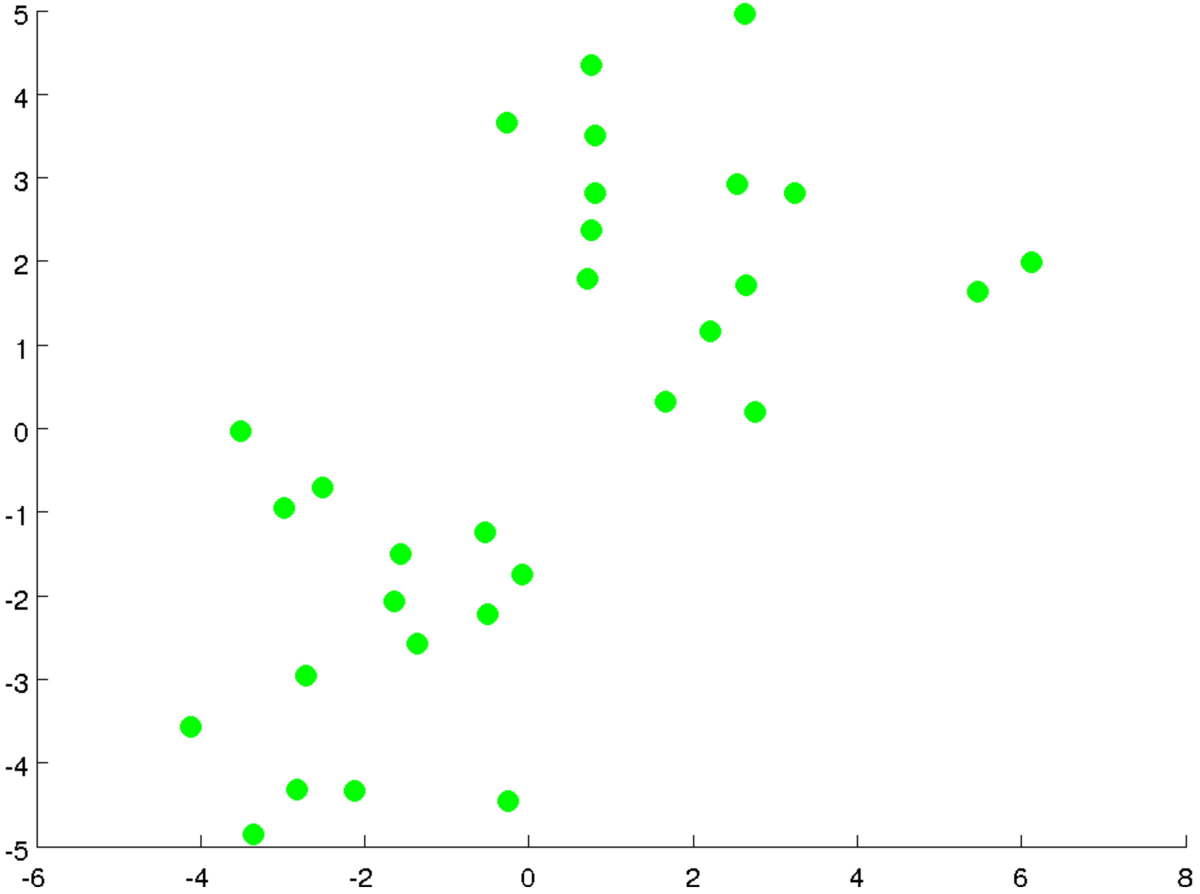
Astronomical data analysis



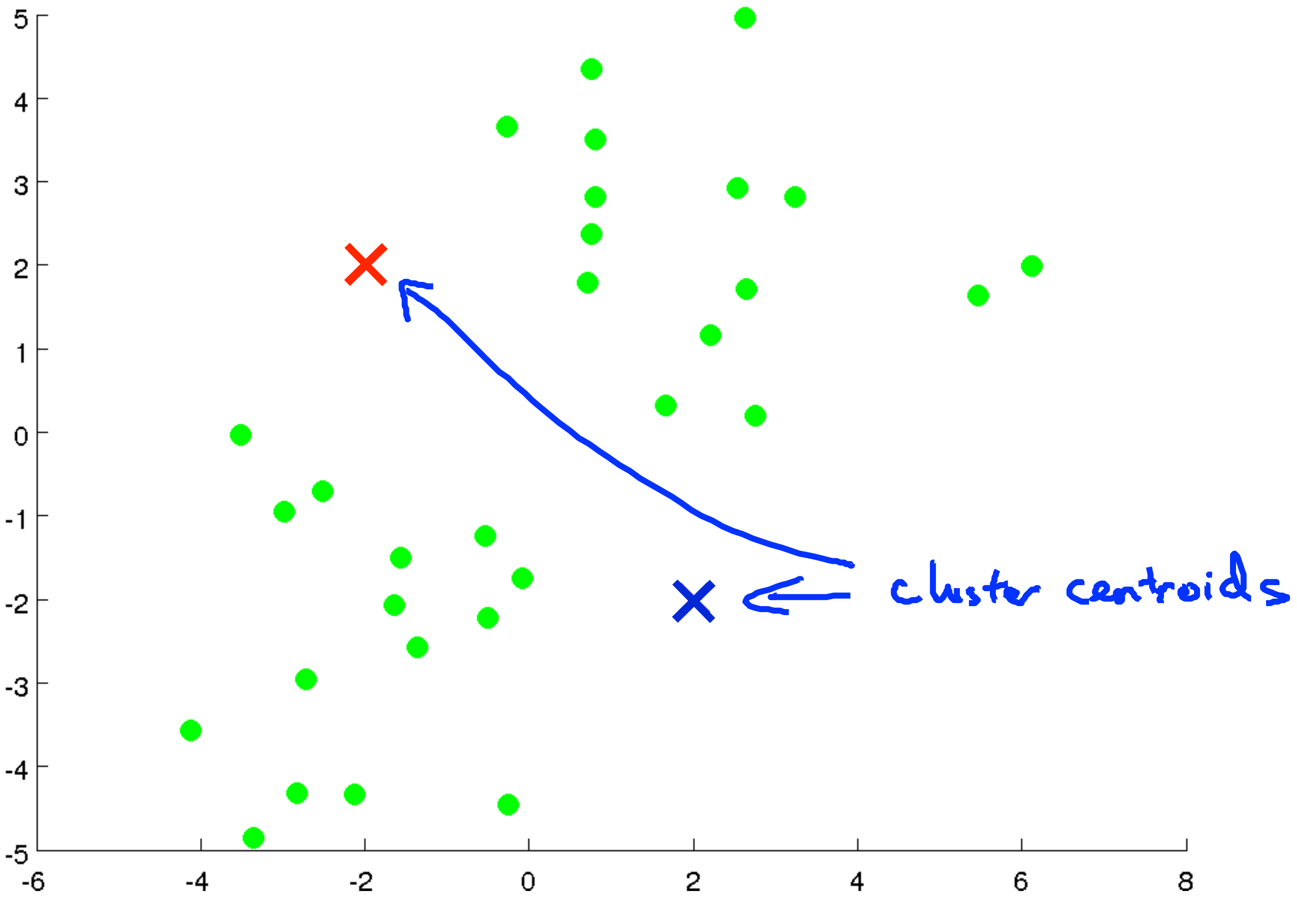
Machine Learning

Clustering

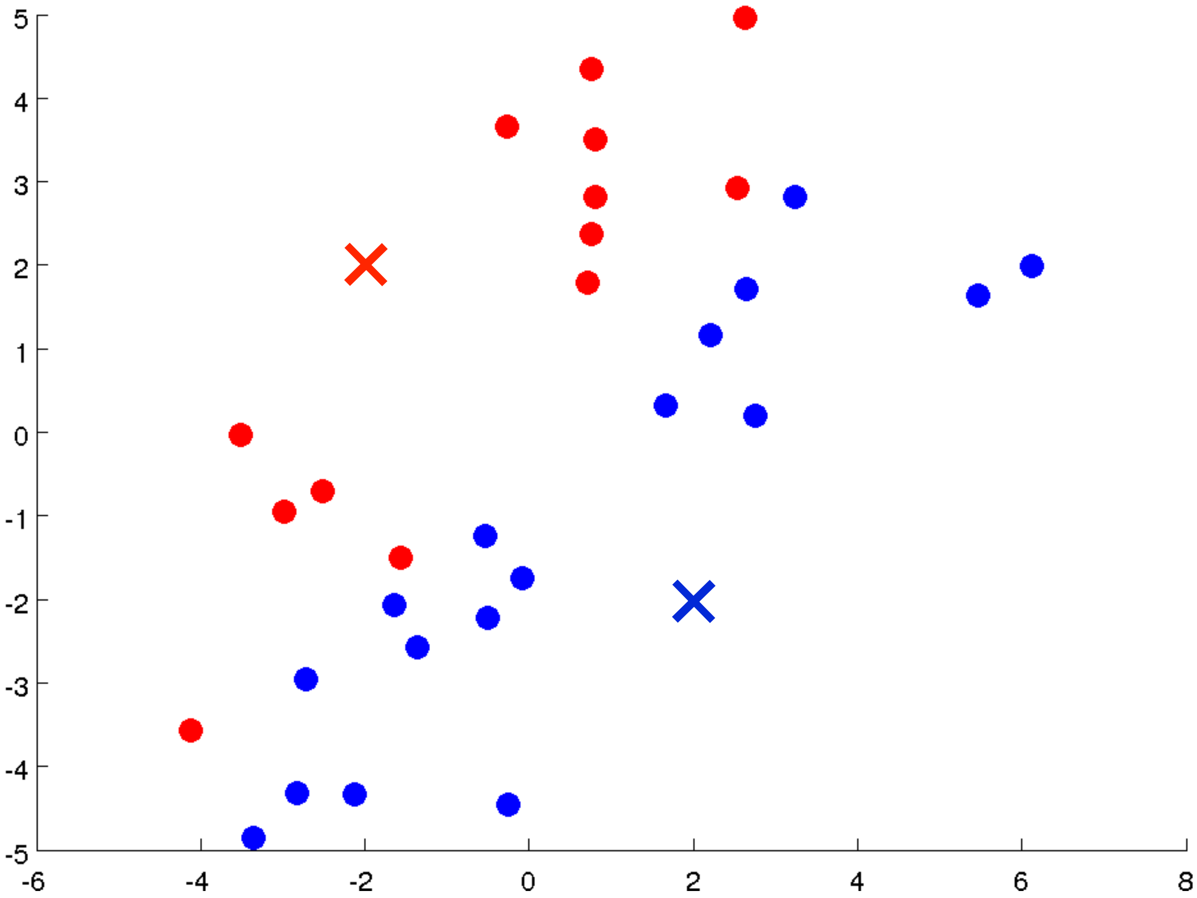
K-means algorithm



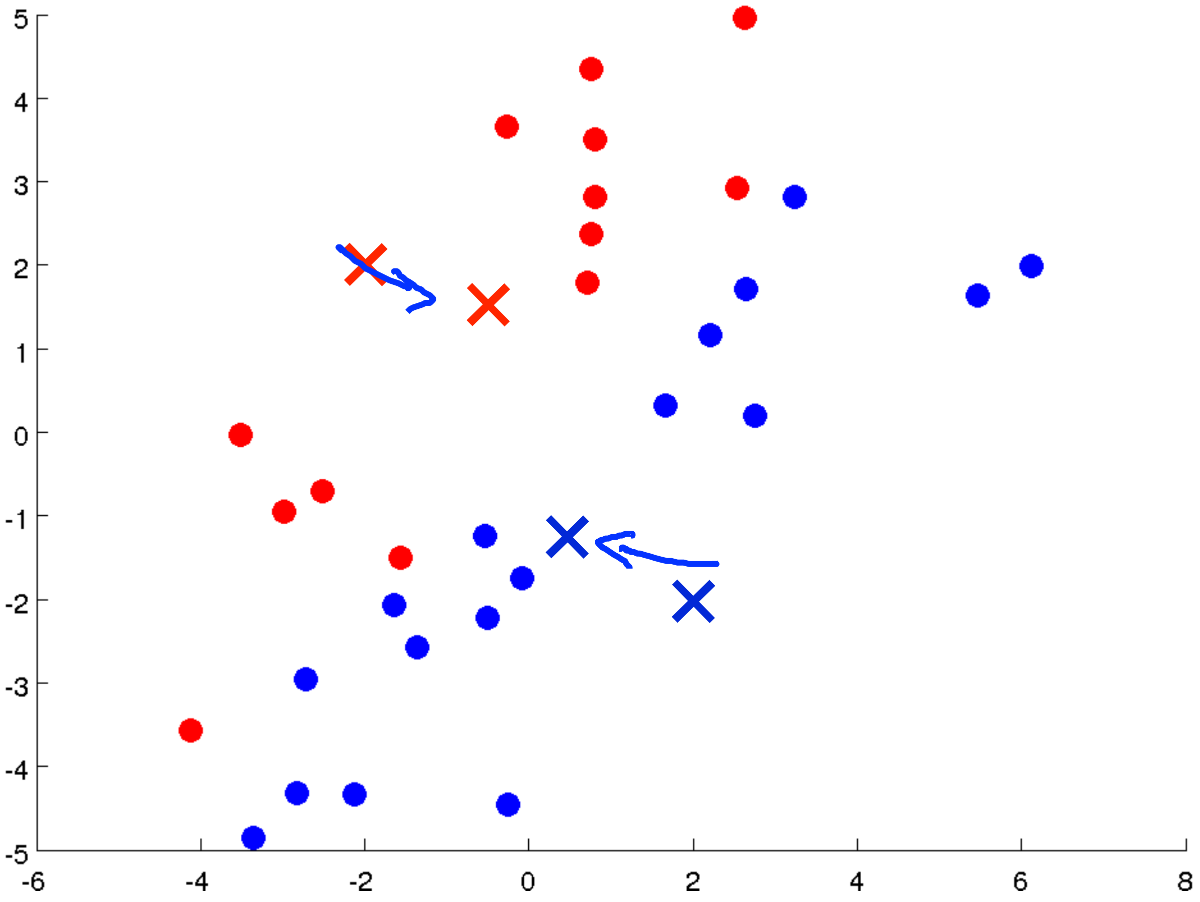
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8



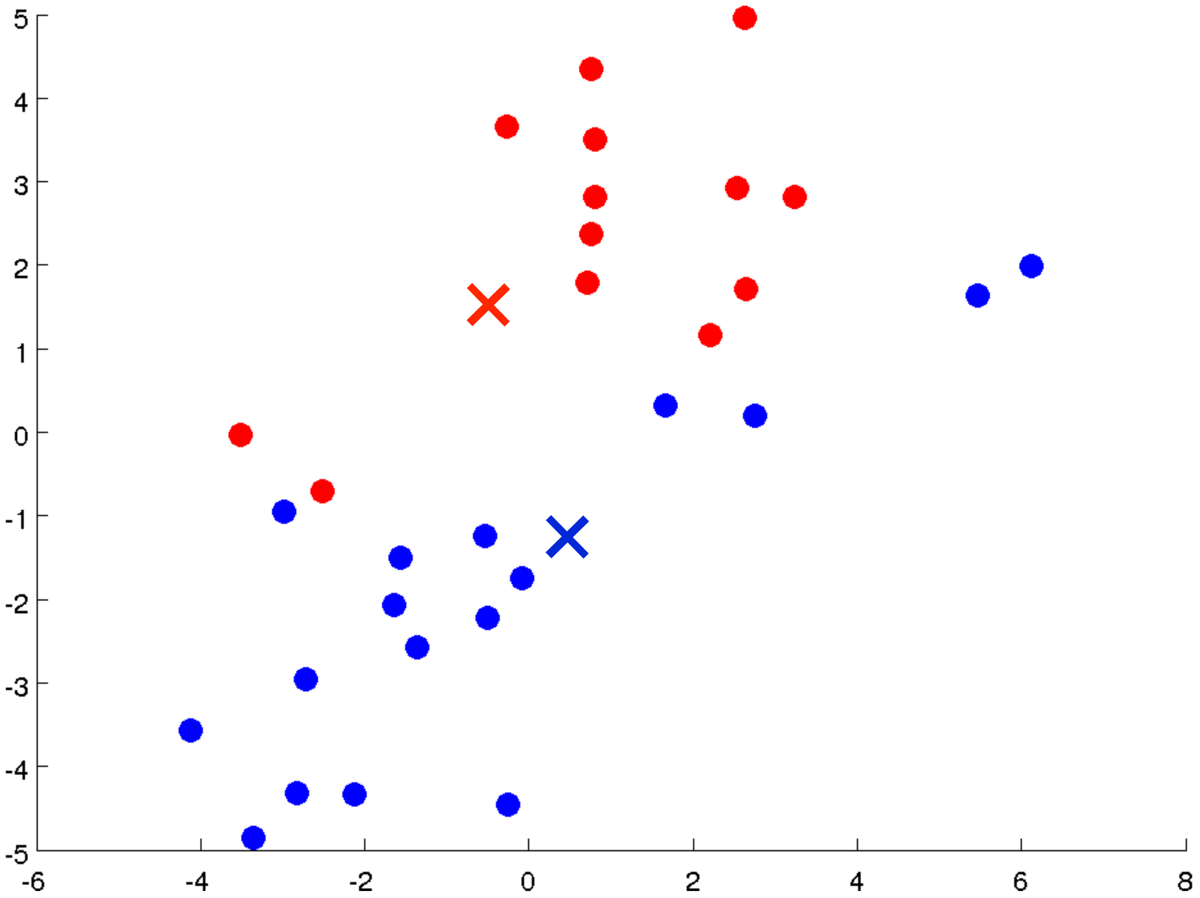
cluster centroids
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8



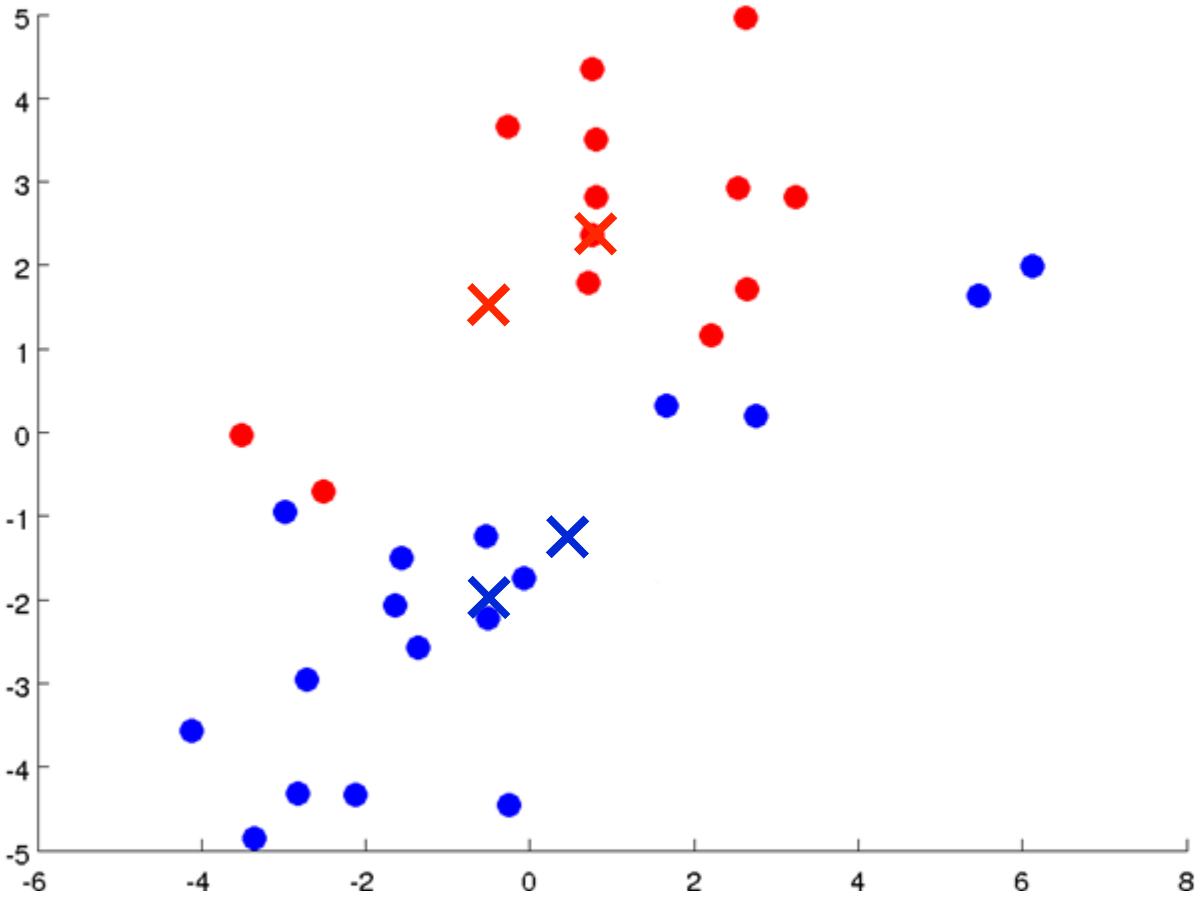
Andrew Ng



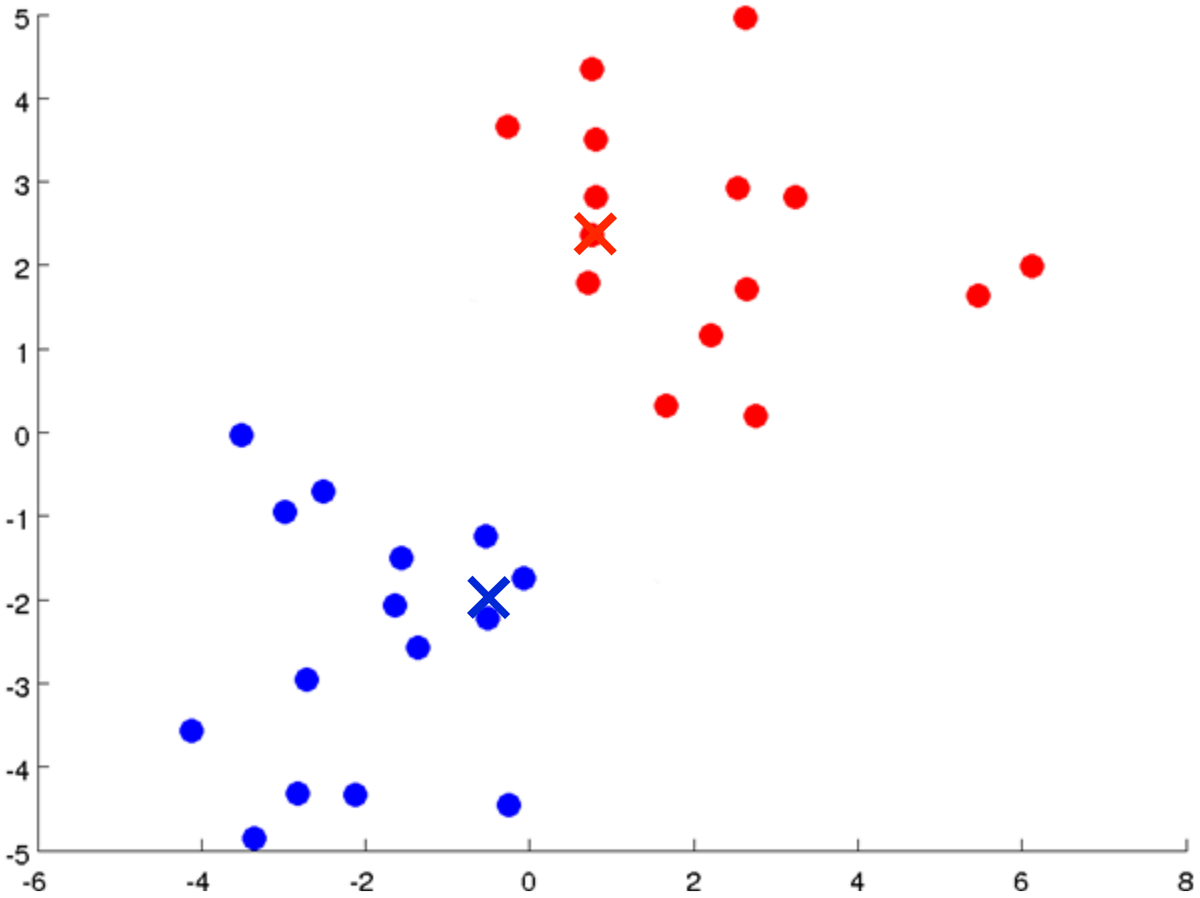
Andrew Ng



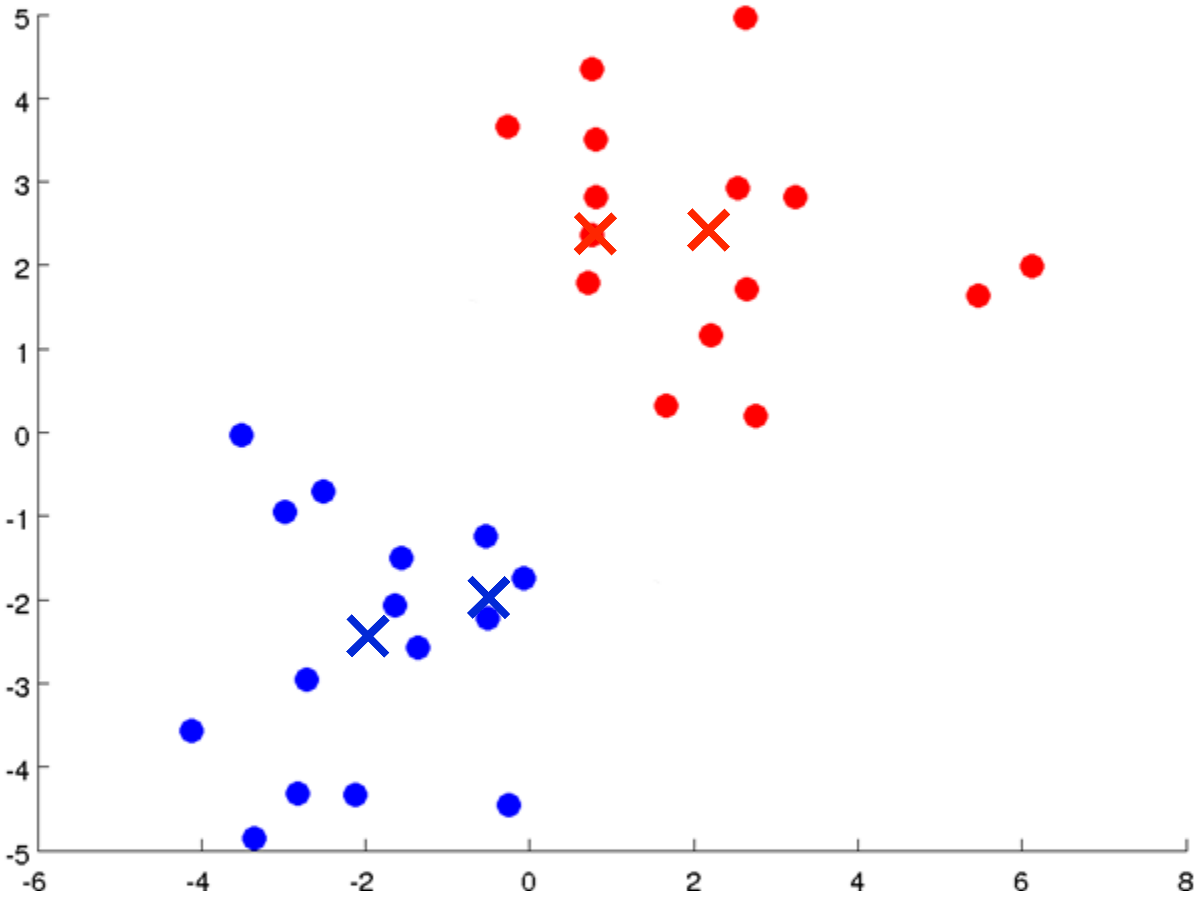
Andrew Ng



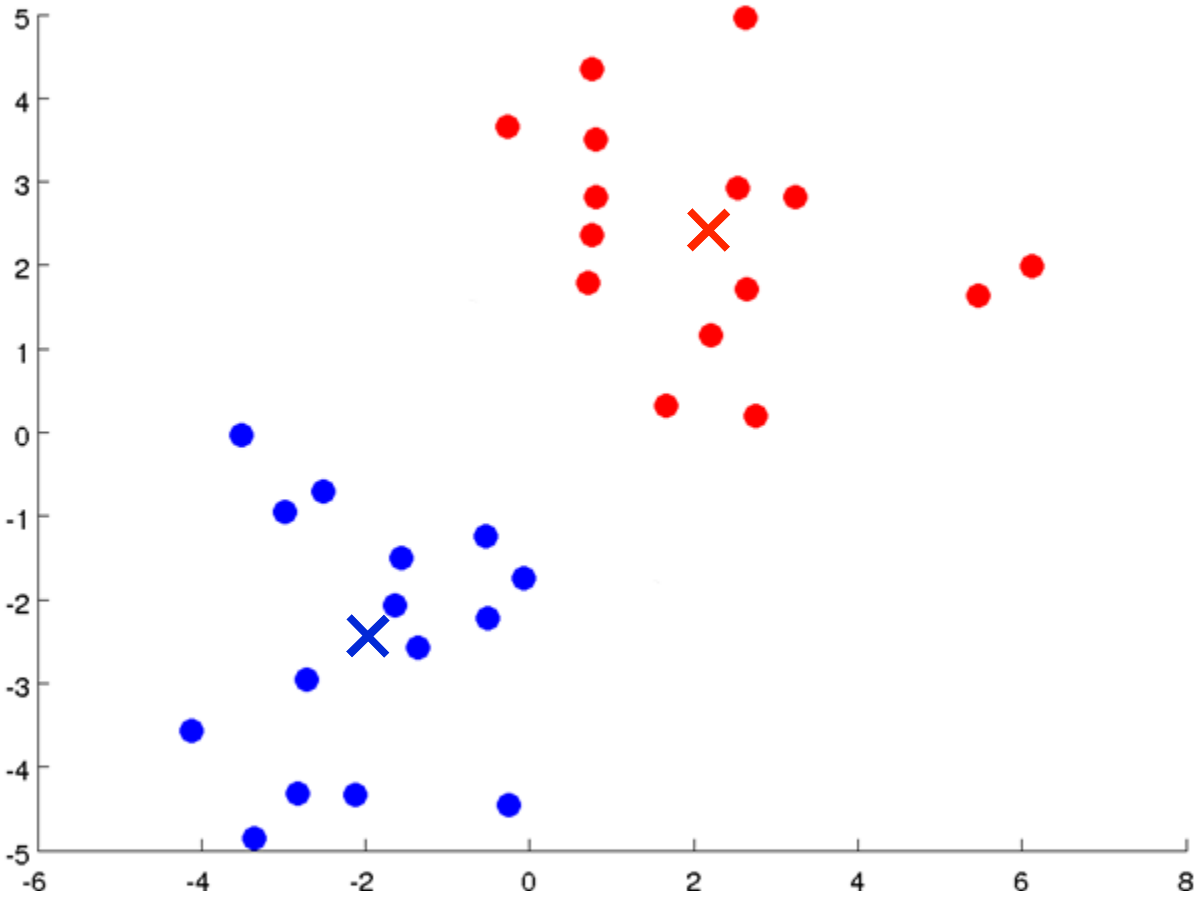
Andrew Ng



Andrew Ng



Andrew Ng



Andrew Ng

K-means algorithm

Input:

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

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

K-means algorithm

μ_1 × μ_2 ×

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

Repeat {

Cluster assignment step

for i = 1 to m

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

$\min_k \|x^{(i)} - \mu_k\|^2$ → $c^{(i)}$

Move centroid

for k = 1 to K

→ μ_k := average (mean) of points assigned to cluster k

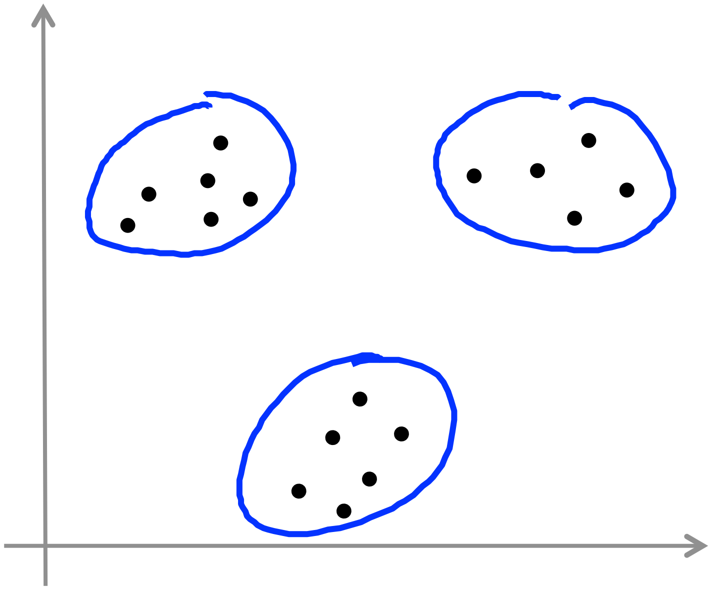
$x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$ → $c^{(1)} = 2, c^{(5)} = 2, c^{(6)} = 2, c^{(10)} = 2$

$$\mu_2 = \frac{1}{4}\left[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}\right] \in \mathbb{R}^n$$

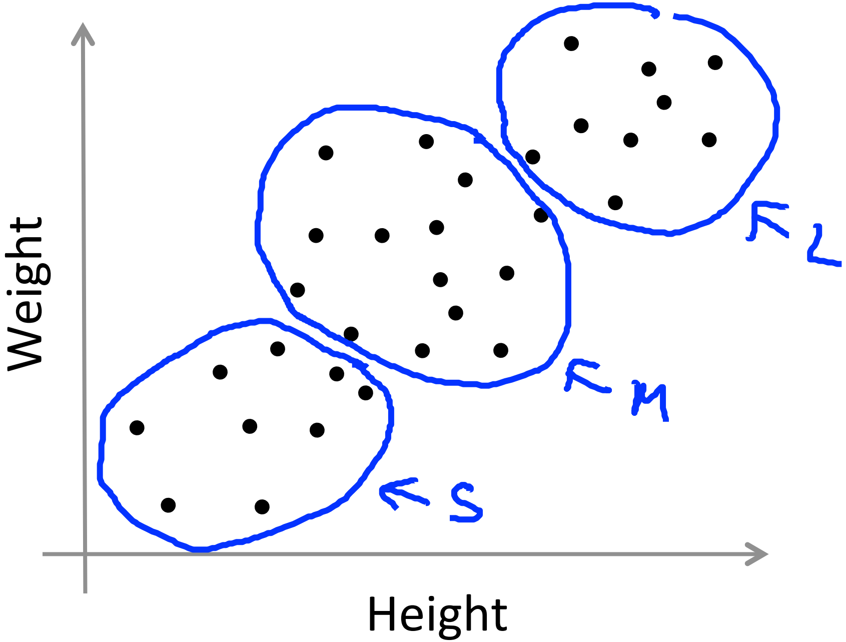
}

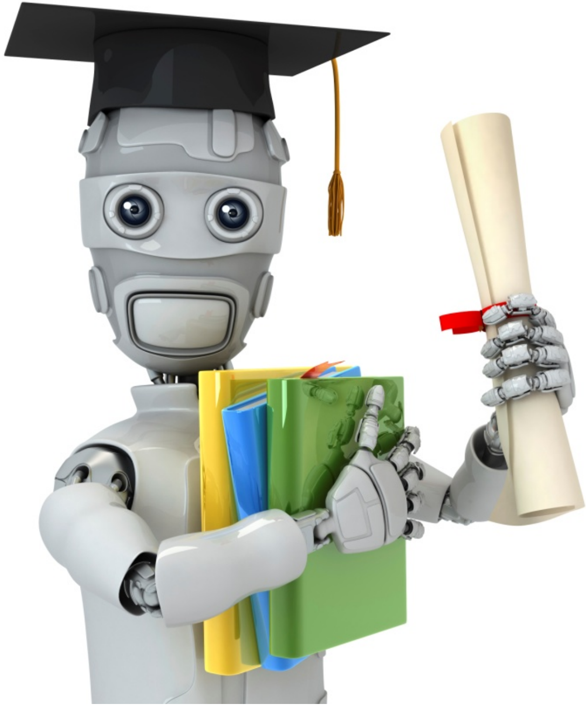
K-means for non-separated clusters

S, M, L



T-shirt sizing





Machine Learning

Clustering

Optimization objective

K-means optimization objective

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

→ μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$) K $k \in \{1,2,\ldots,K\}$

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

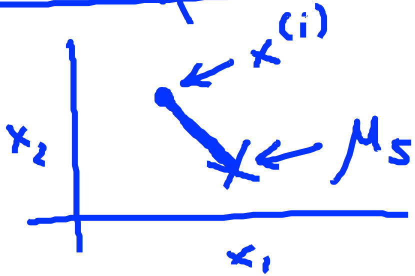
$x^{(i)} \rightarrow 5$ $c^{(i)} = 5$ $\mu_{c^{(i)}} = \mu_5$

Optimization objective:

$$\rightarrow \quad J(c^{(1)},\ldots,c^{(m)},\mu_1,\ldots,\mu_K) = \frac{1}{m}\sum_{i=1}^{m} \boxed{||x^{(i)} - \mu_{c^{(i)}}||^2} \leftarrow$$

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

Distortion



K-means algorithm

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

Cluster assignment step

Minimize $J(\ldots)$ wrt $\boxed{c^{(1)}, c^{(2)}, \ldots, c^{(m)}}$

(holding $\mu_1, \ldots, \mu_K$ fixed)

Repeat {

for i = 1 to m

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

move centroid

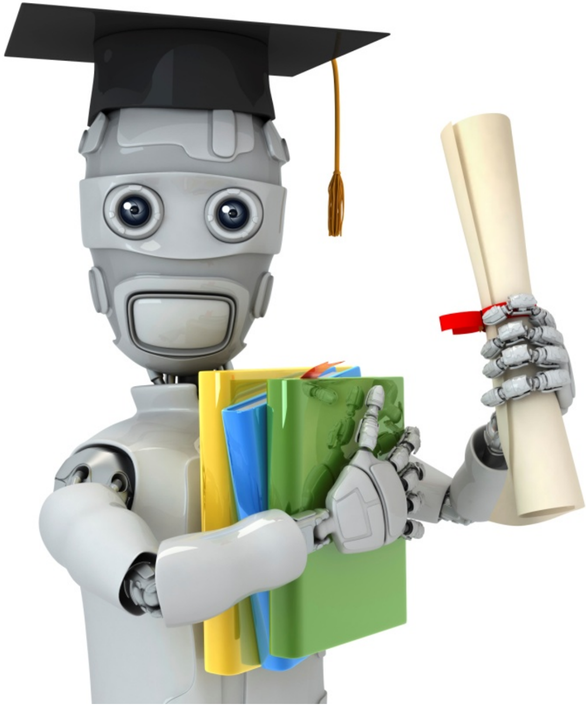
for k = 1 to K

μ_k := average (mean) of points assigned to cluster k

}

minimize $J(\ldots)$ wrt $\boxed{\mu_1, \ldots, \mu_K}$

Andrew Ng



Machine Learning

Clustering

Random initialization

K-means algorithm

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

Repeat {

for i = 1 to m

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

for k = 1 to K

μ_k := average (mean) of points assigned to cluster k

}

Random initialization

$K=2$

Should have $K < m$

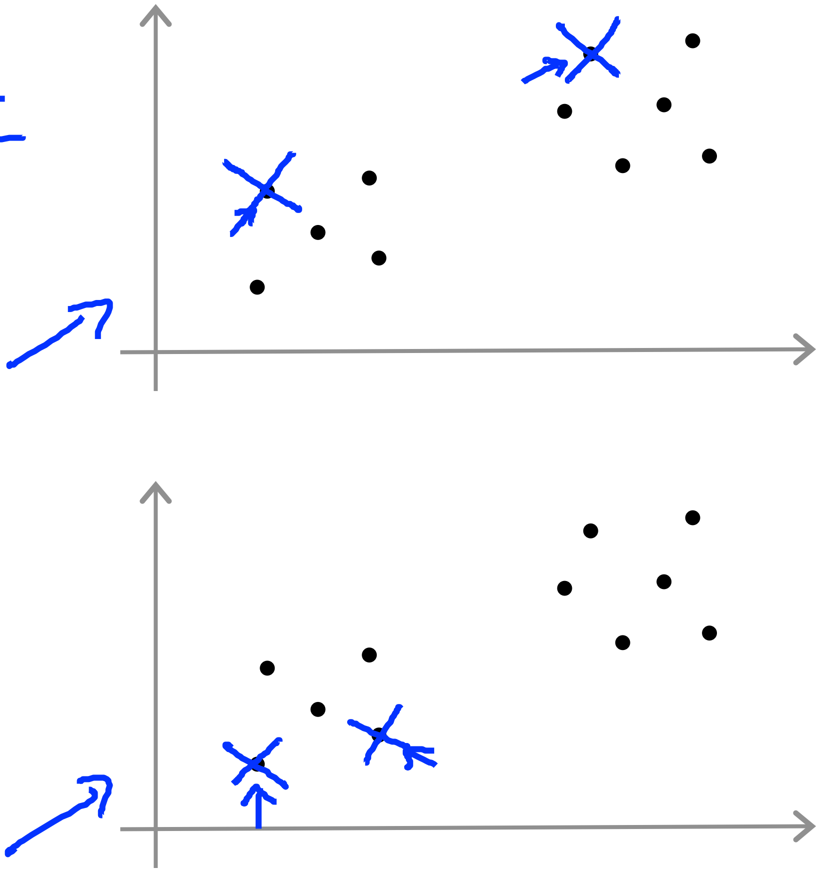
Randomly pick K training examples.

Set $\mu_1, \dots, \mu_K$ equal to these K examples.

$\mu_1 = x^{(i)}$

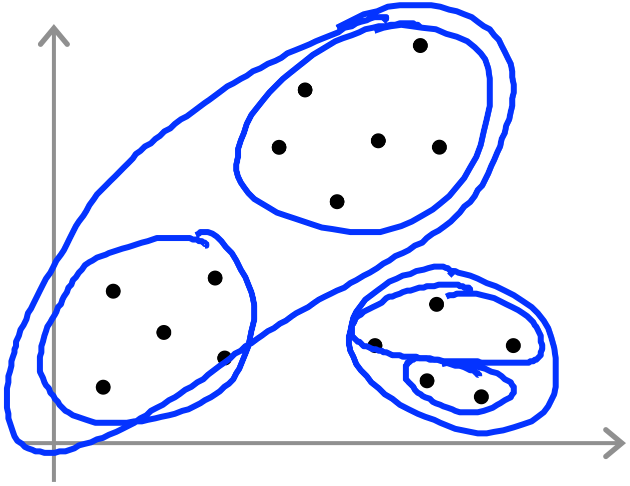
$\mu_2 = x^{(j)}$

$\vdots$

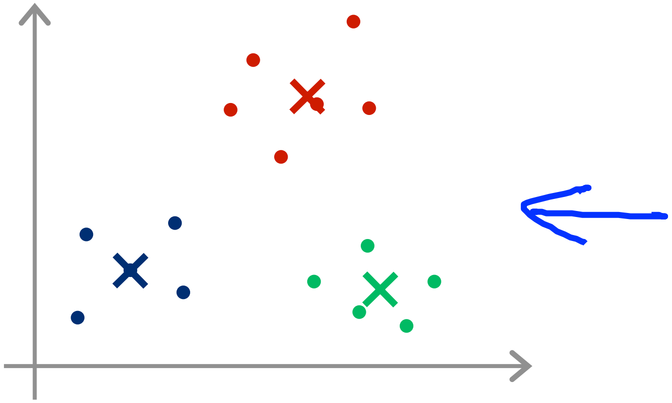


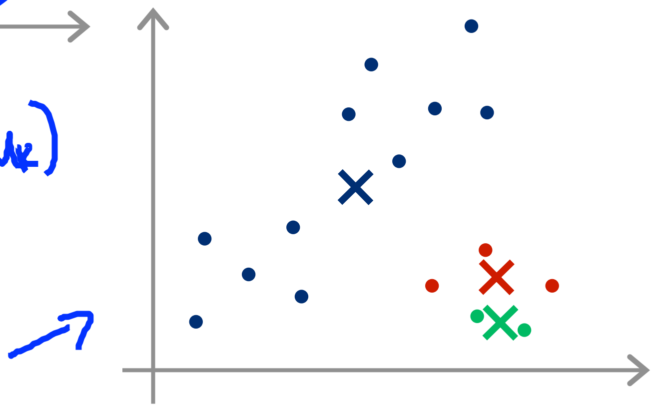
Andrew Ng

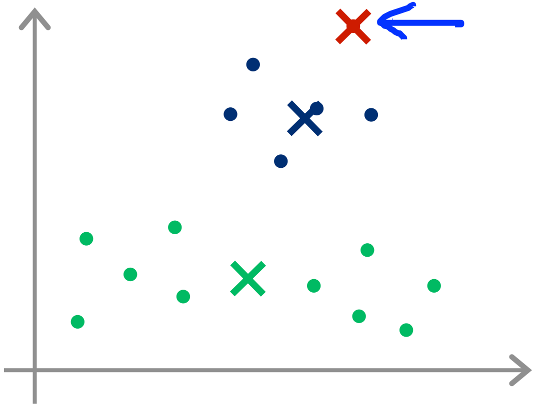
Local optima



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







Random initialization

For i = 1 to 100 {

Randomly initialize K-means.

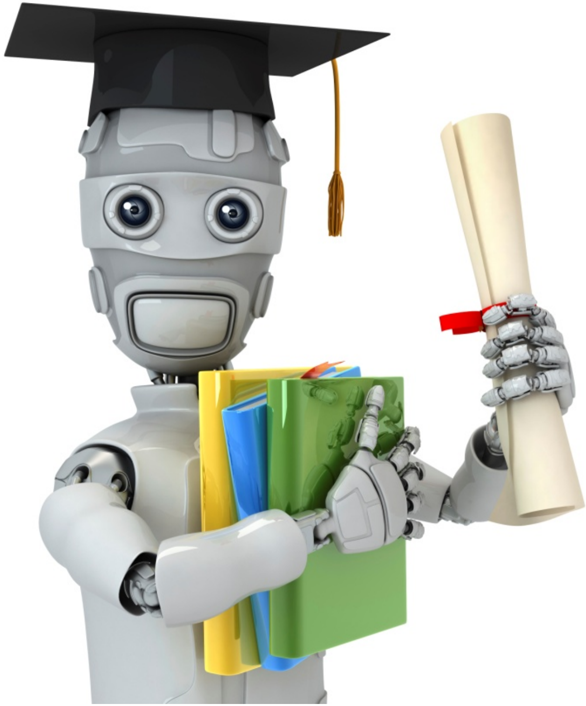
Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.

Compute cost function (distortion)

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

}

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

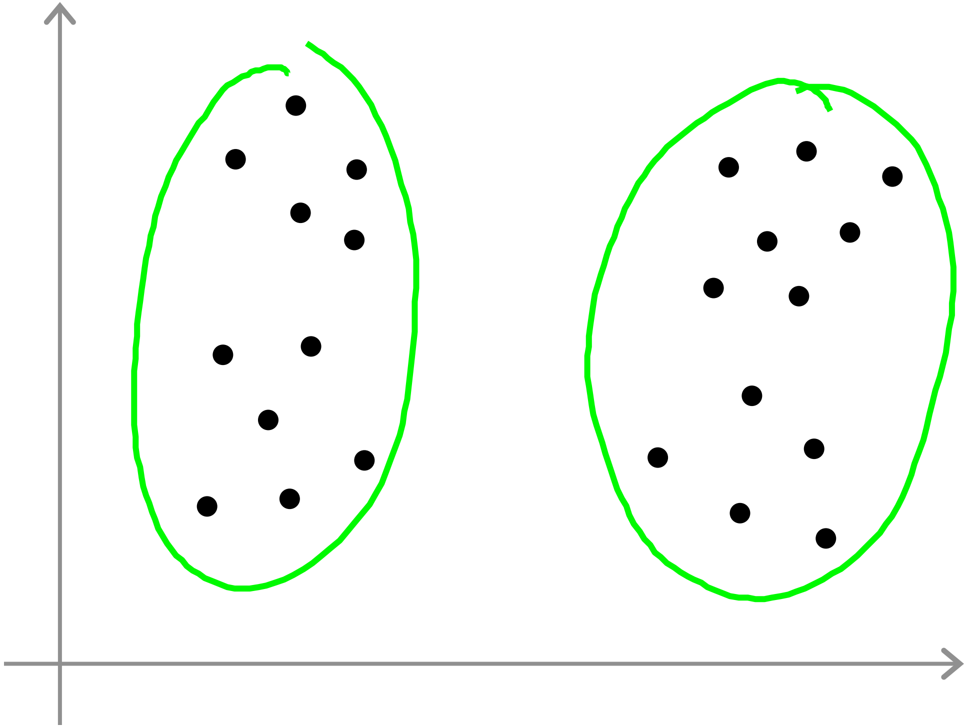


Machine Learning

Clustering

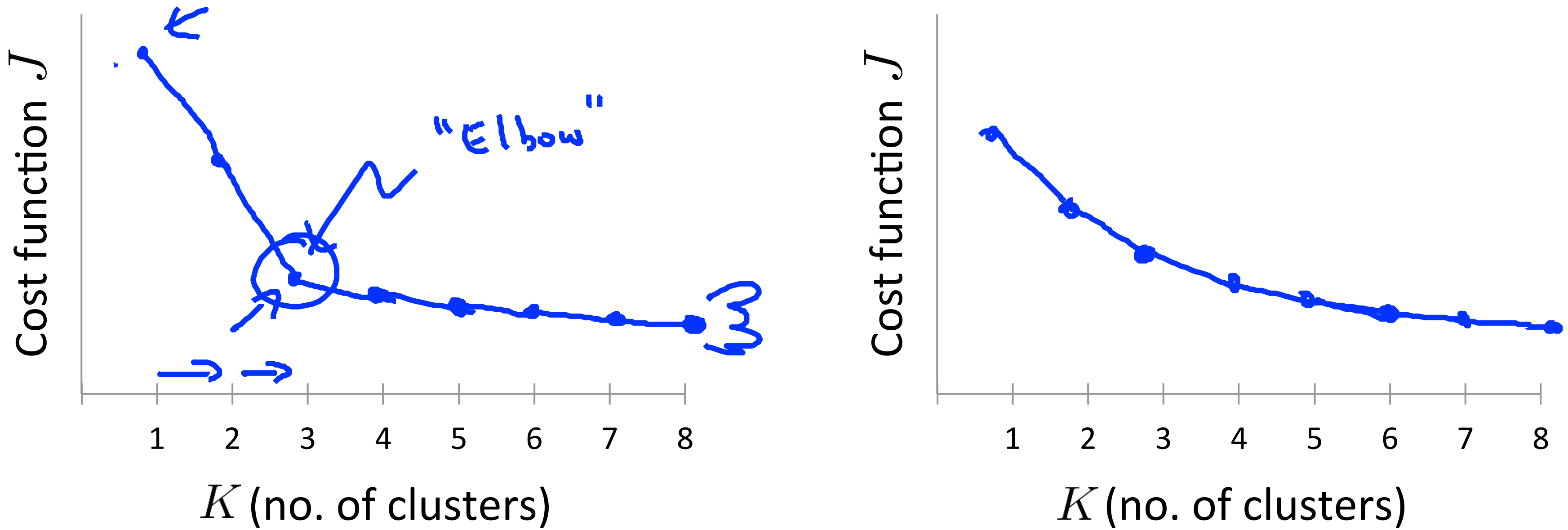
Choosing the number of clusters

What is the right value of K?



Choosing the value of K

Elbow method:



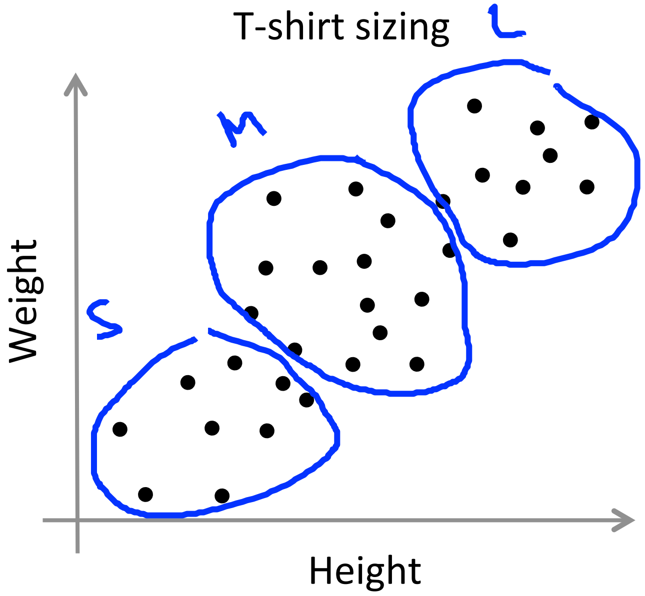
Choosing the value of K

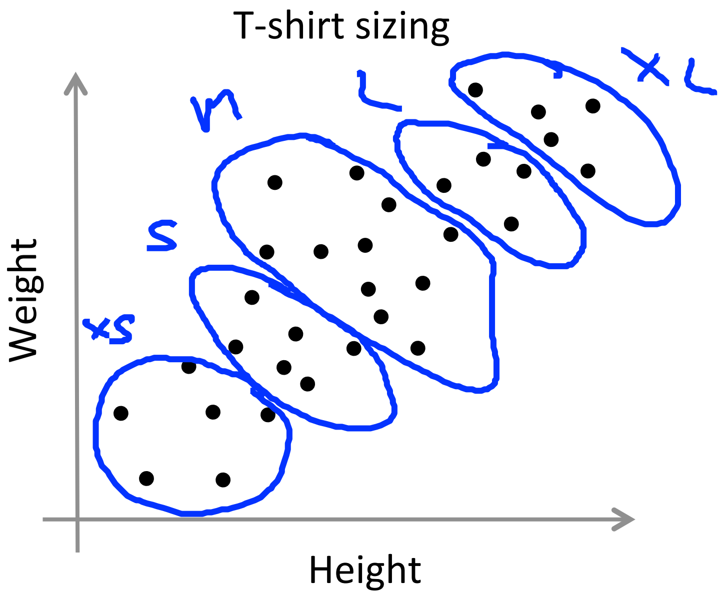
Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

K=3 S, M, L

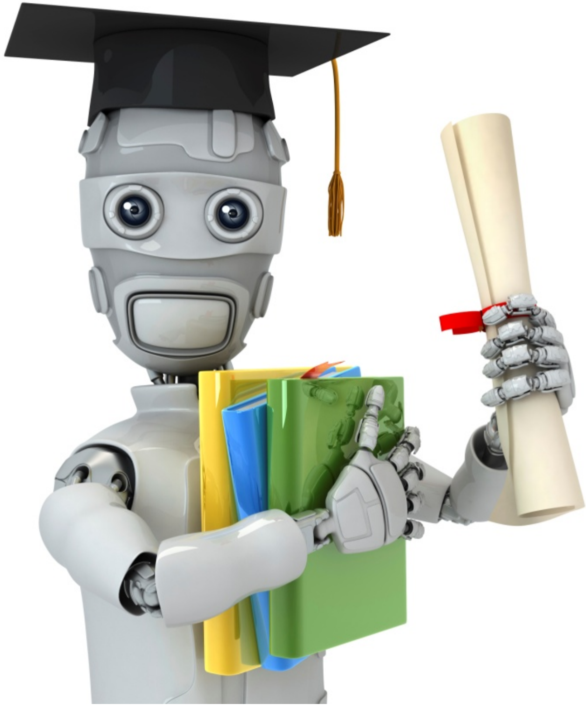
K=5 XS, S, M, L, XL

E.g.





Andrew Ng

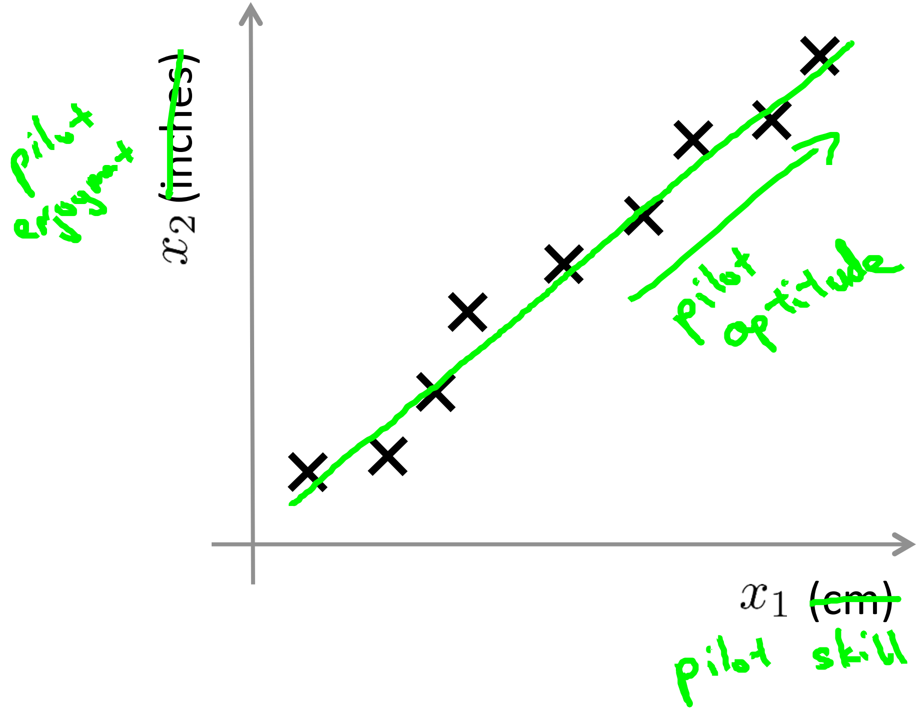


Machine Learning

Dimensionality Reduction

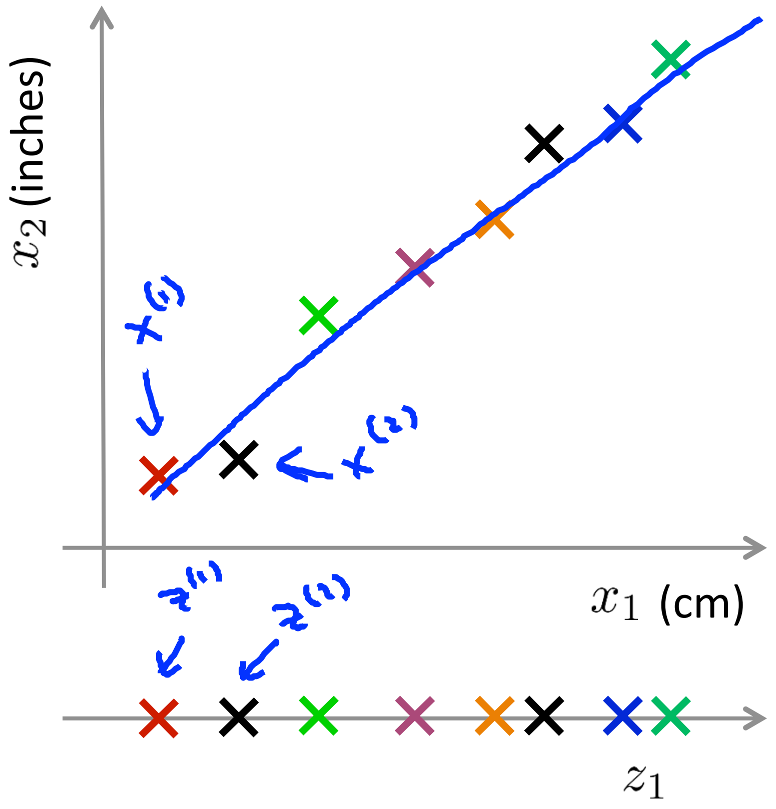
Motivation I: Data Compression

Data Compression



Reduce data from 2D to 1D

Data Compression



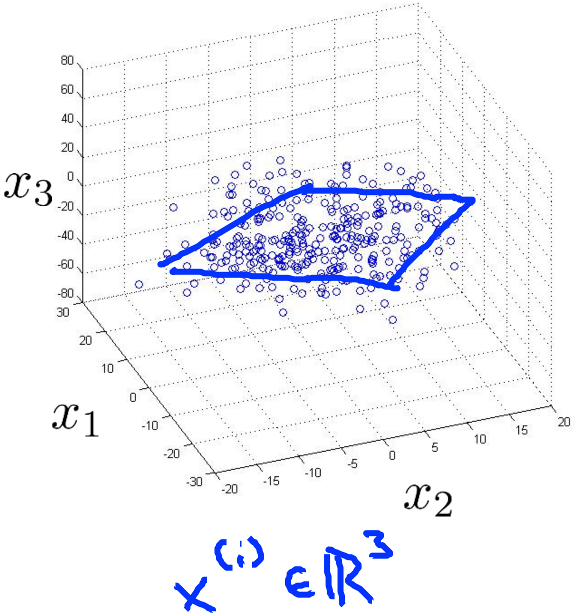
Reduce data from 2D to 1D

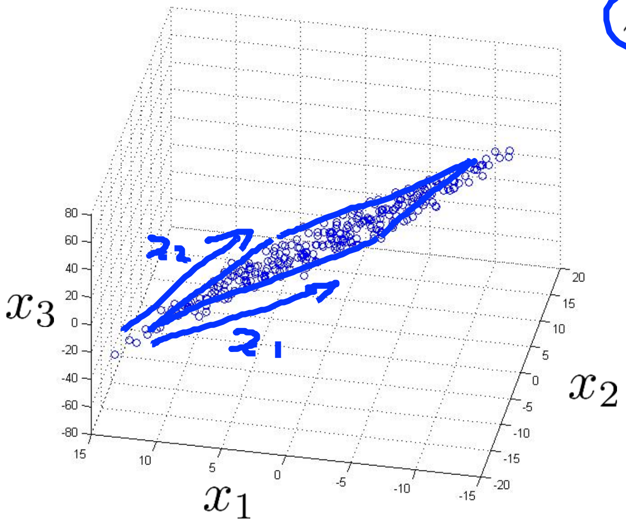
$$x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$$
$$x^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$$
$$\vdots$$
$$x^{(m)} \in \mathbb{R}^2 \rightarrow z^{(m)} \in \mathbb{R}$$

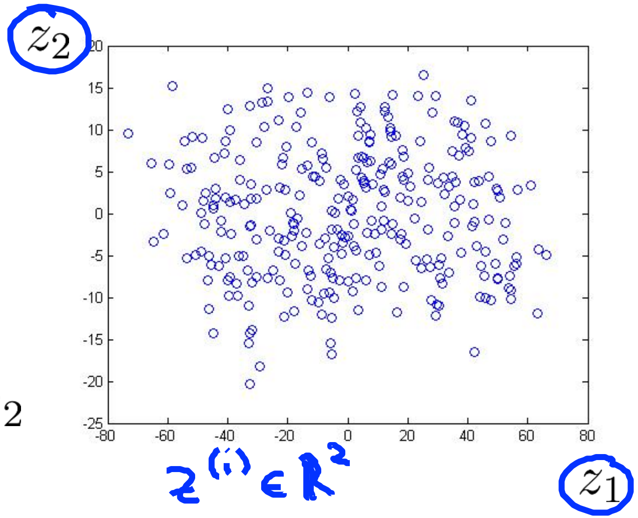
Data Compression

1000D → 100D

Reduce data from 3D to 2D

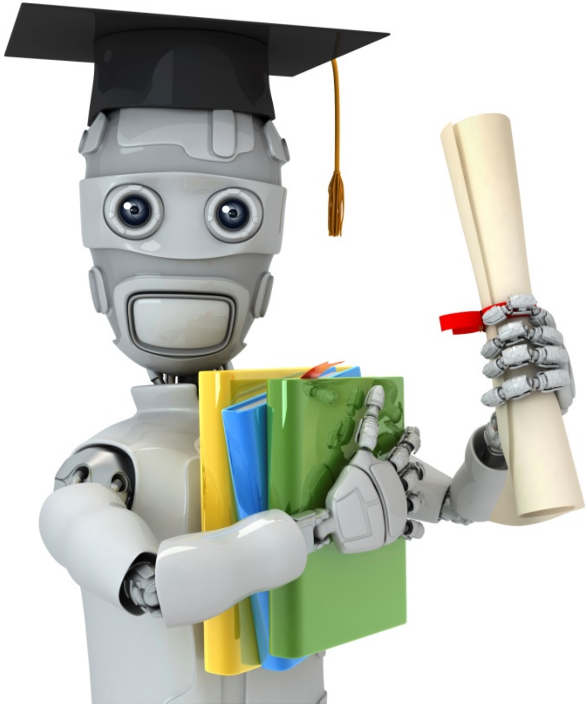






$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \qquad z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$

Andrew Ng



Machine Learning

Dimensionality Reduction

Motivation II: Data Visualization

Data Visualization

$x \in \mathbb{R}^{50}$ $x^{(i)} \in \mathbb{R}^{50}$

Country	x_1 GDP (trillions of US$)	x_2 Per capita GDP (thousands of intl. $)	x_3 Human Develop-ment Index	x_4 Life expectancy	x_5 Poverty Index (Gini as percentage)	x_6 Mean household income (thousands of US$)	…
Canada	1.577	39.17	0.908	80.7	32.6	67.293	…
China	5.878	7.54	0.687	73	46.9	10.22	…
India	1.632	3.41	0.547	64.7	36.8	0.735	…
Russia	1.48	19.84	0.755	65.5	39.9	0.72	…
Singapore	0.223	56.69	0.866	80	42.5	67.1	…
USA	14.527	46.86	0.91	78.3	40.8	84.3	…
…	…	…	…	…	…	…	

[resources from en.wikipedia.org]

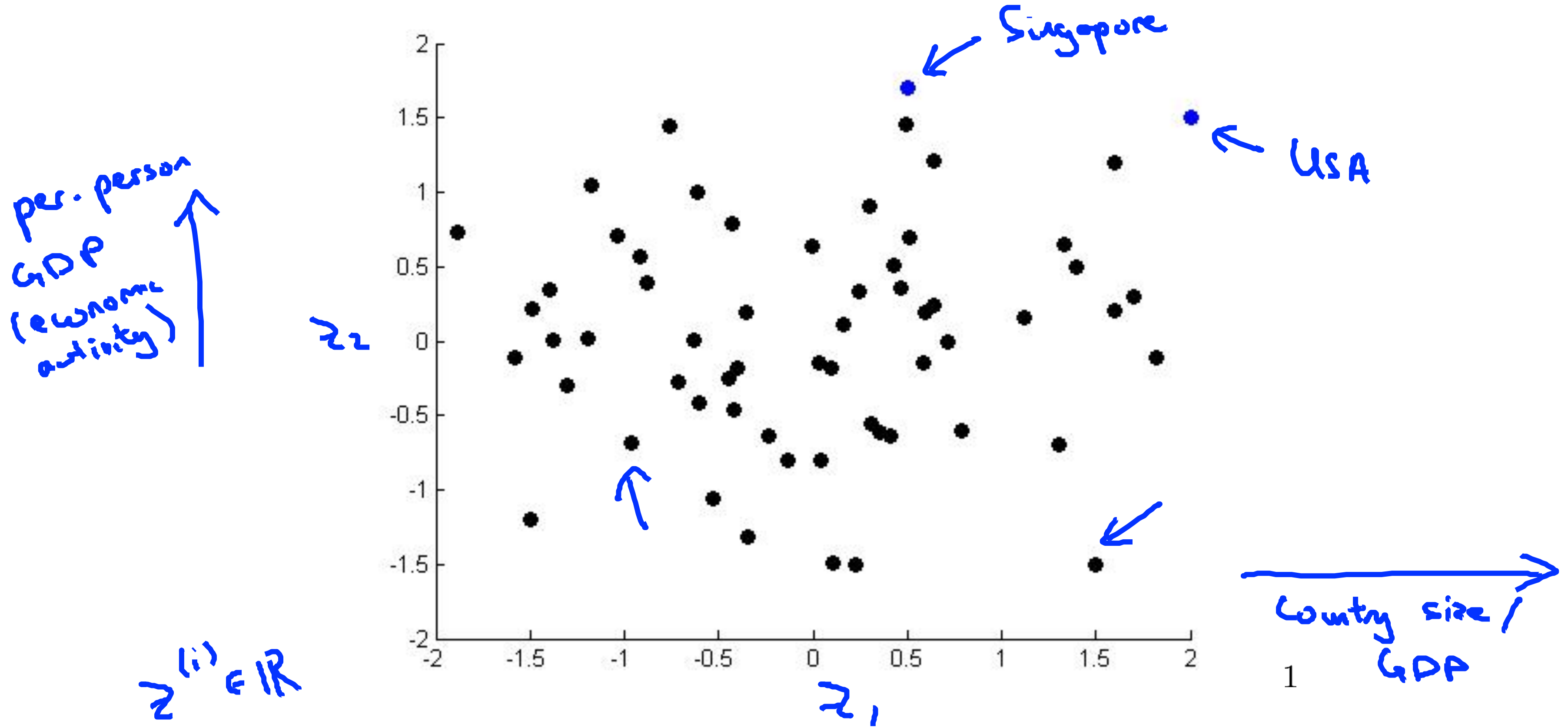
Data Visualization

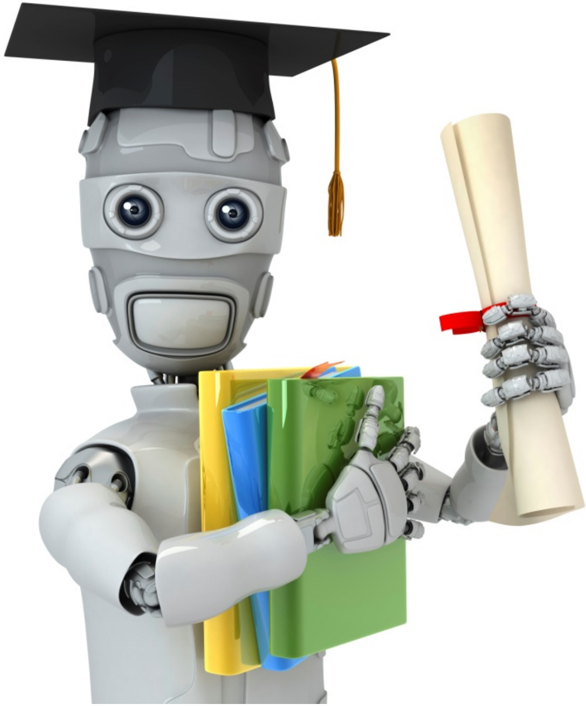
$z^{(i)} \in \mathbb{R}^2$

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

Reduce data from 50D to 2D

Data Visualization



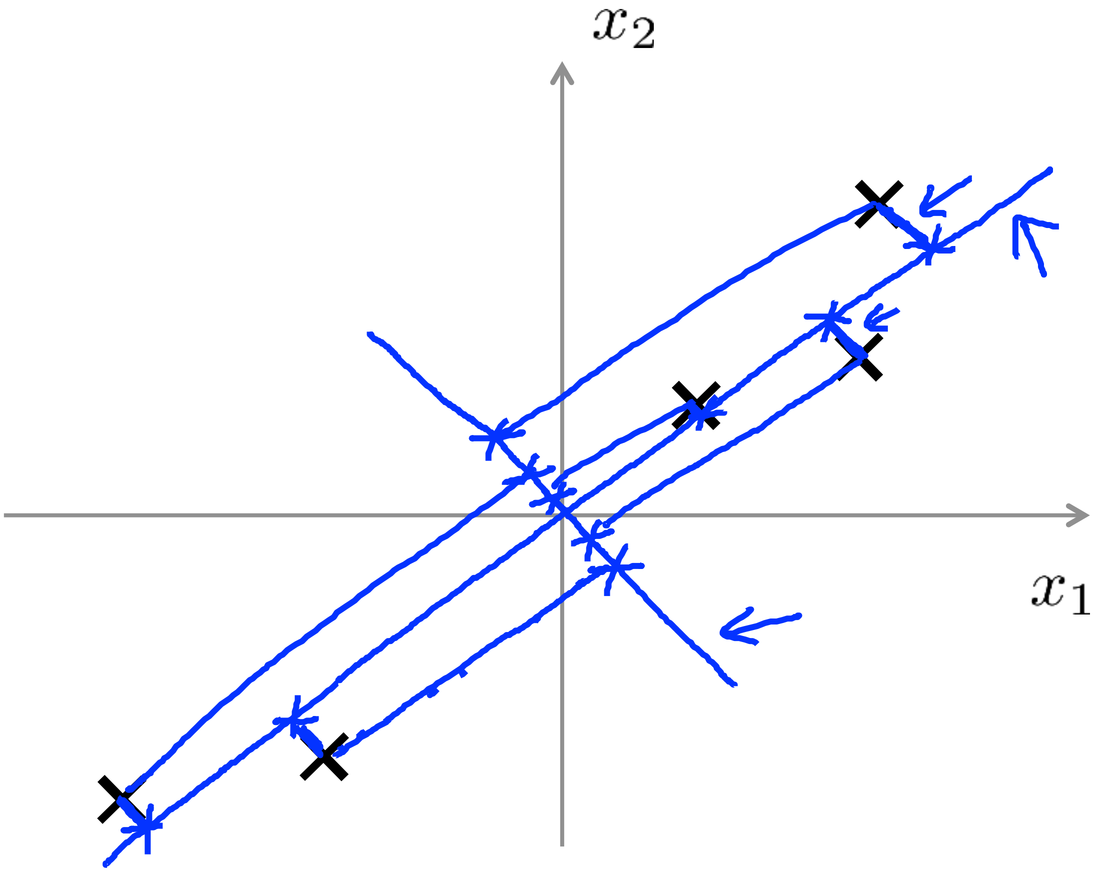


Machine Learning

Dimensionality Reduction

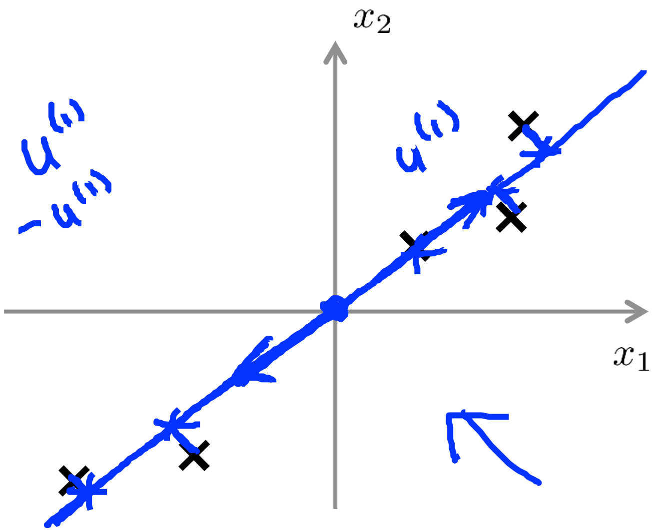
Principal Component Analysis problem formulation

Principal Component Analysis (PCA) problem formulation

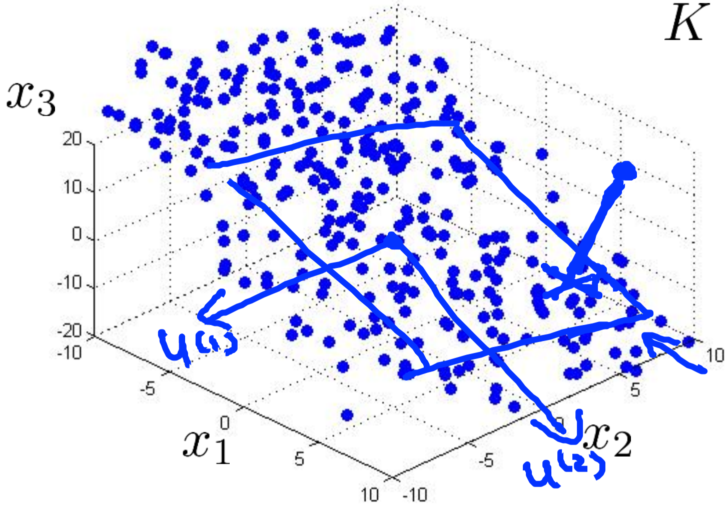


$x \in \mathbb{R}^2$

Principal Component Analysis (PCA) problem formulation



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

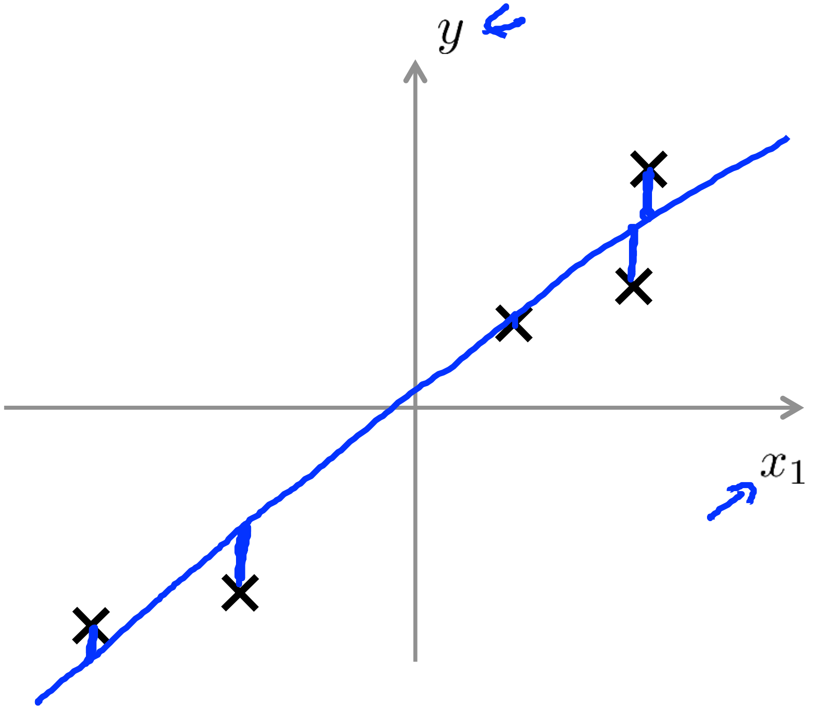


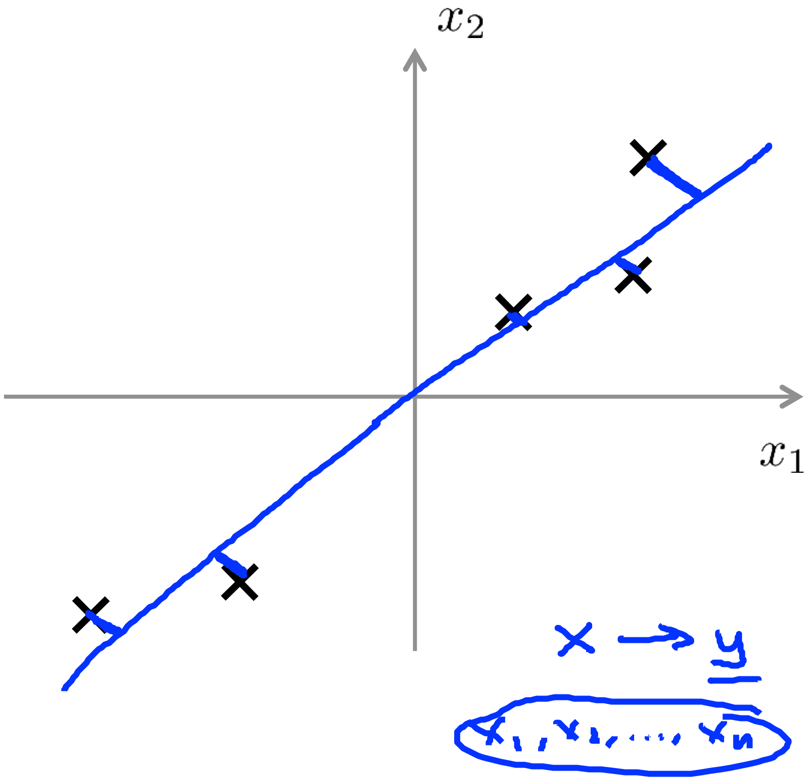
Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

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

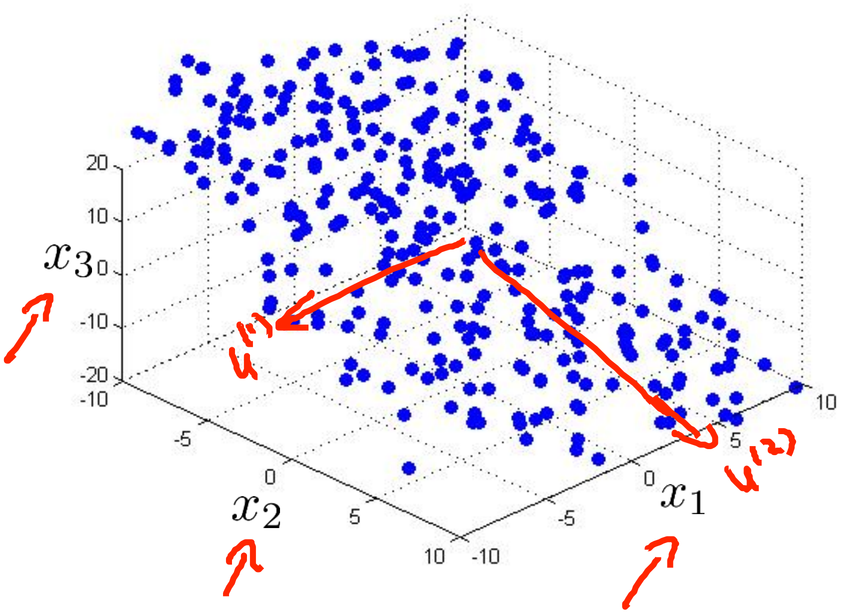
Andrew Ng

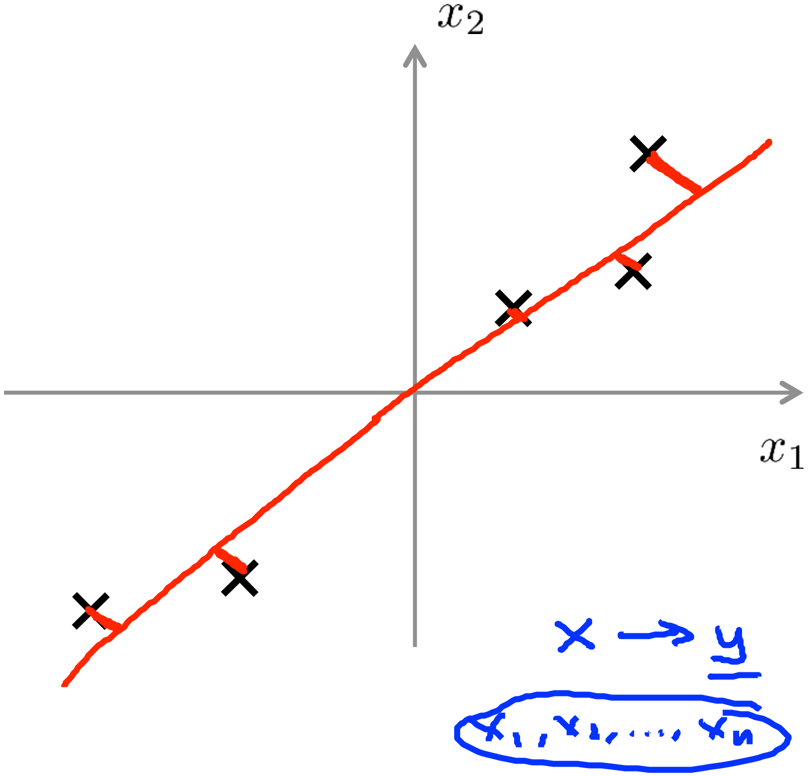
PCA is not linear regression

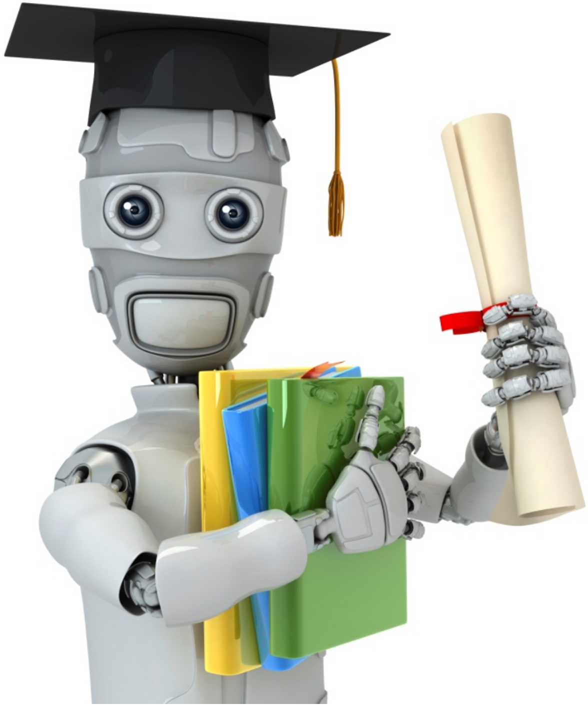




PCA is not linear regression







Machine Learning

Dimensionality Reduction

Principal Component Analysis algorithm

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

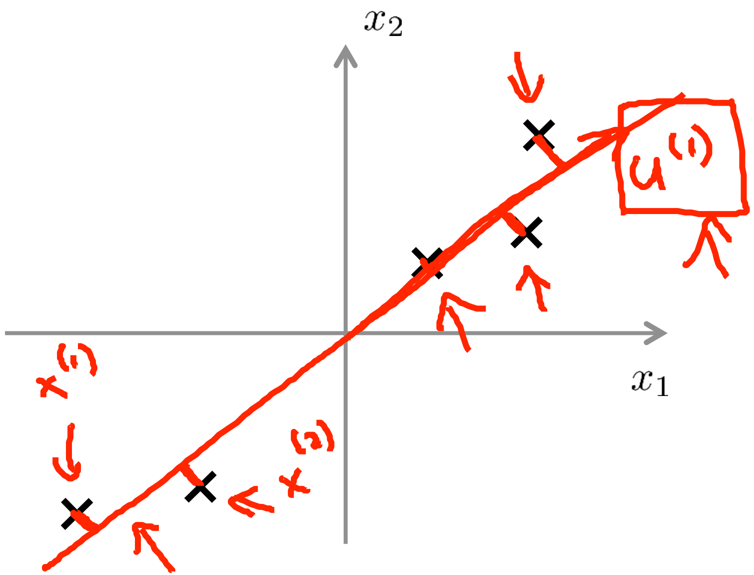
$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

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

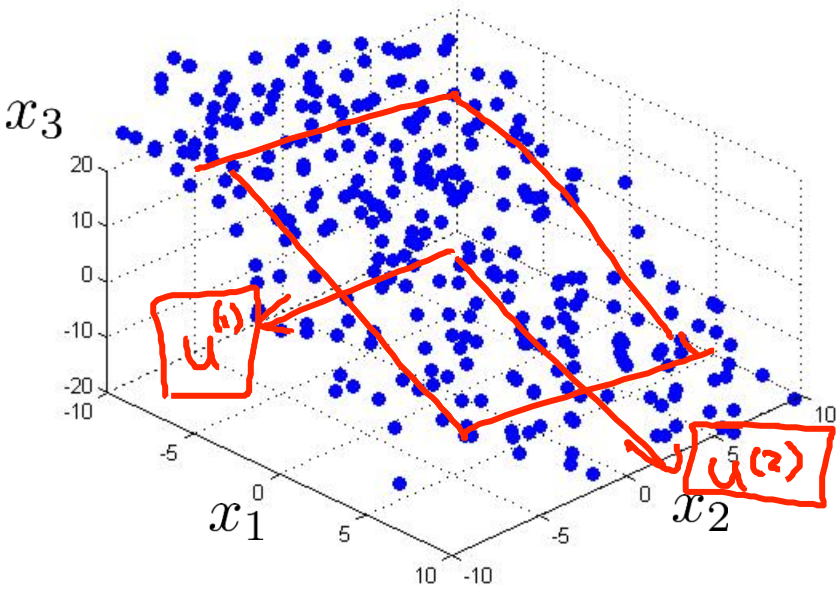
If different features on different scales (e.g., $x_1 =$ size of house, $x_2 =$ number of bedrooms), scale features to have comparable range of values.

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

Principal Component Analysis (PCA) algorithm



Reduce data from 2D to 1D



Reduce data from 3D to 2D

$z^{(i)}$ $\quad x^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}$

$z^{(2)}$ $\quad z_1$

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

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

$n \times n$

Sigma

Compute "eigenvectors" of matrix Σ :

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

$n \times n$ matrix.

→ Singular value decomposition

eig(Sigma)

$$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \dots & u^{(n)} \\ | & | & | & & | \end{bmatrix}$$

k

$U \in \mathbb{R}^{n\times n}$

$u^{(1)}, \dots, u^{(k)}$

Principal Component Analysis (PCA) algorithm

From `[U,S,V] = svd(Sigma)`, we get:

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

k

$x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$

$$z^{(i)} = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}}_{n \times k}^{T} x^{(i)} = \underbrace{\underbrace{\begin{bmatrix} — & (u^{(1)})^T & — \\ & \vdots & \\ — & (u^{(k)})^T & — \end{bmatrix}}_{k \times n} \underset{n \times 1}{x^{(i)}}}_{k \times 1}$$

U_{reduce}

$z \in \mathbb{R}^k$

Principal Component Analysis (PCA) algorithm summary

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

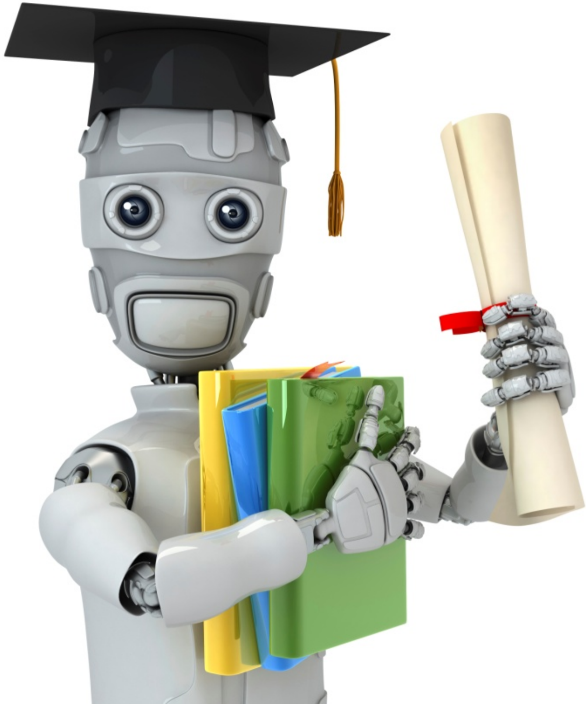
```
Sigma =
```
$$\frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$$

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

$\boxed{\text{Sigma} = (1/m) * X' * X;}$

```
[U,S,V] = svd(Sigma);
Ureduce = U(:,1:k);
z = Ureduce'*x;
```

$x \in \mathbb{R}^n$ $\cancel{x_0 = 1}$

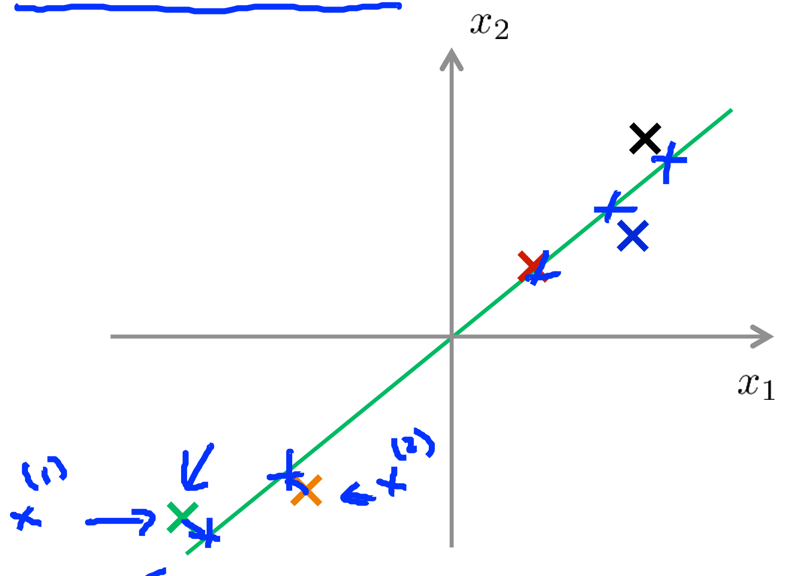


Machine Learning

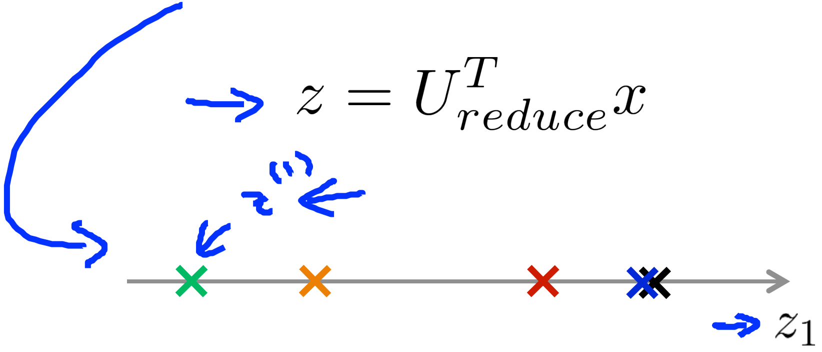
Dimensionality Reduction

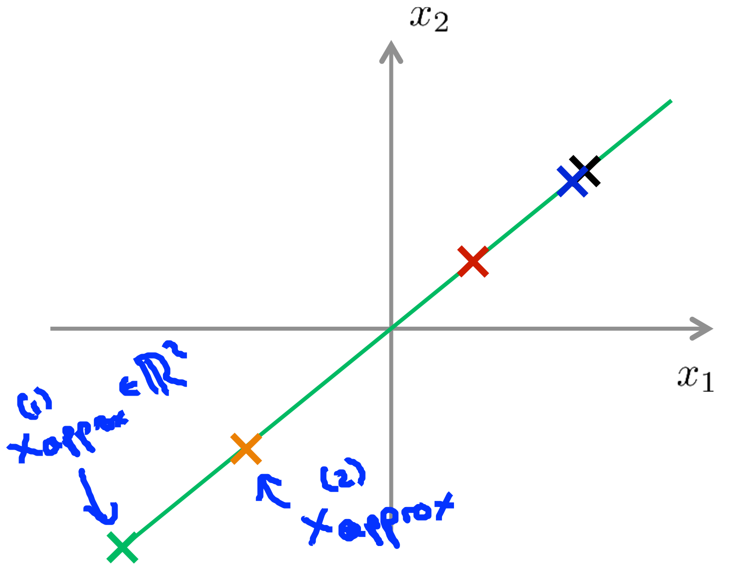
Reconstruction from compressed representation

Reconstruction from compressed representation



$$\rightarrow \quad z = U_{reduce}^T x$$

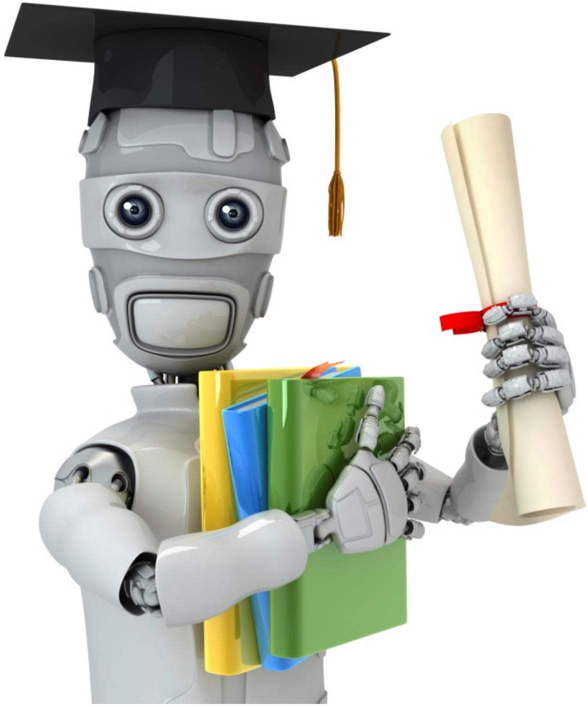




$$z \in \mathbb{R} \rightarrow x \in \mathbb{R}^2$$

$$\underbrace{x_{approx}^{(1)}}_{\in \mathbb{R}^n} = \underbrace{\underbrace{U_{reduce}}_{n \times k} \cdot \underbrace{z^{(1)}}_{k \times 1}}_{n \times 1}$$

$\tilde{x}$



Machine Learning

Dimensionality Reduction

Choosing the number of principal components

Choosing k (number of principal components)

Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2$

Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2$

Typically, choose k to be smallest value so that

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

$\cancel{0.05}$ $\cancel{5\%}$

0.10 (10%)

"~~99%~~ of variance is retained"

~~95%~~ 90%

Choosing k (number of principal components)

Algorithm:

Try PCA with ~~$k = 1$~~ ~~$k=2$~~ ~~$k=3$~~ $k=4$ ...

Compute $U_{reduce}, \underline{z^{(1)}}, \underline{z^{(2)}}, \ldots, \underline{z^{(m)}}, x^{(1)}_{approx}, \ldots, x^{(m)}_{approx}$

Check if

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

$\underline{k = 17}$

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

$$S = \begin{bmatrix} S_{11} & & & & 0 \\ & S_{22} & & & \\ & & S_{33} & & \\ & & & \ddots & \\ 0 & & & & S_{nn} \end{bmatrix}$$

$k=3$

For given k

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01$$

$$\boxed{\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq \underline{0.99}}$$

Andrew Ng

Choosing k (number of principal components)

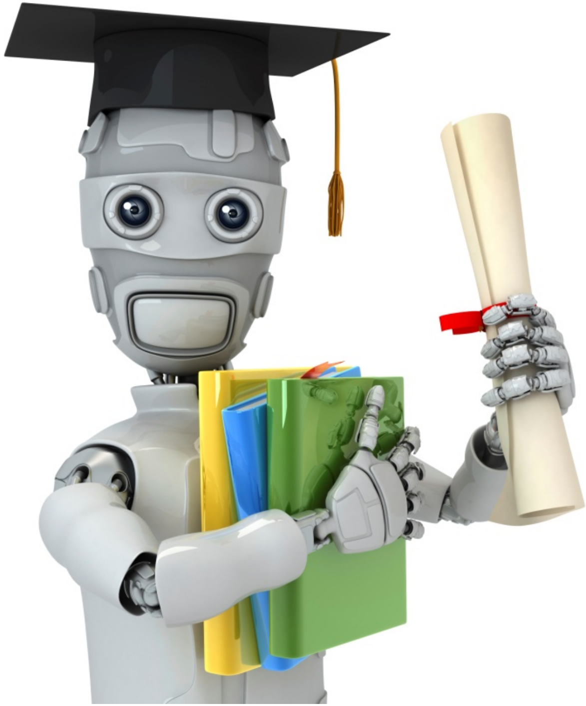
```
[U,S,V] = svd(Sigma)
```

Pick smallest value of k for which

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

$k=100$

(99% of variance retained)



Machine Learning

Dimensionality Reduction

Advice for applying PCA

Supervised learning speedup

$x^{(i)} \in \mathbb{R}^{10{,}000}$

100

100

$\rightarrow \boxed{(x^{(1)}, y^{(1)})}, (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

Extract inputs:

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

$\downarrow PCA$

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

$x \downarrow z$

U_{reduce}

New training set:

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

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

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

$x \rightarrow z$

Application of PCA

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

Choose k by % of variance retain

- Visualization

k = 2 or k = 3

Bad use of PCA: To prevent overfitting

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

1000

10000

Thus, fewer features, less likely to overfit.

Bad!

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

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

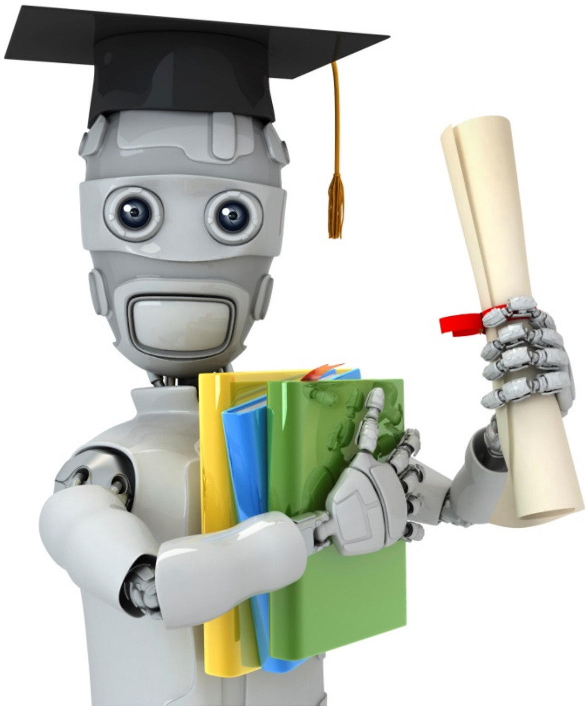
PCA is sometimes used where it shouldn't be

Design of ML system:

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

How about doing the whole thing without using PCA?

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



Machine Learning

Anomaly detection

Problem motivation

Anomaly detection example

Aircraft engine features:

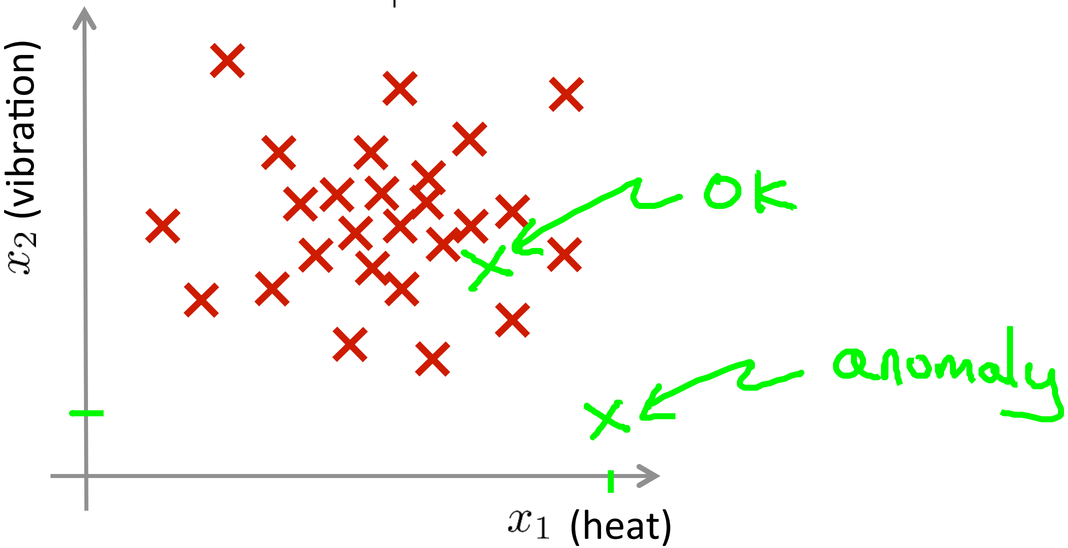
x_1 = heat generated

x_2 = vibration intensity

...

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

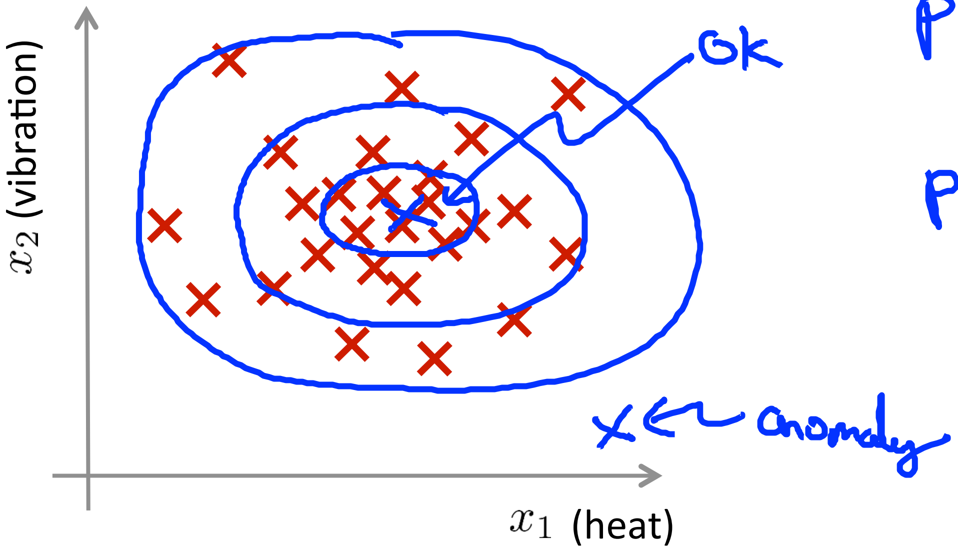
New engine: x_{test}



Density estimation

- Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$
- Is x_{test} anomalous?

Model $p(x)$.



$p(x_{test}) < \varepsilon \rightarrow$ flag anomaly

$p(x_{test}) \geq \varepsilon \rightarrow$ OK

Anomaly detection example

- Fraud detection:
  - $x^{(i)}$ = features of user i's activities
  - Model $p(x)$ from data.
  - Identify unusual users by checking which have $p(x) < \varepsilon$

  x_1

  x_2

  x_3

  x_4

  $p(x)$

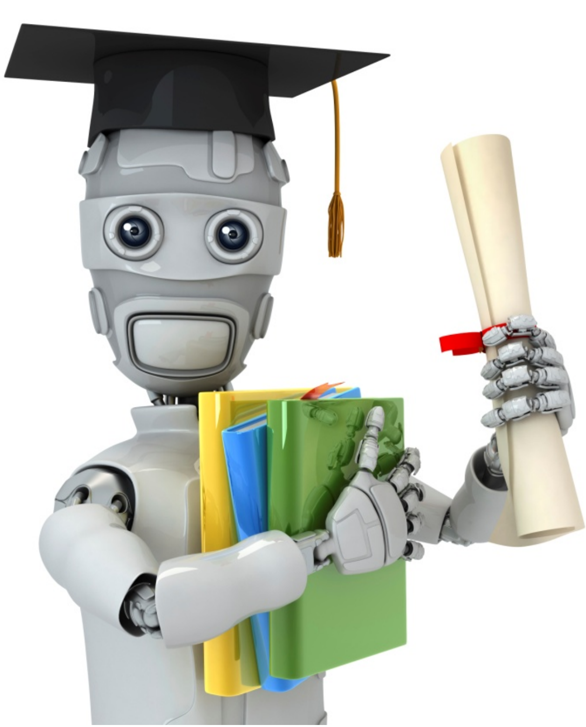
- Manufacturing
- Monitoring computers in a data center.
  - $x^{(i)}$ = features of machine i

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

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

    ...

    $p(x) < \varepsilon$



Machine Learning

Anomaly detection

Gaussian distribution

Gaussian (Normal) distribution

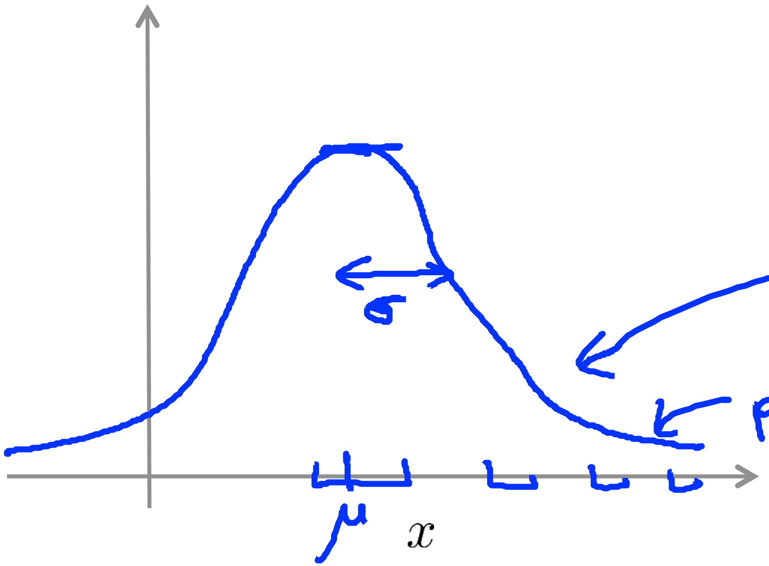
Say $x \in \mathbb{R}$. If x is a distributed Gaussian with mean μ , variance σ^2 .

$x \sim \mathcal{N}(\mu, \sigma^2)$

"distributed as"

σ standard deviation

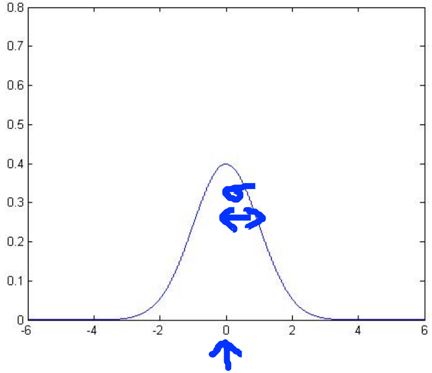
$$p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



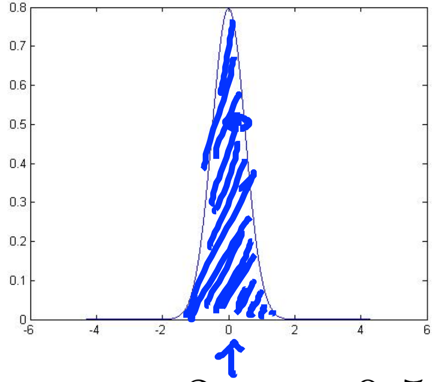
$p(x;\mu,\sigma^2)$

Gaussian distribution example

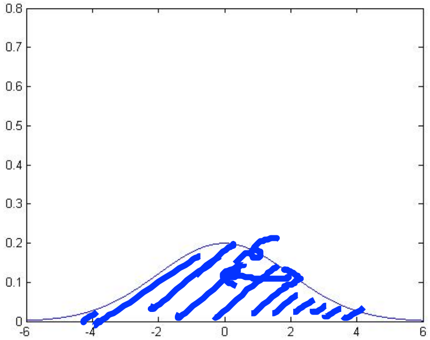
$\mu = 0, \sigma = 1$



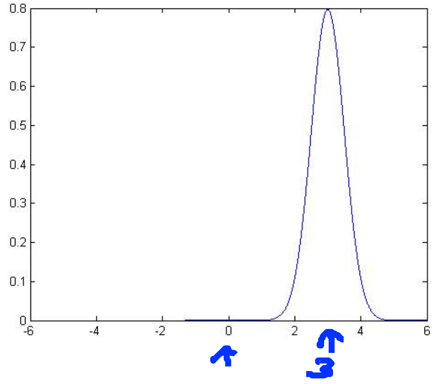
$\mu = 0, \sigma = \underline{0.5}$ $\sigma^2 = 0.25$



$\mu = 0, \sigma = 2$



$\mu = 3, \sigma = 0.5$

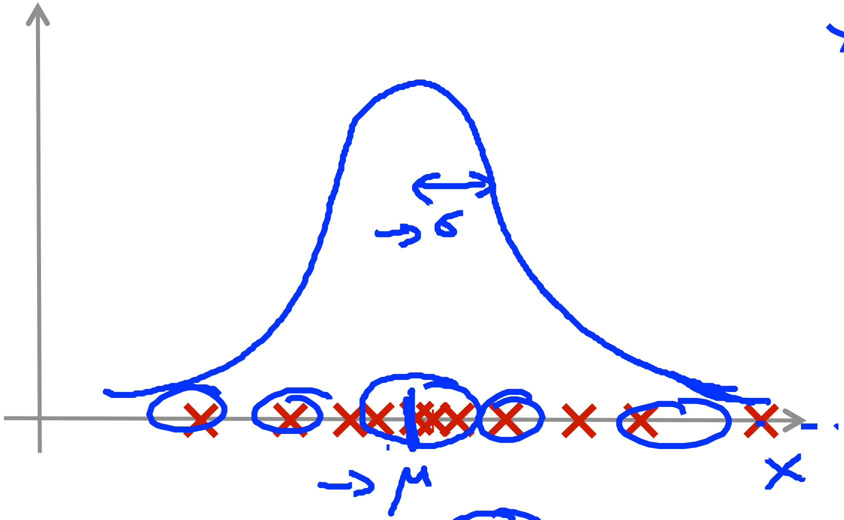


Andrew Ng

Parameter estimation

→ Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ $\quad \underline{x^{(i)} \in \mathbb{R}}$

$x^{(i)} \sim \mathcal{N}(\mu, \sigma^2)$

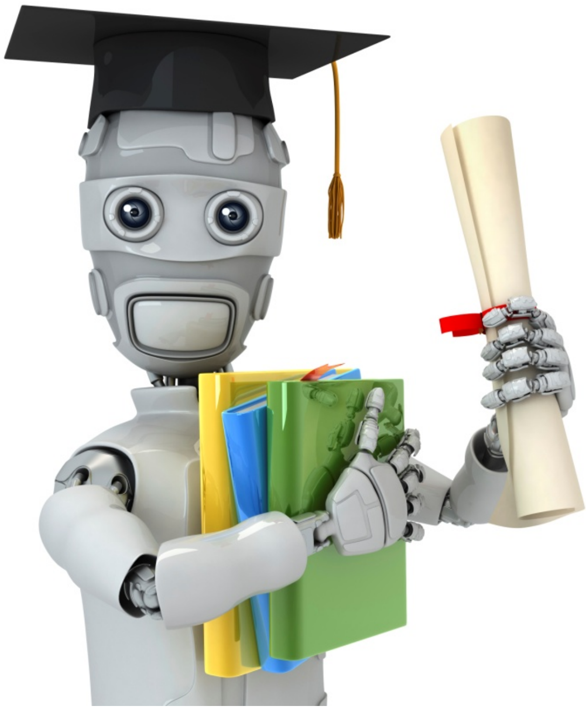


$\rightarrow \mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$

$\rightarrow \sigma^2 = \frac{1}{m} \sum_{i=1}^{m} \underbrace{(x^{(i)} - \mu)^2}$

$m-1$ $\quad \frac{1}{m-1}$ ←

Andrew Ng



Machine Learning

Anomaly detection

Algorithm

Density estimation

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

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

$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$

$x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$

$x_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$

$\rightarrow p(x)$

$$= \boxed{p(x_1; \mu_1, \sigma_1^2)\, p(x_2; \mu_2, \sigma_2^2)\, p(x_3; \mu_3, \sigma_3^2) \cdots p(x_n; \mu_n, \sigma_n^2)}$$

$$= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n$$

$$\prod_{i=1}^{n} i = 1 \times 2 \times 3 \times \ldots \times n$$

Anomaly detection algorithm

1. Choose features x_i that you think might be indicative of anomalous examples.

$\{x^{(1)}, \ldots, x^{(m)}\}$

2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

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

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

$p(x_j; \mu_j, \sigma_j^2)$

$\mu_1, \mu_2, \ldots, \mu_n$

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

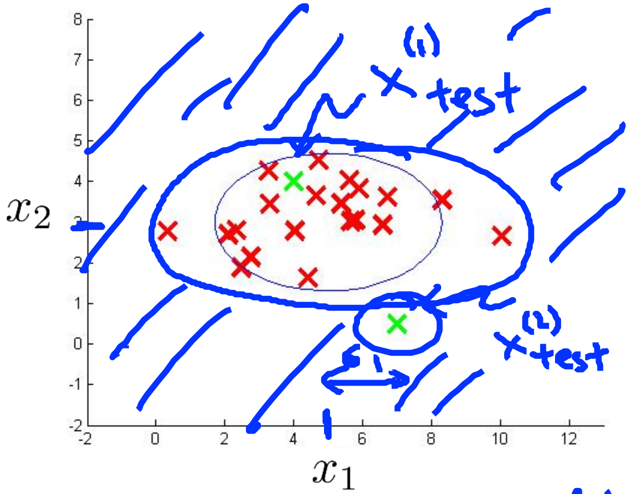
3. Given new example x, compute $p(x)$:

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

Anomaly if $p(x) < \varepsilon$

Anomaly detection example

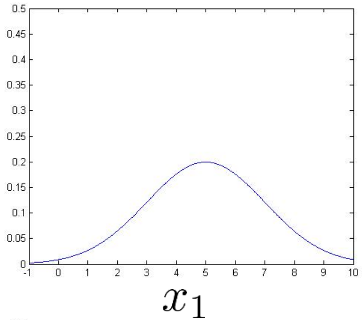
$\rightarrow p(x) = p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2)$



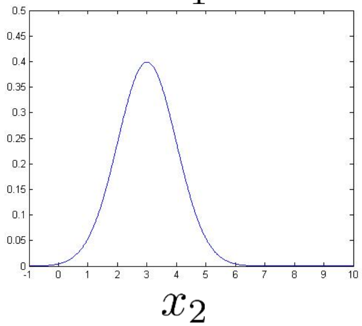
σ_1^2, σ_2^2

$\sigma_1^2 = 4$

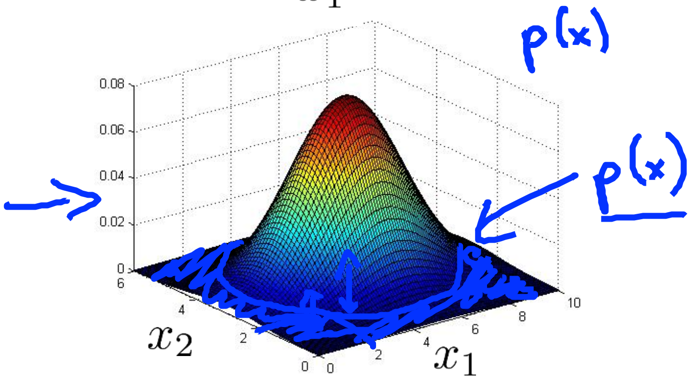
$$\mu_1 = 5, \sigma_1 = 2$$
$$\mu_2 = 3, \sigma_2 = 1$$



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

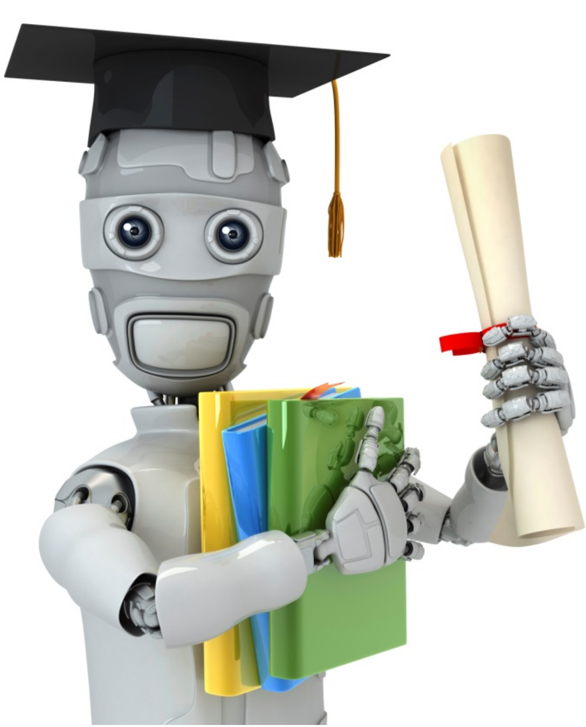


$p(x_2; \mu_2, \sigma_2^2)$



$$\varepsilon = 0.02$$

$$p(x_{test}^{(1)}) = 0.0426 \quad > \varepsilon$$
$$p(x_{test}^{(2)}) = 0.0021 \quad < \varepsilon$$



Machine Learning

Anomaly detection

Developing and evaluating an anomaly detection system

The importance of real-number evaluation

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

- Assume we have some labeled data, of anomalous and non-anomalous examples. ($y = 0$ if normal, $y = 1$ if anomalous).
- Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (assume normal examples/not anomalous)
- Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \ldots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$
- Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \ldots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

$y=1$

Aircraft engines motivating example

$\boxed{10000}$ good (normal) engines

$\boxed{20}$ flawed engines (anomalous) 2-50 $y=1$

$\mu_1, \sigma_1^2, \ldots, \mu_n, \sigma_n^2$

Training set: $\boxed{6000}$ good engines $(y=0)$ $p(x) = p(x_1; \mu_1, \sigma_1^2) \cdots p(x_n; \mu_n, \sigma_n^2)$

CV: $\boxed{2000}$ good engines ($y = 0$), $\boxed{10}$ anomalous ($y = 1$)

Test: $\boxed{2000}$ good engines ($y = 0$), $\boxed{10}$ anomalous ($y = 1$)

Alternative:

Training set: $\boxed{6000}$ good engines

CV: $\boxed{4000}$ good engines ($y = 0$), 10 anomalous ($y = 1$)

Test: $\boxed{4000}$ good engines ($y = 0$), $\boxed{10}$ anomalous ($y = 1$)

Algorithm evaluation

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

- Fit model $p(x)$ on training set $\{x^{(1)}, \ldots, x^{(m)}\}$
- On a cross validation/test example x, predict

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

$y=0$

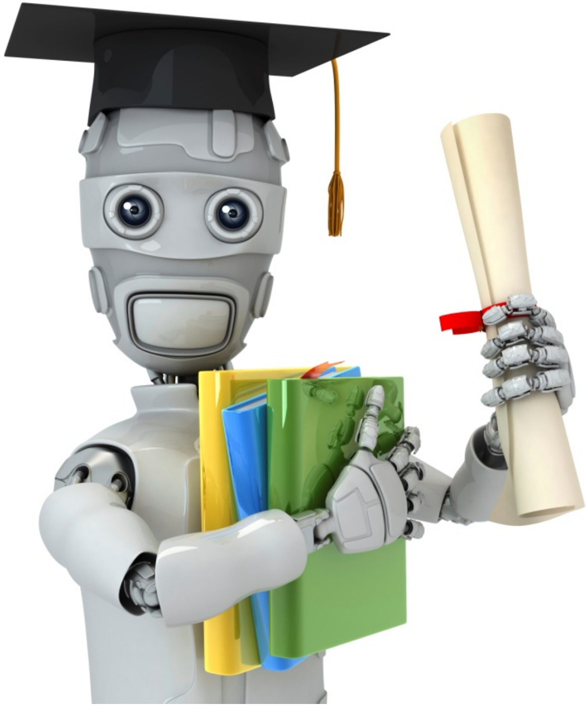
Possible evaluation metrics:

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

CV

Test set

Can also use cross validation set to choose parameter ε



Machine Learning

Anomaly detection

Anomaly detection vs. supervised learning

Anomaly detection vs. Supervised learning

Anomaly detection

- Very small number of positive examples ($y = 1$). (0-20 is common).
- Large number of negative ($y = 0$) examples. $\boxed{p(x)}$
- Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.

Supervised learning

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

Spam

Anomaly detection vs. Supervised learning

Anomaly detection

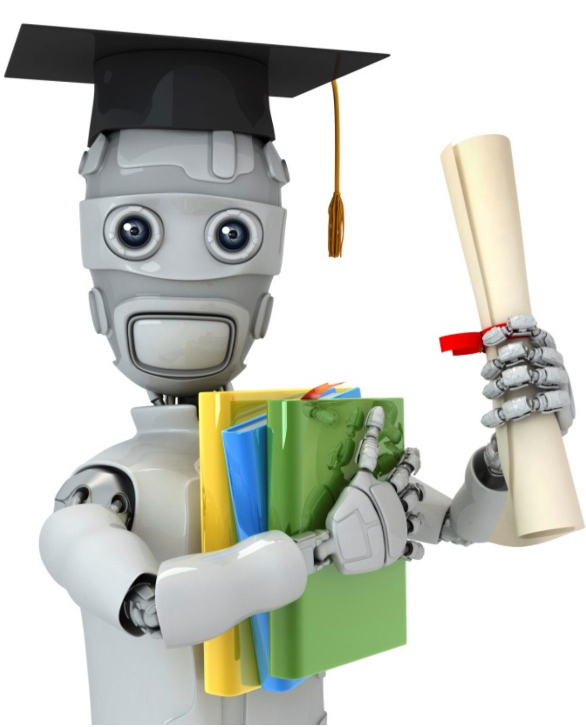
- Fraud detection y=1
- Manufacturing (e.g. aircraft engines)
- Monitoring machines in a data center

⋮

Supervised learning

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

⋮



Machine Learning

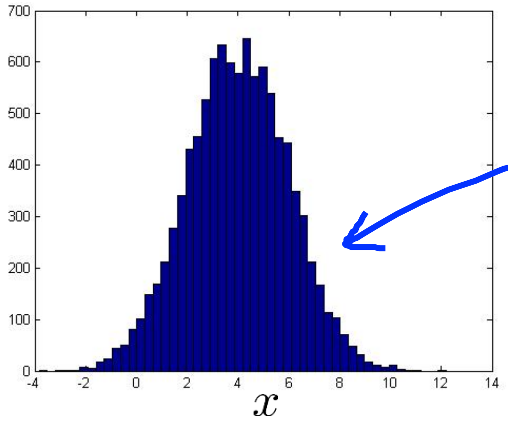
Anomaly detection

Choosing what features to use

Non-gaussian features

$p(x_i; \mu_i, \sigma_i^2)$

hist



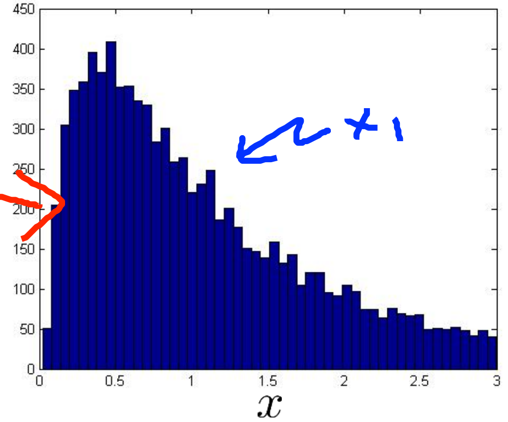
$x_1 \leftarrow \log(x_1)$

$x_2 \leftarrow \log(x_2+1)$

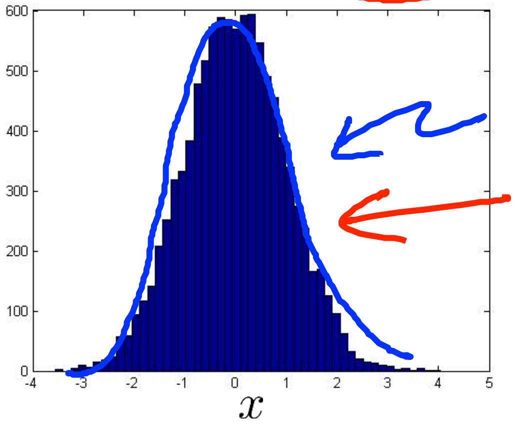
$\log(x_2 + c)$

$x_3 \leftarrow \sqrt{x_3} = x_3^{\frac{1}{2}}$

$x_4 \leftarrow x_4^{\frac{1}{3}}$



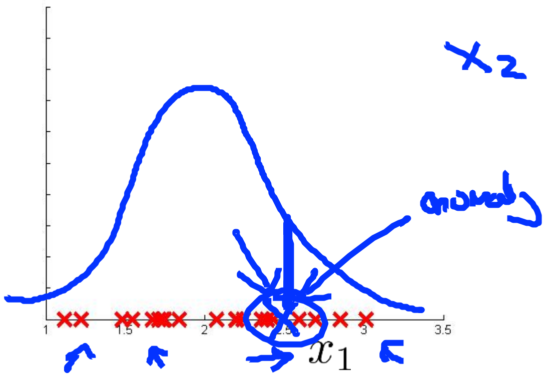
$\log(x)$

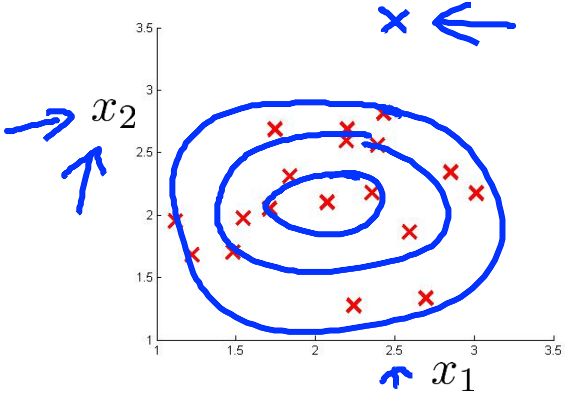


Error analysis for anomaly detection

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

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





Monitoring computers in a data center

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

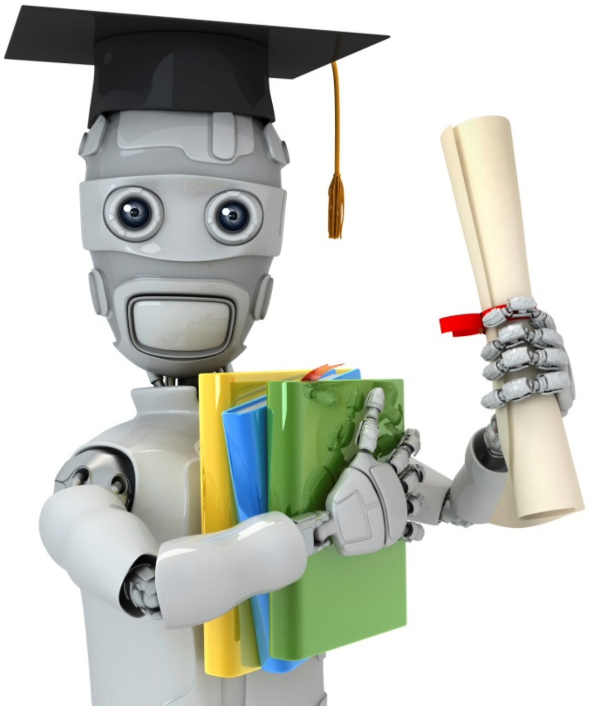
x_1 = memory use of computer

x_2 = number of disk accesses/sec

x_3 = CPU load

x_4 = network traffic

$$x_5 = \frac{\text{CPU load}}{\text{network traffic}} \qquad x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$$

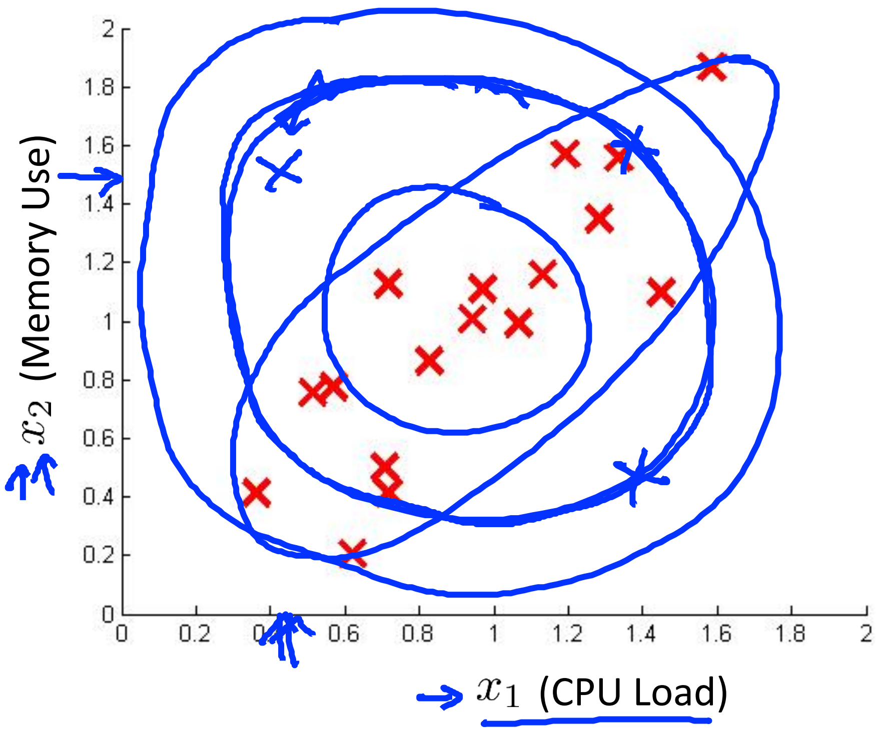


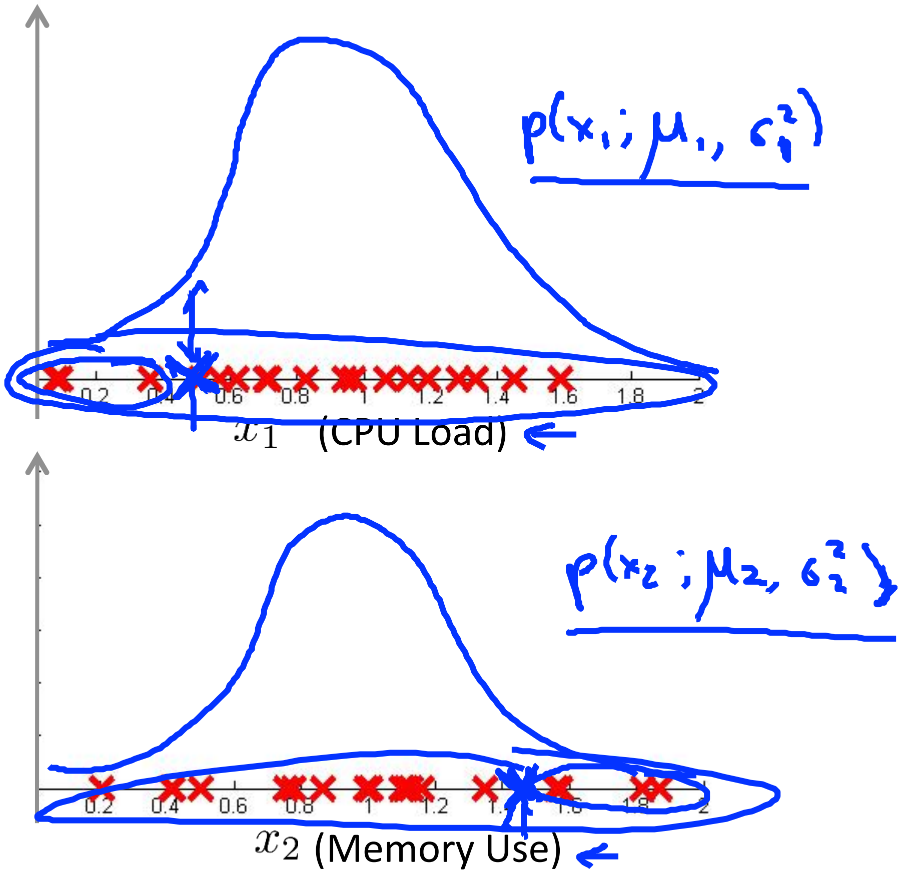
Machine Learning

Anomaly detection

Multivariate Gaussian distribution

Motivating example: Monitoring machines in a data center





Andrew Ng

Multivariate Gaussian (Normal) distribution

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

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

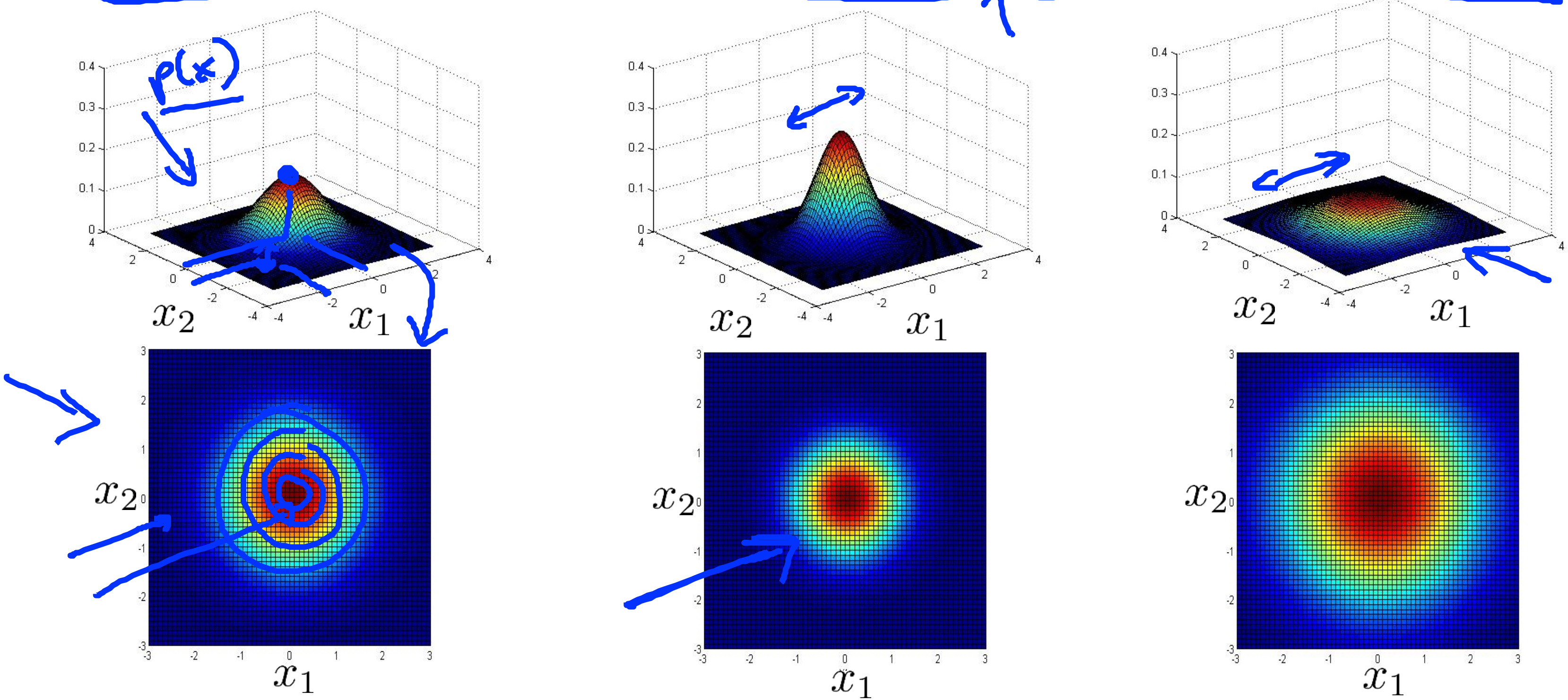
Parameters: $\mu \in \mathbb{R}^n$, $\boxed{\Sigma \in \mathbb{R}^{n \times n}}$ (covariance matrix)

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

$|\Sigma|$ = determinant of Σ | det(Sigma)

Multivariate Gaussian (Normal) examples

$$\boxed{\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}} \qquad \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boxed{\Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}} \qquad \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boxed{\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}}$$



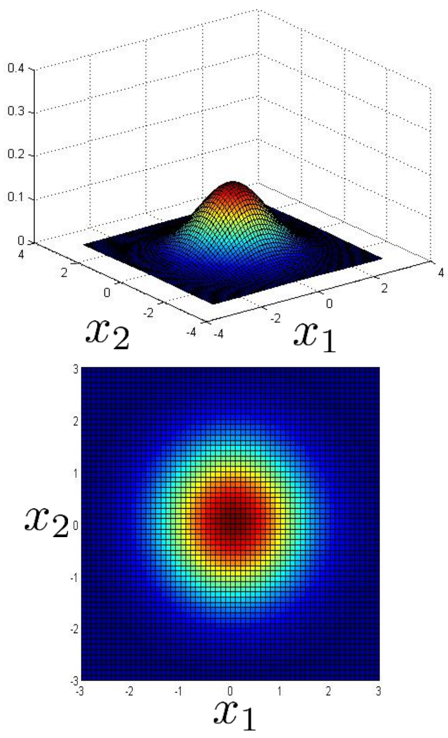
Andrew Ng

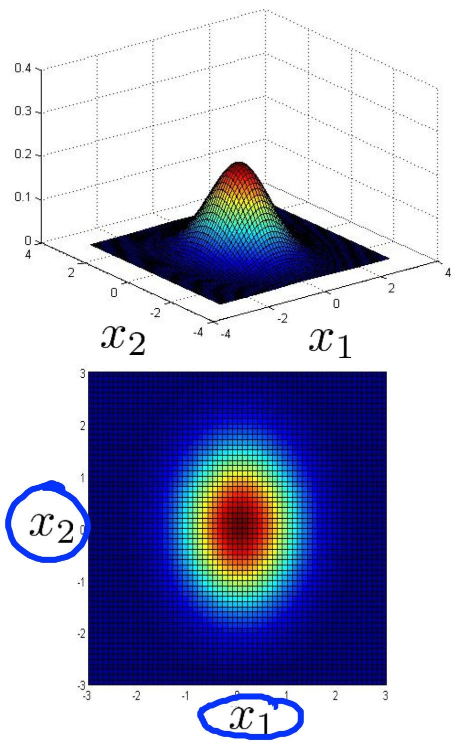
Multivariate Gaussian (Normal) examples

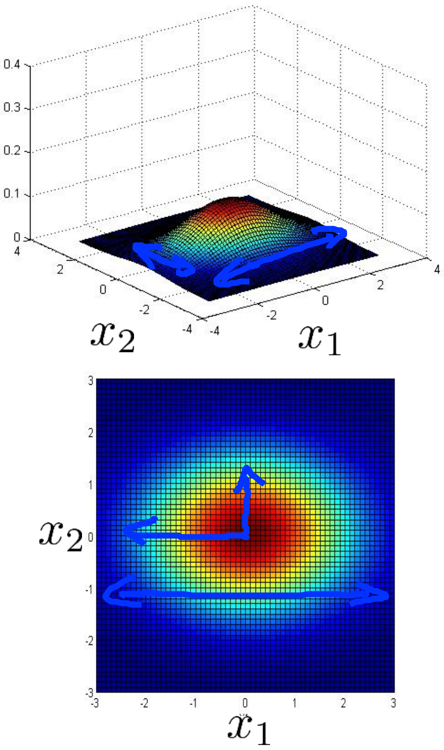
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

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

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

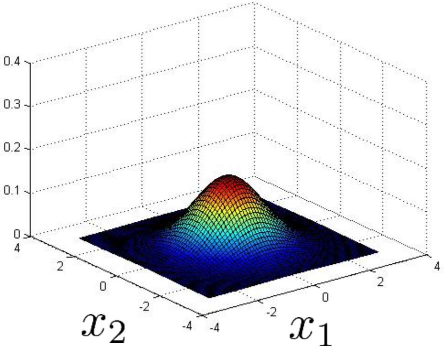


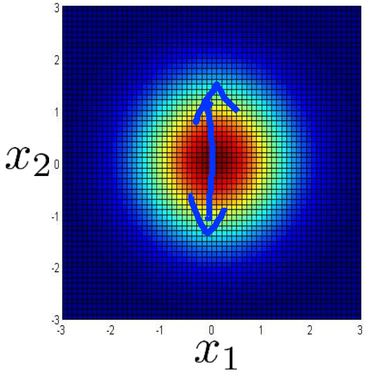




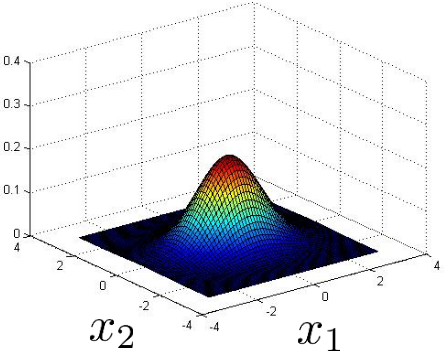
Multivariate Gaussian (Normal) examples

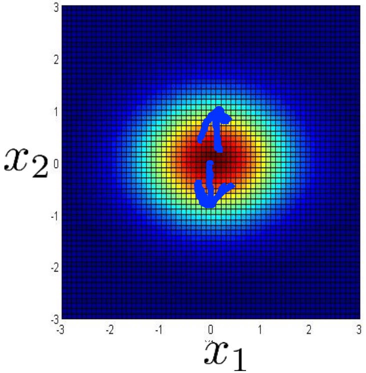
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



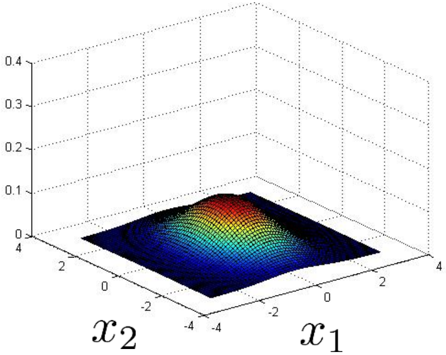


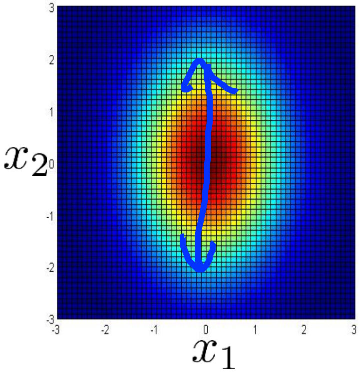
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \end{bmatrix}$$





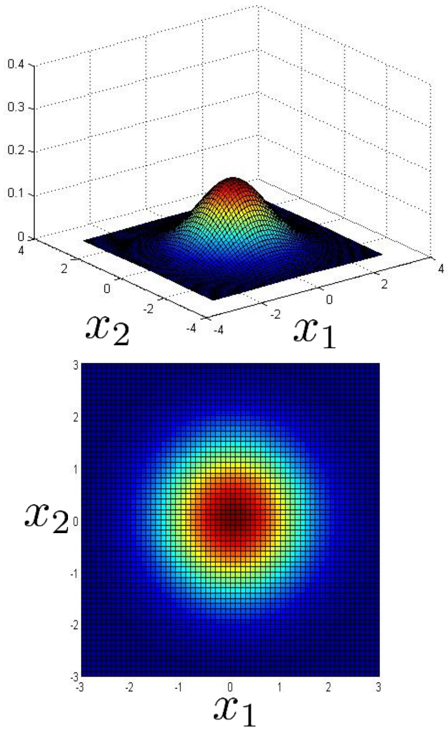
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$



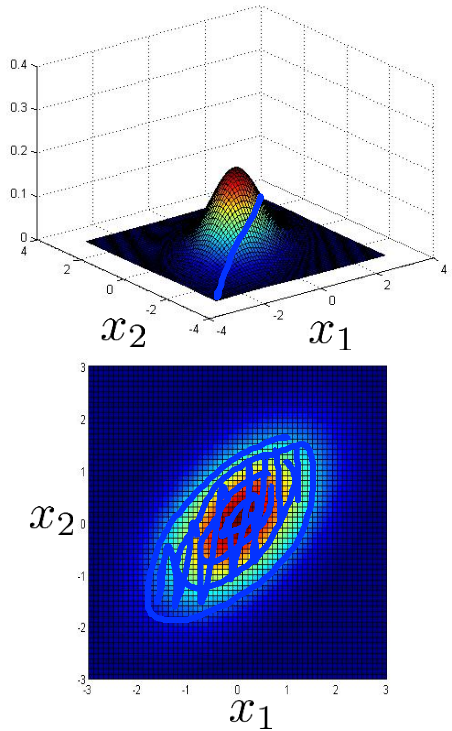


Multivariate Gaussian (Normal) examples

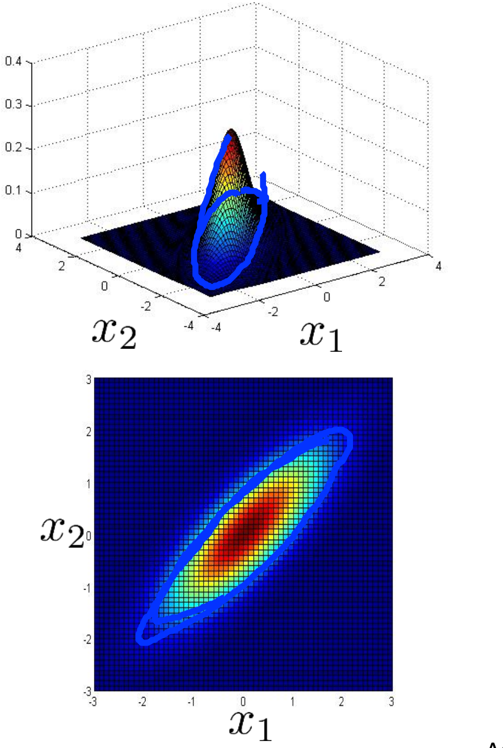
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



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



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

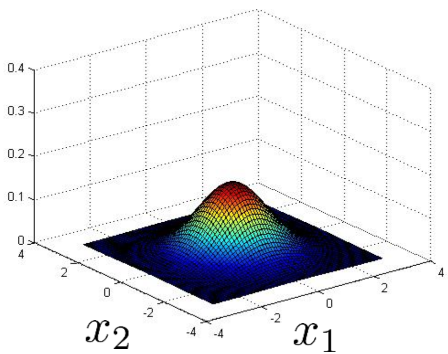


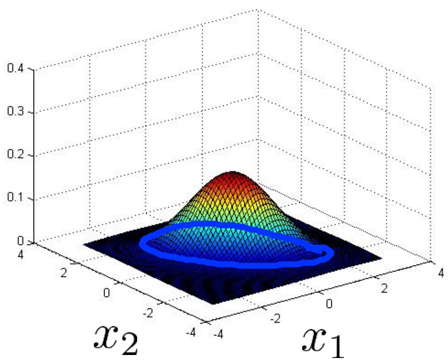
Multivariate Gaussian (Normal) examples

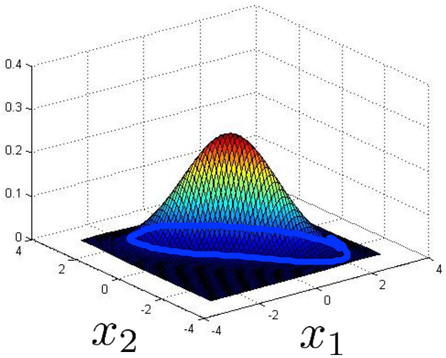
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

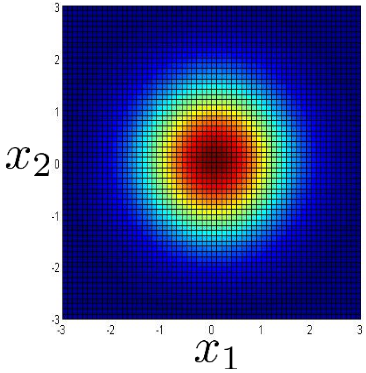
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$

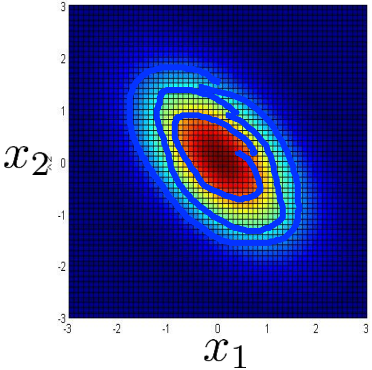
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$$

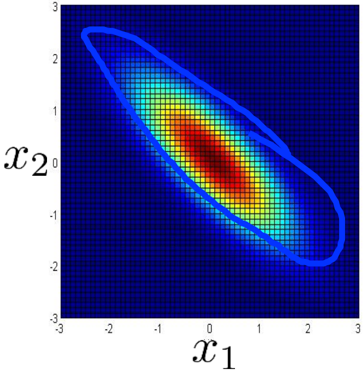










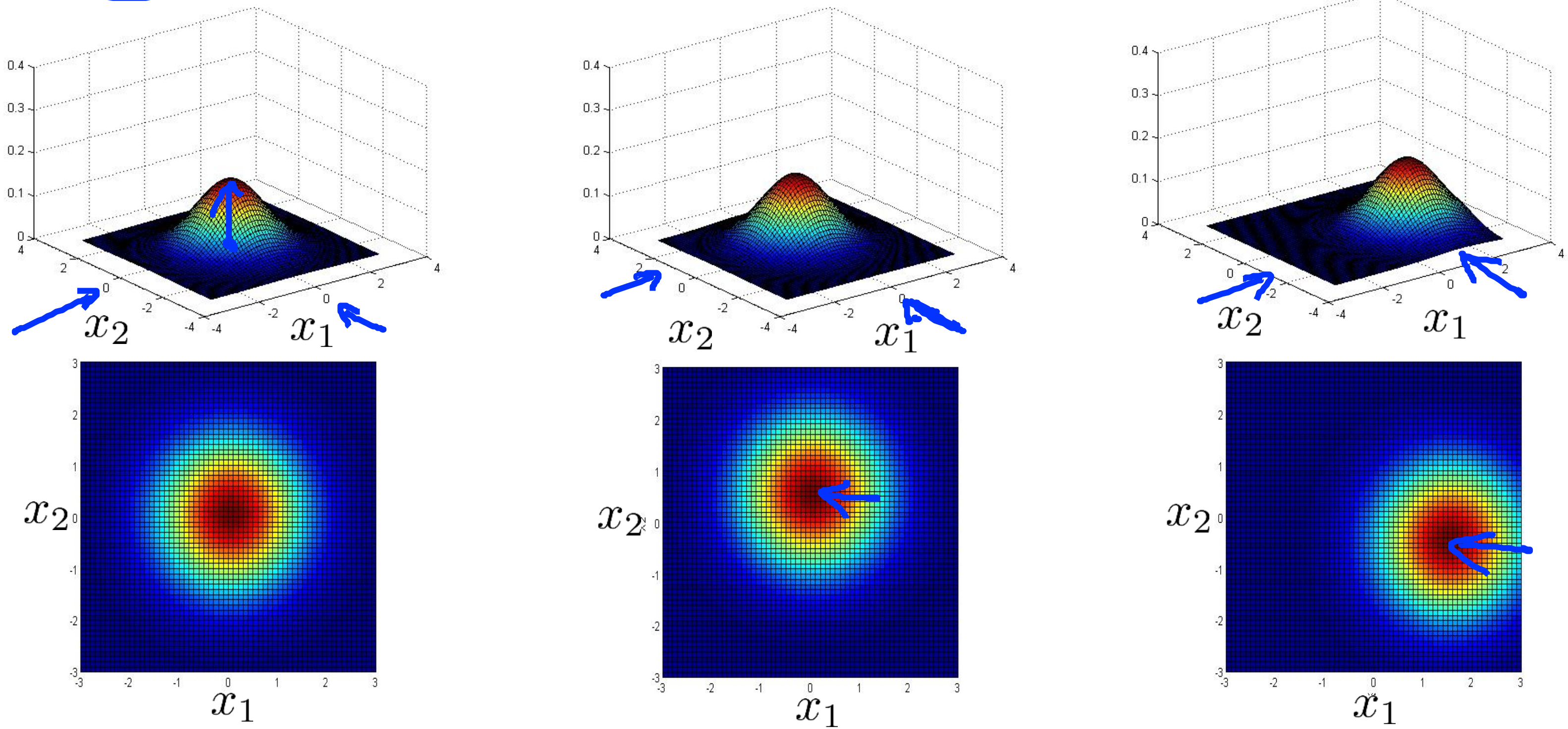


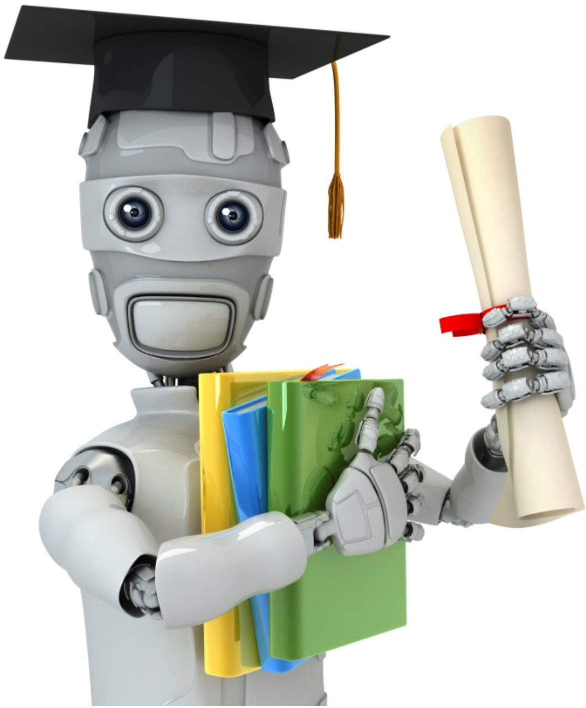
Multivariate Gaussian (Normal) examples

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

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

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





Machine Learning

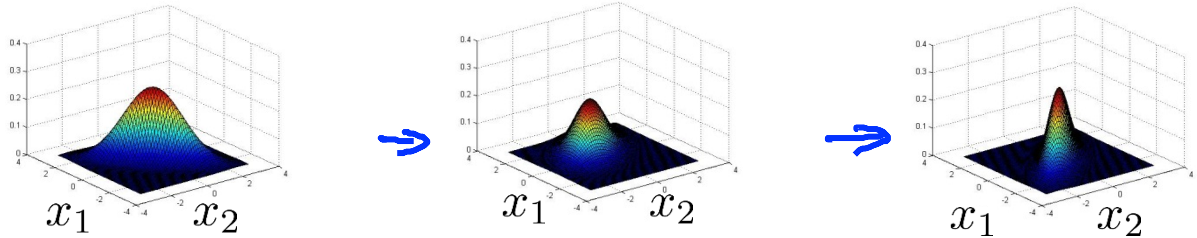
Anomaly detection

Anomaly detection using the multivariate Gaussian distribution

Multivariate Gaussian (Normal) distribution

Parameters μ, Σ $\mu \in \mathbb{R}^n$ $\Sigma \in \mathbb{R}^{n \times n}$

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



Parameter fitting:

Given training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\} \leftarrow$ $x \in \mathbb{R}^n$

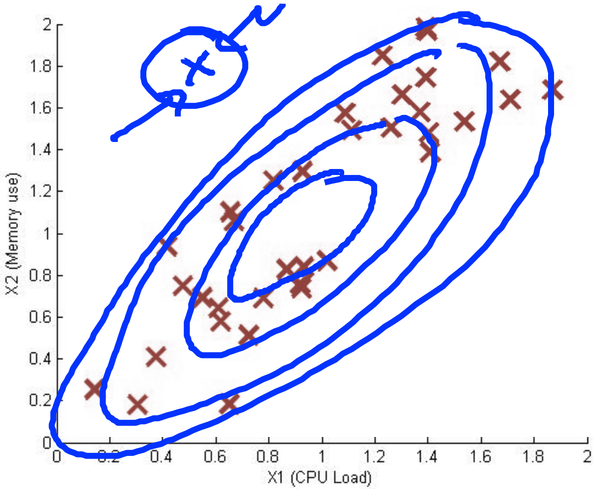
$$\rightarrow \boxed{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \rightarrow \boxed{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T$$

Anomaly detection with the multivariate Gaussian

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

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

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



2. Given a new example x, compute

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

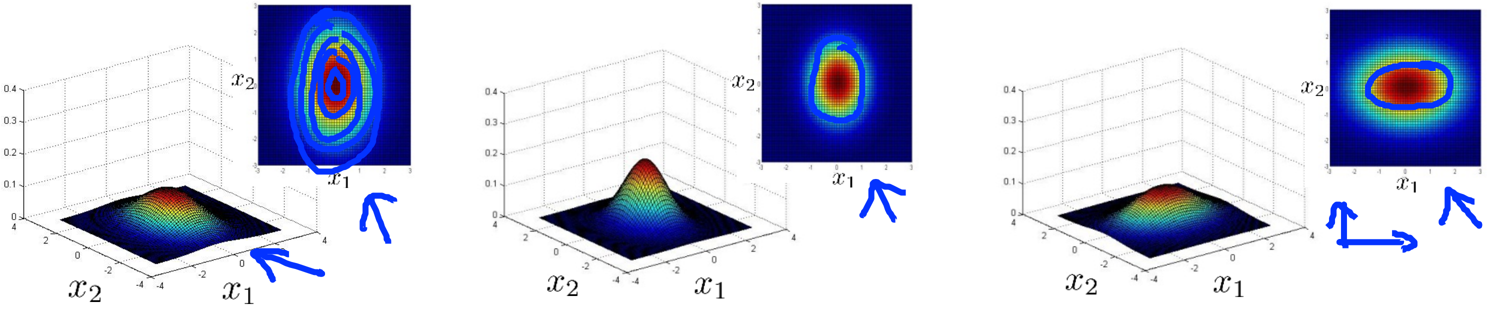
Flag an anomaly if $p(x) < \varepsilon$

Andrew Ng

Relationship to original model

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

p(x)



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

Corresponds to multivariate Gaussian

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

where $\Sigma =$

Andrew Ng

Original model vs. Multivariate Gaussian

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

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

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

$$\boxed{x_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{memory}}}$$

Automatically captures correlations between features

$\Sigma \in \mathbb{R}^{n \times n}$ $\quad$ Σ^{-1}

Computationally cheaper (alternatively, scales better to large n)

$n = 10{,}000$, $\quad$ $n = 100{,}000$

Computationally more expensive

Σ $\quad$ $\sim \frac{n^2}{2}$

$x_1 = \cancel{x_2}$

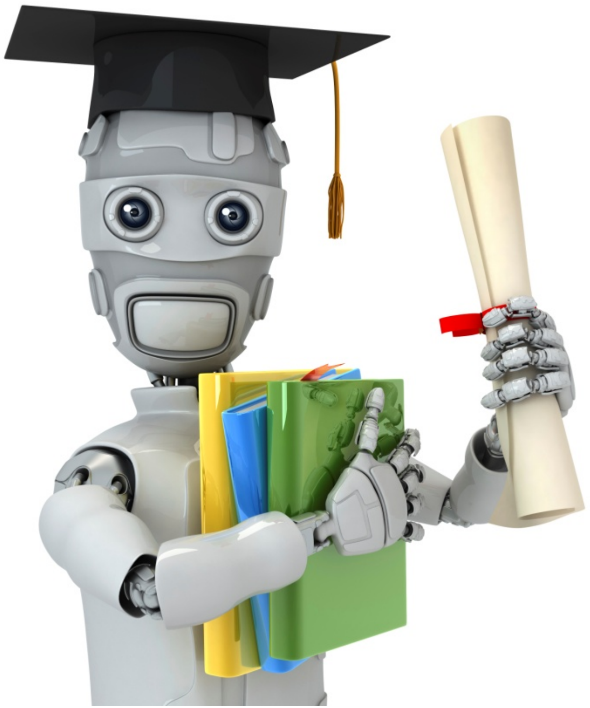
$\cancel{x_3} = x_4 + x_5$

OK even if m (training set size) is small

Must have $\boxed{m > n}$ or else Σ is non-invertible.

$\boxed{m \geq 10n}$

Andrew Ng



Machine Learning

Recommender Systems

Problem formulation

Example: Predicting movie ratings

- User rates movies using ~~one~~ zero to five stars

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	? 4.5	? 0	0
Cute puppies of love	? 5	4	0	? 0
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	? 4

$n_u = 4$ $\qquad$ $n_m = 5$

n_u = no. users

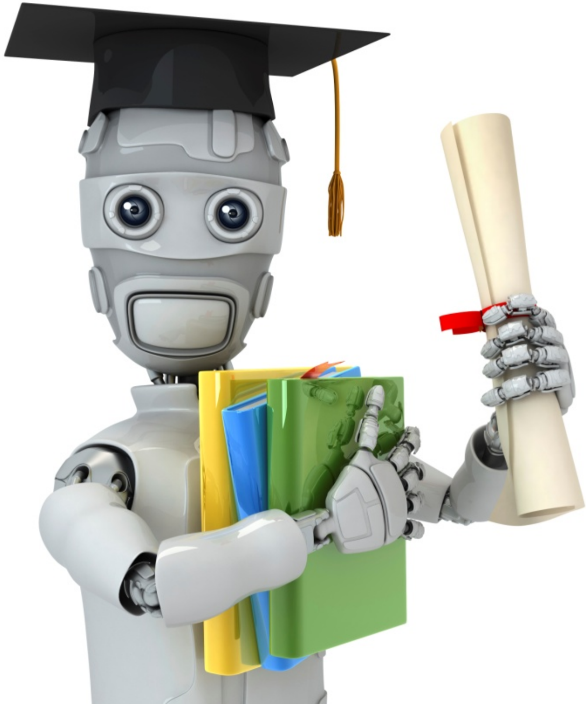
n_m = no. movies

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

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

$0, \ldots, 5$

Andrew Ng



Machine Learning

Recommender Systems

Content-based recommendations

Content-based recommender systems

$n_u = 4, \; n_m = 5$

$x_0 = 1$

$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$
$x^{(1)}$ Love at last 1	5	5	0	0
$x^{(2)}$ Romance forever 2	5	?	?	0
$x^{(3)}$ Cute puppies of love 3	? 4.95	4	0	?
Nonstop car chases 4	0	0	5	4
$x^{(5)}$ Swords vs. karate 5	0	0	5	?

$n = 2$

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

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

$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix} \longleftrightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$

$(\theta^{(1)})^T x^{(3)} = 5 \times 0.99$
$= 4.95$

Problem formulation

- $r(i,j) = 1$ if user j has rated movie i (0 otherwise)
- $y^{(i,j)} =$ rating by user j on movie i (if defined)

- $\theta^{(j)}$ = parameter vector for user j
- $x^{(i)}$ = feature vector for movie i
- For user j, movie i, predicted rating: $(\theta^{(j)})^T(x^{(i)})$

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

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

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

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

Andrew Ng

Optimization objective:

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

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

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

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

$\Theta^{(1)}, \ldots, \Theta^{(n_u)}$

Optimization algorithm:

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

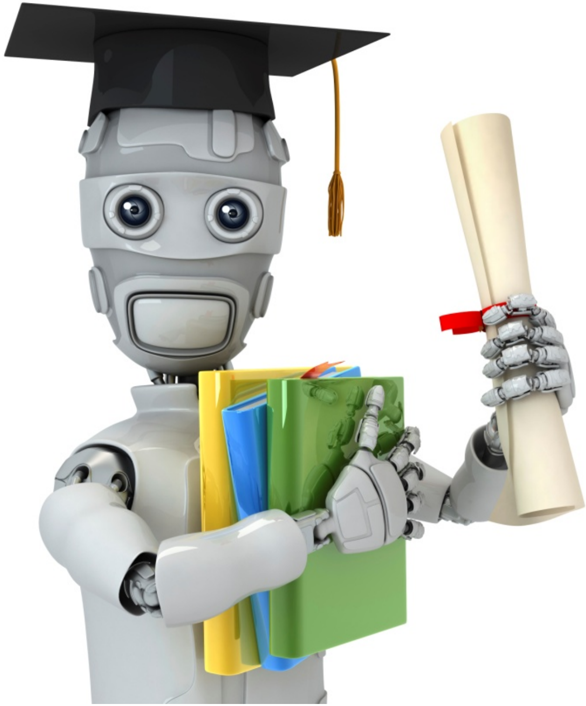
Gradient descent update:

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

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

$$\cancel{\frac{1}{m^{(j)}}}$$

Andrew Ng



Machine Learning

Recommender Systems

Collaborative filtering

Problem motivation

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

Problem motivation

$x_0 = 1$

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 (romance)	x_2 (action)
$x^{(1)}$ ~~Love at last~~	5	5	0	0	? 1.0	? 0.0
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs. karate	0	0	5	?	?	?

$$x^{(1)} = \begin{bmatrix} 1 \\ 1.0 \\ 0.0 \end{bmatrix}$$

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

$\theta^{(j)}$

$x^{(i)}$

$(\theta^{(1)})^T x^{(1)} \approx 5$

$(\theta^{(2)})^T x^{(1)} \approx 5$

$(\theta^{(3)})^T x^{(1)} \approx 0$

$(\theta^{(4)})^T x^{(1)} \approx 0$

Optimization algorithm

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

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

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

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

Collaborative filtering

$r^{(i,j)}$

$y^{(i,j)}$

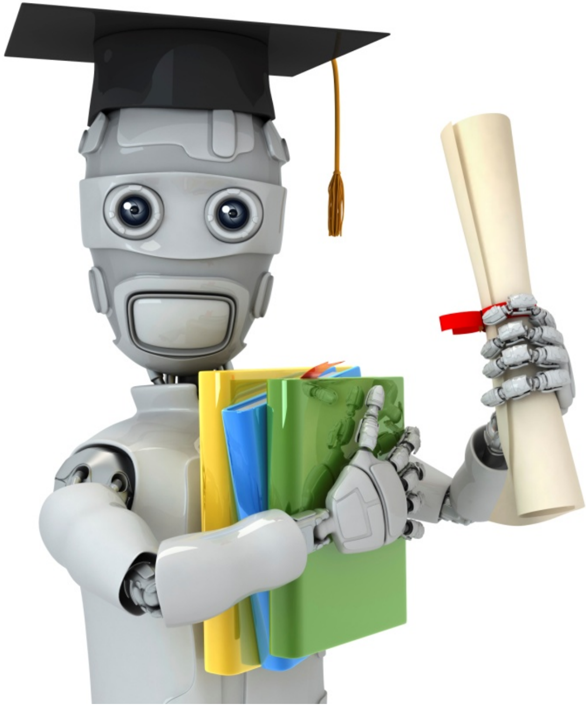
Given $x^{(1)}, \ldots, x^{(n_m)}$ (and movie ratings),

can estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$,

can estimate $x^{(1)}, \ldots, x^{(n_m)}$

Guess $\Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \ldots$



Machine Learning

Recommender Systems

Collaborative filtering algorithm

Collaborative filtering optimization objective

$(i,j) : r(i,j)=1$

$\cancel{x_0}$ $\cancel{\theta_0}$ $x \in \mathbb{R}^n$ $\theta \in \mathbb{R}^n$ $\cancel{x \in \mathbb{R}^{n+1}}$ $x_1 = 1$

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

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

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

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

Minimizing $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$ simultaneously:

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

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)} \\ \theta^{(1)},\ldots,\theta^{(n_u)}}} J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$$

$\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \ldots$

Andrew Ng

Collaborative filtering algorithm

~~$x_0 = 1$~~ $x \in \mathbb{R}^n$, $\theta \in \mathbb{R}^n$

1. Initialize $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

$\cancel{\theta_0}$, θ_1, ..., θ_n

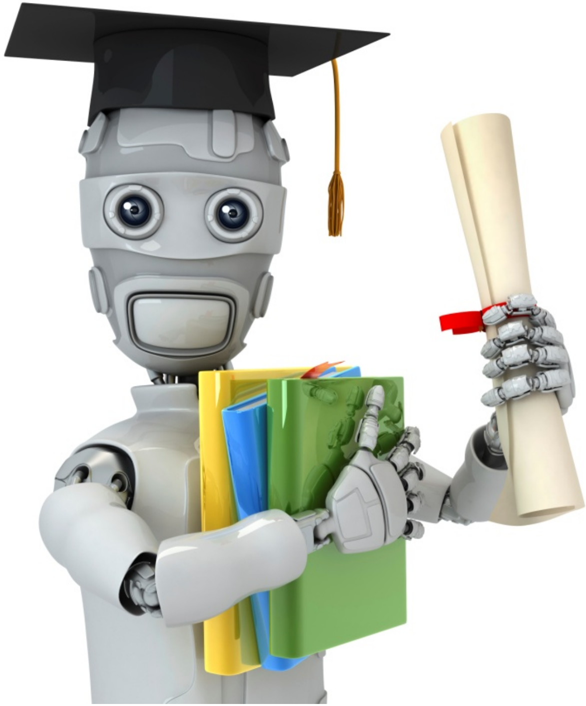
$$x_k^{(i)} := x_k^{(i)} - \alpha \underbrace{\left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)} \right)}_{\frac{\partial}{\partial x_k^{(i)}} J(\ldots)}$$

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

3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

$(\theta^{(j)})^T (x^{(i)})$

Andrew Ng



Machine Learning

Recommender Systems

Vectorization: Low rank matrix factorization

Collaborative filtering

$n_m = 5$
$n_u = 4$

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

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

$y^{(i,j)}$

Collaborative filtering

$X \Theta^T$

$(\Theta^{(j)})^T (x^{(i)})$

(i,j)

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Predicted ratings:

$$\begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \ldots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \ldots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \ldots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}$$

$$X = \begin{bmatrix} - (x^{(1)})^T - \\ - (x^{(2)})^T - \\ \vdots \\ - (x^{(n_m)})^T - \end{bmatrix} \qquad \Theta = \begin{bmatrix} - (\theta^{(1)})^T - \\ - (\theta^{(2)})^T - \\ \vdots \\ - (\theta^{(n_u)})^T - \end{bmatrix}$$

→ Low rank matrix factorization

Finding related movies

For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$.

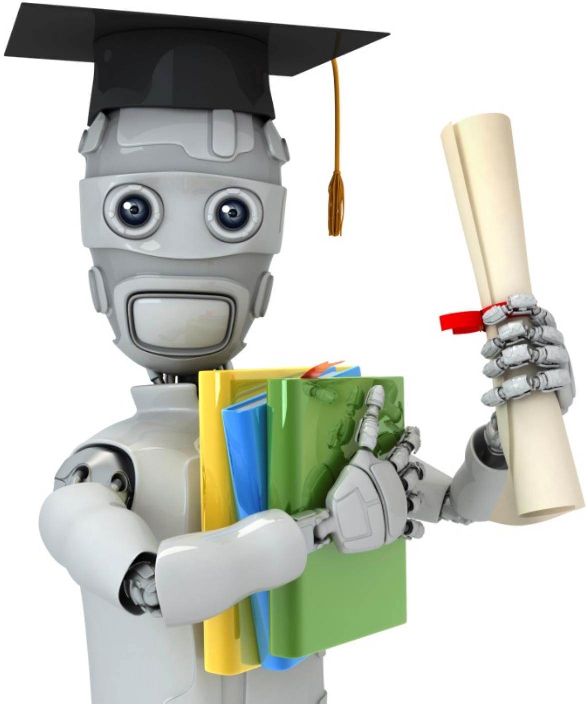
→ x_1 = romance, x_2 = action, x_3 = comedy, x_4 = ...

How to find movies j related to movie i?

small $\|x^{(i)} - x^{(j)}\|$ → movie j and i are "similar"

5 most similar movies to movie i:

Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.



Machine Learning

Recommender Systems

Implementational detail: Mean normalization

Users who have not rated any movies

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

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

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

$n = 2$ $\quad \theta^{(5)} \in \mathbb{R}^2$ $\quad \theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\frac{\lambda}{2}\left[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2\right]$

$(\theta^{(5)})^T x^{(i)} = 0$

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \begin{matrix} \leftarrow 2.5 \\ \leftarrow 2.5 \\ \leftarrow 2 \\ \\ \\ \end{matrix} \qquad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow \underline{Y} = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

$\downarrow$ learn $\underline{\Theta^{(j)}}, \underline{x^{(i)}}$

For user j, on movie i predict:

$$\rightarrow (\Theta^{(j)})^T (x^{(i)}) + \mu_i$$

User 5 (Eve):

$$\underline{\Theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}}$$

$$\underbrace{(\Theta^{(5)})^T (x^{(i)})}_{\rightarrow 0} + \boxed{\mu_i}$$