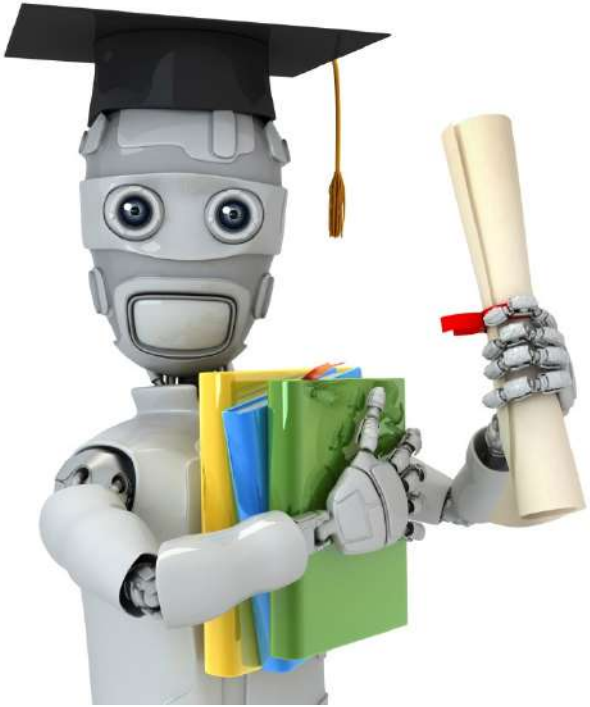
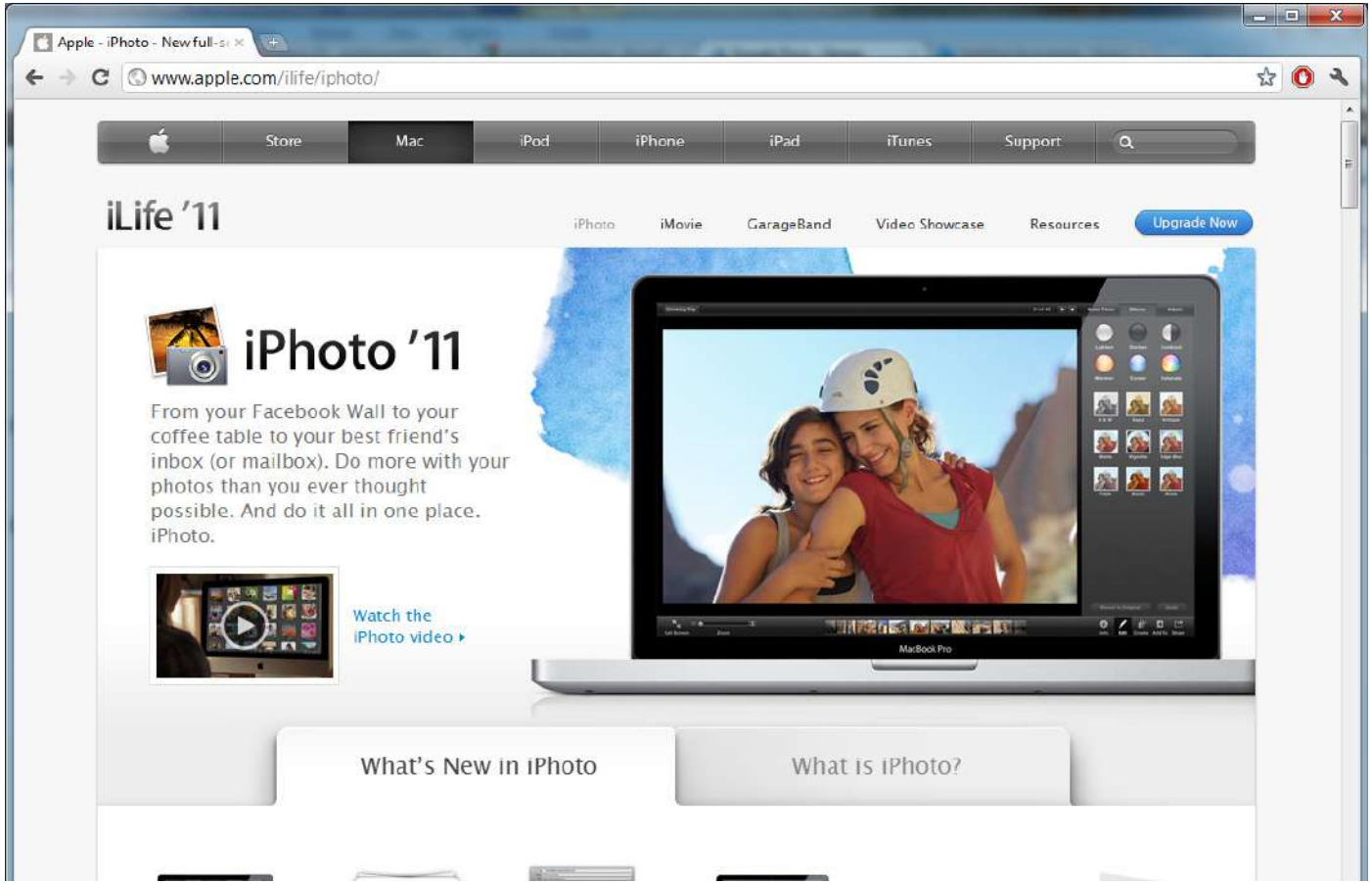


Introduction

Welcome

Machine Learning


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?

Andrew Ng

Andrew Ng

Machine Learning

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

Examples:

- Database mining

  Large datasets from growth of automation/web.

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

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

Machine Learning

- Grew out of work in AI
- |

Exam

- |

ıg

- /

ıost of

Machine Learning

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

Examples:

- Database mining

  Large datasets from growth of automation/web.

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

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

Machine Learning

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

Examples:

- Database mining

  Large datasets from growth of automation/web.

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

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

  E.g., Amazon, Netflix product recommendations

Machine Learning

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

Examples:

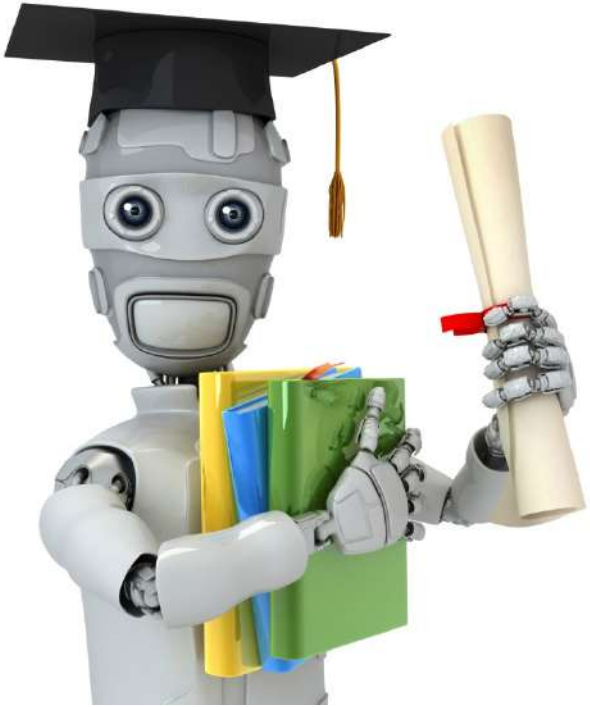
- Database mining

  Large datasets from growth of automation/web.

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

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

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



Machine Learning

Introduction

What is machine learning

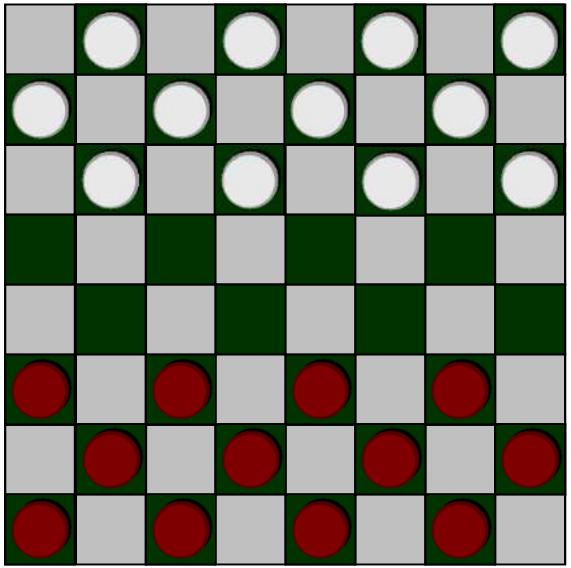
Machine Learning definition

Machine Learning definition

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

Machine Learning definition

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



Machine Learning definition

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

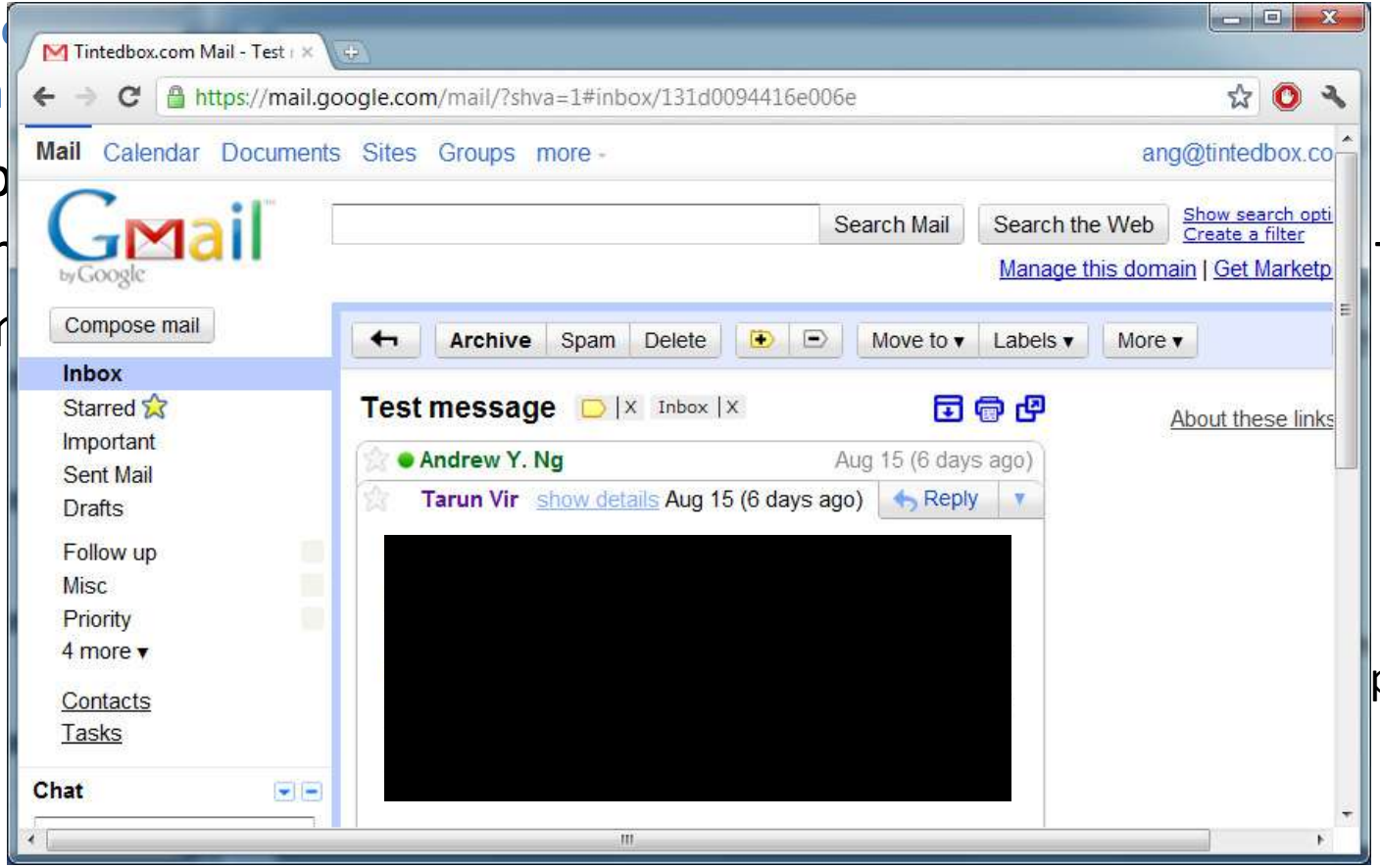
"A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

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

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

"A computer program is said to *learn* from experience E with respect to s on T, a

Supp or do not r filter spam pam.



"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?

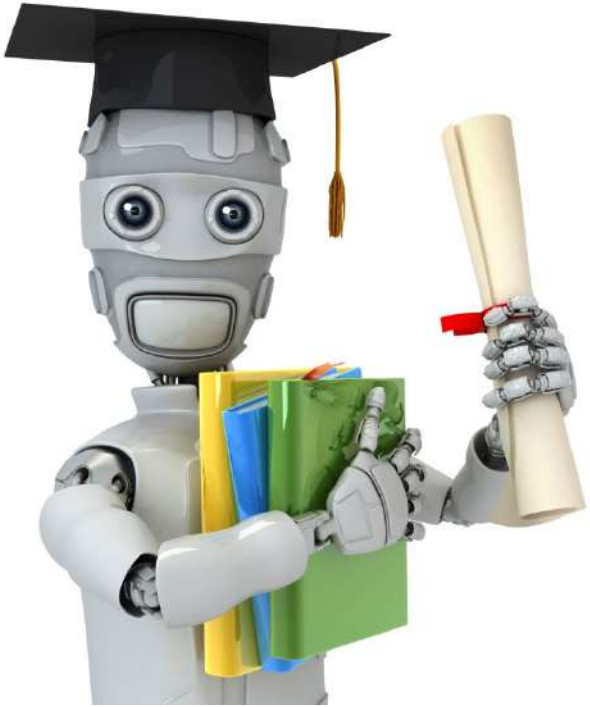
- Classifying emails as spam or not spam. T
- Watching you label emails as spam or not spam. E
- The number (or fraction) of emails correctly classified as spam/not spam. P
- None of the above—this is not a machine learning problem.

Machine learning algorithms:

- Supervised learning
- Unsupervised learning

Others: Reinforcement learning, recommender systems.

Also talk about: Practical advice for applying learning algorithms.

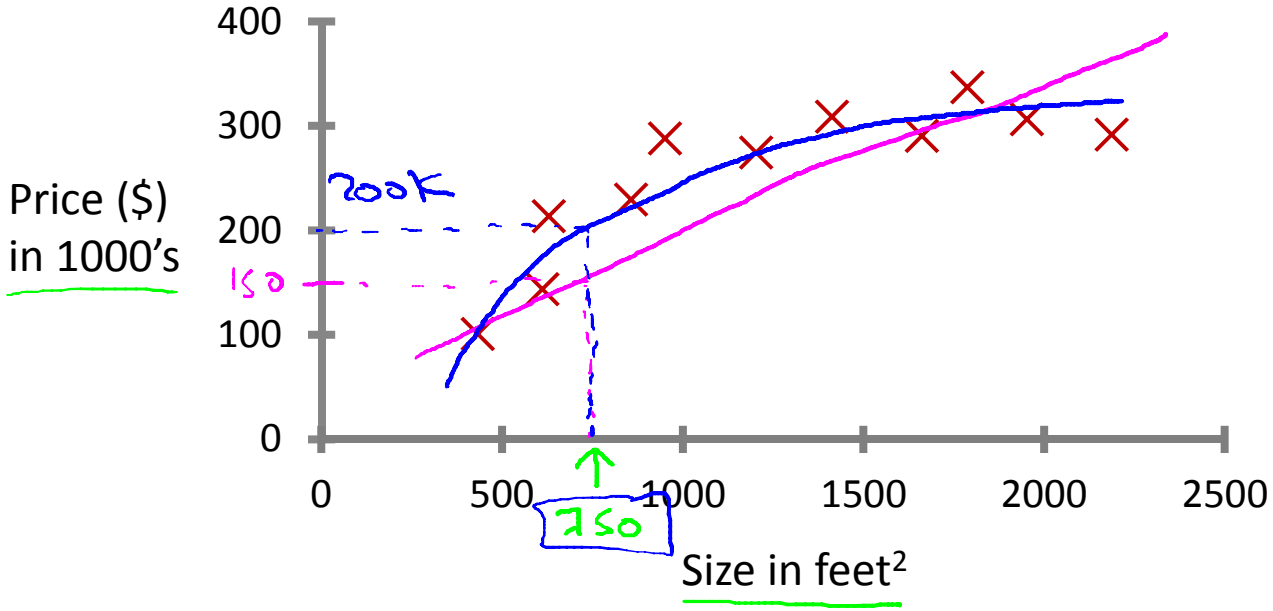


Machine Learning

Introduction

Supervised Learning

Housing price prediction.

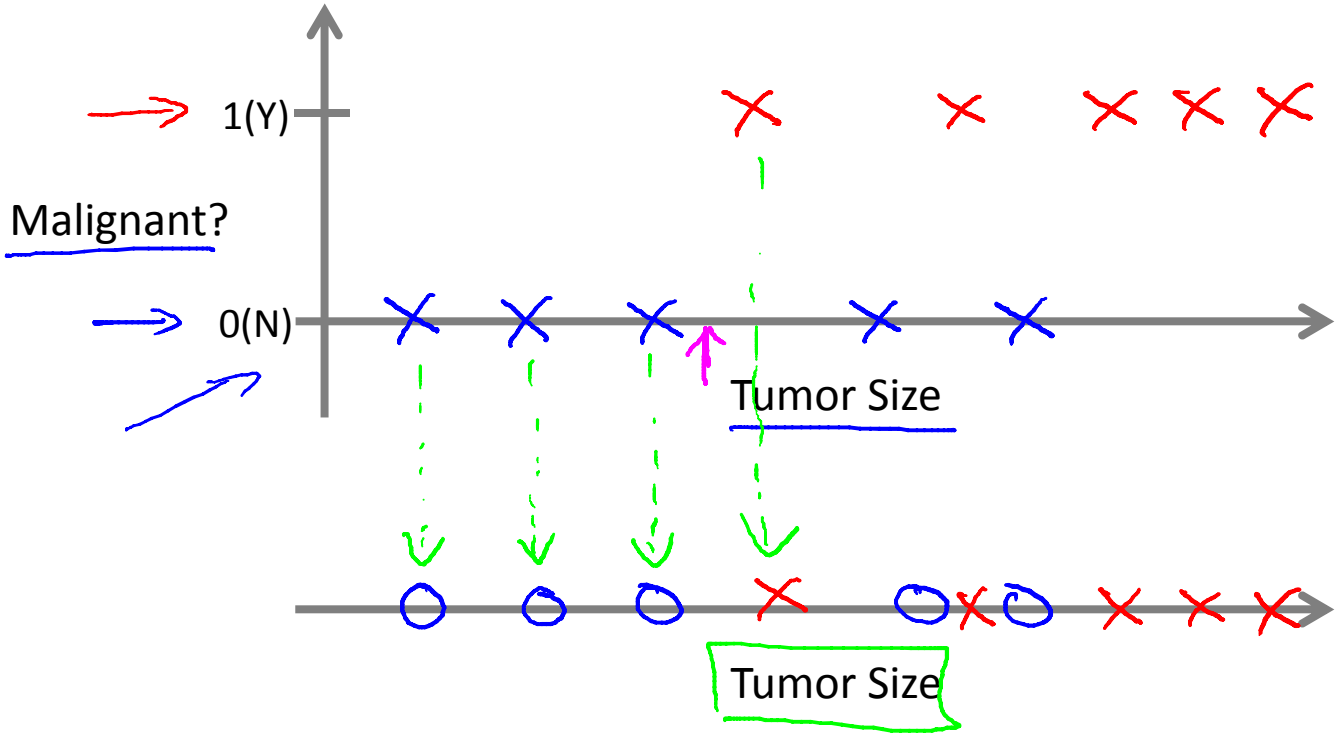


Supervised Learning

"right answers" given

Regression: Predict continuous valued output (price)

Breast cancer (malignant, benign)



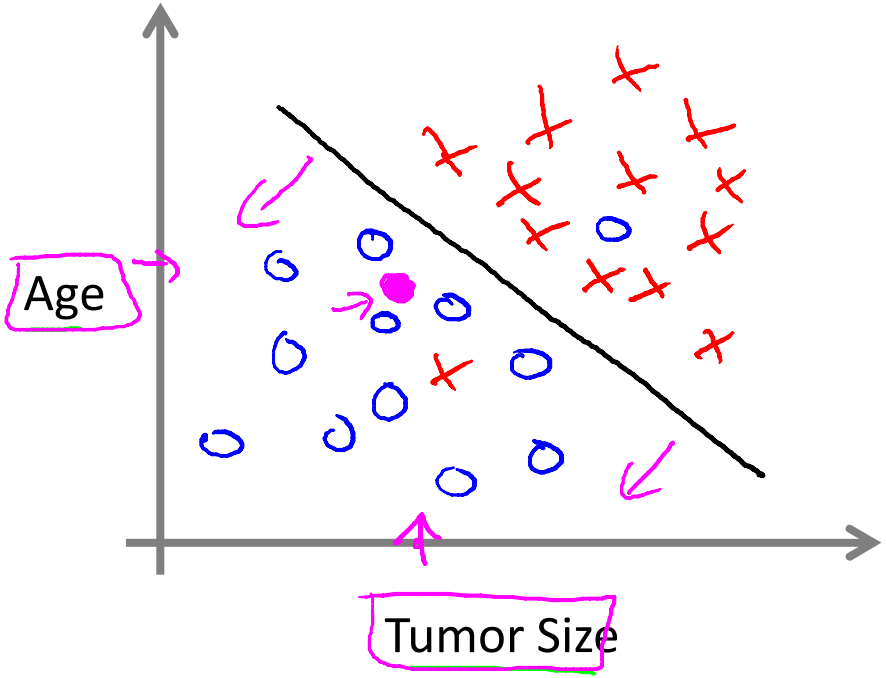
Classification

Discrete valued output (0 or 1)

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.

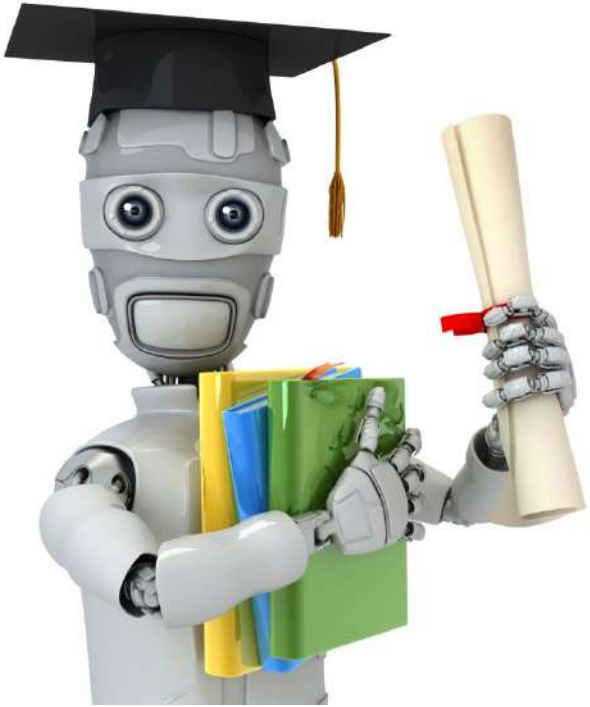
Problem 2: You'd like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised.

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.

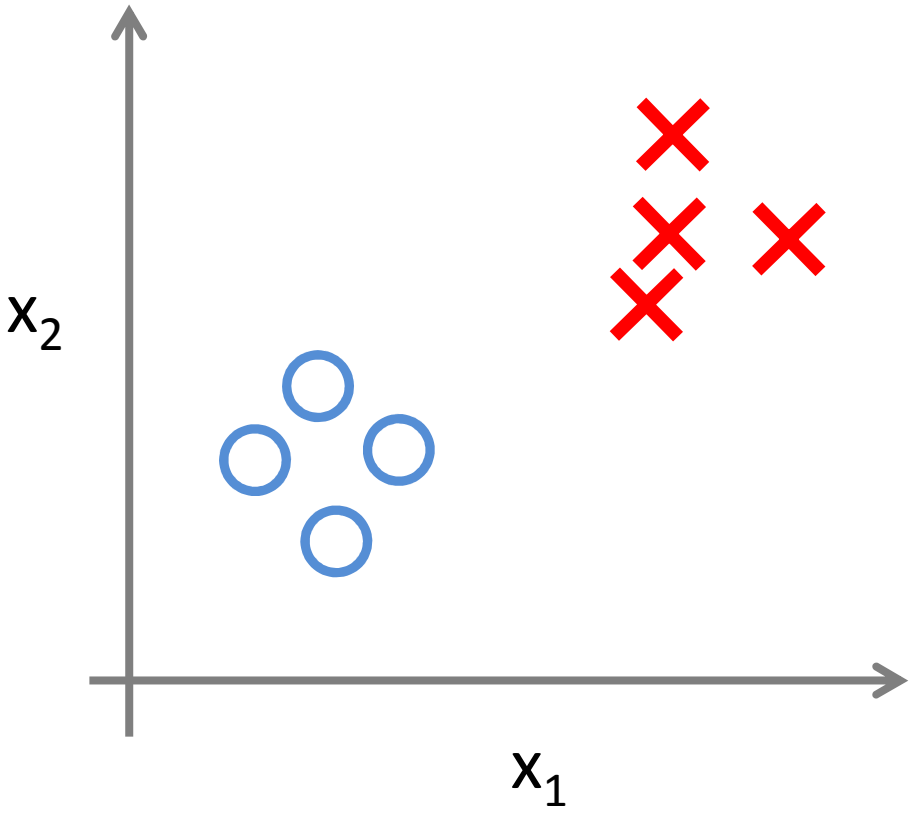


Machine Learning

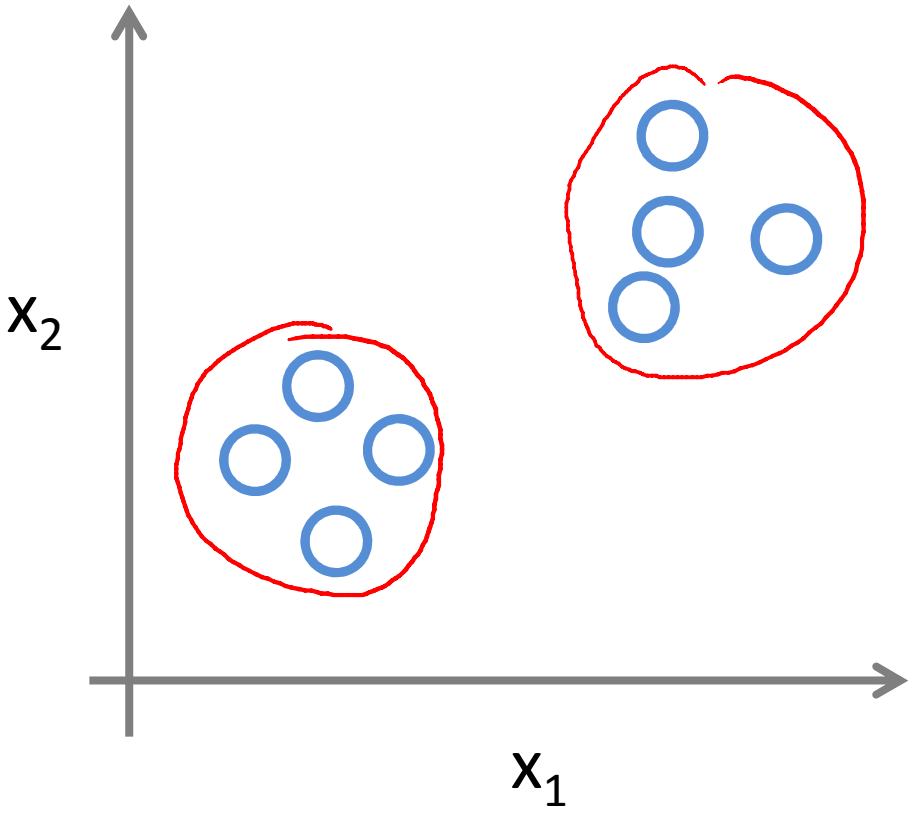
Introduction

Unsupervised Learning

Supervised Learning



Unsupervised Learning



Google News

news.google.com

Web Images Videos Maps News Shopping Gmail more ▼

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

Google news

Search News | Search the Web

Advanced news search

U.S. edition ▼ Add a section »

Top Stories

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

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

All news

- Headlines
- Images

Top Stories

Christine O'Donnell »

White House official denies Tea Party-focused ad campaign

CNN International - Ed Henry - 1 hour ago

Democratic sources say the White House is not considering an ad campaign tying Republicans to the Tea Party. Washington (CNN) -- A top White House official sharply denied a report that claims President Obama's political advisers are weighing a national ...

Tea Party is misplacing the blame, former President Bill Clinton claims New York Daily News

GOP tea party backer defends Christine O'Donnell The Associated Press

Atlanta Journal Constitution - Politics Daily - MyFox Washington DC - Salon

all 726 news articles »

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 »

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 »

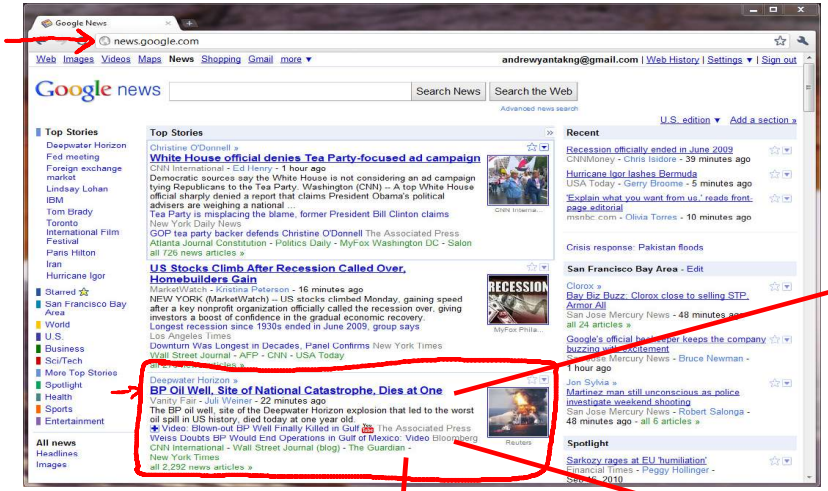
Google's official beekeeper keeps the company buzzing with excitement
San Jose Mercury News - Bruce Newman - 1 hour ago

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

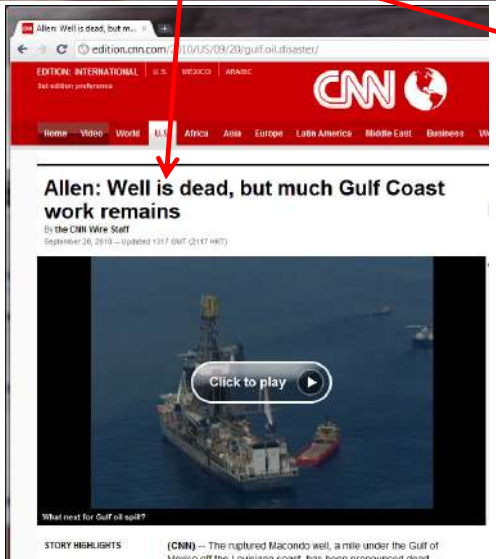
Spotlight

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

Andrew Ng

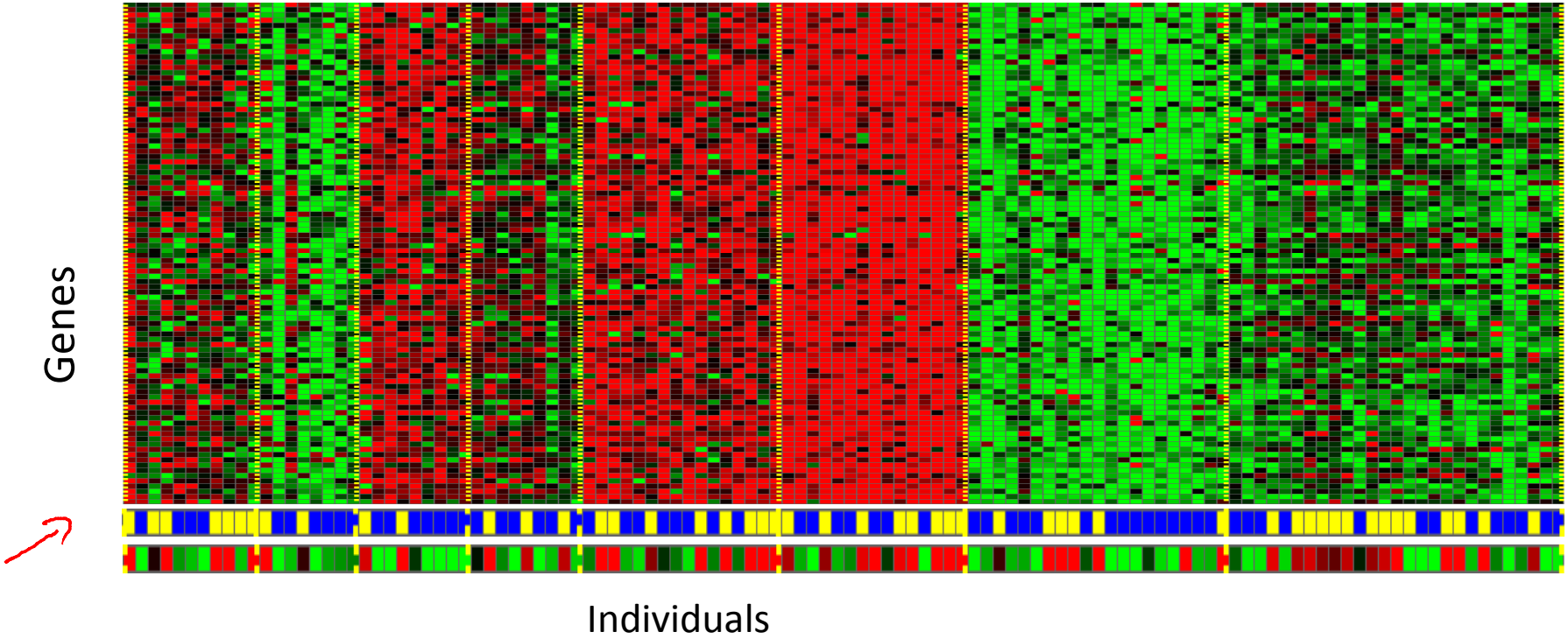

Google News
news.google.com
Web Images Videos Maps News Shopping Gmail more ▾
andrewyantakng@gmail.com | Web History | Settings ▾ | Sign out
Google news
Search News
Search the Web
Advanced news search
U.S. edition ▾ Add a section »
Top Stories
Deepwater Horizon
Fed meeting
Foreign exchange market
Lindsay 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 »
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 1,292 news articles »
Deepwater Horizon »
BP Oil Well, Site of National Catastrophe, Dies at One
Vanity Fair - Juli Weiner - 22 minutes ago
The BP oil well, site of the Deepwater Horizon explosion that led to the worst oil spill in US history, died today at one year old.
Video: Blown-out BP Well Finally Killed in Gulf The Associated Press
Weiss Doubts BP Would End Operations in Gulf of Mexico: Video Bloomberg
CNN International - Wall Street Journal (blog) - The Guardian - New York Times
all 2,292 news articles »
Recent
Recession officially ended in June 2009
CNNMoney - Chris Isidore - 39 minutes ago
Hurricane Igor lashes Bermuda
USA Today - Gerry Broome - 5 minutes ago
'Explain what you want from us,' reads front-page editorial
msnbc.com - Olivia Torres - 10 minutes ago
Crisis response: Pakistan floods
San Francisco Bay Area - Edit
Clorox »
Bay Biz Buzz: Clorox close to selling STP, Armor All
San Jose Mercury News - 48 minutes ago
all 24 articles »
Google's official here's per keeps the company buzzing with excitement
San Jose Mercury News - Bruce Newman - 1 hour ago
Jon Sykes »
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 20, 2010
RECESSION
MyFox Philadelphia
Reuters

BP Kills Macondo, But Its ...
blogs.wsj.com/source/2010/09/20/bp-kills-macondo-but-its-legacy-lives-on/
THE WALL STREET JOURNAL.
THE SOURCE
Financial Services
Transport
Leisure
Insurance
Oil & Gas
Sport
Caught on the Web
Betting
Technology
BP Kills Macondo, But Its Legacy Lives On
Article
Comments (2)
By James Herron
BP confirmed late Sunday that the Macondo well that leaked almost five million barrels of oil into the Gulf of Mexico has been permanently sealed, but the well will continue to affect BP and the wider oil industry for many years.
The most immediate worry for BP and its shareholders is how the authorities will apportion blame for the spill. BP's own investigation
About The Source
Most Recent
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

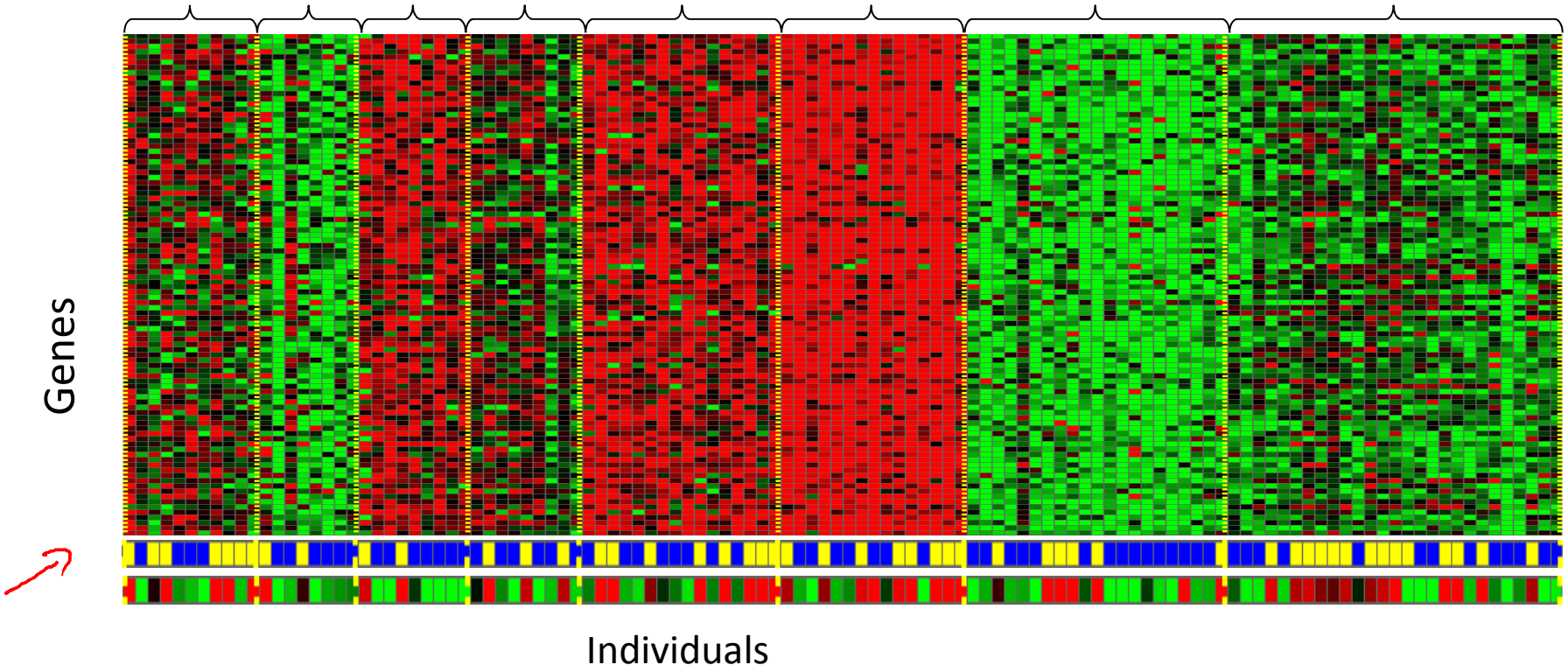

Allen: Well is dead, but m...
edition.cnn.com/2010/US/09/20/gulf.oil.disaster/
EDITION: INTERNATIONAL
U.S.
MEXICO
ARABIC
CNN
Home
Video
World
U.S.
Africa
Asia
Europe
Latin America
Middle East
Business
Allen: Well is dead, but much Gulf Coast work remains
By the CNN Wire Staff
September 20, 2010 -- Updated 1317 GMT (2117 HKT)
Click to play
What next for Gulf oil spill?
STORY HIGHLIGHTS
(CNN) -- The ruptured Macondo well, a mile under the Gulf of

BP oil spill cost hits nearl...
www.guardian.co.uk/environment/2010/sep/20/bp-oil-spill-dee
Mobile site
Sign in
Register
Text larger
smaller
About us
guardian.co.uk
News
Sport
Comment
Culture
Business
Money
Life & style
Business
BP
BP oil spill cost hits nearly $10bn
BP has set up a $20bn compensation fund after the Deepwater Horizon disaster, which has so far paid out 19,000 claims totalling more than $240m
Julia Kollewe
guardian.co.uk, Monday 20 September 2010 08.33 BST
Article history

Andrew Ng

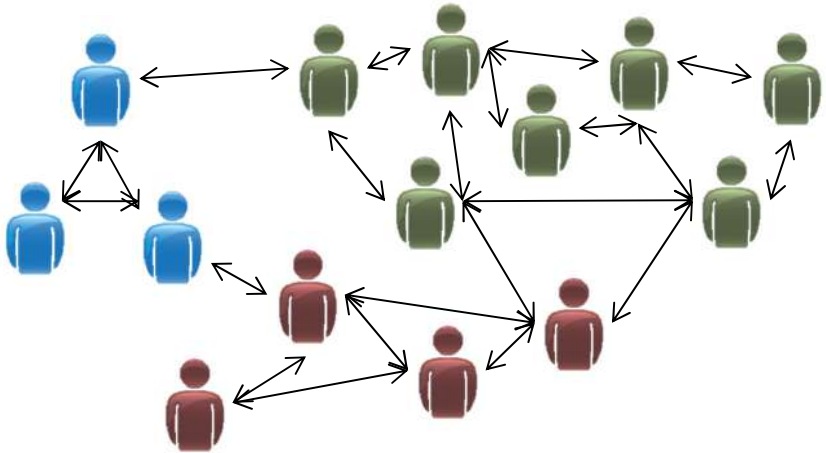


[Source: Daphne Koller]

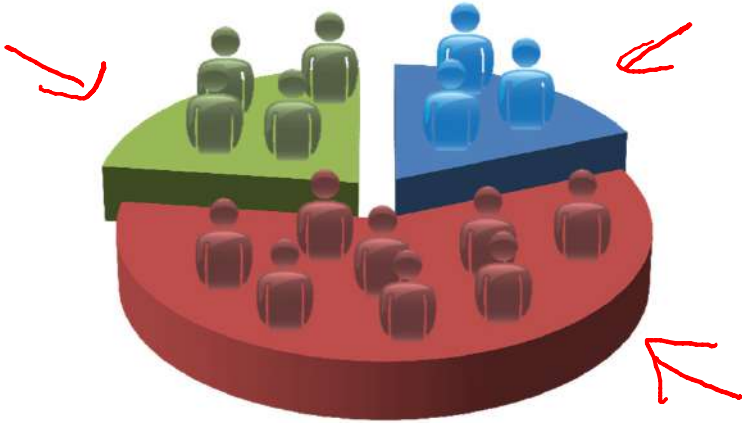


[Source: Daphne Koller]

Organize computing clusters



Social network analysis

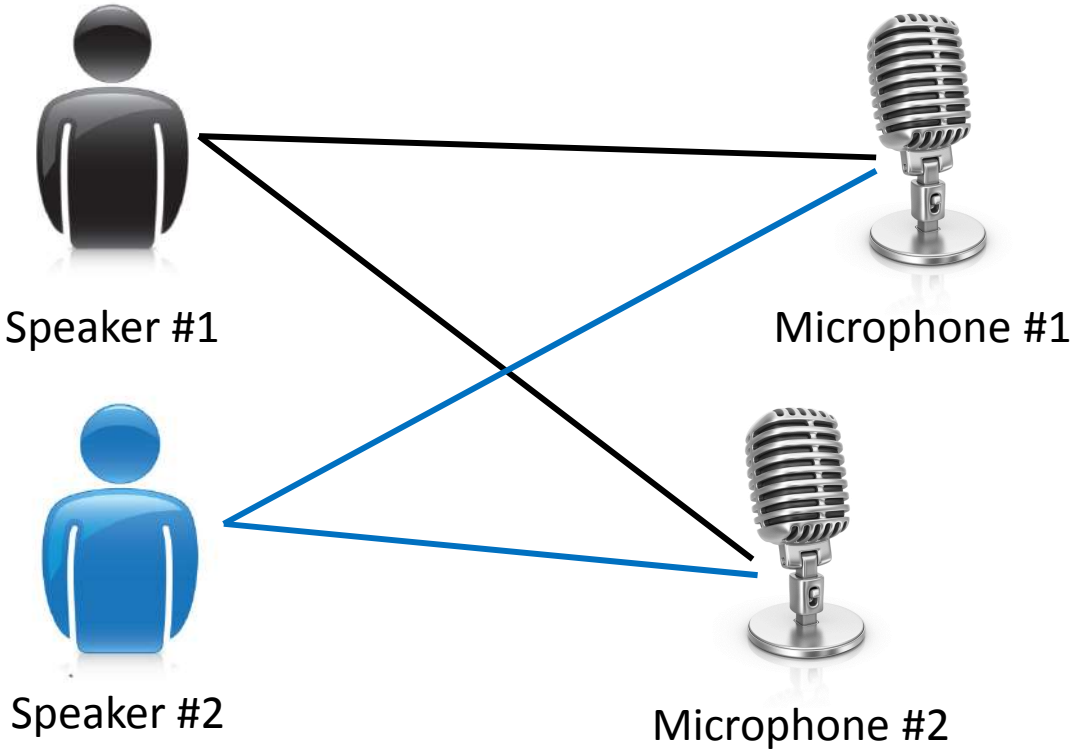


Market segmentation

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.]

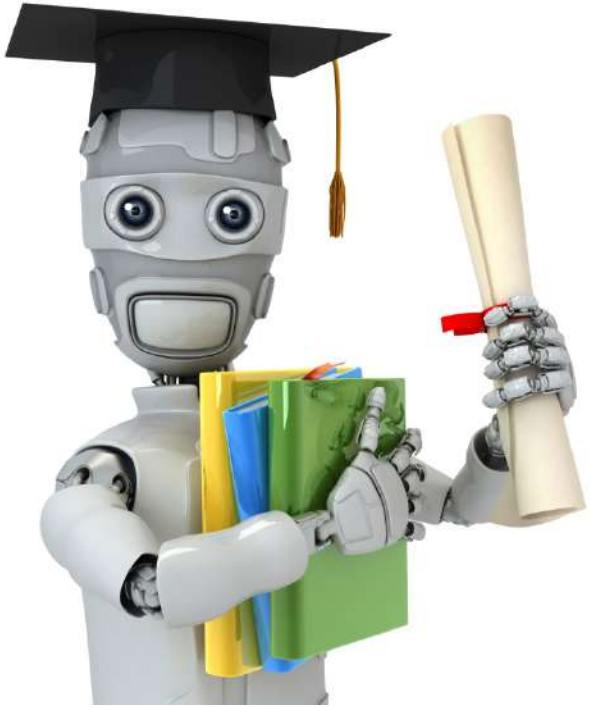
Cocktail party problem algorithm

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

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

Of the following examples, which would you address using an <u>unsupervised</u> 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

Logistic Regression

Classification

Classification

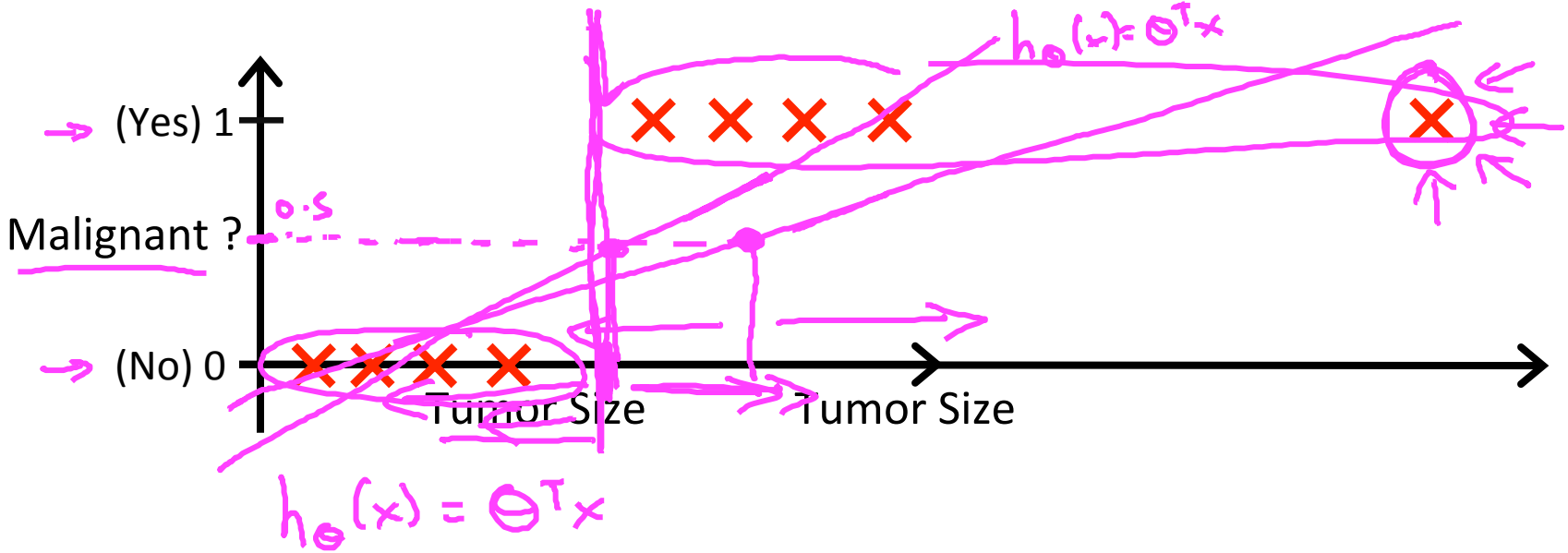
- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign ?

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

0: "Negative Class" (e.g., benign tumor)

1: "Positive Class" (e.g., malignant tumor)

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



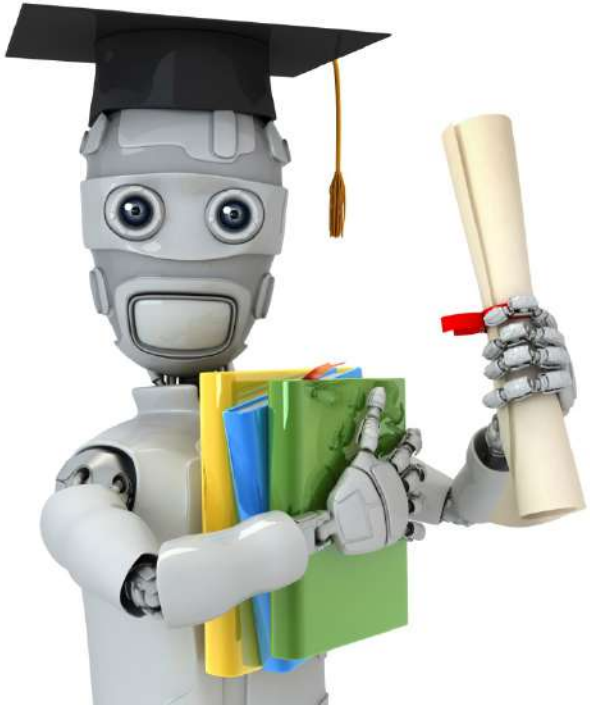
- Threshold classifier output $h_\theta(x)$ at 0.5:
  - If $h_\theta(x) \geq 0.5$, predict "y = 1"
  - If $h_\theta(x) < 0.5$, predict "y = 0"

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

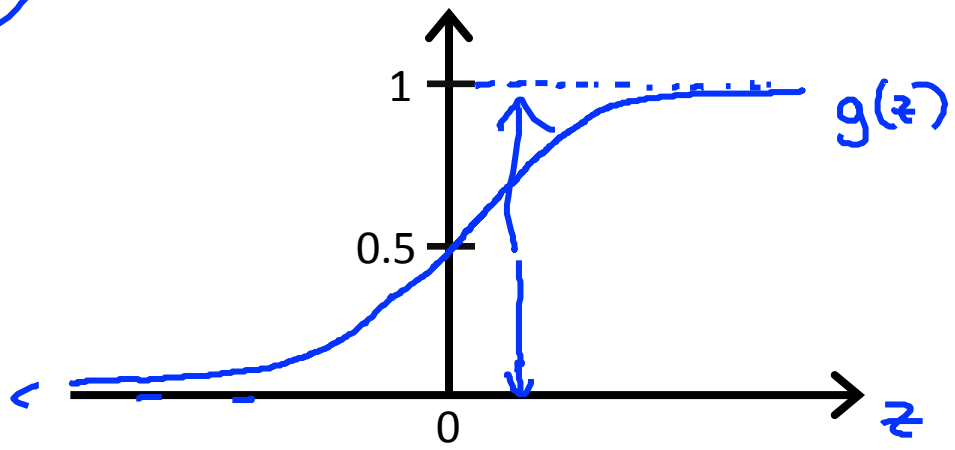
Want $\boxed{0 \le h_\theta(x) \le 1}$

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

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

$\theta^T x$

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



Parameters θ.

Sigmoid function

Logistic function

Interpretation of Hypothesis Output

$h_\theta(x)$

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

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

$$h_\theta(x) = 0.7$$

$y=1$

Tell patient that 70% chance of tumor being malignant

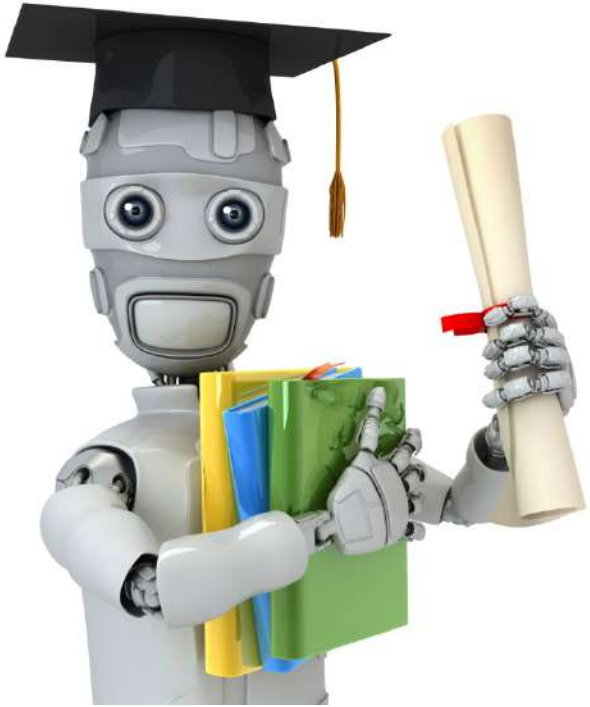
$h_\theta(x) = P(y=1|x;\theta)$

$y = 0$ or 1

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

$$P(y=0|\text{[redacted]}) + \boxed{P(y=1|\text{[redacted]})} = 1$$

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



Machine Learning

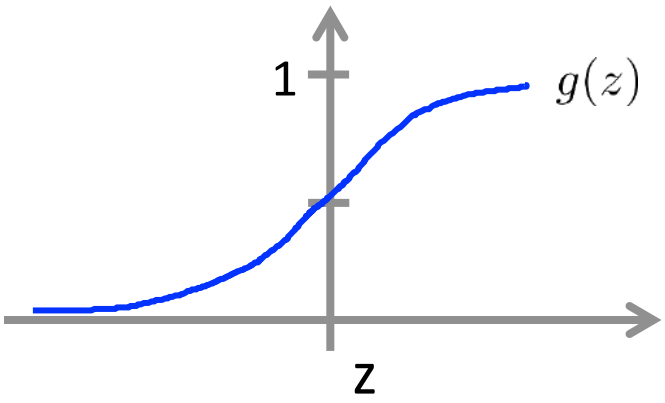
Logistic Regression

Decision boundary

Logistic regression

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

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



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

$\theta^T x \geq 0$

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

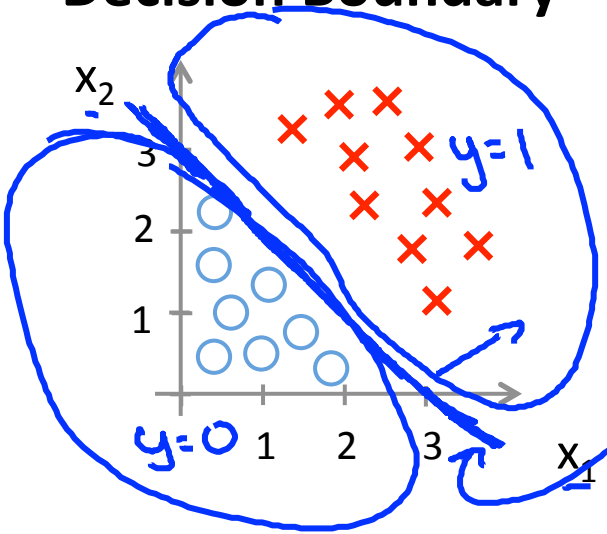
$h_\theta(x) = g(\theta^T x$

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

$\theta^T x < 0$

$g(z) < 0.5$
when $z < 0$

Decision Boundary



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

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

$\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

Decision boundary

Predict "$y = 1$" if $\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0$

$x_1 + x_2 \geq 3$

x_1, x_2

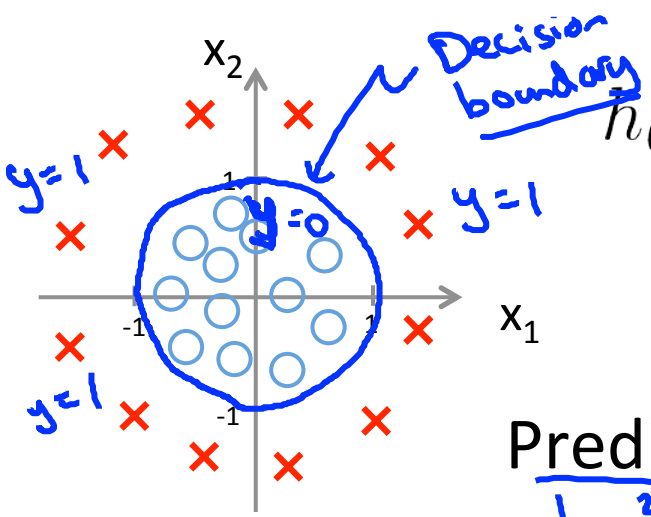
$h_\theta(x) = 0.5$

$\boxed{x_1 + x_2 = 3}$

$x_1 + x_2 < 3$

$y = 0$

Non-linear decision boundaries



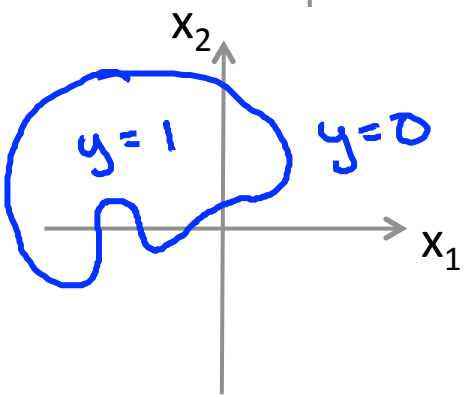
$$h_\theta(x) = g(\overset{=-1}{\theta_0} + \overset{=0}{\theta_1 x_1} + \overset{=0}{\theta_2 x_2} + \theta_3 x_1^2 + \theta_4 x_2^2)$$

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

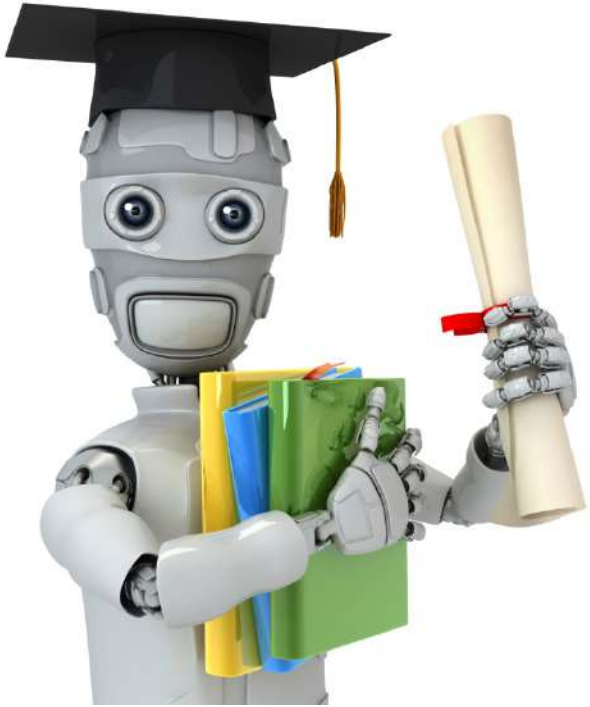
Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

$\boxed{x_1^2 + x_2^2 = 1}$

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



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



Machine Learning

Logistic Regression

Cost function

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

m examples

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

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

How to choose parameters θ ?

Cost function

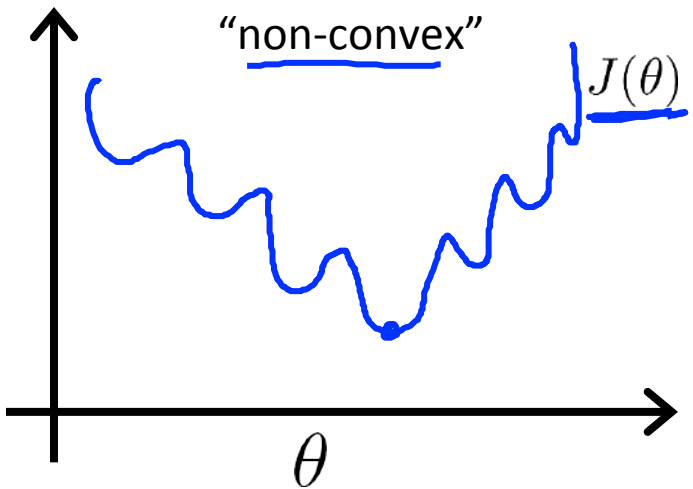
~~Linear~~ regression: $J(\theta) = \boxed{\frac{1}{m}\sum_{i=1}^{m} \cancel{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}}$

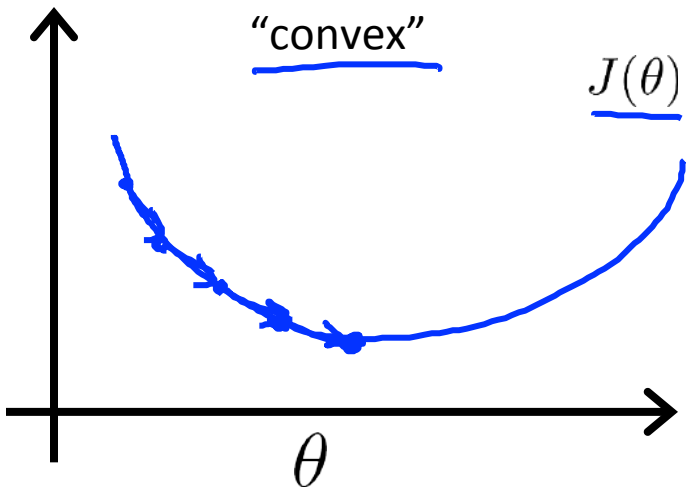
Logistic

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

$$\mathrm{Cost}(h_\theta(x^{[redacted]}), y^{[redacted]}) = \boxed{\frac{1}{2}\left(h_\theta(x^{[redacted]}) - y^{[redacted]}\right)^2}$$

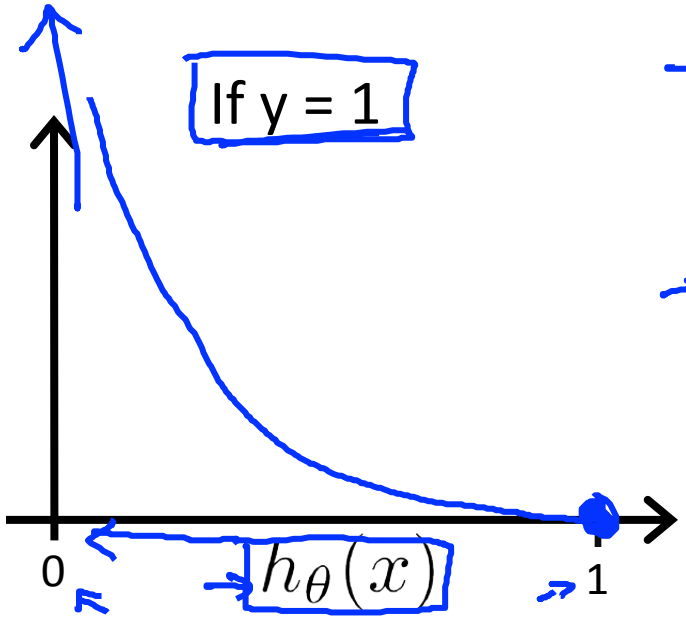
$$\boxed{\frac{1}{1+e^{-\theta^T x}}}$$





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 $h_\theta(x) \to 0$

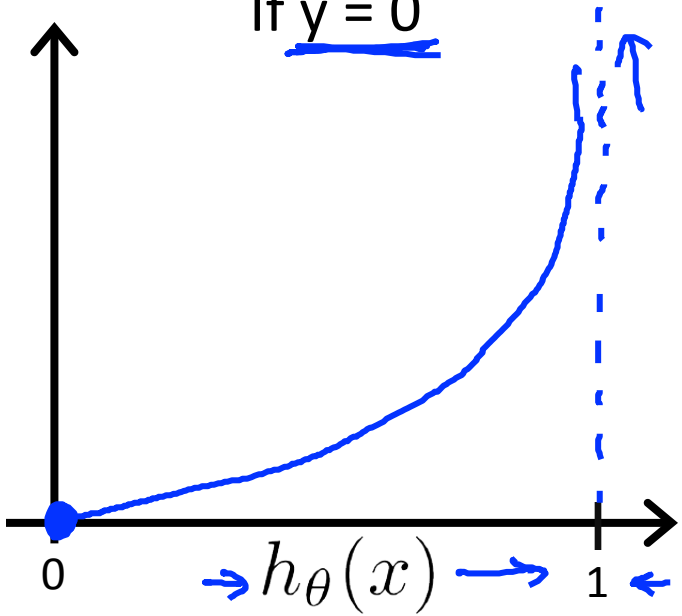
$Cost \to \infty$

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.

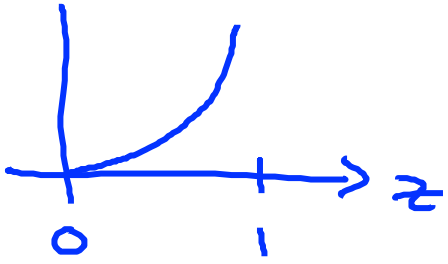
Logistic regression cost function

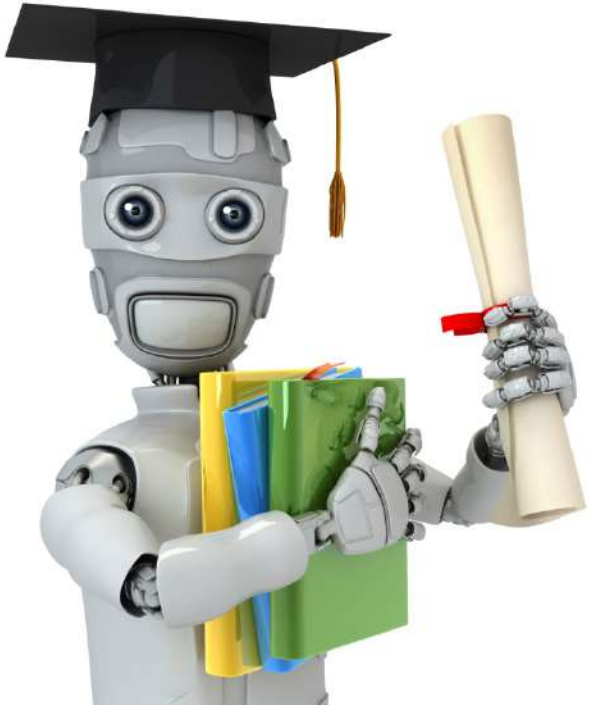
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ \boxed{-\log(1 - h_\theta(x))} & \text{if } y = 0 \end{cases}$$

If y = 0



$-\log(1-z)$





Logistic Regression

Simplified cost function and gradient descent

Machine Learning

Logistic regression cost function

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

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

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

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

$y = 1$, $(1-y) = 0$; $y = 0$, $(1-y) = 1$

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

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 :

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

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

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

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

Repeat {

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

}

(simultaneously update all θ_j)

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

Gradient Descent

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

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

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

for $i = 0$ to n

Repeat {

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

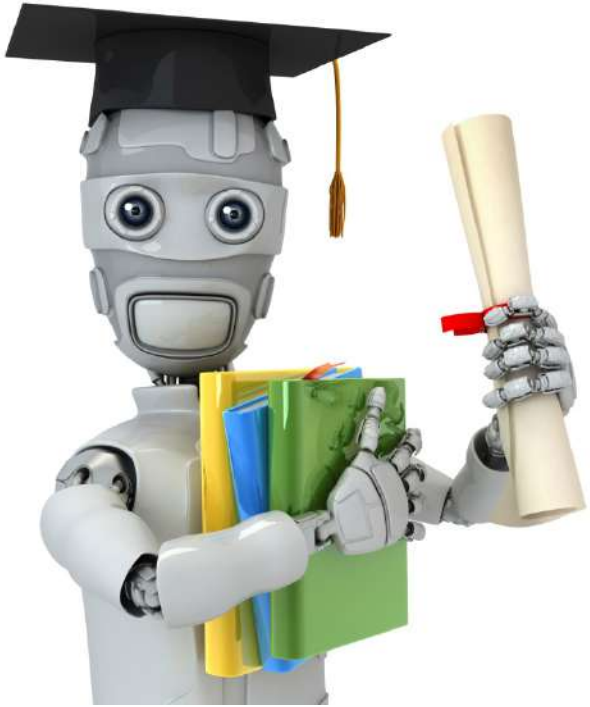
(simultaneously update all θ_j)

}

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

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

Algorithm looks identical to linear regression!



Machine Learning

Logistic Regression

Advanced optimization

Optimization algorithm

Cost function $J(\theta)$. 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 $\{$

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

$\}$

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);
```

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

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

theta(1)

theta(2)

theta(n+1)

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

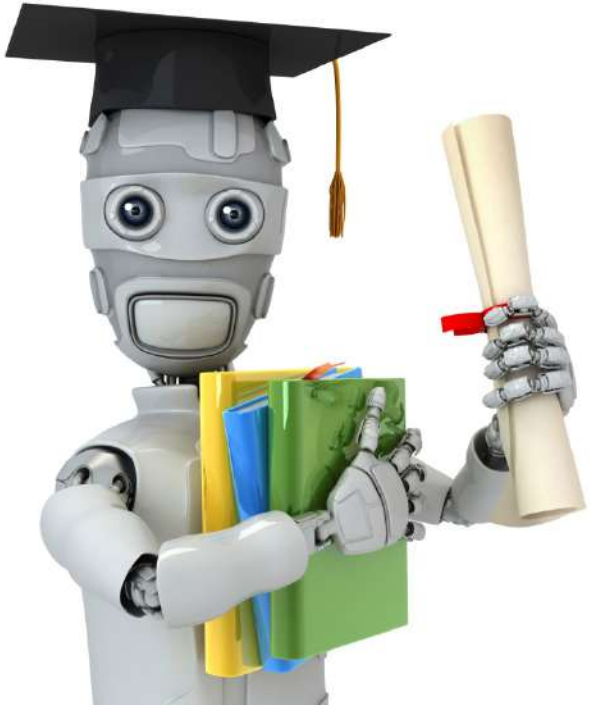
jVal = [code to compute $J(\theta)$];

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

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

$\vdots$

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

$y=1$ $y=2$ $y=3$ $y=4$

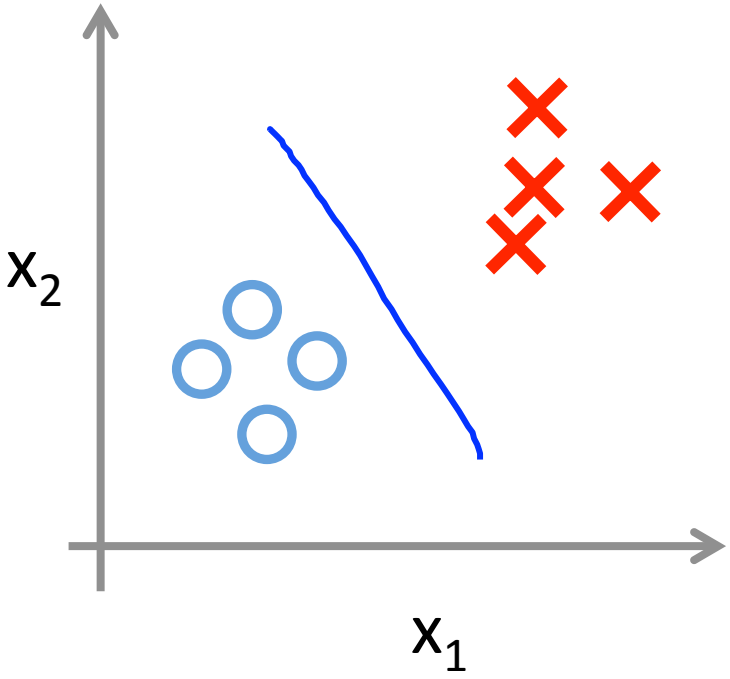
Medical diagrams: Not ill, Cold, Flu

$y=1$ 2 3

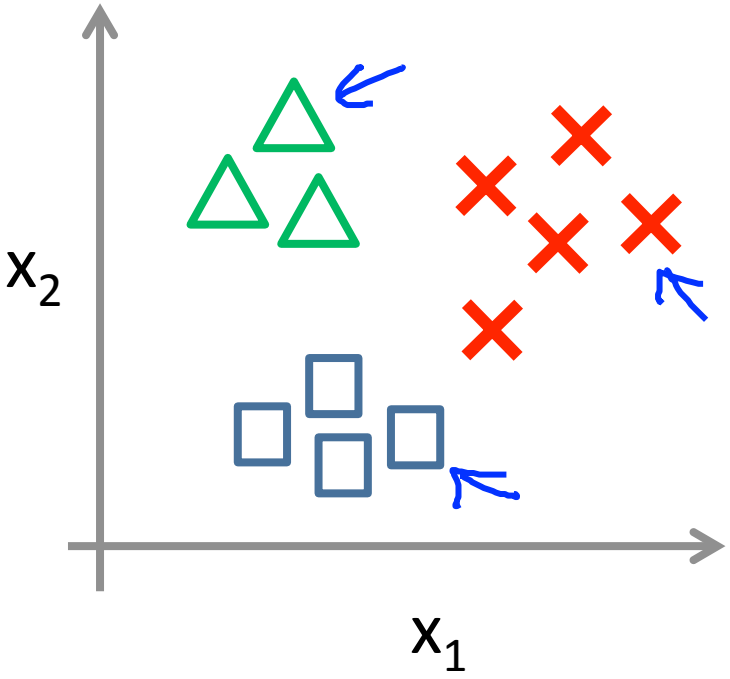
Weather: Sunny, Cloudy, Rain, Snow

$y=1$ 2 3 4

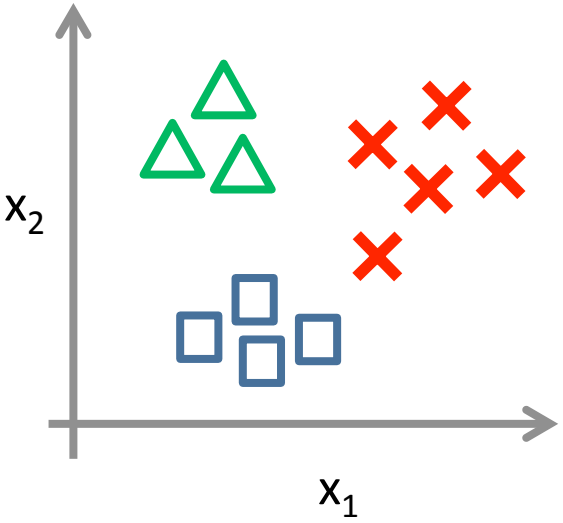
Binary classification:

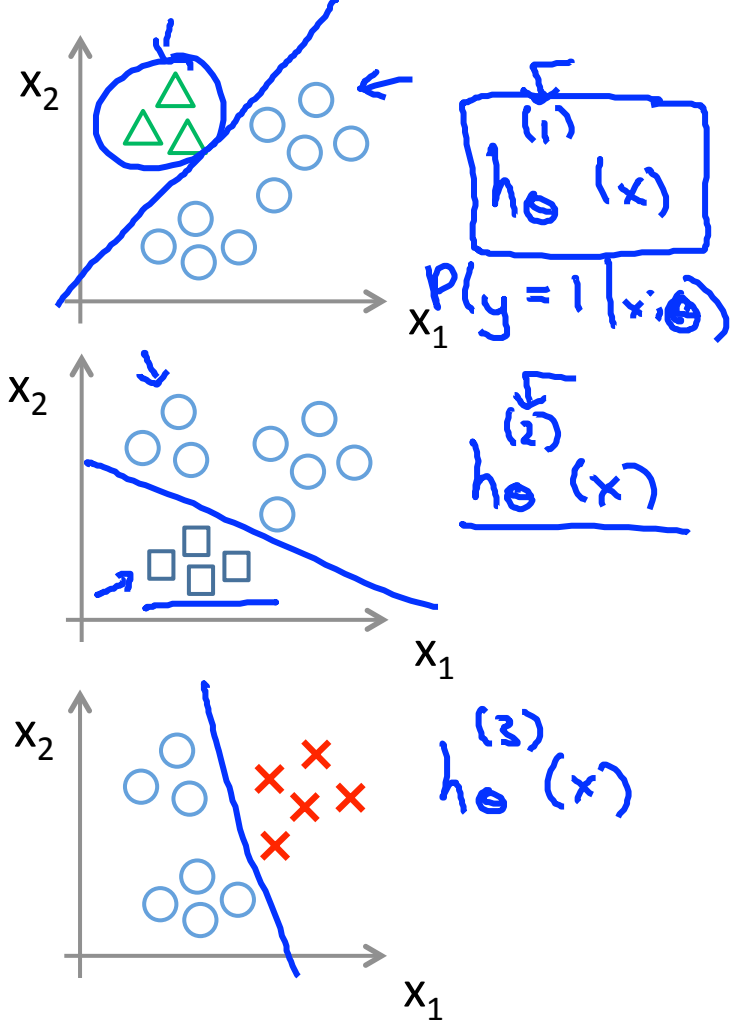


Multi-class classification:



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





Class 1: △

Class 2: □

Class 3: ✕

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

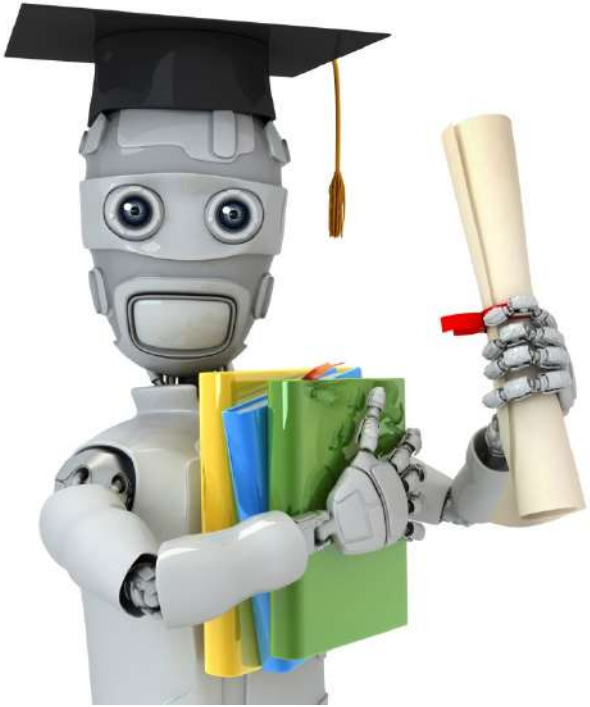
Andrew Ng

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

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

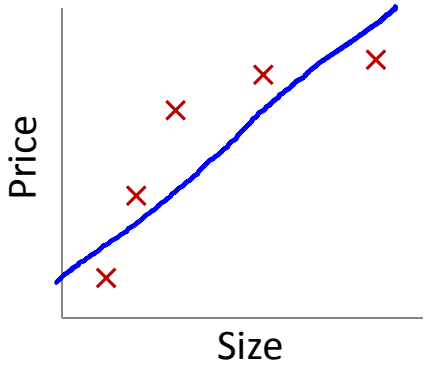


Machine Learning

Regularization

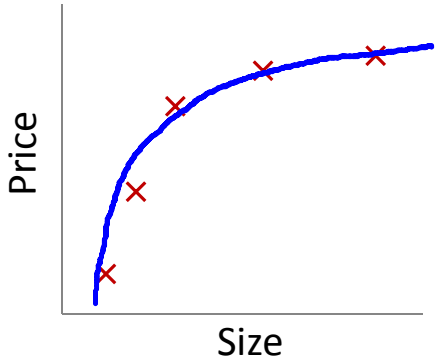
The problem of overfitting

Example: Linear regression (housing prices)



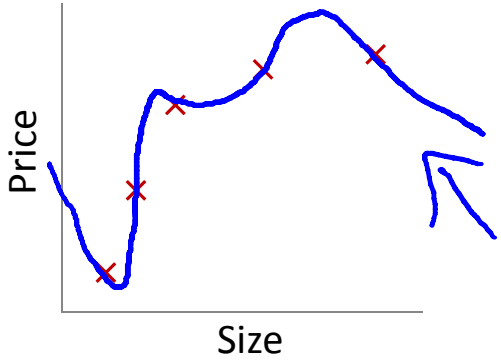
$\rightarrow \theta_0 + \theta_1 x$

"Underfit" "High bias"



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

"Just right"

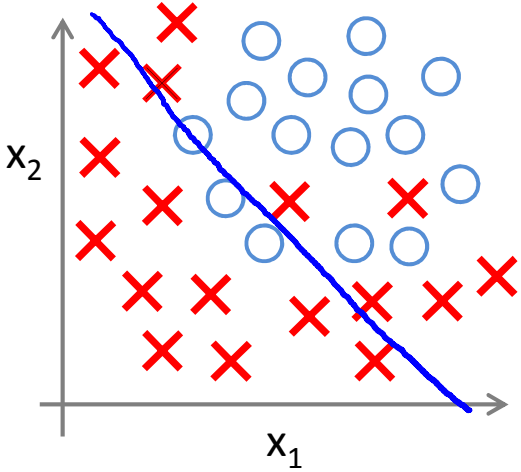


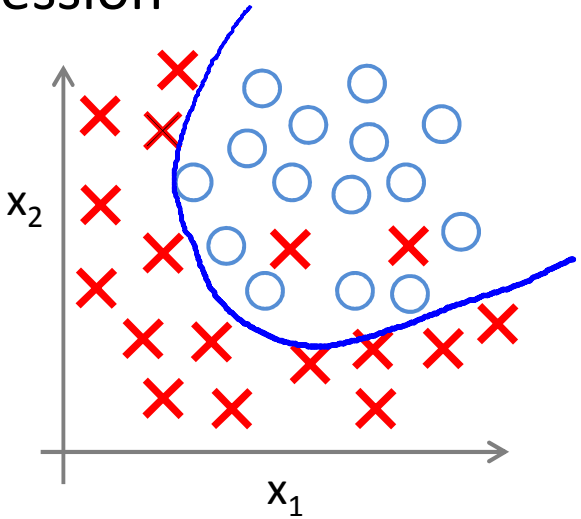
$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

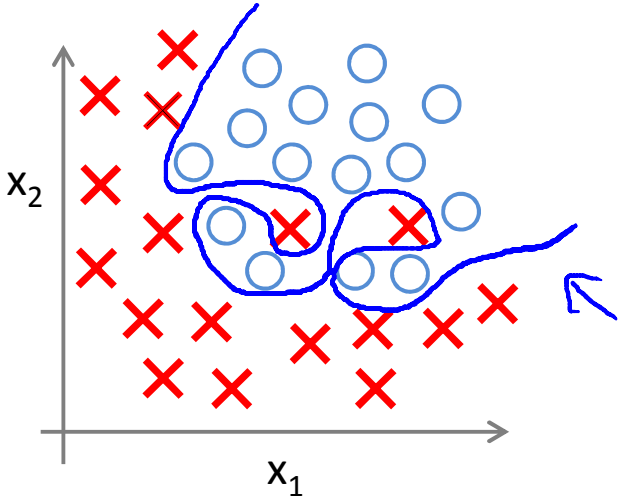
"Overfit" "High variance"

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

Example: Logistic regression







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

(g = sigmoid function)

"Underfit"

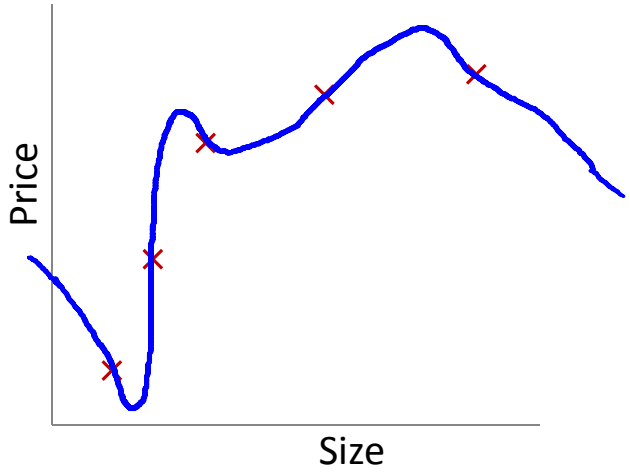
$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$

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

"Overfit"

Addressing overfitting:

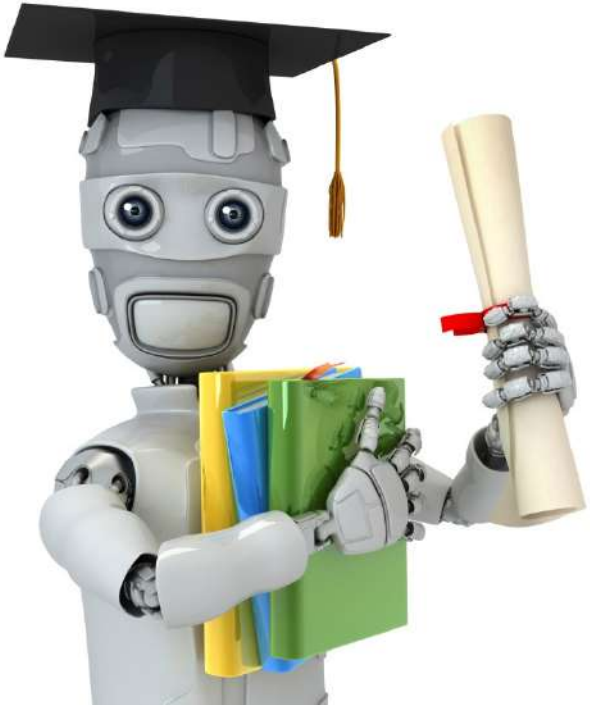
$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}



Addressing overfitting:

Options:

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

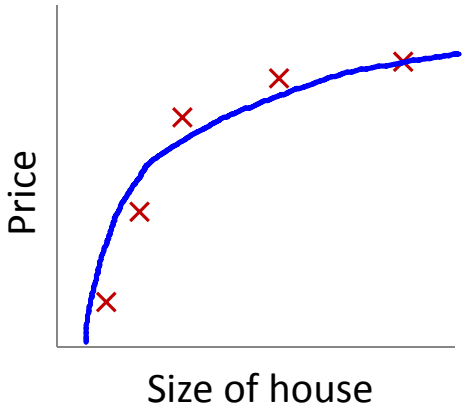


Machine Learning

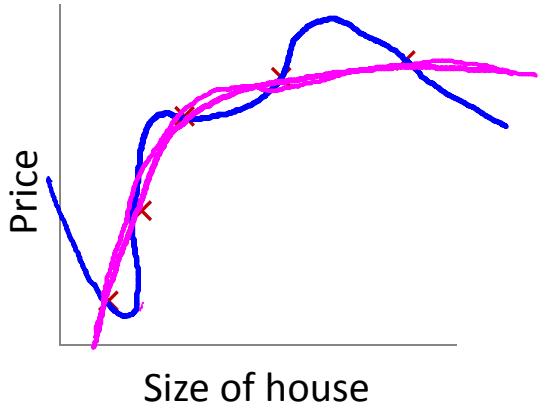
Regularization

Cost function

Intuition



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



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

Suppose we penalize and make θ_3, θ_4 really small.

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

$\theta_3 \approx 0$ $\quad$ $\theta_4 \approx 0$

Regularization.

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

- "Simpler" hypothesis
- Less prone to overfitting

$\boxed{\theta_3, \theta_4} \approx 0$

Housing:

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

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

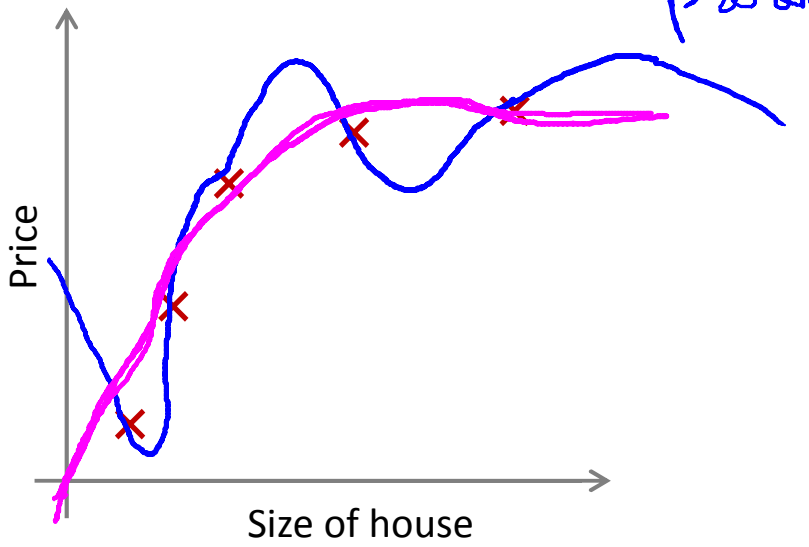
$\cancel{\theta_0}$ $\theta_1, \theta_2, \theta_3, \ldots, \theta_{100}$ $\cancel{\theta_0}$

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

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

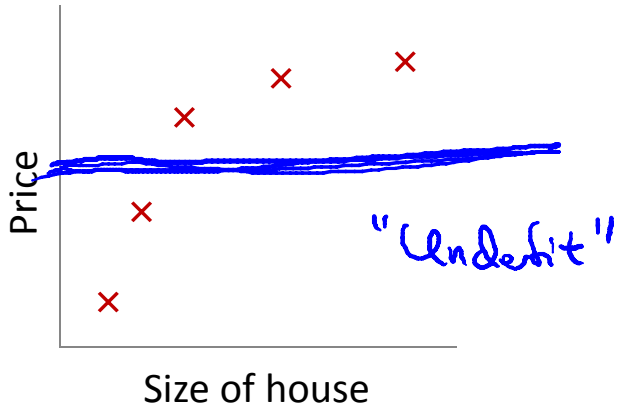
What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

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

In regularized linear regression, we choose θ to minimize

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

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



$h_\theta(x)$

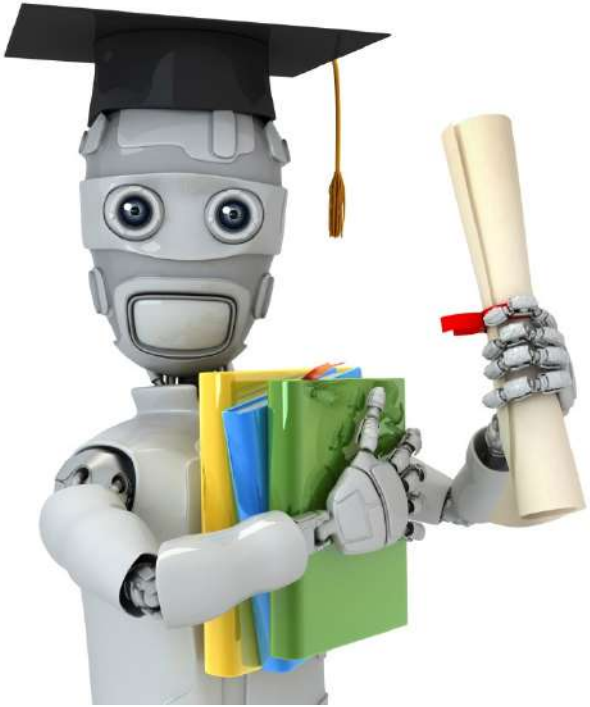
$$\theta_0 + \cancel{\theta_1 x} + \cancel{\theta_2 x^2} + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

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

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

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

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



Machine Learning

Regularization

Regularized linear regression

Regularized linear regression

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

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

Gradient descent

θ_0 $\theta_1, \theta_2, \ldots, \theta_n$

Repeat {

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

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

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

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

}

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

$\rightarrow J(\theta)$

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

0.99

$\theta_j \times 0.99$

$\boxed{\theta_j^{\text{I}}}$

Andrew Ng

Normal equation

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

$m \times (n+1)$

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

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

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

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

Non-invertibility (optional/advanced).

Suppose $m \leq n$,

(#examples) (#features)

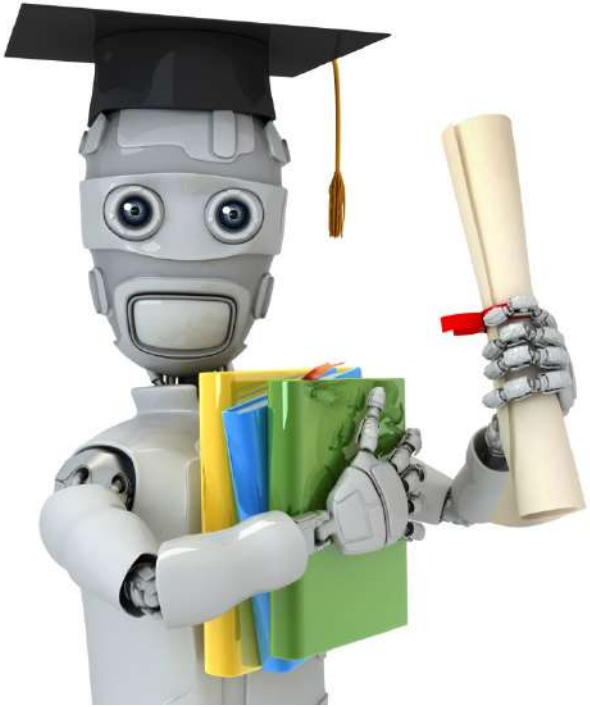
$$\theta = \underbrace{(X^T X)}_{\text{non-invertible / singular}}{}^{-1} X^T y$$

pinv

inv

If $\lambda > 0$,

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

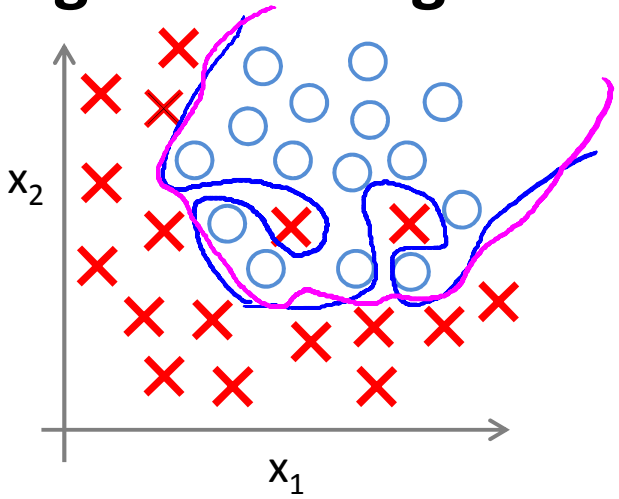


Machine Learning

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:

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

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

Gradient descent

Repeat {

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

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

$$(j = \cancel{0}, \underbrace{1, 2, 3, \ldots, n}_{\theta_1 \ldots \theta_n})$$

}

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

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 $J(\theta)$];

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

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

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

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

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

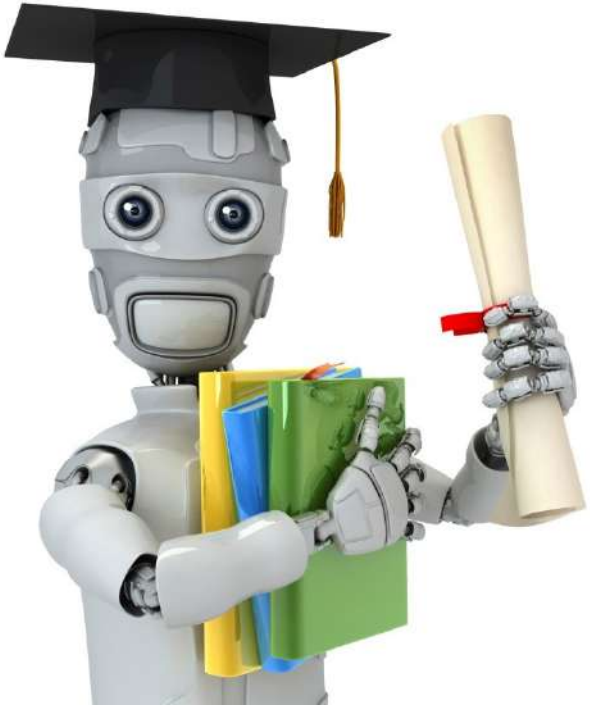
$J(\theta)$

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

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

⋮

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

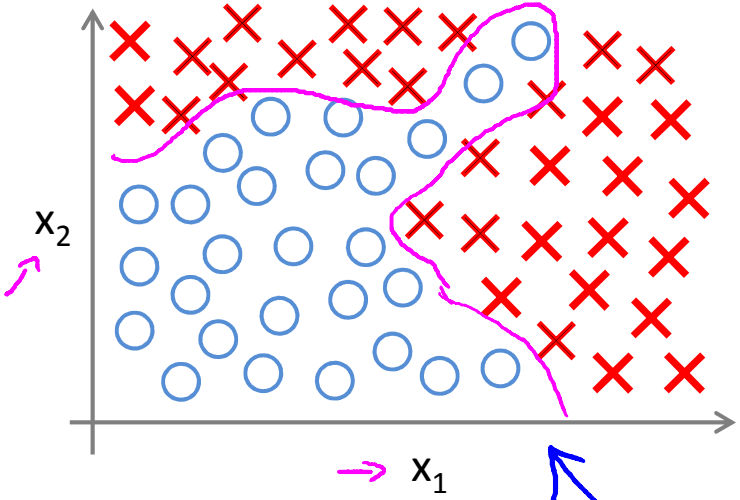


Machine Learning

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 feature $\quad O(n^2)$

$\approx \frac{n^2}{2}$

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

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

$O(n^3)$

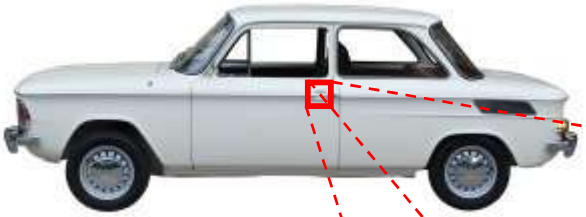
$170{,}000$

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

$n = 100$

What is this?

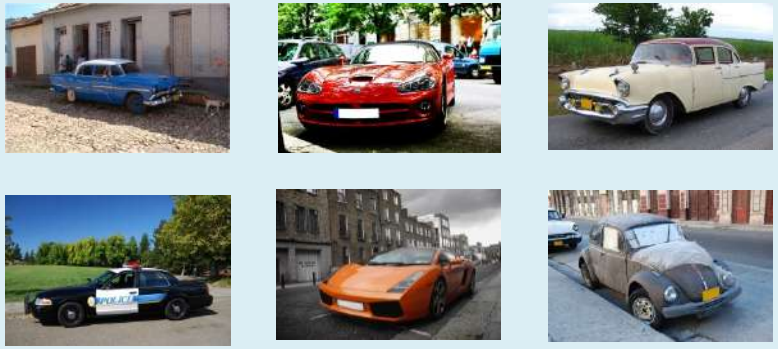
You see this:



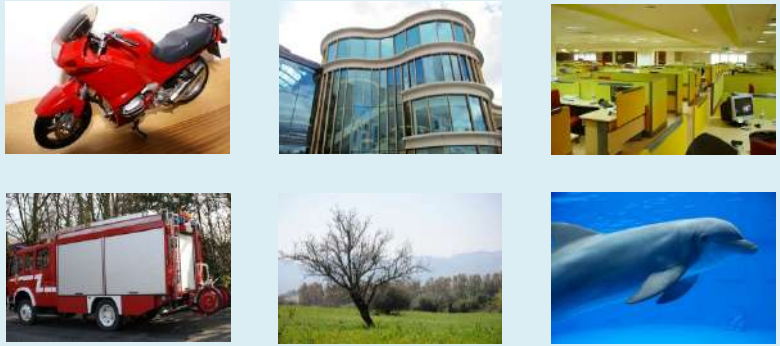
But the camera sees this:

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



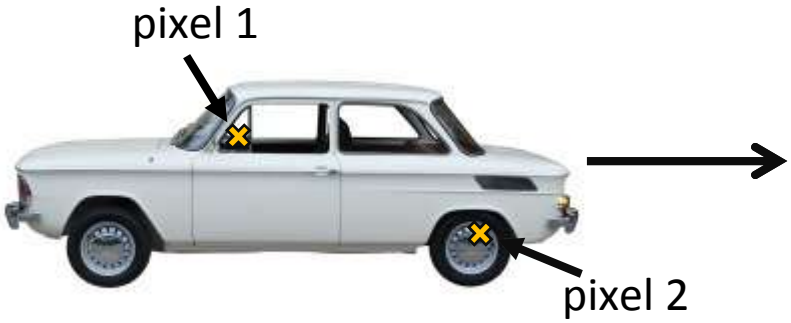
Cars



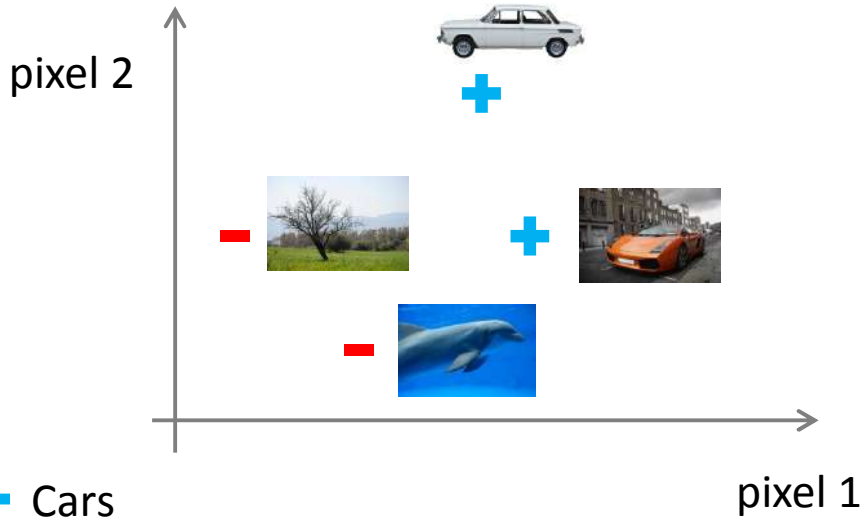
Not a car

Testing:

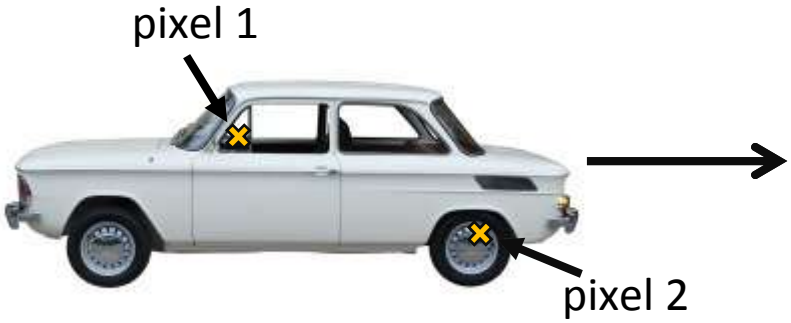
What is this?



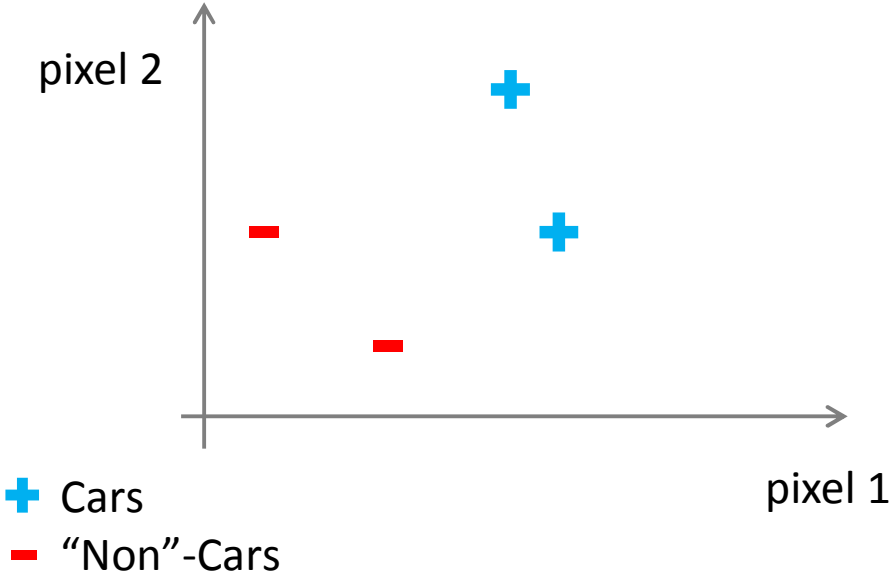
Learning Algorithm

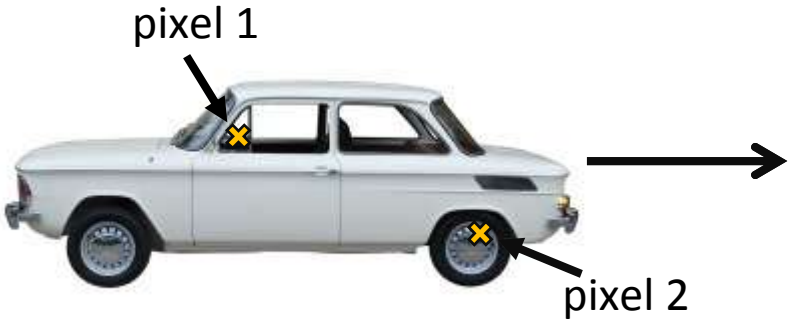


+ Cars
− "Non"-Cars

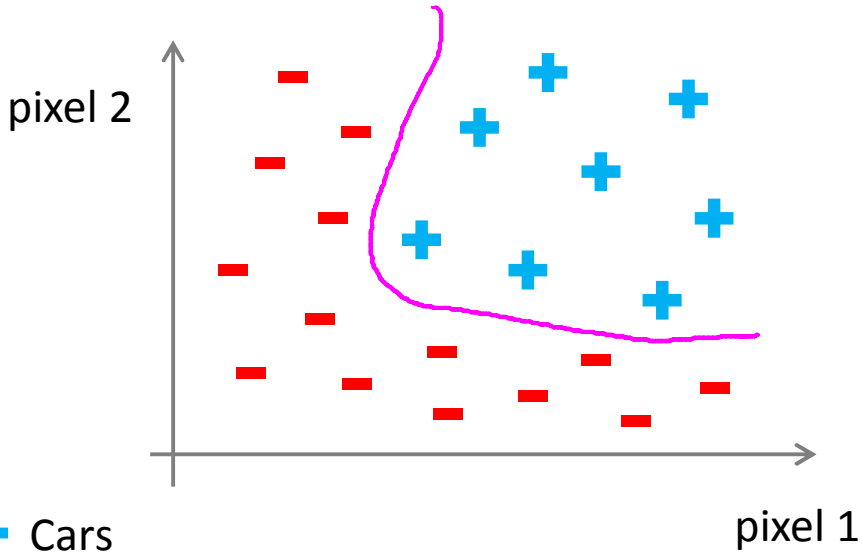


Learning Algorithm





Learning Algorithm



50 x 50 pixel images→ 2500 pixels

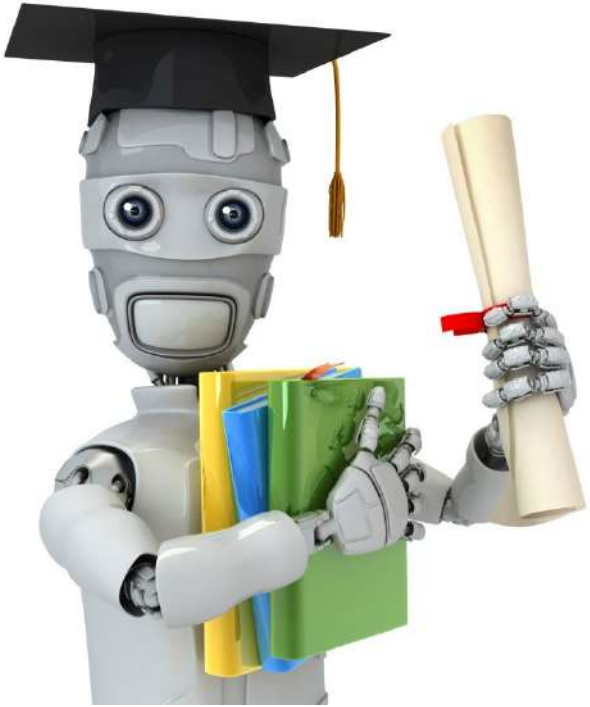
$\boxed{n = 2500}$ (7500 if RGB)

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

0-255

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

Andrew Ng



Machine Learning

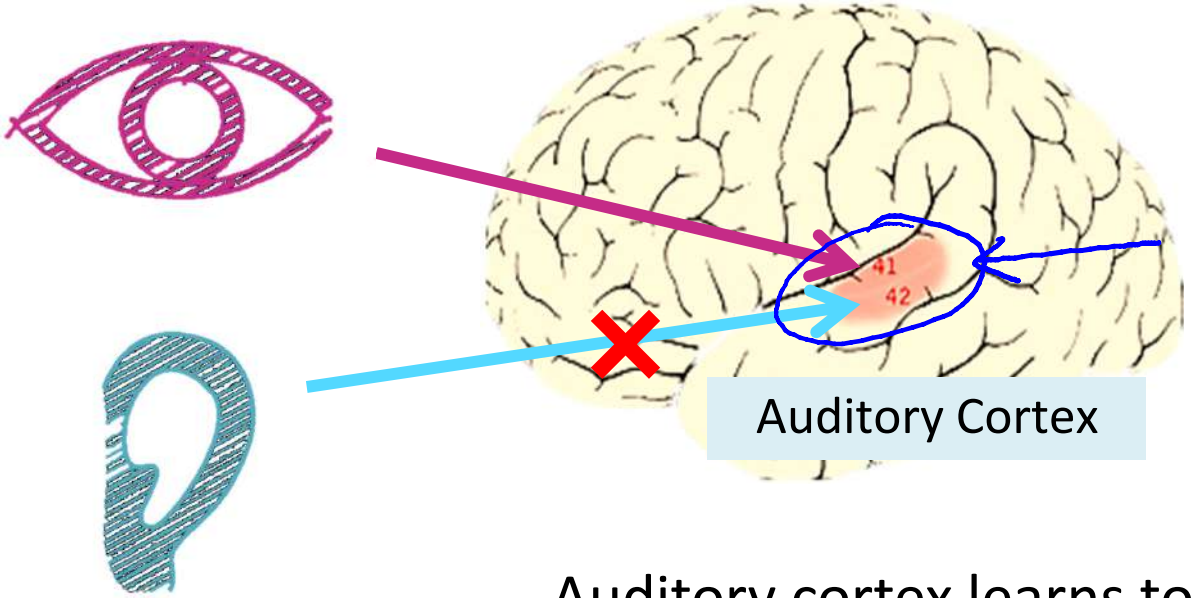
Neural Networks: Representation

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

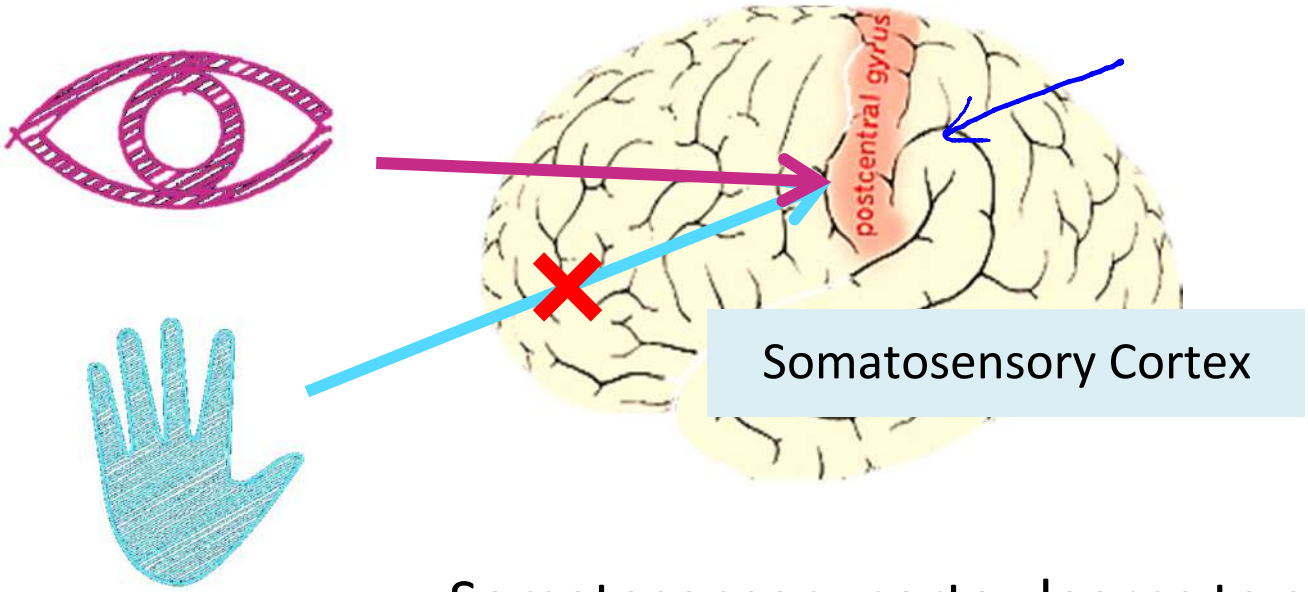
The “one learning algorithm” hypothesis



Auditory cortex learns to see

[Roe et al., 1992]

The “one learning algorithm” hypothesis



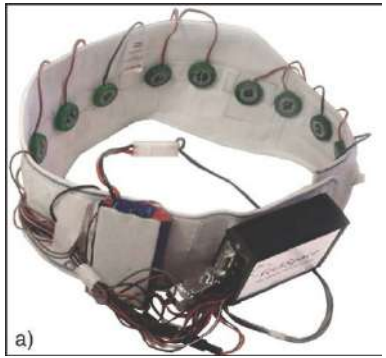
Somatosensory cortex learns to see

[Metin & Frost, 1989]

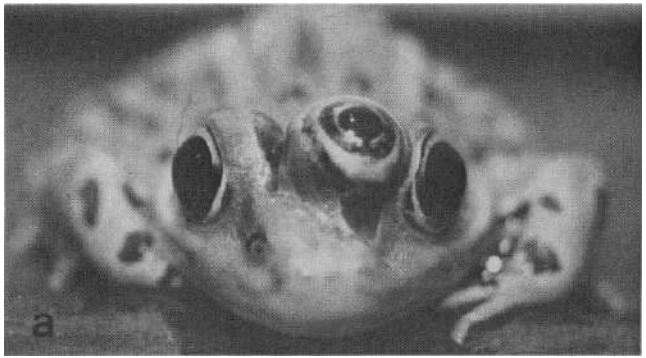
Sensor representations in the brain

Seeing with your tongue

Human echolocation (sonar)

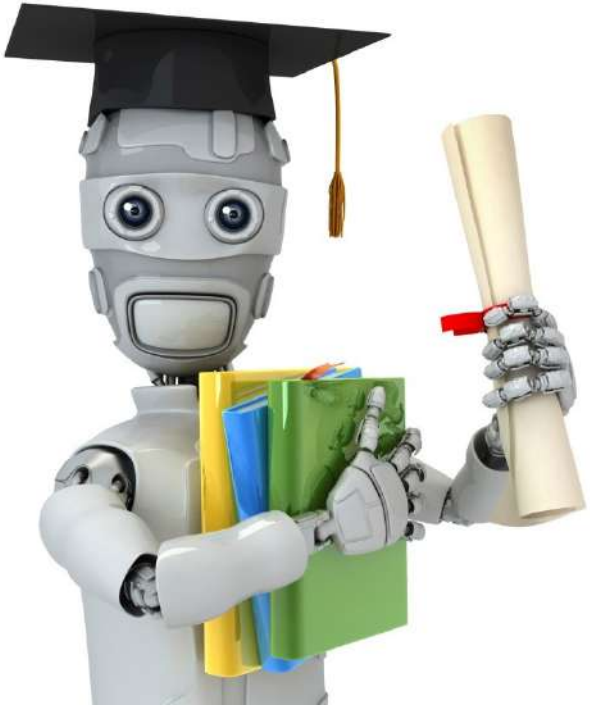


Haptic belt: Direction sense



Implanting a 3rd eye

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

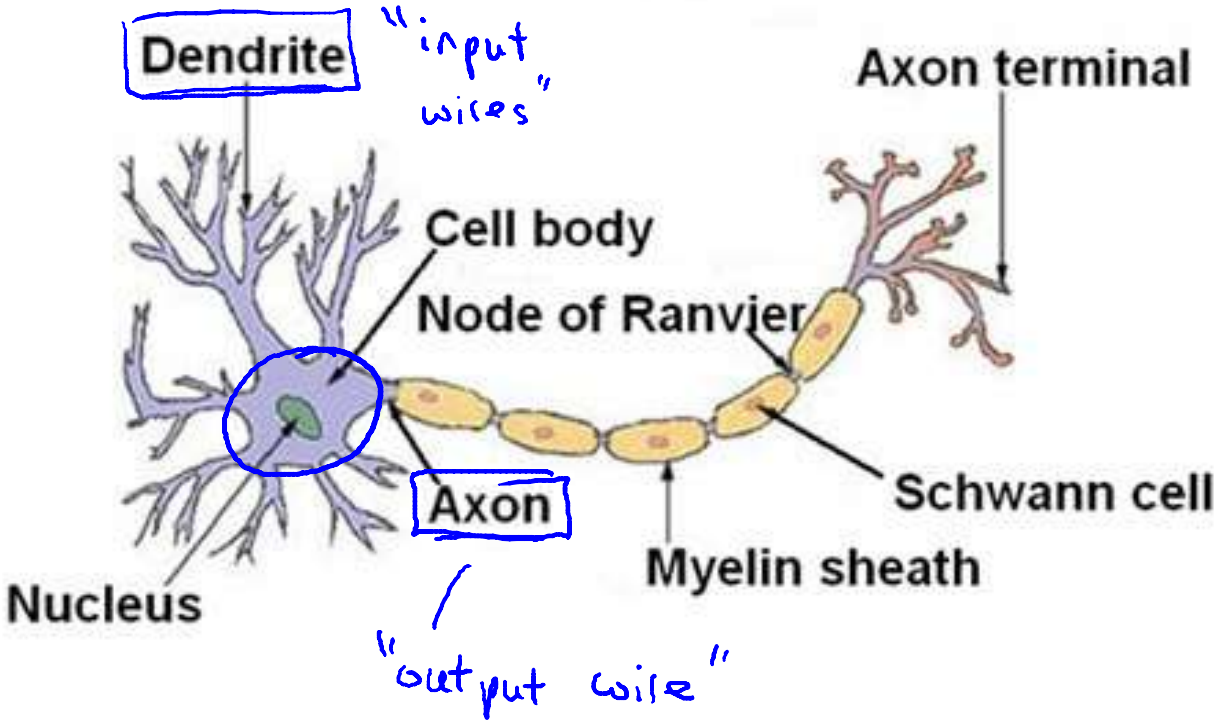


Machine Learning

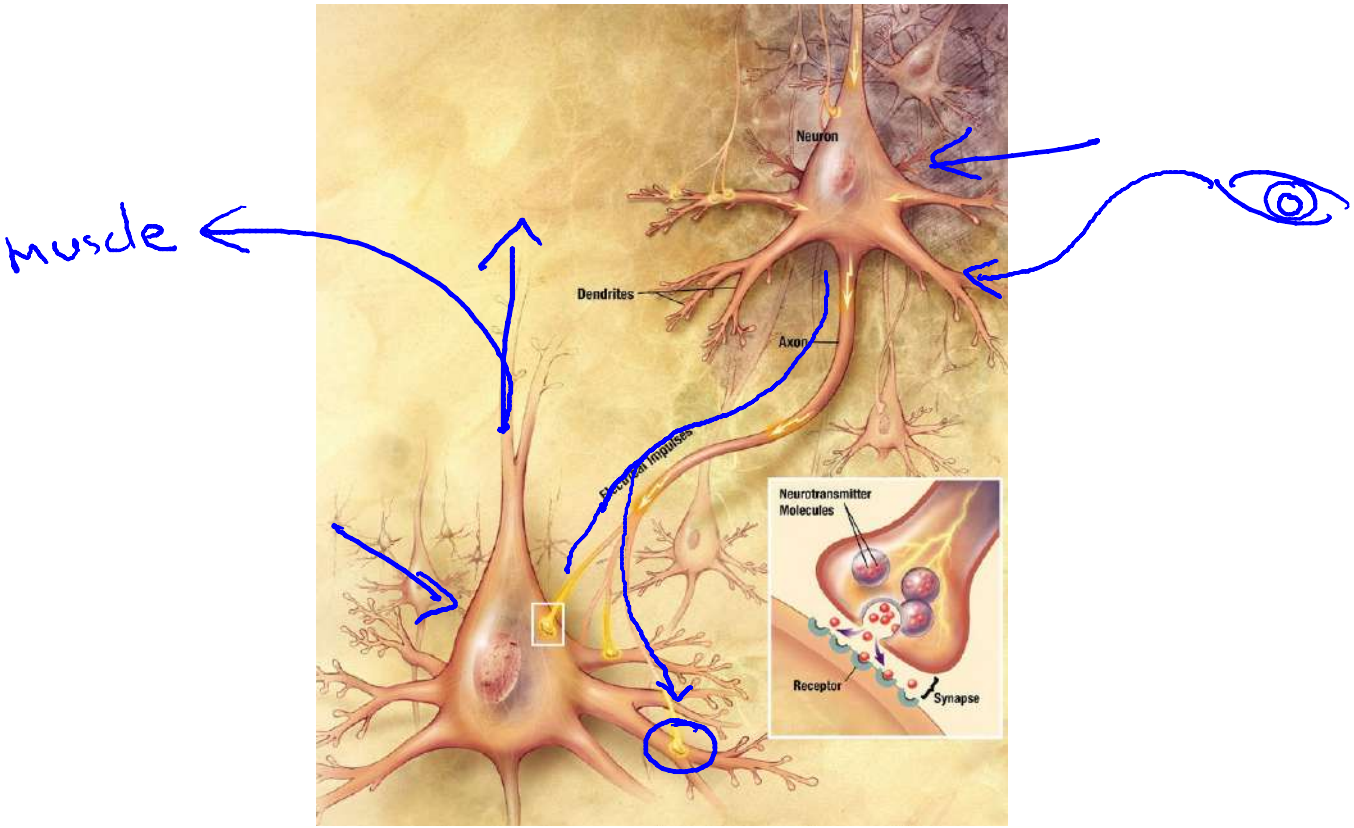
Neural Networks: Representation

Model representation I

Neuron in the brain

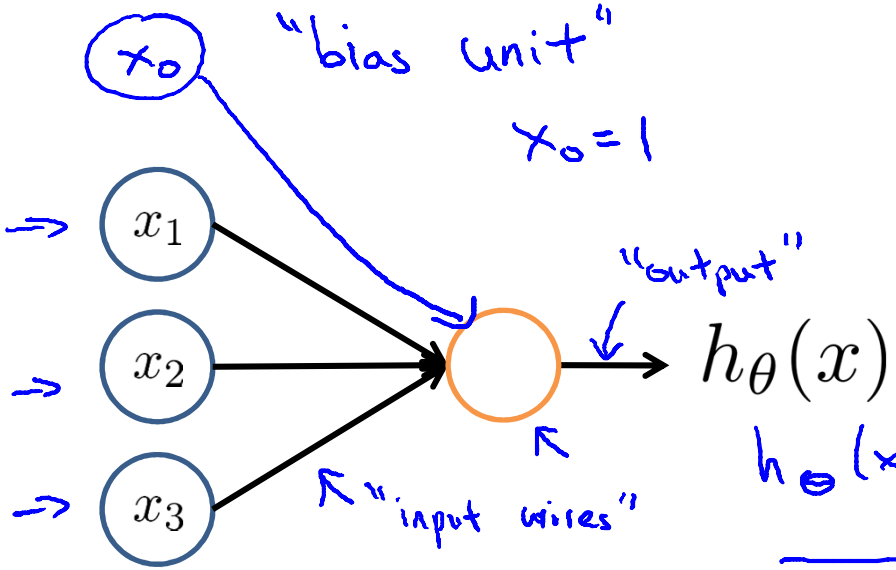


Neurons in the brain



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

Neuron model: Logistic unit



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

"weights"

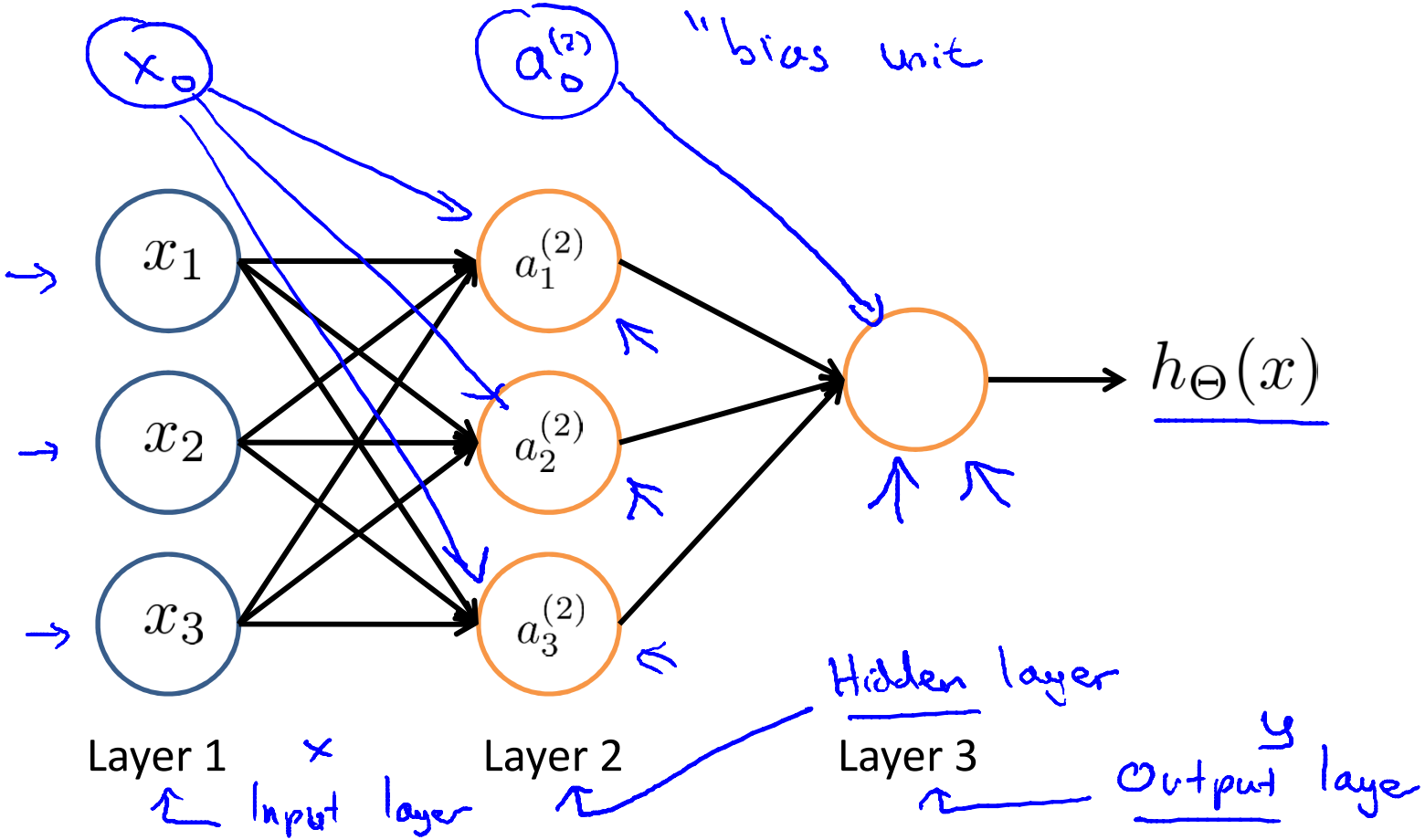
(parameters)

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

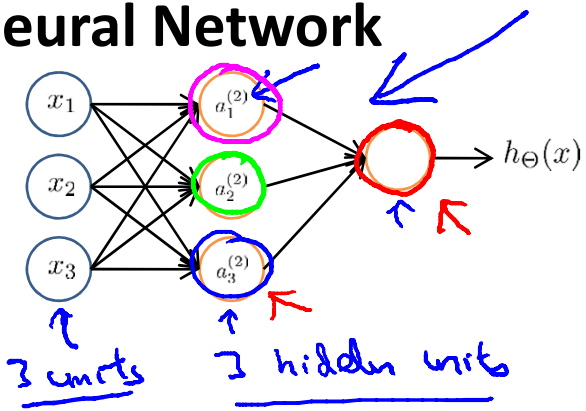
Sigmoid (logistic) activation function.

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

Neural Network



Neural Network



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

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

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

$h_\Theta(x)$

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

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

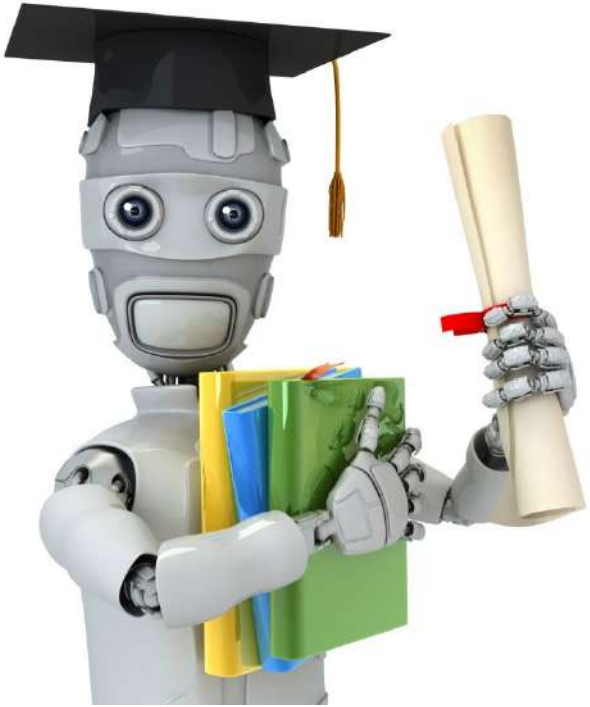
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$\Theta^{(2)}$

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

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

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

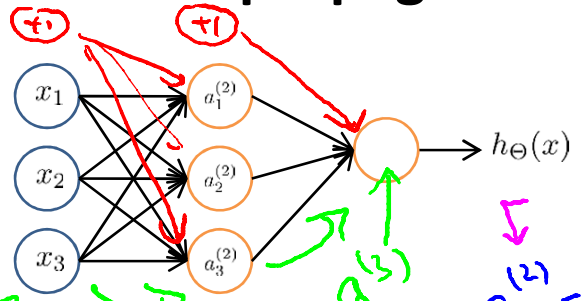


Machine Learning

Neural Networks: Representation

Model representation II

Forward propagation: Vectorized implementation



$a^{(1)} = x$

$a_1^{(3)}$

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

$z_1^{(2)}$

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

$z_2^{(2)}$

$z_3^{(2)}$

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

$a^{(1)} = x$

$a_3^{(2)} = g(z_3^{(2)})$

$z^{(3)}$

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

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

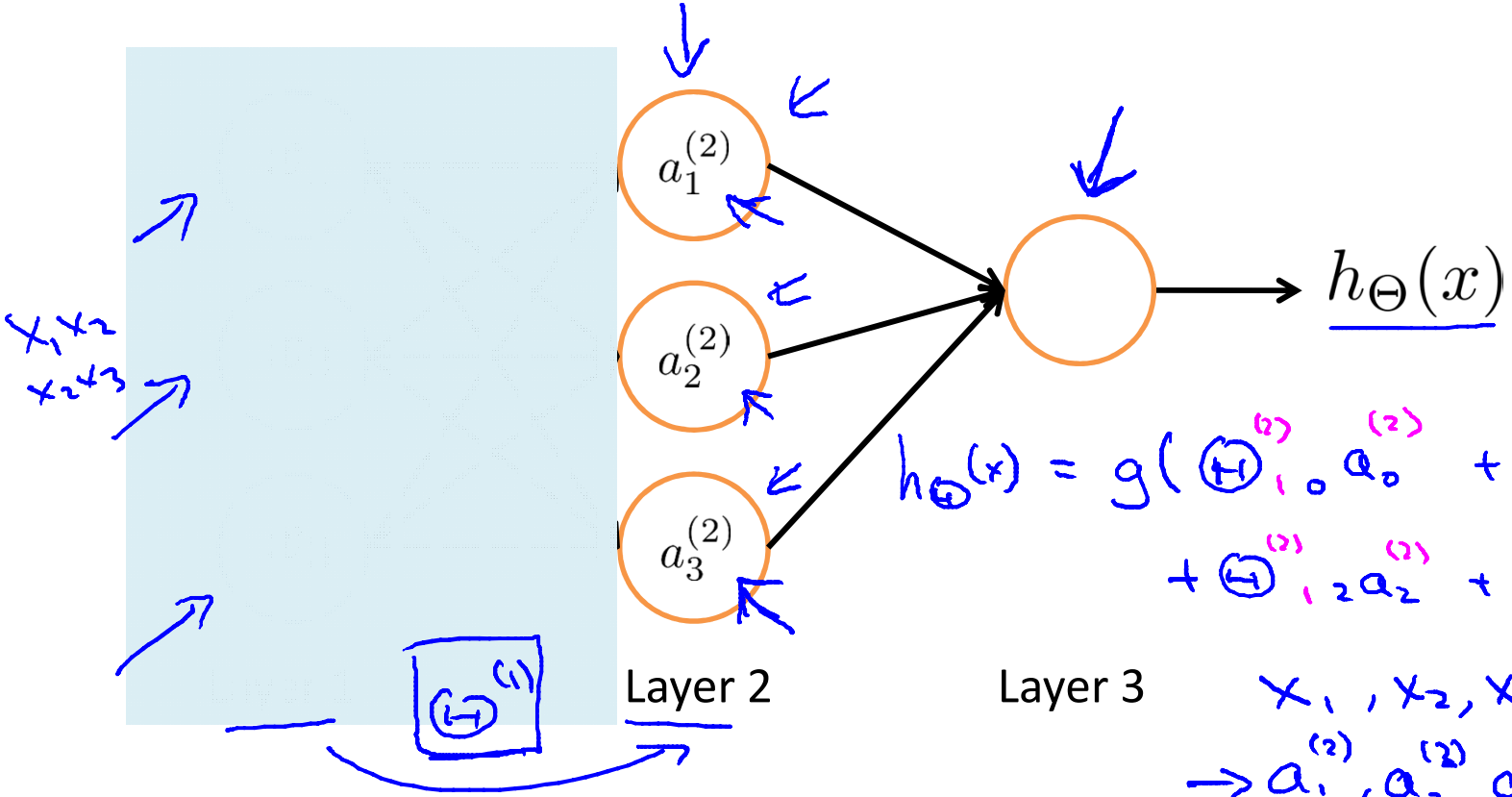
$\mathbb{R}^3$ $\mathbb{R}^3$

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

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

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

Neural Network learning its own features

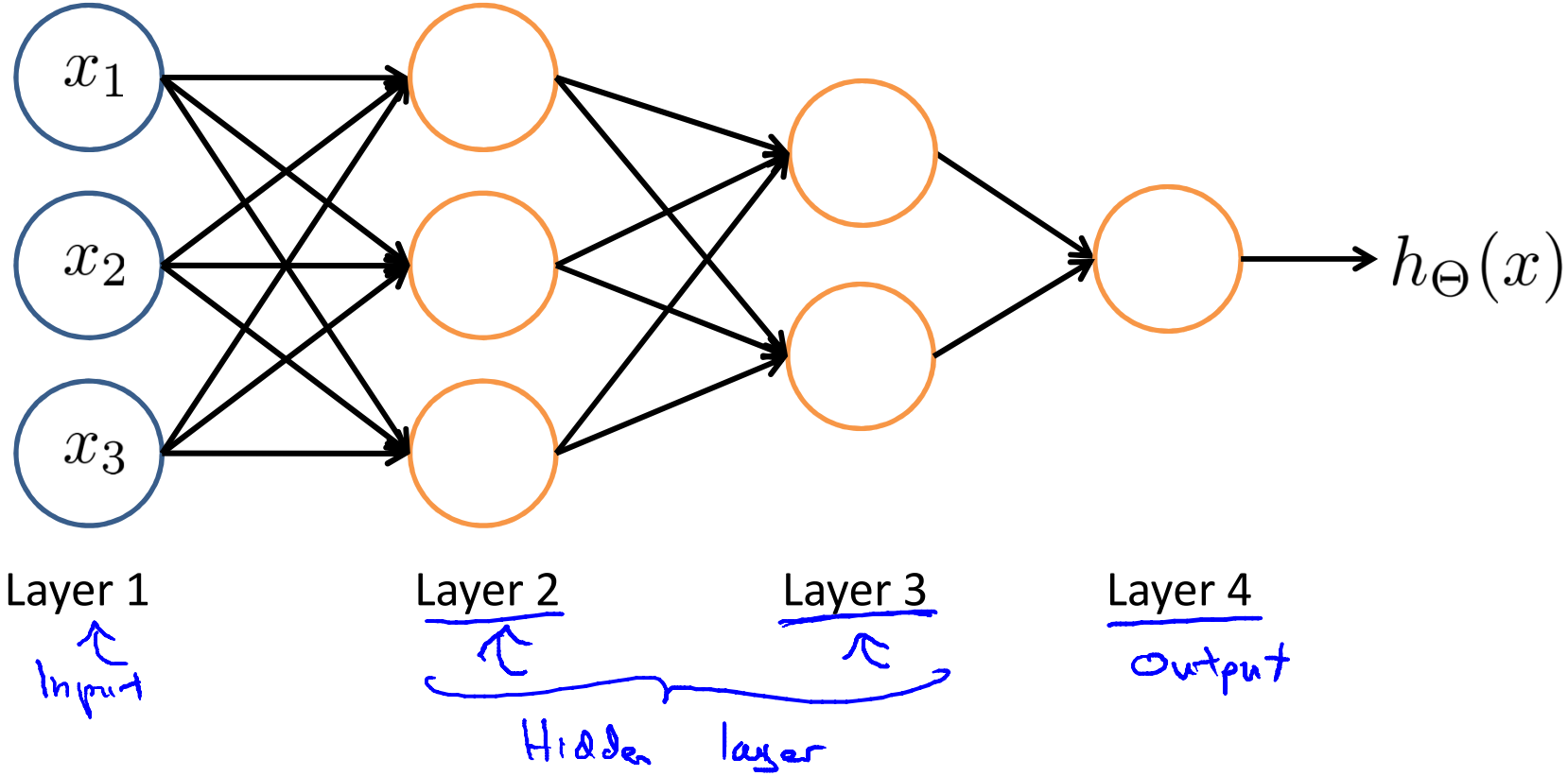


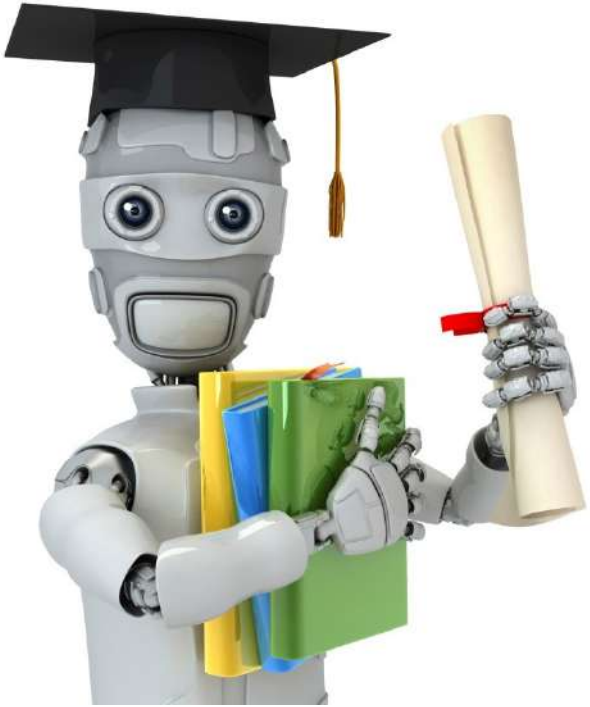
$$h_\Theta(x) = g(\Theta^{(2)}_{10} a_0^{(2)} + \Theta^{(1)}_{11} a_1^{(2)} + \Theta^{(2)}_{12} a_2^{(2)} + \Theta^{(2)}_{13} a_3^{(2)})$$

x_1, x_2, x_3

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

Other network architectures





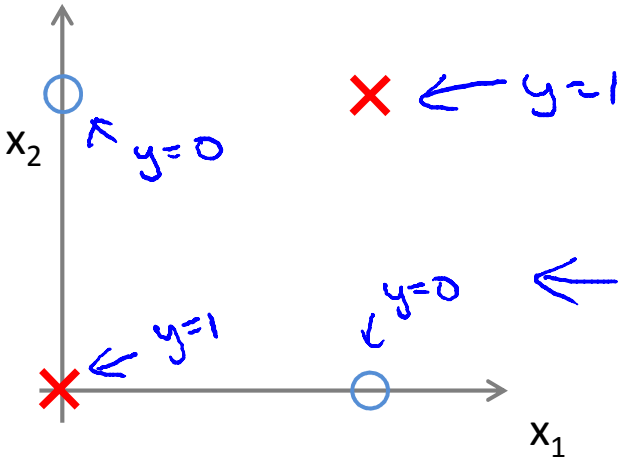
Machine Learning

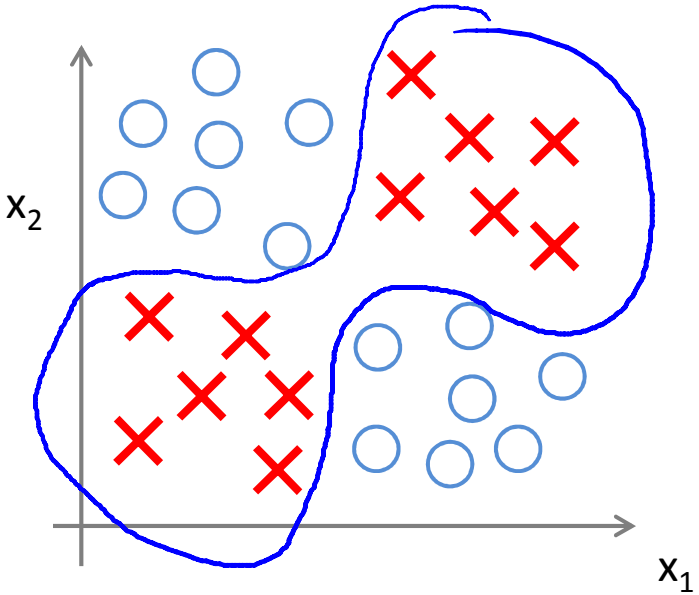
Neural Networks: Representation

Examples and intuitions I

Non-linear classification example: XOR/XNOR

x_1, x_2 are binary (0 or 1).





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

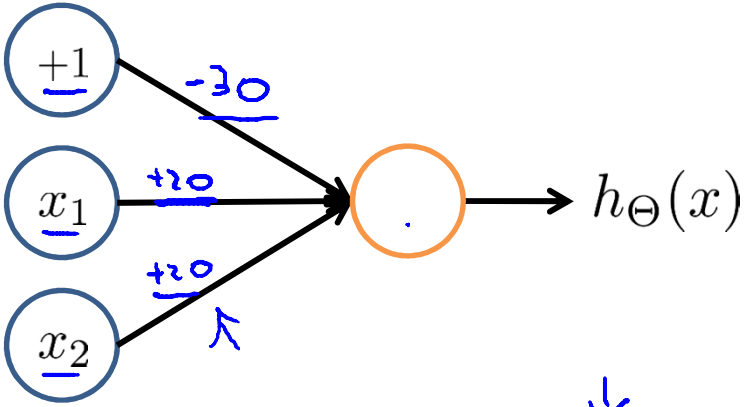
$$x_1 \text{ XNOR } x_2$$

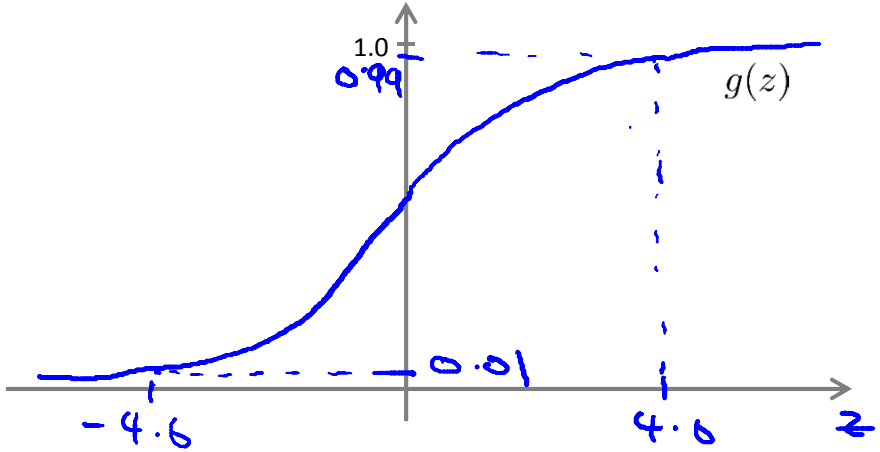
$$\text{NOT } (x_1 \text{ XOR } x_2)$$

Simple example: AND

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

$y = x_1 \text{ AND } x_2$





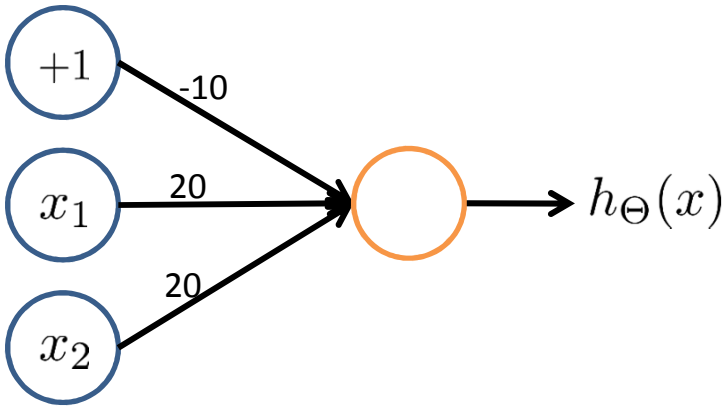
$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$

$\Theta^{(1)}_{10}$ $\Theta^{(1)}_{11}$ $\Theta^{(1)}_{12}$

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	$g(10) \approx 1$

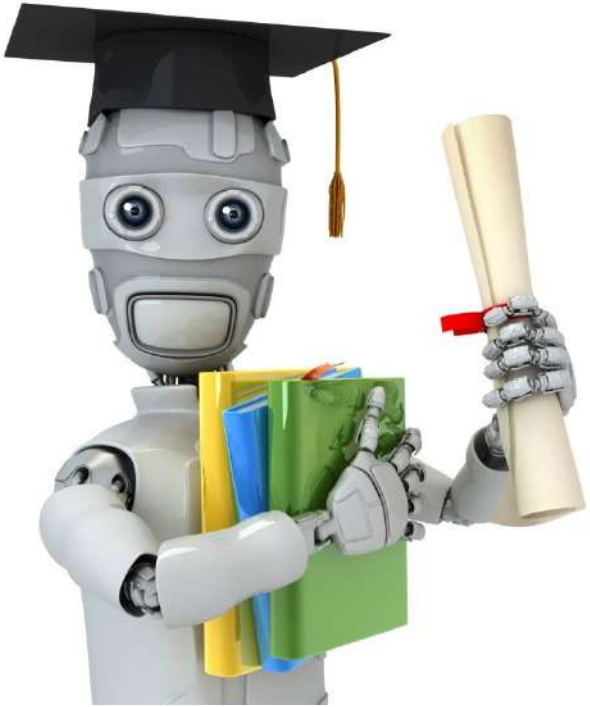
$h_\Theta(x) \approx x_1 \text{ AND } x_2$

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

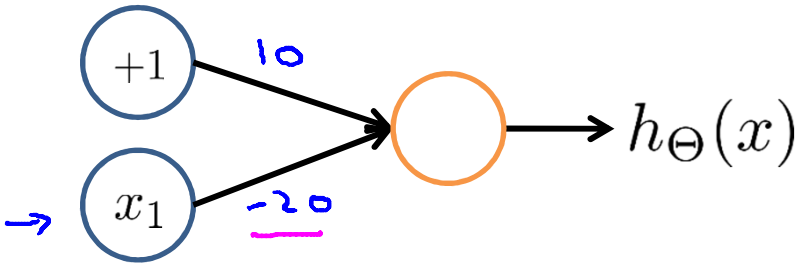
Neural Networks: Representation

Examples and intuitions II

$\rightarrow x_1$ AND x_2 $\qquad$ $\rightarrow x_1$ OR x_2 $\qquad$ $\{0,1\}$.

Negation:

NOT x_1



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

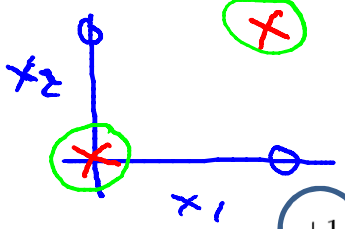
$$h_\Theta(x) = g(10 - 20x_1)$$

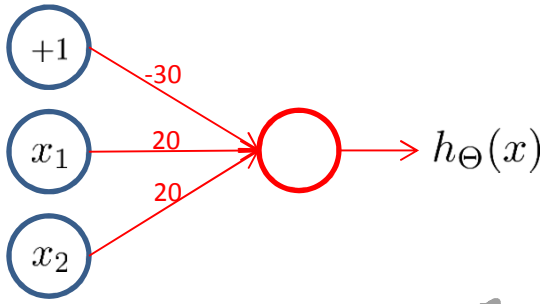
$\rightarrow$ (NOT x_1) AND (NOT x_2)

(=1 if and only if

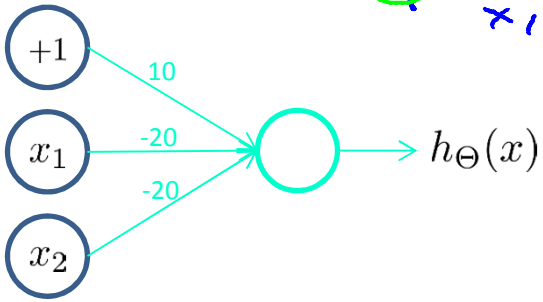
$\rightarrow x_1 = x_2 = 0$

Putting it together: x_1 XNOR x_2

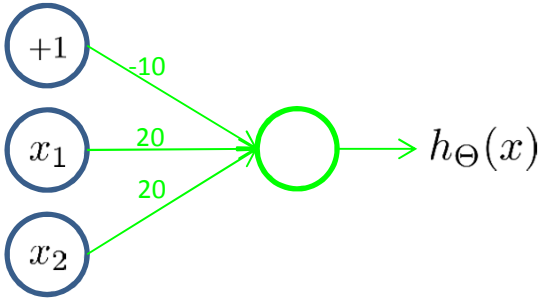




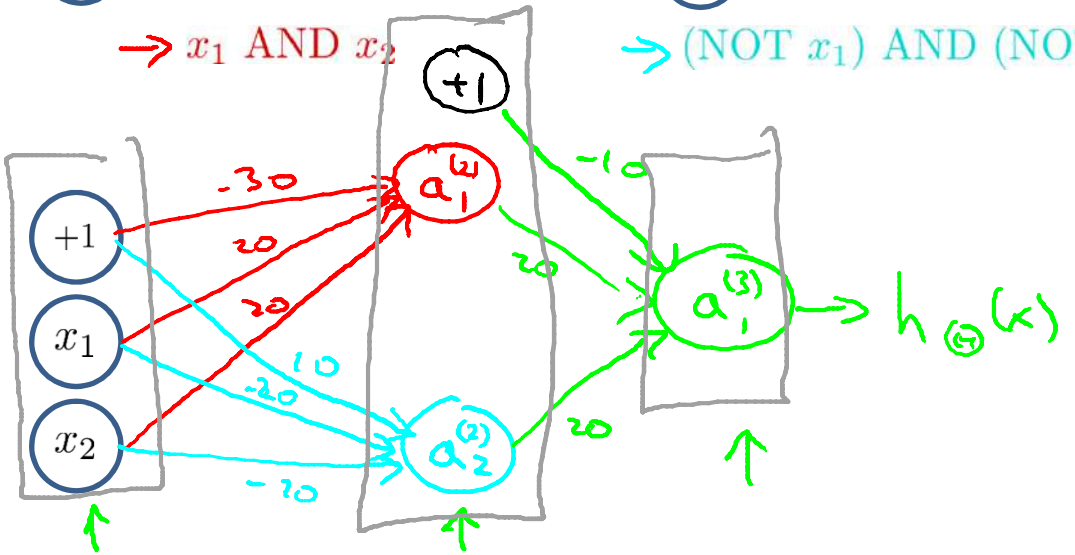
x_1 AND x_2



(NOT x_1) AND (NOT x_2)

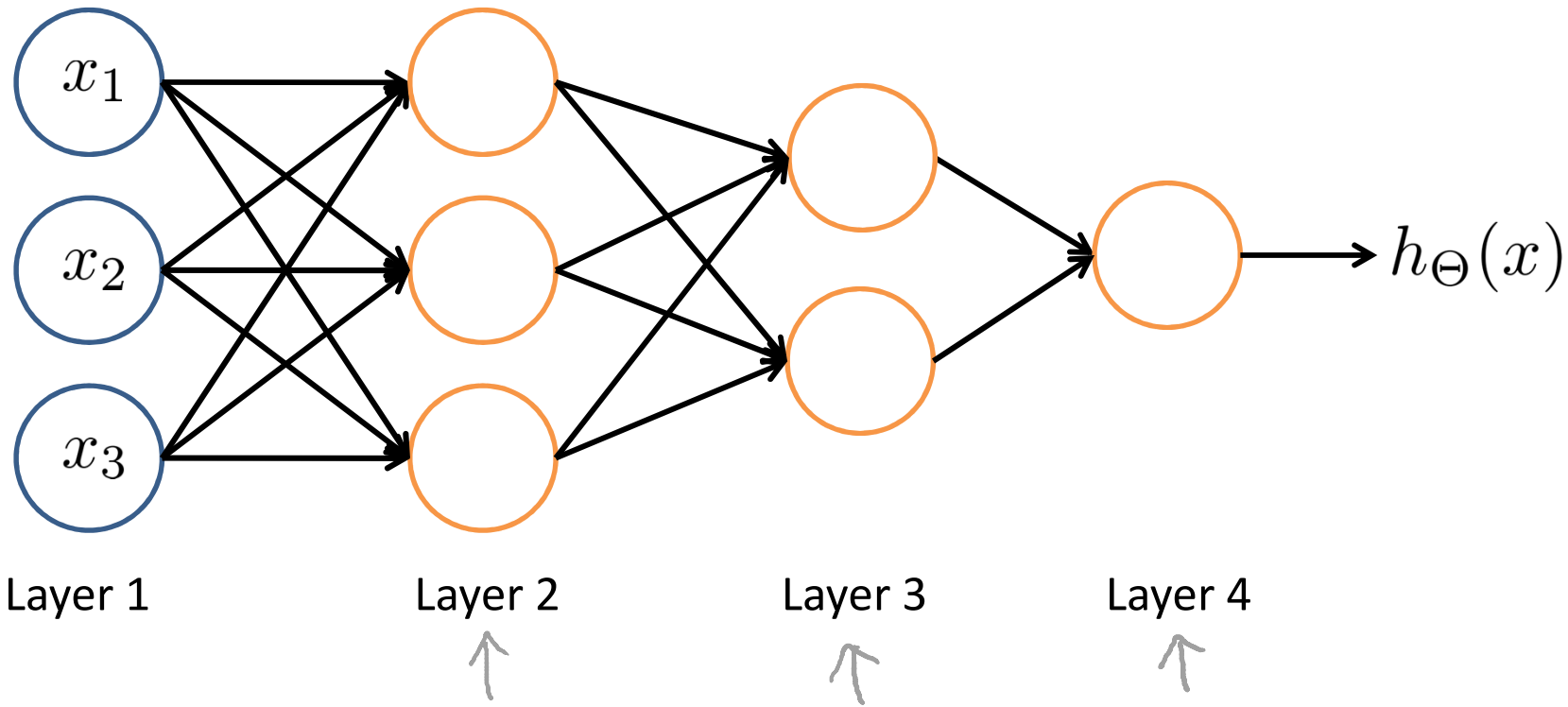


x_1 OR x_2

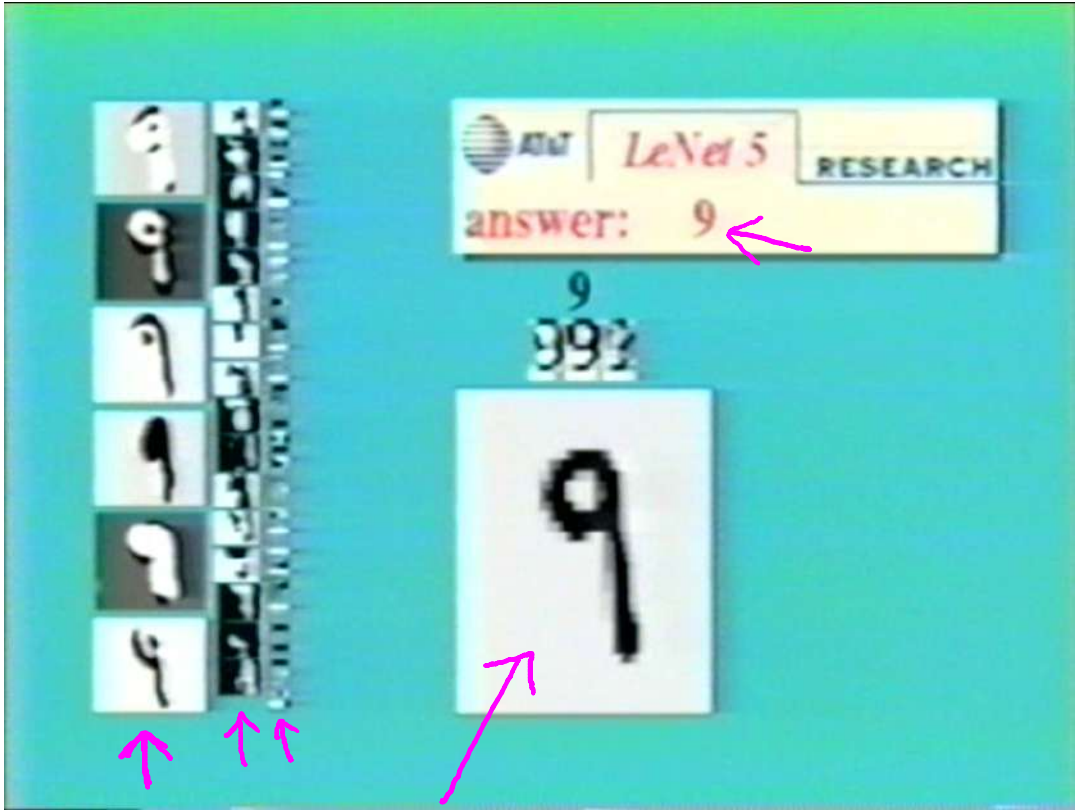


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Neural Network intuition

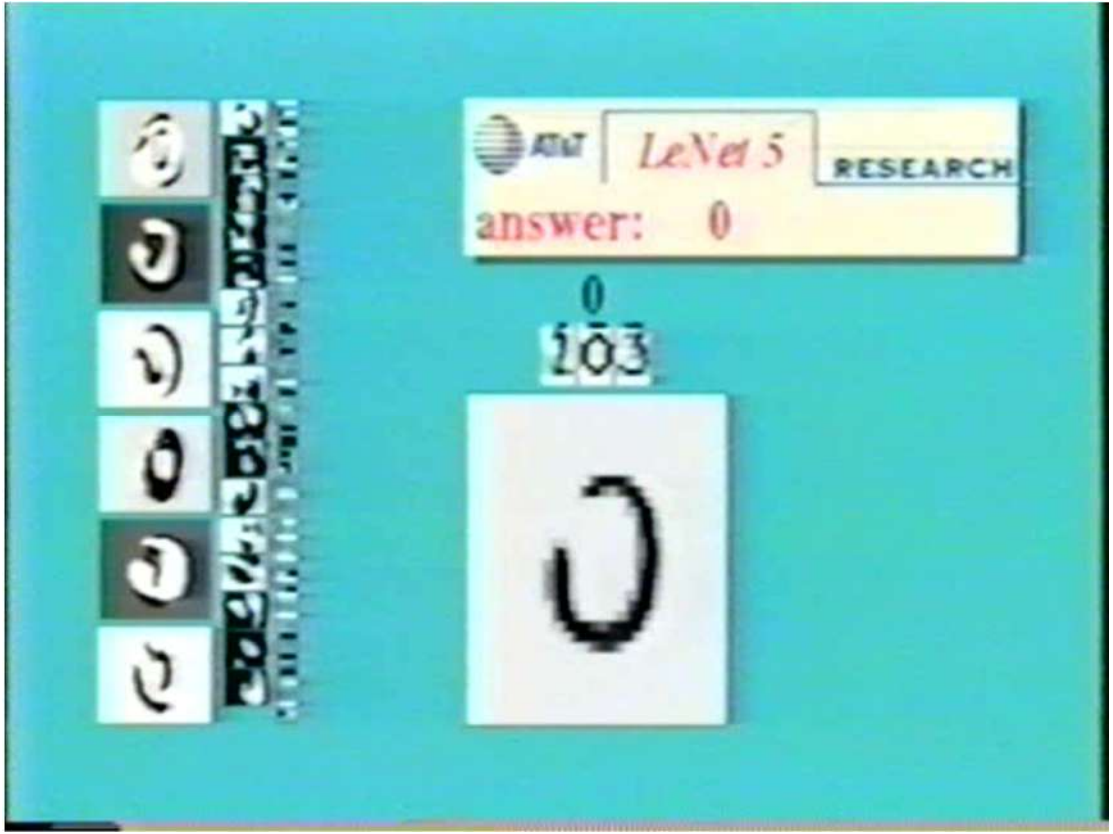


Handwritten digit classification

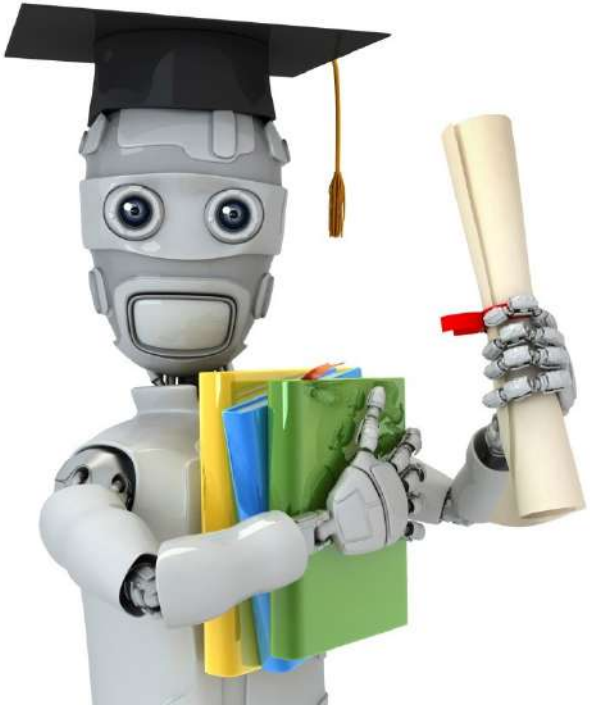


[Courtesy of Yann LeCun]

Handwritten digit classification



[Courtesy of Yann LeCun]



Machine Learning

Neural Networks: Representation

Multi-class classification

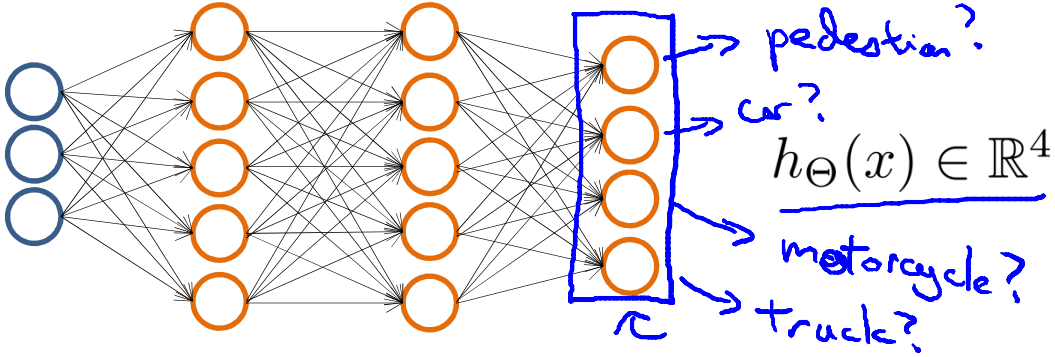
Multiple output units: One-vs-all.

Pedestrian

Car

Motorcycle

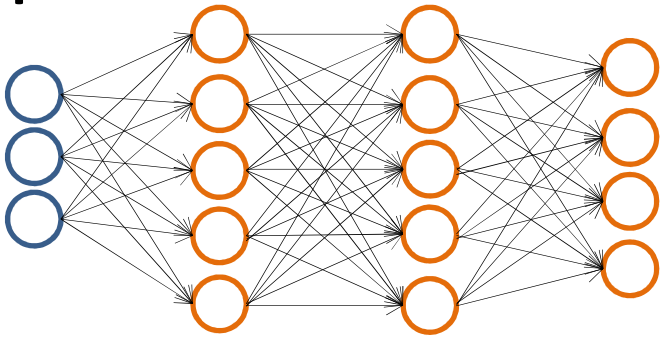
Truck



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

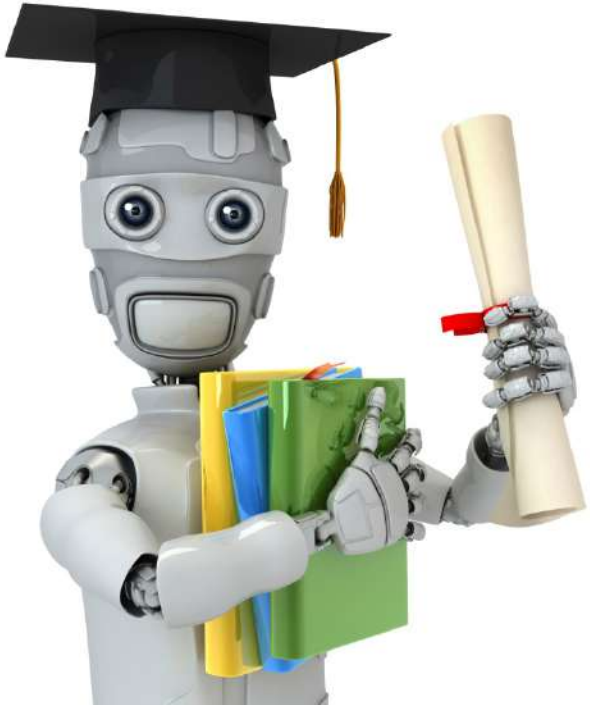
pedestrian car motorcycle truck

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

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

$h_\Theta(x^{(i)}) \approx y^{(i)}$

$\mathbb{R}^4$

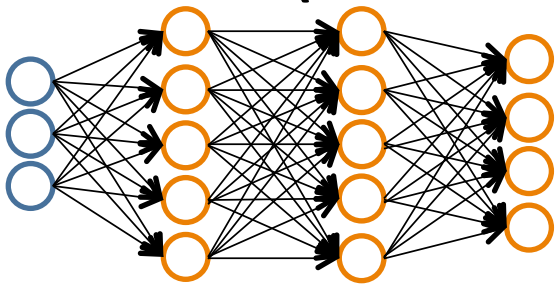


Machine Learning

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

Binary classification

$y = 0 \text{ or } 1$

1 output unit

Multi-class classification (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:

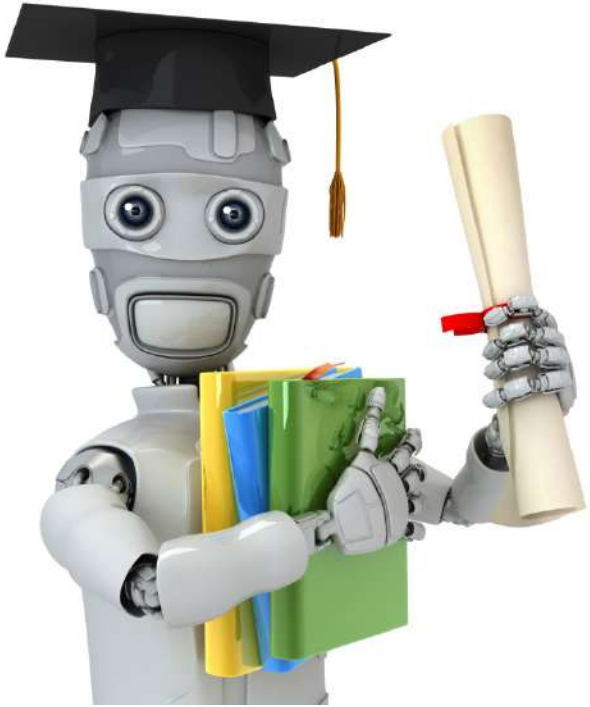
$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Neural network:

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

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

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



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:

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

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

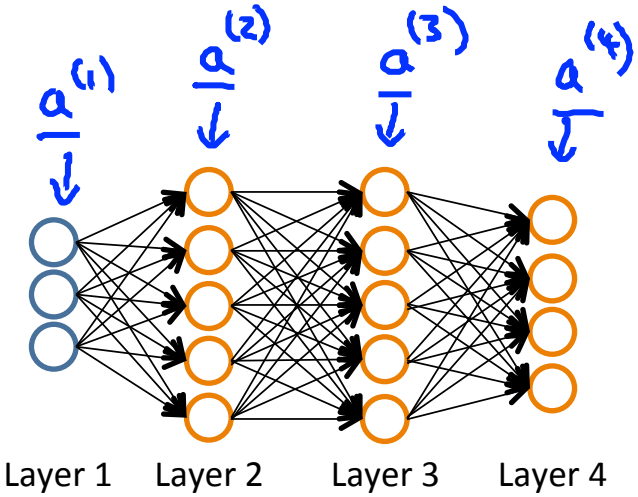
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$

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

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

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

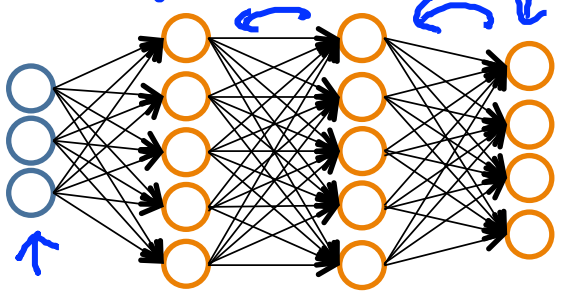
$$a^{(4)} = h_\Theta(x) = g(z^{(4)})$$



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)

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

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

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

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

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

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

(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). (used 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 $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$ $\cancel{\delta^{(1)}}$

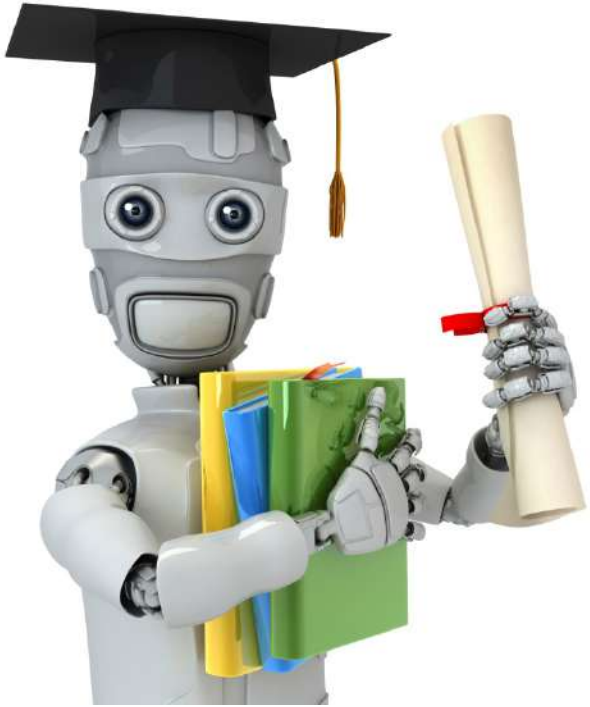
$\boxed{\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}}$ $\boxed{ij}$

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

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

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

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

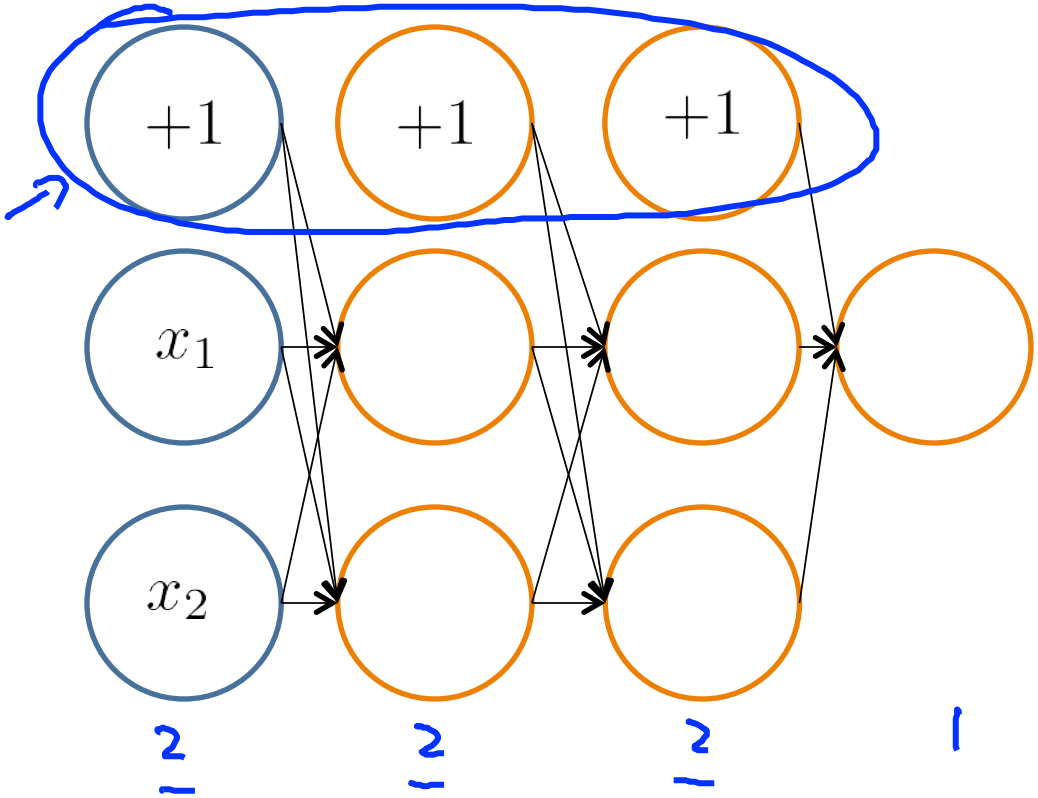


Machine Learning

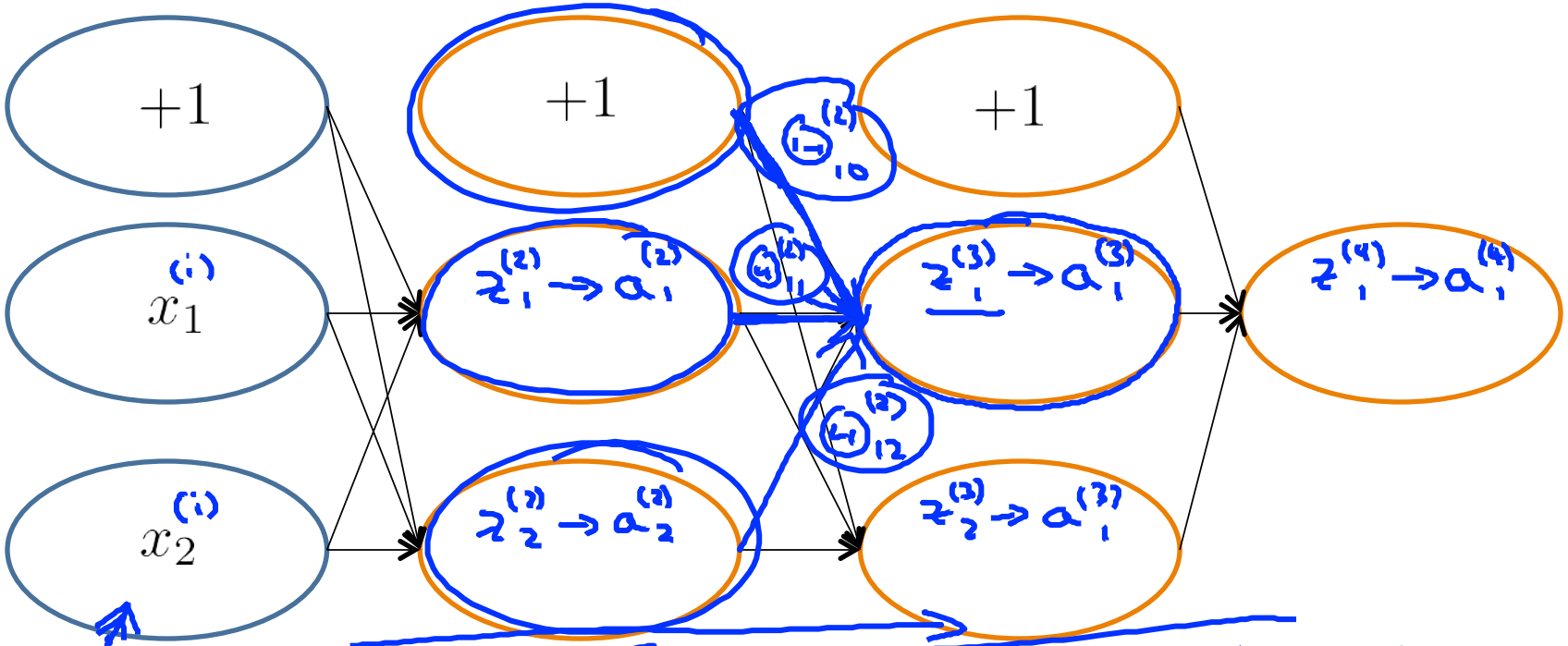
Neural Networks: Learning

Backpropagation intuition

Forward Propagation



Forward Propagation



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

$$z_1^{(3)} = \Theta_{10}^{(2)} \times 1 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} \cdot a_1^{(2)}$$

What is backpropagation doing?

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

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

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

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

Note: Mistake on lecture, it is supposed to be 1-h(x).

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

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

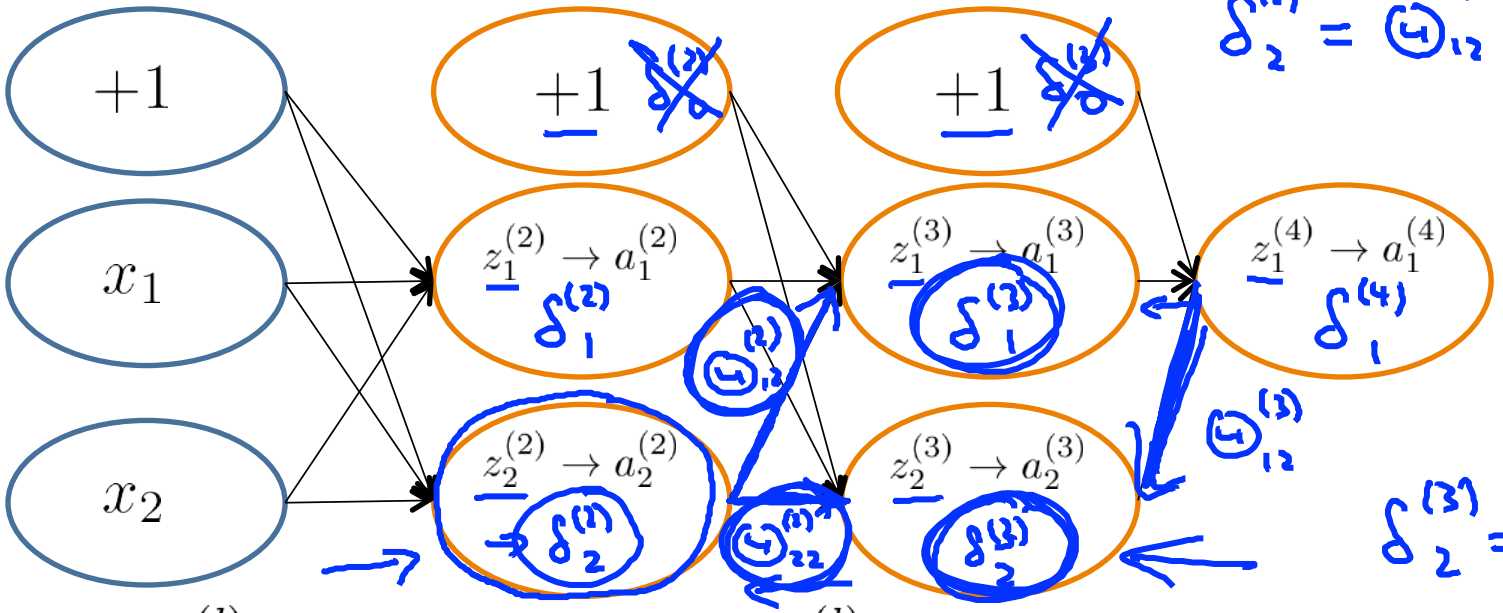
I.e. how well is the network doing on example i?

$y^{(i)}$

Forward Propagation

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

$\delta_2^{(2)} = \Theta_{12}^{(2)} \delta_1^{(3)} + \Theta_{12}^{(2)} \delta_2^{(3)}$



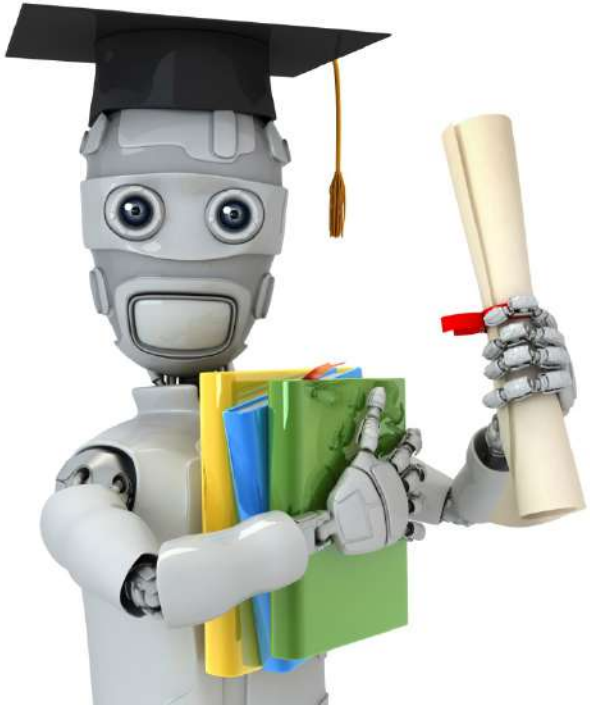
$\delta_2^{(3)} = \Theta_{12}^{(3)} \cdot \delta_1^{(4)}$.

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

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost(i)}$ (for $j \geq 0$), where

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

Andrew Ng



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

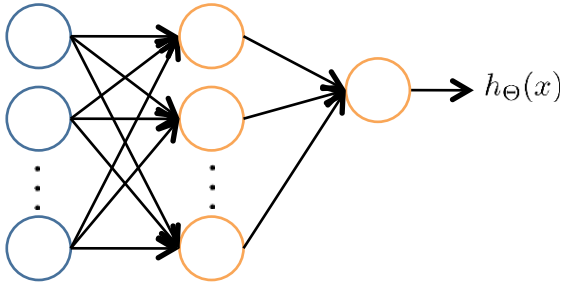
Example

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

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

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

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



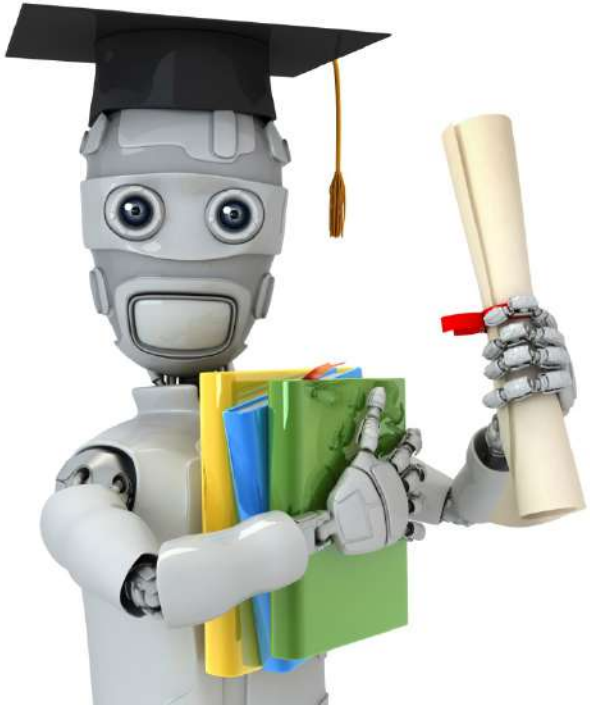

```
thetaVec = [ Theta1(:); Theta2(:); Theta3(:)];
DVec = [D1(:); D2(:); D3(:)];

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

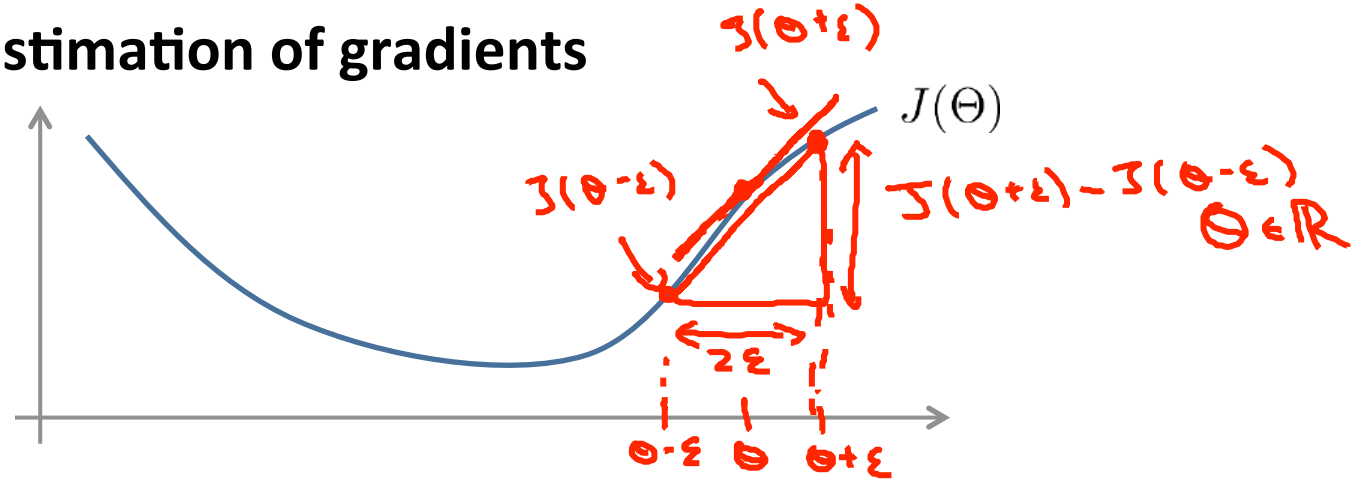


Machine Learning

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)`

Parameter vector θ

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

$\theta = [\theta_1, \theta_2, \theta_3, \ldots, \theta_n]$

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

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

$$\vdots$$

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

```
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:

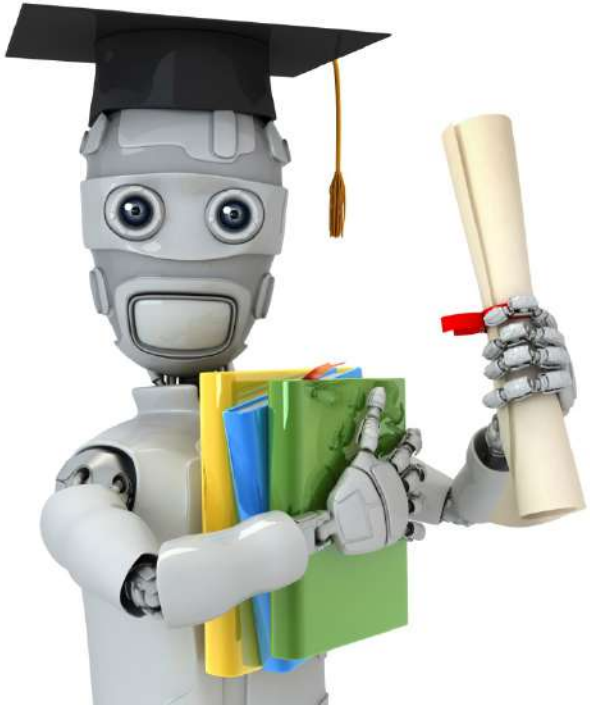
- Implement backprop to compute `DVec` (unrolled $D^{(1)}, D^{(2)}, D^{(3)}$).
- Implement numerical gradient check to compute `gradApprox`.
- Make sure they give similar values.
- Turn off gradient checking. Using backprop code for learning.

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 very slow.



Machine Learning

Neural Networks: Learning

Random initialization

Initial value of Θ

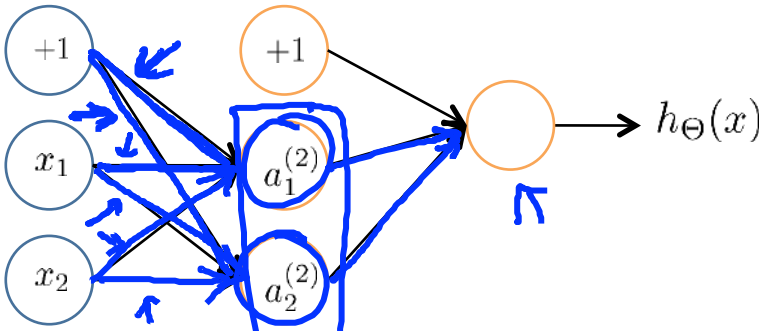
For gradient descent and advanced optimization method, need initial value for Θ.

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

Consider gradient descent

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

Zero initialization



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

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

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

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. $-\epsilon \leq \Theta_{ij}^{(l)} \leq \epsilon$)

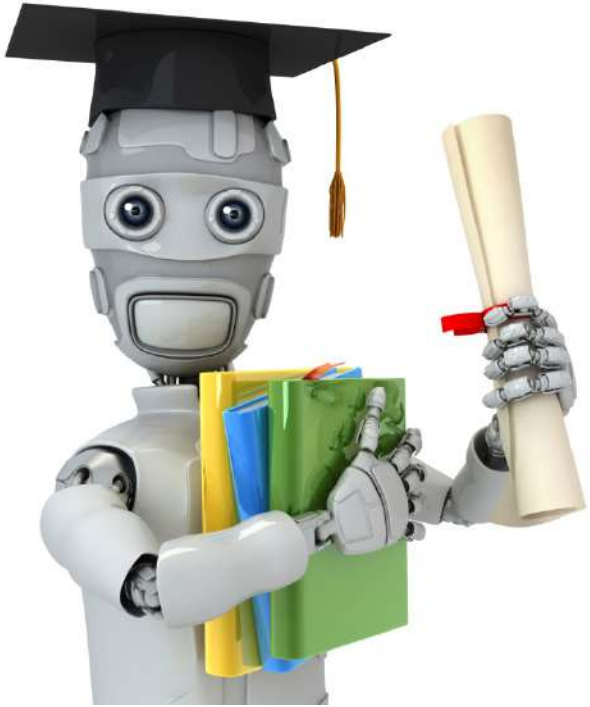
E.g.

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;
```



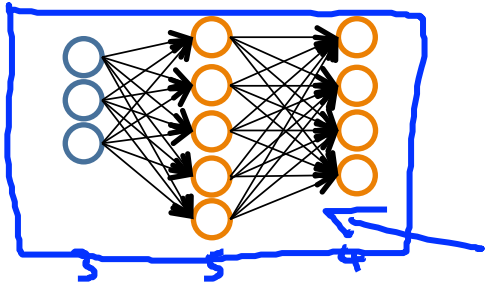
Machine Learning

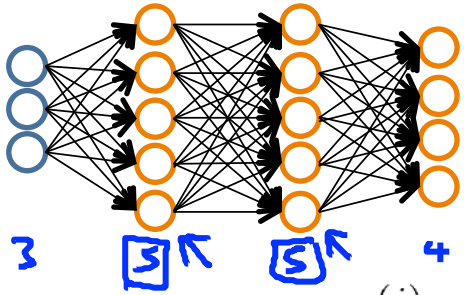
Neural Networks: Learning

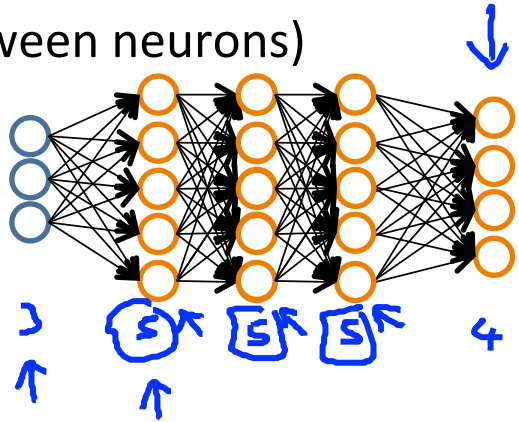
Putting it together

Training a neural network

Pick a network architecture (connectivity pattern between neurons)







- No. of input units: Dimension of features $x^{(i)}$
- No. output units: Number of classes
- Reasonable default: 1 hidden layer, or if >1 hidden layer, have same no. of hidden units in every layer (usually the more the better)

$y \in \{1, 2, 3, \ldots, 10\}$

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

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

Training a neural network

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

```
for i = 1:m
```
$\{ \quad (x^{(1)}, y^{(1)}) \quad (x^{(2)}, y^{(2)}), \ldots\ldots (x^{(m)}, y^{(m)})$

- Perform forward propagation and backpropagation using example $(x^{(i)}, y^{(i)})$
  (Get activations $a^{(l)}$ and delta terms $\delta^{(l)}$ for $l = 2, \ldots, L$).

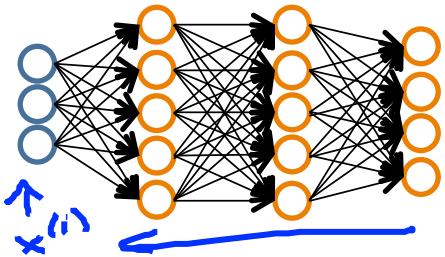
$\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$

$\ldots$

$\}$

$\ldots$

compute $\frac{\partial}{\partial \Theta^{(l)}_{ijk}} J(\Theta)$.

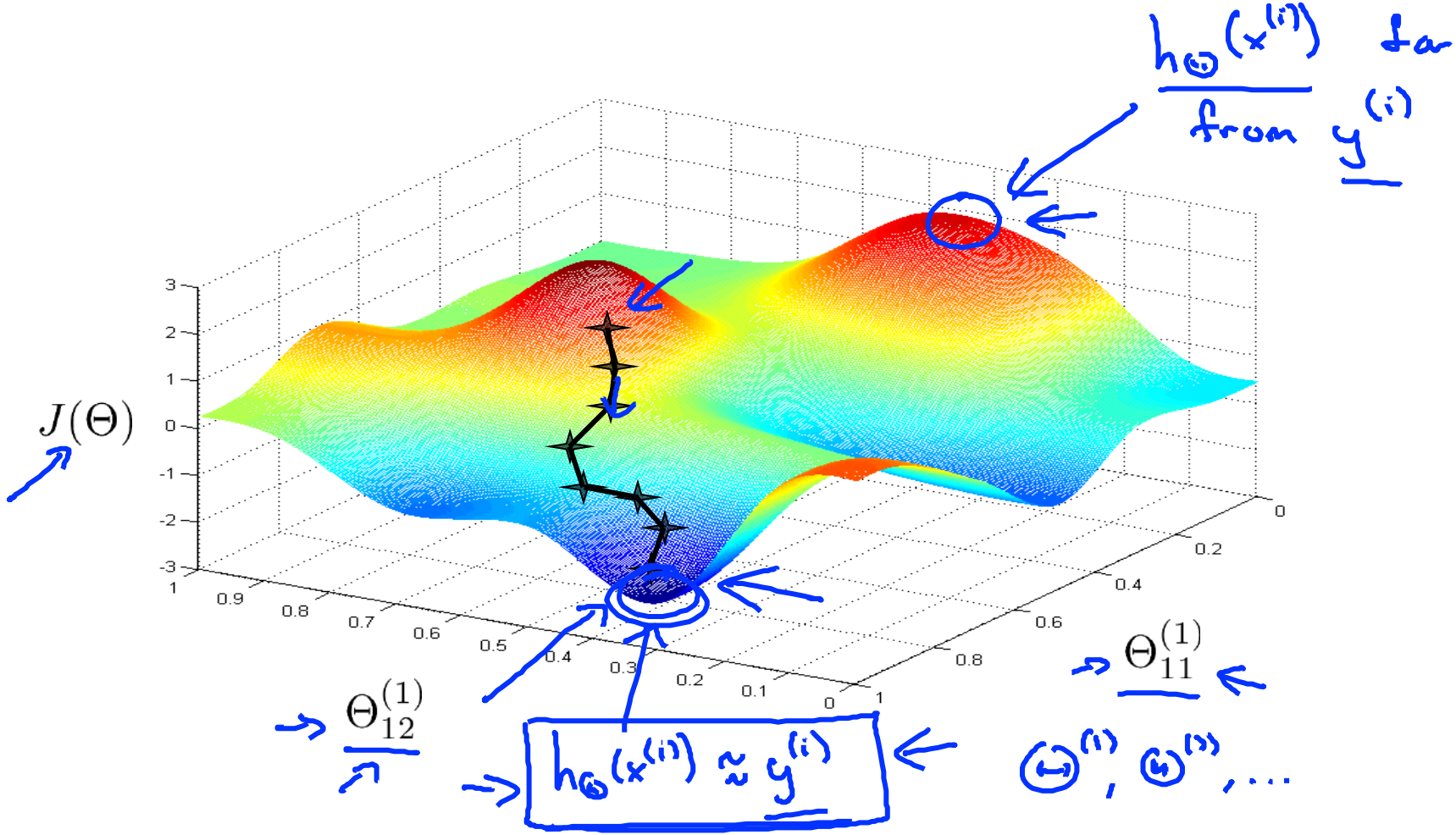


Training a neural network

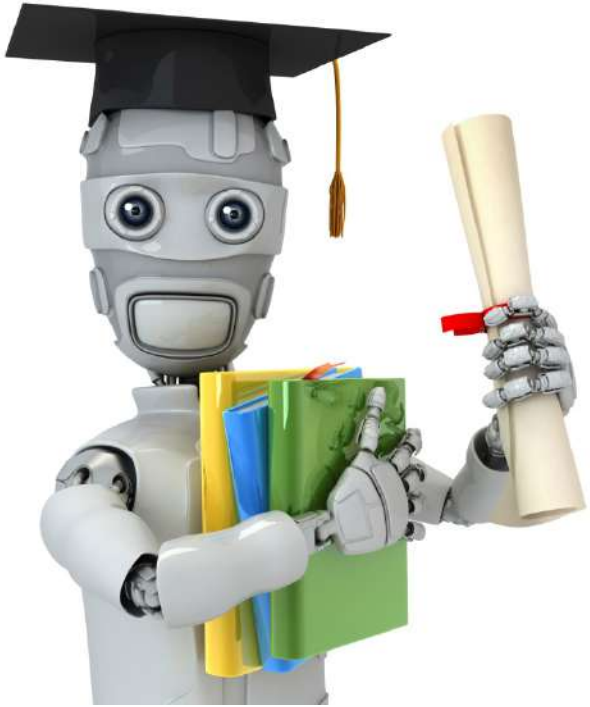
5. Use gradient checking to compare $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$ computed using backpropagation vs. using numerical estimate of gradient of $J(\Theta)$.
   - Then disable gradient checking code.
6. Use gradient descent or advanced optimization method with backpropagation to try to minimize $J(\Theta)$ as a function of parameters Θ

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

$J(\Theta)$ — non-convex.


$h_\Theta(x^{(i)})$ far from $y^{(i)}$
$J(\Theta)$
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
$\Theta_{11}^{(1)}$
$\Theta_{12}^{(1)}$
$h_\Theta(x^{(i)}) \approx y^{(i)}$
$\Theta^{(1)}, \Theta^{(2)}, \ldots$

Andrew Ng

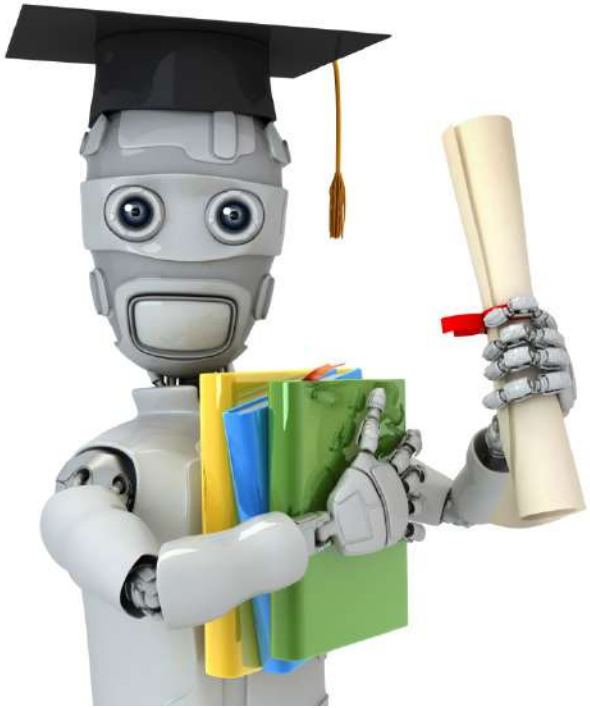


Machine Learning

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.

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

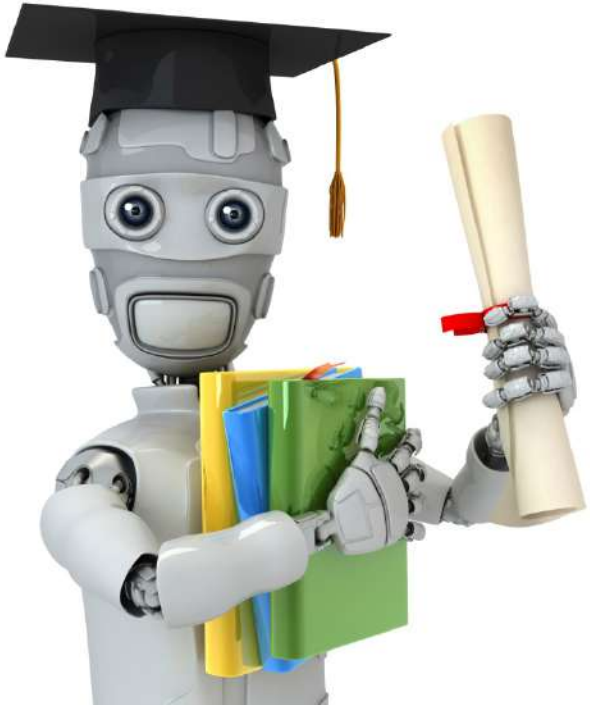
However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- 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.

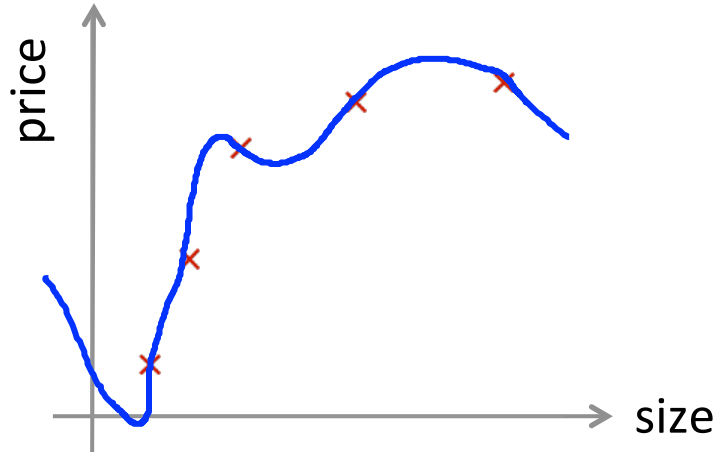


Machine Learning

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

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

30%

Test set

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

m_{test} = no. of test example

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

Andrew Ng

Training/testing procedure for linear regression

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

70%

- Compute test set error:

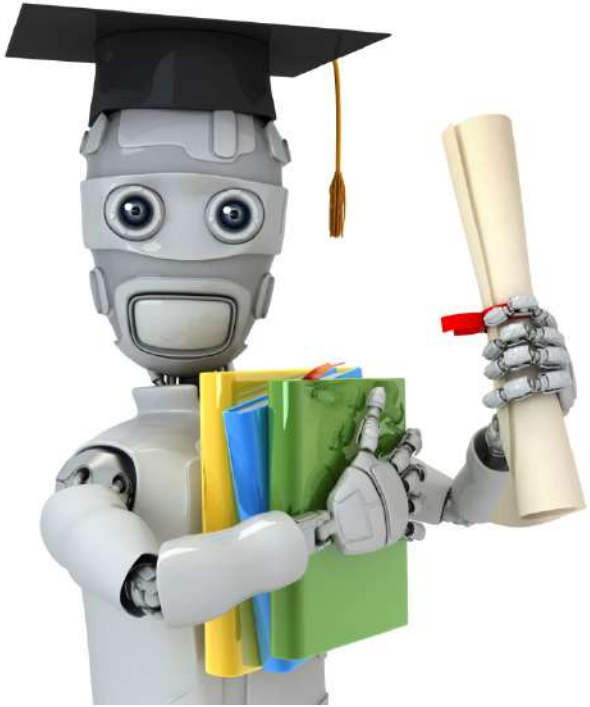
$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right)^2$$

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):

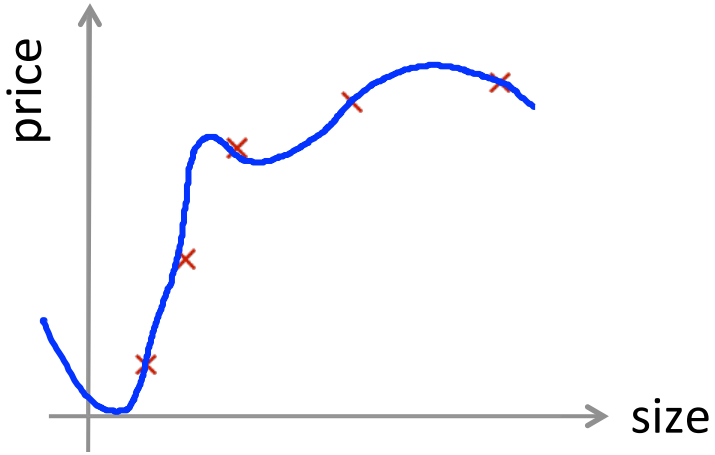


Machine Learning

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.

d = degree of polynomial

Model selection

$d=1$ 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. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3 \longrightarrow \Theta^{(3)} \longrightarrow J_{test}(\Theta^{(3)})$

$\vdots$

$d=10$ 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	
60%	2400	369	Training set
	1416	232	
	3000	540	
	1985	300	
20%	1534	315	Cross validation set (CV)
	1427	199	
20%	1380	212	test set
	1494	243	

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

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

m_{cv} = no. of cv examples

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

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

m_{test}

Train/validation/test error

Training error:

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

$J(\theta)$

Cross Validation error:

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

Test error:

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

Model selection

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

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

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

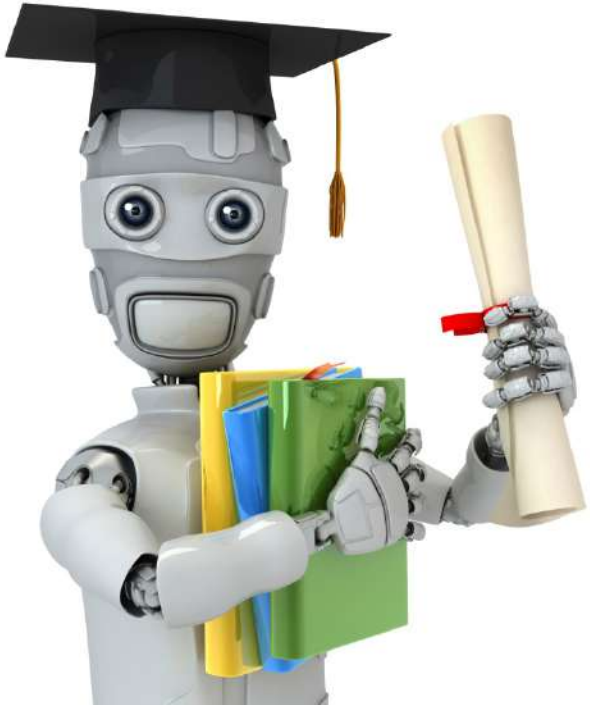
$\vdots$ $\vdots$ $\vdots$ $J_{cv}(\Theta^{(4)})$

$d=10$ 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 $\underline{J_{test}(\theta^{(4)})} \leftarrow$

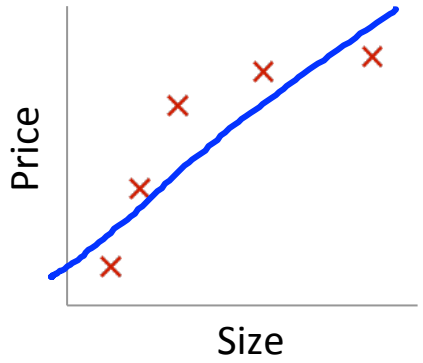


Machine Learning

Advice for applying machine learning

Diagnosing bias vs. variance

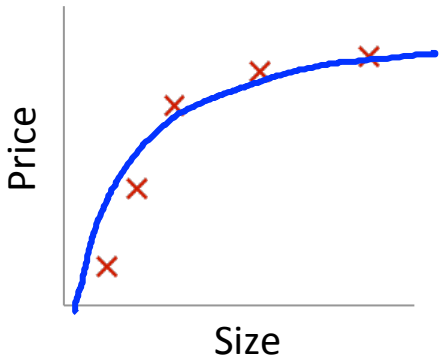
Bias/variance



$$\theta_0 + \theta_1 x$$

High bias
(underfit)

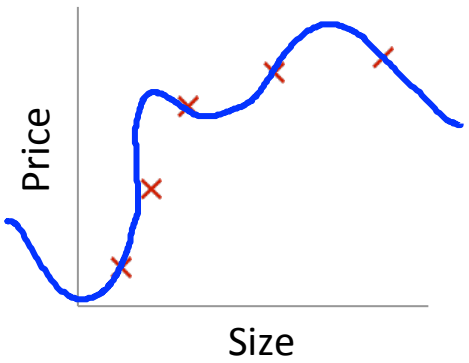
$d=1$



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

"Just right"

$d=2$



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

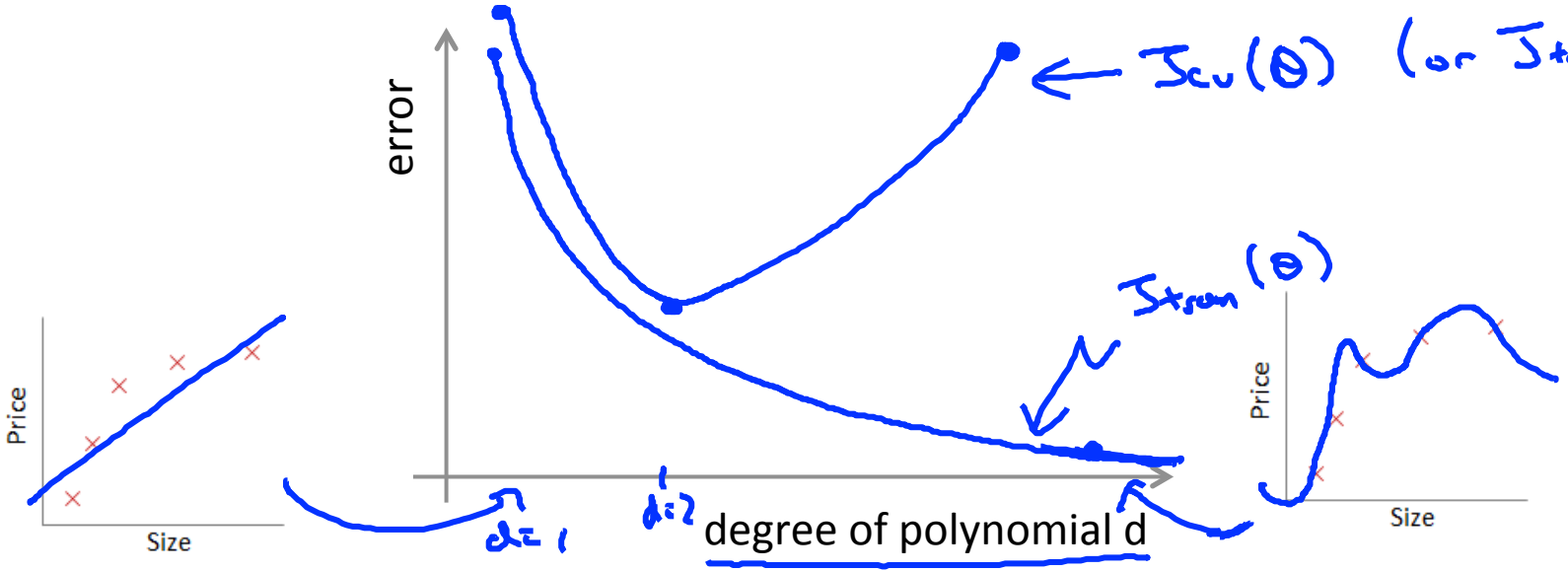
High variance
(overfit)

$d=4$

Bias/variance

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

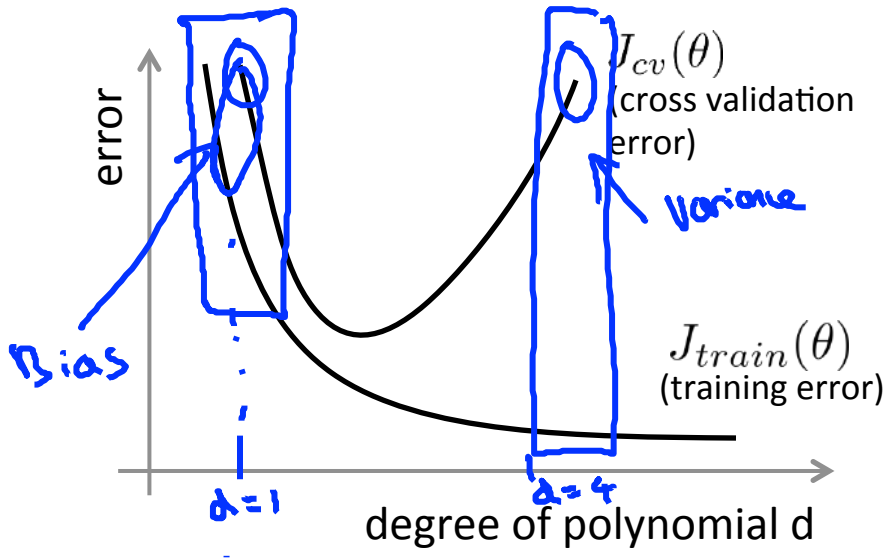
Cross validation error: $J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$ (or $J_{test}(\theta)$)



Andrew Ng

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

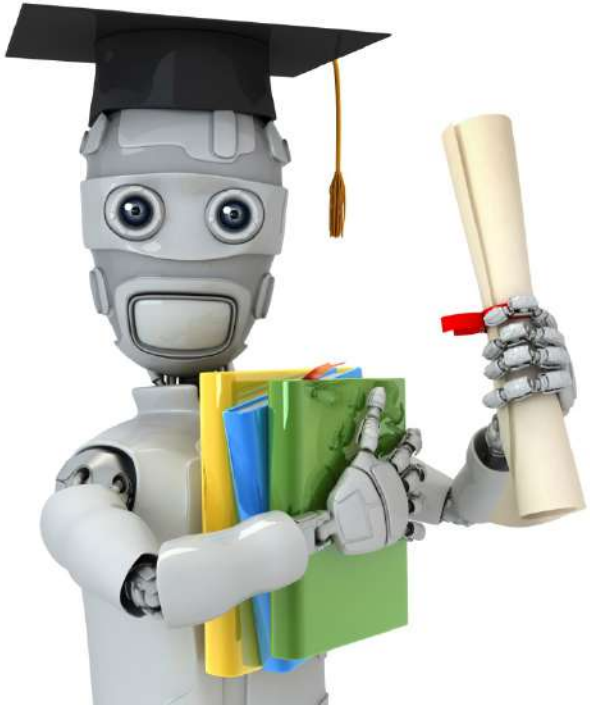
$J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit):

→ $J_{train}(\theta)$ will be low

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

$\gg$



Machine Learning

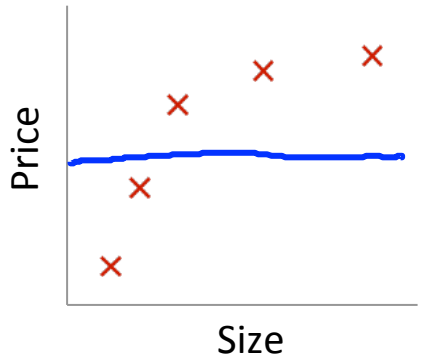
Advice for applying machine learning

Regularization and bias/variance

Linear regression with regularization

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

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

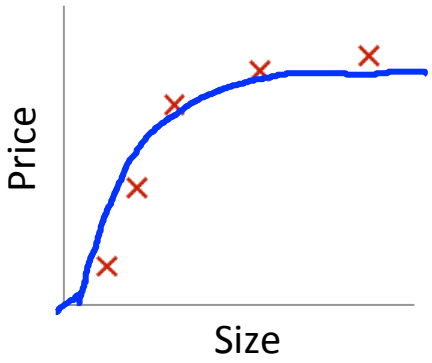


Large λ

High bias (underfit)

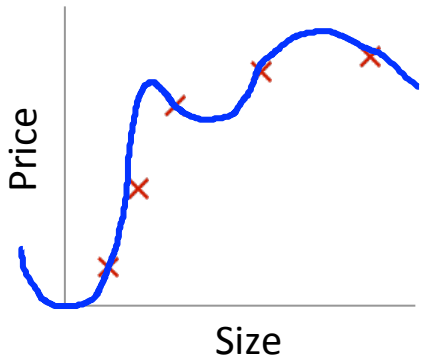
$\lambda = 10000.\ \theta_1 \approx 0, \theta_2 \approx 0, \ldots$

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



Intermediate λ

"Just right"



Small λ

High variance (overfit)

$\lambda = 0$

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 $\lambda = 0.01$ $\longrightarrow$ $\min_\theta J(\theta) \rightarrow \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$
3. Try $\lambda = 0.02$ $\longrightarrow$ $\theta^{(3)} \rightarrow J_{cv}(\theta^{(3)})$
4. Try $\lambda = 0.04$ $\vdots$
5. Try $\lambda = 0.08$ $\longrightarrow$ $\theta^{(5)}$ $J_{cv}(\theta^{(5)})$

   $\vdots$

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

0.24

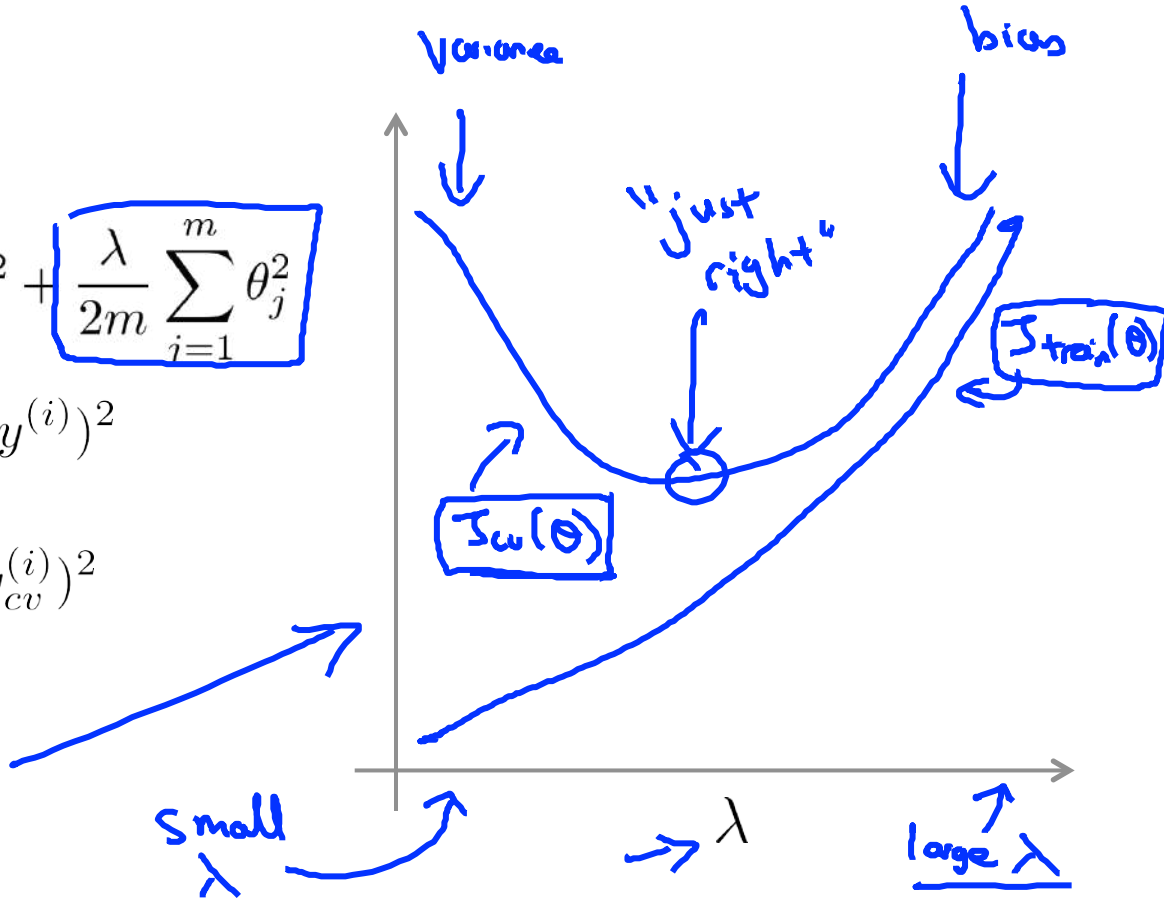
Pick (say) $\theta^{(5)}$. Test error: $J_{test}(\theta^{(5)})$

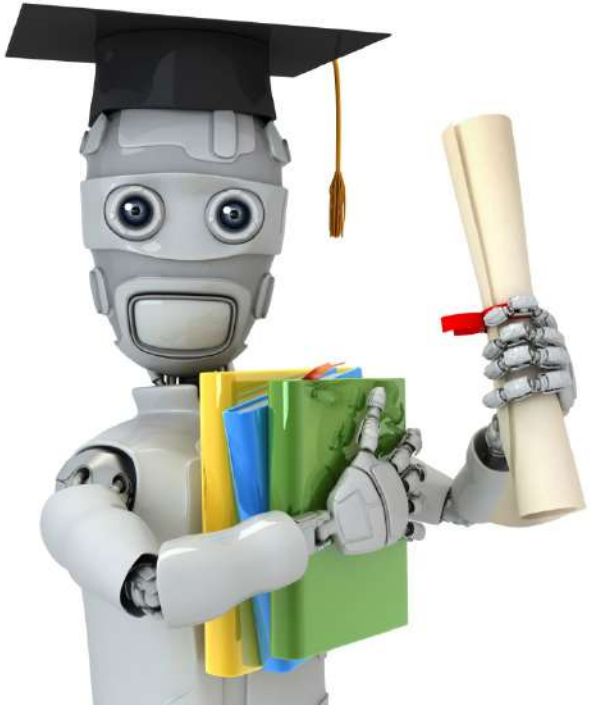
Bias/variance as a function of the regularization parameter λ

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

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

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





Machine Learning

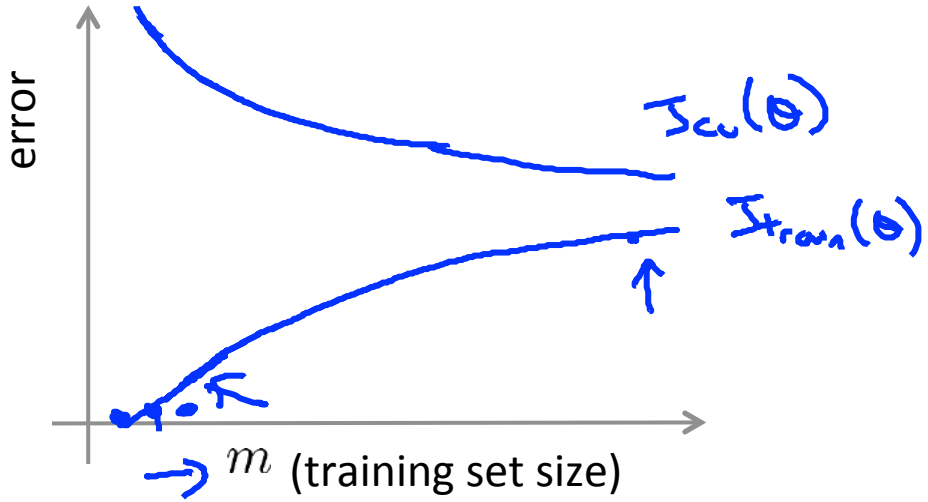
Advice for applying machine learning

Learning curves

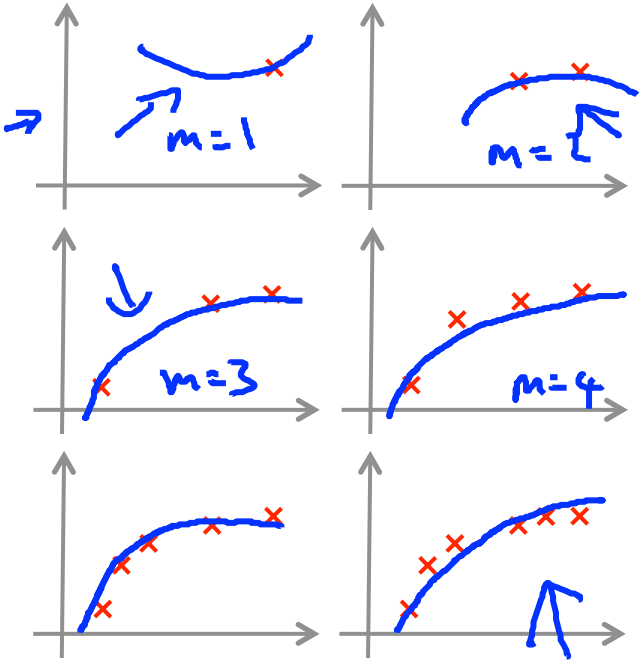
Learning curves

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

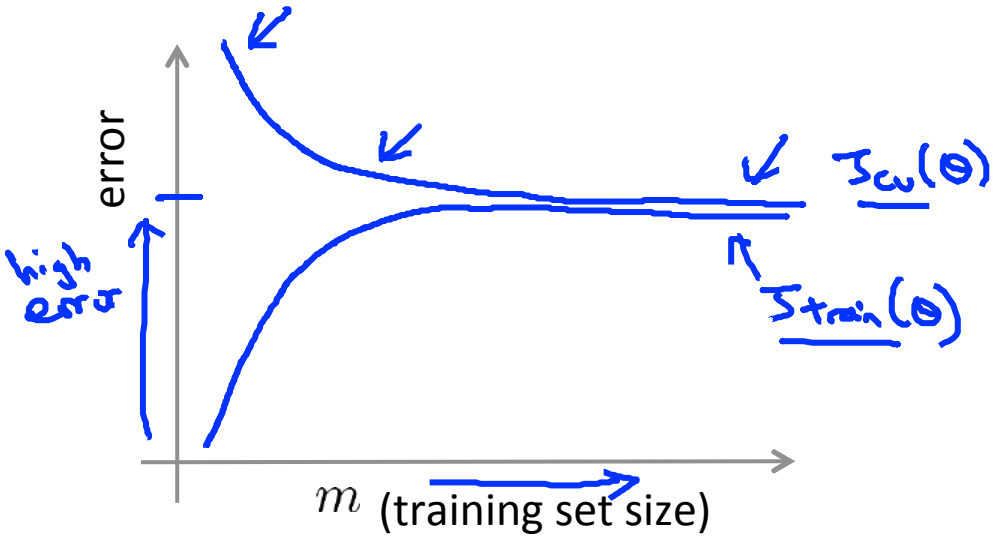
$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

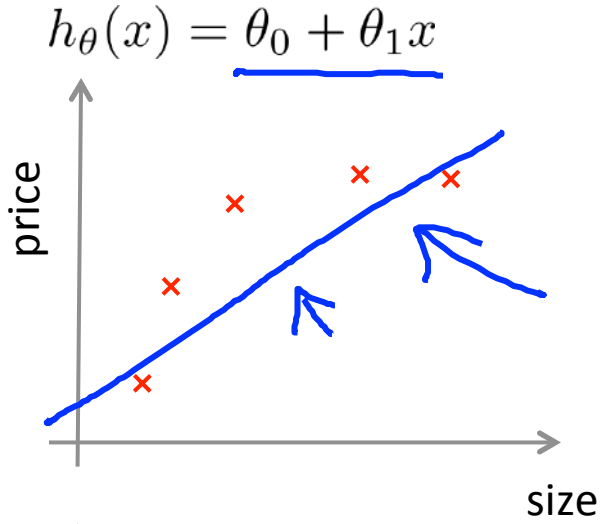


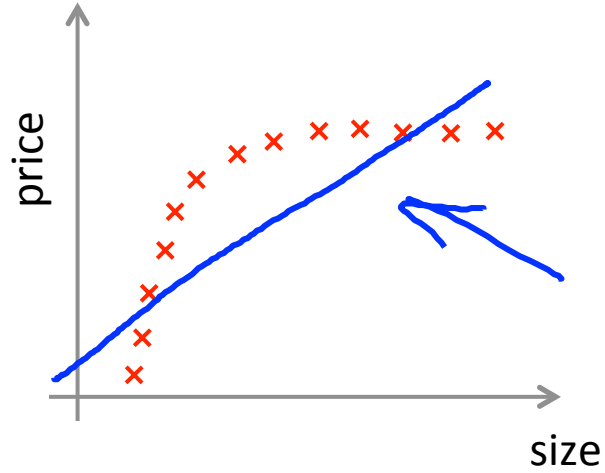
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$



High bias





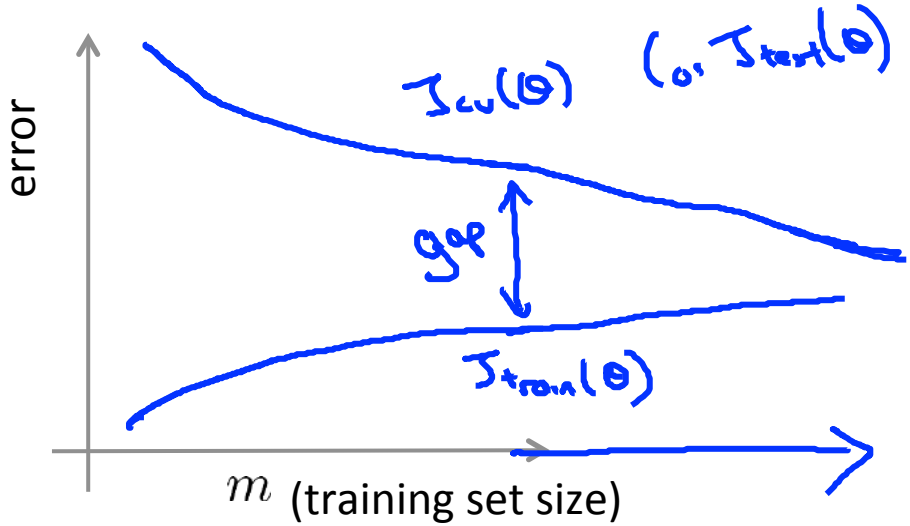


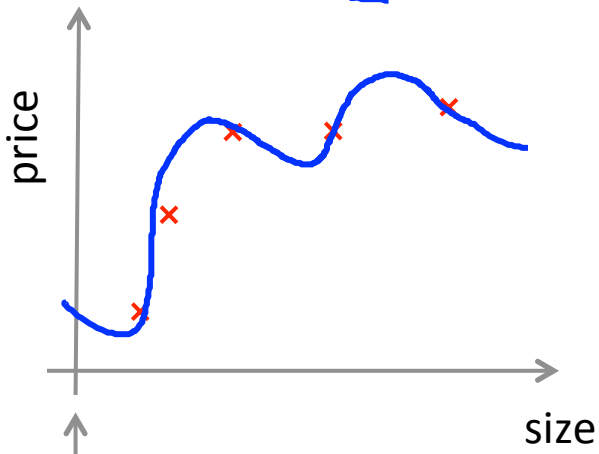
If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.

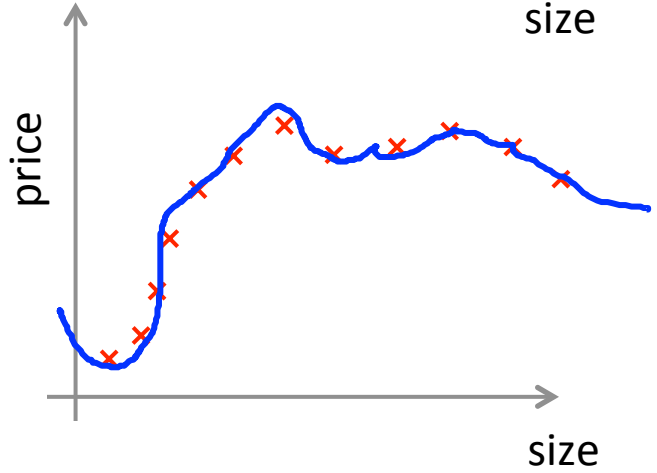
High variance

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

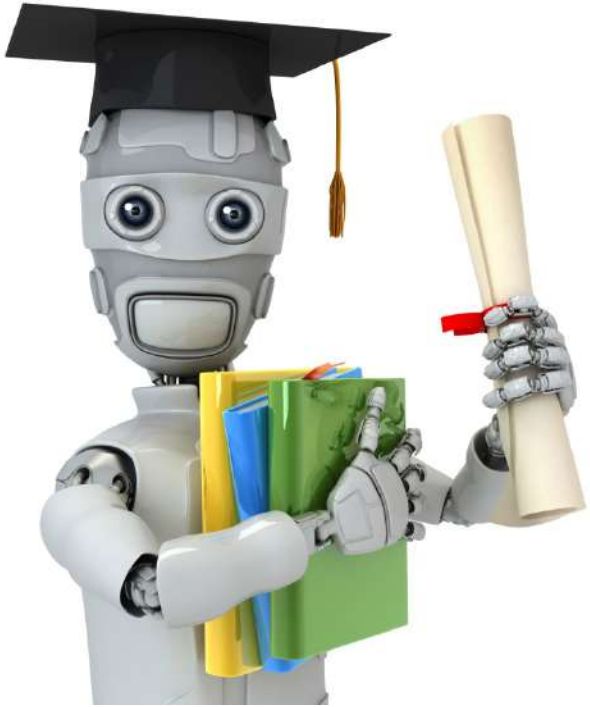
(and small λ)







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



Machine Learning

Advice for applying machine learning

Deciding what to try next (revisited)

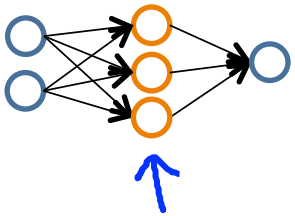
Debugging a learning algorithm:

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

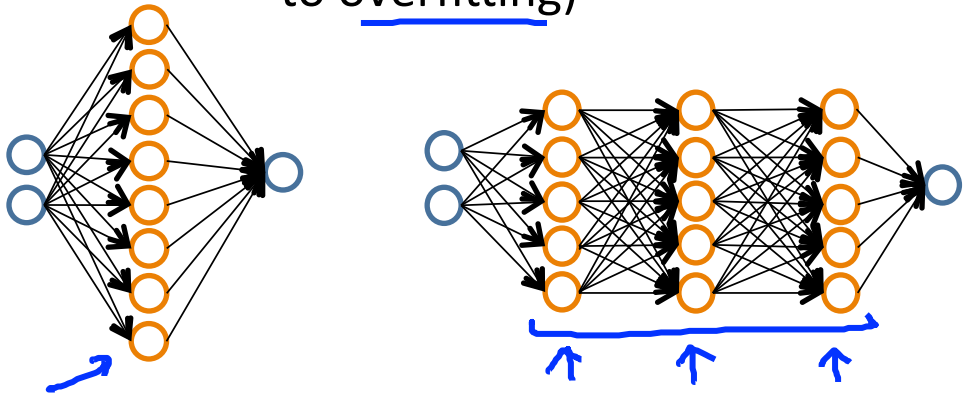
Neural networks and overfitting

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



Computationally cheaper

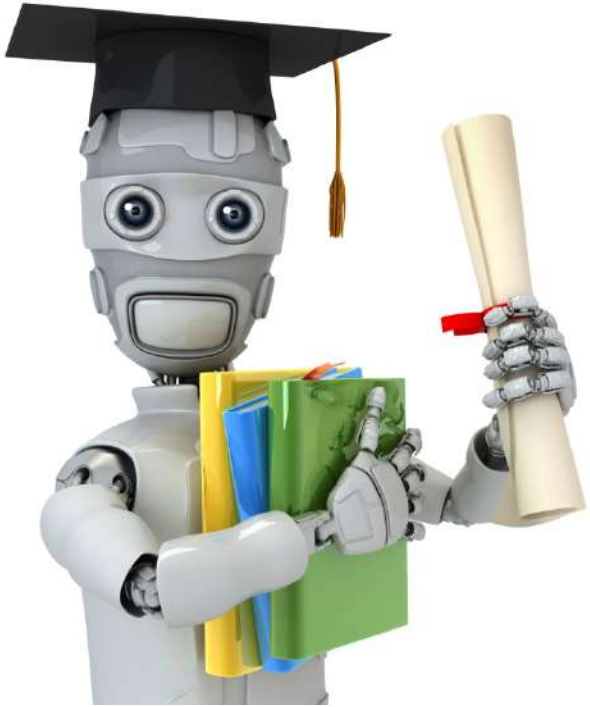
"Large" neural network (more parameters; more prone to overfitting)



Computationally more expensive.

Use regularization (λ) to address overfitting.

$J_{cv}(\Theta)$



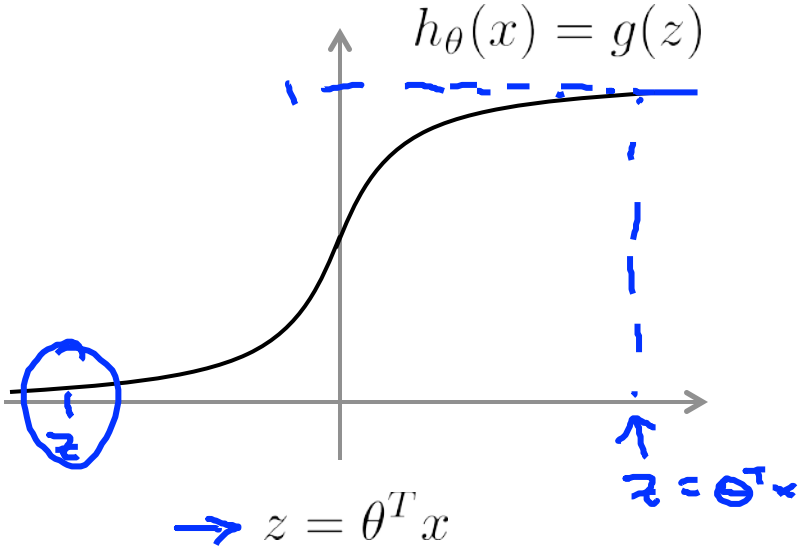
Machine Learning

Support Vector Machines

Optimization objective

Alternative view of logistic regression

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



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

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

Alternative view of logistic regression

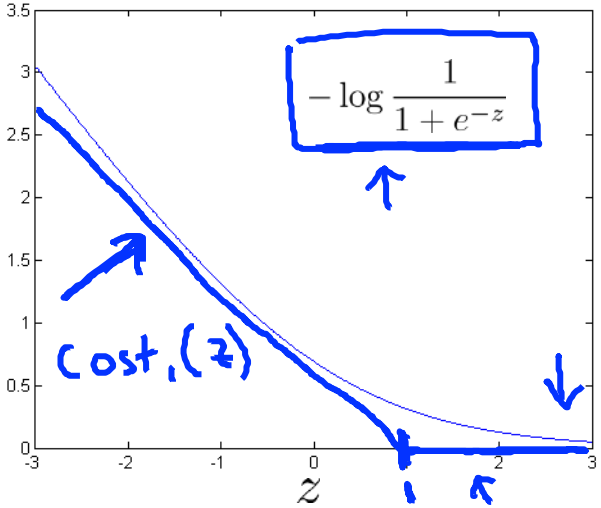
(x, y)

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

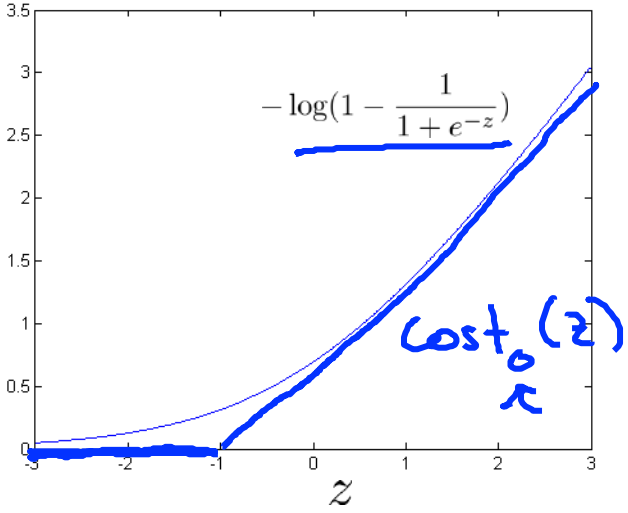
$$= -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y)\log\left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$$

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

$z = \theta^T x$



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



Support vector machine

Logistic regression:

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

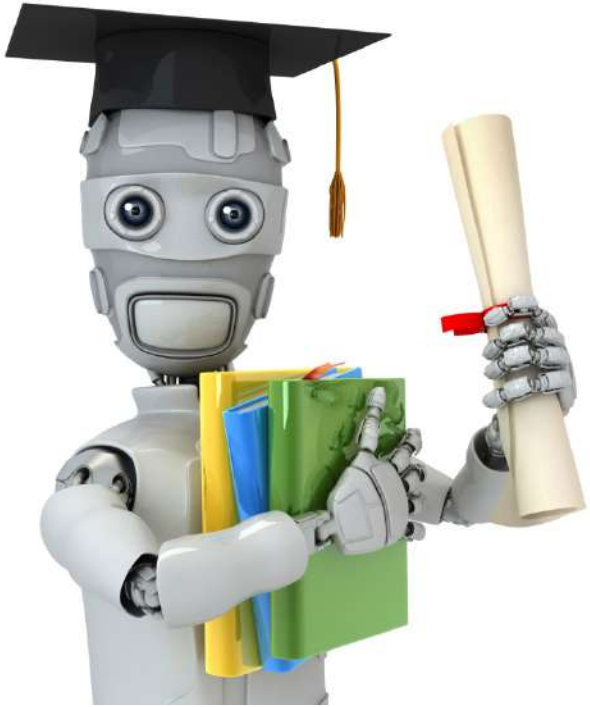
Support vector machine:

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

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:



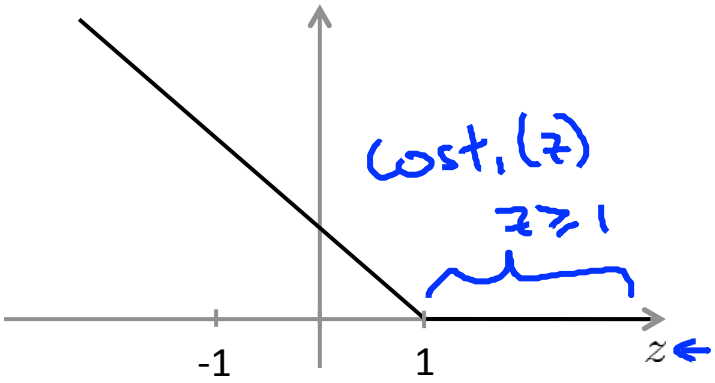
Machine Learning

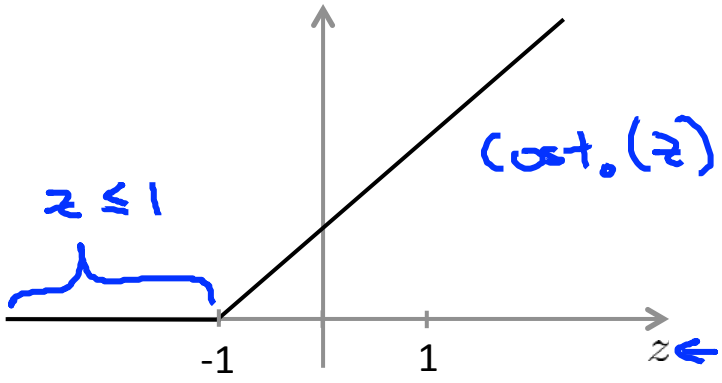
Support Vector Machines

Large Margin Intuition

Support Vector Machine

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





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

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

$C = 100{,}000$

SVM Decision Boundary

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

$= 0$

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

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

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

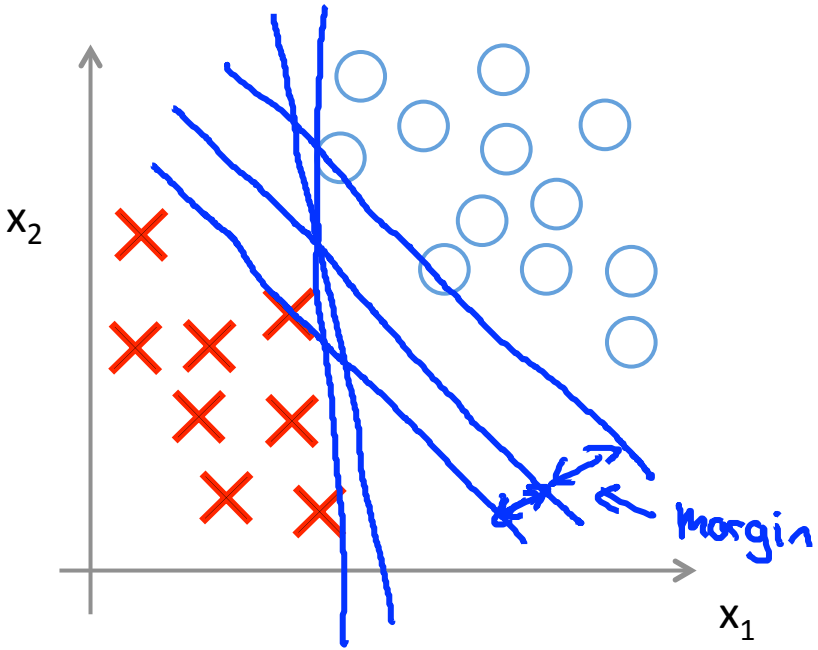
s.t. $\theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$

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

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

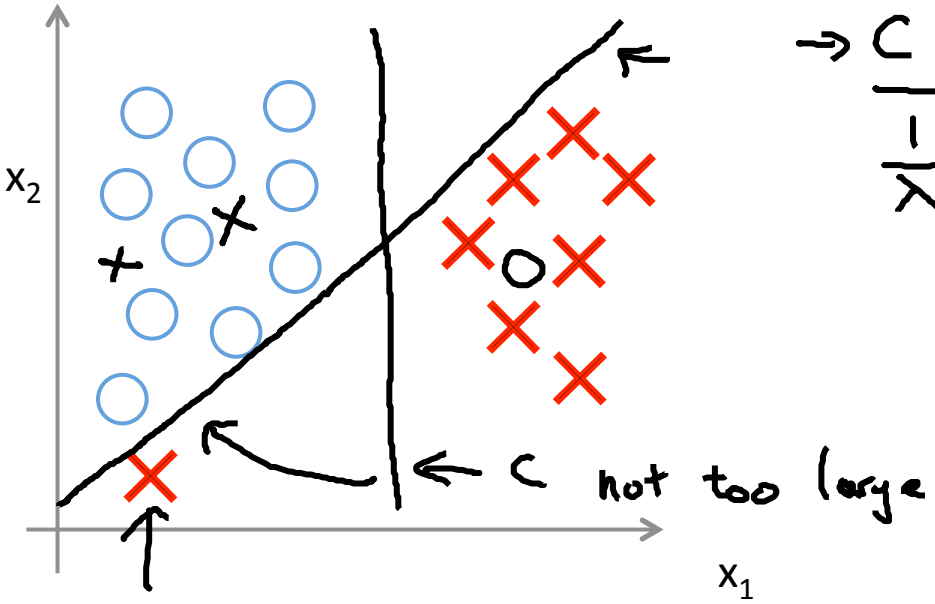
$\theta^T x^{(i)} \leq -1$

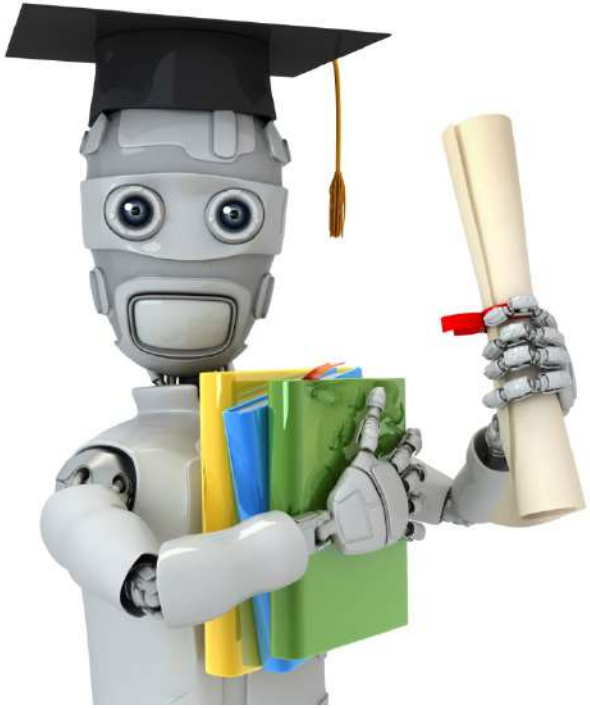
SVM Decision Boundary: Linearly separable case



Large margin classifier

Large margin classifier in presence of outliers



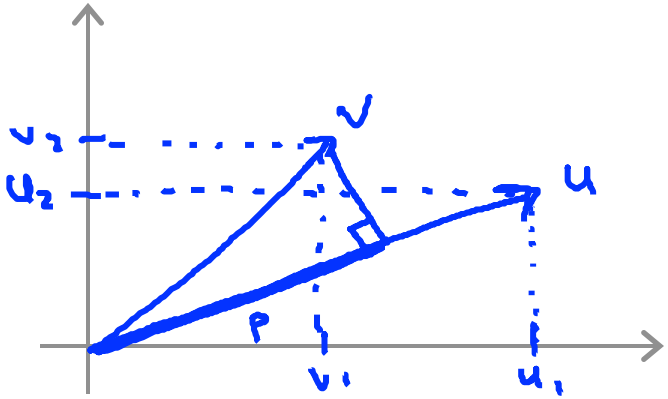


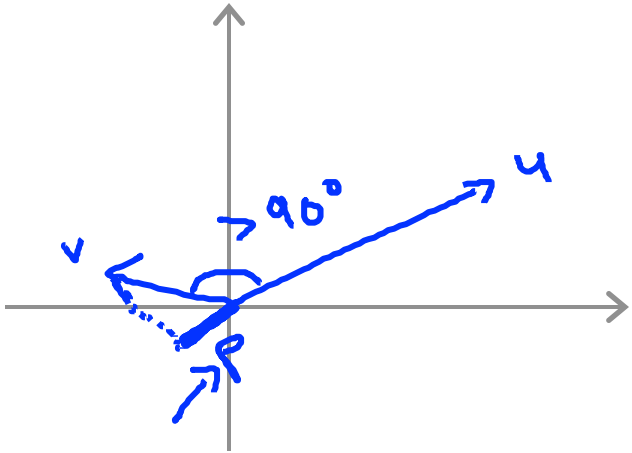
Machine Learning

Support Vector Machines

The mathematics behind large margin classification (optional)

Vector Inner Product





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

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

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

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

p = length of projection of v onto u.

Signed

$u^T v = \underline{p} \cdot \underline{\|u\|} \leftarrow \qquad = v^T u$

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

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

$p < 0$

SVM Decision Boundary

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

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

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

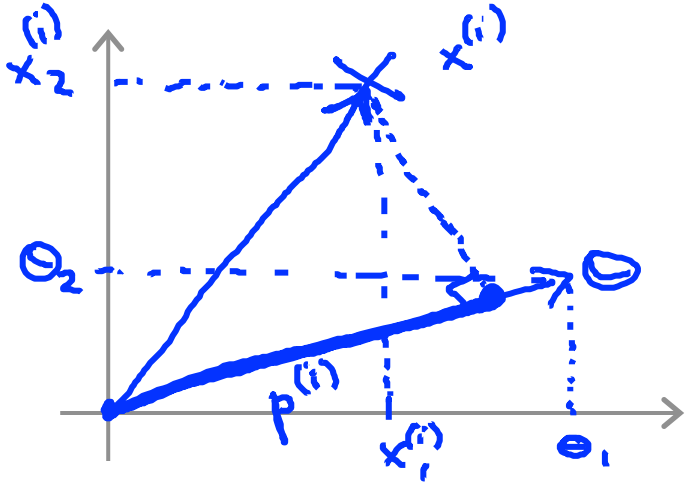
$$\theta^T x^{(i)} \leq -1 \quad \text{if } y^{(i)} = 0$$

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

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

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

$$u^T v$$



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

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

SVM Decision Boundary

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

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

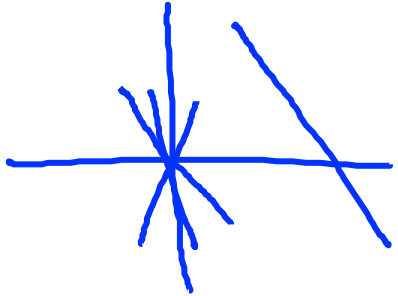
$$p^{(i)} \cdot \|\theta\| \leq -1 \quad \text{if } y^{(i)} = 1$$

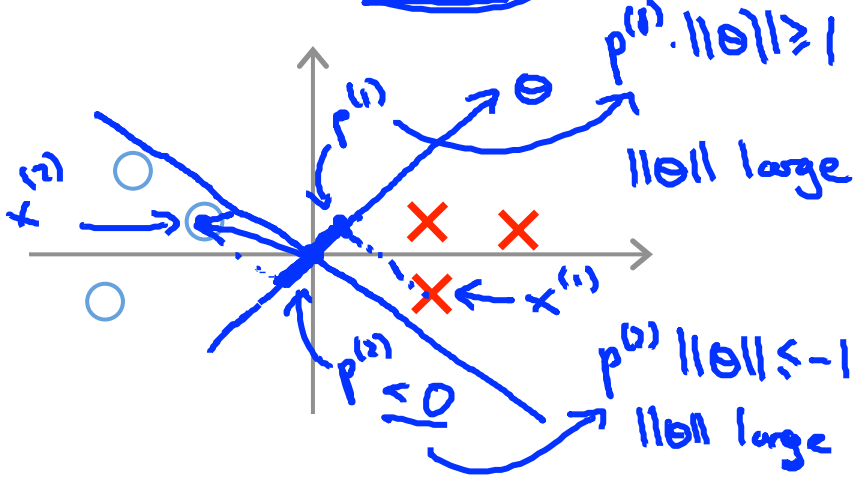
C very large

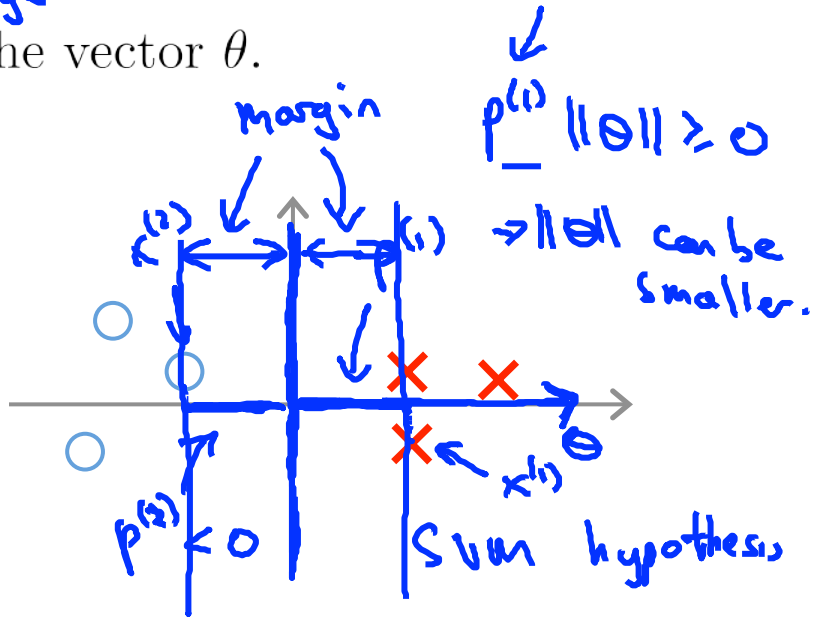
where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

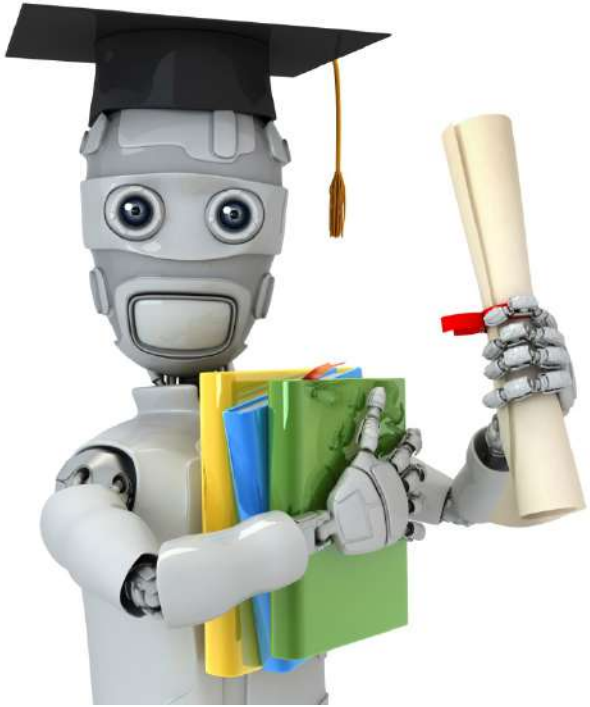
Simplification: $\theta_0 = 0$

$\theta_0 \neq 0$







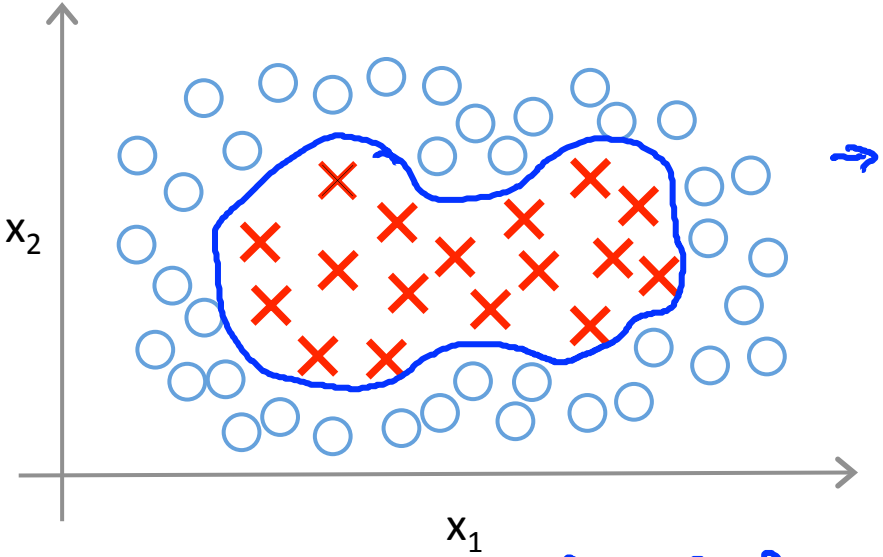


Support Vector Machines

Kernels I

Machine Learning

Non-linear Decision Boundary



Predict $y = 1$ if

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

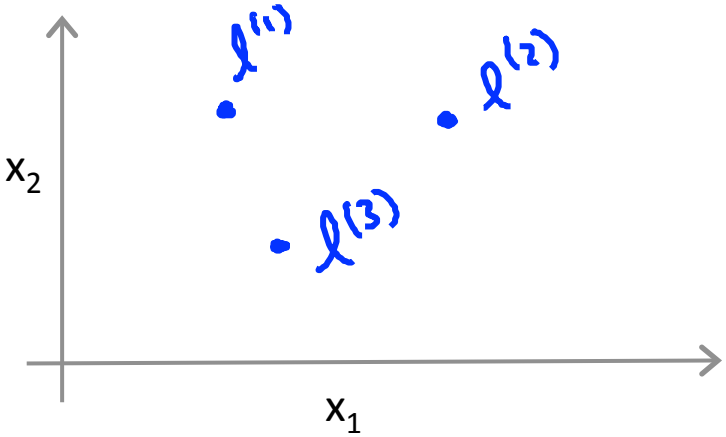
$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \cdots \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

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

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

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

$\|w\|$

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

Given x:

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

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

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

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

Kernel (Gaussian kernels)

Kernels and Similarity

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

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

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

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

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

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

x

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

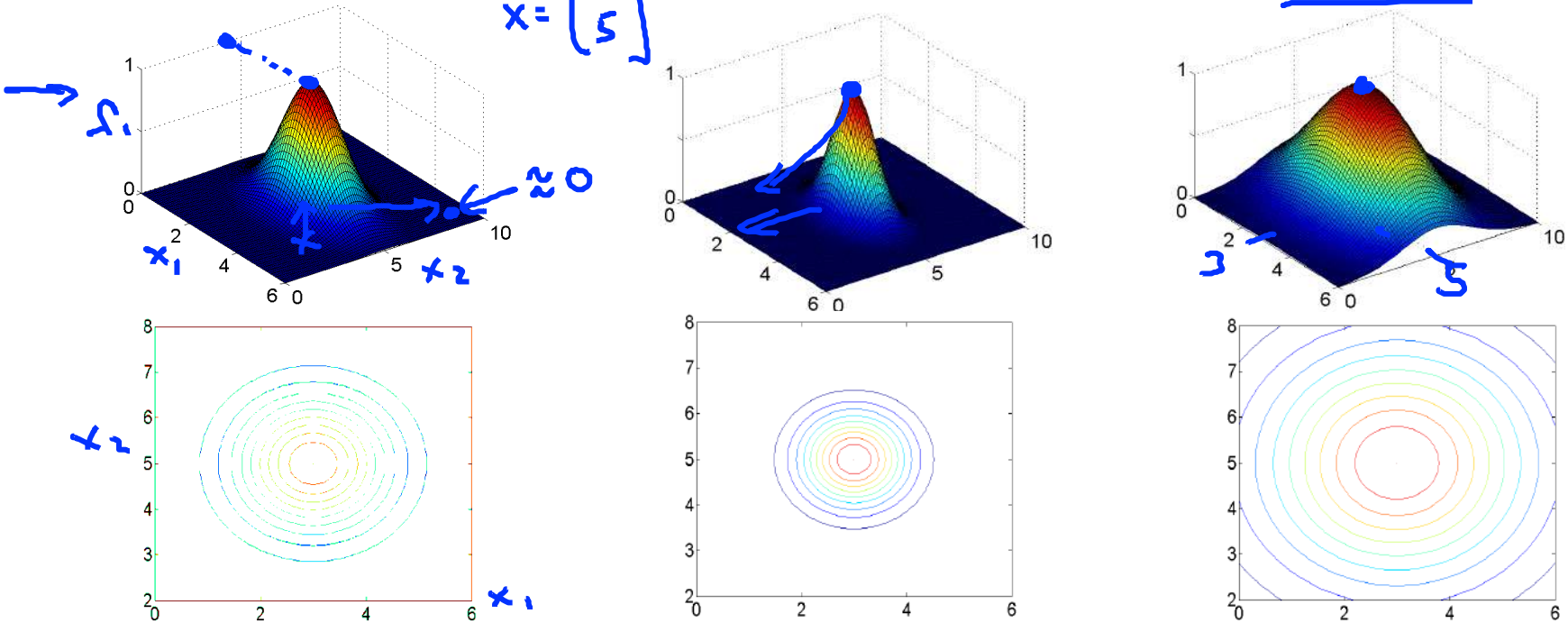
$$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0.$$

Example:

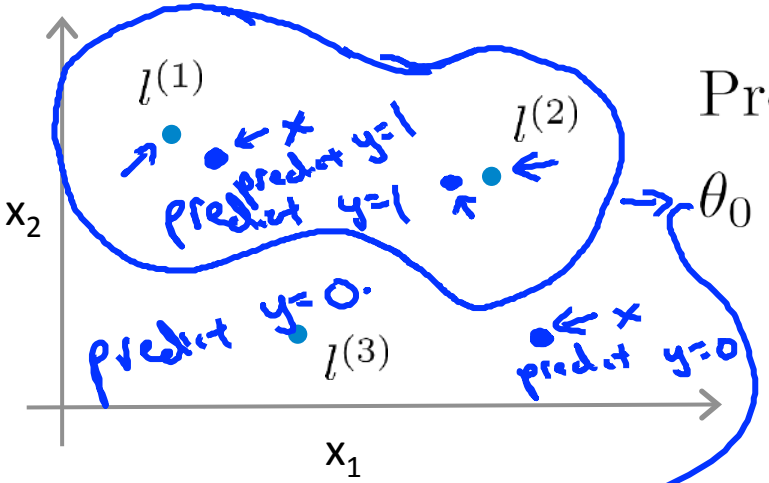
$$l^{(1)} = \boxed{\begin{bmatrix} 3 \\ 5 \end{bmatrix}}, \quad f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\boxed{\sigma^2}}\right)$$

$\sigma^2 = 1$ $\qquad$ $\sigma^2 = 0.5$ $\qquad$ $\sigma^2 = 3$

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



Andrew Ng



Predict "1" when

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

x

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

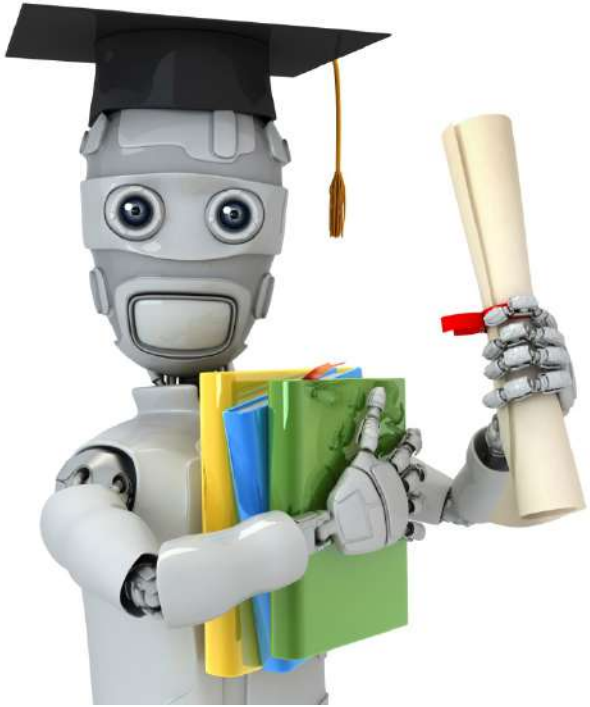
$f_1 \approx 1, \quad f_2 \approx 0, \quad f_3 \approx 0.$

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

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

$f_1, f_2, f_3 \approx 0$

$\theta_0 + \theta_1 f_1 + \dots \approx -0.5 < 0$

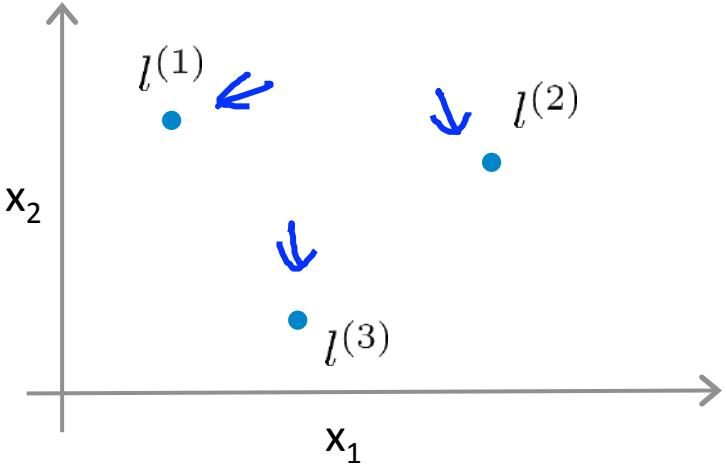


Machine Learning

Support Vector Machines

Kernels II

Choosing the landmarks

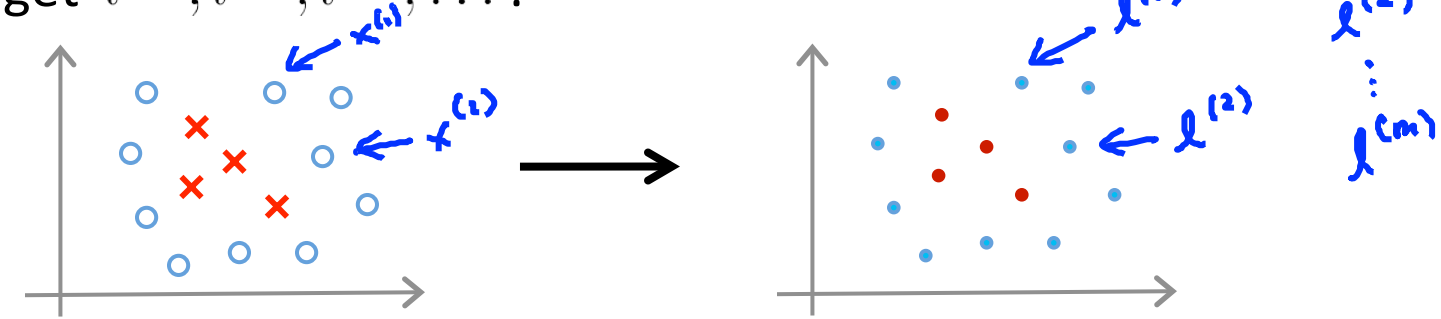


Given x:

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

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

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



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:

$$f_1 = \text{similarity}(x, l^{(1)})$$

$x^{(1)}$

$$f_2 = \text{similarity}(x, l^{(2)})$$

$$\ldots$$

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

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

$x^{(i)} \rightarrow$

$$f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)})$$

$$f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)})$$

$$\vdots$$

$$f_m^{(i)} = \text{sim}(x^{(i)}, l^{(m)})$$

$x^{(i)}$

$$f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)}) = \exp\left(-\frac{0}{2\sigma^2}\right) = 1$$

$x^{(i)} \in \mathbb{R}^{n+1}$ (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 $f \in \mathbb{R}^{m+1}$ $\quad \theta \in \mathbb{R}^{m+1}$

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

Training:

$n = m$

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

$\cancel{\theta^T x^{(i)}}$ $\quad \theta^T f^{(i)}$ $\quad m = m$ $\quad \rightarrow \theta_0$

$$\sum_j \theta_j^2 = \theta^T \theta \leftarrow \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$$

$\|\theta\|^2$

(ignore θ_0)

$\theta^T M \theta$ $\quad M = 10{,}000$

SVM parameters:

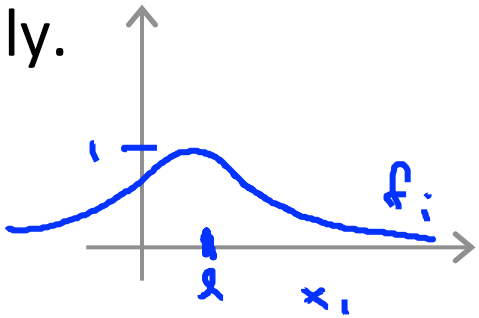
C ($= \frac{1}{\lambda}$). → Large C: Lower bias, high variance. (small λ)

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

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

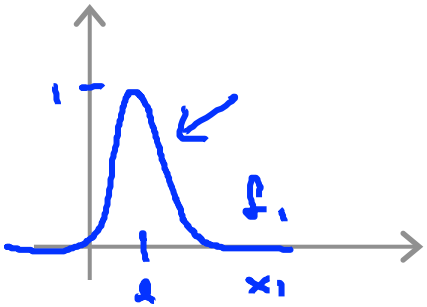
→ Higher bias, lower variance.

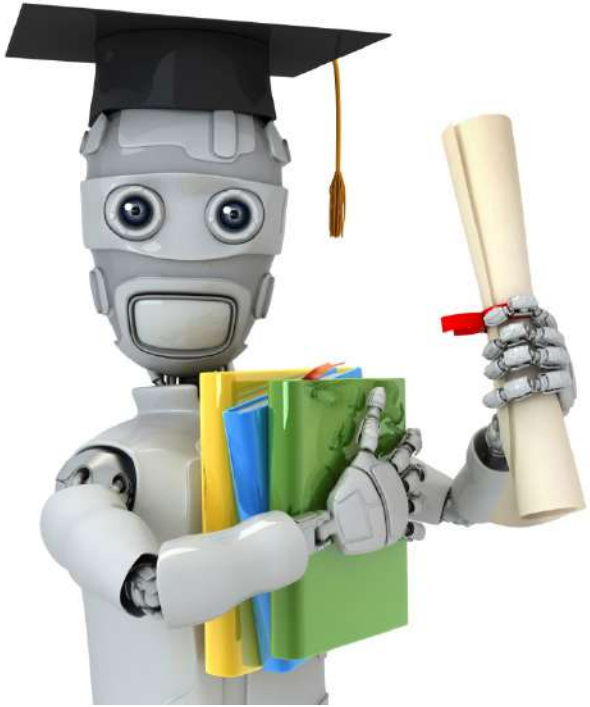
$\exp\left(-\frac{\|x-l^{(i)}\|^2}{2\sigma^2}\right)$



Small σ^2: Features f_i vary less smoothly.

Lower bias, higher variance.





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 $\theta^T x \geq 0$

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

→ 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
```

$$x \rightarrow \begin{matrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{matrix}$$

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

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

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

$v = x - l$

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

$= \underbrace{(x_1 - l_1)^2}_{1000 \text{ feet}^2} + \underbrace{(x_2 - l_2)^2}_{1\text{-}5 \text{ bedrooms}} + \cdots + (x_n - l_n)^2$

Other choices of kernel

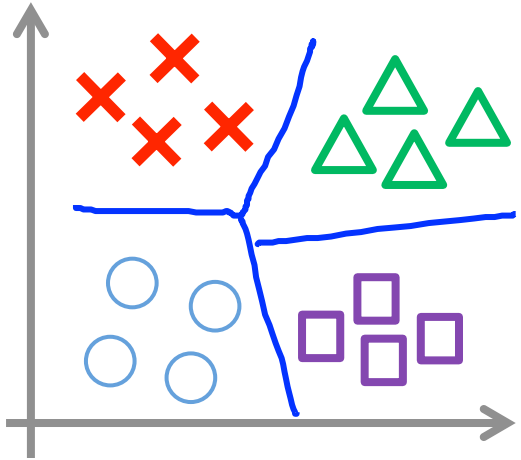
Note: Not all similarity functions $\text{similarity}(x, l)$ make valid kernels.
(Need to satisfy technical condition called "Mercer's Theorem" to make sure SVM packages' optimizations run correctly, and do not diverge).

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

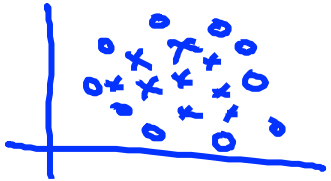
Pick class i with largest $(\theta^{(i)})^T x$

$y=1$ $y=2$ $\ldots$ $\theta=K$

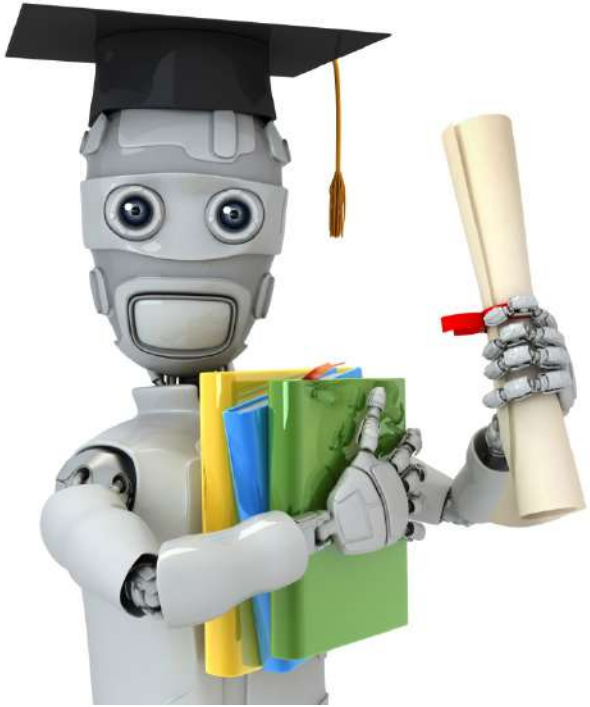
Logistic regression vs. SVMs

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: ($n = 1 - 1000$, $m = 50{,}000+$)
  - 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.

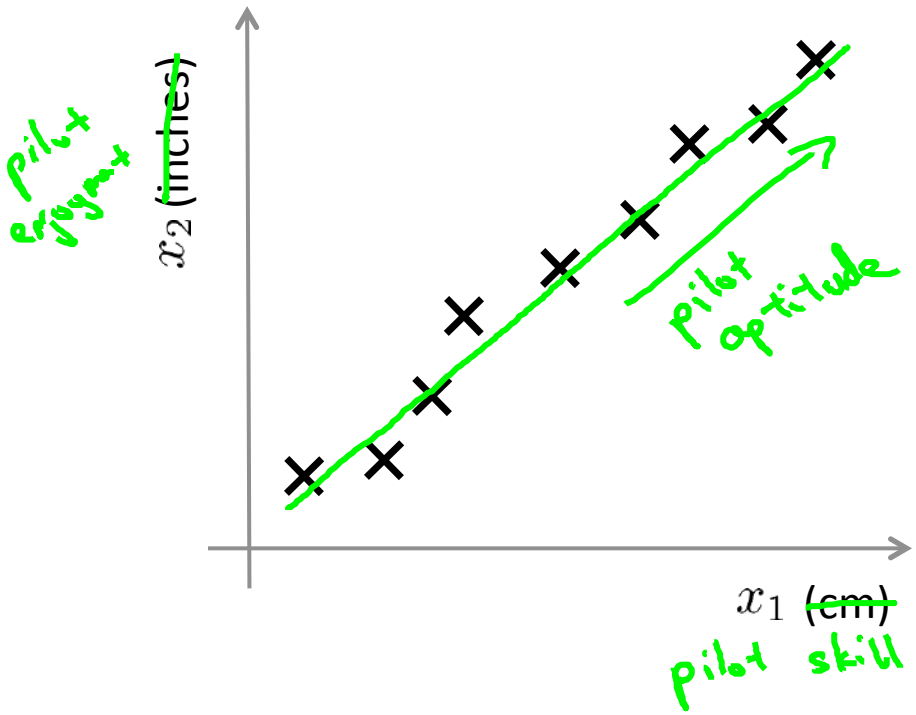


Machine Learning

Dimensionality Reduction

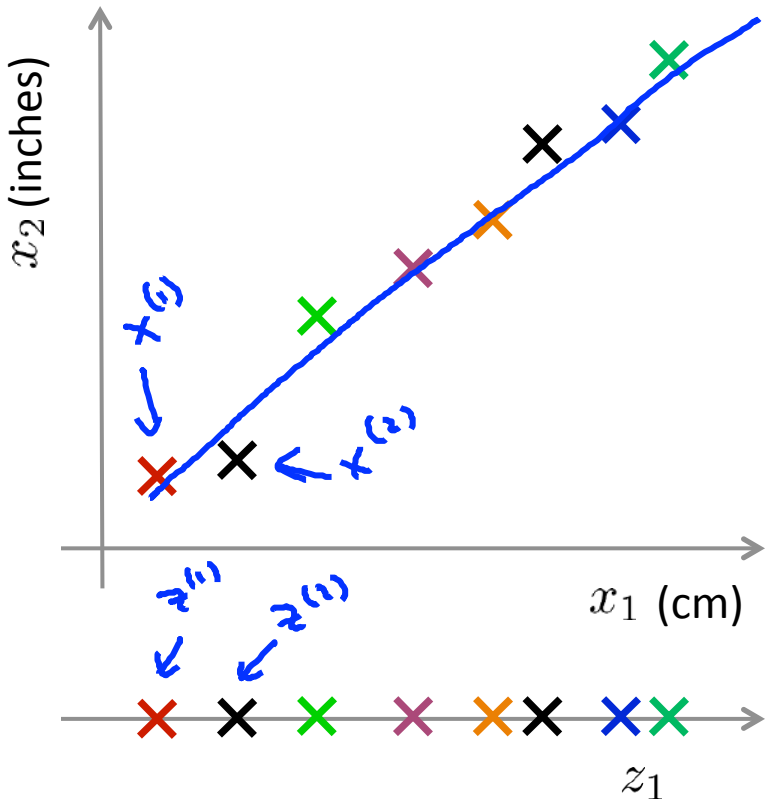
Motivation I: Data Compression

Data Compression



Reduce data from 2D to 1D

Data Compression



Reduce data from 2D to 1D

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

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

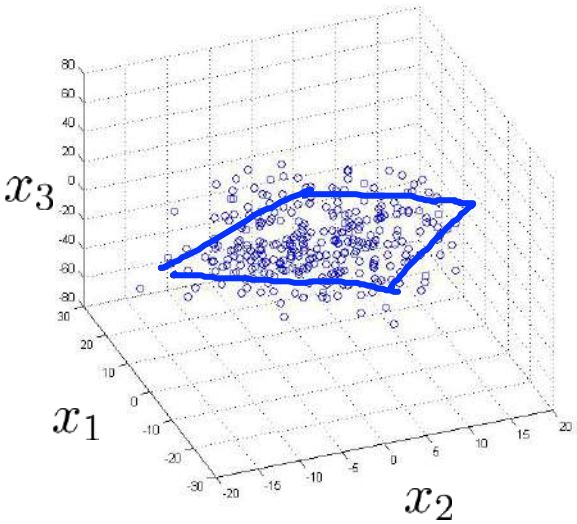
$$\vdots$$

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

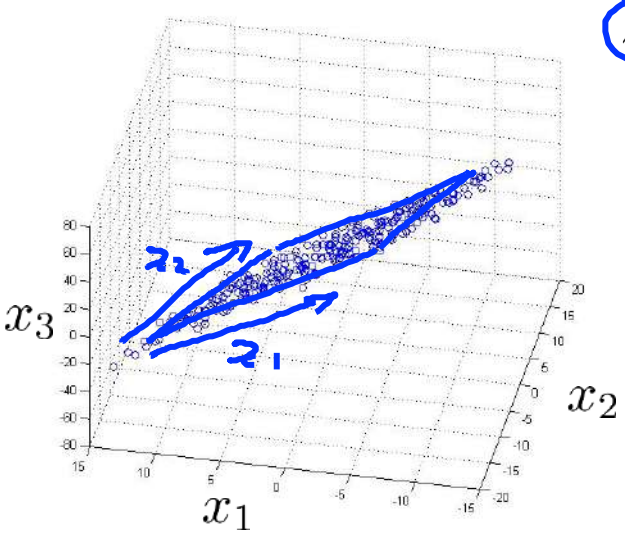
Data Compression

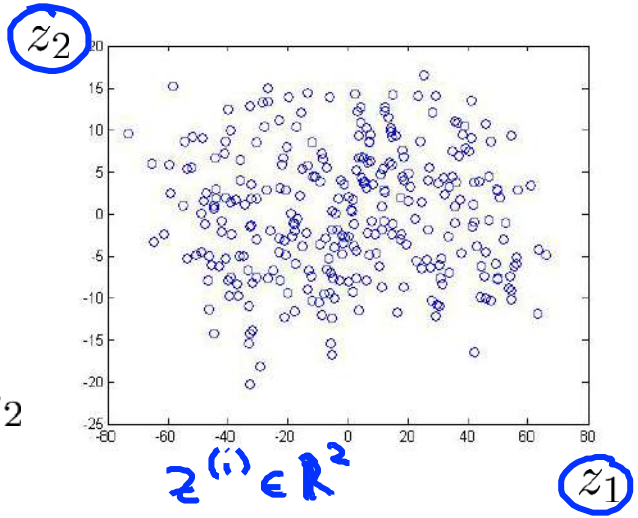
1000D → 100D

Reduce data from 3D to 2D



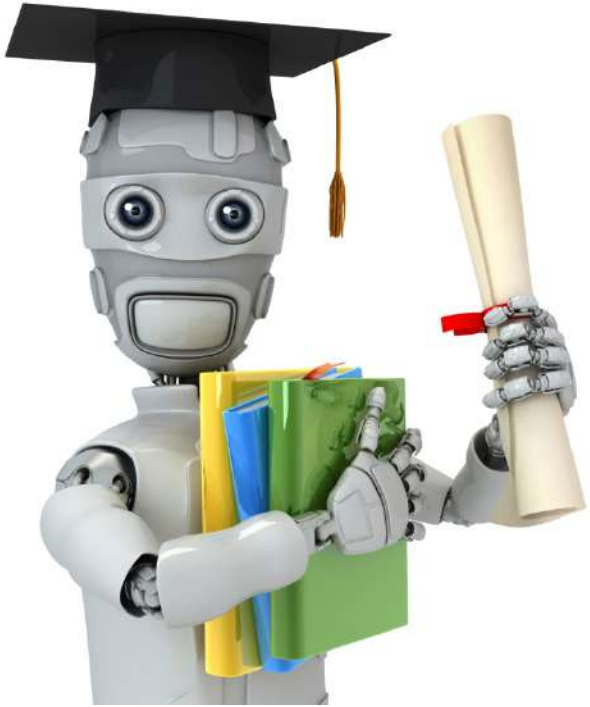
$x^{(i)} \in \mathbb{R}^3$





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

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



Machine Learning

Dimensionality Reduction

Motivation II: Data Visualization

Data Visualization

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

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]

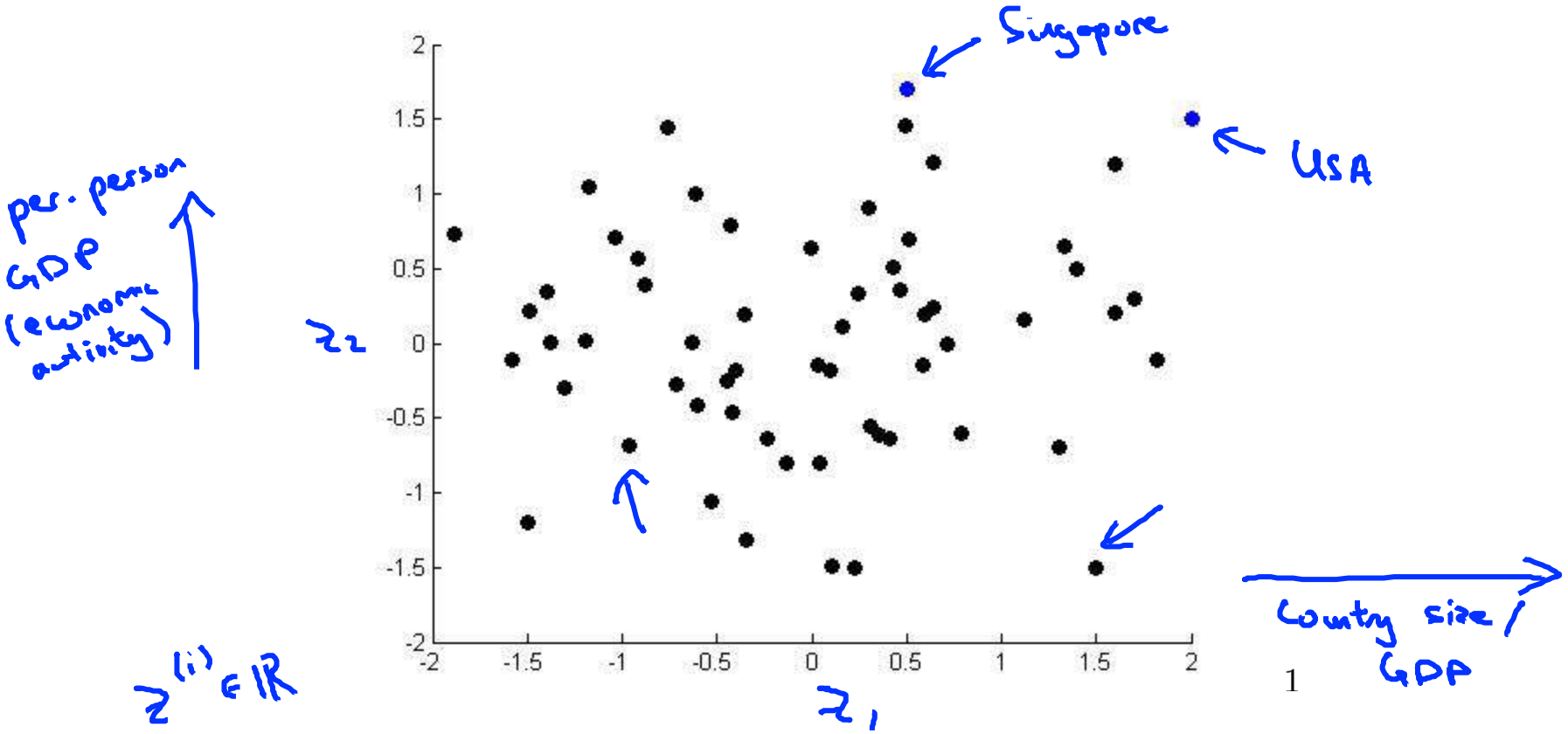
Data Visualization

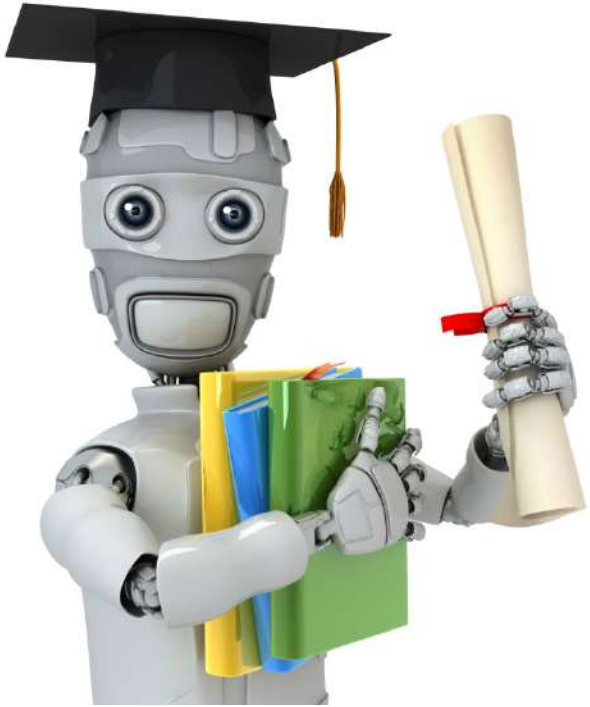
$z^{(i)} \in \mathbb{R}^2$

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





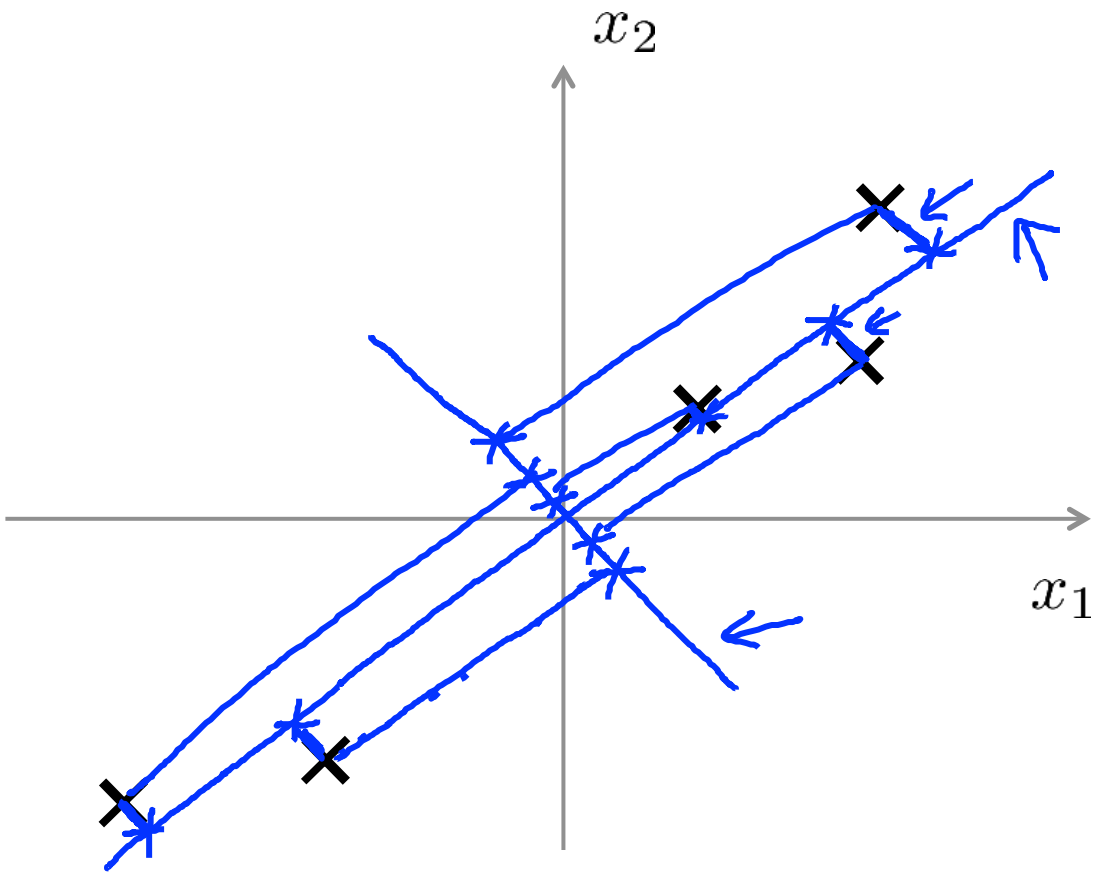
Machine Learning

Dimensionality Reduction

Principal Component Analysis problem formulation

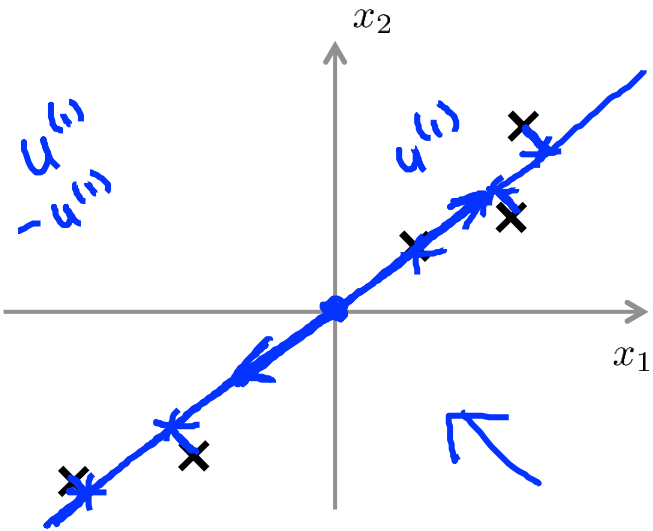
Principal Component Analysis (PCA) problem formulation

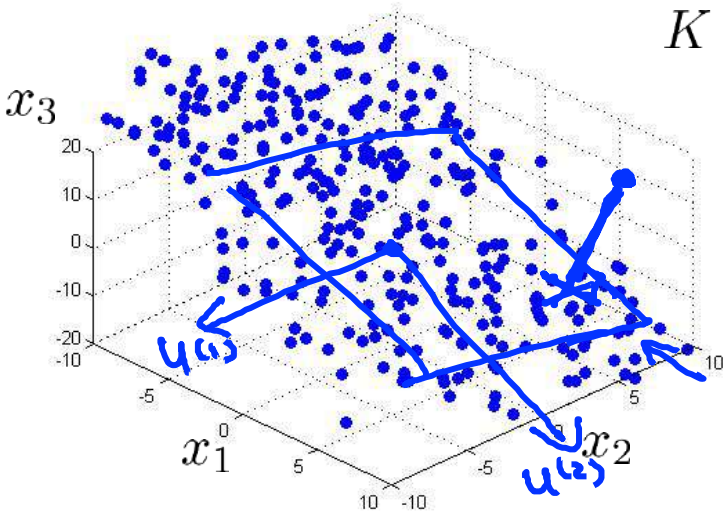
$x \in \mathbb{R}^2$



Principal Component Analysis (PCA) problem formulation

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

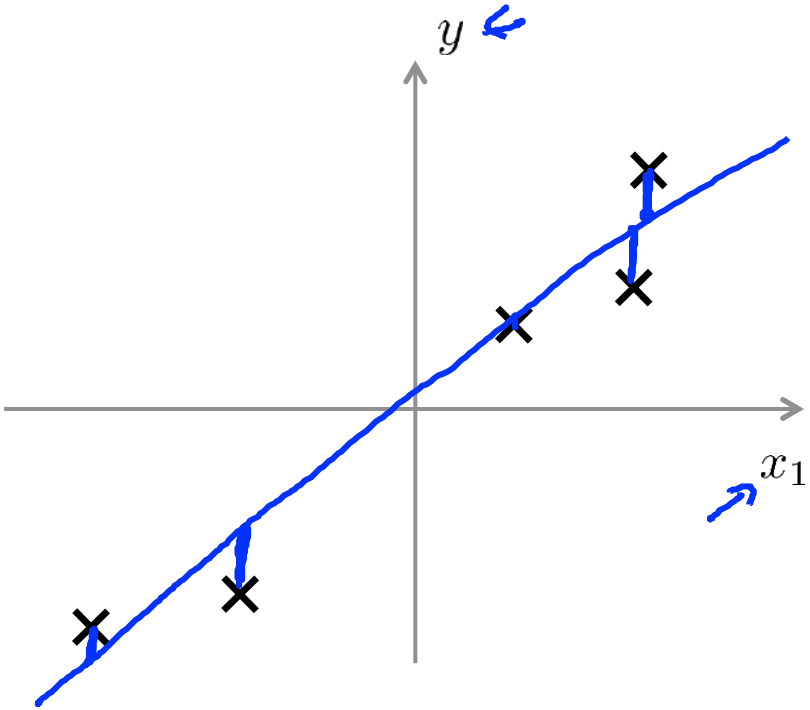


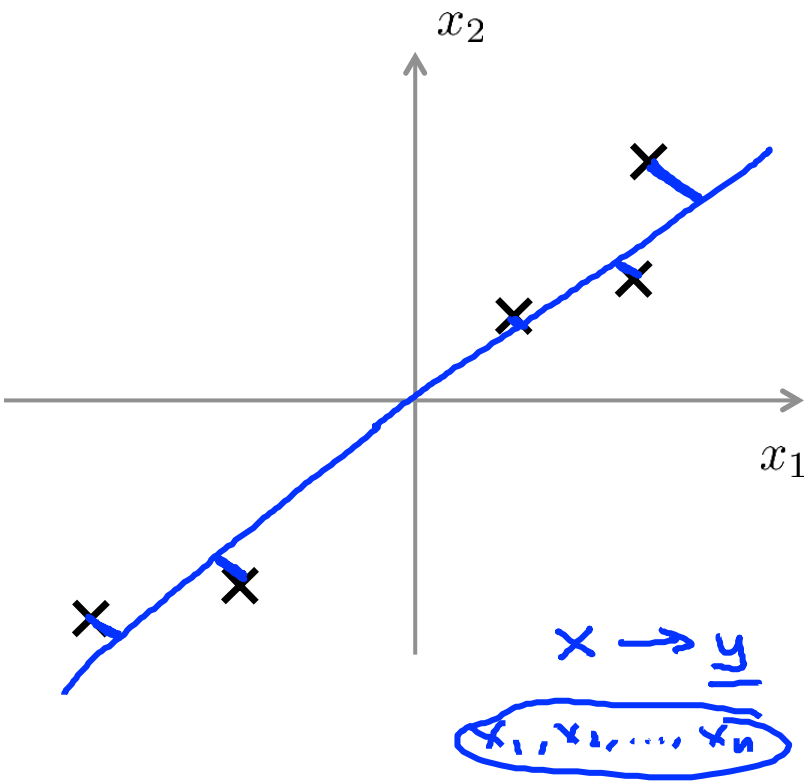


Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

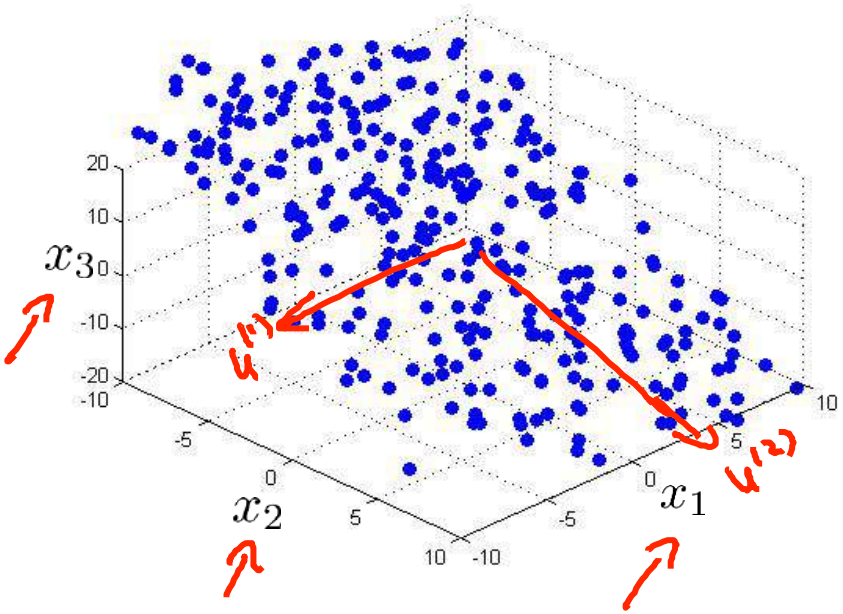
Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

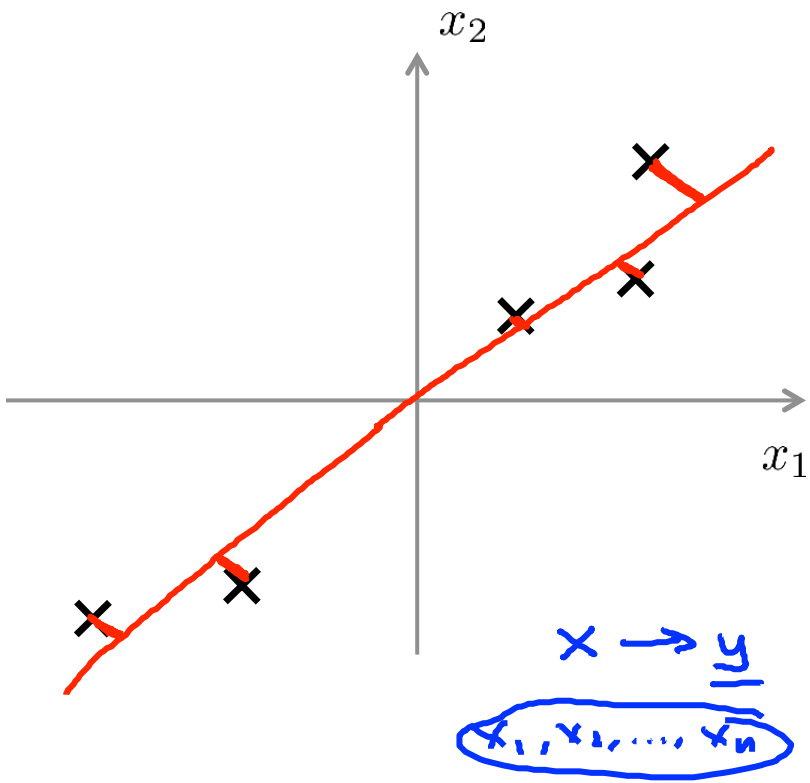
PCA is not linear regression

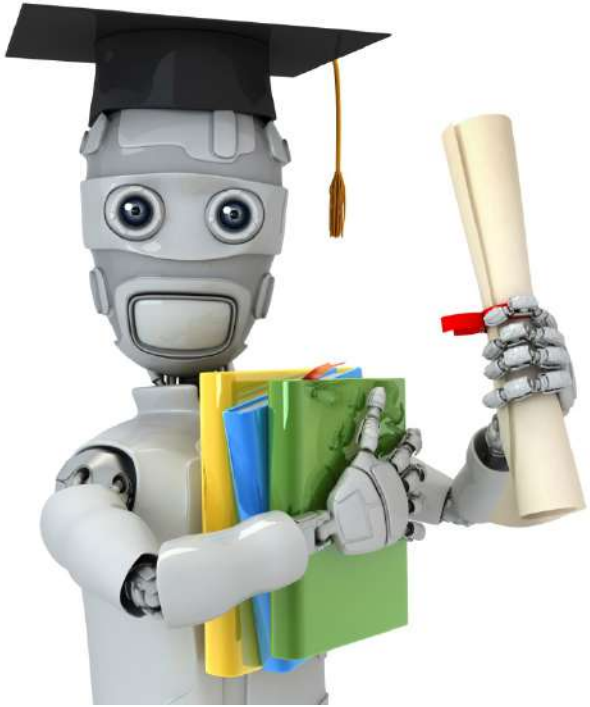




PCA is not linear regression







Machine Learning

Dimensionality Reduction

Principal Component Analysis algorithm

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

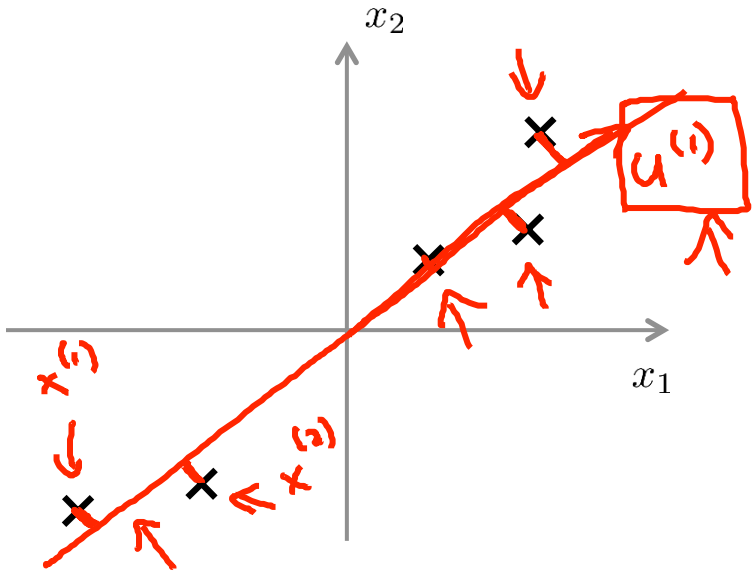
$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

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.

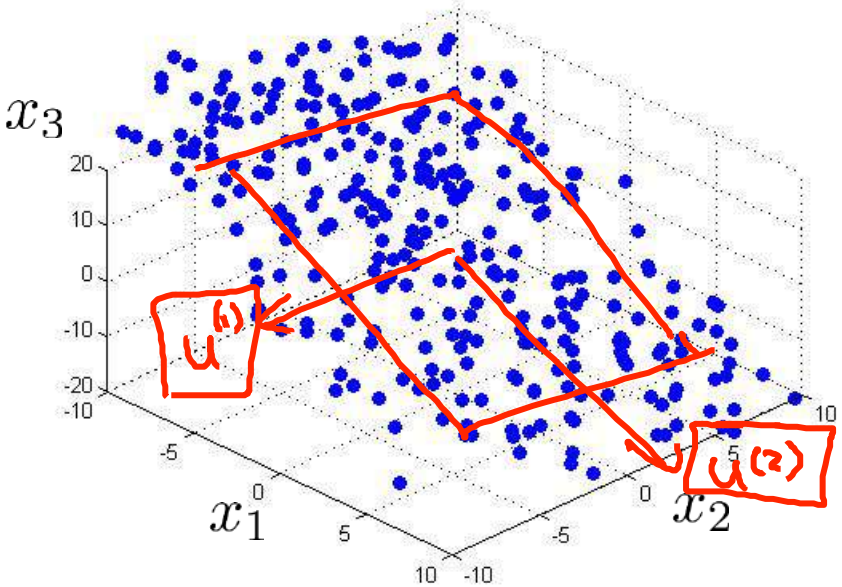
$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

Principal Component Analysis (PCA) algorithm



Reduce data from 2D to 1D

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



Reduce data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

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

$n \times n$

Sigma

Compute "eigenvectors" of matrix Σ:

→ Singular value decomposition

eig (Sigma)

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

$n \times n$ matrix

$$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(n)} \\ | & | & | & & | \end{bmatrix}$$

k

$U \in \mathbb{R}^{n\times n}$

$u^{(1)}, \ldots, u^{(k)}$

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

$x \in \mathbb{R}^n \longrightarrow z \in \mathbb{R}^k$

$$z^{(i)} = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T}_{n \times k} x^{(i)} = \underbrace{\underbrace{\begin{bmatrix} — (u^{(1)})^T — \\ \vdots \\ — (u^{(k)})^T — \end{bmatrix}}_{k \times n} \underbrace{x^{(i)}}_{n \times 1}}_{k \times 1}$$

$z \in \mathbb{R}^k$

U_{reduce}

Principal Component Analysis (PCA) algorithm summary

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

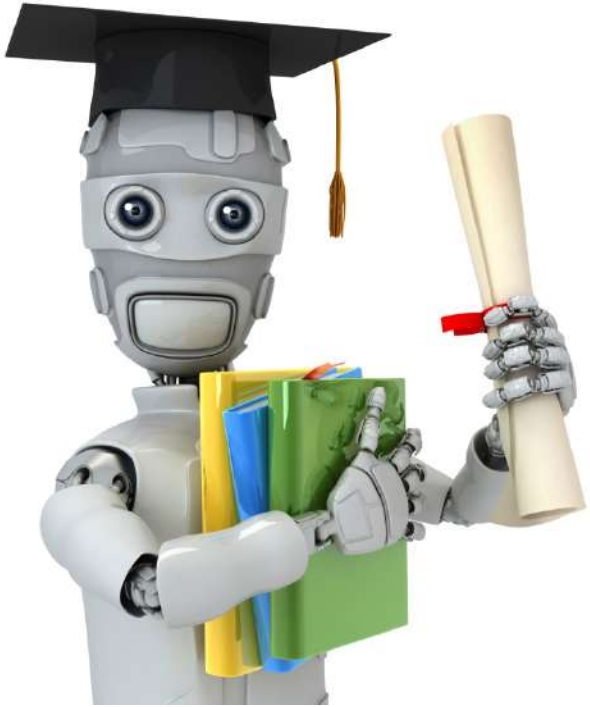
$$\texttt{Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$$

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

$$\boxed{\texttt{Sigma = (1/m) * X' * X;}}$$

```
[U,S,V] = svd(Sigma);
Ureduce = U(:,1:k);
z = Ureduce'*x;
```

$x \in \mathbb{R}^n$ $\quad$ ~~$x_0 = 1$~~

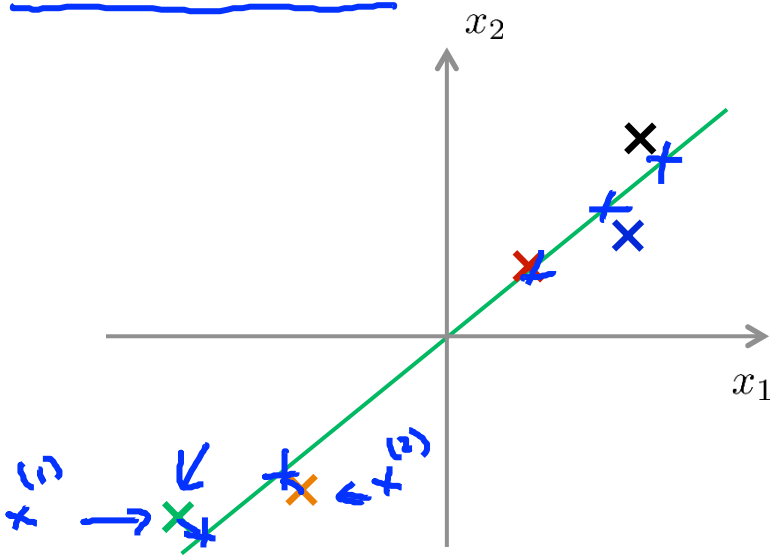


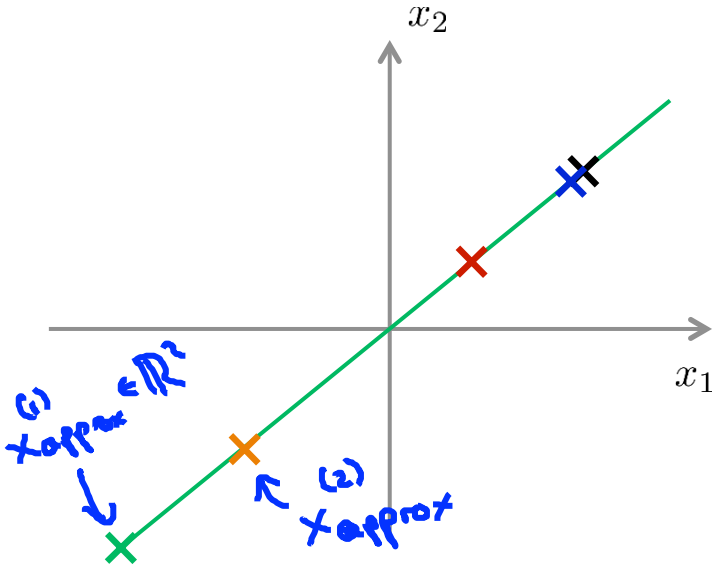
Machine Learning

Dimensionality Reduction

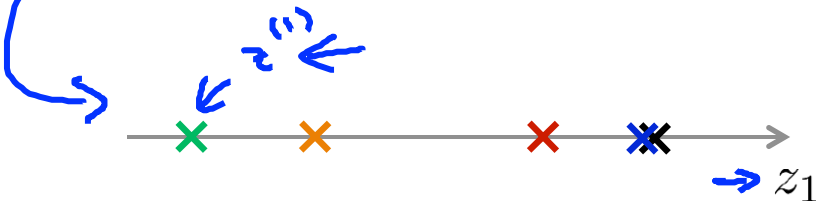
Reconstruction from compressed representation

Reconstruction from compressed representation





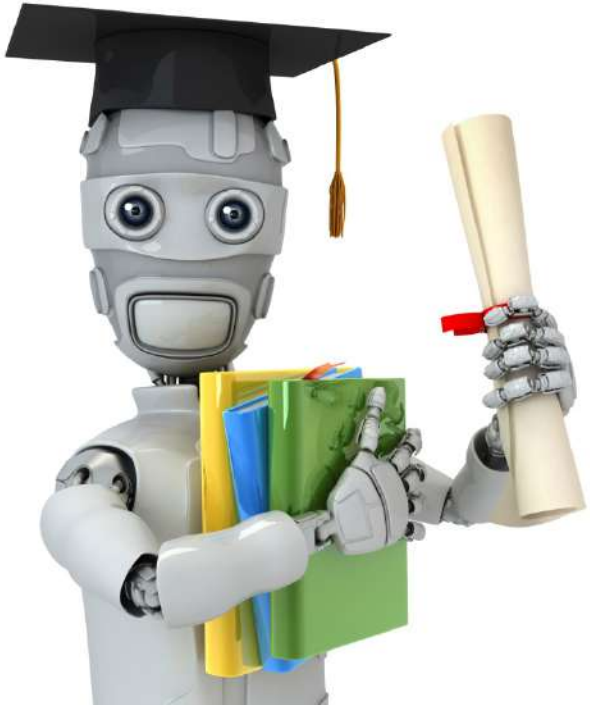
$$z = U_{reduce}^T x$$



$z \in \mathbb{R} \rightarrow x \in \mathbb{R}^2$

$$\underbrace{x_{approx}^{(1)}}_{\approx x \;\; \mathbb{R}^n} = \underbrace{\underbrace{U_{reduce}}_{n \times k} \cdot \underbrace{z^{(1)}}_{k \times 1}}_{n \times 1}$$

Andrew Ng



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

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

$\cancel{0.05}$ 0.10 $\cancel{5\%}$ (10%)

"~~99%~~ of variance is retained"

~~95%~~ 90%

Choosing k (number of principal components)

Algorithm:

Try PCA with $\cancel{k = 1}$ $\cancel{k=2}$ $\cancel{k=3}$ $k=4$ ...

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

Check if

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

$k = 17$

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

$$S = \begin{bmatrix} S_{11} & & & 0 \\ & S_{22} & & \\ & & S_{33} & \\ & & & \ddots \\ 0 & & & & S_{nn} \end{bmatrix}$$

For given k

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01$$

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

$k=3$

Choosing k (number of principal components)

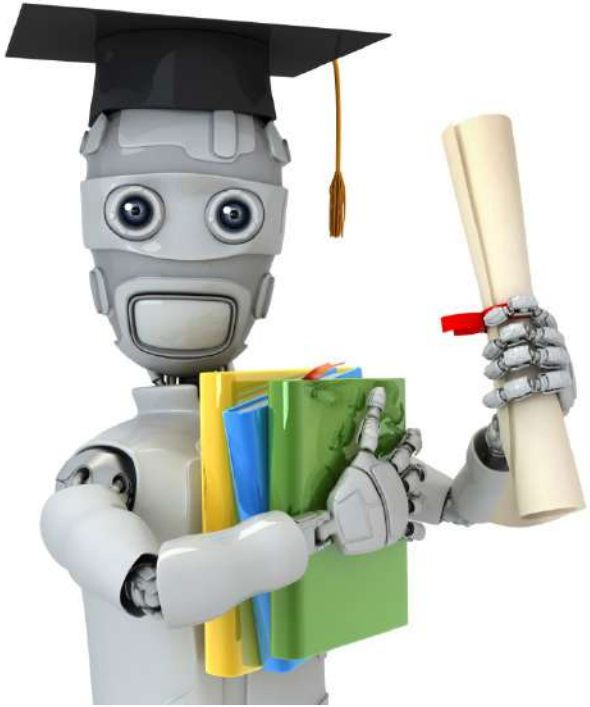
```
[U,S,V] = svd(Sigma)
```

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)



Dimensionality Reduction

Advice for applying PCA

Machine Learning

Supervised learning speedup

$x^{(i)} \in \mathbb{R}^{10,000}$

100

100

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

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:

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

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

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.

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

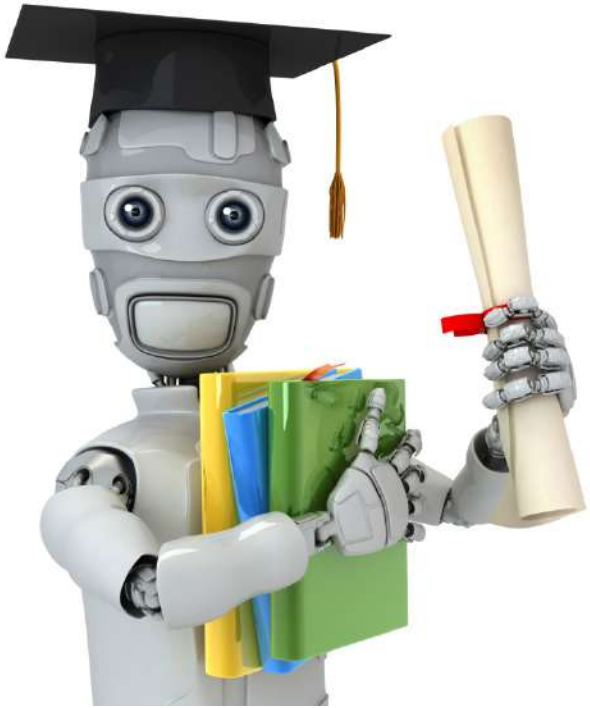
PCA is sometimes used where it shouldn't be

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 $z^{(i)}$.



Machine Learning

Recommender Systems

Problem formulation

Example: Predicting movie ratings

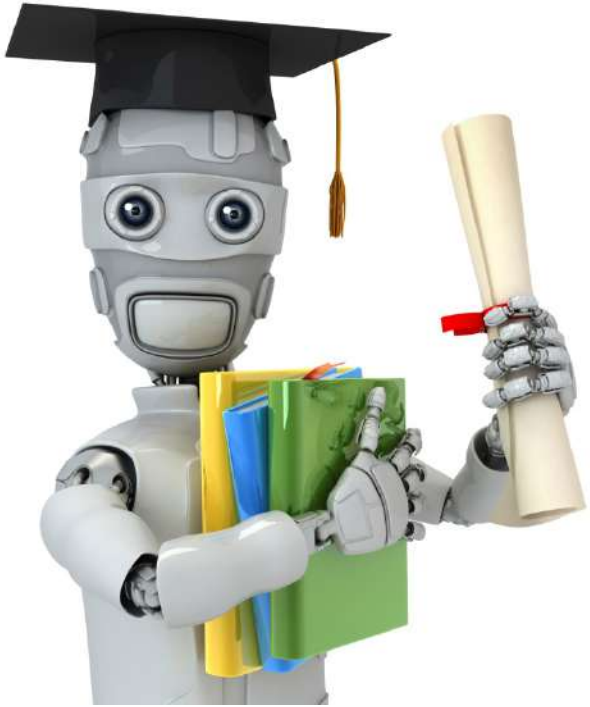
- User rates movies using ~~one~~ zero to five stars

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

$n_u = 4$ $n_m = 5$

- 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 $r(i,j) = 1$)

$0, \ldots, 5$



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 $\theta^{(j)}$:

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

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:

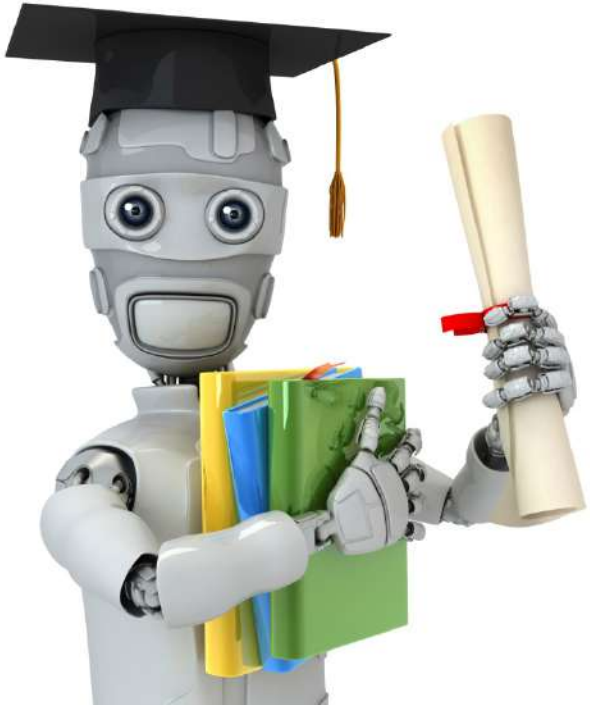
$$\underbrace{\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2}_{J(\theta^{(1)},\ldots,\theta^{(n_u)})}$$

Gradient descent update:

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

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \underbrace{\left(\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \underline{\lambda\theta_k^{(j)}}\right)}_{\frac{\partial}{\partial\theta_k^{(j)}} J(\theta^{(1)},\ldots,\theta^{(n_u)})} \quad \underline{(\text{for } k \neq 0)}$$

$\cancel{\frac{1}{m^{(j)}}}$



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	?	?	?

$$x^{(1)} = \begin{bmatrix} 1 \\ 1.0 \\ 0.0 \end{bmatrix}$$

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

$\theta^{(j)}$

$x^{(1)}$

$(\theta^{(1)})^T x^{(1)} \approx 5$

$(\theta^{(2)})^T x^{(1)} \approx 5$

$(\theta^{(3)})^T x^{(1)} \approx 0$

$(\theta^{(4)})^T x^{(1)} \approx 0$

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

$r^{(i,j)}$

$y^{(i,j)}$

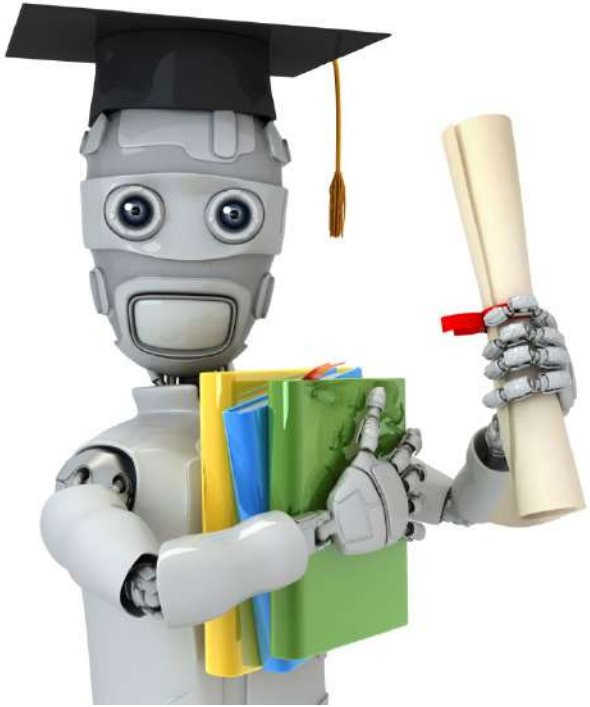
Given $x^{(1)}, \ldots, x^{(n_m)}$ (and movie ratings),

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

$(i,j) : r(i,j) = 1$

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

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

$x_0 = 1$

θ_0

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

$x_1 = 1$

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

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

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

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

Minimizing $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$ simultaneously:

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

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)} \\ \theta^{(1)},\ldots,\theta^{(n_u)}}} J(x^{(1)},\ldots,x^{(n_m)},\theta^{(1)},\ldots,\theta^{(n_u)})$$

$\theta \to x \to \theta \to x \to \ldots$

Andrew Ng

Collaborative filtering algorithm

$\cancel{x_0 = 1}$ $\quad x \in \mathbb{R}^n, \theta \in \mathbb{R}^n$

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 $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

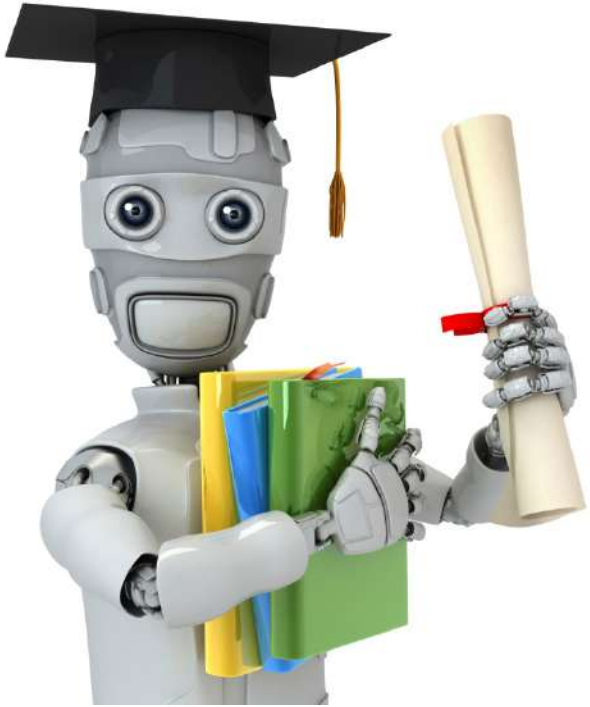
$\cancel{\theta_0}$ θ_1 $\ldots$ θ_n

$$x_k^{(i)} := x_k^{(i)} - \alpha \underbrace{\left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)}_{\frac{\partial}{\partial x_k^{(i)}} J(\ldots)}$$

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

3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

$(\theta^{(j)})^T (x^{(i)})$



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)

Predicted ratings:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

$$\begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \dots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \dots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \dots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}$$

$$X = \begin{bmatrix} -(x^{(1)})^T- \\ -(x^{(2)})^T- \\ \vdots \\ -(x^{(n_m)})^T- \end{bmatrix} \qquad \Theta = \begin{bmatrix} -(\theta^{(1)})^T- \\ -(\theta^{(2)})^T- \\ \vdots \\ -(\theta^{(n_u)})^T- \end{bmatrix}$$

→ Low rank matrix factorization

Finding related movies

For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$.

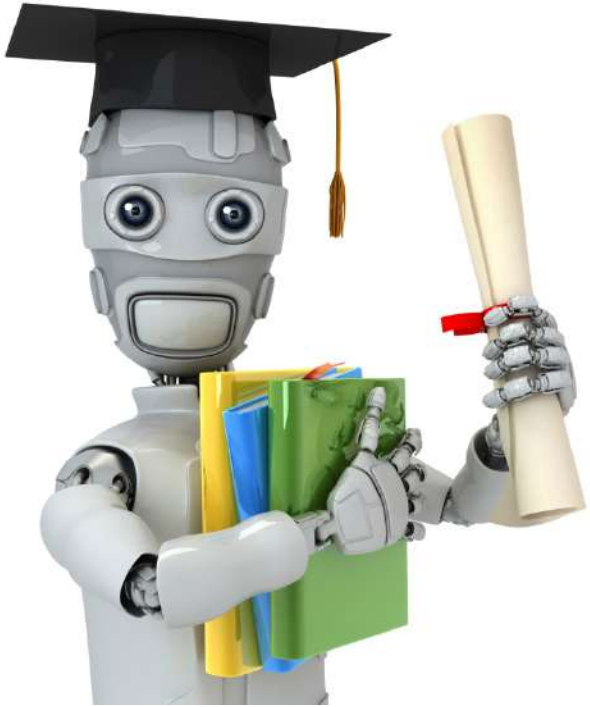
$\rightarrow$ x_1 = romance, x_2 = action, x_3 = comedy, x_4 = ...

How to find movies j related to movie i?

small $\|x^{(i)} - x^{(j)}\|$ $\rightarrow$ 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

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

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

$n = 2$ $\quad \theta^{(5)} \in \mathbb{R}^2$ $\quad \theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\quad \frac{\lambda}{2}\left[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2\right]$

$$(\theta^{(5)})^T x^{(i)} = 0$$

Andrew Ng

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \begin{matrix} 2.5 \\ 2.5 \\ 2 \\ \vdots \\ \\ \end{matrix}$$

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow \underline{Y} = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

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):

$$\underline{\Theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}}$$

$$\underbrace{(\Theta^{(5)})^T (x^{(i)})}_{\rightarrow 0} + \boxed{\mu_i}$$