

# Data Analytics (IT350)

## *Course Introduction*



Dr. Anand Kumar M  
Department of Information Technology  
National Institute of Technology-Karnataka (NITK)  
Surathkal, Mangalore.

# Data Analytics (IT350)

- **Category:**
- Programme Core (PC)
- **Credits (L-T-P):**
- (3-0-2) 4
- **Introduction to Data analysis:** statistical modelling, total information awareness, Bonferroni's Principle; **TYPES of DATA – Preprocessing-POSt PRoCESSING-Distributed File systems:** MapReduce and Spark; **Dimensionality Reduction:** PCA, SVD; **Finding Similar Items:** Distance Measures, Near Neighbour Search; Mining Data Streams; **Mining Social-Network Graphs:** graph centrality concepts, clustering, community detection, partitioning, overlapping community detection; **Applications of Large-scale Machine Learning,** **Machine Learning Basics**, Neural Network Models like Multi-Layer Perceptron (MLP), Recurrent Neural Networks (RNN), Convolutional Neural Network (CNN), Long Short Term Memory (LSTM).

# **Texts and References:**

- **Texts and References:**

- Ian Goodfellow, Y. Bengio and A. Courville, "Deep Learning", MIT Press, 2016.
- Jure Leskovec et al., "Mining of Massive Datasets" Cambridge University Press, 2014
- <http://www.mmmds.org>
- Recent Research Papers

# Evaluation Plan

- **Course Assessment Methods:**
- **(both continuous and semester-end assessment)**
- Mini Project=30%
- Lab Exercises/Exam=20%
- Mid Sem Exam=20%
- End Sem Exam=30%

# Assessment Type and COs

Assessment Type	Course Outcomes (COs)			
	CO1	CO2	CO3	CO4
Mid Sem Theory Exam	X	X	X	
Quizzes	X	X	X	
End Sem Theory Exam	X	X	X	
Lab Continuous Evaluations - Assignments		X	X	X
Mini / Minor Project			X	X

# Course Minor Project (30%)

- IEEE/ACM Reputed Journals
- Core Conferences
- Implementation
  - Title/Topic/Team Proposal
  - Midsem Eval (10)
  - End sem Eval (15)
- Plagiarism free Report
- *Conf/Journal Publication (Bonus Marks)*

# **IMPORTANT NOTE**

- 1. Team Formation – 5<sup>th</sup> Jan 2022**
- 2. Course Mini / Minor Project Proposal (Title Confirmation) - Jan 12th, 2022**
- 3. Mid Sem Project Progress Presentation with Report- Feb 9th, 2022**
- 4. Final Project Presentation with Report and Demo - March 31st, 2022**

# Why Data?

# DATA

bases  
Structures  
Mining  
Science  
Analytics



Data Analytics (IT350) Dr. Anand Kumar M  
(NITK)

# *Quantitative VS Qualitative*

**Data** is a set of variables, which can be quantitative or qualitative.

## *Quantitative Data*



Data can be measured



Data types include numbers and statistics



Data answers to the following questions: **how much?** **how many?** and **how often?**



## *Qualitative Data*



Data cannot be measured



Data types include **text**, **pictures**, **video** and **audio**



Data answers to the following questions: **how?** and **why?**



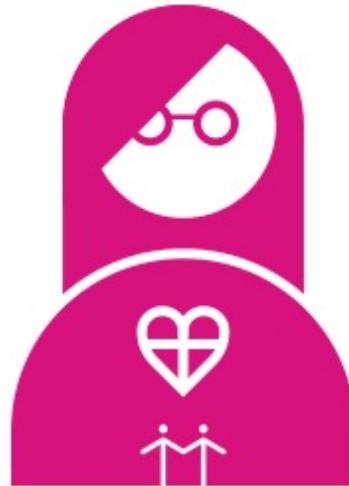
Colors Palette

# Types of Data About Us



## Personal

Name  
Address  
Email  
Telephone number  
IP address  
MAC address  
Online identifiers (cookies)  
Location data



## Sensitive

Gender  
Race  
Religion  
Political memberships  
Genetics  
Biometrics  
Health  
Sexual orientation  
Criminal record



## Behavioural

Browsing history  
Search history  
Purchase history  
Preferences  
Relationships  
Likes  
Dislikes  
Shares



## Societal

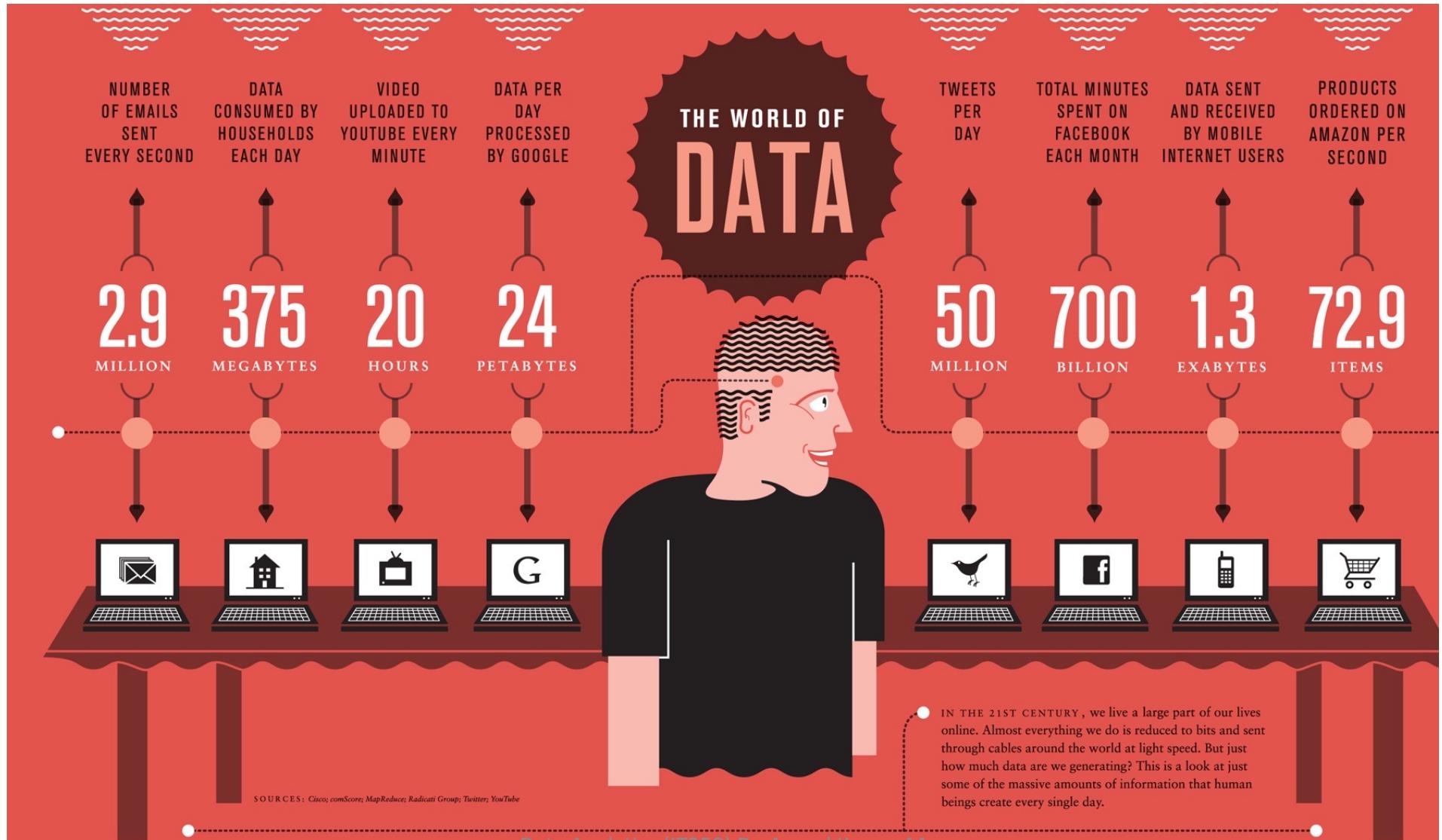
Census data  
Demographics  
Travel patterns  
Crime statistics  
Clinical trial results  
School performance indicators  
A&E waiting times

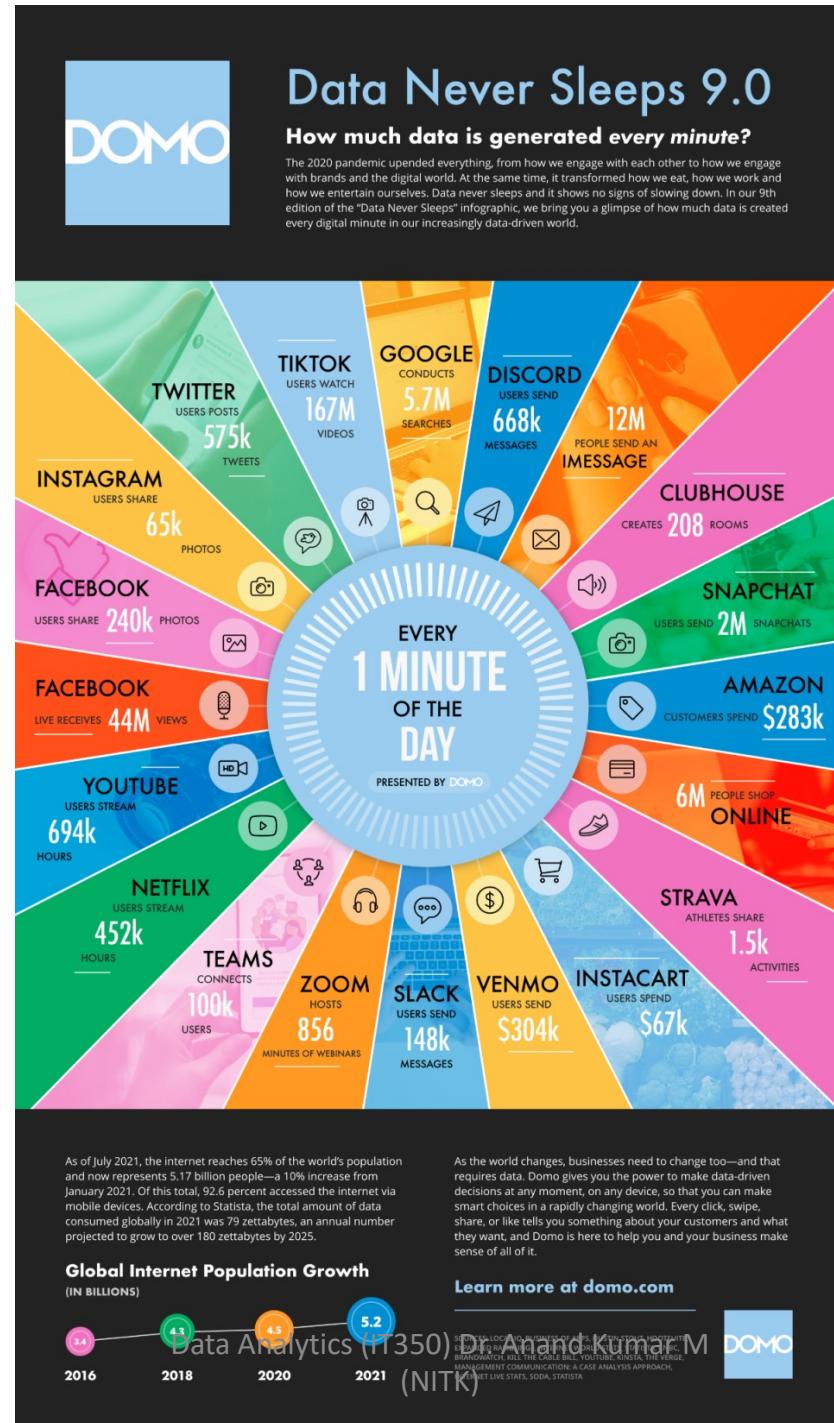


As people

{NEXT}

As society





## Data Analyst

## Data Scientist

4

Less pay, less experience

3

Static modelling

2

Basic programming

1

Historical data analysis

More pay, more experience

Predictive modelling

Advanced programming

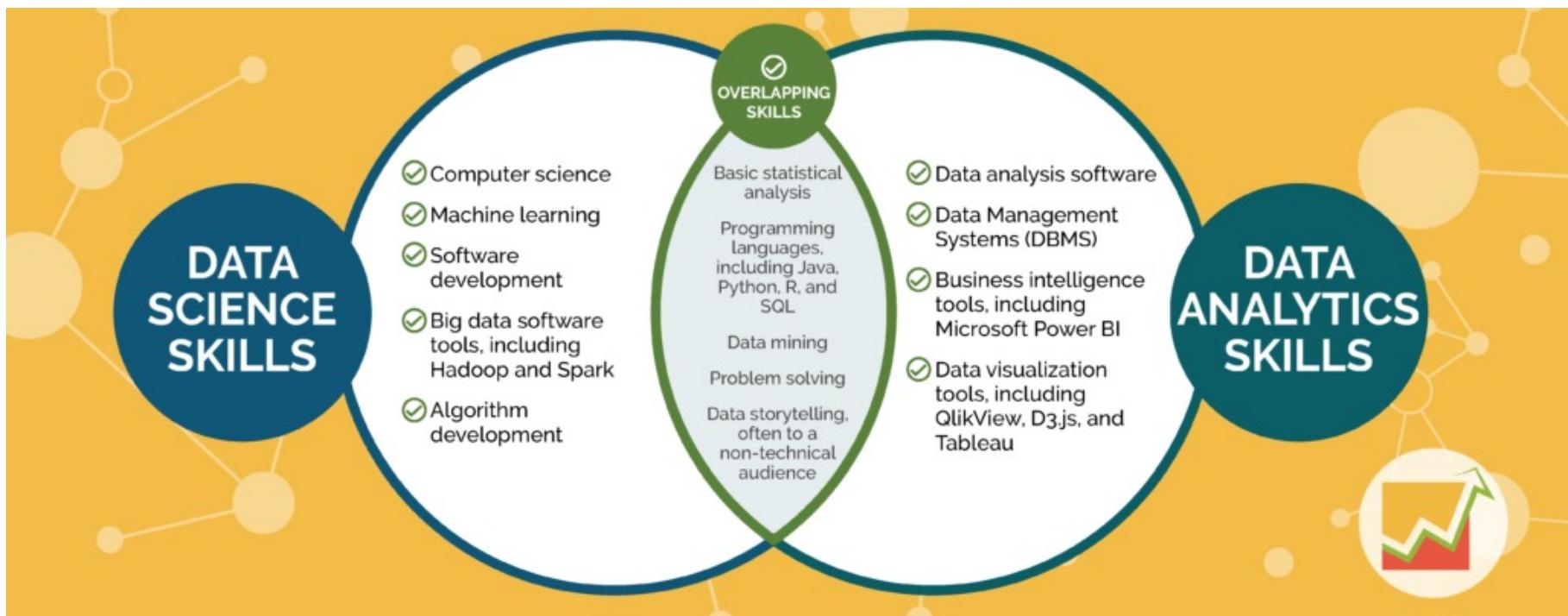
Future predictions



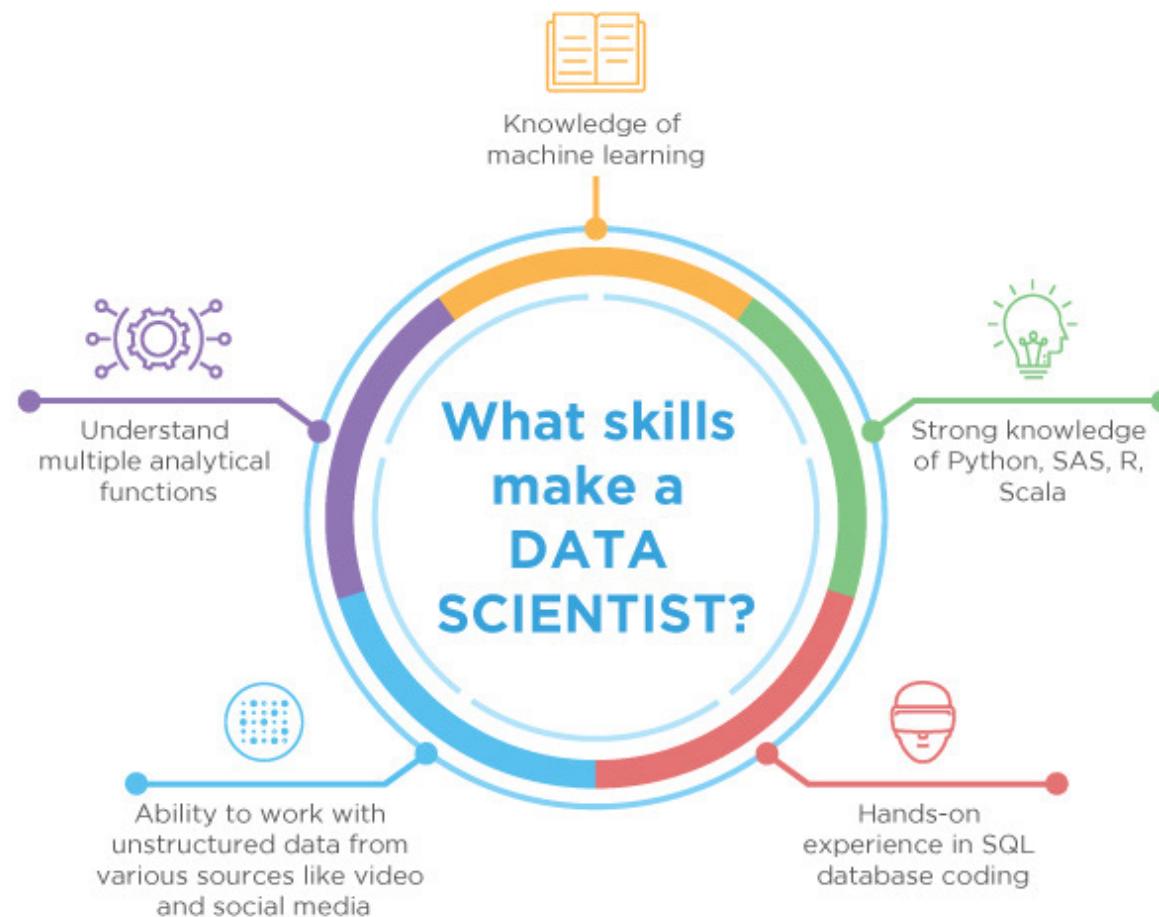


# *Data Science vs Data Analytics*

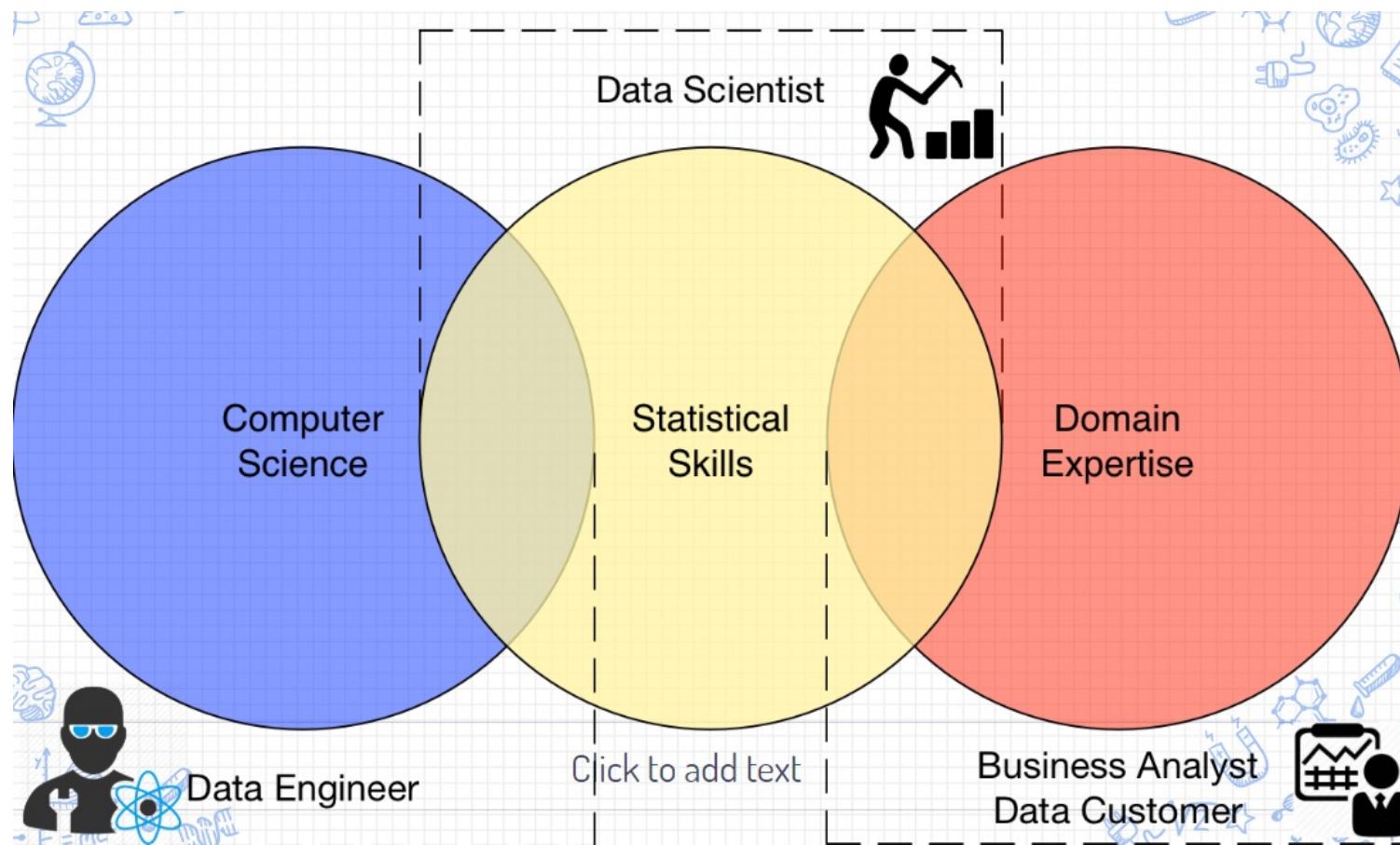
	Data Science	Data Analytics
SKILLSET	<ul style="list-style-type: none"><li>• Data Modelling</li><li>• Predictive Analytics</li><li>• Advanced Statistics</li><li>• Engineering/Programming</li></ul>	<ul style="list-style-type: none"><li>• BI Tools</li><li>• Intermediate Statistics</li><li>• Solid Programming Skills</li><li>• Regular Expression (SQL)</li></ul>
SCOPE	Macro	Micro
EXPLORATION	<ul style="list-style-type: none"><li>• Search Engine Exploration</li><li>• Machine Learning</li><li>• Artificial Intelligence</li><li>• Big data - Often Unstructured</li></ul>	<ul style="list-style-type: none"><li>• Data Visualization Techniques</li><li>• Designing Principles</li><li>• Big Data - Mostly Structured</li></ul>
GOALS	Discover New Questions to Drive Innovation	Use Existing Information to Uncover Actionable Data

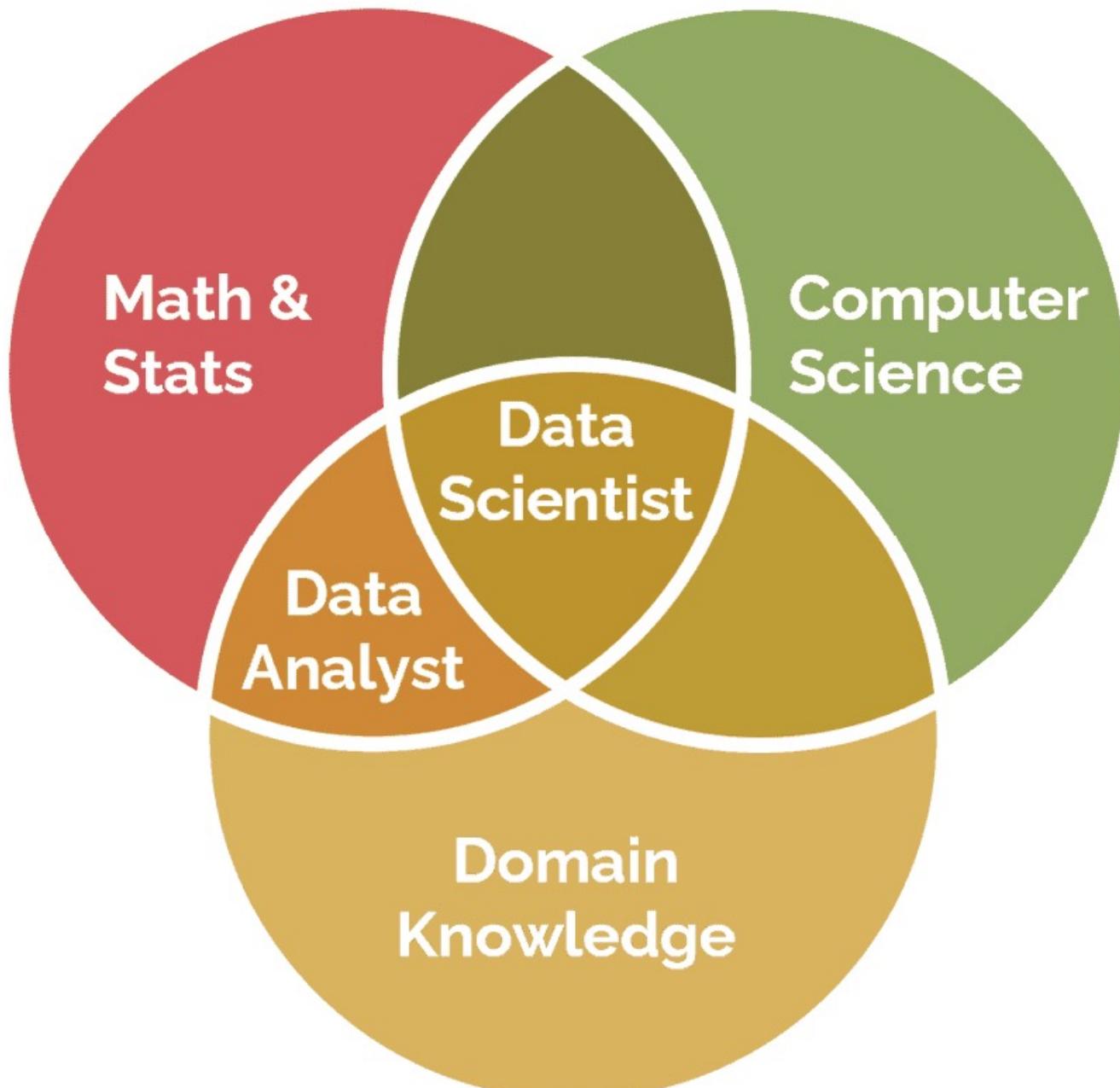


# DATA SCIENTIST



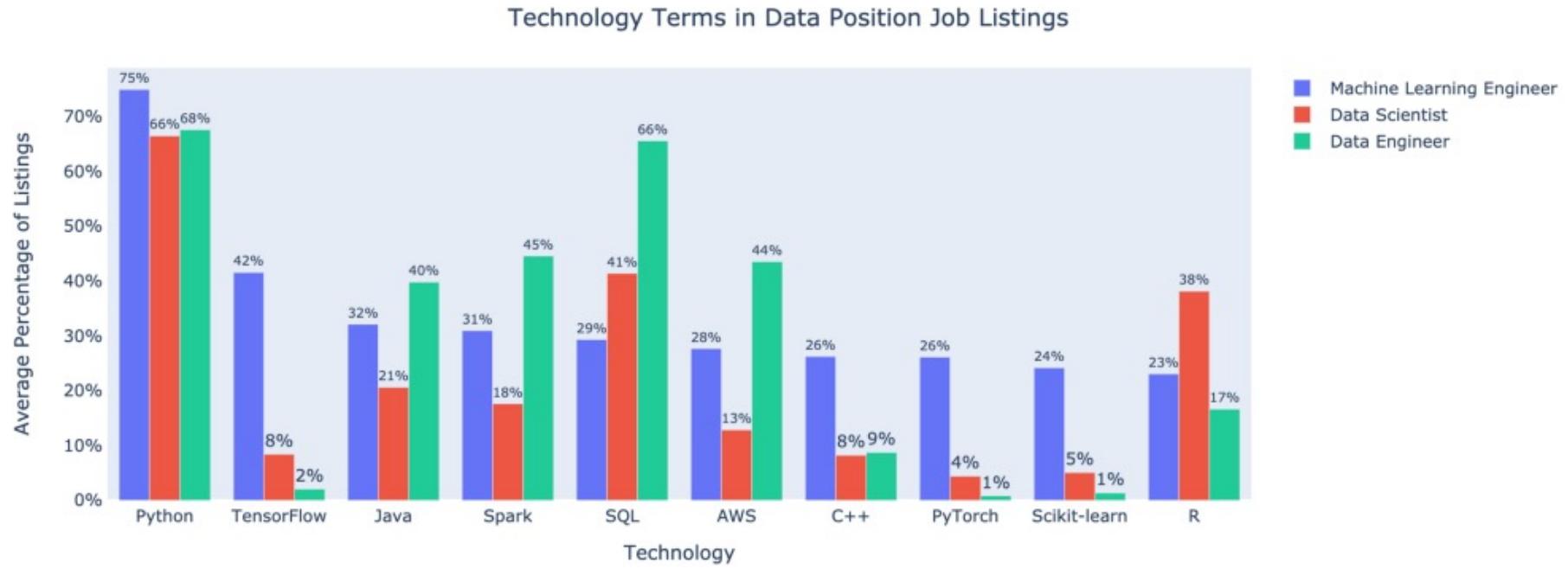
# DATA SCIENTIST





## 10 Most In-Demand Data Science Skills in 2021





<https://towardsdatascience.com/the-most-in-demand-tech-skills-for-machine-learning-engineers-7109751997d1>

Data Analytics (IT350) Dr. Anand Kumar M  
(NITK)

## DATA SCIENTIST ROLES AND SALARIES IN DEMAND

Roles in demand	2-5 years experience salary	5-8 years experience salary scale	8-12 years experience salary scale
Data Scientist (Technical, Functional, Techno-Functional)			
Data Analytics (programming knowledge of R, Python)			
Machine Learning			
Data Analyst (who can do Statistical Programming)			
Data Scientist (who can work on tools like Matplotlib, Google's Looker, Tableau for visualisation, business intelligence )	₹ 8-30 lakh per annum (varies depending on experience and tier-1,2 and 3 colleges)	₹ 16-50 lakh per annum (varies depending on experience and tier-1,2 and 3 colleges)	₹ 32-75 lakh per annum (varies depending on experience and tier-1,2 and 3 colleges)
Data Scientist + Engineering, with knowledge of tools like Spark, Hadoop and cloud platforms such as AWS for data processing			
Data Scientist ( tools Scikit-learn, Pandas, NumPy, SciPy for data analytics and mining)			
Data Scientist (machine learning applications like TensorFlow, PyTorch, Keras)			
Data Scientist (advanced and deep learning tools like Caffe, Alteryx)			

Source: Xpheno, a specialised staffing firm  
 Data Analytics (11550) Dr. Anand Kumar M  
 (NITK)



# WHERE ARE THE JOBS AND WHO IS HIRING

## Locations

Bangalore

Gurgaon

Hyderabad

Mumbai

Pune

Chennai

Kolkata

Noida

Delhi

## Majorly Hiring

R&D / Labs

Products

Startups/eCommerce

Digital Wallet/Platforms

Banking/Financial Services

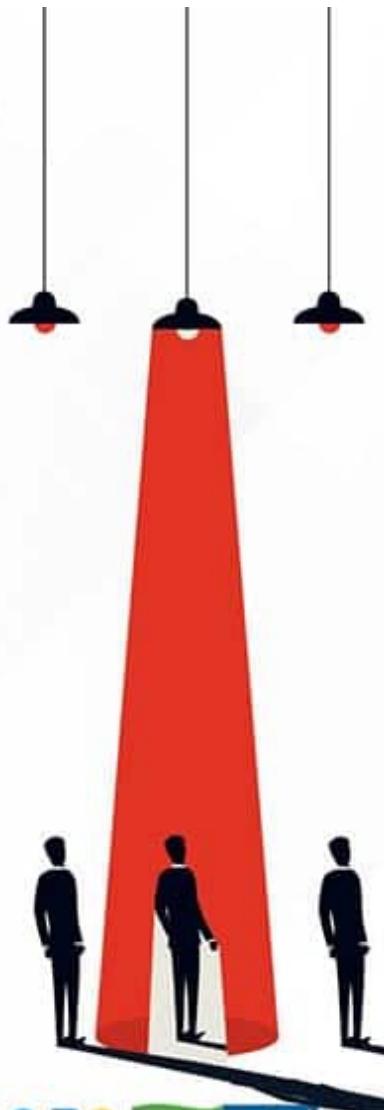
Global in-house centres

Payment/Gateway

Big 4 Consulting Firms

Global Consulting Firms

Software Services



Source: Xpheno, a specialised staffing firm





## DATA SCIENTISTS

A.K.A. STATISTICIANS,  
DATA MANAGERS

### SKILLS

Mathematics,  
Programming  
Communication



### TOOLS

SQL, Python, R



## DATA ENGINEERS

A.K.A. DATA ARCHITECTS,  
DATABASE ADMINISTRATORS

### SKILLS

Programming  
Mathematics  
Big Data



### TOOLS

Hadoop, NoSQL, Python



## DATA ANALYSTS

A.K.A. BUSINESS ANALYSTS

### SKILLS

Statistics  
Communication  
Business Knowledge



### TOOLS

Excel, Tableau, SQL



# How It All Fits Together

## High dim. data

Locality  
sensitive  
hashing

Clustering

Dimensional  
ity  
reduction

## Graph data

PageRank,  
SimRank

Community  
Detection

Spam  
Detection

## Infinite data

Filtering  
data  
streams

Web  
advertising

Queries on  
streams

## Machine learning

SVM

Decision  
Trees

Perceptron,  
kNN

## Apps

Recommen  
der systems

Association  
Rules

Duplicate  
document  
detection



# What is Data Mining?

- Given lots of data
- Discover patterns and models that are:
  - Valid: hold on new data with some certainty
  - Useful: should be possible to act on the item
  - Unexpected: non-obvious to the system
  - Understandable: humans should be able to interpret the pattern

# Data Mining Tasks

- **Descriptive methods**
  - Find human-interpretable patterns that describe the data
    - **Example:** Clustering
- **Predictive methods**
  - Use some variables to predict unknown or future values of other variables
    - **Example:** Recommender systems

# What is Data Mining?



- Data mining is the use of **efficient** techniques for the analysis of **very large** collections of data and the extraction of **useful** and possibly **unexpected** patterns in data.
- “Data mining is the discovery of **models** for data” (Rajaraman, Ullman)
  - We can have the following types of models
    - Models that **explain** the data (e.g., a single function)
    - Models that **predict** the future data instances.
    - Models that **summarize** the data
    - Models that **extract** the most prominent **features** of the data.

# Statistical Modeling

- Statisticians view data mining as the construction of a **statistical model**, that is, an underlying distribution from which the visible data is drawn.
- Suppose our data is a **set of numbers**. This data is much simpler than data that would be data-mined, but it will serve as an example.
- A statistician might **decide that the data comes from a Gaussian distribution** and use a formula to compute the most likely parameters of this Gaussian.
- The **mean and standard deviation** of this Gaussian distribution completely characterize the distribution and would become the model of the data.

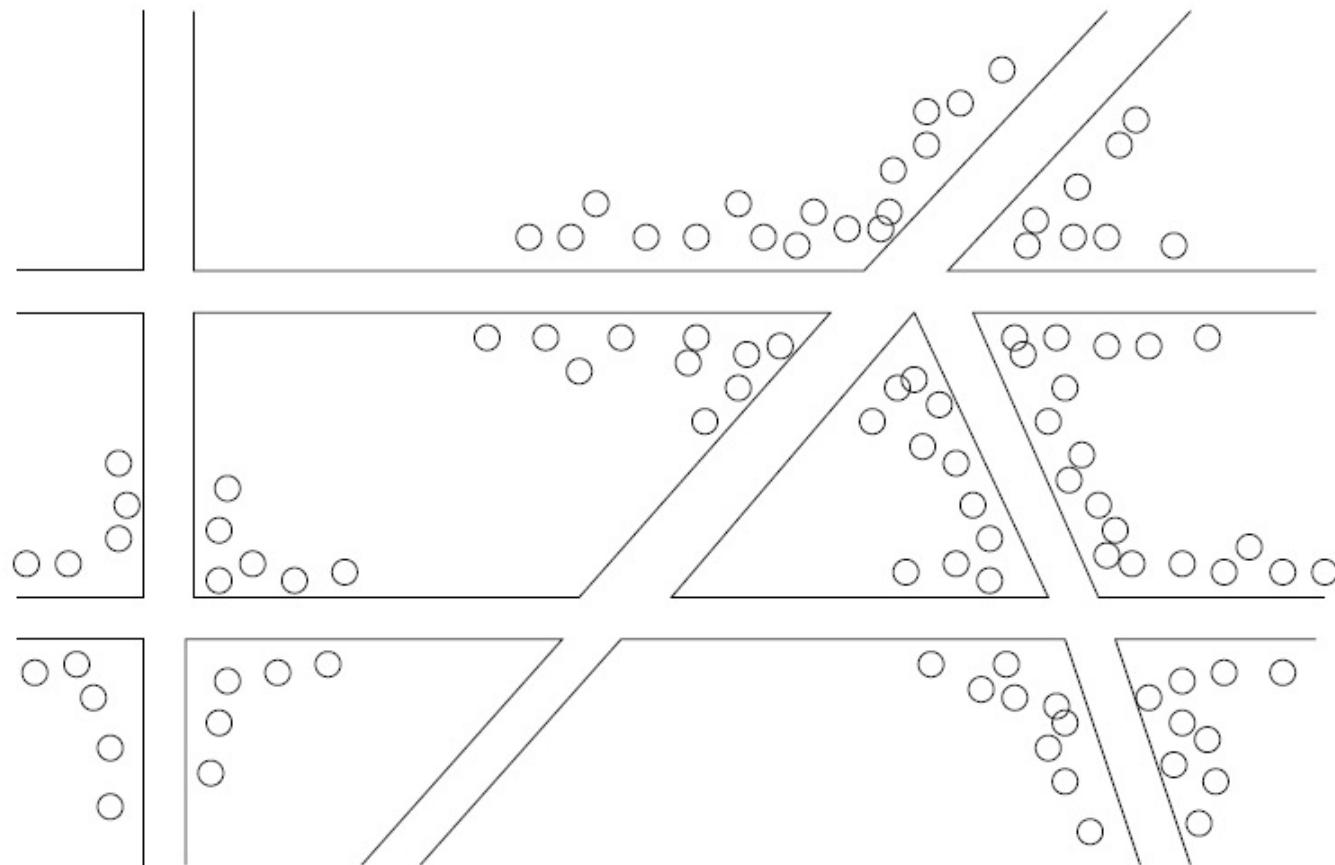
# Machine Learning

- Machine-learning practitioners use the data as a **training set**, to train an algorithm.
- The typical case where **machine learning is a good approach is when we have little idea of what we are looking for in the data.**
- Machine learning **has not proved successful in situations** where we can describe the goals of the mining more directly.
- A Company used machine learning to locate people's resumes on the Web.

# Summarization

- One of the most interesting forms of summarization is the **PageRank** idea, which made Google successful.
- *entire complex structure of the Web is summarized by a single number for each page.*
- **Clustering**-data is viewed as points in a multidimensional space. Points that are “close” in this space are assigned to the same cluster.

# Cholera Cases on a Map of London



# Feature Extraction

- Frequent Itemsets-We look for *small sets of items that appear together in many baskets*, and these “frequent itemsets” are the characterization of the data that we seek
- **Similar Items**- Amazon can look for “similar” customers and recommend something many of these customers have bought. This process is called “**collaborative filtering**.”

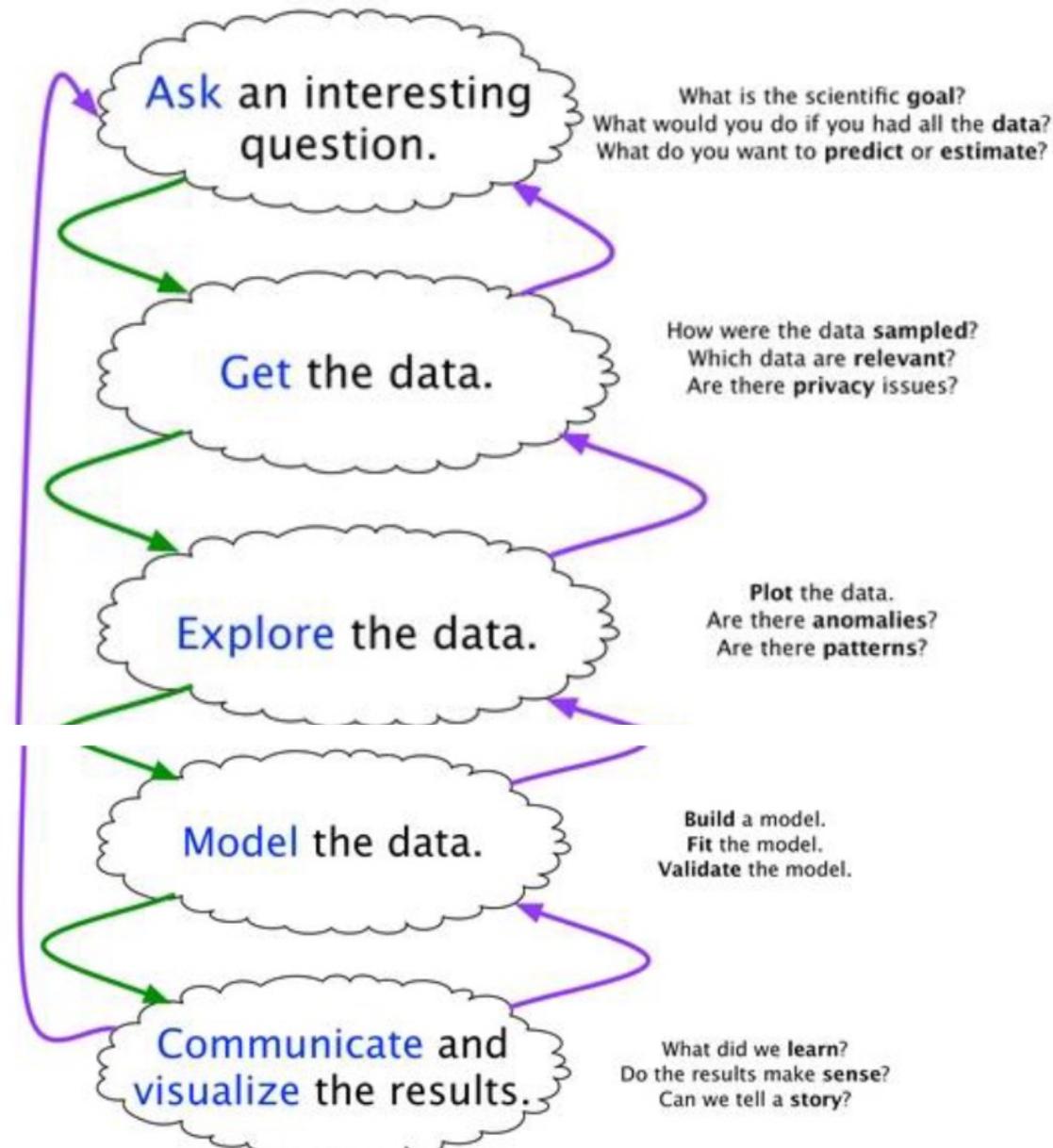
# The data analysis pipeline

- Mining is not the only step in the analysis process



- **Preprocessing:** real data is noisy, incomplete and inconsistent. **Data cleaning** is required to make sense of the data
  - Techniques: Sampling, Dimensionality Reduction, Feature selection.
  - A dirty work, but it is often the most important step for the analysis.
- **Post-Processing:** Make the data actionable and useful to the user
  - Statistical analysis of importance
  - Visualization.
- Pre- and Post-processing are often data mining tasks as well

# THE DATA SCIENCE WORKFLOW



# Meaningfulness of Analytic Answers

- A risk with “Data mining” is that an analyst can “discover” patterns that are meaningless
- Statisticians call it **Bonferroni’s principle**:
  - Roughly, if you look in more places for interesting patterns than your amount of data will support, you are bound to find crap



# Bonferroni's principle

- Certain amount of data - events of a certain type within that data
- expect events of this type to occur, even if the data is completely random-
- the number of occurrences of these events will grow as the size of the data grows
- These occurrences are “**bogus**,” in the sense that they have **no cause other than that random data will always have some number of unusual features**.

# IT350 DATA ANALYTICS

## LECTURE 1

---

Introduction – Data- Types -Examples

# What is data mining?

- After years of data mining there is still no unique answer to this question.
- A tentative definition:



Data mining is the use of **efficient** techniques for the analysis of **very large** collections of data and the extraction of **useful** and possibly **unexpected** patterns in data.



# Why do we need data mining?

- Really, really huge amounts of raw data!!
  - In the digital age, TB of data is generated by the second
    - Mobile devices, digital photographs, web documents.
    - Facebook updates, Tweets, Blogs, User-generated content
    - Transactions, sensor data, surveillance data
    - Queries, clicks, browsing
  - Cheap storage has made possible to maintain this data
- Need to analyze the raw data to extract knowledge

# Why do we need data mining?

- “The data is the computer”
  - Large amounts of **data** can be more **powerful** than complex **algorithms** and models
    - Google has solved many Natural Language Processing problems, simply by looking at the data
    - Example: misspellings, synonyms
  - Data is power!
    - Today, the collected data is one of the biggest **assets** of an online company
      - Query logs of Google
      - The friendship and updates of Facebook
      - Tweets and follows of Twitter
      - Amazon transactions
  - We need a way to harness the **collective intelligence**

# The data is also very **complex**

- Multiple **types** of data: tables, text, time series, images, graphs, etc
- **Spatial** and **temporal** aspects
- **Interconnected** data of different types:
  - From the mobile phone we can collect, location of the user, friendship information, check-ins to venues, opinions through twitter, status updates in FB, images though cameras, queries to search engines

# Example: transaction data

- Billions of real-life customers:
  - WALMART: 20M transactions per day
  - AT&T 300 M calls per day
  - Credit card companies: billions of transactions per day.
- The point cards allow companies to collect information about specific users

# Example: document data

- Web as a document repository: estimated 50 billions of web pages
- Wikipedia: 4.5 million articles (and counting)
- Online news portals: steady stream of 100's of new articles every day
- Twitter: ~500 million tweets every day

# Example: network data

- Web: 50 billion pages linked via hyperlinks
- Facebook: 1.23 billion users
- Twitter: 270 million users
- Blogs: 250 million blogs worldwide, presidential candidates run blogs

# Example: genomic sequences

- <http://www.1000genomes.org/page.php>
- Full sequence of 1000 individuals
- 3 billion nucleotides per person → 3 trillion nucleotides
- Lots more data in fact: medical history of the persons, gene expression data

# Medical data

- Wearable devices can measure your heart rate, blood sugar, blood pressure, and other signals about your health. Medical records are becoming available to individuals
  - Wearable computing
- Brain imaging
  - Images that monitor the activity in different areas of the brain under different stimuli
    - TB of data that need to be analyzed.
- Gene and Protein interaction networks
  - It is rare that a single gene regulates deterministically the expression of a condition.
  - There are complex networks and probabilistic models that govern the protein expression.

# Example: environmental data

- Climate data (just an example)

<http://www.ncdc.gov/oa/climate/ghcn-monthly/index.php>

- “a database of temperature, precipitation and pressure records managed by the National Climatic Data Center, Arizona State University and the Carbon Dioxide Information Analysis Center”
- “6000 temperature stations, 7500 precipitation stations, 2000 pressure stations”
  - Spatiotemporal data

# Behavioral data

- Mobile phones today record a large amount of information about the user behavior
  - GPS records position
  - Camera produces images
  - Communication via phone and SMS
  - Text via facebook updates
  - Association with entities via check-ins
- Amazon collects all the items that you browsed, placed into your basket, read reviews about, purchased.
- Google and Bing record all your browsing activity via toolbar plugins. They also record the queries you asked, the pages you saw and the clicks you did.
- Data collected for millions of users on a daily basis

# So, what is Data?

- Collection of data **objects** and their **attributes**
- An attribute is a property or characteristic of an object
  - Examples: eye color of a person, temperature, etc.
  - Attribute is also known as **variable**, **field**, **characteristic**, or **feature**
- A collection of attributes describe an object
  - Object is also known as **record**, **point**, **case**, **sample**, **entity**, or **instance**

Objects

Attributes

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

**Size:** Number of objects

**Dimensionality:** Number of attributes

**Sparsity:** Number of populated object-attribute pairs

# Types of Attributes

- There are different types of attributes
  - Categorical
    - Examples: eye color, zip codes, words, rankings (e.g, good, fair, bad), height in {tall, medium, short}
    - Nominal (no order or comparison) vs Ordinal (order but not comparable)
  - Numeric
    - Examples: dates, temperature, time, length, value, count.
    - Discrete (counts) vs Continuous (temperature)
    - Special case: Binary attributes (yes/no, exists/not exists)

# Numeric Record Data

- If data objects have the same **fixed set** of **numeric attributes**, then the data objects can be thought of as **points** in a multi-dimensional space, where each **dimension** represents a distinct attribute
- Such data set can be represented by an **n-by-d data matrix**, where there are **n** rows, one for each object, and **d** columns, one for each attribute

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

# Categorical Data

- Data that consists of a collection of records, each of which consists of a **fixed set of categorical attributes**

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	High	No
2	No	Married	Medium	No
3	No	Single	Low	No
4	Yes	Married	High	No
5	No	Divorced	Medium	Yes
6	No	Married	Low	No
7	Yes	Divorced	High	No
8	No	Single	Medium	Yes
9	No	Married	Medium	No
10	No	Single	Medium	Yes

# Document Data

- Each document becomes a 'term' vector,
  - each term is a component (attribute) of the vector,
  - the value of each component is the number of times the corresponding term occurs in the document.
  - **Bag-of-words** representation – no ordering

	team	coach	pla	ball	score	game	n	wi	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0	
Document 3	0	1	0	0	1	2	2	0	3	0	

# Transaction Data

- Each record (transaction) is a **set of items**.

<i>TID</i>	<i>Items</i>
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

- A set of items can also be represented as a **binary vector**, where each attribute is an item.
- A document can also be represented as a **set of words** (no counts)

**Sparsity:** average number of products bought by a customer

# Ordered Data

- Genomic **sequence** data

```
GGTTCCGCCTTCAGCCCCGCGCC  
CGCAGGGCCCGCCCCGCGGCCGTC  
GAGAAGGGCCC GCCCTGGCGGGCG  
GGGGGAGGC GGGGCCGCCGAGC  
CCAACCGAGTCCGACCAGGTGCC  
CCCTCTGCTCGGCCTAGACCTGA  
GCTCATTAGGC GGCAGCGGACAG  
GCCAAGTAGAACACGCGAAGCGC  
TGGGCTGCCTGCTGCGACCAGGG
```

- Data is a long **ordered** string

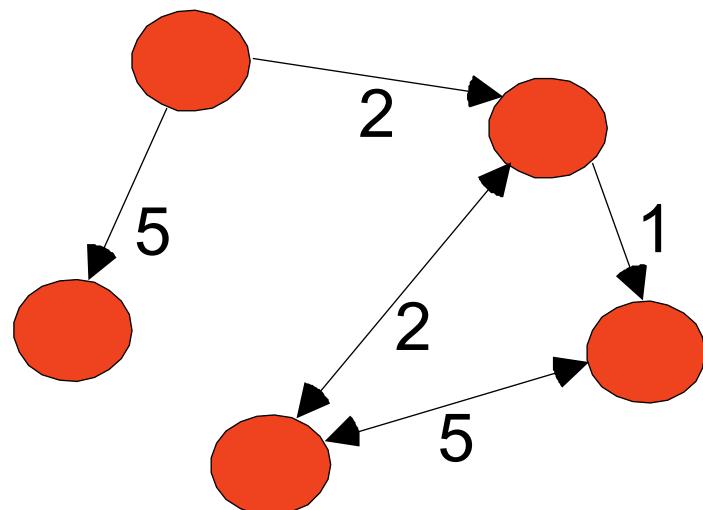
# Ordered Data

- Time series
  - Sequence of ordered (over “time”) numeric values.



# Graph Data

- Examples: Web graph and HTML Links
- Facebook graph of Friendships
- Twitter follow graph
- The connections between brain neurons



In this case the data consists of **pairs**:

Who links to whom

# Types of data

- **Numeric data**: Each object is a point in a multidimensional space
- **Categorical data**: Each object is a vector of categorical values
- **Set data**: Each object is a set of values (with or without counts)
  - Sets can also be represented as binary vectors, or vectors of counts
- **Ordered sequences**: Each object is an ordered sequence of values.
- **Graph data**

# What can you do with the data?

- Suppose that you are the owner of a supermarket and you have collected billions of **market basket** data. What information would you extract from it and how would you use it?

<i>TID</i>	<i>Items</i>
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Product placement

Catalog creation

Recommendations

- What if this was an online store?

# What can you do with the data?

- Suppose you are a search engine and you have a **toolbar log** consisting of

- pages browsed,
- queries,
- pages clicked,
- ads clicked

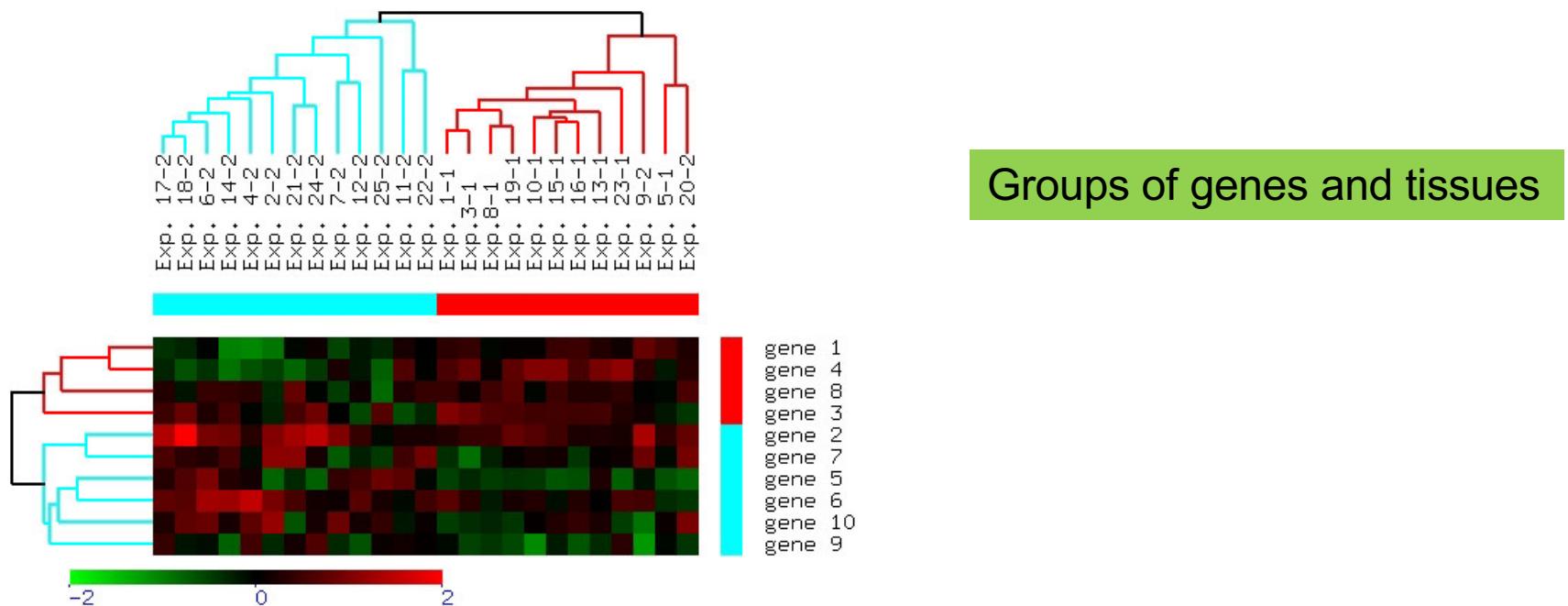
Ad click prediction

Query reformulations

each with a **user id** and a **timestamp**. What information would you like to get our of the data?

# What can you do with the data?

- Suppose you are biologist who has **microarray expression data**: thousands of genes, and their expression values over thousands of different settings (e.g. tissues). What information would you like to get out of your data?



# What can you do with the data?

- Suppose you are a stock broker and you observe the fluctuations of multiple stocks over time. What information would you like to get our of your data?



# What can you do with the data?

- You are the owner of a social network, and you have full access to the social graph, what kind of information do you want to get out of your graph?

- Who is the most important node in the graph?
- What is the shortest path between two nodes?
- How many friends two nodes have in common?
- How does information spread on the network?

# Why data mining?

- **Commercial** point of view
  - Data has become the key competitive advantage of companies
    - Examples: Facebook, Google, Amazon
  - Being able to extract useful information out of the data is key for exploiting them commercially.
- **Scientific** point of view
  - Scientists are at an unprecedented position where they can collect TB of information
    - Examples: Sensor data, astronomy data, social network data, gene data
  - We need the tools to analyze such data to get a better understanding of the world and advance science
- **Scale** (in data **size** and feature **dimension**)
  - Why not use traditional analytic methods?
  - Enormity of data, **curse of dimensionality**
  - The amount and the complexity of data does not allow for manual processing of the data. We need automated techniques.

# Big data

- The new trend in data mining...
  - An all-encompassing term to describe problems in science, industry, everyday life where there are huge amounts of data that need to be stored, maintained and analyzed to produce **value**.
- The overall idea:
  - Every activity generates data
    - Wearable computing, Internet of Things, Brain Imaging, Urban behavior
  - If we collect and understand this data we can improve life
    - E.g., Urban computing, Health informatics.

# Why data mining?

There is also [this](#) reason...

*"The success of companies like Google, Facebook, Amazon, and Netflix, not to mention Wall Street firms and industries from manufacturing and retail to healthcare, is increasingly driven by better tools for extracting meaning from very large quantities of data. 'Data Scientist' is now the hottest job title in Silicon Valley."* – Tim O'Reilly

## Data Scientist: The Sexiest Job of the 21st Century

by Thomas H. Davenport and D.J. Patil

Comments (87)



Artwork: **Tamar Cohen**, *Andrew J Buboltz*, 2011, silk screen on a page from a high school yearbook, 8.5" x 12"

### RELATED

[Executive Summary](#)

### ALSO AVAILABLE

- [Buy PDF](#)

# What is Data Mining again?

- “Data mining is the analysis of (often large) observational data sets to find **unsuspected relationships** and to **summarize** the data in novel ways that are both **understandable and useful** to the data analyst” (Hand, Mannila, Smyth)
- “Data mining is the discovery of **models** for data” (Rajaraman, Ullman)
  - We can have the following types of models
    - Models that **explain** the data (e.g., a single function)
    - Models that **predict** the future data instances.
    - Models that **summarize** the data
    - Models that **extract** the most prominent **features** of the data.

# What is data mining again?

- The **industry** point of view: The analysis of **huge amounts of data** for extracting useful and actionable information, which is then integrated into **production** systems in the form of new features of products
  - **Data Scientists** should be good at **data analysis, math, statistics**, but also be able to **code** with huge amounts of data and use the extracted information to **build** products.

# What can we do with data mining?

- Some examples:
  - Frequent itemsets and Association Rules extraction
  - Coverage
  - Clustering
  - Classification
  - Ranking
  - Exploratory analysis

# Frequent Itemsets and Association Rules

- Given a set of records each of which contain some number of items from a given collection:
  - Identify sets of items (**itemsets**) occurring frequently together
  - Produce **dependency rules** which will predict occurrence of an item based on occurrences of other items.

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Itemsets Discovered:

{Milk, Coke}

{Diaper, Milk}

Rules Discovered:

{Milk} --> {Coke}

{Diaper, Milk} --> {Beer}

# Frequent Itemsets: Applications

- Text mining: finding associated phrases in text
  - There are lots of documents that contain the phrases “association rules”, “data mining” and “efficient algorithm”
- Recommendations:
  - Users who buy this item often buy this item as well
  - Users who watched James Bond movies, also watched Jason Bourne movies.
  - Recommendations make use of item and user similarity

# Association Rule Discovery: Application

- Supermarket **shelf management**.
  - Goal: To identify items that are bought together by sufficiently many customers.
  - Approach: Process the point-of-sale data collected with barcode scanners to find dependencies among items.
  - A classic rule --
    - If a customer buys diaper and milk, then he is very likely to buy beer.
    - So, don't be surprised if you find six-packs stacked next to diapers!

# Clustering Definition

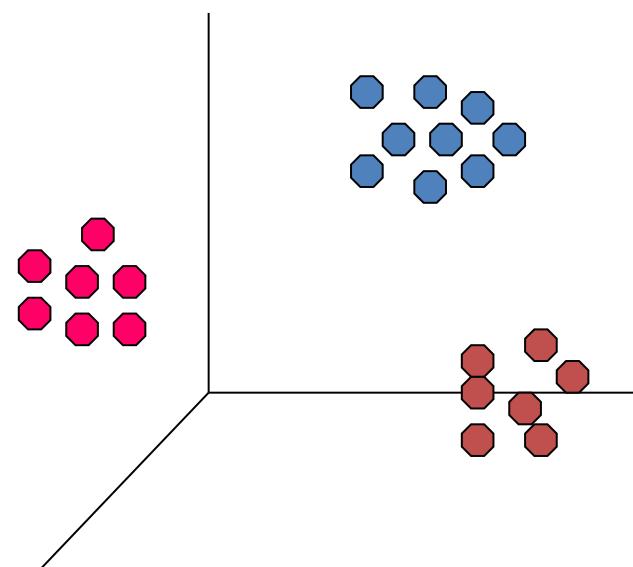
- Given a set of data points, each having a set of attributes, and a similarity measure among them, find **clusters** such that
  - Data points in one **cluster** are **more similar** to one another.
  - Data points in **separate clusters** are **less similar** to one another.
- Similarity Measures?**
  - Euclidean Distance if attributes are continuous.
  - Other Problem-specific Measures.

# Illustrating Clustering

Euclidean Distance Based Clustering in 3-D space.

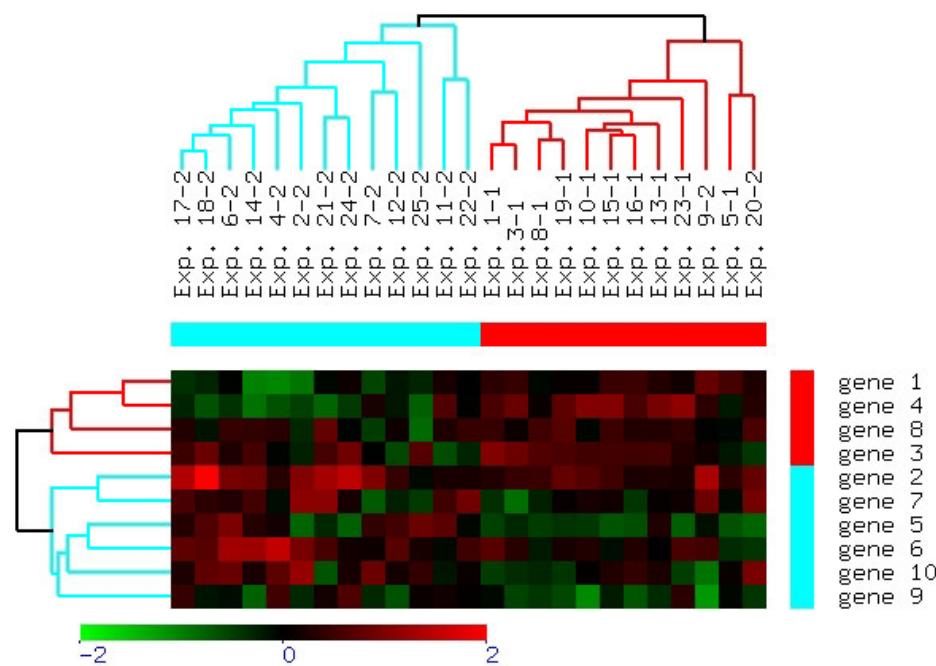
Intracluster distances  
are minimized

Intercluster distances  
are maximized



# Clustering: Application 1

- Bioinformatics applications:
  - Goal: Group genes and tissues together such that genes are coexpressed on the same tissues



# Clustering: Application 2

- Document Clustering:
  - Goal: To find groups of documents that are similar to each other based on the important terms appearing in them.
  - Approach: To identify frequently occurring terms in each document. Form a similarity measure based on the frequencies of different terms. Use it to cluster.
  - Gain: Information Retrieval can utilize the clusters to relate a new document or search term to clustered documents.

# Coverage

- Given a set of customers and items and the transaction relationship between the two, select a small set of items that “**covers**” all users.
  - For each user there is at least one item in the set that the user has bought.
- Application:
  - Create a catalog to send out that has at least one item of interest for every customer.

# Classification: Definition

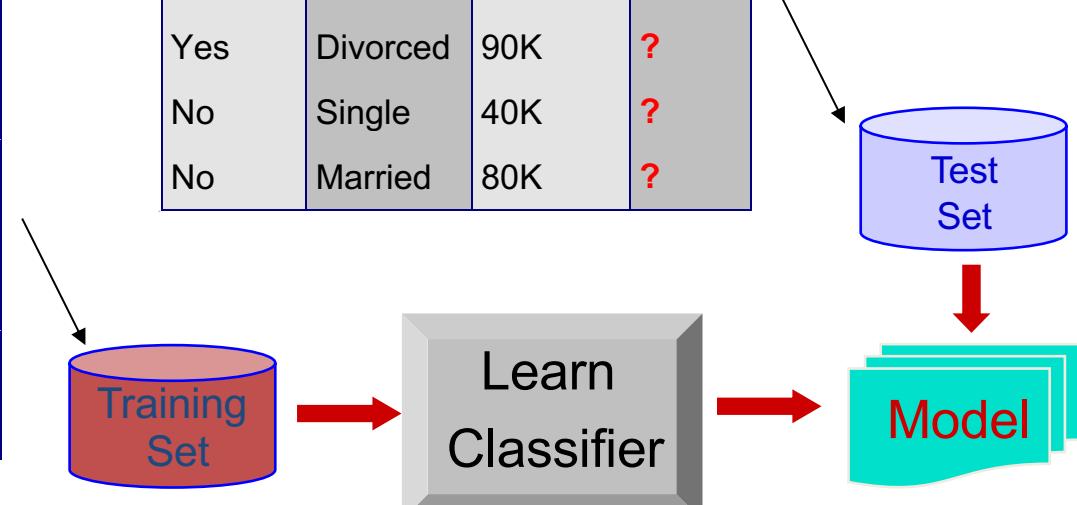
- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

# Classification Example

categorical  
categorical  
continuous  
class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?



# Classification: Application 1

- Ad Click Prediction
  - Goal: Predict if a user that visits a web page will click on a displayed ad. Use it to target users with high click probability.
  - Approach:
    - Collect data for users over a period of time and record who clicks and who does not. The {click, no click} information forms the **class attribute**.
    - Use the history of the user (web pages browsed, queries issued) as the features.
    - Learn a classifier model and test on new users.

# Classification: Application 2

- Fraud Detection
  - Goal: Predict fraudulent cases in credit card transactions.
  - Approach:
    - Use credit card transactions and the information on its account-holder as attributes.
    - When does a customer buy, what does he buy, how often he pays on time, etc
    - **Label** past transactions as fraud or fair transactions. This forms the class attribute.
    - Learn a model for the class of the transactions.
    - Use this model to detect fraud by observing credit card transactions on an account.

# Network data analysis

- **Link Analysis Ranking:** Given a collection of web pages that are linked to each other, rank the pages according to importance (**authoritativeness**) in the graph
  - Intuition: A page gains authority if it is linked to by another page.
- Application: When retrieving pages, the authoritativeness is factored in the ranking.
  - This is the idea that made **Google** a success around 2000

# Network data analysis

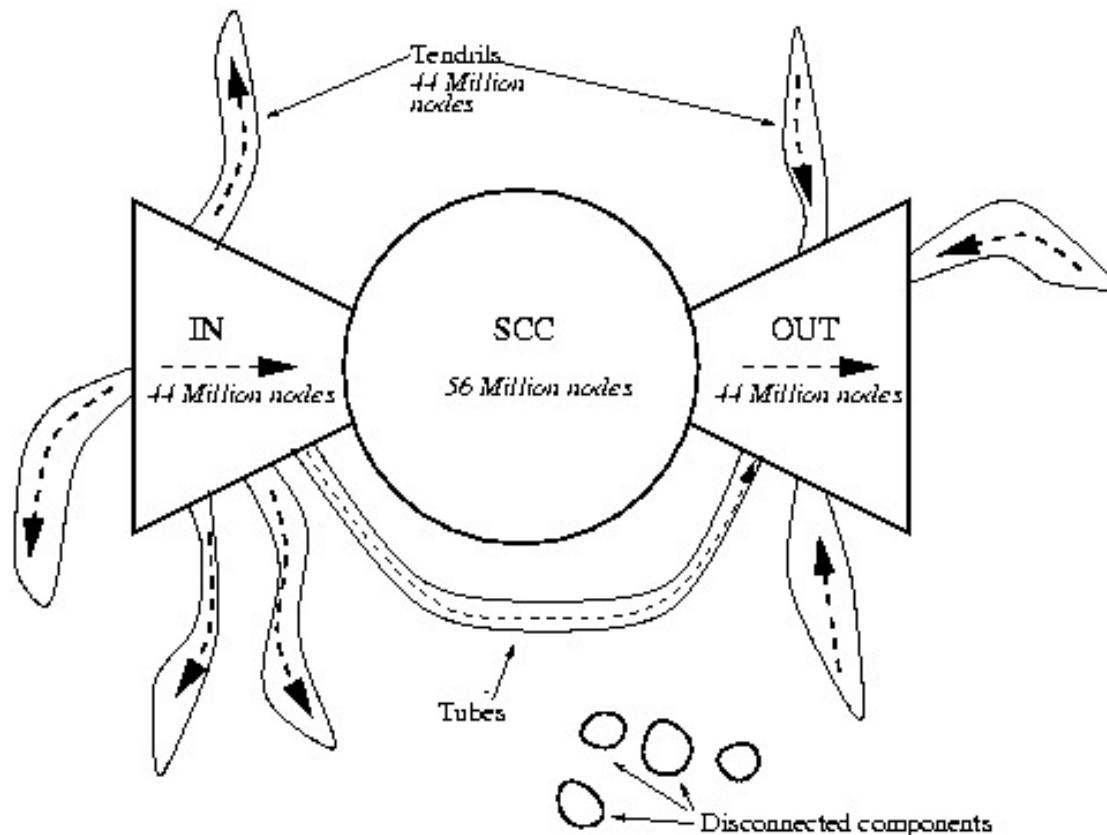
- Given a social network can you predict which individuals will connect in the future?
  - Triadic closure principle: Links are created in a way that usually closes a triangle
    - If both Bob and Charlie know Alice, then they are likely to meet at some point.
- Application: Friend/Connection **recommendations** in social networks

# Exploratory Analysis

- Trying to understand the data as a **physical phenomenon**, and describe them with simple metrics
  - What does the web graph look like?
  - How often do people repeat the same query?
  - Are friends in facebook also friends in twitter?
- The important thing is to find the right **metrics** and ask the right **questions**
- It helps our understanding of the world, and can lead to **models** of the phenomena we observe.

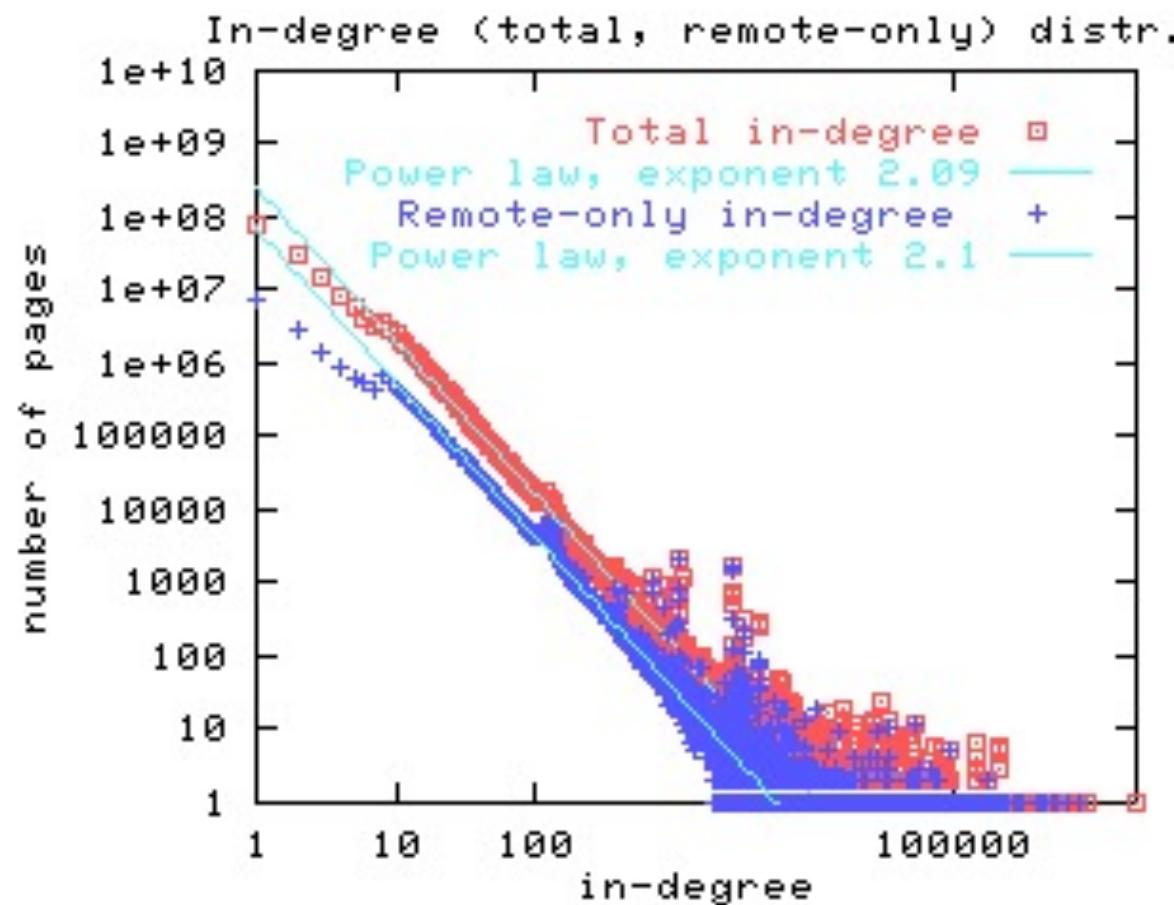
# Exploratory Analysis: The Web

- What is the structure and the properties of the web?



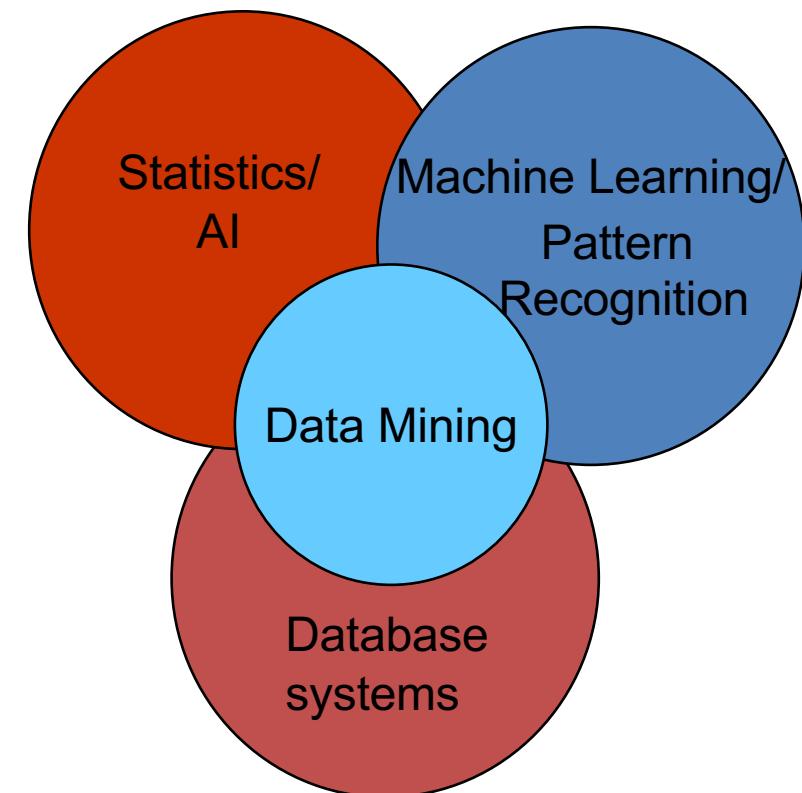
# Exploratory Analysis: The Web

- What is the distribution of the incoming links?



# Connections of Data Mining with other areas

- Draws ideas from machine learning/AI, pattern recognition, statistics, and database systems
- Traditional Techniques may be unsuitable due to
  - Enormity of data
  - High dimensionality of data
  - Heterogeneous, distributed nature of data
  - Emphasis on the use of data



# Cultures

- **Databases**: concentrate on large-scale (non-main-memory) data.
- **AI** (machine-learning): concentrate on complex methods, small data.
  - In today's world data is more important than algorithms
- **Statistics**: concentrate on models.

# Models vs. Analytic Processing

- To a database person, data-mining is an extreme form of **analytic processing** – queries that examine large amounts of data.
  - Result is the query answer.
- To a statistician, data-mining is the inference of models.
  - Result is the parameters of the model.

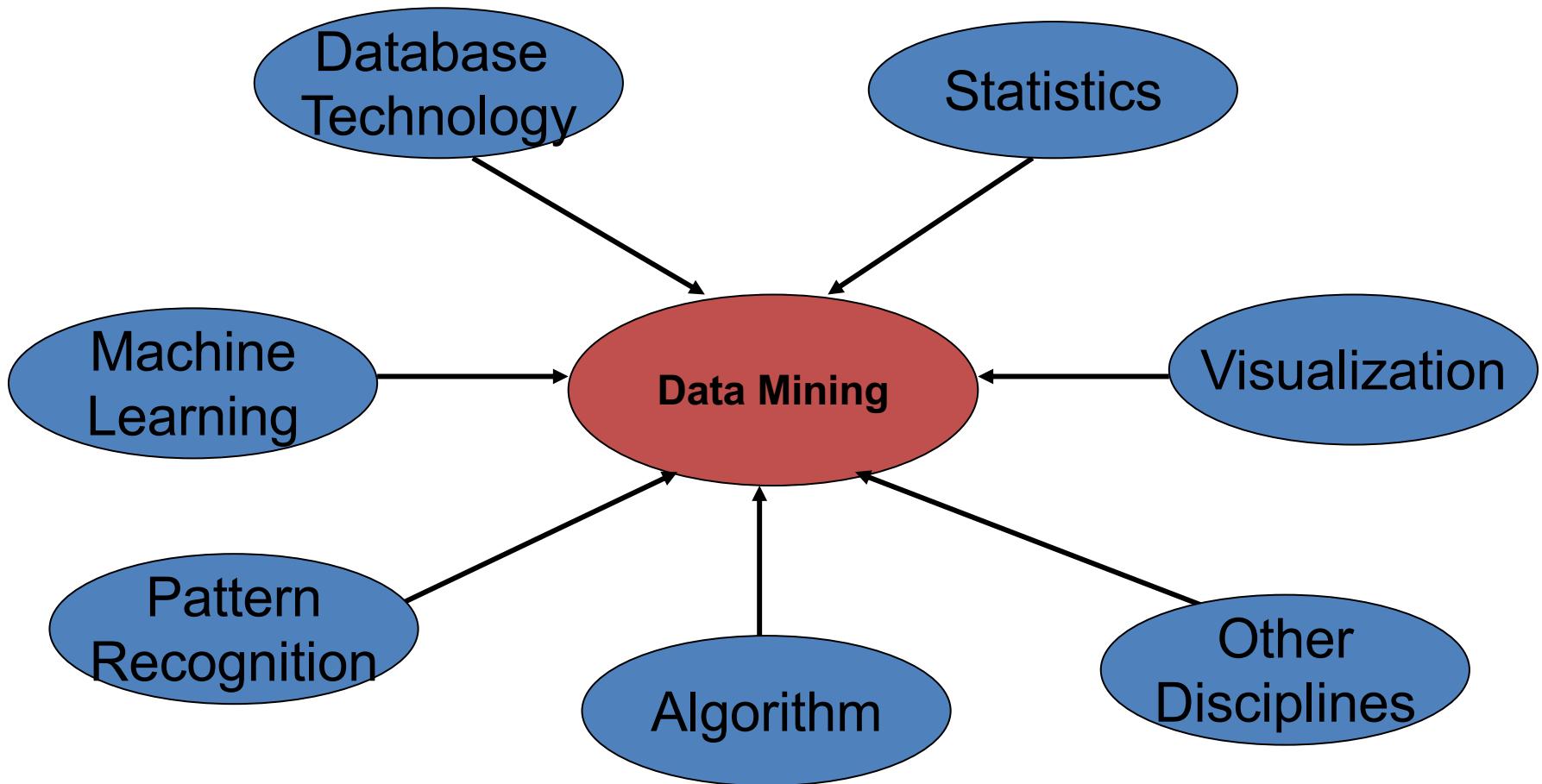
## (Way too Simple) Example

- Given a billion numbers, a DB person would compute their average and standard deviation.
- A statistician might fit the billion points to the best Gaussian distribution and report the mean and standard deviation *of that distribution*.

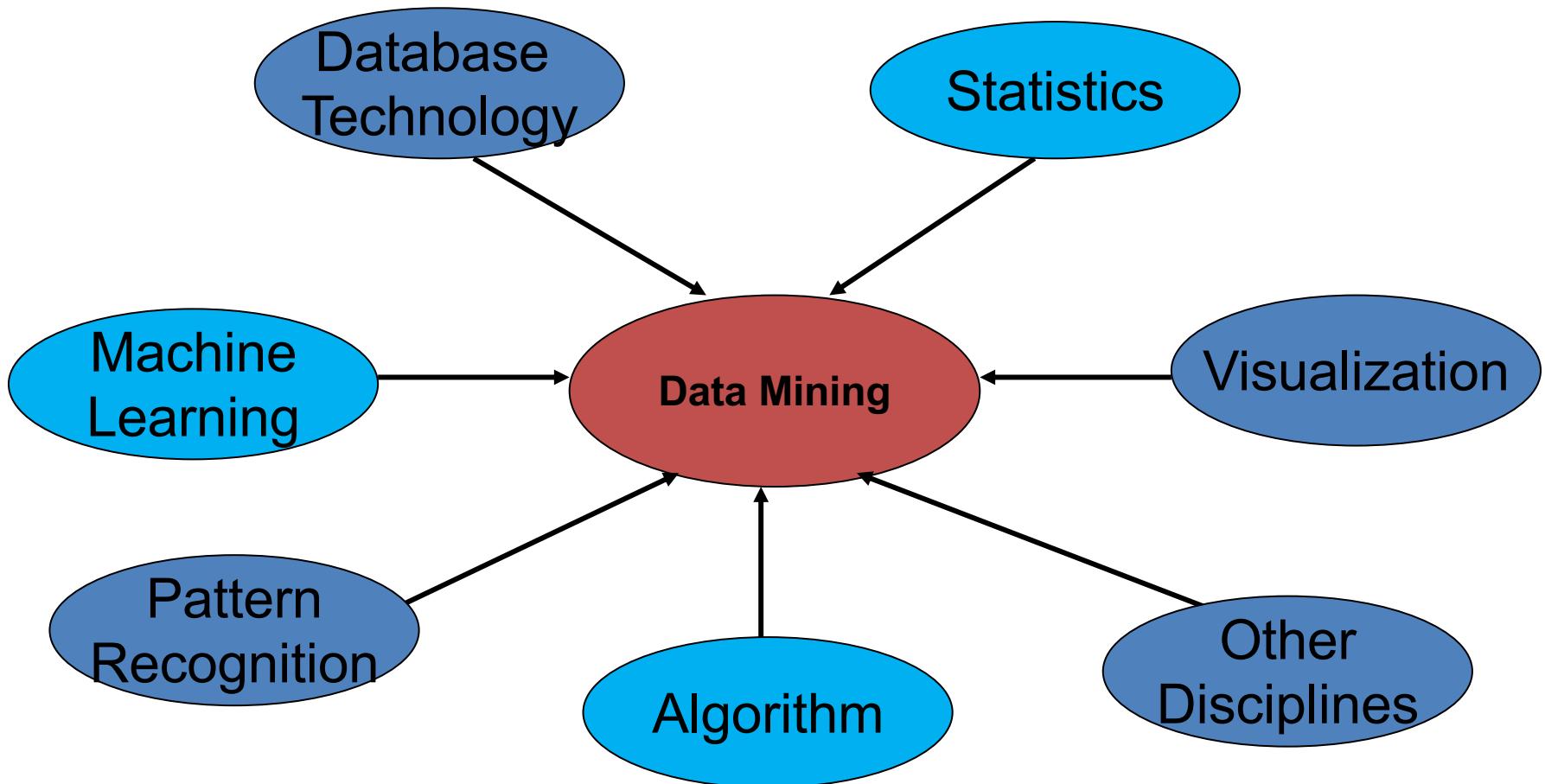
# New era of data mining

- Boundaries are becoming less clear
  - Today data mining and machine learning are synonymous. It is assumed that these algorithms should scale. It is clear that statistical inference is used for building the models.

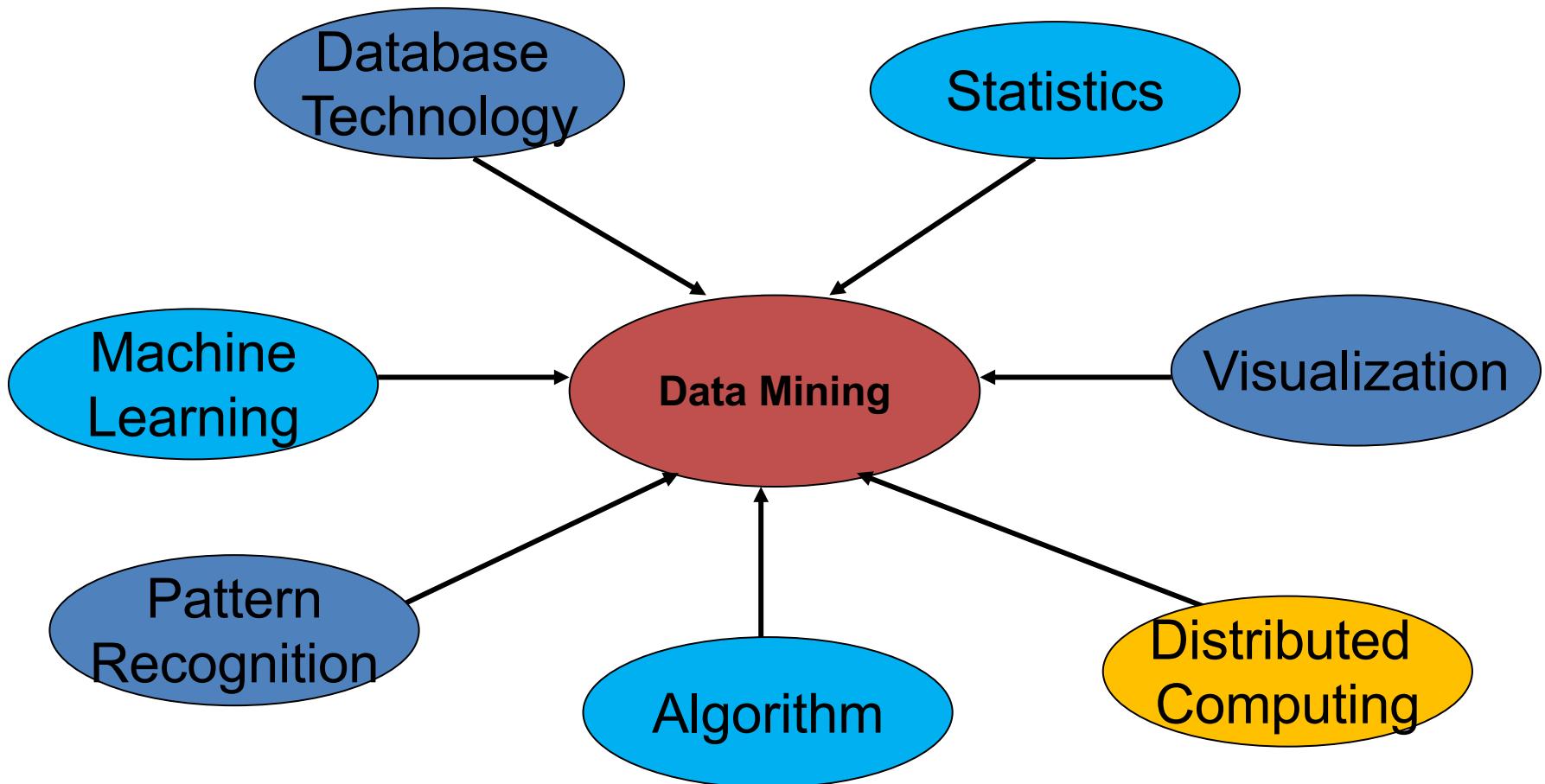
# Data Mining: Confluence of Multiple Disciplines



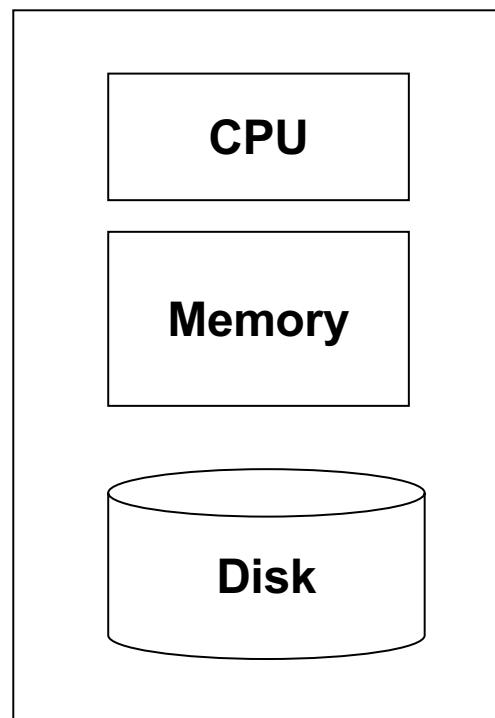
# Data Mining: Confluence of Multiple Disciplines



# Data Mining: Confluence of Multiple Disciplines



# Single-node architecture



Machine Learning, Statistics

“Classical” Data Mining

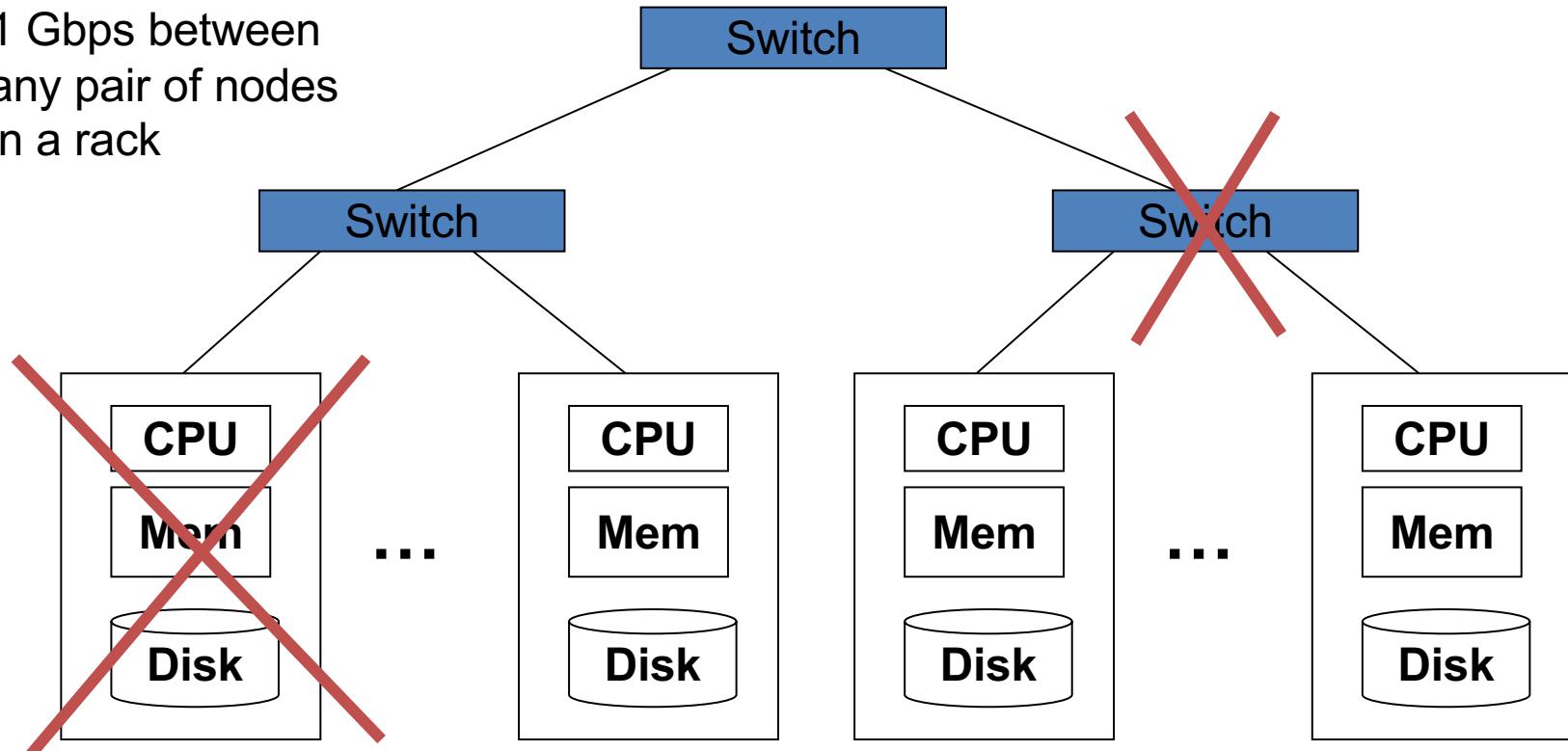
# Commodity Clusters

- Web data sets can be very large
  - Tens to hundreds of terabytes
  - Cannot mine on a single server
- Standard architecture emerging:
  - Cluster of commodity Linux nodes, Gigabit ethernet interconnect
  - Google GFS; Hadoop HDFS; Kosmix KFS
- Typical usage pattern
  - Huge files (100s of GB to TB)
  - Data is rarely updated in place
  - Reads and appends are common
- How to organize computations on this architecture?
  - Map-Reduce paradigm

# Cluster Architecture

2-10 Gbps backbone between racks

1 Gbps between  
any pair of nodes  
in a rack



Each rack contains 16-64 nodes

# Map-Reduce paradigm

- Map the data into key-value pairs
  - E.g., map a document to word-count pairs
- Group by key
  - Group all pairs of the same word, with lists of counts
- Reduce by aggregating
  - E.g. sum all the counts to produce the total count.

**Note to other teachers and users of these slides:** We would be delighted if you found this our material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: <http://www.mmds.org>

# Dimensionality Reduction: SVD & PCA

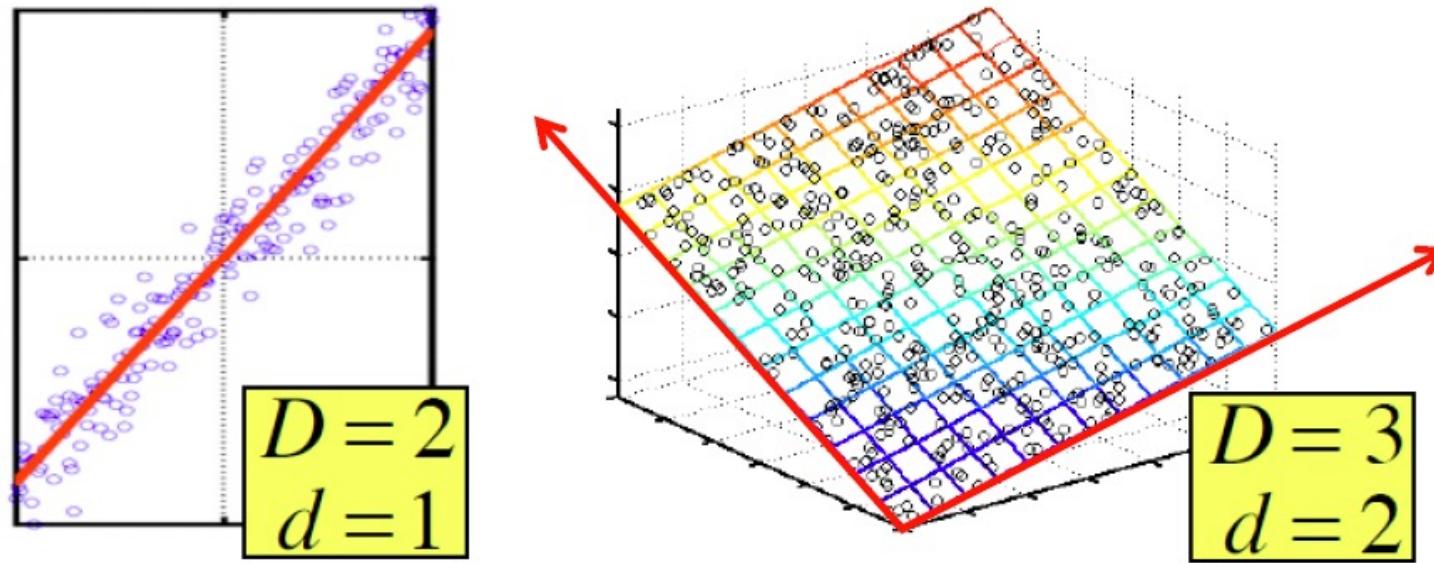
Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman  
Stanford University

<http://www.mmds.org>



# Dimensionality Reduction



- **Assumption:** Data lies on or near a low  $d$ -dimensional subspace
- **Axes of this subspace are effective representation of the data**

# Dimensionality Reduction

- Compress / reduce dimensionality:
  - $10^6$  rows;  $10^3$  columns; no updates
  - Random access to any cell(s); small error: OK

customer	day	We	Th	Fr	Sa	Su
		7/10/96	7/11/96	7/12/96	7/13/96	7/14/96
ABC Inc.		1	1	1	0	0
DEF Ltd.		2	2	2	0	0
GHI Inc.		1	1	1	0	0
KLM Co.		5	5	5	0	0
Smith		0	0	0	2	2
Johnson		0	0	0	3	3
Thompson		0	0	0	1	1

The above matrix is really “2-dimensional.” All rows can be reconstructed by scaling [1 1 1 0 0] or [0 0 0 1 1]

# Rank of a Matrix

- **Q:** What is **rank** of a matrix **A**?
- **A:** Number of **linearly independent** columns of **A**
- **For example:**
  - Matrix  $\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$  has rank **r=2**
  - **Why?** The first two rows are linearly independent, so the rank is at least 2, but all three rows are linearly dependent (the first is equal to the sum of the second and third) so the rank must be less than 3.
- **Why do we care about low rank?**
  - We can write **A** as two “basis” vectors: [1 2 1] [-2 -3 1]
  - And new coordinates of : [1 0] [0 1] [1 1]

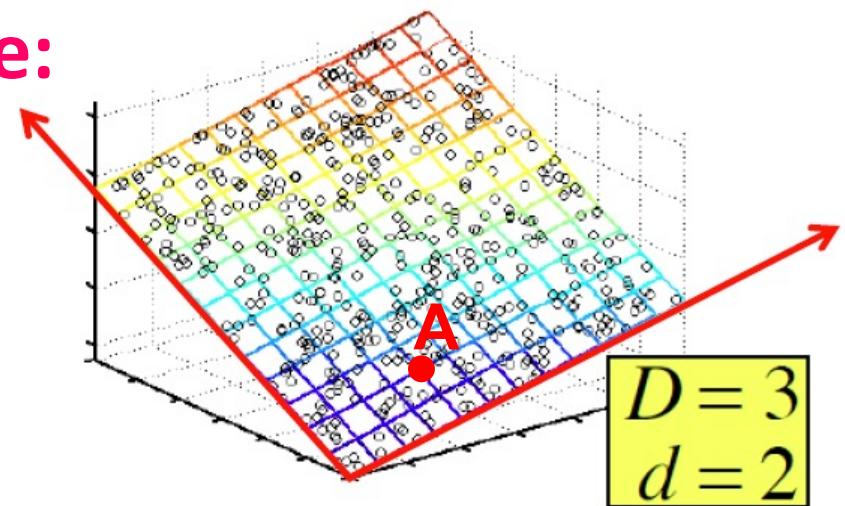
# Rank is “Dimensionality”

- Cloud of points 3D space:

- Think of point positions

as a matrix:  $\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$

1 row per point:



- We can rewrite coordinates more efficiently!

- Old basis vectors: [1 0 0] [0 1 0] [0 0 1]

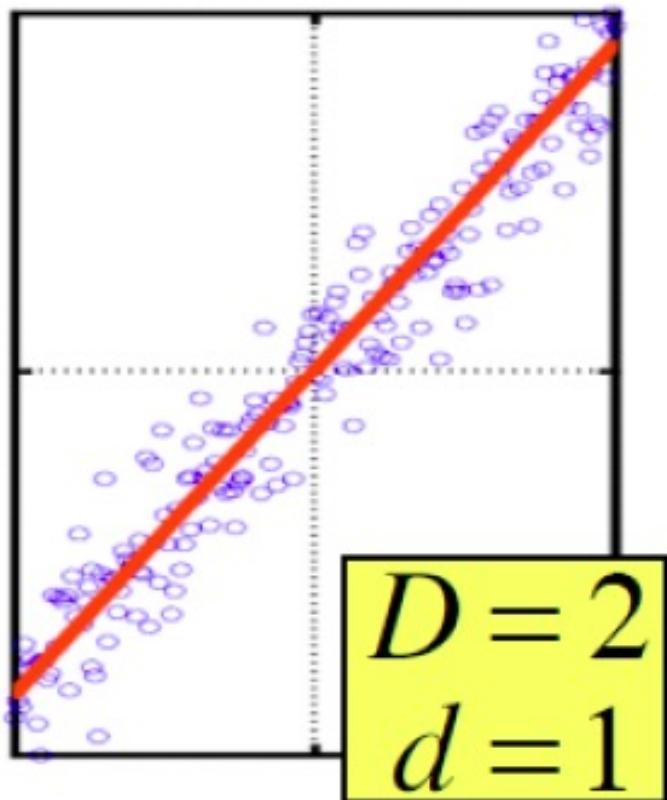
- New basis vectors: [1 2 1] [-2 -3 1]

- Then A has new coordinates: [1 0]. B: [0 1], C: [1 1]

- Notice: We reduced the number of coordinates!

# Dimensionality Reduction

- Goal of dimensionality reduction is to discover the axis of data!



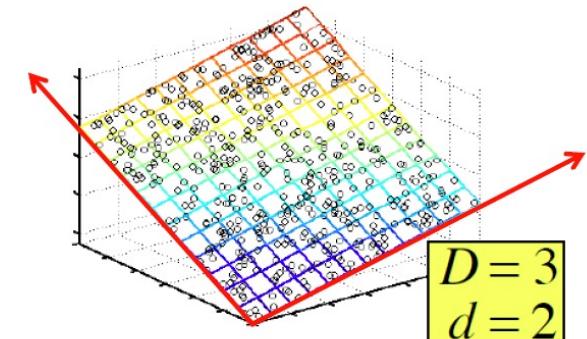
Rather than representing every point with 2 coordinates we represent each point with 1 coordinate (corresponding to the position of the point on the red line).

By doing this we incur a bit of **error** as the points do not exactly lie on the line

# Why Reduce Dimensions?

## Why reduce dimensions?

- **Discover hidden correlations/topics**
  - Words that occur commonly together
- **Remove redundant and noisy features**
  - Not all words are useful
- **Interpretation and visualization**
- **Easier storage and processing of the data**



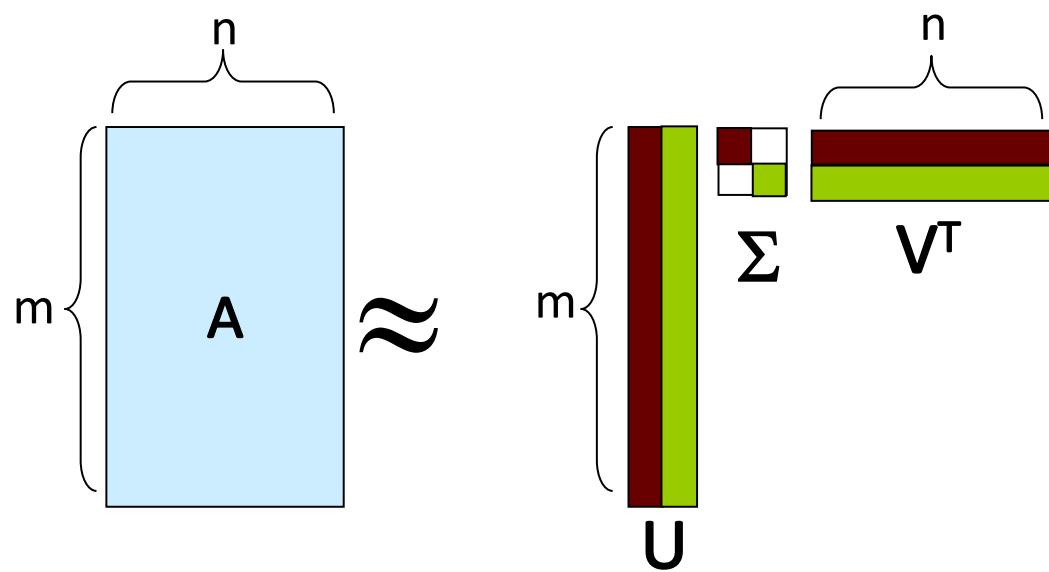
# SVD - Definition

$$\mathbf{A}_{[m \times n]} = \mathbf{U}_{[m \times r]} \Sigma_{[r \times r]} (\mathbf{V}_{[n \times r]})^T$$

- **A: Input data matrix**
  - $m \times n$  matrix (e.g.,  $m$  documents,  $n$  terms)
- **U: Left singular vectors**
  - $m \times r$  matrix ( $m$  documents,  $r$  concepts)
- **$\Sigma$ : Singular values**
  - $r \times r$  diagonal matrix (strength of each ‘concept’)  
( $r$  : rank of the matrix A)
- **V: Right singular vectors**
  - $n \times r$  matrix ( $n$  terms,  $r$  concepts)

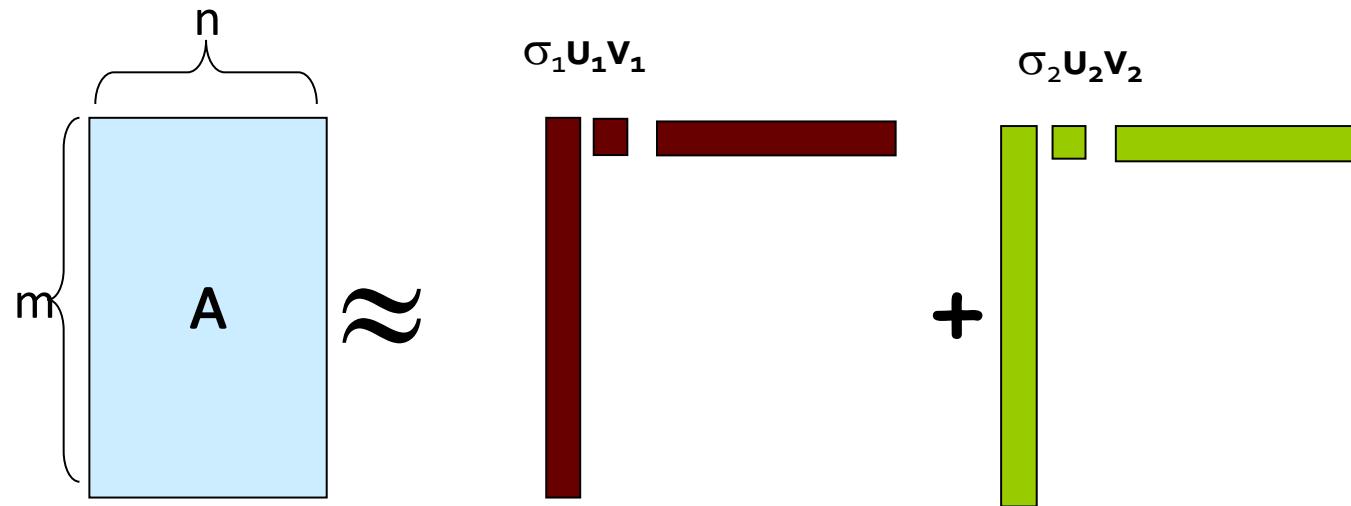
# SVD

$$\mathbf{A} \approx \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^\top$$



# SVD

$$\mathbf{A} \approx \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^\top$$



$\sigma_i$  ... scalar  
 $\mathbf{u}_i$  ... vector  
 $\mathbf{v}_i$  ... vector

# SVD - Properties

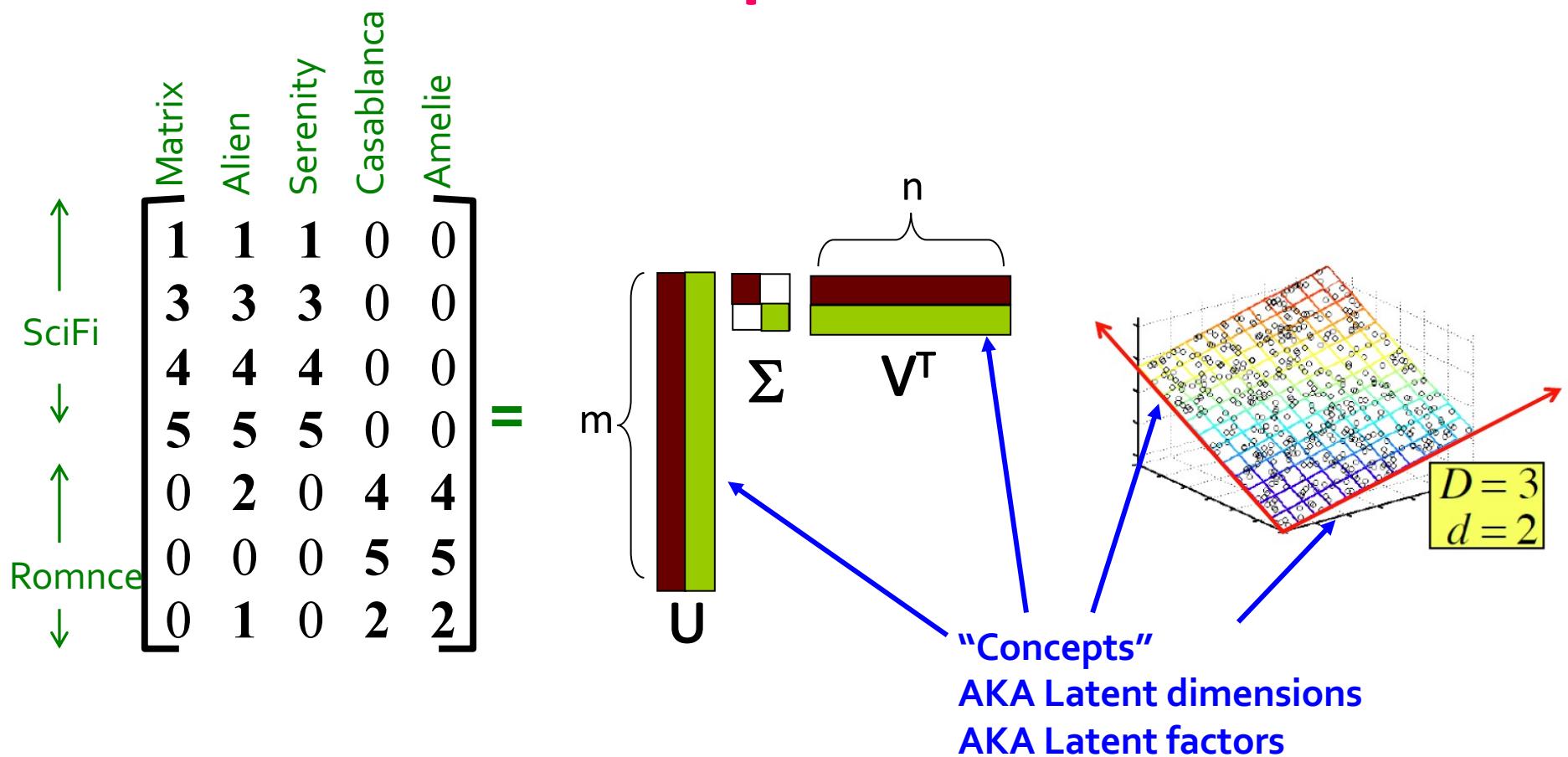
It is **always** possible to decompose a real matrix  $\mathbf{A}$  into  $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$ , where

- $\mathbf{U}, \Sigma, \mathbf{V}$ : unique
- $\mathbf{U}, \mathbf{V}$ : column orthonormal
  - $\mathbf{U}^T \mathbf{U} = \mathbf{I}; \mathbf{V}^T \mathbf{V} = \mathbf{I}$  ( $\mathbf{I}$ : identity matrix)
  - (Columns are orthogonal unit vectors)
- $\Sigma$ : diagonal
  - Entries (**singular values**) are **positive**, and sorted in decreasing order ( $\sigma_1 \geq \sigma_2 \geq \dots \geq 0$ )

Nice proof of uniqueness: <http://www.mpi-inf.mpg.de/~bast/ir-seminar-wso4/lecture2.pdf>

# SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$  - example: Users to Movies



# SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$  - example: Users to Movies

$$\begin{array}{c}
 \text{Matrix} \\
 \begin{array}{c}
 \uparrow \quad \downarrow \\
 \text{SciFi} \\
 \downarrow \quad \uparrow \\
 \text{Romance}
 \end{array}
 \end{array}
 \left[ \begin{array}{ccccc}
 1 & 1 & 1 & 0 & 0 \\
 3 & 3 & 3 & 0 & 0 \\
 4 & 4 & 4 & 0 & 0 \\
 5 & 5 & 5 & 0 & 0 \\
 0 & 2 & 0 & 4 & 4 \\
 0 & 0 & 0 & 5 & 5 \\
 0 & 1 & 0 & 2 & 2
 \end{array} \right]
 = \left[ \begin{array}{ccc}
 0.13 & 0.02 & -0.01 \\
 0.41 & 0.07 & -0.03 \\
 0.55 & 0.09 & -0.04 \\
 0.68 & 0.11 & -0.05 \\
 0.15 & -0.59 & 0.65 \\
 0.07 & -0.73 & -0.67 \\
 0.07 & -0.29 & 0.32
 \end{array} \right]
 \times \left[ \begin{array}{ccc}
 12.4 & 0 & 0 \\
 0 & 9.5 & 0 \\
 0 & 0 & 1.3
 \end{array} \right]
 \times \left[ \begin{array}{ccccc}
 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
 0.40 & -0.80 & 0.40 & 0.09 & 0.09
 \end{array} \right]$$

# SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$  - example: Users to Movies

$$\begin{array}{c}
 \text{Matrix} \\
 \begin{array}{c}
 \uparrow \quad \downarrow \\
 \text{SciFi} \quad \text{Romance}
 \end{array}
 \end{array}
 \left[ \begin{array}{ccccc}
 1 & 1 & 1 & 0 & 0 \\
 3 & 3 & 3 & 0 & 0 \\
 4 & 4 & 4 & 0 & 0 \\
 5 & 5 & 5 & 0 & 0 \\
 0 & 2 & 0 & 4 & 4 \\
 0 & 0 & 0 & 5 & 5 \\
 0 & 1 & 0 & 2 & 2
 \end{array} \right]
 = \left[ \begin{array}{ccc}
 0.13 & 0.02 & -0.01 \\
 0.41 & 0.07 & -0.03 \\
 0.55 & 0.09 & -0.04 \\
 0.68 & 0.11 & -0.05 \\
 0.15 & -0.59 & 0.65 \\
 0.07 & -0.73 & -0.67 \\
 0.07 & -0.29 & 0.32
 \end{array} \right]
 \times \left[ \begin{array}{ccc}
 12.4 & 0 & 0 \\
 0 & 9.5 & 0 \\
 0 & 0 & 1.3
 \end{array} \right]
 \times \left[ \begin{array}{ccccc}
 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
 0.40 & -0.80 & 0.40 & 0.09 & 0.09
 \end{array} \right]$$

SciFi-concept      Romance-concept

# SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$  - example:

$U$  is “user-to-concept” similarity matrix

$$\begin{array}{c}
 \text{Matrix} \\
 \begin{bmatrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ 
 \text{SciFi} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} & = & \begin{bmatrix} \text{SciFi-concept} & \text{Romance-concept} \\ 
 0.13 & 0.02 & -0.01 \\ 
 0.41 & 0.07 & -0.03 \\ 
 0.55 & 0.09 & -0.04 \\ 
 0.68 & 0.11 & -0.05 \\ 
 0.15 & -0.59 & 0.65 \\ 
 0.07 & -0.73 & -0.67 \\ 
 0.07 & -0.29 & 0.32 \end{bmatrix} \\
 \downarrow & & & \\
 \text{Romnce} & & & 
 \end{array}$$

$\times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times$

$$\begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 
 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 
 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$  - example:

Matrix

	Alien	Serenity	Casablanca	Amelie	
SciFi	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
Romance	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

$=$

	SciFi-concept		
	0.13	0.02	-0.01
	0.41	0.07	-0.03
	0.55	0.09	-0.04
	0.68	0.11	-0.05
	0.15	-0.59	0.65
	0.07	-0.73	-0.67
	0.07	-0.29	0.32

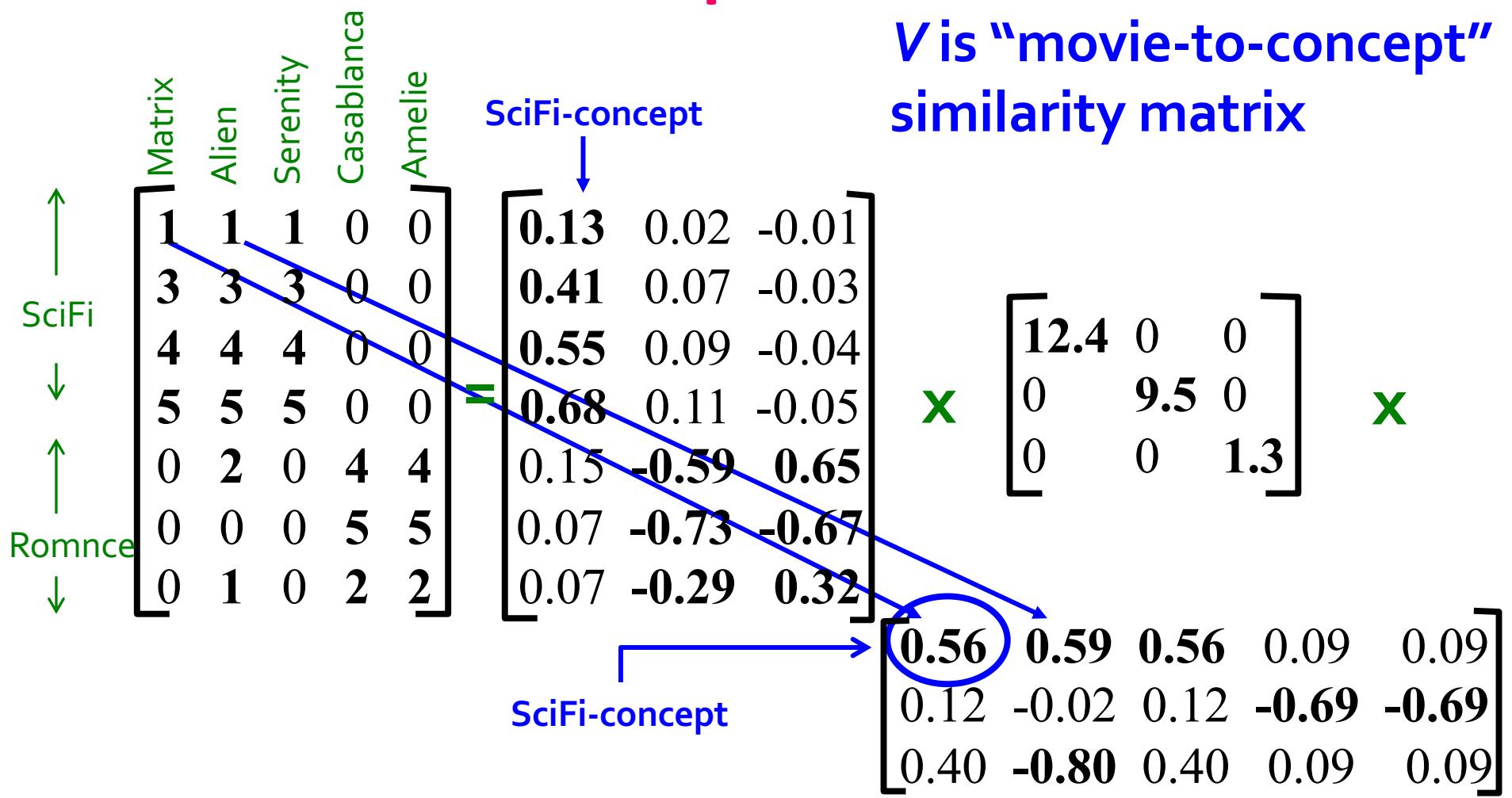
"strength" of the SciFi-concept

X	12.4	0	0	X
0	9.5	0	0	
0	0	1.3	0	

$$0.56 \quad 0.59 \quad 0.56 \quad 0.09 \quad 0.09 \\ 0.12 \quad -0.02 \quad 0.12 \quad -0.69 \quad -0.69 \\ 0.40 \quad -0.80 \quad 0.40 \quad 0.09 \quad 0.09$$

# SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$  - example:



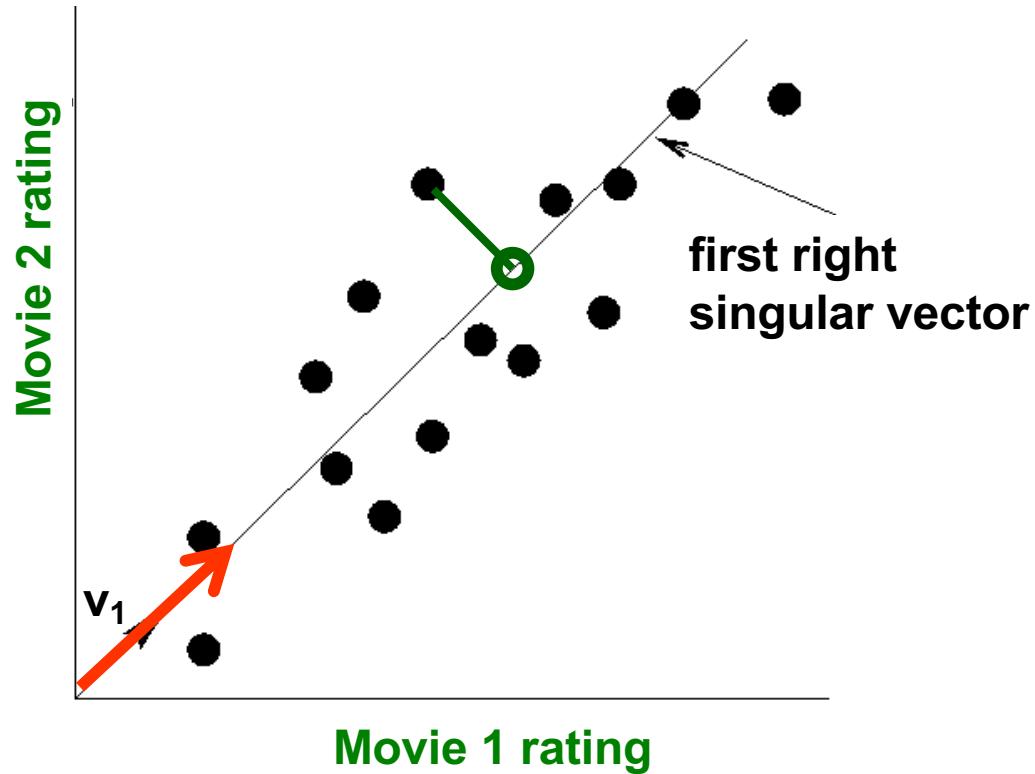
# SVD - Interpretation #1

‘movies’, ‘users’ and ‘concepts’:

- $U$ : user-to-concept similarity matrix
- $V$ : movie-to-concept similarity matrix
- $\Sigma$ : its diagonal elements:  
‘strength’ of each concept

# Dimensionality Reduction with SVD

# SVD – Dimensionality Reduction



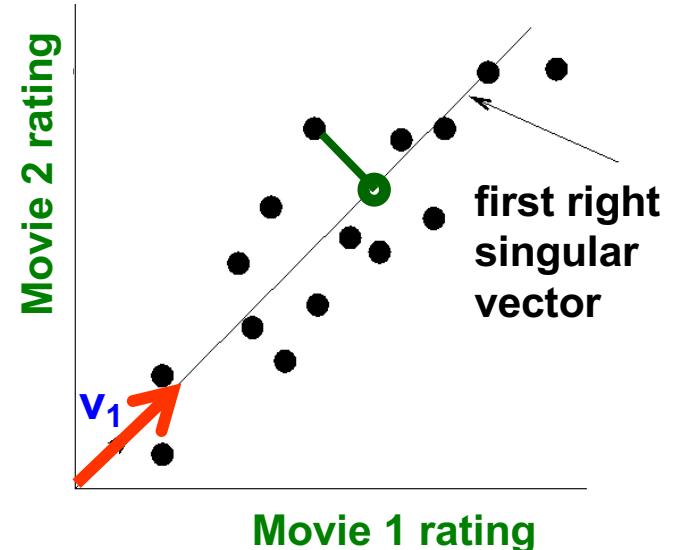
- Instead of using two coordinates  $(x, y)$  to describe point locations, let's use only one coordinate ( $z$ )
- Point's position is its location along vector  $v_1$
- **How to choose  $v_1$ ? Minimize reconstruction error**

# SVD – Dimensionality Reduction

- **Goal:** Minimize the sum of reconstruction errors:

$$\sum_{i=1}^N \sum_{j=1}^D \|x_{ij} - z_{ij}\|^2$$

- where  $x_{ij}$  are the “old” and  $z_{ij}$  are the “new” coordinates
- **SVD gives ‘best’ axis to project on:**
  - ‘best’ = minimizing the reconstruction errors
  - In other words, **minimum reconstruction error**

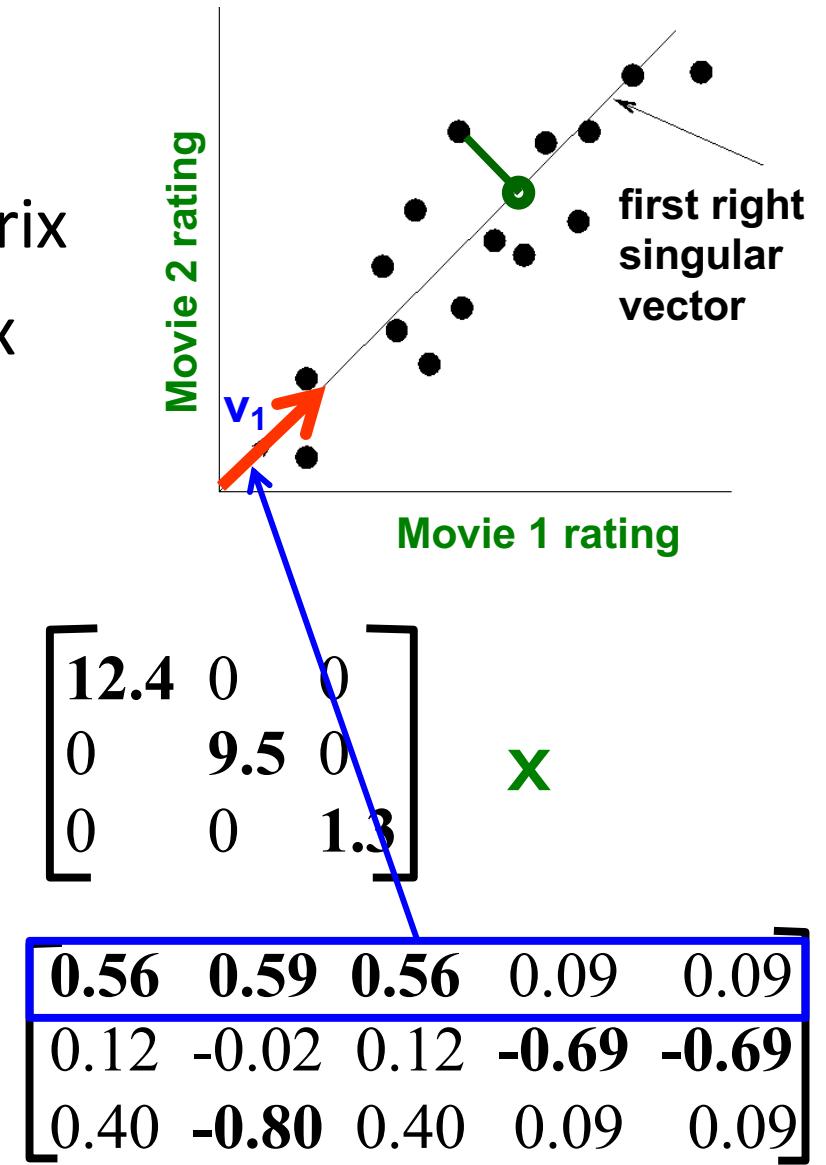


# SVD - Interpretation #2

## ■ $A = U \Sigma V^T$ - example:

- $V$ : “movie-to-concept” matrix
- $U$ : “user-to-concept” matrix

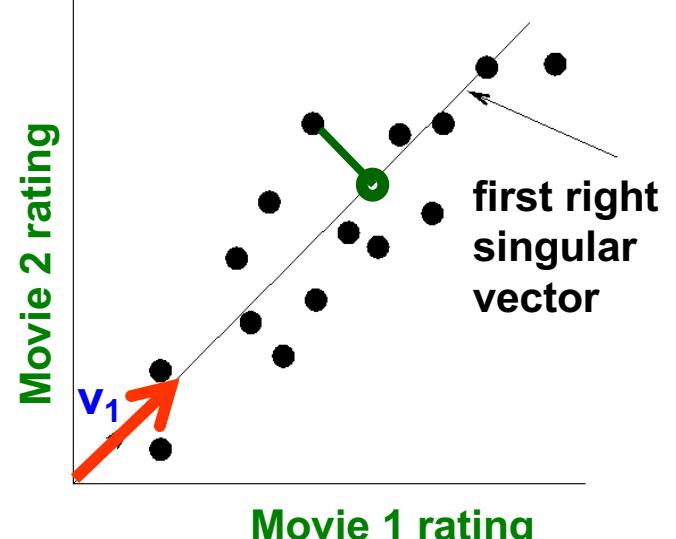
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}$$



# SVD - Interpretation #2

- $A = U \Sigma V^T$  - example:

variance ('spread')  
on the  $v_1$  axis



$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

**variance ('spread')  
on the  $v_1$  axis**

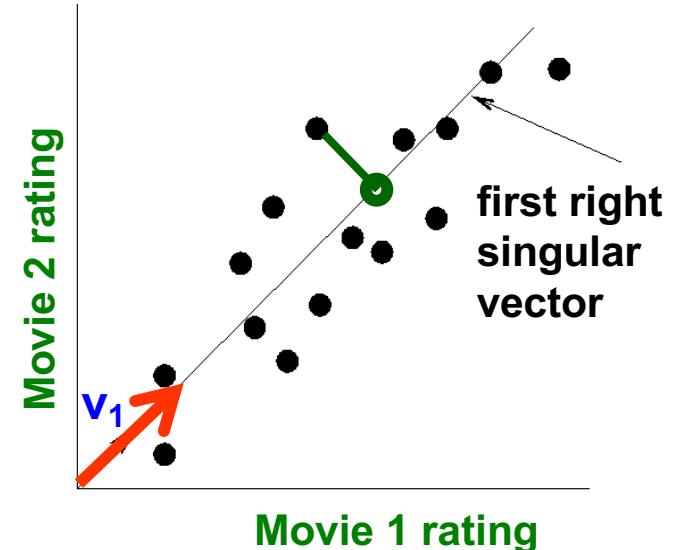
# SVD - Interpretation #2

$A = U \Sigma V^T$  - example:

- $U \Sigma$ : Gives the coordinates of the points in the projection axis

1	1	1	0	0
3	3	3	0	0
4	4	4	0	0
5	5	5	0	0
0	2	0	4	4
0	0	0	5	5
0	1	0	2	2

Projection of users  
on the “Sci-Fi” axis  
 $(U \Sigma)^T$ :



1.61	0.19	-0.01
5.08	0.66	-0.03
6.82	0.85	-0.05
8.43	1.04	-0.06
1.86	-5.60	0.84
0.86	-6.93	-0.87
0.86	-2.75	0.41

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & \cancel{1.3} \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & \cancel{1.3} \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

The diagram illustrates the SVD decomposition of a matrix. On the left is the original matrix X. To its right is the SVD decomposition:  $X \approx U \Sigma V^T$ . The matrix U has columns that are scaled versions of the original rows. The diagonal matrix  $\Sigma$  contains the singular values. The matrix  $V$  is shown as a 3x5 matrix, where the last two columns are crossed out with red lines, indicating they correspond to the zero singular values. The first column of  $V$  is also crossed out.

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 \\ 0.41 & 0.07 \\ 0.55 & 0.09 \\ 0.68 & 0.11 \\ 0.15 & -0.59 \\ 0.07 & -0.73 \\ 0.07 & -0.29 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \end{bmatrix}$$

# SVD - Interpretation #2

## More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.92 & 0.95 & 0.92 & 0.01 & 0.01 \\ 2.91 & 3.01 & 2.91 & -0.01 & -0.01 \\ 3.90 & 4.04 & 3.90 & 0.01 & 0.01 \\ 4.82 & 5.00 & 4.82 & 0.03 & 0.03 \\ 0.70 & 0.53 & 0.70 & 4.11 & 4.11 \\ -0.69 & 1.34 & -0.69 & 4.78 & 4.78 \\ 0.32 & 0.23 & 0.32 & 2.01 & 2.01 \end{bmatrix}$$

Frobenius norm:

$$\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$$

$$\|A-B\|_F = \sqrt{\sum_{ij} (A_{ij}-B_{ij})^2}$$

is “small”

# SVD – Best Low Rank Approx.

$$A = U \Sigma V^T$$

B is best approximation of A

$$B = \bar{U} \bar{\Sigma} \bar{V}^T$$

# SVD – Best Low Rank Approx.

- Theorem:

Let  $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$  and  $\mathbf{B} = \mathbf{U} \mathbf{S} \mathbf{V}^T$  where

$\mathbf{S}$  = **diagonal  $r \times r$  matrix** with  $s_i = \sigma_i$  ( $i=1\dots k$ ) else  $s_i = 0$   
then  $\mathbf{B}$  is a **best**  $\text{rank}(\mathbf{B})=k$  approx. to  $\mathbf{A}$

What do we mean by “best”:

- $\mathbf{B}$  is a solution to  $\min_B \|A-B\|_F$  where  $\text{rank}(B)=k$

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}_{m \times n} = \begin{pmatrix} u_{11} & \dots & u_{r1} \\ \vdots & \ddots & \vdots \\ u_{m1} & & u_{rn} \end{pmatrix}_{m \times r} \begin{pmatrix} \sigma_{11} & \Sigma_0 & \dots \\ 0 & \ddots & \\ \vdots & & r \times r \end{pmatrix} \begin{pmatrix} v_{11} & \dots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}_{r \times n}$$

$$\|A - B\|_F = \sqrt{\sum_{ij} (A_{ij} - B_{ij})^2}$$

Details!

# SVD – Best Low Rank Approx.

- Theorem: Let  $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$  ( $\sigma_1 \geq \sigma_2 \geq \dots$ ,  $\text{rank}(\mathbf{A})=r$ )  
then  $\mathbf{B} = \mathbf{U} \mathbf{S} \mathbf{V}^T$ 
  - $\mathbf{S}$  = diagonal  $r \times r$  matrix where  $s_i = \sigma_i$  ( $i=1\dots k$ ) else  $s_i=0$  is a best rank- $k$  approximation to  $\mathbf{A}$ :
  - $\mathbf{B}$  is a solution to  $\min_B \|\mathbf{A}-\mathbf{B}\|_F$  where  $\text{rank}(\mathbf{B})=k$

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}_{m \times n} = \begin{pmatrix} u_{11} & \dots & u_{mr} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}_{m \times r} \begin{pmatrix} \sigma_{11} & & & \\ 0 & \ddots & & \\ & \ddots & \ddots & \\ & & 0 & \sigma_{rr} \end{pmatrix}_{r \times r} \begin{pmatrix} v_{11} & \dots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}_{r \times n}$$

- We will need 2 facts:
  - $\|M\|_F = \sum_i (q_{ii})^2$  where  $M=PQR$  is SVD of  $M$
  - $\mathbf{U} \Sigma \mathbf{V}^T - \mathbf{U} \mathbf{S} \mathbf{V}^T = \mathbf{U} (\Sigma - \mathbf{S}) \mathbf{V}^T$

# SVD – Best Low Rank Approx.

Details!

- We will need 2 facts:

- $\|M\|_F = \sum_k (q_{kk})^2$  where  $M = PQR$  is SVD of  $M$

$$\|M\| = \sum_i \sum_j (m_{ij})^2 = \sum_i \sum_j \left( \sum_k \sum_\ell p_{ik} q_{k\ell} r_{\ell j} \right)^2$$

$$\|M\| = \sum_i \sum_j \sum_k \sum_\ell \sum_n \sum_m p_{ik} q_{k\ell} r_{\ell j} p_{in} q_{nm} r_{mj}$$

$\sum_i p_{ik} p_{in}$  is 1 if  $k = n$  and 0 otherwise

- $U \Sigma V^T - U S V^T = U (\Sigma - S) V^T$

We apply:

- P column orthonormal
- R row orthonormal
- Q is diagonal

Details!

# SVD – Best Low Rank Approx.

- $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$ ,  $\mathbf{B} = \mathbf{U} \mathbf{S} \mathbf{V}^T$  ( $\sigma_1 \geq \sigma_2 \geq \dots \geq 0$ ,  $\text{rank}(A)=r$ )
  - $\mathbf{S}$  = diagonal  $n \times n$  matrix where  $s_i = \sigma_i$  ( $i=1 \dots k$ ) else  $s_i = 0$   
**then**  $\mathbf{B}$  is solution to  $\min_{\mathbf{B}} \|\mathbf{A} - \mathbf{B}\|_F$ ,  $\text{rank}(\mathbf{B})=k$
- Why?

$$\min_{B, \text{rank}(B)=k} \|A - B\|_F = \min \|\Sigma - S\|_F = \min_{s_i} \sum_{i=1}^r (\sigma_i - s_i)^2$$

We used:  $\mathbf{U} \Sigma \mathbf{V}^T - \mathbf{U} \mathbf{S} \mathbf{V}^T = \mathbf{U} (\Sigma - \mathbf{S}) \mathbf{V}^T$

- We want to choose  $s_i$  to minimize  $\sum_i (\sigma_i - s_i)^2$
- Solution is to set  $s_i = \sigma_i$  ( $i=1 \dots k$ ) and other  $s_i = 0$

$$= \min_{s_i} \sum_{i=1}^k (\sigma_i - s_i)^2 + \sum_{i=k+1}^r \sigma_i^2 = \sum_{i=k+1}^r \sigma_i^2$$

# SVD - Interpretation #2

**Equivalent:**

**'spectral decomposition' of the matrix:**

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} | & | \\ u_1 & u_2 \\ | & | \end{bmatrix} \times \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \end{bmatrix} \times \begin{bmatrix} v_1 & \\ & v_2 \end{bmatrix}$$

# SVD - Interpretation #2

**Equivalent:**

'spectral decomposition' of the matrix

$$\begin{array}{c} \xleftarrow{\quad m \quad} \\ \uparrow \\ \left[ \begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{array} \right] = \end{array} \begin{array}{c} \xleftarrow{\quad k \text{ terms} \quad} \\ \sigma_1 \quad \xrightarrow{\quad U_1 \quad} \quad \xleftarrow{\quad V^T_1 \quad} + \quad \sigma_2 \quad \xrightarrow{\quad U_2 \quad} \quad \xleftarrow{\quad V^T_2 \quad} + \dots \\ n \times 1 \quad 1 \times m \end{array}$$

Assume:  $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots \geq 0$

Why is setting small  $\sigma_i$  to 0 the right thing to do?

Vectors  $u_i$  and  $v_i$  are unit length, so  $\sigma_i$  scales them.

So, zeroing small  $\sigma_i$  introduces less error.

# SVD - Interpretation #2

**Q: How many  $\sigma$ s to keep?**

**A:** Rule-of-a thumb:

**keep 80-90% of 'energy' =  $\sum_i \sigma_i^2$**

$$\begin{array}{c} \text{← } m \text{ →} \\ \uparrow \quad \downarrow \\ \left[ \begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{array} \right] = \sigma_1 \mathbf{U}_1 \mathbf{V}^T_1 + \sigma_2 \mathbf{U}_2 \mathbf{V}^T_2 + \dots \end{array}$$

**Assume:  $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots$**

# SVD - Complexity

- **To compute SVD:**
  - $O(nm^2)$  or  $O(n^2m)$  (whichever is less)
- **But:**
  - Less work, if we just want singular values
  - or if we want first  $k$  singular vectors
  - or if the matrix is sparse
- **Implemented in** linear algebra packages like
  - LINPACK, Matlab, SPlus, Mathematica ...

# SVD - Conclusions so far

- **SVD:  $A = U \Sigma V^T$ : unique**
  - $U$ : user-to-concept similarities
  - $V$ : movie-to-concept similarities
  - $\Sigma$  : strength of each concept
- **Dimensionality reduction:**
  - keep the few largest singular values (80-90% of ‘energy’)
  - SVD: picks up linear correlations

# Relation to Eigen-decomposition

- SVD gives us:
  - $A = U \Sigma V^T$
- Eigen-decomposition:
  - $A = X \Lambda X^T$
  - A is symmetric
  - U, V, X are orthonormal ( $U^T U = I$ ),
  - $\Lambda$ ,  $\Sigma$  are diagonal
- Now let's calculate:
  - $AA^T =$
  - $A^T A = V \Sigma^T U^T (U \Sigma V^T) = V \Sigma \Sigma^T V^T$

# Relation to Eigen-decomposition

- SVD gives us:

- $A = U \Sigma V^T$

- Eigen-decomposition:

- $A = X \Lambda X^T$

- A is symmetric

- U, V, X are orthonormal ( $U^T U = I$ ),

- $\Lambda$ ,  $\Sigma$  are diagonal

- Now let's calculate:

- $AA^T = U\Sigma V^T (U\Sigma V^T)^T = U\Sigma V^T (V\Sigma^T U^T) = U\Sigma\Sigma^T U^T$

- $A^T A = V \Sigma^T U^T (U\Sigma V^T) = V \Sigma \Sigma^T V^T$

Shows how to compute  
SVD using eigenvalue  
decomposition!

$$\begin{array}{c} \downarrow \\ X \Lambda^2 X^T \\ \downarrow \quad \downarrow \quad \downarrow \\ U \Sigma \Sigma^T U^T \\ \uparrow \quad \uparrow \quad \uparrow \\ X \Lambda^2 X^T \end{array}$$

# Example of SVD & Conclusion

# Case study: How to query?

- Q: Find users that like ‘Matrix’
- A: Map query into a ‘concept space’ – how?

$$\begin{array}{c} \text{↑ SciFi} \\ \text{↓} \\ \text{↑ Romance} \\ \text{↓} \end{array} \begin{matrix} \text{Matrix} \\ \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{matrix} \left[ \begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{array} \right] = \left[ \begin{array}{ccc} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{array} \right] \times \left[ \begin{array}{ccc} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{array} \right] \times \left[ \begin{array}{ccccc} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{array} \right]$$

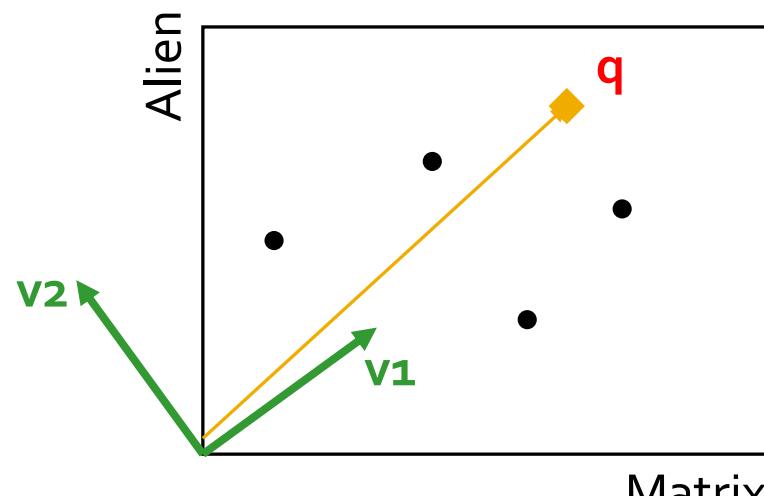
J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, <http://www.mmds.org>

# Case study: How to query?

- Q: Find users that like ‘Matrix’
- A: Map query into a ‘concept space’ – how?

$$q = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{bmatrix} \quad \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Project into concept space:  
Inner product with each  
'concept' vector  $v_i$

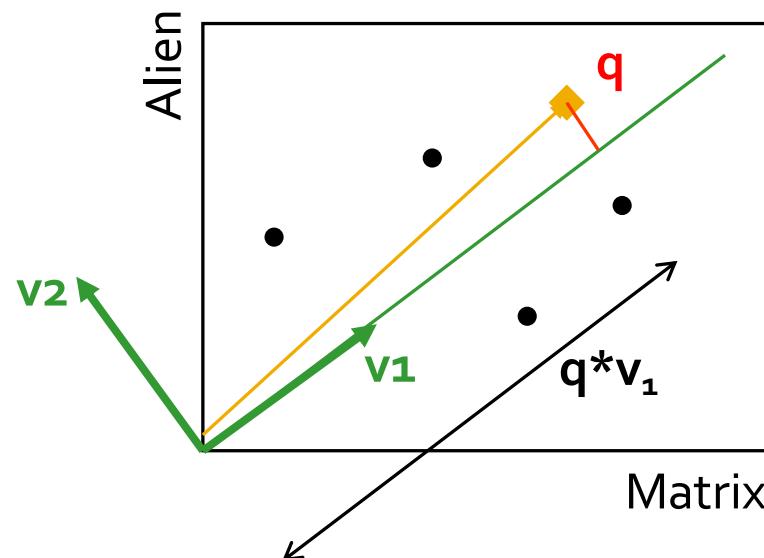


# Case study: How to query?

- Q: Find users that like ‘Matrix’
- A: Map query into a ‘concept space’ – how?

$$q = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{bmatrix} \quad \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Project into concept space:  
Inner product with each  
'concept' vector  $v_i$



# Case study: How to query?

Compactly, we have:

$$\mathbf{q}_{\text{concept}} = \mathbf{q} \mathbf{V}$$

E.g.:

$$\mathbf{q} = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix} = \begin{bmatrix} 2.8 \\ 0.6 \end{bmatrix}$$

SciFi-concept

movie-to-concept  
similarities ( $\mathbf{V}$ )

# Case study: How to query?

- How would the user  $d$  that rated ('Alien', 'Serenity') be handled?

$$d_{\text{concept}} = d \mathbf{V}$$

E.g.:

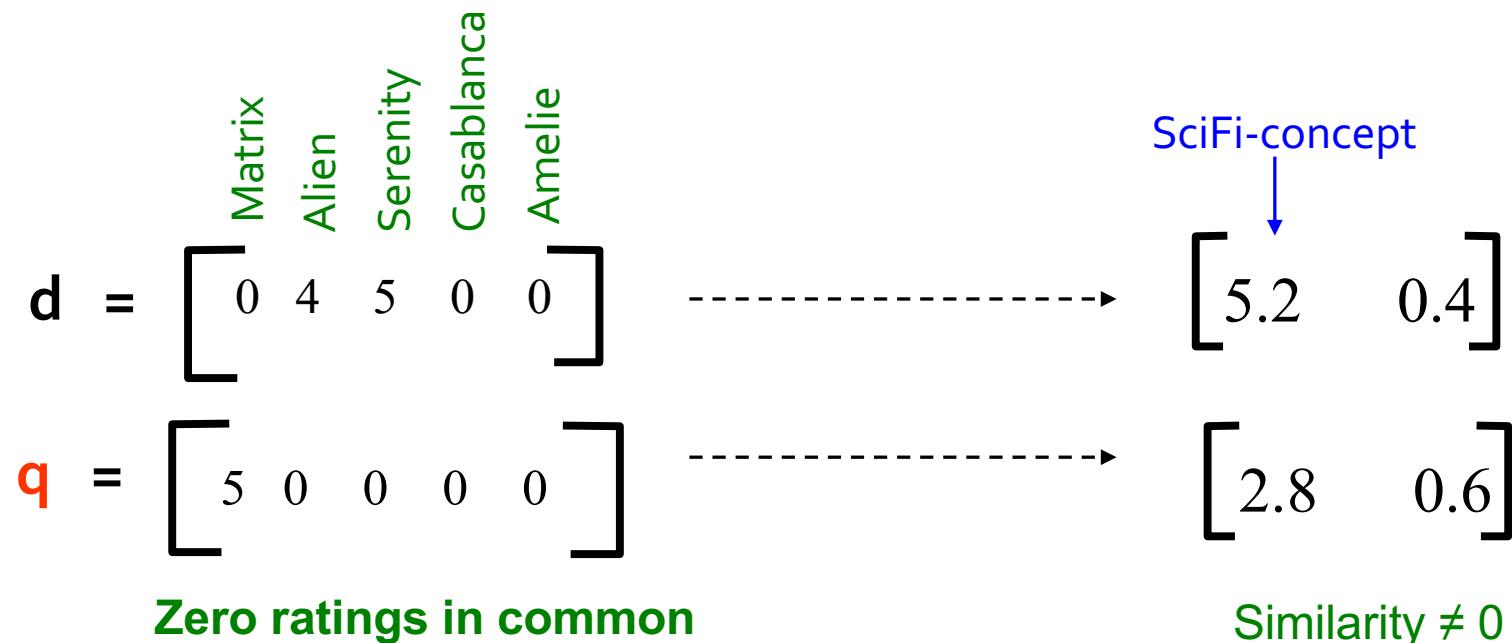
$$\mathbf{q} = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix}$$

movie-to-concept  
similarities ( $\mathbf{V}$ )

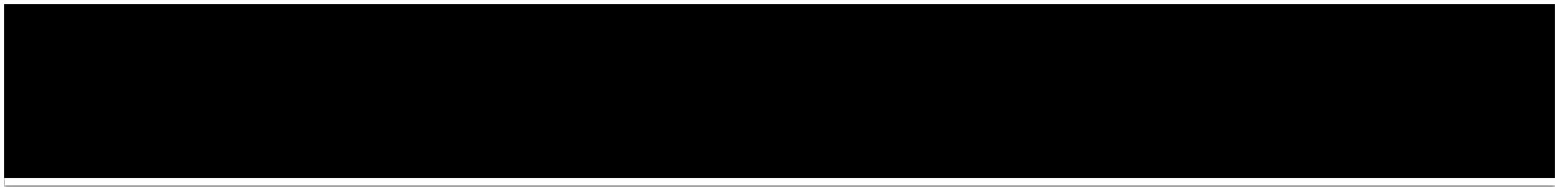
SciFi-concept  
↓  
 $= \begin{bmatrix} 5.2 & 0.4 \end{bmatrix}$

# Case study: How to query?

- **Observation:** User  $d$  that rated ('Alien', 'Serenity') will be **similar** to user  $q$  that rated ('Matrix'), although  $d$  and  $q$  have **zero ratings in common!**









# PCA

# PCA

- Useful to **learn lower dimensional representations** of the data.
- Powerful **unsupervised learning** techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.
- PCA learns a representation that has **lower dimensionality** than the original input.
- It also learns a representation whose **elements have no linear correlation** with each other

# PCA

- To achieve full independence, a representation learning algorithm must also **remove the nonlinear relationships between variables.**

# PCA

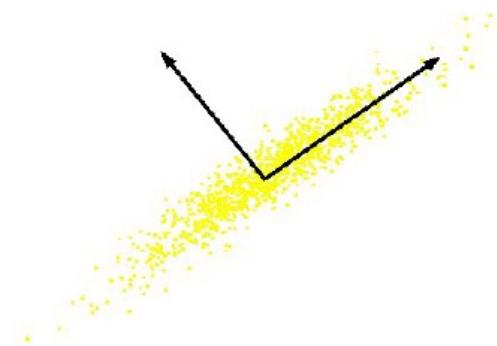
- When you apply **transformation** to the original data, the **axis corresponding to the principal eigenvector** is the one along which the points are most “spread out,”
- More precisely, this axis is the one along which the **variance of the data is maximized**
- **The high-dimensional data** can be replaced by its projection onto the most important axes. These axes are the **ones corresponding to the largest eigenvalues**.
- Thus, the **original data** is approximated by data that has many **fewer dimensions** and that summarizes well the original data.

# PCA Useful for

- Visualization
- Further processing by machine learning algorithms
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions à better generalization
- Noise removal (improving data quality)

# Principal Component Analysis (PCA)

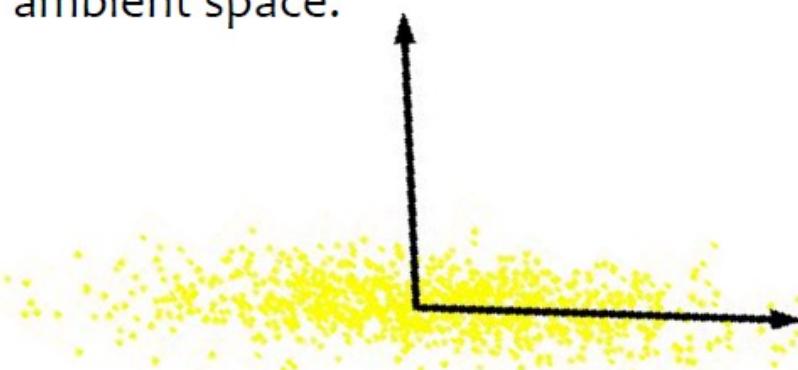
**What is PCA:** Unsupervised technique for extracting variance structure from high dimensional datasets.



- PCA is an orthogonal projection or transformation of the data into a (possibly lower dimensional) subspace so that the variance of the projected data is maximized.

# Principal Component Analysis (PCA)

Intrinsically lower dimensional than the dimension of the ambient space.



Only one relevant feature

If we rotate data, again only one coordinate is more important.

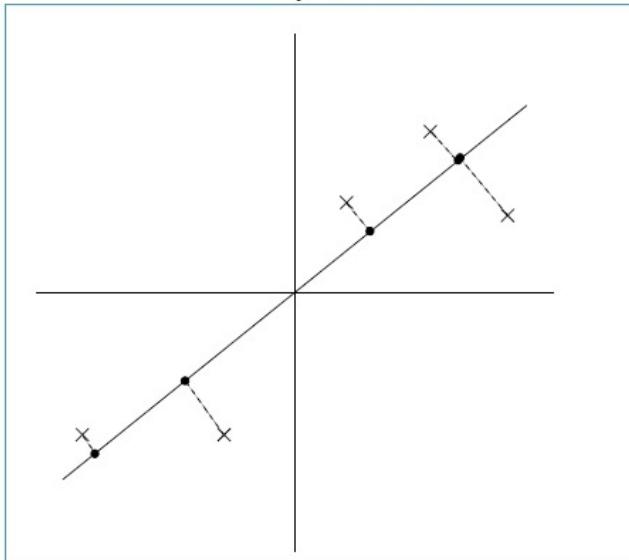


Both features are relevant

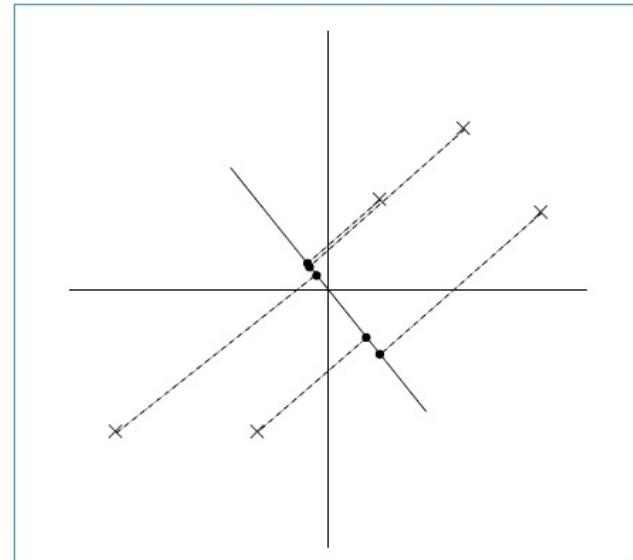
# Maximizing the Variance

- Consider the two projections below
- Which maximizes the variance?

Option A



Option B



# Principal Components Analysis

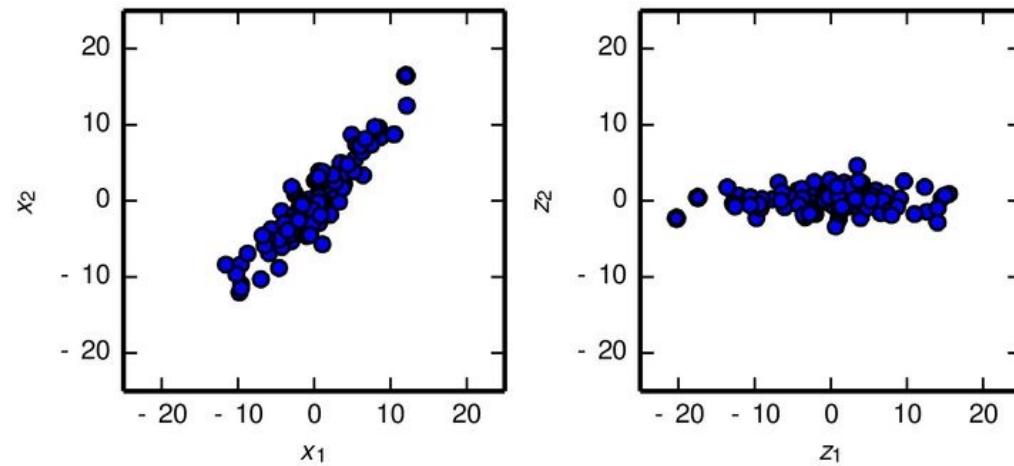


Figure 5.8

(Goodfellow 2016)

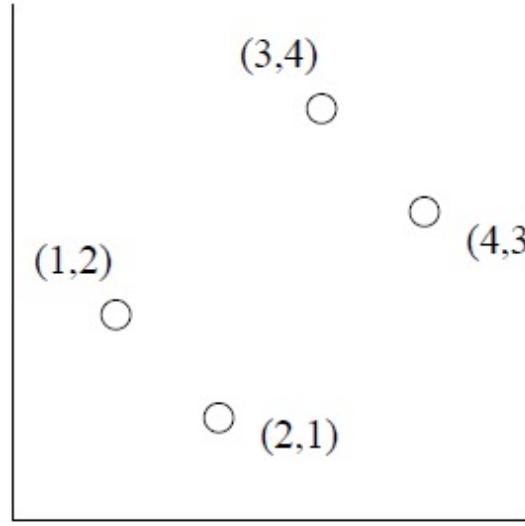


Figure 11.1: Four points in a two-dimensional space

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

Compute  $M^T M$ , which is

$$M^T M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} = \begin{bmatrix} 30 & 28 \\ 28 & 30 \end{bmatrix}$$

We may find the eigenvalues of the matrix above by solving the equation

$$(30 - \lambda)(30 - \lambda) - 28 \times 28 = 0$$

- Eigen values are 58 and 2

$$\begin{bmatrix} 30 & 28 \\ 28 & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 58 \begin{bmatrix} x \\ y \end{bmatrix}$$

When we multiply out the matrix and vector we get two equations

$$\begin{aligned} 30x + 28y &= 58x \\ 28x + 30y &= 58y \end{aligned}$$

Both equations tell us the same thing:  $x = y$ . Thus, the unit eigenvector corresponding to the principal eigenvalue 58 is

$$\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

For the second eigenvalue, 2, we perform the same process. Multiply out

$$\begin{bmatrix} 30 & 28 \\ 28 & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 2 \begin{bmatrix} x \\ y \end{bmatrix}$$

to get the two equations

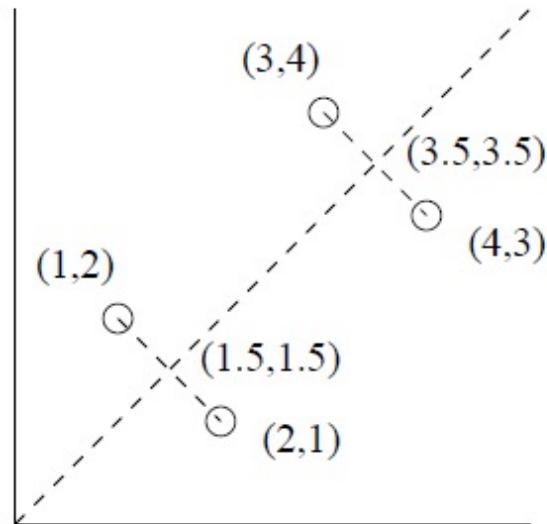
$$\begin{aligned} 30x + 28y &= 2x \\ 28x + 30y &= 2y \end{aligned}$$

Both equations tell us the same thing:  $x = -y$ . Thus, the unit eigenvector corresponding to the principal eigenvalue 2 is

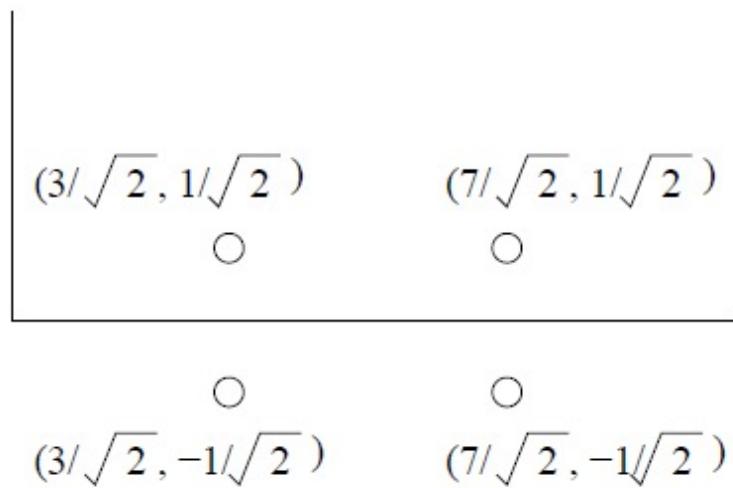
$$\begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$E = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

$$ME = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 3/\sqrt{2} & 1/\sqrt{2} \\ 3/\sqrt{2} & -1/\sqrt{2} \\ 7/\sqrt{2} & 1/\sqrt{2} \\ 7/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$



# New Coordinates



**Note to other teachers and users of these slides:** We would be delighted if you found this our material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: <http://www.mmds.org>

# Finding Similar Items: Locality Sensitive Hashing

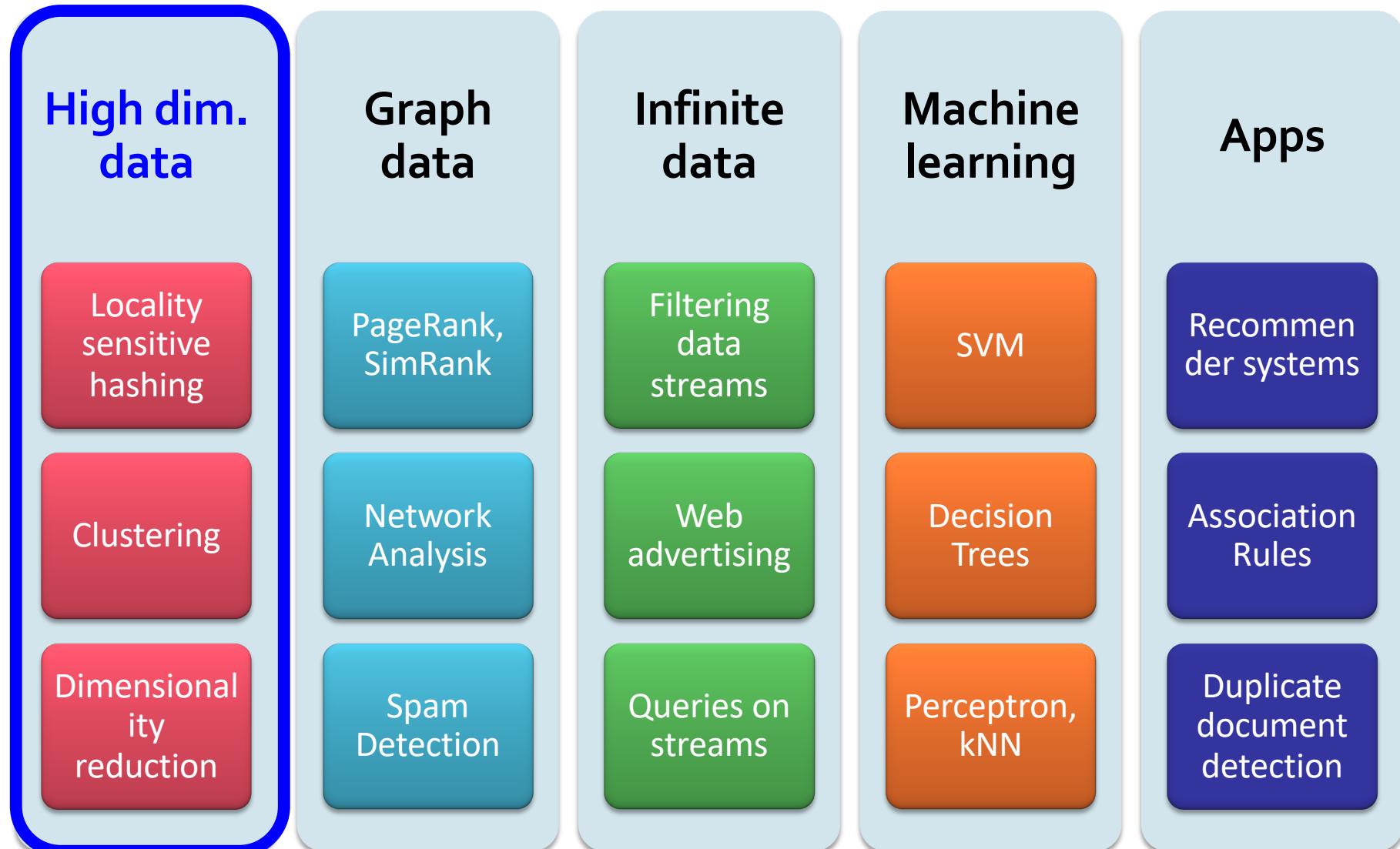
Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman  
Stanford University

<http://www.mmds.org>



# New thread: High dim. data



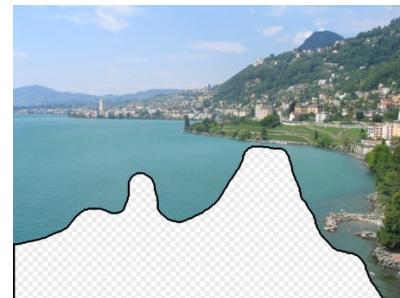
[Hays and Efros, SIGGRAPH 2007]

# Scene Completion Problem



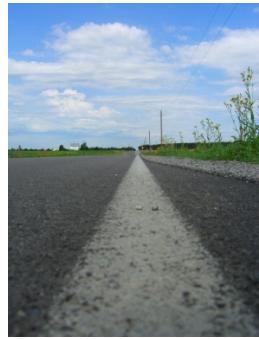
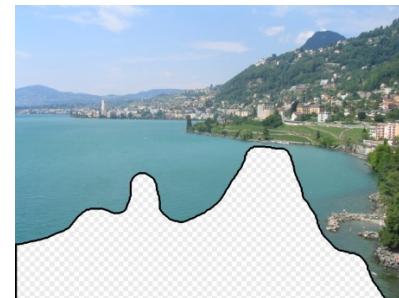
[Hays and Efros, SIGGRAPH 2007]

# Scene Completion Problem



[Hays and Efros, SIGGRAPH 2007]

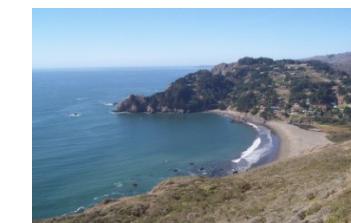
# Scene Completion Problem



10 nearest neighbors from a collection of 20,000 images

[Hays and Efros, SIGGRAPH 2007]

# Scene Completion Problem

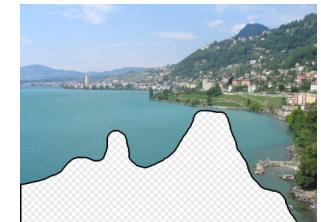


**10 nearest neighbors from a collection of 2 million images**

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, <http://www.mmmds.org>

# A Common Metaphor

- Many problems can be expressed as finding “similar” sets:
  - Find near-neighbors in high-dimensional space
- Examples:
  - Pages with similar words
    - For duplicate detection, classification by topic
  - Customers who purchased similar products
    - Products with similar customer sets
  - Images with similar features
    - Users who visited similar websites



# Problem for Today's Lecture

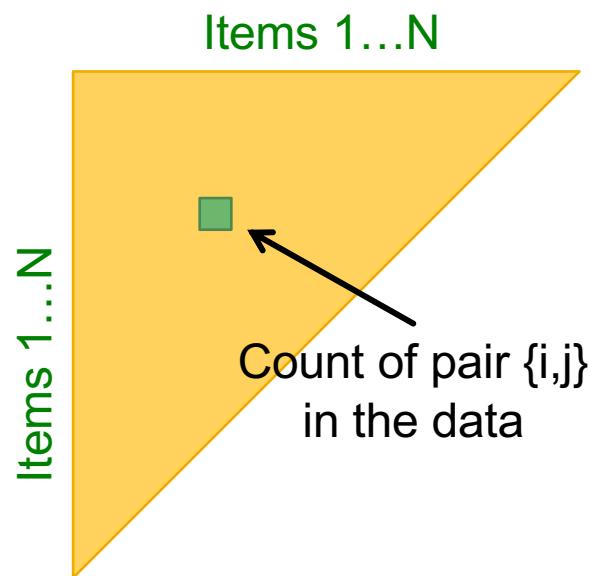
- Given: High dimensional data points  $x_1, x_2, \dots$ 
  - For example: Image is a long vector of pixel colors

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow [1 2 1 0 2 1 0 1 0]$$

- And some distance function  $d(x_1, x_2)$ 
  - Which quantifies the “distance” between  $x_1$  and  $x_2$
- Goal: Find all pairs of data points  $(x_i, x_j)$  that are within some distance threshold  $d(x_i, x_j) \leq s$
- Note: Naïve solution would take  $O(N^2)$  ☹  
where  $N$  is the number of data points
- MAGIC: This can be done in  $O(N)$ !! How?

# Relation to Previous Lecture

- **Last time:** Finding frequent pairs

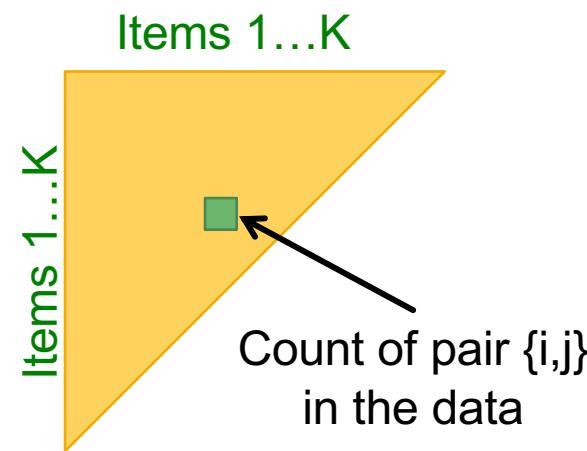


## Naïve solution:

Single pass but requires space quadratic in the number of items

N ... number of distinct items

K ... number of items with support  $\geq s$



## A-Priori:

First pass: Find frequent singletons

For a pair to be a **frequent pair candidate**, its singletons have to be frequent!

Second pass:

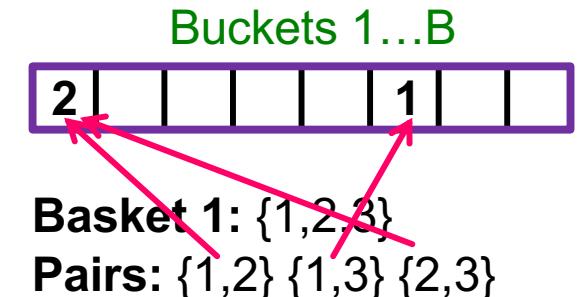
**Count only candidate pairs!**

# Relation to Previous Lecture

- **Last time:** Finding frequent pairs
- **Further improvement:** PCY

- **Pass 1:**

- Count exact frequency of each item:
- Take pairs of items  $\{i,j\}$ , hash them into  $B$  buckets and count of the number of pairs that hashed to each bucket:



# Relation to Previous Lecture

- **Last time:** Finding frequent pairs
- **Further improvement:** PCY

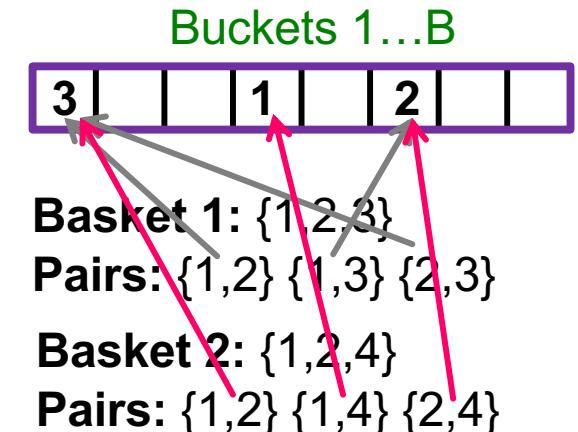
- **Pass 1:**

- Count exact frequency of each item:
- Take pairs of items  $\{i,j\}$ , hash them into  $B$  buckets and count of the number of pairs that hashed to each bucket:



- **Pass 2:**

- For a pair  $\{i,j\}$  to be a **candidate for a frequent pair**, its singletons  $\{i\}$ ,  $\{j\}$  have to be frequent and the pair has to hash to a frequent bucket!



# Relation to Previous Lecture

- Last time: Find pairs of similar documents
- Future: Find pairs of similar documents

## Previous lecture: A-Priori

### Main idea: Candidates

Instead of keeping a count of each pair, only keep a count of candidate pairs!

## Today's lecture: Find pairs of similar docs

### Main idea: Candidates

-- Pass 1: Take documents and hash them to buckets such that documents that are similar hash to the same bucket

-- Pass 2: Only compare documents that are **candidates** (i.e., they hashed to a same bucket)

Benefits: Instead of  $O(N^2)$  comparisons, we need  $O(N)$  comparisons to find similar documents

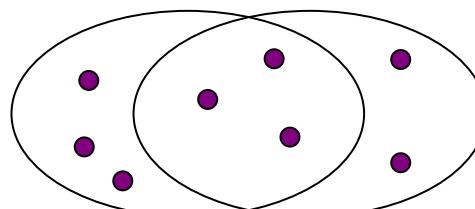
3}

{2,4}

# Finding Similar Items

# Distance Measures

- **Goal: Find near-neighbors in high-dim. space**
  - We formally define “near neighbors” as points that are a “small distance” apart
- For each application, we first need to define what “**distance**” means
- **Today: Jaccard distance/similarity**
  - The **Jaccard similarity** of two **sets** is the size of their intersection divided by the size of their union:  
 $sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$
  - **Jaccard distance:**  $d(C_1, C_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|$



3 in intersection  
8 in union  
Jaccard similarity= 3/8  
Jaccard distance = 5/8

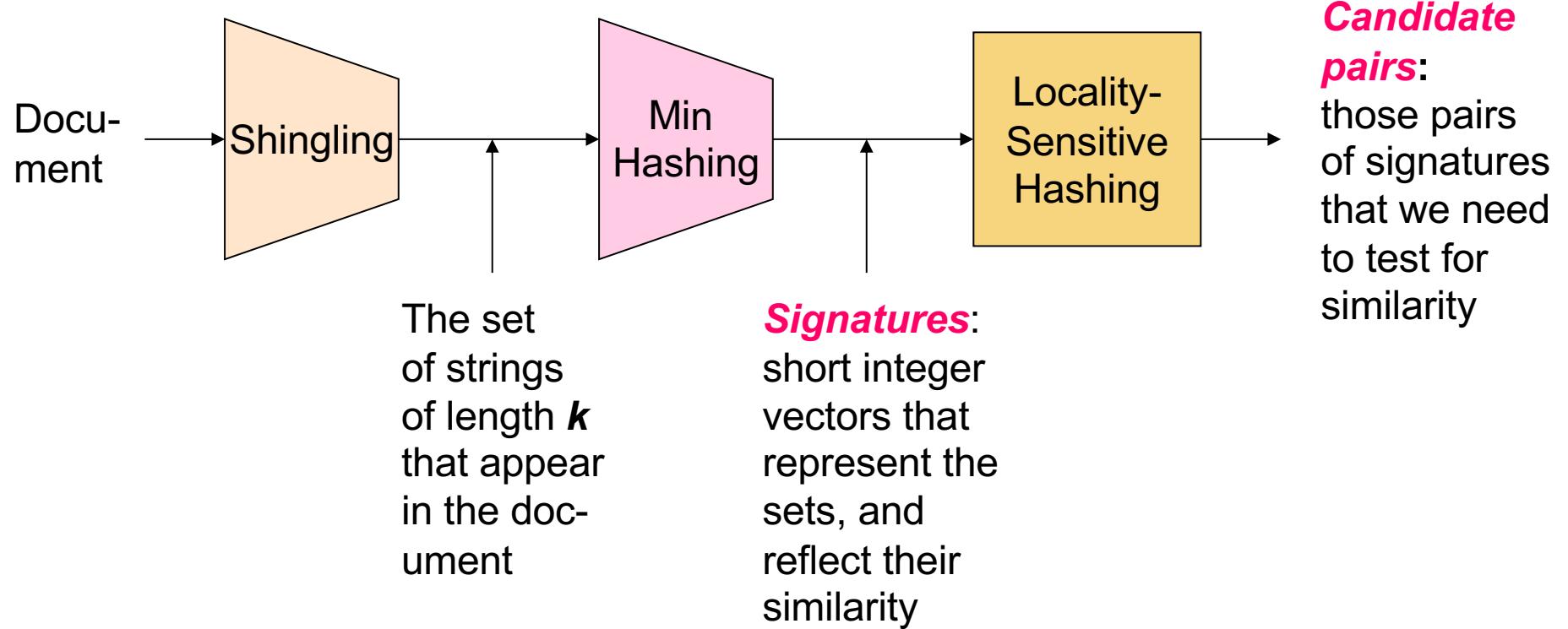
# Task: Finding Similar Documents

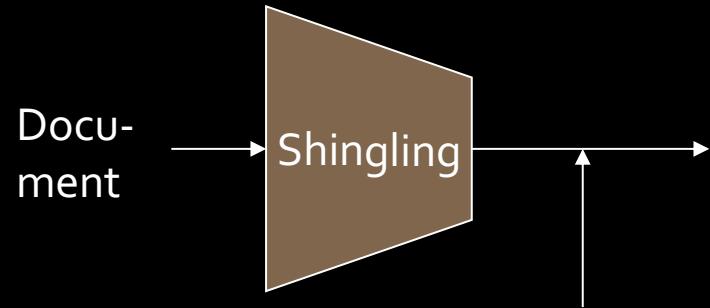
- **Goal:** Given a large number ( $N$  in the millions or billions) of documents, find “near duplicate” pairs
- **Applications:**
  - Mirror websites, or approximate mirrors
    - Don’t want to show both in search results
  - Similar news articles at many news sites
    - Cluster articles by “same story”
- **Problems:**
  - Many small pieces of one document can appear out of order in another
  - Too many documents to compare all pairs
  - Documents are so large or so many that they cannot fit in main memory

# 3 Essential Steps for Similar Docs

1. ***Shingling:*** Convert documents to sets
2. ***Min-Hashing:*** Convert large sets to short signatures, while preserving similarity
3. ***Locality-Sensitive Hashing:*** Focus on pairs of signatures likely to be from similar documents
  - **Candidate pairs!**

# The Big Picture





The set  
of strings  
of length  $k$   
that appear  
in the doc-  
ument

# Shingling

Step 1: *Shingling*: Convert documents to sets

# Documents as High-Dim Data

- Step 1: *Shingling*: Convert documents to sets
- Simple approaches:
  - Document = set of words appearing in document
  - Document = set of “important” words
  - Don’t work well for this application. Why?
- Need to account for ordering of words!
- A different way: *Shingles*!

# Define: Shingles

- A *k-shingle* (or *k-gram*) for a document is a sequence of  $k$  tokens that appears in the doc
  - Tokens can be *characters*, *words* or something else, depending on the application
  - Assume tokens = characters for examples
- **Example:**  $k=2$ ; document  $D_1 = \text{abcab}$   
Set of 2-shingles:  $S(D_1) = \{\text{ab}, \text{bc}, \text{ca}\}$ 
  - **Option:** Shingles as a bag (multiset), count ab twice:  $S'(D_1) = \{\text{ab}, \text{bc}, \text{ca}, \text{ab}\}$

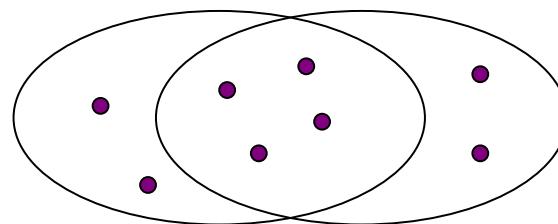
# Compressing Shingles

- To **compress long shingles**, we can **hash** them to (say) 4 bytes
- **Represent a document by the set of hash values of its  $k$ -shingles**
  - **Idea:** Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared
- **Example:**  $k=2$ ; document  $D_1 = \text{abcab}$   
Set of 2-shingles:  $S(D_1) = \{\text{ab}, \text{bc}, \text{ca}\}$   
Hash the singles:  $h(D_1) = \{1, 5, 7\}$

# Similarity Metric for Shingles

- Document  $D_1$  is a set of its  $k$ -shingles  $C_1 = S(D_1)$
- Equivalently, each document is a 0/1 vector in the space of  $k$ -shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse
- A natural similarity measure is the **Jaccard similarity**:

$$sim(D_1, D_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$

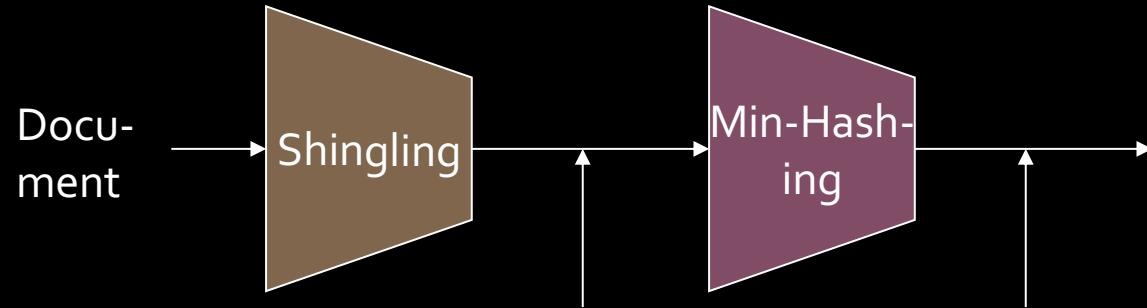


# Working Assumption

- **Documents that have lots of shingles in common have similar text, even if the text appears in different order**
- **Caveat:** You must pick  $k$  large enough, or most documents will have most shingles
  - $k = 5$  is OK for short documents
  - $k = 10$  is better for long documents

# Motivation for Minhash/LSH

- Suppose we need to find near-duplicate documents among  $N = 1$  million documents
- Naïvely, we would have to compute pairwise Jaccard similarities for every pair of docs
  - $N(N - 1)/2 \approx 5 * 10^{11}$  comparisons
  - At  $10^5$  secs/day and  $10^6$  comparisons/sec, it would take 5 days
- For  $N = 10$  million, it takes more than a year...



The set  
of strings  
of length  $k$   
that appear  
in the doc-  
ument

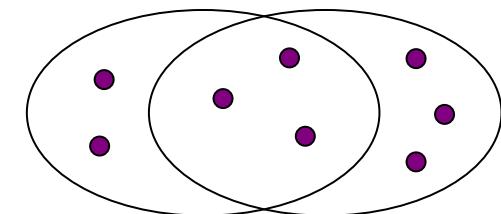
***Signatures:***  
short integer  
vectors that  
represent the  
sets, and  
reflect their  
similarity

## MinHashing

Step 2: ***Minhashing:*** Convert large sets to  
short signatures, while preserving similarity

# Encoding Sets as Bit Vectors

- Many similarity problems can be formalized as **finding subsets that have significant intersection**
- **Encode sets using 0/1 (bit, boolean) vectors**
  - One dimension per element in the universal set
- Interpret **set intersection as bitwise AND**, and **set union as bitwise OR**
- **Example:**  $C_1 = 10111$ ;  $C_2 = 10011$ 
  - Size of intersection = 3; size of union = 4,
  - **Jaccard similarity** (not distance) =  $3/4$
  - **Distance:**  $d(C_1, C_2) = 1 - (\text{Jaccard similarity}) = 1/4$



# From Sets to Boolean Matrices

- **Rows** = elements (shingles)
- **Columns** = sets (documents)
  - 1 in row  $e$  and column  $s$  if and only if  $e$  is a member of  $s$
  - Column similarity is the Jaccard similarity of the corresponding sets (rows with value 1)
  - **Typical matrix is sparse!**
- **Each document is a column:**
  - **Example:**  $\text{sim}(C_1, C_2) = ?$ 
    - Size of intersection = 3; size of union = 6, Jaccard similarity (not distance) = 3/6
    - $d(C_1, C_2) = 1 - (\text{Jaccard similarity}) = 3/6$

	Documents			
Shingles	1	1	1	0
1	1	0	1	
0	1	0	1	
0	0	0	1	
1	0	0	1	
1	1	1	0	
1	0	1	0	

# Outline: Finding Similar Columns

- **So far:**
  - Documents → Sets of shingles
  - Represent sets as boolean vectors in a matrix
- **Next goal: Find similar columns while computing small signatures**
  - **Similarity of columns == similarity of signatures**

# Outline: Finding Similar Columns

- **Next Goal: Find similar columns, Small signatures**
- **Naïve approach:**
  - **1) Signatures of columns:** small summaries of columns
  - **2) Examine pairs of signatures** to find similar columns
    - **Essential:** Similarities of signatures and columns are related
  - **3) Optional:** Check that columns with similar signatures are really similar
- **Warnings:**
  - Comparing all pairs may take too much time: **Job for LSH**
    - These methods can produce false negatives, and even false positives (if the optional check is not made)

# Hashing Columns (Signatures)

- **Key idea:** “hash” each column  $C$  to a small *signature*  $h(C)$ , such that:
  - (1)  $h(C)$  is small enough that the signature fits in RAM
  - (2)  $\text{sim}(C_1, C_2)$  is the same as the “similarity” of signatures  $h(C_1)$  and  $h(C_2)$

- **Goal:** Find a hash function  $h(\cdot)$  such that:
  - If  $\text{sim}(C_1, C_2)$  is high, then with high prob.  $h(C_1) = h(C_2)$
  - If  $\text{sim}(C_1, C_2)$  is low, then with high prob.  $h(C_1) \neq h(C_2)$

- **Hash docs into buckets. Expect that “most” pairs of near duplicate docs hash into the same bucket!**

# Min-Hashing

- **Goal: Find a hash function  $h(\cdot)$  such that:**
  - if  $\text{sim}(C_1, C_2)$  is high, then with high prob.  $h(C_1) = h(C_2)$
  - if  $\text{sim}(C_1, C_2)$  is low, then with high prob.  $h(C_1) \neq h(C_2)$
- **Clearly, the hash function depends on the similarity metric:**
  - Not all similarity metrics have a suitable hash function
- **There is a suitable hash function for the Jaccard similarity:** It is called **Min-Hashing**

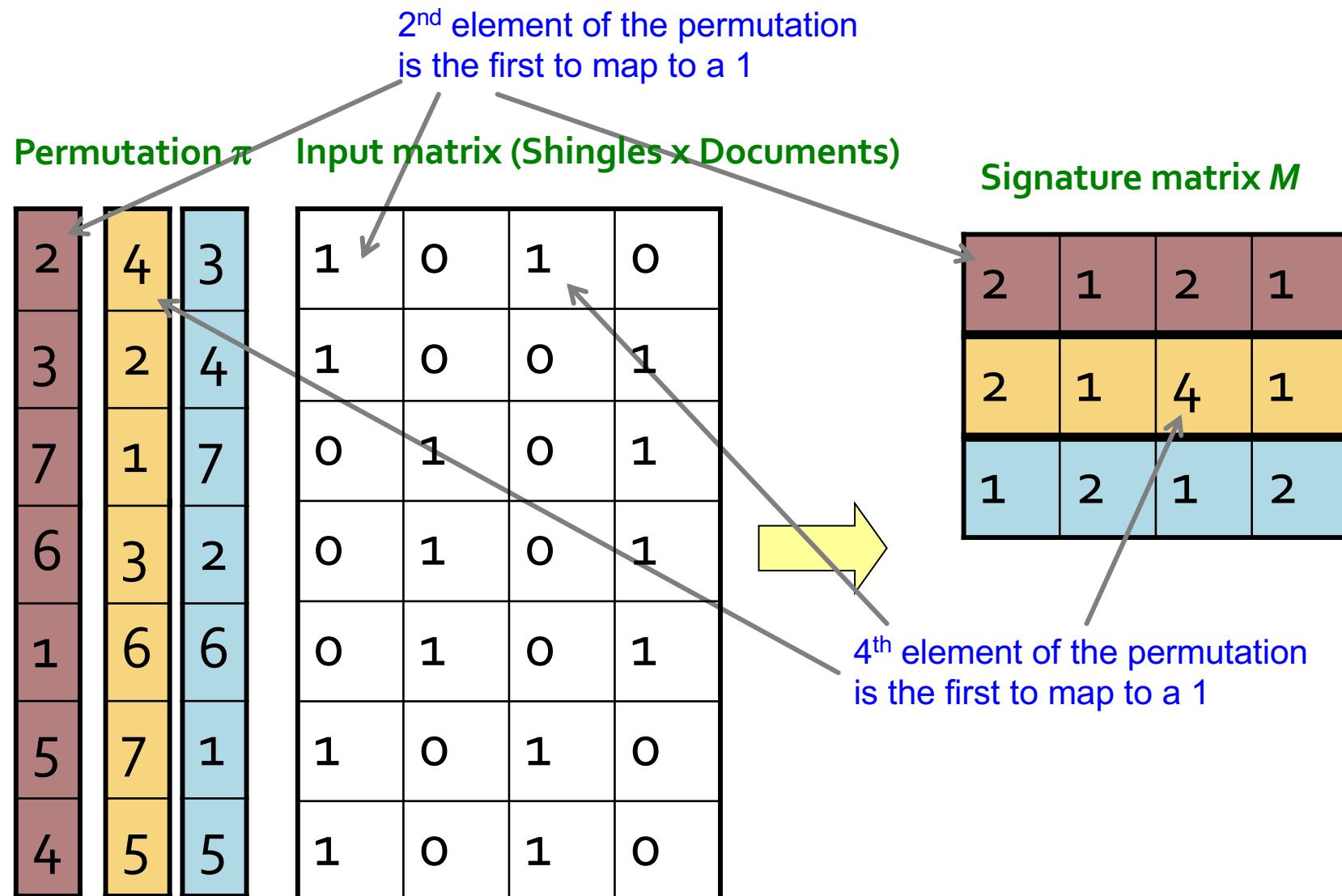
# Min-Hashing

- Imagine the rows of the boolean matrix permuted under **random permutation**  $\pi$
- Define a “**hash**” function  $h_\pi(C)$  = the index of the **first** (in the permuted order  $\pi$ ) row in which column  $C$  has value 1:  
$$h_\pi(C) = \min_\pi \pi(C)$$
- Use several (e.g., 100) independent hash functions (that is, permutations) to create a signature of a column

# Min-Hashing Example

Note: Another (equivalent) way is to store row indexes:

1	5	1	5
2	3	1	3
6	4	6	4



# The Min-Hash Property

0	0
0	0
<b>1</b>	<b>1</b>
0	0
0	1
1	0

- Choose a random permutation  $\pi$
- Claim:  $\Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$
- Why?
  - Let  $X$  be a doc (set of shingles),  $y \in X$  is a shingle
  - Then:  $\Pr[\pi(y) = \min(\pi(X))] = 1/|X|$ 
    - It is equally likely that any  $y \in X$  is mapped to the **min** element
  - Let  $y$  be s.t.  $\pi(y) = \min(\pi(C_1 \cup C_2))$
  - Then either:  $\pi(y) = \min(\pi(C_1))$  if  $y \in C_1$ , or  
 $\pi(y) = \min(\pi(C_2))$  if  $y \in C_2$   
One of the two  
cols had to have  
1 at position  $y$
  - So the prob. that both are true is the prob.  $y \in C_1 \cap C_2$
  - $\Pr[\min(\pi(C_1)) = \min(\pi(C_2))] = |C_1 \cap C_2| / |C_1 \cup C_2| = \text{sim}(C_1, C_2)$

# Similarity for Signatures

- We know:  $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- Now generalize to multiple hash functions
- The *similarity of two signatures* is the fraction of the hash functions in which they agree
- **Note:** Because of the Min-Hash property, the similarity of columns is the same as the expected similarity of their signatures

# Min-Hashing Example

Permutation  $\pi$

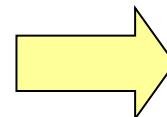
2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Input matrix (Shingles x Documents)

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix  $M$

2	1	2	1
2	1	4	1
1	2	1	2



Similarities:

Col/Col  
Sig/Sig

1-3	2-4	1-2	3-4
0.75	0.75	0	0
0.67	1.00	0	0

# Min-Hash Signatures

- Pick K=100 random permutations of the rows
- Think of  $\text{sig}(C)$  as a column vector
- $\text{sig}(C)[i]$  = according to the  $i$ -th permutation, the index of the first row that has a 1 in column C
$$\text{sig}(C)[i] = \min (\pi_i(C))$$
- Note: The sketch (signature) of document C is small ~100 bytes!
- We achieved our goal! We “compressed” long bit vectors into short signatures

# Implementation Trick

- **Permuting rows even once is prohibitive**
- **Row hashing!**
  - Pick  $K = 100$  hash functions  $k_i$
  - Ordering under  $k_i$  gives a random row permutation!
- **One-pass implementation**
  - For each column  $C$  and hash-func.  $k_i$ , keep a “slot” for the min-hash value
  - Initialize all  $\text{sig}(C)[i] = \infty$
  - **Scan rows looking for 1s**
    - Suppose row  $j$  has 1 in column  $C$
    - Then for each  $k_i$ :
      - If  $k_i(j) < \text{sig}(C)[i]$ , then  $\text{sig}(C)[i] \leftarrow k_i(j)$

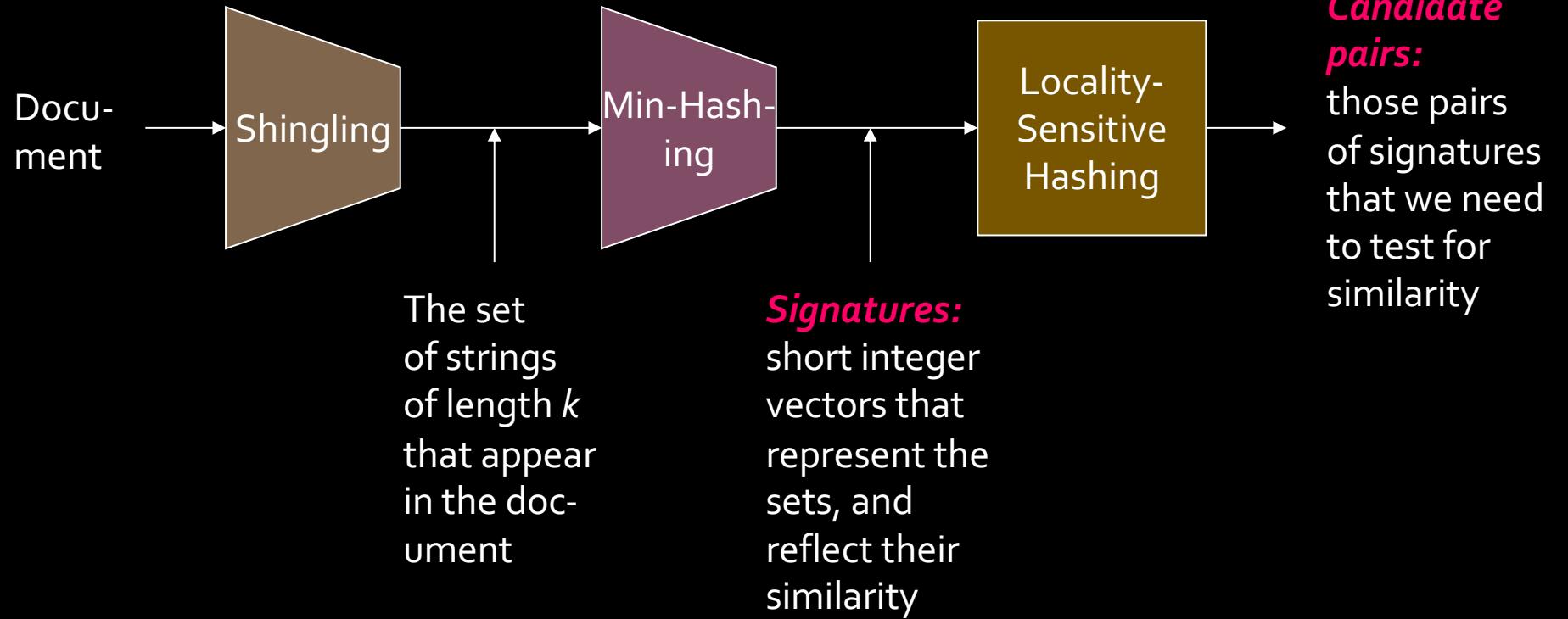
**How to pick a random hash function  $h(x)$ ?**

**Universal hashing:**

$h_{a,b}(x) = ((a \cdot x + b) \bmod p) \bmod N$   
where:

a, b ... random integers

p ... prime number ( $p > N$ )



## Locality Sensitive Hashing

**Step 3: *Locality-Sensitive Hashing:***

Focus on pairs of signatures likely to be from similar documents

# LSH: First Cut

2	1	4	1
1	2	1	2
2	1	2	1

- **Goal:** Find documents with Jaccard similarity at least  $s$  (for some similarity threshold, e.g.,  $s=0.8$ )
- **LSH – General idea:** Use a function  $f(x,y)$  that tells whether  $x$  and  $y$  is a *candidate pair*: a pair of elements whose similarity must be evaluated
- **For Min-Hash matrices:**
  - Hash columns of *signature matrix  $M$*  to many buckets
  - Each pair of documents that hashes into the same bucket is a *candidate pair*

# Candidates from Min-Hash

2	1	4	1
1	2	1	2
2	1	2	1

- Pick a similarity threshold  $s$  ( $0 < s < 1$ )
- Columns  $x$  and  $y$  of  $M$  are a **candidate pair** if their signatures agree on at least fraction  $s$  of their rows:  
 $M(i, x) = M(i, y)$  for at least frac.  $s$  values of  $i$ 
  - We expect documents  $x$  and  $y$  to have the same (Jaccard) similarity as their signatures

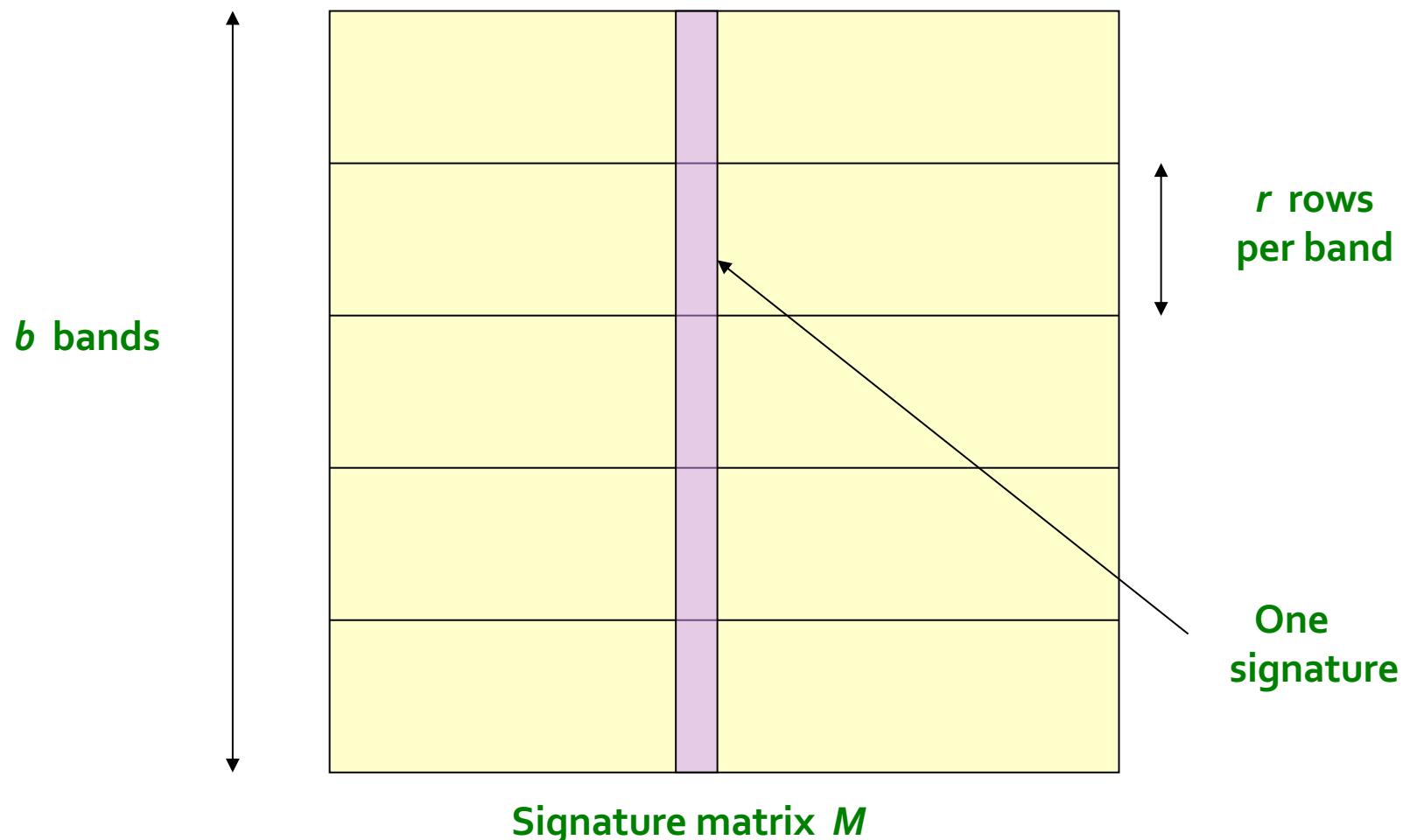
# LSH for Min-Hash

2	1	4	1
1	2	1	2
2	1	2	1

- **Big idea: Hash columns of signature matrix  $M$  several times**
- Arrange that (only) similar columns are likely to **hash to the same bucket**, with high probability
- **Candidate pairs are those that hash to the same bucket**

# Partition $M$ into $b$ Bands

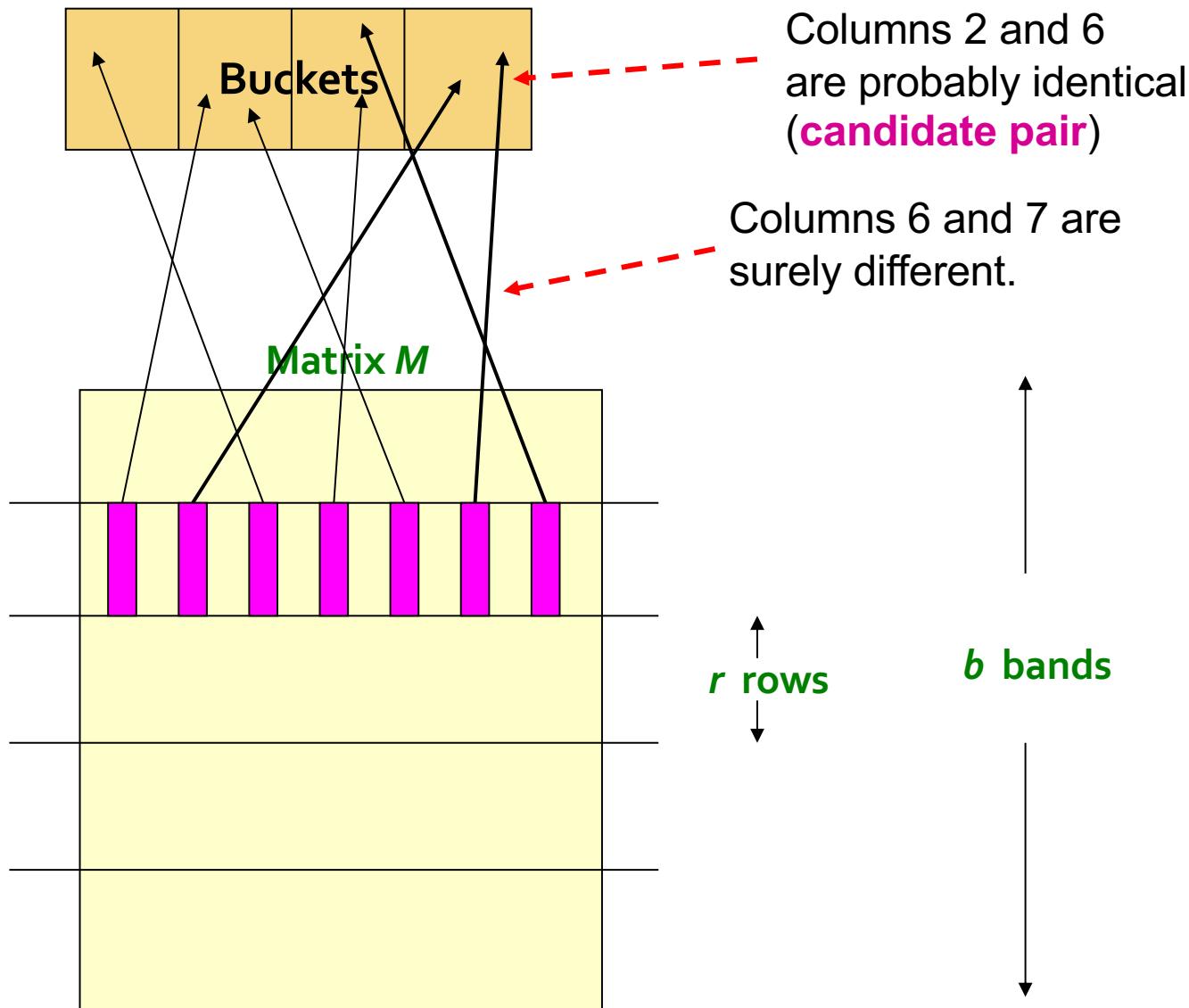
2	1	4	1
1	2	1	2
2	1	2	1



# Partition $M$ into Bands

- Divide matrix  $M$  into  $b$  bands of  $r$  rows
- For each band, hash its portion of each column to a hash table with  $k$  buckets
  - Make  $k$  as large as possible
- ***Candidate*** column pairs are those that hash to the same bucket for  $\geq 1$  band
- Tune  $b$  and  $r$  to catch most similar pairs, but few non-similar pairs

# Hashing Bands



# Simplifying Assumption

- There are **enough buckets** that columns are unlikely to hash to the same bucket unless they are **identical** in a particular band
- Hereafter, we assume that “**same bucket**” means “**identical in that band**”
- Assumption needed only to simplify analysis, not for correctness of algorithm

# Example of Bands

2	1	4	1
1	2	1	2
2	1	2	1

Assume the following case:

- Suppose 100,000 columns of  $M$  (100k docs)
- Signatures of 100 integers (rows)
- Therefore, signatures take 40Mb
- Choose  $b = 20$  bands of  $r = 5$  integers/band
- **Goal:** Find pairs of documents that are at least  $s = 0.8$  similar

# $C_1, C_2$ are 80% Similar

2	1	4	1
1	2	1	2
2	1	2	1

- Find pairs of  $\geq s=0.8$  similarity, set  $b=20, r=5$
- Assume:  $\text{sim}(C_1, C_2) = 0.8$ 
  - Since  $\text{sim}(C_1, C_2) \geq s$ , we want  $C_1, C_2$  to be a candidate pair: We want them to hash to at least 1 common bucket (at least one band is identical)
- Probability  $C_1, C_2$  identical in one particular band:  $(0.8)^5 = 0.328$
- Probability  $C_1, C_2$  are not similar in all of the 20 bands:  $(1-0.328)^{20} = 0.00035$ 
  - i.e., about 1/3000th of the 80%-similar column pairs are false negatives (we miss them)
- We would find 99.965% pairs of truly similar documents

# $C_1, C_2$ are 30% Similar

2	1	4	1
1	2	1	2
2	1	2	1

- Find pairs of  $\geq s=0.8$  similarity, set  $b=20, r=5$
- Assume:  $\text{sim}(C_1, C_2) = 0.3$ 
  - Since  $\text{sim}(C_1, C_2) < s$  we want  $C_1, C_2$  to hash to **NO common buckets** (all bands should be different)
- Probability  $C_1, C_2$  identical in one particular band:  $(0.3)^5 = 0.00243$
- Probability  $C_1, C_2$  identical in at least 1 of 20 bands:  $1 - (1 - 0.00243)^{20} = 0.0474$ 
  - In other words, approximately 4.74% pairs of docs with similarity 0.3% end up becoming **candidate pairs**
    - They are **false positives** since we will have to examine them (they are candidate pairs) but then it will turn out their similarity is below threshold  $s$

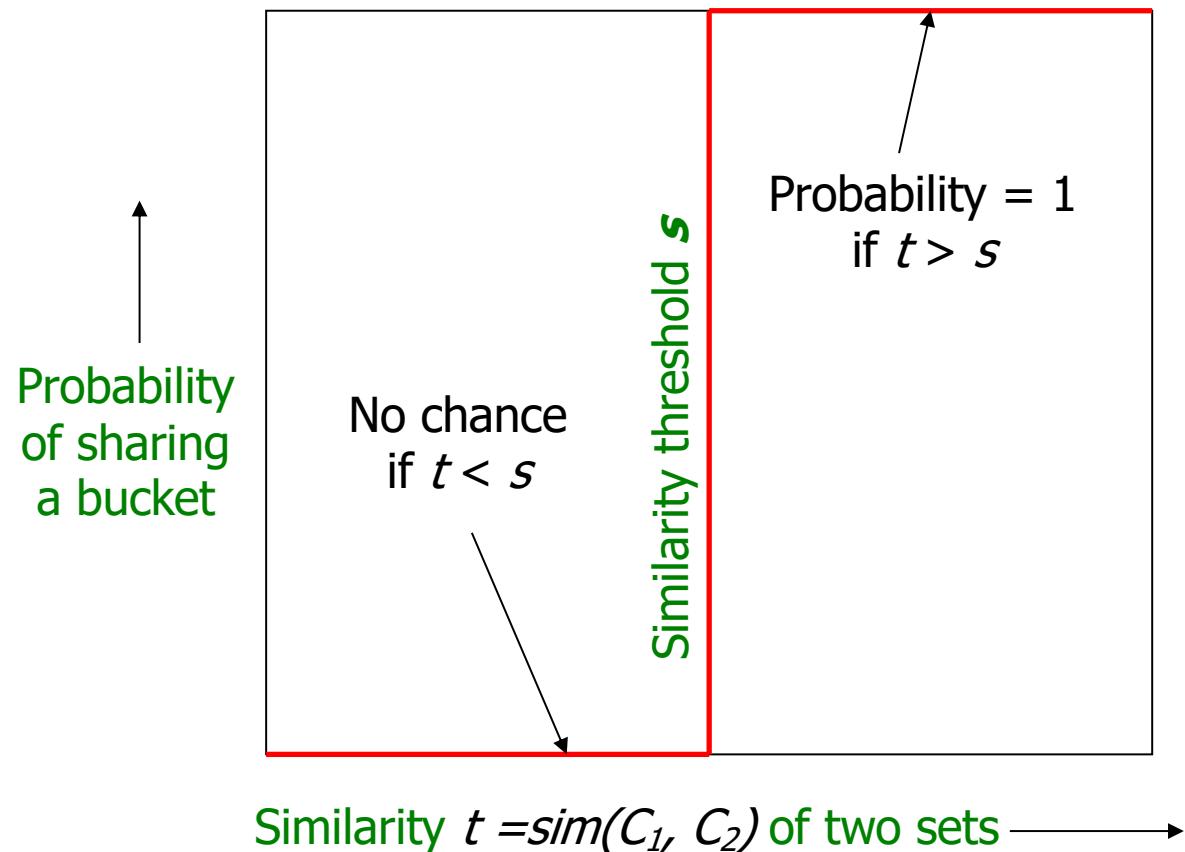
# LSH Involves a Tradeoff

2	1	4	1
1	2	1	2
2	1	2	1

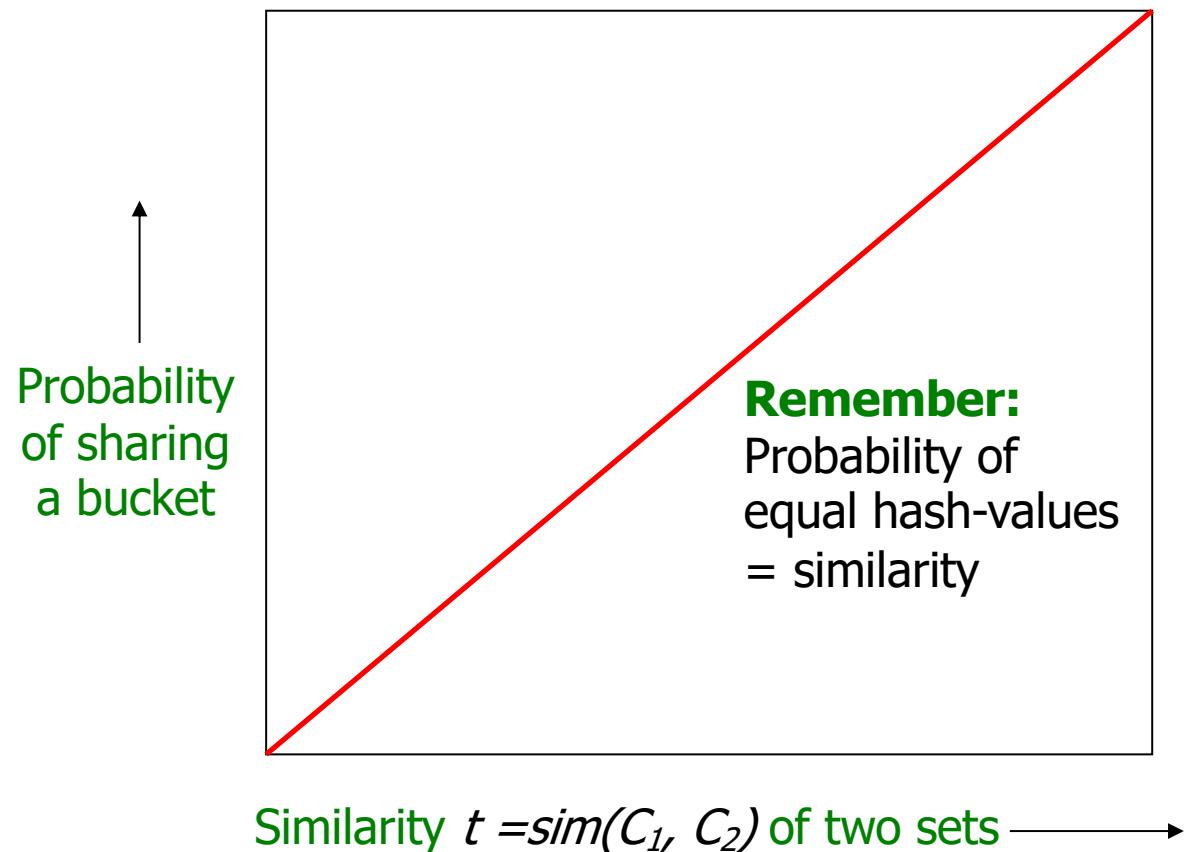
- **Pick:**

- The number of Min-Hashes (rows of  $M$ )
- The number of bands  $b$ , and
- The number of rows  $r$  per band  
to balance false positives/negatives
- **Example:** If we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up

# Analysis of LSH – What We Want



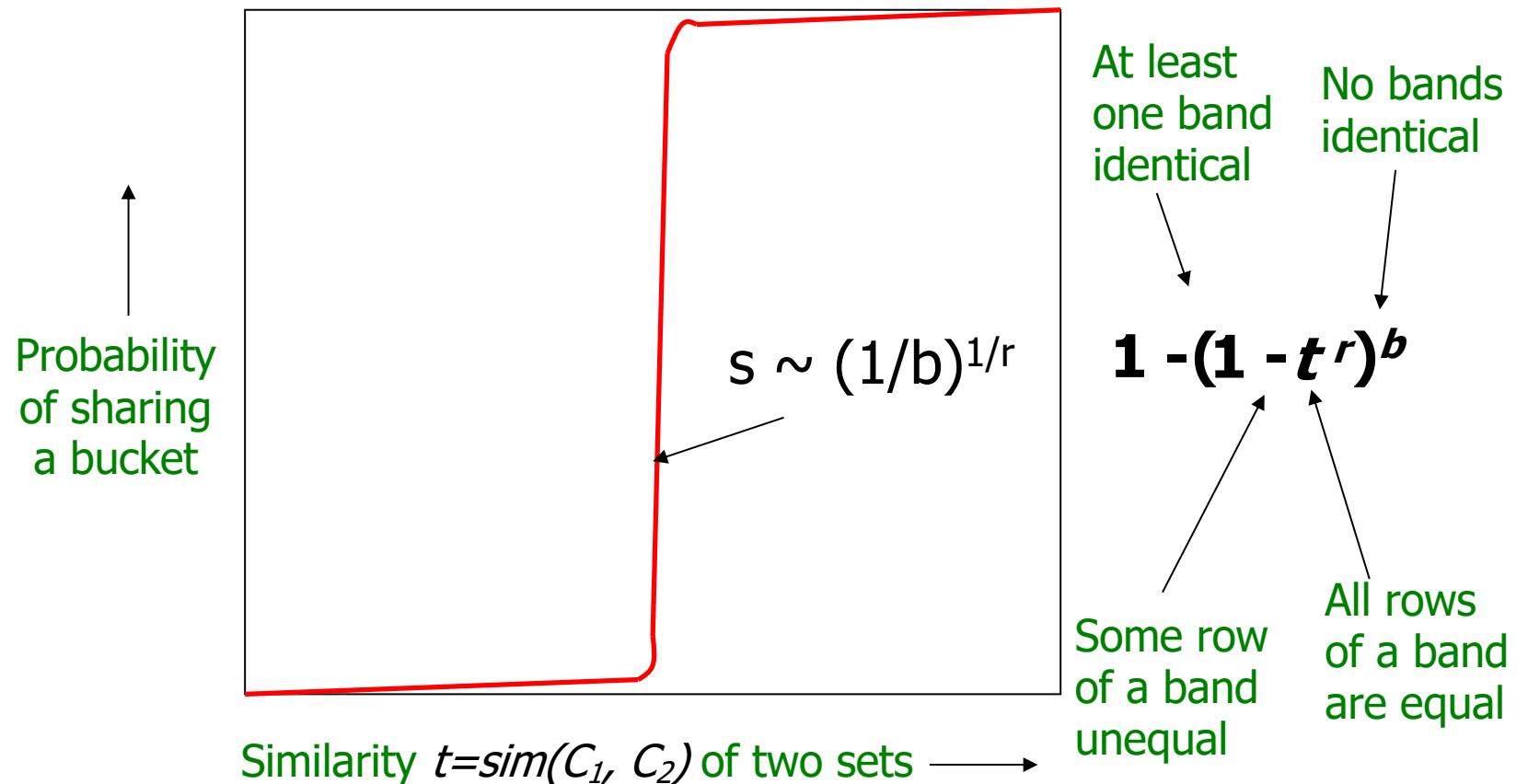
# What 1 Band of 1 Row Gives You



# ***b bands, r rows/band***

- Columns  $C_1$  and  $C_2$  have similarity  $t$
- Pick any band ( $r$  rows)
  - Prob. that all rows in band equal =  $t^r$
  - Prob. that some row in band unequal =  $1 - t^r$
- Prob. that no band identical =  $(1 - t^r)^b$
- Prob. that at least 1 band identical =  
$$1 - (1 - t^r)^b$$

# What $b$ Bands of $r$ Rows Gives You



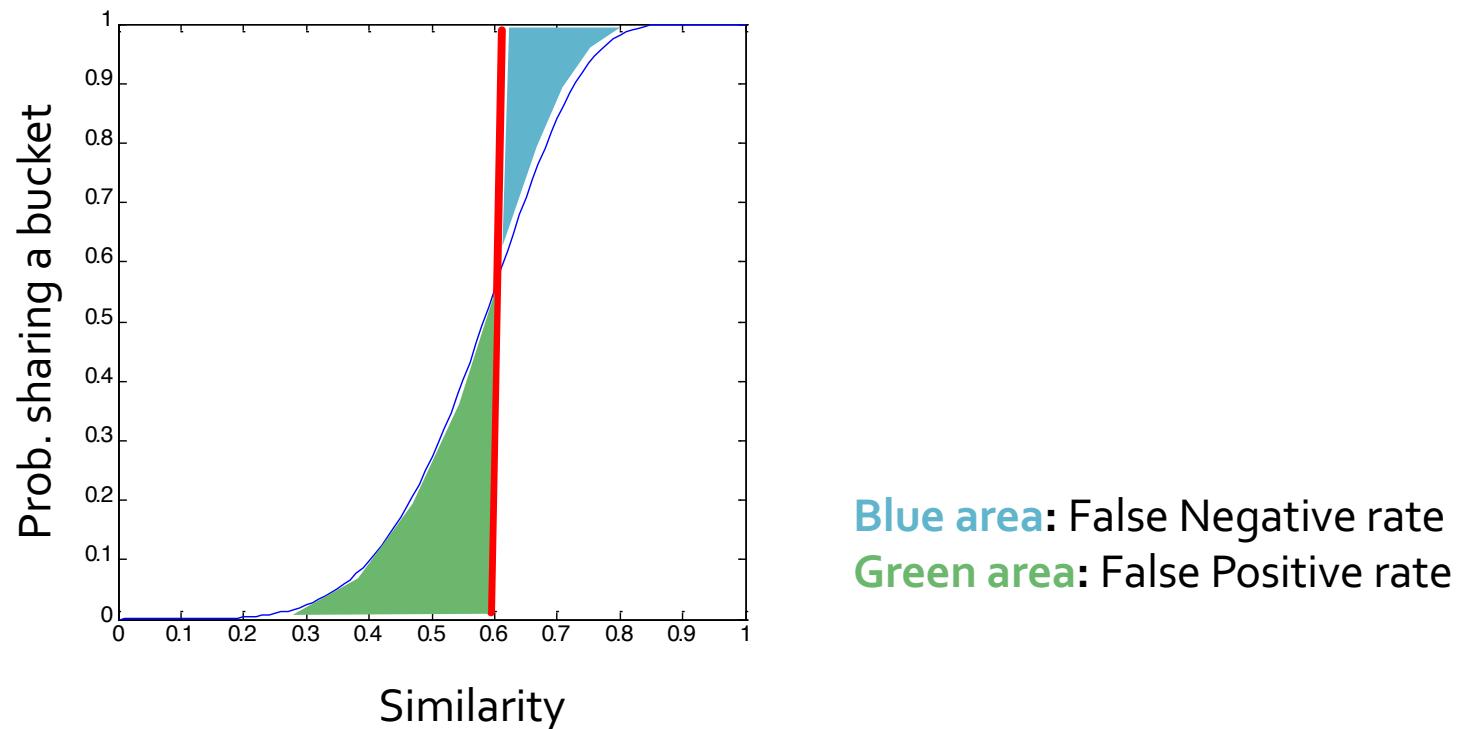
# Example: $b = 20$ ; $r = 5$

- Similarity threshold  $s$
- Prob. that at least 1 band is identical:

$s$	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

# Picking $r$ and $b$ : The S-curve

- Picking  $r$  and  $b$  to get the best S-curve
  - 50 hash-functions ( $r=5$ ,  $b=10$ )



# LSH Summary

- Tune  $M$ ,  $b$ ,  $r$  to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures
- Check in main memory that **candidate pairs** really do have **similar signatures**
- **Optional:** In another pass through data, check that the remaining candidate pairs really represent similar documents

# Summary: 3 Steps

- **Shingling:** Convert documents to sets
  - We used hashing to assign each shingle an ID
- **Min-Hashing:** Convert large sets to short signatures, while preserving similarity
  - We used **similarity preserving hashing** to generate signatures with property  $\Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$
  - We used hashing to get around generating random permutations
- **Locality-Sensitive Hashing:** Focus on pairs of signatures likely to be from similar documents
  - We used hashing to find **candidate pairs** of similarity  $\geq s$

As the next set of slides are not proper so its better to check the lectures in the drive and listen to the last 4 lectures which are:

- 1) Distance Measure - 1 video
- 2) Mining data streams - 3 videos

**Note to other teachers and users of these slides:** We would be delighted if you found this our material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: <http://www.mmds.org>

# Clustering

Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman  
Stanford University

<http://www.mmds.org>



# High Dimensional Data

## High dim. data

Locality  
sensitive  
hashing

Clustering

Dimensional  
ity  
reduction

## Graph data

PageRank,  
SimRank

Community  
Detection

Spam  
Detection

## Infinite data

Filtering  
data  
streams

Web  
advertising

Queries on  
streams

## Machine learning

SVM

Decision  
Trees

Perceptron,  
kNN

## Apps

Recommen  
der systems

Association  
Rules

Duplicate  
document  
detection

# High Dimensional Data

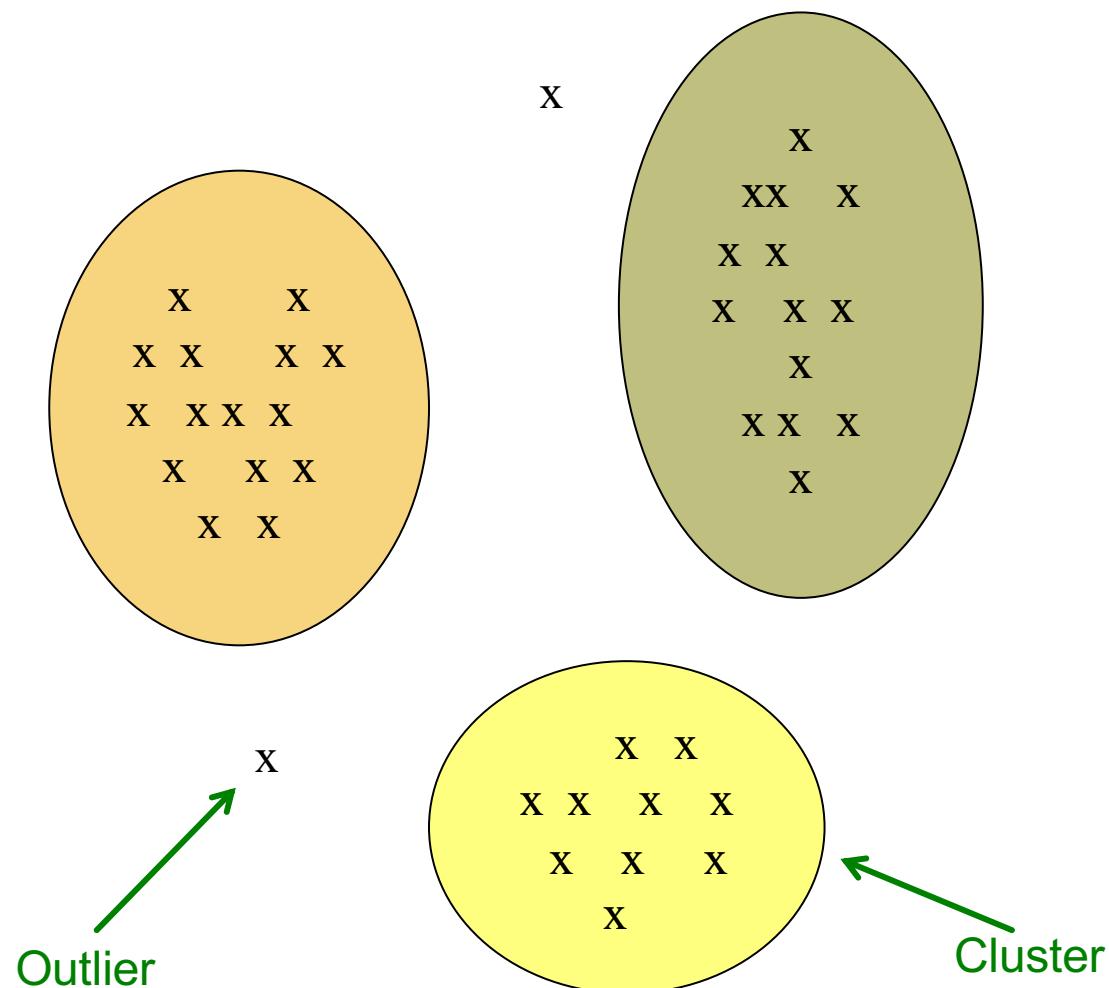
- Given a cloud of data points we want to understand its structure



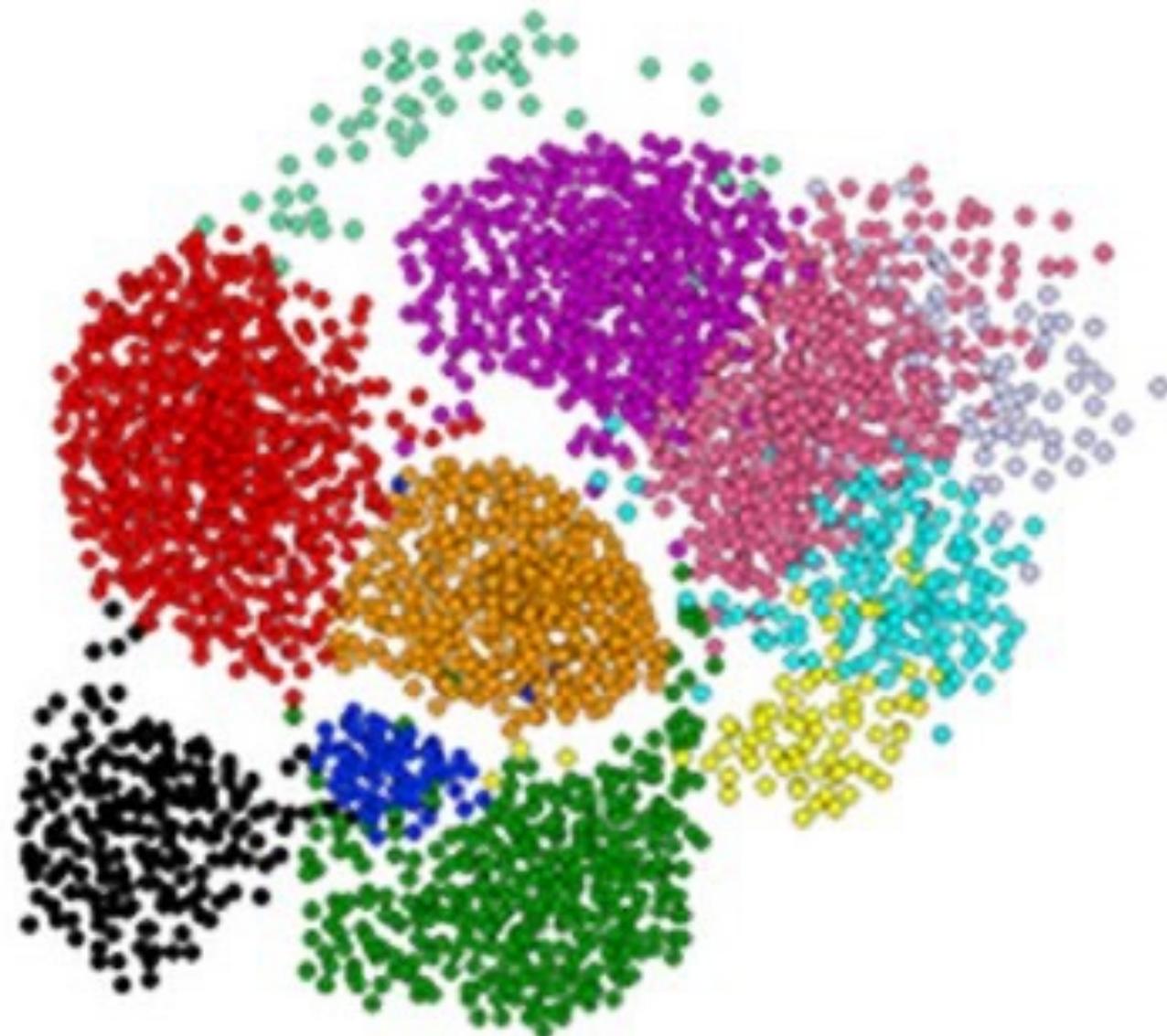
# The Problem of Clustering

- Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***, so that
  - Members of a cluster are close/similar to each other
  - Members of different clusters are dissimilar
- **Usually:**
  - Points are in a high-dimensional space
  - Similarity is defined using a distance measure
    - Euclidean, Cosine, Jaccard, edit distance, ...

# Example: Clusters & Outliers



# Clustering is a hard problem!



# Why is it hard?

- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy
- And in most cases, looks are **not** deceiving
  
- Many applications involve not 2, but 10 or 10,000 dimensions
- **High-dimensional spaces look different:**  
Almost all pairs of points are at about the same distance

# Clustering Problem: Galaxies

- A catalog of 2 billion “sky objects” represents objects by their radiation in 7 dimensions (frequency bands)
- Problem: Cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.
- Sloan Digital Sky Survey



# Clustering Problem: Music CDs

- **Intuitively:** Music divides into categories, and customers prefer a few categories
  - But what are categories really?
- Represent a CD by a set of customers who bought it:
- Similar CDs have similar sets of customers, and vice-versa

# Clustering Problem: Documents

## Finding topics:

- Represent a document by a vector  $(x_1, x_2, \dots, x_k)$ , where  $x_i = 1$  iff the  $i^{\text{th}}$  word (in some order) appears in the document
  - It actually doesn't matter if  $k$  is infinite; i.e., we don't limit the set of words
- **Documents with similar sets of words may be about the same topic**

# Cosine, Jaccard, and Euclidean

- As with CDs we have a choice when we think of documents as sets of words or shingles:
  - Sets as vectors: Measure similarity by the cosine distance
  - Sets as sets: Measure similarity by the Jaccard distance
  - Sets as points: Measure similarity by Euclidean distance

# Overview: Methods of Clustering

## ■ Hierarchical:

### ■ Agglomerative (bottom up):

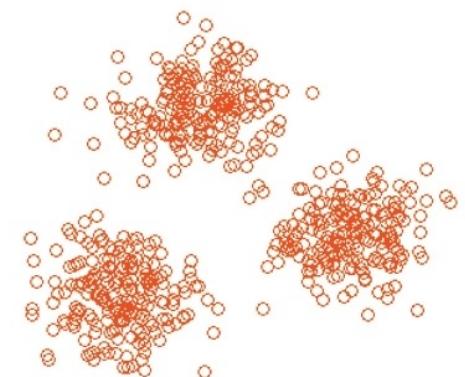
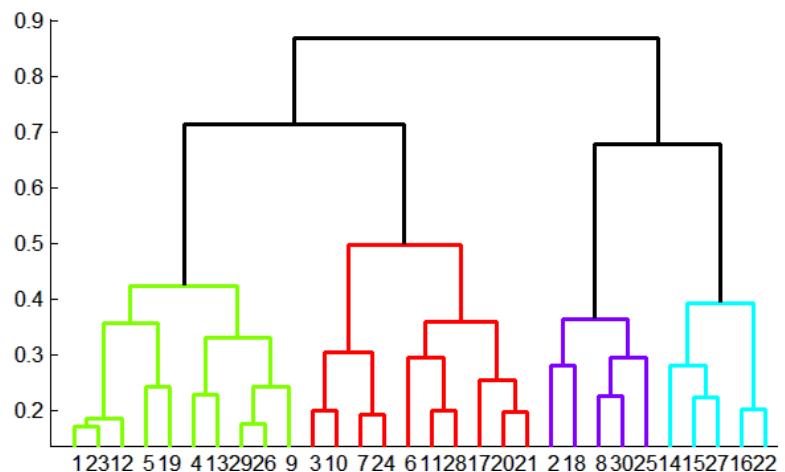
- Initially, each point is a cluster
- Repeatedly combine the two “nearest” clusters into one

### ■ Divisive (top down):

- Start with one cluster and recursively split it

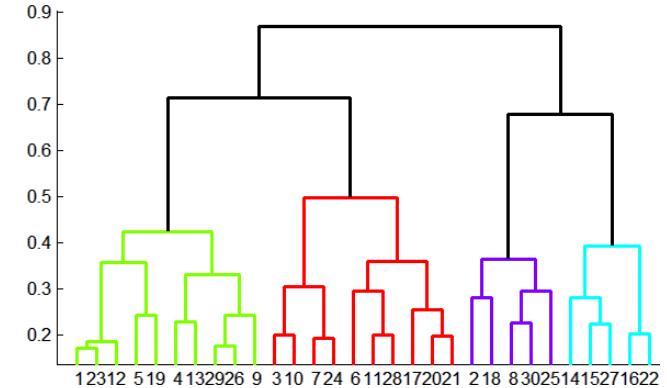
## ■ Point assignment:

- Maintain a set of clusters
- Points belong to “nearest” cluster



# Hierarchical Clustering

- **Key operation:**  
**Repeatedly combine two nearest clusters**

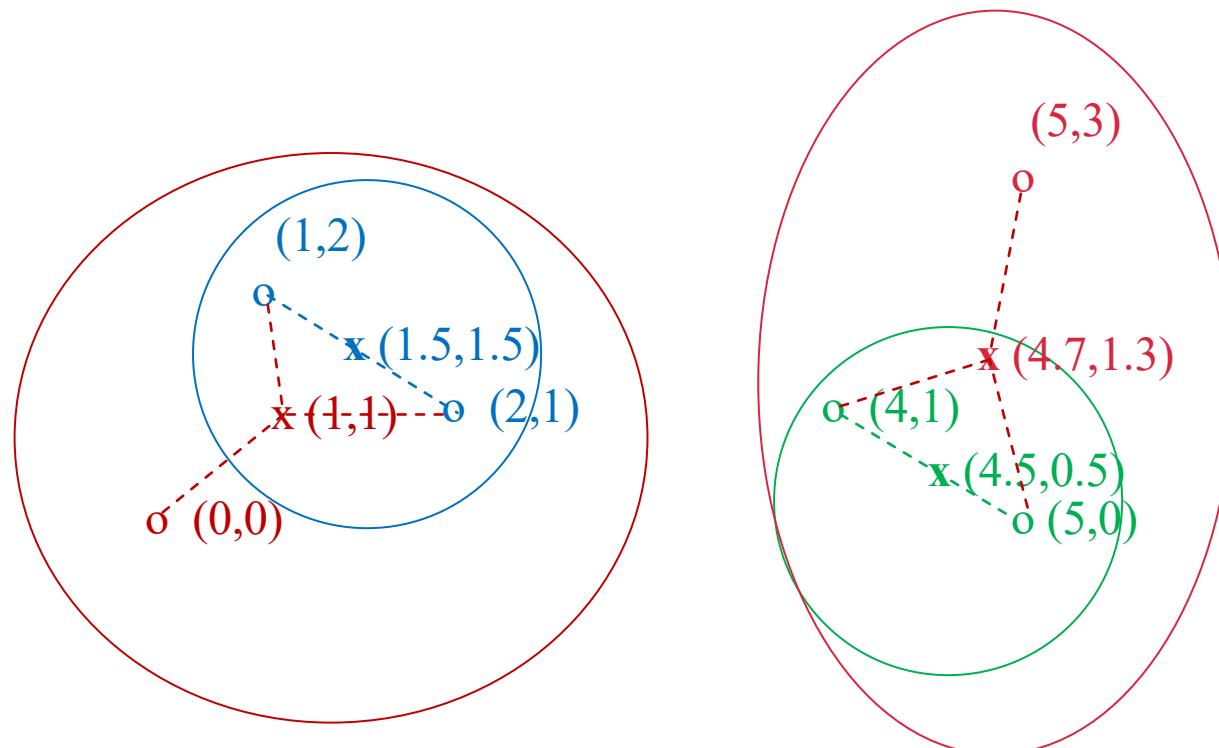


- **Three important questions:**
  - 1) How do you represent a cluster of more than one point?
  - 2) How do you determine the “nearness” of clusters?
  - 3) When to stop combining clusters?

# Hierarchical Clustering

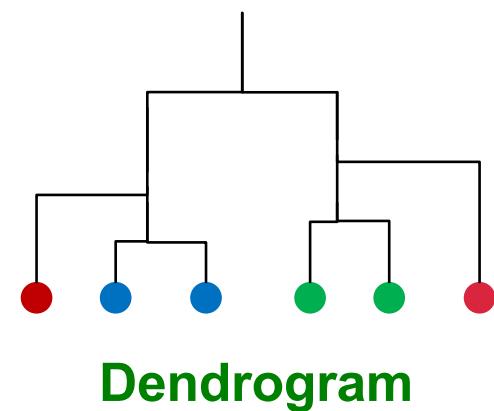
- **Key operation:** Repeatedly combine two nearest clusters
- **(1) How to represent a cluster of many points?**
  - **Key problem:** As you merge clusters, how do you represent the “location” of each cluster, to tell which pair of clusters is closest?
- **Euclidean case:** each cluster has a *centroid* = average of its (data)points
- **(2) How to determine “nearness” of clusters?**
  - Measure cluster distances by distances of centroids

# Example: Hierarchical clustering

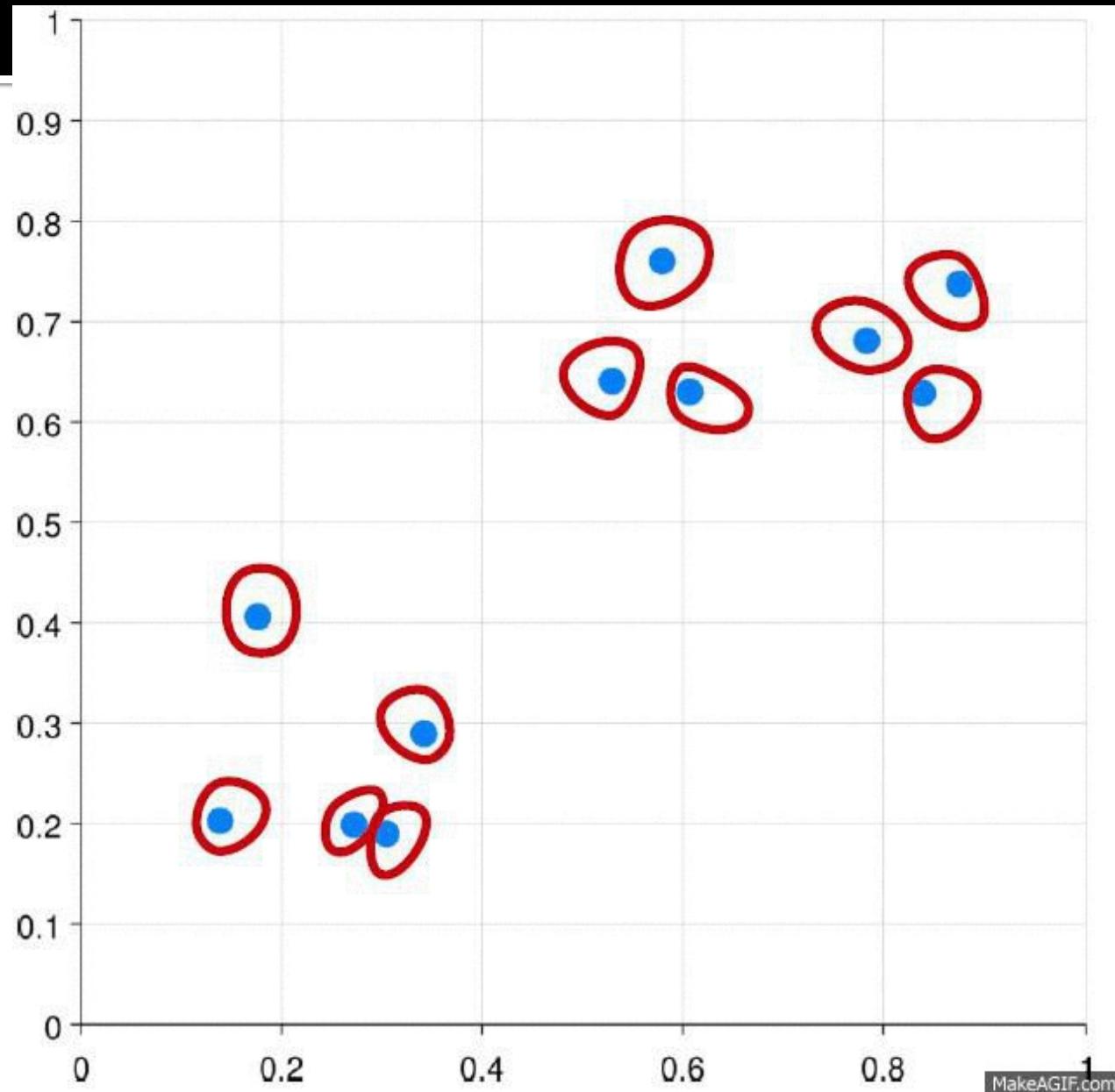


## Data:

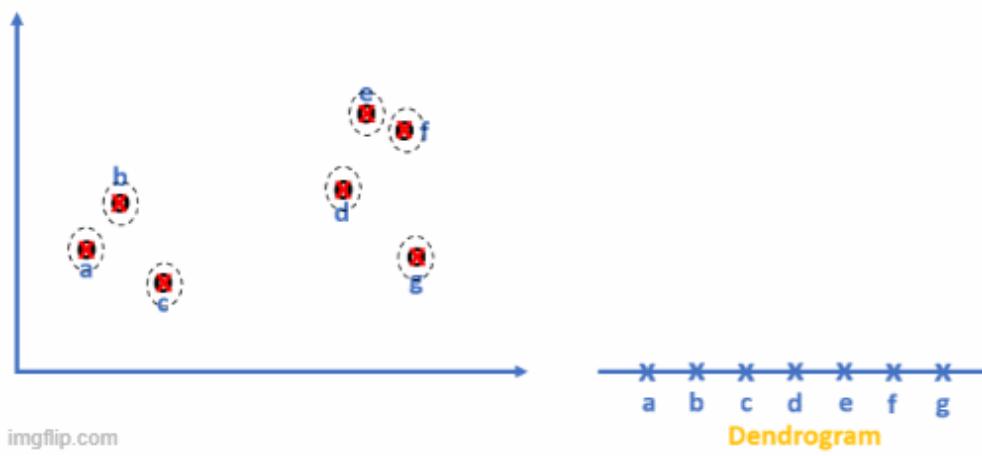
o ... data point  
x ... centroid



Dendrogram



MakeAGIF.com



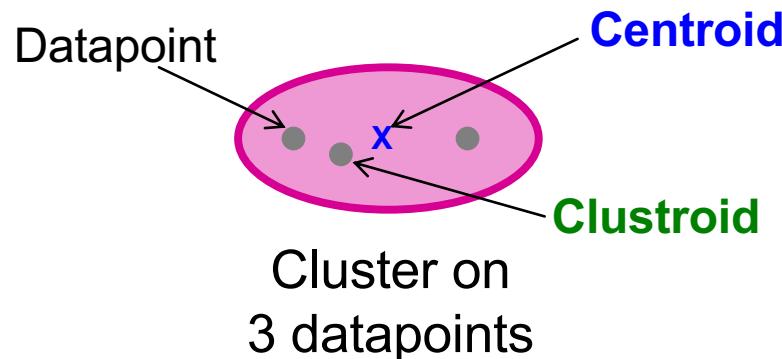
# And in the Non-Euclidean Case?

## What about the Non-Euclidean case?

- The only “locations” we can talk about are the points themselves
  - i.e., there is no “average” of two points
- Approach 1:
  - (1) How to represent a cluster of many points?  
*clustroid* = (data)point “closest” to other points
  - (2) How do you determine the “nearness” of clusters? Treat clustroid as if it were centroid, when computing inter-cluster distances

# “Closest” Point?

- (1) How to represent a cluster of many points?  
*clustroid* = point “closest” to other points
- Possible meanings of “closest”:
  - Smallest maximum distance to other points
  - Smallest average distance to other points
  - Smallest sum of squares of distances to other points
    - For distance metric  $d$  clustroid  $c$  of cluster  $C$  is:  $\min_c \sum_{x \in C} d(x, c)^2$



**Centroid** is the avg. of all (data)points in the cluster. This means centroid is an “artificial” point.

**Clustroid** is an **existing** (data)point that is “closest” to all other points in the cluster.

**Example 7.7:** Suppose we are using edit distance, and a cluster consists of the four points abcd, aecdb, abecb, and ecdab. Their distances are found in the following table:

	ecdab	abecb	aecdb
abcd	5	3	3
aecdb	2	2	
abecb	4		

If we apply the three criteria for being the centroid to each of the four points of the cluster, we find:

Point	Sum	Max	Sum-Sq
abcd	11	5	43
aecdb	7	3	17
abecb	9	4	29
ecdab	11	5	45

# Defining “Nearness” of Clusters

- (2) How do you determine the “nearness” of clusters?
  - Approach 1: K known
  - Approach 2:  
**Intercluster distance** = minimum of the distances between any two points, one from each cluster
  - Approach 3:  
Pick a notion of “**cohesion**” of clusters, e.g., maximum distance from the clustroid
    - Merge clusters whose **union** is most cohesive

# *k*-means clustering

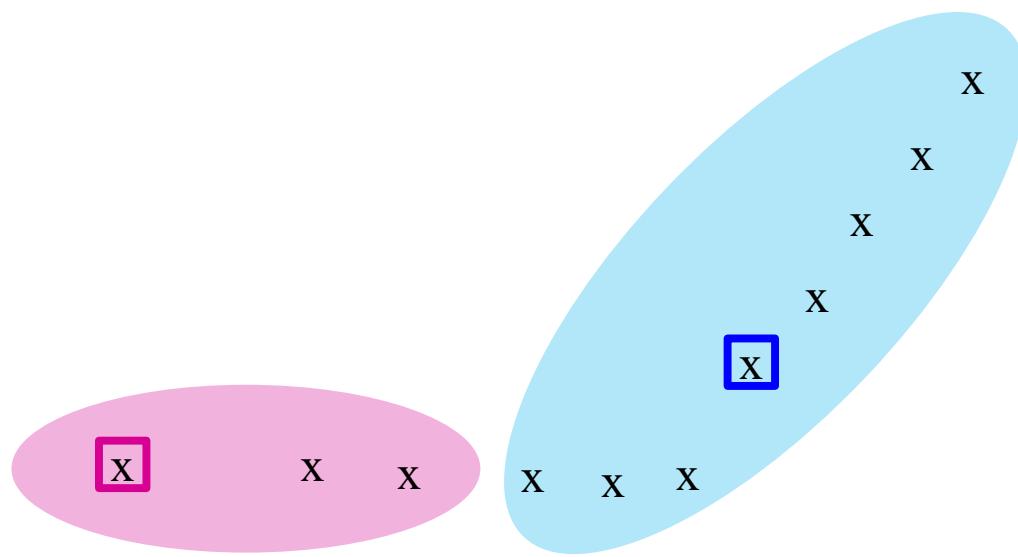
# **$k$ -means Algorithm(s)**

- Assumes Euclidean space/distance
- Start by picking  $k$ , the number of clusters
- Initialize clusters by picking one point per cluster
  - **Example:** Pick one point at random, then  $k-1$  other points, each as far away as possible from the previous points

# Populating Clusters

- 1) For each point, place it in the cluster whose current centroid it is nearest
- 2) After all points are assigned, update the locations of centroids of the  $k$  clusters
- 3) Reassign all points to their closest centroid
  - Sometimes moves points between clusters
- **Repeat 2 and 3 until convergence**
  - **Convergence:** Points don't move between clusters and centroids stabilize

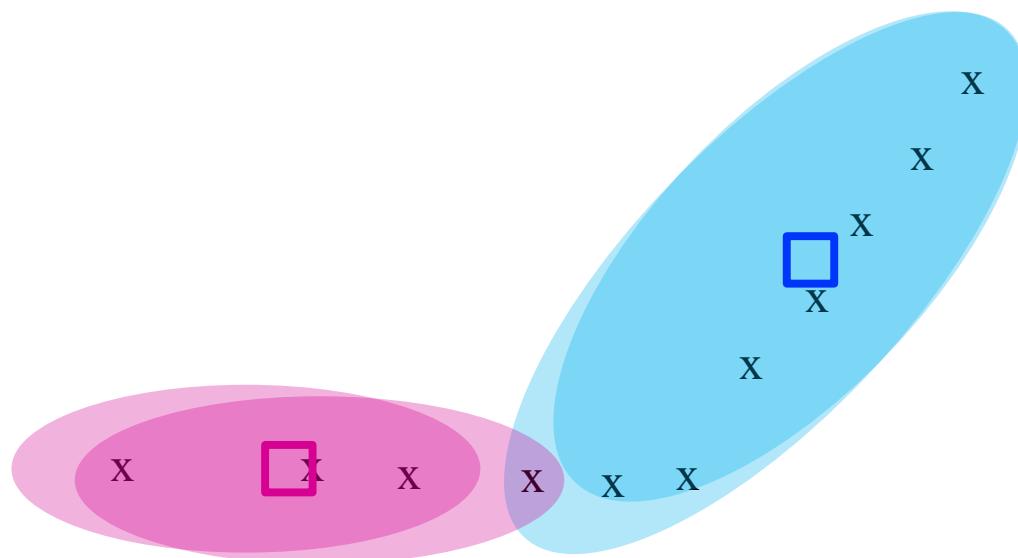
# Example: Assigning Clusters



$\text{X}$  ... data point  
□ ... centroid

**Clusters after round 1**

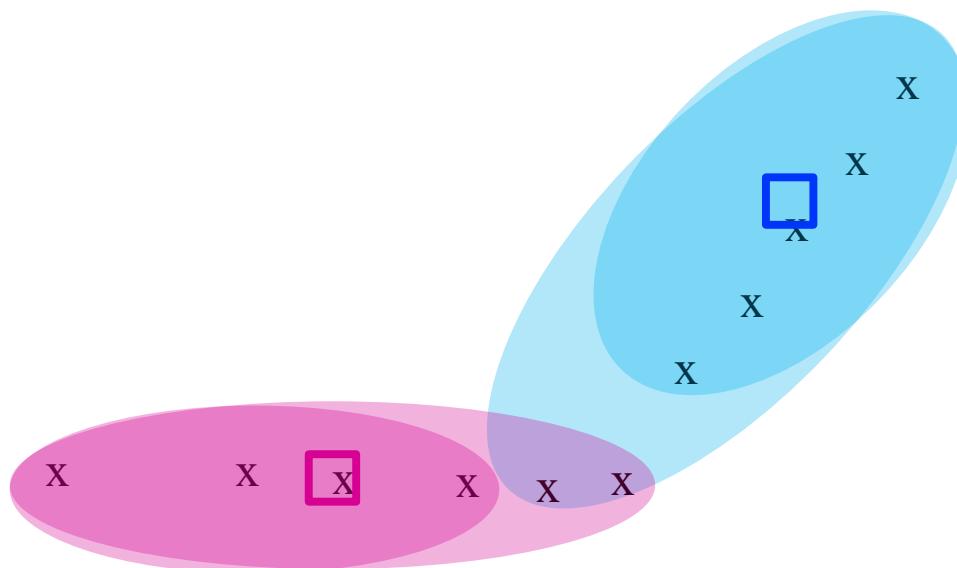
# Example: Assigning Clusters



$\text{X}$  ... data point  
 $\square$  ... centroid

**Clusters after round 2**

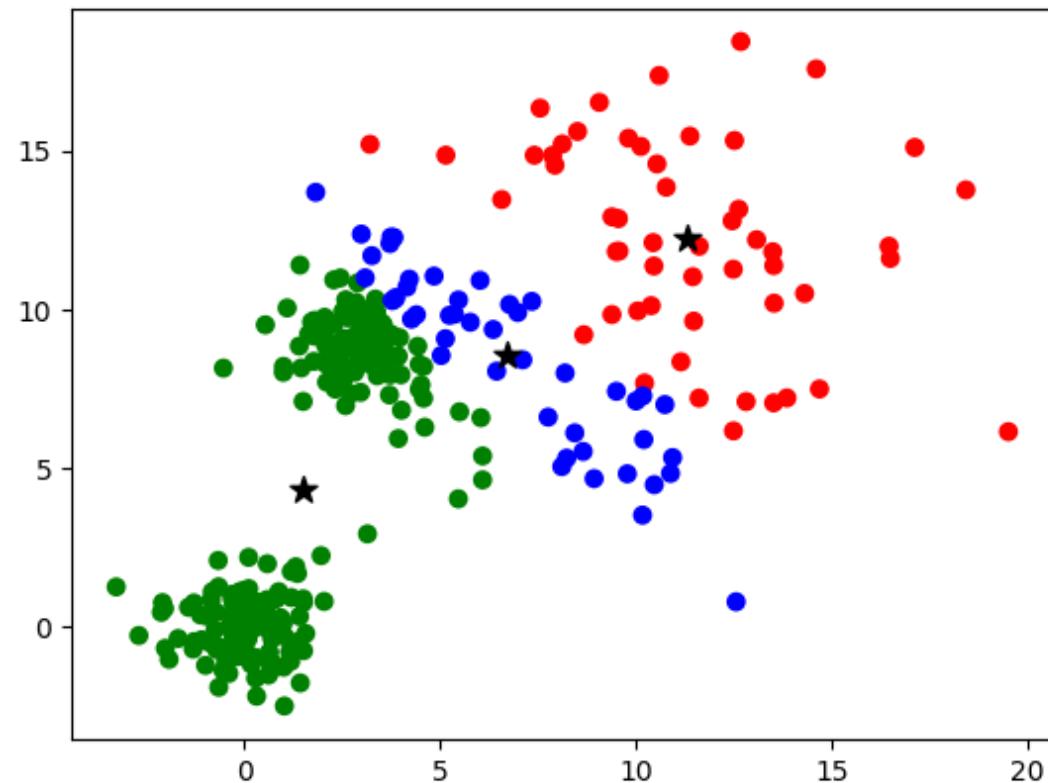
# Example: Assigning Clusters



$\text{X}$  ... data point  
 $\square$  ... centroid

**Clusters at the end**

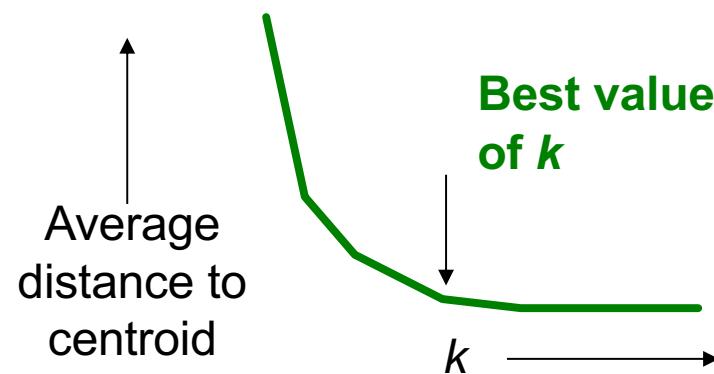
Kmeans Iteration 1



# Getting the $k$ right

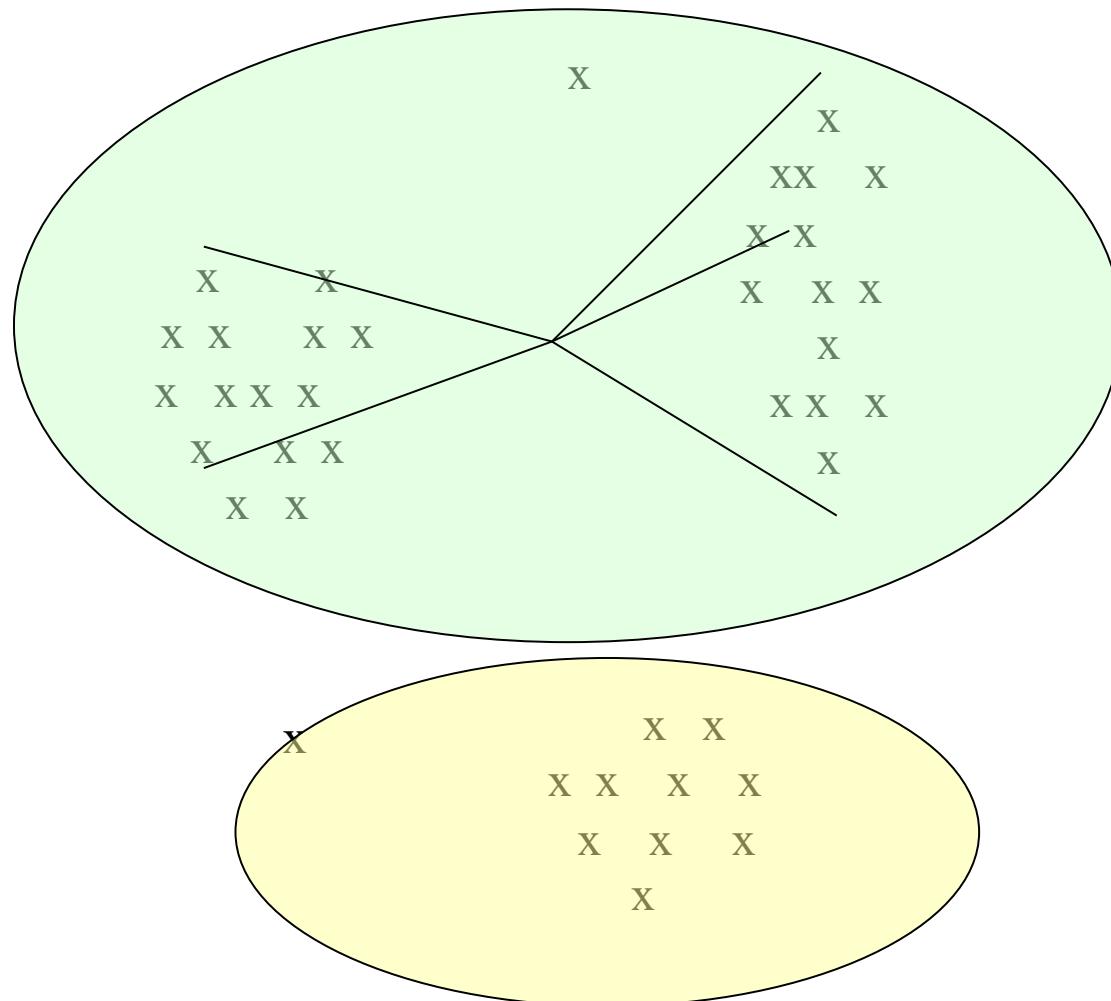
## How to select $k$ ?

- Try different  $k$ , looking at the change in the average distance to centroid as  $k$  increases
- Average falls rapidly until right  $k$ , then changes little



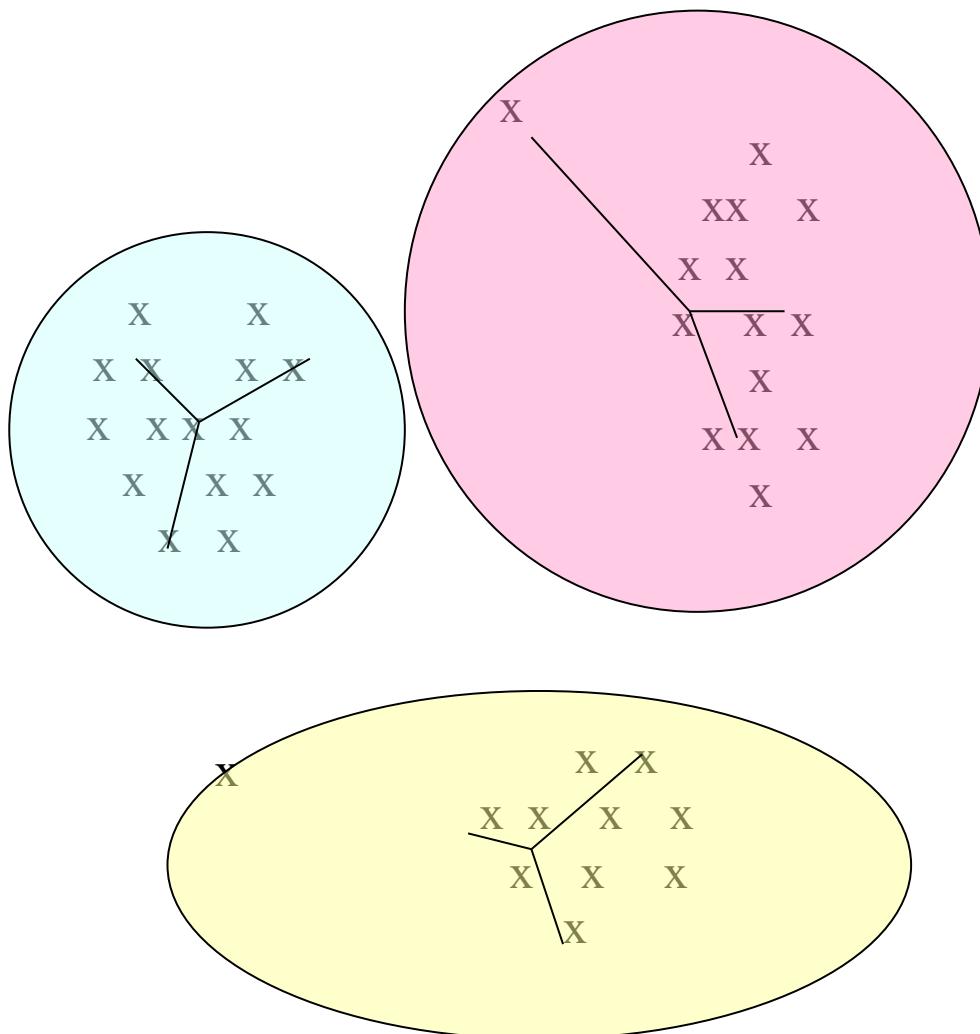
# Example: Picking $k$

Too few;  
many long  
distances  
to centroid.



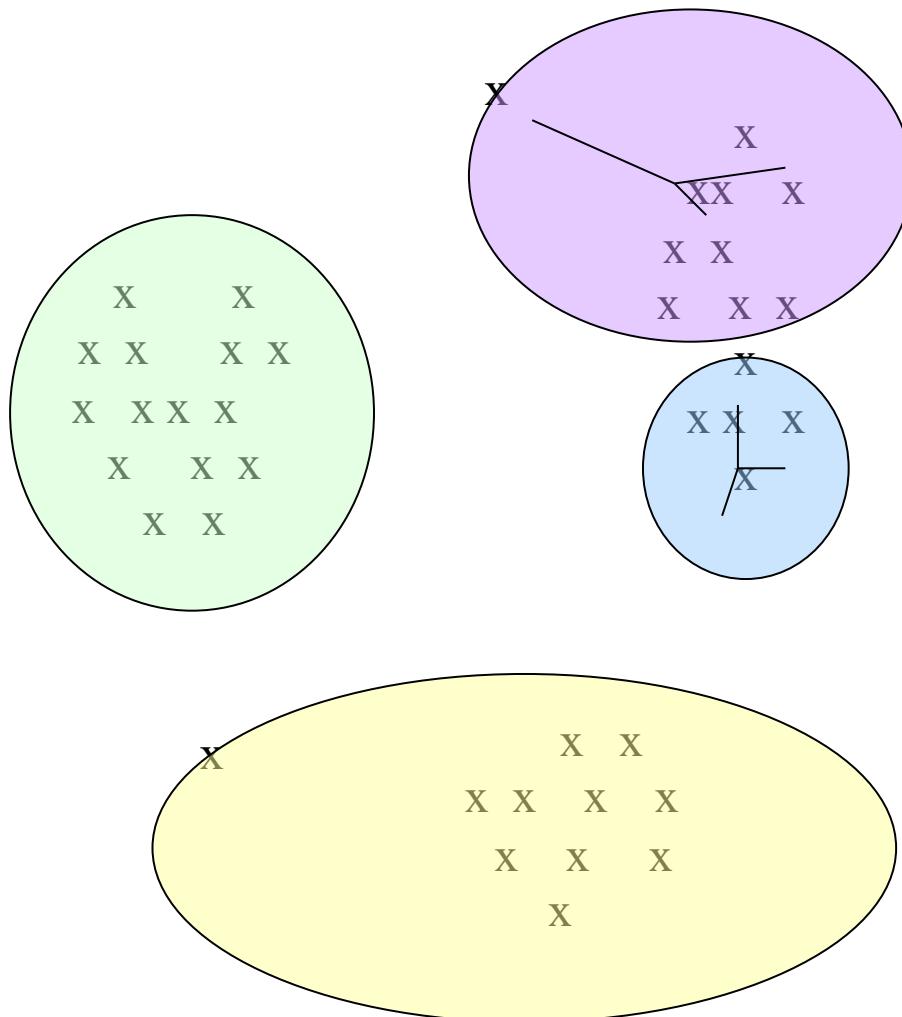
# Example: Picking $k$

Just right;  
distances  
rather short.



# Example: Picking $k$

**Too many;**  
little improvement  
in average  
distance.



# Summary

- **Clustering:** Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***
- **Algorithms:**
  - Agglomerative **hierarchical clustering**:
    - Centroid and clustroid
  - ***k-means*:**
    - Initialization, picking  $k$

**Note to other teachers and users of these slides:** We would be delighted if you found this our material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: <http://www.mmds.org>

# Large-Scale Machine Learning: k-NN, Perceptron

Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman  
Stanford University

<http://www.mmds.org>



# New Topic: Machine Learning!

High dim.  
data

Locality  
sensitive  
hashing

Clustering

Dimensional  
ity  
reduction

Graph  
data

PageRank,  
SimRank

Community  
Detection

Spam  
Detection

Infinite  
data

Filtering  
data  
streams

Web  
advertising

Queries on  
streams

Machine  
learning

SVM

Decision  
Trees

Perceptron,  
kNN

Apps

Recommen  
der systems

Association  
Rules

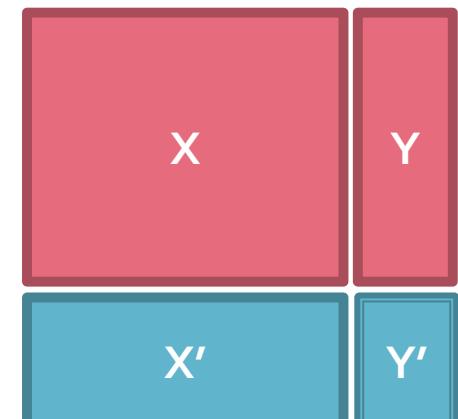
Duplicate  
document  
detection

# Supervised Learning

- Would like to do **prediction**:  
**estimate** a function  $f(x)$  so that  $y = f(x)$

- Where **y** can be:
  - **Real number**: Regression
  - **Categorical**: Classification
  - Complex object:
    - Ranking of items, Parse tree, etc.

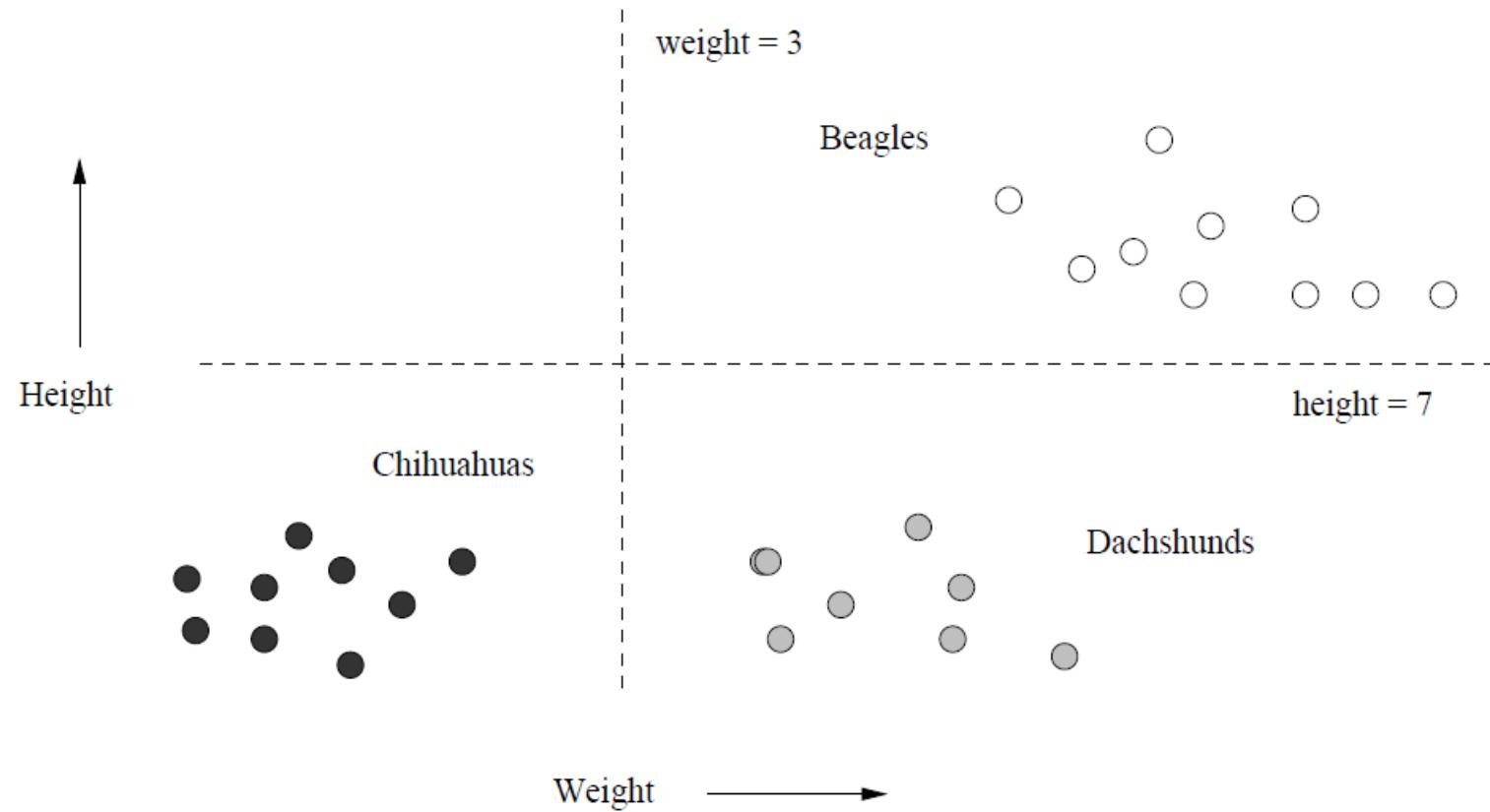
- Data is **labeled**:
  - Have many pairs  $\{(x, y)\}$ 
    - **x** ... vector of binary, categorical, real valued features
    - **y** ... class ( $\{+1, -1\}$ , or a real number)



Training and test set

Estimate  $y = f(x)$  on  $X, Y$ .  
Hope that the same  $f(x)$   
also works on unseen  $X', Y'$

# Example



```
if (height > 7) print Beagle  
else if (weight < 3) print Chihuahua  
else print Dachshund;
```

# Training set/ Test set/Val

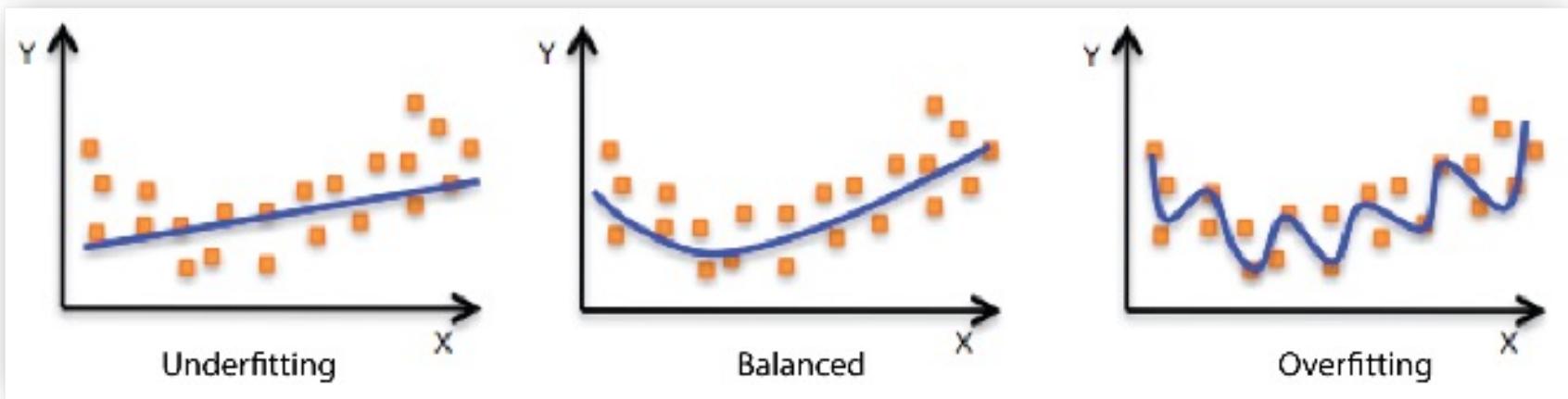
- In some cases, we withhold two sets of training examples,
- a *validation set (sometimes called a development set)*, as well as the *test set*.
- The *difference is that the validation set is used to help design the model, while the test set is used only to determine how good the model is.*
- The problem addressed by a validation set is that many machine-learning algorithms tend to *overfit the data*
- By using the validation set, and seeing how well the classifier works on that, we can tell if the classifier is overfitting the data.

# Overfitting

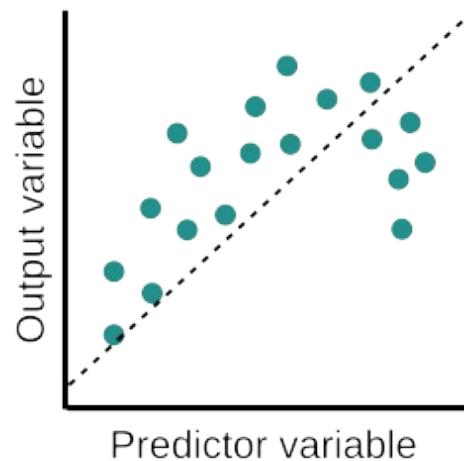


**THE BEST WAY TO  
EXPLAIN OVERFITTING**

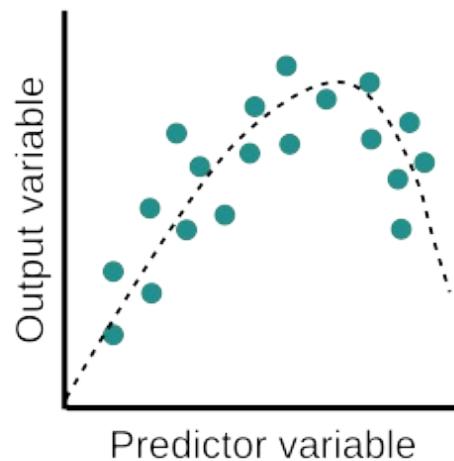
# Over fitting underfitting



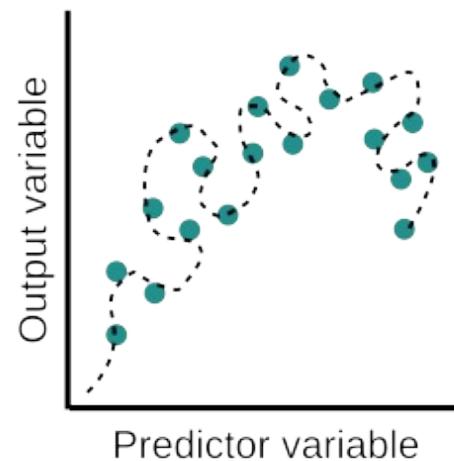
Underfit



Optimal



Overfit



# Large Scale Machine Learning

- We will talk about the following methods:
  - Perceptron and Winnow algorithms
  - Support Vector Machines

# Linear models: Perceptron

- Example: **Spam filtering**

	viagra	learning	the	dating	nigeria	spam?
$\vec{x}_1 = ($	1	0	1	0	0 $)$	$y_1 = 1$
$\vec{x}_2 = ($	0	1	1	0	0 $)$	$y_2 = -1$
$\vec{x}_3 = ($	0	0	0	0	1 $)$	$y_3 = 1$

- Instance space  $x \in X$  ( $|X| = n$  data points)
  - Binary or real-valued feature vector  $x$  of word occurrences
  - $d$  features (words + other things,  $d \sim 100,000$ )
- Class  $y \in Y$ 
  - $y$ : Spam (+1), Ham (-1)

# Linear models for classification

## ■ Binary classification:

$$f(\mathbf{x}) = \begin{cases} +1 & \text{if } w_1 x_1 + w_2 x_2 + \dots + w_d x_d \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

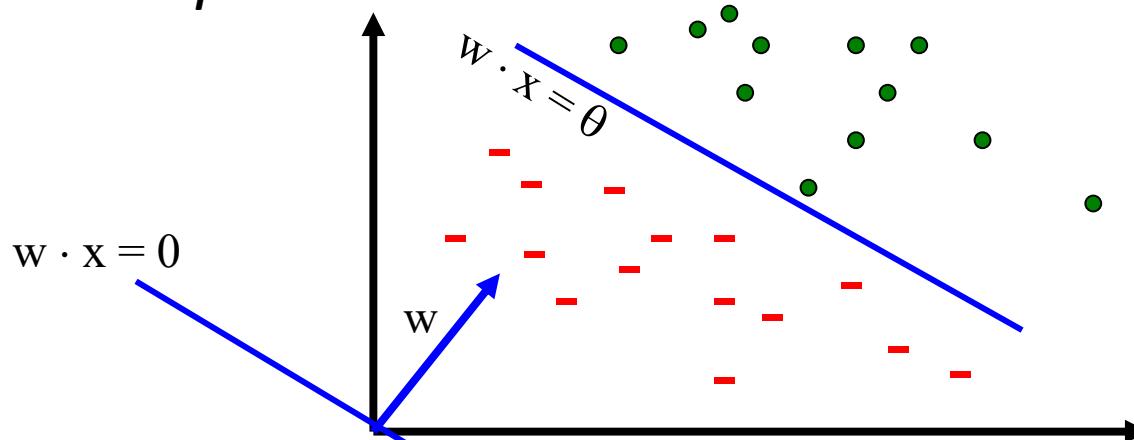
Decision boundary is linear

## ■ Input: Vectors $\mathbf{x}^{(j)}$ and labels $y^{(j)}$

- Vectors  $\mathbf{x}^{(j)}$  are real valued where  $\|\mathbf{x}\|_2 = 1$

## ■ Goal: Find vector $w = (w_1, w_2, \dots, w_d)$

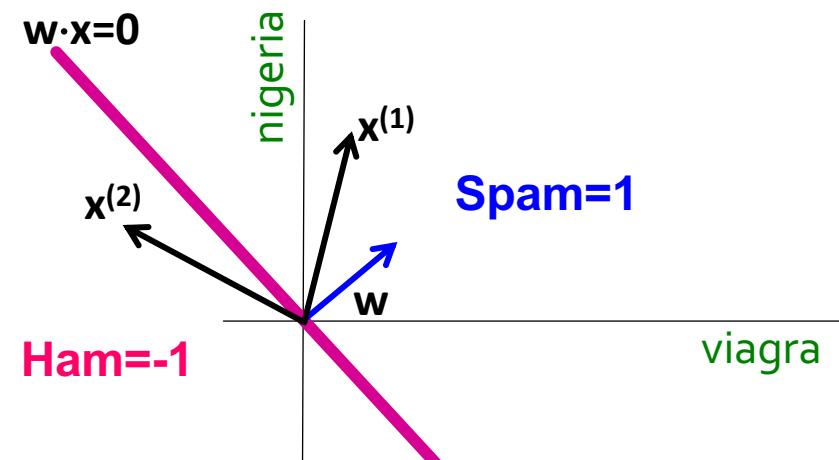
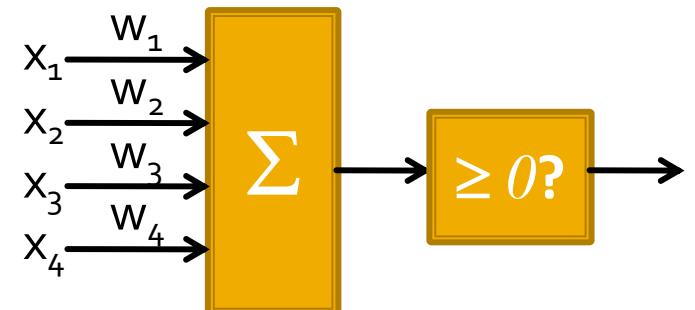
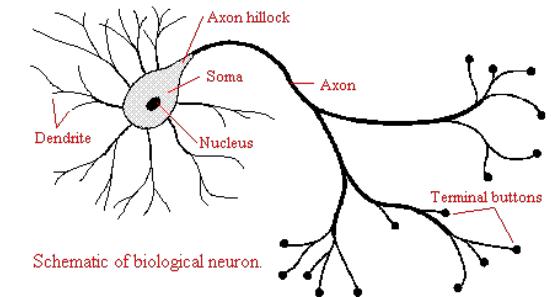
- Each  $w_i$  is a real number



Note:  
 $\mathbf{x} \Leftrightarrow \langle \mathbf{x}, 1 \rangle \quad \forall \mathbf{x}$   
 $\mathbf{w} \Leftrightarrow \langle \mathbf{w}, -\theta \rangle$

# Perceptron [Rosenblatt '58]

- (very) Loose motivation: Neuron
- Inputs are feature values
- Each feature has a weight  $w_i$
- Activation is the sum:
  - $f(x) = \sum_i w_i x_i = w \cdot x$
- If the  $f(x)$  is:
  - Positive: Predict +1
  - Negative: Predict -1

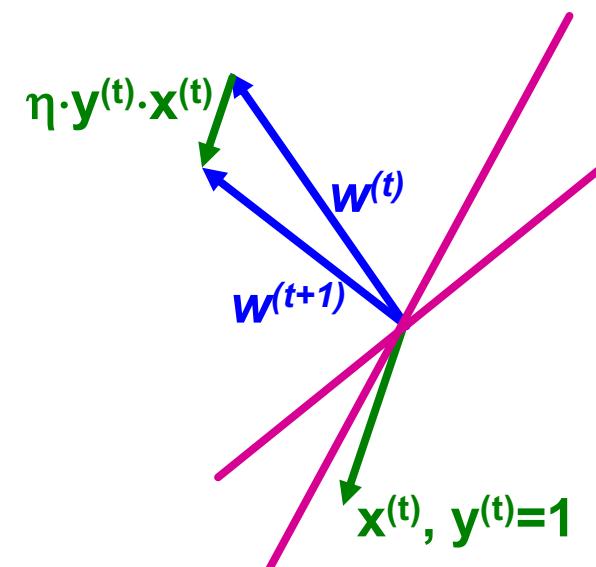


# Perceptron: Estimating $w$

- Perceptron:  $y' = \text{sign}(w \cdot x)$
- How to find parameters  $w$ ?

- Start with  $w_0 = 0$
- Pick training examples  $x^{(t)}$  one by one (from disk)
- Predict class of  $x^{(t)}$  using current weights
  - $y' = \text{sign}(w^{(t)} \cdot x^{(t)})$
- If  $y'$  is correct (i.e.,  $y_t = y'$ )
  - No change:  $w^{(t+1)} = w^{(t)}$
- If  $y'$  is wrong: adjust  $w^{(t)}$ 
$$w^{(t+1)} = w^{(t)} + \eta \cdot y^{(t)} \cdot x^{(t)}$$
  - $\eta$  is the learning rate parameter
  - $x^{(t)}$  is the t-th training example
  - $y^{(t)}$  is true t-th class label ( $\{+1, -1\}$ )

Note that the Perceptron is a conservative algorithm: it ignores samples that it classifies correctly.

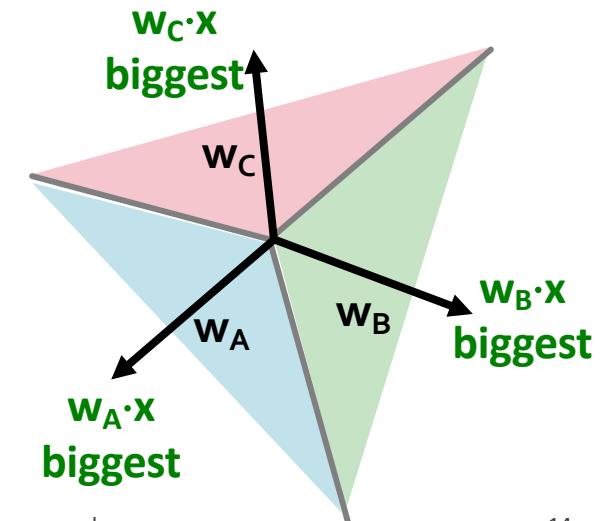


# Updating the Learning Rate

- Perceptron will oscillate and won't converge
- When to stop learning?
- (1) Slowly decrease the learning rate  $\eta$ 
  - A classic way is to:  $\eta = c_1/(t + c_2)$ 
    - But, we also need to determine constants  $c_1$  and  $c_2$
- (2) Stop when the training error stops chaining
- (3) Have a small test dataset and stop when the test set error stops decreasing
- (4) Stop when we reached some maximum number of passes over the data

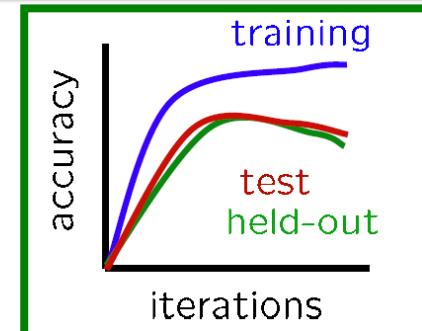
# Multiclass Perceptron

- What if more than 2 classes?
  - Weight vector  $w_c$  for each class c
    - Train one class vs. the rest:
    - Example: 3-way classification  $y = \{A, B, C\}$
    - Train 3 classifiers:  $w_A$ : A vs. B,C;  $w_B$ : B vs. A,C;  $w_C$ : C vs. A,B
  - Calculate activation for each class
- $$f(x,c) = \sum_i w_{c,i} x_i = w_c \cdot x$$
- Highest activation wins
  - $c = \arg \max_c f(x,c)$

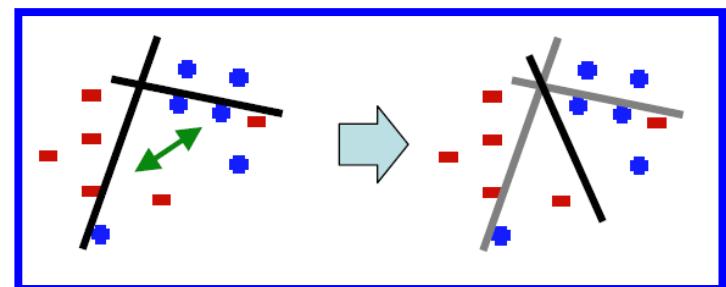


# Issues with Perceptrons

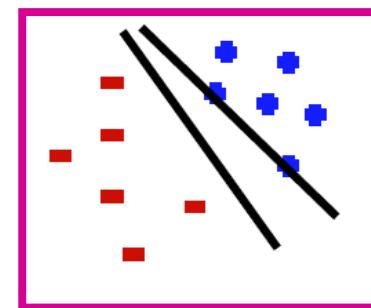
- **Overfitting:**



- **Regularization:** If the data is not separable weights dance around



- **Mediocre generalization:**
  - Finds a “barely” separating solution

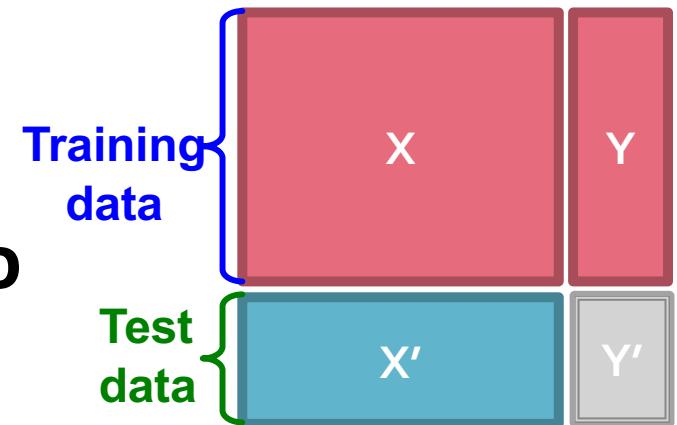


# More generally: Supervised Learning

- Would like to do **prediction**:  
estimate a function  $f(x)$  so that  $y = f(x)$
- Where **y** can be:
  - **Real number**: Regression
  - **Categorical**: Classification
  - **Complex object**:
    - Ranking of items, Parse tree, etc.
- Data is **labeled**:
  - Have many pairs  $\{(x, y)\}$ 
    - $x$  ... vector of binary, categorical, real valued features
    - $y$  ... class ( $\{+1, -1\}$ , or a real number)

# Supervised Learning

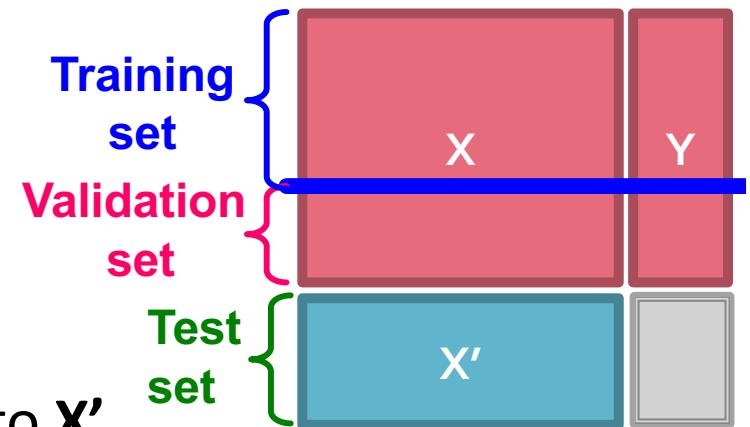
- **Task:** Given data  $(X, Y)$  build a model  $f()$  to predict  $Y'$  based on  $X'$
- **Strategy:** Estimate  $y = f(x)$  on  $(X, Y)$ .  
Hope that the same  $f(x)$  also works to predict unknown  $Y'$ 
  - The “hope” is called **generalization**
    - **Overfitting:** If  $f(x)$  predicts well  $Y$  but is unable to predict  $Y'$
  - **We want to build a model that generalizes well to unseen data**
    - But Jure, how can we well on data we have never seen before?!?



# Supervised Learning

- **Idea:** Pretend we do not know the data/labels we actually do know

- Build the model  $f(x)$  on the training data  
See how well  $f(x)$  does on the test data
    - If it does well, then apply it also to  $X'$



- **Refinement: Cross validation**

- Splitting into training/validation set is brutal
  - Let's split our data  $(X, Y)$  into 10-folds (buckets)
  - Take out 1-fold for validation, train on remaining 9
  - Repeat this 10 times, report average performance

# Linear models for classification

- **Binary classification:**

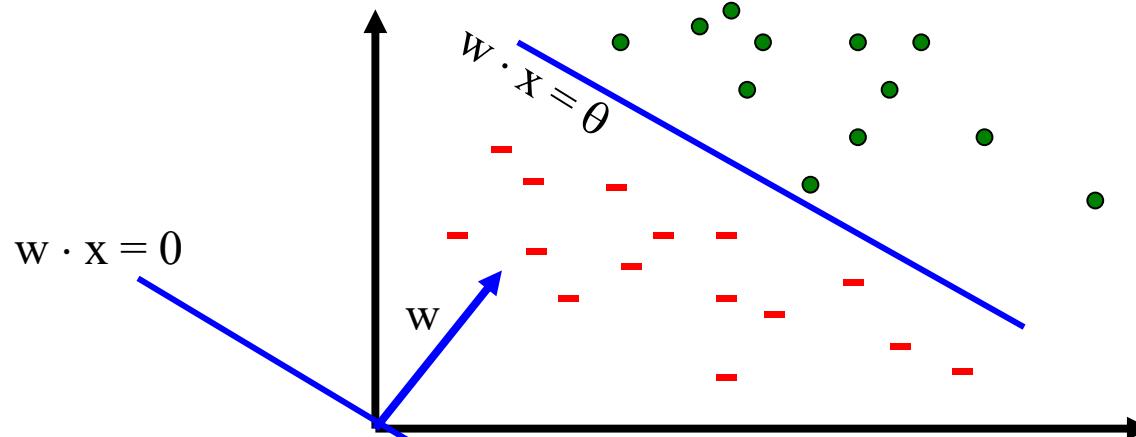
$$f(\mathbf{x}) = \begin{cases} +1 & \text{if } w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \dots + w^{(d)}x^{(d)} \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

- **Input:** Vectors  $\mathbf{x}_j$  and labels  $y_j$

- Vectors  $\mathbf{x}_j$  are real valued where  $\|\mathbf{x}\|_2 = 1$

- **Goal:** Find vector  $w = (w^{(1)}, w^{(2)}, \dots, w^{(d)})$

- Each  $w^{(i)}$  is a real number

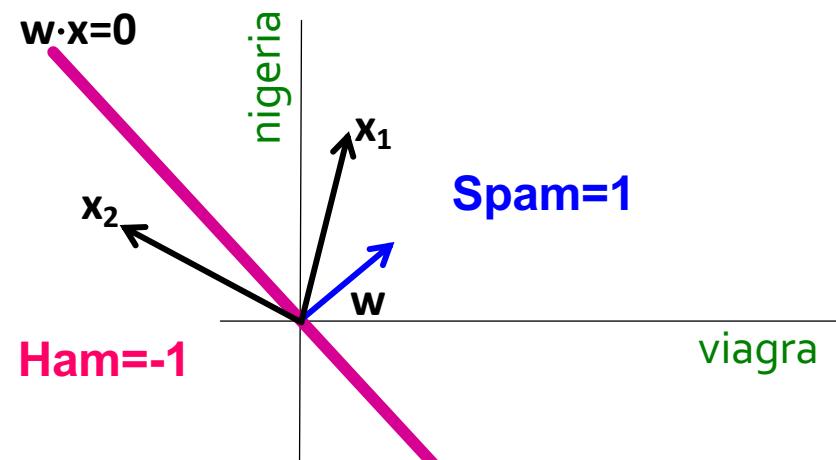
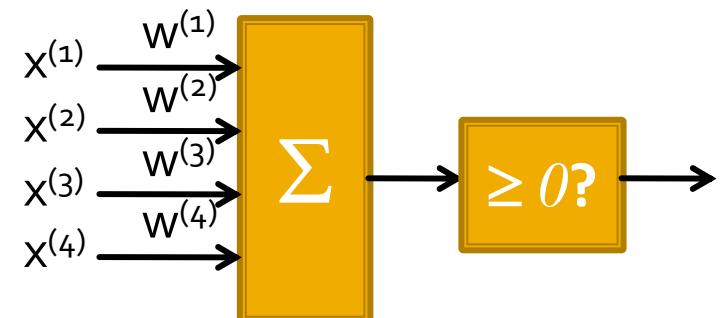
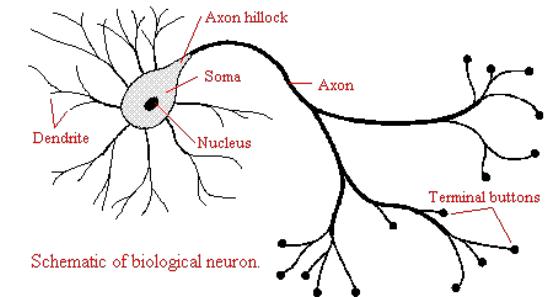


Decision  
boundary  
is linear

**Note:**  
 $\mathbf{x} \rightarrow \langle \mathbf{x}, 1 \rangle \quad \forall \mathbf{x}$   
 $\mathbf{w} \rightarrow \langle \mathbf{w}, -\theta \rangle$

# Perceptron [Rosenblatt '58]

- **(Very) loose motivation:** Neuron
- Inputs are feature values
- Each feature has a weight  $w_i$
- **Activation is the sum:**
  - $f(x) = \sum_i^d w^{(i)} x^{(i)} = w \cdot x$
- If the  $f(x)$  is:
  - Positive: Predict +1
  - Negative: Predict -1



# Perceptron

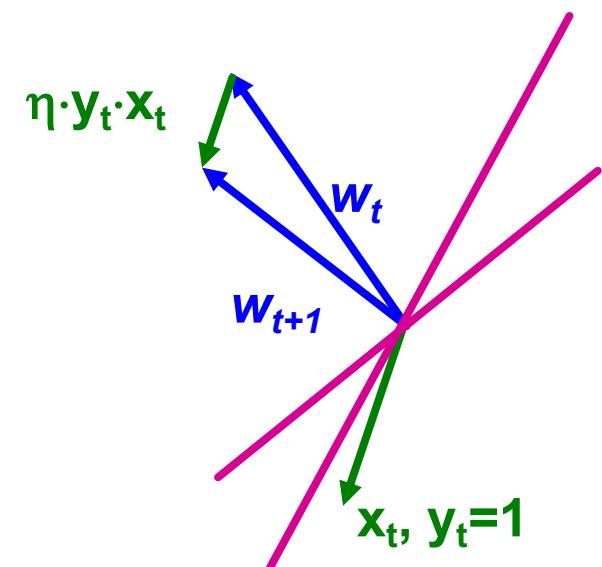
- **Perceptron:  $y' = \text{sign}(w \cdot x)$**
- **How to find parameters  $w$ ?**

- Start with  $w_0 = 0$
- Pick training examples  $x_t$  **one by one**
- Predict class of  $x_t$  using current  $w_t$ 
  - $y' = \text{sign}(w_t \cdot x_t)$
- **If  $y'$  is correct (i.e.,  $y_t = y'$ )**
  - No change:  $w_{t+1} = w_t$
- **If  $y'$  is wrong:** Adjust  $w_t$

$$w_{t+1} = w_t + \eta \cdot y_t \cdot x_t$$

- $\eta$  is the learning rate parameter
- $x_t$  is the t-th training example
- $y_t$  is true t-th class label ( $\{+1, -1\}$ )

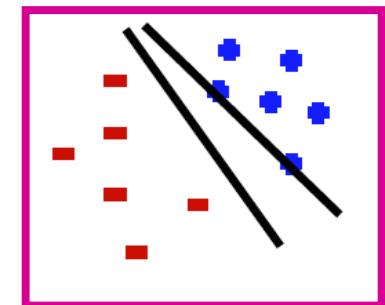
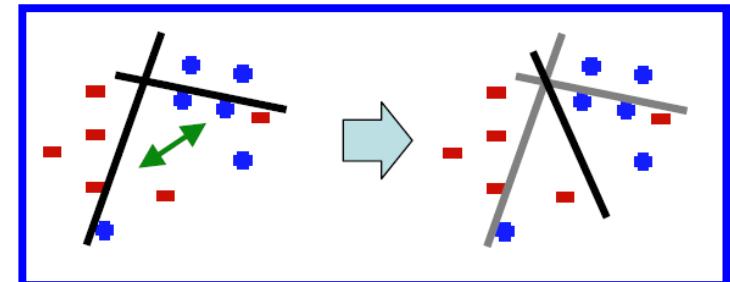
Note that the Perceptron is a conservative algorithm: it ignores samples that it classifies correctly.



# Perceptron: The Good and the Bad

- **Good: Perceptron convergence theorem:**
  - If there exist a set of weights that are consistent (i.e., the data is linearly separable) the Perceptron learning algorithm will converge
- **Bad: Never converges:**

If the data is not separable weights dance around indefinitely
- **Bad: Mediocre generalization:**
  - Finds a “barely” separating solution



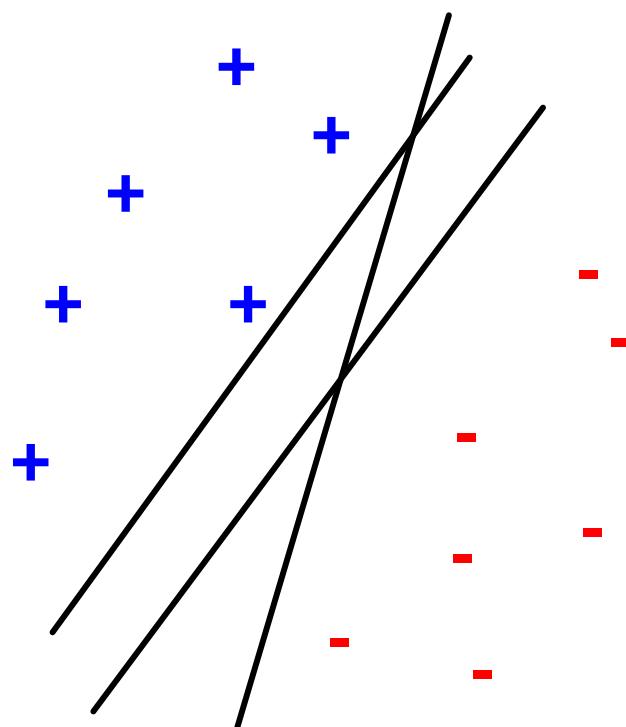
# Updating the Learning Rate

- Perceptron will oscillate and won't converge
- So, when to stop learning?
- (1) Slowly decrease the learning rate  $\eta$ 
  - A classic way is to:  $\eta = c_1/(t + c_2)$ 
    - But, we also need to determine constants  $c_1$  and  $c_2$
- (2) Stop when the training error stops chaining
- (3) Have a small test dataset and stop when the test set error stops decreasing
- (4) Stop when we reached some maximum number of passes over the data

# **Support Vector Machines**

# Support Vector Machines

- Want to separate "+" from "-" using a line



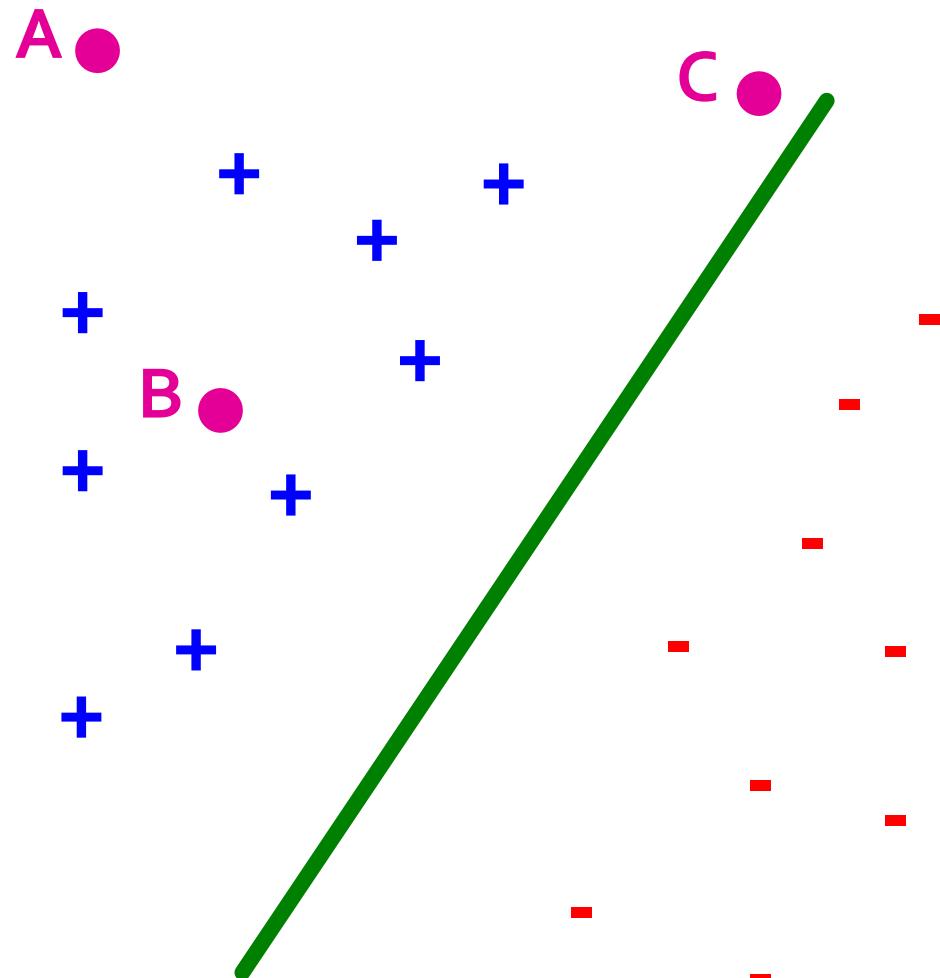
Data:

- Training examples:
  - $(x_1, y_1) \dots (x_n, y_n)$
- Each example  $i$ :
  - $x_i = (x_i^{(1)}, \dots, x_i^{(d)})$
  - $x_i^{(j)}$  is real valued
  - $y_i \in \{-1, +1\}$
- Inner product:

$$w \cdot x = \sum_{j=1}^d w^{(j)} x^{(j)}$$

Which is best linear separator (defined by  $w$ )?

# Largest Margin

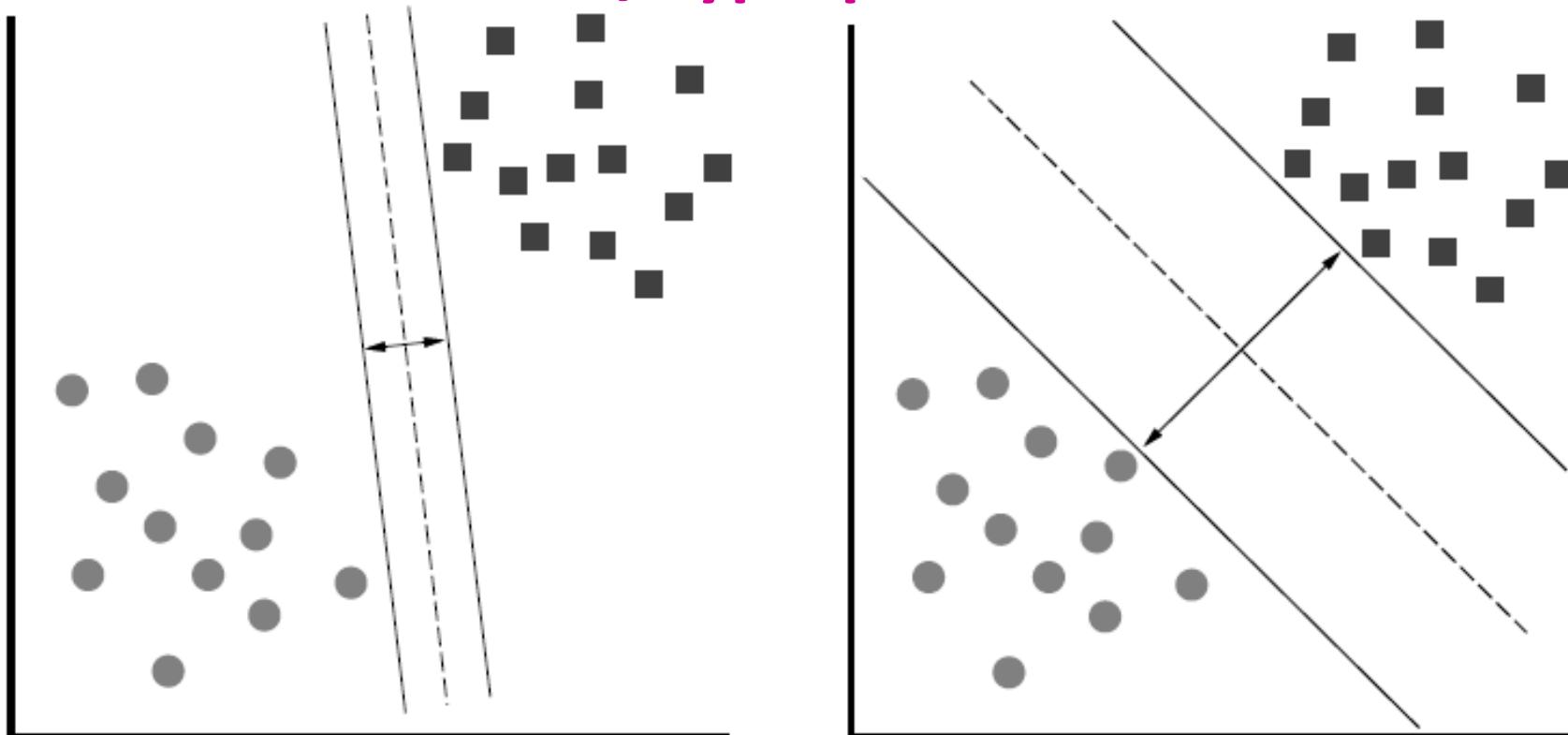


- Distance from the separating hyperplane corresponds to the “confidence” of prediction
- Example:

- We are more sure about the class of A and B than of C

# Largest Margin

- Margin  $\gamma$ : Distance of closest example from the decision line/hyperplane

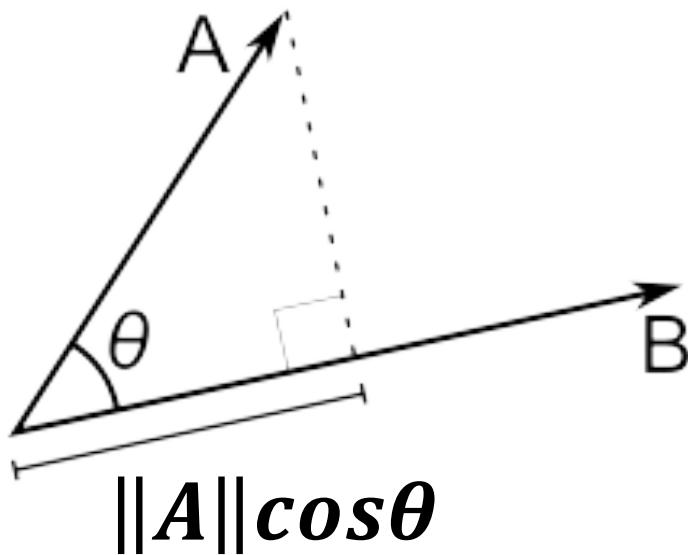


The reason we define margin this way is due to theoretical convenience and existence of generalization error bounds that depend on the value of margin.

# Why maximizing $\gamma$ a good idea?

- Remember: Dot product

$$A \cdot B = \|A\| \cdot \|B\| \cdot \cos \theta$$



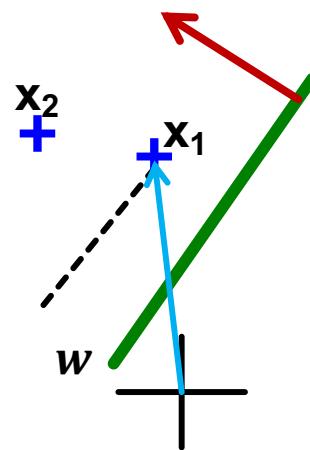
$$\|A\| = \sqrt{\sum_{j=1}^d (A^{(j)})^2}$$

# Why maximizing $\gamma$ a good idea?

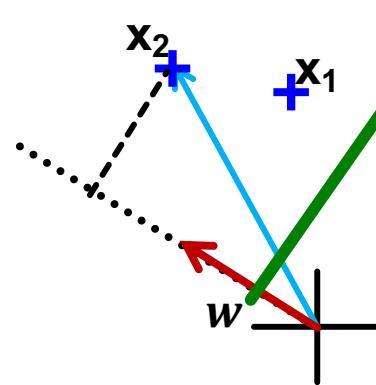
- Dot product

$$A \cdot B = \|A\| \|B\| \cos \theta$$

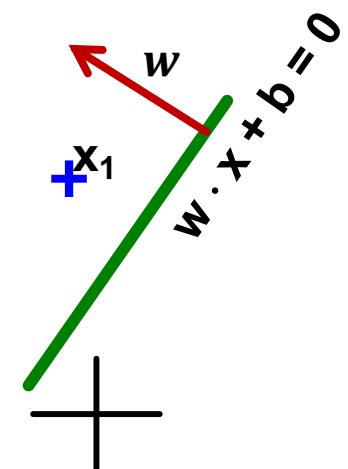
- What is  $w \cdot x_1, w \cdot x_2$ ?



In this case  
 $\gamma_1 \approx \|w\|^2$

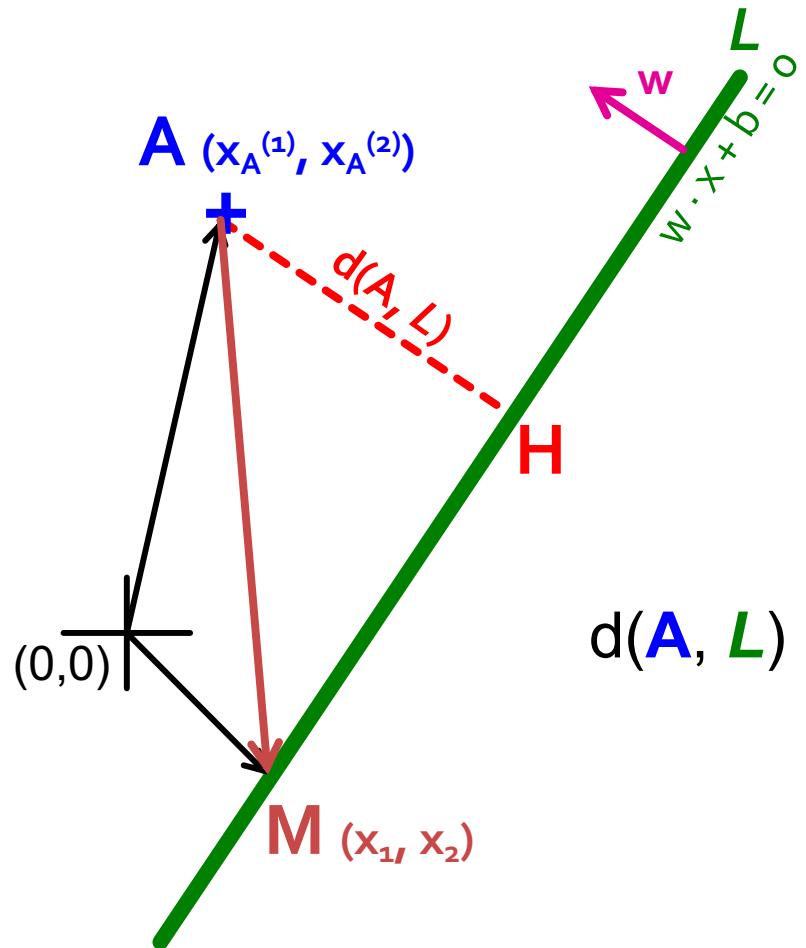


In this case  
 $\gamma_2 \approx 2\|w\|^2$



- So,  $\gamma$  roughly corresponds to the margin
  - Bigger  $\gamma$  bigger the separation

# What is the margin?



Let:

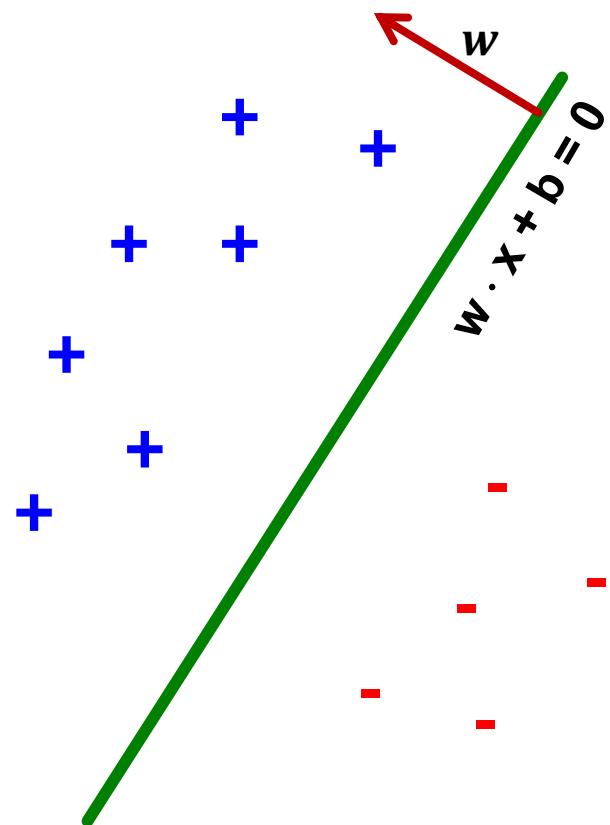
- Line  $L$ :  $w \cdot x + b = 0$
- $w = (w^{(1)}, w^{(2)})$
- Point  $A = (x_A^{(1)}, x_A^{(2)})$
- Point  $M$  on a line  $= (x_M^{(1)}, x_M^{(2)})$

Note we assume  
 $\|w\|_2 = 1$

$$\begin{aligned}d(A, L) &= |AH| \\&= |(A-M) \cdot w| \\&= |(x_A^{(1)} - x_M^{(1)}) w^{(1)} + (x_A^{(2)} - x_M^{(2)}) w^{(2)}| \\&= x_A^{(1)} w^{(1)} + x_A^{(2)} w^{(2)} + b \\&= w \cdot A + b\end{aligned}$$

Remember  $x_M^{(1)}w^{(1)} + x_M^{(2)}w^{(2)} = -b$   
since  $M$  belongs to line  $L$

# Largest Margin



- Prediction =  $\text{sign}(w \cdot x + b)$
- “Confidence” =  $(w \cdot x + b) y$
- For i-th datapoint:  
$$\gamma_i = (w \cdot x_i + b) y_i$$
- Want to solve:  
$$\max_w \min_i \gamma_i$$
- Can rewrite as  
$$\max_{w, \gamma} \gamma$$
  
$$s.t. \forall i, y_i (w \cdot x_i + b) \geq \gamma$$

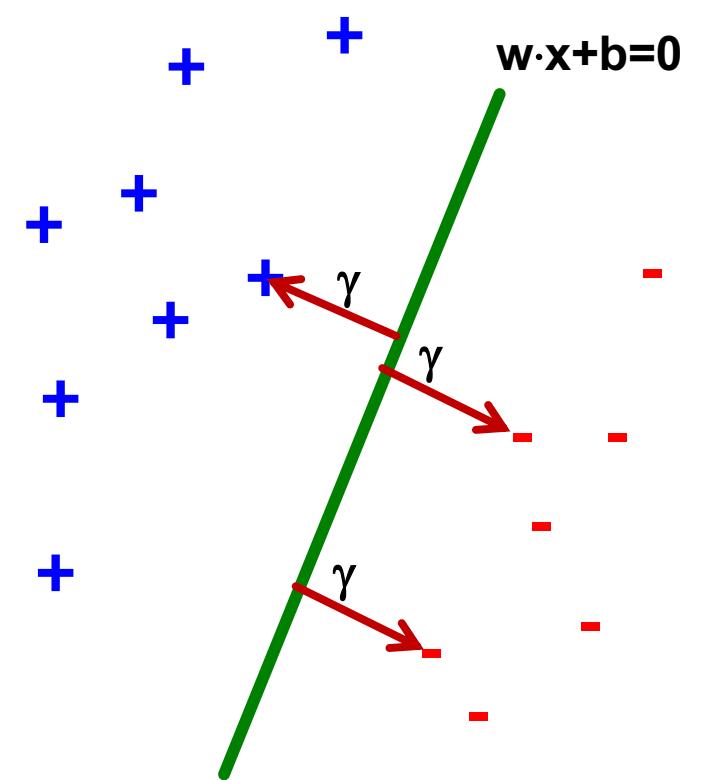
# Support Vector Machine

- **Maximize the margin:**
  - Good according to intuition, theory (VC dimension) & practice

$$\max_{w, \gamma} \gamma$$

$$s.t. \forall i, y_i(w \cdot x_i + b) \geq \gamma$$

- $\gamma$  is margin ... distance from the separating hyperplane

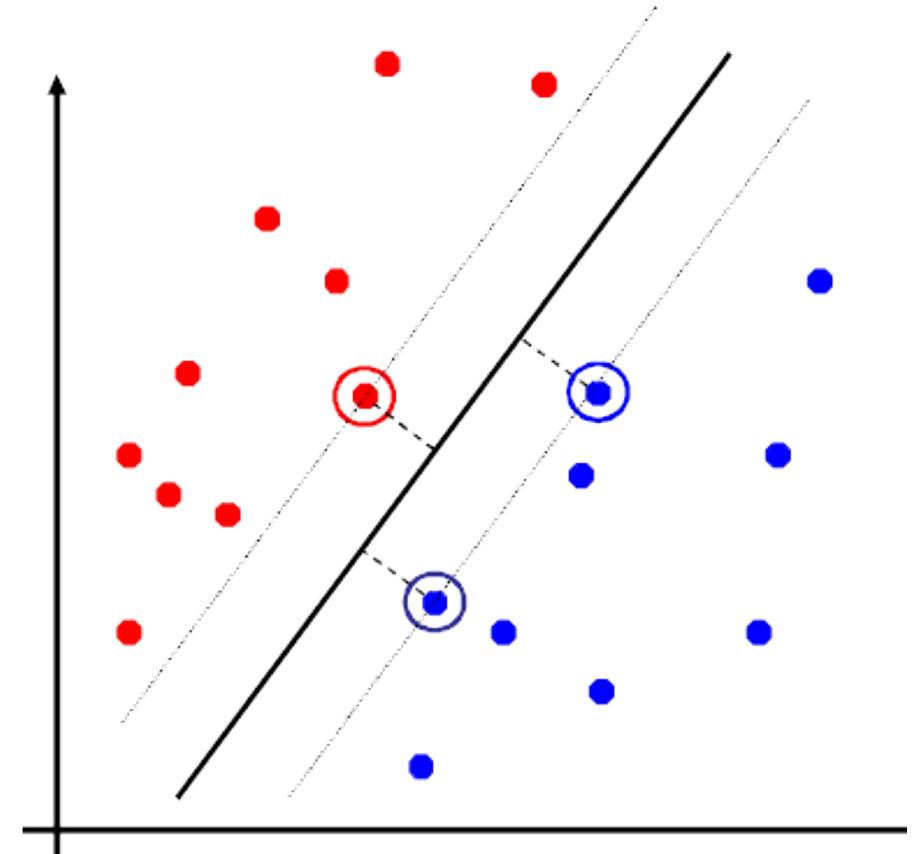


# Support Vector Machines: Deriving the margin

# Support Vector Machines

- Separating hyperplane is defined by the support vectors

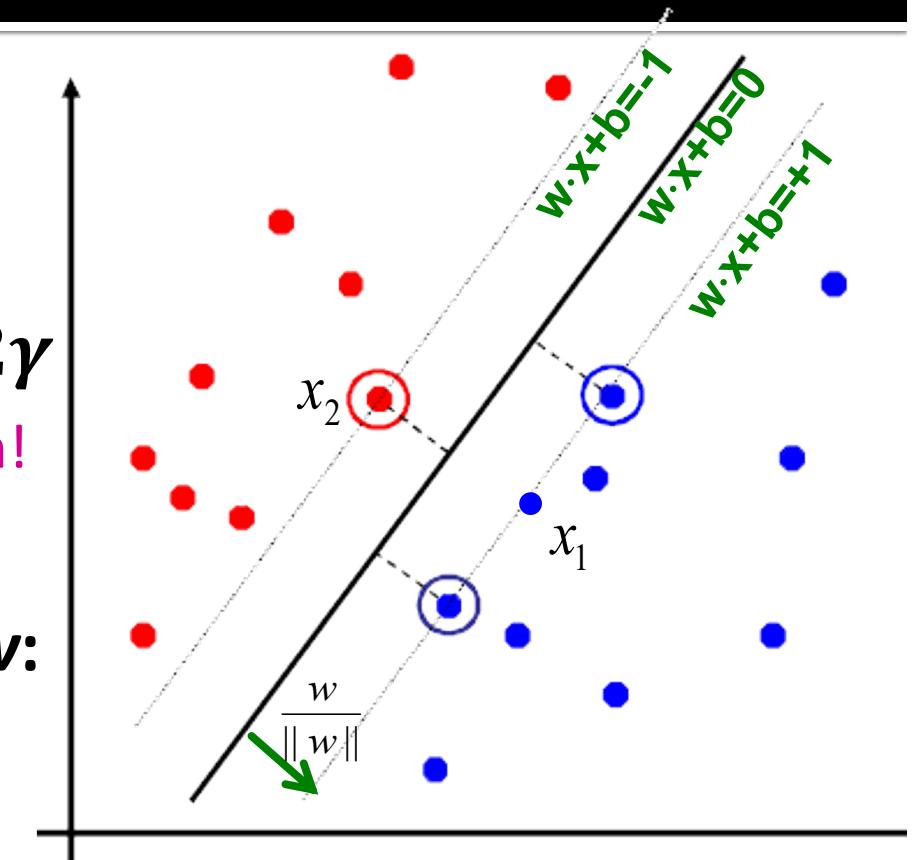
- Points on +/- planes from the solution
- If you knew these points, you could ignore the rest
- Generally,  $d+1$  support vectors (for  $d$  dim. data)



# Canonical Hyperplane: Problem

## ■ Problem:

- Let  $(w \cdot x + b)y = \gamma$   
then  $(2w \cdot x + 2b)y = 2\gamma$ 
  - Scaling  $w$  increases margin!



## ■ Solution:

- Work with normalized  $w$ :

$$\gamma = \left( \frac{w}{\|w\|} \cdot x + b \right) y$$

- Let's also require **support vectors**  $x_j$  to be on the plane defined by:  $w \cdot x_j + b = \pm 1$

$$\|w\| = \sqrt{\sum_{j=1}^d (w^{(j)})^2}$$

# Canonical Hyperplane: Solution

- Want to maximize margin  $\gamma$ !
- What is the relation between  $x_1$  and  $x_2$ ?

- $x_1 = x_2 + 2\gamma \frac{w}{\|w\|}$

- We also know:

- $w \cdot x_1 + b = +1$
- $w \cdot x_2 + b = -1$

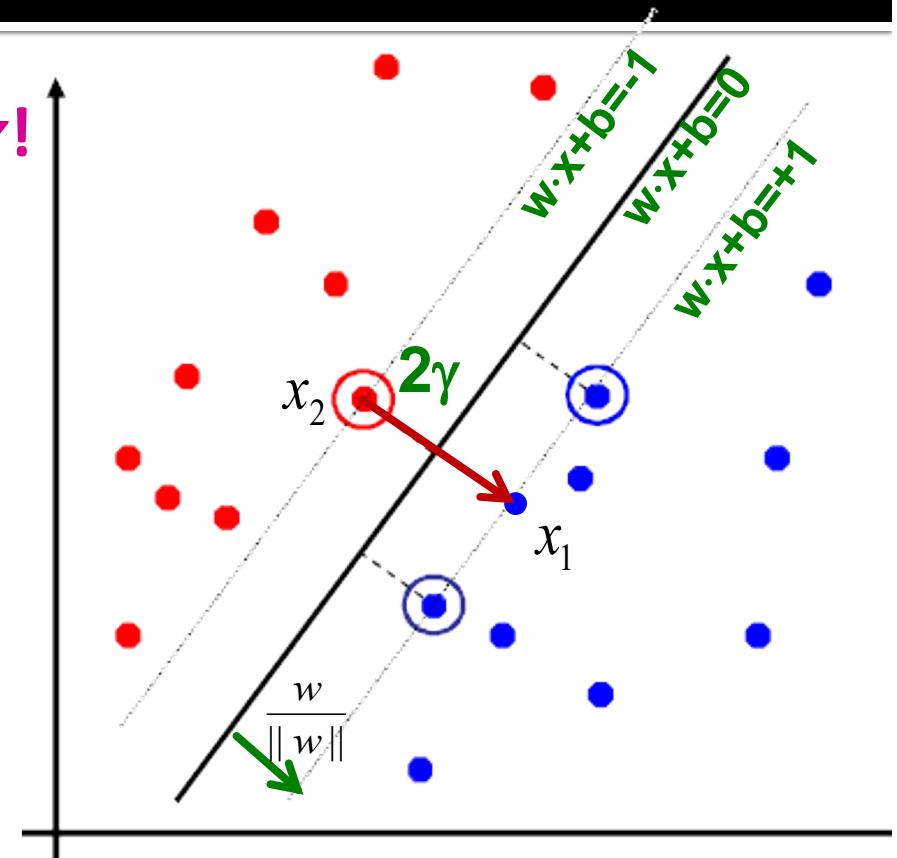
- So:

- $w \cdot x_1 + b = +1$

- $w \left( x_2 + 2\gamma \frac{w}{\|w\|} \right) + b = +1$

- $w \cdot x_2 + b + 2\gamma \frac{w \cdot w}{\|w\|} = +1$

$-1$



$$\Rightarrow \gamma = \frac{\|w\|}{w \cdot w} = \frac{1}{\|w\|}$$

Note:  
 $w \cdot w = \|w\|^2$

# Maximizing the Margin

- We started with

$$\max_{w, \gamma} \gamma$$

$$s.t. \forall i, y_i(w \cdot x_i + b) \geq \gamma$$

But  $w$  can be arbitrarily large!

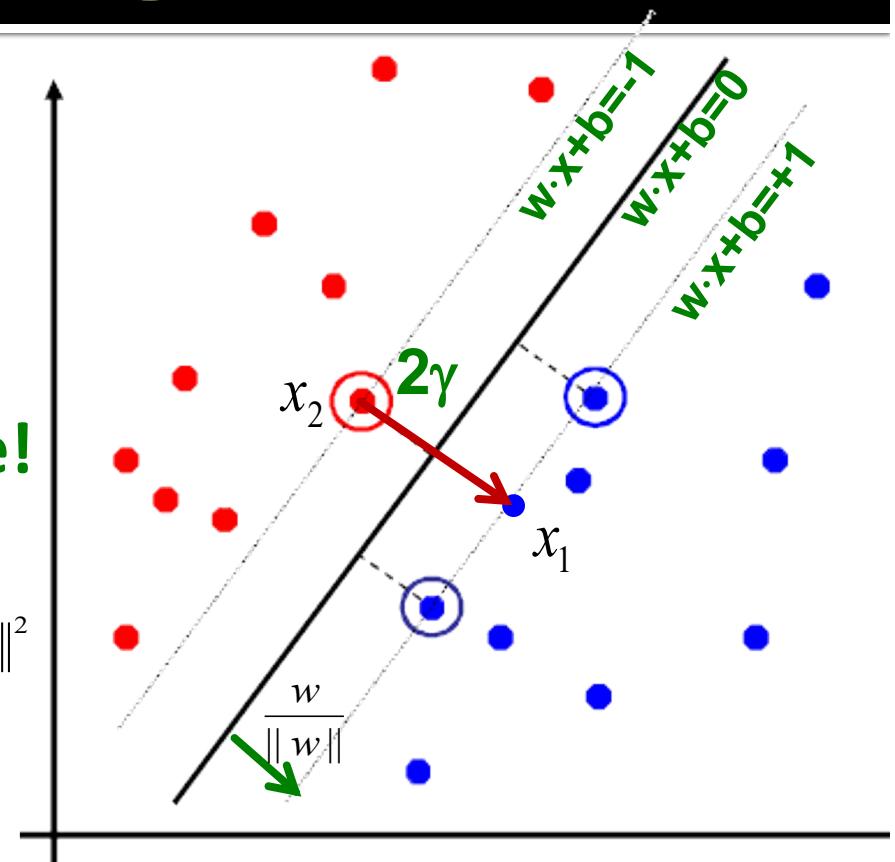
- We normalized and...

$$\arg \max \gamma = \arg \max \frac{1}{\|w\|} = \arg \min \|w\| = \arg \min \frac{1}{2} \|w\|^2$$

- Then:

$$\min_w \frac{1}{2} \|w\|^2$$

$$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1$$



This is called SVM with “hard” constraints

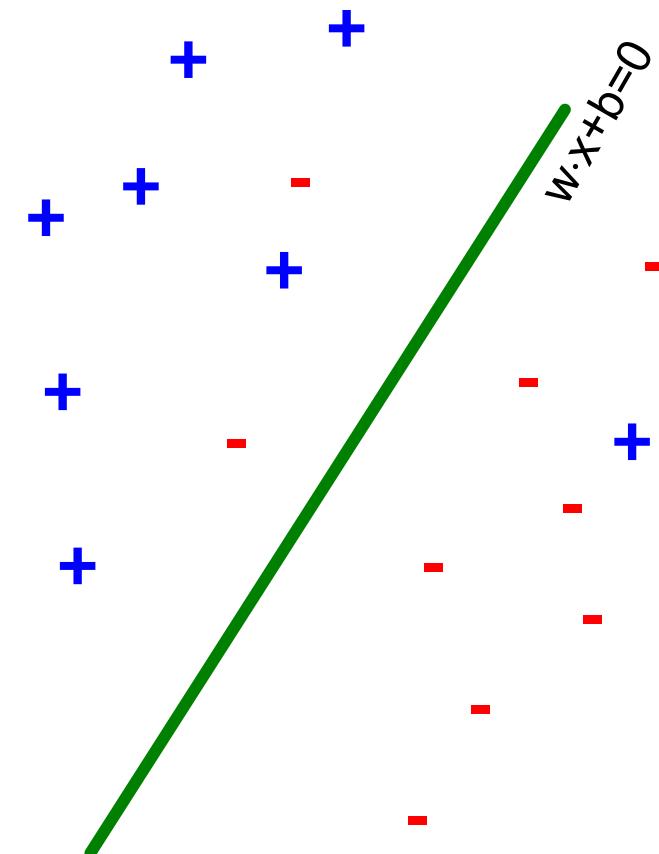
# Non-linearly Separable Data

- If data is **not separable** introduce **penalty**:

$$\min_w \frac{1}{2} \|w\|^2 + C \cdot (\# \text{number of mistakes})$$

$$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1$$

- Minimize  $\|w\|^2$  plus the number of training mistakes
  - Set **C** using cross validation
- 
- How to penalize mistakes?
  - All mistakes are not equally bad!



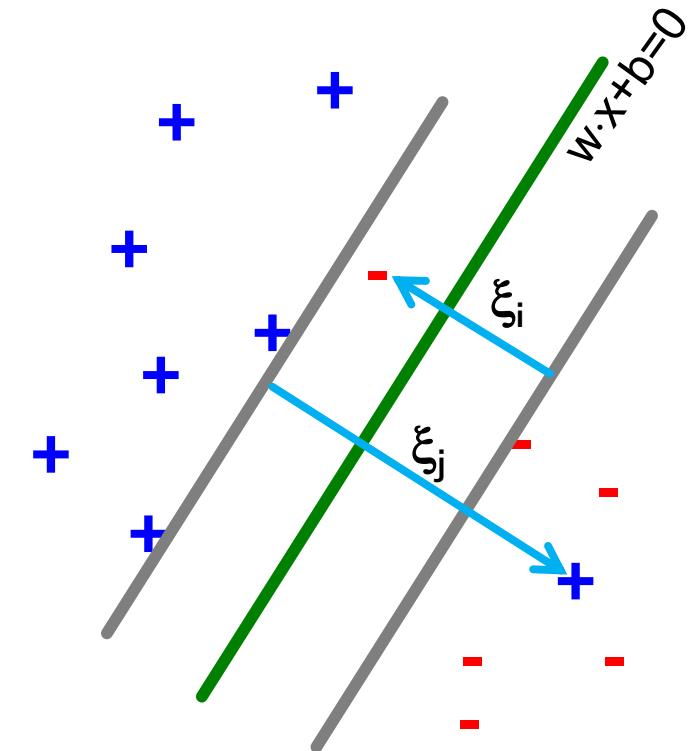
# Support Vector Machines

- Introduce slack variables  $\xi_i$

$$\min_{w,b,\xi_i \geq 0} \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^n \xi_i$$

$$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1 - \xi_i$$

- If point  $x_i$  is on the wrong side of the margin then get penalty  $\xi_i$



For each data point:  
If margin  $\geq 1$ , don't care  
If margin  $< 1$ , pay linear penalty

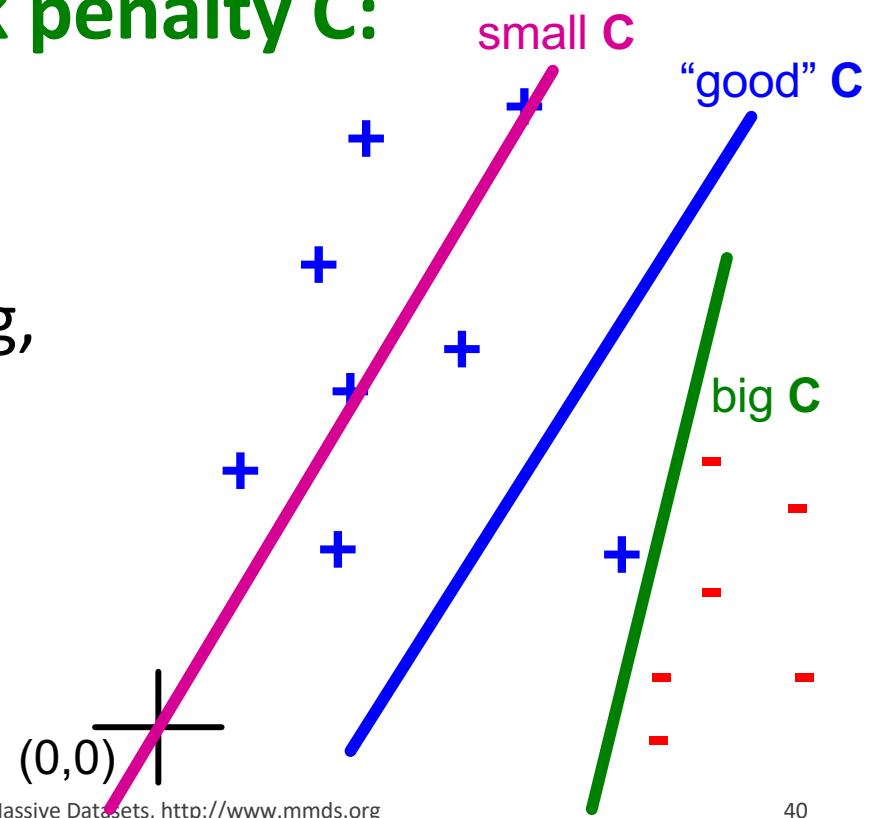
# Slack Penalty $C$

$$\min_w \frac{1}{2} \|w\|^2 + C \cdot (\# \text{number of mistakes})$$

$$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1$$

- **What is the role of slack penalty  $C$ :**

- **$C=\infty$ :** Only want to  $w, b$  that separate the data
- **$C=0$ :** Can set  $\xi_i$  to anything, then  $w=0$  (basically ignores the data)



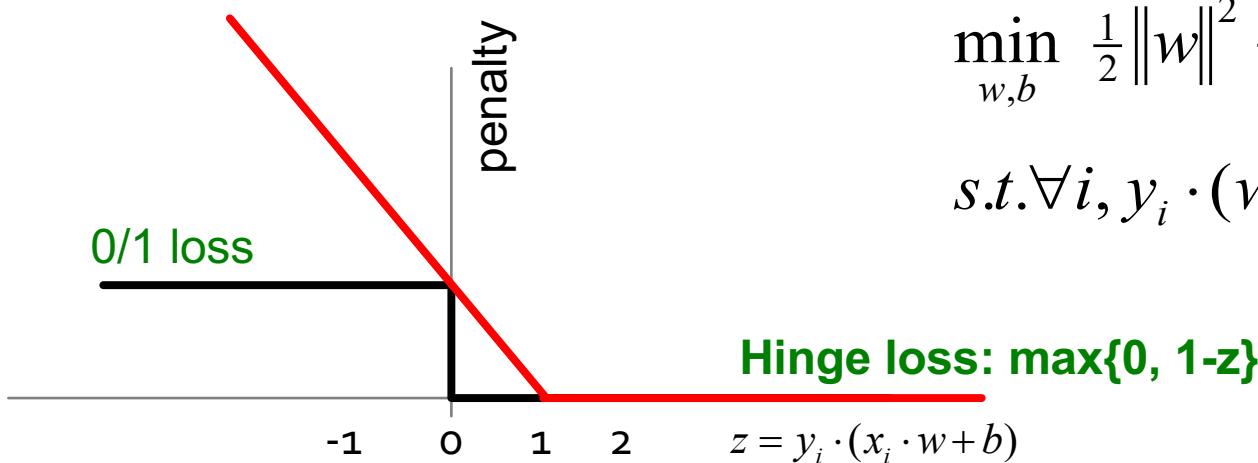
# Support Vector Machines

- SVM in the “natural” form

$$\arg \min_{w,b} \frac{1}{2} \underbrace{w \cdot w}_{\text{Margin}} + C \cdot \sum_{i=1}^n \max \{0, 1 - y_i (w \cdot x_i + b)\}$$

↑  
Regularization parameter  
Empirical loss  $L$  (how well we fit training data)

- SVM uses “Hinge Loss”:



$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$$
$$s.t. \forall i, y_i \cdot (w \cdot x_i + b) \geq 1 - \xi_i$$

# **Support Vector Machines: How to compute the margin?**

# SVM: How to estimate $w$ ?

$$\min_{w,b} \frac{1}{2} w \cdot w + C \cdot \sum_{i=1}^n \xi_i$$

$$s.t. \forall i, y_i \cdot (x_i \cdot w + b) \geq 1 - \xi_i$$

- **Want to estimate  $w$  and  $b$ !**
  - **Standard way:** Use a solver!
    - **Solver:** software for finding solutions to “common” optimization problems
- **Use a quadratic solver:**
  - Minimize quadratic function
  - Subject to linear constraints
- **Problem:** Solvers are inefficient for big data!

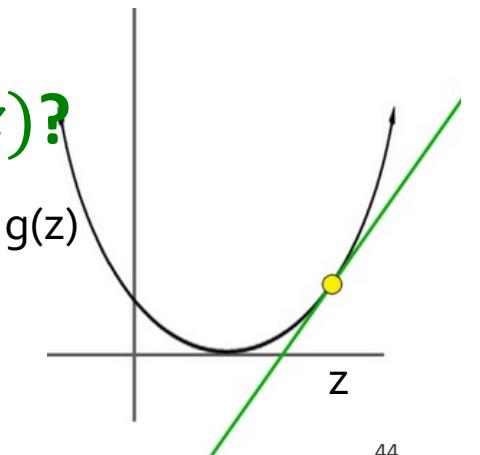
# SVM: How to estimate $w$ ?

- Want to estimate  $w, b$ !
- Alternative approach:
  - Want to minimize  $f(w, b)$ :

$$f(w, b) = \frac{1}{2} w \cdot w + C \cdot \sum_{i=1}^n \max \left\{ 0, 1 - y_i \left( \sum_{j=1}^d w^{(j)} x_i^{(j)} + b \right) \right\}$$

- Side note:
  - How to minimize convex functions  $g(z)$ ?
  - Use gradient descent:  $\min_z g(z)$
  - Iterate:  $z_{t+1} \leftarrow z_t - \eta \nabla g(z_t)$

$$\begin{aligned} & \min_{w,b} \frac{1}{2} w \cdot w + C \sum_{i=1}^n \xi_i \\ & s.t. \forall i, y_i \cdot (x_i \cdot w + b) \geq 1 - \xi_i \end{aligned}$$



# SVM: How to estimate $w$ ?

- Want to minimize  $f(w, b)$ :

$$f(w, b) = \frac{1}{2} \sum_{j=1}^d (w^{(j)})^2 + C \sum_{i=1}^n \max \left\{ 0, 1 - y_i \left( \sum_{j=1}^d w^{(j)} x_i^{(j)} + b \right) \right\}$$

**Empirical loss  $L(x_i, y_i)$**

- Compute the gradient  $\nabla f(w, b)$  w.r.t.  $w^{(j)}$

$$\nabla f^{(j)} = \frac{\partial f(w, b)}{\partial w^{(j)}} = w^{(j)} + C \sum_{i=1}^n \frac{\partial L(x_i, y_i)}{\partial w^{(j)}}$$

$$\frac{\partial L(x_i, y_i)}{\partial w^{(j)}} = 0 \quad \text{if } y_i(w \cdot x_i + b) \geq 1$$
$$= -y_i x_i^{(j)} \quad \text{else}$$

# SVM: How to estimate $w$ ?

## ■ Gradient descent:

**Iterate until convergence:**

- For  $j = 1 \dots d$

- **Evaluate:**  $\nabla f^{(j)} = \frac{\partial f(w, b)}{\partial w^{(j)}} = w^{(j)} + C \sum_{i=1}^n \frac{\partial L(x_i, y_i)}{\partial w^{(j)}}$

- **Update:**  
 $w^{(j)} \leftarrow w^{(j)} - \eta \nabla f^{(j)}$

$\eta$ ...learning rate parameter

C... regularization parameter

## ■ Problem:

- Computing  $\nabla f^{(j)}$  takes  $O(n)$  time!
  - $n$  ... size of the training dataset

# SVM: How to estimate $w$ ?

## ■ Stochastic Gradient Descent

- Instead of evaluating gradient over all examples evaluate it for each **individual** training example

$$\nabla f^{(j)}(x_i) = w^{(j)} + C \cdot \frac{\partial L(x_i, y_i)}{\partial w^{(j)}}$$

## ■ Stochastic gradient descent:

Iterate until convergence:

- For  $i = 1 \dots n$ 
  - For  $j = 1 \dots d$ 
    - Compute:  $\nabla f^{(j)}(x_i)$
    - Update:  $w^{(j)} \leftarrow w^{(j)} - \eta \nabla f^{(j)}(x_i)$

We just had:

$$\nabla f^{(j)} = w^{(j)} + C \sum_{i=1}^n \frac{\partial L(x_i, y_i)}{\partial w^{(j)}}$$

Notice: no summation over  $i$  anymore

# **Support Vector Machines: Example**

# Example: Text categorization

- Example by Leon Bottou:

- Reuters RCV1 document corpus
  - Predict a category of a document
    - One vs. the rest classification
- $n = 781,000$  training examples (documents)
- 23,000 test examples
- $d = 50,000$  features
  - One feature per word
  - Remove stop-words
  - Remove low frequency words

# Example: Text categorization

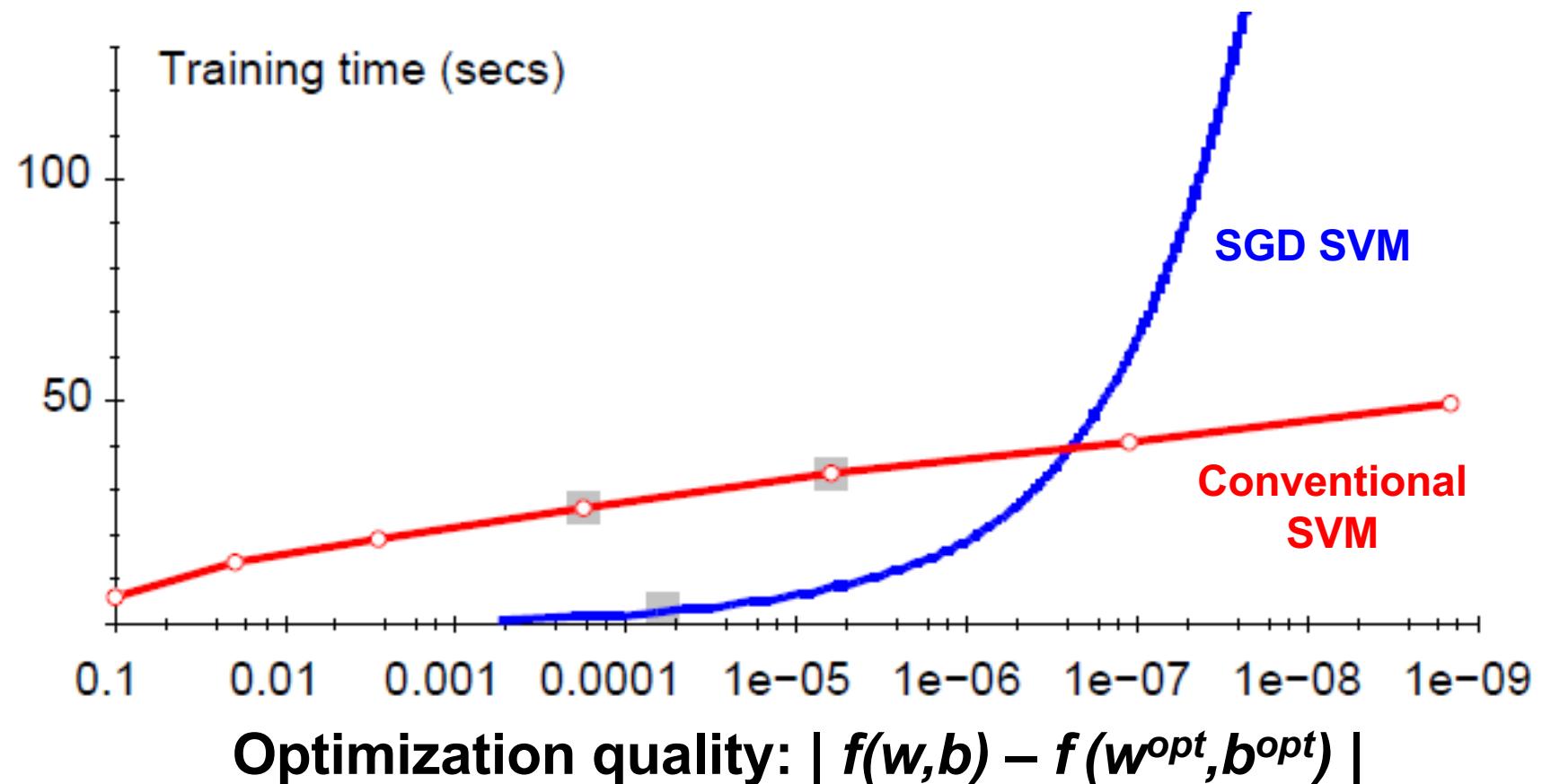
## ■ Questions:

- (1) Is SGD successful at minimizing  $f(\mathbf{w}, b)$ ?
- (2) How quickly does SGD find the min of  $f(\mathbf{w}, b)$ ?
- (3) What is the error on a test set?

	<i>Training time</i>	<i>Value of <math>f(\mathbf{w}, b)</math></i>	<i>Test error</i>
Standard SVM	23,642 secs	0.2275	6.02%
“Fast SVM”	66 secs	0.2278	6.03%
<b>SGD SVM</b>	1.4 secs	0.2275	6.02%

- (1) SGD-SVM is successful at minimizing the value of  $f(\mathbf{w}, b)$
- (2) SGD-SVM is super fast
- (3) SGD-SVM test set error is comparable

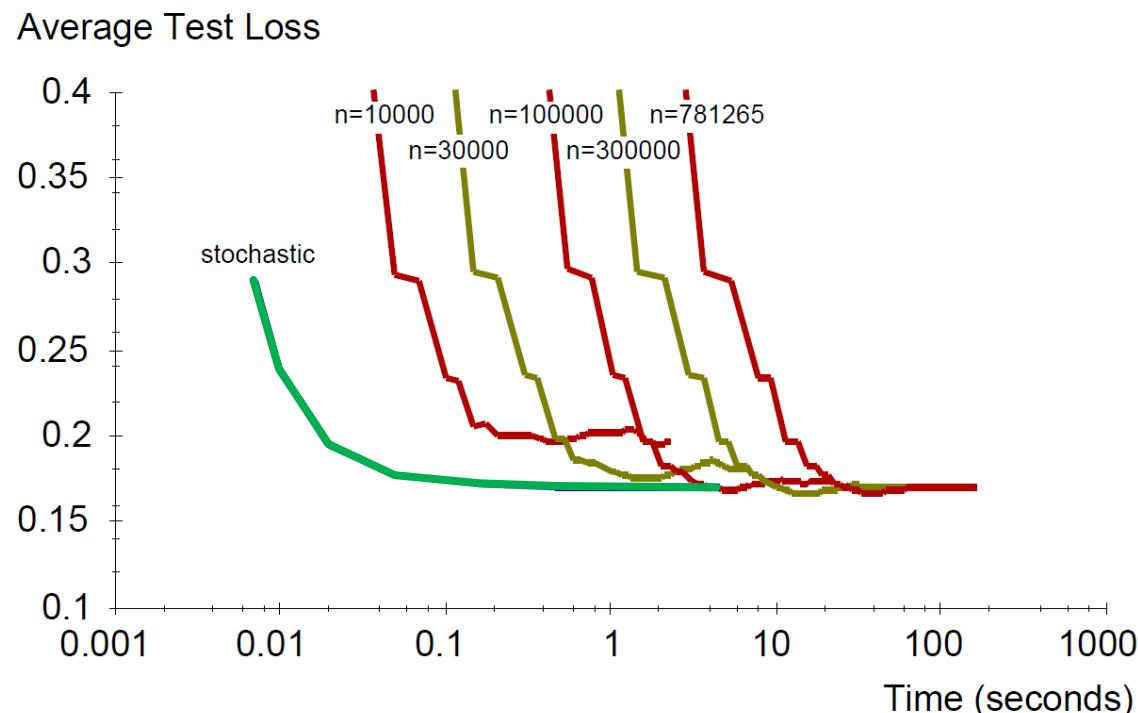
# Optimization “Accuracy”



For optimizing  $f(w,b)$  *within reasonable* quality  
SGD-SVM is super fast

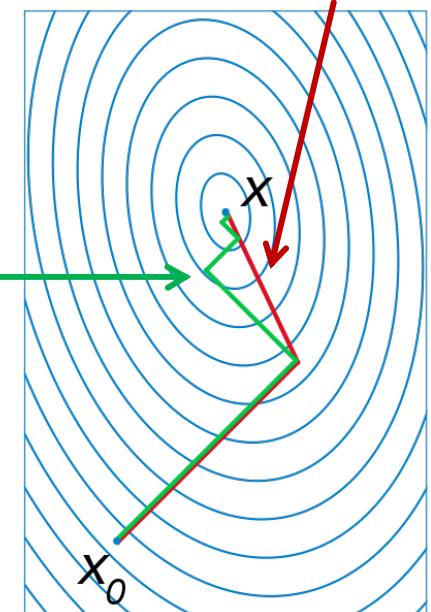
# SGD vs. Batch Conjugate Gradient

- SGD on full dataset vs. Conjugate Gradient on a sample of  $n$  training examples



**Bottom line:** Doing a simple (but fast) SGD update many times is better than doing a complicated (but slow) CG update a few times

Theory says: Gradient descent converges in linear time  $k$ . Conjugate gradient converges in  $\sqrt{k}$ .



$k$ ... condition number

# Practical Considerations

- Need to choose learning rate  $\eta$  and  $t_0$

$$w_{t+1} \leftarrow w_t - \frac{\eta_t}{t + t_0} \left( w_t + C \frac{\partial L(x_i, y_i)}{\partial w} \right)$$

- Leon suggests:

- Choose  $t_0$  so that the expected initial updates are comparable with the expected size of the weights
- Choose  $\eta$ :
  - Select a **small subsample**
  - Try various rates  $\eta$  (e.g., 10, 1, 0.1, 0.01, ...)
  - Pick the one that most reduces the cost
  - Use  $\eta$  for next 100k iterations on the full dataset

# Practical Considerations

## ■ Sparse Linear SVM:

- Feature vector  $x_i$  is sparse (contains many zeros)
  - Do not do:  $x_i = [0, 0, 0, 1, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, \dots]$
  - But represent  $x_i$  as a sparse vector  $x_i = [(4, 1), (9, 5), \dots]$
- Can we do the SGD update more efficiently?

$$w \leftarrow w - \eta \left( w + C \frac{\nabla L}{\nabla L(x_i, y_i)} \right)$$

## ■ Approximated in 2 steps:

$$w \leftarrow w - \eta C \frac{\partial L(x_i, y_i)}{\partial w}$$

$$w \leftarrow w(1 - \eta)$$

**cheap:**  $x_i$  is sparse and so few coordinates  $j$  of  $w$  will be updated  
**expensive:**  $w$  is not sparse, all coordinates need to be updated

# Practical Considerations

## ■ Solution 1: $w = s \cdot v$

- Represent vector  $w$  as the product of scalar  $s$  and vector  $v$
- Then the update procedure is:

$$\text{■ (1)} \quad v = v - \eta C \frac{\partial L(x_i, y_i)}{\partial w}$$

$$\text{■ (2)} \quad s = s(1 - \eta)$$

## ■ Solution 2:

- Perform only step (1) for each training example
- Perform step (2) with lower frequency and higher  $\eta$

Two step update procedure:

$$(1) \quad w \leftarrow w - \eta C \frac{\partial L(x_i, y_i)}{\partial w}$$

$$(2) \quad w \leftarrow w(1 - \eta)$$

# Practical Considerations

## ■ Stopping criteria:

### How many iterations of SGD?

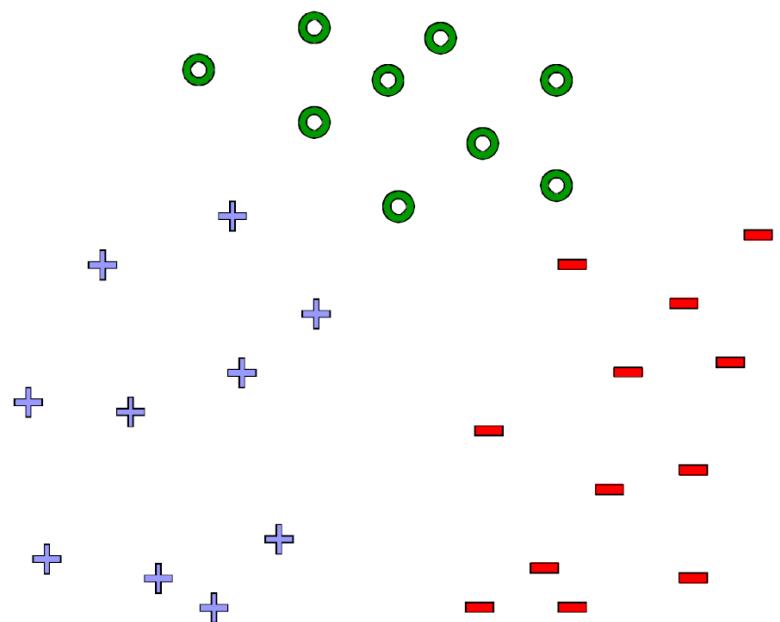
- Early stopping with cross validation

- Create a validation set
  - Monitor cost function on the validation set
  - Stop when loss stops decreasing

- Early stopping

- Extract two disjoint subsamples **A** and **B** of training data
  - Train on **A**, stop by validating on **B**
  - Number of epochs is an estimate of  $k$
  - Train for  $k$  epochs on the full dataset

# What about multiple classes?



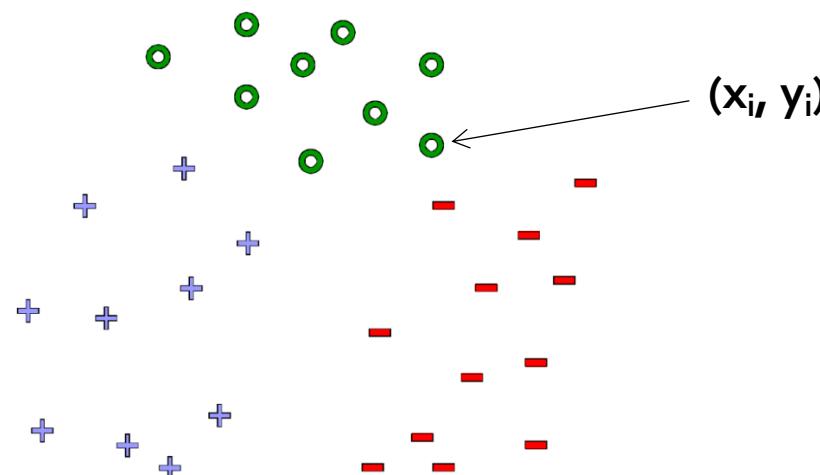
- **Idea 1:**  
**One against all**  
Learn 3 classifiers
  - + vs. {o, -}
  - - vs. {o, +}
  - o vs. {+, -}Obtain:  
 $w_+ b_+$ ,  $w_- b_-$ ,  $w_o b_o$
- **How to classify?**
- Return class  $c$   
 $\arg \max_c w_c x + b_c$

# Learn 1 classifier: Multiclass SVM

## ■ Idea 2: Learn 3 sets of weights simultaneously!

- For each class  $c$  estimate  $w_c, b_c$
- Want the correct class to have highest margin:

$$w_{y_i} x_i + b_{y_i} \geq 1 + w_c x_i + b_c \quad \forall c \neq y_i, \forall i$$



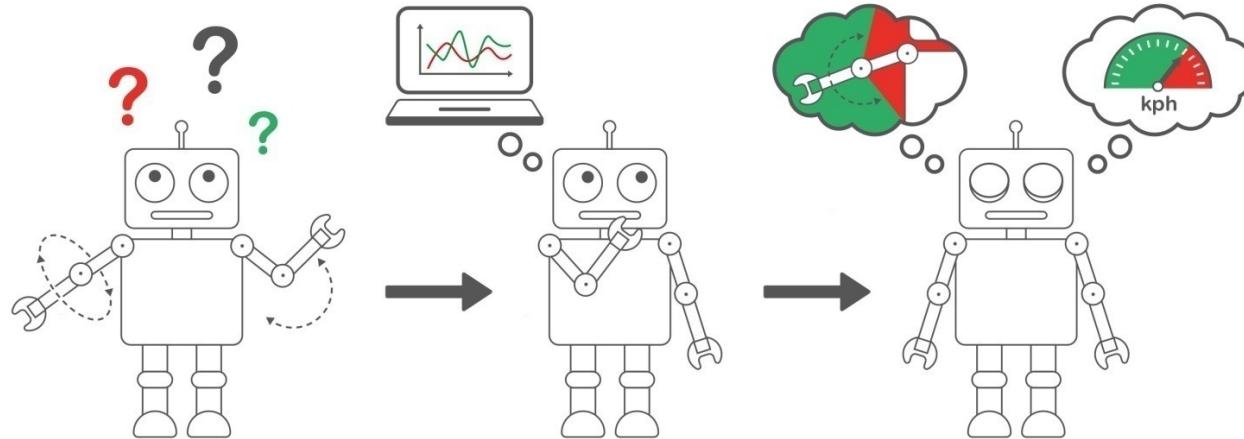
# Multiclass SVM

- Optimization problem:

$$\min_{w,b} \frac{1}{2} \sum_c \|w_c\|^2 + C \sum_{i=1}^n \xi_i \quad \forall c \neq y_i, \forall i$$

$$w_{y_i} \cdot x_i + b_{y_i} \geq w_c \cdot x_i + b_c + 1 - \xi_i \quad \xi_i \geq 0, \forall i$$

- To obtain parameters  $w_c, b_c$  (for each class  $c$ ) we can use similar techniques as for 2 class SVM
- SVM is widely perceived a very powerful learning algorithm



# Introduction to Machine Learning and Deep Learning



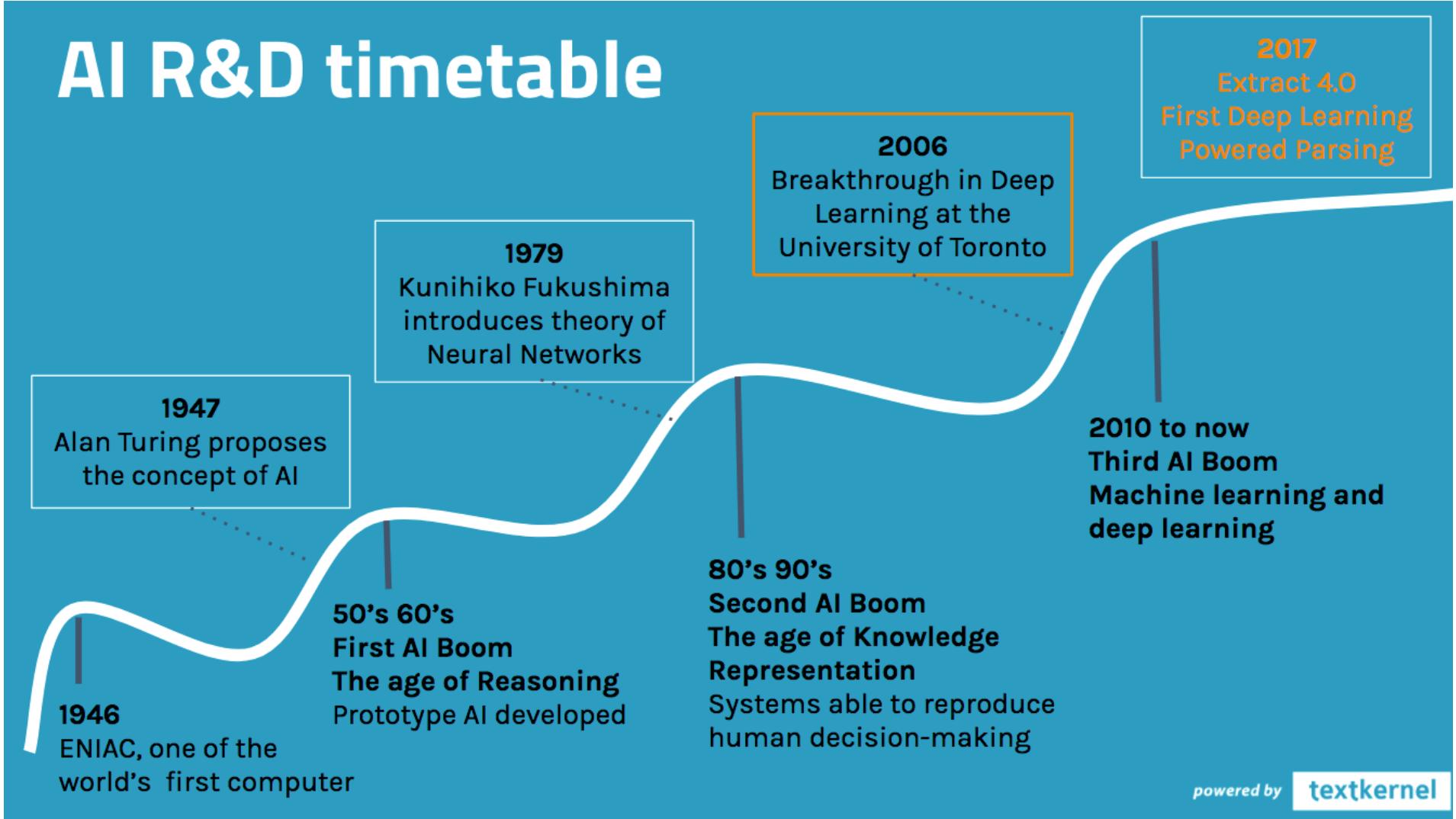
**Dr. Anand Kumar M**  
Assistant Professor-I,  
Department of Information Technology  
National Institute of Technology Karnataka  
Surathkal  
[m\\_anandkumar@nitk.edu.in](mailto:m_anandkumar@nitk.edu.in)

# Outline & Content

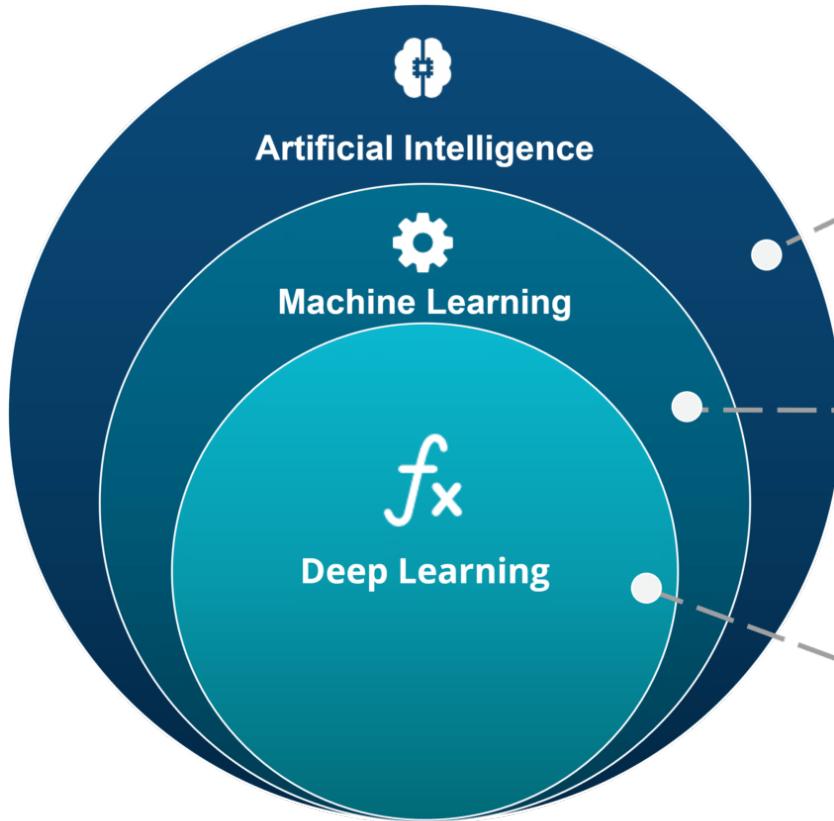
---

- AI/ML/DL
- Machine Learning
- Deep learning?
- Why Deep Learning
- Applications
- Conclusion

# AI R&D timetable



powered by  textkernel



## ARTIFICIAL INTELLIGENCE

A technique which enables machine to mimic human behaviour

## MACHINE LEARNING

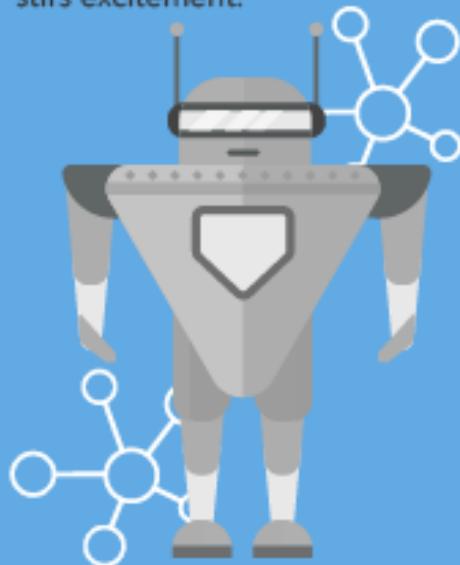
Subset of AI technique which uses statistical methods to enable machine to improve with experience

## DEEP LEARNING

Subset of ML which make the computation of multi-layer neural network feasible

# ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



1950's

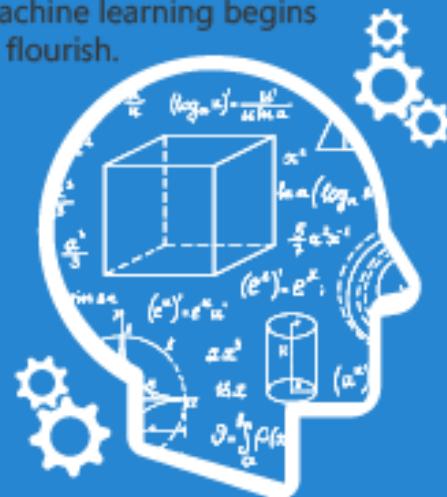
1960's

1970's

1980's

# MACHINE LEARNING

Machine learning begins to flourish.



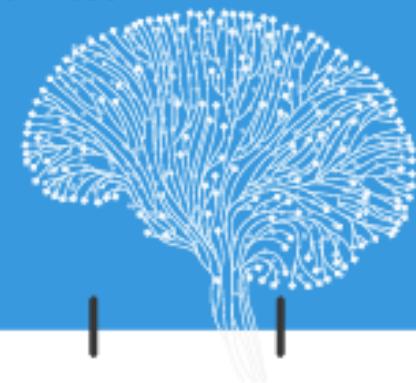
1990's

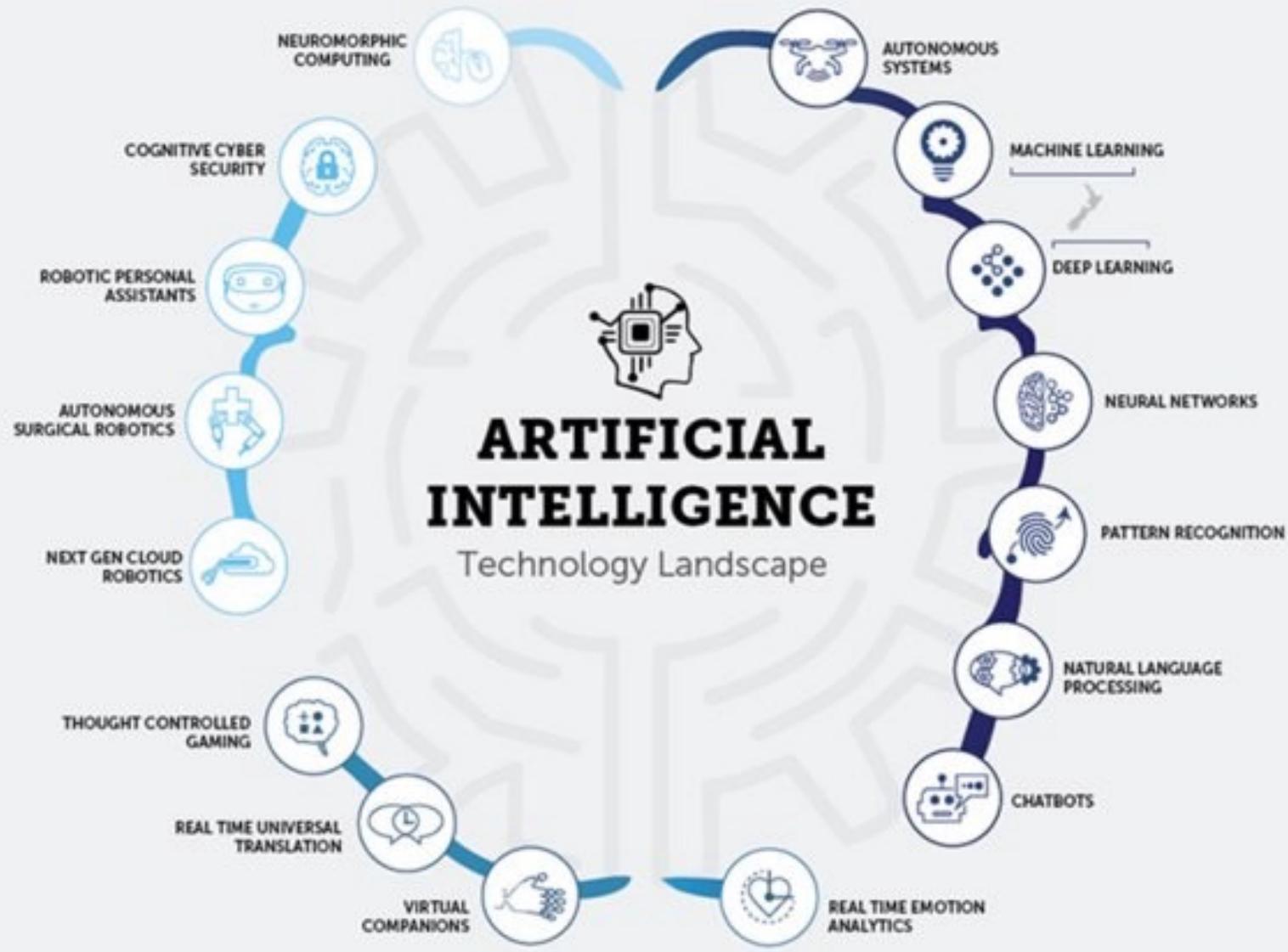
2000's

2010's

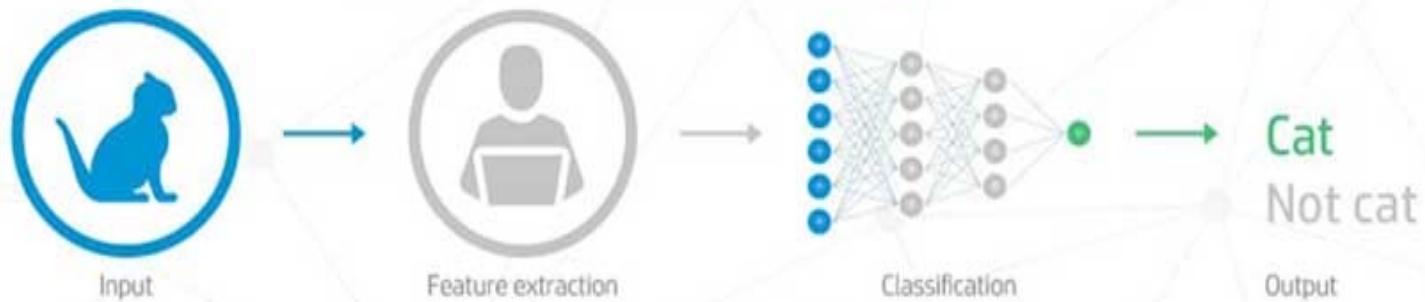
# DEEP LEARNING

Deep learning breakthroughs drive AI boom.

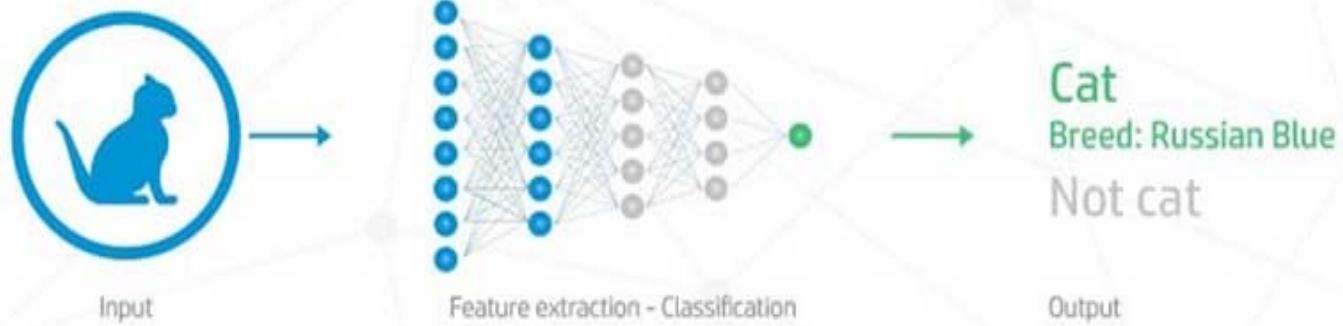




## Machine learning



## Deep learning



# ML vs DL

## Machine Learning

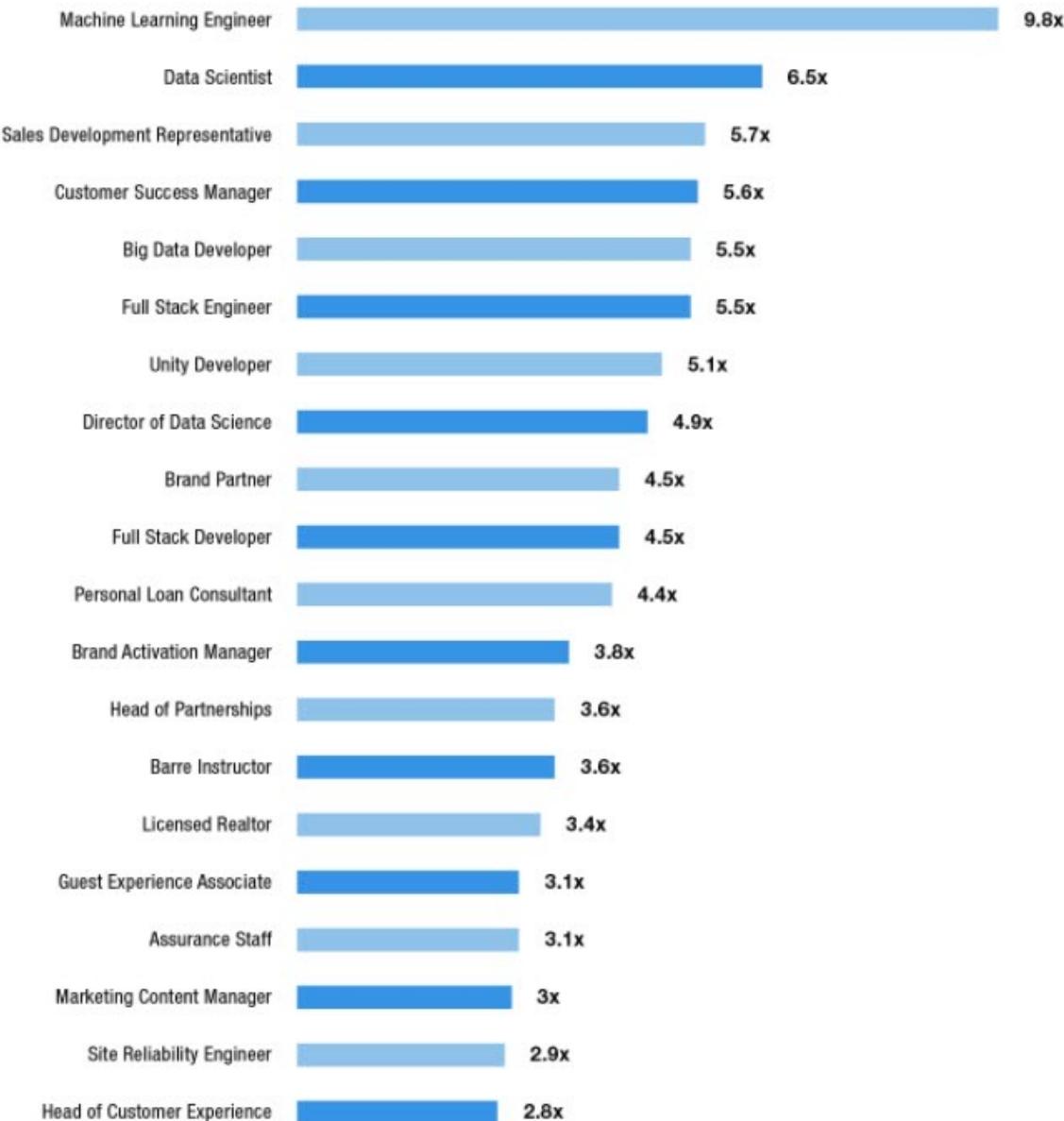
- ⊕ Good results with small data sets
- ⊕ Quick to train a model
- ⊖ Need to try different features and classifiers to achieve best results
- ⊖ Accuracy plateaus

## Deep Learning

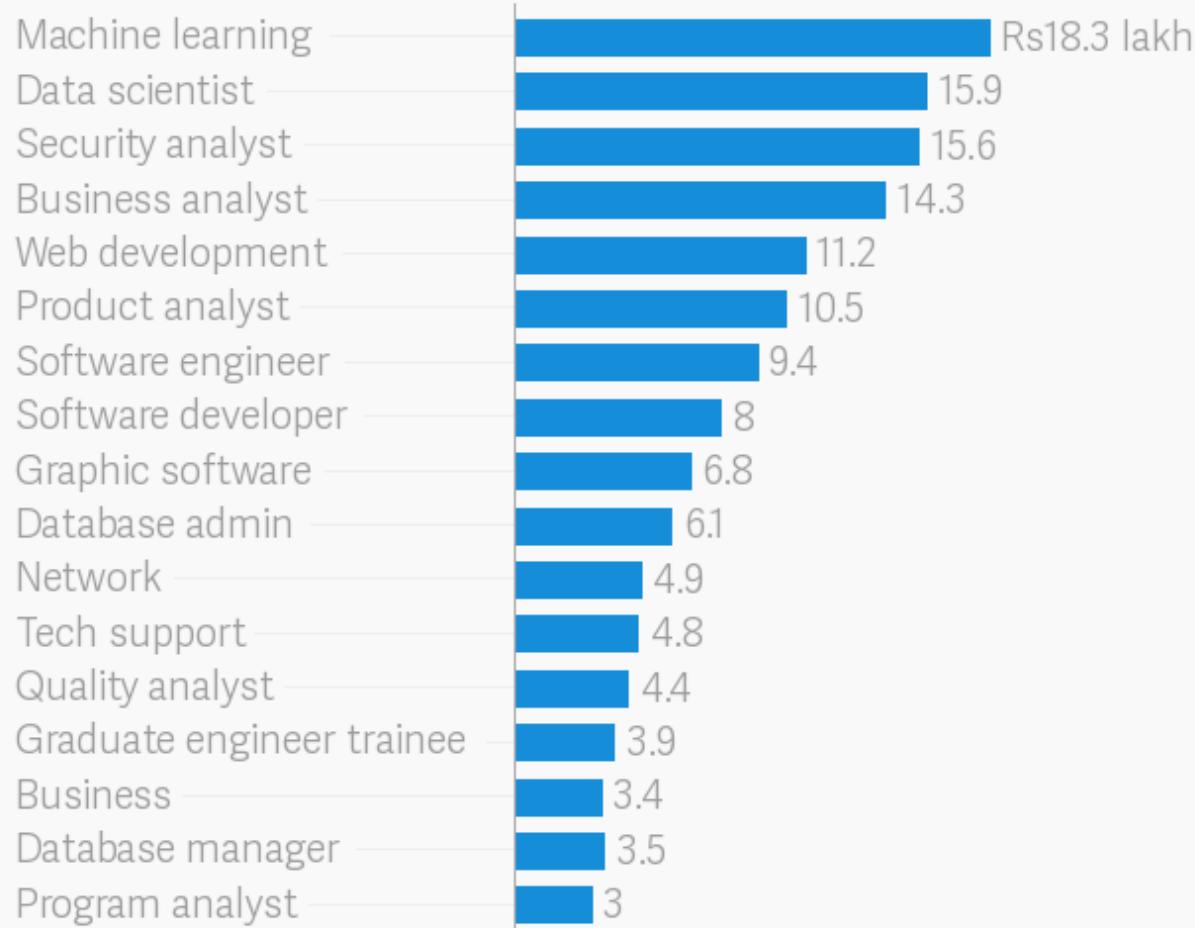
- ⊖ Requires very large data sets
- ⊖ Computationally intensive
- ⊕ Learns features and classifiers automatically
- ⊕ Accuracy is unlimited

## Top 20 Emerging Jobs

LinkedIn Economic Graph



## Top job roles in CS/IT in India



DATA

| Data: Mettl Campus Hiring Report 2018



COMMITTED TO  
IMPROVING THE STATE  
OF THE WORLD

# The Jobs Landscape in 2022

emerging  
roles,  
global  
change  
by 2022

133  
Million

declining  
roles,  
global  
change  
by 2022

75  
Million

## Top 10 Emerging

1. Data Analysts and Scientists
2. AI and Machine Learning Specialists
3. General and Operations Managers
4. Software and Applications Developers and Analysts
5. Sales and Marketing Professionals
6. Big Data Specialists
7. Digital Transformation Specialists
8. New Technology Specialists
9. Organisational Development Specialists
10. Information Technology Services

## Top 10 Declining

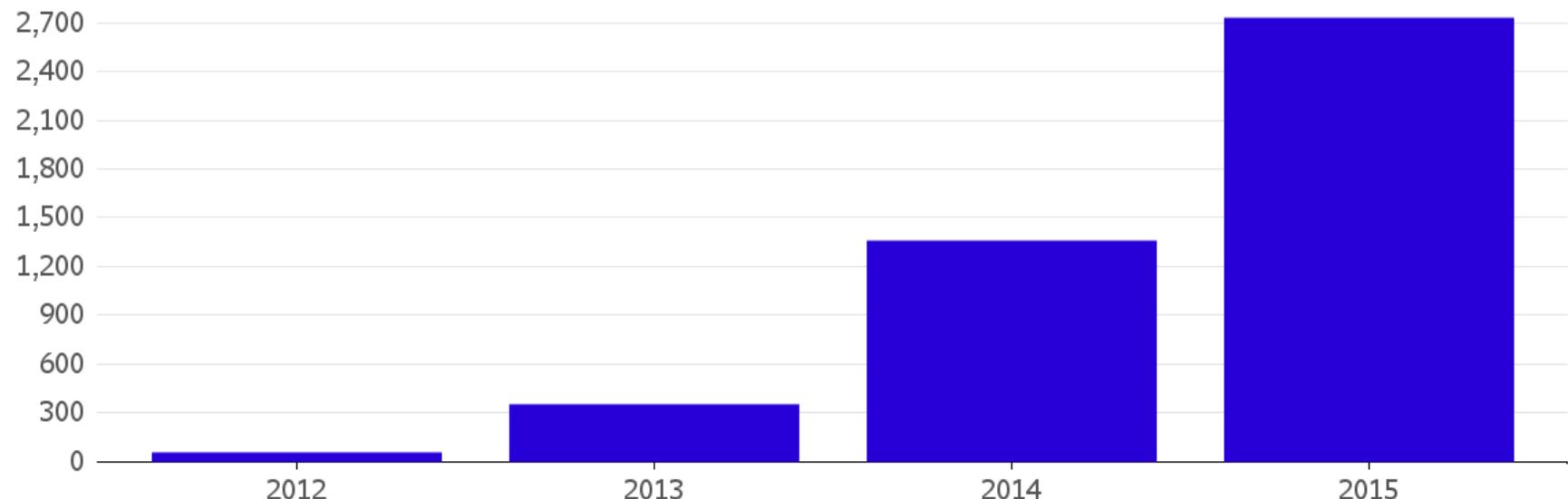
1. Data Entry Clerks
2. Accounting, Bookkeeping and Payroll Clerks
3. Administrative and Executive Secretaries
4. Assembly and Factory Workers
5. Client Information and Customer Service Workers
6. Business Services and Administration Managers
7. Accountants and Auditors
8. Material-Recording and Stock-Keeping Clerks
9. General and Operations Managers
10. Postal Service Clerks

Source: Future of Jobs Report 2018, World Economic Forum

# Deep Learning at Google

## Artificial Intelligence Takes Off at Google

Number of software projects within Google that uses a key AI technology, called Deep Learning.

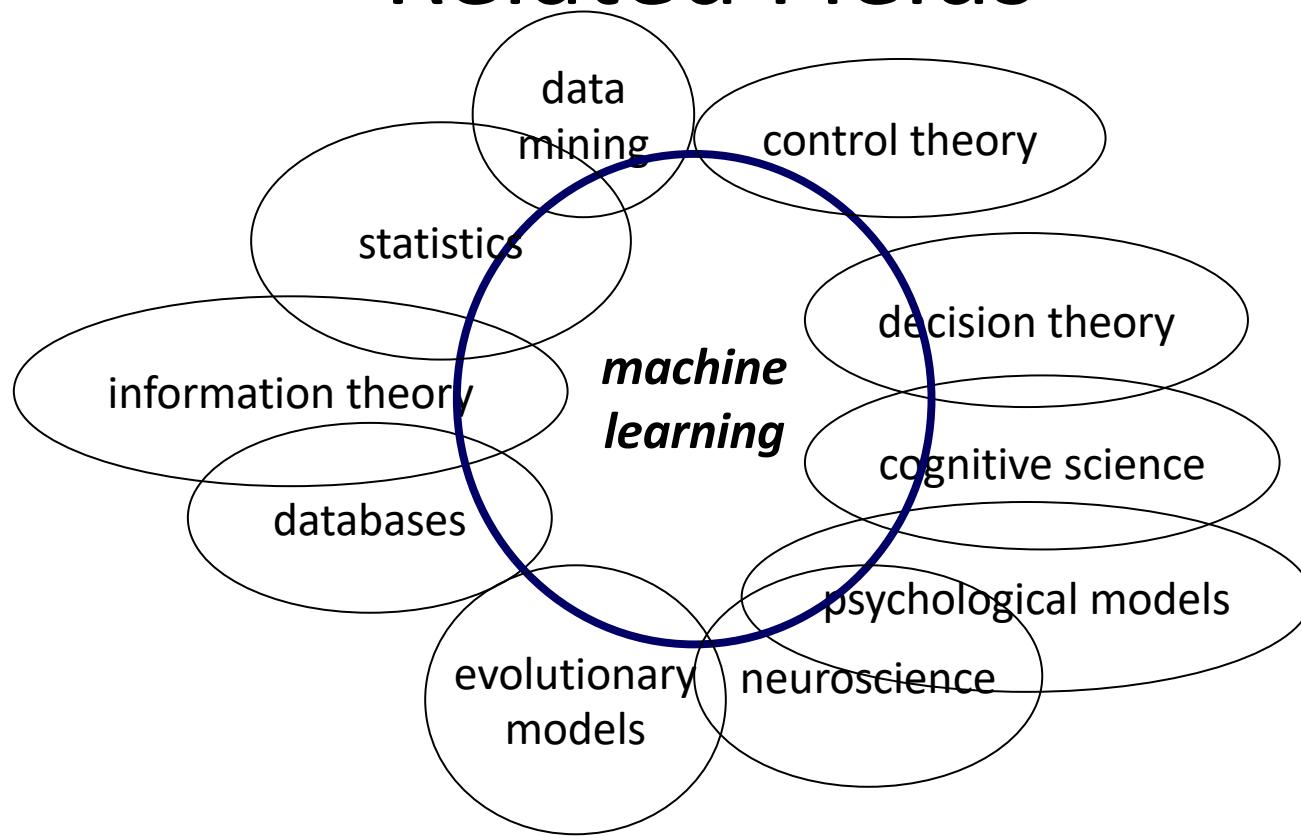


Source: Google

Note: 2015 data does not incorporate data from Q4

Bloomberg

# Related Fields



*Machine learning* is primarily concerned with the accuracy and effectiveness of the *computer system*.

# What is Machine Learning?

- It is very hard to write programs that solve problems like recognizing a face.
  - We don't know what program to write because we don't know how our brain does it.
  - Even if we had a good idea about how to do it, the program might be awfully complicated.
- Instead of writing a program by hand, we **collect lots of examples** that specify the correct output for a given input.
- A machine learning algorithm then takes these examples and produces a program that does the job.
  - The program produced by the **learning algorithm may look very different from a typical hand-written program**. It may contain millions of numbers.
  - If we do it right, the **program works for new cases as well as the ones we trained it on**.

# Machine Learning

- **Herbert Alexander Simon:**  
“Learning is any process by which a system improves performance from experience.”
- “Machine Learning is concerned with computer programs that automatically improve their performance through experience. “

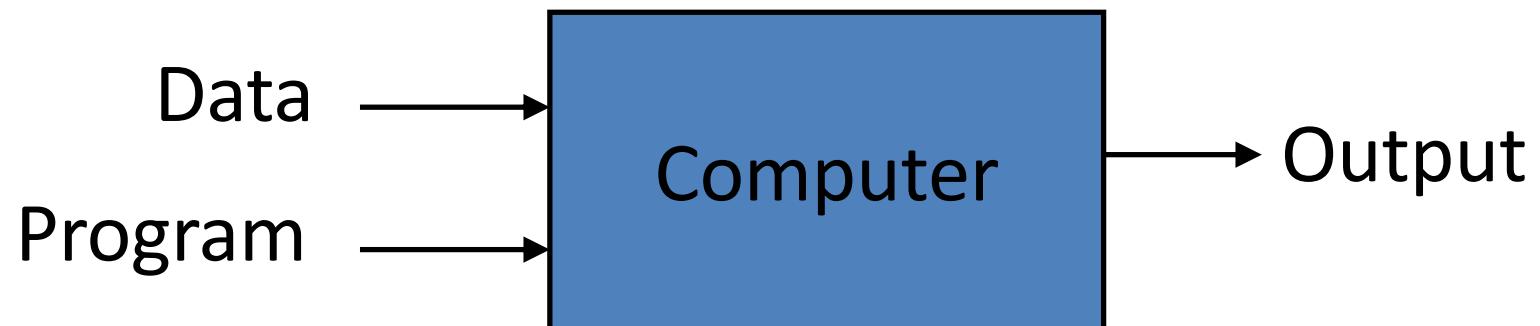


**Herbert Simon**  
[Turing Award 1975](#)  
[Nobel Prize in Economics 1978](#)

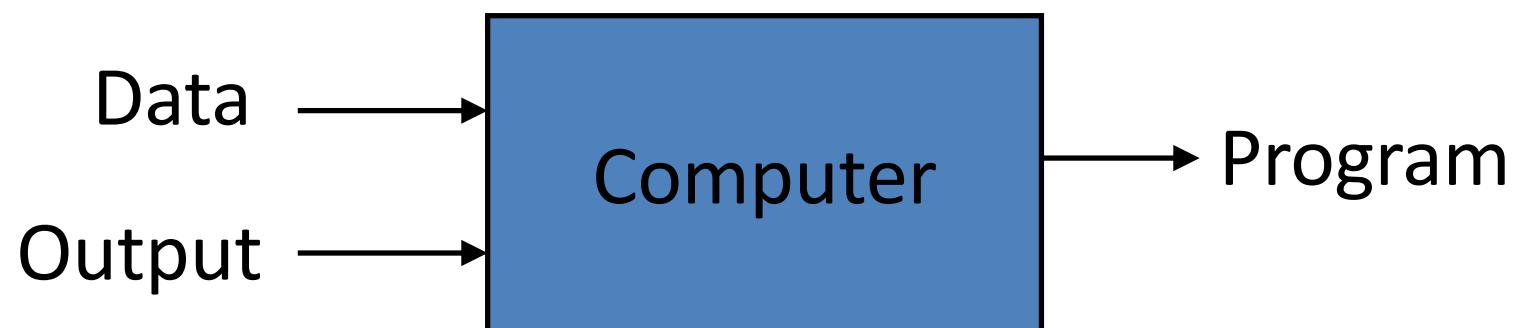
# Why now?

- Flood of available data (especially with the advent of the Internet)
- Increasing computational power
- Growing progress in available algorithms and theory developed by researchers
- Increasing support from industries

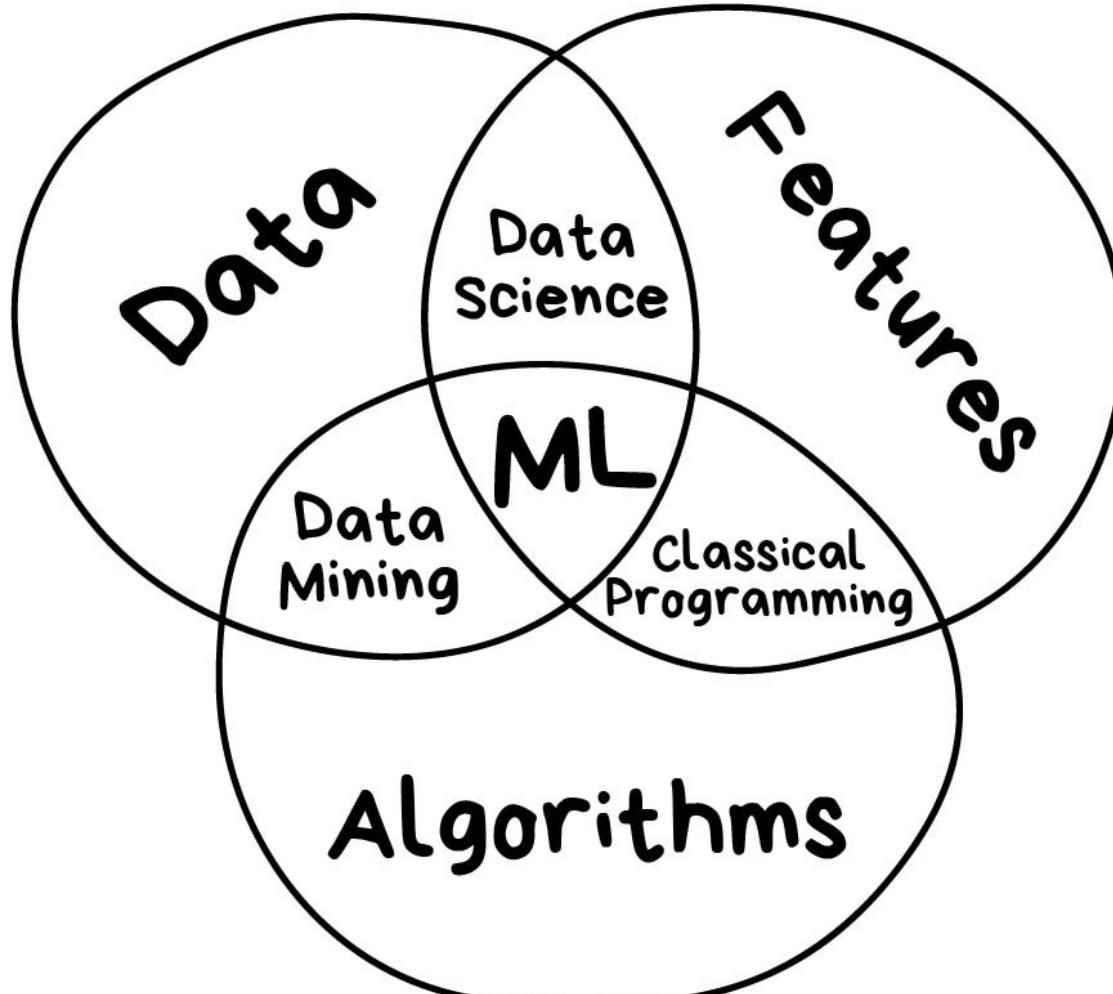
## Traditional Programming



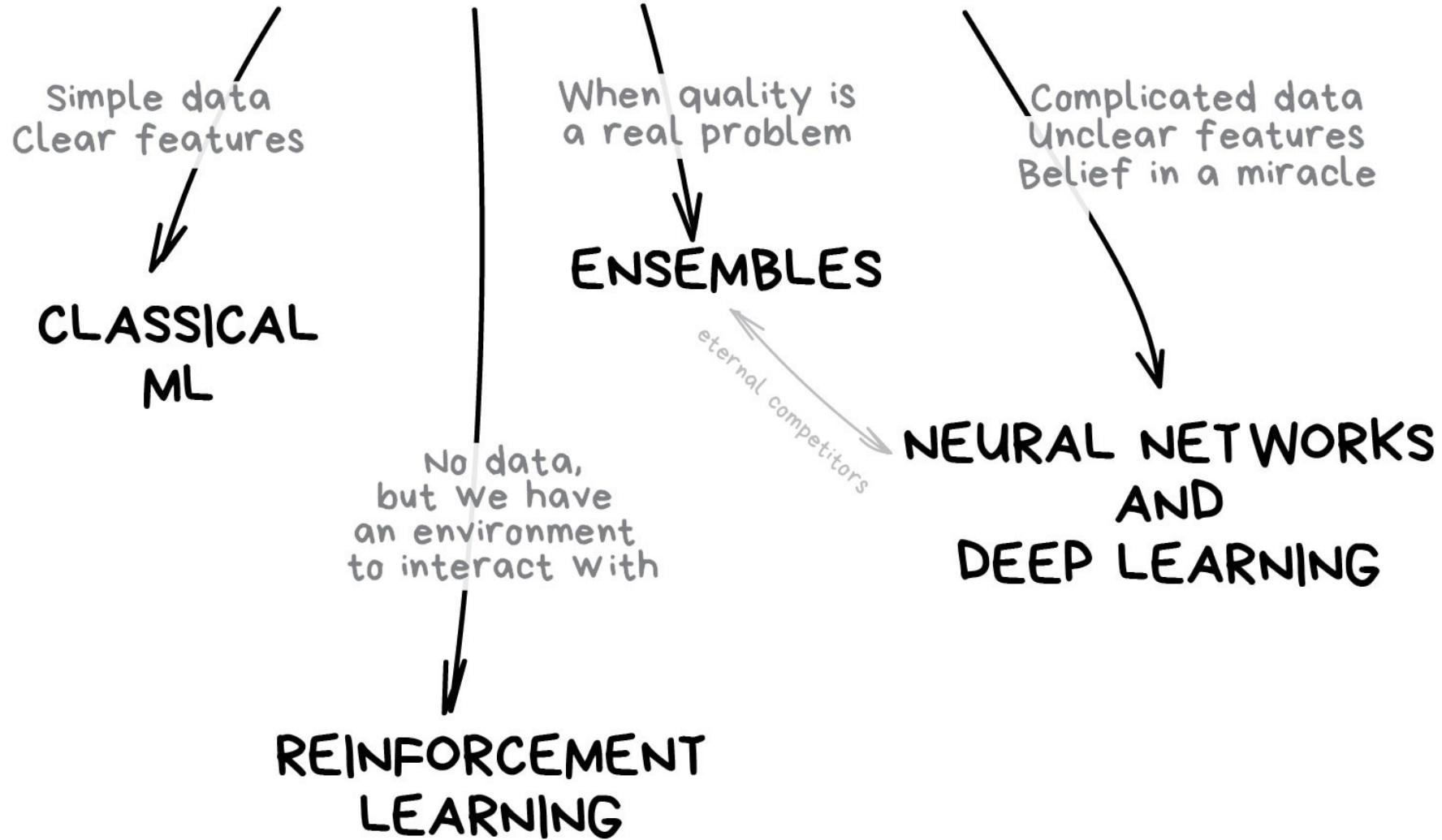
## Machine Learning



# Three components of machine learning



# THE MAIN TYPES OF MACHINE LEARNING



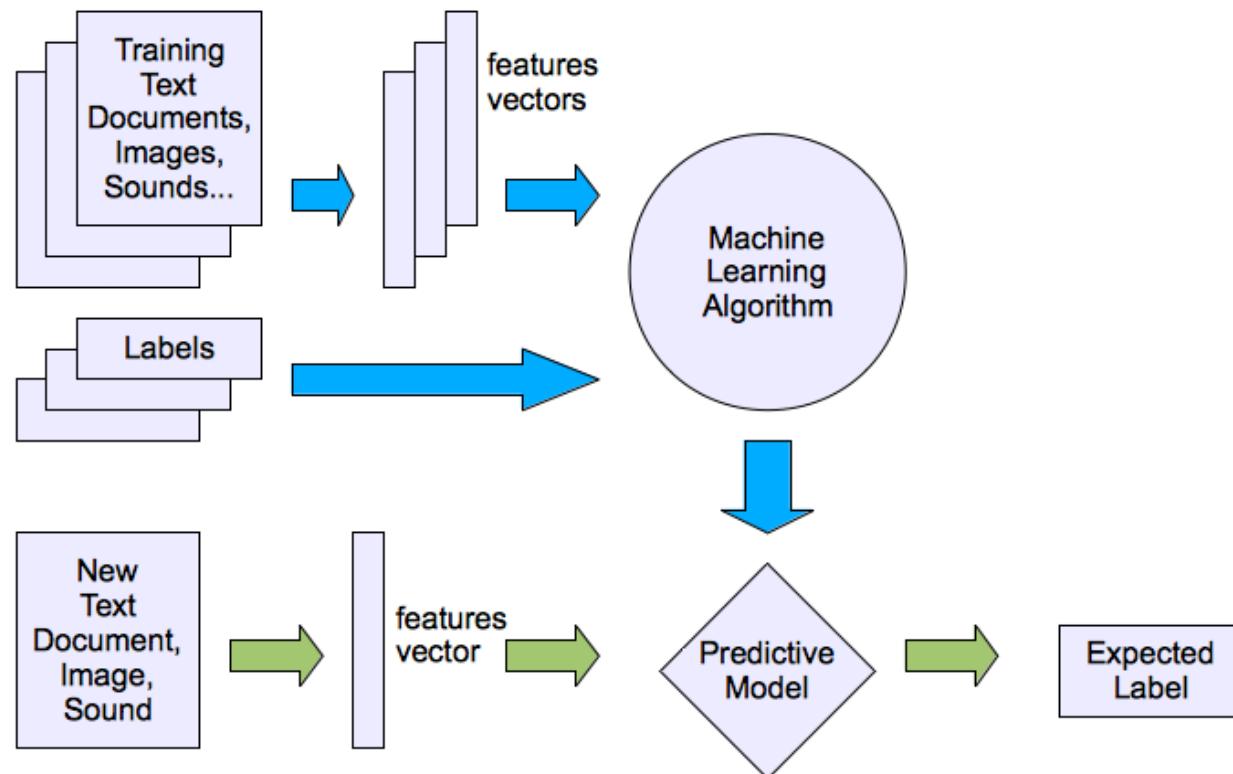
# Data

- Want to detect spam? Get samples of spam messages. Want to forecast stocks? Find the price history. Want to find out user preferences?

# Machine learning structure

---

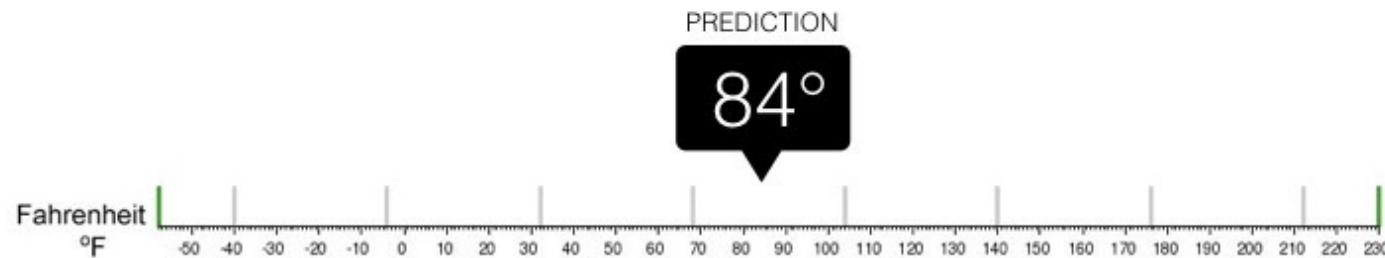
- Supervised learning





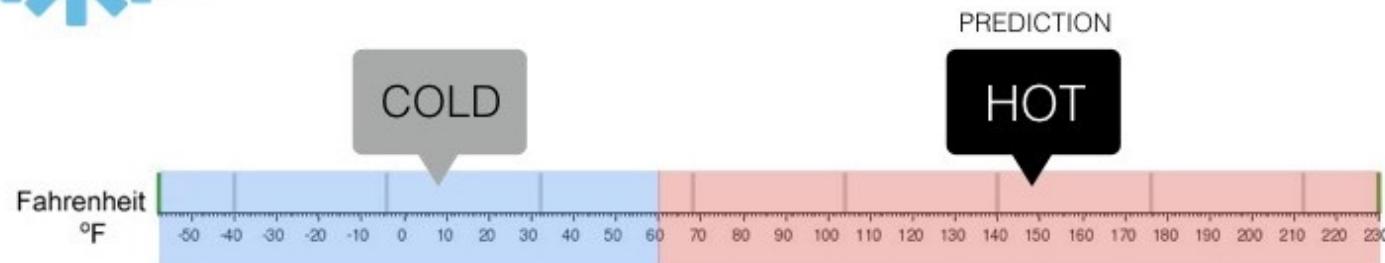
## Regression

What is the temperature going to be tomorrow?

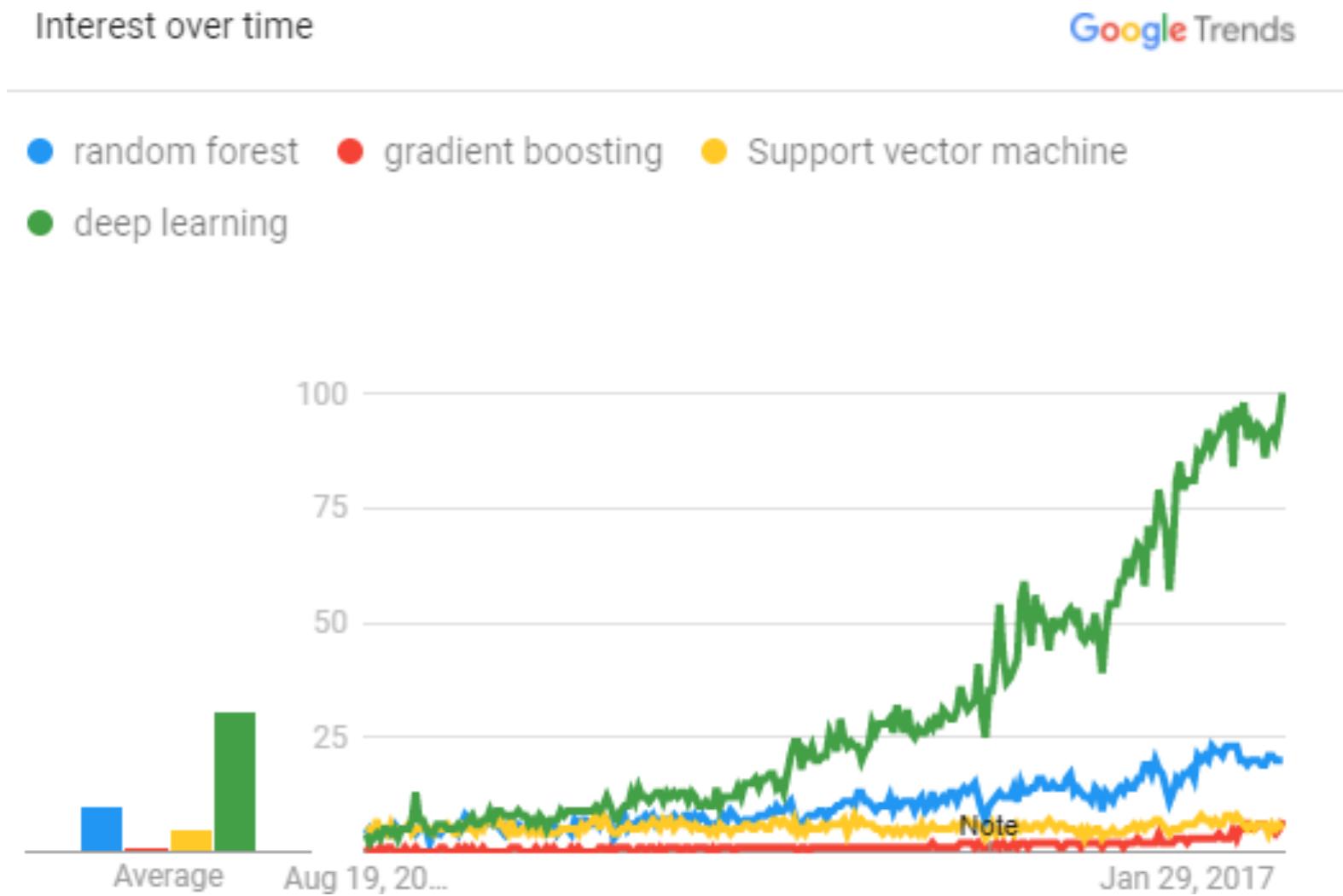


## Classification

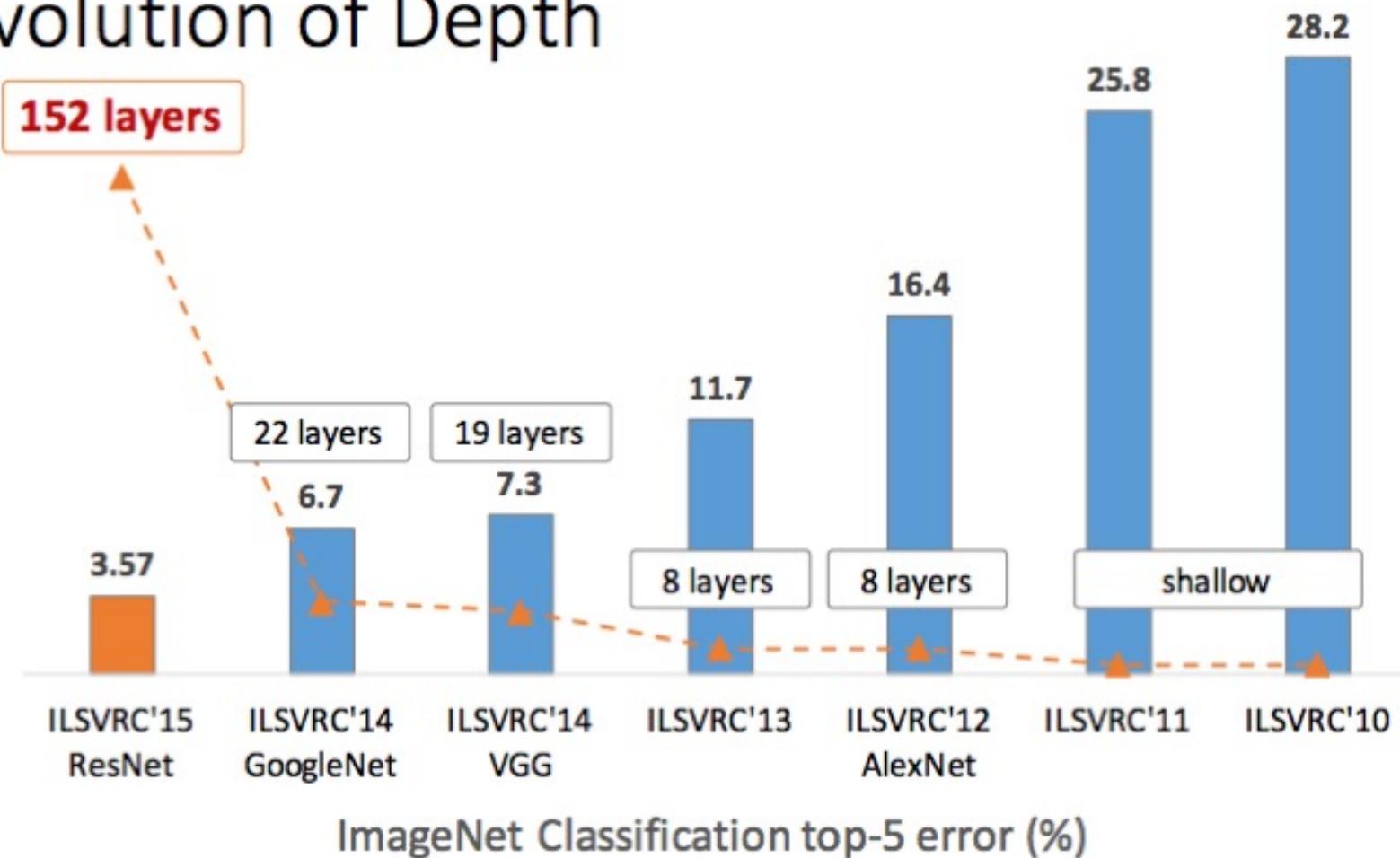
Will it be Cold or Hot tomorrow?



# Google trends



# Revolution of Depth



**So, 1. what exactly is deep learning ?**

**And, 2. why is it generally better than other methods on image, speech and certain other types of data?**

So, 1. what exactly is deep learning ?

And, 2. why is it generally better than other methods on image, speech and certain other types of data?

### The short answers

1. ‘Deep Learning’ means using a neural network with several layers of nodes between input and output
  
2. the series of layers between input & output do feature identification and processing in a series of stages, just as our brains seem to.

hmmm... OK, but:

**3. multilayer neural networks have been around for  
25 years. What's actually new?**

hmmm... OK, but:

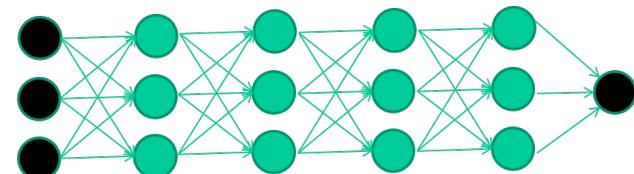
### 3. multilayer neural networks have been around for 25 years. What's actually new?

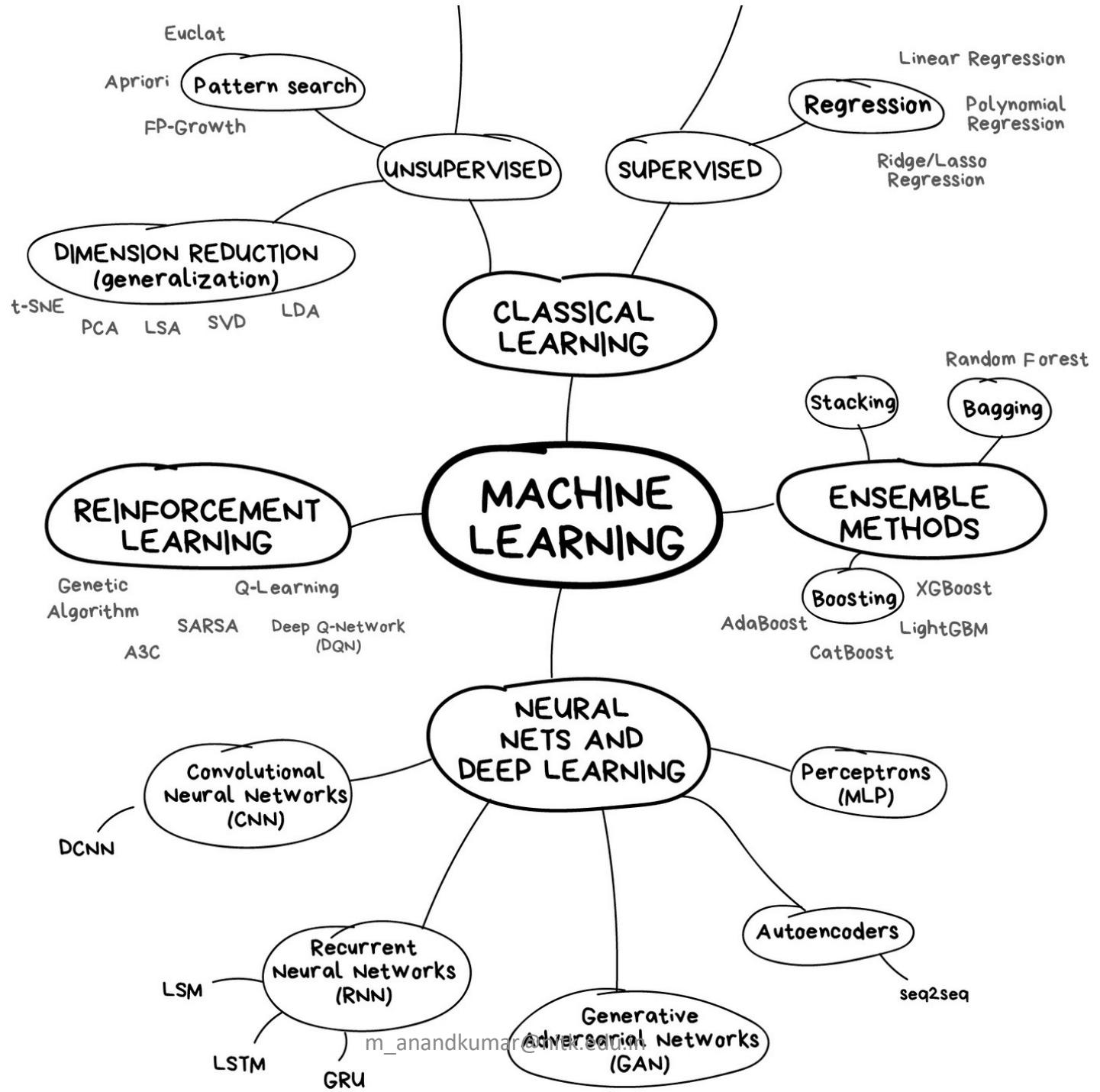
**we have always had good algorithms for learning the weights in networks with 1 hidden layer**



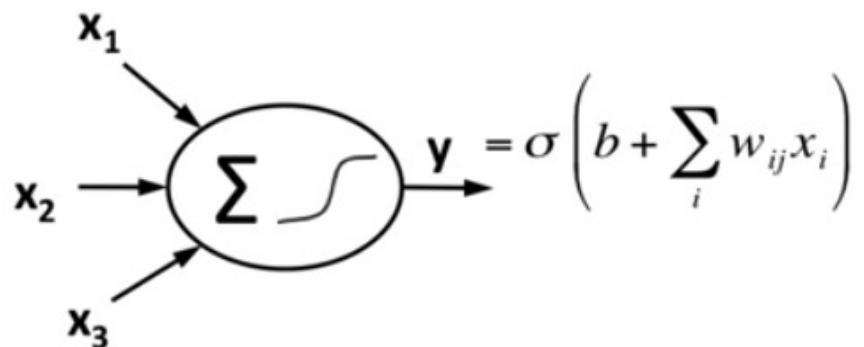
**but these algorithms are not good at learning the weights for networks with more hidden layers**

**what's new is: algorithms for training many-layer networks**

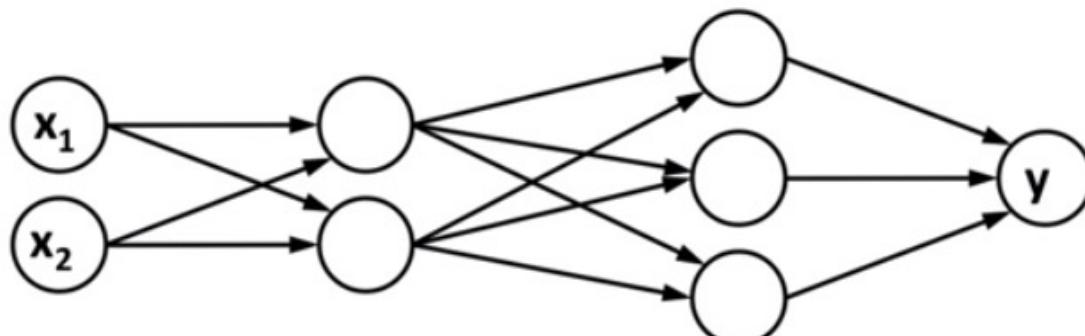




# Types of Neural Networks

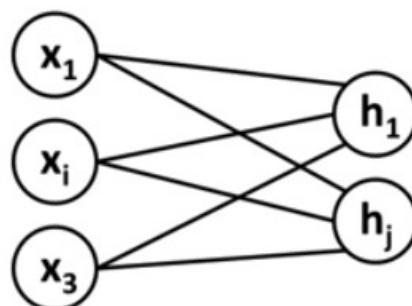
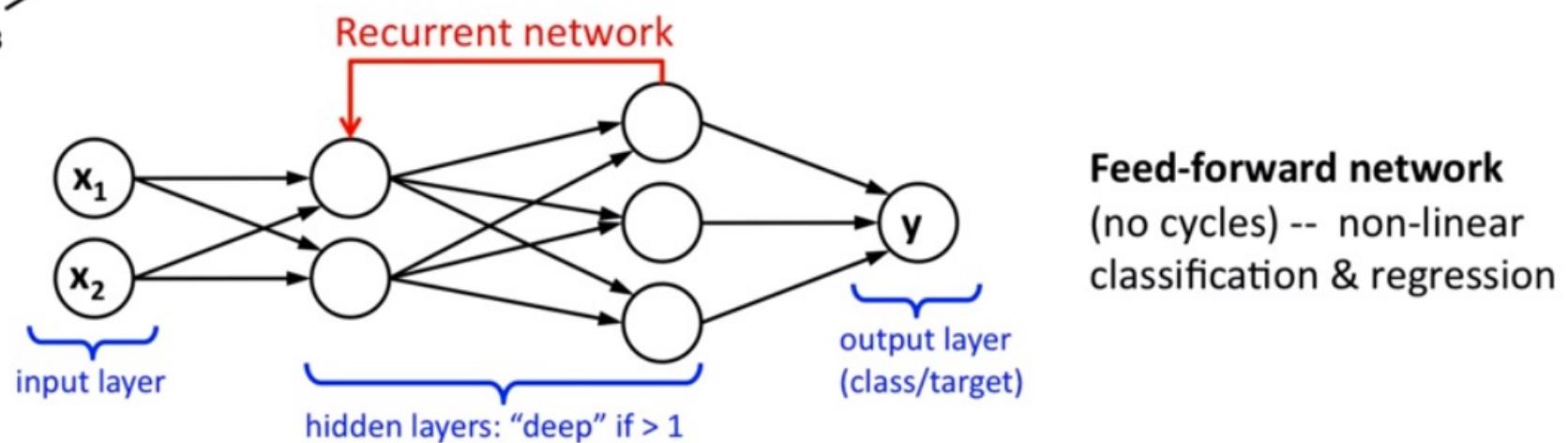
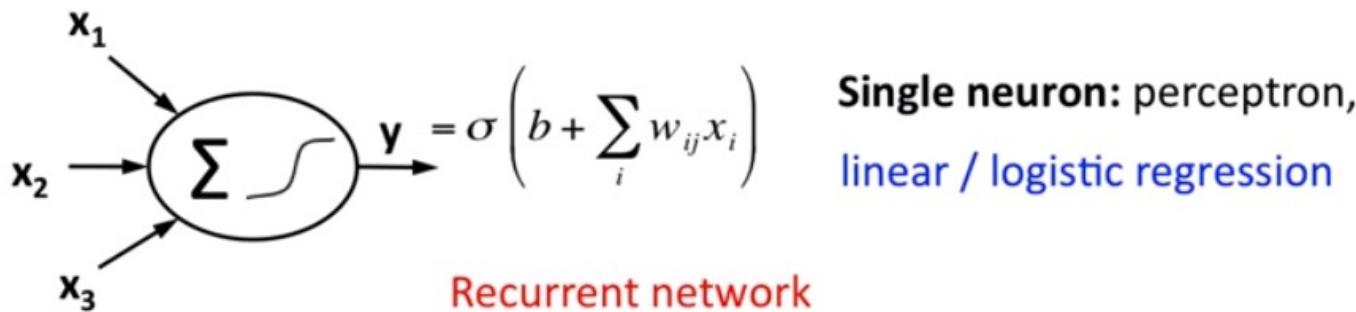


**Single neuron:** perceptron,  
linear / logistic regression



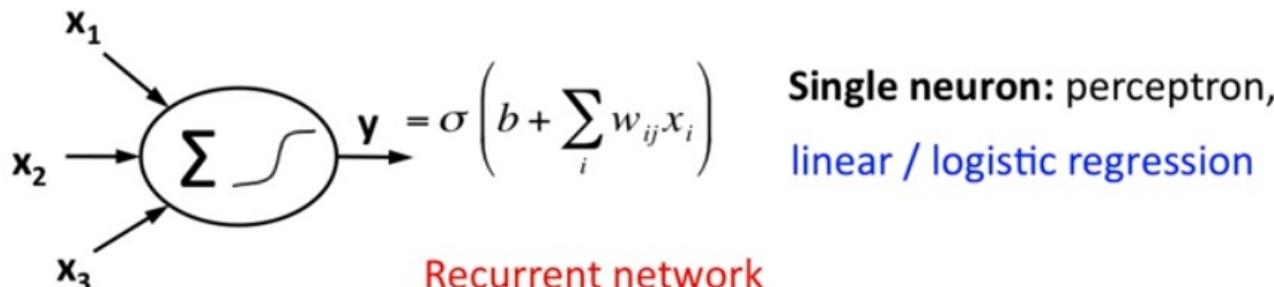
**Feed-forward netwo**  
(no cycles) -- non-lin  
classification & regre

# Types of Neural Networks

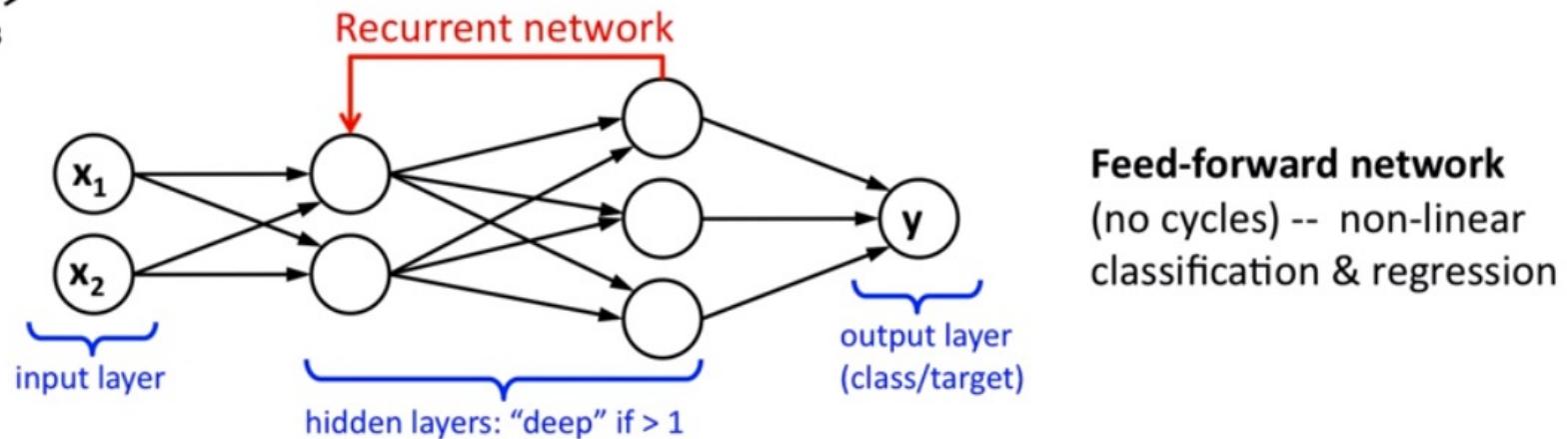


**Symmetric (RBM)**  
unsupervised, trained  
to maximize likelihood  
of input data

# Types of Neural Networks



**Single neuron:** perceptron,  
linear / logistic regression



**Feed-forward network**  
(no cycles) -- non-linear  
classification & regression

$$\sigma\left(\beta_i + \sum_j w_{ij} h_j\right) = \begin{array}{c} x_1 \\ x_i \\ x_3 \end{array} \rightarrow \begin{array}{c} h_1 \\ h_j \end{array}$$

same set of weights

**P (input | hidden)**

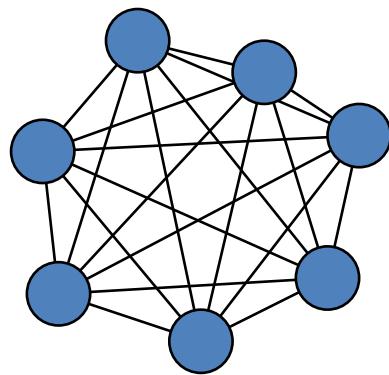
$$\sigma\left(\beta_i + \sum_j w_{ij} h_j\right) =$$

**P (hidden | input)**

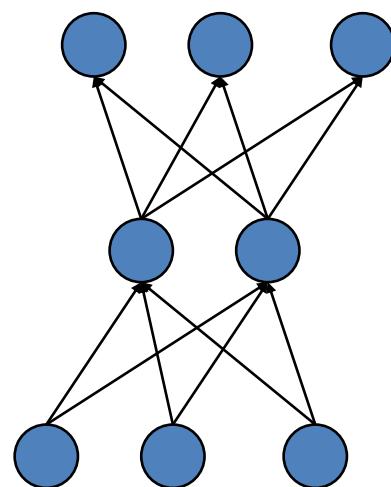
$$\sigma\left(b_j + \sum_i w_{ij} x_i\right) =$$

**Symmetric (RBM)**  
unsupervised, trained  
to maximize likelihood  
of input data

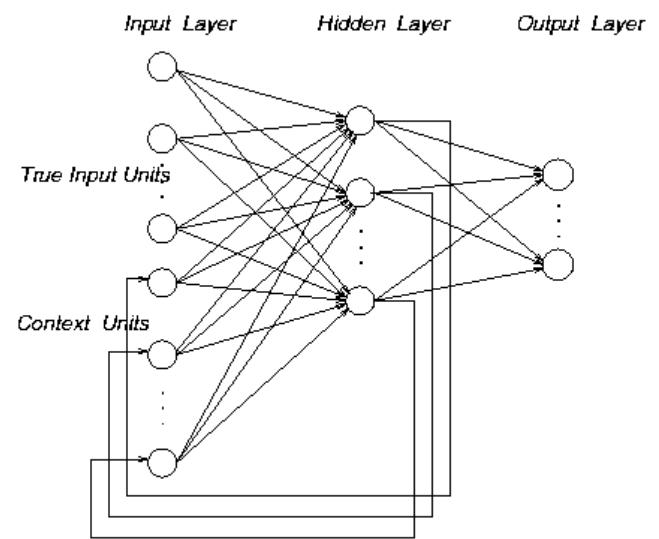
# Topologies of Neural Networks



*completely  
connected*



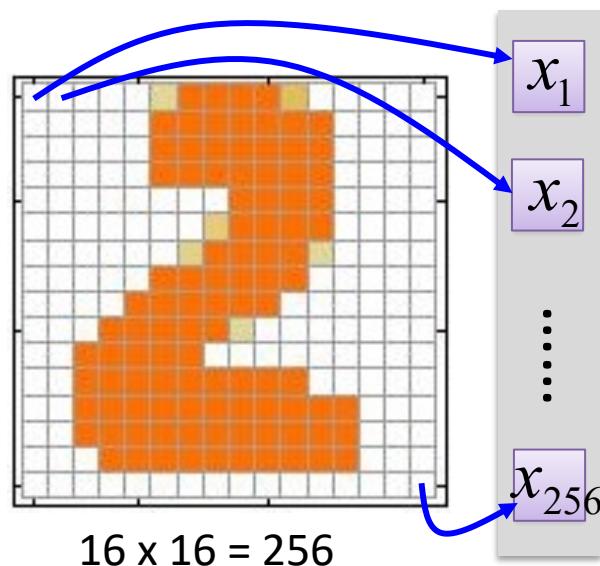
*feedforward  
(directed, a-cyclic)*



*recurrent  
(feedback connections)*

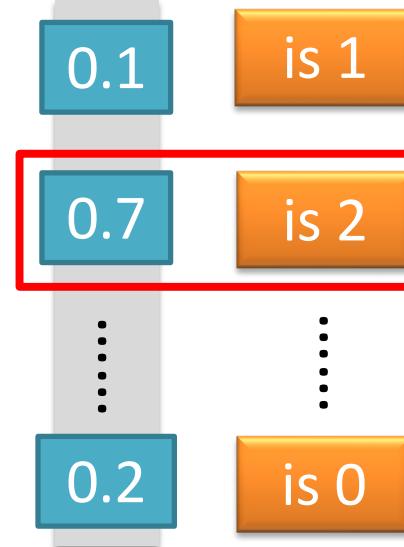
# Handwriting Digit Recognition

**Input**



Ink → 1  
No ink → 0

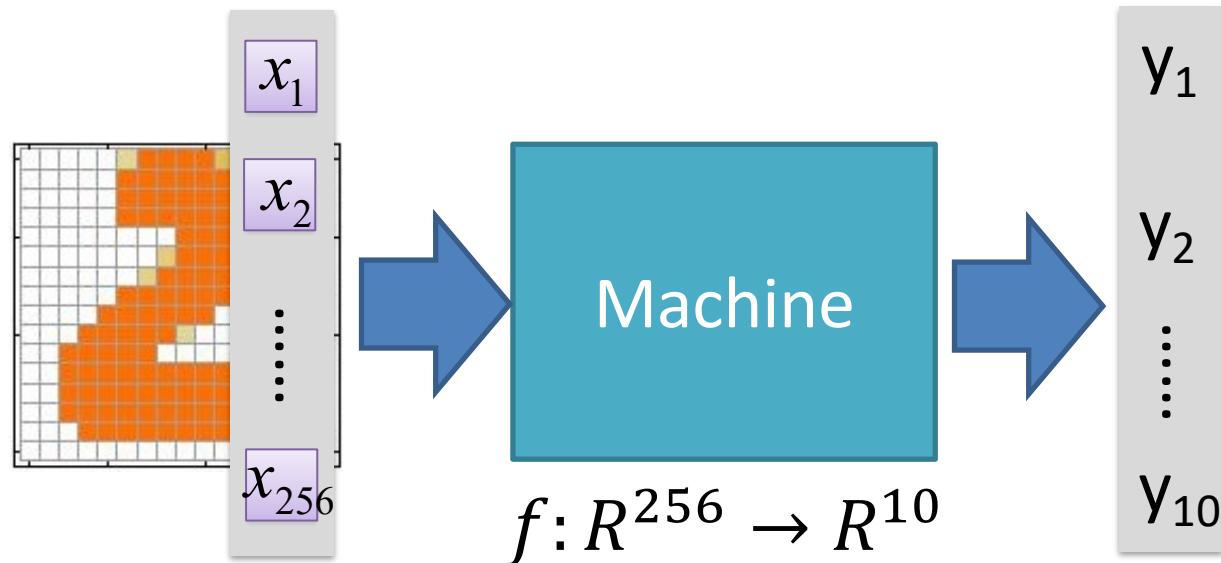
**Output**



Each dimension represents the confidence of a digit.

# Example Application

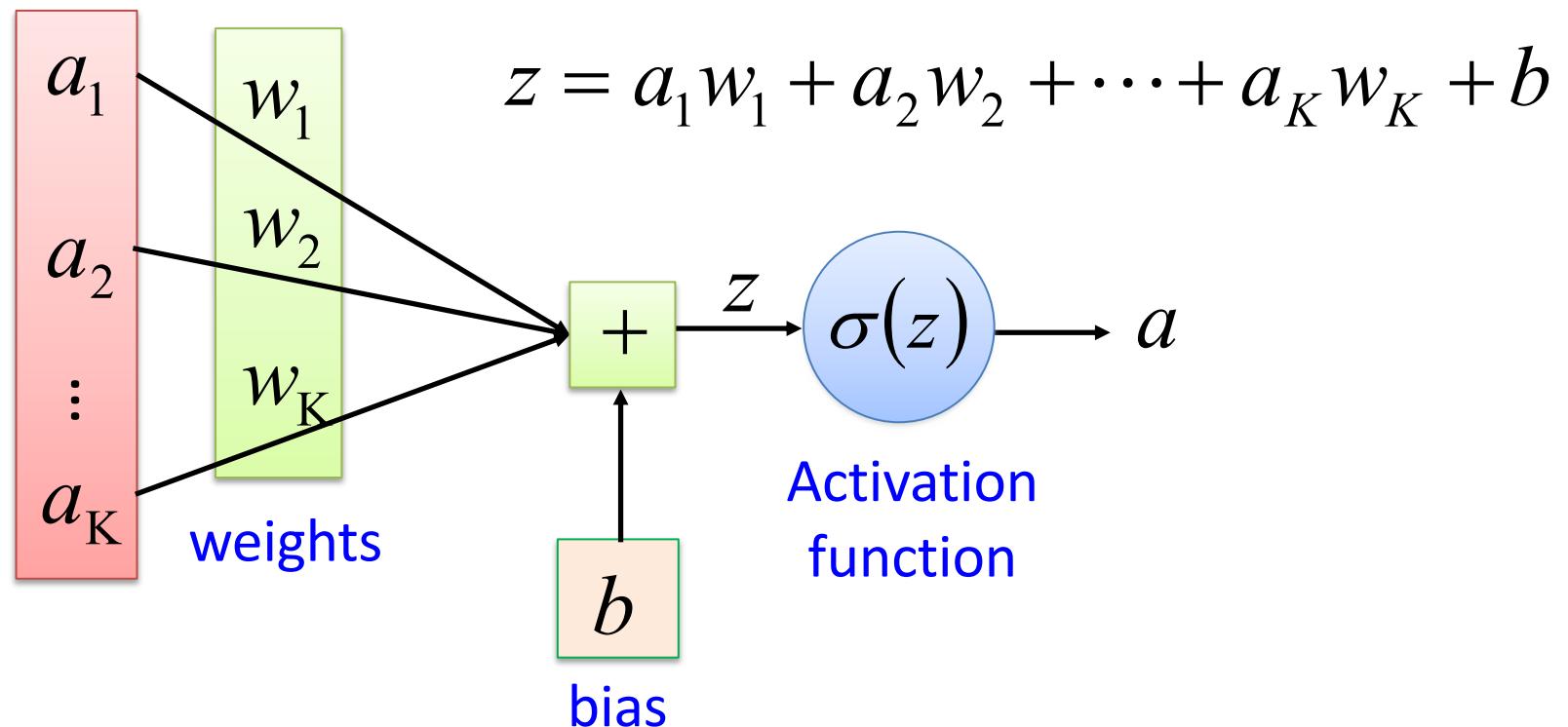
- Handwriting Digit Recognition



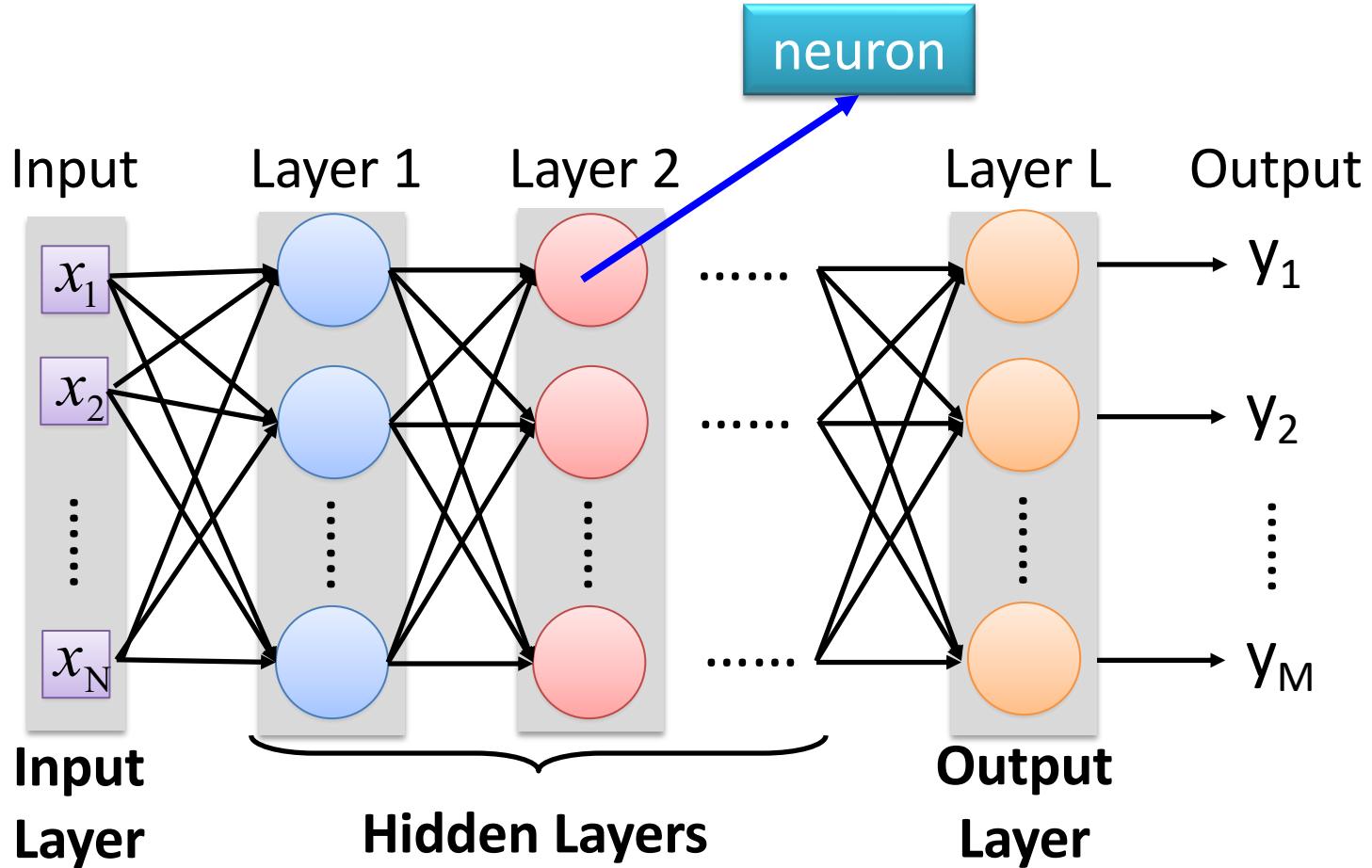
In deep learning, the function  $f$  is represented by neural network

# Element of Neural Network

**Neuron**     $f: R^K \rightarrow R$

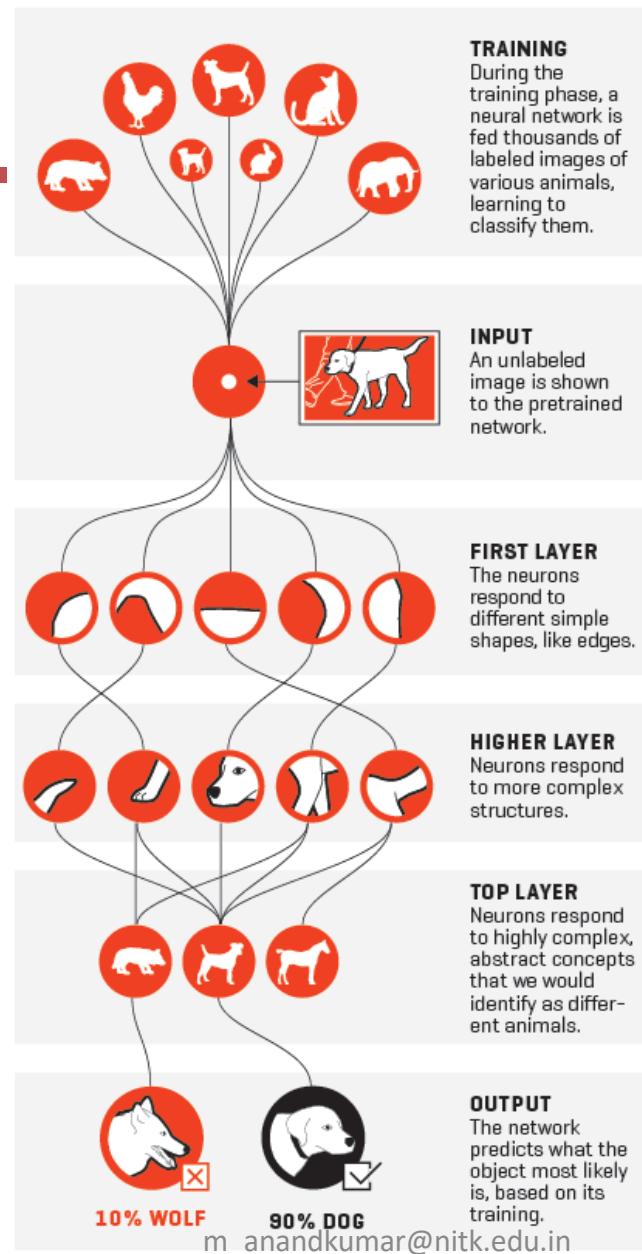


# Neural Network

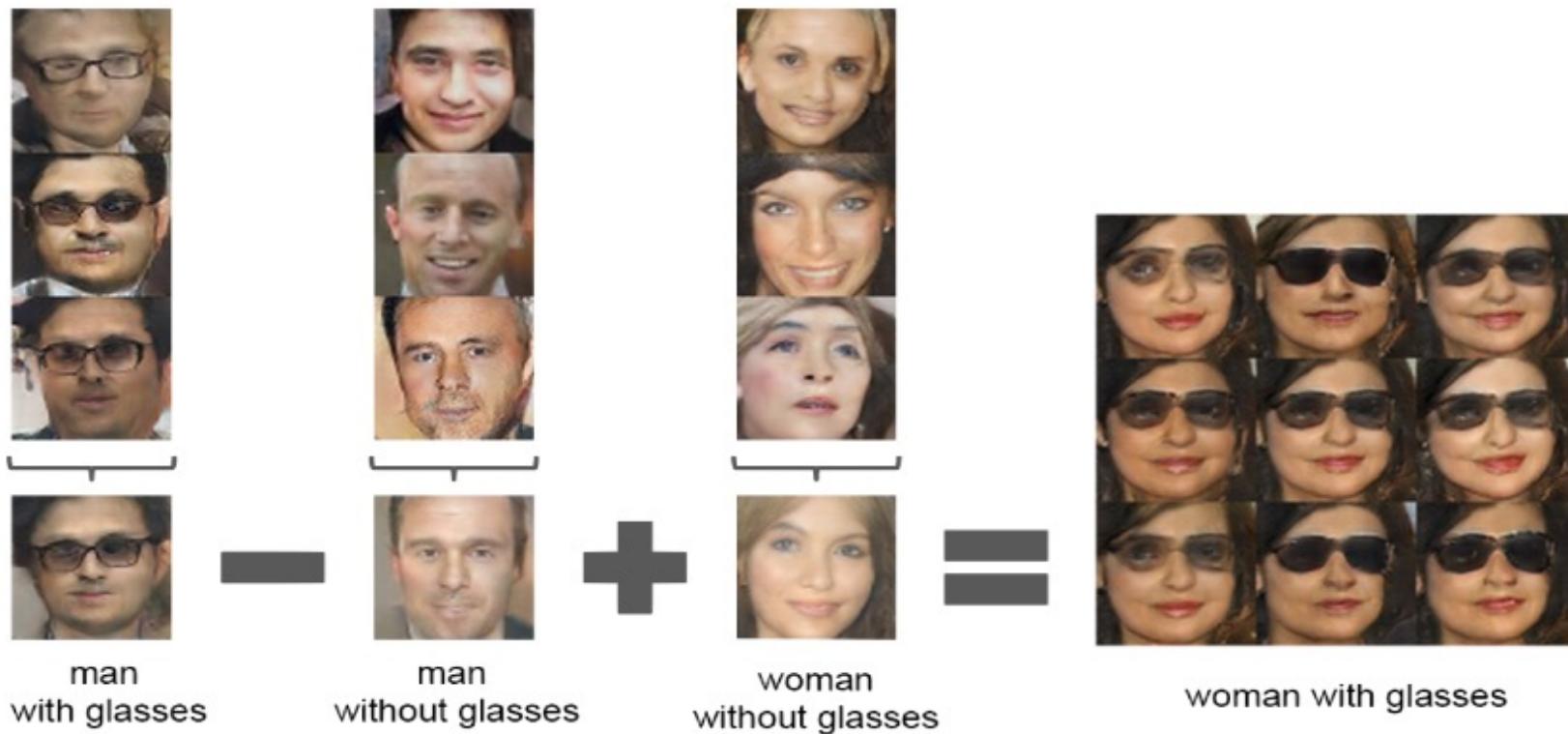


Deep means many hidden layers

## HOW NEURAL NETWORKS RECOGNIZE A DOG IN A PHOTO



## Latent vectors capture interesting patterns...



Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

# Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging
- [Your favorite area]

# Applications (conti..)

- Spam Email Detection
- Machine Translation (Language Translation)
- Image Search (Similarity)
- Clustering (KMeans) : Amazon
- Recommendations
- Classification : Google News
- Text Summarization - Google News
- Rating a Review/Comment: Yelp
- Fraud detection : Credit card Providers
- Decision Making : e.g. Bank/Insurance sector
- Sentiment Analysis
- Speech Understanding – iPhone with Siri
- Face Detection – Facebook's Photo tagging

# Similar/Duplicate Images

About 81 results (0.70 seconds)

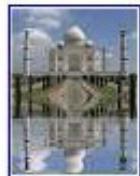


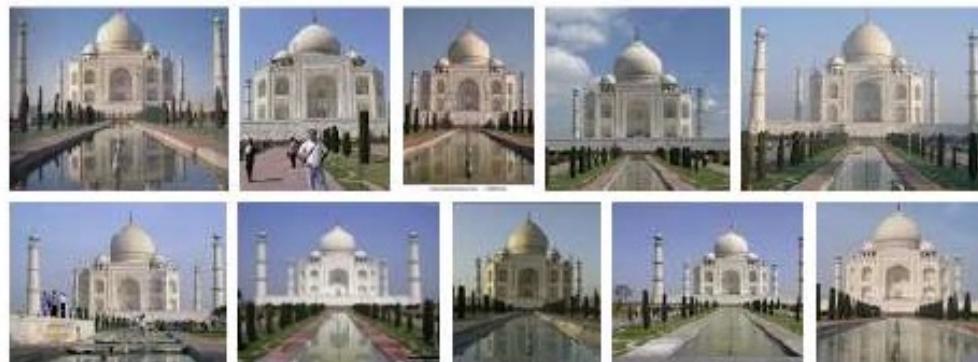
Image size:  
250 × 321

No other sizes of this image found.

Best guess for this image: [taj mahal](#)

## Visually similar images

[Report images](#)



## Remember

### Features ?

(Feature Extraction)

Can be :

- Width
- Height
- Contrast
- Brightness
- Position
- Hue
- Colors

### Check this :

LIRE (Lucene Image REtrieval)  
library -

<https://code.google.com/p/lire/>

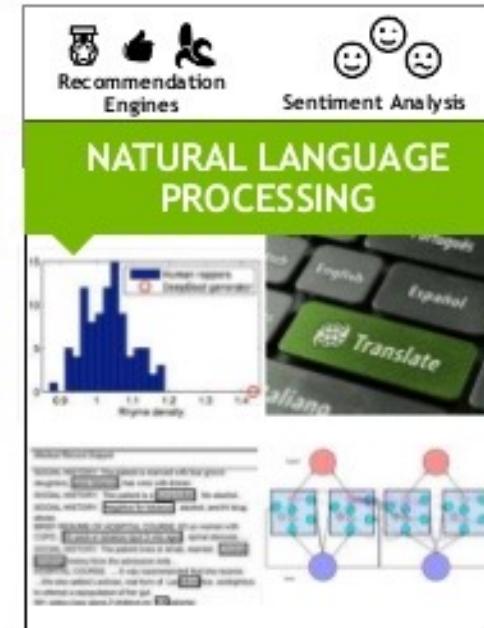
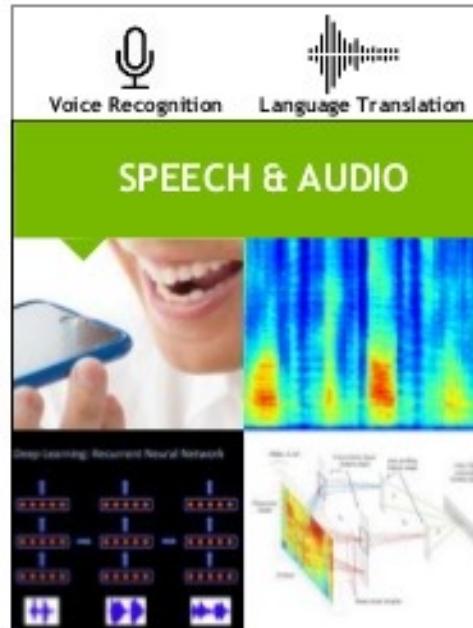
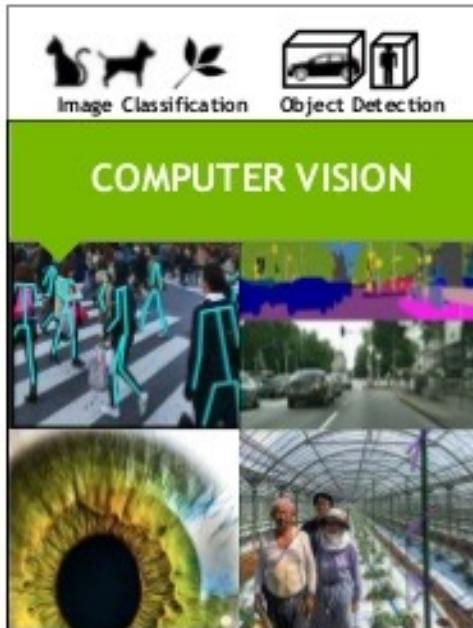
Credit: <https://www.google.co.in/>

---

# Popular Frameworks/Tools

- Weka
- Carrot2
- Gate
- OpenNLP
- LingPipe
- Stanford NLP
- Mallet – Topic Modelling
- Gensim – Topic Modelling (Python)
- Apache Mahout
- MLlib – Apache Spark
- scikit-learn - Python
- LIBSVM : Support Vector Machines
- and many more...

# AI APPLICATIONS



# 20 DEEP LEARNING Applications



1 Self Driving Cars

2 Entertainment

3 Visual Recognition

4 Virtual Assistants

5 Fraud Detection

6 Natural Language Processing

7 News Aggregation and Fraud News Detection

8 Detecting Developmental Delay in Children

9 Colourisation of Black and White images

10 Adding sounds to silent movies

Healthcare

11

Personalisations

12

Automatic Machine Translation

13

Automatic Handwriting Generation

14

Demographic & Election Predictions

15

16 Automatic Game Playing

17 Language Translations

18 Pixel Restoration

19 Photo Descriptions

20 Deep Dreaming

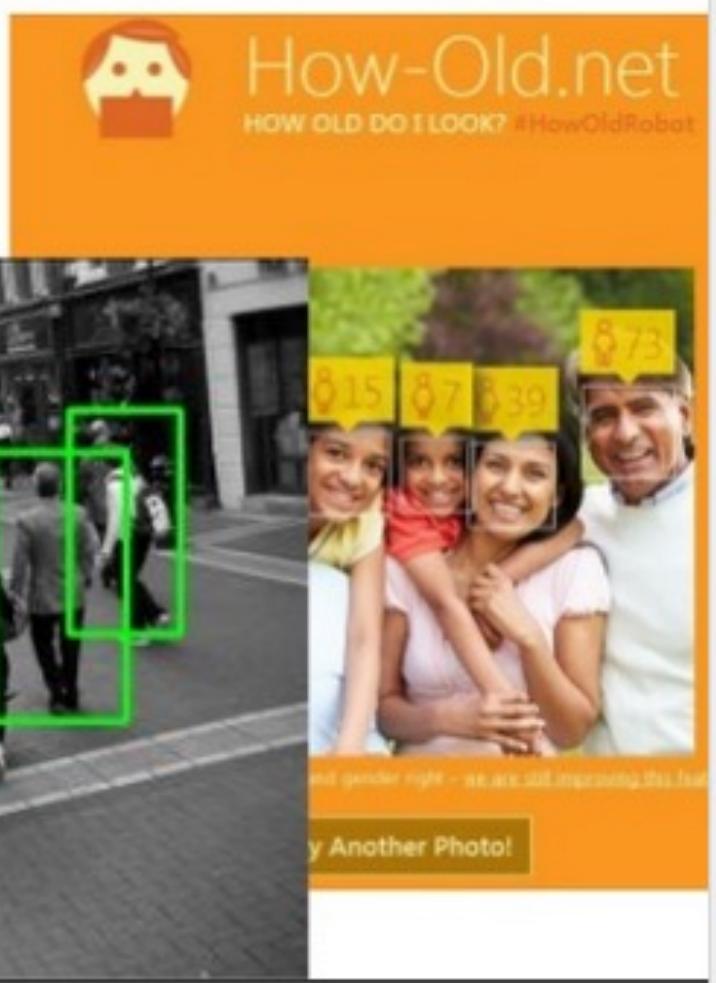
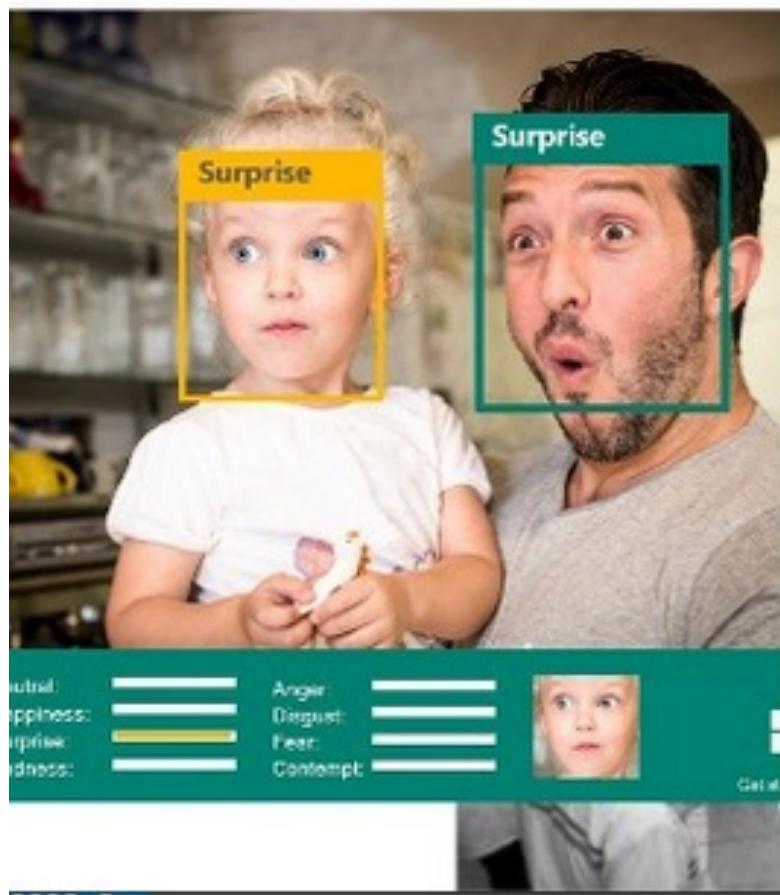
# Image Translation



# Applications

- Deep Learning AI is revolutionizing the filmmaking process as cameras learn to study human body language to imbibe in virtual characters.
- A deep learning model tends to associate the video frames with a database of pre-recorded sounds to select appropriate sounds for the scene
- <https://youtu.be/0FW99AQmMc8>
- <http://news.mit.edu/2016/artificial-intelligence-produces-realistic-sounds-0613>

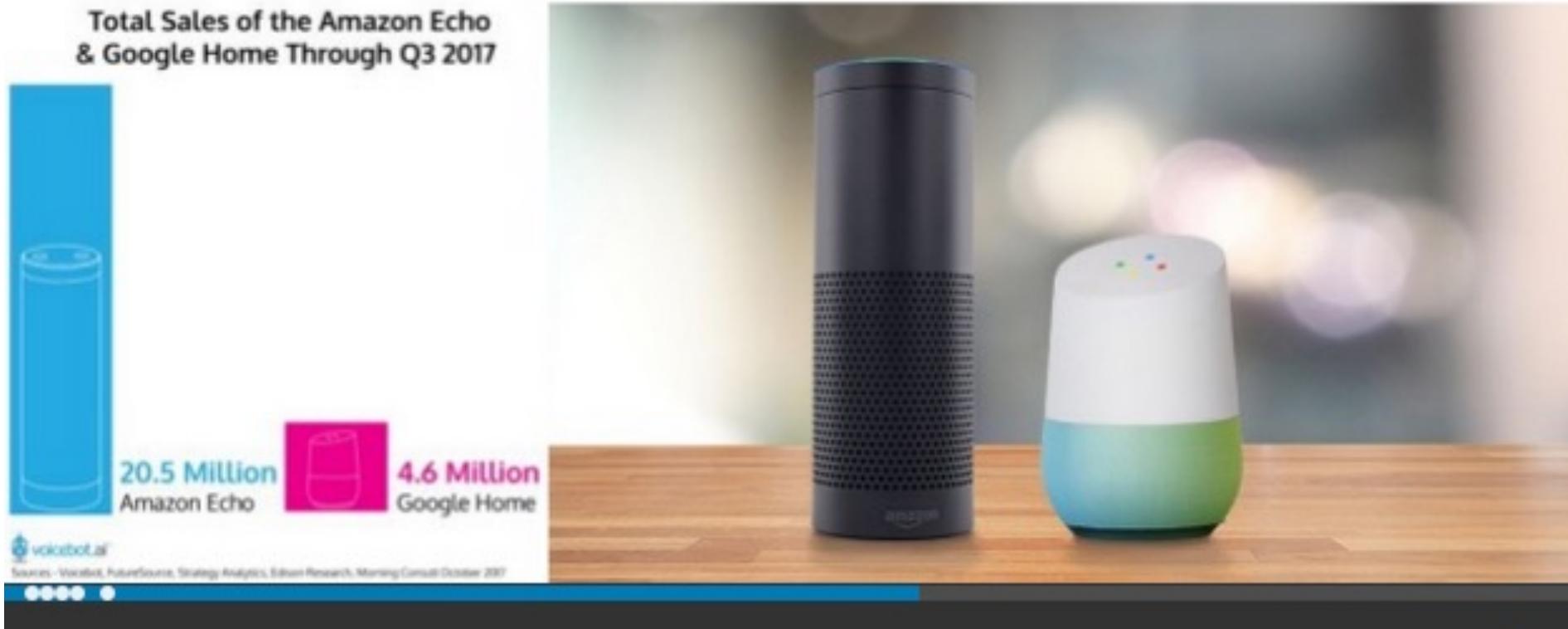
## Example: Object Detection



## Case: Amazon Echo

Amazon Alexa is in more than 20 million devices. The vast majority of these are in the Amazon Echo portfolio.

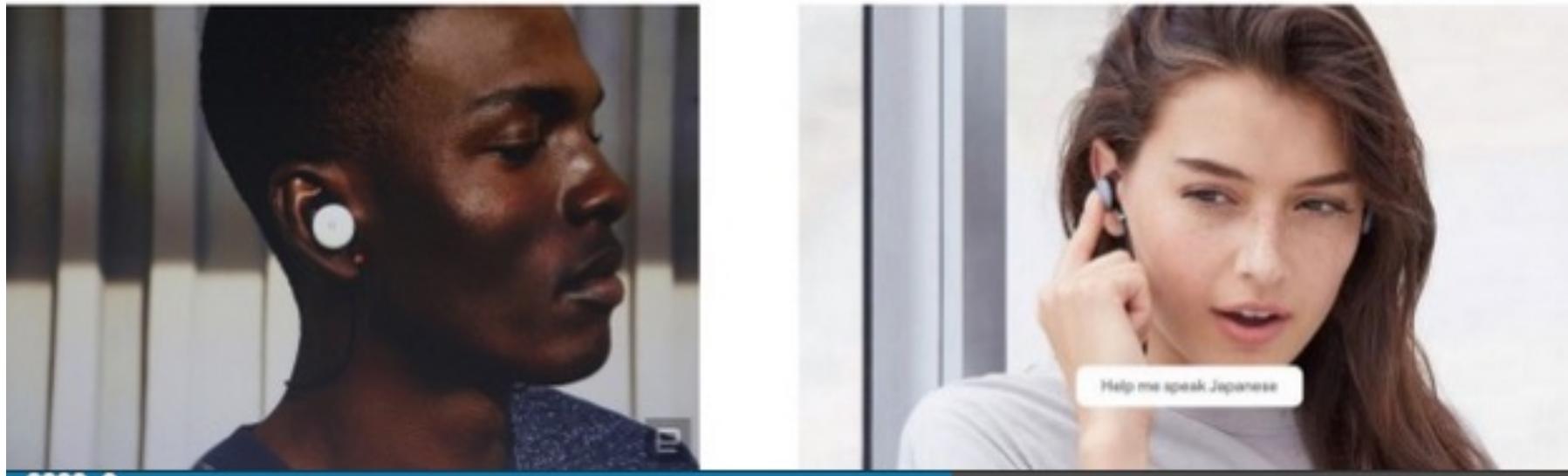
<https://www.voicebot.ai/2017/10/27/bezos-says-20-million-amazon-alexa-devices-sold/>



## Case: Google Pixel Buds

Google packed its headphones (in combination with the Pixel 2) with the power to translate between 40 languages, literally in real-time. The company has finally done what science fiction and countless Kickstarters have been promising us, but failing to deliver on, for years. This technology could fundamentally change how we communicate across the global community.

<https://www.engadget.com/2017/10/04/google-pixel-buds-translation-change-the-world/>



## 10 AI Applications That Could Change Health Care

APPLICATION	POTENTIAL ANNUAL VALUE BY 2026	KEY DRIVERS FOR ADOPTION
Robot-assisted surgery	\$40B	Technological advances in robotic solutions for more types of surgery
Virtual nursing assistants	20	Increasing pressure caused by medical labor shortage
Administrative workflow	18	Easier integration with existing technology infrastructure
Fraud detection	17	Need to address increasingly complex service and payment fraud attempts
Dosage error reduction	16	Prevalence of medical errors, which leads to tangible penalties
Connected machines	14	Proliferation of connected machines/devices
Clinical trial participation	13	Patent cliff; plethora of data; outcomes-driven approach
Preliminary diagnosis	5	Interoperability/data architecture to enhance accuracy
Automated image diagnosis	3	Storage capacity; greater trust in AI technology
Cybersecurity	2	Increase in breaches; pressure to protect health data

# Machine Learning in Healthcare



*Diseases Identification & Diagnosis*



*Personalized Medicine/  
Treatment*



*Drug Discovery &  
Manufacturing*



*Smart Health Records*



*Medical Imaging*



*Diseases Prediction*

# APPLICATIONS OF MACHINE LEARNING IN HEALTHCARE



Better Imaging &  
Diagnostic Techniques



Detecting Diseases  
in Early Stage



Providing Personalized  
Treatment



Clinical Decision  
Support



Drug Discovery &  
Research



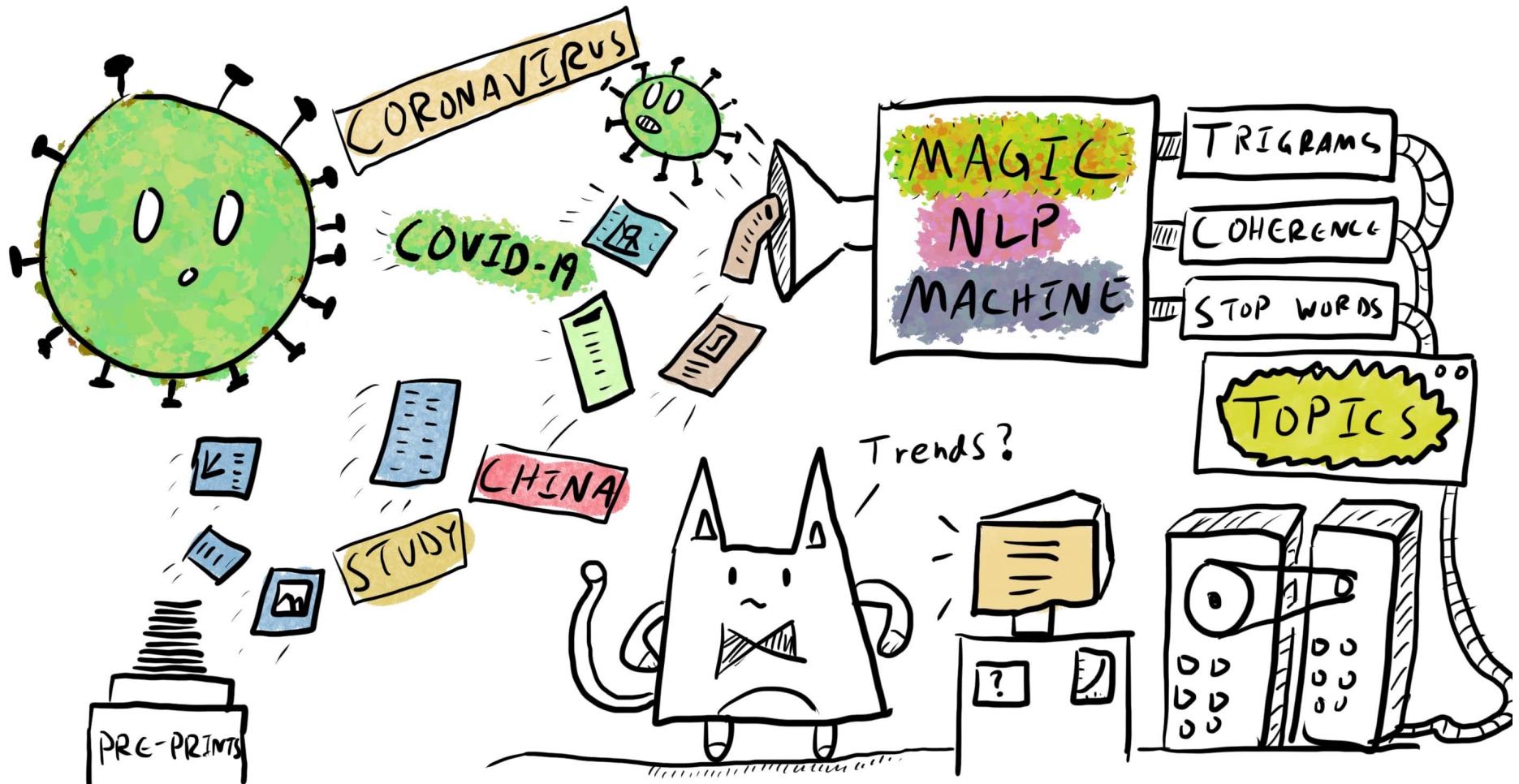
Preventing Medical  
Insurance Frauds

MACHINE HACK



## Predict A Doctor's Consultation Fee Hackathon

# Bio-NLP



<https://towardsdatascience.com/summarising-the-latest-research-on-coronavirus-with-nlp-and-topic-modelling-28b867ad9860>

m\_anandkumar@nitk.edu.in

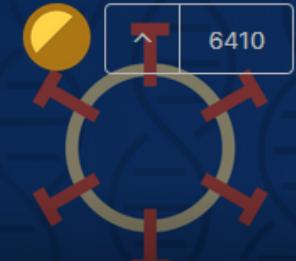
# Covid NLP datasets

Dataset

## COVID-19 Open Research Dataset Challenge (CORD-19)

An AI challenge with AI2, CZI, MSR, Georgetown, NIH & The White House

 Allen Institute For AI and 8 collaborators • updated 9 days ago (Version 12)



[Allen Institute for AI](#) Open Research Dataset (CORD-19), over 47,000 scholarly articles, including over 36,000 with full text, about COVID-19 and the coronavirus family of viruses for use by the global research community.

## DRAVIDIAN



## HASOC-Dravidian-CodeMix - FIRE 2020

Organized by dravidiancodemixed - Current server time: June 23, 2020, 3:27 a.m. UTC

### ► Current

End

First phase

Competition Ends

June 19, 2020, 6:53 p.m. UTC

Never

Learn the Details

Phases

Participate

Results

Overview

Bharathi Raja Chakravarthi, PhD Researcher, Insight SFI Research Centre for Data Analytics, Data Science and National University of Ireland Galway

Evaluation

Dr. Anand Kumar, Assistant Professor, Department of Information Technology, National Institute of Technology Karnataka Surathkal, India

Organizers

Important Dates

Dr John P. McCrae, Lecturer-above-the-bar, Insight SFI Research Centre for Data Analytics, Data Science and National University of Ireland Galway

Terms and Conditions

Prof. K P Soman, Head, CEN, Amrita Vishwa Vidyapeetham

[View Details](#) [Edit Details](#) [Delete](#) [CEN - Amrita Vishwa Vidyapeetham](#)

## Identification of informative COVID-19 English Tweets

For this task, participants are asked to develop systems that automatically identify whether an English Tweet related to the novel coronavirus (COVID-19) is informative or not. Such informative Tweets provide information about recovered, suspected, confirmed and death cases as well as location or travel history of the cases.

**Data is released on June 21, 2020!**

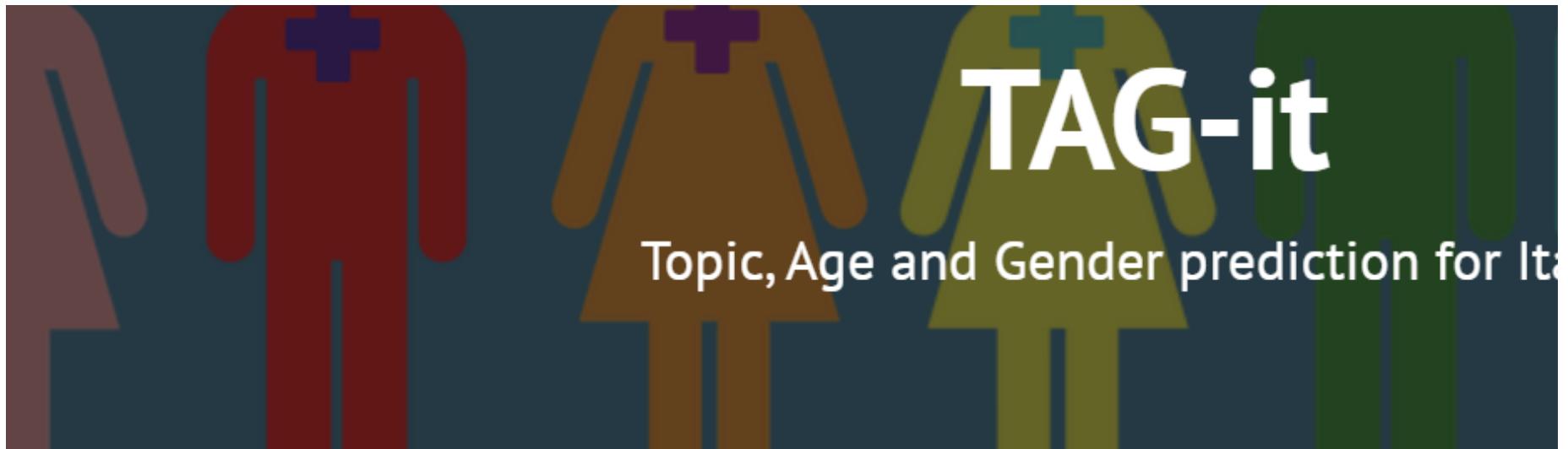
**Official valuation will be between August 17, 2020 and August 21, 2020 (Please register [here](#) to participate).**

There is a [mailing list](#) for future announcements.

## Introduction

The goals of our shared task are: (1) To develop a language processing task that potentially impacts research and downstream applications, and (2) To provide the community with a new dataset for identifying informative COVID-19 English Tweets.

As of mid-June 2020, the COVID-19 outbreak has led to about [445K deaths and 8.2M+ infected patients from 215 regions & countries](#), creating fear and panic for people all around the world. Recently, much attention has been paid to building monitoring systems (e.g. [The Johns Hopkins Coronavirus Dashboard](#)) to track the development of the outbreak and to provide users the information related to the virus, e.g. any new suspicious/confirmed cases near/in the users' regions. Note that most of the "official" sources used in the tracking tools



## OVERVIEW

TAG-it is a profiling task for Italian.

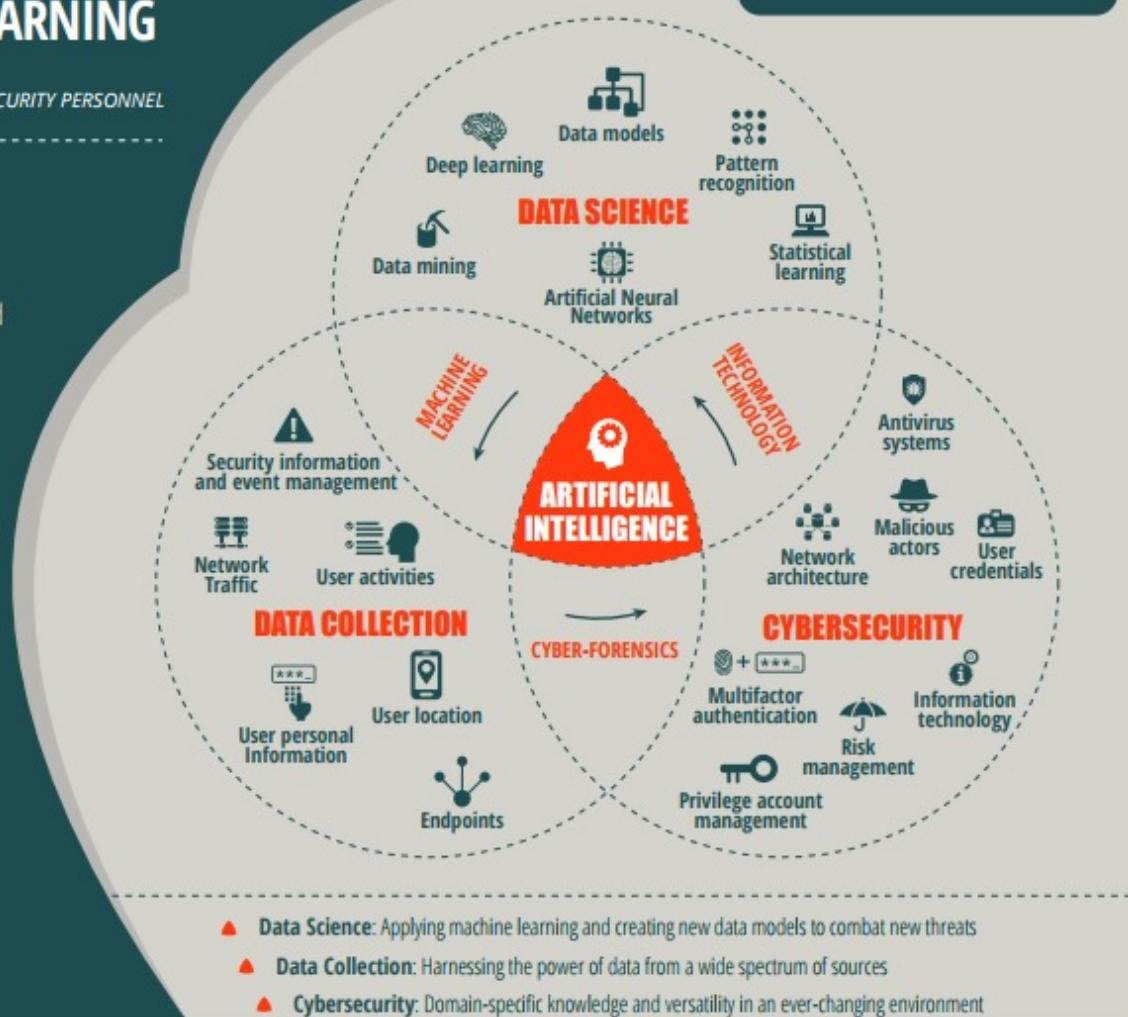
This can be seen as a follow-up of the [GxG](#) task organised in the context of EVALITA 2018 though with some differences. GxG was concerned with gender prediction, and had two distinctive traits: (i) models were trained a tested *cross-genre*, and (ii) evidence per author was for some genres (Twitter and YouTube) extremely limited ( tweet or one comment). The combination of these two aspects yielded scores that were comparatively lower than those observed in other campaigns, and for other languages. One of the core reasons for training the models cross genre was to remove as much as possible genre-specific traits, but also topic-related features. The two would basically coincide in most n-gram-based models, which are standard for this task.

## WHAT CAN MACHINE LEARNING DO FOR CYBERSECURITY?

A POTENT NEW ARSENAL FOR IT AND CYBERSECURITY PERSONNEL

-  User entity behavioral analytics, deep learning, automation
-  Assist IT professionals and defend against new cyberthreats
-  Better predictive models, lower FPR, distill new metrics
-  Fraud and anomaly detection
-  Defend against new cyberthreats
-  Better use of internal data and global repositories
-  Tackle device influx and enhanced data loss prevention (DLP) solutions

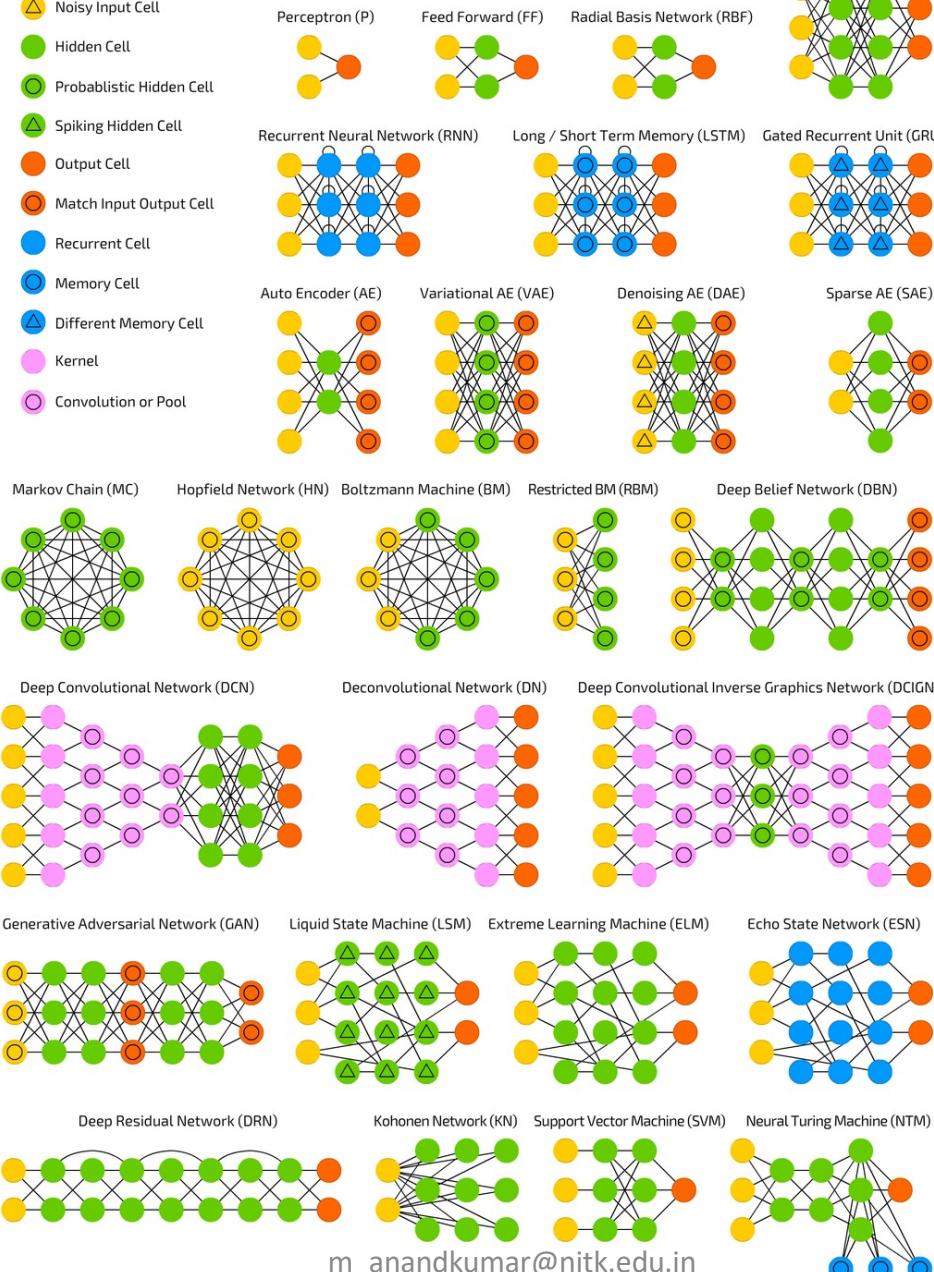
### Analytics and Forensics



# Neural Networks

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

©2016 Fjodor van Veen - asimovinstitute.org



m\_anandkumar@nitk.edu.in

62

- <http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>
- <https://www.mygreatlearning.com/blog/deep-learning-applications/>
- <http://www.r2d3.us/visual-intro-to-machine-learning-part-2/>
- [http://www.hpc.lsu.edu/training/weekly-materials/2016-Fall/machine\\_learning\\_qb2\\_fall\\_2016.pdf](http://www.hpc.lsu.edu/training/weekly-materials/2016-Fall/machine_learning_qb2_fall_2016.pdf)
- [https://vas3k.com/blog/machine learning/](https://vas3k.com/blog/machine_learning/)

# Deep Feed Forward Networks and Backpropagation Algorithms

Dr. Anand Kumar M  
Assistant Professor,  
Dept of IT  
NIT-K, Surathkal

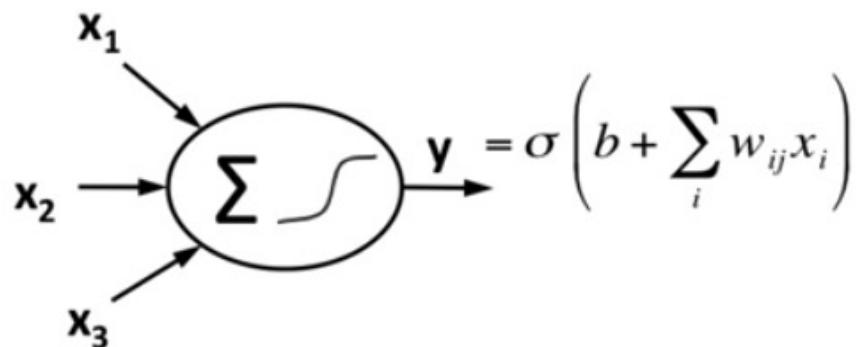
# Road Map

- Types of NN
- Feed Forward NN
- Example
- Back Propagation -example

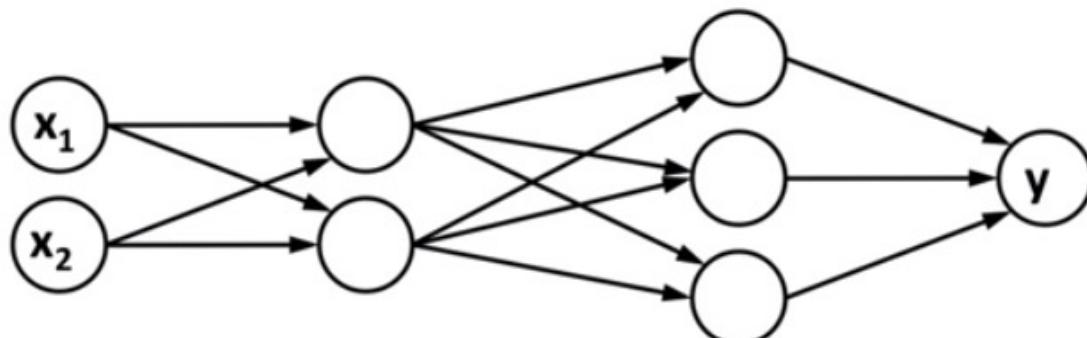
# Architecture

This section presents the architecture of the network that is most commonly used with the backpropagation algorithm –  
**the multilayer feedforward network**

# Types of Neural Networks

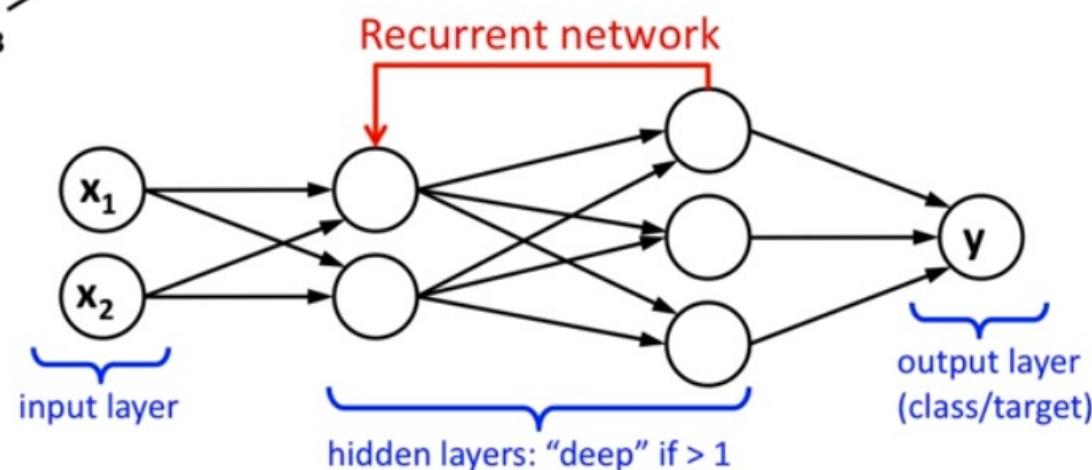
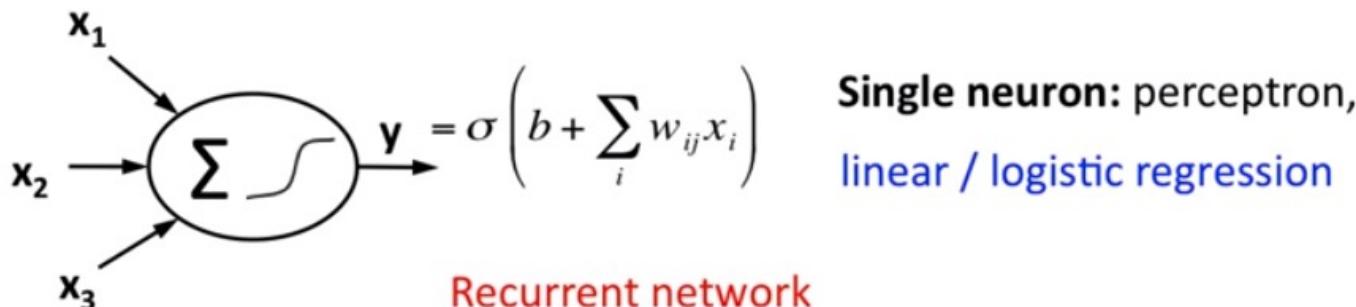


**Single neuron:** perceptron,  
linear / logistic regression

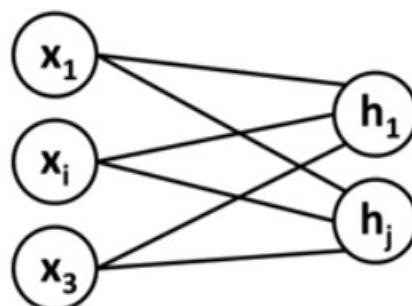


**Feed-forward netwo**  
(no cycles) -- non-lin  
classification & regre

# Types of Neural Networks

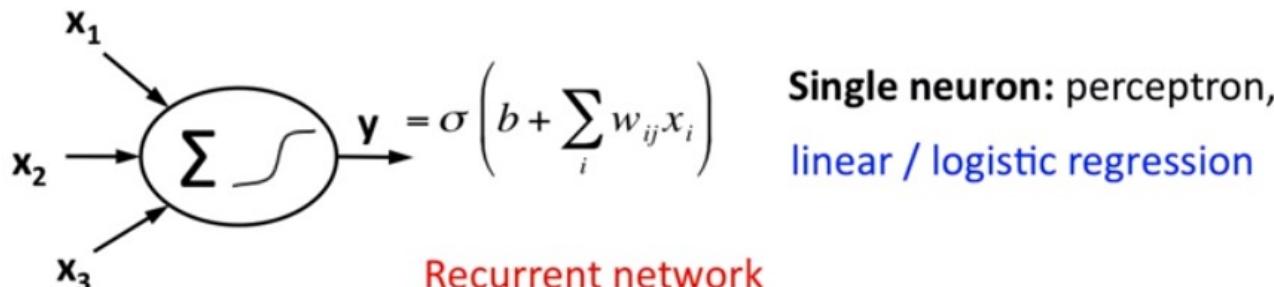


**Feed-forward network**  
(no cycles) -- non-linear classification & regression

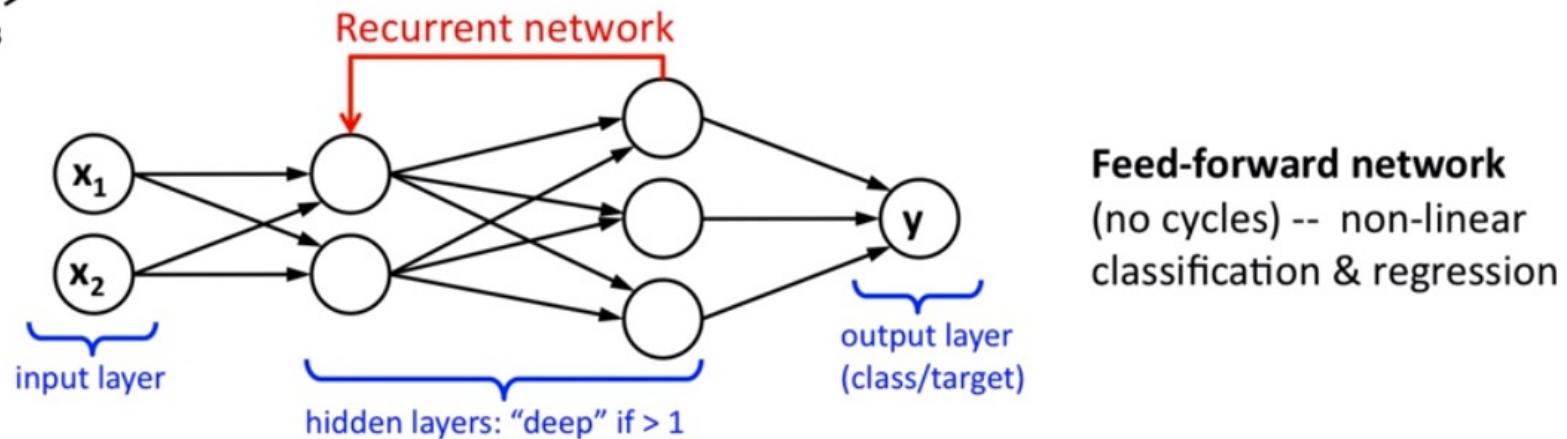


**Symmetric (RBM)**  
unsupervised, trained  
to maximize likelihood  
of input data

# Types of Neural Networks



**Single neuron:** perceptron,  
linear / logistic regression



**Feed-forward network**  
(no cycles) -- non-linear  
classification & regression

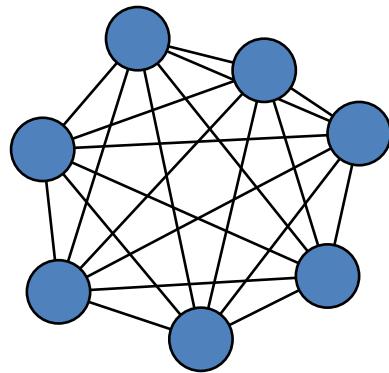
$$\sigma\left(\beta_i + \sum_j w_{ij} h_j\right) = \begin{array}{c} x_1 \\ x_i \\ x_3 \end{array} \rightarrow \begin{array}{c} h_1 \\ h_j \end{array}$$

same set of weights

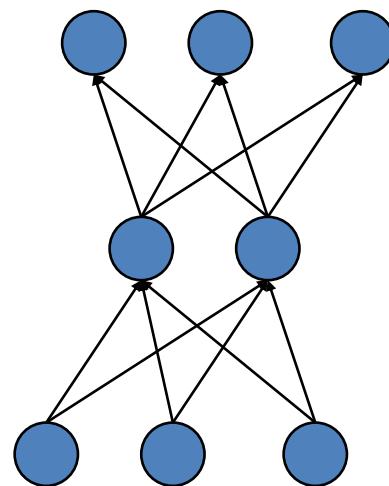
$$P(\text{input} \mid \text{hidden}) \quad P(\text{hidden} \mid \text{input})$$
$$= \sigma\left(b_j + \sum_i w_{ij} x_i\right)$$

**Symmetric (RBM)**  
unsupervised, trained  
to maximize likelihood  
of input data

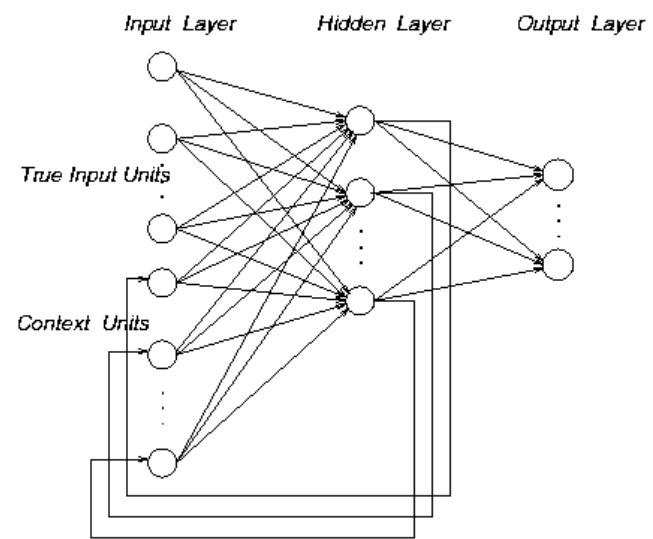
# Topologies of Neural Networks



*completely  
connected*



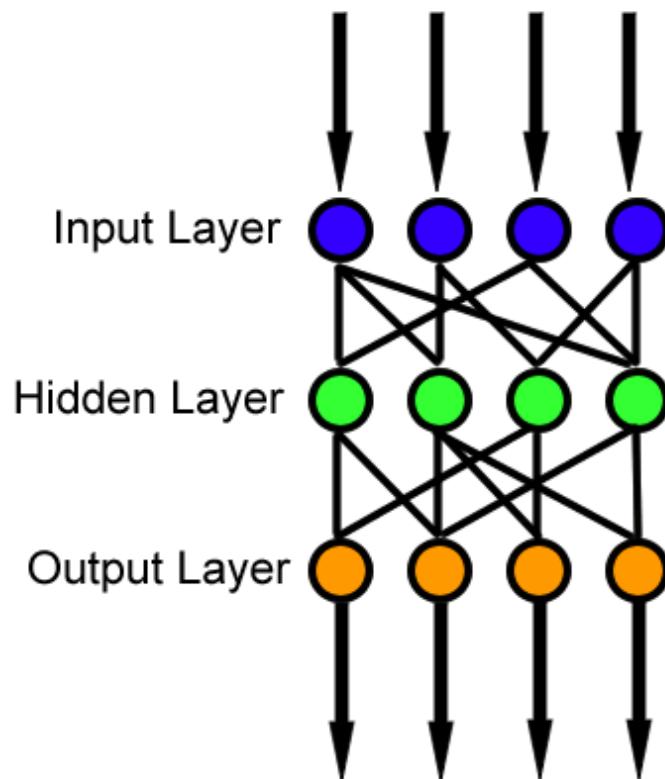
*feedforward  
(directed, a-cyclic)*



*recurrent  
(feedback connections)*

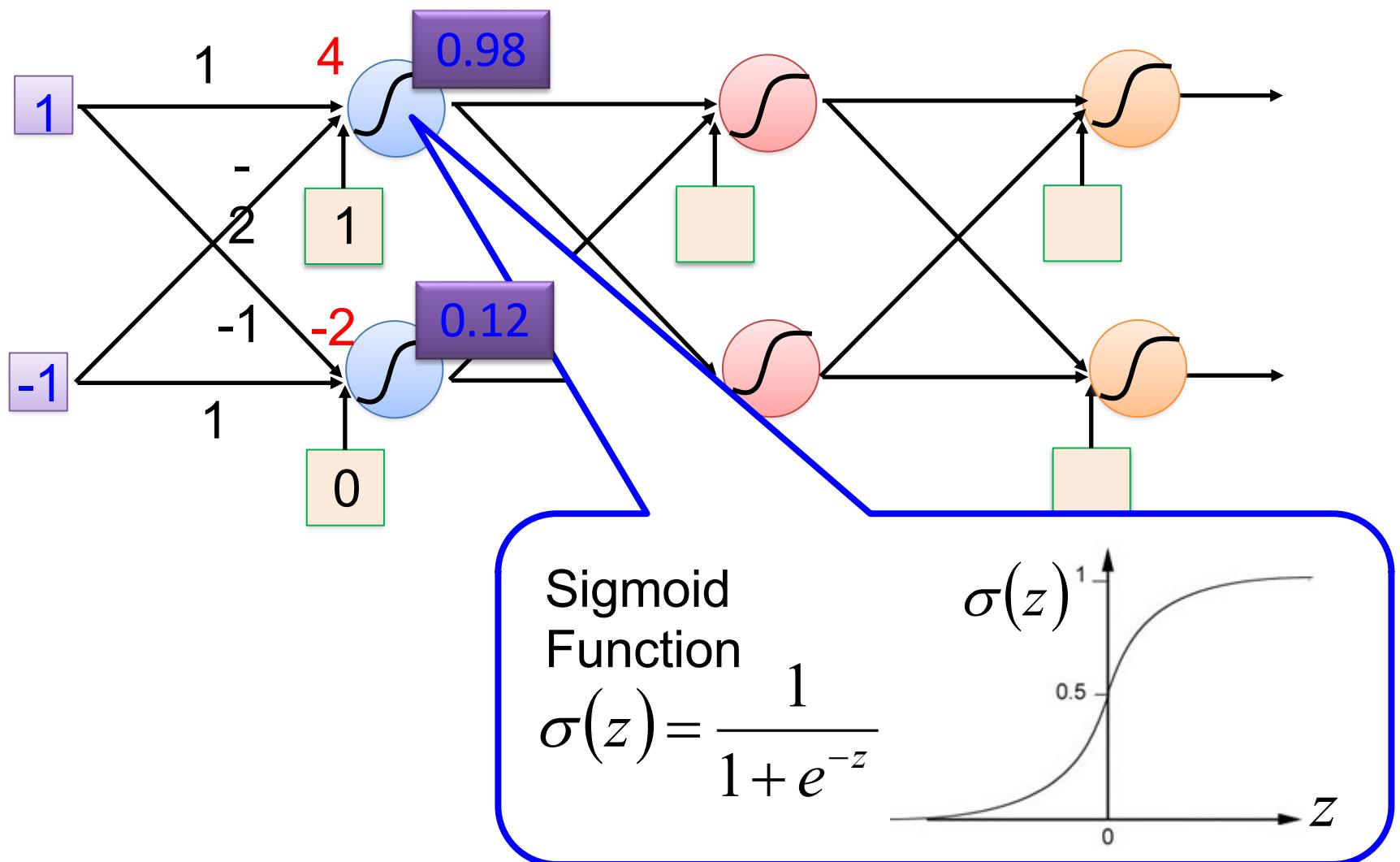
# Feed Forward Network

- In a feed forward network information always moves one direction; it never goes backwards.
- 

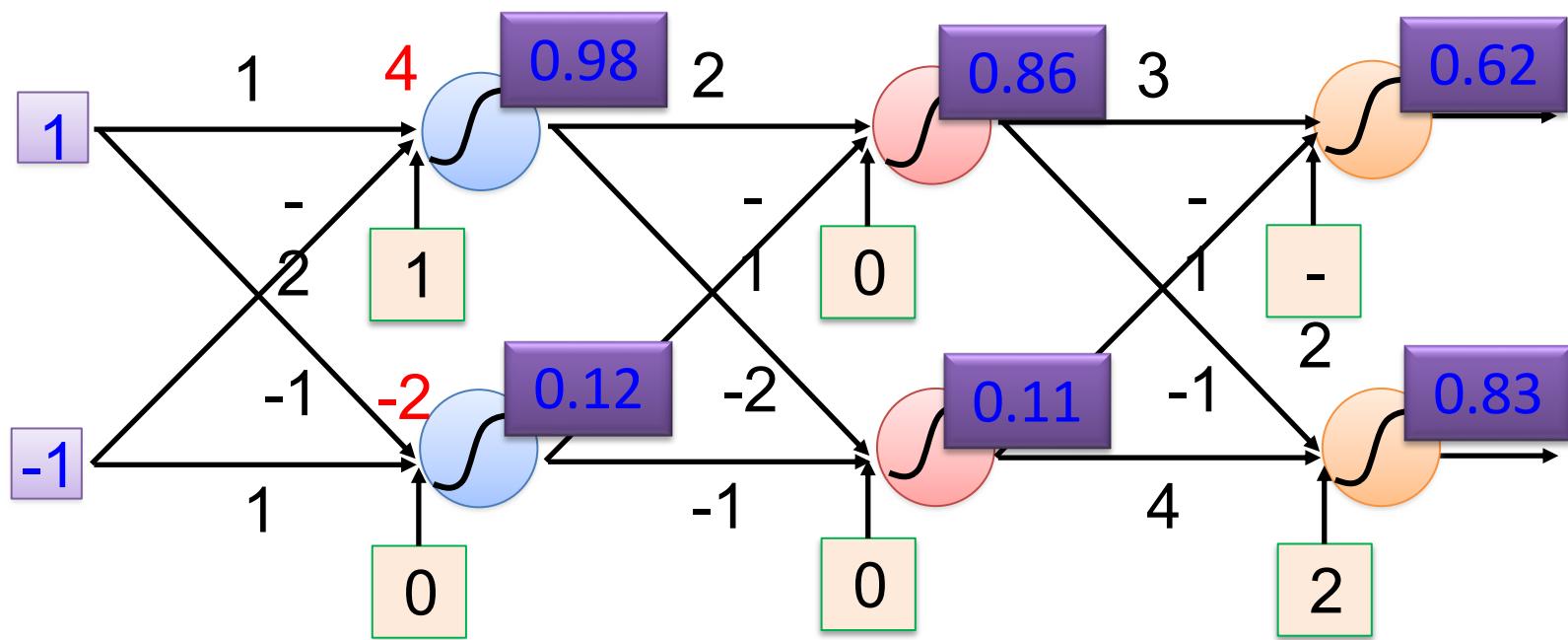




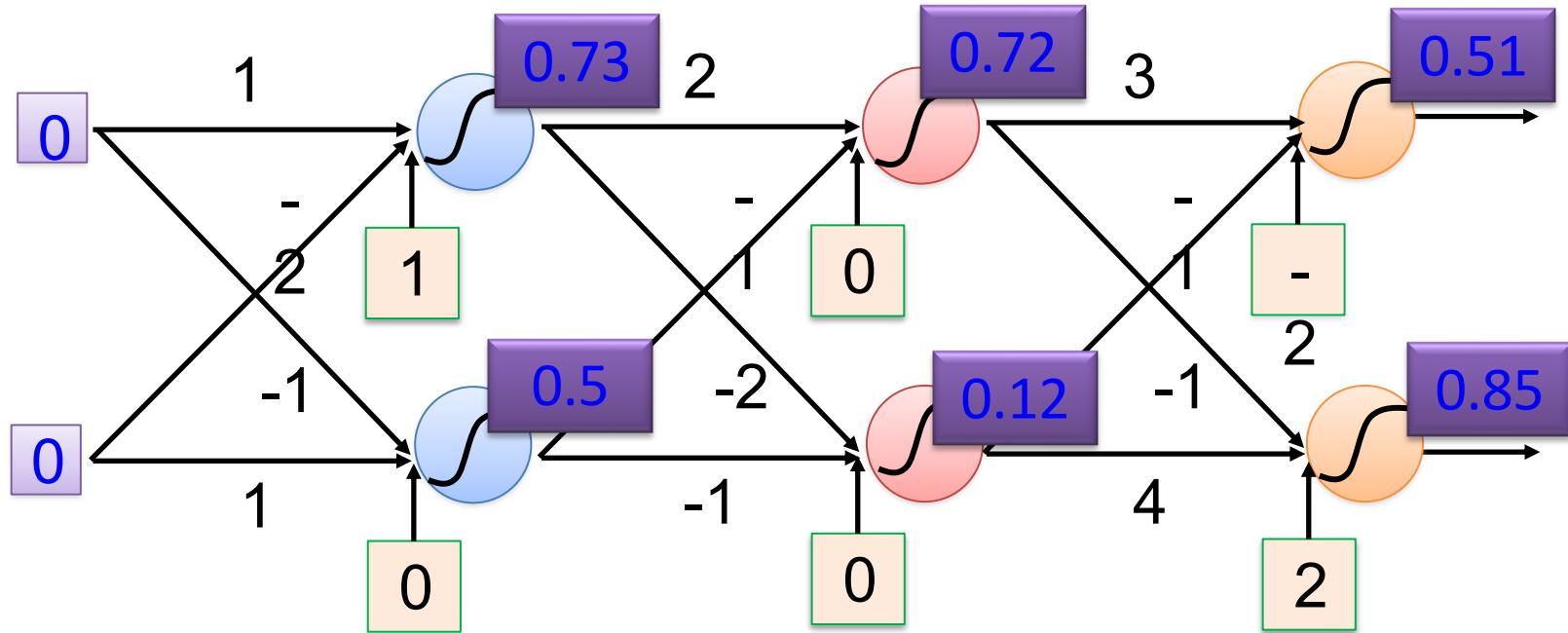
# Example of Neural Network



# Example of Neural Network



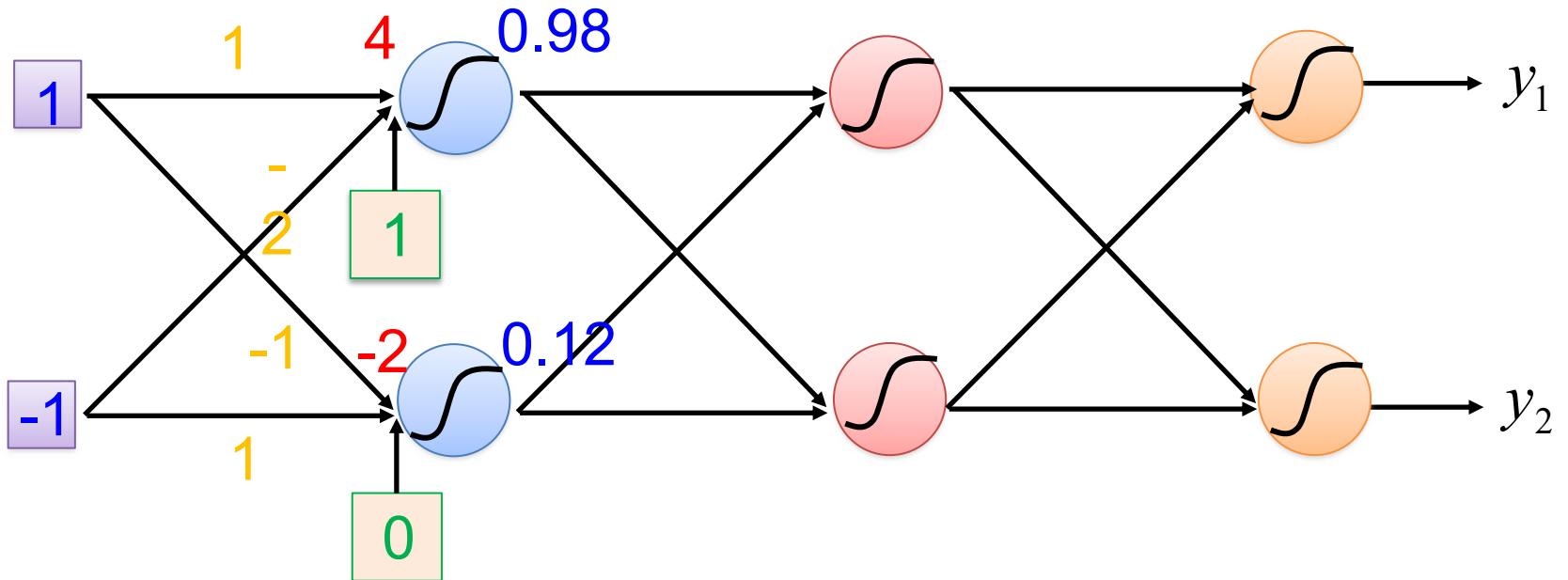
# Example of Neural Network



$$f: \mathbb{R}^2 \rightarrow \mathbb{R}^2 \quad f \left( \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix} \quad f \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix}$$

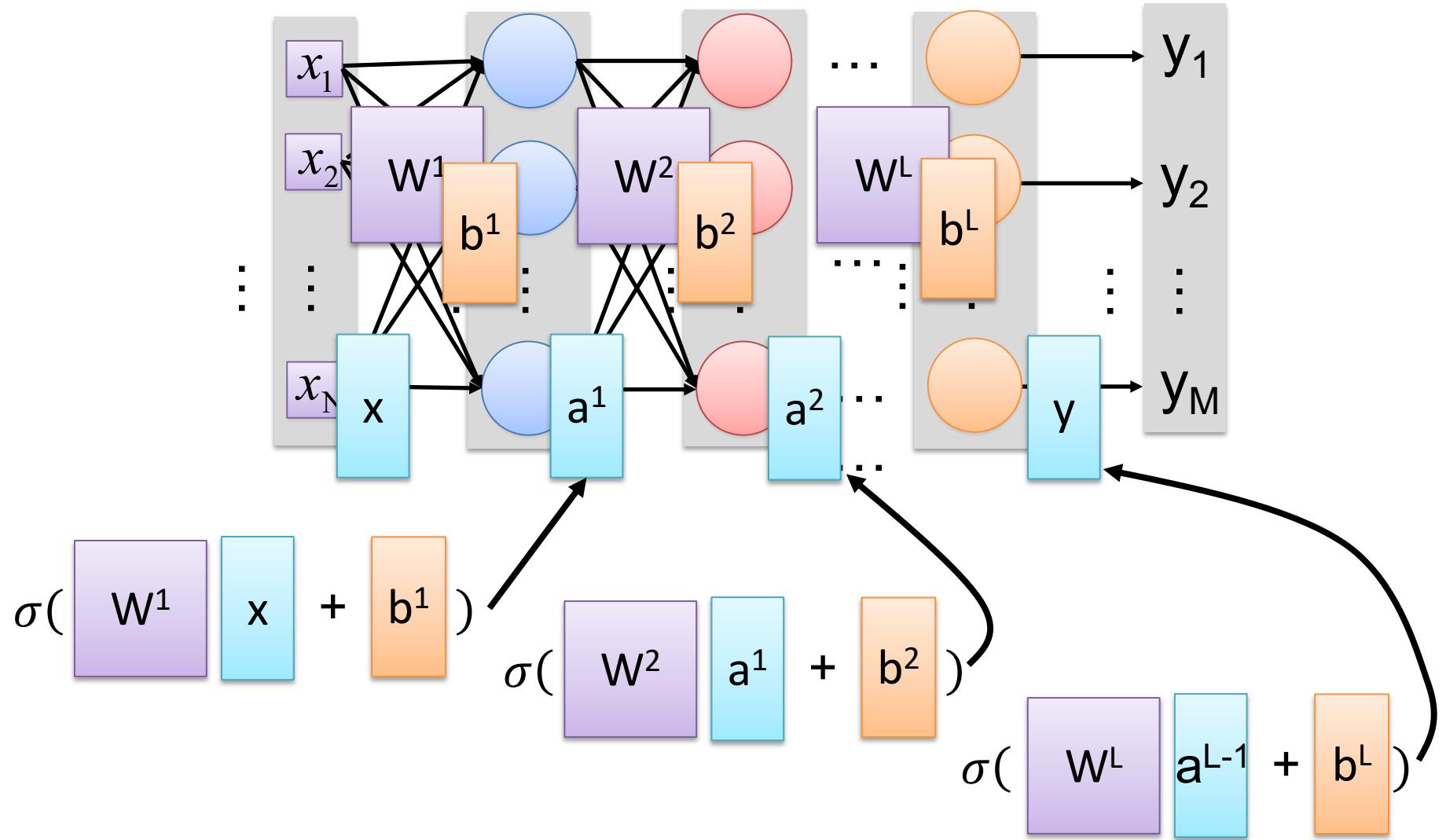
Different parameters define different function

# Matrix Operation

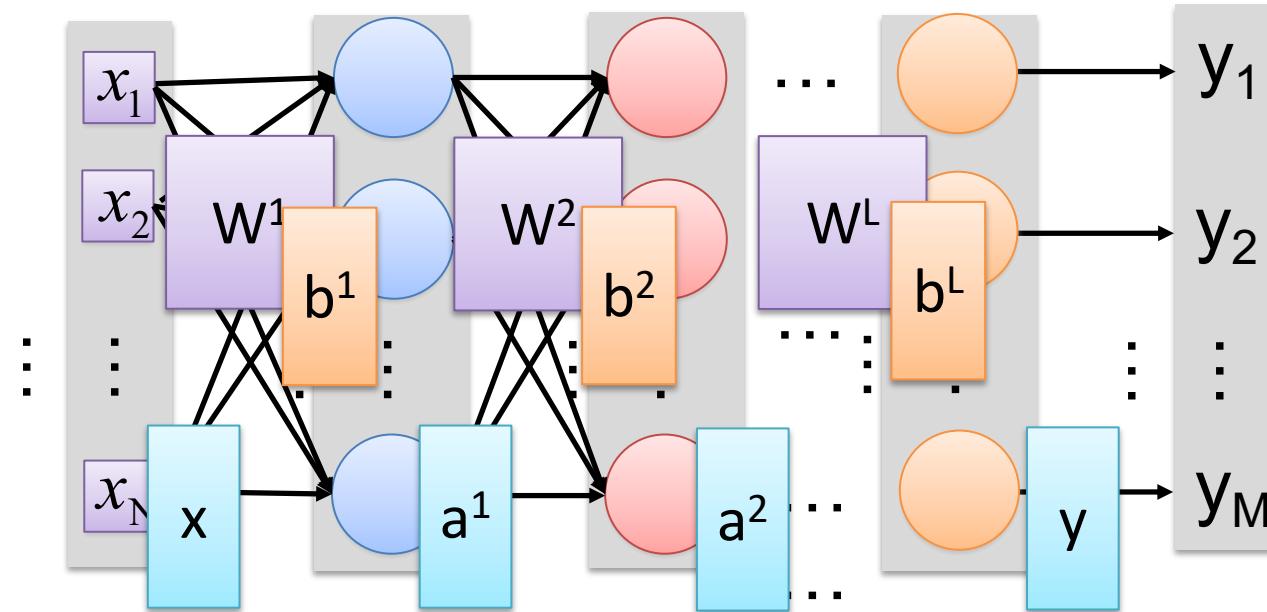


$$\sigma \left( \underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}} \right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

# Neural Network



# Neural Network

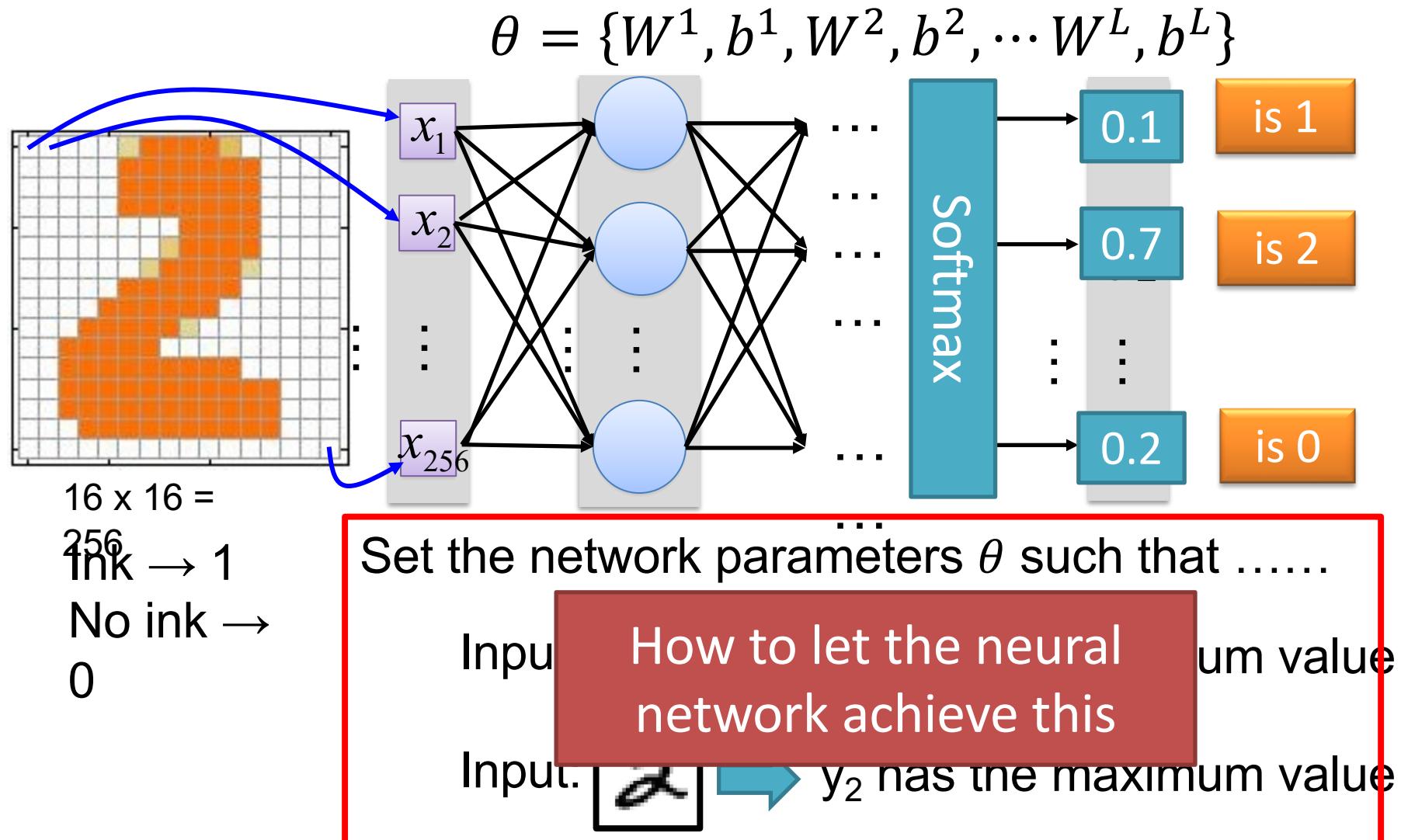


$$y = f(x)$$

Using parallel computing  
techniques to speed up matrix  
operation

$$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

# How to set network parameters



# Training Data

- Preparing training data: images and their labels



“5”



“0”



“4”



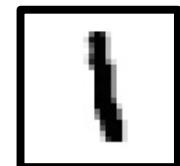
“1”



“9”



“2”



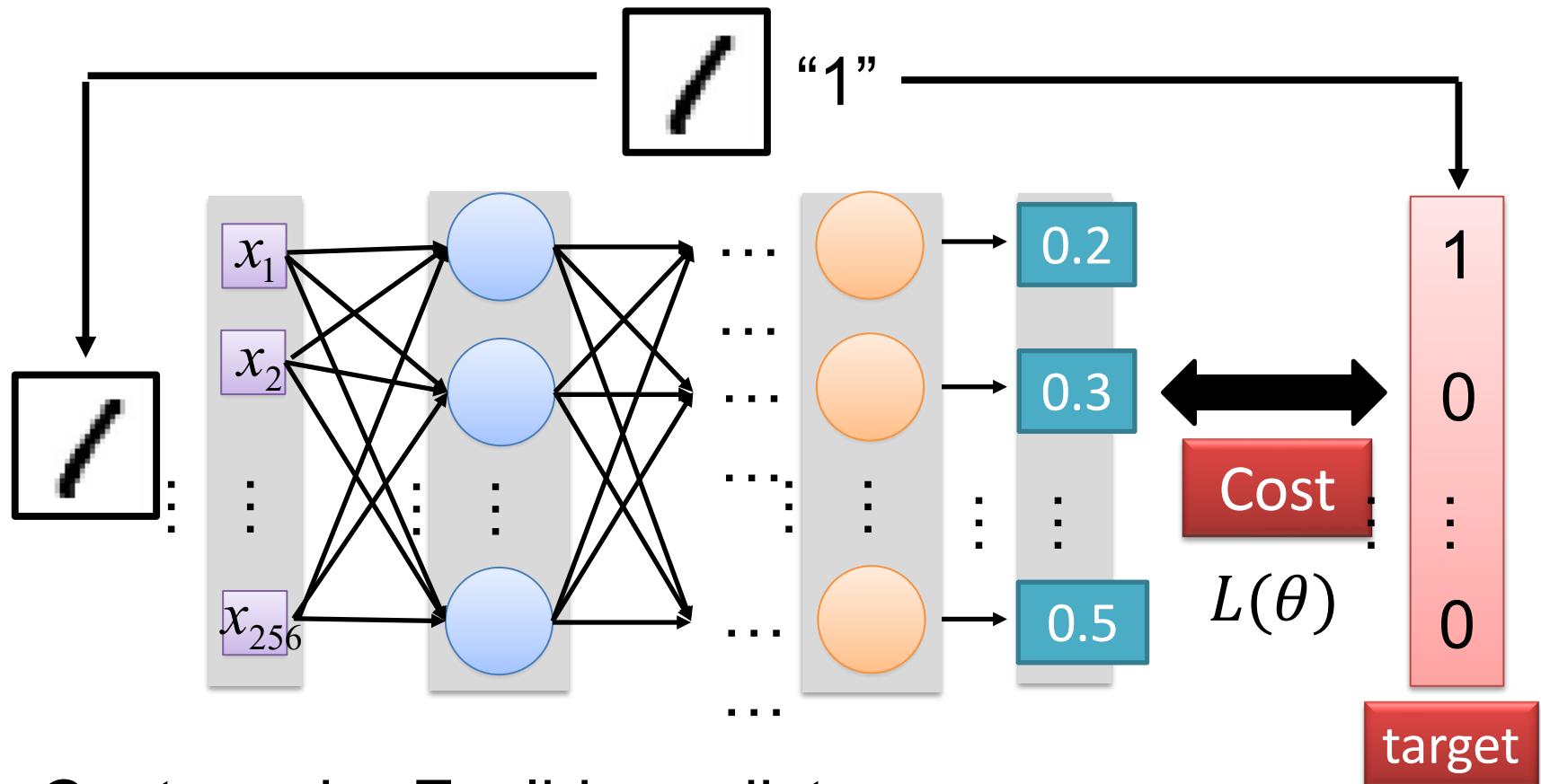
“1”



“3”

Using the training data to find  
the network parameters.

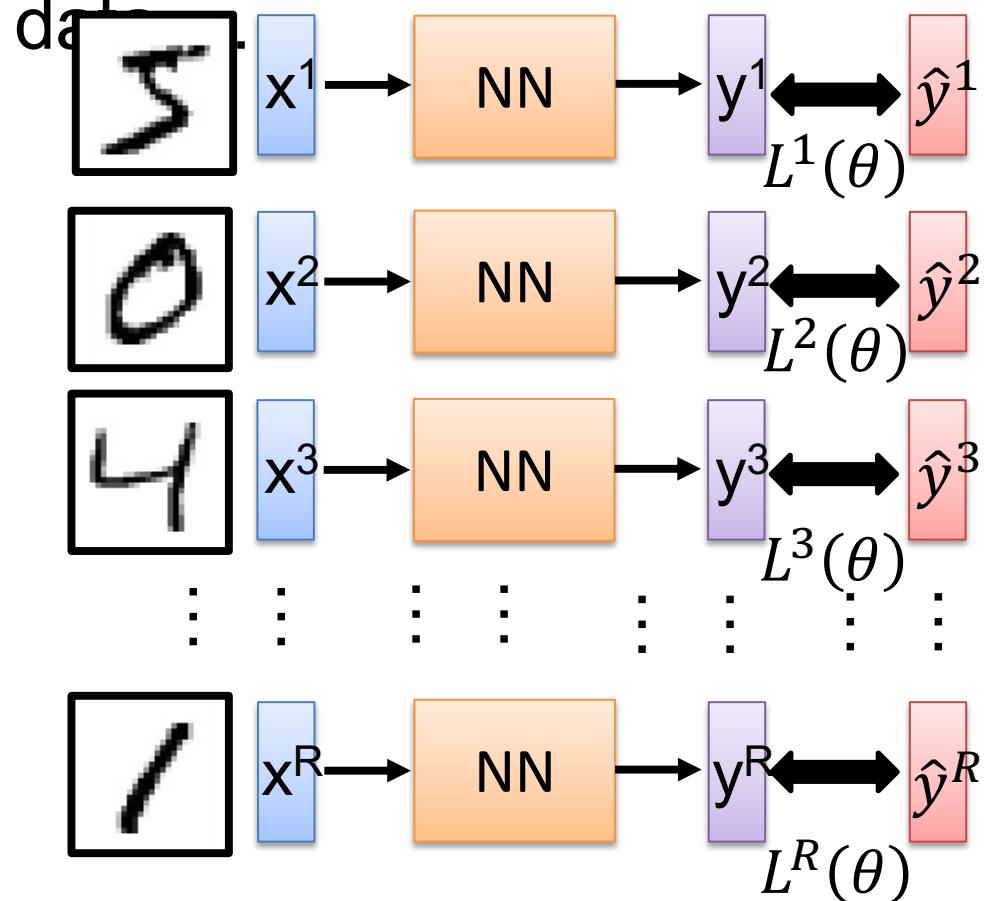
Given a set of network parameters  $\theta$ , each example has a cost value.



Cost can be Euclidean distance or cross entropy of the network output and target

# Total Cost

For all training data



Total Cost:

$$C(\theta) = \sum_{r=1}^R L^r(\theta)$$

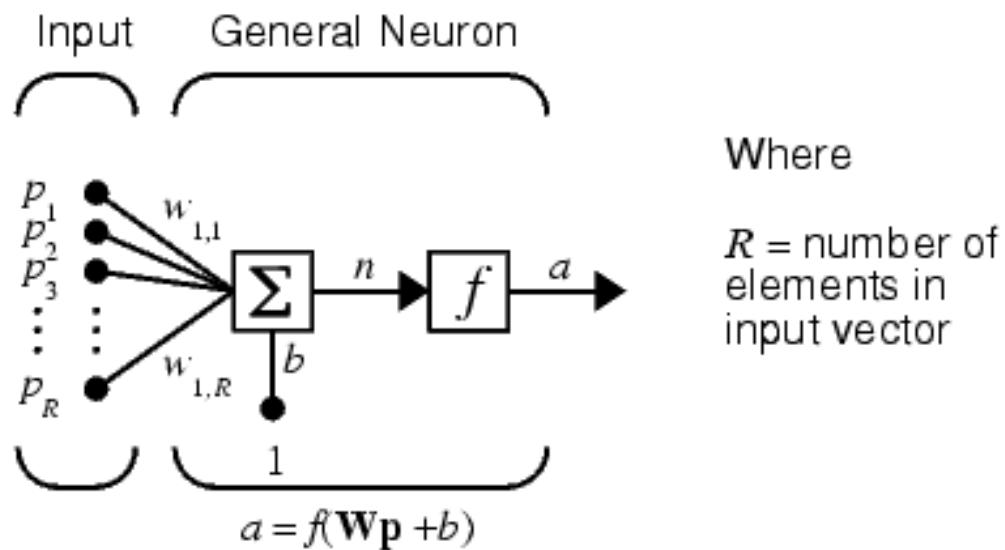
How bad the network parameters  $\theta$  is on this task

Find the network parameters  $\theta^*$  that minimize this value

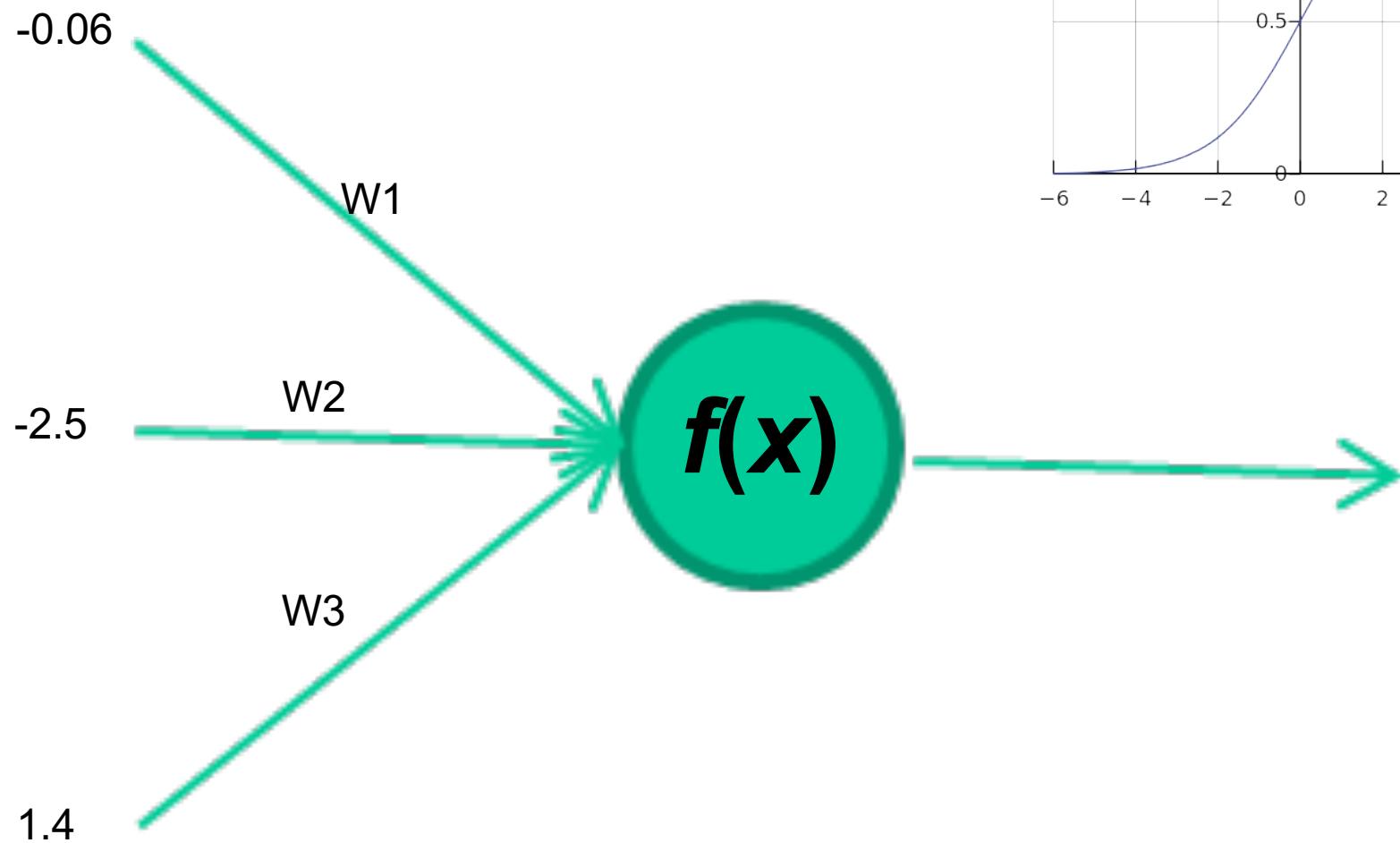
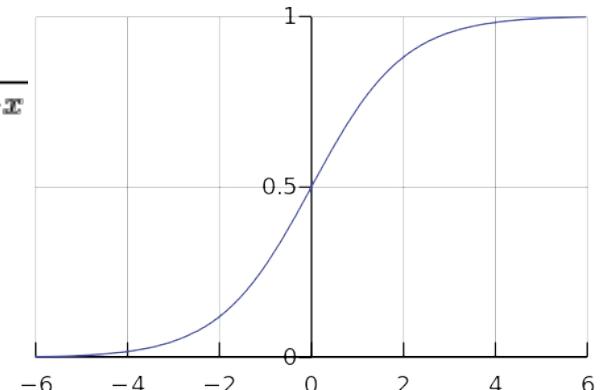
# Architecture

## Neuron Model

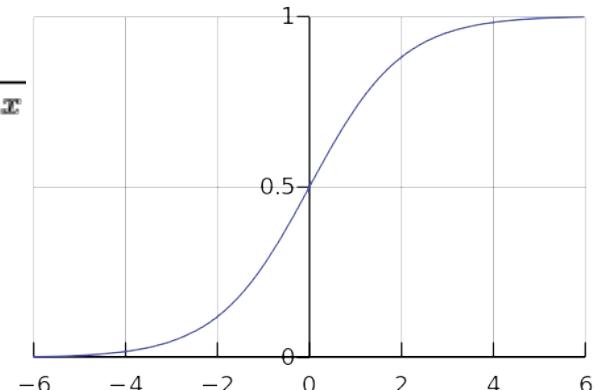
An elementary neuron with R inputs is shown below. Each input is weighted with an appropriate w. The sum of the weighted inputs and the bias forms the input to the transfer function f. Neurons can use any **differentiable transfer function** f to generate their output.



$$f(x) = \frac{1}{1 + e^{-x}}$$



$$f(x) = \frac{1}{1 + e^{-x}}$$



-0.06

2.7

-2.5

-8.6

0.002

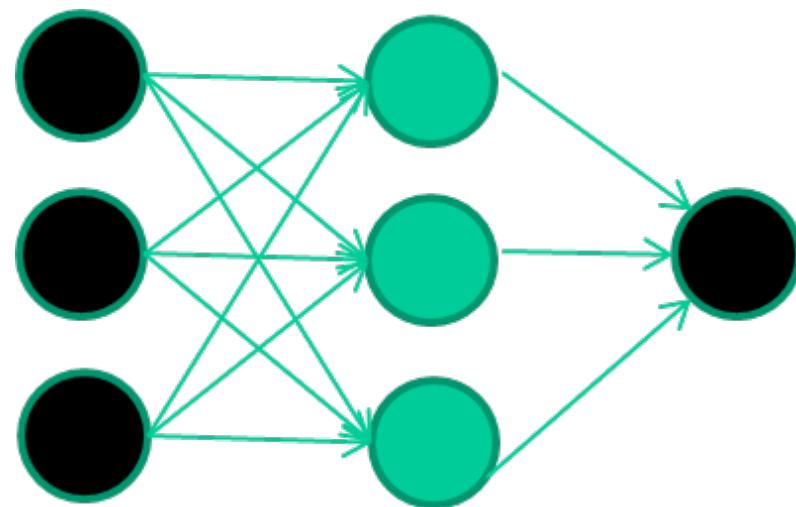
1.4

**$f(x)$**

$$x = -0.06 \times 2.7 + 2.5 \times 8.6 + 1.4 \times 0.002 = 21.34$$

*A dataset*

<i>Fields</i>	<i>class</i>
1.4 2.7 1.9	0
3.8 3.4 3.2	0
6.4 2.8 1.7	1
4.1 0.1 0.2	0
etc ...	

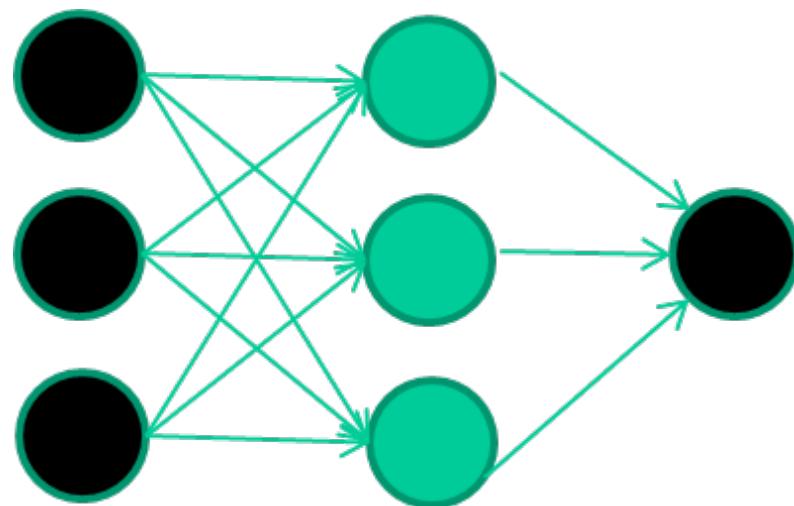


*Training the neural network*

**Fields**           **class**

1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc ...



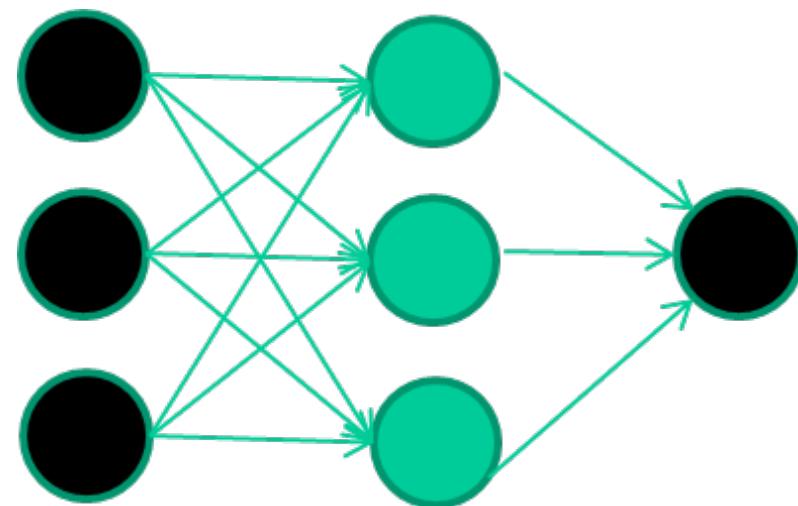
*Training data*

**Fields**            **class**

1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc ...

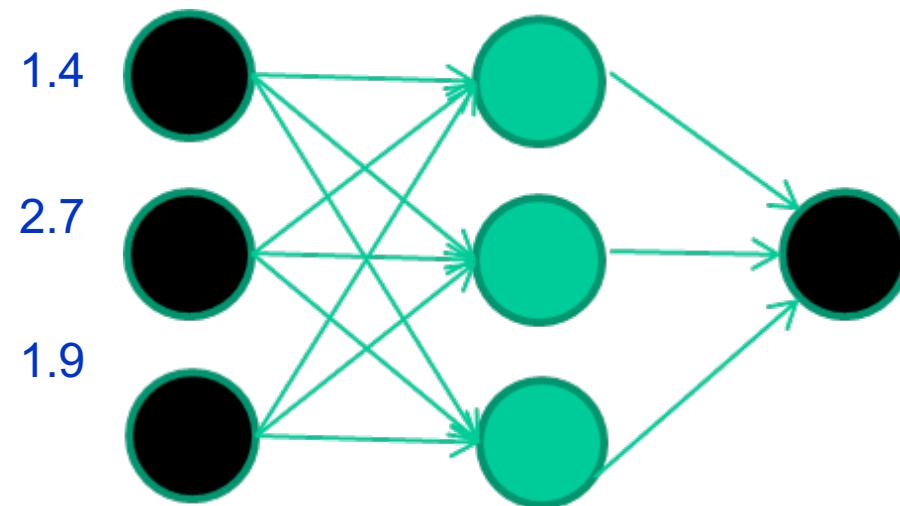
**Initialise with random weights**



*Training data*

<b>Fields</b>			
			<b>class</b>
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc ...			

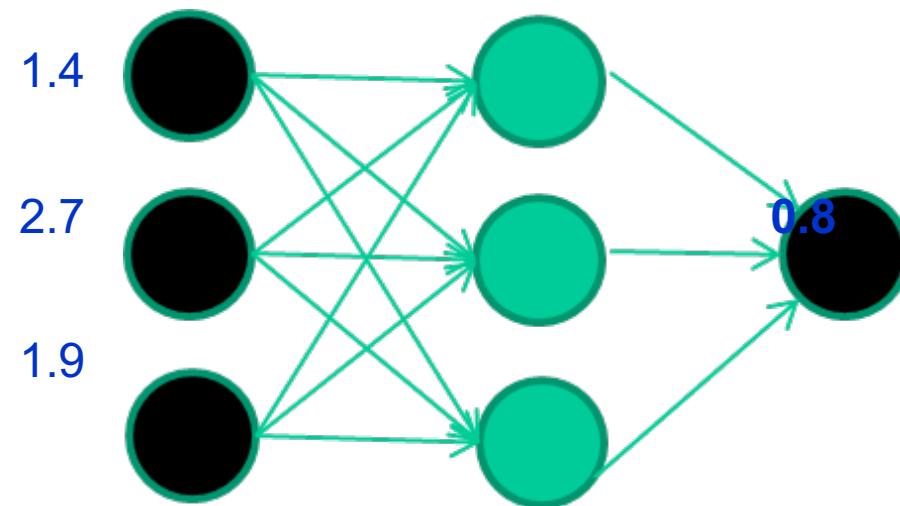
Present a training pattern



*Training data*

<b>Fields</b>			
			<b>class</b>
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc ...			

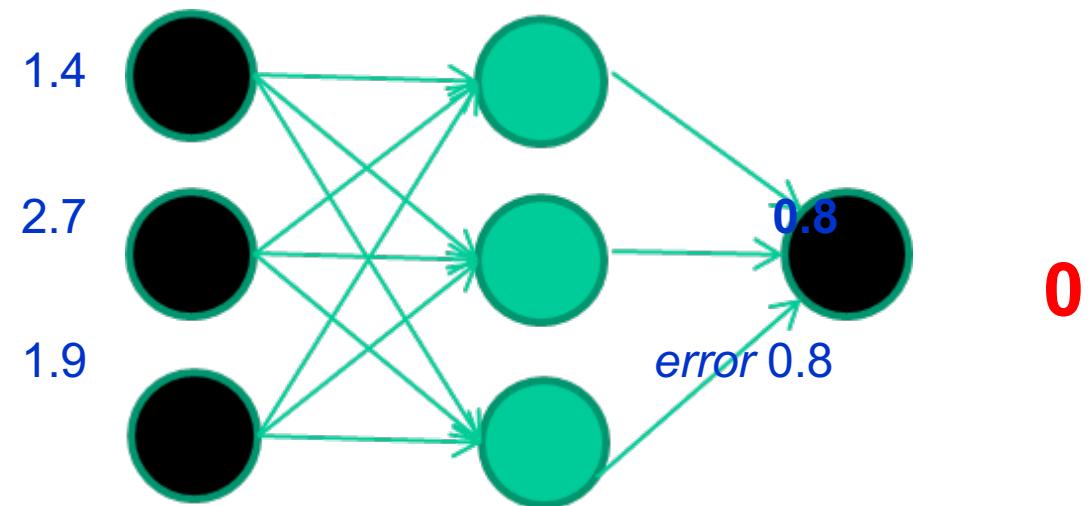
**Feed it through to get output**



*Training data*

<b>Fields</b>	<b>class</b>		
	<b>class</b>		
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc ...			

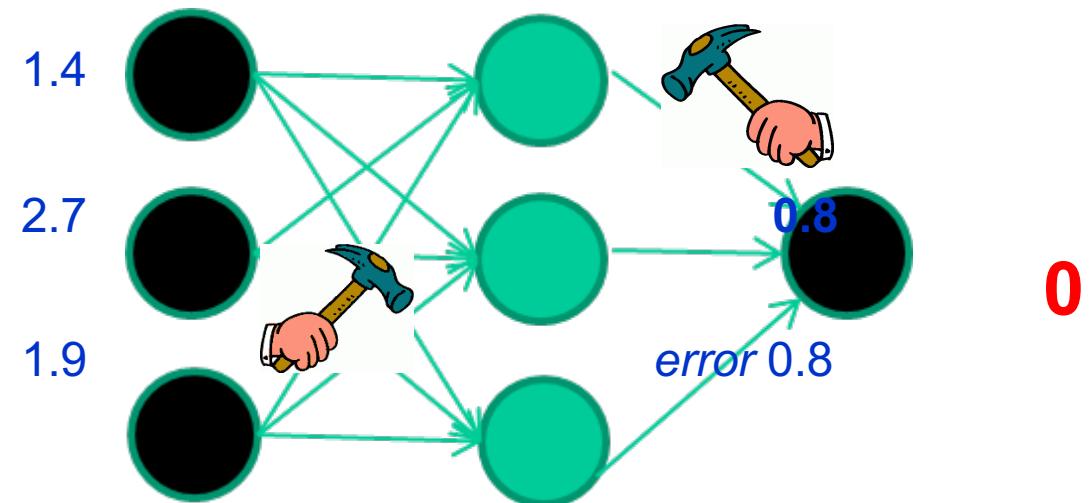
**Compare with target output**



*Training data*

<b>Fields</b>				<b>class</b>
1.4	2.7	1.9	0	
3.8	3.4	3.2	0	
6.4	2.8	1.7	1	
4.1	0.1	0.2	0	
etc ...				

**Adjust weights based on error**



*Training data*

**Fields**                   **class**

1.4 2.7 1.9 0

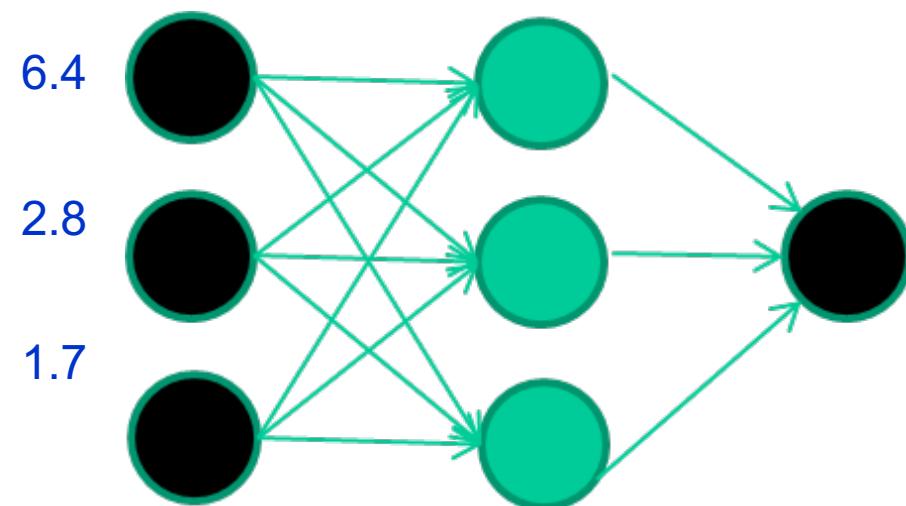
3.8 3.4 3.2 0

6.4 2.8 1.7 1

4.1 0.1 0.2 0

etc

Present a training pattern



*Training data*

**Fields**                   **class**

1.4 2.7 1.9 0

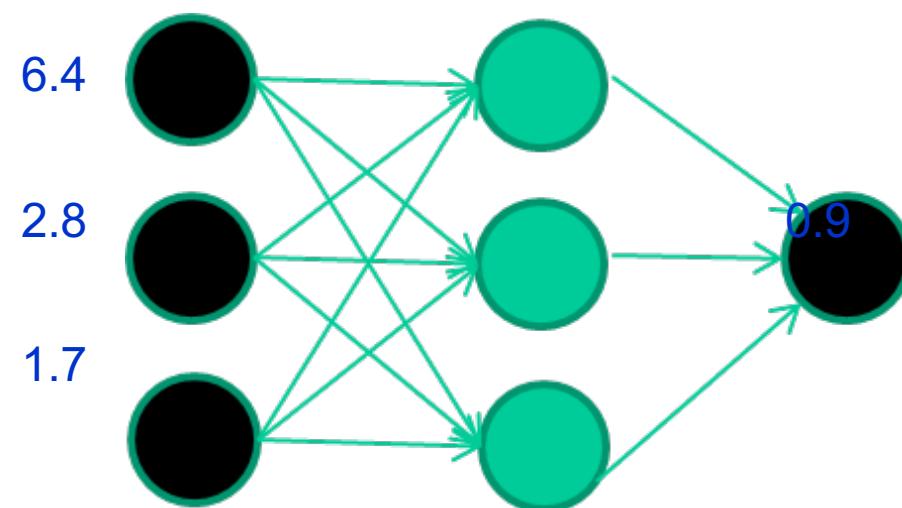
3.8 3.4 3.2 0

6.4 2.8 1.7 1

4.1 0.1 0.2 0

etc

Feed it through to get output



*Training data*

**Fields**                    **class**

1.4 2.7 1.9 0

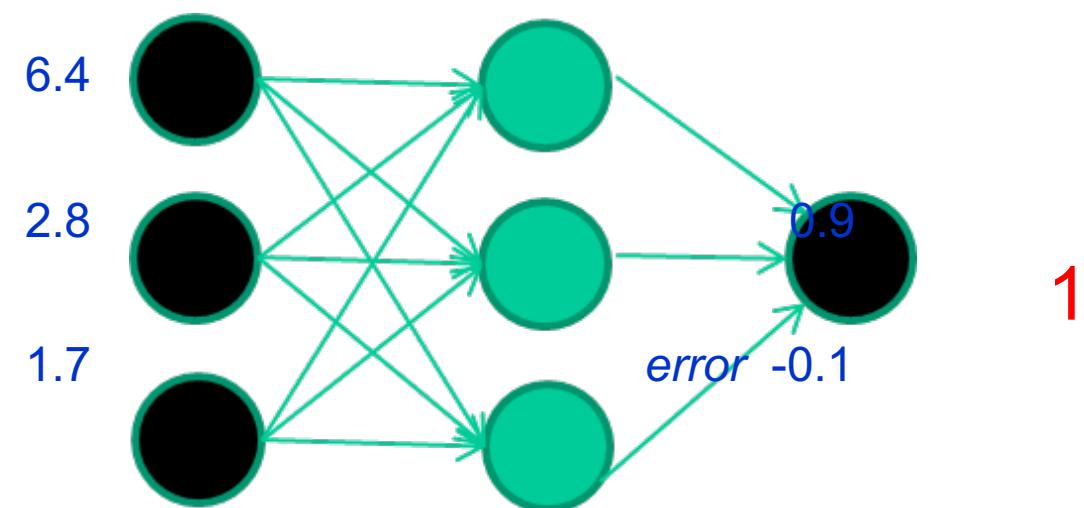
3.8 3.4 3.2 0

6.4 2.8 1.7 1

4.1 0.1 0.2 0

etc

Compare with target output



*Training data*

**Fields**                    **class**

1.4 2.7 1.9 0

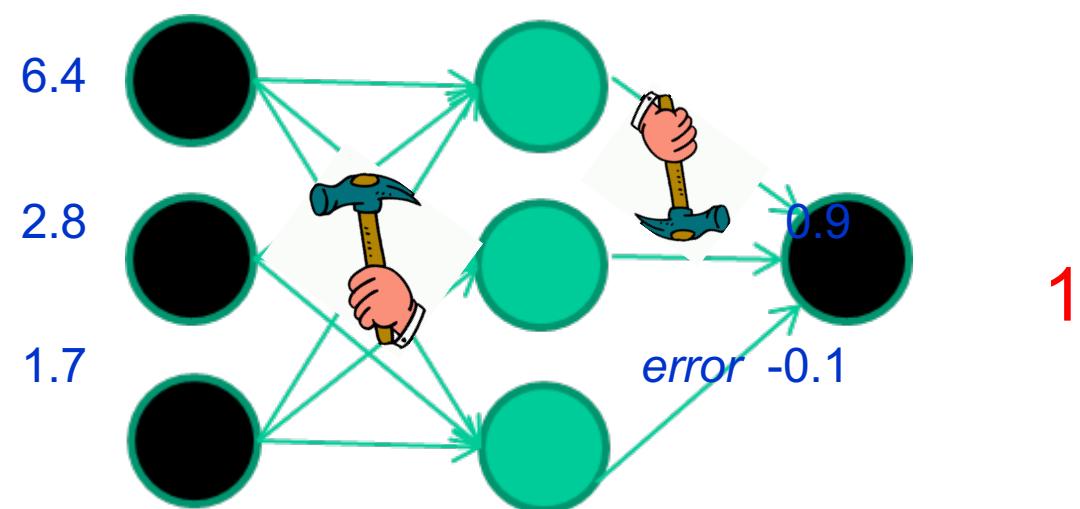
3.8 3.4 3.2 0

6.4 2.8 1.7 1

4.1 0.1 0.2 0

etc

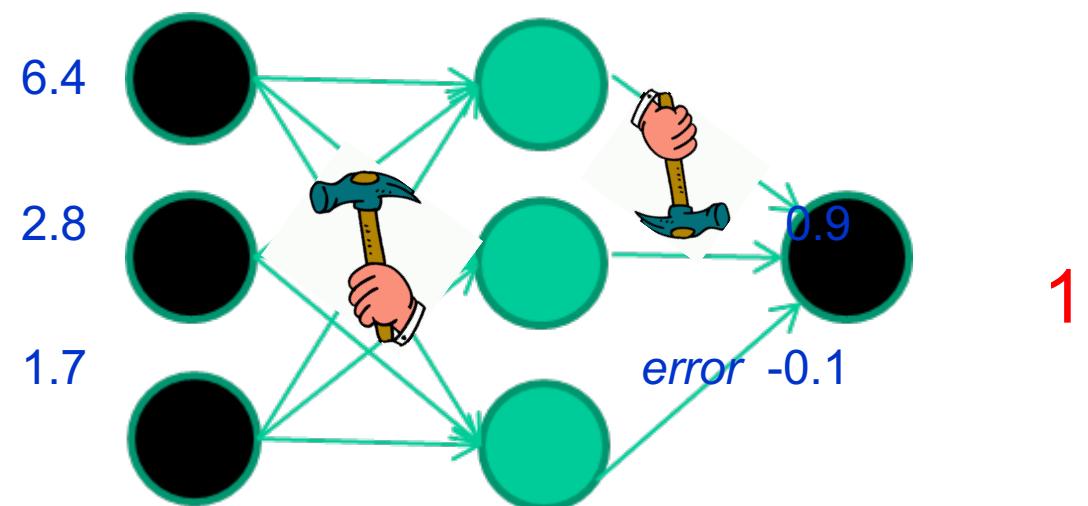
**Adjust weights based on error**



*Training data*

<b>Fields</b>	<b>class</b>		
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc			

**And so on ....**

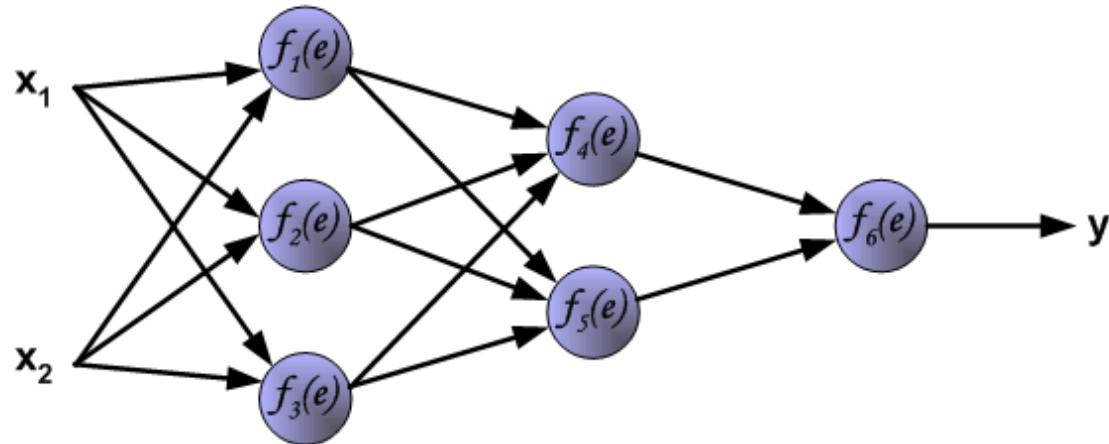


Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments

*Algorithms for weight adjustment are designed to make changes that will reduce the error*

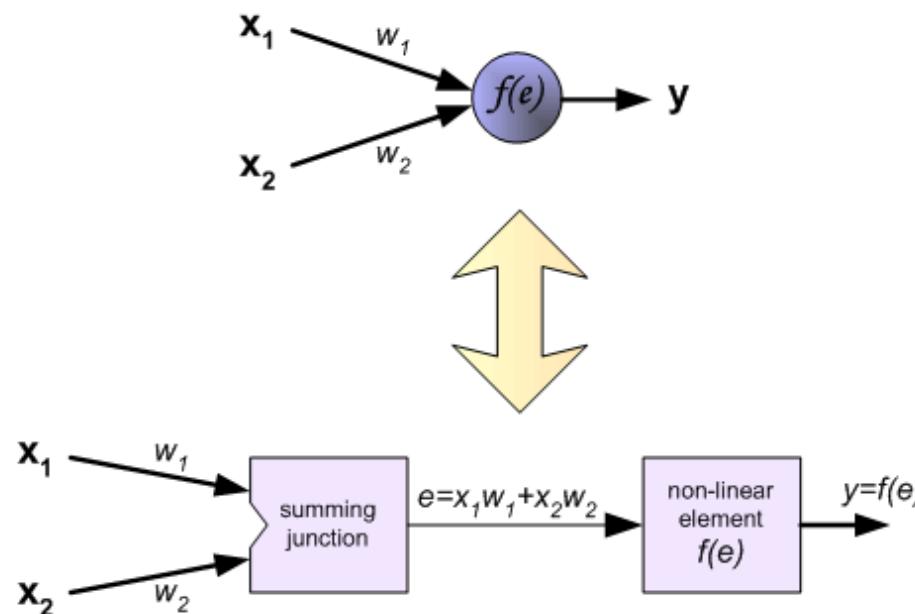
# Learning Algorithm: Backpropagation

The following slides describes **teaching process** of multi-layer neural network employing **backpropagation** algorithm. To illustrate this process the three layer neural network with two inputs and one output, which is shown in the picture below, is used:



# Learning Algorithm: Backpropagation

Each neuron is composed of two units. First unit adds products of weights coefficients and input signals. The second unit realise nonlinear function, called neuron transfer (activation) function. Signal  $e$  is adder output signal, and  $y = f(e)$  is output signal of nonlinear element. Signal  $y$  is also output signal of neuron.



# Learning Algorithm: Backpropagation

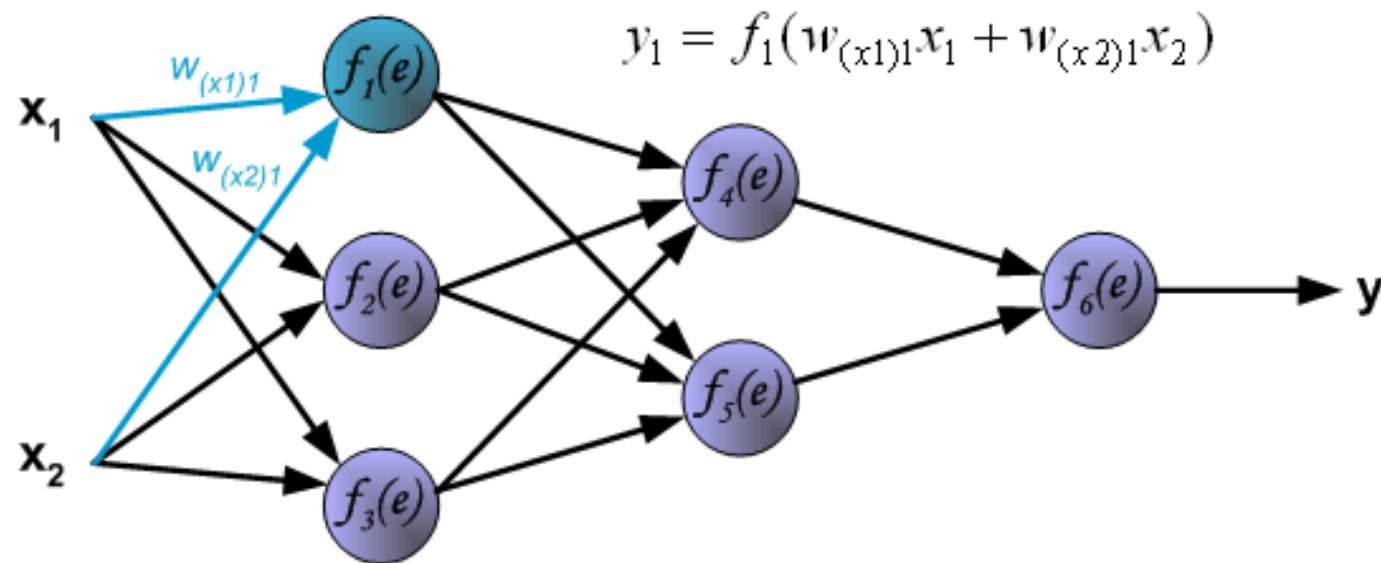
To teach the neural network we need training data set. The training data set consists of input signals ( $x_1$  and  $x_2$ ) assigned with corresponding target (desired output) z.

The network training is an iterative process. In each iteration weights coefficients of nodes are modified using new data from training data set. Modification is calculated using algorithm described below:

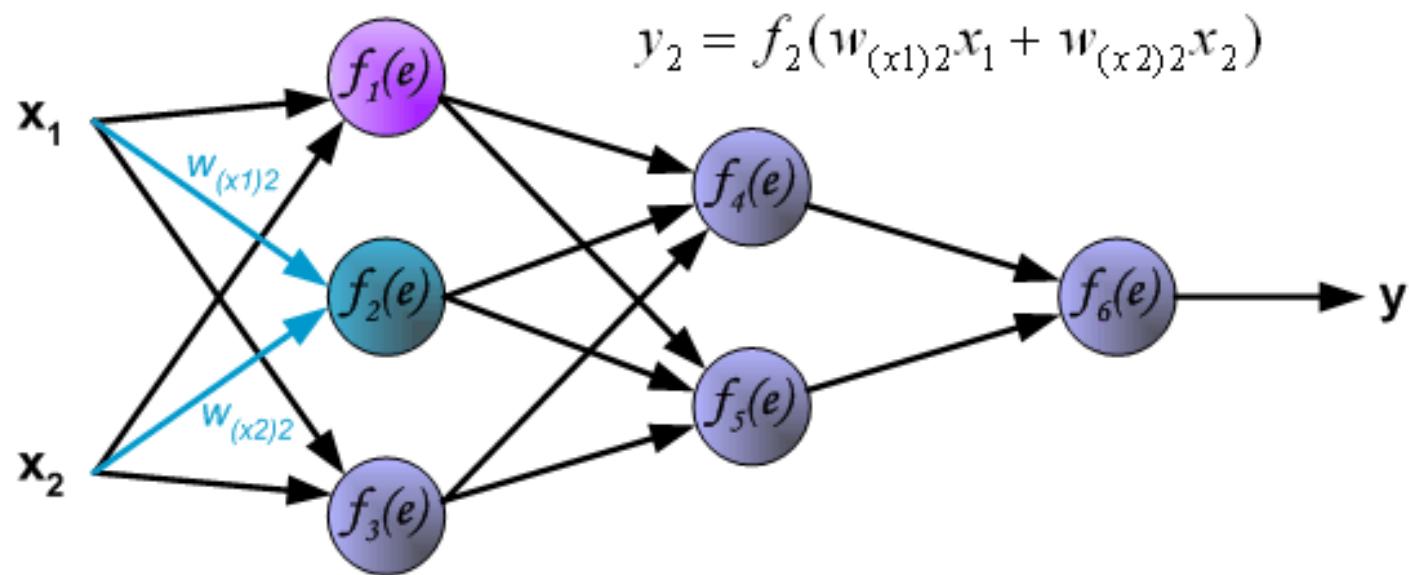
Each teaching step starts with forcing both input signals from training set. After this stage we can determine output signals values for each neuron in each network layer.

# Learning Algorithm: Backpropagation

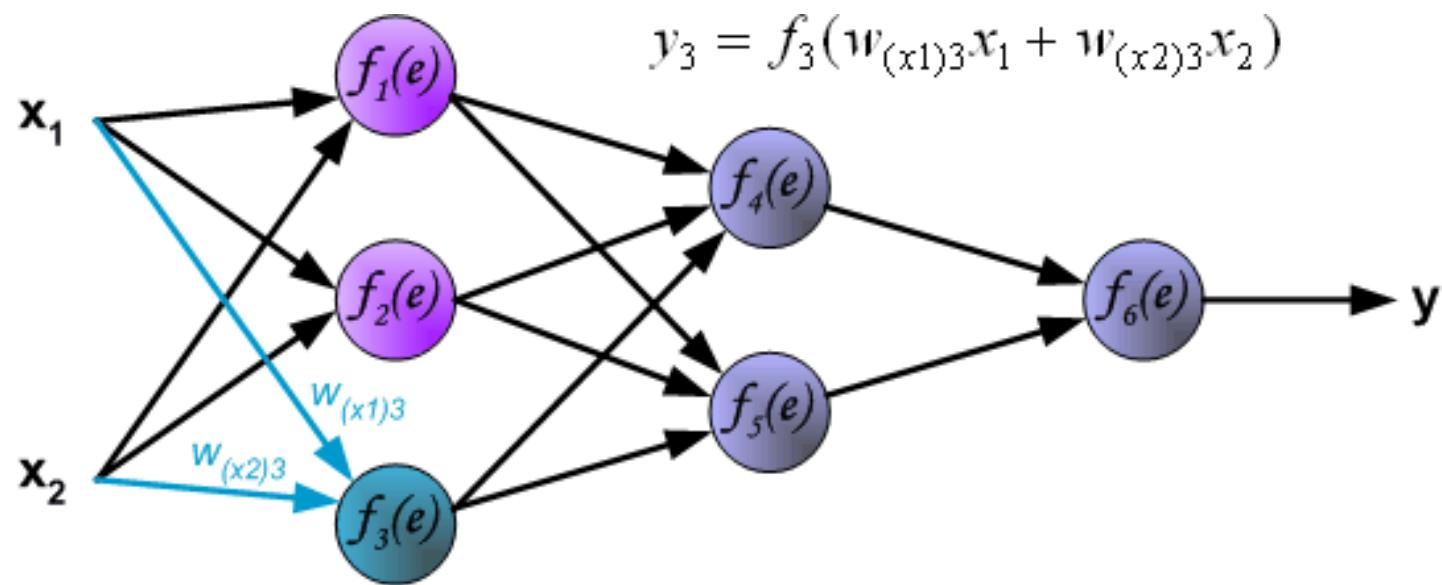
Pictures below illustrate how signal is propagating through the network,  
Symbols  $w_{(xm)n}$  represent weights of connections between network input  $x_m$  and  
neuron  $n$  in input layer. Symbols  $y_n$  represents output signal of neuron  $n$ .



# Learning Algorithm: Backpropagation

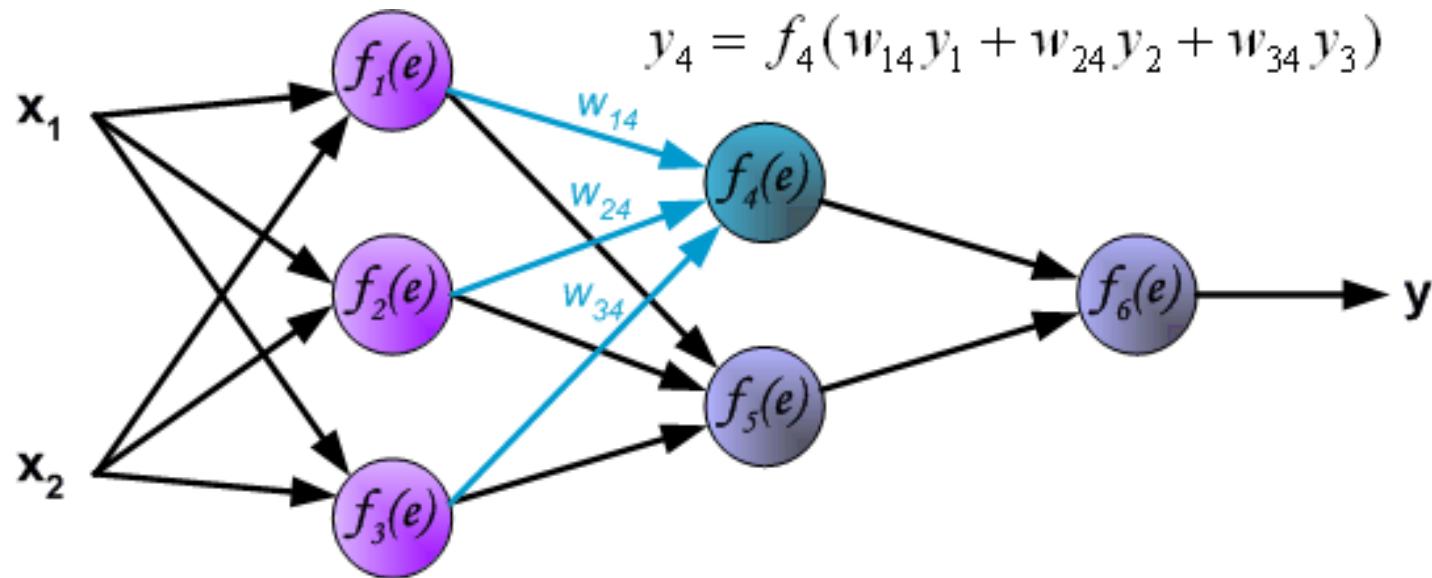


# Learning Algorithm: Backpropagation

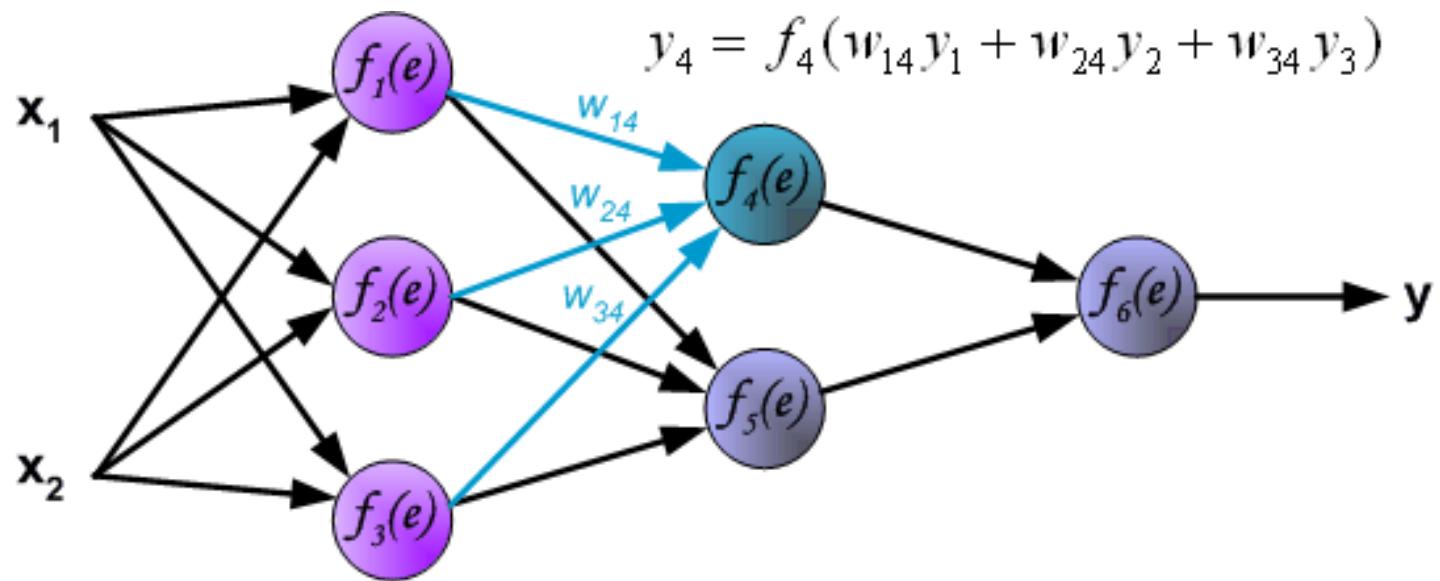


# Learning Algorithm: Backpropagation

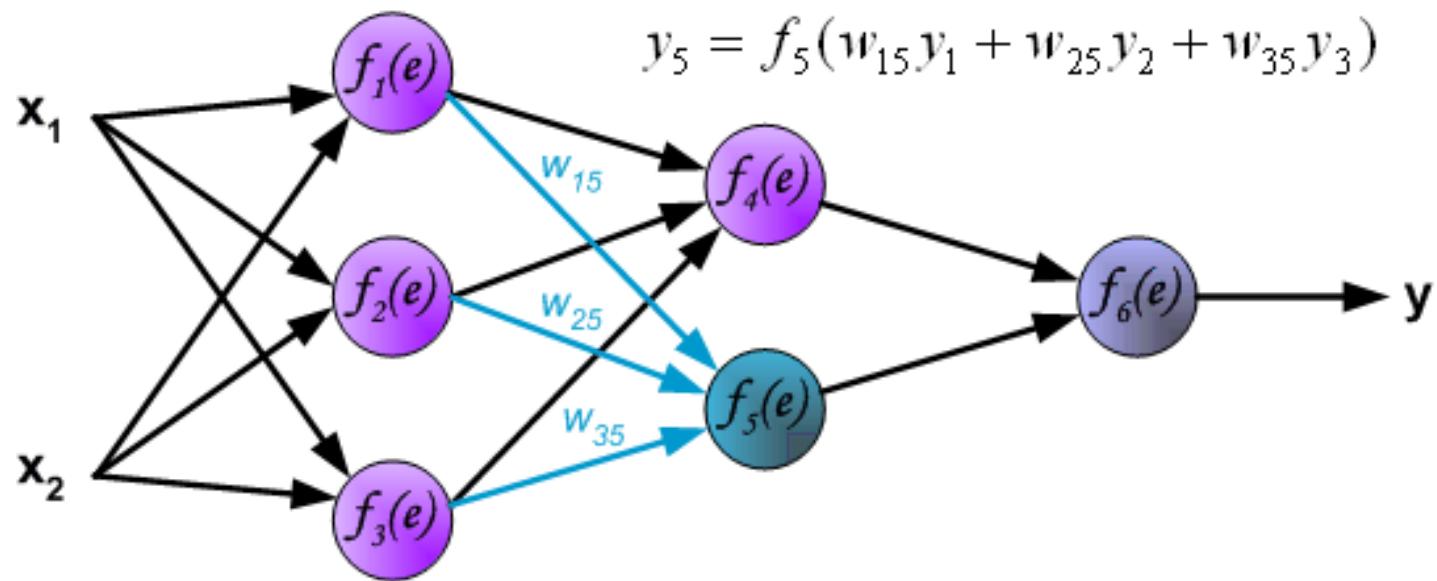
Propagation of signals through the hidden layer. Symbols  $w_{mn}$  represent weights of connections between output of neuron  $m$  and input of neuron  $n$  in the next layer.



# Learning Algorithm: Backpropagation

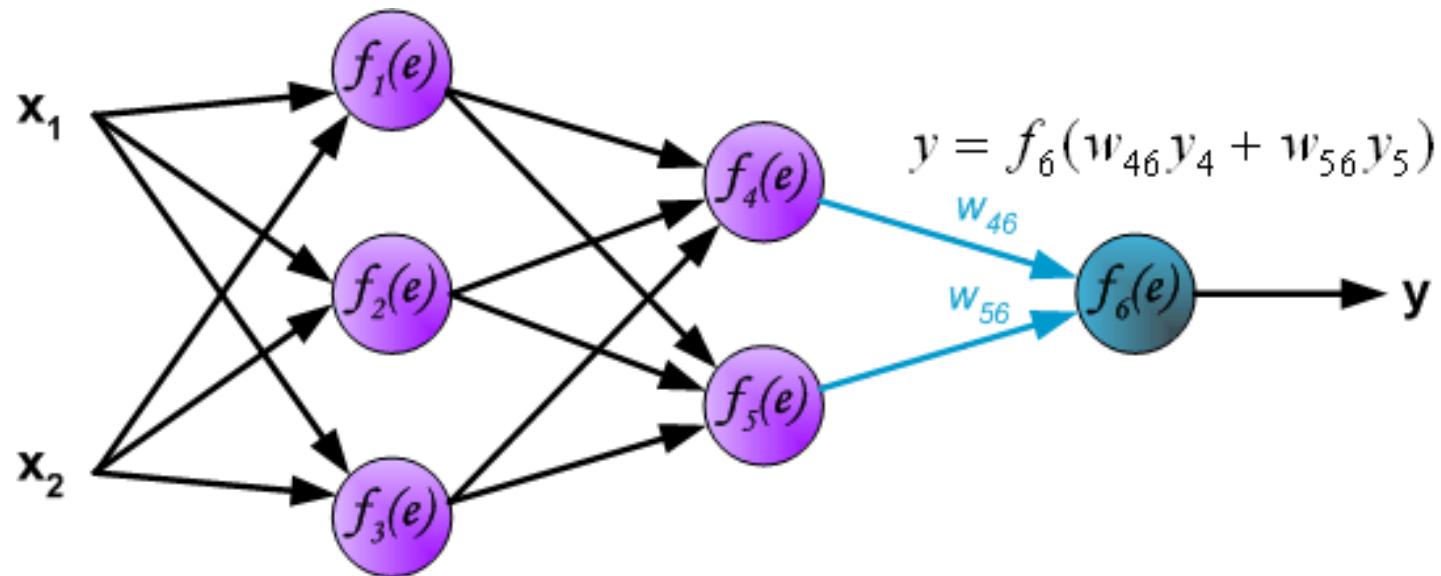


# Learning Algorithm: Backpropagation



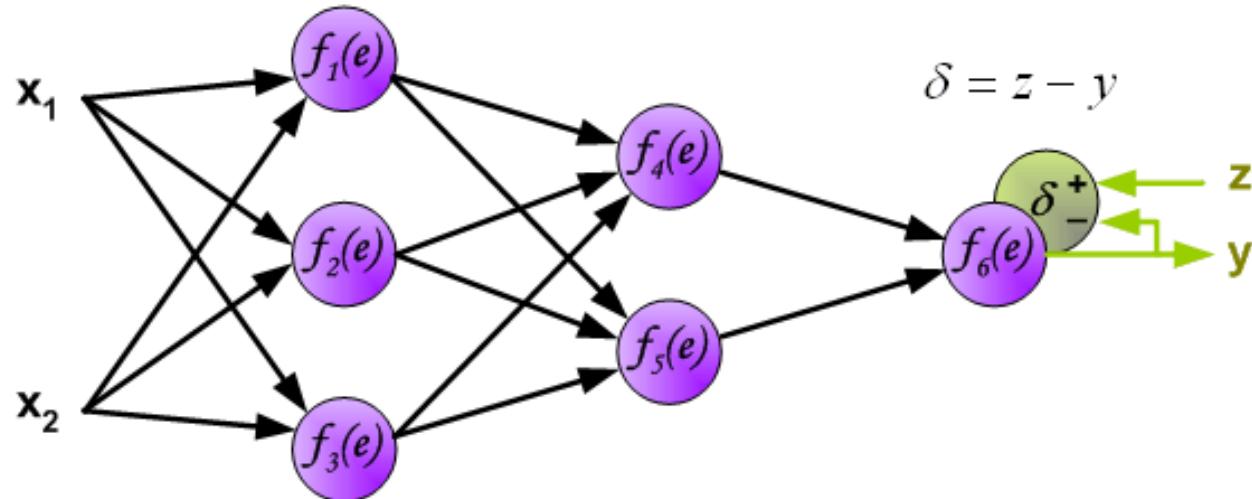
# Learning Algorithm: Backpropagation

Propagation of signals through the output layer.



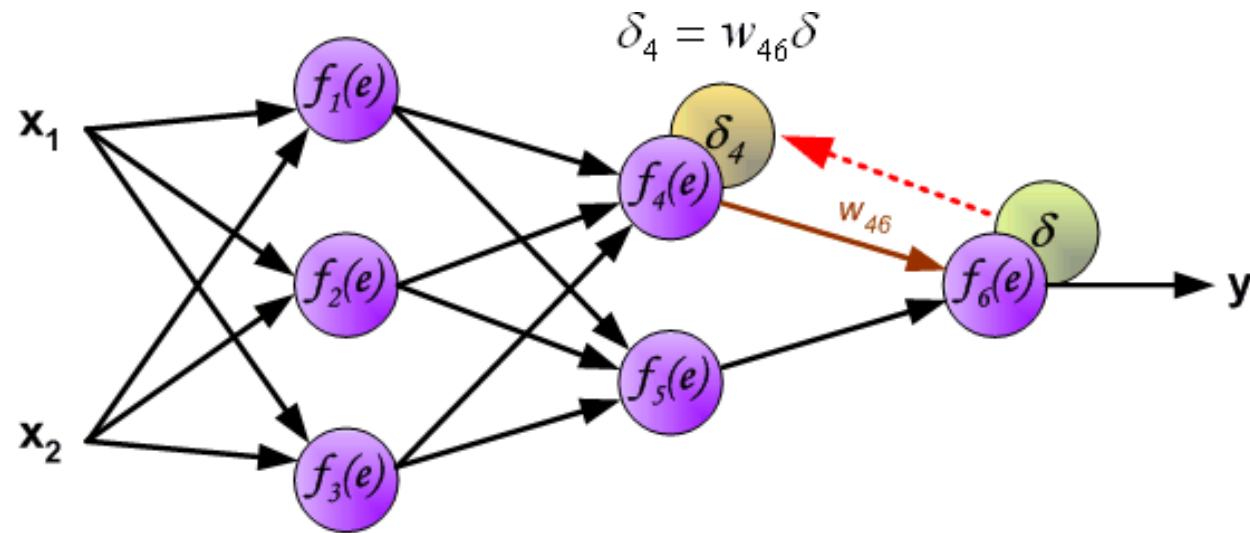
# Learning Algorithm: Backpropagation

In the next algorithm step the output signal of the network  $y$  is compared with the desired output value (the target), which is found in training data set. The difference is called error signal  $d$  of output layer neuron



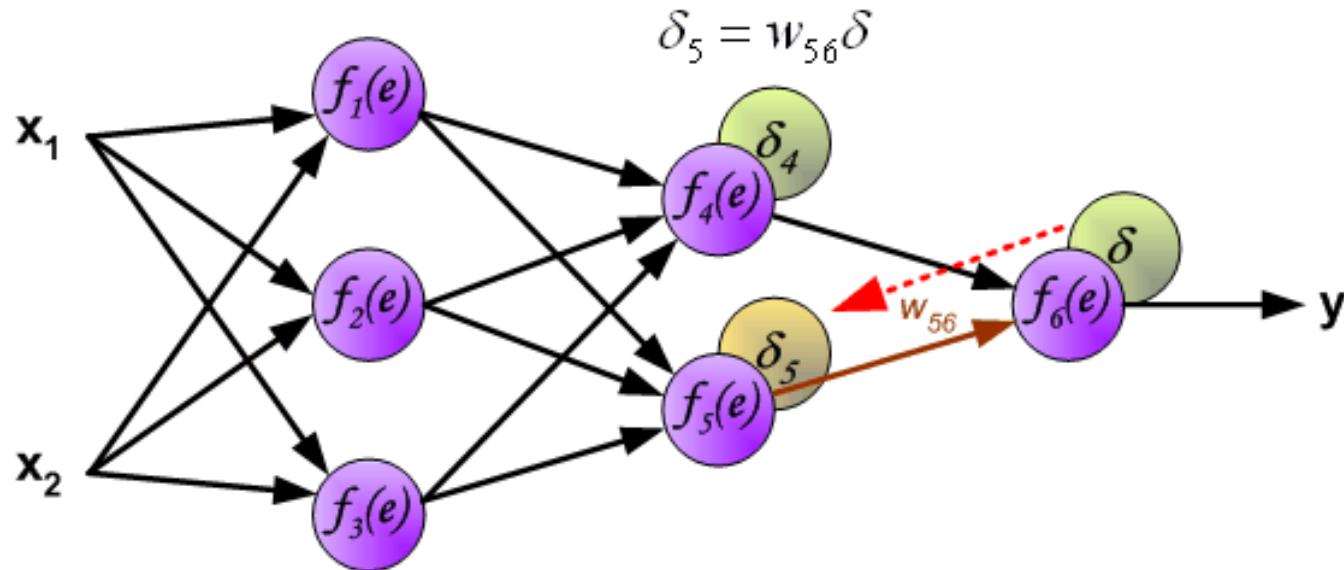
# Learning Algorithm: Backpropagation

The idea is to propagate error signal  $d$  (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.



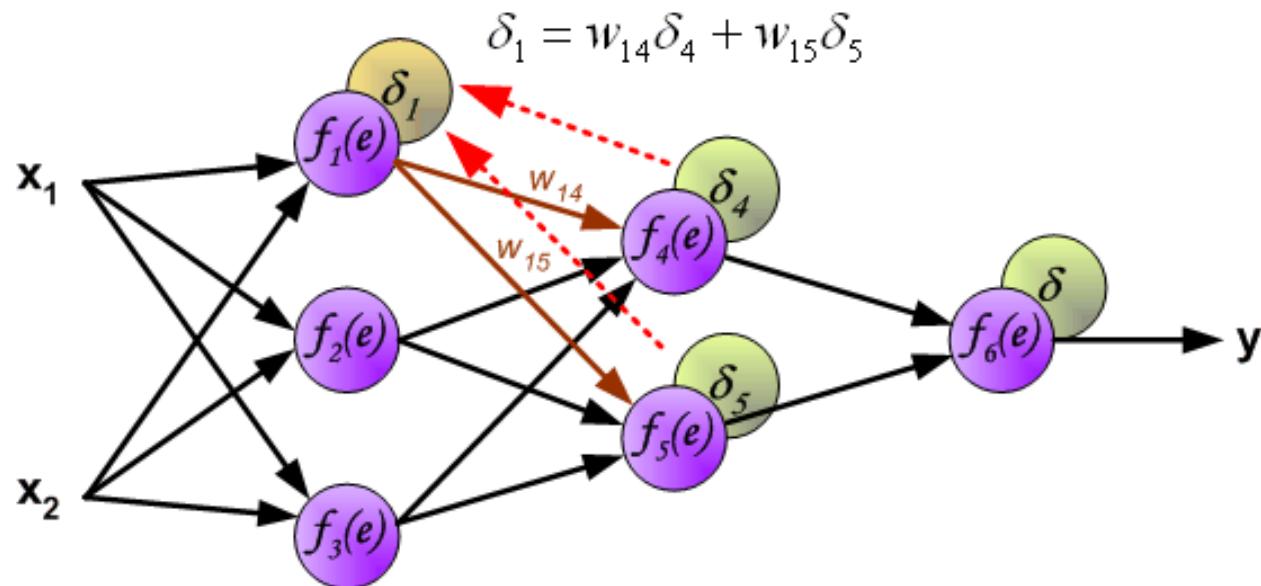
# Learning Algorithm: Backpropagation

The idea is to propagate error signal  $d$  (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.



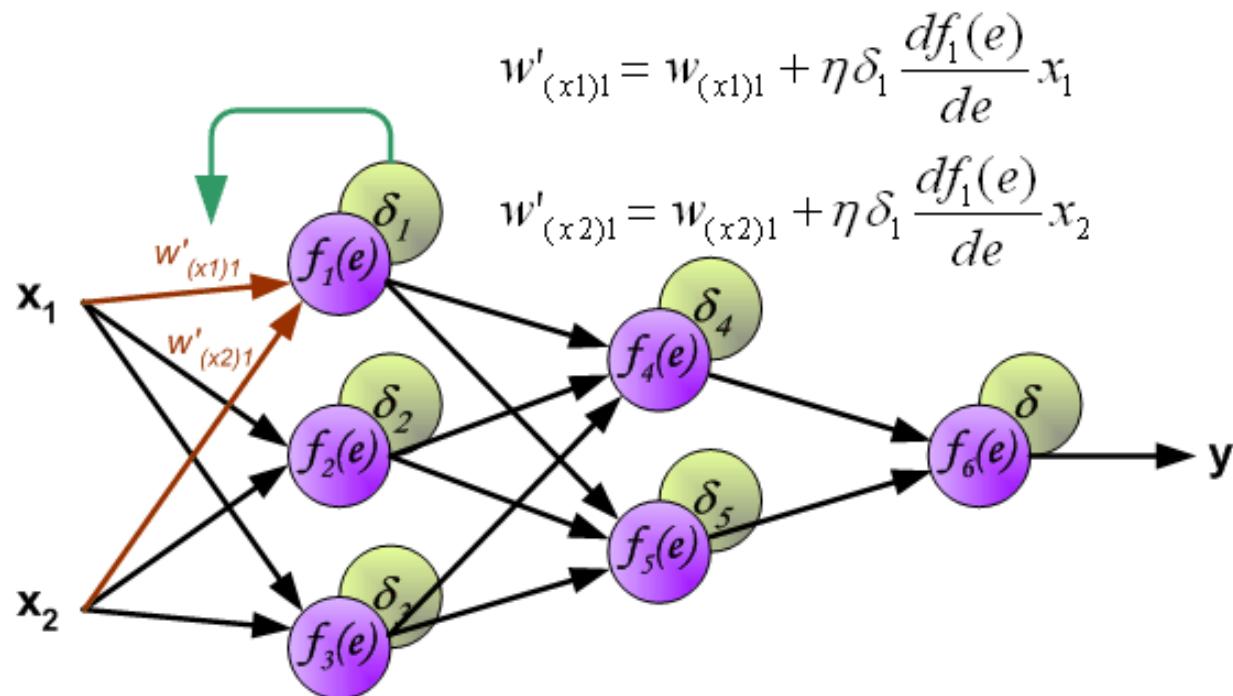
# Learning Algorithm: Backpropagation

The weights' coefficients  $w_{mn}$  used to propagate errors back are equal to this used during computing output value. Only the direction of data flow is changed (signals are propagated from output to inputs one after the other). This technique is used for all network layers. If propagated errors came from few neurons they are added. The illustration is below:



# Learning Algorithm: Backpropagation

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below  $df(e)/de$  represents derivative of neuron activation function (which weights are modified).

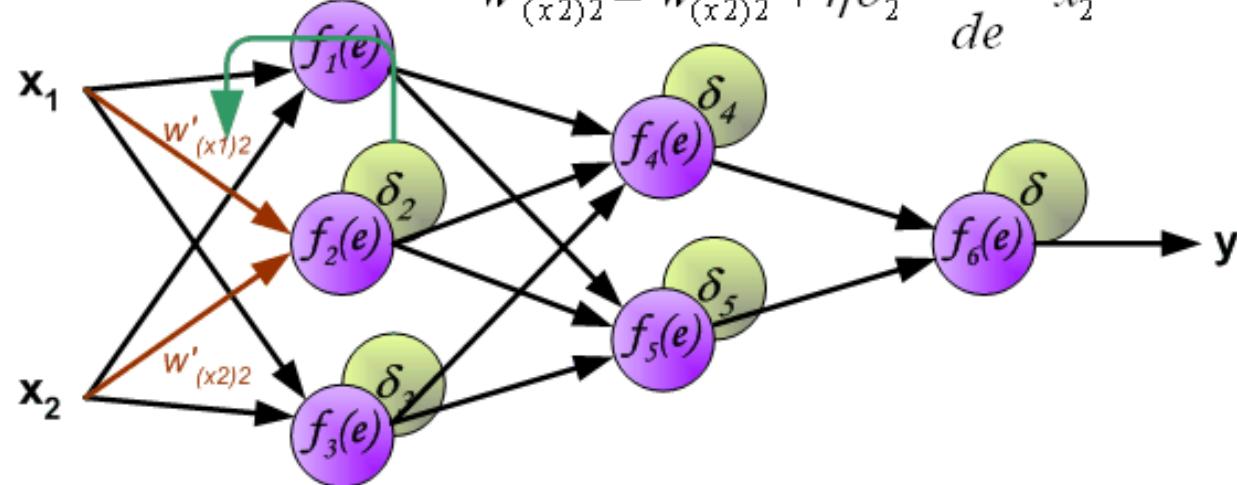


# Learning Algorithm: Backpropagation

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below  $df(e)/de$  represents derivative of neuron activation function (which weights are modified).

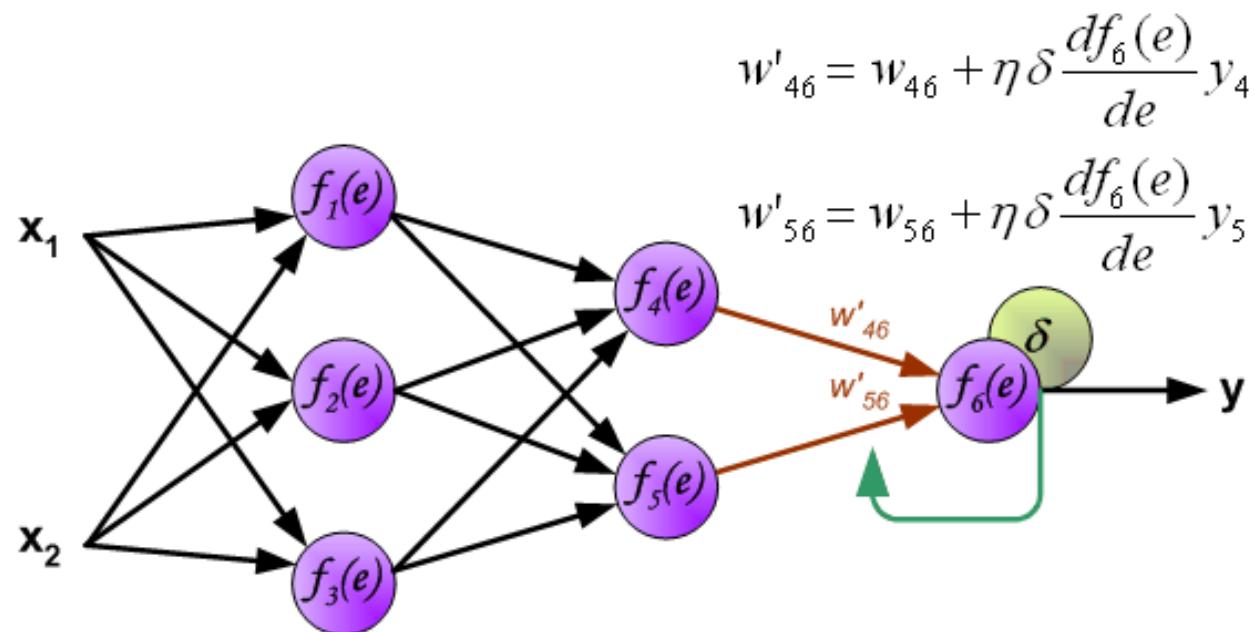
$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$



# Learning Algorithm: Backpropagation

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below  $df(e)/de$  represents derivative of neuron activation function (which weights are modified).



<https://playground.tensorflow.org/>

https://google-  
developers.appspot.com/machine-  
learning/crash-course/backprop-  
scroll/

A mostly complete chart of  
**Neural Networks**

©2016 Fjodor van Veen - asimovinstitute.org

Backfed Input Cell

Input Cell

Noisy Input Cell

Hidden Cell

Probabilistic Hidden Cell

Spiking Hidden Cell

Output Cell

Match Input Output Cell

Recurrent Cell

Memory Cell

Different Memory Cell

Kernel

Convolution or Pool

Perceptron (P)

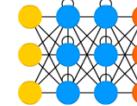
Feed Forward (FF)

Radial Basis Network (RBF)

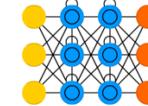
Deep Feed Forward (DFF)



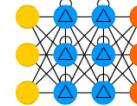
Recurrent Neural Network (RNN)



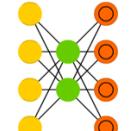
Long / Short Term Memory (LSTM)



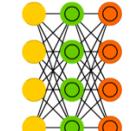
Gated Recurrent Unit (GRU)



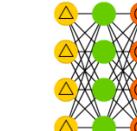
Auto Encoder (AE)



Variational AE (VAE)



Denoising AE (DAE)



Sparse AE (SAE)



Markov Chain (MC)



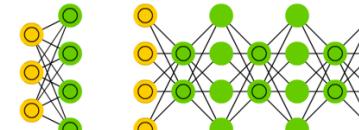
Hopfield Network (HN)



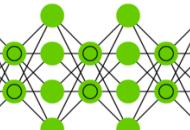
Boltzmann Machine (BM)



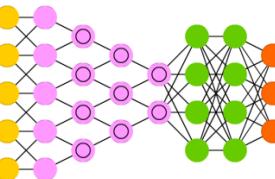
Restricted BM (RBM)



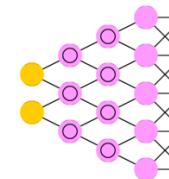
Deep Belief Network (DBN)



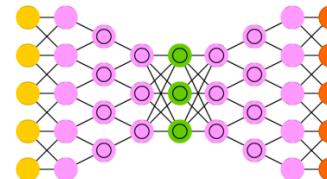
Deep Convolutional Network (DCN)



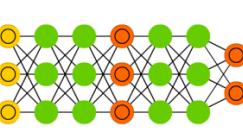
Deconvolutional Network (DN)



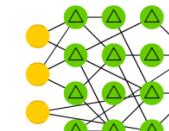
Deep Convolutional Inverse Graphics Network (DCIGN)



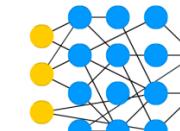
Generative Adversarial Network (GAN)



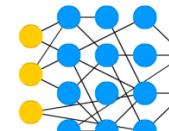
Liquid State Machine (LSM)



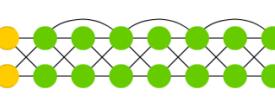
Extreme Learning Machine (ELM)



Echo State Network (ESN)



Deep Residual Network (DRN)



Kohonen Network (KN)



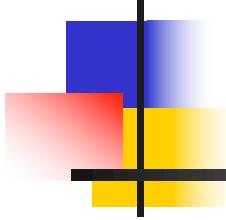
Support Vector Machine (SVM)



Neural Turing Machine (NTM)



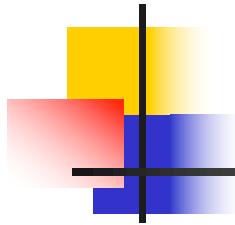
Thank You



# Convolutional Networks

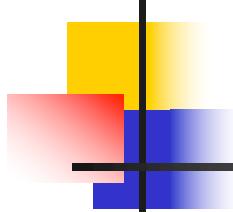
Anand Kumar M  
IT, NITK

Slides Modified from Dr. Charles Tappert / Good



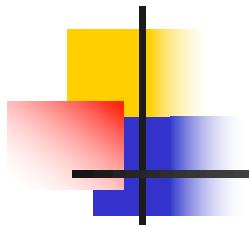
# Chapter 9 Sections

- Introduction
- 1 The Convolution Operation
- 2 Motivation
- 3 Pooling
- 4 Convolution and Pooling as an Infinitely Strong Prior
- 5 Variants of the Basic Convolution Function
- 6 Data Types



# Introduction

- Convolutional networks are also known as convolutional neural networks (CNNs)
- Specialized for data having grid-like topology
  - 1D grid – time series data
  - 2D grid – image data
- Definition
  - Convolutional networks use **convolution in place of general matrix multiplication** in at least one layer
  - Neural network convolution does not correspond to convolution used in engineering and mathematics



# 1. The Convolution Operation

- Convolution is an operation on two functions
  - Section begins with general convolution example
    - Signal smoothing in locating spaceship with a laser sensor
- CNN convolutions (not general convolution)
  - First function is network input  $x$ , second is kernel  $w$
  - **Tensors** refer to the multidimensional arrays
    - E.g., input data and parameter arrays, thus TensorFlow
  - The convolution **kernel** is usually a sparse matrix in contrast to the usual fully-connected weight matrix

# 2D Convolution

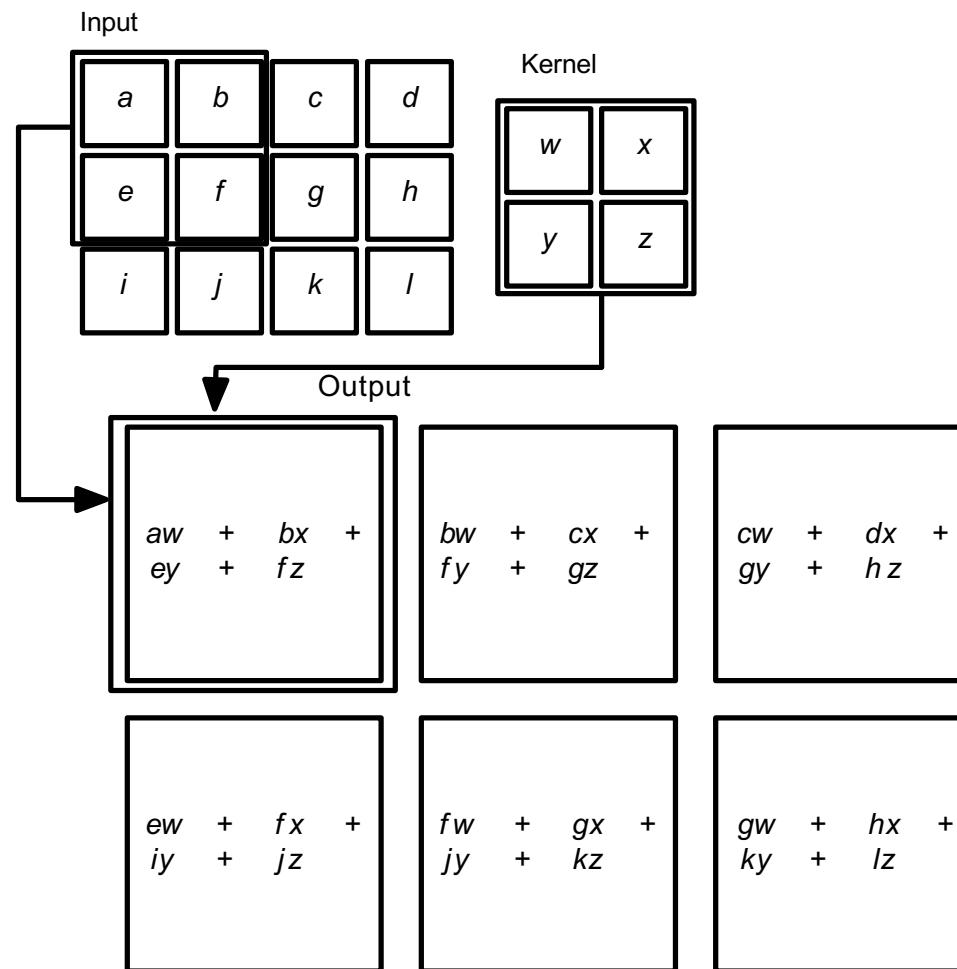


Figure 9.5

(Goodfellow 2016)

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

\*

1	0	-1
1	0	-1
1	0	-1

=

6		

$7x1 + 4x1 + 3x1 +$   
 $2x0 + 5x0 + 3x0 +$   
 $3x-1 + 3x-1 + 2x-1$   
 $= 6$

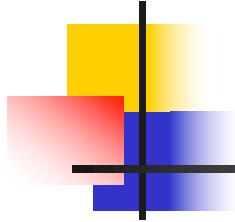
Finally, we often use convolutions over more than one axis at a time. For example, if we use a two-dimensional image  $I$  as our input, we probably also want to use a two-dimensional kernel  $K$ :

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n). \quad (9.4)$$

Convolution is commutative, meaning we can equivalently write

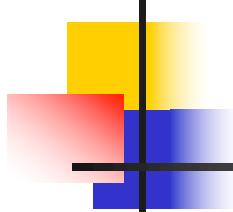
$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n). \quad (9.5)$$

## Figure 9.6



## 2. Motivation

- Convolution leverages three important ideas that help improve machine learning systems
  1. Sparse interactions
  2. Parameter sharing
  3. Equivariant representations



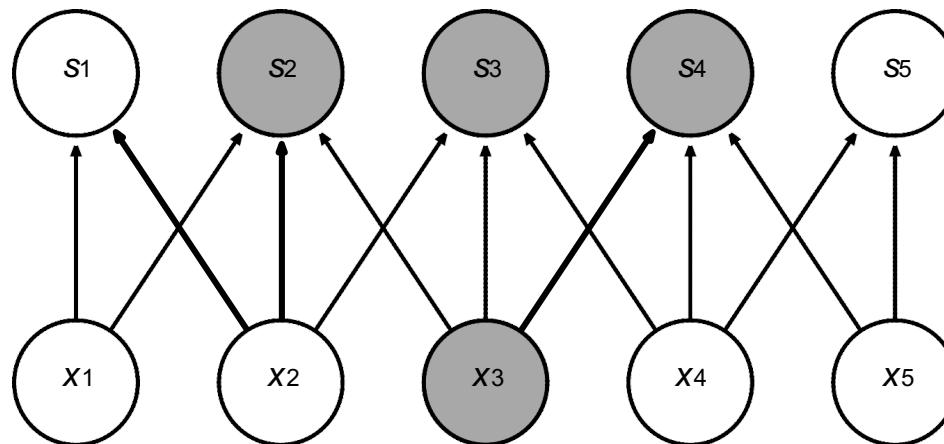
## 2. Motivation

# Sparse Interactions

- Fully connected traditional networks
  - $m$  inputs in a layer and  $n$  outputs in next layer
  - requires  $O(m \times n)$  runtime (per example)
- Sparse interactions
  - Also called **sparse connectivity** or **sparse weights**
  - Accomplished by making kernel smaller than input
    - $k \ll m$  requires  $O(k \times n)$  runtime (per example)
    - $k$  is typically several orders of magnitude smaller than  $m$

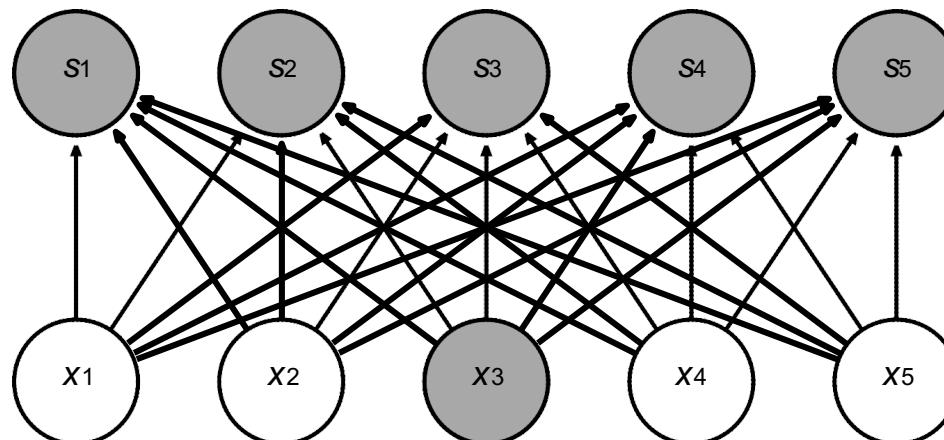
# Sparse Connectivity

Sparse connections due to small convolution kernel



Viewed from below

Dense connections  
Fully connected

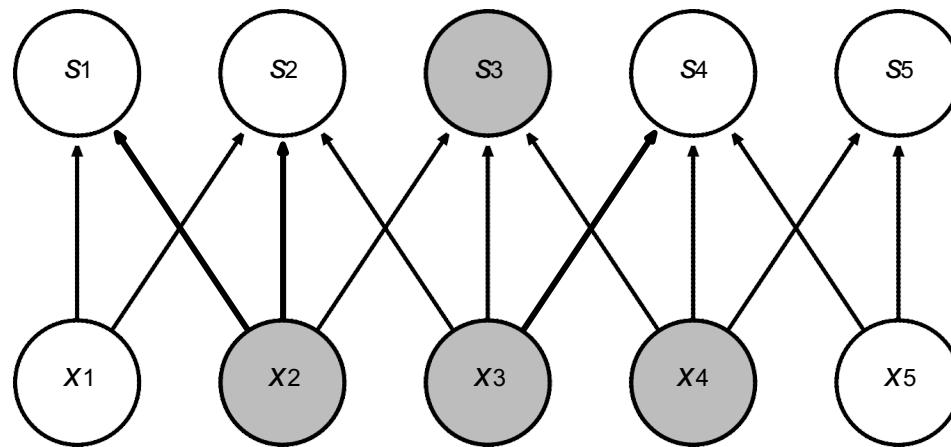


*highlight one input unit,  $X_3$ , and highlight the output units in  $S$  that are affected by this unit*

Figure 9.9

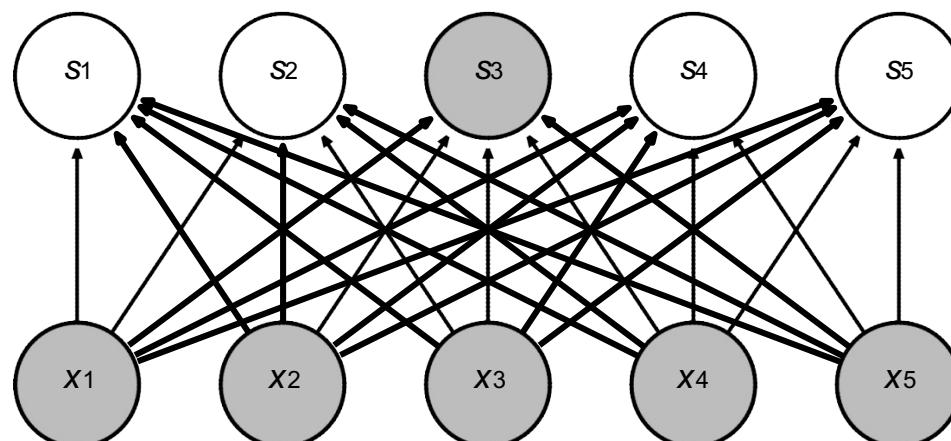
# Sparse Connectivity

Sparse connections due to small convolution kernel



Viewed from above (receptive fields)

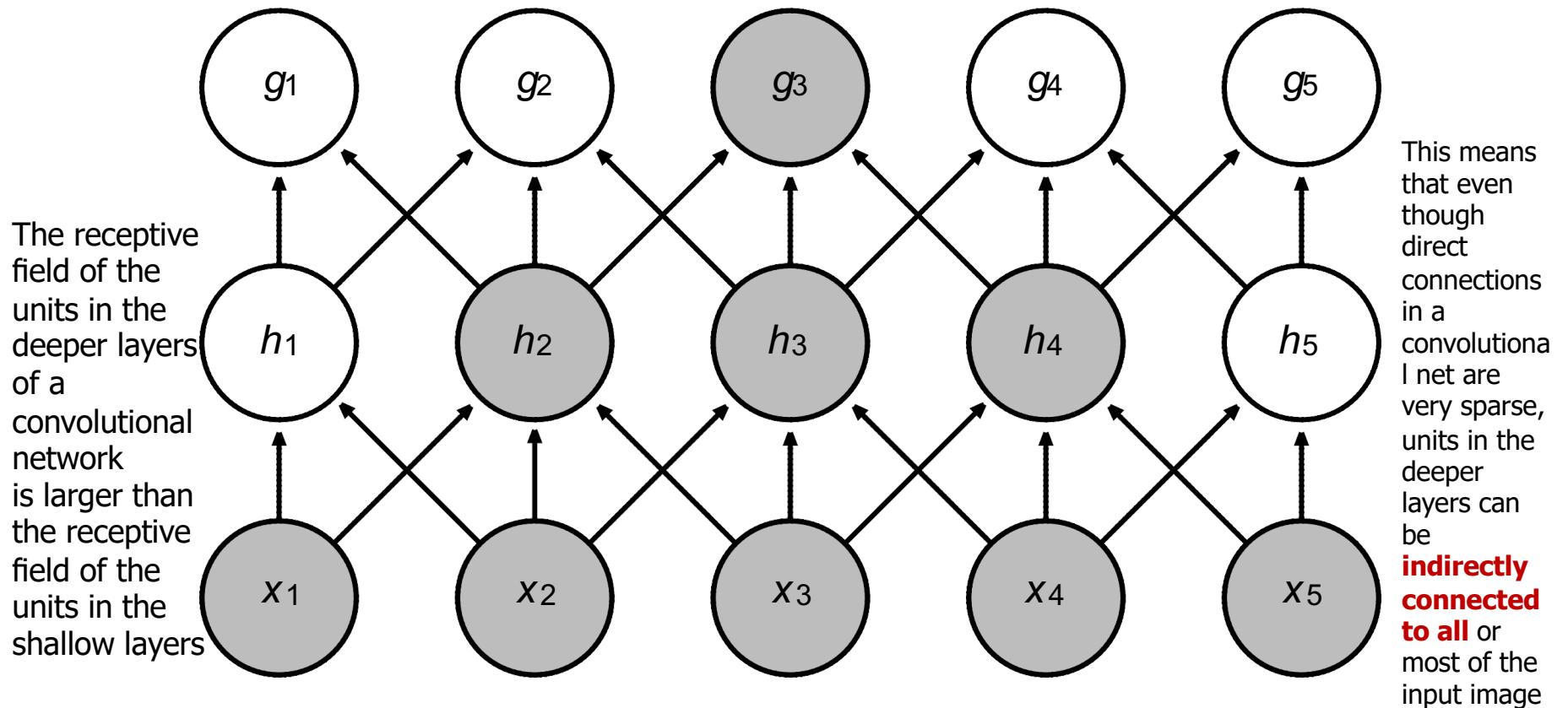
Dense connections  
Fully connected



We highlight one output unit, s3, and highlight the input units in X that affect this unit. These **units are known as the receptive field** of s3

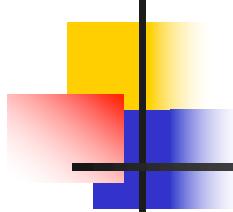
Figure

# Growing Receptive Fields



Deeper layer units have larger receptive fields

Figure



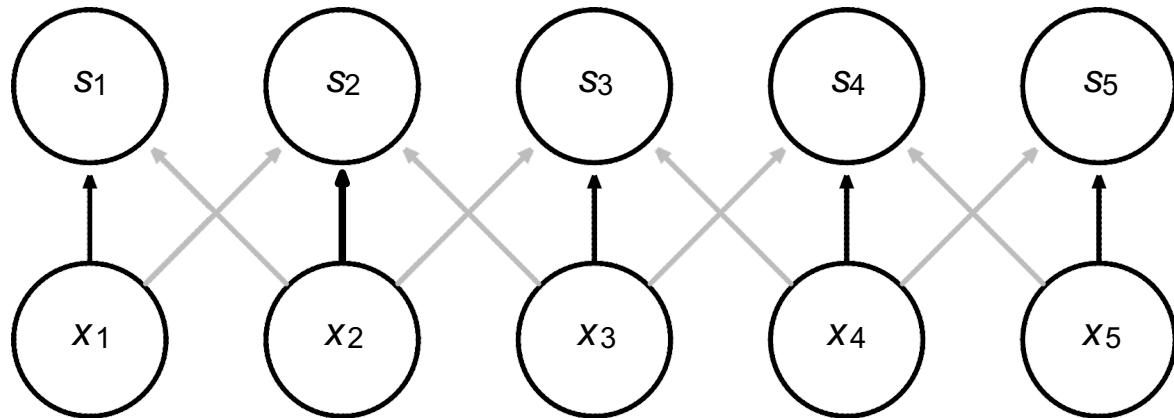
## 2. Motivation Parameter Sharing

- In traditional neural networks
  - Each element of the weight matrix is unique
- **Parameter sharing** mean using the same parameters for more than one model function
  - The network has **tied weights**
  - Reduces storage requirements to  $k$  parameters
  - Does not affect forward prop runtime  $O(k \times n)$

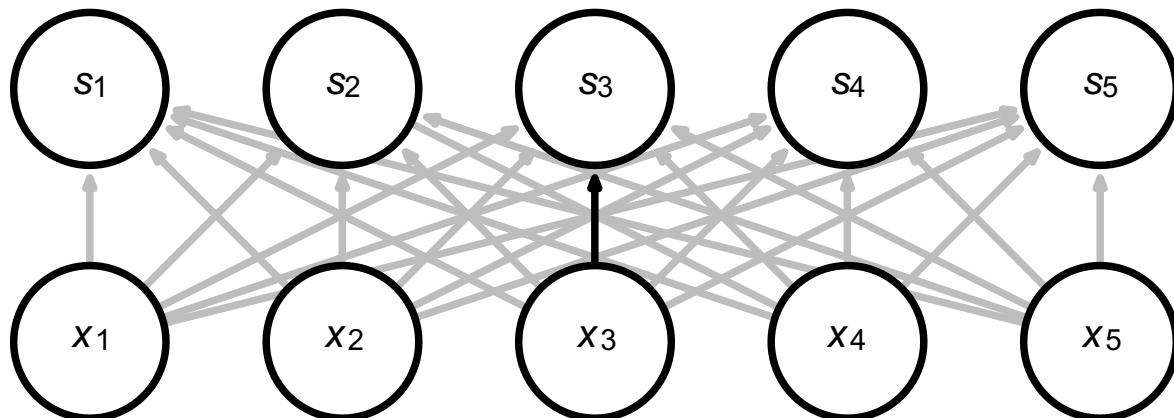
# Parameter Sharing

Black arrows = particular parameter

Convolution  
shares the same  
parameters  
across all spatial  
locations



Traditional matrix  
multiplication  
does not share  
any parameters



Figure

# Edge Detection by Convolution



Input

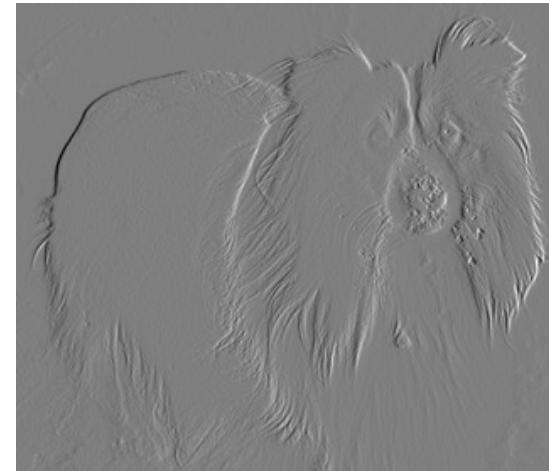


-1 1

Kernel

$k = 2$

Right image = each orig pixel – left pixel  
detects edges



Output

Figure

# Efficiency of Convolution

Input size: 320 by 280

Kernel size: 2 by 1

Output size: 319 by 280

	Convolution	Dense matrix Fully connected	Sparse matrix
Stored floats Each weight	2	$319*280*320*280$ $> 8e9$	$2*319*280 =$ 178,640
Float mults+adds Forward computation	$319*280*3 =$ 267,960	$> 16e9$	Same as convolution (267,960)

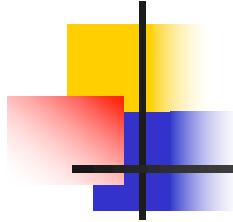
Figure

(GuruFellow 2016)

## 2. Motivation

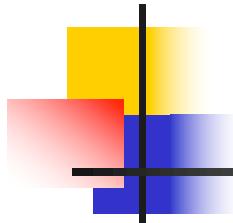
# Equivariant Representations

- For an Equivariant function, if the input changes, the output change in the same way
- For convolution, a particular form of parameter sharing causes **equivariance to translation**
  - In image processing, detecting edges is useful in the first layer, and edges appear more or less everywhere in the image
    - *Equivariant to translation means that a translation of input features results in an equivalent translation of output*
    - *Invariant to translation means that a translation of input features does not change the output at all*



## 3. Pooling

- The pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs
  - **Max pooling** reports the maximum output within a rectangular neighborhood
  - **Average pooling** reports the average output
- Pooling helps make the representation approximately invariant to small input translations



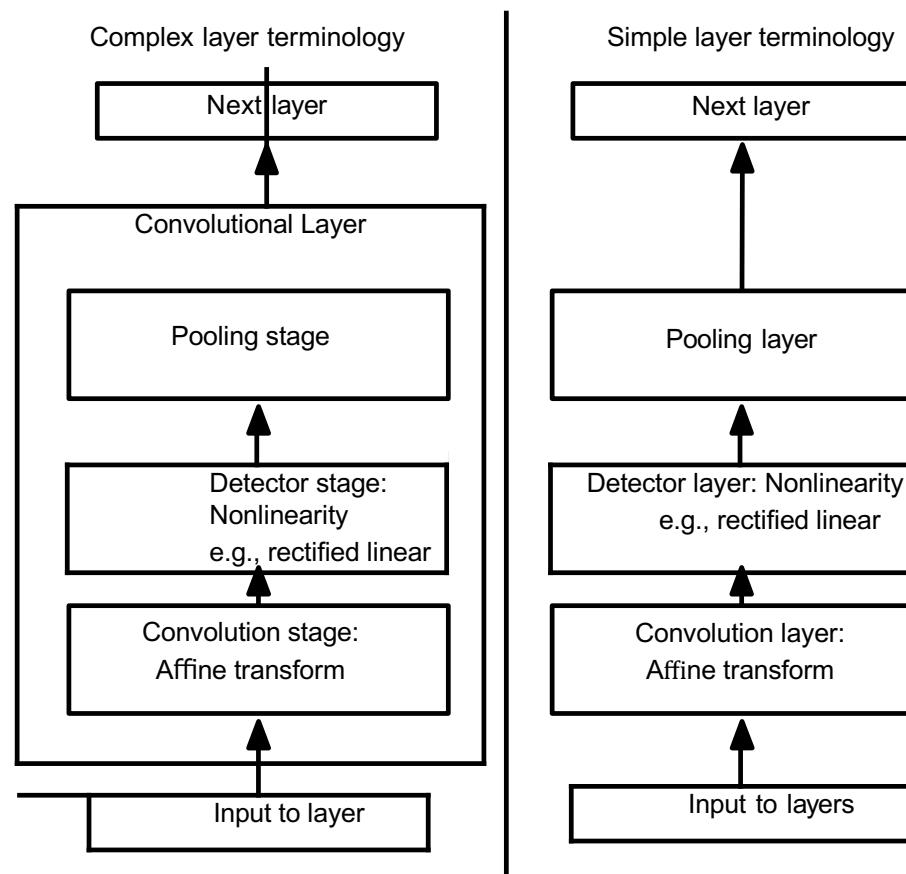
12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

$2 \times 2$  Max-Pool

20	30
112	37

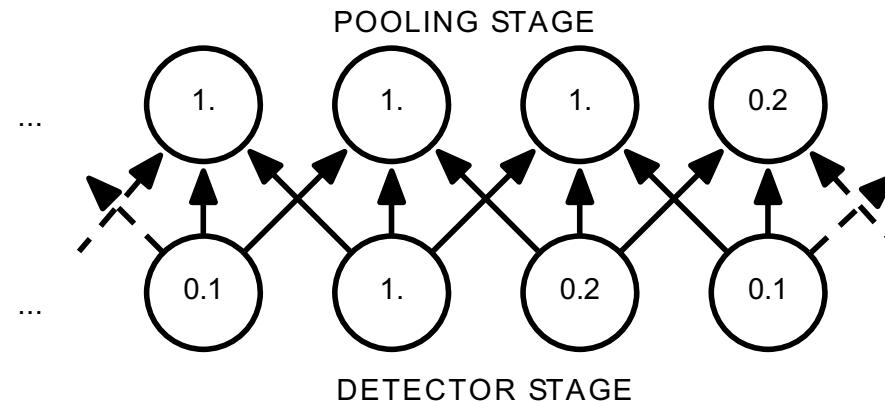
# Convolutional Network Components

Two common terminologies



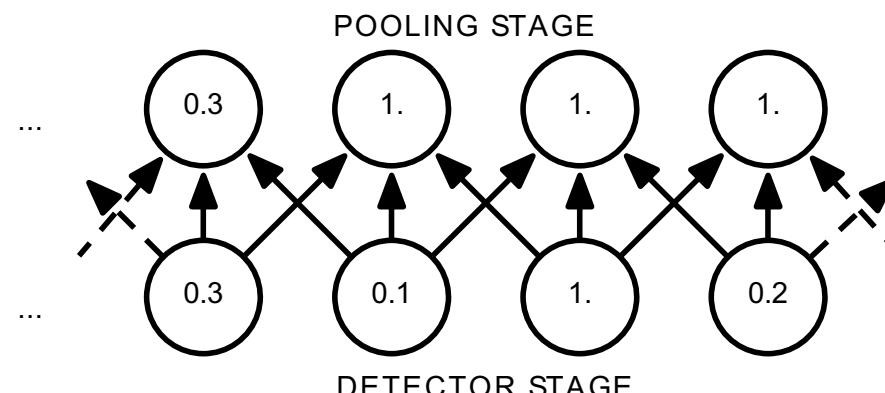
Figure

# Max Pooling and Invariance to Translation



The top row shows the outputs of max pooling, with a stride of one pixel between pooling regions and a pooling region width of three pixels.

Same network  
with input  
shifted one  
pixel to right

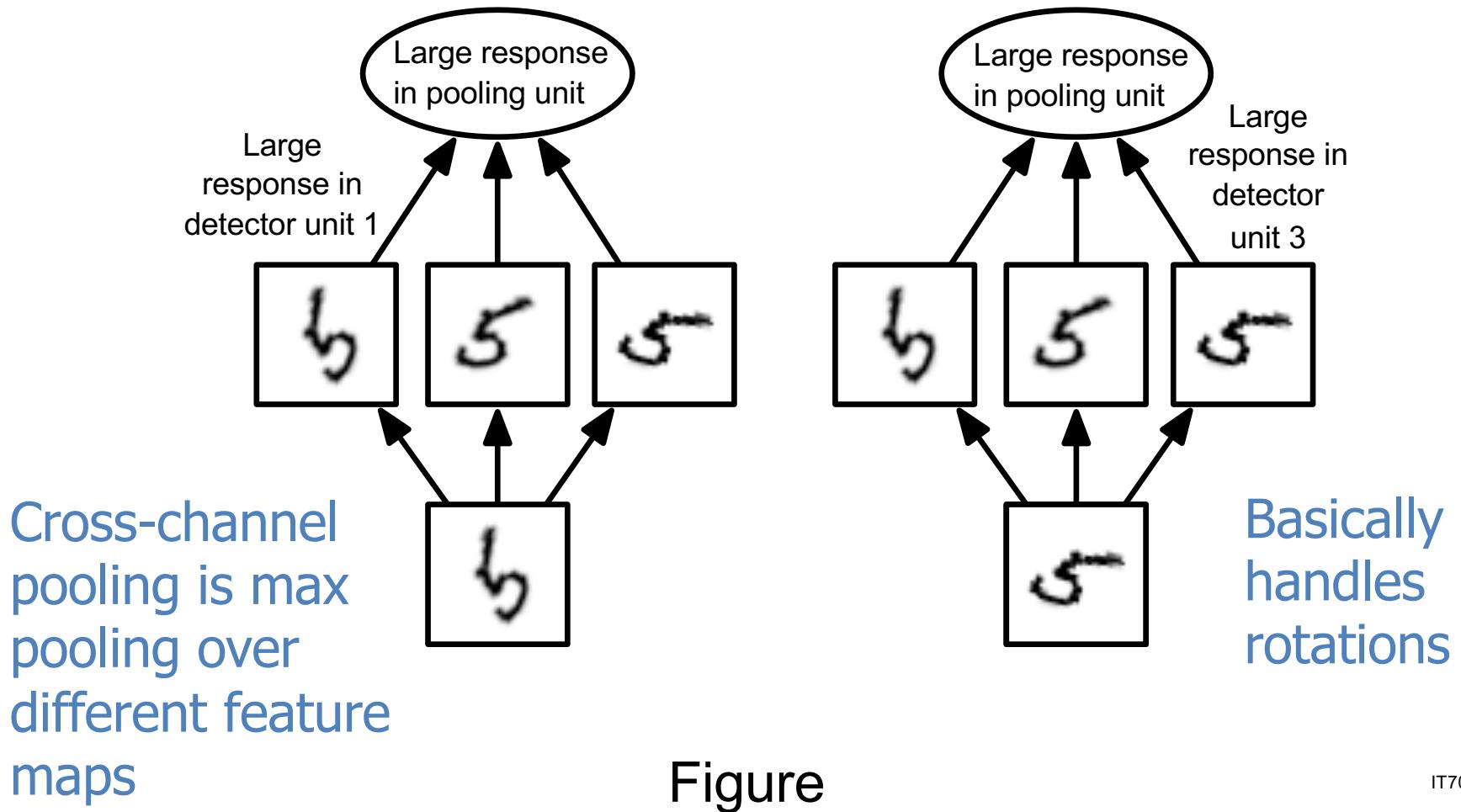


... Little change  
in pooling  
stage ...

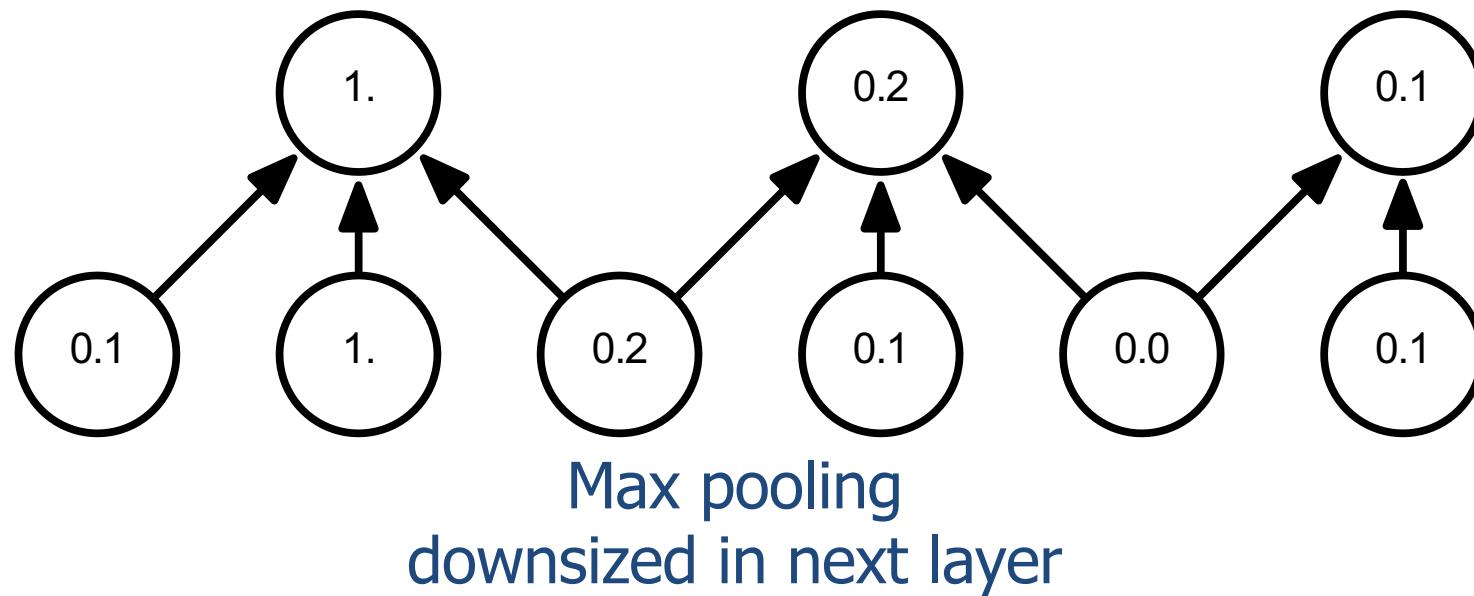
Every value in the bottom row has changed, but only half of the values in the top row have changed

Figure

# Cross-Channel Pooling and Invariance to Learned Transformations



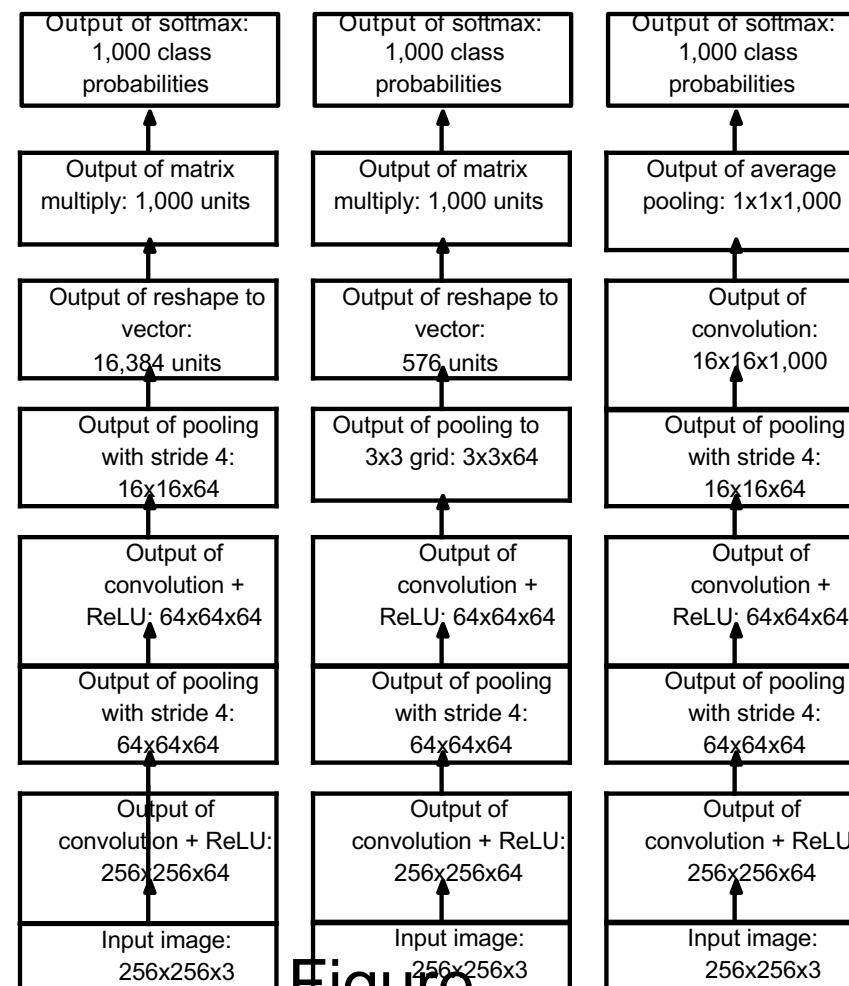
# Pooling with Downsampling



Pooling with downsampling. Here we use max pooling with a pool width of three and a stride between pools of two. This reduces the representation size by a factor of two

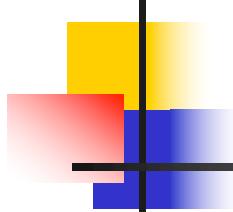
Figure

# Example Classification Architectures



Figure

Figure 9.11

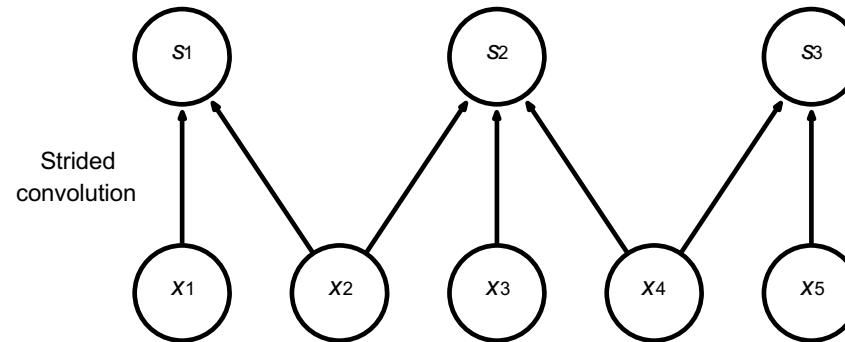


## 5. Variants of the Basic Convolution Function

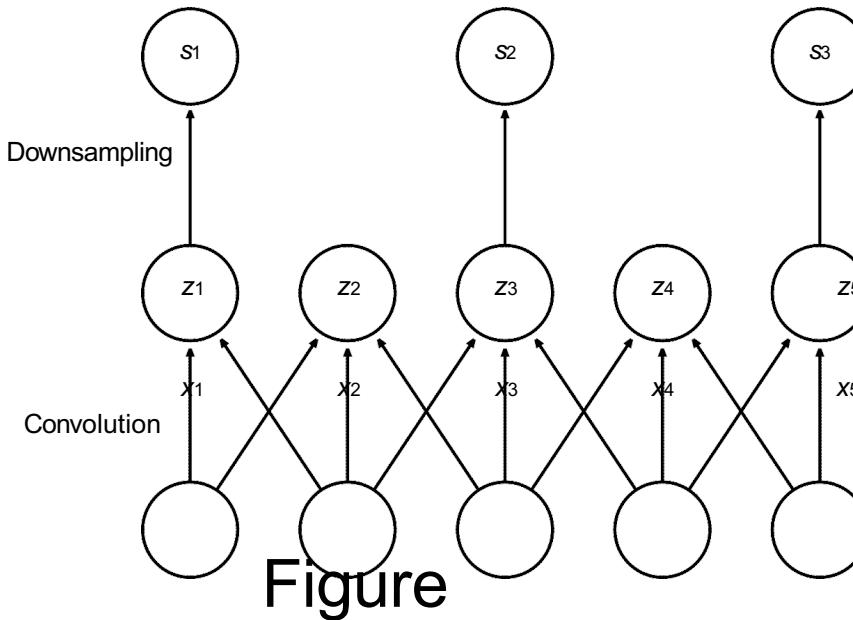
- **Stride** is the amount of downsampling
  - Can have separate strides in different directions
- **Zero padding** avoids layer-to-layer shrinking
- **Unshared convolution**
  - Like convolution but without sharing
- **Partial connectivity between channels**
- **Tiled convolution**
  - Cycle between shared parameter groups

# Convolution with Stride

Stride  
of two



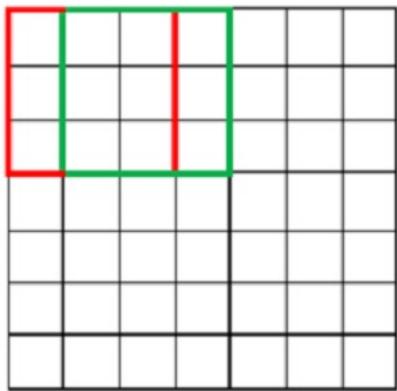
Equivalent to  
above but  
computationally  
wasteful



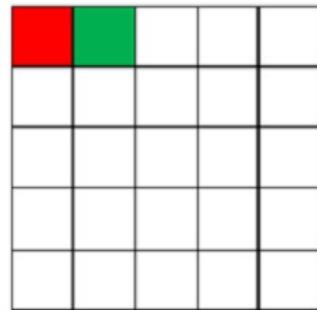
Figure

Figure  
9.12

7 x 7 Input Volume

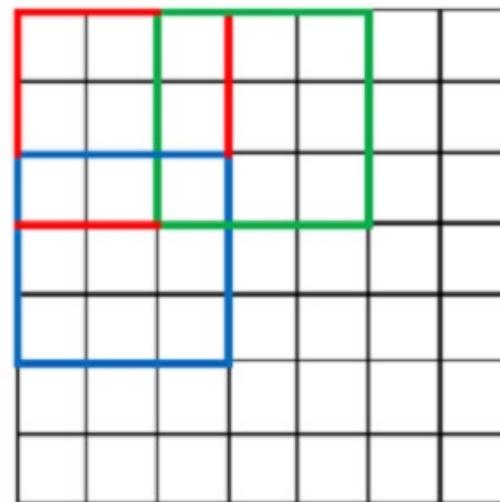


5 x 5 Output Volume

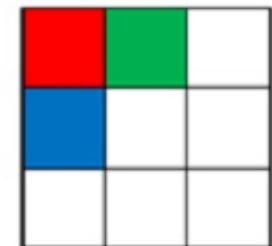


Stride with value 1 (default)

7 x 7 Input Volume



3 x 3 Output Volume



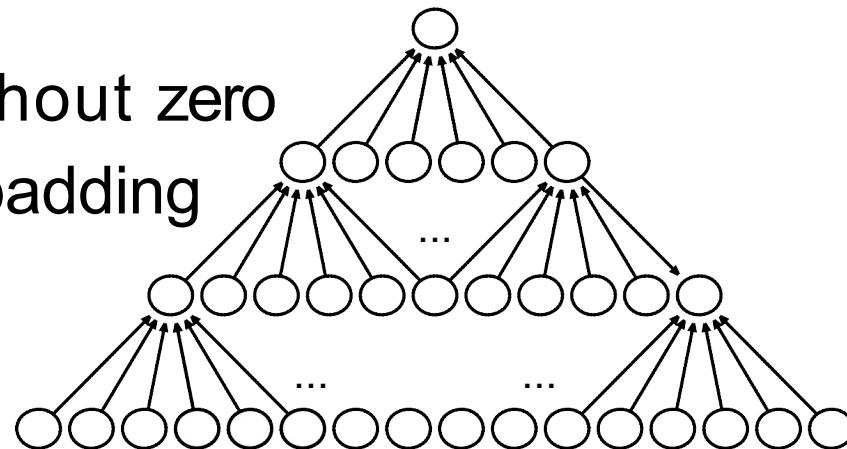
Figure

Stride with value 2

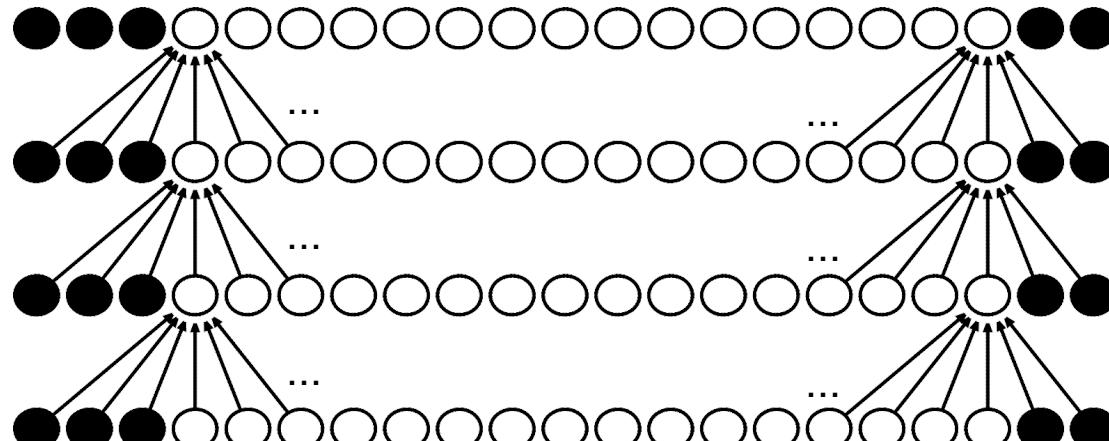
# Zero Padding Controls Size

Kernel  
width  
of six

Without zero  
padding



With zero  
padding  
Prevents  
shrinking

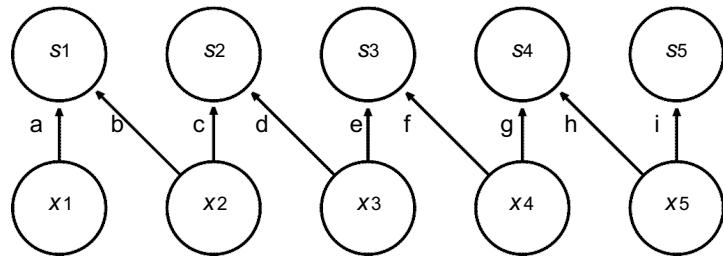


Figure

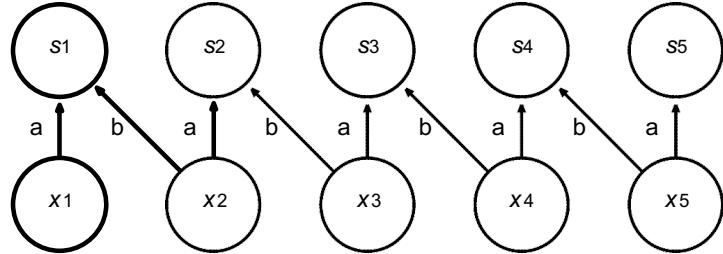
Figure 9.13

# Kinds of Connectivity

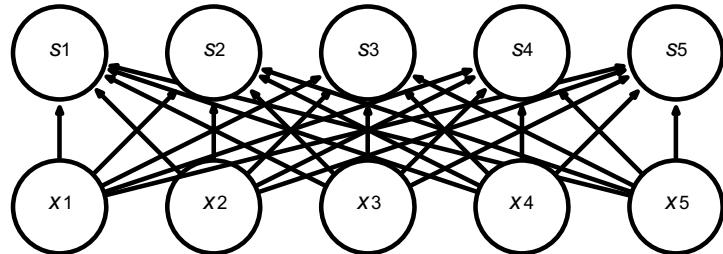
Unshared convolution



Local connection:  
like convolution,  
but no sharing



Convolution

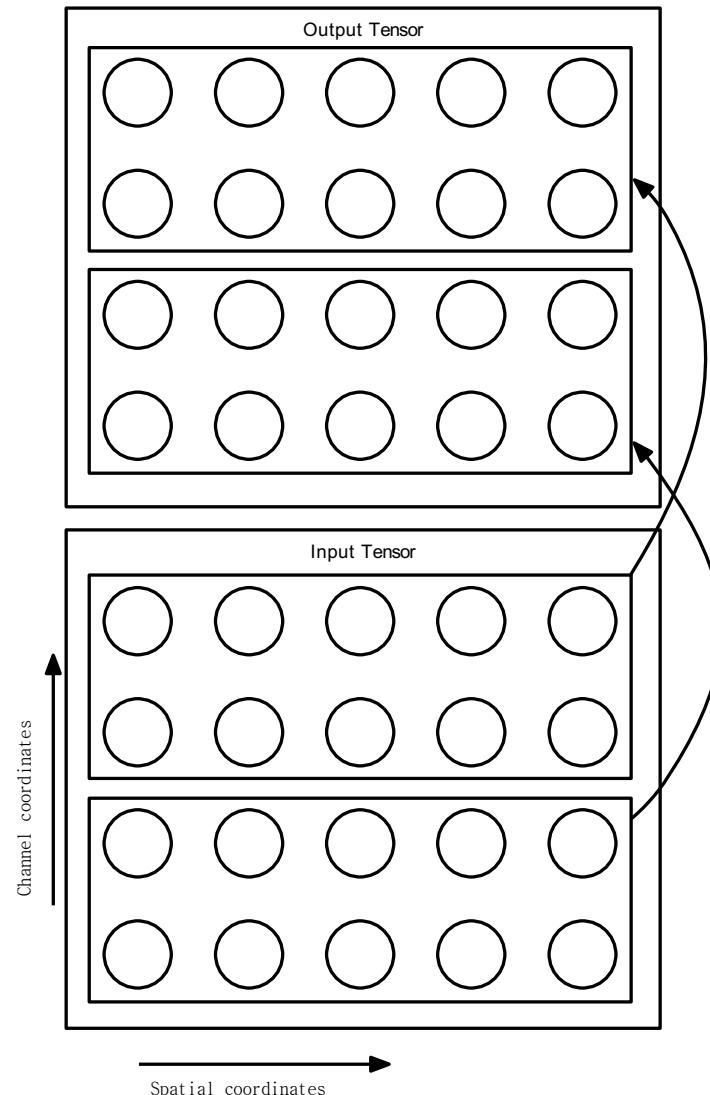


Fully connected

Figure 9.14

# Partial Connectivity Between Channels

Each output channel is a function of only a subset of the input channels

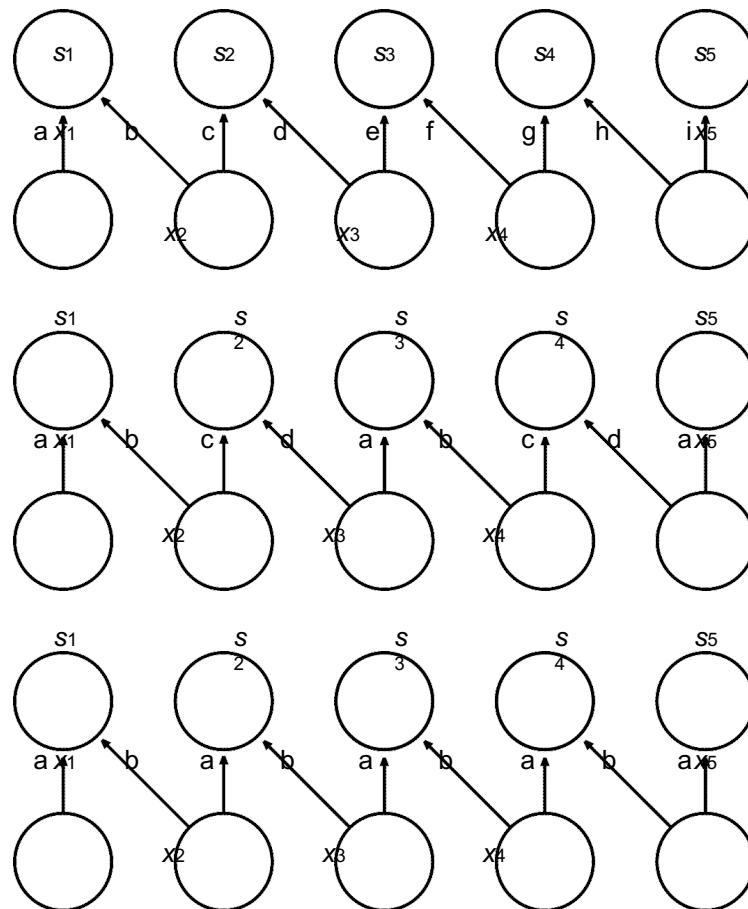


Figure

Figure 9.15

# Tiled convolution

Tiled convolution has a set of  $t$  different kernels. Here we illustrate the case of  $t = 2$ .

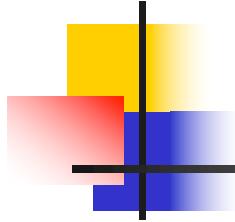


Local connection  
(no sharing)

**Tiled convolution**  
(cycle between shared  
parameter groups )

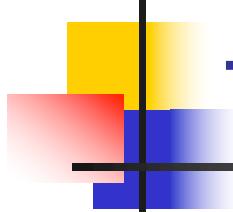
Standard convolution  
(one group shared  
everywhere)

Figure 9.16



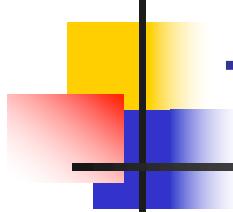
## 7. Data Types

- Single channel examples:
  - 1D audio waveform
  - 2D audio data after Fourier transform
    - Frequency versus time
- Multi-channel examples:
  - 2D color image data
    - Three channels: red pixels, green pixels, blue pixels
    - Each channel is 2D for the image



# 11. Convolutional Networks and the History of Deep Learning

- Convolutional networks have played an important role in the history of deep learning
  - Application of neuroscience to machine learning
  - First deep models to perform well
  - First important commercial applications
  - Used to win many contests
  - Some of first deep networks trained with back-prop
  - Performed well decades ago to pave the way toward acceptance of neural networks in general



# 11. Convolutional Networks and the History of Deep Learning

- Convolutional networks allow specialized neural networks for grid-structured topology
  - Most successful on 2D image topology
  - For 1D sequential data we use recurrent networks



# Sequence Modeling: Recurrent and Recursive Nets

Dr. Anand Kumar M

Dept. Of IT

National Institute of Technology Karnataka (NITK)

[M\\_anandkumar@nitk.edu.in](mailto:M_anandkumar@nitk.edu.in)

# Outline

- Motivation
- Unfolding Computational Graph
- RNN –variants
- BPTT
- Bi-RNN
- Encoder-Decoder
- DRN and Recursive NN
- LSTM

# Motivation

- Recurrent neural networks , or RNNs, are a family of neural networks for processing **sequential data**.
- Much as a *convolutional network* is a neural network that is specialized for processing a *grid of values X* such as an image.
- Recurrent neural network is a neural network that is specialized for processing a sequence of values  $x(1), \dots, x(\tau)$

# Motivation

- recurrent networks can also process sequences of **variable length**.
- **Parameter sharing** makes it possible to extend and apply the model to examples of different lengths and generalize across them.
- If we had separate parameters for each value of the time index, **we could not generalize to sequence lengths** not seen during training

# Motivation

- The **convolution** operation allows a network to share parameters across **time but is shallow**.
- The **output of convolution** is a sequence where *each member of the output is a function of a small number of neighboring members of the input.*
- IN RNN, *each member of the output is a function of the previous members of the output.*

# Unfolding Computational Graphs

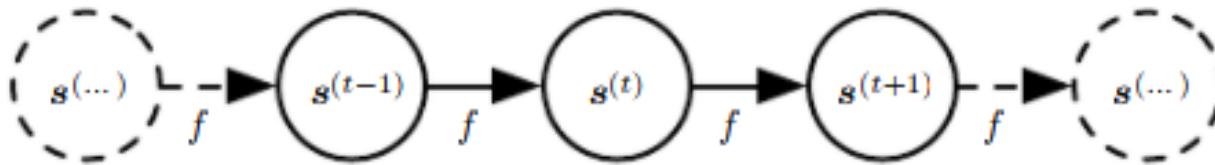


Figure 10.1: The classical dynamical system described by equation 10.1, illustrated as an unfolded computational graph. Each node represents the state at some time  $t$ , and the function  $f$  maps the state at  $t$  to the state at  $t + 1$ . The same parameters (the same value of  $\theta$  used to parametrize  $f$ ) are used for all time steps.

For example, consider the classical form of a dynamical system:

$$s^{(t)} = f(s^{(t-1)}; \theta), \quad (10.1)$$

where  $s^{(t)}$  is called the state of the system.

Equation 10.1 is recurrent because the definition of  $s$  at time  $t$  refers back to the same definition at time  $t - 1$ .

For a finite number of time steps  $\tau$ , the graph can be unfolded by applying the definition  $\tau - 1$  times. For example, if we unfold equation 10.1 for  $\tau = 3$  time steps, we obtain

$$s^{(3)} = f(s^{(2)}; \theta) \quad (10.2)$$

$$= f(f(s^{(1)}; \theta); \theta). \quad (10.3)$$

# Unfolding Computational Graphs

- As another example, let us consider a dynamical system driven by an external signal  $X_t$ .

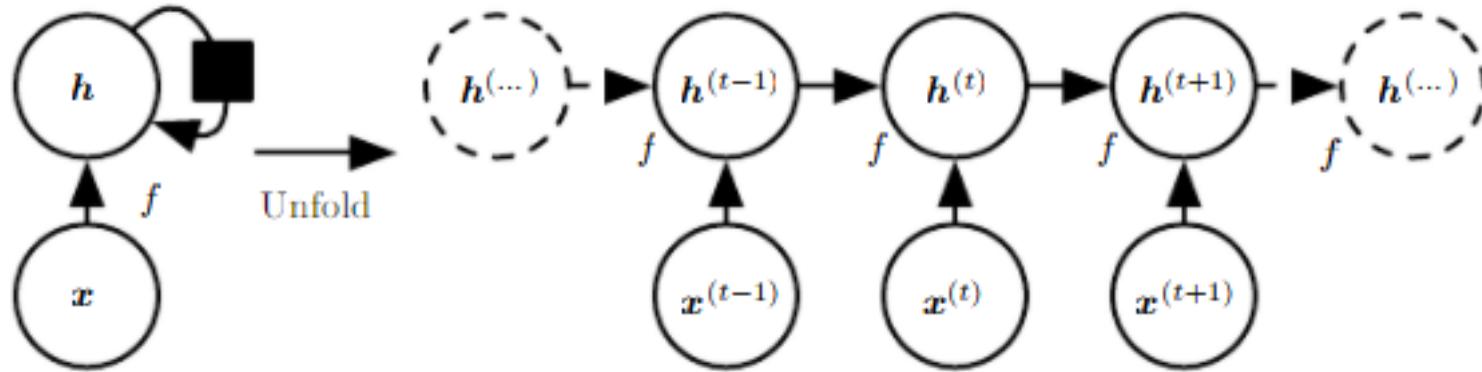


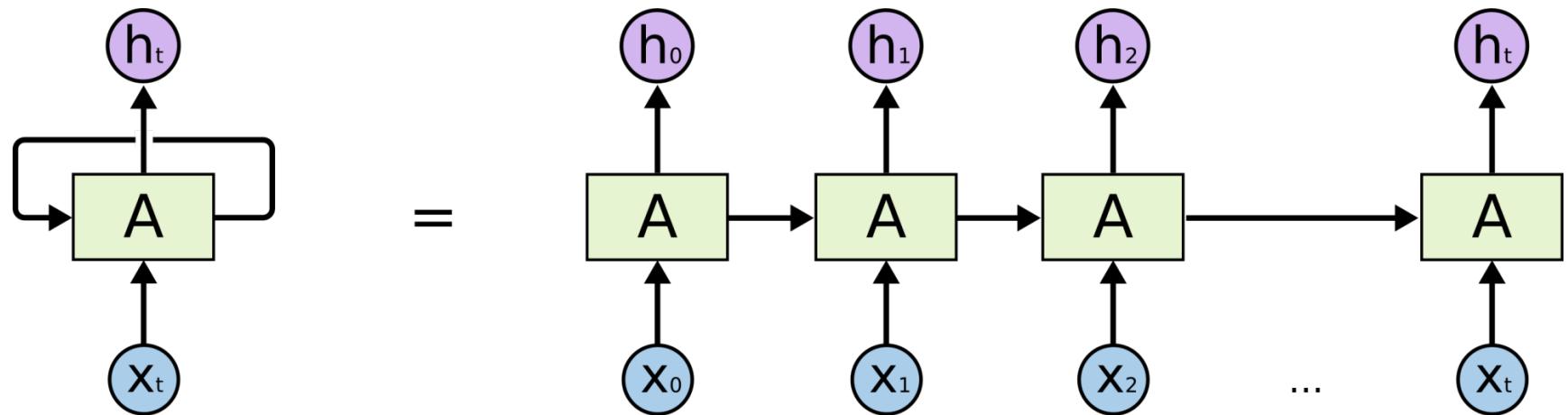
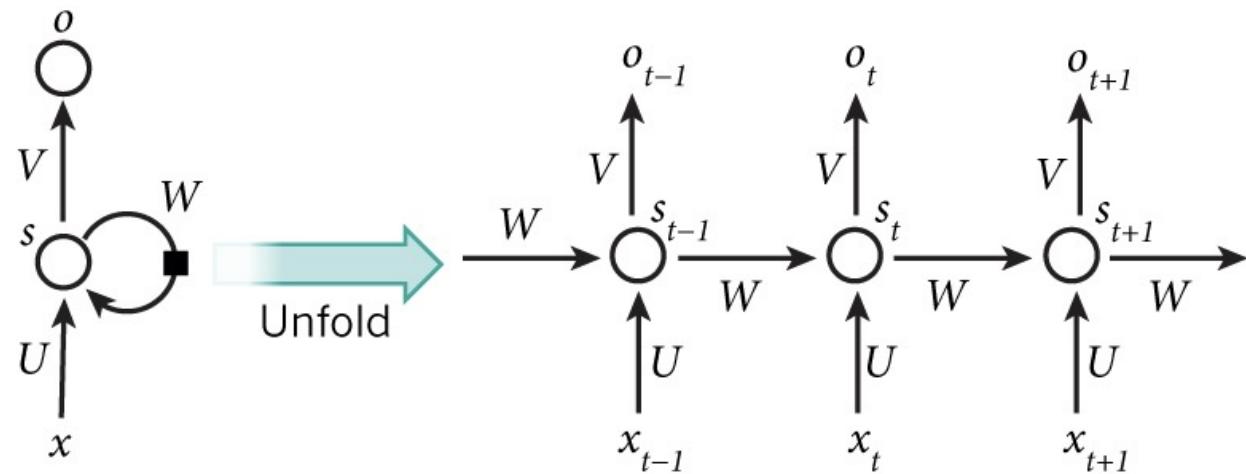
Figure 10.2: A recurrent network with no outputs. This recurrent network just processes information from the input  $x$  by incorporating it into the state  $h$  that is passed forward through time. (*Left*) Circuit diagram. The black square indicates a delay of a single time step. (*Right*) The same network seen as an unfolded computational graph, where each node is now associated with one particular time instance.

$$\textcolor{brown}{h}^{(t)} = f(\textcolor{brown}{h}^{(t-1)}, \textcolor{violet}{x}^{(t)}; \theta),$$

# Unfolding Computational Graphs

- The unfolding process thus introduces **two** major advantages:
  - Regardless of the sequence length, the *learned model always has the same input size*, because it is specified in terms of transition from one state to another state, rather than specified in terms of a variable-length history of states.
  - It is possible to use the *same transition function f with the same parameters* at every time step

# Recurrent Neural Network



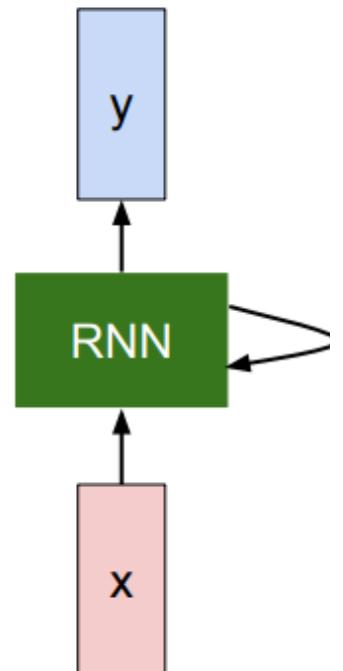
# RNN

## Recurrent Neural Network

We can process a sequence of vectors  $\mathbf{x}$  by applying a **recurrence formula** at every time step:

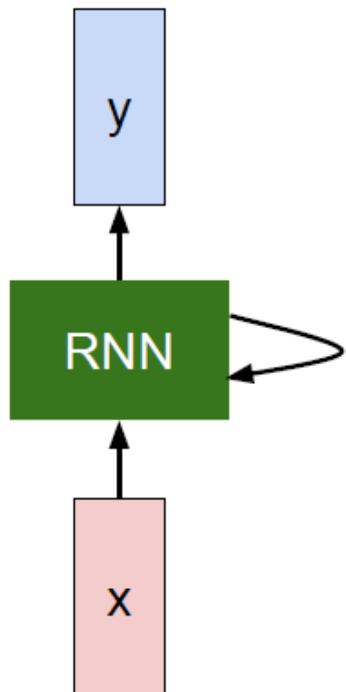
$$h_t = f_W(h_{t-1}, x_t)$$

new state      \old state      input vector at  
some function      some time step  
with parameters W



# (Vanilla) Recurrent Neural Network

The state consists of a single “*hidden*” vector  $\mathbf{h}$ :



$$h_t = f_W(h_{t-1}, x_t)$$

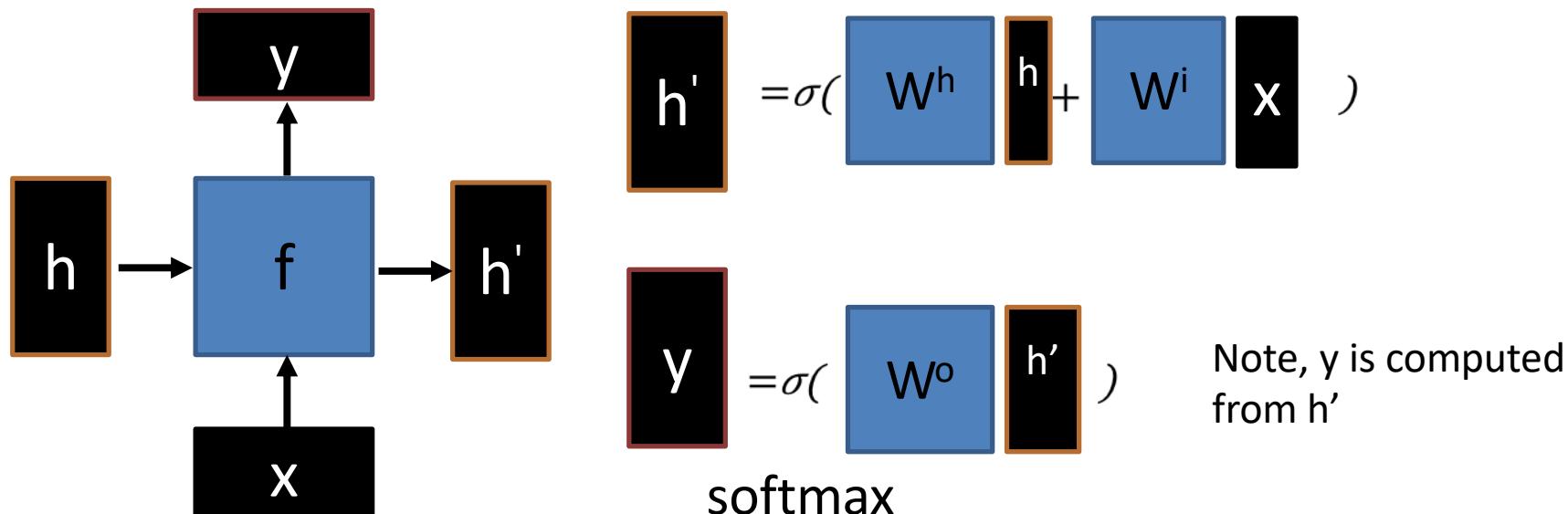


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

# Naïve RNN

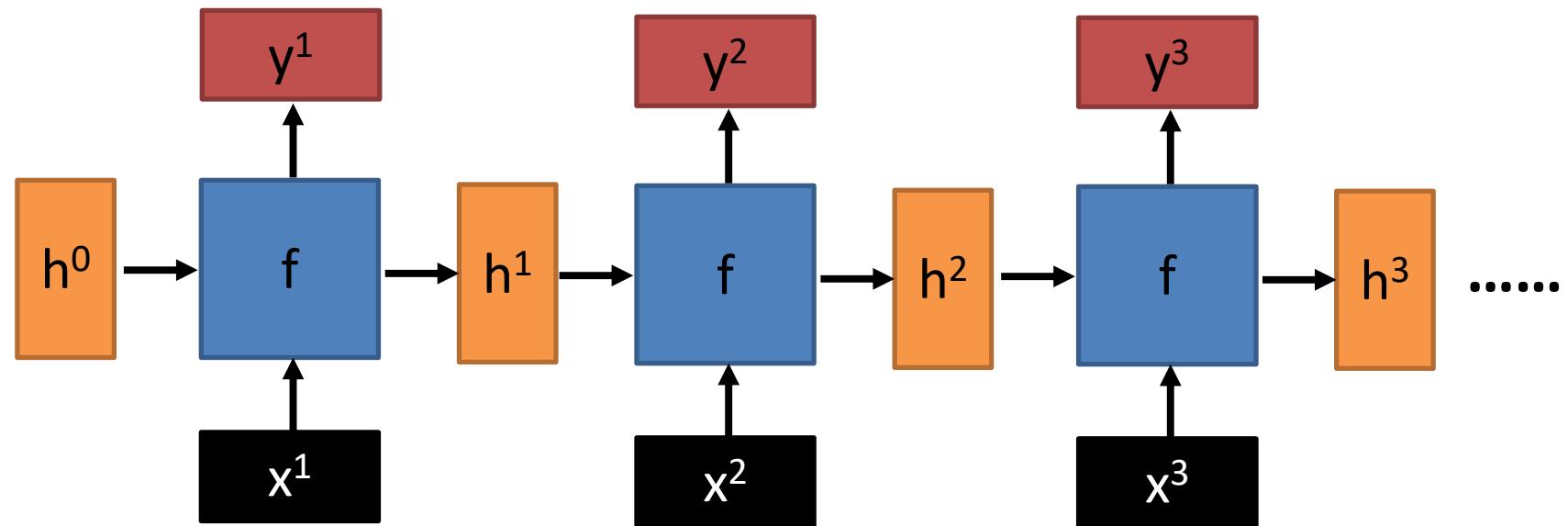
- Given function  $f: h', y = f(h, x)$



# How does RNN reduce complexity?

- Given function  $f: h', y = f(h, x)$

$h$  and  $h'$  are vectors with the same dimension



No matter how long the input/output sequence is, we only need one function  $f$ . If  $f$ 's are different, then it becomes a feedforward NN. This may be treated as another compression from fully connected network.

# Recurrent Neural Networks

- Some examples of important design patterns for recurrent neural networks include the following.
- Recurrent networks that produce an output at each time step and have recurrent connections between hidden units, illustrated in figure 10.3
- Recurrent networks that produce an output at each time step and have recurrent connections only from the output at one time step to the hidden units at the next time step, illustrated in figure 10.4
- Recurrent networks with recurrent connections between hidden units, that read an entire sequence and then produce a single output, illustrated in figure 10.5

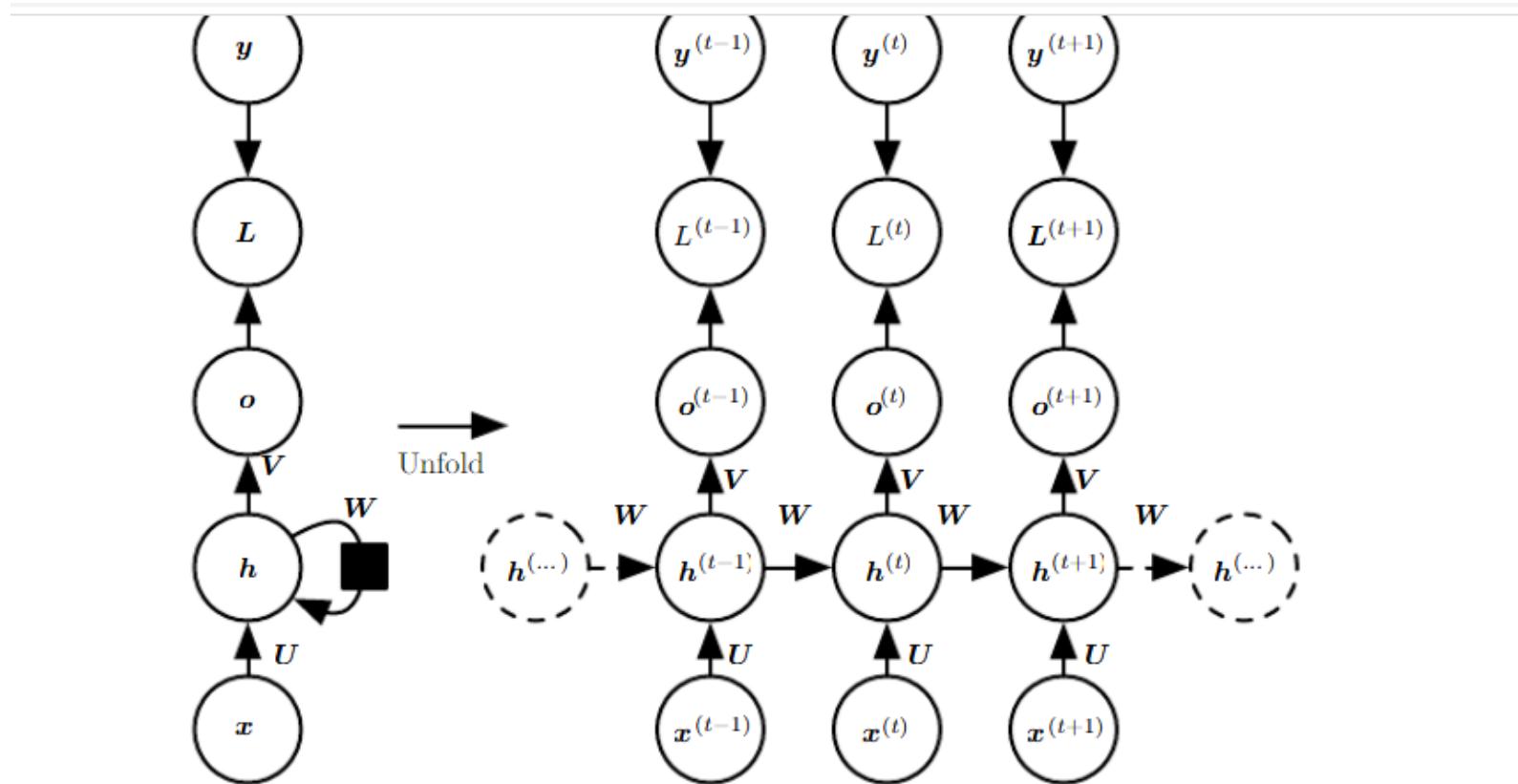
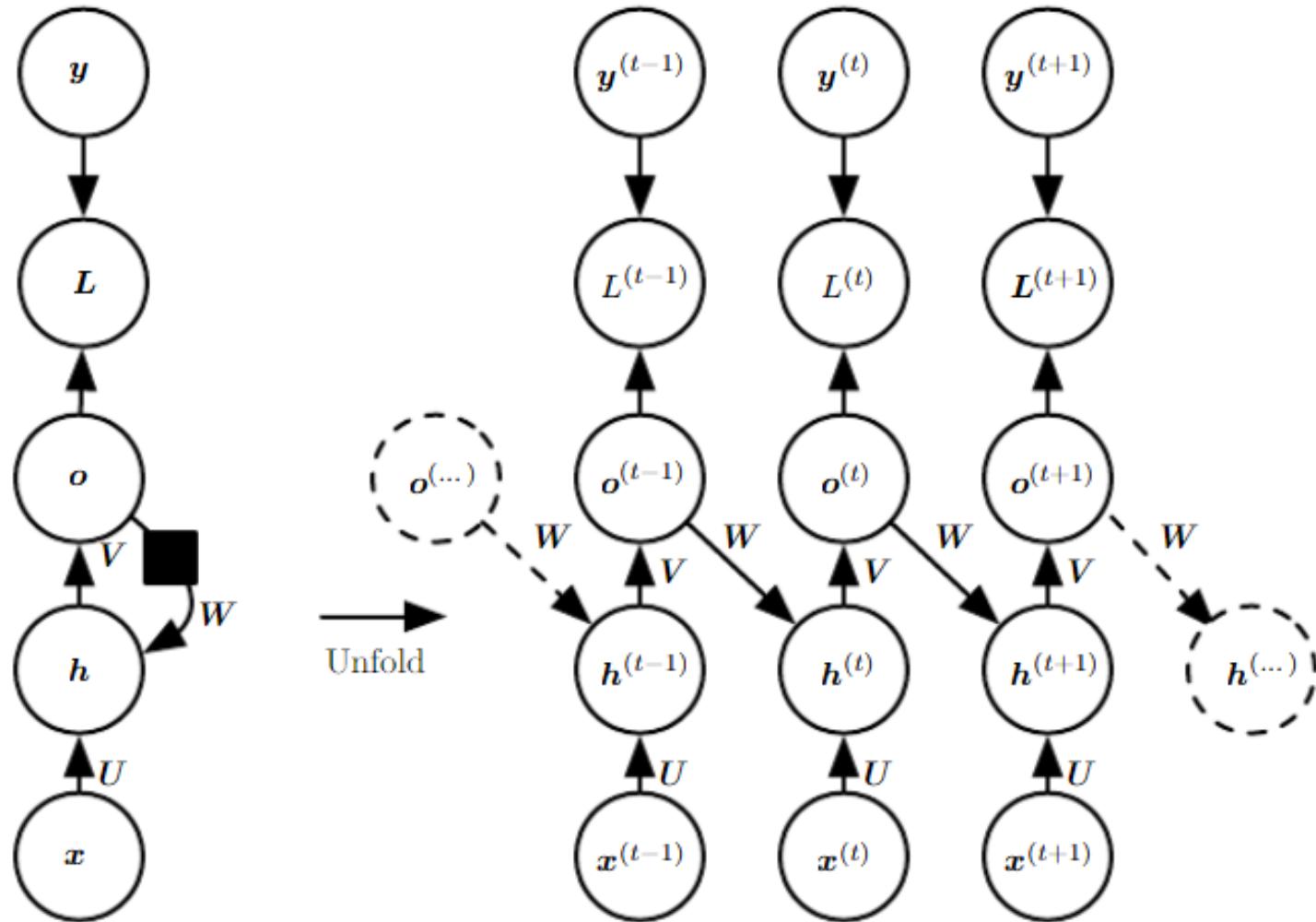
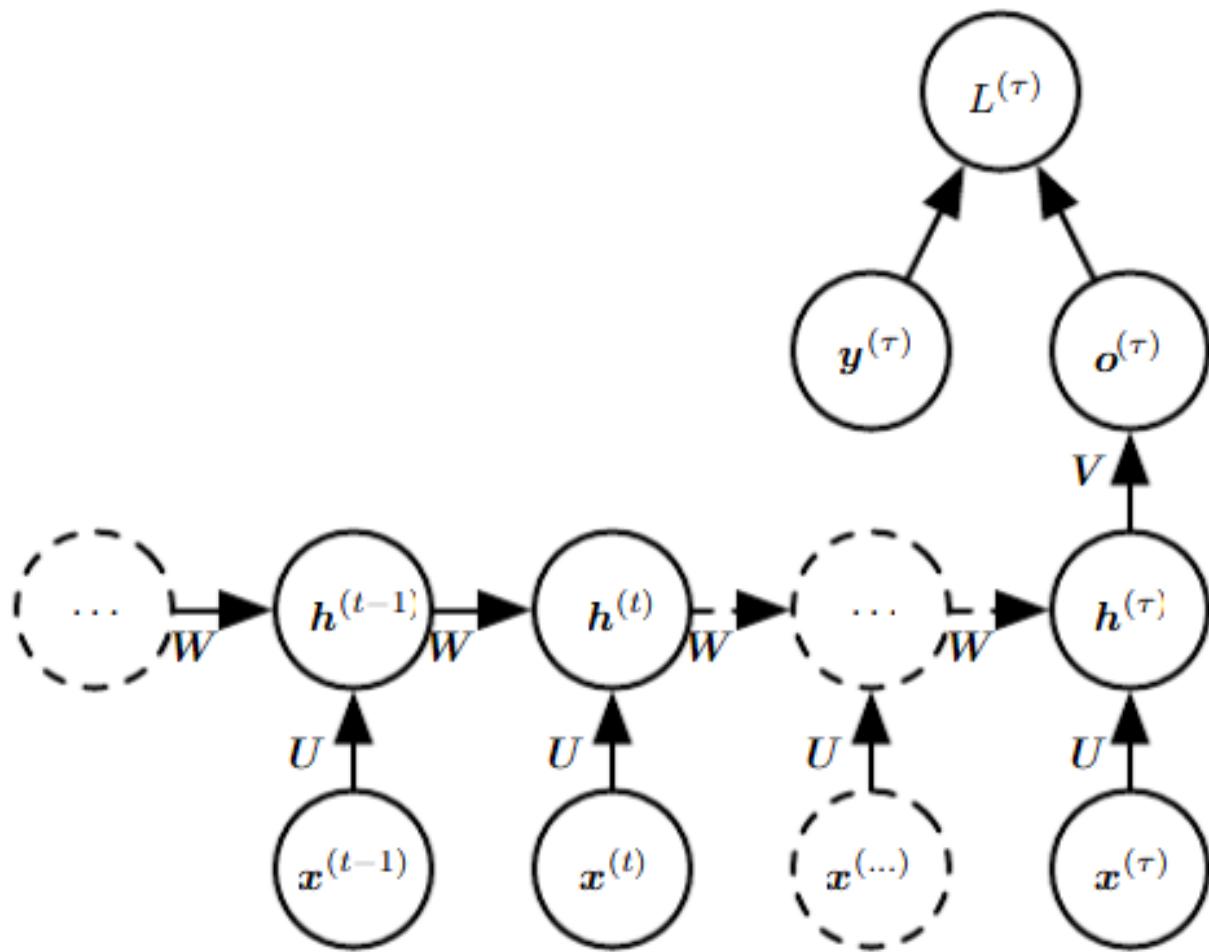


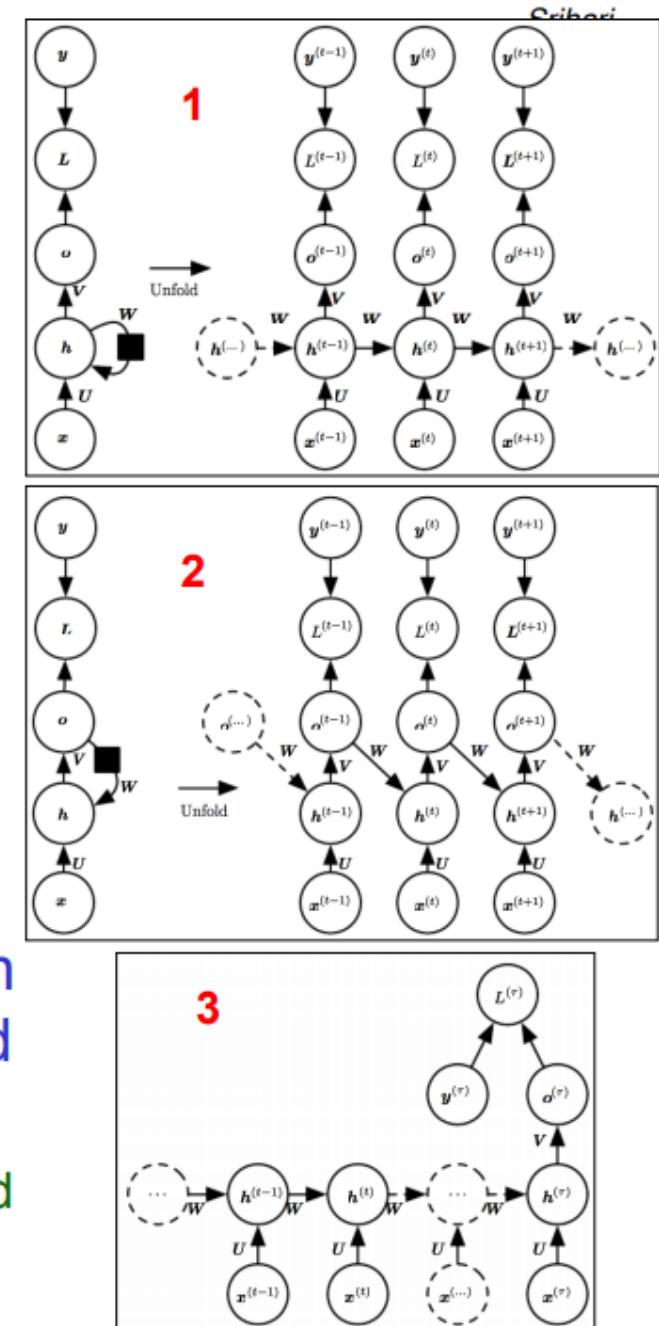
Figure 10.3: The computational graph to compute the training loss of a recurrent network that maps an input sequence of  $\mathbf{x}$  values to a corresponding sequence of output  $\mathbf{o}$  values.



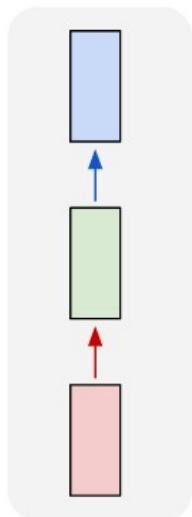


## Three design patterns of RNNs

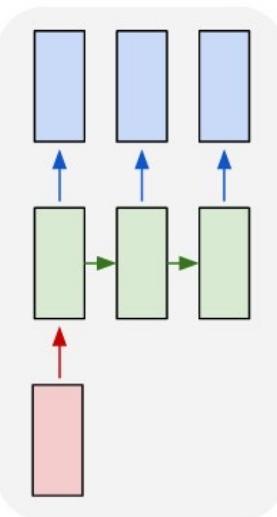
1. Produce output at each time step and have recurrent connections between hidden units
2. Produce output at each time step and have recurrent connections only from output at one time step to hidden units at next time step
3. Recurrent connections between hidden units to read entire input sequence and produce a single output
  - Can summarize sequence to produce a fixed size representation for further processing



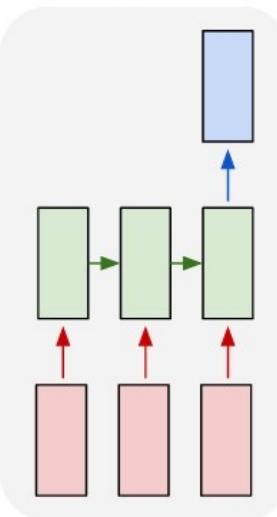
one to one



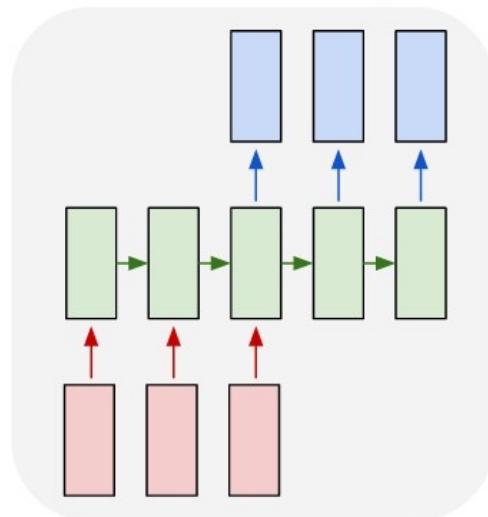
one to many



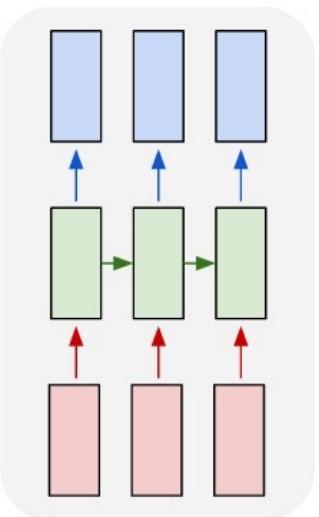
many to one



many to many



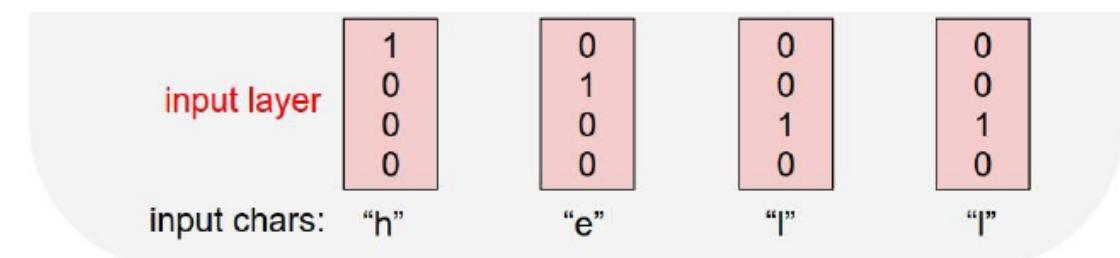
many to many



# Example: Character-level Language Model

Vocabulary:  
[h,e,l,o]

Example training  
sequence:  
**“hello”**

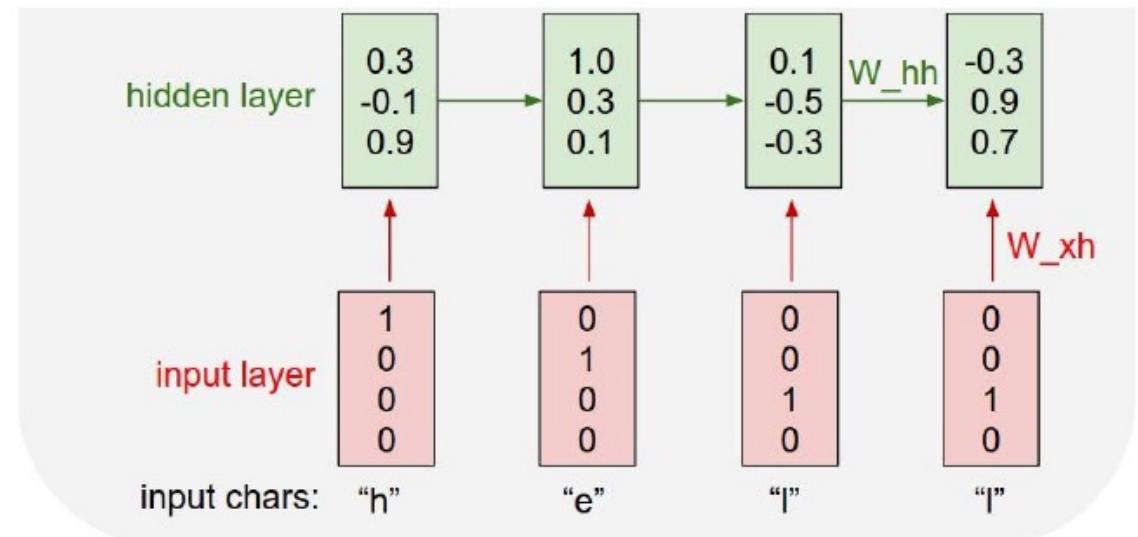


## Example: Character-level Language Model

Vocabulary:  
[h,e,l,o]

Example training  
sequence:  
**“hello”**

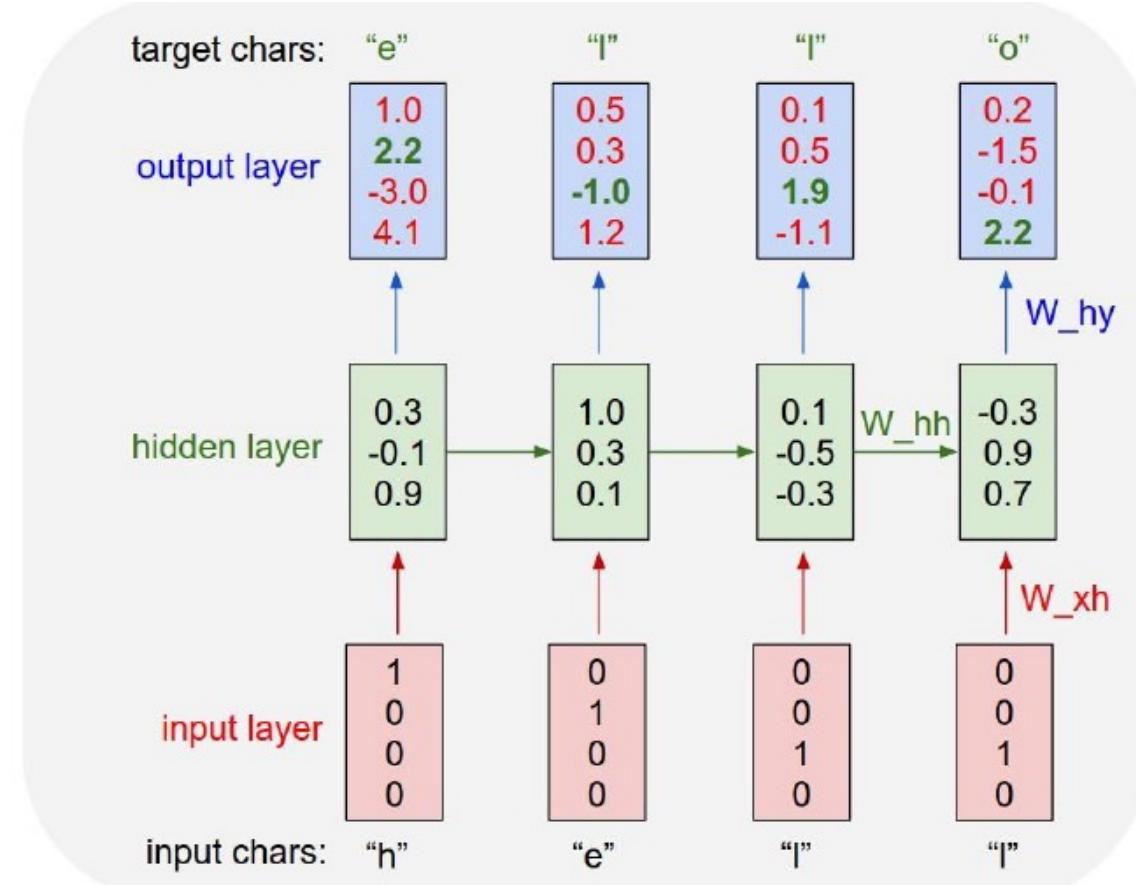
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



## Example: Character-level Language Model

Vocabulary:  
[h,e,l,o]

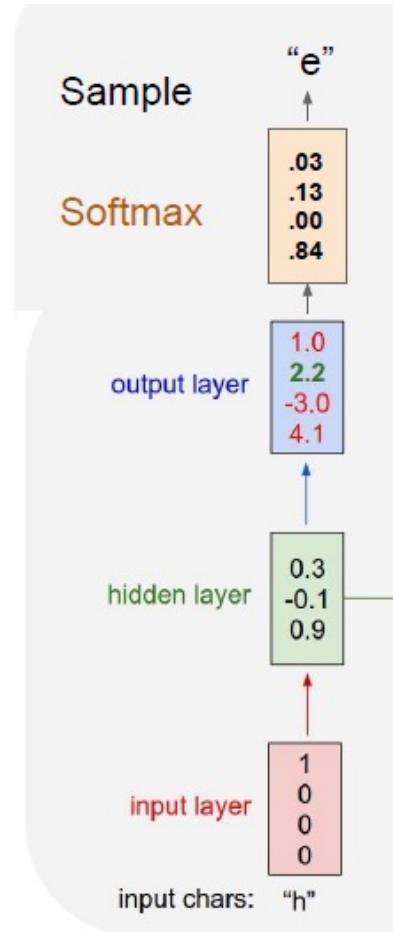
Example training  
sequence:  
**“hello”**



# Example: Character-level Language Model Sampling

Vocabulary:  
[h,e,l,o]

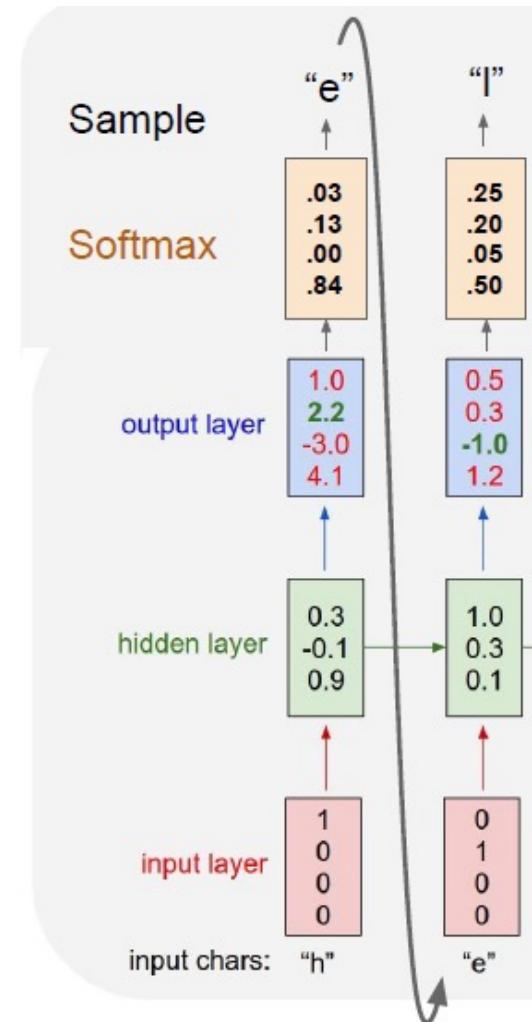
At test-time sample  
characters one at a time,  
feed back to model



# Example: Character-level Language Model Sampling

Vocabulary:  
[h,e,l,o]

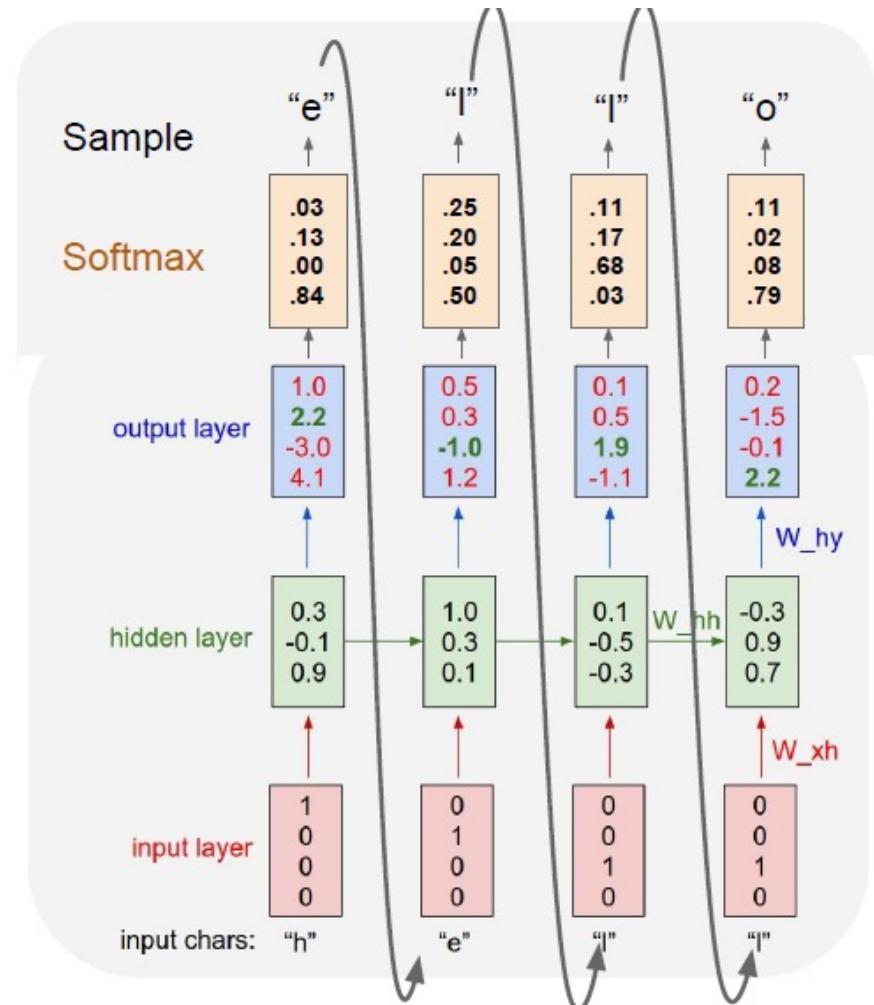
At test-time sample  
characters one at a time,  
feed back to model



# Example: Character-level Language Model Sampling

Vocabulary:  
[h,e,l,o]

At test-time sample  
characters one at a time,  
feed back to model



# Bi-RNN

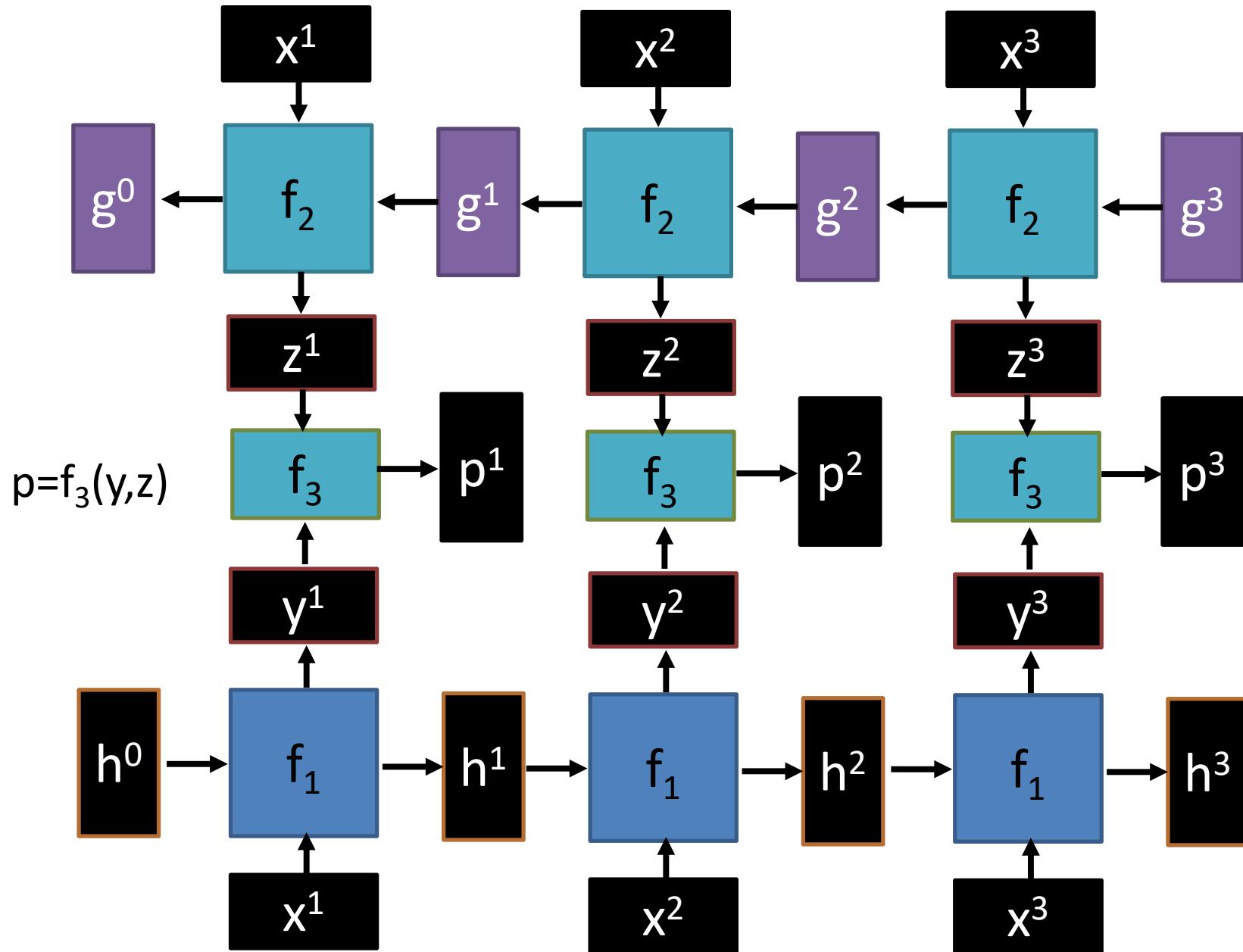
- In many applications, however, we want to output a prediction of  $y(t)$  that may depend on the whole input sequence.
- They have been extremely successful in applications where that need arises, such as handwriting recognition, Speech recognition and Bioinformatics.

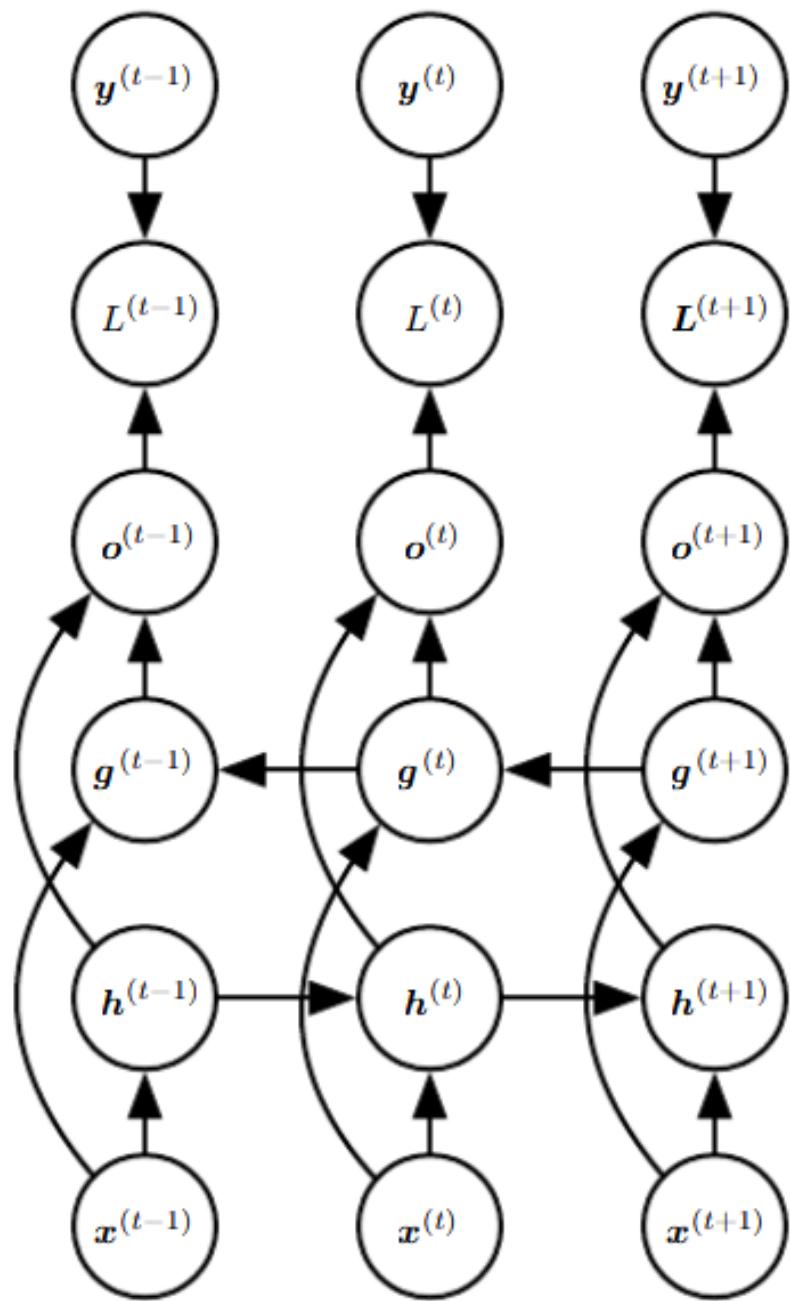
# Bi-RNN

- As the name suggests, bidirectional RNNs combine an RNN that moves forward through time, **beginning from the start of the sequence**, with another RNN that moves backward through time, **beginning from the end of the sequence**.

# Bidirectional RNN

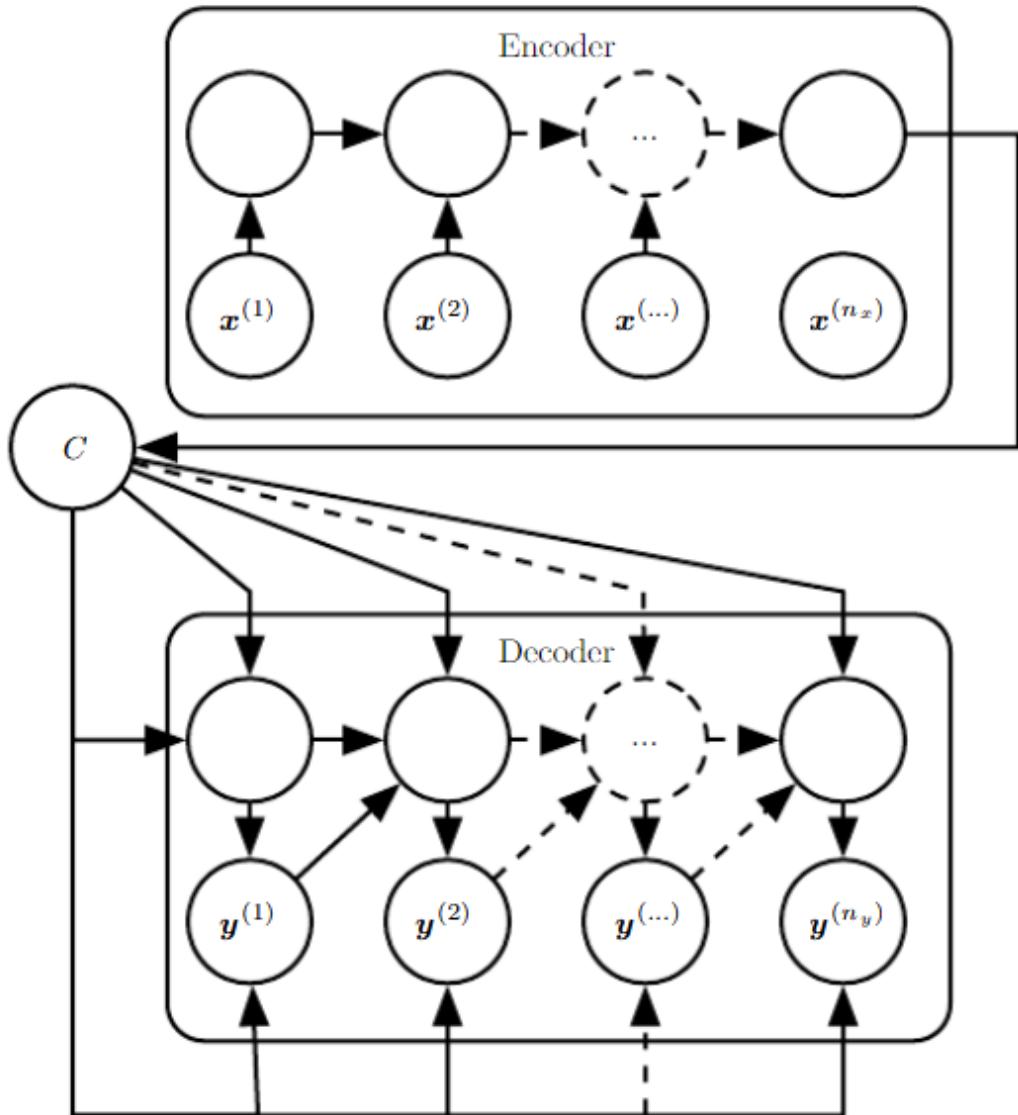
$$y, h = f_1(x, h) \quad z, g = f_2(g, x)$$





# Encoder-Decoder Sequence-to-Sequence Architectures

- RNN can map an input sequence to a fixed-size vector.
- An RNN can map a fixed-size vector to a sequence.
- an RNN can map an input sequence to an output sequence of the same length.
- *RNN can be trained to map an input sequence to an output sequence which is not necessarily of the same length.*



Example of an encoder-decoder or sequence-to- sequence RNN architecture, for learning to generate an output sequence  $y$  given an input sequence  $x$ .

It is composed of an encoder RNN that reads the input sequence as well as a decoder RNN that generates the output sequence (or computes the probability of a given output sequence).

The final hidden state of the encoder RNN is used to compute a generally fixed-size context variable  $C$ , which represents a semantic summary of the input sequence and is given as input to the decoder RNN

# Deep Recurrent Net

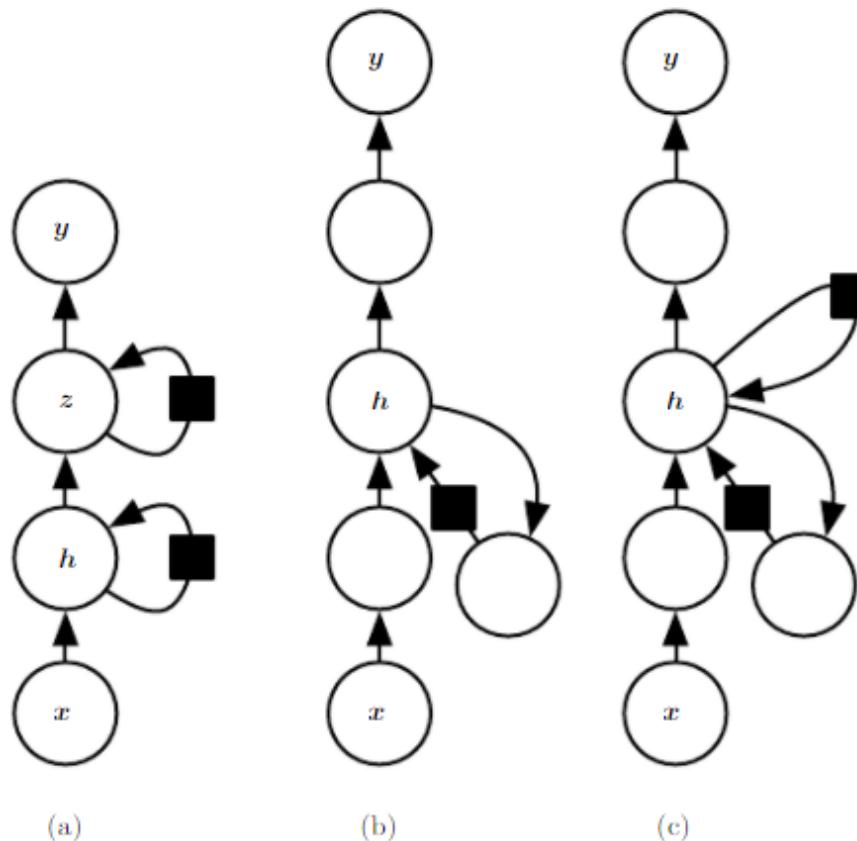
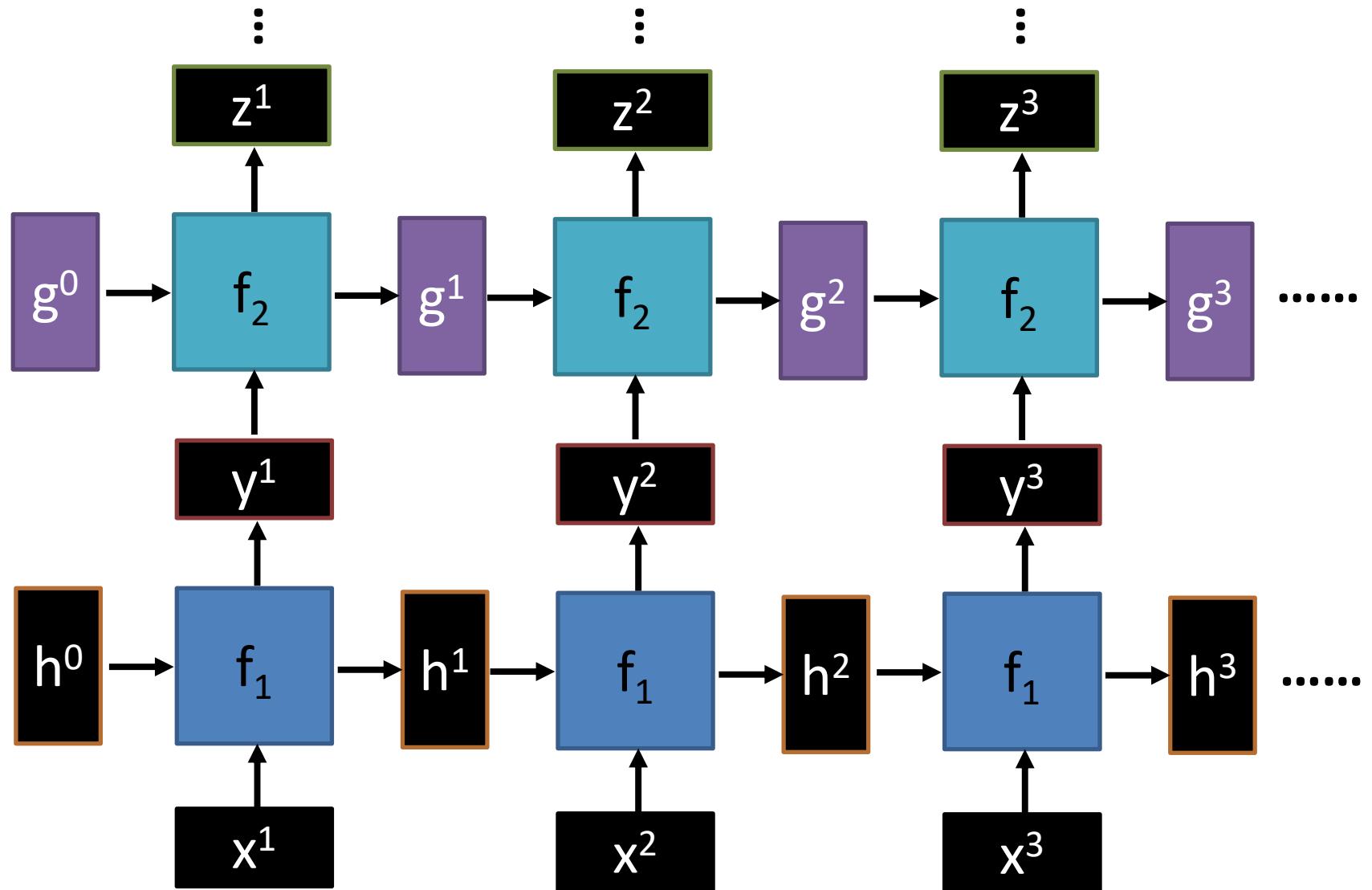


Figure 10.13: A recurrent neural network can be made deep in many ways (Pascanu *et al.*, 2014a). (a) The hidden recurrent state can be broken down into groups organized hierarchically. (b) Deeper computation (e.g., an MLP) can be introduced in the input-to-hidden, hidden-to-hidden, and hidden-to-output parts. This may lengthen the shortest

# Deep RNN

$$h', y = f_1(h, x), g', z = f_2(g, y)$$

...



# Recursive NN

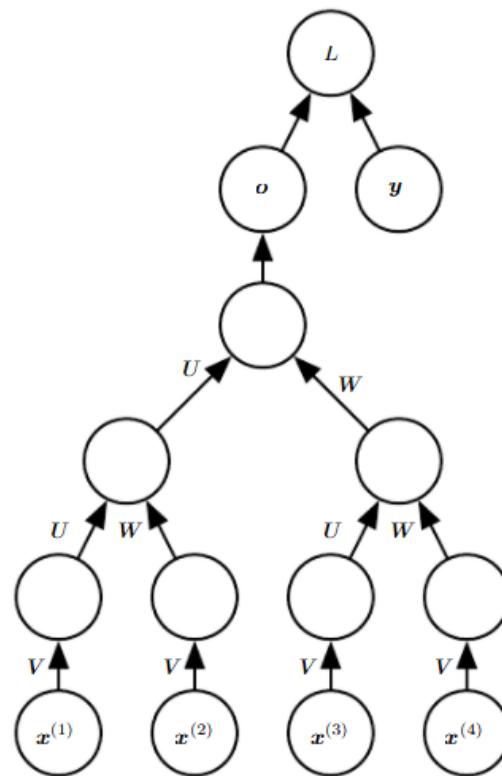
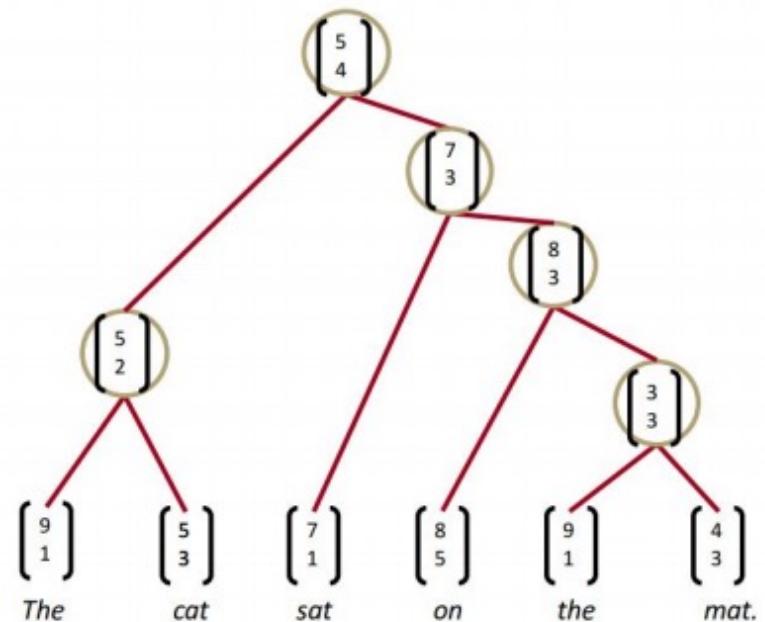


Figure 10.14: A recursive network has a computational graph that generalizes that of the recurrent network from a chain to a tree. A variable-size sequence  $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(t)}$  can be mapped to a fixed-size representation (the output  $\mathbf{o}$ ), with a fixed set of parameters (the weight matrices  $\mathbf{U}, \mathbf{V}, \mathbf{W}$ ). The figure illustrates a supervised learning case in which some target  $\mathbf{y}$  is provided that is associated with the whole sequence.

# Recursive Neural Network

- Apply when there's tree structure in data
  - For natural language use The Stanford Parser to build the syntax tree given a sentence

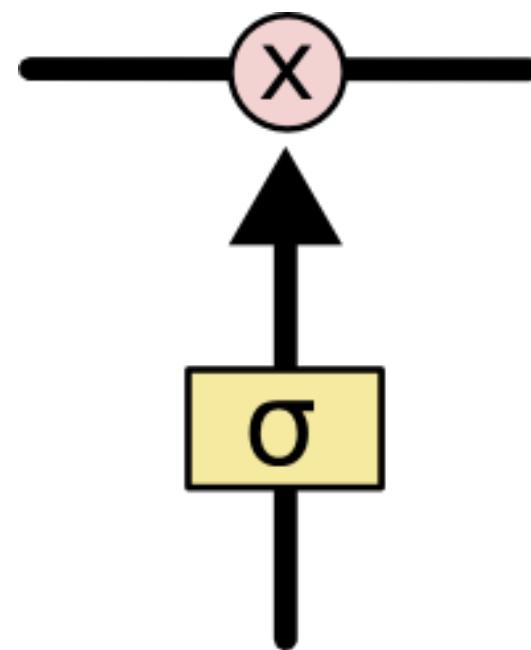
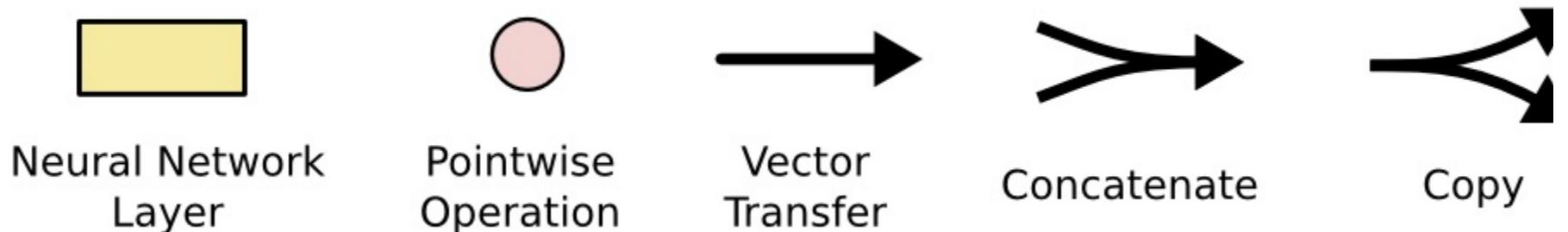
<http://cs224d.stanford.edu/lectures/CS224d-Lecture10.pdf>  
<https://nlp.stanford.edu/software/lex-parser.shtml>



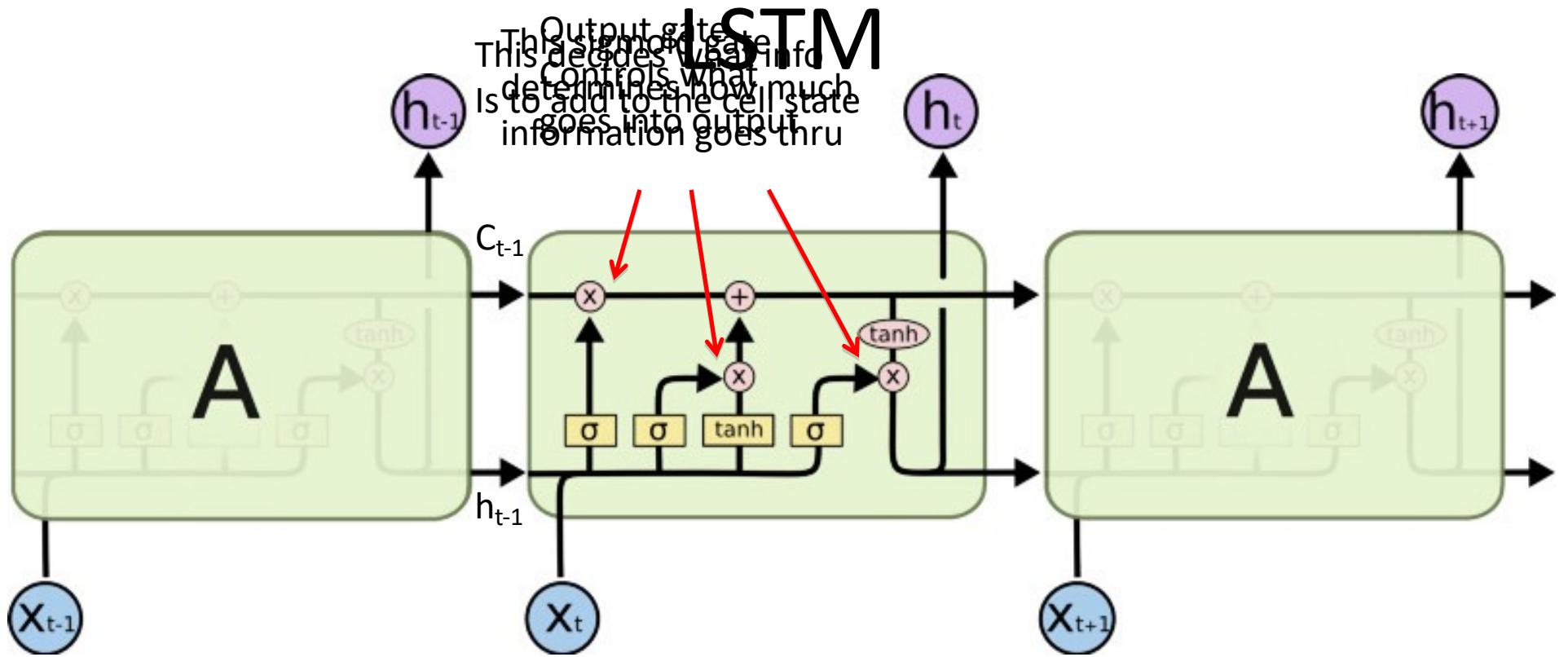
# Problems with naive RNN

- When dealing with a time series, it tends to forget old information. When there is a distant relationship of unknown length, we wish to have a “memory” to it.
- Vanishing gradient problem.

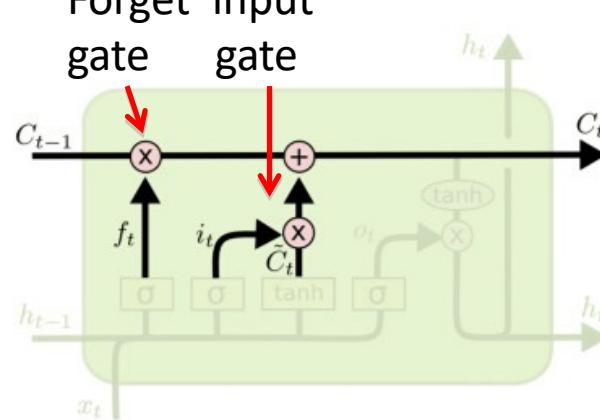
# LSTM



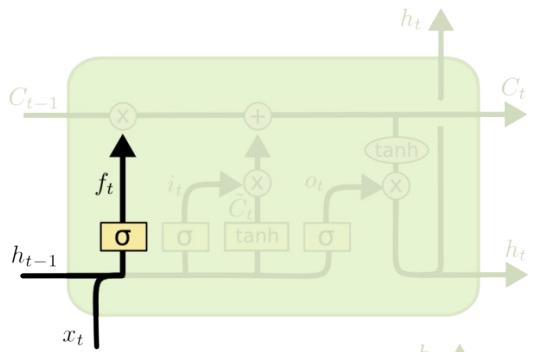
The sigmoid layer outputs numbers between 0-1 determine how much each component should be let through. Pink X gate is point-wise multiplication.



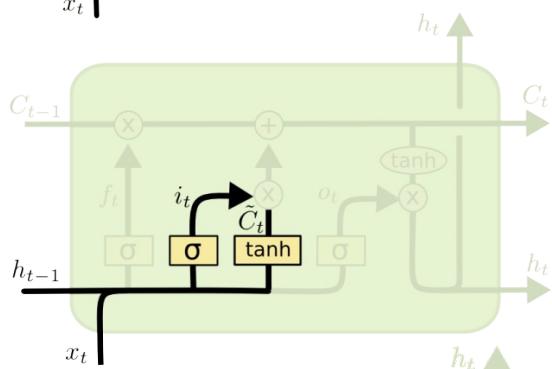
The core idea is this cell state  
Why sigmoid or tanh:  
 $C_t$ , it is changed slowly, with  
 Sigmoid: 0,1 gating as switch.  
 only minor linear interactions.  
 Vanishing gradient problem in  
 It is very easy for information  
 LSTM is handled already.  
 to flow along it unchanged.  
 ReLU replaces tanh ok?



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

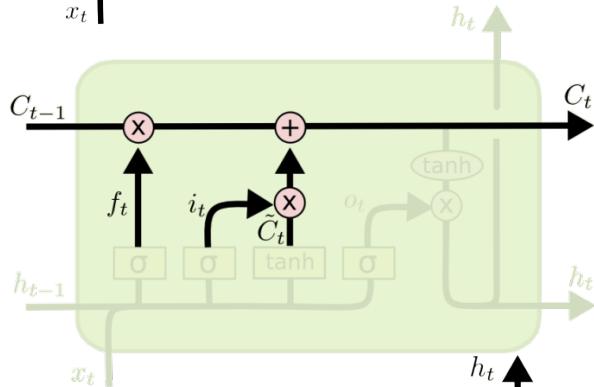


$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

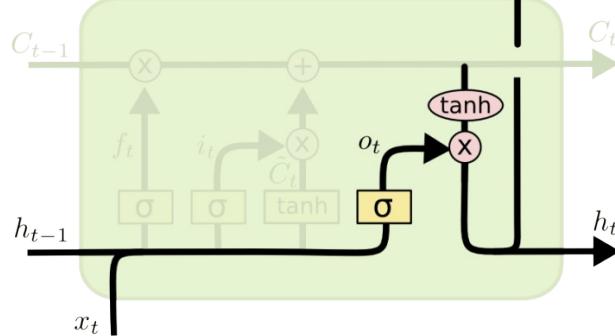


$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

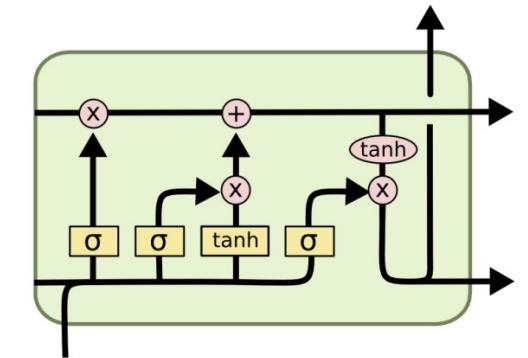


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$o_t = \sigma (W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

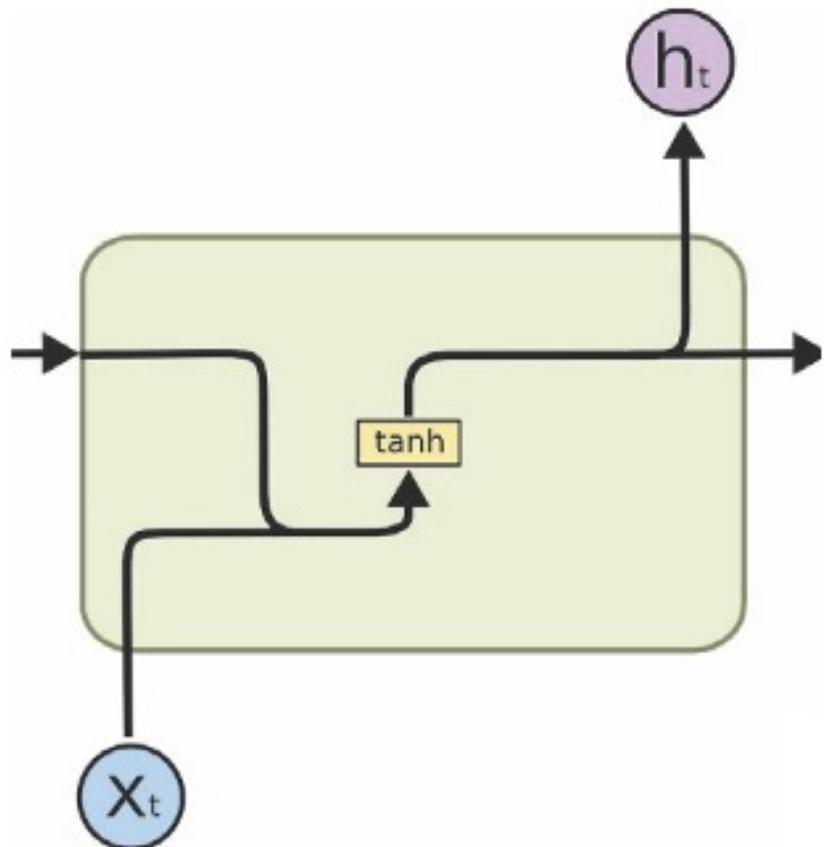


$i_t$  decides what component is to be updated.  
 $C'_t$  provides change contents

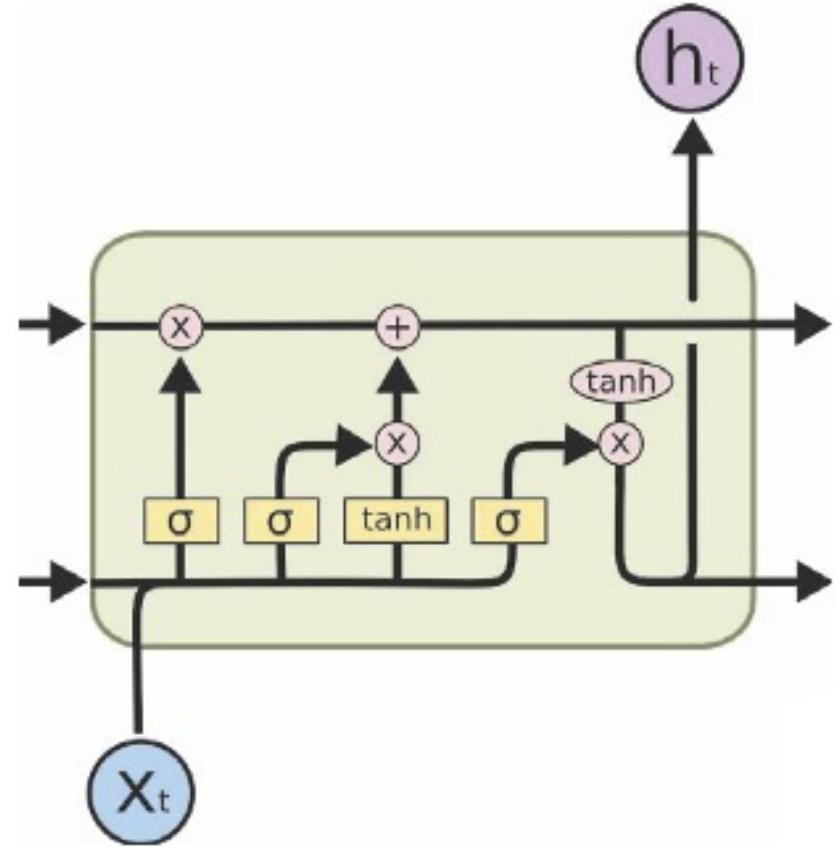
Updating the cell state

Decide what part of the cell state to output

# RNN vs LSTM

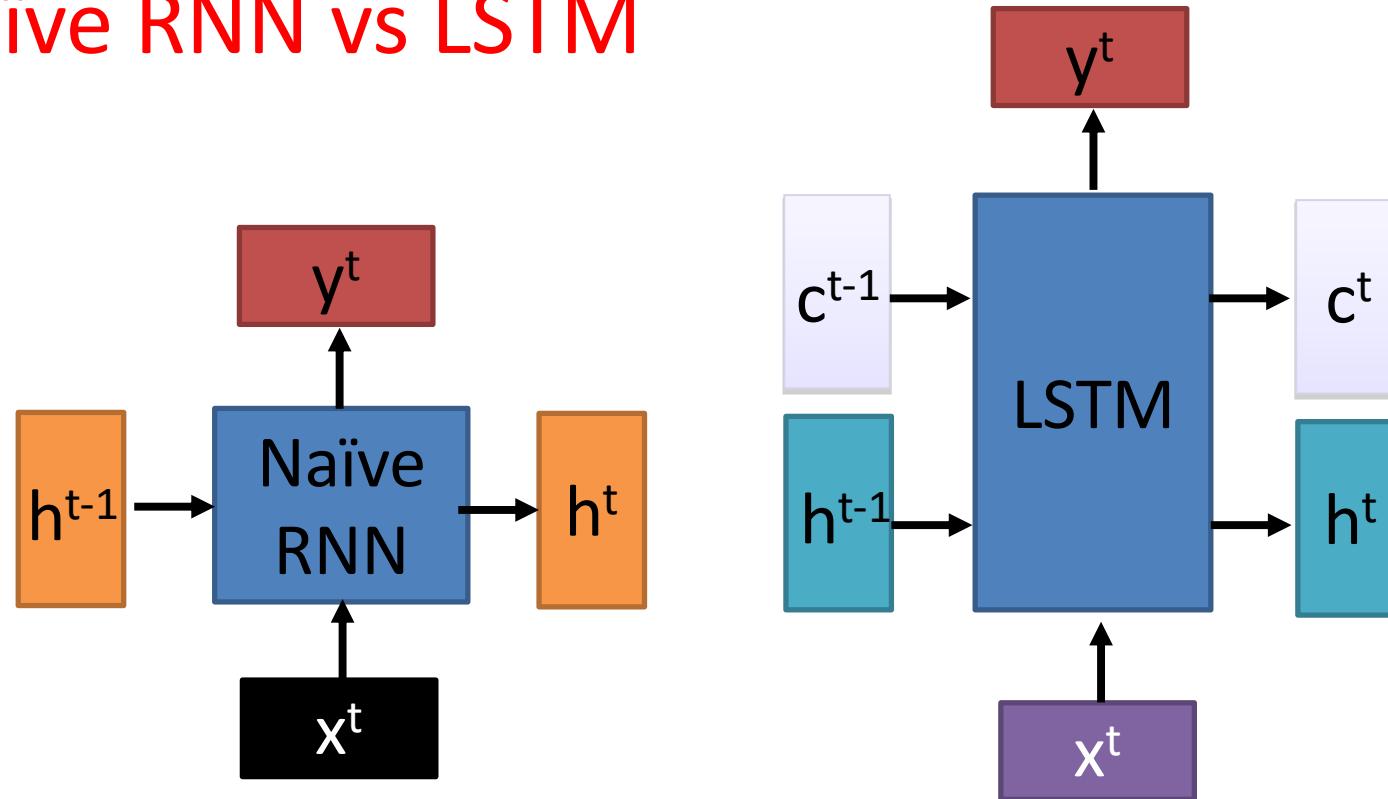


(a) RNN



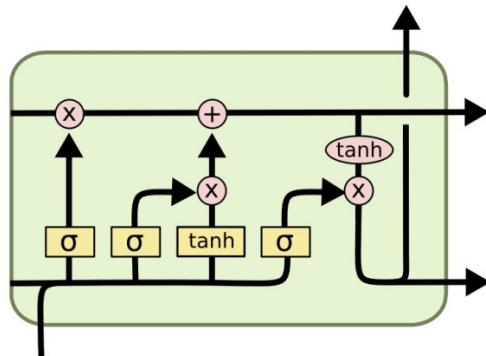
(b) LSTM

## Naïve RNN vs LSTM



$c$  changes slowly  $\rightarrow c^t$  is  $c^{t-1}$  added by something

$h$  changes faster  $\rightarrow h^t$  and  $h^{t-1}$  can be very different



These 4 matrix computation should be done concurrently.

$$z = \tanh(W h^{t-1} + x^t)$$

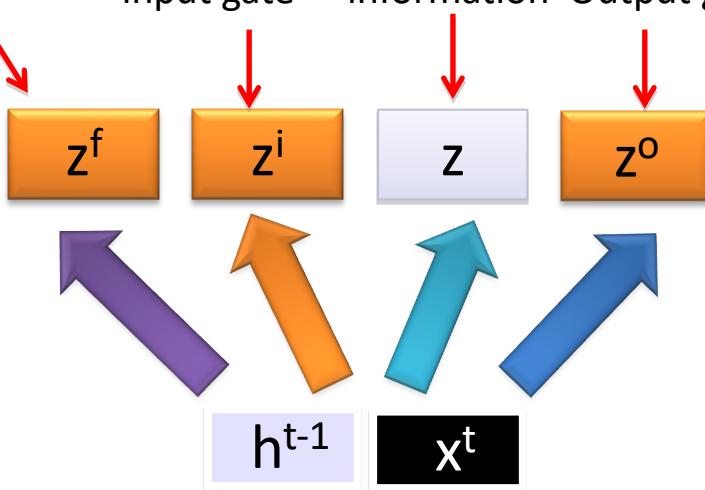
$$z^i = \sigma(W^i h^{t-1} + x^t)$$

$$z^f = \sigma(W^f h^{t-1} + x^t)$$

$$z^o = \sigma(W^o h^{t-1} + x^t)$$

$c^{t-1}$

Controls  
forget gate      Controls  
input gate      Updating  
information      Controls  
Output gate



Information flow of LSTM

# Applications

# Language Modeling

- Tell story
- 
- “Heeeeeel”
- ⇒ “Heeeloolllell”
- ⇒ “Hellooo”
- ⇒ “Hello”

tyntd-iafhatawiaoahrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e  
plia tklrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓ train more

"Tmont thithey" fomesscerliund  
Keushey. Thom here  
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwyl fil on aseterlome  
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓ train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of  
her hearly, and behs to so arwage fiving were to it belege, pavu say falling misfort  
how, and Gogition is so overelical and ofter.

↓ train more

"Why do what that day," replied Natasha, and wishing to himself the fact the  
princess, Princess Mary was easier, fed in had oftened him.  
Pierre aking his soul came to the packs and drove up his father-in-law women.

[http://cs231n.stanford.edu/slides/2017/cs231n\\_2017\\_lecture10.pdf](http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf)

# Image Captioning: Example Results

Captions generated using [neuraltalk](#)  
All images are [CC0 Public domain](#):  
[cat suitcase](#), [cat tree](#), [dog](#), [bear](#),  
[surfers](#), [tennis](#), [giraffe](#), [motorcycle](#)



*A cat sitting on a suitcase on the floor*



*A cat is sitting on a tree branch*



*A dog is running in the grass with a frisbee*



*A white teddy bear sitting in the grass*



*Two people walking on the beach with surfboards*



*A tennis player in action on the court*



*Two giraffes standing in a grassy field*



*A man riding a dirt bike on a dirt track*

# Image Captioning: Failure Cases



*A woman is holding a cat in her hand*



*A person holding a computer mouse on a desk*



*A woman standing on a beach holding a surfboard*



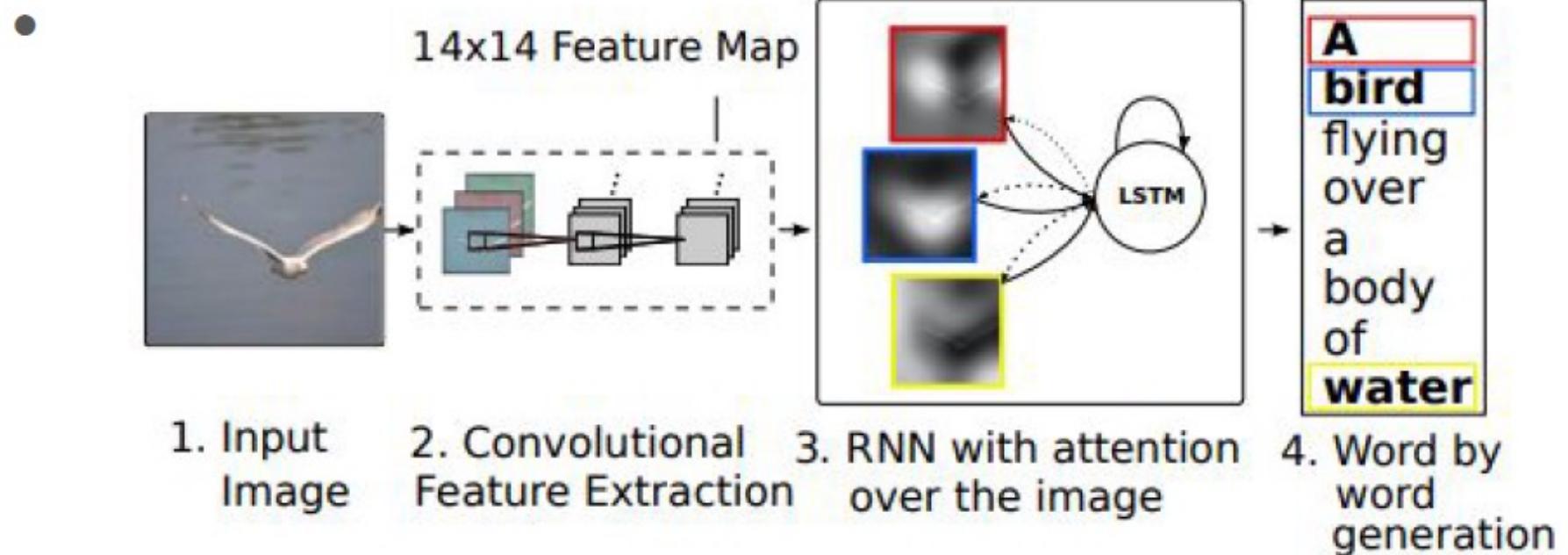
*A bird is perched on a tree branch*



*A man in a baseball uniform throwing a ball*

Captions generated using [neuraltalk2](#).  
All images are [CC0 Public domain](#): fur  
coat, handstand, spider web, baseball

# Image Attention: Image Captioning



Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International Conference on Machine Learning. 2015.

# Image Attention: Image Captioning



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



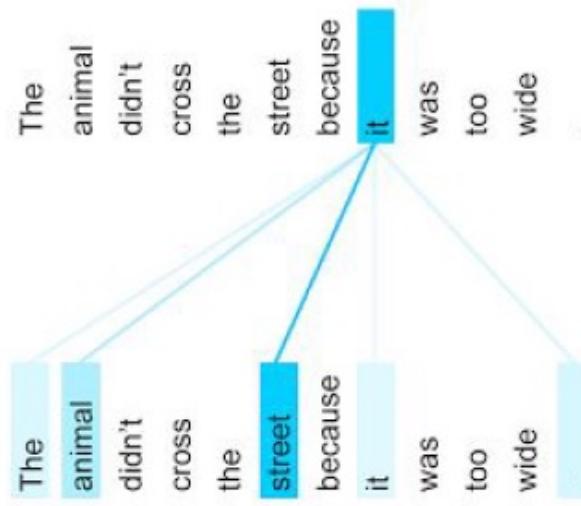
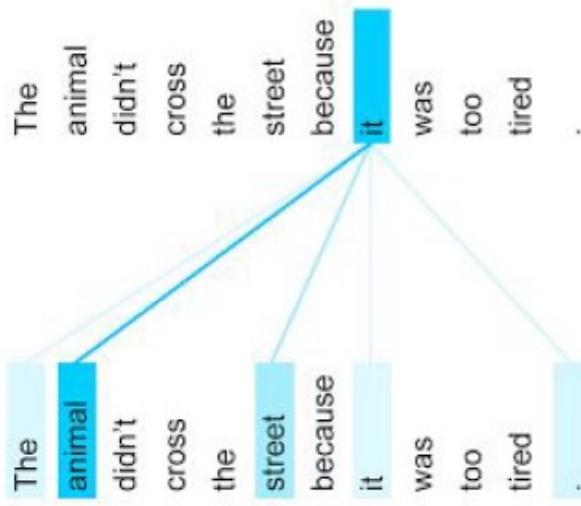
A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

# Attention is All You Need

- The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads)



# Generating Sequence

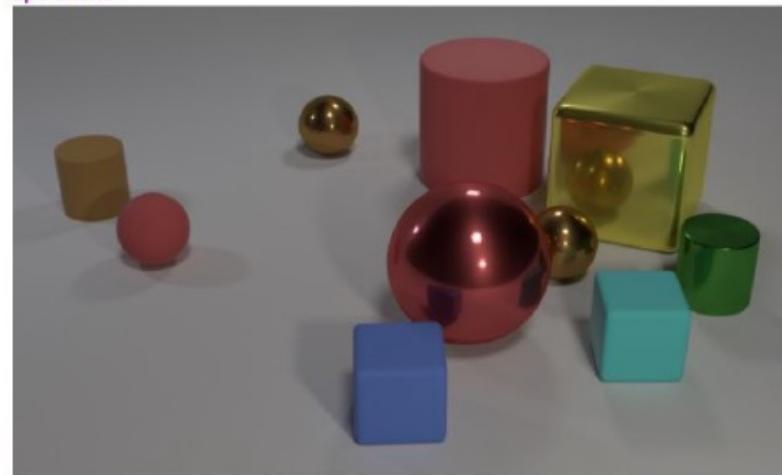
- Language modeling
  - Input: "A"
  - Output: "A quick brown fox jumps over the lazy dog."
- Handwriting stroke generation
  - Awesome Recurrent Neural Networks
  - Awesome Recurrent Neural Networks
  - Awesome Recurrent Neural Networks

# Visual Question Answering

- Reason the **relations** among Objects in image
- 
- “**What size** is the **cylinder** that is **left of** the **brown metal** thing that is **left of** the **big sphere**”
- 
- Dataset
  - CLEVR

<https://distill.pub/2016/augmented-rnns/>  
<http://cs.stanford.edu/people/jcjohns/clevr/>

Questions in CLEVR test various aspects of visual reasoning including **attribute identification**, **counting**, **comparison**, **spatial relationships**, and **logical operations**.



Q: Are there an **equal number** of **large things** and **metal spheres**?

Q: **What size** is the **cylinder** that is **left of** the **brown metal** thing that is **left of** the **big sphere**?

Q: There is a **sphere** with the **same size as** the **metal cube**; is it **made of the same material as** the **small red sphere**?

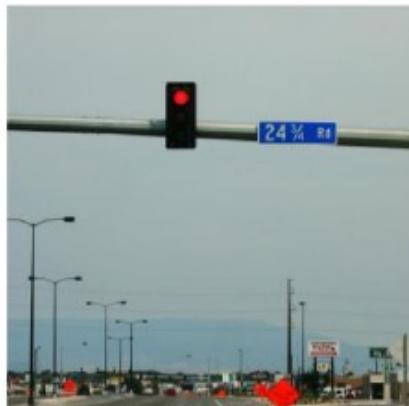
Q: **How many** objects are **either small cylinders or red** things?

# Visual Question Answering



**Q:** What endangered animal is featured on the truck?

- A: A bald eagle.
- A: A sparrow.
- A: A humming bird.
- A: A raven.



**Q:** Where will the driver go if turning right?

- A: Onto 24 1/4 Rd.
- A: Onto 25 1/4 Rd.
- A: Onto 23 1/4 Rd.
- A: Onto Main Street.



**Q:** When was the picture taken?

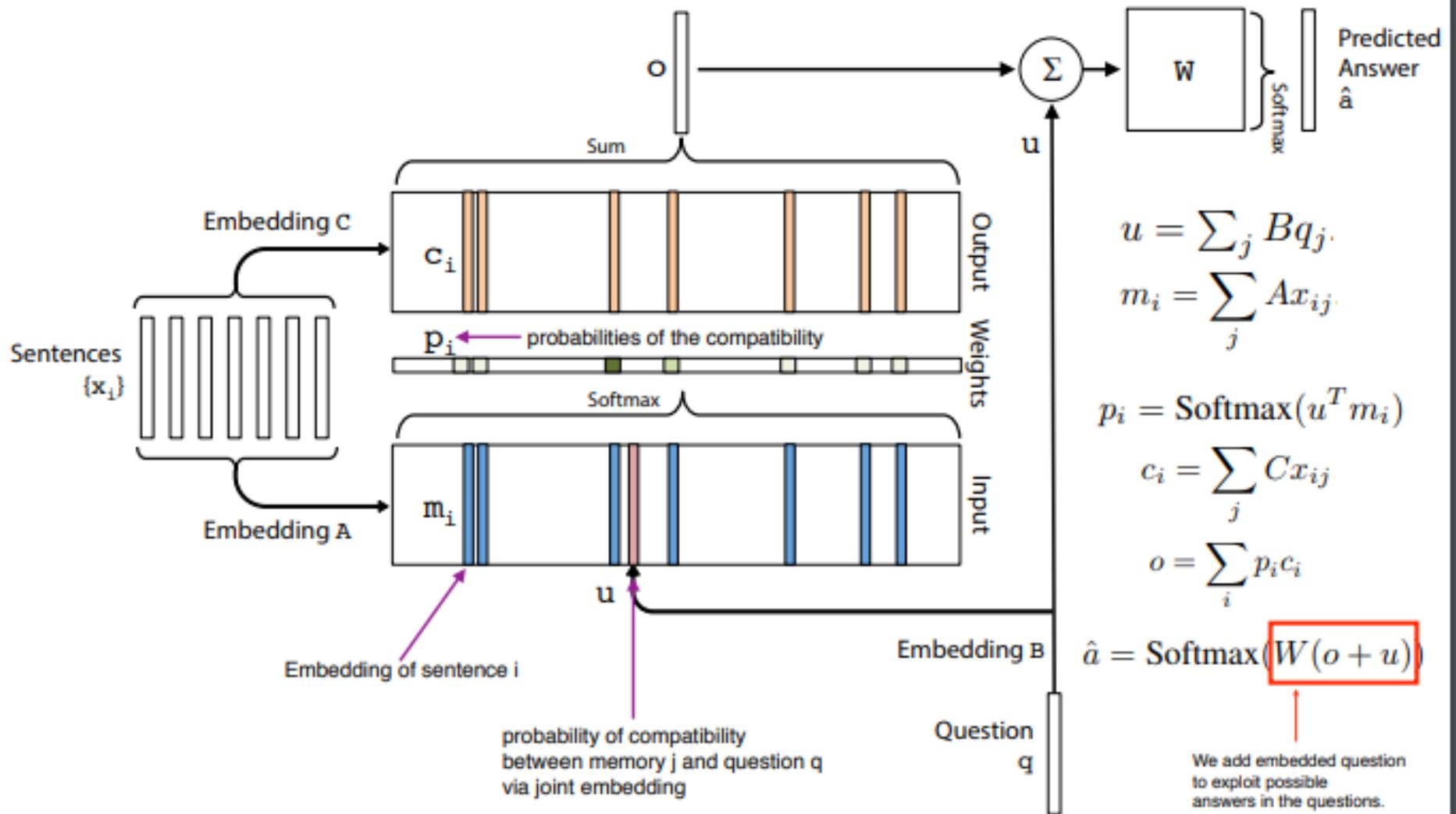
- A: During a wedding.
- A: During a bar mitzvah.
- A: During a funeral.
- A: During a Sunday church service.



**Q:** Who is under the umbrella?

- A: Two women.
- A: A child.
- A: An old man.
- A: A husband and a wife.

# End-to-End Memory Networks



# Question Answering

1. **Mary** moved to the **bathroom**
2. **John** went to the **hallway**
3. **Where** is **Mary**?
4. Answer: **bathroom**

Weston, Jason, Sumit Chopra, and Antoine Bordes. "Memory networks." arXiv preprint arXiv:1410.3916 (2014).  
Sukhbaatar, Sainbayar, Jason Weston, and Rob Fergus. "End-to-end memory networks." Advances in neural information processing systems. 2015.  
Andreas, Jacob, et al. "Learning to compose neural networks for question answering." arXiv preprint arXiv:1601.01705 (2016)  
<http://cs.umd.edu/~miyyer/data/deepqa.pdf>  
<https://research.fb.com/downloads/babi/>

## End-to-end Memory Networks

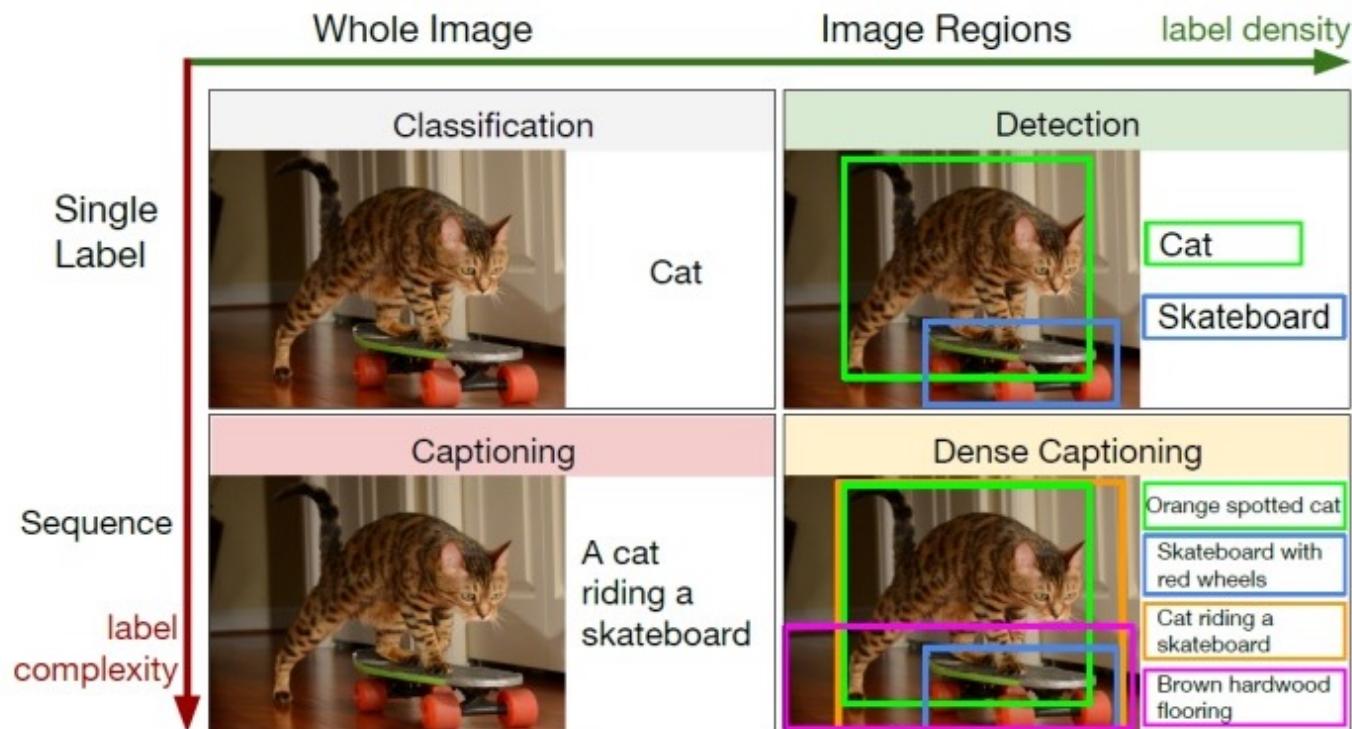
Story (1: 1 supporting fact)	Support	Hop 1	Hop 2	Hop 3
Daniel went to the bathroom.		0.00	0.00	0.03
Mary travelled to the hallway.		0.00	0.00	0.00
John went to the bedroom.		0.37	0.02	0.00
John travelled to the bathroom.	yes	0.60	0.98	0.96
Mary went to the office.		0.01	0.00	0.00
Where is John? Answer: bathroom Prediction: bathroom				
Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow				
Story (2: 2 supporting facts)	Support	Hop 1	Hop 2	Hop 3
John dropped the milk.		0.06	0.00	0.00
John took the milk there.	yes	0.88	1.00	0.00
Sandra went back to the bathroom.		0.00	0.00	0.00
John moved to the hallway.	yes	0.00	0.00	1.00
Mary went back to the bedroom.		0.00	0.00	0.00
Where is the milk? Answer: hallway Prediction: hallway				
Story (18: size reasoning)	Support	Hop 1	Hop 2	Hop 3
The suitcase is bigger than the chest.	yes	0.00	0.88	0.00
The box is bigger than the chocolate.		0.04	0.05	0.10
The chest is bigger than the chocolate.	yes	0.17	0.07	0.90
The chest fits inside the container.		0.00	0.00	0.00
The chest fits inside the box.		0.00	0.00	0.00
Does the suitcase fit in the chocolate? Answer: no Prediction: no				

# DenseCap: Fully Convolutional Localization Networks for Dense Captioning

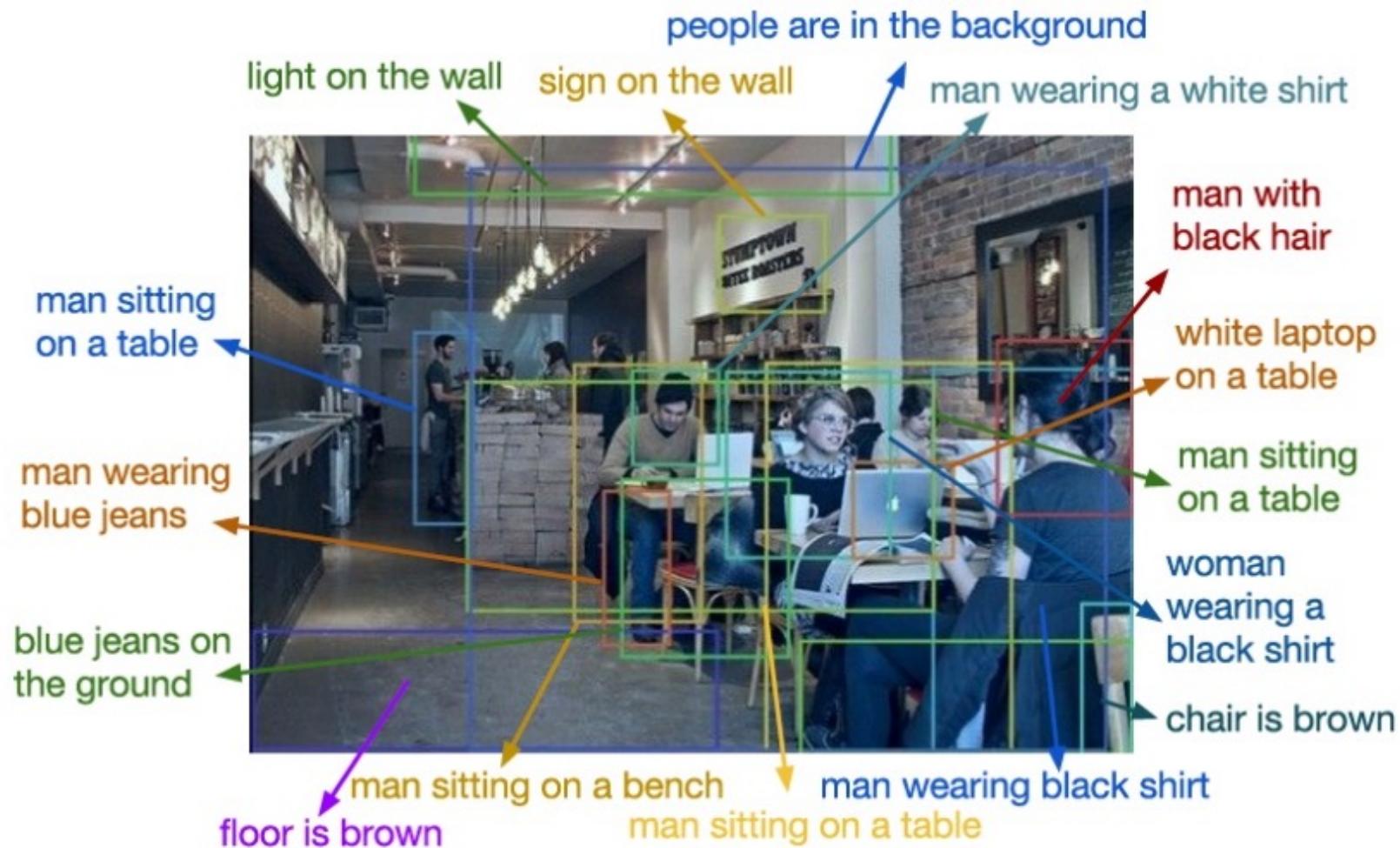
Justin Johnson\* Andrey Karpathy\* Li Fei-Fei

Department of Computer Science, Stanford University

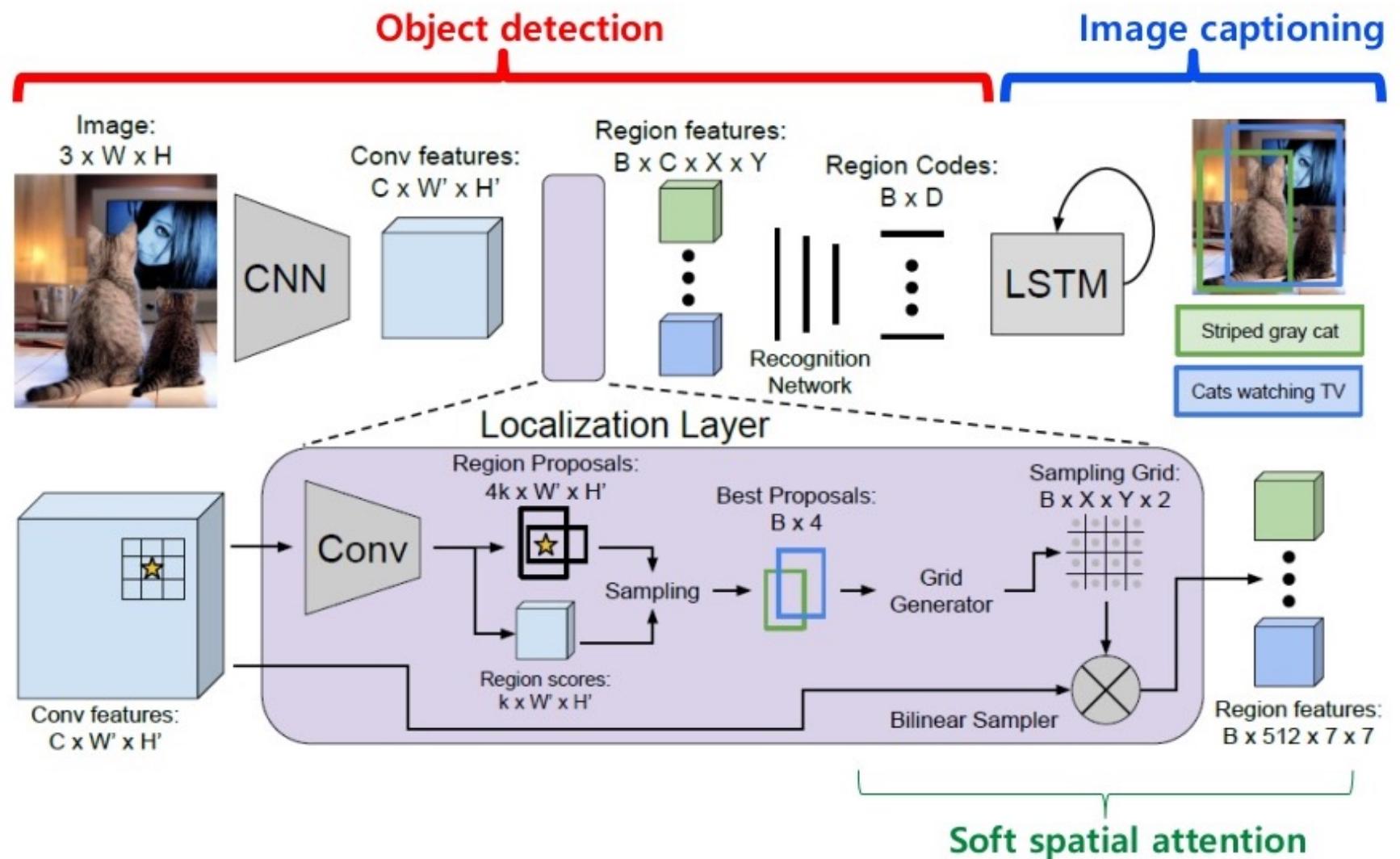
{jcjohns, karpathy, feifeili}@cs.stanford.edu

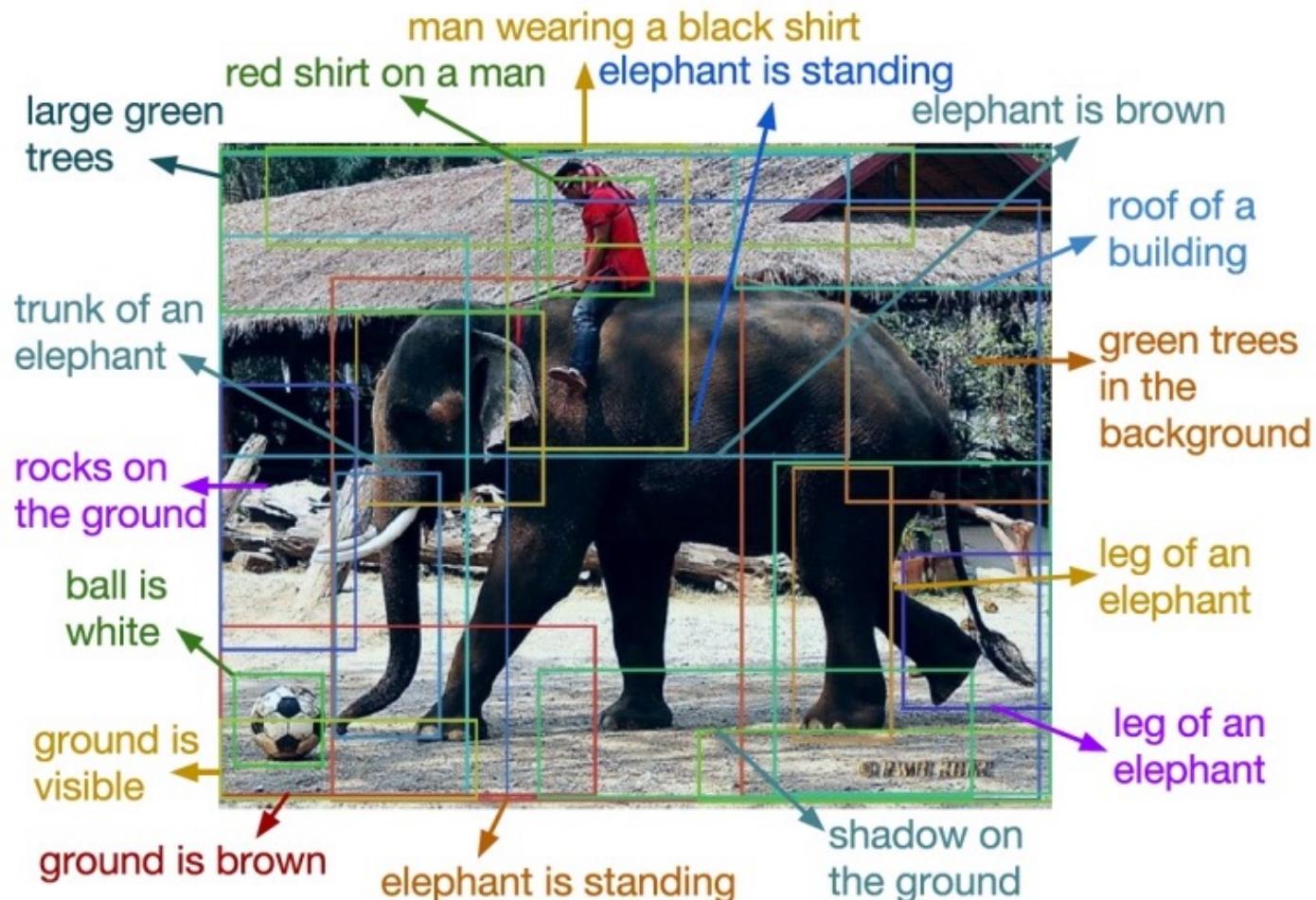


# DenseCap

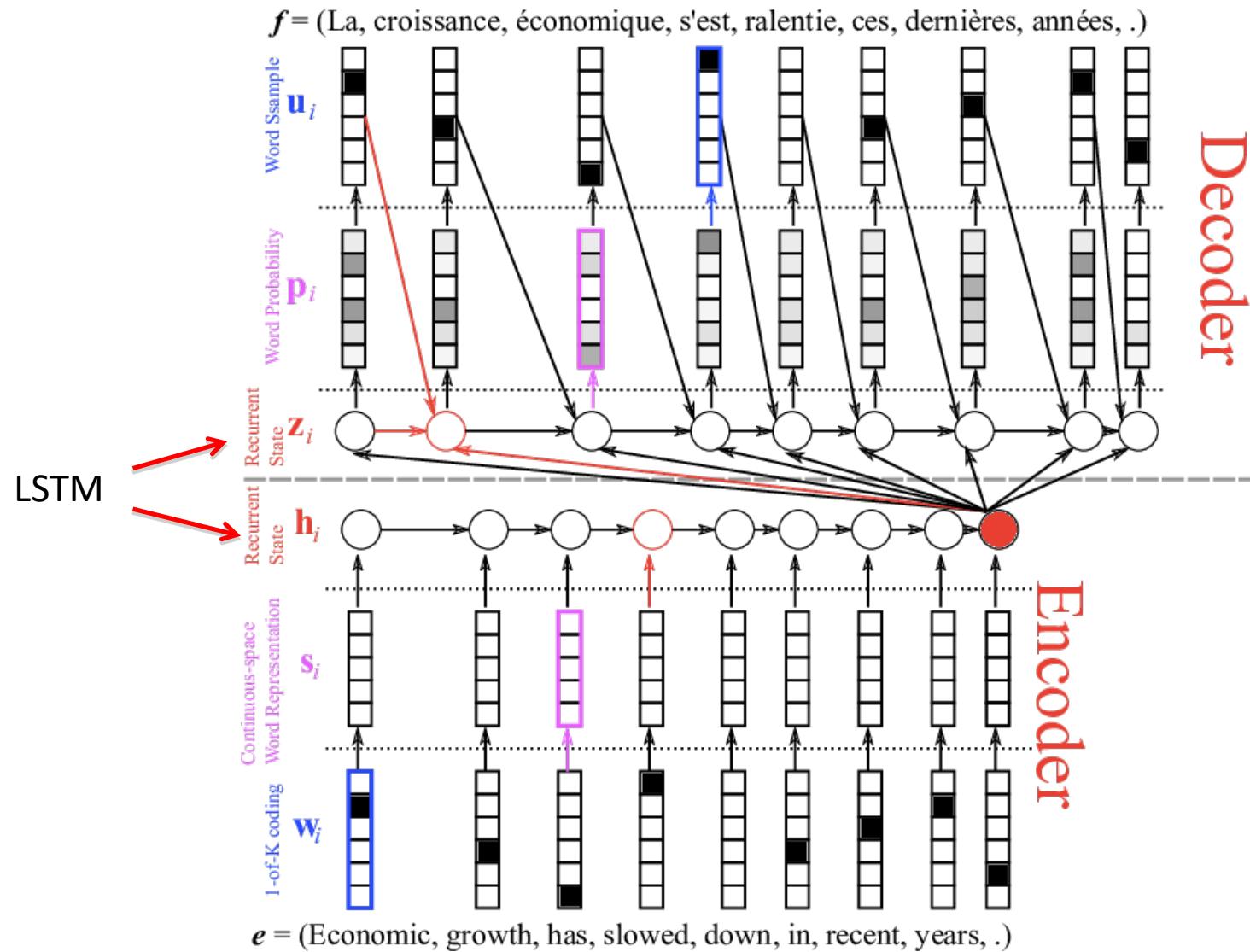


# Model Architecture

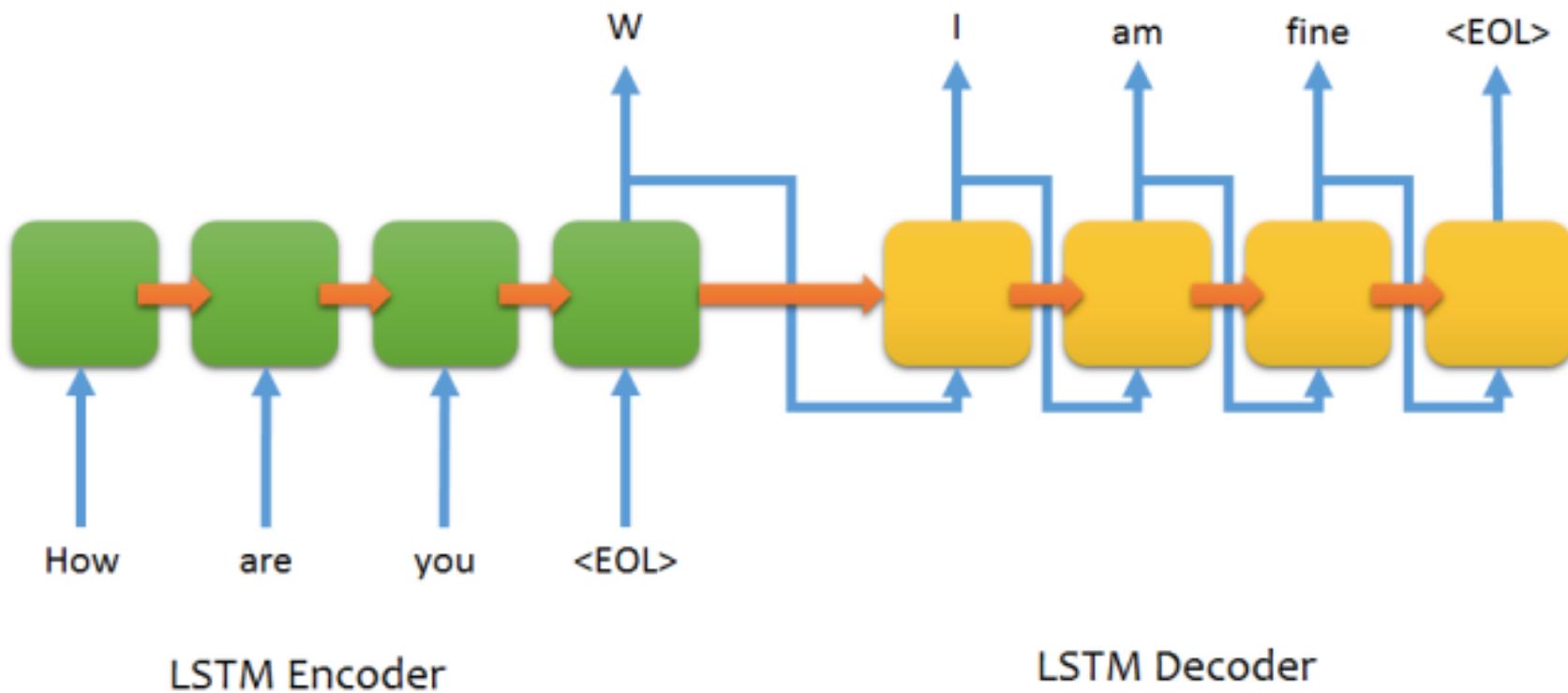




# Neural machine translation



# Sequence to sequence chat model



# Chat with context

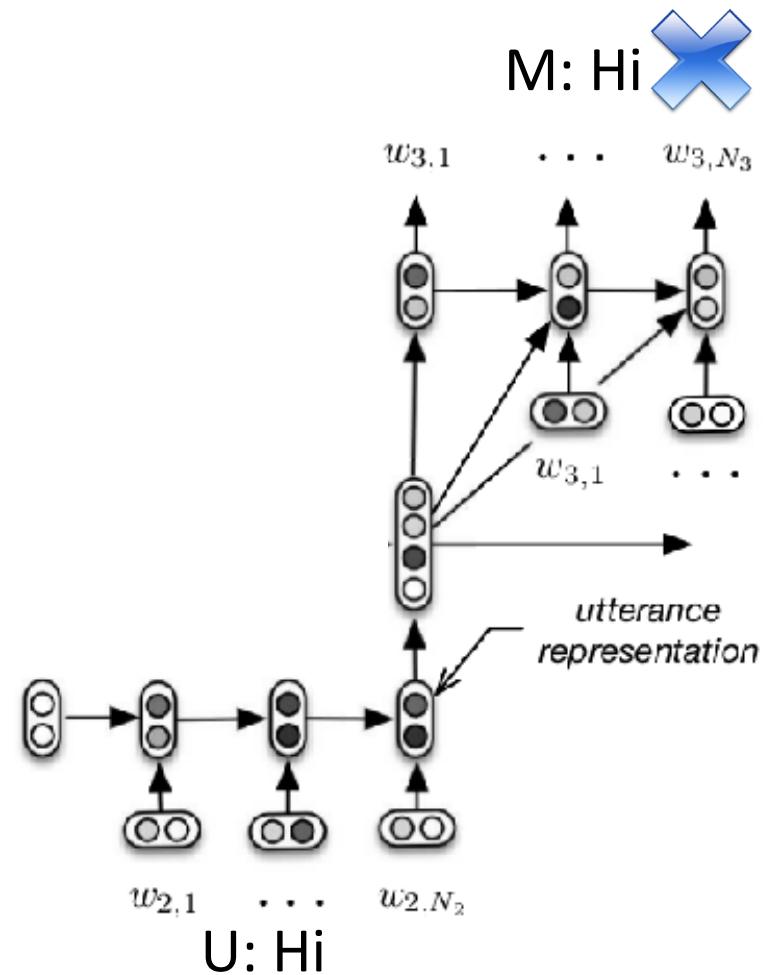
M: Hello

U: Hi

M: Hi

M: Hello

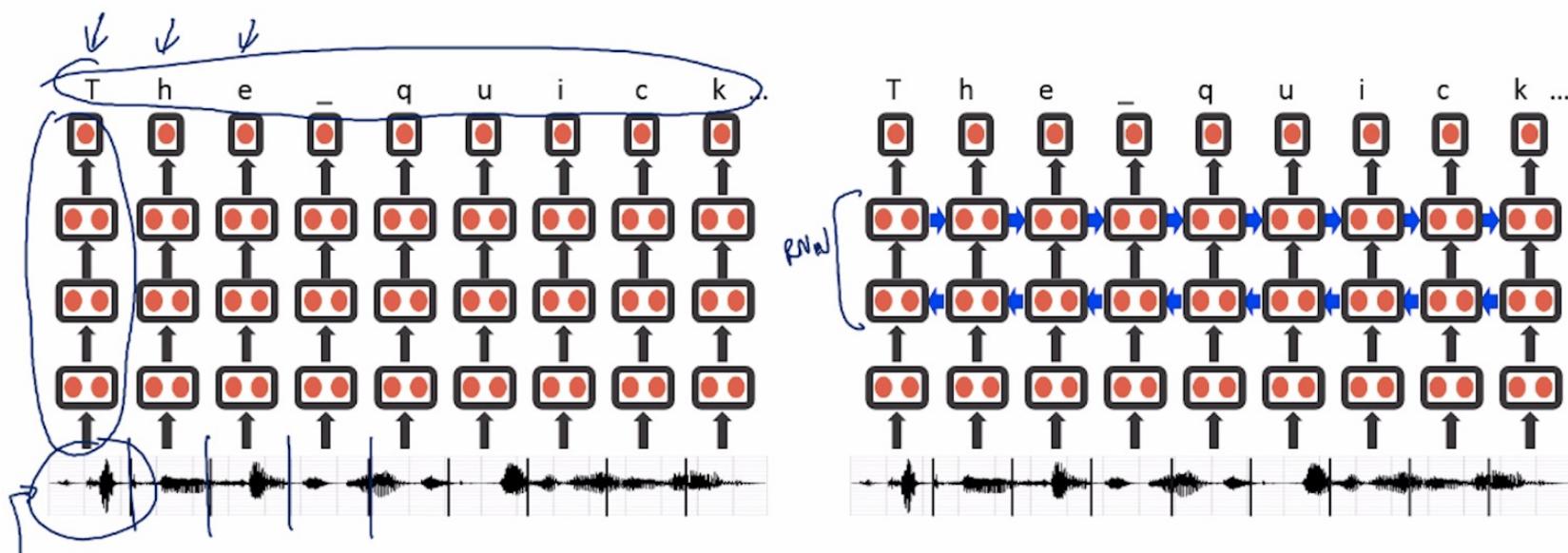
M: Hi 



Serban, Iulian V., Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau, 2015  
"Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models."

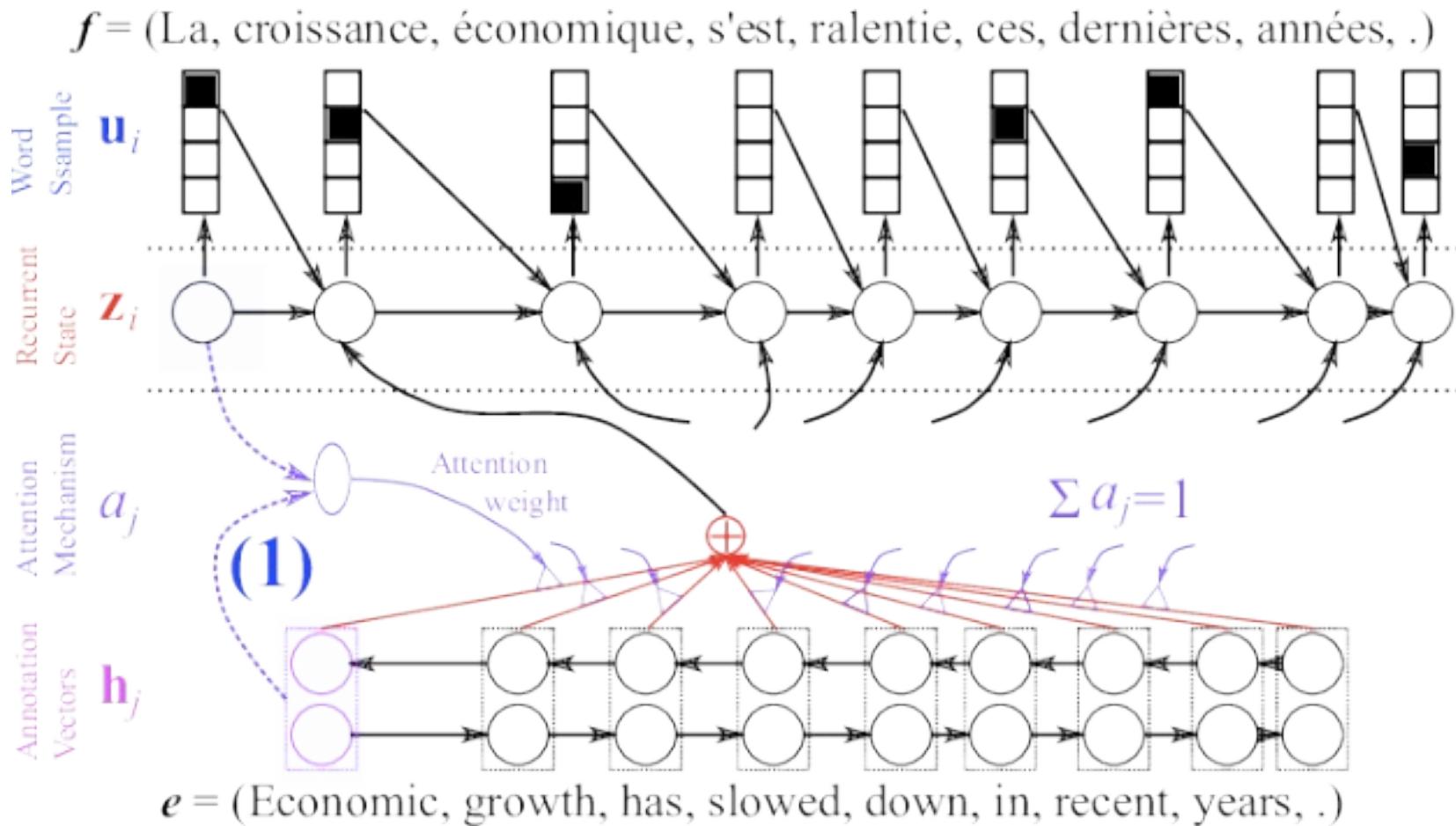
# Baidu's speech recognition using RNN

Speech recognition example (Deep Speech)

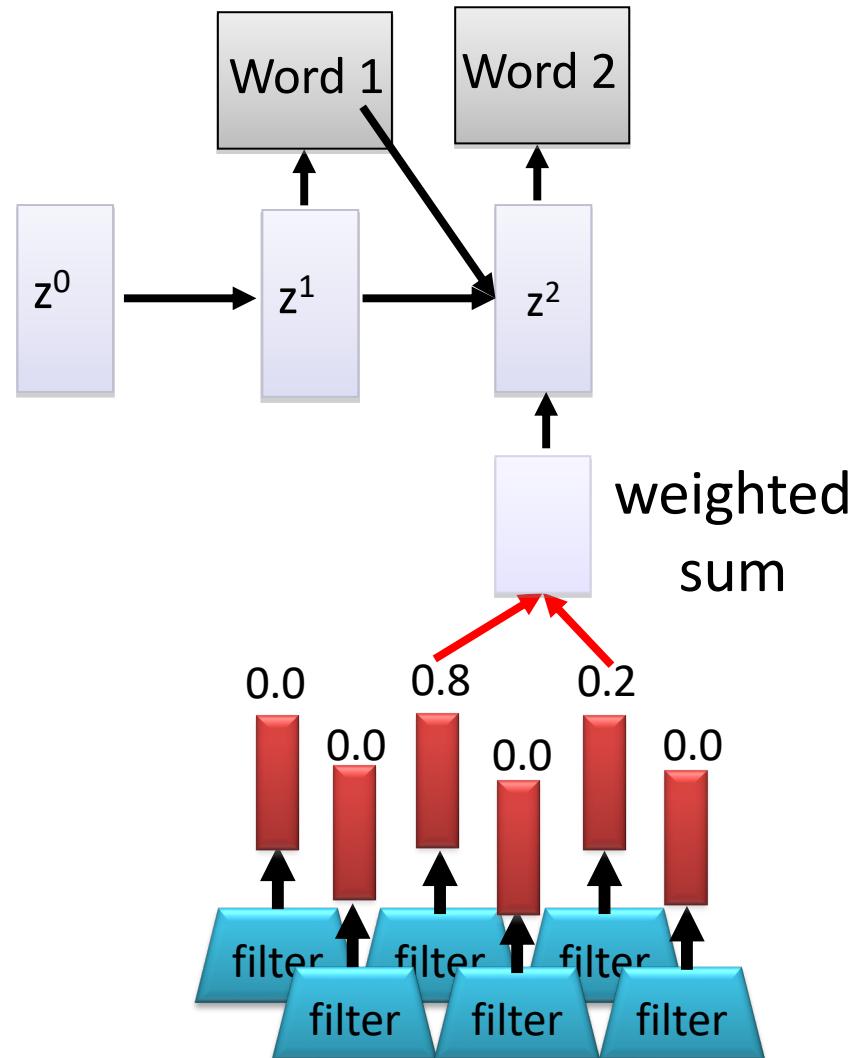
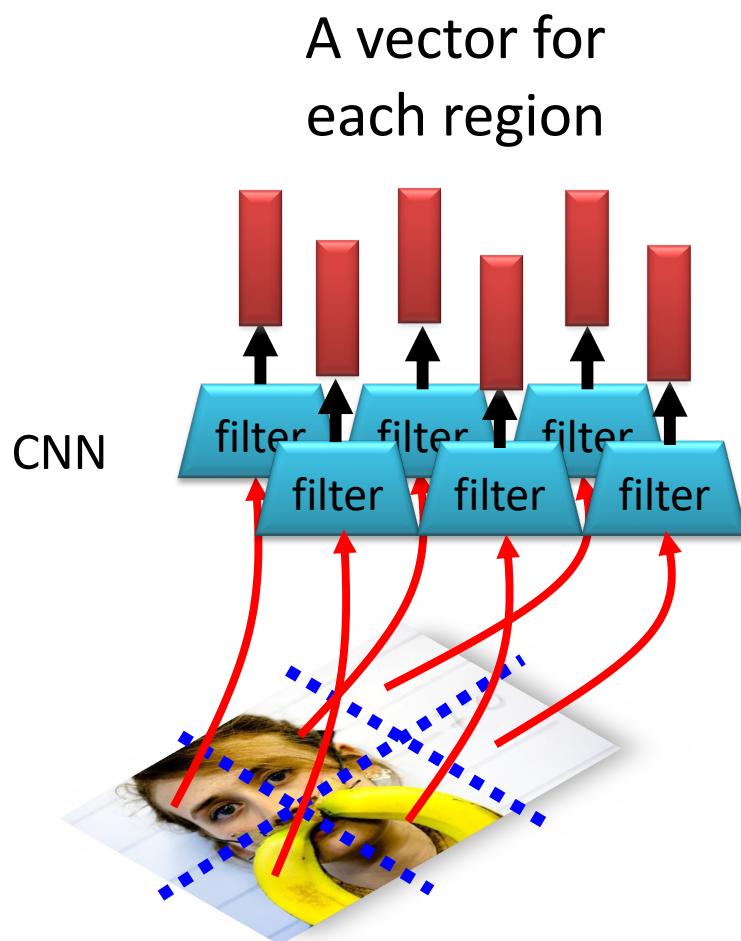


Andrew Ng

# Attention



# Image Caption Generation



# Image Caption Generation



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML, 2015

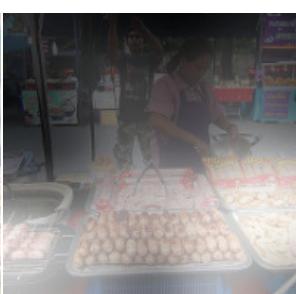
# Image Caption Generation



A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.



A person is standing on a beach with a surfboard.

A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML, 2015



**Ref:** A man and a woman ride a motorcycle  
A **man** and a **woman** are **talking** on the **road**



**Ref:** A woman is frying food  
**Someone** is **frying** a **fish** in a **pot**

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, Aaron Courville, "Describing Videos by Exploiting Temporal Structure", ICCV, 2015

# Demo

# Social Network

---



**No introduction required**



**Really?**

We still need to understand a  
few properties



disclaimer: the brand logos are used here entirely for educational purpose <sup>2</sup>

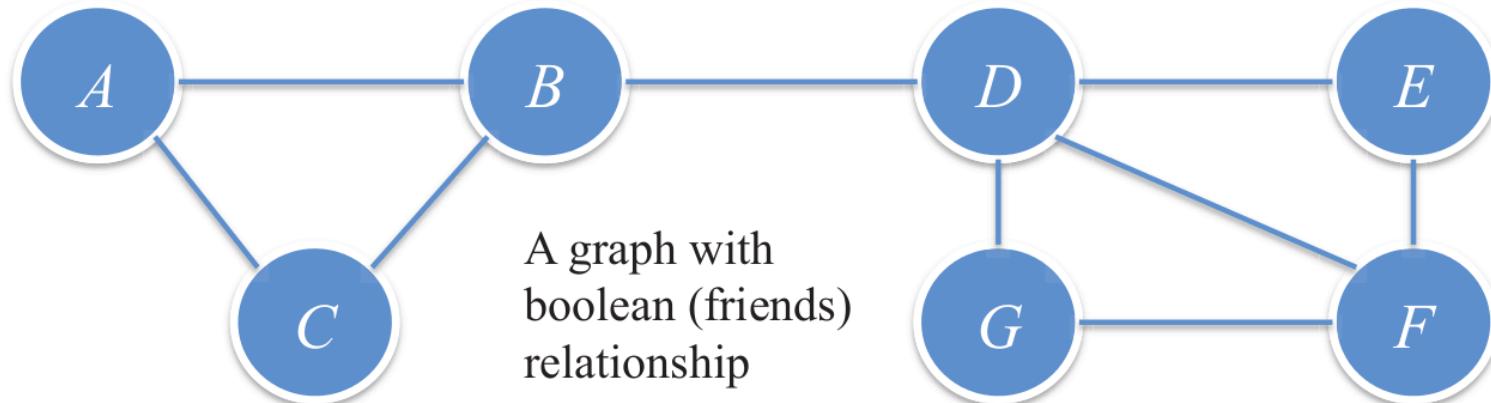
# Social Network

---

- A collection of **entities**
  - Typically people, but could be something else too
- At least one **relationship** between entities of the network
  - For example: friends
  - Sometimes **boolean**: two people are either friends or they are not
  - May have a **degree**
  - **Discrete** degree: friends, family, acquaintances, or none
  - Degree – **real number**: the fraction of the average day that two people spend talking to each other
- An assumption of **nonrandomness** or **locality**
  - Hard to formalize
  - Intuition: that relationships tend to cluster
  - If entity A is related to both B and C, then the probability that B and C are related is higher than average (random)

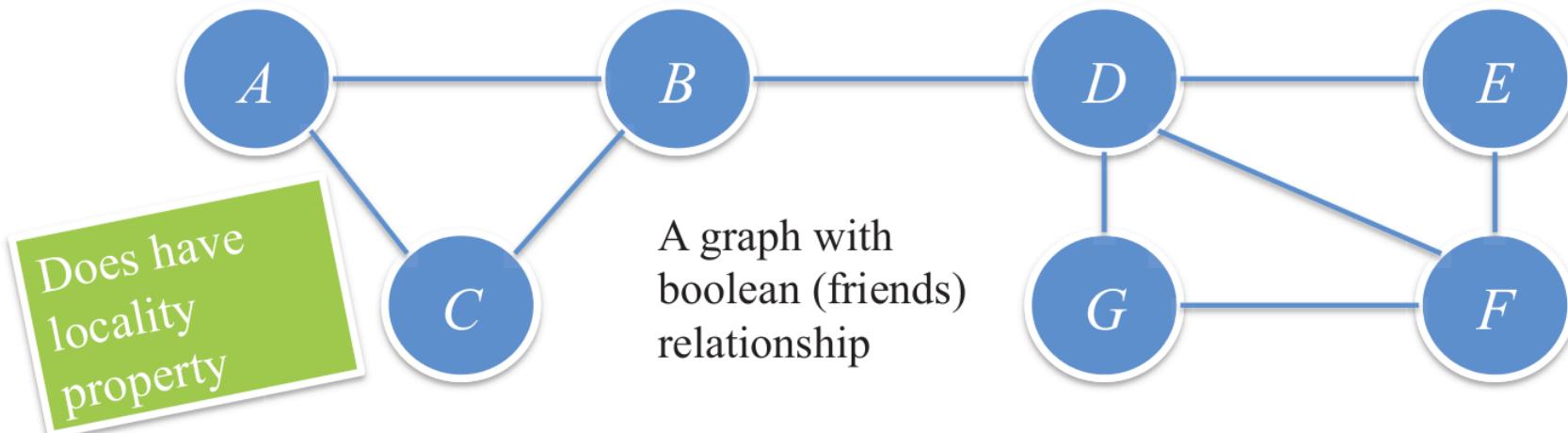
# Social Network as a Graph

---



- Check for the non-randomness criterion
- In a random graph  $(V,E)$  of 7 nodes and 9 edges, if  $XY$  is an edge,  $YZ$  is an edge, what is the probability that  $XZ$  is an edge?
  - For a large random graph, it would be close to  $|E|/(|V|C_2) = 9/21 \sim 0.43$
  - Small graph:  $XY$  and  $YZ$  are already edges, so compute within the rest
  - So the probability is  $(|E|-2)/(|V|C_2-2) = 7/19 = 0.37$
- Now let's compute what is the probability for this graph in particular

# Social Network as a Graph



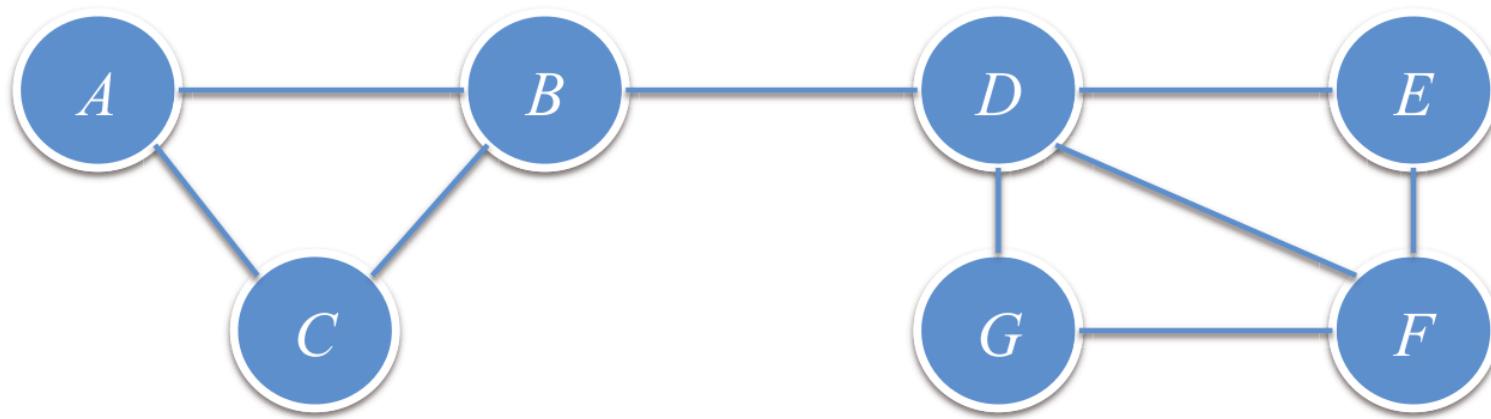
- For each  $X$ , check possible  $YZ$  and check if  $YZ$  is an edge or not
- Example: if  $X = A$ ,  $YZ = \{BC\}$ , it is an edge

$X=$	$YZ=$	<i>Yes/Total</i>
$A$	$BC$	1/1
$B$	$AC, AD, CD$	1/3
$C$	$AB$	1/1
$D$	$BE, BG, BF, EF, EG, FG$	2/6

$X=$	$YZ=$	<i>Yes/Total</i>
$E$	$DF$	1/1
$F$	$DE, DG, EG$	2/3
$G$	$DF$	1/1
<i>Total</i>		9/16 ~ <b>0.56</b>

# Types of Social (or Professional) Networks

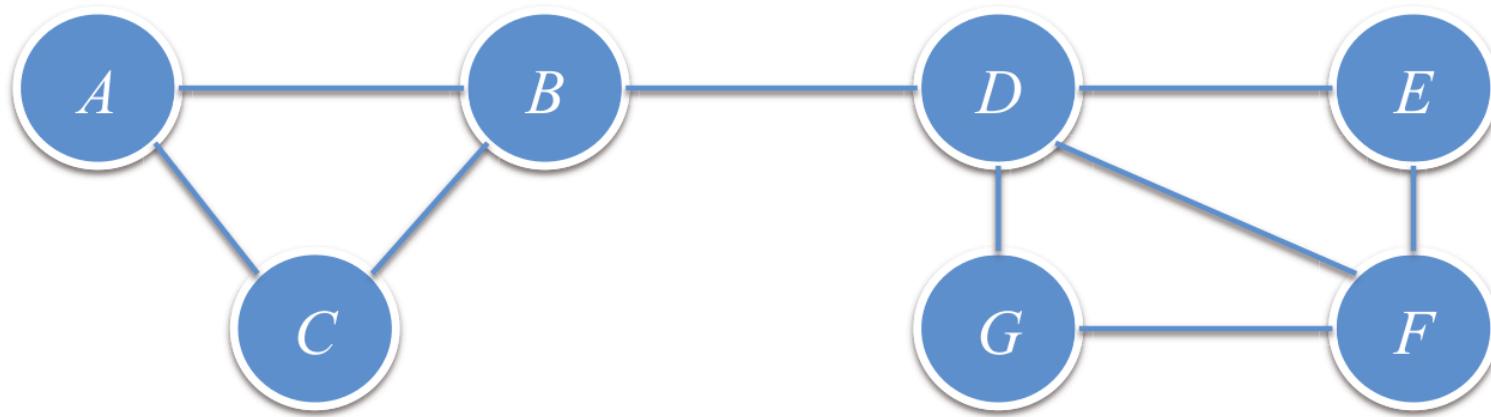
---



- Of course, the “social network”. But also several other types
- Telephone network
- Nodes are phone numbers
- $AB$  is an edge if  $A$  and  $B$  talked over phone within the last one week, or month, or ever
- Edges could be weighted by the number of times phone calls were made, or total time of conversation

# Types of Social (or Professional) Networks

---



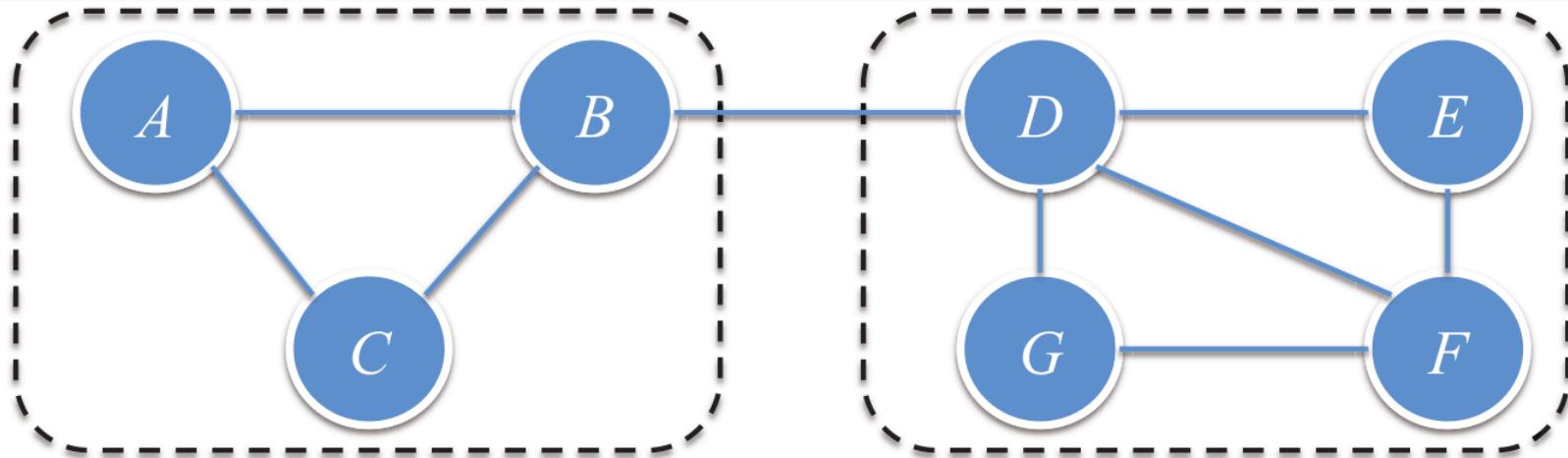
- Email network: nodes are email addresses
- $AB$  is an edge if  $A$  and  $B$  sent mails to each other within the last one week, or month, or ever
  - One directional edges would allow spammers to have edges
- Edges could be weighted
- Other networks: collaboration network – authors of papers, jointly written papers or not
- Also networks exhibiting locality property

# Clustering of Social Network Graphs

---

- Locality property → there are clusters
- Clusters are communities
  - People of the same institute, or company
  - People in a photography club
  - Set of people with “Something in common” between them
- Need to define a distance between points (nodes)
- In graphs with weighted edges, different distances exist
- For graphs with “friends” or “not friends” relationship
  - Distance is 0 (friends) or 1 (not friends)
  - Or 1 (friends) and infinity (not friends)
  - Both of these violate the triangle inequality
  - Fix triangle inequality: distance = 1 (friends) and 1.5 or 2 (not friends) or length of shortest path

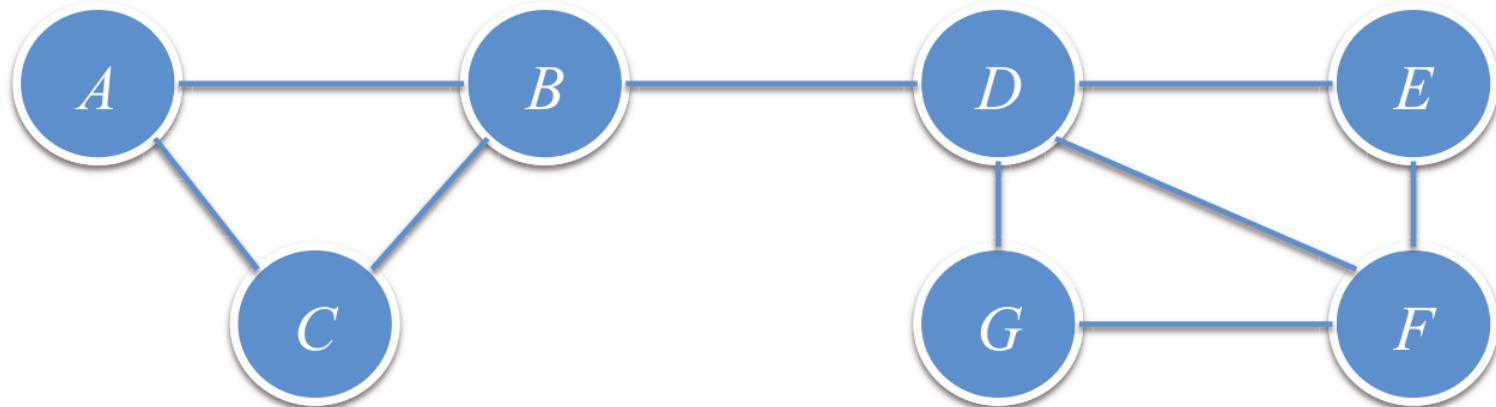
# Traditional Clustering



- Intuitively, two communities
- Traditional clustering depends on the distance
  - Likely to put two nodes with small distance in the same cluster
  - Social network graphs would have cross-community edges
  - Severe merging of communities likely
- May join  $B$  and  $D$  (and hence the two communities) with not so low probability

# Betweenness of an Edge

---

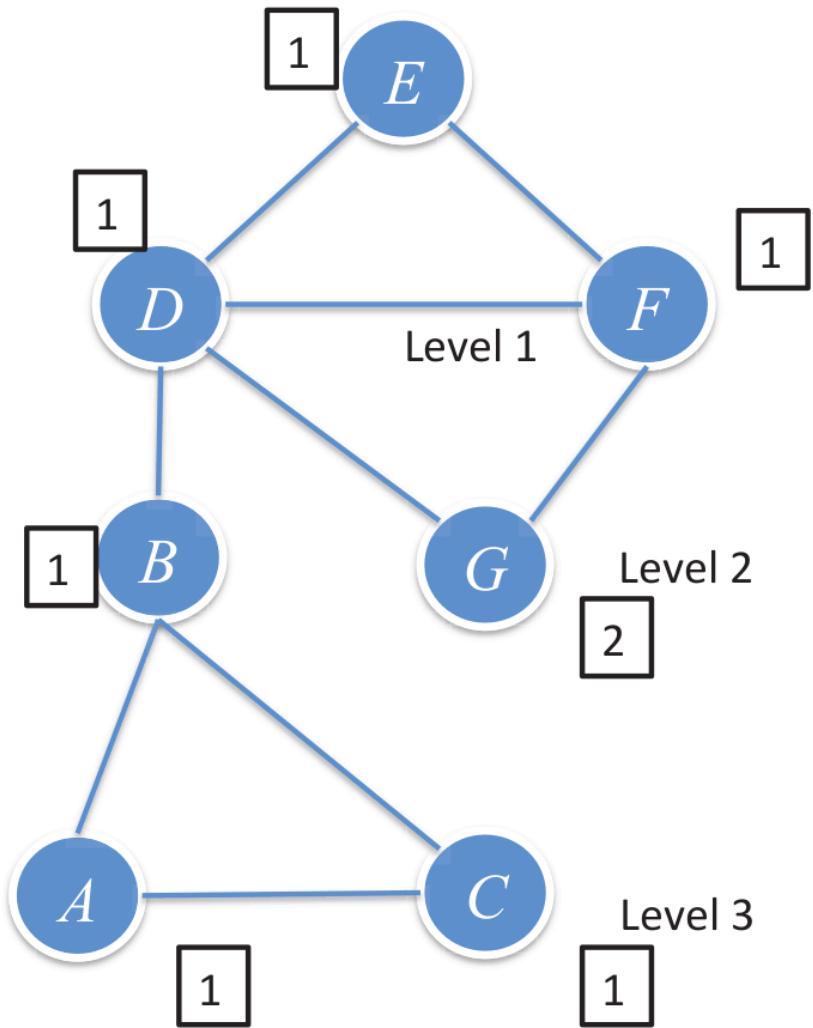


- Betweenness of an edge AB: #of pairs of nodes (X, Y) such that AB lies on the shortest path between X and Y
  - There can be more than one shortest paths between X and Y
  - Credit AB the fraction of those paths which include the edge AB
- High score of betweenness means?
  - The edge runs “between” two communities
- Betweenness gives a better measure
  - Edges such as *BD* get a higher score than edges such as *AB*
- Not a distance measure, may not satisfy triangle inequality. Doesn’t matter!

# The Girvan – Newman Algorithm

- Step 1 – BFS: Start at a node  $X$ , perform a BFS with  $X$  as root
- Observe: level of node  $Y = \text{length of shortest path from } X \text{ to } Y$
- Edges between level are called “DAG” edges
  - Each DAG edge is part of at least one shortest path from  $X$
- Step 2 – Labeling: Label each node  $Y$  by the number of shortest paths from  $X$  to  $Y$

Calculate *betweenness* of edges



# The Girvan – Newman Algorithm

Step 3 – credit sharing:

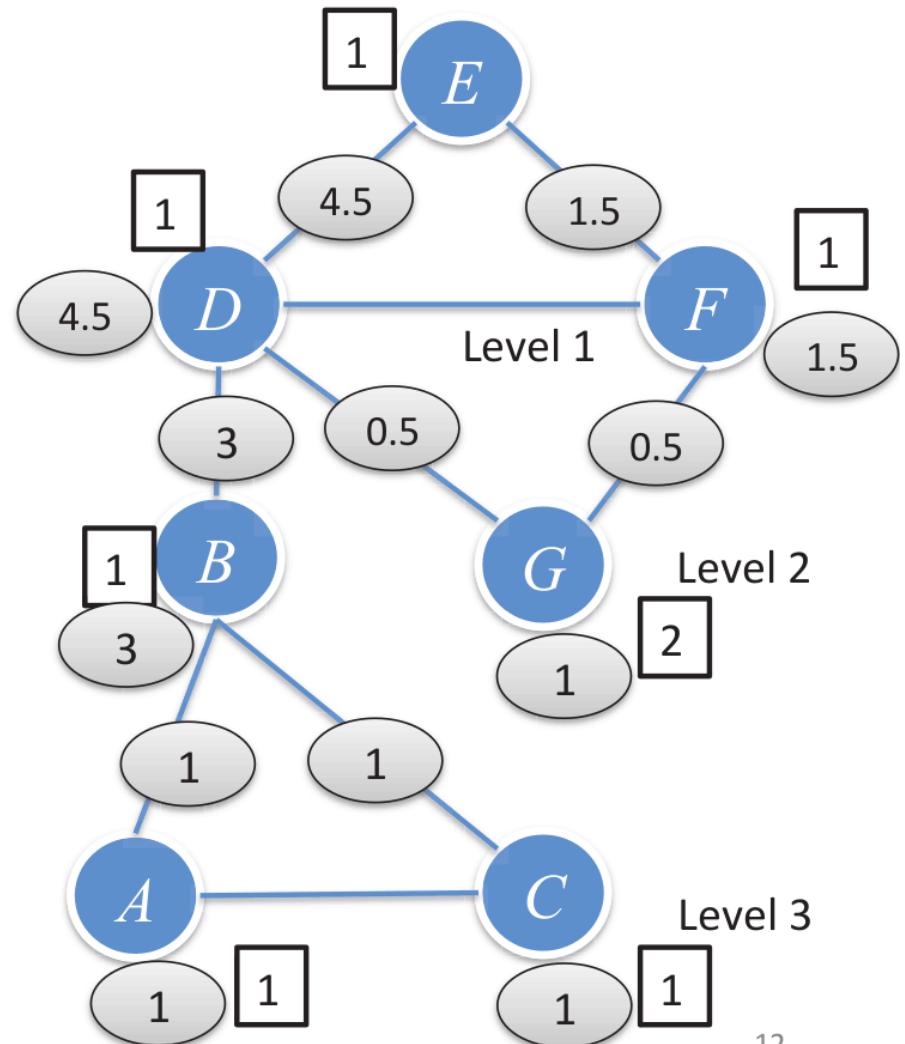
- Each leaf node gets credit 1
- Each non-leaf node gets  $1 + \text{sum}(\text{credits of the DAG edges to the level below})$
- Credit of DAG edges: Let  $Y_i$  ( $i=1, \dots, k$ ) be parents of  $Z$ ,  $p_i = \text{label}(Y_i)$

$$\text{credit}(Y_i, Z) = \frac{\text{credit}(Z) \times p_i}{(p_1 + \dots + p_k)}$$

- Intuition: a DAG edge  $Y_iZ$  gets the share of credit of  $Z$  proportional to the #of shortest paths from  $X$  to  $Z$  going through  $Y_iZ$

Finally: Repeat Steps 1, 2 and 3 with each node as root. For each edge, betweenness = sum credits obtained in all iterations / 2

Calculate betweenness of edges



# Computation in practice

---

- Complexity:  $n$  nodes,  $e$  edges
  - BFS starting at each node:  $O(e)$
  - Do it for  $n$  nodes
  - Total:  $O(ne)$  time
  - Very expensive
- Method in practice
  - Choose a random subset  $W$  of the nodes
  - Compute credit of each edge starting at each node in  $W$
  - Sum and compute betweenness
  - A reasonable approximation

# Finding Communities using Betweenness

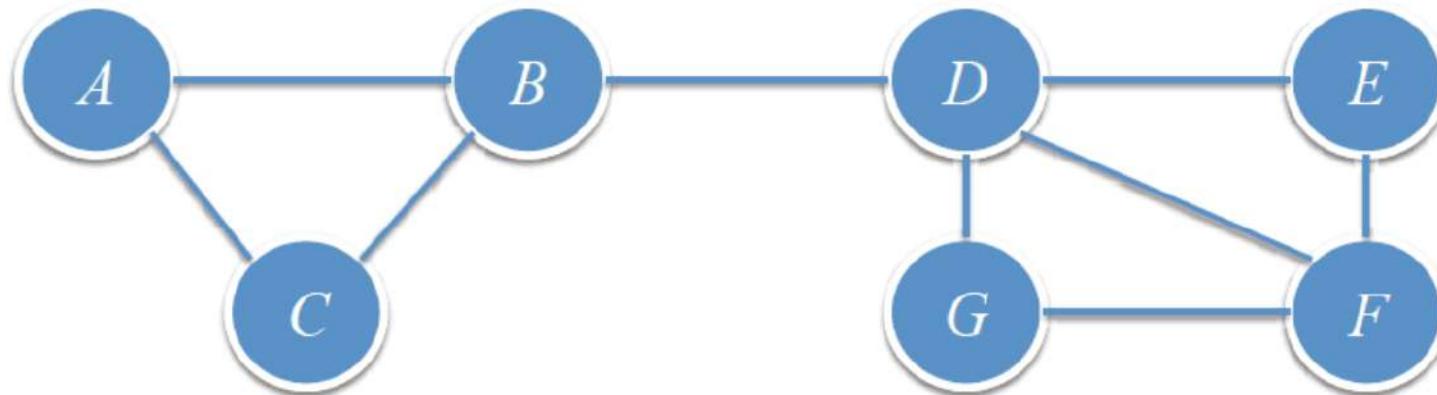
---

Method 1:

- Keep adding edges (among existing ones) starting from lowest betweenness
- Gradually join small components to build large connected components

# Betweenness and Girvan Newman Algorithm

# Betweenness of an Edge

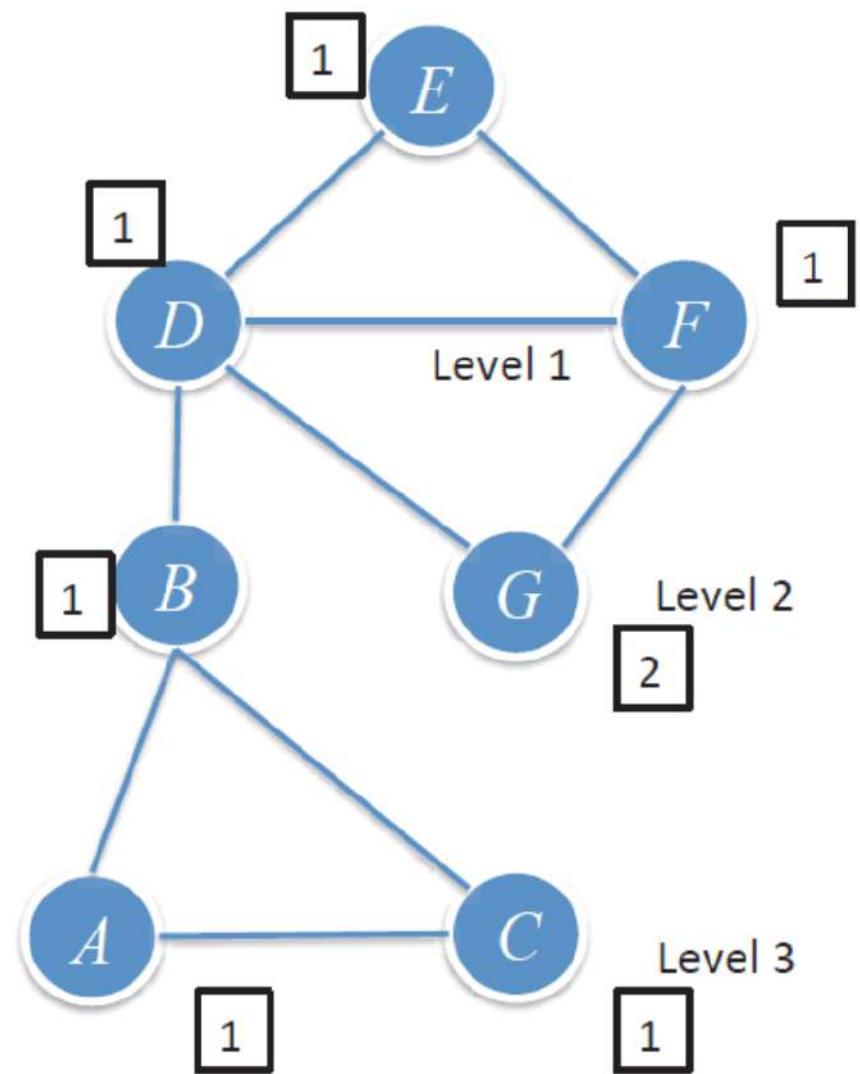


- Betweenness of an edge AB: #of pairs of nodes (X,Y) such that AB lies on the shortest path between X and Y
  - There can be more than one shortest paths between X and Y
  - Credit AB the fraction of those paths which include the edge AB
- High score of betweenness means?
  - The edge runs “between” two communities
- Betweenness gives a better measure
  - Edges such as *BD* get a higher score than edges such as *AB*
- Not a distance measure, may not satisfy triangle inequality. Doesn’t matter!

# The Girvan – Newman Algorithm

- Step 1 – BFS: Start at a node  $X$ , perform a BFS with  $X$  as root
- Observe: level of node  $Y =$  length of shortest path from  $X$  to  $Y$
- Edges between level are called “DAG” edges
  - Each DAG edge is part of at least one shortest path from  $X$
- Step 2 – Labeling: **Label** each node  $Y$  by the number of shortest paths from  $X$  to  $Y$

Calculate *betweenness* of edges



# The Girvan – Newman Algorithm

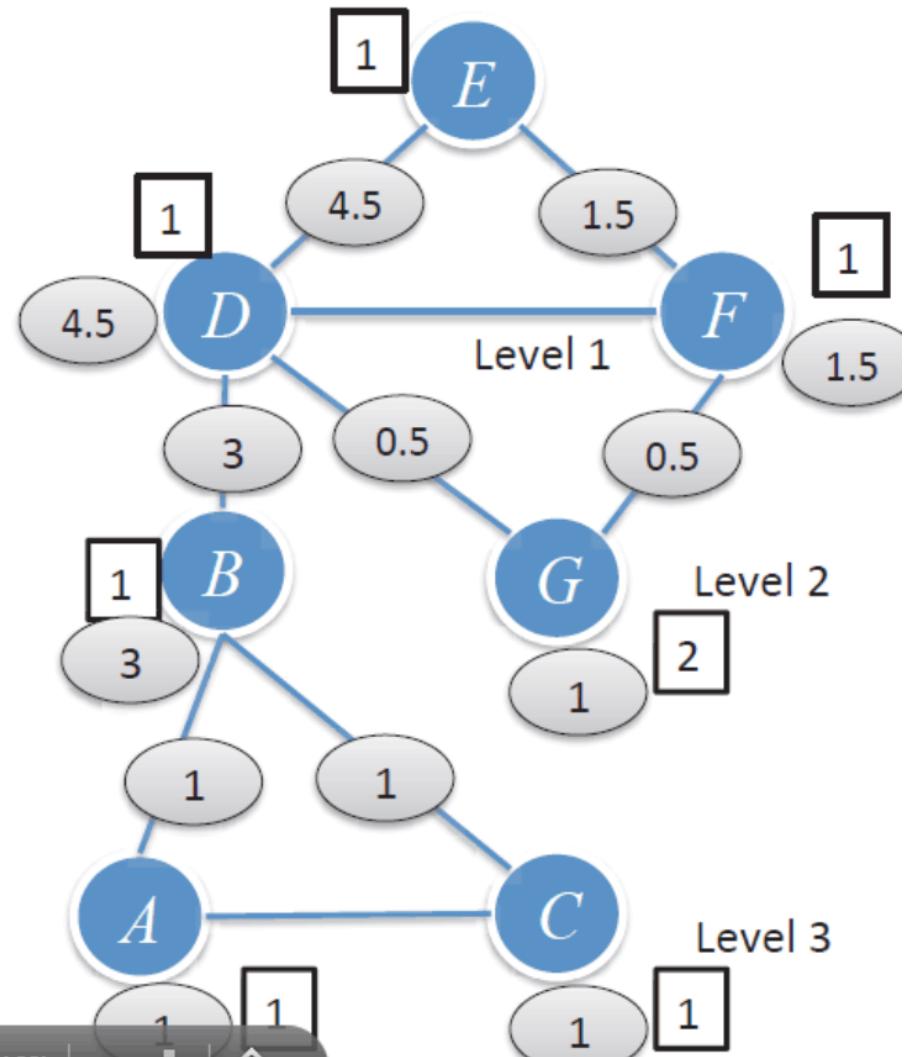
## Step 3 – credit sharing:

- Each leaf node gets credit 1
- Each non-leaf node gets  $1 + \text{sum}(\text{credits of the DAG edges to the level below})$
- Credit of DAG edges: Let  $Y_i$  ( $i=1, \dots, k$ ) be parents of  $Z$ ,  $p_i = \text{label}(Y_i)$   
$$\text{credit}(Y_i, Z) = \frac{\text{credit}(Z) \times p_i}{(p_1 + \dots + p_k)}$$

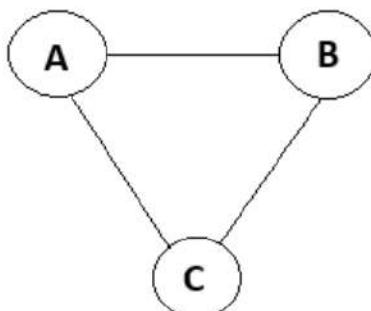
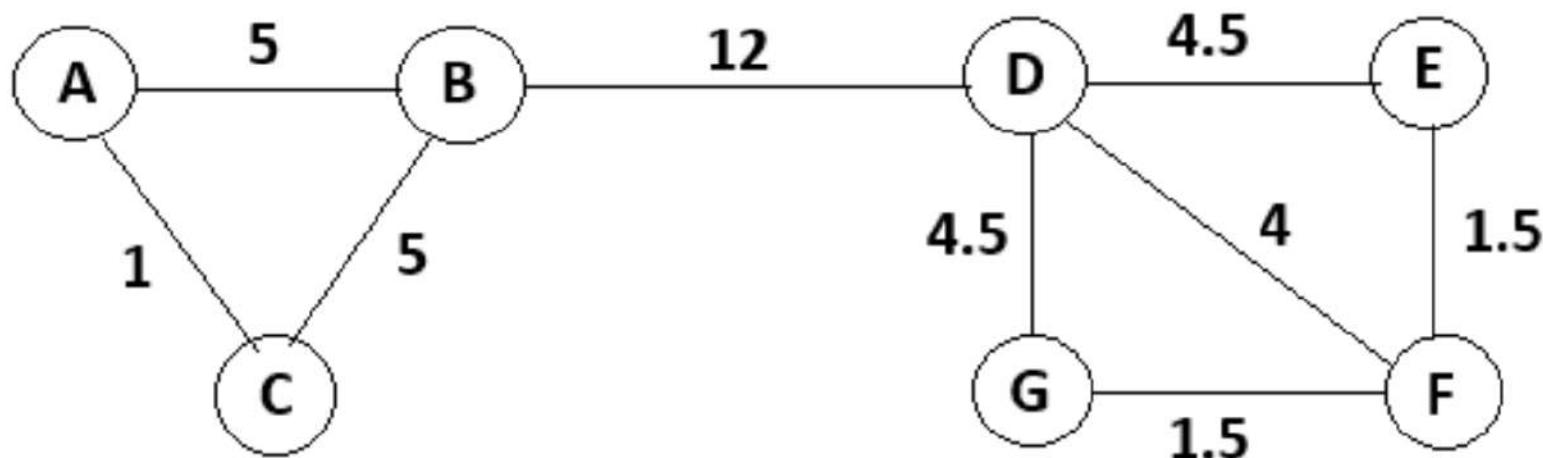
- Intuition: a DAG edge  $Y_iZ$  gets the share of credit of  $Z$  proportional to the #of shortest paths from  $X$  to  $Z$  going through  $Y_iZ$

Finally: Repeat Steps 1, 2 and 3 with each node as root. For each edge, betweenness = sum credits obtained in all iterations / 2

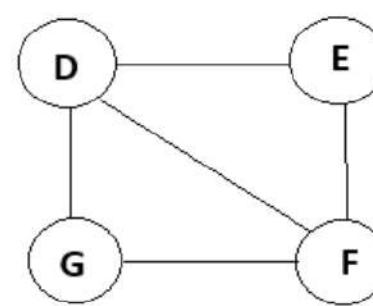
Calculate betweenness of edges



## Graph with betweenness value



C1



C2

Communities by clustering.

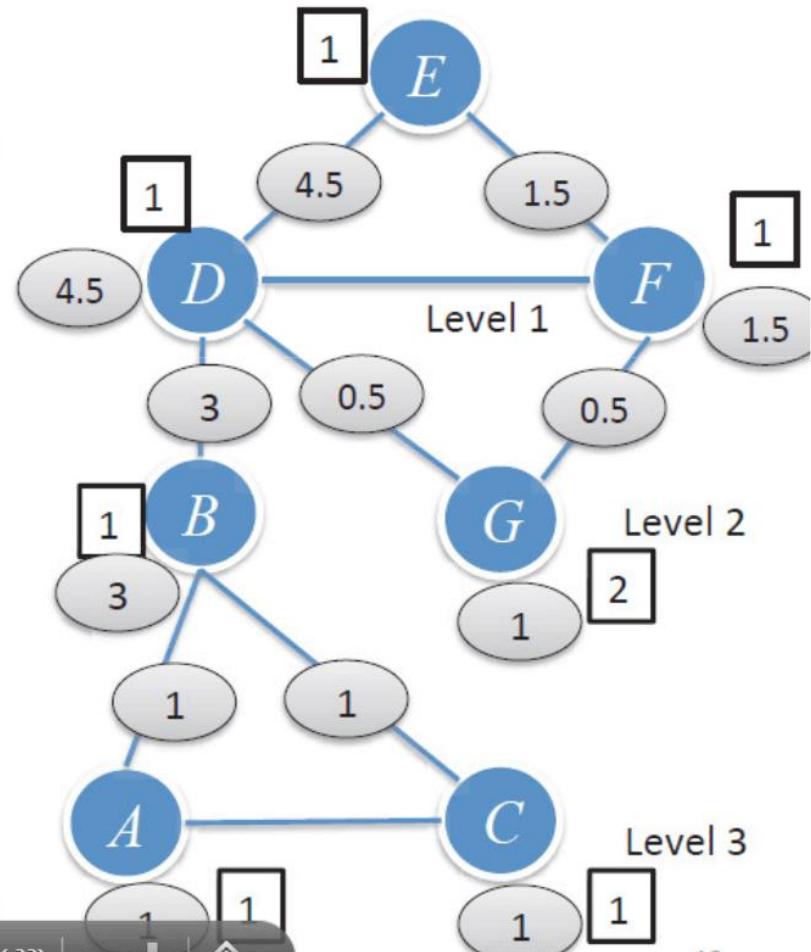
# The Girvan – Newman Algorithm

Step 3 – credit sharing:

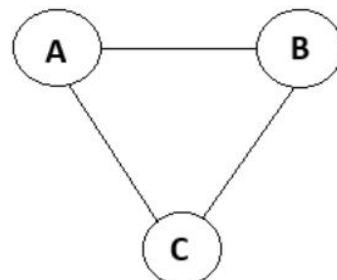
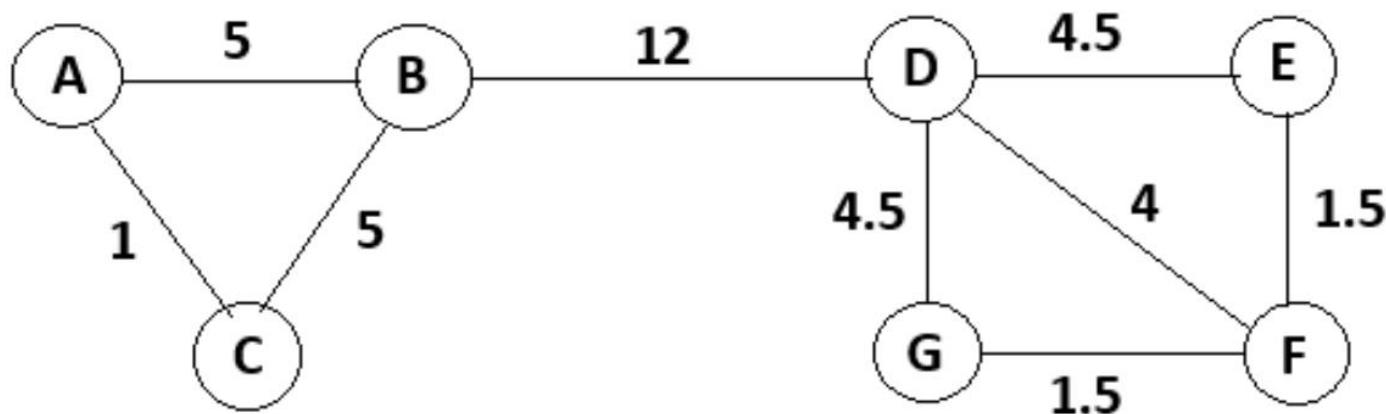
- Each leaf node gets credit 1
- Each non-leaf node gets  $1 + \text{sum}(\text{credits of the DAG edges to the level below})$
- Credit of DAG edges: Let  $Y_i$  ( $i=1, \dots, k$ ) be parents of  $Z$ ,  $p_i = \text{label}(Y_i)$   
$$\text{credit}(Y_i, Z) = \frac{\text{credit}(Z) \times p_i}{(p_1 + \dots + p_k)}$$
- Intuition: a DAG edge  $Y_iZ$  gets the share of credit of  $Z$  proportional to the #of shortest paths from  $X$  to  $Z$  going through  $Y_iZ$

Finally: Repeat Steps 1, 2 and 3 with each node as root. For each edge, betweenness = sum credits obtained in all iterations / 2

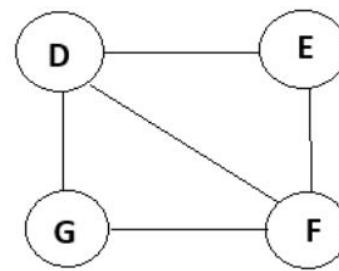
Calculate betweenness of edges



## Graph with betweenness value



C1



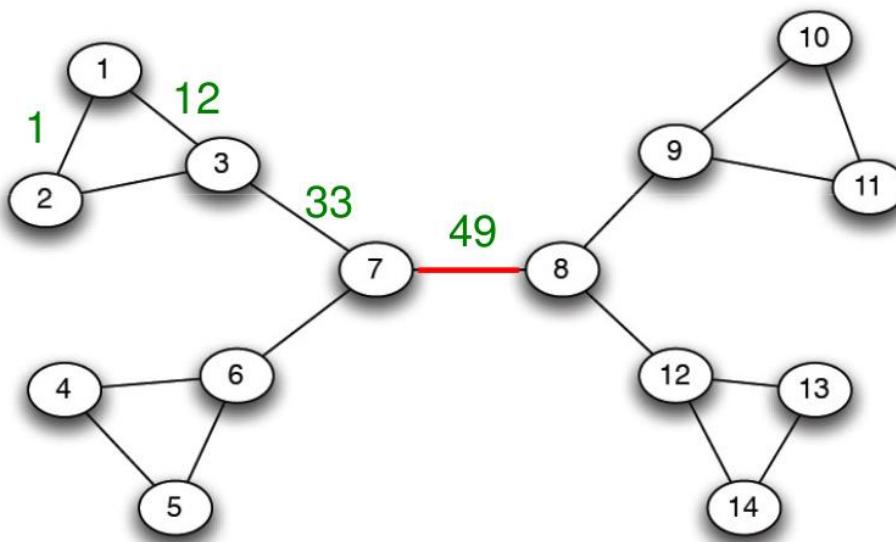
C2

Communities by clustering.

# Method 1: Girvan-Newman

- Divisive hierarchical clustering based on the notion of edge **betweenness**:  
**Number of shortest paths passing through the edge**
- **Girvan-Newman Algorithm:**
  - » Undirected unweighted networks
  - **Repeat until no edges are left:**
    - Calculate betweenness of edges
    - Remove edges with highest betweenness
  - Connected components are communities
  - Gives a hierarchical decomposition of the network

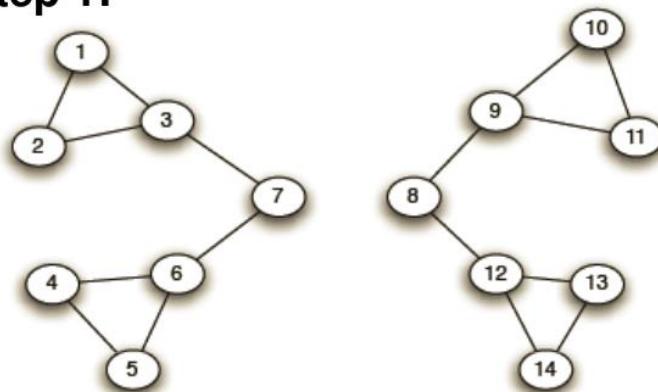
# Girvan-Newman: Example



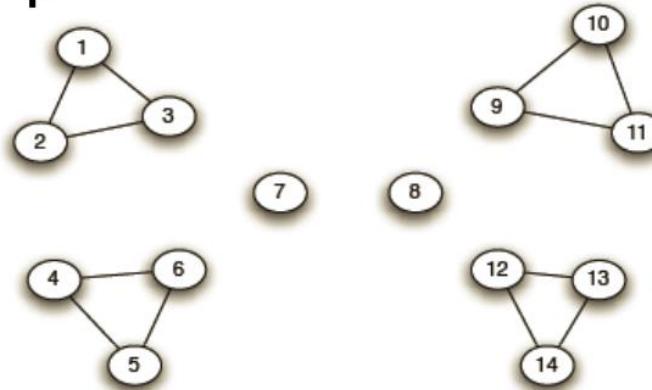
Need to re-compute  
betweenness at  
every step

# Girvan-Newman: Example

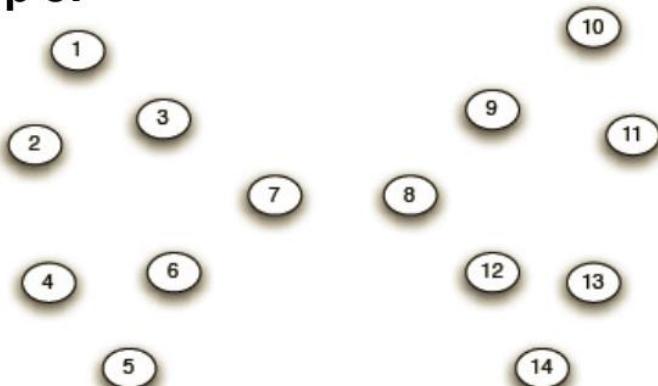
**Step 1:**



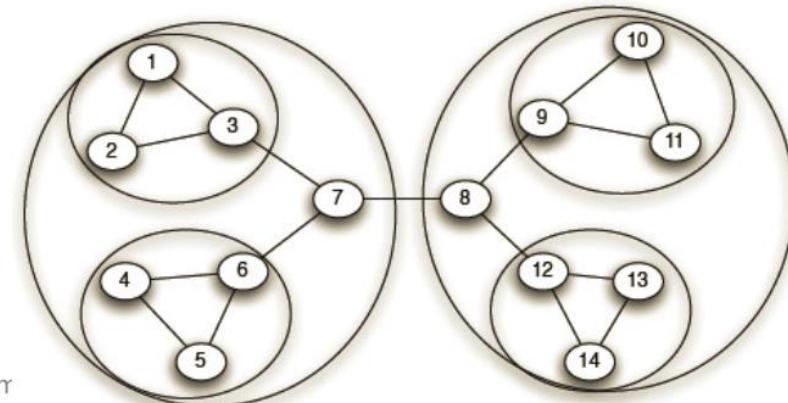
**Step 2:**



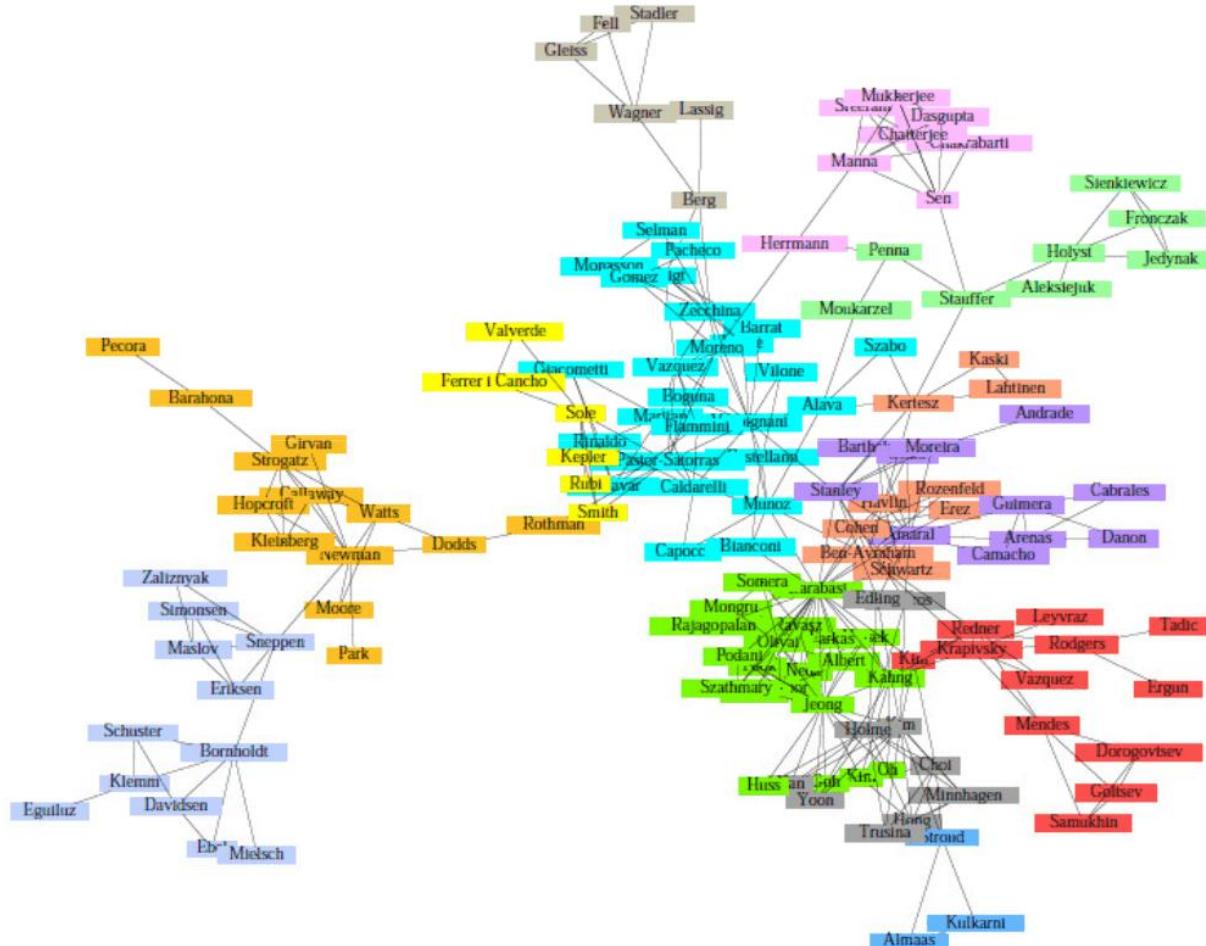
**Step 3:**



**Hierarchical network decomposition:**



# Girvan-Newman: Results

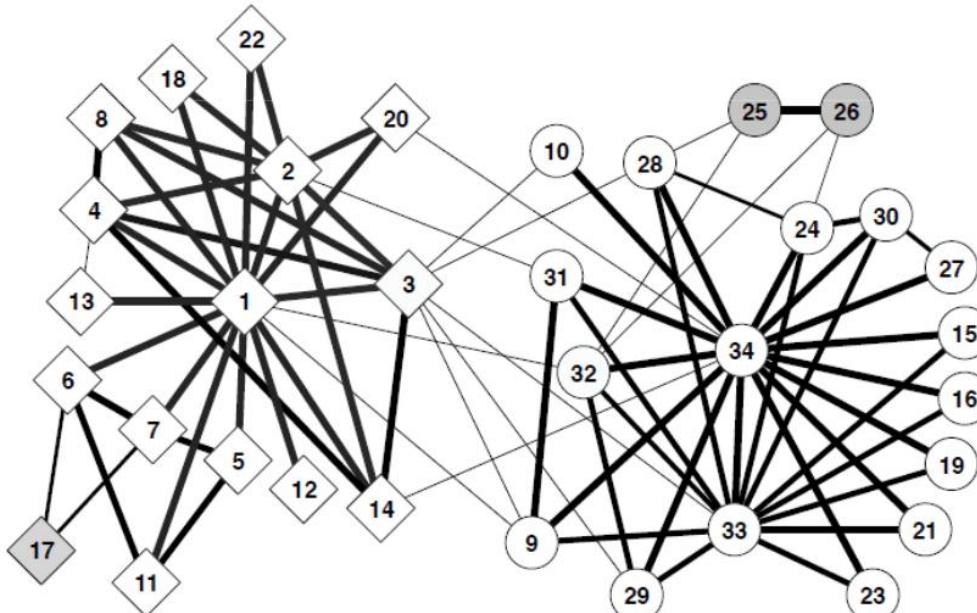


Communities in physics collaborations

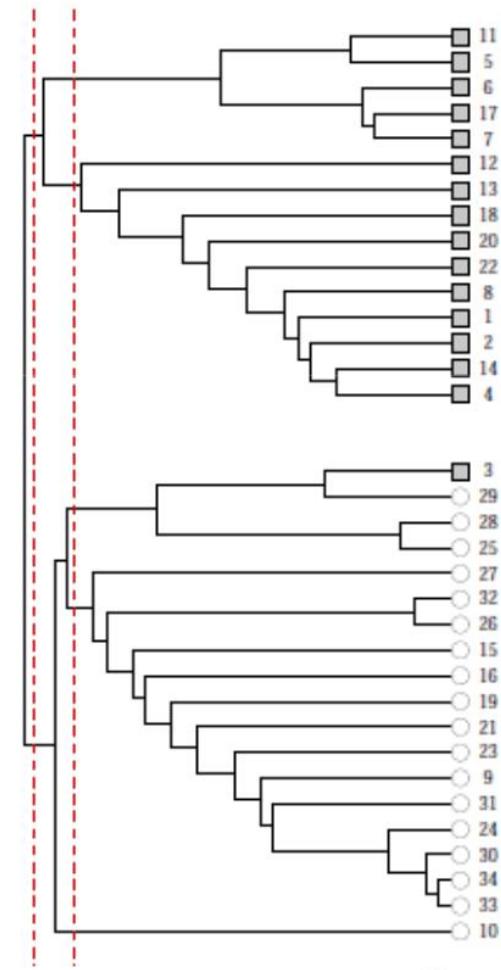
<http://www.mmds.org>

# Girvan-Newman: Results

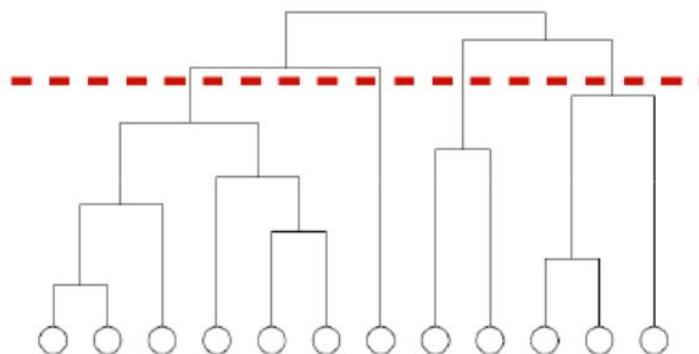
- Zachary's Karate club:  
Hierarchical decomposition



J. Leskovec, A. Rajaraman, J. Ullman:  
Mining of Massive Datasets,  
<http://www.mmds.org>



# WE NEED TO RESOLVE 2 QUESTIONS

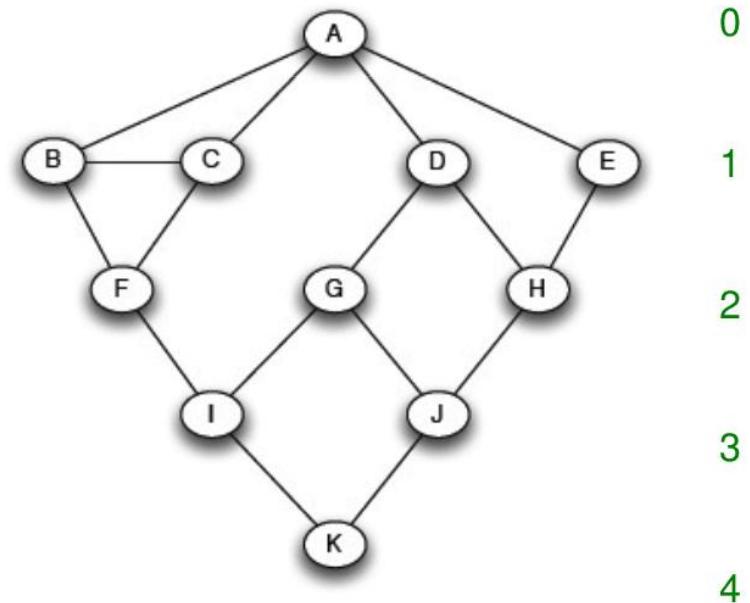
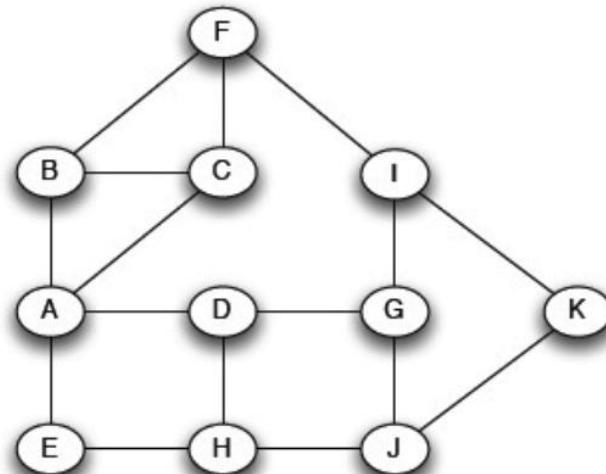


1. How to compute betweenness?
2. How to select the number of clusters?

J. Leskovec, A. Rajaraman, J. Ullman:  
Mining of Massive Datasets,  
<http://www.mmds.org>

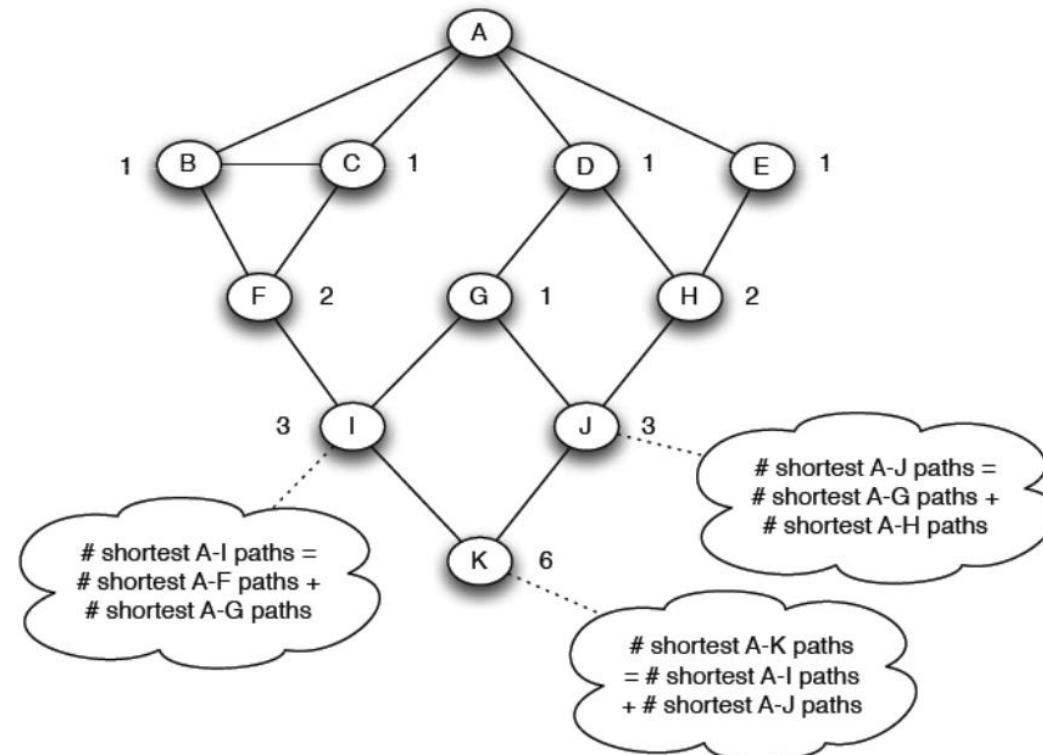
# How to Compute Betweenness?

- Want to compute betweenness of paths starting at node A
- Breath first search starting from A:



# How to Compute Betweenness?

- Count the number of shortest paths from  $A$  to all other nodes of the network:

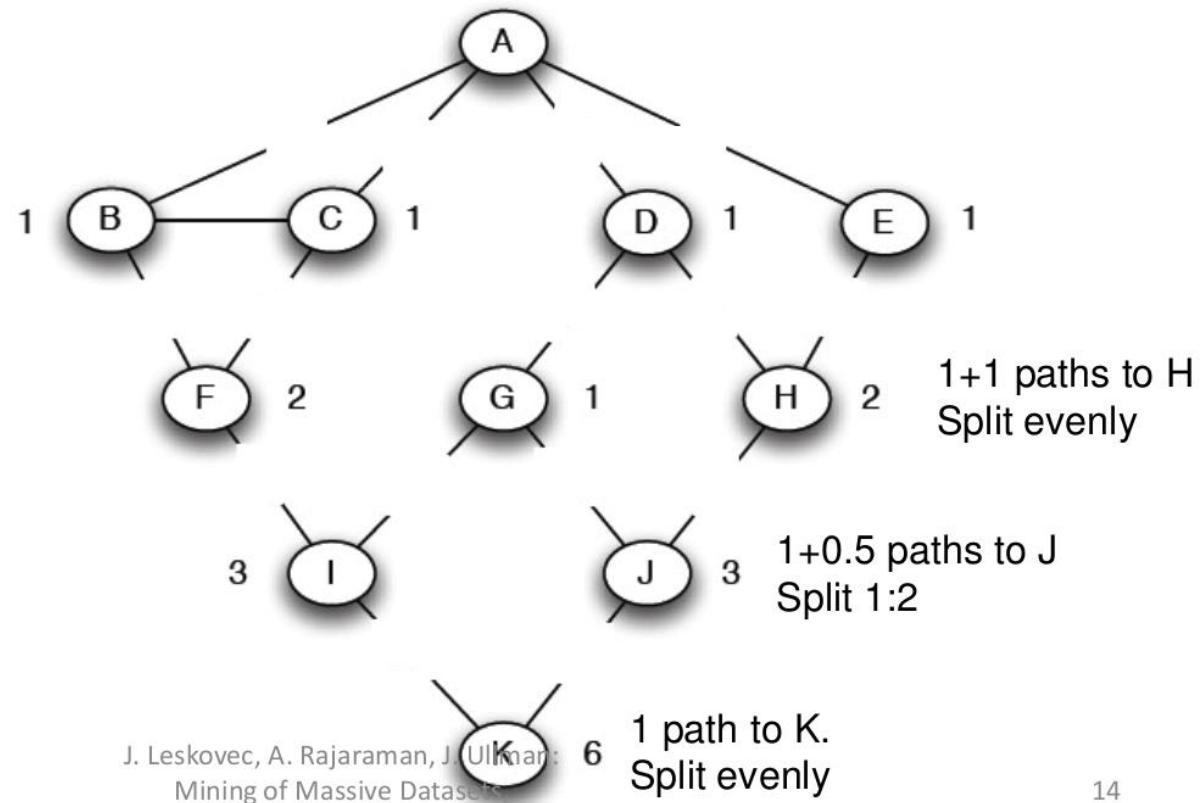


# How to Compute Betweenness?

- **Compute betweenness by working up the tree:** If there are multiple paths count them fractionally

The algorithm:

- Add edge flows:
  - node flow =  $1 + \sum_{\text{child edges}}$
  - split the flow up based on the parent value
- Repeat the BFS procedure for each starting node  $U$

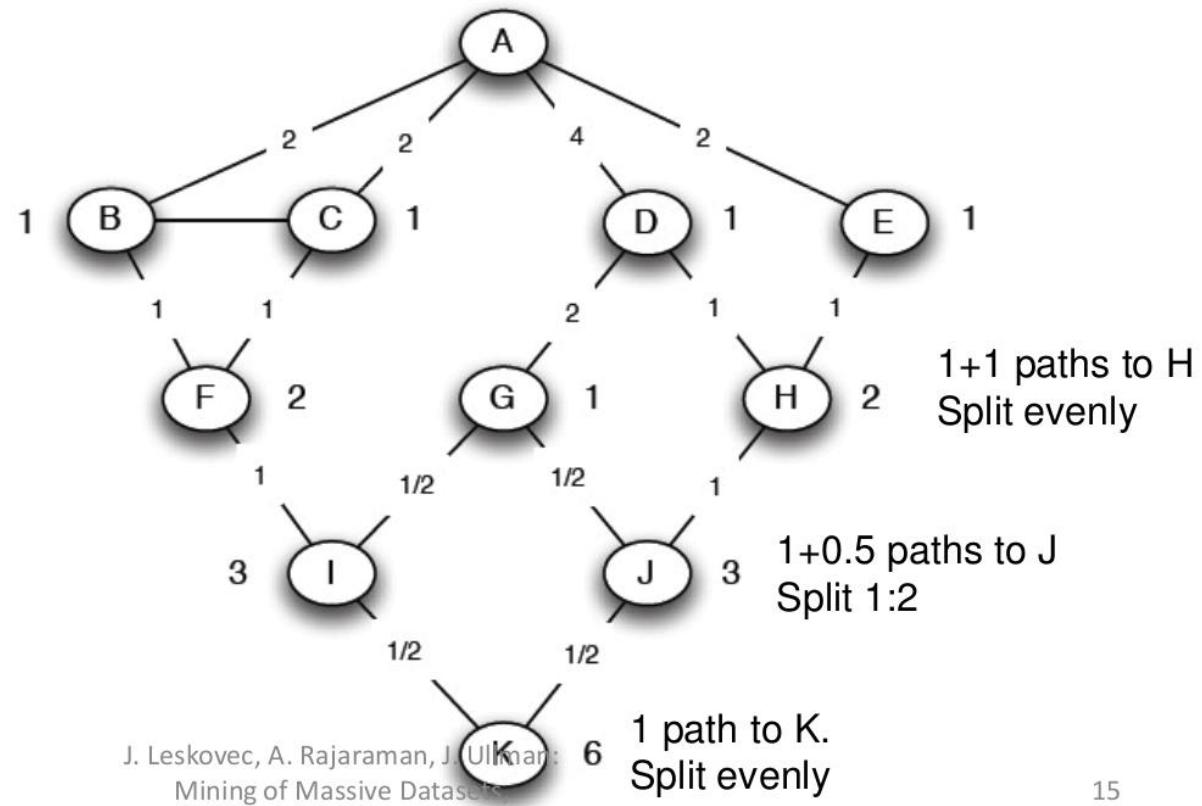


# How to Compute Betweenness?

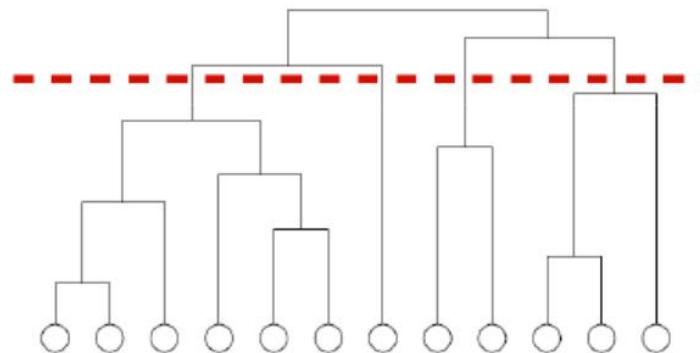
- **Compute betweenness by working up the tree:** If there are multiple paths count them fractionally

The algorithm:

- Add edge flows:
  - node flow =  $1 + \sum_{\text{child edges}}$
  - split the flow up based on the parent value
- Repeat the BFS procedure for each starting node  $U$



# WE NEED TO RESOLVE 2 QUESTIONS



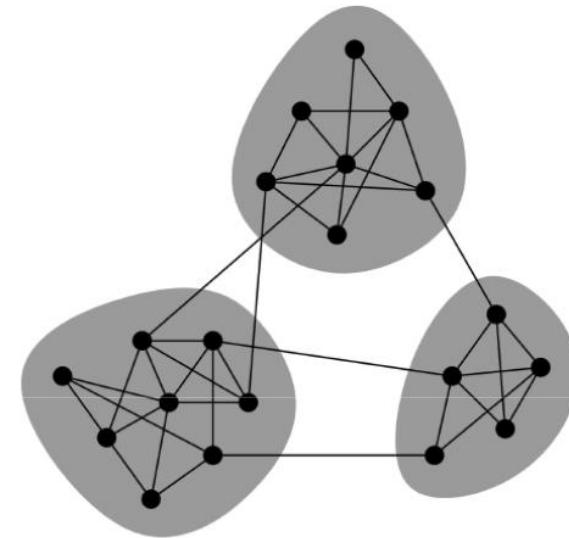
1. How to compute betweenness?
2. How to select the number of clusters?

J. Leskovec, A. Rajaraman, J. Ullman:  
Mining of Massive Datasets,  
<http://www.mmds.org>

# Network Communities

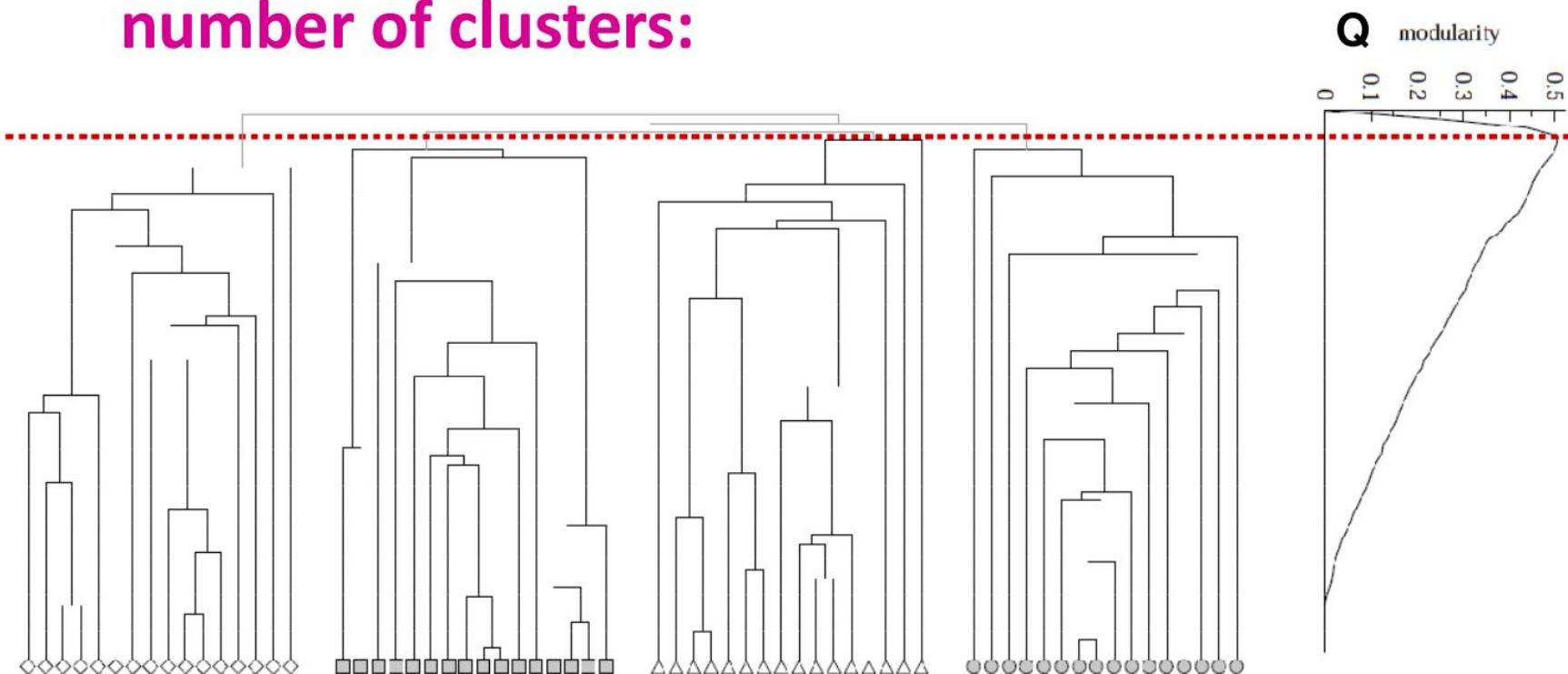
- **Communities:** sets of **tightly connected nodes**
- Define: **Modularity  $Q$** 
  - A measure of how well a network is partitioned into communities
  - Given a partitioning of the network into groups  $s \in S$ :

$$Q \propto \sum_{s \in S} [ (\text{\# edges within group } s) - (\text{expected \# edges within group } s) ]$$



# Modularity: Number of clusters

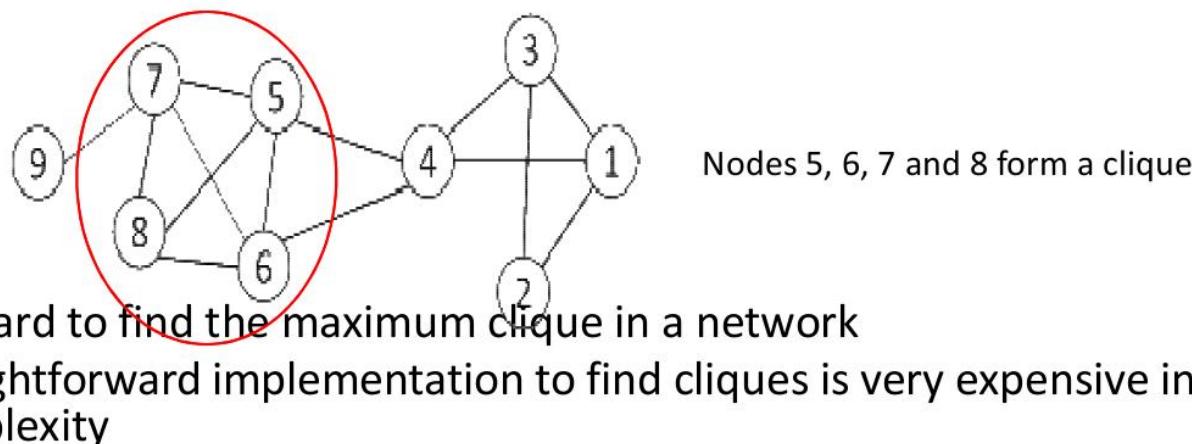
- Modularity is useful for selecting the number of clusters:



# **OVERLAPPING COMMUNITIES**

# Cliques

- **Clique**: a maximum complete subgraph in which all nodes are adjacent to each other

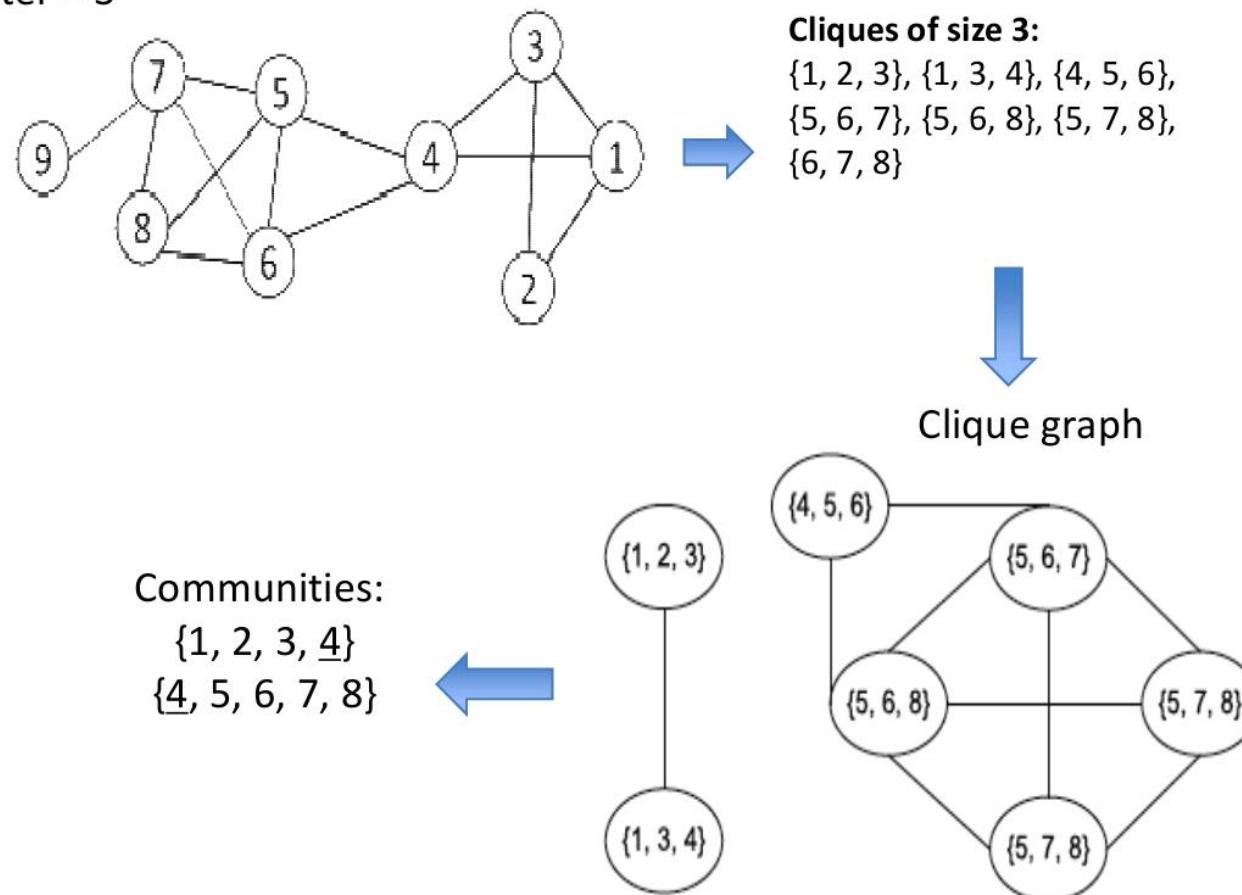


# Clique Percolation Method (CPM)

- Normally use cliques as **a core or a seed** to find larger communities
- Clique Percolation Method to find **overlapping** communities (diagram on next page)
  - **Input**
    - A parameter  $k$ , and a network
  - **Procedure**
    - Find out all cliques of size  $k$  in a given network
    - Construct a clique graph: two cliques are adjacent if they share  $k-1$  nodes
    - The nodes depicted in the labels of each connected components in the clique graph form a community

# CPM Example

Parameter = 3

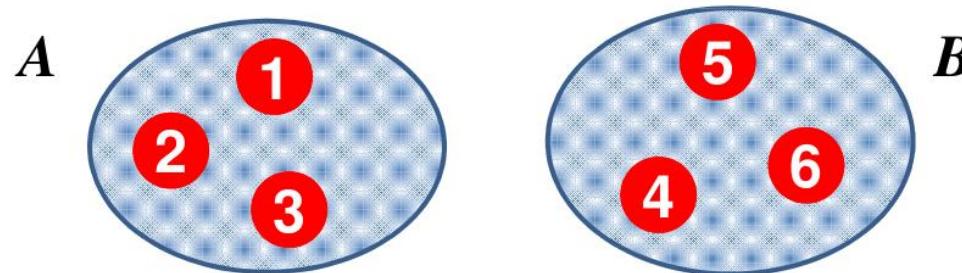
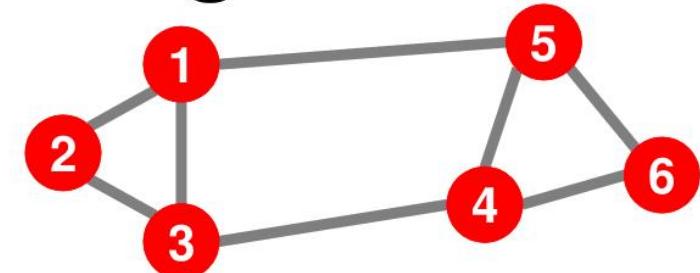


# Partitioning of Graphs

## Graph Spectral Clustering

# Graph Partitioning

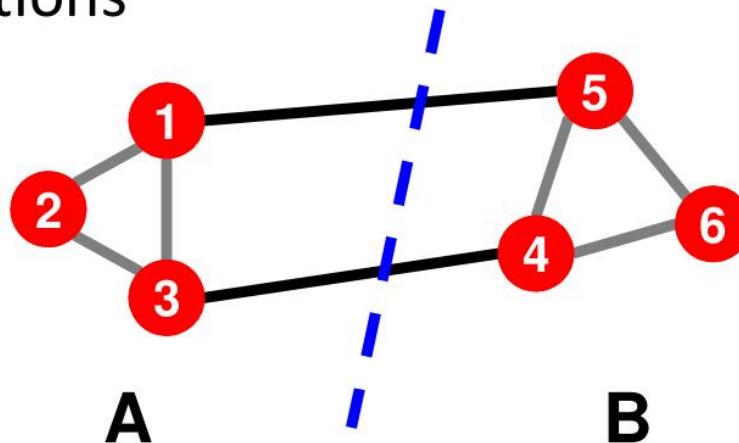
- Undirected graph  $G(V, E)$ :
- Bi-partitioning task:
  - Divide vertices into two disjoint groups  $A, B$



- Questions:
  - How can we define a “good” partition of  $G$ ?
  - How can we efficiently identify such a partition?

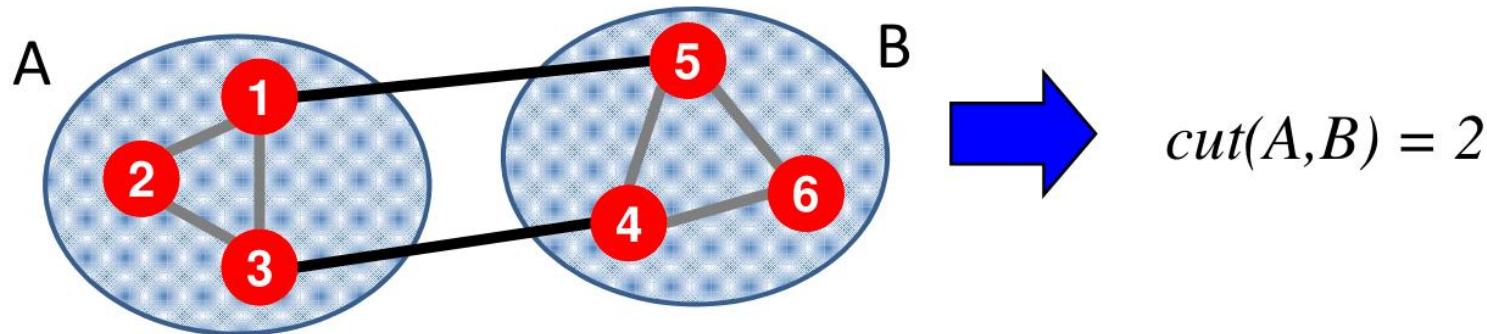
# Graph Partitioning

- **What makes a good partition?**
  - Maximize the number of within-group connections
  - Minimize the number of between-group connections



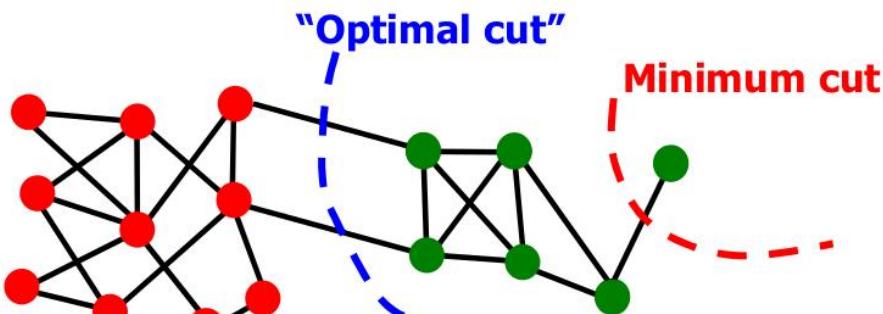
# Graph Cuts

- Express partitioning objectives as a function of the “edge cut” of the partition
- **Cut:** Set of edges with only one vertex in a group:  
$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$$



# Graph Cut Criterion

- Criterion: **Minimum-cut**
  - Minimize weight of connections between groups
- Degenerate case:



- Problem:
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity

# Graph Cut Criteria

- Criterion: **Normalized-cut** [Shi-Malik, '97]
  - Connectivity between groups relative to the density of each group

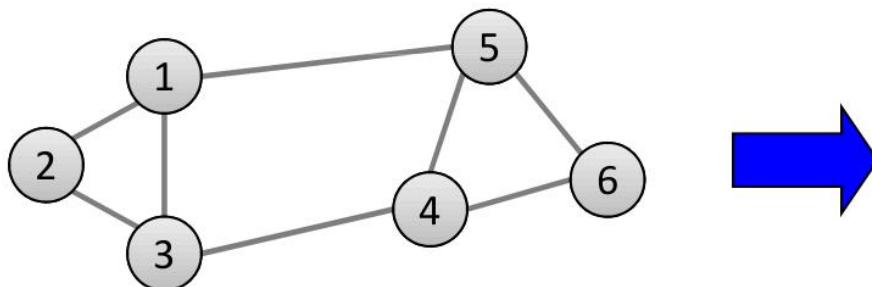


$\text{vol}(A)$ : total weight of the edges with at least one endpoint in  $A$ :  $\text{vol}(A) = \sum_{i \in A} k_i$

- Why use this criterion?
  - Produces more balanced partitions
- How do we efficiently find a good partition?
  - Problem: Computing optimal cut is NP-hard

# Matrix Representations

- **Adjacency matrix ( $A$ ):**
  - $n \times n$  matrix
  - $A = [a_{ij}]$ ,  $a_{ij} = 1$  if edge between node  $i$  and  $j$

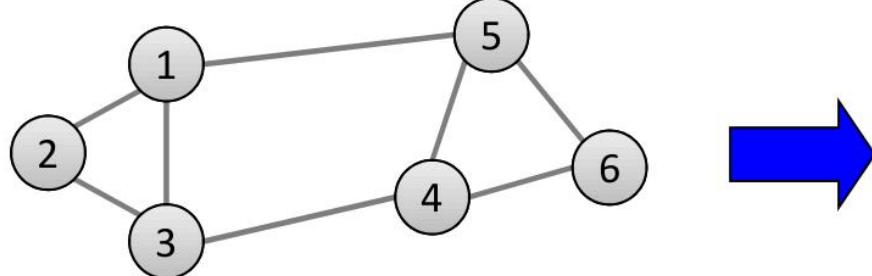


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- **Important properties:**
  - Symmetric matrix
  - Eigenvectors are real and orthogonal

# Matrix Representations

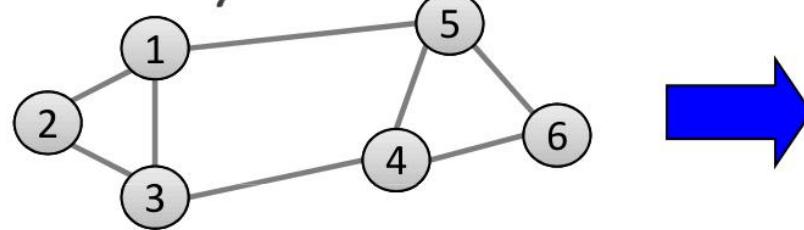
- **Degree matrix (D):**
  - $n \times n$  diagonal matrix
  - $D = [d_{ii}]$ ,  $d_{ii}$  = degree of node  $i$



	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

# Matrix Representations

- **Laplacian matrix (L):**
  - $n \times n$  symmetric matrix



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

\* Given real  $G$  on  $n$  nodes and  $m$  edges, construct rewired network  $G'$ :  
- Same degree distribution but random connections  
- Consider  $G'$  as a multigraph  
- The expected number of edges between nodes  $i$  and  $j$  of degrees  $K_i$  and  $K_j$  equals to:  $K_i \cdot \frac{K_j}{2m} = \frac{K_i K_j}{2m}$   
- The expected number of edges in (multigraph)  $G'$ :  
 $= \frac{1}{2} \sum_{i \in N} \sum_{j \in N} \frac{K_i K_j}{2m} = \frac{1}{2} \cdot 2m \sum_{i \in N} K_i (\sum_{j \in N} K_j) =$   
 $= \frac{1}{2m} 2m \cdot 2m = m$

- **What is trivial eigenpair?**
  - $x = (1, \dots, 1)$  then  $L \cdot x = \mathbf{0}$  and so  $\lambda = \lambda_1 = 0$
- **Important properties:**
  - **Eigenvalues** are non-negative real numbers
  - **Eigenvectors** are real and orthogonal

# Spectral Graph Partitioning

- $A$ : adjacency matrix of undirected  $\mathbf{G}$ 
  - $A_{ij} = 1$  if  $(i, j)$  is an edge, else  $0$
- $x$  is a vector in  $\Re^n$  with components  $(x_1, \dots, x_n)$ 
  - Think of it as a label/value of each node of  $G$
- **What is the meaning of  $A \cdot x$ ?**

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$
$$y_i = \sum_{j=1}^n A_{ij} x_j = \sum_{(i,j) \in E} x_j$$

- **Entry  $y_i$  is a sum of labels  $x_j$  of neighbors of  $i$**

# What is the meaning of $Ax$ ?

- **$j^{th}$  coordinate of  $A \cdot x$ :**

- Sum of the  $x$ -values of neighbors of  $j$
- Make this a new value at node  $j$

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$A \cdot x = \lambda \cdot x$

- **Spectral Graph Theory:**

- Analyze the “spectrum” of matrix representing  $G$
- **Spectrum:** Eigenvectors  $\mathbf{x}_i$  of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues  $\lambda_i$ :

$$\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$$

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

J. Leskovec, A. Rajaraman, J. Ullman:

Mining of Massive Datasets,  
<http://www.mmds.org>

# Example: d-regular graph

- Suppose all nodes in  $G$  have degree  $d$  and  $G$  is connected
- **What are some eigenvalues/vectors of  $G$ ?**

$A \cdot x = \lambda \cdot x$  What is  $\lambda$ ? What  $x$ ?

- Let's try:  $x = (1, 1, \dots, 1)$
- Then:  $A \cdot x = (d, d, \dots, d) = \lambda \cdot x$ . So:  $\lambda = d$
- We found eigenpair of  $G$ :  $x = (1, 1, \dots, 1)$ ,  $\lambda = d$

Remember the meaning of  $y = A \cdot x$ :

$$y_j = \sum_{i=1}^n A_{ij}x_i = \sum_{(j,i) \in E} x_i$$

# Facts about the Laplacian $L$

- (a) All eigenvalues are  $\geq 0$
- (b)  $x^T L x = \sum_{ij} L_{ij} x_i x_j \geq 0$  for every  $x$
- (c)  $L = N^T \cdot N$

- That is,  $L$  is positive semi-definite
- **Proof:**

- (c) $\Rightarrow$ (b):  $x^T L x = x^T N^T N x = (xN)^T (Nx) \geq 0$ 
  - As it is just the square of length of  $Nx$
- (b) $\Rightarrow$ (a): Let  $\lambda$  be an eigenvalue of  $L$ . Then by (b)  $x^T L x \geq 0$  so  $x^T L x = x^T \lambda x = \lambda x^T x \Rightarrow \lambda \geq 0$
- (a) $\Rightarrow$ (c): is also easy! Do it yourself.

# Eigen values and Eigen vectors

The second eigenvector has three positive and three negative components. It makes the unsurprising suggestion that one group should be  $\{1, 2, 3\}$ , the nodes with positive components, and the other group should be  $\{4, 5, 6\}$ .  $\square$

Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	-5	-1	-1	-1
	1	2	4	-2	1	0
	1	1	1	3	-1	1
	1	-1	-5	-1	1	1
	1	-2	4	-2	-1	0
	1	-1	1	3	1	-1

# Finding Overlapping Communities