

COMP4331: Introduction to Data Mining

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Why Data Mining?



We are experiencing an explosive growth of data

- 5 exabytes (10^{18} bytes) of data were created by human until 2003 (kilo, mega, giga, tera; peta; exa; zetta); today this amount of information is created in two days
- in 2012, digital world of data was expanded to 2.72 zettabytes
- it is predicted to double every two years
- IBM indicates that every day 2.5 exabytes of data is created

Major Sources of Abundant Data

- society: news, social networking (Facebook, YouTube)
- business: web, e-commerce, transactions, stocks
- science: sensors, bioinformatics, scientific simulations

Example

- Facebook has 955 million monthly active accounts using 70 languages, 140 billion photos uploaded, 125 billion friend connections, every day 30 billion pieces of content and 2.7 billion likes and comments have been posted
- Every minute, 48 hours of video are uploaded and every day, 4 billion views performed on YouTube
- 1 billion Tweets every 72 hours from more than 140 million active users on Twitter
- 571 new websites are created every minute of the day
- every day 10 billion text messages are sent
- by the year 2020, 50 billion devices will be connected to networks and the internet

Why Data Mining?

We are drowning in data, but starving for knowledge

- while data **size** and **complexity** rapidly increase, the number of data analysts remains relatively small

Example

Within the next decade, number of information will increase by 50 times; however number of information technology specialists who keep up with all that data will increase by 1.5 time

- traditional techniques are simply inapplicable
- we need to find efficient ways to **analyze** the vast quantities of raw data to extract **knowledge**

Some Motivating Examples

- Business: A book store wishes to make **recommendations** to customers based on other customers' previous purchases
- Science: A bioinformatics lab wishes to find DNA **similarities** among different organisms
- Society: Either a company (for marketing purposes) or a lab (for research purposes) wishes to identify the most **influential** users in a social network



What is Data Mining?

- extraction of **interesting** (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from huge amount of data
- exploration and analysis, by automatic or semi-automatic means, of large quantities of data in order to discover **meaningful** patterns

What is Data Mining?...

Is everything “Data Mining”?

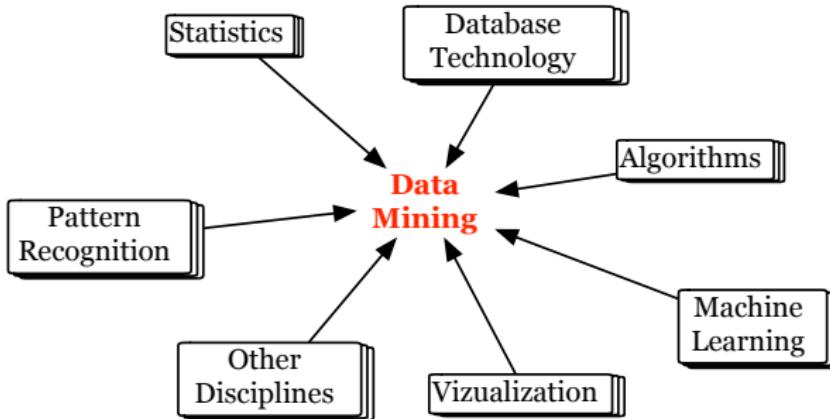
- simple search or query processing should not be confused with data mining

What is Not Data Mining?

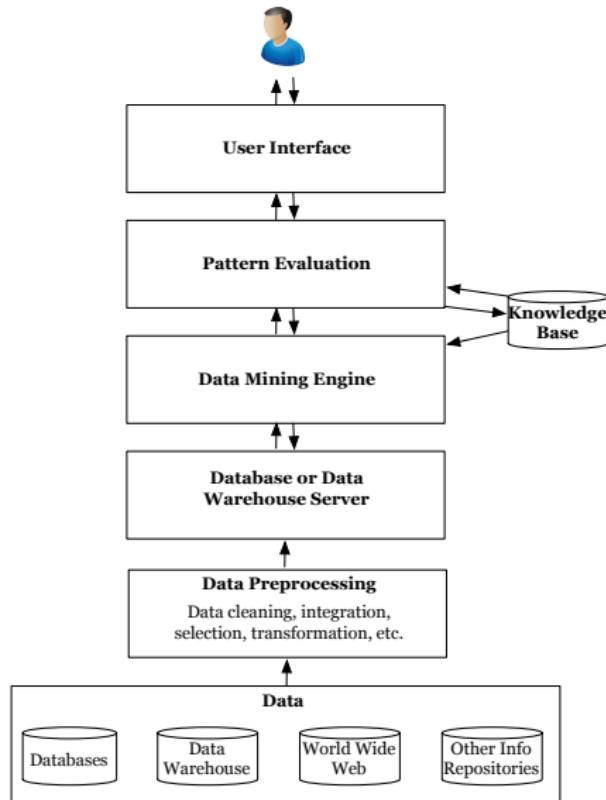
- look up phone number in a phone directory
- query a web search engine for information about “Amazon”

What is Data Mining?...

Data mining is a confluence of several disciplines



Data Mining Architecture



Architecture of a typical data mining system

On What Kind of Data?

Various data repositories

- relational data
- data warehouses
- transactional data
- graph data
- sequence data
- time series
- spatial data
- text & multimedia data

Relational Data

- a relational database consists of a set of **tables**, each of which consisting of a set of **attributes** (or columns or fields), and containing a large set of **records** (or tuples or rows)

Example

Employee

<u>EID</u>	<u>Name</u>	<u>Address</u>	<u>Position</u>	<u>Salary</u>
0023	A. Smith	122 Lake Ave., Chicago, IL	Manager	200,000\$
...

Branch

<u>BID</u>	<u>Name</u>	<u>Address</u>
005	City Square	356 Michigan Ave., Chicago, IL
...

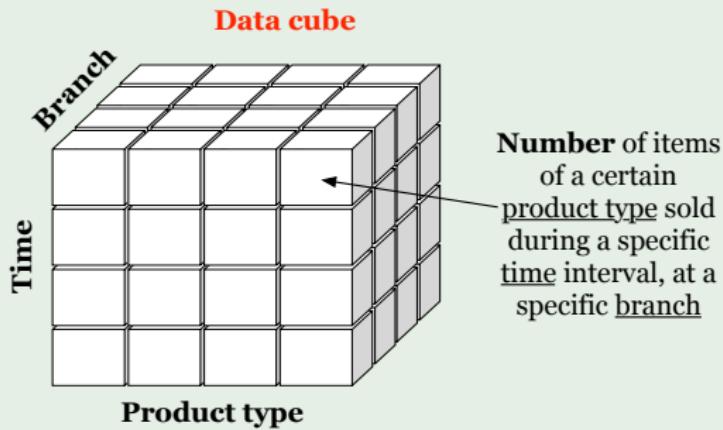
Works_At

<u>EID</u>	<u>BID</u>
0023	005
...	...

Data Warehouse

- a data repository of information collected from different sources stored under a unified scheme, and it usually resides at a single site
- the stored data provide information from a **historical perspective** and they are usually **summarized**
- the physical structure is typically a **multidimensional data cube**

Example



Transactional Data

- a special type of relational data, where every record is a **transaction** and involves a set of items

Example (a set of transactions)

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

Graph Data

- capture **relationships** among objects

Example (social network)



- **ordered** sequences of events with or without a concrete notion of time

Example

Genomic Sequence Data

```
GGTTCCGCCTTCAGCCCCGCGCC  
CGCAGGGCCCGCCCGCGCCGTC  
GAGAAGGGCCCGCCTGGCGGGCG  
GGGGGAGGCAGGGCCGCCGAGC  
CCAACCGAGTCCGACCAGGTGCC  
CCCTCTGCTCGGCCTAGACCTGA  
GCTCATTAGGCAGCGGACAG  
GCCAAGTAGAACACGCGAAGCGC  
TGGGCTGCCTGCTGCGACCAGGG
```

Time Series Data

- a special type of sequence data, where the values or events are obtained over repeated measurements of **time** (e.g., hourly, daily, weekly)

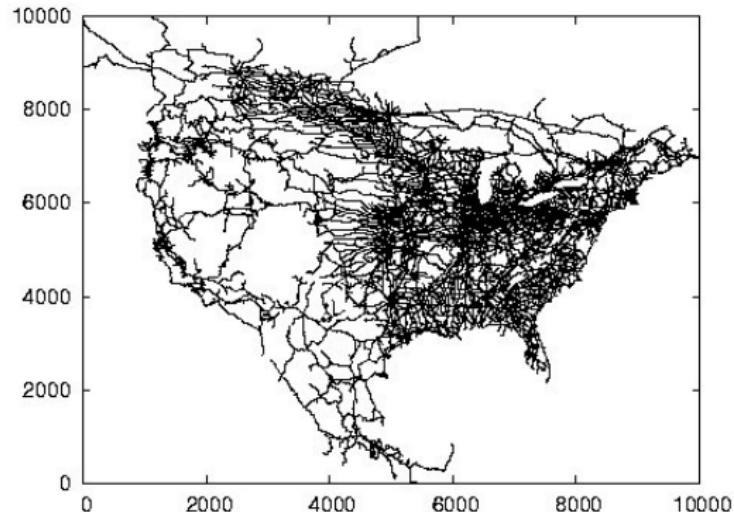
Example (Apple vs. Google Stock)



Spatial Data

- contain **geographical** attributes (such as spatial coordinates or areas)

Example (Road Network of North America (NA))



Text & Multimedia Data

- text databases contain **word descriptions** for objects
- multimedia databases store **image**, **audio**, and **video** data

Example

Document Term Vectors

	'Team'	'Coach'	'Timeout'
Doc#1	3	0	0
Doc#2	0	7	0

Image



Major data mining tasks

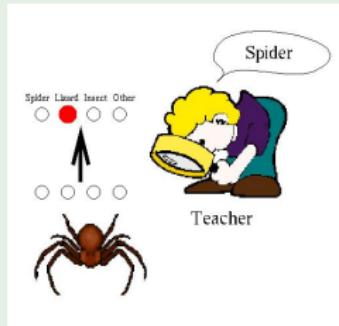
- Classification and regression
- Cluster analysis
- Association analysis

Classification and Regression

Classification

- we have a set of records called **training set**
- each record contains various attributes, among which there is a **categorical** (i.e., discrete) attribute referred to as **class**

Example



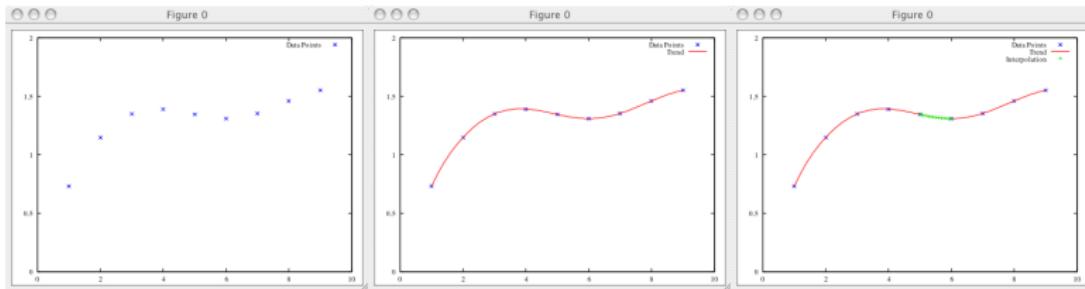
Regression

- classification predicts categorical attribute values; regression predicts **numerical** attribute values

Classification and Regression...

How to **predict** the value of a **new** (i.e., previously unseen) record?

- what about an old record?



- we explore the training set and devise a function called **model**
- the model takes as input a set of attributes values, and returns a value for the class attribute
- we then predict the class of the new record based on the model

Classification and Regression

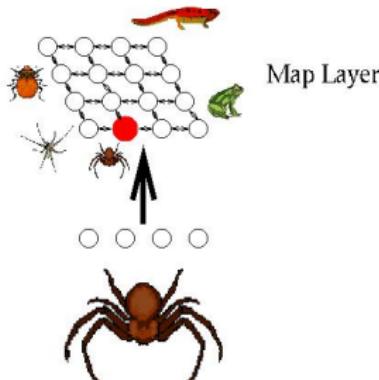
Example (Direct Marketing)

- suppose that we run an electronics consumer store
- we wish to reduce the cost of mailing by targeting only the set of consumers that are likely to buy a new product
- we collect a dataset of consumers that bought a similar product introduced before, as well as various demographic, lifestyle, and other information about them (**training set**)
- this {buy, don't buy} decision forms the **class** attribute
- we use the above information to devise a classifier **model**
- when reviewing the potential mail recipients, we **predict** if they are likely to buy the new product based on the model

Clustering

Given a set of objects, each having a set of attributes, and a **similarity measure** among them, find **clusters** (i.e., groups) such that

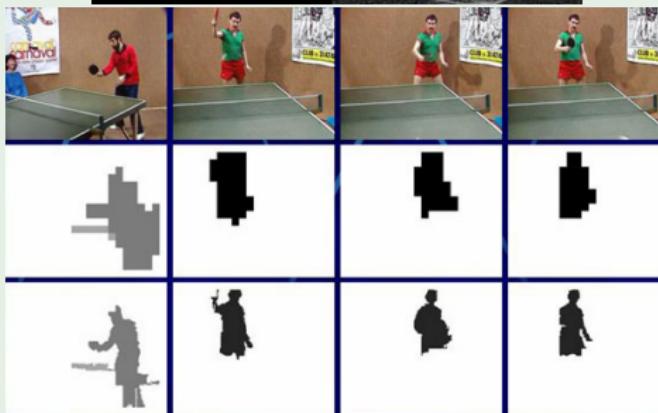
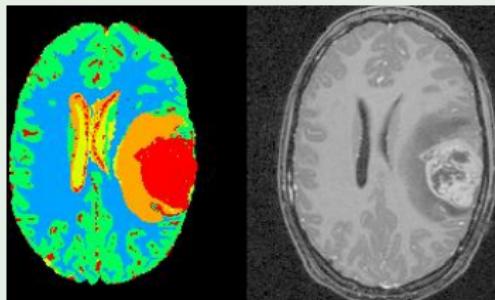
- objects in one cluster are more similar to one another
- objects in separate clusters are less similar to one another



- unlike classification, clustering analyzes objects **without** consulting a known class label

Clustering...

Example (image segmentation)



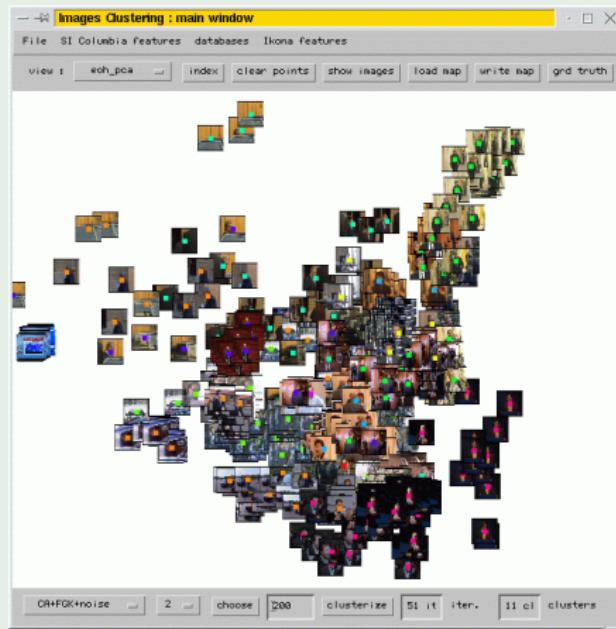
Clustering...

Example

Star Trek IV 0.024 Star Trek II 0.023 Star Trek VI 0.023 Star Trek III 0.021 The Fifth Element 0.018	Dr. Strangelove 0.029 A Clockwork Orange 0.020 Delicatessen 0.018 Cinema Paradiso 0.018 Brazil 0.017717
The Rock 0.553 Eraser 0.232 Independence Day (ID4) 0.089 Mission: Impossible 0.077 Trainspotting 0.021	The Piano 0.288 The Remains of the Day 0.077 In the Name of the Father 0.067 Forrest Gump 0.052 Shadowlands 0.047

Clustering...

Example (content-based image retrieval)



Outlier Detection

- outlier is an object that is “far away” from any cluster
- clustering can be used

Example (Fraud Detection)

- collect old transactions of a credit card holder
- cluster the transactions based on the location and/or the amount of money spent
- detect whether an incoming transaction is considerably dissimilar to all clusters

Association Analysis

Example

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

A supermarket wishes to find the products that most frequently co-occur in the customer transactions, in order to strategize effective promotions

Goal

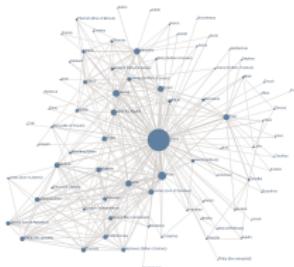
Given a **transactional database**, find the sets of objects that **frequently appear** within the **same** transactions

- also called **frequent pattern mining**

Emerging Data Mining Functionalities

Social network analysis

- mainly motivated by the rapid proliferation of social networking (Facebook, YouTube, etc.)



Example (possible tasks)

- discover social **communities** using similarity metrics
- model the **strength** of a social link based on the interaction between the users
- identify the most **influential** users in the social network (for **viral marketing** purposes)

Summary

- We explained what data mining is and why it is important
- We described the architecture of a typical data mining system
- We outlined various repositories used for data mining
- We overviewed the major data mining tasks

Data Preprocessing: Introduction

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Why Data Preprocessing?

real-world data is **dirty**

incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data (i.e., summaries)

- e.g., *occupation* = “ ”

Example

- “not applicable” data values upon collection (e.g., address)
- different considerations between the time when the data was collected and when it is analyzed
- human/hardware/software problems

Why Data Preprocessing?...

noisy: containing errors or outliers

- e.g., *salary* = “-10”

Example

- faulty data collection instruments
- human or computer error at data entry
- errors in data transmission

inconsistent: containing discrepancies in codes or names

- e.g., *age* = “42” and *birthday* = “03/07/1997”

Example

- different data sources

redundant: containing duplicate records or unnecessary attributes

Example

- integration (i.e., merging) of datasets from different sources

No quality data = no quality mining results!

- quality decisions must be based on quality data
- e.g., duplicate or missing data may cause incorrect or even misleading statistics

Why Data Preprocessing?

data size and complexity gravely affect the performance of the data mining tasks

- the larger the number of data objects to be analyzed, the more expensive the mining task
- the larger the number of attributes and the more complicated their value types, the more expensive the mining task



Data Preprocessing

Data preprocessing is a **preparation stage** before the actual data mining tasks are performed, which attempts to

- remove incomplete, noisy, inconsistent, and redundant data
- reduce the size and complexity of the data in order to refine the quality of the mining results and improve the performance of the mining tasks

Data preprocessing tasks

- data cleaning
- data integration
- data transformation
- data reduction

In this Part I

- data cleaning
- but before describing the above
 - types of attributes
 - descriptive data summarization

In part II

- data integration
- data transformation
- data reduction

Types of Attributes

A **dataset** is a collection of **objects**

- alternative names for “object” include “record”, “tuple”, “point”, “case”, “sample”, “entity”, and “instance”

An object is characterized by a set of **attributes**

Example

name, address, eye color, temperature

- alternative names for “attribute” include “variable”, “field”, “characteristic”, and “feature”

Attribute values are numbers or symbols assigned to an attribute

Example

student_name = ‘John’

- alternative names for “attribute value” include “value” and “feature-value”

Types of Attributes...

Categorical

- **nominal**: provide enough information to distinguish one object from another

Example

zip codes, employee ID numbers, eye color, gender

- **binary** attributes: assume only two values (e.g., yes/no, true/false, 0/1)
- **ordinal**: provide enough information to **order** objects

Example

grades, $\{good, better, best\}$

Numeric (continuous)

Examples

calendar dates, temperature in Celsius or Fahrenheit

Statistical Descriptions of Data

- gives the overall picture of the data
- involves
 - measuring the **central tendency**
 - measuring the **dispersion**
 - **graphical display** of descriptive summaries

Central Tendency

- the most common measure is the (arithmetic) **mean** (or average)
- let x_1, x_2, \dots, x_N be N observations
- their (sample) mean is calculated as:

$$\bar{x} = \frac{\sum_{i=1}^N x_i}{N}$$

- sometimes each value x_i is associated with a weight that signifies its importance. In this case, the **weighted mean** is:

$$\bar{x} = \frac{\sum_{i=1}^N w_i x_i}{\sum_{i=1}^N w_i}$$

The mean is sensitive to extreme values

Central Tendency...



Example (Scoring for individual diving events)

- panel of seven judges
- the two highest and lowest scores of the panel are thrown out
- the rest of the scores are added together and multiplied by a difficulty rating
- multiplied by 0.6 (for easy comparison with other events)

Trimmed mean: disregards the low and high extremes

Example

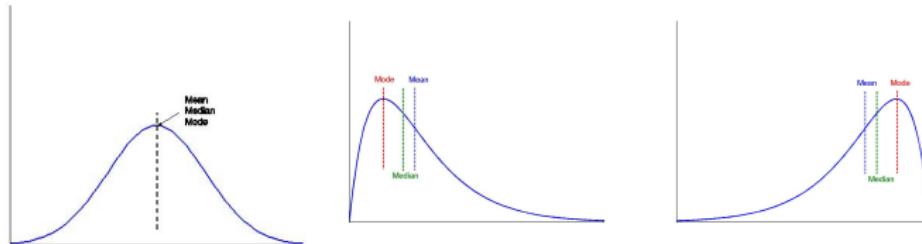
- data: 0,4,5,6,7,7,8,10,11,18
- 10% trimmed mean omits 0 and 18
- trimmed mean: $\frac{4+5+6+7+7+8+10+11}{8} = 7.25$

Central Tendency...

- a measure that is not sensitive to extreme values is the **median**, which represents the middle value of an ordered set of observations

Example

- $\text{median}(1, 5, 2, 8, 7) = 5$
- $\text{median}(1, 6, 2, 8, 7, 2) = (2 + 6)/2 = 4$
- **mode**: the value that occurs most frequently in the set
- **midrange**: average of the largest and smallest values in the data



Dispersion

- Let x_1, x_2, \dots, x_N be a set of (numerical) observations.
- range**: difference between the largest and smallest value
- k th percentile**: value x_i with the property that k percent of the data are smaller than x_i (**what percentile is the median?**)
- quartiles**: 25th percentile (denoted by Q_1), 50th percentile, and 75th percentile (denoted by Q_3)
- interquartile range**:

$$IQR = Q_3 - Q_1$$

- five-number summary**: consists of *minimum*, Q_1 , *median*, Q_3 , *maximum* (in this order)
 - gives a good impression of the center, spread, shape and distribution of the data

Dispersion

- **variance** $\text{var}(X) = E[(X - \mu)^2]$
- given a set of observations x_1, x_2, \dots, x_N :

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 = \frac{1}{N} \left[\sum_{i=1}^N x_i^2 - \frac{1}{N} \left(\sum_{i=1}^N x_i \right)^2 \right]$$

- **standard deviation** σ : square root of variance σ^2
 - σ indicates the spread of the values around the mean
 - $\sigma = 0$ when there is no spread, i.e., when all observations have the same value. Otherwise, $\sigma > 0$

Dispersion

- **variance** $\text{var}(X) = E[(X - \mu)^2]$
- given a set of observations x_1, x_2, \dots, x_N :

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 = \frac{1}{N} \left[\sum_{i=1}^N x_i^2 - \frac{1}{N} \left(\sum_{i=1}^N x_i \right)^2 \right]$$

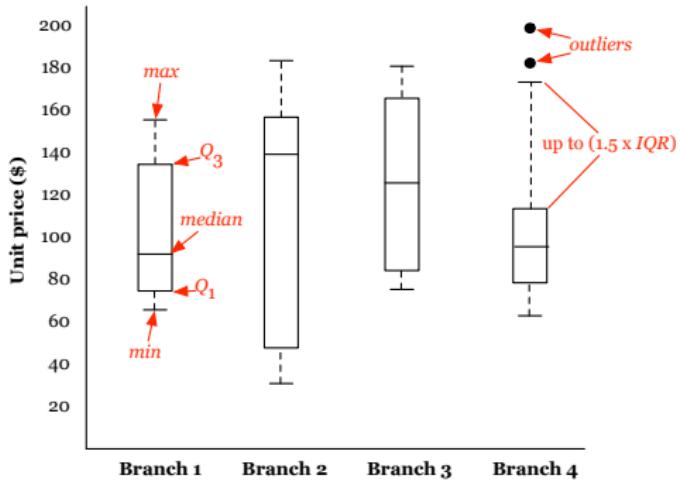
- **standard deviation** σ : square root of variance σ^2
 - σ indicates the spread of the values around the mean
 - $\sigma = 0$ when there is no spread, i.e., when all observations have the same value. Otherwise, $\sigma > 0$

Graphic Display

- usually useful to provide graphic displays of the data, in order to get some first impression of their characteristics
- examples of graphic displays include
 - boxplots
 - histograms
 - scatter plots

Boxplot

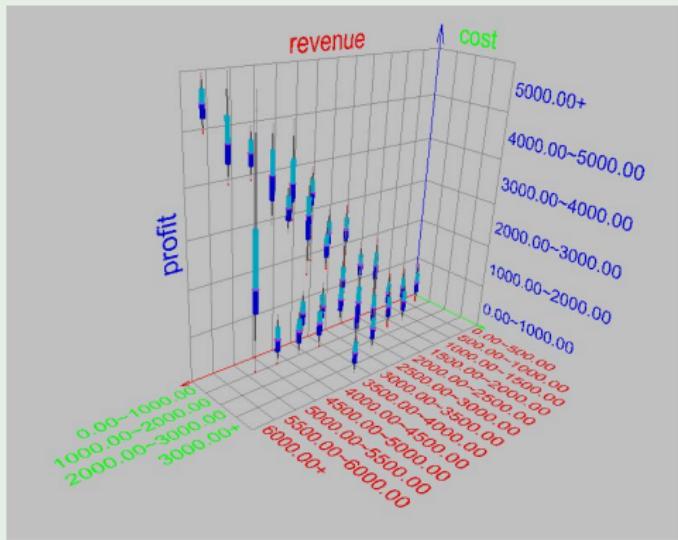
- boxplots incorporate the **five-number summary**



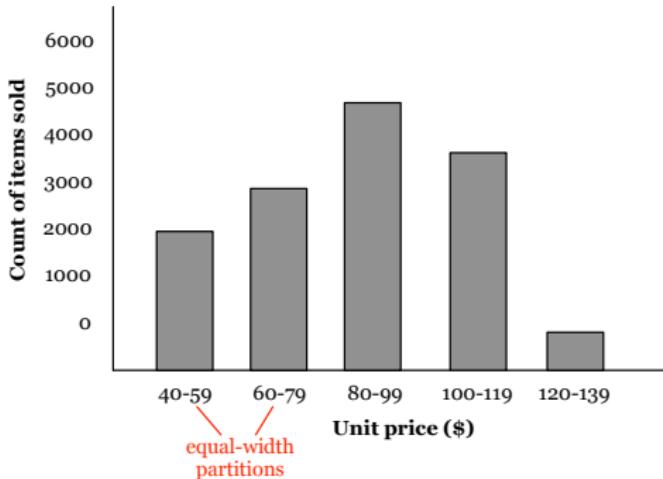
- the ends of the box are at the **first and third quartiles**
 - height of the box is IQR
- **median** is marked by a line within the box
- **outlier**: usually, a value higher/lower than $1.5 \times \text{IQR}$
- **whiskers**: two lines outside the box extended to **minimum** and **maximum**

Boxplot...

Example (3-D boxplot)



Histogram

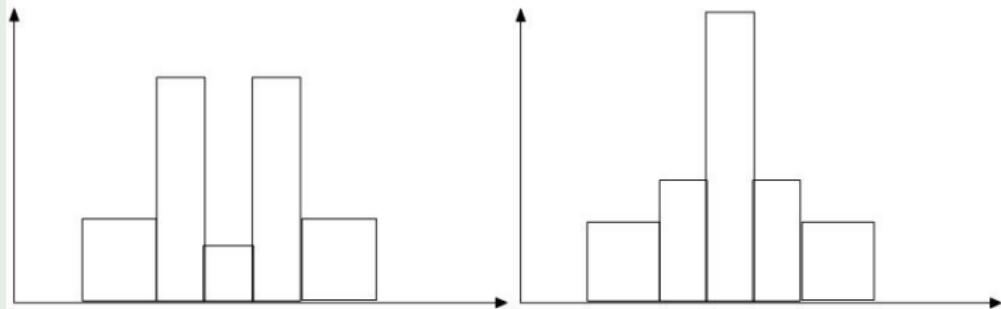


- divide data into buckets and store average (sum) for each bucket

Histogram...

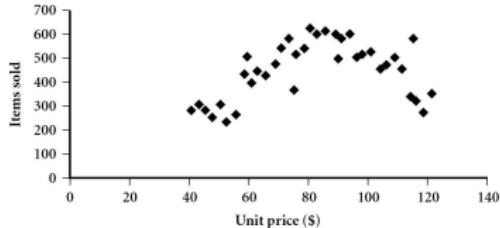
Two histograms may have the same boxplot representation (i.e., the same values for: min, Q1, median, Q3, max), do they have the same data distributions?

Example

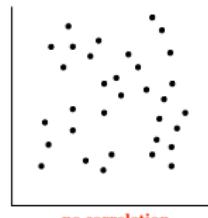
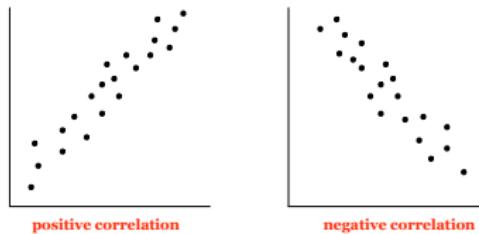


Scatter Plot

- determine whether there appears to be a relationship, pattern, or trend between **two numerical attributes**

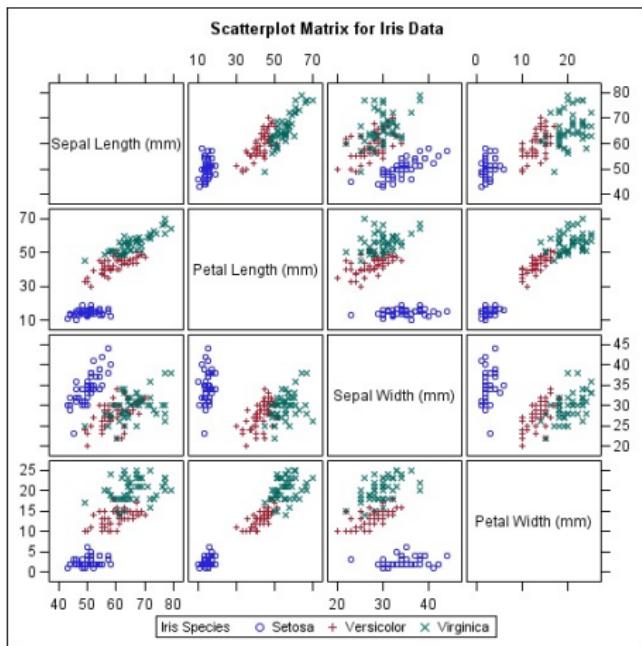


- each pair of values is treated as a pair of coordinates



Example: Iris Flower Data Set

- consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor)
- 4 features were measured from each sample
 - length and width of the sepals and petals



Data Preprocessing: Data Cleaning

Data cleaning attempts to

- fill in missing values
 - e.g., Occupation = “”
- smooth out noise, outliers
 - e.g., Salary = “-10”
- correct inconsistencies in the data
 - e.g., Age = “42”, Birthday = “03/07/2010”
 - e.g., discrepancy between duplicate records

Data Cleaning: Missing Values

What to do with **missing values?**

- ignore the record
 - usually done when class label is missing (when doing classification)
 - ignoring the tuple → cannot make use of the remaining attributes' values in the tuple
- fill in the missing value **manually**
 - tedious + infeasible?
- use a **global constant** to fill in the missing value
 - e.g., "unknown" (a new class?!)
- use the **attribute mean** to fill in the value
- use the attribute mean of **a certain class** (to which the record belongs)
- use the **most probable value** to fill in the missing value
 - extra tools may be needed to compute the most probable value

How to smooth out noise?

- remove **outliers** found by graphic display (explained in previous slides)
- **binning** (explained in the next slide)
- **regression**
 - smooth by fitting the data into regression functions
 - explained in later lectures
- **clustering**
 - detect and remove outliers
 - explained in later lectures

Binning

离散化

- **smooths** a sorted value by consulting its “neighborhood” (i.e., the values around it)
- e.g., smoothing by bin means/medians

Example

sorted data for *price* in \$: 4, 8, 15, 21, 21, 24, 25, 28, 34

- **partition into equal-sized bins**
Bin 1: 4, 8, 15 Bin 2: 21, 21, 24 Bin 3: 25, 28, 34
- **smoothing by bin means**
Bin 1: 9, 9, 9 Bin 2: 22, 22, 22 Bin 3: 29, 29, 29

Inconsistencies

How to correct inconsistencies?

- some cases are easy to detect (or even fix), provided that we possess some **domain knowledge**, also called **metadata** (i.e., data about the data)

Example

A person's height should not be negative

- other cases are much trickier, in which case we may need to consult an **external source of information**

Example

check the customer's address in a reimbursement form against the customer database of the insurance company

Summary

- We provided an introduction in data preprocessing
- We covered the first preprocessing technique, namely data cleaning

Data Preprocessing: Data Integration

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Data Integration

Data integration combines data from multiple sources into a coherent data store



What should we consider during data integration?

- entity identification problem
- data value conflicts
- data redundancy

Entity Identification Problem

Do two objects from different data sources refer to the same entity?

Example

Is the record that has $customer_id = 234$ (from one source) equivalent to that where $cust_num = 234$ (from the other source)?

Metadata can help

元资料：描述其他资料的资料

- e.g., for each attribute, look at the name, meaning, data type, range of values permitted, etc

Data Value Conflicts

Example

For the **same entity**, **attribute values** from different sources may differ

- e.g., *weight* measured in kilograms or pounds

Example

Attribute *total_sales* in one database may refer to the total sales of a company branch, whereas in another it may refer to the total sales for all branches in a specific region

once again, **metadata** may help

- e.g., What are the acceptable values for each attribute? What is the range of values? What is the standard deviation?

Data Redundancy

record redundancy

- there are two or more identical tuples for a unique entity

attribute redundancy

- one attribute may be “derived” from another attribute, or a set of attributes

Example

annual_income may be derived by *annual_revenue* and *annual_expenses*

- may be able to be detected by measuring how related two attributes are

Numerical Attributes: Correlation Coefficient

Given N tuples, are **numerical** attributes A and B correlated?

Let

- a_i, b_i : values of attribute A and B for the i th tuple
- \bar{A}, \bar{B} : respective means
- σ_A, σ_B : respective standard deviations

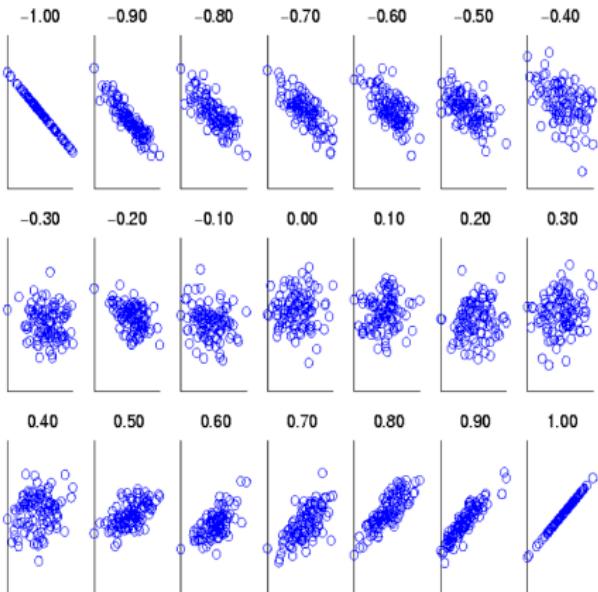
The correlation coefficient is given by

$$r_{A,B} = \frac{\frac{1}{N} \sum_{i=1}^N (a_i - \bar{A})(b_i - \bar{B})}{\sigma_A \sigma_B} = \frac{\sum_{i=1}^N a_i b_i - N \bar{A} \bar{B}}{N \sigma_A \sigma_B}$$

Correlation Coefficient...

$$-1 \leq r_{A,B} \leq +1$$

- $r_{A,B} > 0$
 - A and B are positively correlated
- $r_{A,B} < 0$
 - A and B are negatively correlated
- $r_{A,B} = 0$
 - A and B are uncorrelated



Categorical Attributes: χ^2 test

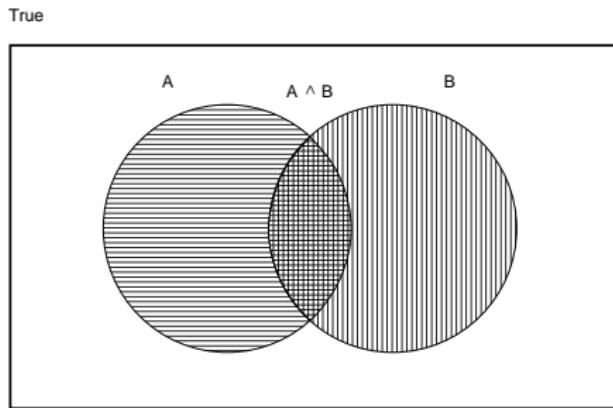
- A and B be two **categorical** attributes
- A has c distinct values, B has r distinct values

are A and B **independent?**

Revision: Axioms for Probability

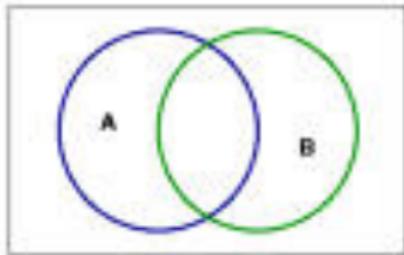
- All probabilities are between 0 and 1: $0 \leq P(A) \leq 1$
- Necessarily true propositions have probability 1: $P(\text{True}) = 1$
- Necessarily false propositions have probability 0: $P(\text{False}) = 0$
- The probability of a disjunction:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$



Conditional Probability

- let A and B be two events such that $P(A) > 0$
- $P(B|A)$: probability of B given that A has occurred



$$P(B|A) = \frac{P(A \cap B)}{P(A)}, \quad P(A \cap B) = P(A)P(B|A)$$

- probability that both A and B occur is equal to the probability that A occurs times the probability that B occurs given that A has occurred

Independence

- two random variables X and Y are **independent** if

$$P(X|Y) = P(X), \text{ or } P(Y|X) = P(Y)$$

- knowledge about X contains **no** information about Y
- equivalently, $P(X, Y) = P(X)P(Y)$

Example

- X : result of tossing a fair coin for the first time; Y : result of second tossing of the same coin
- X : result of US election; Y : your grades in this course

Question: Are these independent?

X : midterm exam grade; Y : final exam grade

χ^2 Test

- Let the distinct values of A be $\{a_1, a_2\} = \{\text{male}, \text{female}\}$
- Let the distinct values of B be $\{b_1, b_2\} = \{\text{fiction}, \text{non_fiction}\}$
- Create a 2x2 **contingency table**, putting the values of A as column labels, and those of B as row labels

	<i>male</i>	<i>female</i>	<i>Total</i>
<i>fiction</i>	250	200	450
<i>non_fiction</i>	50	1000	1050
<i>Total</i>	300	1200	1500

- In every cell (a_i, b_j) : **observed frequency** o_{ij}
 - the actual count of records that have $A=a_i$ and $B=b_j$

are A and B **independent**?

χ^2 Test...

- For every cell (a_i, b_j) compute the **expected frequency** e_{ij} of the event that $A=a_i$ and $B=b_j$, assuming A and B are **independent**:
 - recall that you have N tuples

$$\begin{aligned} e_{ij} &= N \times P(A = a_i \wedge B = b_j) \\ &= N \times P(A = a_i) \times P(B = b_j) \\ &= \frac{1}{N} (count(A = a_i) \times count(B = b_j)) \end{aligned}$$

	<i>male</i>	<i>female</i>	<i>Total</i>
<i>fiction</i>	250 (90)	200 (360)	450
<i>non_fiction</i>	50 (210)	1000 (840)	1050
<i>Total</i>	300	1200	1500

- observed \simeq expected \rightarrow A and B are independent
- observed $\not\simeq$ expected \rightarrow A and B are dependent

χ^2 Test...

- compute the χ^2 value using the following formula:

$$\chi^2 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}} = \sum_{i=1}^c \sum_{j=1}^r \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

- large $\chi^2 \rightarrow$ more likely A and B are related

- hypothesis testing

- hypothesis: A and B are independent

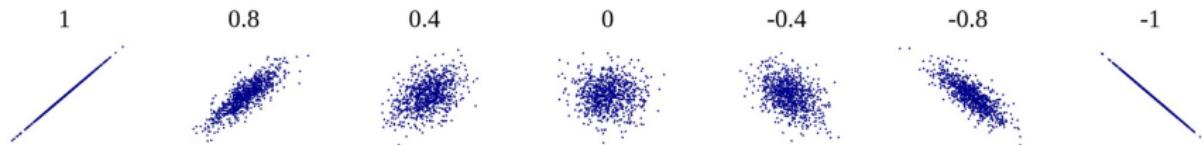
- compute the degrees of freedom as $(r-1) \times (c-1)$

Degrees of freedom	χ^2 value										
1	0.004	0.02	0.06	0.15	0.46	1.07	1.64	2.71	3.84	6.64	10.83
2	0.10	0.21	0.45	0.71	1.39	2.41	3.22	4.60	5.99	9.21	13.82
Level of significance	0.95	0.90	0.80	0.70	0.50	0.30	0.20	0.10	0.05	0.01	0.001
	Non-significant										
	Significance										

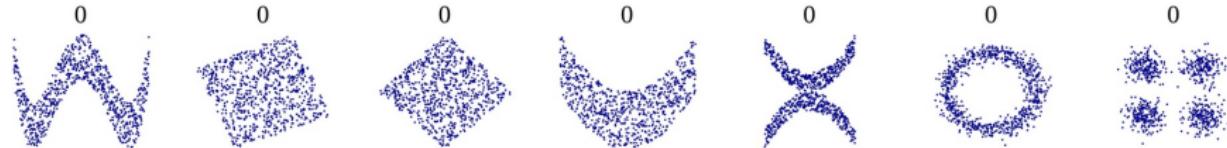
- reject the hypothesis (typically) at significance level 0.001, if χ^2 is larger than the corresponding value at the table
 - In our example, $\chi^2 = 507.93 > 10.83$ and, thus, A and B are not independent (the hypothesis is rejected)

Correlation and Dependence

Zero correlation coefficient = Independent?



- independence \Rightarrow uncorrelated
- uncorrelated \Rightarrow independence X



Data Preprocessing: Data Transformation

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

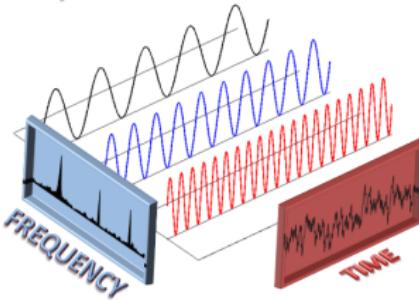
Fall 2017

Data Transformation

Goal: modify the data in order to improve data mining performance

- **attribute/feature construction**

- create **new** attributes (features) that can capture the important information in a data set more effectively than the original ones
- e.g., Fourier transform, wavelet transform



- **normalization:** scaled to fall within a smaller, specified range
 - min-max normalization
 - z-score normalization
- **discretization**

Why Data Normalization?

Example

one variable is 100 times larger than another (on average)

- your model may be better behaved if you normalize (standardize) the two variables to be approximately equivalent

Min-Max Normalization

- \min_A, \max_A : minimum and maximum values of attribute A
- maps a value v of A to a new range $[new_min_A, new_max_A]$ as

$$v' = \frac{v - \min_A}{\max_A - \min_A} (new_max_A - new_min_A) + new_min_A$$

Example

- let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]
- then \$73,000 is mapped to $\frac{73600 - 12000}{98000 - 12000} = 0.716$
- preserves the relationships among the original values
- typically, we map values to range [0.0, 1.0] or [-1.0, 1.0]

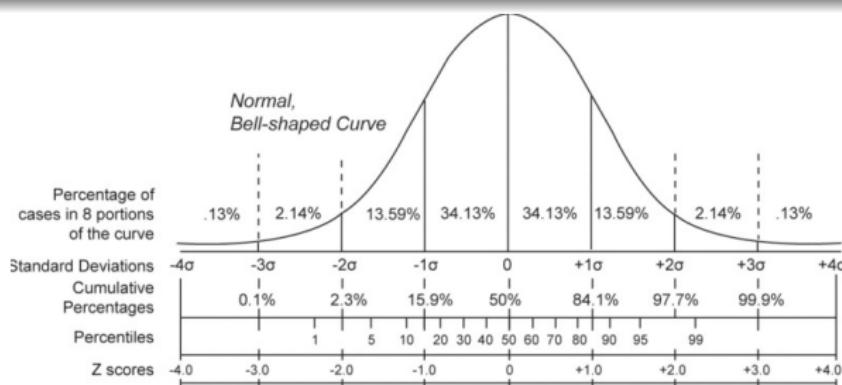
Z-Score Normalization

how many standard deviations from the average that your data lies?

- the new value v' is calculated as $v' = \frac{v - \bar{A}}{\sigma_A}$

Example

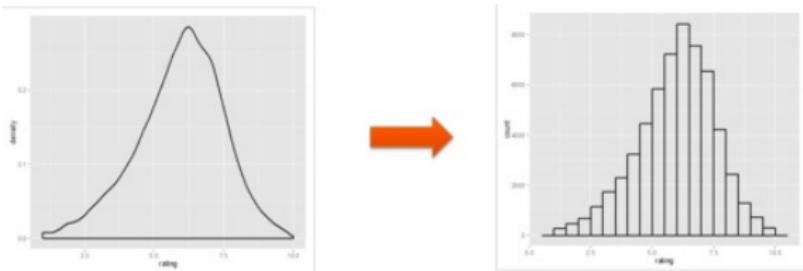
- let $\mu = 54,000, \sigma = 16,000$. Then $\frac{73000 - 54000}{16000} = 1.225$



- useful when we do not know the minimum and maximum of an attribute, or when we have outliers

Data Transformation: Discretization

- divides the range of a **continuous** attribute (e.g., *age*) into intervals
 - assign these intervals labels such as $0 - 10, 11 - 20, \dots$, or *youth, adult, senior*, etc
- reduces data size
- example methods
 - **histograms** (discussed in the previous lecture)
 - the new dataset consists of the bucket labels



- **cluster analysis** (will be discussed in a later lecture)
- **decision-tree analysis** (will be discussed in a later lecture)

Data Preprocessing: Data Reduction

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Data Reduction

Data reduction obtains a **reduced representation** of the dataset, while allowing (almost) the same analytical results to be produced

Why data reduction?

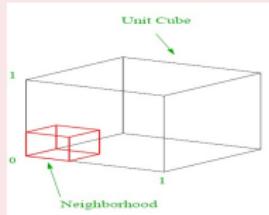
- a database/data warehouse may store terabytes of data
- complex data analysis may take a very long time to run on the complete data set

Data reduction strategies

- **dimensionality reduction**: reduce the number of attributes
 - principal components analysis (PCA)
 - feature selection
- use a smaller form of data representation
 - regression, histograms, clustering, sampling, data cube aggregation

Motivation

Suppose that points are uniformly distributed in a d -dimensional unit hypercube. If we want to construct a hypercube neighborhood to capture a fraction r of the observations, what is the edge length ℓ of this cube?



- volume of cube: $\ell^d = r$; we have $\ell = r^{1/d}$

$$d = 1$$

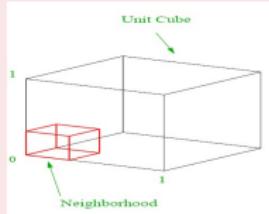
- if $r = 0.01$ then $\ell = 0.01$; if $r = 0.1$ then $\ell = 0.1$

$$d = 10$$

- if $r = 0.01$ then $\ell = 0.63$; if $r = 0.1$, then $\ell = 0.80$
- in order to capture 1% (or 10%) of the data, we must cover of the range of each input

Motivation

Suppose that points are uniformly distributed in a d -dimensional unit hypercube. If we want to construct a hypercube neighborhood to capture a fraction r of the observations, what is the edge length ℓ of this cube?



- volume of cube: $\ell^d = r$; we have $\ell = r^{1/d}$

$$d = 1$$

- if $r = 0.01$ then $\ell = 0.01$; if $r = 0.1$ then $\ell = 0.1$

$$d = 10$$

- if $r = 0.01$ then $\ell = 0.63$; if $r = 0.1$, then $\ell = 0.80$
- in order to capture 1% (or 10%) of the data, we must cover 63% (80%) of the range of each input

Motivation...

If $n = 100$ represents a dense sample for one single input, how large should n be in order to have the same sampling density with $d = 10$?

- $n = 100^{10}$
- the number of required points increases **exponentially** to maintain the same sampling density

Curse of dimensionality

- when dimensionality increases, data becomes increasingly **sparse**
- density and distance between points → less meaningful

Dimensionality Reduction

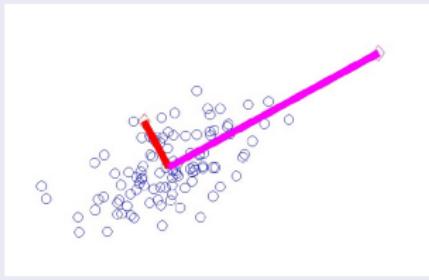
- avoid the curse of dimensionality
- help eliminate irrelevant features and reduce noise
- reduce time and space required in data mining
- allow easier visualization

Dimensionality reduction techniques

- principal component analysis
- feature selection

Principal Component Analysis (PCA)

Find a **projection** that captures the largest amount of **variation** in data



- the original data are projected onto a **lower-dimensional** space, resulting in dimensionality reduction

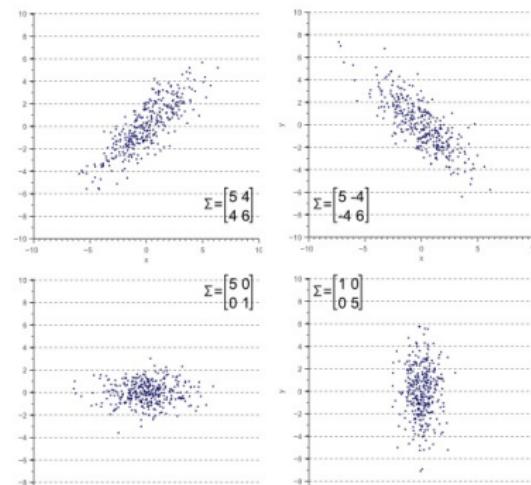
How to Measure Variation in Data?

- 1d: variance
- 2d: variance and covariance

$$\frac{1}{N} \sum_{i=1}^N (a_i - \bar{A})(b_i - \bar{B})$$

- correlation coefficient?

Given covariance matrix $\begin{bmatrix} 5 & 4 \\ 4 & 6 \end{bmatrix}$, what will the data look like?



How to Measure Variation in Data?...

- 3d: attributes (x, y, z)

$$\begin{bmatrix} \text{cov}(x, x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, x) & \text{cov}(y, y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{cov}(z, z) \end{bmatrix}$$

- covariance matrix

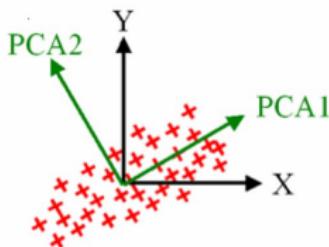
$$\mathbf{C} = \frac{1}{n} \sum_{k=1}^n (\mathbf{x}_k - \bar{\mathbf{x}})(\mathbf{x}_k - \bar{\mathbf{x}})^T$$

projection that captures the largest amount of variation
= projection \mathbf{w} s.t. $\text{var}(\mathbf{w}^T \mathbf{x})$ is maximized

PCA

find the **eigenvectors** of the covariance matrix **C**

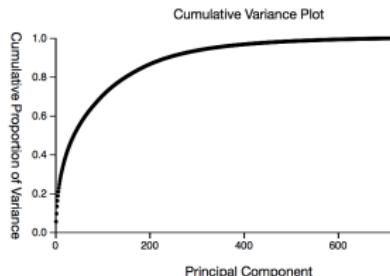
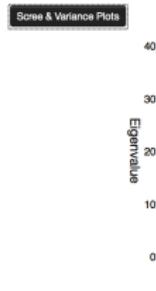
- $\mathbf{Cv} = \lambda\mathbf{v}$
- \mathbf{v} : **principal components**
- λ : eigenvalue
 - measures the variance magnitude in the direction of the eigenvector
 - decreasing eigenvalue \rightarrow decreasing “significance” or strength



Proportion of Variance Explained

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_d}$$

- λ_i are sorted in **descending** order
- decreasing eigenvalue → decreasing “significance” or strength
- data size reduced by eliminating components with **small** eigenvalues



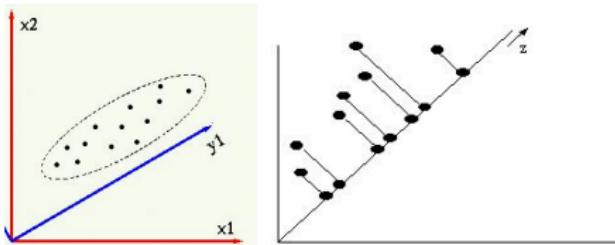
Example

X_1	X_2
19	63
39	74
30	87
30	23
15	35
15	43
15	32
30	73

$$C = \begin{bmatrix} 75 & 106 \\ 106 & 482 \end{bmatrix}; \text{ eigenvectors:}$$

- $e_1 = [-0.98, -0.21], \lambda_1 = 51.8;$
 $e_2 = [0.21, -0.98], \lambda_2 = 560.2$
- the eigenvector is more important! keep!
- obtain the final data as

$$y_i = [x_{i1} \ x_{i2}] \begin{bmatrix} 0.21 \\ -0.98 \end{bmatrix} = 0.21x_{i1} - 0.98x_{i2}$$



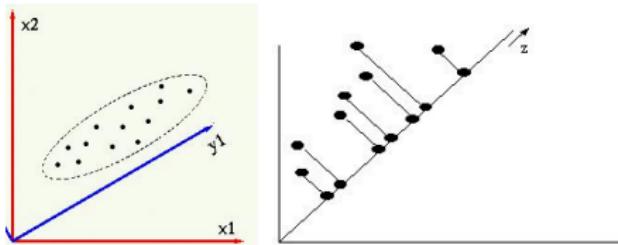
Example

X_1	X_2
19	63
39	74
30	87
30	23
15	35
15	43
15	32
30	73

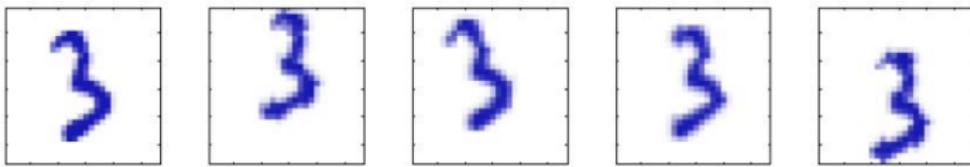
$$C = \begin{bmatrix} 75 & 106 \\ 106 & 482 \end{bmatrix}; \text{ eigenvectors:}$$

- $e_1 = [-0.98, -0.21], \lambda_1 = 51.8;$
 $e_2 = [0.21, -0.98], \lambda_2 = 560.2$
- the second eigenvector is more important! keep!
- obtain the final data as

$$y_i = [x_{i1} \ x_{i2}] \begin{bmatrix} 0.21 \\ -0.98 \end{bmatrix} = 0.21x_{i1} - 0.98x_{i2}$$

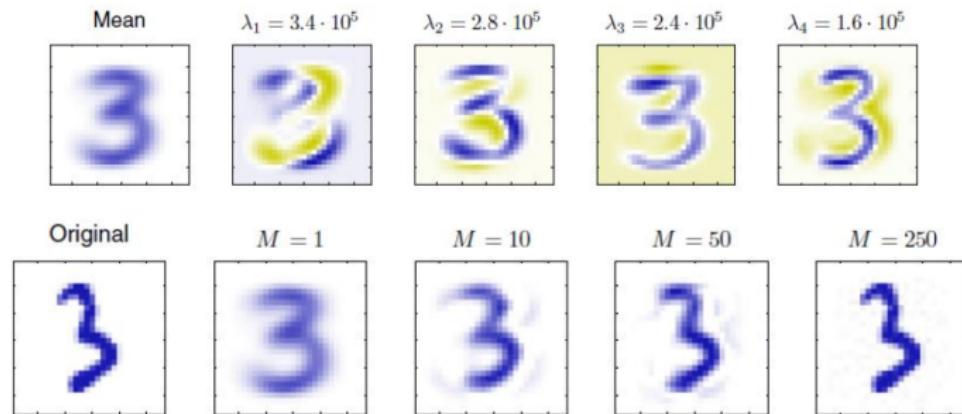


Example



- a collection of 100×100 images created from one image by introducing random displacement and rotation

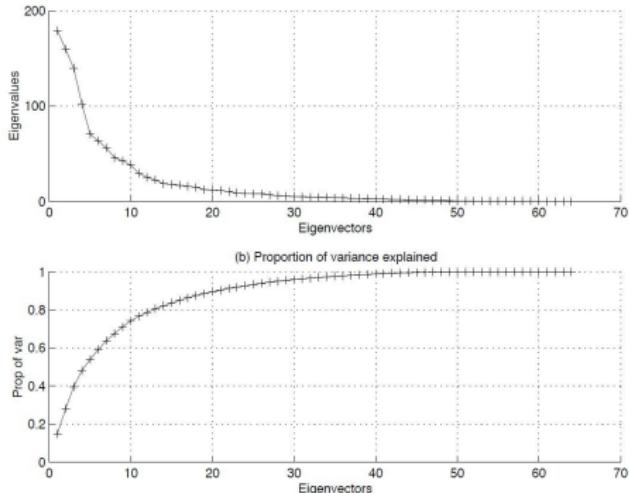
eigenvectors



How to Choose the Number of PCs (k)?

Use the proportion of variance explained:

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_d} \quad (\lambda_i \text{ are sorted in descending order})$$



- e.g., stop at proportion of variance > 0.9

Attribute/Feature Subset Selection

- another way to reduce dimensionality of data
- **redundant attributes**
 - duplicate much or all of the information contained in one or more other attributes

Example

“purchase price of a product” and “the amount of sales tax paid”

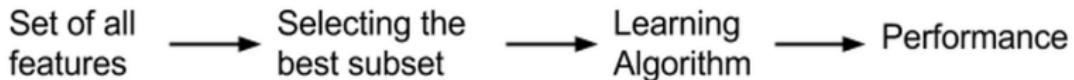
- **irrelevant attributes**
 - contain no information that is useful for the data mining task

Example

students' ID is often irrelevant to the task of predicting students' GPA

Goal

Select the minimum possible **subset** of attributes, such that the quality of the data mining task is not compromised



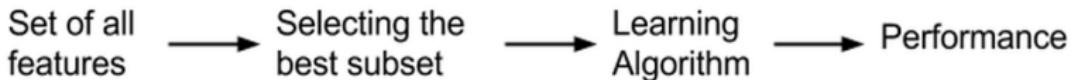
Challenging



- there are 2^d possible attribute combinations of d
- very difficult to test all possible (**exponential**) combinations of attributes

Goal

Select the minimum possible **subset** of attributes, such that the quality of the data mining task is not compromised



Challenging



- there are 2^d possible attribute combinations of d
- very difficult to test all possible (**exponential**) combinations of attributes

Typical Attribute Selection Methods

Forward selection	Backward elimination
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$
Initial reduced set: $\{\}$ $=> \{A_1\}$ $=> \{A_1, A_4\}$ $=>$ Reduced attribute set: $\{A_1, A_4, A_6\}$	$=> \{A_1, A_3, A_4, A_5, A_6\}$ $=> \{A_1, A_4, A_5, A_6\}$ $=>$ Reduced attribute set: $\{A_1, A_4, A_6\}$

① greedy forward selection

- the **best** single-attribute is picked first
- the next best attribute condition to the first, ...

which attribute is the **best**?

- e.g., correlation between the attribute and target value

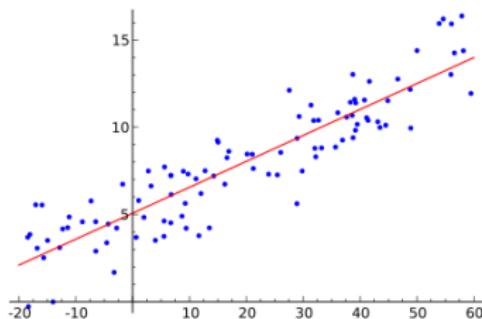
② greedy backward elimination

- repeatedly eliminate the **worst** attribute

③ combined attribute selection and elimination

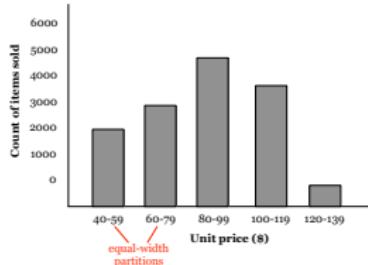
Parametric Methods

- assume the data fits some **model**, estimate model **parameters**, store only the parameters, and discard the data (except possible outliers)
- e.g., in **linear regression** the data can be modeled to fit a straight line (will be studied in a later lecture)

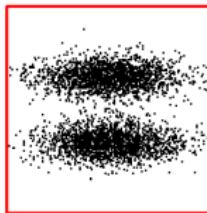


Nonparametric Methods

- do **not** assume models, e.g.,
- histograms (discussed in the previous lecture)



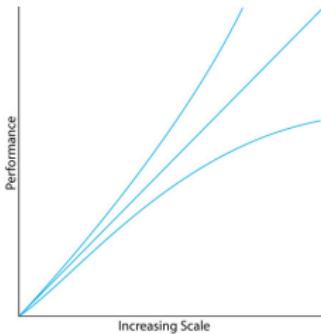
- clustering



- e.g., a cluster of points can be represented by their **centroids**
- clustering will be discussed in a later lecture
- sampling
- data cube aggregation

Sampling

- obtaining a **small sample** to represent the whole data set
- allow a mining algorithm to run in complexity that is potentially **sublinear** to the size of the data



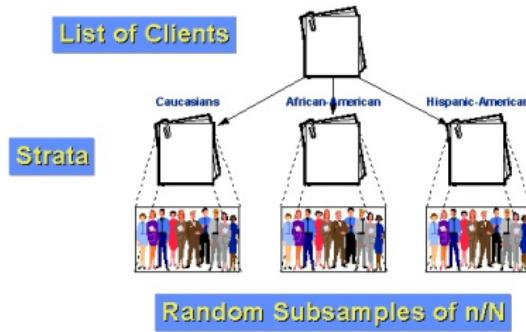
With replacement or not?

- sampling **without replacement**
 - once an object is selected, it is removed from the population
- sampling **with replacement**
 - a selected object is not removed from the population

Sampling...

How to choose a representative subset of the data?

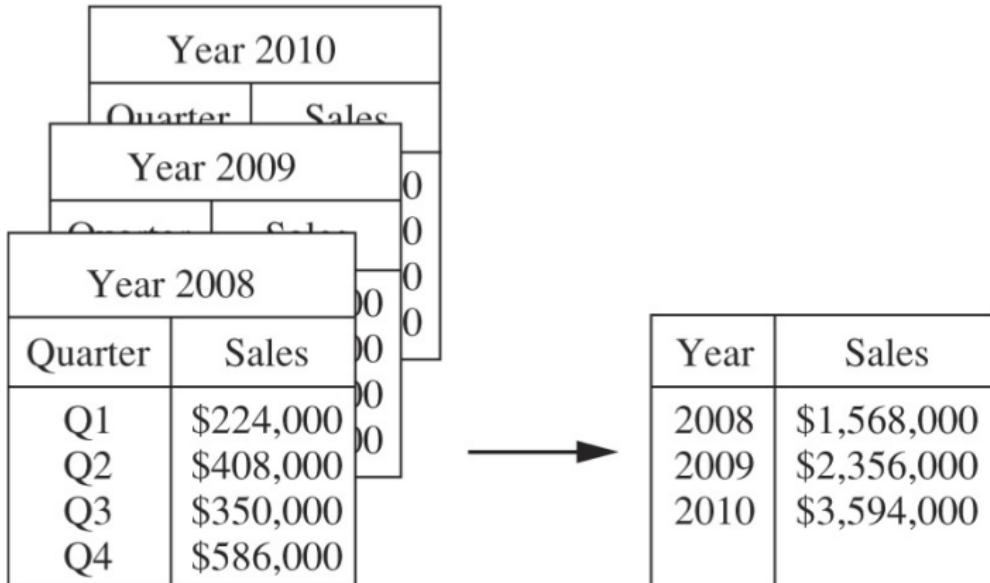
- simple **random sampling**
 - there is an equal probability of selecting any particular item
- **stratified sampling**
 - partition the data set (e.g., by age group), and draw samples from each partition



- useful when the data is skewed

Data Cube Aggregation

- use the smallest representation which is enough to solve the task



Classification: Introduction

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Application Examples

Example

- You are a bank loan officer and need to know which loan applicants are “safe” and which are “risky” for the bank
 - in other words, you want to assign **labels** “safe” or “risky” to a loan applicant

Example

- You are a marketing manager at an electronics consumer shop and want to guess whether a customer will buy a new computer
- “buys” or “doesn’t buy” to a customer

Example

- You are a medical researcher and want to predict which of (say three) specific treatments a patient should receive
- “treat_A” or “treat_B” or “treat_C” to a patient

How To Do That?

1. Gather a set of previous (e.g., archived) data

- such as past loan applicant/customer/patient data
- this dataset is called **training set**

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>class: buys_computer</i>
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle-aged	medium	no	excellent	yes
13	middle-aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- each object in the training set may have a set of attributes (of any type)
- each object must have a categorical attribute that takes as values the desired labels.
 - this attribute is called **class attribute**.
 - the class attribute values must be available in the training set

2. Analyze the training set

- create a **model** (or **classifier**) that takes as input the non-class attribute values and returns a class value

Example

$$f(\text{age}, \text{income})$$
$$f(\text{age} = \text{"young"}, \text{income} = \text{"low"}) = \text{"risky"}$$

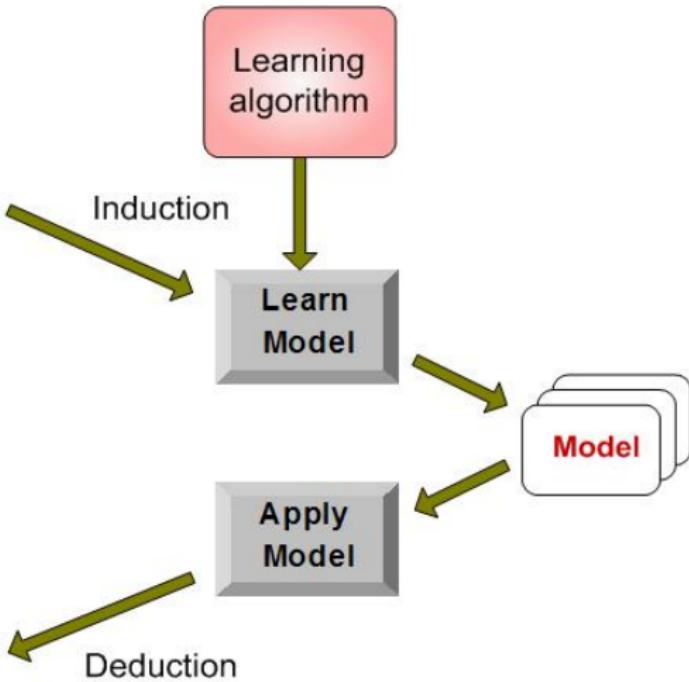
- the classifier can be implemented in different ways
 - decision trees
 - rule-based classification
 - Bayesian classification
 - neural network-based classification
 - support vector machines
 - k*-nearest neighbor classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



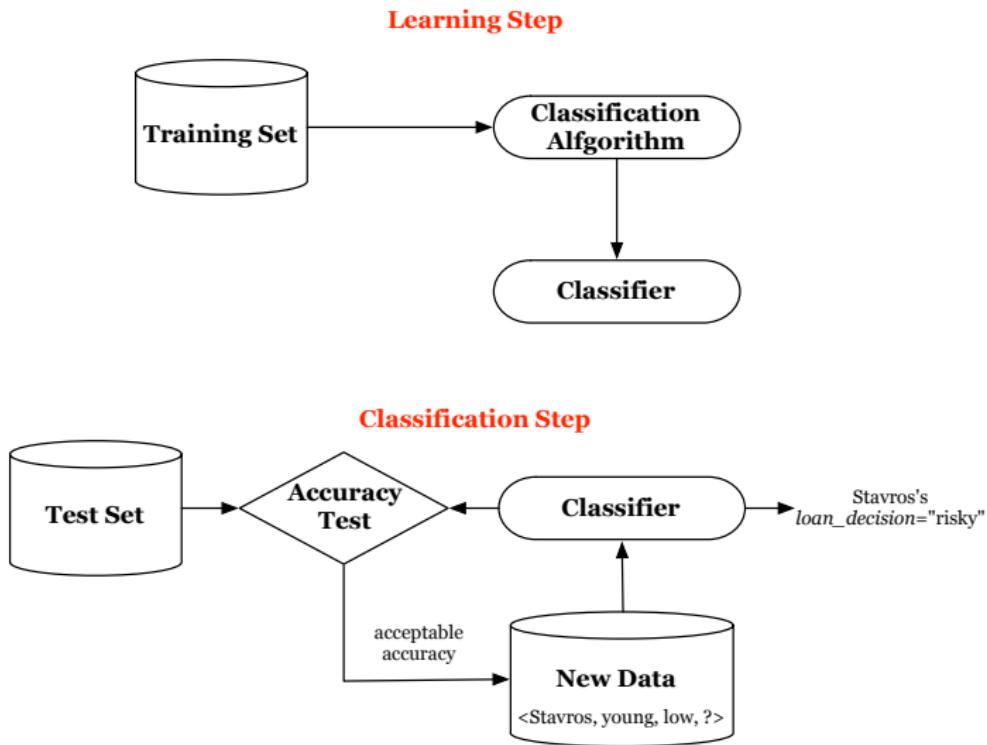
3. Estimate the **accuracy** of the classifier

- find **another** set of previous (archived) data, with the same attributes as the training set
 - the new data tuples should be disjoint from those in the training set.
 - the new dataset is called **test set**
- the accuracy of the classifier will be estimated based on the test set

4. Prediction

- If the estimated accuracy of the classifier is acceptable, use this to **classify** (i.e., assign a class attribute value to) new (i.e., unseen) objects

How To Do That?...



Preprocessing for Classification

- Data cleaning
 - remove/reduce noise
 - treat missing values
- Data reduction
 - remove redundant attributes (correlation analysis)
 - remove irrelevant attributes (attribute subset selection)
 - dimensionality reduction and discretization
- Data transformation
 - normalization

Classification Accuracy

How to measure the accuracy of a classifier?

- suppose we have already selected the training and test sets, and we have created a classifier based on the training set
- we will measure the accuracy based on the **test set**
 - specifically, we will try to predict the class value of every tuple in the test set, and compare it against their actual ones (which are already stored in the test set)

Accuracy Measures: Classification Accuracy

- the percentage of test set tuples that are correctly classified by the classifier
- a useful tool for analyzing how well the classifier can recognize tuples of different classes is the **confusion matrix**

		tuples from class "yes" classified by the classifier as "no"		accuracy of classifying "yes" tuples	
		Predicted class			
Classes		<i>buys_computer="yes"</i>	<i>buys_computer="no"</i>	Total	Accuracy #1 (%)
Actual class	<i>buys_computer="yes"</i>	6,954	46	7,000	99.34
	<i>buys_computer="no"</i>	412	2,588	3,000	86.27
	Total	7,366	2,634	10,000	95.42

total accuracy
 $Acc_1 = (6,954 + 2,588) / 10,000$

- rows and columns are the different classes
- $c_{i,j}$: value of cell at row i and column j
- $c_{i,j}$: number of tuples from class i that are classified as those of class j by the classifier

Can be Misleading

- Let us focus on a two-class problem (e.g., “non_cancer” / “cancer” patients) where
 - number of class C_1 tuples: 9,990
 - number of class C_2 tuples: 10
- If the classifier predicts everything to be class C_1 , then $Acc_1 = 9,990/10,000 = 99.9\%$
- However, this is **misleading** because the classifier does not correctly predict any tuple from C_2

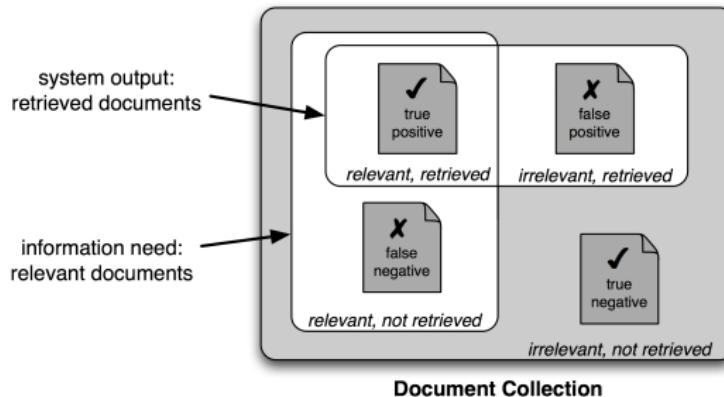
Alternative Accuracy Measures

Consider a two-class problem and the confusion matrix below

C₁ is the main class of interest

Actual class	Predicted class		Total
	C ₁	C ₂	
C ₁	true positives (t_pos)	false negatives (f_neg)	positives (pos)
C ₂	false positives (f_pos)	true negatives (t_neg)	negatives (neg)

- the positives refer to the tuples of the **main class of interest**



Alternative Accuracy Measures...

C₁ is the main class of interest

Predicted class

Actual class	Classes	C_1	C_2	Total
	C_1	true positives (t_{pos})	false negatives (f_{neg})	positives (pos)
C_2	false positives (f_{pos})	true negatives (t_{neg})	negatives (neg)	

- **precision** = $\frac{TP}{TP+FP}$
 - (information retrieval) the fraction of retrieved documents that are relevant to the search (i.e., probability that a (randomly selected) retrieved document is relevant)
- **recall** = $\frac{TP}{TP+FN}$
 - (IR) the fraction of the documents that are relevant to the query that are successfully retrieved (i.e., probability that a (randomly selected) relevant document is retrieved in a search)
- **F-measure** = $\frac{2 \cdot \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$
 - combines precision and recall (harmonic mean)

How do we select the training and test sets?

Holdout method

- suppose we possess a dataset D with $|D|$ tuples
- randomly partition D into two disjoint datasets D_1 and D_2
 - typically $|D_1| = \frac{2}{3}|D|$ and $|D_2| = \frac{1}{3}|D|$
- use D_1 as the training set and D_2 as the test set
- use the accuracy measures to derive the classifier's accuracy

Weakness

- fewer records are available for training because some are used in the test set
 - the classifier may not be as good as when all the records of D are used
- the induced classifier may be highly dependent on the composition of the training and test sets
- a class over-represented in the training set will be under-represented in the test set, and vice versa



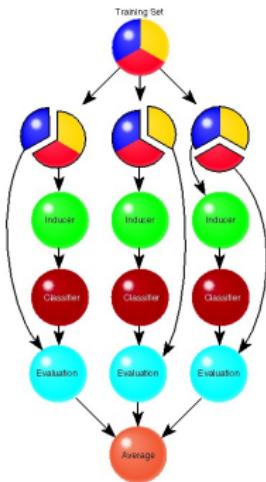
Random subsampling, cross-validation

- all these use several different training sets and test sets
- for every new training set we select, we create a new classifier that is tested against the respective new test set
- the final accuracy estimate is the average over these accuracies

Repeated Subsampling

- repeat the holdout method k times (creating a new classifier each time)
- the overall accuracy estimate is the average of the classifier accuracies obtained from each iteration

k -Fold Cross-Validation



- suppose we possess a dataset D with $|D|$ tuples
- partition D into k disjoint datasets D_1, D_2, \dots, D_k (called **folds**) of approximately equal size
- training and testing is performed k times
- in iteration i , D_i is selected as the test set (**validation set**), and the rest collectively serve as the training set

k -Fold Cross-Validation

Each example is used

- in the validation set once, and
- in the training set for the other $k - 1$ folds

A high proportion of the available data ($1 - \frac{1}{k}$) is used for training, while making use of all the data in computing the error

How many times do we need to perform training?

- typically, $k \sim 10$ is considered reasonable

If $k = |D|$

- train on $|D| - 1$ examples and validate on 1 example
- called **leave-one-out cross-validation**
- useful for small data sets

k -fold Stratified Cross-validation

Another problem:

- examples in the training set of each fold may not be representative
- e.g., all the examples of a certain class are missing → the classifier cannot learn to predict this class

How to ensure that each class is represented with **approximately equal proportions** in both the training and validation sets?

- partition the $|D|$ examples into k folds such that each class is uniformly distributed among the k folds
- the class distribution in each fold is similar to that in the original data set
- typically, use 10-fold (stratified) cross-validation

Cross-validation...

Which classifier should we eventually create in order to make predictions for the future (i.e., unseen) tuples?

- create the classifier using the entire dataset (D)
- it contains the maximum possible data we can find and thus may lead to better accuracy

We covered the basics on classification

- We described some example applications
- We defined the data classification process
- We showed how to select training/test sets and evaluate the classifier's accuracy

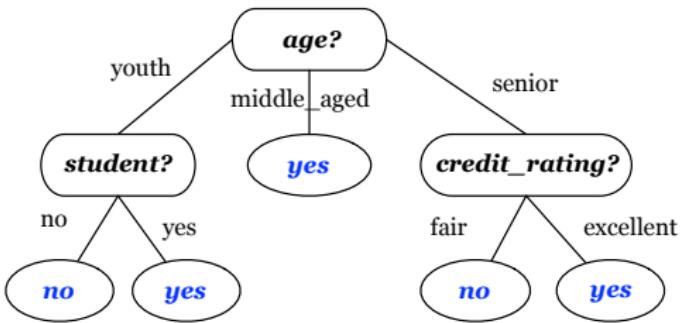
Decision Tree

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Decision Tree



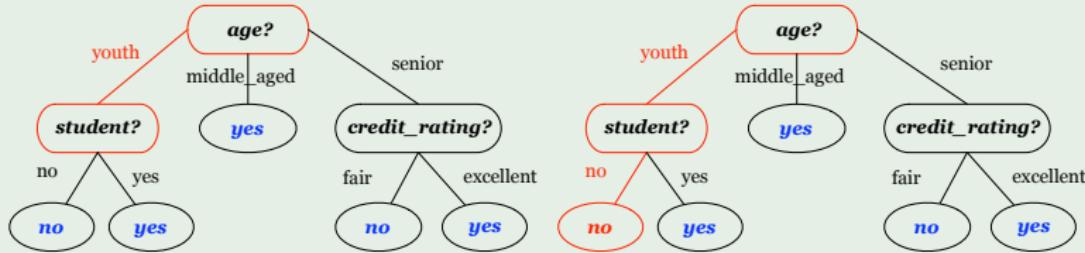
- each **internal node** denotes a test on an attribute
- each **branch** represents an outcome of the test
- each **leaf node** holds a class label

Prediction

- given a new (unseen) tuple: the associated class is unknown

Example

$\langle \text{Name}, \text{age}, \text{student}, \text{credit_rating}, \text{buys_computer} \rangle = \langle \text{Name}, \text{age}, \text{student}, \text{credit_rating}, \text{buys_computer} \rangle =$
 $\langle \text{Stavros, youth, no, excellent, ?} \rangle = \langle \text{Stavros, youth, no, excellent, no} \rangle$



- test the attribute values of the tuple against the decision tree
 - start from the root and trace a **path** to a leaf node (top-down), based on the attribute values of the tuple
- the class value included in this leaf is assigned to the tuple

Building a Decision Tree

- there are several popular decision tree **algorithms**, including **ID3**, **C4.5** and **CART**
- we will describe a **general greedy framework** followed by the majority of the algorithms
- next, we will discuss some components of this framework (which essentially differentiate the various algorithms) in more detail, as well as some other additional issues on decision tree induction

- top-down and recursive

Input

- node N
 - the first time the algorithm is called, N is the root of the tree
- dataset D of training tuples
 - initially the entire training set
- *attribute_list*: holds the set of attributes
 - initially the attributes that remained after data preprocessing
- *attribute_selection_method*: a heuristic process for selecting the attribute that “best” discriminates the given tuples according to class

output

- decision tree

Example

RID	age	income	student	credit_rating	class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Steps

Step 1

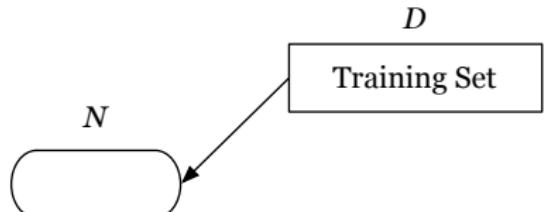
- associate node N with dataset D
- trivial; meant to emphasize that each decision tree node represents a subset of the original (entire) training set

Step 2

- end this process if one of the terminating conditions is satisfied (will be discussed soon)

Step 1

The algorithm is called with the initial single root node N , the entire training set as D , $\{age, income, student, credit_rating\}$ as the *attribute_list*, and a certain *attribute_selection_method*



Step 2

No terminating condition is satisfied yet

Step 3

Call *attribute_selection_method*

- use a **splitting criterion** to select an attribute to test at node N
 - try to “best” partition D into subsets, such that each subset is as “pure” as possible
 - a subset of D is **pure**, if it contains tuples belonging to the **same** class
- this test will result in creating new nodes (as children of N), each of which representing a subset of D

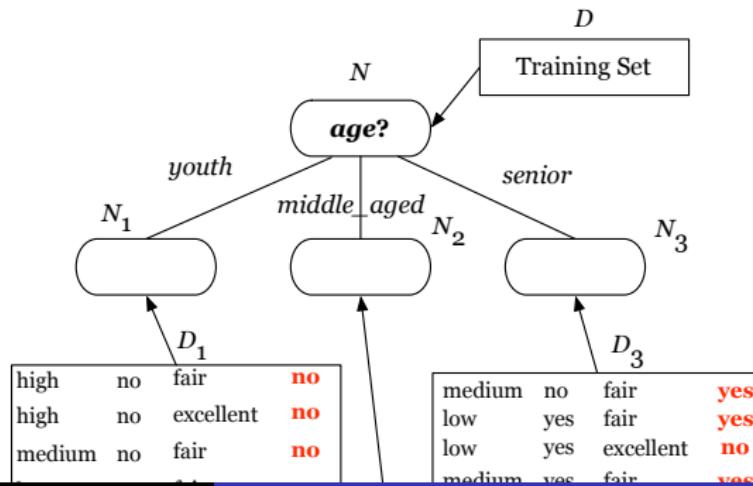
Step 3

Suppose that the *age* is selected as the splitting attribute by the *attribute_selection_method*

Step 4

Three branches and children are created for N

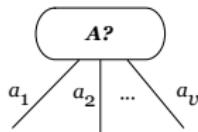
Also D is partitioned into three datasets D_1 , D_2 and D_3 according to the *age* attribute values of



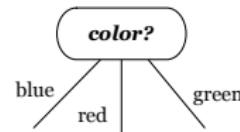
Step 4

- a branch for each of the outcomes of the splitting criterion
- a new node N_i is created for each branch i
- if the number of new nodes is m , D is partitioned accordingly into m subsets D_1, D_2, \dots, D_m
 - each D_i contains the tuples that satisfy the splitting criterion outcome of branch i

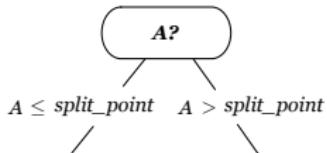
A is discrete-valued



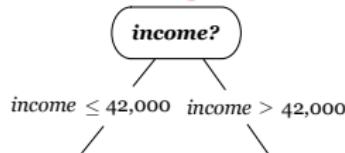
Example



A is continuous-valued



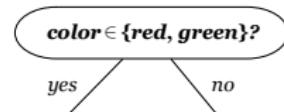
Example



A is discrete-valued and binary



Example



Example...

Step 3

Suppose that the *age* is selected as the splitting attribute by the *attribute_selection_method*

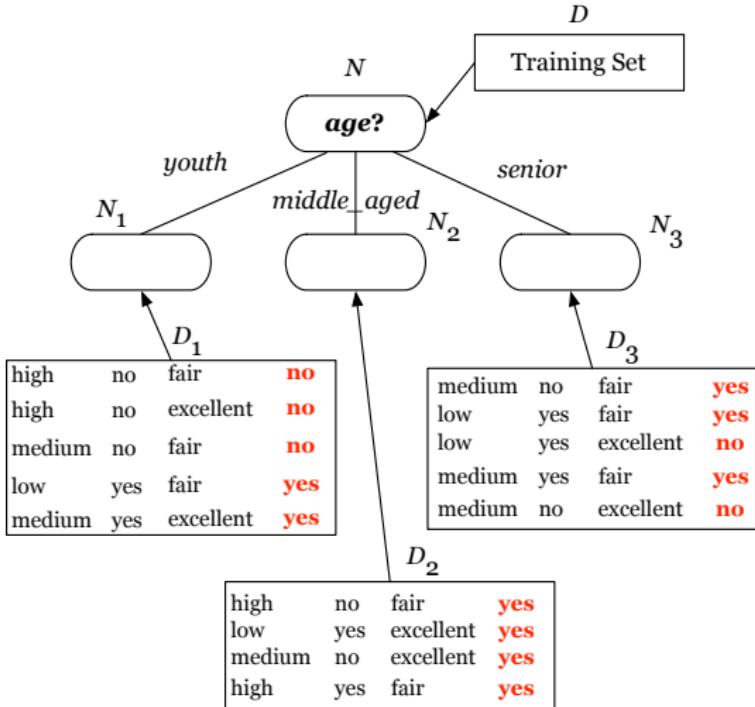
Step 4

Three branches and children are created for *N*

Also *D* is partitioned into three datasets *D*₁, *D*₂ and *D*₃ according to the *age* attribute values of the tuples

Step 5

The algorithm is recursively called for (*N*₁, *D*₁), (*N*₂, *D*₂) and (*N*₃, *D*₃), after removing *age* from *attribute_list*



The tuples in *D*₁, *D*₂ and *D*₃ are of the form
<income, student, credit_rating, class>

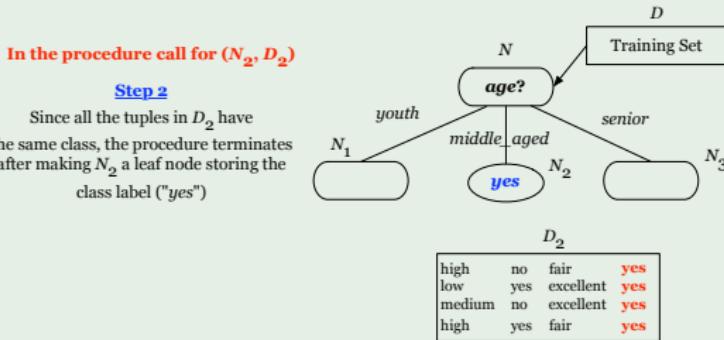
Step 5

- remove the splitting attribute A from $attribute_list$
- call $Decision_Tree_Induction(N_i, D_i, attribute_list, attribute_selection_method)$ recursively for every newly created (N_i, D_i) pair

Terminating Conditions

- ① all of the tuples in partition D belong to the same class
 - in this case, N becomes a **leaf** and is labeled with that class

Example

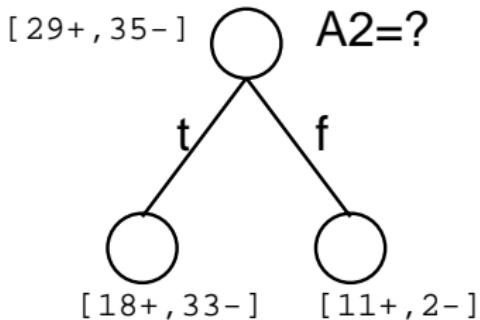
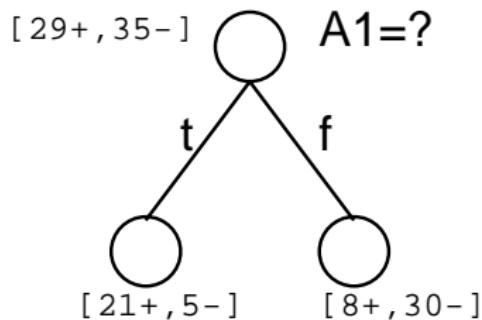


- ② there are no remaining attributes in *attribute_list* to help partitioning the tuples of D further
 - N becomes a **leaf** and is labeled with the **majority class** in D
- ③ D is empty
 - N becomes a **leaf** and is labeled with the **majority class** of its parent's dataset

Attribute Selection Measures

- ideally, the best splitting criterion is the one that decomposes D into subsets having only tuples of a single class (these subsets are called **pure**)
- since it may not be always possible to select a splitting criterion that derives only pure subsets, the attribute selection measure provides a **ranking** for each attribute
- the attribute with the **best score** is selected as the splitting attribute
- we will study three attribute selection measures
 - **Information Gain** (used in ID3)
 - **Gain Ratio** (used in C4.5)
 - **Gini Index** (used in CART)

Which Attribute is Best?



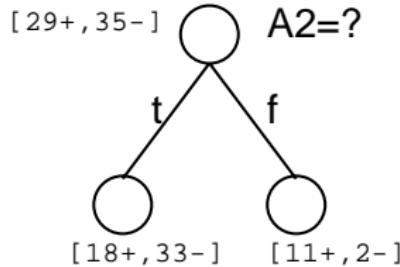
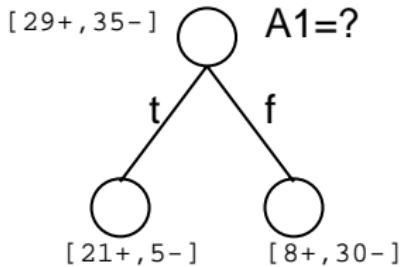
Example

I am thinking of an integer between 1 and 1,000 – what is it?
What is the first question you would ask?

- “Is it 752?”, or
- “Is it a prime number between 123 and 239?”, or
- “Is it between 1 and 500?”

Which answer provides the most information?

Idea

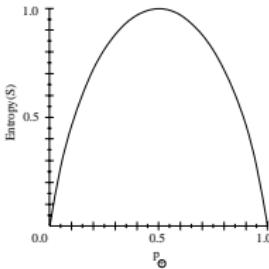


Want attributes that split examples into sets that are relatively **pure** in one label

How close is a set of instances to having just one label?

Entropy: Intuitive Notion

- measures the impurity, uncertainty, irregularity, surprise
- suppose we have two discrete classes (in general, can have > 2)
 - D : a sample of training examples
 - p_{\oplus} : proportion of positive examples in D
 - p_{\ominus} : proportion of negative examples in D
- optimal purity** (impurity/uncertainty= 0): either
 - $p_{\oplus} = 1, p_{\ominus} = 0$
 - $p_{\oplus} = 0, p_{\ominus} = 1$
- least pure** (maximum impurity/uncertainty):
 - $p_{\oplus} = 0.5, p_{\ominus} = 0.5$



Entropy: Definition

Definition

$$\text{Entropy}(D) \equiv -p_{\oplus} \log_b p_{\oplus} - p_{\ominus} \log_b p_{\ominus}$$

- if $p_{\oplus} = 0$, take $p_{\oplus} \log_b p_{\oplus} = 0$

What units is entropy measured in?

Depends on the base b of the log

- $b = e$: **nats**
- $b = 2$: **bits** (adopted here)

Example

Entropy of a fair coin = 1 bit

Example

D is a collection of 14 examples, 9 positive and 5 negative

$$\text{Entropy}([9+, 5-]) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$

Properties

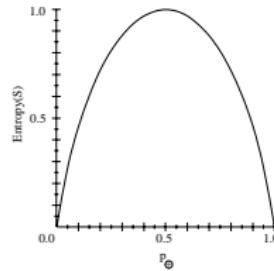
All members of D belong to the same class

- \rightarrow entropy = 0

D has an equal number of +ve and -ve examples

- \rightarrow entropy = 1

Otherwise, entropy is between 0 and 1

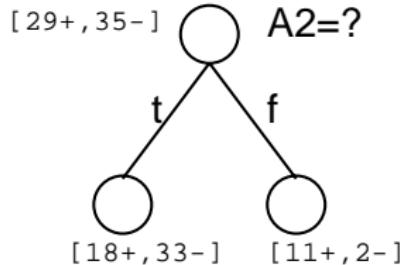
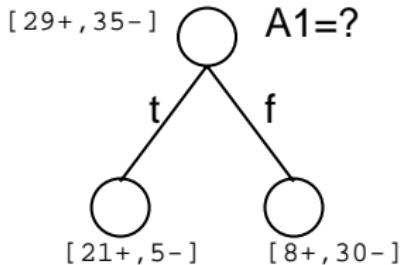


In general, if the target attribute can take on m different values

$$\text{Entropy}(D) = \sum_{i=1}^m -p_i \log_2 p_i$$

- entropy is between 0 and $\log_2 m$

Information Gain



What is the **uncertainty removed** by splitting on the value of A ?

Definition

The **information gain** of D relative to attribute A is the expected **reduction** in entropy caused by knowing the value of A

- D_v : the set of examples in D where attribute A has value v

$$Gain(D, A) \equiv Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} Entropy(D_v)$$

Example

RID	age	income	student	credit_rating	class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- $\text{entropy}(D) = 0.940$
- we have to compute $\text{Gain}(A)$ for every attribute A

Example

- let us first focus on $A = \text{age}$

$$\text{entropy}(D_{\text{youth}}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5}$$

$$\text{entropy}(D_{\text{middle_aged}}) = -\frac{4}{4} \log_2 \frac{4}{4}$$

$$\text{entropy}(D_{\text{senior}}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5}$$

$$\begin{aligned}\text{weighted average} &= \frac{5}{14} \times \text{entropy}(D_{\text{youth}}) + \frac{4}{14} \times \text{entropy}(D_{\text{middle_aged}}) \\ &\quad + \frac{5}{14} \times \text{entropy}(D_{\text{senior}}) \\ &= 0.694 \text{ bits}\end{aligned}$$

- information gain for age :

$$\text{Gain}(D, \text{age}) = 0.940 - 0.694 = 0.246 \text{ bits}$$

- Similarly, $\text{Gain}(D, \text{income}) = 0.029$,

$$\text{Gain}(D, \text{student}) = 0.151, \text{ and } \text{Gain}(D, \text{credit_rating}) = 0.048$$

- split according to age

Continuous-Valued Attributes

Example

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

Discretize the continuous-valued attributes

- Create a boolean attribute A_c that is true if $A < c$ and false otherwise

How to pick the threshold c ?

Continuous-Valued Attributes

Example

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

- ① sort the attribute values in the training set ($\{v_1, v_2, \dots, v_m\}$)
- ② generate a set of candidate thresholds midway between these values
 - cut at 44, 54, 66, 76, 85
 - there are thus $m - 1$ candidate thresholds
- ③ evaluate these candidate thresholds by the information gain

Example

Example

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

6 items, total weight 6.0

Att A1

Cut at 44.000 (gain 0.191):

Cut at 54.000 (gain 0.459):

Cut at 66.000 (gain 0.082):

Cut at 76.000 (gain 0.000):

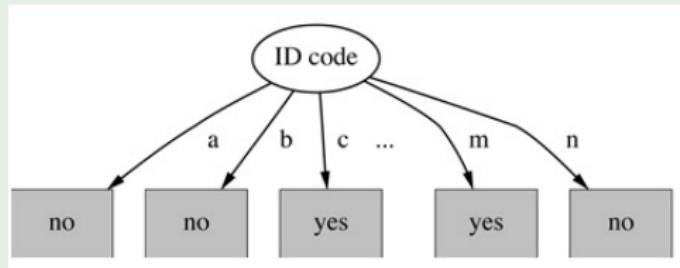
Cut at 85.000 (gain 0.191):

Best cut: 54.000

Attribute Selection Measures: Gain Ratio

Example

- if we try to split according to a unique identifier (e.g., *product_id*)



- there will be as many partitions as the number of tuples, and each partition will have a single tuple
- entropy after splitting = 0, which means that the gain is maximized, and *product_id* is chosen as the splitting criterion
- obviously, such a partitioning is useless for classification

Information gain is **biased** towards tests with many outcomes

Gain Ratio

- gain ratio is an extension of information gain, where a large number of partitions is **penalized**
- penalty should be
 - large when data is evenly spread
 - small when all data belong to one branch
- first compute $SplitInfo_A(D)$, which measures the **entropy** of the **partitioning** according to the v distinct values of A :

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \frac{|D_j|}{|D|}$$

- gain ratio is:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

- we select as the splitting criterion the A that leads to the **highest** $GainRatio(A)$ value

Example

RID	age	income	student	credit-rating	class:buy_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle-aged	medium	no	excellent	yes
13	middle-aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Using attribute income

1st partition (low) D1 has **4 tuples**

2nd partition (medium) D2 has **6 tuples**

3rd partition (high) D3 has **4 tuples**

$$Gain(income) = 0.029$$

$$GainRatio(income) = \frac{0.029}{0.926} = 0.031$$

$$\begin{aligned} SplitInfo_{income}(D) &= -\frac{4}{14} \log_2 \left(\frac{4}{14} \right) - \frac{6}{14} \log_2 \left(\frac{6}{14} \right) - \frac{4}{14} \log_2 \left(\frac{4}{14} \right) \\ &= 0.926 \end{aligned}$$

Attribute Selection Measures: Gini Index

- suppose that the class attribute of D has m distinct class values C_1, C_2, \dots, C_m
- $|D|$: cardinality of D , $|C_i|$: number of tuples in D having class label C_i
- probability p_i that a tuple of D belongs to class C_i : $|C_i|/|D|$
- Gini index

$$Gini(D) = 1 - \sum_{i=1}^m p_i^2$$

- $Gini(D)$ is small if most of the tuples belong to a few classes

Gini Index...

- if data set D is split on A into two subsets D_1 and D_2 , the gini index is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- reduction in impurity

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- select as the splitting criterion the A that leads to the highest $\Delta gini(A)$ value

Example: PlayTennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Over-fitting in Decision Trees

Tree output

```
A1 = overcast: + (4.0)
A1 = sunny:
|   A3 = high: - (3.0)
|   A3 = normal: + (2.0)
A1 = rain:
|   A4 = weak: + (3.0)
|   A4 = strong: - (2.0)
```

Over-fitting in Decision Trees

Example

Consider adding noisy training example #15:

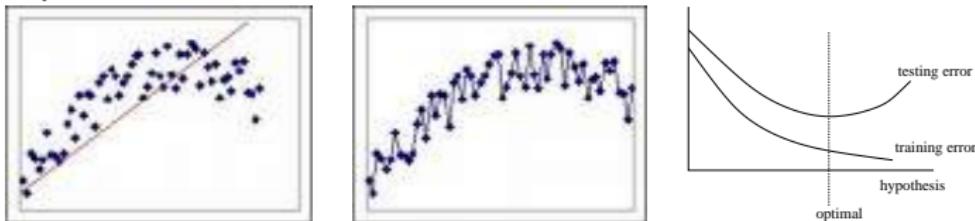
Sunny, Hot, Normal, Strong, PlayTennis = No

A1 = overcast: + (4.0)	A1 = overcast: + (4.0)
A1 = sunny:	A1 = sunny:
A3 = high: - (3.0)	A2 = hot: - (3.0)
A3 = normal: + (2.0)	A2 = cold: + (1.0)
A1 = rain:	A2 = mild:
A4 = weak: + (3.0)	A3 = high: - (1.0)
A4 = strong: - (2.0)	A3 = normal: + (1.0)
	A1 = rain:
	A4 = weak: + (3.0)
	A4 = strong: - (2.0)

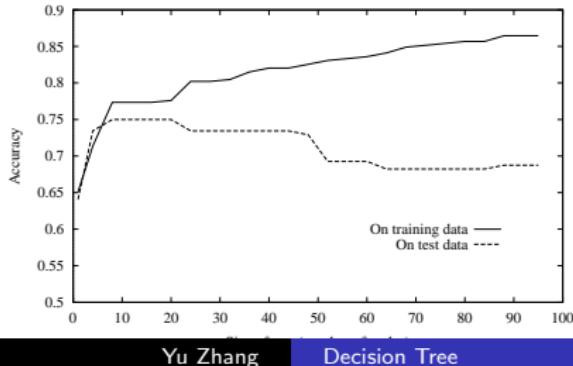
- more complex than the original tree
- fits the noisy data better than the old tree

Overfitting

- special **anomalies** of the training set may have been incorporated in the tree



- may further affect the accuracy of the tree on the test set (i.e., during prediction)
- sometimes smaller trees are preferable because of their interpretability



Tackling Overfitting

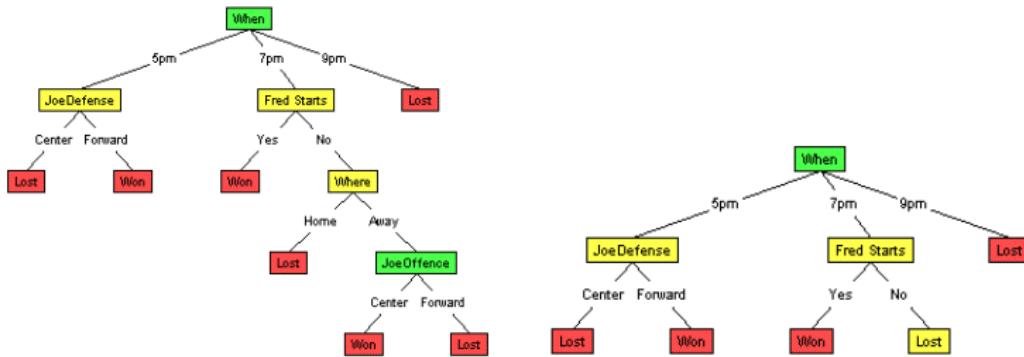
- early-stopping
- pruning

Early-Stopping

- add extra terminating conditions at Step 2 of the decision tree induction algorithm
 - stop if the number of tuples is fewer than some user-specified threshold
- may be difficult to choose an appropriate threshold

Pruning

- grow the whole tree, then prune



How to prune a decision node?

- remove the **subtree** rooted at that node
- make it a leaf node

The leaves that are left may not necessarily be pure

- assign it the most common classification of the training examples affiliated with that node

Pruning...

The pruned tree usually misclassifies **more** training examples than the unpruned tree

When to stop pruning?

- uses a validation set
- stop if the pruned tree performs worse than the original on the validation set

Which node to prune?

- remove the one that most improves validation set accuracy
- **greedy** approach

Reduced-error pruning

Do until further pruning is harmful:

- ① Evaluate impact on validation set of pruning each possible node
- ② Greedily remove the one that most improves validation set accuracy

Summary

- We explained a greedy framework for decision tree induction
- We discussed three attribute selection measures that facilitate the induction process
- We presented tree pruning methods to address tree overfitting

Bayesian Classification

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Example

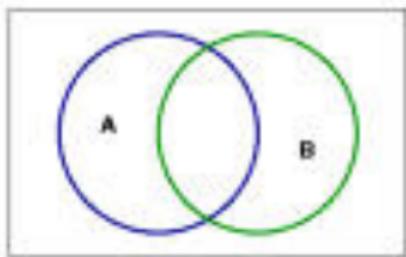
- customers, described by attributes *age* and *income*
- want to predict whether new customers are going to buy a new computer or not
- class attribute: *buys_computer*; possible values: {yes, no}
- an unseen tuple: *age =youth*, *income= 45K*

What is the **probability** that it belongs to class yes (or no)?

- based on the **Bayes rule**

Revision: Conditional Probability

- Let A and B be two events such that $P(A) > 0$
- $P(B|A)$: probability of B given that A has occurred



$$P(B|A) = \frac{P(A \cap B)}{P(A)}, \quad P(A \cap B) = P(A)P(B|A)$$

- probability that both A and B occur is equal to the probability that A occurs times the probability that B occurs given that A has occurred
- For any three events A_1, A_2, A_3 :

$$P(A_1 \cap A_2 \cap A_3) = P(A_1)P(A_2|A_1)P(A_3|A_1 \cap A_2)$$

Mutually Exclusive Events

Two events are **mutually exclusive** if they cannot occur at the same time

Example

A single card is chosen at random from a standard deck of 52 playing cards

- E_1 : the card chosen is a five, E_2 : the card chosen is a king
- mutually exclusive?



If events A_1, \dots, A_n are mutually exclusive with $\sum_{i=1}^n P(A_i) = 1$

$$P(B) = P(A_1)P(B|A_1) + P(A_2)P(B|A_2) + \dots + P(A_n)P(B|A_n)$$

Bayes Rule

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} = \frac{P(D|h)P(h)}{\sum_h P(D|h)P(h)}$$

- $P(h)$: prior probability of hypothesis h
 - initial probability that h holds, before observing the training data
- $P(h|D)$: posterior probability of h after observing the data D
- $P(D|h)$: likelihood of observing the data D given hypothesis h
- $P(D)$: probability that training data D will be observed

Example: Medical Diagnosis

Given:

- $P(\text{Cough}|\text{LungCancer}) = 0.8$
- $P(\text{LungCancer}) = 0.005$
- $P(\text{Cough}) = 0.05$

What is $P(\text{LungCancer}|\text{Cough})$?

$$\begin{aligned} & P(\text{LungCancer}|\text{Cough}) \\ &= \frac{P(\text{Cough}|\text{LungCancer})P(\text{LungCancer})}{P(\text{Cough})} \\ &= \frac{0.8 \times 0.005}{0.05} = 0.08 \end{aligned}$$

Returning to Our Previous Example

Example

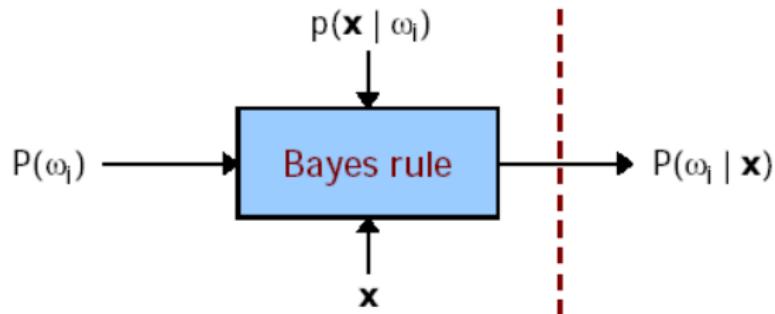
- customers, described by attributes *age* and *income*
- want to predict whether new customers are going to buy a new computer or not
- class attribute: *buys_computer*; possible values: {yes, no}
- an unseen tuple: *age =youth*, *income= 45K*
- $P(\text{yes} | (\text{youth}, 45\text{K}))$: probability that a customer will buy the computer, given that her *age* is *youth* and his/her *income* is 45K
- $P((\text{youth}, 45\text{K}) | \text{yes})$: probability that a customer has *age = youth* and *income = 45K*, given that he/she has bought the computer
- $P(\text{yes})$: probability that a customer buys the computer
- $P((\text{youth}, 45\text{K}))$: probability that a customer's *age* is *youth* and the *income* is 45K

Bayes Rule...

$$P(\omega_i | \mathbf{x}) = \frac{p(\mathbf{x} | \omega_i) P(\omega_i)}{p(\mathbf{x})}$$

Given

Compute



- relates the prior probability (**before** observing D) and the posterior probability (**after** observing D)

How to predict the class of tuple \mathbf{x} ?

- ① computes probability $P(C_i|\mathbf{x})$ for every possible class C_i
- ② assigns \mathbf{x} to the class C_i that has the **maximum posterior probability** (MAP) $P(C_i|\mathbf{x})$
 - $P(\mathbf{x})$ is constant for all classes → only needs to be maximize $P(\mathbf{x}|C_i)P(C_i)$

Example

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, 0.008 of the entire population have this cancer.

Does the patient have cancer or not?

$$P(\text{cancer}) = 0.008 \quad P(\neg\text{cancer}) = 0.992$$

$$P(+|\text{cancer}) = 0.98 \quad P(-|\text{cancer}) = 0.02$$

$$P(+|\neg\text{cancer}) = 0.03 \quad P(-|\neg\text{cancer}) = 0.97$$

$$P(+|\text{cancer})P(\text{cancer}) = 0.98(0.008) = 0.0078$$

$$P(+|\neg\text{cancer})P(\neg\text{cancer}) = 0.03(0.992) = 0.0298$$

MAP decision = $\neg\text{cancer}$

Naive Bayes Classifier

How to estimate probabilities $P(\mathbf{x}|C_i)$ and $P(C_i)$?

- estimate these probabilities based on training data!

$P(C_i)$

- simply compute $P(C_i) = |C_i|/|D|$,
 - $|C_i|$: number of tuples in the training set D having class C_i
 - $|D|$: total number of tuples in D

$P(\mathbf{x}|C_i)$

Can we estimate this probability by the fraction of tuples in D that belong to class C_i and have the attribute values described in \mathbf{x} ?

- NO, unless we have a very large amount of data in D .
Otherwise, the estimate is not going to be reliable
- Instead, the naive Bayes classifier assumes **conditional independence**

Revision: Independence

- Two random variables X and Y are **independent** if

$$P(X|Y) = P(X), \text{ or } P(Y|X) = P(Y)$$

- Knowledge about X contains **no** information about Y
- Equivalently, $P(X, Y) = P(X)P(Y)$
- If n Boolean variables (X_1, \dots, X_n) are independent

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i)$$

Example

- X : result of tossing a fair coin for the first time; Y : result of second tossing of the same coin
- X : result of US election; Y : your grades in this course

Question: Are these independent?

X : midterm exam grade; Y : final exam grade

Example



- There is a bag of 100 coins. 10 coins were made by a malfunctioning machine and are biased toward head. Tossing such a coin results in head 80% of the time. The other coins are fair.
- Randomly draw a coin from the bag and toss it a few times
- X_i : result of the i th tossing

Are X_i 's independent of each other?

- If I get 9 heads in first 10 tosses, then the coin is probably a biased coin. Hence the next tossing will be more likely to result in a head than a tail.

Example...

- Y : whether the coin is produced by the malfunctioning machine

Are X_i 's conditionally independent given Y ?

- If the coin is not biased, the probability of getting a head in one toss is $1/2$ regardless of the results of other tosses
- If the coin is biased, the probability of getting a head in one toss is 80% regardless of the results of other tosses
- If I already knew whether the coin is biased or not, learning the value of X_i does not give me additional information about X_j

Conditional Independence

- Absolute independence is a very strong requirement, seldom met
- Two random variables X and Y are **conditionally independent given Z** if

$$P(X|Y, Z) = P(X|Z)$$

- Given Z , knowledge about X contains no information about Y
 - Y might contain some information about X
 - however all the information about X contained in Y are also contained in Z

Example

$$P(\text{Thunder}|\text{Rain, Lightning}) = P(\text{Thunder}|\text{Lightning})$$

Conditional Independence...

$$P(X|Y, Z) = P(X|Z)$$

- Equivalently, $P(Y|X, Z) = P(Y|Z)$ (why?)

$$\begin{aligned} P(Y|X, Z) &= P(X|Y, Z)P(Y|Z)/P(X|Z) \\ &= P(X|Z)P(Y|Z)/P(X|Z) = P(Y|Z) \end{aligned}$$

- Equivalently, $P(X, Y|Z) = P(X|Z)P(Y|Z)$ (why?)

$$P(X, Y|Z) = P(X|Y, Z)P(Y|Z) = P(X|Z)P(Y|Z)$$

$$P(\mathbf{x}|C_i)$$

- assumes that the attributes are conditionally independent given the class label
- recall that $\mathbf{x} = (x_1, x_2, \dots, x_n)$
- compute

$$P(\mathbf{x}|C_i) = \prod_{k=1}^n P(x_k|C_i)$$

- this greatly reduces the computation cost

$$P(\mathbf{X}|C_i)$$

attribute A_k is **categorical**

- $P(x_k|C_i) \leftarrow N_{k,C_i}/N_{C_i}$
 - N_{C_i} : number of training examples that belong to C_i
 - N_{k,C_i} : number of examples that belong to C_i and $A_k = x_k$
- fraction of tuples in D that belong to C_i , whose A_k attribute is x_k

attribute A_k is **continuous-valued**

- ① either discretize A_k ; or
- ② estimate $P(x_k|C_i)$ based on some distribution (e.g., **normal distribution**)

$$P(x_k|C_i) = \frac{1}{\sqrt{2\pi}\sigma_{k,C_i}} e^{-\frac{(x_k - \mu_{k,C_i})^2}{2\sigma_{k,C_i}^2}}$$

- μ_{k,C_i} : average of the attribute values of A_k for the tuples belonging to C_i
- σ_{k,C_i} : corresponding standard deviation

Naive Bayes Classifier

Naive_Bayes_Learn(*examples*)

```
begin
    for each class  $C_i$  do
        estimate  $P(C_i)$ ;
        for each attribute  $k$  do
            | estimate  $P(x_k|C_i)$  ;
        end
    end
end
```

Classify_New_Instance(\mathbf{x})

```
begin
     $v_{NB} = \arg \max_{C_i} P(C_i) \prod_k P(x_k|C_i)$ 
end
```

Example

RID	age	income	student	credit_rating	class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Example

- We first compute the prior probability for each class:

$$P(C_1) = 9/14 = 0.643$$

$$P(C_2) = 5/14 = 0.357$$

- To derive $P(\mathbf{x}|C_i)$ for $i = 1, 2$, we need to compute the following:

$$P(\text{age} = \text{youth}|C_1) = 2/9 = 0.222$$

$$P(\text{age} = \text{youth}|C_2) = 3/5 = 0.600$$

$$P(\text{income} = \text{medium}|C_1) = 4/9 = 0.444$$

$$P(\text{income} = \text{medium}|C_2) = 2/5 = 0.400$$

$$P(\text{student} = \text{yes}|C_1) = 6/9 = 0.667$$

$$P(\text{student} = \text{yes}|C_2) = 1/5 = 0.200$$

$$P(\text{credit_rating} = \text{fair}|C_1) = 6/9 = 0.667$$

$$P(\text{credit_rating} = \text{fair}|C_2) = 2/5 = 0.400$$

Example

- Given the previous probabilities, we obtain

$$\begin{aligned}P(\mathbf{x}|C_1) &= P(\text{age} = \text{youth}|C_1) \times P(\text{income} = \text{medium}|C_1) \\&\quad \times P(\text{student} = \text{yes}|C_1) \times P(\text{credit_rating} = \text{fair}|C_1) \\&= 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044\end{aligned}$$

- Similarly,

$$P(\mathbf{x}|C_2) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019$$

- Finally, we calculate

$$P(\mathbf{x}|C_1)P(C_1) = 0.044 \times 0.643 = 0.028$$

$$P(\mathbf{x}|C_2)P(C_2) = 0.019 \times 0.357 = 0.007$$

- This implies that \mathbf{x} should be assigned C_1 , i.e.,
 $\text{buys_computer} = \text{yes}$

Naive Bayes Classifier

- Recall that $P(x_k|C_i) = N_{k,C_i}/N_{C_i}$
 - N_{C_i} : number of tuples of class C_i in D
 - N_{k,C_i} : number of tuples of C_i in D , with attribute A_k equal to x_k

What if there is no tuple of C_i having $A_k = x_k$?

- $\rightarrow P(x_k|C_i) = 0$
- $\rightarrow P(x|C_i)$ will be zero as well, which means that the effects of all the other probabilities will be canceled

One trick to avoid this is **Laplacian correction**

- modify $P(x_k|C_i)$ for every different $A_k = x_k$ to

$$P(x_k|C_i) = \frac{N_{k,C_i} + 1}{N_{C_i} + c}$$

- c is the total number of different x_k values, i.e., the number of distinct values for attribute A_k

Example

Example

- Number of tuples of C_1 with $income = low$: 0
- Number of tuples of C_1 with $income = medium$: 990
- Number of tuples of C_1 with $income = high$: 10
- Total number of tuples of C_1 : 1000
- Observe that $P(income = low|C_1) = 0/1,000 = 0.0$
- We fix it by changing probabilities as follows:

$$P(income = low|C_1) = \frac{1}{1,003}$$

$$P(income = medium|C_1) = \frac{991}{1,003}$$

$$P(income = high|C_1) = \frac{11}{1,003}$$

Advantages

- easy to implement
- good results obtained in many cases

Disadvantages

- assumption: class conditional independence, therefore potential loss of accuracy
- but it works surprisingly well anyway!

Summary

- We discussed Bayesian classification
- We explained in detail one popular Bayesian classifier, namely the naive Bayes classifier

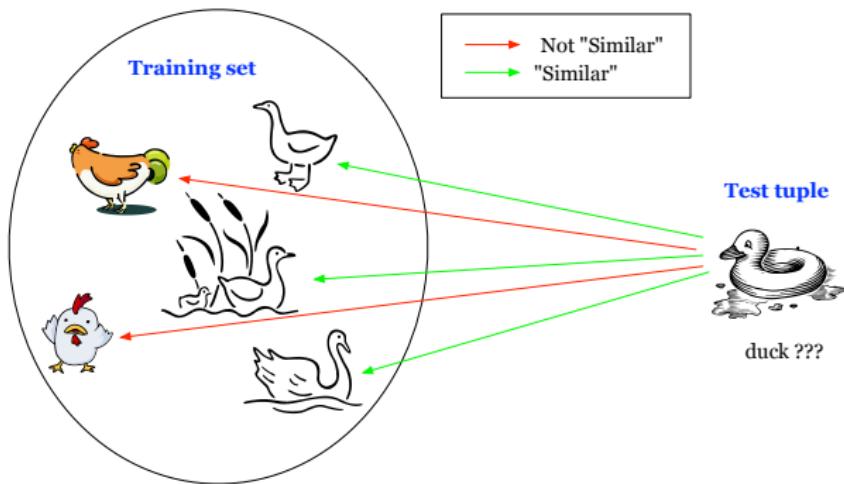
Nearest-Neighbor Classification

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Basic Idea



*"If it walks like a duck, quacks like a duck, and looks like a duck,
then it is probably a duck!"*

kNN Classification

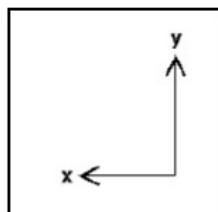
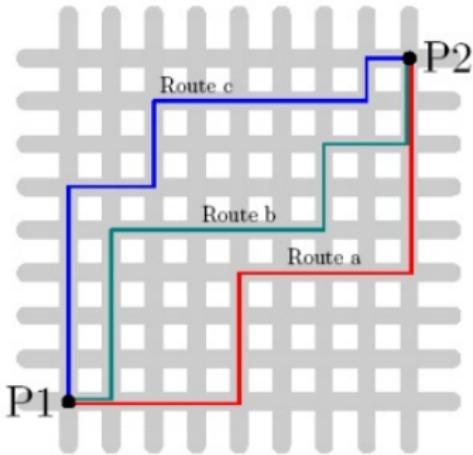
- every tuple in the training set is described by n numeric attributes
 - each tuple can be perceived as a point in a n -dimensional space
- two tuples \mathbf{x}_1 and \mathbf{x}_2 are “close” or “near” or “similar” if they have a “small” distance based on a **distance metric**
- the most popular metric is the **Euclidean distance** (ℓ_2 -distance)
 - $\mathbf{x}_1 = (x_{11}, x_{12}, \dots, x_{1n})$ and $\mathbf{x}_2 = (x_{21}, x_{22}, \dots, x_{2n})$
 - Euclidean distance between \mathbf{x}_1 and \mathbf{x}_2 is

$$dist(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{i=1}^n (x_{1i} - x_{2i})^2}$$

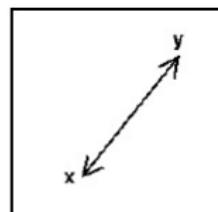
Distance Metrics

Other distances can also be used

- e.g., ℓ_1 -distance: (**Manhattan distance**) $\sum_{i=1}^n |x_{1i} - x_{2i}|$



Manhattan



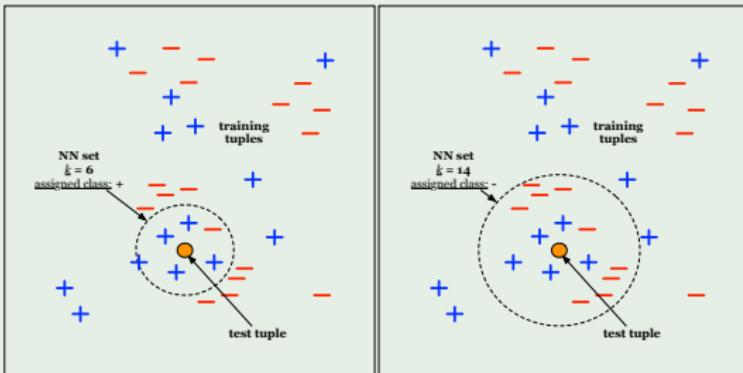
Euclidean

k -Nearest-Neighbor (k NN) Classifier

Given an unknown tuple

- search for its k nearest training tuples (“neighbors”) based on a pre-defined distance metric
- assign the unknown tuple to the majority class of its k nearest neighbor set

Example (assume Euclidean distance)



Data Preprocessing

- distance metric is affected by the **ranges** of the attribute values
- large ranges (e.g., *income*) may outweigh small ranges (e.g., *age*) in the distance computation
- perform **data normalization** on the training set, such that all values fall within range $[0, 1]$
 - e.g., transform a value $v \in [min_A, max_A]$ of an attribute A to

$$v' = \frac{v - min_A}{max_A - min_A} \quad (\text{min-max normalization})$$

What about **categorical data**?

- suppose that we measure the Euclidean distance between \mathbf{x}_1 and \mathbf{x}_2
- need to calculate $x_{1i} - x_{2i}$ on a categorical attribute A_i
- assign “0” if the values are identical, and “1” otherwise

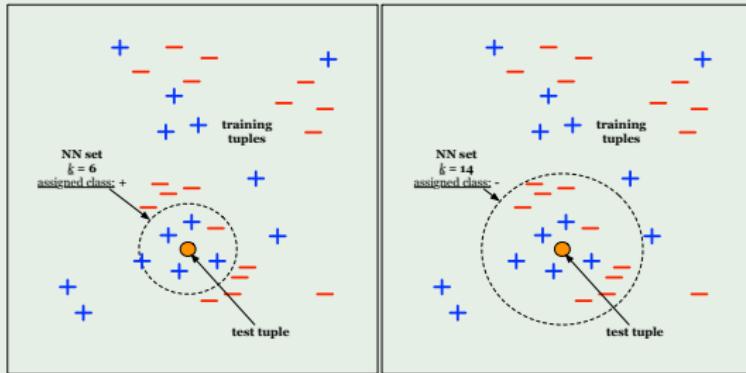
Curse of Dimensionality

- when dimensionality increases, data becomes increasingly sparse
- density and distance between points → less meaningful
- as a result, a nearest neighbor query may become non-meaningful
- in particular, distance between neighbors could be dominated by irrelevant attributes
 - dimension reduction
 - use weights to attribute higher importance to some attributes (so far we have implied that all attributes are assigned an equal weight)
 - e.g., $dist(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{i=1}^n \beta_i (x_{1i} - x_{2i})^2}$

Notes

- k is a user-defined **parameter** which affects the **accuracy** of the classifier

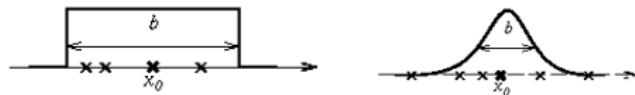
Example (assume Euclidean distance)



- k **too small** → the classifier becomes sensitive to noise
- k **too large** → the neighborhood may “mix” points from different classes
- can tune k by **cross-validation**

Notes...

- on prediction, assign the unknown tuple to the **majority class** of its k nearest neighbor set



- can also take into account its **distance** from its neighbors
 - idea: give greater weights to closer neighbors
 - e.g., weight the contribution of each of the k neighbors according to their distance to the query x_q : $w = \frac{1}{d(x_q, x_k)^2}$

Lazy Learning

decision trees, naive Bayes classifiers (and many other classifiers)

- build a model as soon as the training set is available (before seeing the test examples)
- **eager** learning

*k*NN classifier

- just **store** all training examples
- **lazy** learning

	eager	lazy
different new instances	estimates based on the same function	estimates based on different functions
approximation to the target function	global	local
computation on training	a lot	little
computation on testing	little	a lot

Rule-Based Classification

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

What is Rule-based Classification?

- use a set of **IF-THEN** rules for classification
 $\text{IF } condition_1 \text{ AND } condition_2 \text{ AND } \dots \text{ THEN } class_label$

Example

IF $age = youth$ **AND** $student = yes$ **THEN** $buys_computer = yes$

- “IF” part: **antecedent** or **precondition**
- “THEN” part: **consequent**
- if the rule precondition holds for some tuple (i.e., every condition/attribute test is true), then we say that the rule precondition is **satisfied** and that the rule **covers** the tuple

How to predict the class of a tuple \mathbf{X} ?

Example

rule R : IF $age = youth$ AND $student = yes$ THEN $buys_computer = yes$

tuple: $(age = youth, income = medium,$
 $student = yes, credit_rating = fair)$

triggers the rule R

If R is the only rule triggered by \mathbf{X} ,

- \mathbf{X} is **classified** according to the class label of R
 - $buys_computer = yes$ in the example above

If more than one rules are triggered

- need a **conflict resolution strategy** (discussed soon) to determine which rule will fire to classify \mathbf{X}

Some Terminologies: Coverage and Accuracy

- D : dataset (with $|D|$ tuples)
- Let us focus on a rule R
- coverage of R

$$\text{coverage}(R) = \frac{n_{\text{covers}}}{|D|}$$

- n_{covers} : number of tuples in D covered by R
- accuracy of R

$$\text{accuracy}(R) = \frac{n_{\text{correct}}}{n_{\text{covers}}}$$

- n_{correct} : number of tuples in D correctly classified by R
(supposing that it fires for all the covered tuples)

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>class: buys_computer</i>
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- $R: \text{IF } age = \text{youth} \text{ AND } student = \text{yes} \text{ THEN } buys_computer = \text{yes}$
- $\text{coverage}(R) = 2/14 = 14.28\%$
- $\text{accuracy}(R) = 2/2 = 100\%$

Exhaustive Rule Set

- A rule set is **exhaustive** if every tuple can be covered by at least one rule

RID	age	income	student	credit_rating	class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

R_1 : IF *age = middle_aged* THEN *buys_computer = yes*

R_2 : IF *age = senior* AND *credit_rating = excellent* THEN *buys_computer = yes*

R_3 : IF *age = senior* AND *credit_rating = fair* THEN *buys_computer = no*

- exhaustive? no

What if the rule set is not exhaustive?

- we set up a **default rule**
- an unknown tuple not triggering any rule will be classified according to this default rule
- the class label of the default rule can be
 - the **majority class of D** , or
 - the **majority class of the uncovered tuples of D**

Mutually Exclusive Rule Set

- A rule set is **mutually exclusive** if every tuple can be covered by at most one rule

RID	age	income	student	credit_rating	class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

R_1 : IF $age = youth$ AND $student = yes$ THEN $buys_computer = yes$

R_2 : IF $age = youth$ AND $credit_rating = fair$ THEN $buys_computer = no$

- mutually exclusive? no

What if the rule set is not mutually exclusive?

- as mentioned previously, we follow a **conflict resolution strategy** to determine the class of an unknown tuple
- popular strategies
 - size ordering
 - class-based ordering
 - rule-based ordering

Size Ordering

- assigns the highest priority to the triggering rule with the **largest precondition size** (i.e., the biggest number of attribute tests)
- intuitively, this means that the unknown tuple is classified according to the rule with the “toughest” requirements

Class-Based Ordering

- ① group the rules based on the **classes** they predict
- ② sort the groups based on some criterion, e.g.,
 - (a) the group that predicts the **most frequent** class goes first, the group that predicts the second most frequent class goes next, etc.
 - (b) the groups are sorted according to the **accuracy per class**
 - (c) ...
- within each group, rules predict the same class → rules do not need to be ordered
- ③ an unknown tuple is classified based on the first rule it triggers in the resulting list of rules

Rule-Based Ordering

- rules are organized into a long list (decision list) according to some criterion, e.g.,
 - rule accuracy
 - rule coverage
 - rule size (i.e., number of attribute tests in the precondition)
 - advice from domain experts
- an unknown tuple is classified by the first rule it triggers in the decision list

Example

R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds
R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes
R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals
R4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles
R5: (Live in Water = sometimes) \rightarrow Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

Rule Generation

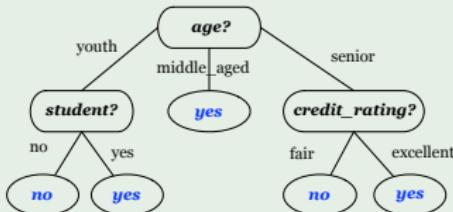
- Rule extraction from a decision tree
- Rule induction with sequential covering

Rule Extraction from a Decision Tree

- in comparison to the decision tree, these rules may be easier to understand by humans
- rule extraction from a decision tree is an indirect method
 - it operates on an already created tree, and not on the training set directly
- C4.5rules is a popular method

Rule Extraction from a Decision Tree

Example



R_1 : IF $age = youth$ AND $student = no$ THEN $buys_computer = no$

R_2 : IF $age = youth$ AND $student = yes$ THEN $buys_computer = yes$

R_3 : IF $age = middle_aged$ THEN $buys_computer = yes$

R_4 : IF $age = senior$ AND $credit_rating = excellent$ THEN $buys_computer = yes$

R_5 : IF $age = senior$ AND $credit_rating = fair$ THEN $buys_computer = no$

- create **one rule for every root-to-leaf path** in the decision tree
- every splitting criterion along the path is logically AND to form the rule precondition (IF part)
- the leaf holds the class prediction, which is assigned to the rule consequent (THEN part)

extracted set of rules is exhaustive? yes; mutually exclusive? yes

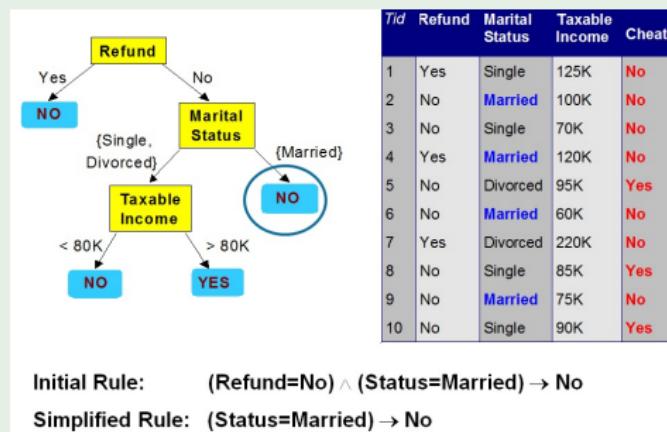
Rule Pruning: Motivation

- derive one rule for every root-to-leaf path in the tree
- the extracted rules cannot be much simpler than the decision tree itself
 - sometimes may be even more difficult to interpret the rules than the tree
- therefore, after extracting the rules, we may wish to **prune** some attribute tests from the preconditions of the rules in order to **simplify** them

How to Prune?

- test every condition in the IF part
- if the existence of this condition does not improve the **accuracy** of the rule: **remove** this condition from the rule
- repeat for every attribute test in a rule, and for all the rules

Example



after pruning, the resulting rule set may not be mutually exclusive and/or exhaustive

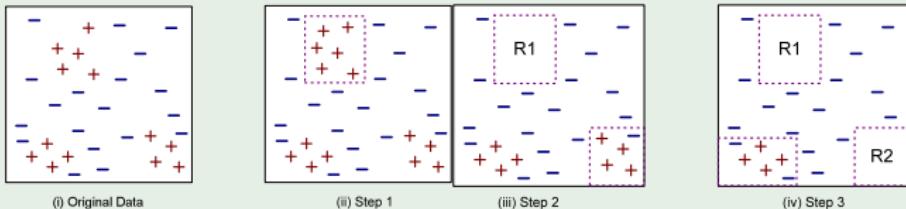
- may need a default rule and conflict resolution strategy

Rule Induction with Sequential Covering

- rules can be **directly** extracted from the training set, without requiring a decision tree
- **RIPPER** is a popular such algorithm
- rules are learned for **one class at a time**
 - cf. decision-tree induction: learning a set of rules **simultaneously**

Basic Algorithm

Example



- suppose we focus on class C_i and the current training set is D
 - ① **scan** D and learn the “**best**” rule R_{best} for C_i and D
 - this is achieved through a function `Learn_One_Rule`
 - ② **remove** from D all the tuples covered by R_{best}
 - ③ repeat the process until a **terminating condition** is met
 - e.g., when there are no tuples, or a returned rule is below a user-specified quality level
- when the terminating condition is satisfied for C_i , we move on to extract the rules for the next class

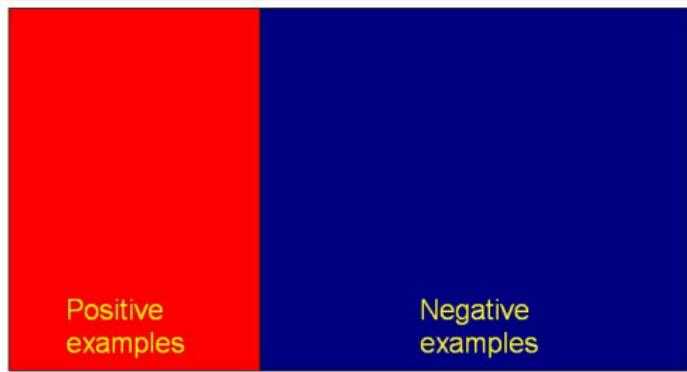
why do we need to eliminate tuples?

- otherwise, the next rule is identical to previous rule

Learn_One_Rule: Greedy Algorithm

- input: current training set D , current class C_i , a list of attributes with their values ($attribute_vals$)
- ① start with the “empty” rule $IF\ THEN\ C_i$
- ② consider every possible test that can be added to the rule
 - selected from $attribute_vals$
 - choose the “best” attribute test based on a **quality measure**, and adds it to the IF part of the previously “empty” rule $IF\ condition_1\ THEN\ C_i$
- ③ choose the **next “best”** attribute test, and AND it to the already existing attribute test in the IF part of the rule $IF\ condition_1\ AND\ condition_2\ THEN\ C_i$
- the rule greedily **grows**, until the resulting rule meets an acceptable **quality level**
- the resulting rule is appended to the final rule list

Example



Learn_One_Rule

Example (growing a rule)

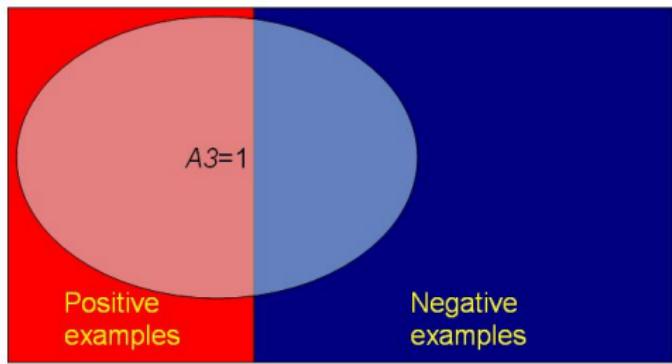
Step 1: IF THEN C_i

Step 2: IF $condition_1$ THEN C_i

Step 3: IF $condition_1$ AND $condition_2$ THEN C_i

Step 4: Quality level reached

Example



Learn_One_Rule

Example (growing a rule)

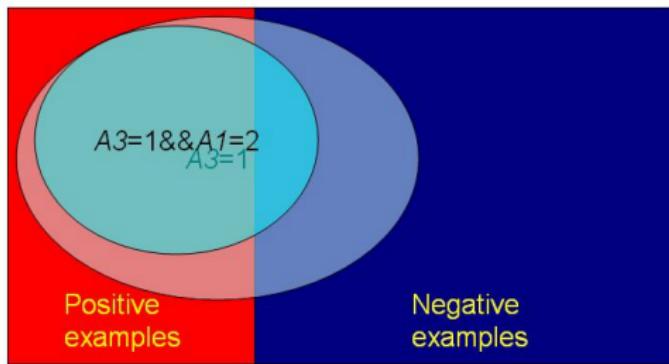
Step 1: IF THEN C_i

Step 2: IF $condition_1$ THEN C_i

Step 3: IF $condition_1$ AND $condition_2$ THEN C_i

Step 4: Quality level reached

Example



Example (growing a rule)

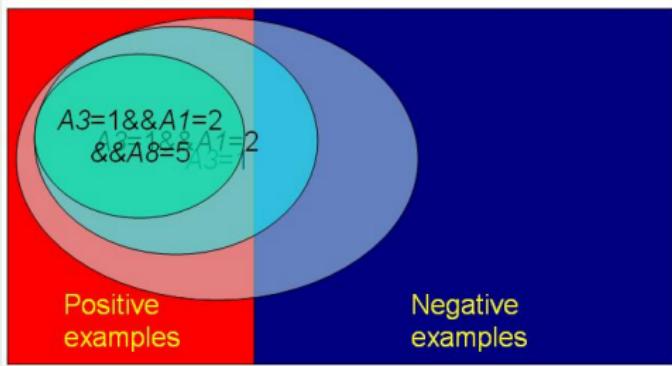
Step 1: IF THEN C_i

Step 2: IF $condition_1$ THEN C_i

Step 3: IF $condition_1$ AND $condition_2$ THEN C_i

Step 4: Quality level reached

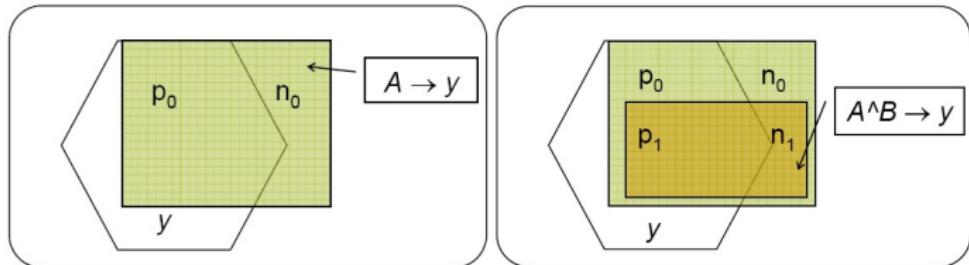
Example



Quality Measure

- one quality measure is the rule **accuracy**
 - however, does not take into account **coverage**
- a different measure: ***FOIL_Gain***
 - R : initial rule
 - R' : rule derived after ANDing an attribute test to R
 - p_0 (n_0): correct (incorrect) classifications of R on D
 - p_1 (n_1): correct (incorrect) classifications of R' on D

$$FOIL_Gain = p_1 \times \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$

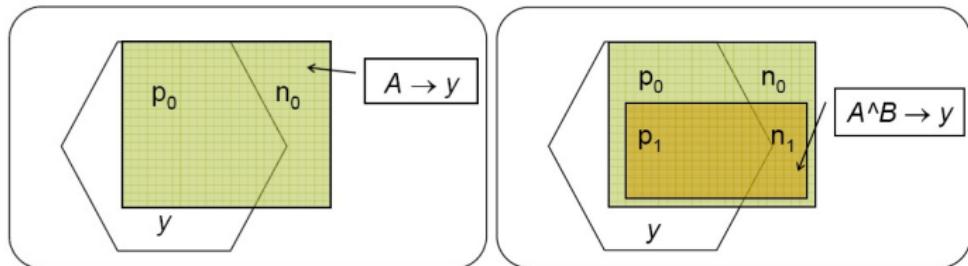


- favor rules with accuracy and cover positive tuples
- choose the attribute test that leads to the **highest *FOIL_Gain***

Quality Measure

- one quality measure is the rule **accuracy**
 - however, does not take into account **coverage**
- a different measure: ***FOIL_Gain***
 - R : initial rule
 - R' : rule derived after ANDing an attribute test to R
 - p_0 (n_0): correct (incorrect) classifications of R on D
 - p_1 (n_1): correct (incorrect) classifications of R' on D

$$FOIL_Gain = p_1 \times \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$

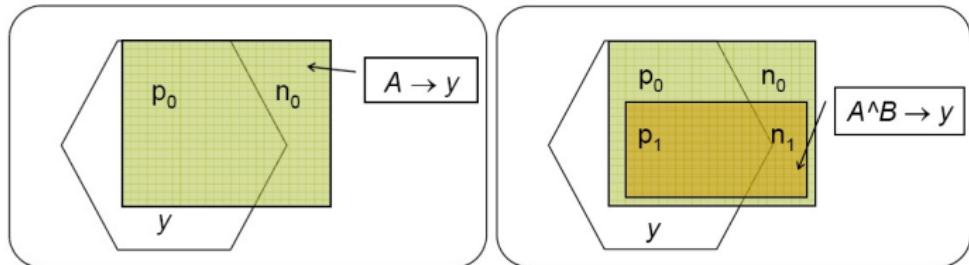


- favor rules with high accuracy and cover positive tuples
- choose the attribute test that leads to the **highest *FOIL_Gain***

Quality Measure

- one quality measure is the rule **accuracy**
 - however, does not take into account **coverage**
- a different measure: ***FOIL_Gain***
 - R : initial rule
 - R' : rule derived after ANDing an attribute test to R
 - p_0 (n_0): correct (incorrect) classifications of R on D
 - p_1 (n_1): correct (incorrect) classifications of R' on D

$$FOIL_Gain = p_1 \times \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$



- favor rules with high accuracy and cover many positive tuples
- choose the attribute test that leads to the **highest *FOIL_Gain***

- similar to rule extraction from decision trees, the resulting rules from the sequential covering algorithm may need pruning

is the resulting rule set exhaustive?

- may not → set up a **default rule**

is the resulting rule set mutually exclusive?

- may not → set up **conflict resolution strategy**

- We explained what rule-based classification is
- We described in detail two methods for learning rule classifiers
 - (indirect) rule extraction from decision trees
 - (direct) rule induction by a sequential covering algorithm

Neural Networks: Perceptrons

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Science Fiction

“Science fiction is the great opportunity to speculate on what could happen” ~ Ray Kurzweil (Director of Engineering, Google)

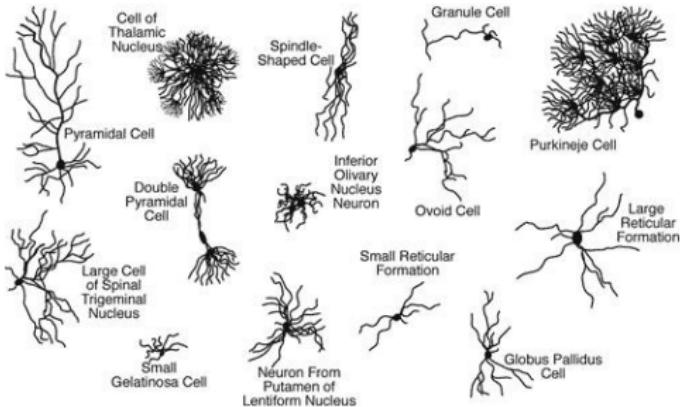
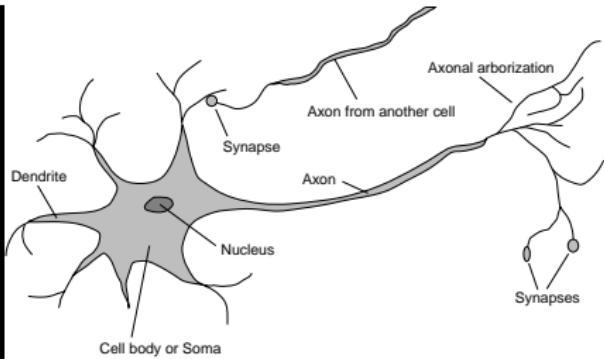
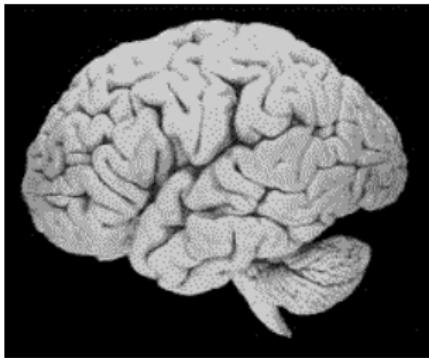


Terminator 2 (1991)

My CPU is a $i+(1)-i$ neural-net processor

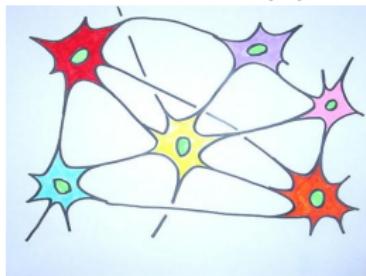
Biological Neurons

Human brain: $i+(1)-i 100,000,000,000$ neurons

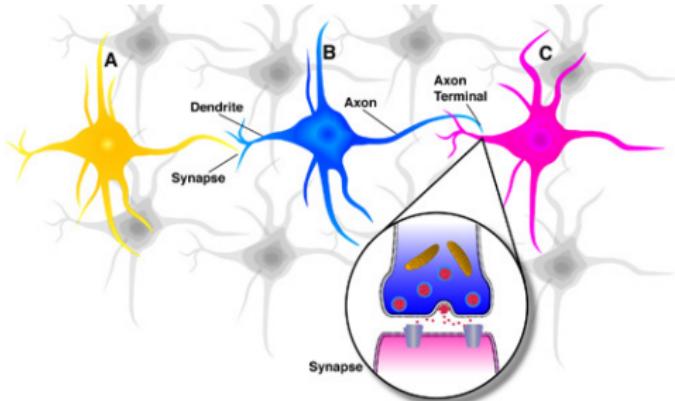


Neuron-to-Neuron

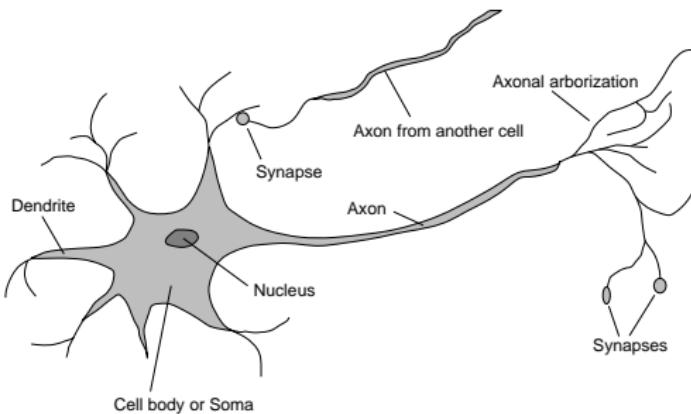
- each neuron receives input from $j+(1)-i$ 1,000 others



- the pulse forms the input to other neurons
- the interface of two neurons is called a **synapse**

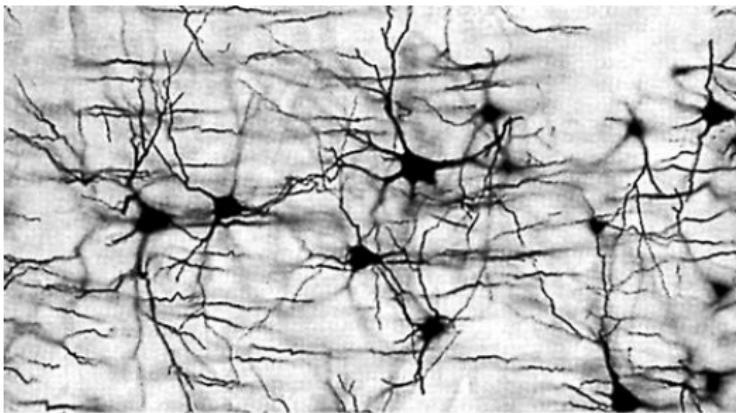


Signal Transmission



- impulses arrive simultaneously
- added together
- chemical transmitter substances are released from the synapses and enter the dendrite
- if sufficiently strong, an electrical pulse is sent down the axon
- the pulse spreads out along the branches of the axon, reaches the synapses and releases transmitters into the bodies of other cells

“Artificial” Neural Networks

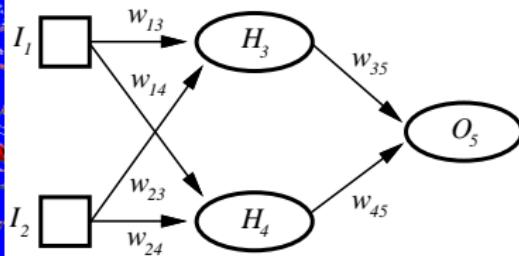
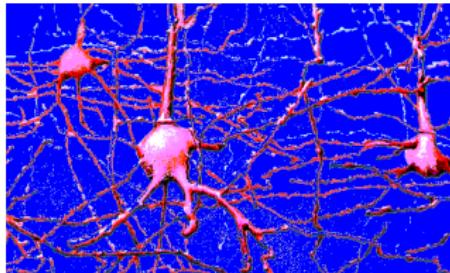




Google official blog on
Using large-scale brain simulations for machine learning and A.I.

Artificial Neural Networks

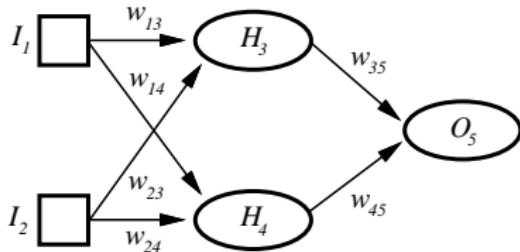
- use complex **networks** of **simple** computing elements as mathematical models to **mimic** the functions of the brain



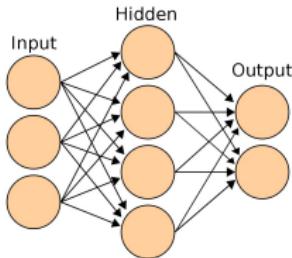
what can artificial neurons in a Google brain learn to detect?



Artificial Neural Networks...



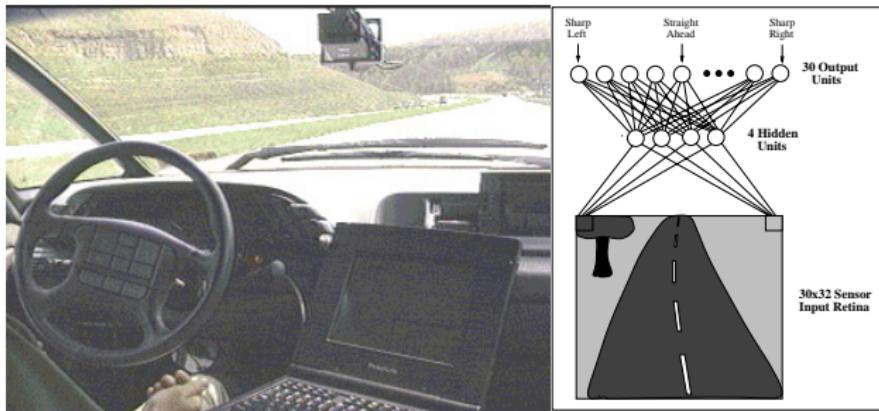
- structure: **Unit, Link, Weight**
 - unit types: **input** units, **hidden** units, **output** units



- learning usually takes place by **updating** the weights

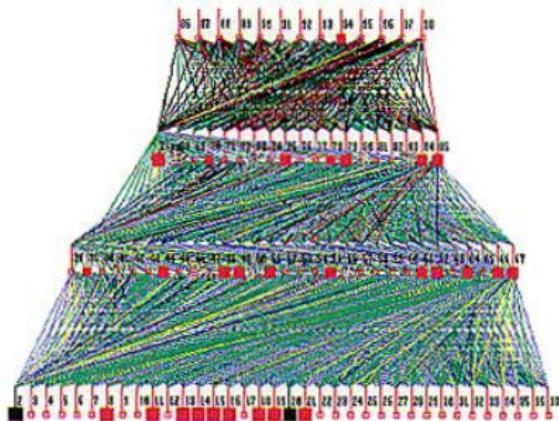
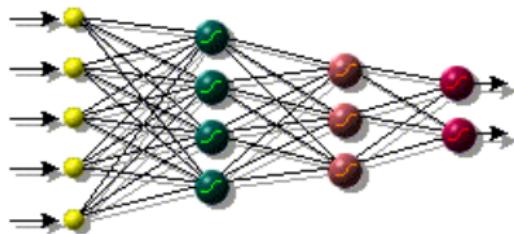
Example Application

ALVINN (Autonomous Land Vehicle In a Neural Network)



- learns to control a vehicle by watching a person drive
- input: video images; output: steering direction

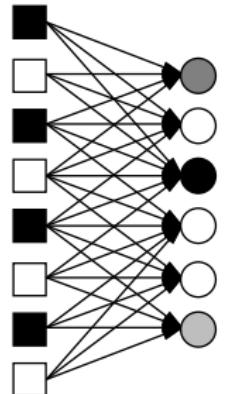
Examples



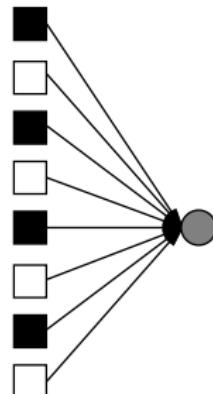
- deep learning (MIT Tech Review)
- Google brain: more than 1 billion connections

Perceptron

a **feed-forward** network with only **one** layer of adjustable (learnable) weights connected to one or more **threshold** units (as output units)



I_j $W_{j,i}$ O_i
Input Units Output Units
Perceptron Network



I_j W_j O
Input Units Output Unit
Single Perceptron

Model

input: I_1, I_2, \dots, I_n

- signals from the other neurons

weights: w_1, w_2, \dots, w_n

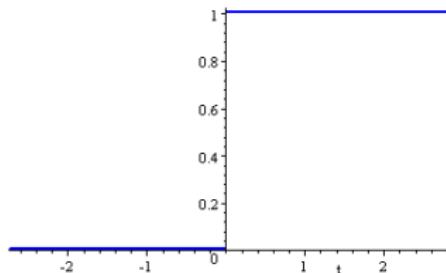
- can be negative

transfer (activation) function:

- relating the input and output

$$O = \text{step}(\sum_{j=1}^n w_j I_j - \theta)$$

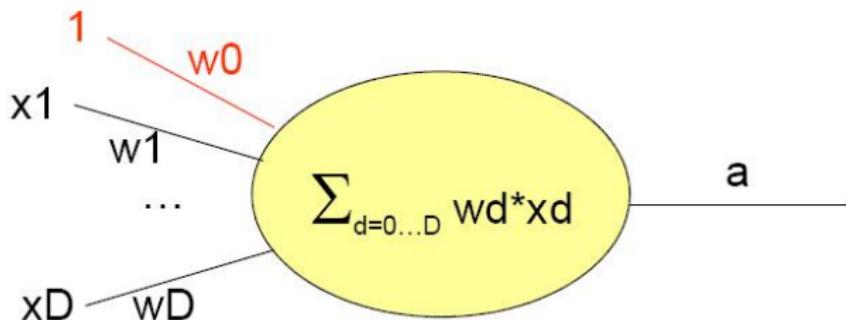
$$\text{step}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \quad (\text{step function})$$



Model...

$$O = \text{step}(\sum_{j=1}^n w_j l_j - \theta)$$

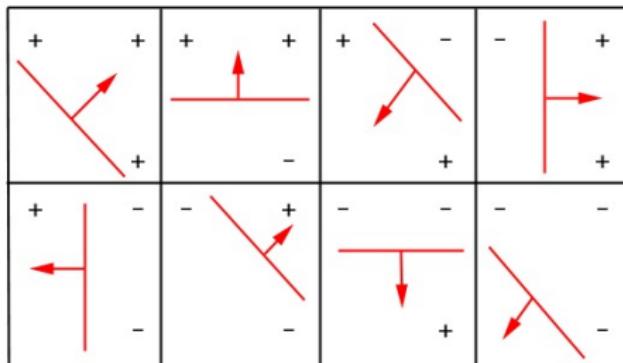
Or, with $w_0 = -\theta$ and $l_0 = 1$



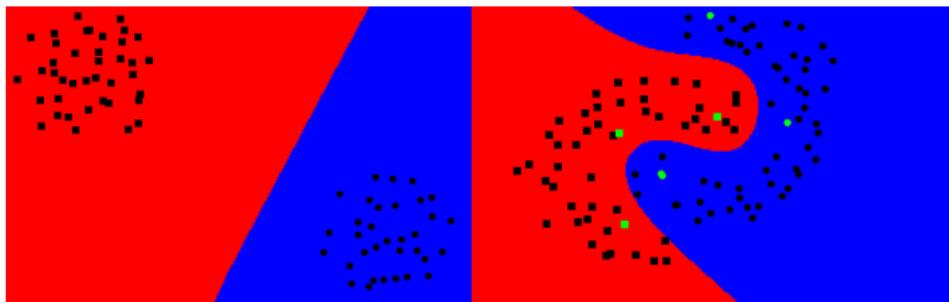
$$O = \text{step} \left(\sum_{j=0}^n w_j l_j \right)$$

Linearly Separable Functions

- a function can be represented by a single perceptron if and only if the function is **linearly separable**



Three points in a plane shattered by a half-space.



Learning Linearly Separable Functions

how to find the appropriate weights?

supervised learning \Rightarrow training examples

Example

apples



not apples

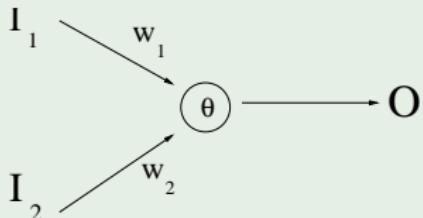


Learning Linearly Separable Functions...

- convert into **features**

Example

	I ₁	I ₂	T
e ₁ :	5	1	0
e ₂ :	2	1	0
e ₃ :	1	1	1
e ₄ :	3	3	1
e ₅ :	4	2	0
e ₆ :	2	3	1



$$O = \text{step}(w_0 + w_1 I_1 + w_2 I_2)$$

Basic Algorithm

- start with some initial values for the weights
- use the perceptron to classify training examples
- modify the weights when errors occur

```
function NEURAL-NETWORK-LEARNING(examples) returns network
    network  $\leftarrow$  a network with randomly assigned weights
    repeat
        for each e in examples do
             $\mathbf{O} \leftarrow$  NEURAL-NETWORK-OUTPUT(network, e)
             $\mathbf{T} \leftarrow$  the observed output values from e
            update the weights in network based on e,  $\mathbf{O}$ , and  $\mathbf{T}$ 
        end
    until all examples correctly predicted or stopping criterion is reached
    return network
```

- an **iterative** algorithm

How to Update the Weights?

idea

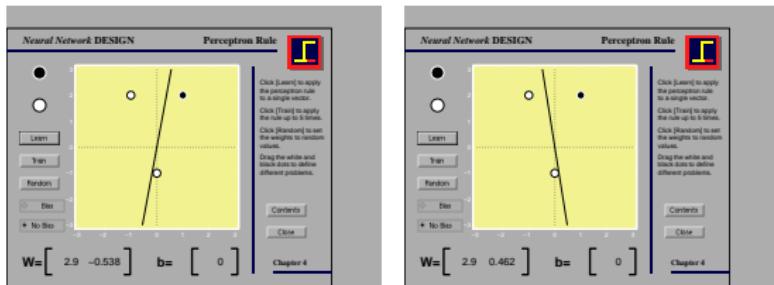
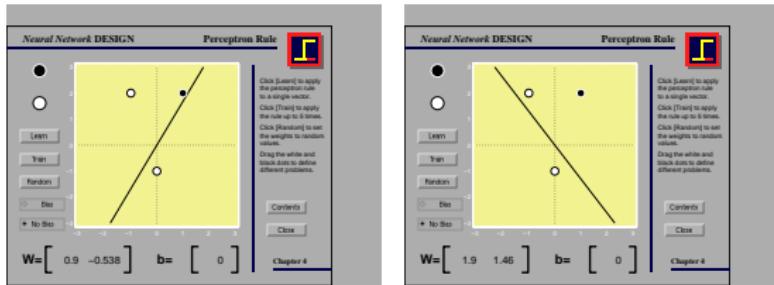
if the **observed output** (T) is different from the **predicted one** (O),
then make **small adjustments** in the weights to reduce the
difference

- if the error $T - O$ is positive, then we need to $\downarrow + (1) - \downarrow$ increase O
- if the error is negative, then we need to $\downarrow + (1) - \downarrow$ decrease O
- each input unit contributes $\downarrow + (1) - \downarrow w_j l_j$ to the total input, so if l_j is positive, an increase in w_j will tend to $\downarrow + (1) - \downarrow$ increase O
- if l_j is negative, an increase in w_j will tend to $\downarrow + (1) - \downarrow$ decrease O

weight update rule

$$w_j \leftarrow w_j + \alpha l_j (T - O), \quad \forall j$$

- α : **learning rate**



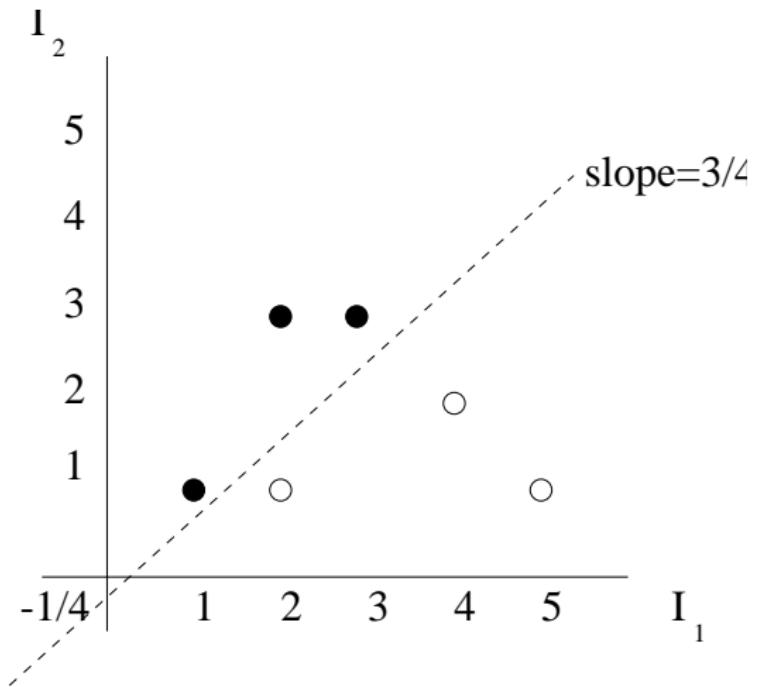
demo

Trace of The Learning Process

Take $\alpha = 1$ and the initial weight values to be 0

Iteration	w^{old}	I	T	O	T = O?	w^{new}
1	(0 0 0)	(1 5 1)	0	1	no	(-1 -5 -1)
2	(-1 -5 -1)	(1 2 1)	0	0	yes	(-1 -5 -1)
3	(-1 -5 -1)	(1 1 1)	1	0	no	(0 -4 0)
4	(0 -4 0)	(1 3 3)	1	0	no	(1 -1 3)
5	(1 -1 3)	(1 4 2)	0	1	no	(0 -5 1)
6	(0 -5 1)	(1 2 3)	1	0	no	(1 -3 4)
7	(1 -3 4)	(1 5 1)	0	0	yes	(1 -3 4)
8	(1 -3 4)	(1 2 1)	0	0	yes	(1 -3 4)
9	(1 -3 4)	(1 1 1)	1	1	yes	(1 -3 4)
10	(1 -3 4)	(1 3 3)	1	1	yes	(1 -3 4)
11	(1 -3 4)	(1 4 2)	0	0	yes	(1 -3 4)
12	(1 -3 4)	(1 2 3)	1	1	yes	(1 -3 4)

Geometric Interpretation of Solution



Neural Networks: Multilayer Perceptrons

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Multi-layer Feedforward Networks

- Generalization of simple perceptrons
- **Multi-layer** perceptrons (MLP)

Output units O_i

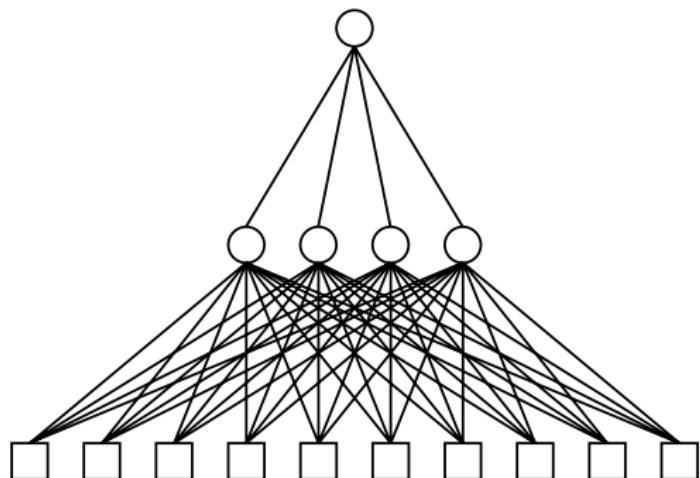
$W_{j,i}$

Hidden units a_j

$W_{k,j}$

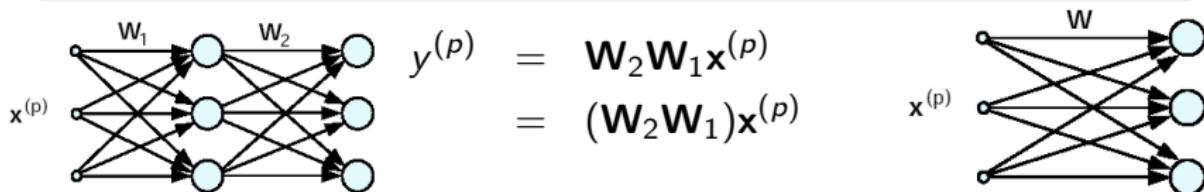
Input units

I_k



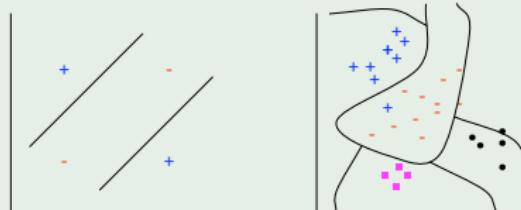
Hidden Unit Transfer Function

if hidden units were linear, then a single-layer neural network with appropriately chosen weights could exactly duplicate those calculations performed by any multi-layer network



- MLPs are capable of expressing a rich variety of nonlinear decision boundaries

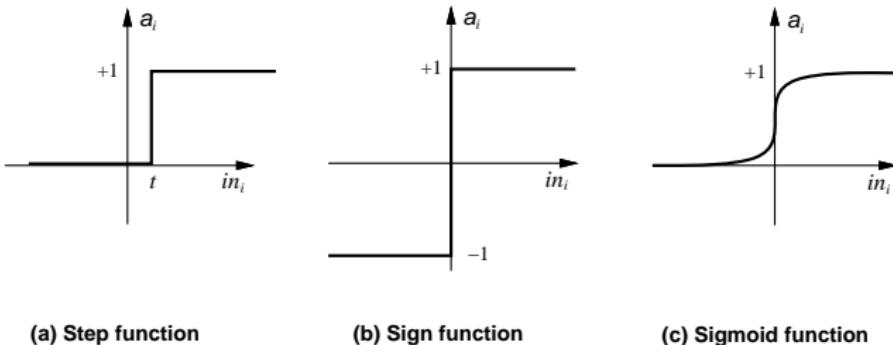
Example



- stems from nonlinearities of the hidden units

Sigmoid Unit

- a unit very much like a perceptron, but based on a **smoothed, differentiable** threshold function: $\sigma(x) = \frac{1}{1+e^{-x}}$

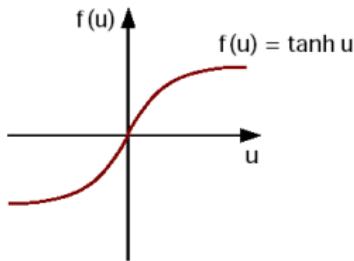


(a) Step function

(b) Sign function

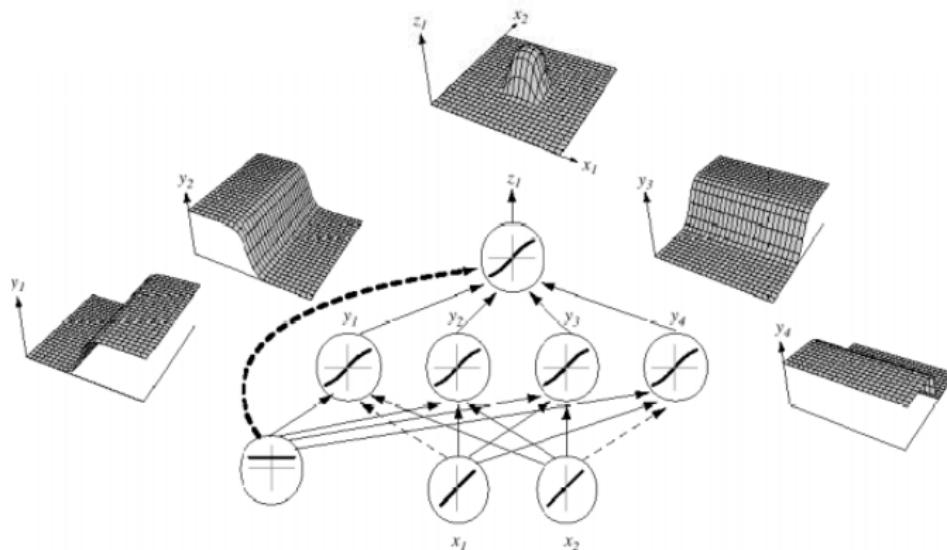
(c) Sigmoid function

- the tanh is also sometimes used in place of the sigmoid function



Universal Approximation

only **one layer** of sigmoid hidden units suffices to **approximate** any well-behaved function to arbitrary precision



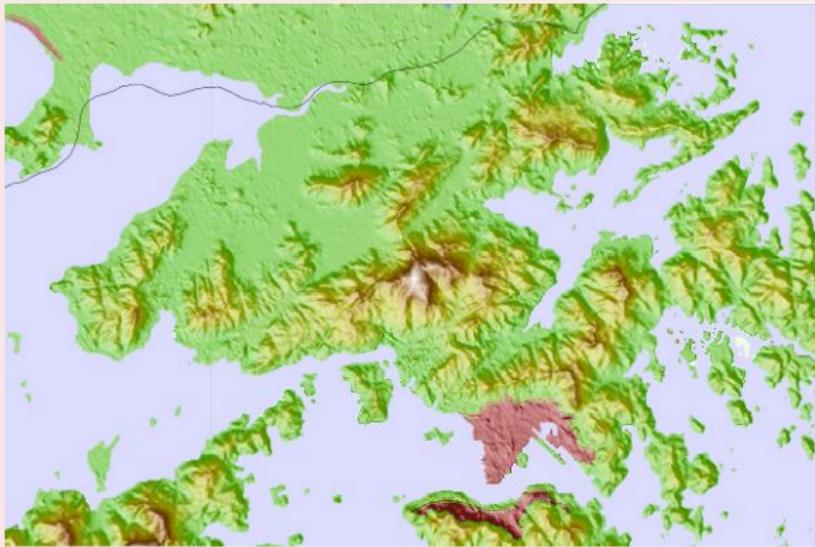
- network with > 2 layers also have universal approximation property

Learning Algorithm

Nonlinear activation functions + multi-layer network

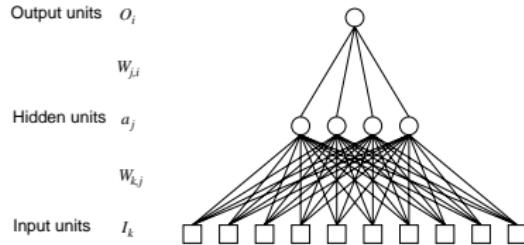
- requires a more sophisticated learning algorithm

How to go to Tai Mo Shan?

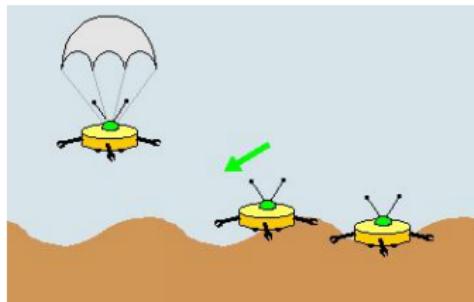


- start at any point and keep going uphill

Finding the Weights

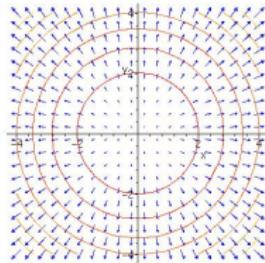
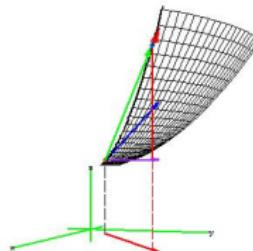


- **error**: a function of the weights
- use **gradient descent** to search the space of possible weight, such that the error is **minimized**



Gradient Descent

- error: $E(\mathbf{w})$
- gradient $\nabla E(\mathbf{w})$ at \mathbf{w} :
$$\left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$



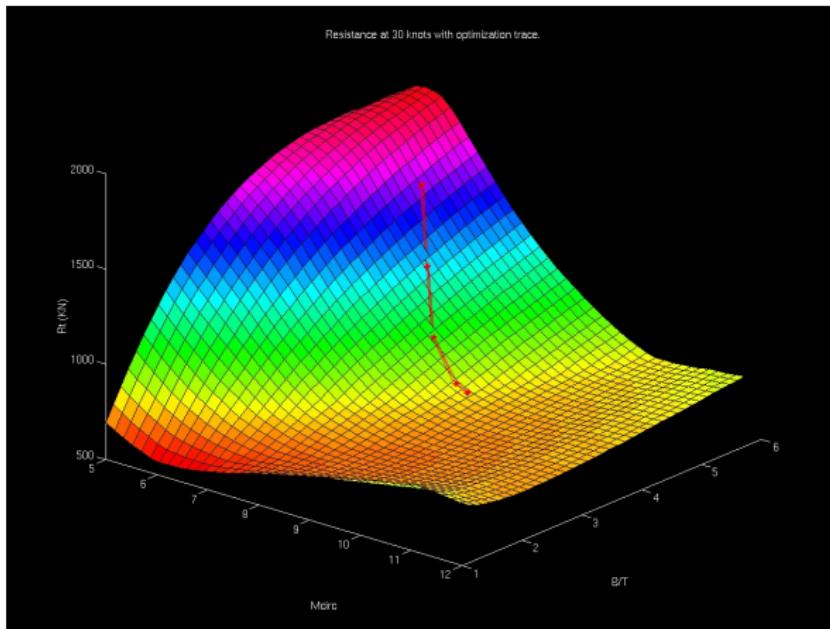
- start with initial value for \mathbf{w}
- repeat until convergence
 - compute the gradient of the error function for current \mathbf{w}
 - move in the opposite direction

move \mathbf{w} :

- **direction:** opposite to $\nabla E(\mathbf{w})$
- **magnitude:** a small fraction of $\nabla E(\mathbf{w})$

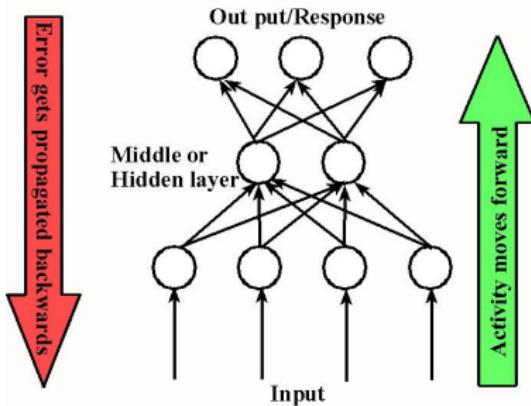
$$\Delta \mathbf{w} = -\eta \nabla E(\mathbf{w}) \quad \text{or} \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Example



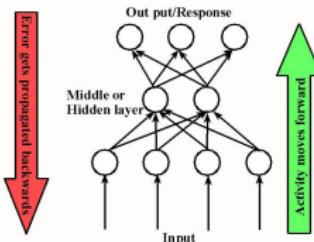
Back-Propagation

- the learning algorithm is more complicated than that of the perceptron
- one popular algorithm is **back-propagation**
 - details not covered in this course



- propagate the **input** **forward** through the network
- propagate the **errors** **backward** through the network in order to compute the gradient vector

Backpropagation Algorithm



Initialize all weights to small random numbers

Until convergence, do

- For each training example, do
 - propagate the **input forward** through the network
 - propagate the **errors backward** through the network
 - for each network weight w_{ji} (weight from i to j)

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

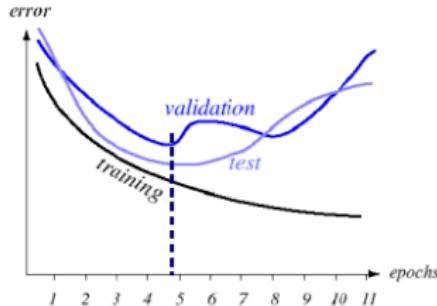
Practical Details

how to initialize the weight values?

- initialize to some small random values

when to stop training?

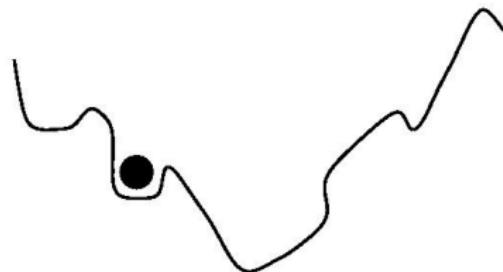
- after a **fixed** number of iterations through the loop
- once the training error falls below some **threshold**
- stop at a minimum of the error on the **validation set**



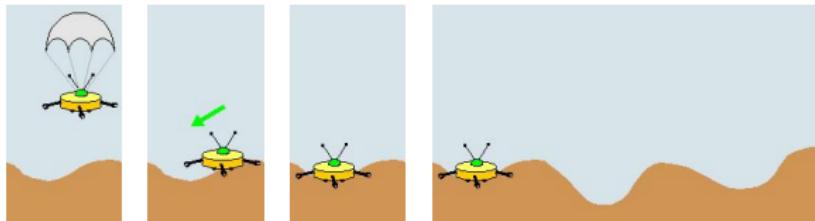
- training can be very slow in networks with multiple hidden layers
- testing is fast

Local Minima

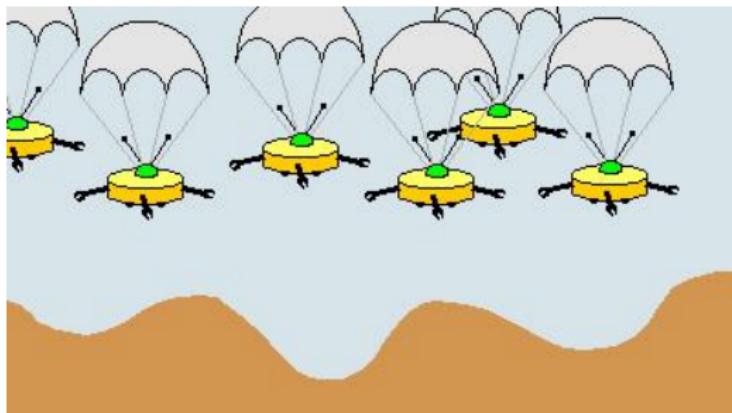
The error surface can have multiple **local minima**



Gradient descent is only guaranteed to converge toward some local minimum, and **not** necessarily to the global minimum



how to escape from locally optimal solutions?

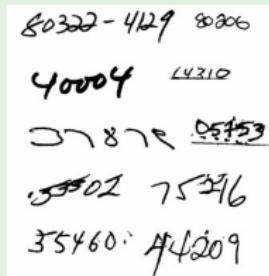


- train **multiple** networks using the same data, but initialize each network with **different** random weights

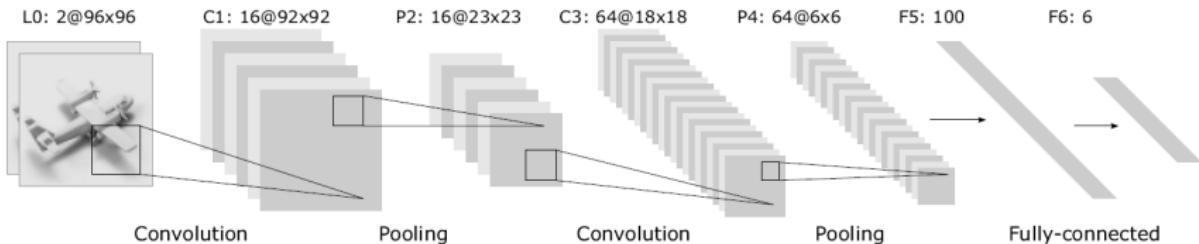
Application: Handwritten Digit Recognition

Example

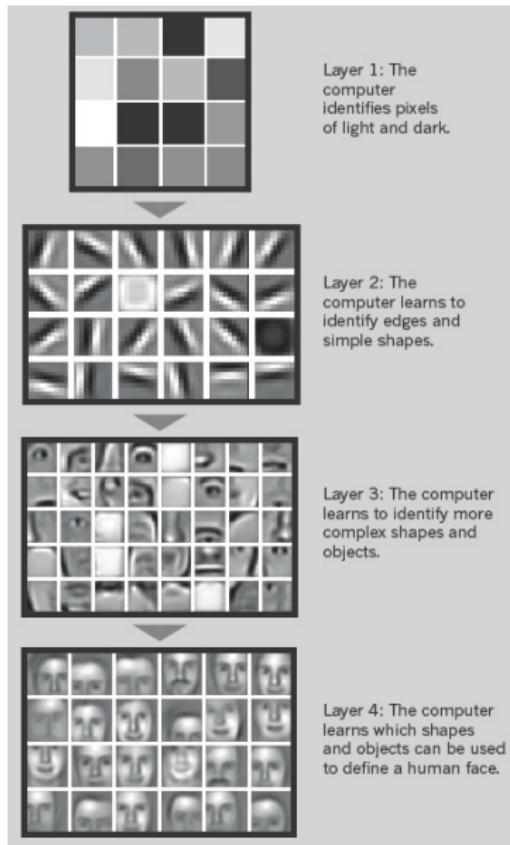
zip code



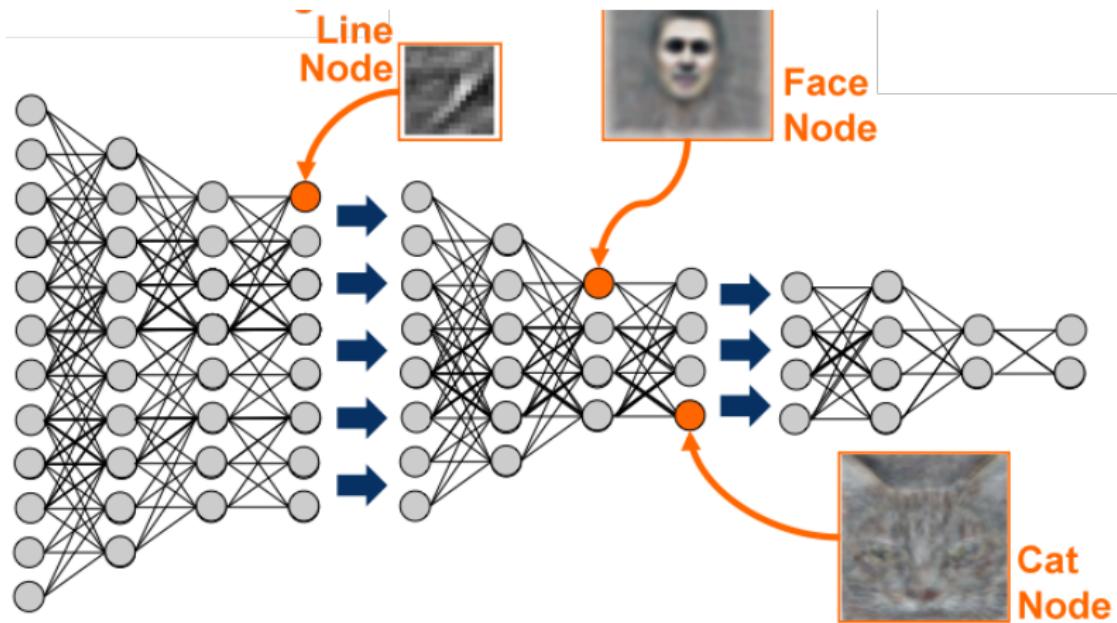
Convolutional neural network



Application: Face Recognition



Application: Image Recognition

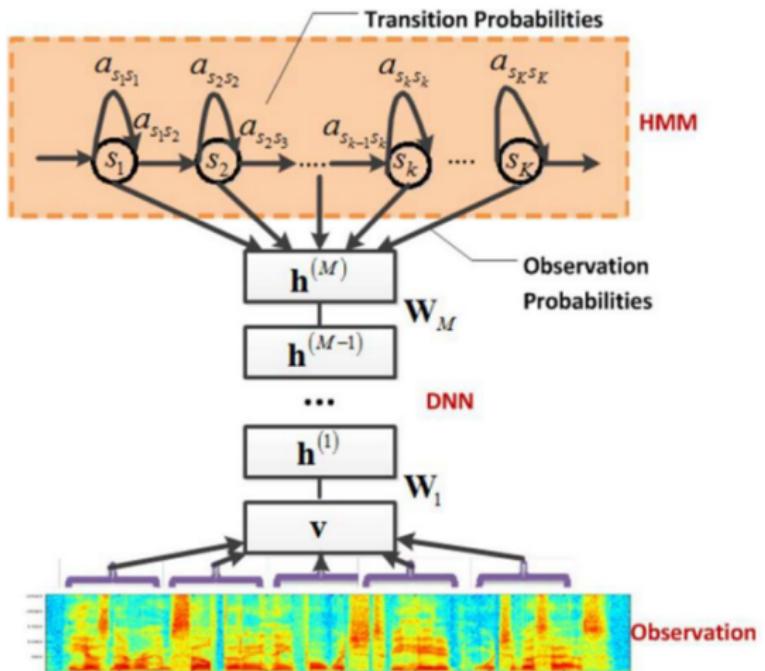


Application: Privacy Protection



- automatically detect and blur faces and license plates
- convolutional neural network
 - input: 64×64 candidate box in the image
 - 2 convolutional layers and 2 sub-sampling layers

Application: Speech Recognition



Dahl, George E., et al. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition." *Audio, Speech, and Language Processing, IEEE Transactions on* 20.1 (2012): 30-42.

Tasks for Which Deep Convolutional Nets are the Best

- handwriting recognition: MNIST
- OCR in the Wild [2011]: StreetView House Numbers
- traffic sign recognition [2011]
- pedestrian detection [2013]
- volumetric brain image segmentation [2009]
- human action recognition [2011]
- object recognition [2012]: ImageNet competition
- scene parsing [2012]:
- scene parsing from depth images [2013]
- speech recognition [2012] (IBM and Google)
- breast cancer cell mitosis detection [2011]

Summary

- we have introduced some basic notions on Artificial Neural Networks (ANNs)
- we described in more detail a simple ANN model, called perceptron, and explained learning and classification using this model
- we briefly discussed a more complicated ANN model, called multilayer perceptron

Support Vector Machines

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

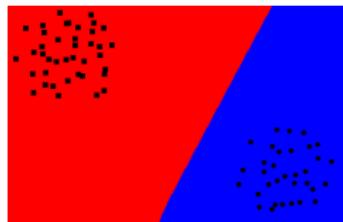
Classification Problem

training set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$

- (\mathbf{x}_i, y_i) : training pattern
- $\mathbf{x}_i \in \mathbb{R}^m$: input
- $y_i \in \{\pm 1\}$: output (label) (in general, can have > 2 classes)

assume that the problem is **linearly separable**

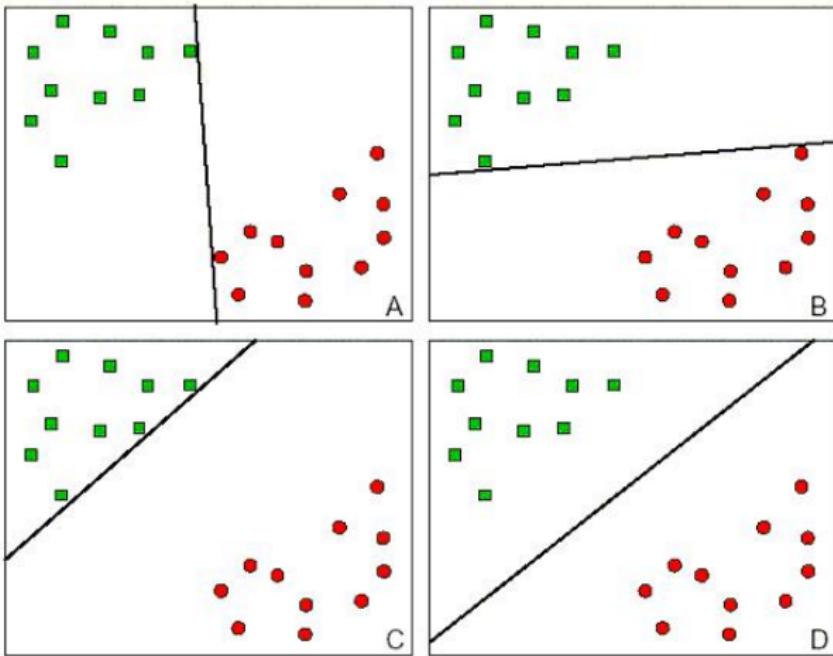
- there exists a linear surface to separate the two classes



- 2-D: line; 3-D: plane; ...; n-D: **hyperplane** ($\mathbf{w}'\mathbf{x} + b = 0$)
 - $\mathbf{w} = (w_1, w_2, \dots, w_n)$: **weight** vector
 - b : scalar

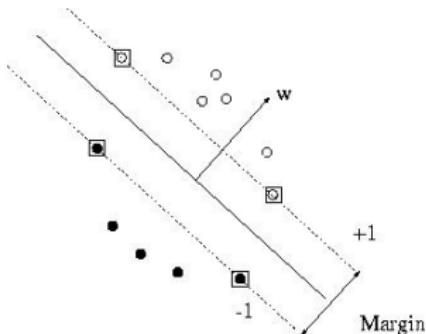
Find hyperplane that perfectly separates the two classes

Multiple Solutions

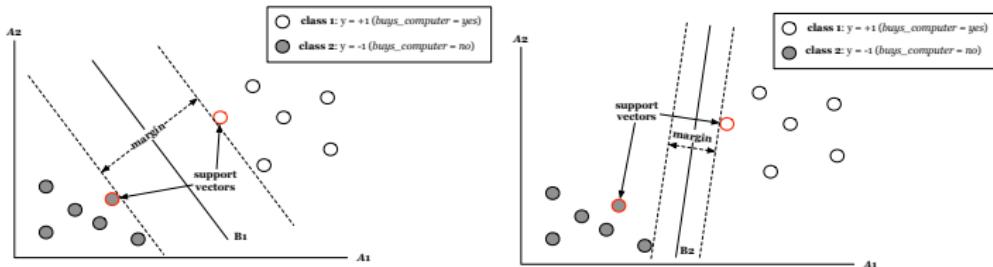


which one is the best?

Margin

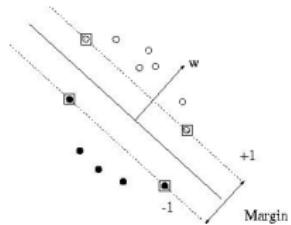


- different decision boundaries have different margins

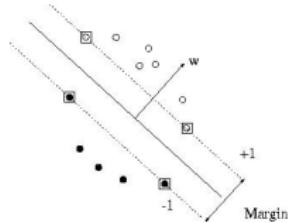


Goal: Find the decision boundary with **maximum margin**

Training



Training



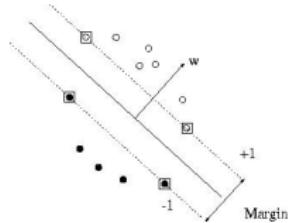
$$\text{margin} = \frac{\mathbf{w}'}{\|\mathbf{w}\|} (\mathbf{x}_1 - \mathbf{x}_2) = \frac{2}{\|\mathbf{w}\|}$$

the hyperplane should **separate** the two classes

$$\mathbf{w}'\mathbf{x} + b \begin{cases} & \text{if } y_i = 1 \\ & \text{if } y_i = -1 \end{cases} \quad \text{or, equivalently } y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1$$

$$\min \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to } y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, n$$

Training



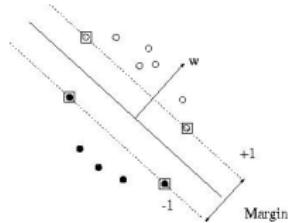
$$\text{margin} = \frac{\mathbf{w}'}{\|\mathbf{w}\|} (\mathbf{x}_1 - \mathbf{x}_2) = \frac{2}{\|\mathbf{w}\|}$$

the hyperplane should **separate** the two classes

$$\mathbf{w}'\mathbf{x} + b \begin{cases} \geq 1 & \text{if } y_i = 1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \quad \text{or, equivalently } y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1$$

$$\min \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to } y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, n$$

Training



$$\text{margin} = \frac{\mathbf{w}'}{\|\mathbf{w}\|} (\mathbf{x}_1 - \mathbf{x}_2) = \frac{2}{\|\mathbf{w}\|}$$

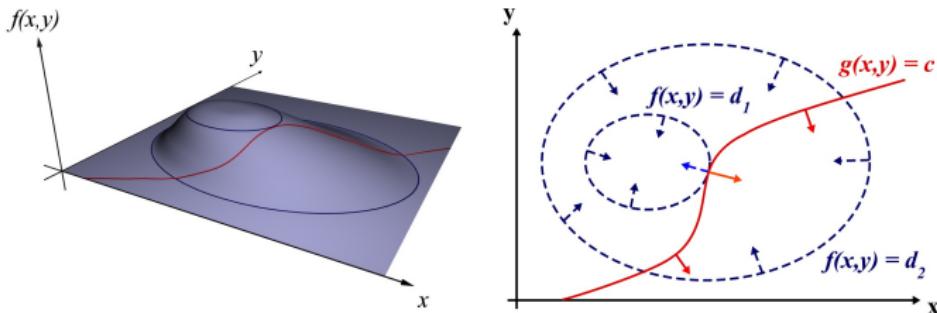
the hyperplane should **separate** the two classes

$$\mathbf{w}'\mathbf{x} + b \begin{cases} \geq 1 & \text{if } y_i = 1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \quad \text{or, equivalently } y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1$$

$$\min \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to } y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, n$$

$$\min \frac{1}{2} \|\mathbf{w}\|^2 \text{ subject to } y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, n$$

constrained optimization

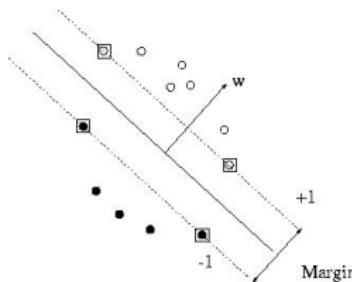


- each **constraint** is associated with a **Lagrange multiplier**
 - $y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1 \leftrightarrow \alpha_i$
 - one pattern (\mathbf{x}_i, y_i) \leftrightarrow one constraint \leftrightarrow one Lagrange multiplier α_i
- can be solved by many optimization packages (e.g. MATLAB optimization toolbox)
 - can then obtain \mathbf{w} and $\alpha_1, \alpha_2, \dots, \alpha_n$
- can be shown that $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$
 - α_i large $\rightarrow (\mathbf{x}_i, y_i)$ important

Support Vectors

after solving the optimization problem, we obtain

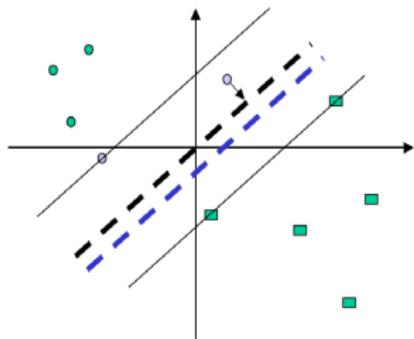
$$f(\mathbf{x}) = \mathbf{w}'\mathbf{x} + b = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i' \mathbf{x} + b$$



- patterns for which $y_i(\mathbf{w}'\mathbf{x}_i + b) > 1$
 - \mathbf{x}_i irrelevant
 - can be shown that $\alpha_i = 0$
- patterns that have $\alpha_i > 0$
 - can be shown that $y_i(\mathbf{w}'\mathbf{x}_i + b) = 1$
 - lie either on H_1 or H_2
- solution is determined by the examples on the margin
(support vectors)
 - critical elements of the training set

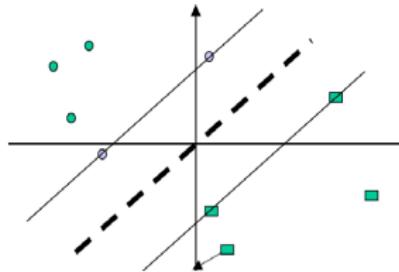
Support Vectors (SVs)...

- if support vectors are moved or changed
 - position of the hyperplane solution would be changed
- if other training points (non-SVs) are removed or moved around, and training repeated
 - the same hyperplane would be found

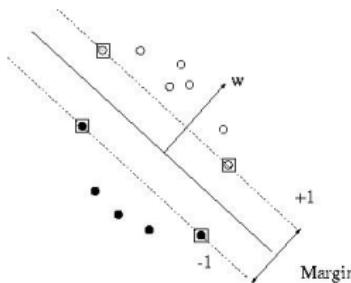


Moving a support vector moves the decision boundary

Moving the other vectors has no effect



How to Perform Testing?

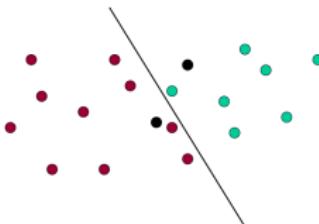


- use $\text{sign}(\mathbf{w}'\mathbf{x} + b)$ or $\text{sign}(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i' \mathbf{x} + b)$

$$\text{sign} \left(\sum_{i=1}^{n_s} \alpha_i y_i \mathbf{x}_i' \mathbf{x} + b \right) = \text{sign} \left(\sum_{i=1}^{n_s} \alpha_i y_i \mathbf{x}_i' \mathbf{x} + b \right)$$

- n_s : number of support vectors

What if Training Data not Linearly Separable?



separate the training set with a minimum number of **errors**

hard-margin SVM

$$\min \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to } y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, n$$

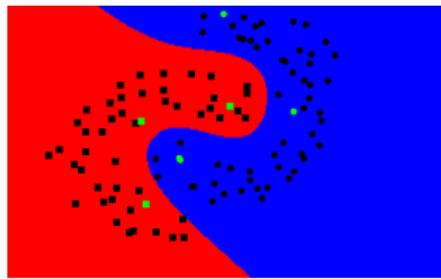
soft-margin SVM

$$\min \frac{1}{\text{margin}} + C \text{ error}$$

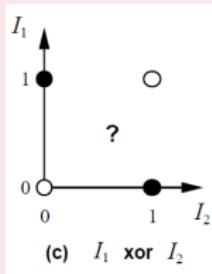
subject to the constraints

- still a similar optimization problem

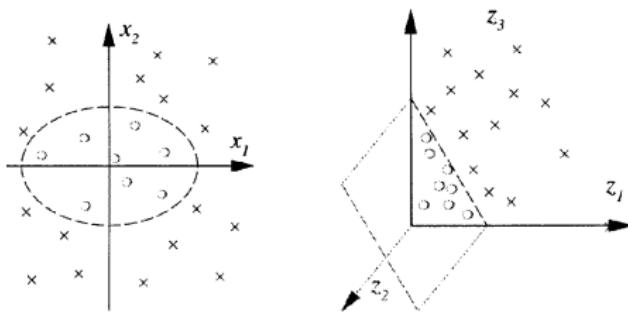
Nonlinear Decision Surface



What about mapping the data to a higher-dimensional space?



Nonlinear Decision Surface...



- ① map the data to a **higher-dimensional space**
 - $\mathbf{x} \mapsto \varphi(\mathbf{x})$
- ② linearly separate them in the new space

Example: All Degree 2 Monomials in \mathbb{R}^2

- $\mathbf{x} = (x_1, x_2)$ becomes $\varphi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$
- $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ becomes $(\varphi(\mathbf{x}_1), y_1), \dots, (\varphi(\mathbf{x}_n), y_n)$
- $\mathbf{x} \mapsto y$ becomes $\varphi(\mathbf{x}) \mapsto y$
- $f = \mathbf{w}'\mathbf{x} + b$ becomes
$$f = \mathbf{w}'\varphi(\mathbf{x}) + b = w_1x_1^2 + \sqrt{2}w_2x_1x_2 + w_3x_2^2 + b$$

Example

- create a d th order polynomial in the input space \mathbb{R}^m
- for $m = 256, d = 5 \rightarrow$ dimensionality 10^{10}

Inner Products

- recall that the prediction function is

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i' \mathbf{x} + b$$

why should inner products be involved?

- intuitively, they provide some measure of **similarity**
- when the data is mapped to another space, inner product will be **changed**

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \varphi(\mathbf{x}_i)' \varphi(\mathbf{x}) + b$$

Example (degree 2 case)

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^2, d = 2$$

$$\begin{aligned}\varphi(\mathbf{x})' \varphi(\mathbf{y}) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)'(y_1^2, \sqrt{2}y_1y_2, y_2^2) \\ &= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \\ &= (x_1y_1 + x_2y_2)^2 \\ &= (\mathbf{x}'\mathbf{y})^2\end{aligned}$$

- dot product can be computed in \mathbb{R}^2

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \varphi(\mathbf{x}_i)' \varphi(\mathbf{x}) + b = \sum_{i=1}^n \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

- $k(\mathbf{x}_i, \mathbf{x}) = (\mathbf{x}'\mathbf{y})^2$
- mapping is defined by a **kernel** function $k(\cdot, \cdot)$

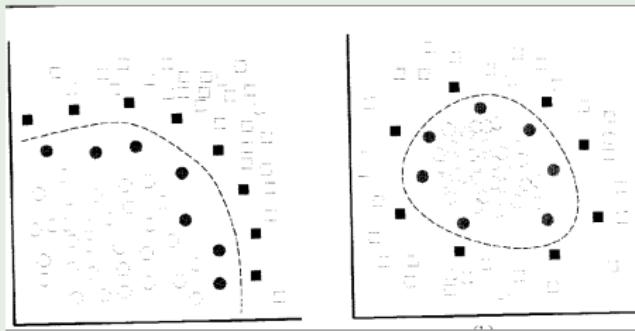
Kernel Trick

- kernel: substitutes the dot product and act as a nonlinear similarity measure
- common kernels
 - **polynomial**: $k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}'_1 \mathbf{x}_2)^d$ or $k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}'_1 \mathbf{x}_2 + 1)^d$
 - corresponds to d th-order polynomial feature space
 - **Gaussian**: $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2 / (2\sigma^2))$
 - σ : controls the width of the Gaussian
 - corresponds to an infinite-dimensional feature space

Any algorithm that depends only on dot products can use the kernel trick!

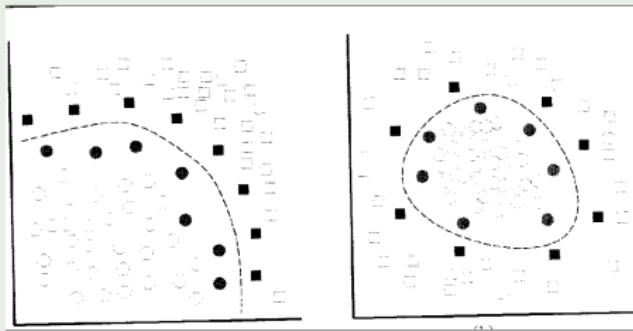
Example

Example



Example

Example



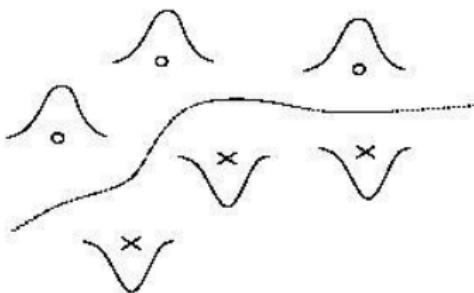
- usually support vectors are very few in number

Gaussian Kernel

recall that the decision rule uses

$$\sum_{i=1}^n \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

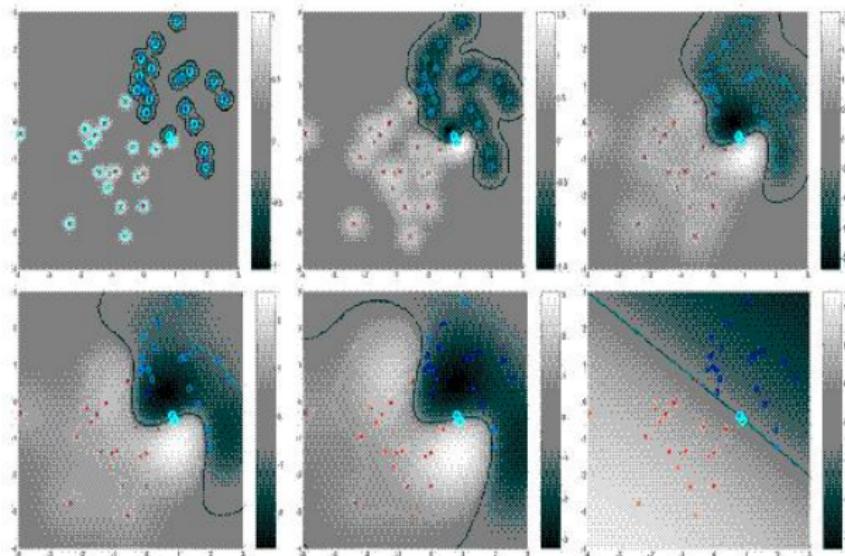
- Gaussian kernel: $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2/2\sigma^2)$
- amounts to putting **bumps** of various sizes on the training set



Varying σ

Example: Two overlapping Gaussians belonging to two classes with means $(-1, -1)$ and $(1, 1)$ and standard deviation 1

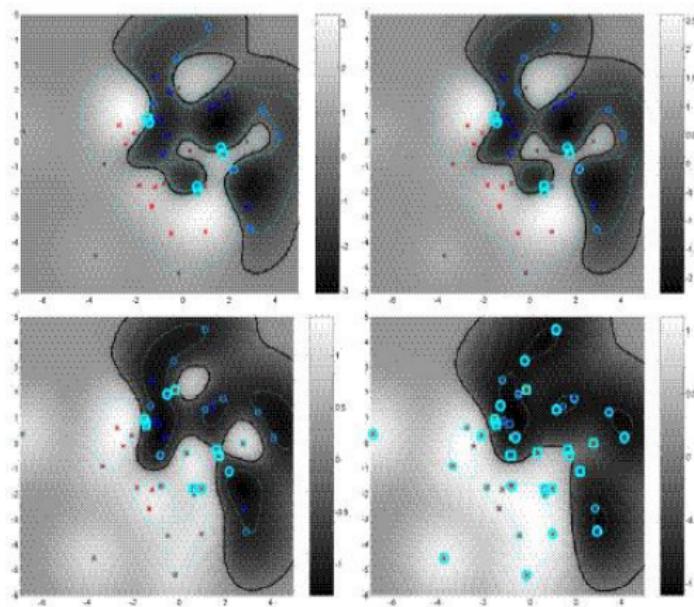
- $\sigma = 0.1, 0.25, 0.5, 0.75, 1.10$



Overfitting?

Varying C

$(\sigma^* = 2, \sigma = 1)$ $C = \inf, 100, 10, 1$



Overfitting?

Summary

- We explained the Support Vector Machine (SVM) classifier

Ensemble Methods

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Motivation

Example

- you are suspected of having a serious illness
- instead of going to a **single** doctor, you consult **multiple** doctors
- if a particular diagnosis occurs more than others → final diagnosis

why?

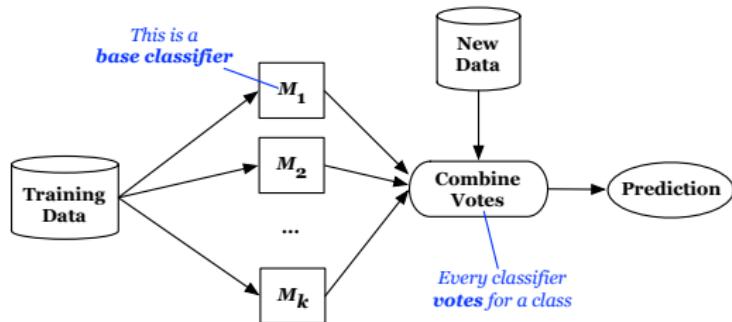
- hypotheses (doctors) are not perfect!
- hypotheses combination → increased accuracy

additionally, you may also **weight** the hypotheses

- assign weights to every doctor, based on the accuracy of their previous diagnoses (on old patients)
- the final diagnosis is based on a combination of these weighted diagnoses

Ensemble

Basic Idea of Ensemble Methods



- improve the performance by aggregating **multiple** classifiers



Ensemble for Classification

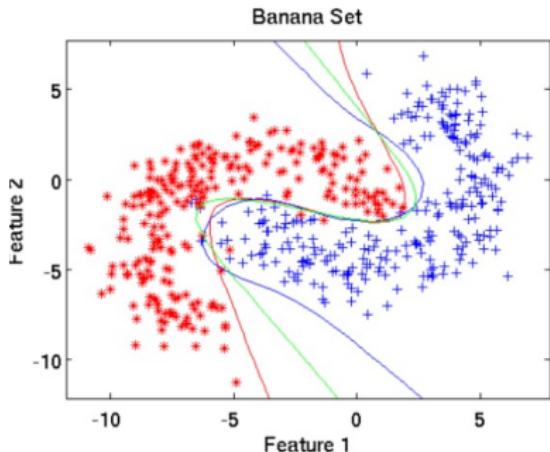
ensemble for **binary** classification consists of

- **hypotheses** $\{h_t(\mathbf{x}) : t = 1, \dots, T\}$, where each hypothesis output ± 1

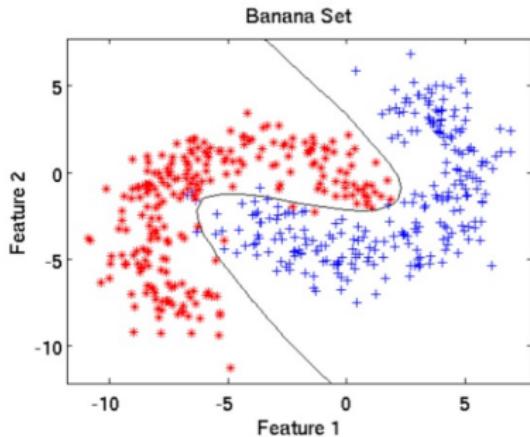
- **weights** $\alpha_1, \alpha_2, \dots, \alpha_T$ where $\alpha_t \geq 0$

output: **weighted majority of the votes**

- $f_{\text{ens}}(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$



Three neural nets generated with default settings [bp, xnc]



Final output from bagging 10 neural nets

Why Does It Work?

Example (two-class problem)

- suppose there are 25 base classifiers
- every base classifier has **error rate** $\epsilon_m = 0.35$
- the ensemble classifier predicts the class label of a test tuple by taking the majority vote of the base classifiers
- assume that the classifiers are **independent**
- probability that the ensemble classifier makes a wrong prediction is:

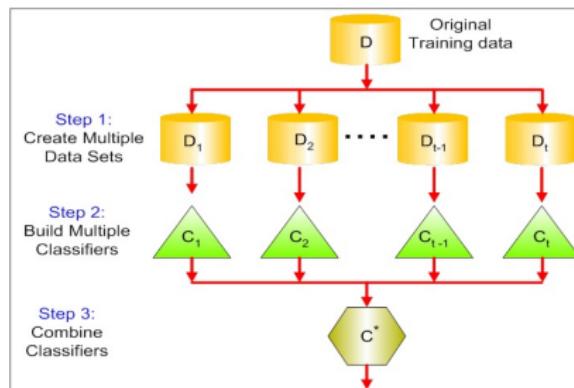
$$\epsilon_{ens} = \sum_{i=13}^{25} \binom{25}{i} \epsilon_m^i (1 - \epsilon_m)^{25-i} = 0.06$$

- $\epsilon_{ens} = 0.06 < \epsilon_m = 0.35$

How to Construct an Ensemble Classifier?

- ① how to generate different hypotheses?
- ② how to combine them (the weights)?

- by manipulating the training set
 - bagging, boosting



- by manipulating the learning algorithm
 - e.g., by changing the **parameters** of the algorithm (e.g., k in knn classification, the network topology in anns, etc.)
- ...

Bagging (Bootstrap aggregating)

for a dataset with N examples

- create the sample (**bootstrap samples**) by sampling the original data set **with replacement**

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- each example has a probability of $1 - (1 - \frac{1}{N})^N$ of being selected **at least once** in the N samples
- for $N \rightarrow \infty$, this number converges to $1 - \frac{1}{e}$, or 0.632
- as a result, each classifier is trained on the average of 63.2% of the training examples

Example (with a simple rule-based classifier)

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \Rightarrow y = 1$

$x > 0.35 \Rightarrow y = -1$

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

$x \leq 0.65 \Rightarrow y = 1$

$x > 0.65 \Rightarrow y = 1$

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.35 \Rightarrow y = 1$

$x > 0.35 \Rightarrow y = -1$

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.3 \Rightarrow y = 1$

$x > 0.3 \Rightarrow y = -1$

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.35 \Rightarrow y = 1$

$x > 0.35 \Rightarrow y = -1$

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \Rightarrow y = -1$

$x > 0.75 \Rightarrow y = 1$

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

$x \leq 0.75 \Rightarrow y = -1$

$x > 0.75 \Rightarrow y = 1$

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \Rightarrow y = -1$

$x > 0.75 \Rightarrow y = 1$

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \Rightarrow y = -1$

$x > 0.75 \Rightarrow y = 1$

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

$x \leq 0.05 \Rightarrow y = -1$

$x > 0.05 \Rightarrow y = 1$

Figure 5.35. Example of bagging.

Example...

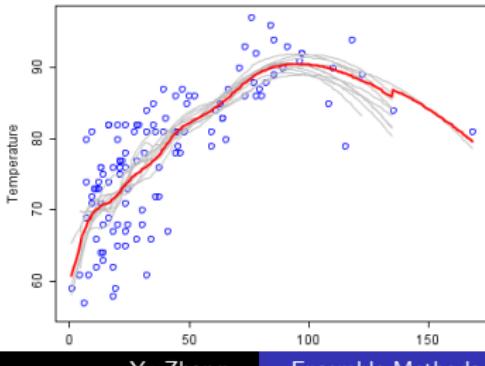
Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

Figure 5.36. Example of combining classifiers constructed using the bagging approach.

- accuracy of ensemble classifier: 100%

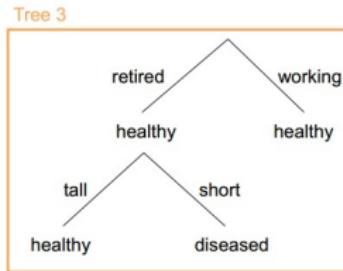
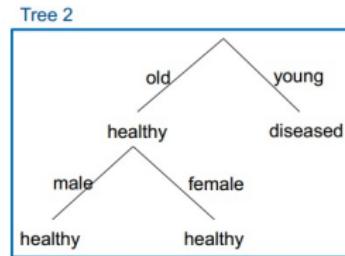
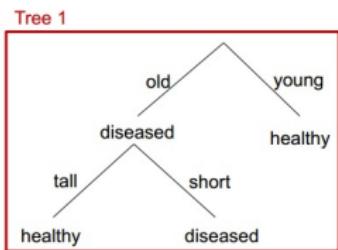
Remarks

- different bootstrap samples → **different** hypotheses to be built
- **unstable** classifier: a small change in the training data can lead to a radically different hypothesis
 - e.g., neural networks and decision trees
 - bagging will have little benefit when used with stable base learning algorithms (i.e., most base learners in the ensemble will be very similar)
 - bagging works particularly well if base classifiers are unstable
- bagging can also be applied to regression problems
 - for a given test tuple, take **average** of the predictions



Random Forests

- ensemble method specifically designed for decision tree classifiers
- grows many trees (unpruned)
- on classifying a new instance: use voting (over all the trees in the forest)



New sample:

old, retired, male, short

Tree predictions:

diseased, healthy, diseased

Majority rule:
diseased

Random Forests...

introduce **two** sources of randomness

- ① bagging: each tree is grown using a bootstrap sample of training data
- ② node splitting
 - method 1
 - randomly select m attributes among all attributes
 - compute information gain, pick the attribute with the large gain to split
 - method 2
 - compute information gain for all attributes
 - select the top m attributes by information gain
 - randomly select one of the m attributes as the splitting node

Trees vs Random Forests

	trees	random forests
speed	fast	slow
insight into decision rules	yes	hard
accuracy	lower	higher

Boosting

bagging

- classifiers are trained independently
- maintains a **constant** probability (of $1/N$) for selecting each example

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

boosting

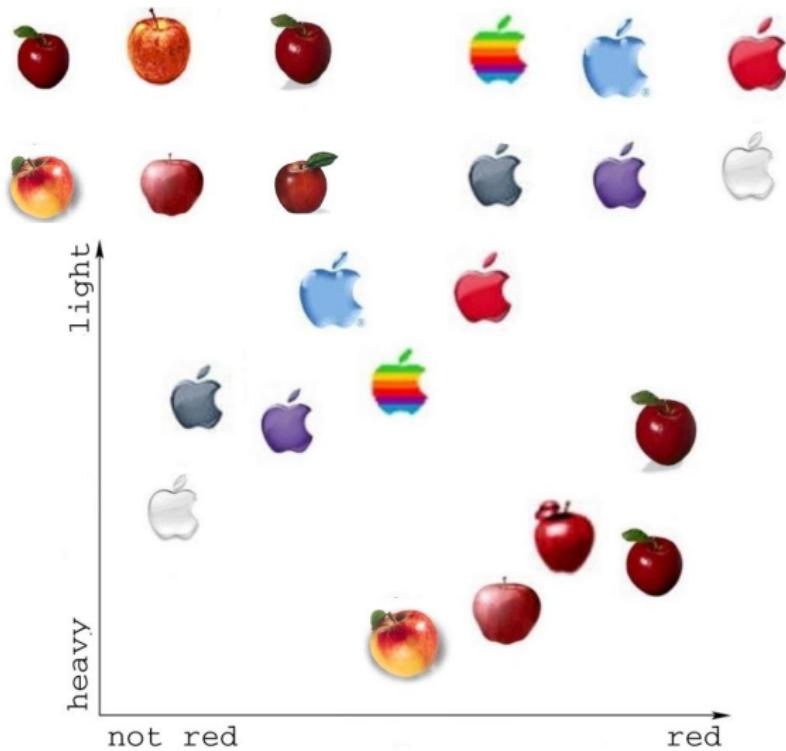
- base classifiers are built one by one
- example selection probability is **adapted** over time based on performance
 - focus more on **previously misclassified examples** ("hard" examples)

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

Boosting...

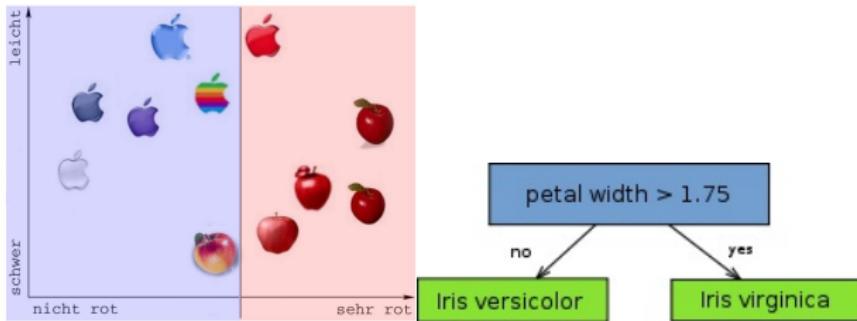
```
for hypothesis  $h_1, h_2, \dots, h_T$  do
    compute weight distribution on examples;
    find hypothesis  $h_t$  on weighted training samples;
    compute weight  $\alpha_t$  for hypothesis  $h_t$  (a function of its
    accuracy)
end
combine hypotheses:  $\sum_{t=1}^T \alpha_t h_t$ 
```

Example



Example...

- a series of classifiers are iteratively learned
- simple hypotheses (weak learner)
 - e.g., cuts on coordinate axes

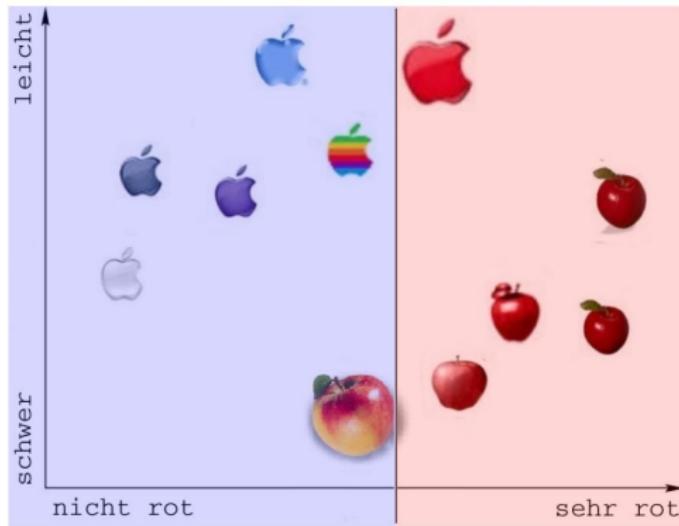


- decision stump



Recompute Weighting

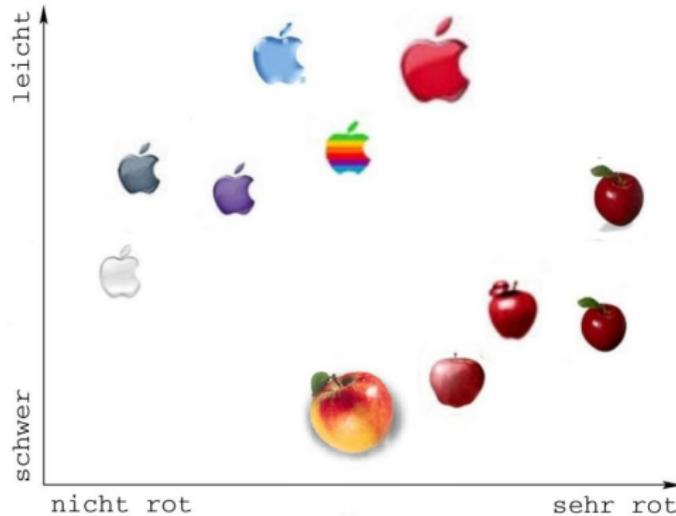
- weights are assigned to each training tuple



- after a classifier h_t is learned, the weights are updated to allow the subsequent classifier, h_{t+1} , to pay more attention to the training tuples that were misclassified by h_t

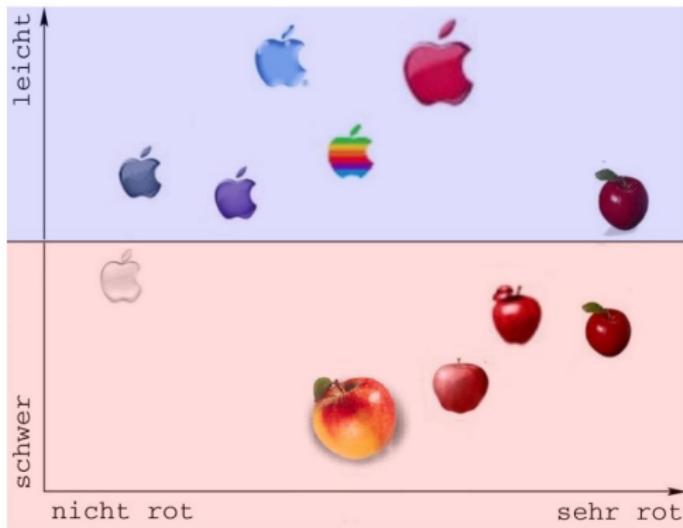
Example

Boosting: 2nd iteration



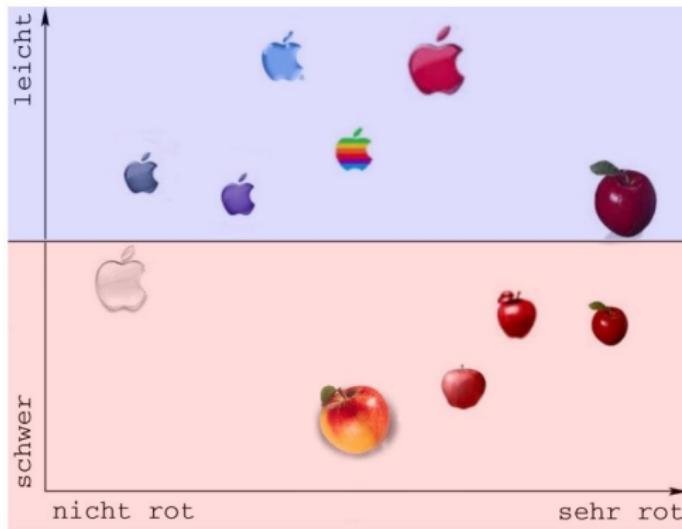
Example

Boosting: 2nd hypothesis



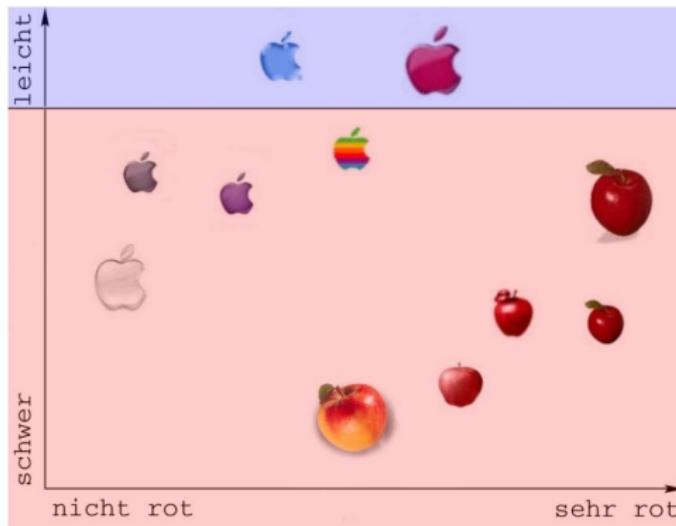
Example

Recompute weighting



Example

Boosting: 3rd hypothesis



Example

Boosting: 4th hypothesis

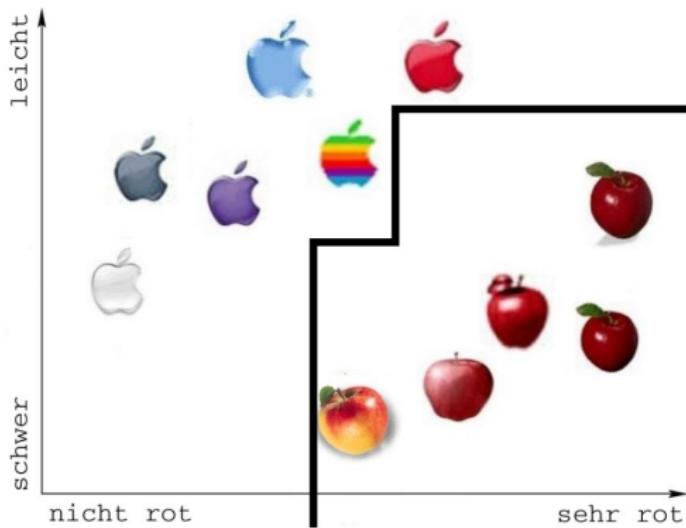


Combination of Hypotheses



- the final h^* combines the votes of each individual classifier, where the **weight of each classifier's vote** is a function of its accuracy

Decision

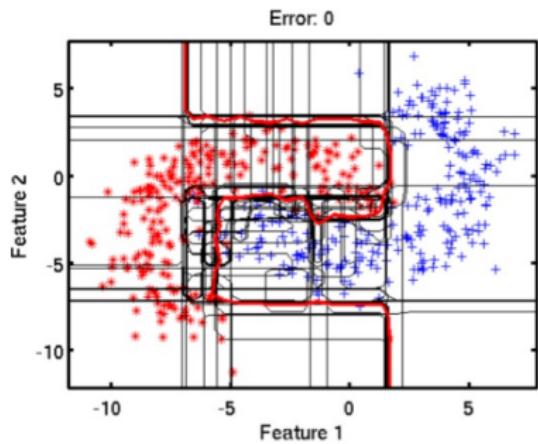


AdaBoost Algorithm

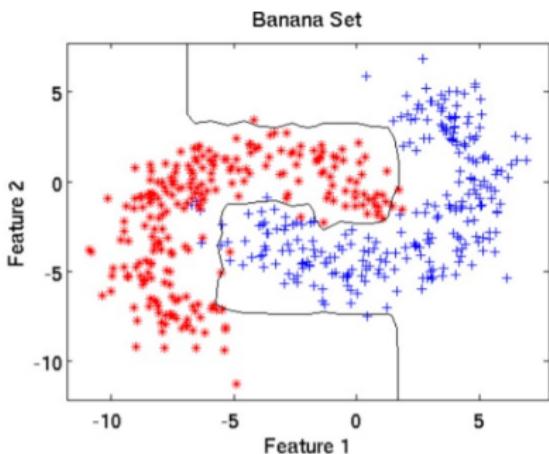
- Input: N examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$
- Initialize: $d_n^{(1)} = 1/N$ for all $n = 1, \dots, N$
- Do for $t = 1, \dots, T$
 - ① train **base learner** according to example distribution $d^{(t)}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto \{\pm 1\}$
 - ② compute weighted error $\epsilon_t = \sum_{n=1}^N d_n^{(t)} I(y_n \neq h_t(\mathbf{x}_n))$
 - $I(\cdot)$: indicator function; output 1 if argument holds, 0 otherwise
 - accurate $h_t \rightarrow \epsilon_t$ small
 - ③ compute **hypothesis weight** $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
 - accurate $h_t \rightarrow \alpha_t$ large
 - ④ update **example distribution** $d_n^{(t+1)} \propto d_n^{(t)} \exp(-\alpha_t y_n h_t(\mathbf{x}_n))$
 - $y_n = h_t(\mathbf{x}_n)$: $y_n h_t(\mathbf{x}_n) = 1 \rightarrow \exp(-\alpha_t y_n h_t(\mathbf{x}_n))$ small
 - $y_n \neq h_t(\mathbf{x}_n)$: $y_n h_t(\mathbf{x}_n) = -1 \rightarrow \exp(-\alpha_t y_n h_t(\mathbf{x}_n))$ large
- Output: final hypothesis

$$f_{\text{ens}}(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$$

Example



AdaBoost using 20 decision trees
with default settings



Final output of AdaBoost with 20
decision trees

- boosting is based on the concept of a **weak learner**
 - an algorithm that performs **slightly better than chance**
 - e.g., on two-class problems, classification accuracy slightly larger than 50%
- a weak learner can be converted to a **strong learner** (that are highly accurate) by boosting
- boosting can also be used with strong learners, but the benefits in this case will be small

Summary

- We introduced the basic concepts behind **ensemble methods**
- We described two ensemble methods that manipulate the training set
 - Bagging
 - Boosting

Clustering: k -means and k -medoids

Yu Zhang

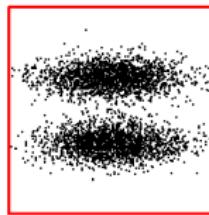
Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

- dataset D
- object $\mathbf{x} = (x_1, x_2, \dots, x_d)$
 - every \mathbf{x} is a point in a d -dimensional space

Clustering: groups the data into clusters

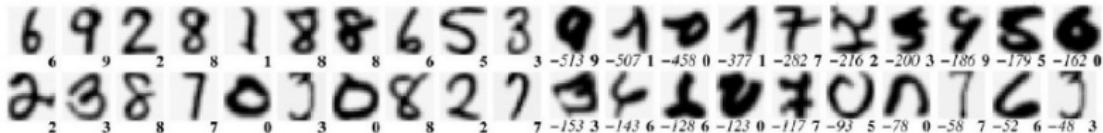
- objects in the same cluster have high similarity
- objects in different clusters are dissimilar to each other



- unlike classification, there is **no class attribute** in clustering
 - this is quite common in large databases, because assigning class labels to a large number of objects can be very costly
- **unsupervised learning** (on the other hand, classification is **supervised learning**)

Applications of Clustering

- business
 - discover distinct groups of customers based on their purchasing patterns
- biology
 - derive plant and animal taxonomies
- geographical data
 - group houses in a city according to house type and geographical location
- outlier detection
 - find credit card transactions that are not ordinary (fraud detection)

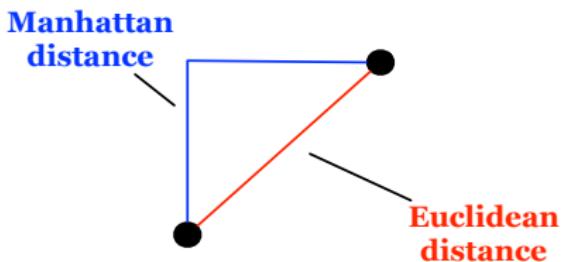


Dissimilarity

- measures how **different** two objects are (lower when objects are more alike)
- minimum dissimilarity is often 0

$$\mathbf{x}_1 = (x_{11}, x_{12}, \dots, x_{1d}) \text{ and } \mathbf{x}_2 = (x_{21}, x_{22}, \dots, x_{2d})$$

- Euclidean distance:** $dist(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{i=1}^d (x_{1i} - x_{2i})^2}$
- Manhattan distance:** $dist(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^d |x_{1i} - x_{2i}|$



- standardization** is necessary if the scales of the attributes vary considerably

Similarity

- measures how **alike** two objects are
 - higher when objects are more alike
 - often falls in the range [0, 1]

Example

$$\text{cosine similarity: } \cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\|\mathbf{x}_1\| \cdot \|\mathbf{x}_2\|}$$

- $\mathbf{x}_1 = (3, 2, 0, 5, 0, 0, 0, 2, 0, 0)$
- $\mathbf{x}_2 = (1, 0, 0, 0, 0, 0, 0, 1, 0, 2)$
- cosine similarity is: 0.31

Similarity Measures for Binary Attributes

- \mathbf{x}_1 and \mathbf{x}_2 : two objects containing d **binary** attributes
 - f_{00} : number of attributes with \mathbf{x}_1 is 0 and \mathbf{x}_2 is 0
 - f_{01} : number of attributes with \mathbf{x}_1 is 0 and \mathbf{x}_2 is 1
 - f_{10} : number of attributes with \mathbf{x}_1 is 1 and \mathbf{x}_2 is 0
 - f_{11} : number of attributes with \mathbf{x}_1 is 1 and \mathbf{x}_2 is 1
- simple matching coefficient: $SMC = \frac{f_{11} + f_{00}}{f_{01} + f_{10} + f_{00} + f_{11}} = \frac{f_{11} + f_{00}}{d}$

Example

- $\mathbf{x}_1 = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0), \mathbf{x}_2 = (0, 0, 0, 0, 0, 0, 1, 0, 0, 1)$
- $f_{00} = 7, f_{01} = 2, f_{10} = 1, f_{11} = 0; SMC = \frac{0+7}{2+1+0+7} = 0.7$

- for a given attribute, sometimes one value may be more important than the other

Example

- attribute: is-Nobel-laureate
- two Nobel laureates are more similar than two non-Nobel-laureates
- Jaccard coefficient: $JC = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$
 - number of negative matches is considered unimportant

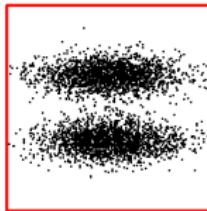
Example

- $JC = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} = \frac{0}{2+1+0} = 0$

Partitioning Methods

Basic idea

- organize the N objects into k partitions ($k < N$), where each partition represents a **cluster**
 - objects within a cluster are similar, whereas objects of different clusters are dissimilar



- the clusters are formed to minimize an **objective criterion**, such as a **distance function**

Will focus on the following partitioning methods

- ***k*-means**
- ***k*-medoids**

k-means

- C_i : a cluster; N_i : number of points in C_i
- **mean** of this cluster: $\mathbf{c}_i = \frac{1}{N_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$
 - can be regarded as the **centroid** or **center of gravity** of the cluster

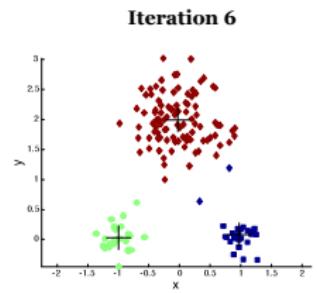
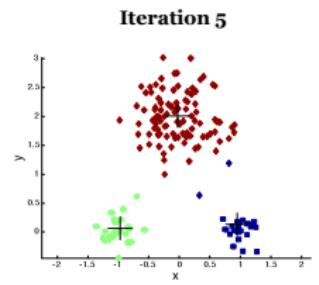
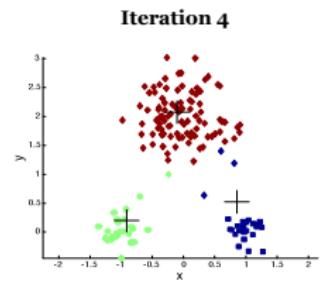
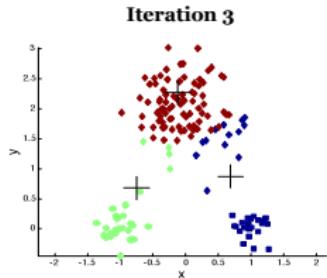
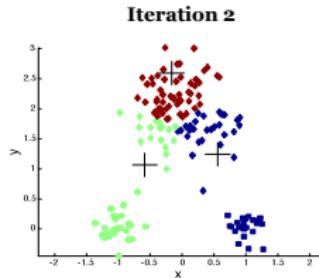
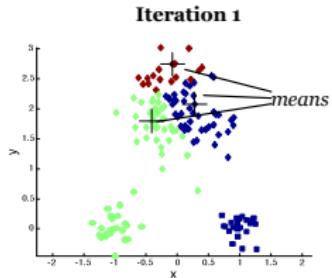
k-means algorithm

input: dataset D , number of clusters k

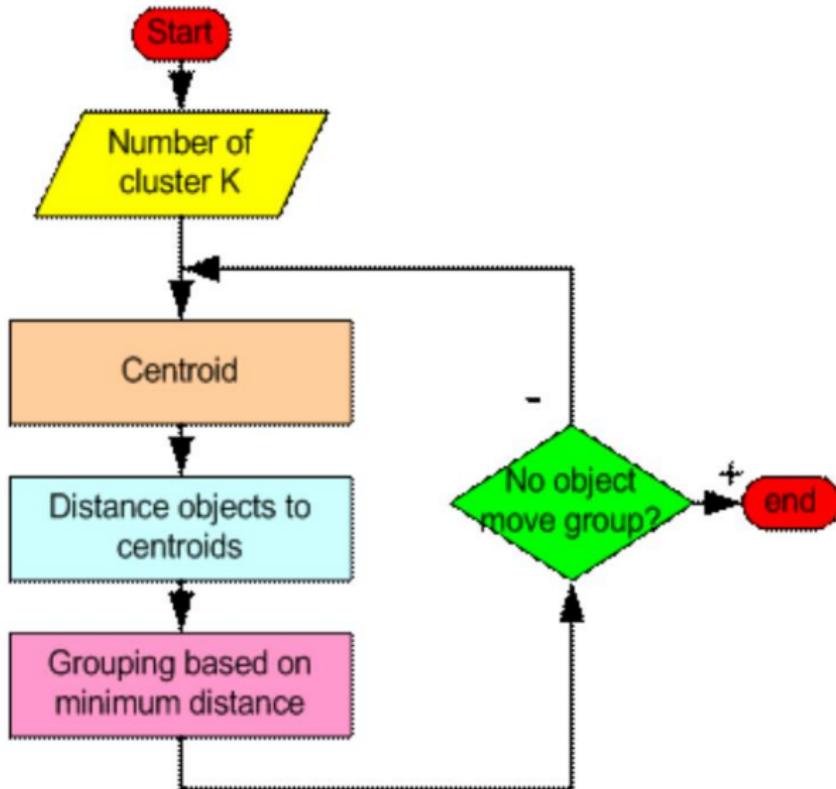
output: k clusters

1. randomly select k points from D as the **initial** means
2. **repeat**
3. form k clusters by assigning each point to its **closest** mean
4. **recompute** the mean for every cluster
5. **until** no changes in the mean
6. **return** the k clusters

Example



demo



Objective function for k -means clustering

- objective criterion that k -means attempts to minimize

$$\min_{\alpha, \{\mathbf{c}_i\}} \sum_{j=1}^n \sum_{i=1}^k \alpha_{ji} \|\mathbf{x}_j - \mathbf{c}_i\|^2 \text{ s.t. } \sum_{i=1}^k \alpha_{ji} = 1 \quad \forall j, \alpha_{ji} \in \{0, 1\} \quad \forall i, j$$

- The first step corresponds to minimizing with respect to α
 - The optimizations of α_j and α_q ($j \neq q$) are independent where $\alpha_j = (\alpha_{j1}, \dots, \alpha_{jk})^T$.
 - can be decomposed into n subproblems with each one formulated as

$$\min_{\alpha_j} \sum_{i=1}^k \alpha_{ji} \|\mathbf{x}_j - \mathbf{c}_i\|^2 \text{ s.t. } \sum_{i=1}^k \alpha_{ji} = 1, \quad \alpha_{ji} \in \{0, 1\} \quad \forall j$$

- an analytical solution:

$$\alpha_{jt} = \begin{cases} 1 & \text{if } t = \arg \min_i \|\mathbf{x}_j - \mathbf{c}_i\|^2 \\ 0 & \text{otherwise} \end{cases}$$

Objective function for k -means clustering

- objective criterion that k -means attempts to minimize

$$\min_{\alpha, \{\mathbf{c}_i\}} \sum_{j=1}^n \sum_{i=1}^k \alpha_{ji} \|\mathbf{x}_j - \mathbf{c}_i\|^2 \text{ s.t. } \sum_{i=1}^k \alpha_{ji} = 1 \quad \forall j, \alpha_{ji} \in \{0, 1\} \quad \forall i, j$$

- The second step corresponds to minimizing with respect to $\{\mathbf{c}_i\}$

- The optimizations of different \mathbf{c}_i 's are independent.
- can be decomposed into k subproblems with each one formulated as

$$\min_{\mathbf{c}_i} \sum_{j:\alpha_{ji}=1} \alpha_{ji} \|\mathbf{x}_j - \mathbf{c}_i\|^2$$

- an analytical solution:

$$\mathbf{c}_i = \frac{\sum_{j:\alpha_{ji}=1} \mathbf{x}_j}{\sum_{j:\alpha_{ji}=1} \alpha_{ji}}$$

Issues: Local Minimum

- objective criterion that k -means attempts to minimize

$$\text{(sum squared error) } SSE = \sum_{i=1}^k \sum_{x \in C_i} \|x - c_i\|^2$$

- converges to a local minimum → suboptimal clustering

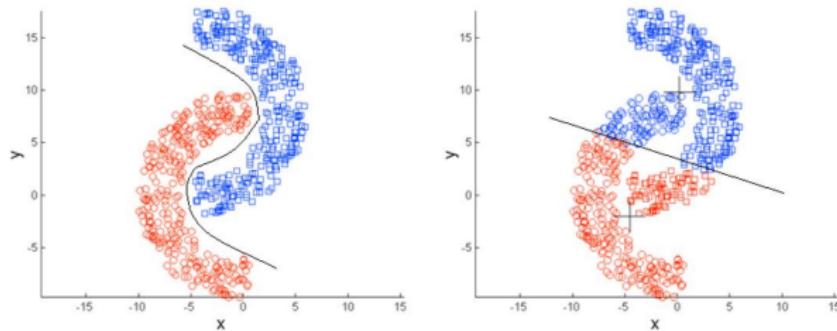


- different initial starts of the means gives different final answers
→ initial selection is very important

how to alleviate this problem?

- ① multiple runs
- ② pick the solution with minimum SSE

Issues: Implicit Assumption on Cluster Shape

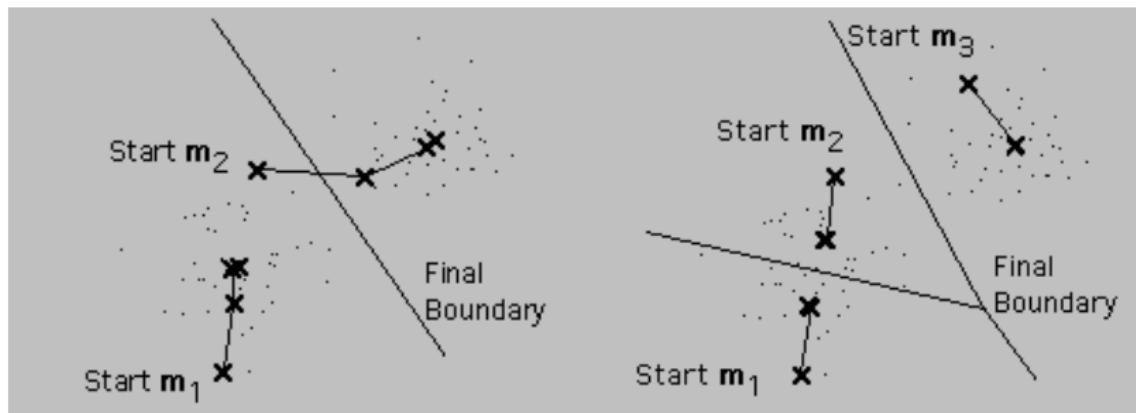


- can get wrong results when clusters have other shapes

Issues: Number of Clusters

You have to pick the number of clusters

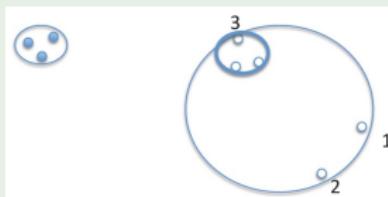
- in general, clustering result depends on k



Issues: Outliers

- the means may be **fictitious** (i.e., non-existent in the dataset)
- k -means is sensitive to **outliers**

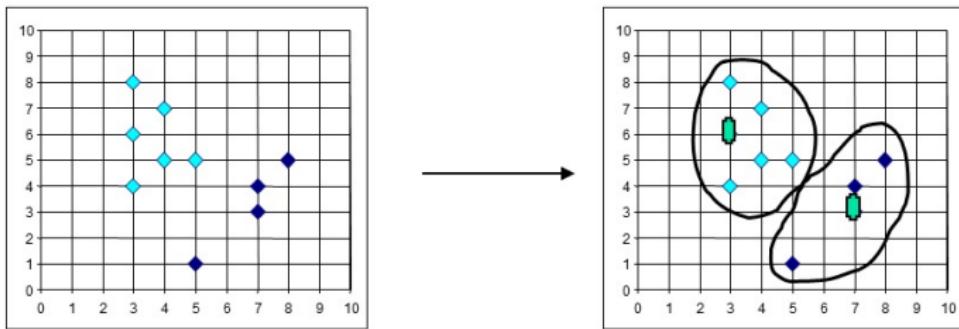
Example



- by removing points 1 and 2, we obtain much tighter clusters

k -Medoids

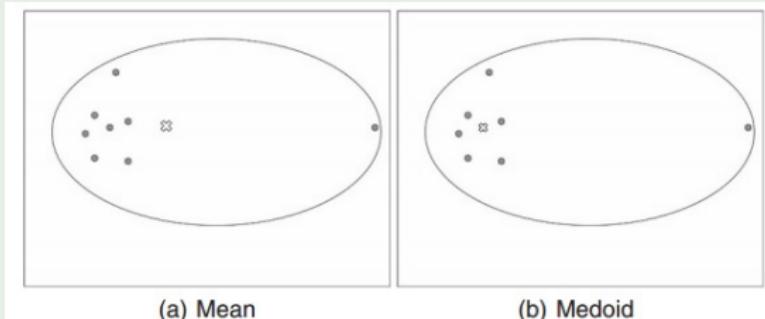
- recall that for k -means: obtained cluster representatives may be fictitious and considerably affected by outliers
- basic idea of k -medoids
 - instead of taking the mean, use the most centrally located object (**medoid**) as cluster representative



k -Medoids...

- k -medoids diminishes the sensitivity to outliers

Example (one outlier on the right)



- recall that k -means tries to minimize

$$SSE = \sum_{i=1}^k \sum_{x \in C_i} \|x - c_i\|^2$$

- k -medoids tries to minimize

$$\text{(sum absolute error)} SAE = \sum_{i=1}^k \sum_{x \in C_i} \|x - m_i\|$$

Partitioning Around Medoids (PAM)

- a popular k -medoid algorithm

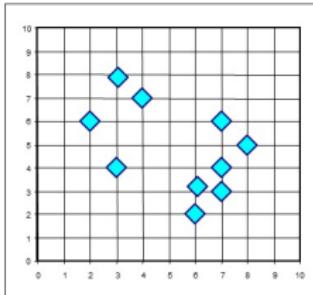
k -Medoids (PAM) Algorithm

input: dataset D , number of clusters k

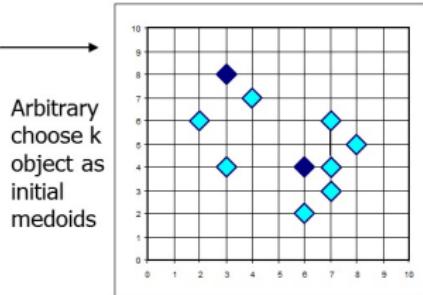
output: k clusters

1. randomly select k points from D as the initial medoids
2. **repeat**
3. form k clusters by assigning each point to its closest medoid, and compute SAE
4. **for** each medoid m ;
5. **for** each non-medoid x
6. **swap** m ; with x and compute SAE
7. select the configuration with the lowest SAE
8. **until** no medoid changes
9. **return** the k clusters

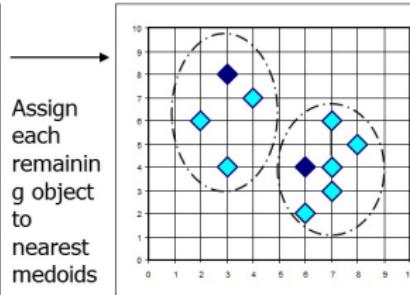
Example



K=2



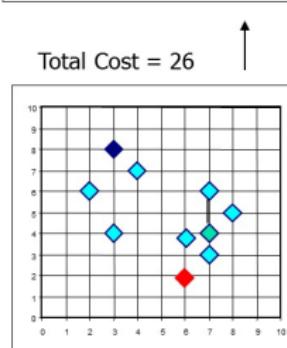
Arbitrary choose k object as initial medoids



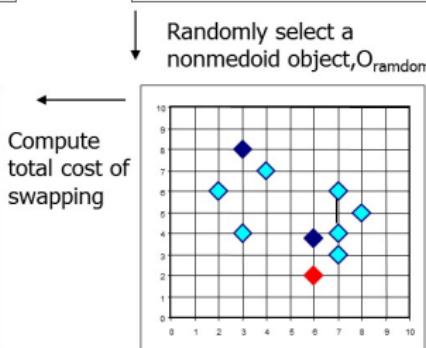
Total Cost = 20

Do loop
Until no change

Swapping O and O_{random}
If quality is improved.

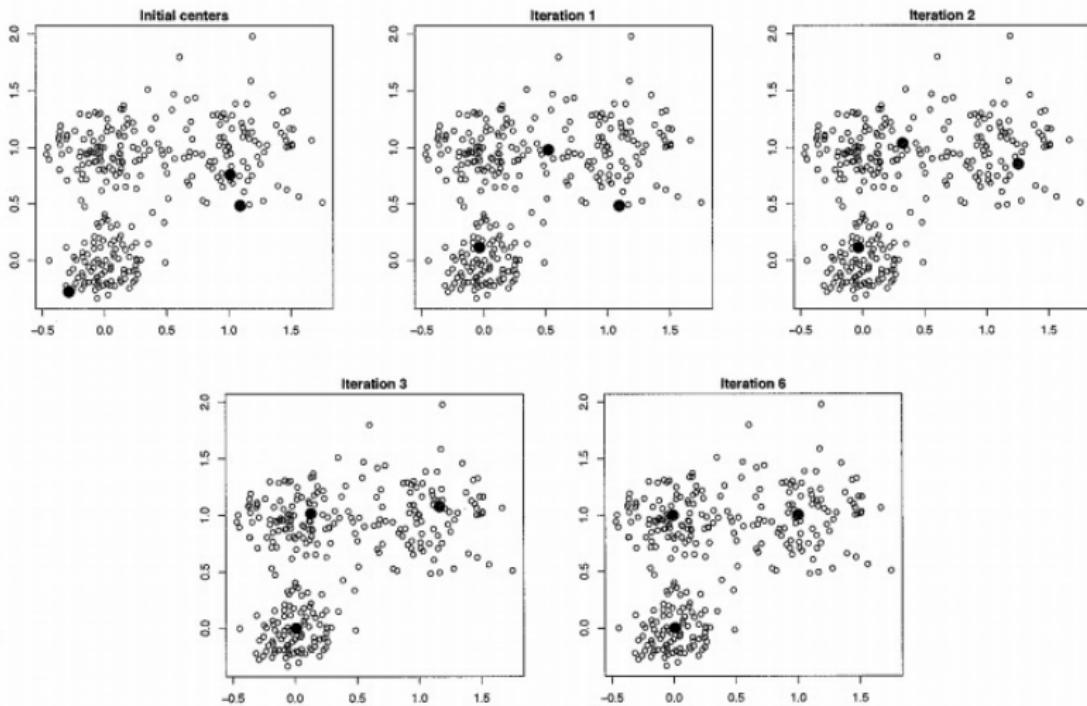


Total Cost = 26



Compute total cost of swapping

Example



demo

Remarks

- k -medoids also converges to a **local minimum** → different starts gives different final answers
- k -medoids is more **robust** than k -means regarding outliers
- k -medoids is more **expensive** than k -means
 - not scalable for large data sets

Summary

- We provided the preliminaries on **clustering**
- We described two clustering methods based on partitioning
 - *k*-means
 - *k*-medoids

Hierarchical Clustering

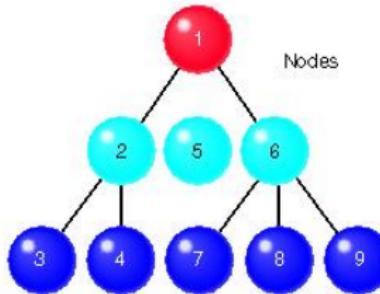
Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Basics

- hierarchical clustering works by grouping data objects into a **tree of clusters**



- clusters → subclusters → subsubclusters → ⋯

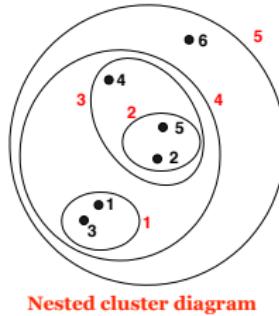
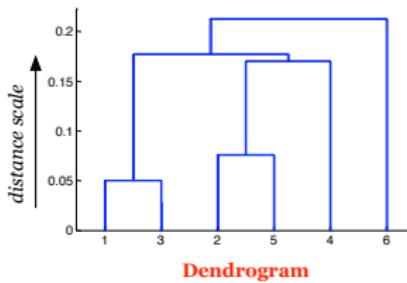
Why do we need hierarchies?

Example

Biology, library book categorization, web directories

Basics

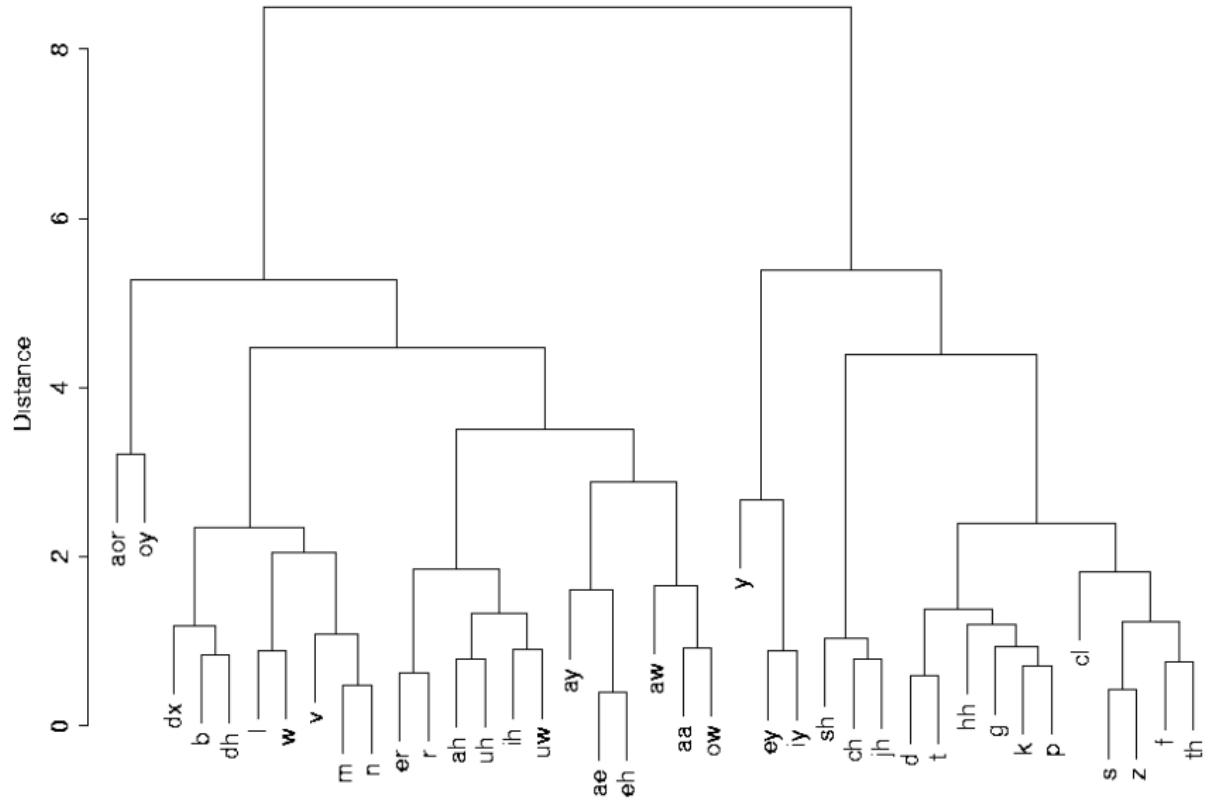
- use a **similarity** or **distance** (**dissimilarity**) matrix
- merge/split **one** cluster at a time
- can be graphically displayed by
 - dendrograms**
 - nested cluster diagrams**



Dendrogram

- given any two samples x and x' , at some level they will be grouped together in the same cluster
- if two samples are in the same cluster at some level $c \rightarrow$ they remain together at all higher levels ($> c$)

Example: Dendrogram of 39 English Sounds



Strengths and Weaknesses

Strengths

- no assumptions on the **number** of clusters
 - any desired number of clusters can be obtained by “**cutting**” the dendrogram at the proper level
- hierarchical clusterings may correspond to **meaningful taxonomies** (e.g., in biological sciences, the Web etc.)

Weaknesses

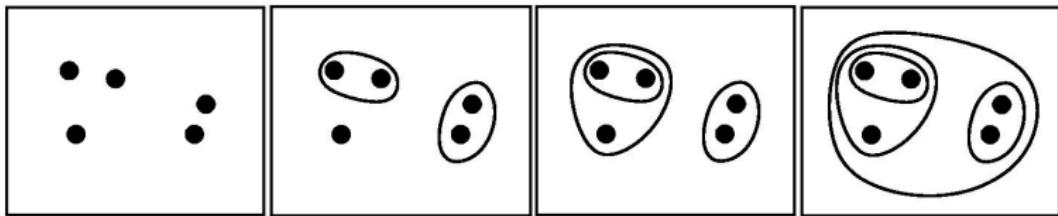
- **computationally intensive**
 - not usable for large datasets
- may lead to poor quality because they are **unable to make adjustments** once a merge/split decision has been executed

Agglomerative vs Divisive

Two types of hierarchical clustering methods:

① **agglomerative** (bottom-up)

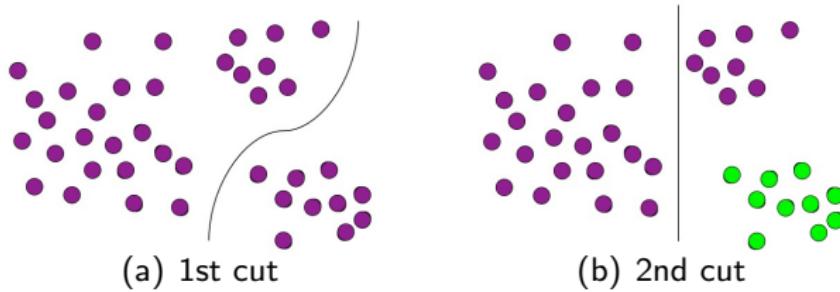
- start with the **points** as individual clusters
- at each step, **merge** the closest pair of clusters until only one cluster (or k clusters) left



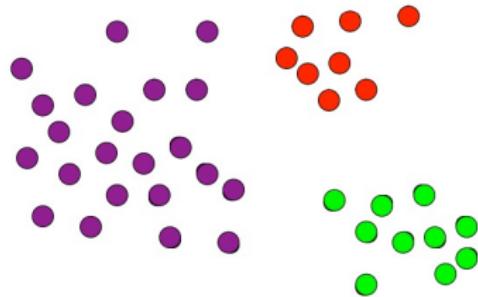
② **divisive** (top-down)

- start with one, all-inclusive **cluster**
- at each step, **split** a cluster until each cluster contains a point (or there are k clusters)

Divisive Hierarchical Clustering: Example



final result



Agglomerative Hierarchical Clustering

input: dataset D of points

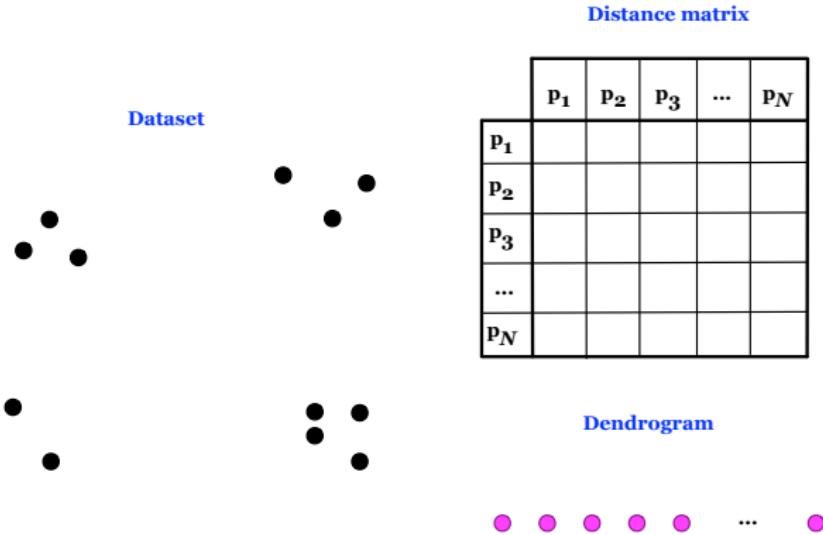
output: tree of clusters

1. compute the distance matrix over D
2. initialize **each** data point as a different cluster
3. **repeat**
4. merge the two **closest** clusters
5. update the distance matrix
6. **until** only one cluster remains

Example

Initial setting

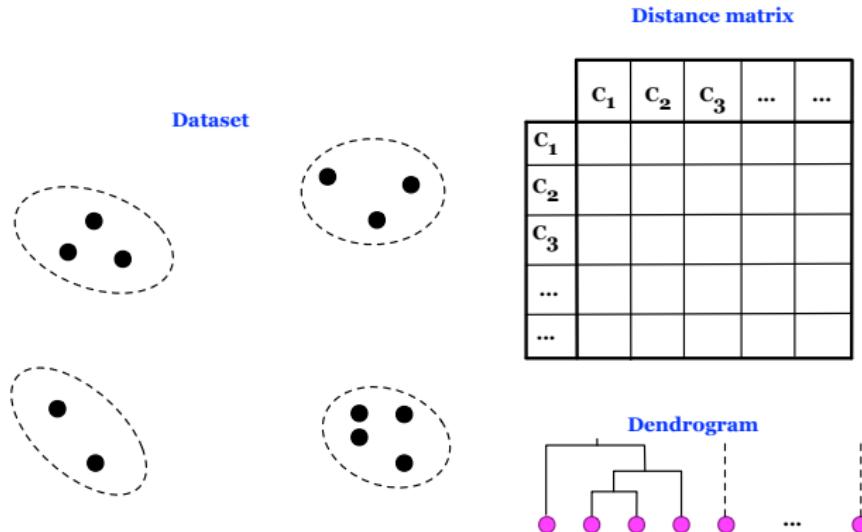
- start with the initial dataset D and compute the distance matrix that records distances between data points



Example...

After some steps

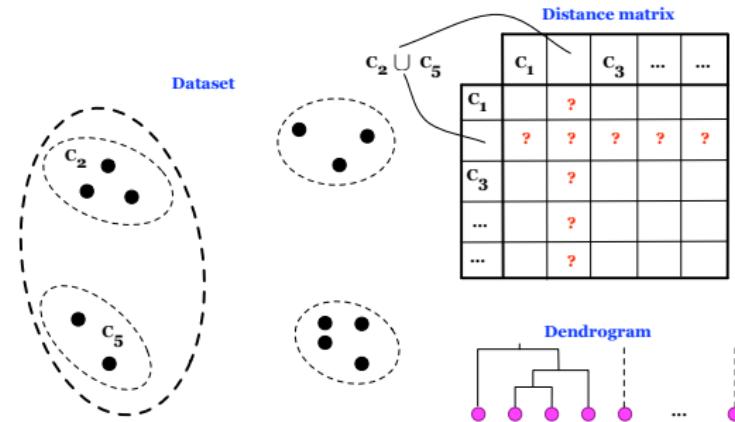
- several clusters have been formed
- also a new distance matrix has been computed, which records distances between clusters



Example...

Merging two closest clusters

- suppose we merge clusters C_2 with C_5 in the figure below



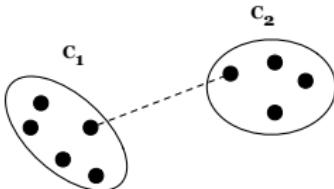
How to update the distance matrix?

Distance between Clusters

- involve computing **distances between clusters**
 - each cluster is a **set** of points
- different definitions of the distance between clusters leads to **different clustering behavior**
- we will explore the following distances:
 - single-link distance
 - complete-link distance
 - group average distance

Single-Link Distance

- let C_1 and C_2 be two clusters



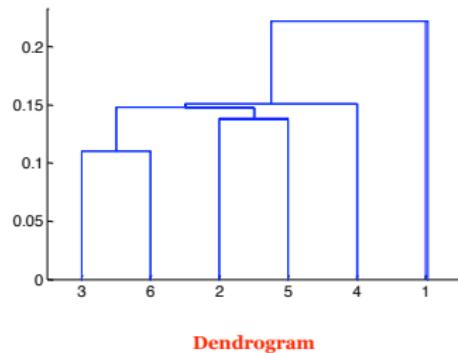
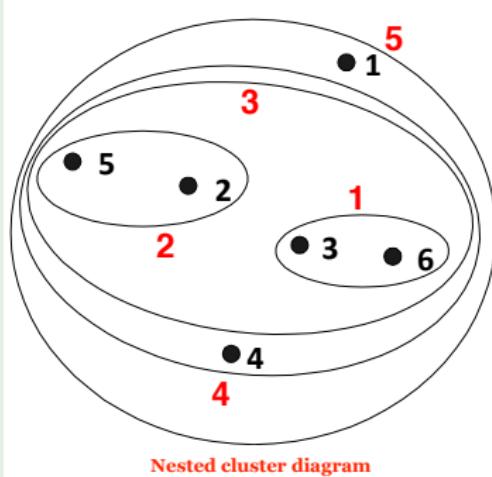
- single-link distance: the **minimum distance** between any object in C_1 and any object in C_2

$$dist_{single}(C_1, C_2) = \min_{\mathbf{x}_1, \mathbf{x}_2} \{ dist(\mathbf{x}_1, \mathbf{x}_2) \mid \mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2 \}$$

- i.e., defined by the **most similar** pair of objects
- depends on a distance metric, such as Euclidean distance

Example

Example

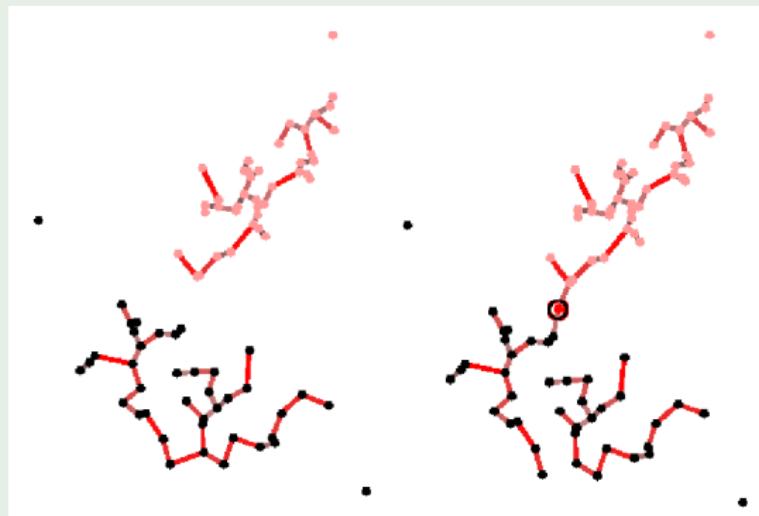


- tends to produce long clusters

Limitation

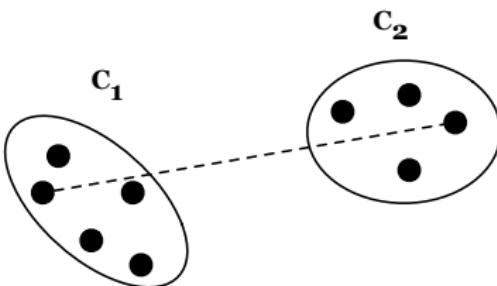
Sensitive to **noise** or slight changes in positions of the data points

Example



- different result due to addition of a new point
- **single-link**

Complete-Link Distance



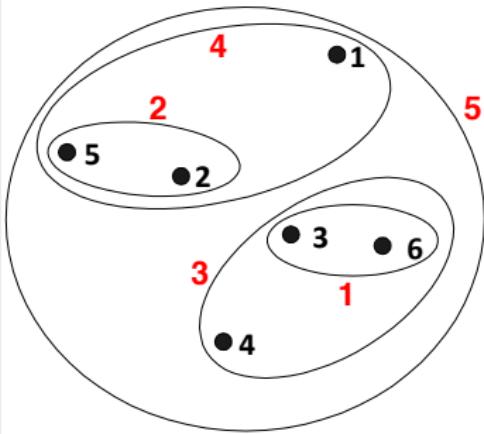
- complete-link distance: the **maximum distance** between any object in C_1 and any object in C_2

$$dist_{complete}(C_1, C_2) = \max_{\mathbf{x}_1, \mathbf{x}_2} \{ dist(\mathbf{x}_1, \mathbf{x}_2) \mid \mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2 \}$$

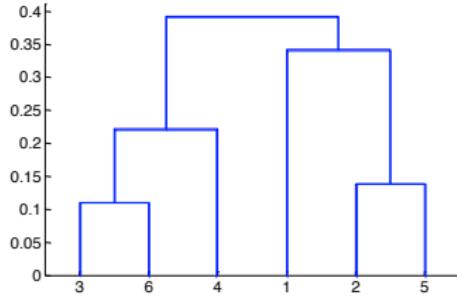
- i.e., defined by the **most dissimilar** pair of objects
- again depends on a distance metric

Example

Example



Nested cluster diagram

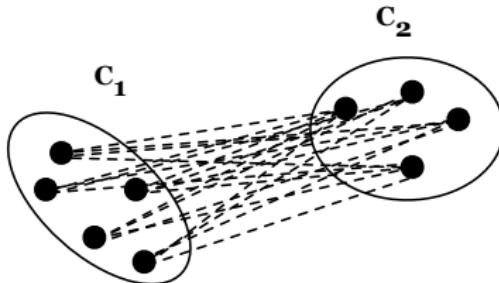


Dendrogram

- at each iteration, the size (largest diameter) of the partition is increased as little as possible
 - tends to produce very **tight** clusters
 - problematic if the true clusters are elongated

Group Average Distance

- let C_1 and C_2 be two clusters, with cardinality N_1 and N_2 , respectively

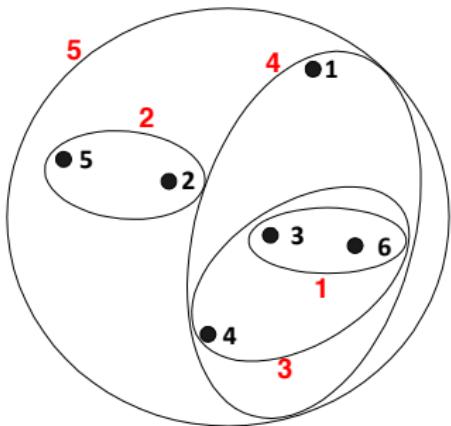


- group average distance: the **average distance** between any object in C_1 and any object in C_2

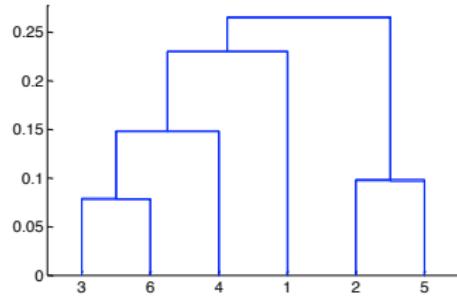
$$dist_{avg}(C_1, C_2) = \frac{1}{N_1 \cdot N_2} \sum_{x_1 \in C_1, x_2 \in C_2} dist(x_1, x_2)$$

- i.e., defined by **all** the objects in the union of the two clusters

Group Average Distance...

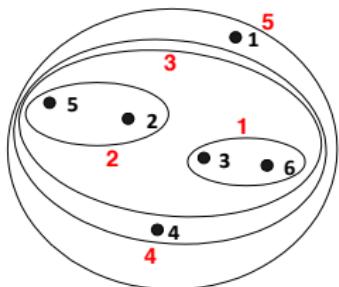


Nested cluster diagram

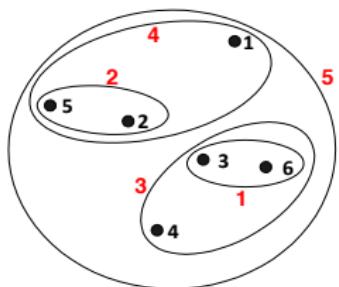


Dendrogram

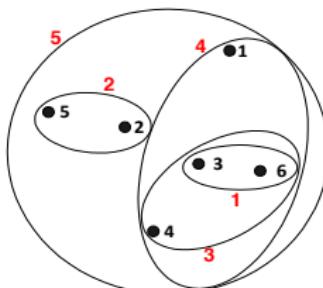
Comparison of Different Results



Single-link



Complete-link



Group average

Summary

- We have discussed some basic notions revolving around hierarchical clustering methods
- We explained one of the two categories of hierarchical clustering, namely agglomerative clustering

Density-Based Clustering

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

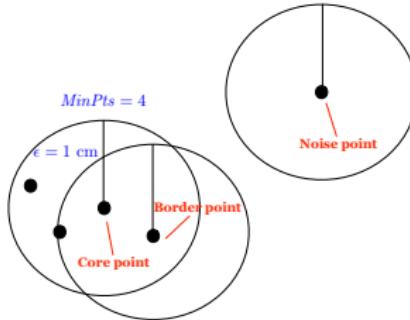
Fall 2017

Density-based Clustering

- regards clusters as **dense** regions of objects in the data space that are separated from regions of **low** density
- a cluster is defined as a maximal set of **density-connected points**
- discovers clusters of **arbitrary** shape
- in this lecture, we will cover **DBSCAN** (Density Based Spatial Clustering of Applications with Noise)

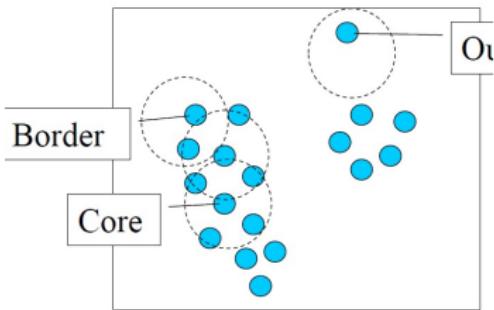
Definitions

- user-defined parameters: ϵ , $MinPts$
- ϵ -neighborhood of a point p
 - points within a radius of ϵ from p (including p)

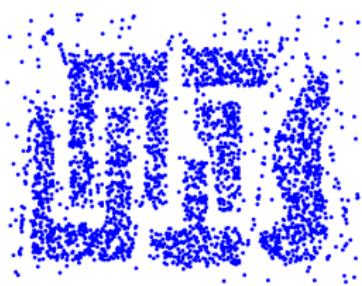


- **core point**
 - if the number of points lying inside the ϵ -neighborhood of p is $\geq MinPts$, then p is a core point
- **border point**
 - not a core point (i.e., has fewer than $MinPts$ within ϵ), but falls within the neighborhood of a core point
- **noise point**
 - neither a core point nor a border point

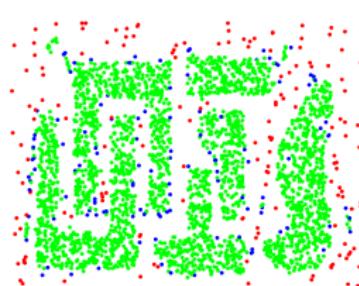
Examples



- core points are points that are at the **interior** of a cluster



Original Data



Core , Border and Noise Data

$MinPts = 4$

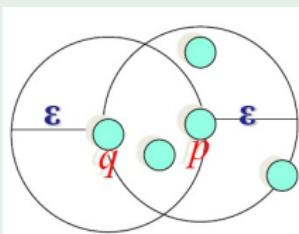
$\epsilon = 10$

Directly Density-Reachability

A point q is **directly density-reachable** from a point p if

- ① p is a core point, and
- ② q is in p 's ϵ -neighborhood

Example



$\text{MinPts} = 4$

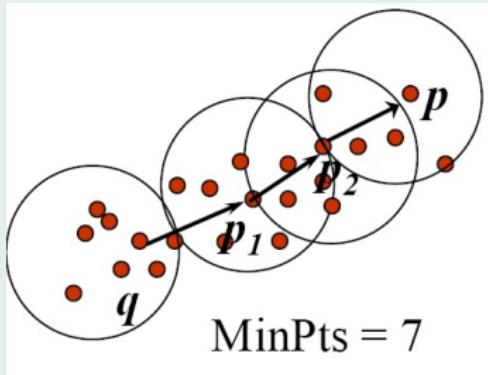
- is q directly density-reachable from p ? yes
- is p directly density-reachable from q ? no

directly density-reachability is **not** symmetric

Density-Reachability

- p is density-reachable from q

Example



- $p \leftarrow p_2 \leftarrow p_1 \leftarrow q$ form a chain
 - p_1 is directly density-reachable from q
 - p_2 is directly density-reachable from p_1
 - p is directly density-reachable from p_2
- is q density-reachable from p ? no

Idea

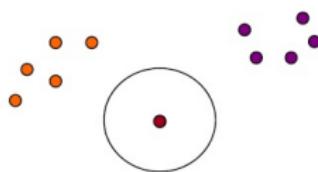
For a **core point** p , the set

$$\{o \mid o \in D \text{ and } o \text{ is density-reachable from } p\}$$

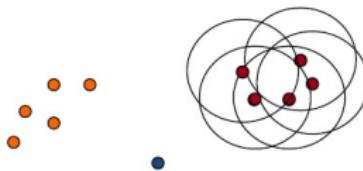
is a **cluster**

Basic steps

- ① choose an arbitrary point from the database D satisfying the core point condition as a seed



- ② retrieve all points that are density-reachable from the seed → obtain the cluster containing the seed



DBSCAN(D , ϵ , MinPts)

```
 $C = 0;$ 
for each unvisited point  $p$  in dataset  $D$  do
    mark  $p$  as visited;
    NeighborPts = regionQuery( $p$ ,  $\epsilon$ );
    if sizeof(NeighborPts) < MinPts then
        mark  $p$  as NOISE
    end
    else
         $C =$  next cluster;
        expandCluster( $p$ , NeighborPts,  $C$ ,  $\epsilon$ , MinPts);
    end
end
```

Note

- the noise point might later be found in a ϵ -neighborhood of a different point and hence be made part of a cluster

regionQuery(p , ϵ)

return all points within p 's ϵ -neighborhood (including p)

`expandCluster(p , NeighborPts, C , ϵ , MinPts)`

If a point is found to be a dense part of a cluster, its ϵ -neighborhood is also part of that cluster

```
add  $p$  to cluster  $C$ ;
for each point  $p'$  in NeighborPts do
    if  $p'$  is not visited then
        mark  $p'$  as visited;
        NeighborPts' = regionQuery( $p'$ ,  $\epsilon$ );
        if sizeof(NeighborPts')  $\geq$  MinPts then // for every point
            within the  $\epsilon$ -neighborhood: if it is also dense
            → its own  $\epsilon$ -neighborhood also added

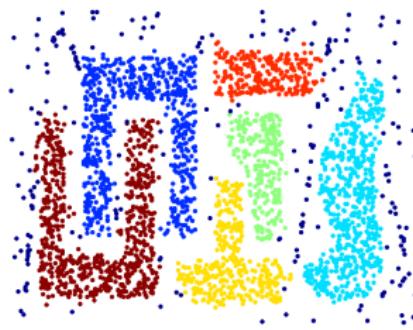
        NeighborPts = NeighborPts joined with NeighborPts'
    end
end
if  $p'$  is not yet member of any cluster then
    add  $p'$  to cluster  $C$ 
end
end
```

Example

- resistant to noise
- can handle clusters of arbitrary shapes and sizes



Original Data



Clusters and noise found by DBSCAN

$MinPts = 4$

$\epsilon = 10$

demo

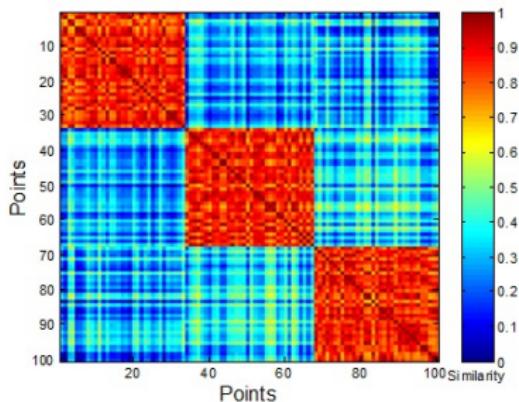
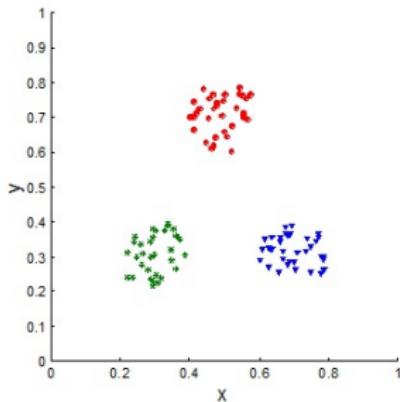
Cluster Validity

- for supervised classification we have a variety of measures to evaluate how good our model is
 - accuracy, precision, recall
- for cluster analysis, the analogous question is

How to evaluate the “goodness” of the resulting clusters?

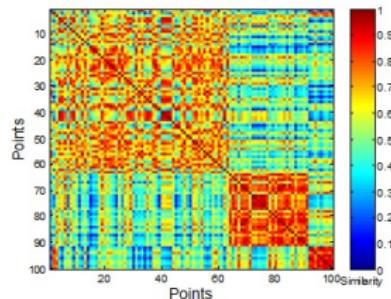
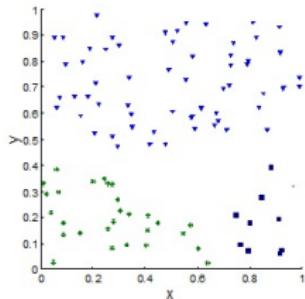
Using Similarity Matrix for Cluster Validation

- order the similarity matrix with respect to cluster labels and inspect visually

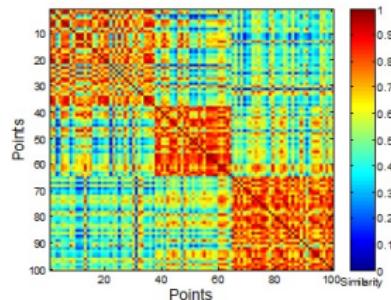
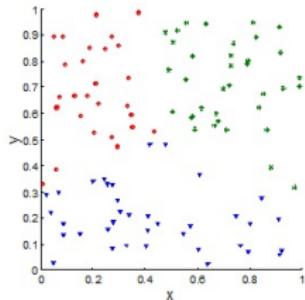


Clusters in Random Data are not so Crisp

DBSCAN



k -means



- We described the third category of clustering, namely **density-based clustering**
- We explained the **DBSCAN** algorithm

Finite Mixture Model

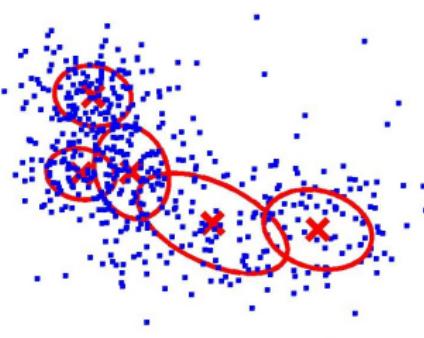
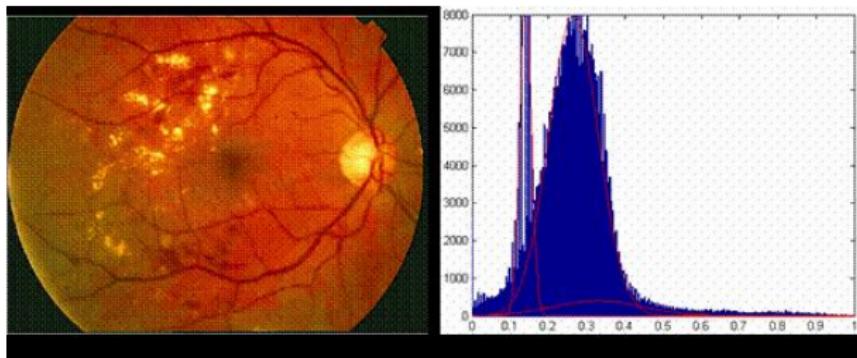
Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

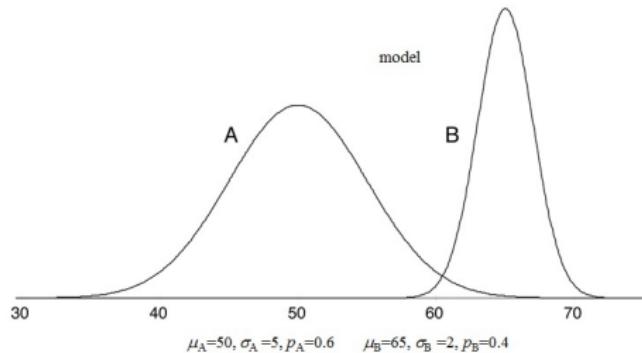
Finite Mixture Model

Probabilistic clustering algorithms model the data using a **mixture of distributions**



Finite Mixture Model...

- usually individual distributions are normal distribution

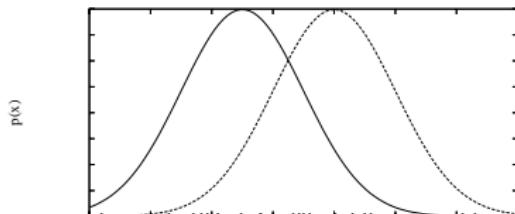


- distributions are combined using **weights**
- they are called **finite** mixtures because only a finite number of components are used

Simple Problem

Given n instances generated by a **mixture** of 2 **known** distributions

- e.g., normal distributions
- $p(x|\omega_1, \theta_1) = N(\mu_1, \Sigma_1), p(x|\omega_2, \theta_2) = N(\mu_2, \Sigma_2)$



- **unknown** prior probabilities: $\pi_1 = P(\omega_1), \pi_2 = P(\omega_2)$
- $\pi_1 + \pi_2 = 1$

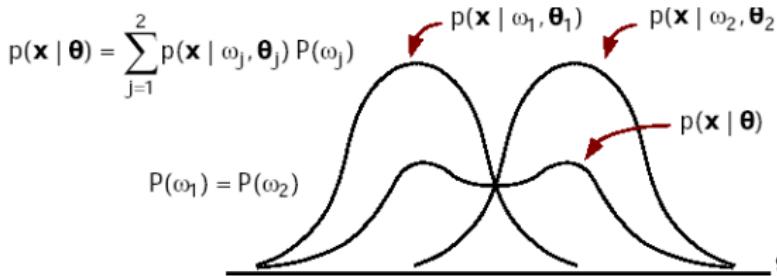
Each instance x generated by

- ① choosing one of the 2 distributions with probability π_1 (or π_2)
- ② generating an instance at random according to that distribution

Do **not** know which instance x_i was generated by which distribution

How to estimate π_1 and π_2 ?

Simple Problem...



In general, with c components, the combined pdf:

$$p(x|\theta) = \sum_{i=1}^c \pi_i p(x|\omega_i, \theta_i) = \sum_{i=1}^c \pi_i p_i(x)$$
$$\theta = (\pi_1, \dots, \pi_{c-1}), \quad \pi_c = 1 - \sum_{i=1}^{c-1} \pi_i$$

- $p(x|\theta)$: **mixture density**
- $p_i(x)$: **component density**
- π_i : **mixing parameter**

Missing Data Problem and Expectation Maximization

Think of full description of each instance as $y_j = \langle x_j, z_{1j}, \dots, z_{cj} \rangle$

- z_{ij} is 1 if x_j generated by i th distribution, 0 otherwise
 - unobservable (hidden variable, **missing data**)
- x_j : observable (observed variable, **incomplete data**)
- $\langle x_j, z_{ij} \rangle$: **complete data**
 - if z_{ij} are observed, then $\pi_i = \sum_{j=1}^n z_{ij}/n$

Expectation Maximization (EM) Algorithm

Basic idea

- ① use the current hypothesis $\langle \pi_1, \dots, \pi_c \rangle$ to estimate the expected values of the hidden variables z_{ij}
- ② recalculate the hypothesis using these expected values for the hidden variables
- ③ go back to Step 1 until convergence

① Pick random initial $h = \langle \pi_1, \dots, \pi_c \rangle$

② Expectation step (E-step):

- assume the current hypothesis $h = \langle \pi_1, \dots, \pi_c \rangle$ holds
- calculate $E[z_{ij}]$ of each hidden variable z_{ij}

$$E[z_{ij}] = \sum z_{ij} P(z_{ij}) = P(z_{ij} = 1)$$

= posterior probability that the j th instance (with observed value x_j) belongs to the i th component

$$= \pi_i p_i(x_j) / \sum_{i=1}^c \pi_i p_i(x_j)$$

③ Maximization step (M-step):

- calculate a new (maximum likelihood) hypothesis

$h' = \langle \pi'_1, \dots, \pi'_c \rangle$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated above

- replace $h = \langle \pi_1, \dots, \pi_c \rangle$ by $h' = \langle \pi'_1, \dots, \pi'_c \rangle$:

$$\pi'_i \leftarrow \frac{\sum_{j=1}^n E[z_{ij}]}{n}, \quad i = 1, \dots, c$$

- each instance contributes an amount which is equal to its (currently assessed) posterior probability of membership of the i th component of the mixture

EM Learning of the Mixture Model

Assume we know that there are c clusters, the combined pdf:

$$p(x|\theta) = \sum_{i=1}^c \pi_i p(x|\omega_i, \theta_i) = \sum_{i=1}^c \pi_i p_i(x)$$

- each component density is univariate normal with **unknown** means μ_1, \dots, μ_c and common **unknown** variance σ^2

$$p_i(x) = (2\pi\sigma^2)^{-1/2} \exp\left\{-\frac{1}{2}\frac{(x-\mu_i)^2}{\sigma^2}\right\}$$

- to learn the clusters we need to determine their parameters
 - $\theta = (\mu_1, \dots, \mu_c, \sigma^2, \pi_1, \dots, \pi_{c-1})'$, $\pi_c = 1 - \sum_{i=1}^{c-1} \pi_i$
- think of full description of each instance as
 $y_j = \langle x_j, z_{1j}, \dots, z_{cj} \rangle$
 - z_{ij} is 1 if x_j generated by i th Gaussian, 0 otherwise

E-Step

- ① Pick random initial $h = \langle \mu_1, \dots, \mu_c, \sigma^2, \pi_1, \dots, \pi_{c-1} \rangle$
- ② E step: can be reduced to computing

$$E[z_{ij}] = \frac{\pi_i p_i(x_j)}{\sum_{i=1}^c \pi_i p_i(x_j)}$$

- ③ M-step:

$$\pi'_i \leftarrow \frac{\sum_{j=1}^n E[z_{ij}]}{n}, \quad i = 1, \dots, c$$

$$\mu'_i \leftarrow \frac{\sum_{j=1}^n E[z_{ij}]x_j}{\sum_{j=1}^n E[z_{ij}]}$$

$$(\sigma')^2 \leftarrow \sum_{i=1}^c \sum_{j=1}^n E[z_{ij}] \frac{(x_j - \mu'_i)^2}{n}$$

Example

Demo

Association Analysis: Introduction

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Motivating Example

- the table below depicts a set of **transactions** at a grocery store



Market basket transactions example

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

- each transaction corresponds to the set of items purchased by a single customer at the checkout counter of the store
- such transactions are also called **market basket transactions**

Which items are purchased together by the customers?

- given a set of transactions, **association analysis** finds **rules** that predict the occurrence of an item in a transaction based on the occurrences of other items in this transaction

Motivating Example...

Market basket transactions example

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

- one rule that can be extracted is that diapers → beer
 - this rule suggests a strong relationship between the sale of diapers and beer: customers who buy diapers also buy beer
- retailers can use this information for cross-marketing, catalog design, customer shopping behavior analysis, ...

Other applications

- medical diagnosis, DNA sequence analysis, web log (clickstream) analysis, ...

Gender	Age	Smoking	Blood pressure	...	Class
M	40 - 50	Y	high	...	abnormal
M	20 - 40	N	normal	...	normal
F	20 - 40	N	normal	...	normal
:	:	:	:	...	:

Binary Representation

- transactional data can be stored in **binary format**
- the binary representation of the data in the example of the previous slides is depicted below

TID	Bread	Milk	Diaper	Beer	Eggs	Coke
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

- a '1' entry implies the presence of the item in the transaction, whereas a '0' entry implies its absence

Support

Market basket transactions example

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

- $I = \{I_1, I_2, \dots, I_n\}$: set of all items appearing in the transactional database
- **itemset**: a set of items (that is a subset of I)
- **support count** of an itemset A ($\text{support_count}(A)$)
 - **number** of transactions that contain A
 - **example**: $\text{support_count}(\{\text{Milk, Diaper, Beer}\}) = 2$
- **support** of an itemset A
 - **fraction** of transactions that contain A
 - let $|T|$ denote the number of transactions in the database
 - $\text{support}(A) = \text{support_count}(A)/|T|$
 - alternatively, the probability that a transaction contains A :
$$\text{support}(A) = P(A)$$
 - **example**: $\text{support}(\{\text{Milk, Diaper, Beer}\}) = 2/5$

Association Rule

- let $A \subset I$ and $B \subset I$ be two itemsets, such that $A \cap B = \emptyset$

Example

$A = \{\text{Milk, Diaper}\}$ and $B = \{\text{Beer}\}$

- association rule: implication of the form
 $A \rightarrow B$ [*support, confidence*]

Example

$\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$ [*support=40%, confidence=67%*]

Support of a Rule $A \rightarrow B$

Market basket transactions example

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

- fraction of transactions that contain both A and B
- $\text{support}(A \rightarrow B) = \text{support_count}(A \cup B) / |T|$
- example: $\text{support}(\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}) = 2/5 = 40\%$
- alternatively, probability that a transaction contains both A and B
 - $\text{support}(A \rightarrow B) = P(A \cup B)$
 - note: $P(A \cup B)$ is NOT probability of A or B

Why use support?

low support rules

- may occur simply by chance
- may not be interesting

Confidence of a Rule $A \rightarrow B$

Market basket transactions example

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

- fraction of transactions containing A that also contain B

$$\text{confidence}(A \rightarrow B) = \frac{\text{support}(A \cup B)}{\text{support}(A)} = \frac{\text{support_count}(A \cup B)}{\text{support_count}(A)}$$

- example: $\text{confidence}(\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}) = 2/3 = 67\%$
- alternatively, conditional probability that a transaction having A also contains B
 - $\text{confidence}(A \rightarrow B) = P(B|A)$

Why use confidence?

- measures the reliability of the inference made by a rule

Strong Rules

- define two parameters
 - minimum support threshold (or minimum support count threshold) min_sup
 - minimum confidence threshold min_conf
- a **strong** association rule satisfies both min_sup and min_conf

How to mine the strong association rules?

Brute-force

- compute the support and confidence for **every** possible rule
- computationally expensive

Decouple the support and confidence requirements

- support of a rule $X \rightarrow Y$ depends on only the support of its corresponding itemset $X \cup Y$

Frequent itemset

- itemset A is **frequent** if it satisfies min_sup
- A is a **frequent k -itemset** if it is frequent and contains k items

Suppose that we have already computed all frequent itemsets

- for each frequent itemset S , generate **all nonempty proper subsets** A of S

- for each A , output rule $A \rightarrow (S - A)$ if

$$\text{confidence}(A \rightarrow (S - A)) = \frac{\text{support_count}(S)}{\text{support_count}(A)} \geq min_conf$$

- this rule automatically satisfies min_sup since it is derived from a frequent itemset
- hence, a **strong** rule

Example

Market basket transactions example

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

- let $\{\text{Diaper}, \text{ Beer}\}$ be a frequent itemset
- its proper nonempty subsets are $\{\text{Diaper}\}$ and $\{\text{Beer}\}$
- the rules are
 - $\{\text{Diaper}\} \rightarrow \{\text{Beer}\}$ with confidence $3/4 = 75\%$
 - $\{\text{Beer}\} \rightarrow \{\text{Diaper}\}$ with confidence $3/3 = 100\%$
- if $\text{min_conf} = 90\%$, rule is strong

Example

Market basket transactions example

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

- let $\{\text{Diaper}, \text{ Beer}\}$ be a frequent itemset
- its proper nonempty subsets are $\{\text{Diaper}\}$ and $\{\text{Beer}\}$
- the rules are
 - $\{\text{Diaper}\} \rightarrow \{\text{Beer}\}$ with confidence $3/4 = 75\%$
 - $\{\text{Beer}\} \rightarrow \{\text{Diaper}\}$ with confidence $3/3 = 100\%$
- if $\text{min_conf} = 90\%$, only the second rule is strong

Another Example

Example

- frequent itemset: $\{i1, i2, i5\}$
- subsets: $\{i1, i2\}$, $\{i1, i5\}$, $\{i2, i5\}$, $\{i1\}$, $\{i2\}$, $\{i5\}$
- resultant association rules
 - $\{i1, i2\} \rightarrow i5$, confidence = $2/4 = 50\%$
 - $\{i1, i5\} \rightarrow i2$, confidence = $2/2 = 100\%$
 - $\{i2, i5\} \rightarrow i1$, confidence = $2/2 = 100\%$
 - $i1 \rightarrow \{i2, i5\}$, confidence = $2/6 = 33\%$
 - $i2 \rightarrow \{i1, i5\}$, confidence = $2/7 = 29\%$
 - $i5 \rightarrow \{i1, i2\}$, confidence = $2/2 = 100\%$
- if the minimum confidence threshold is 70%, then
rules are strong

Another Example

Example

- frequent itemset: $\{i1, i2, i5\}$
- subsets: $\{i1, i2\}$, $\{i1, i5\}$, $\{i2, i5\}$, $\{i1\}$, $\{i2\}$, $\{i5\}$
- resultant association rules
 - $\{i1, i2\} \rightarrow i5$, confidence = $2/4 = 50\%$
 - $\{i1, i5\} \rightarrow i2$, confidence = $2/2 = 100\%$
 - $\{i2, i5\} \rightarrow i1$, confidence = $2/2 = 100\%$
 - $i1 \rightarrow \{i2, i5\}$, confidence = $2/6 = 33\%$
 - $i2 \rightarrow \{i1, i5\}$, confidence = $2/7 = 29\%$
 - $i5 \rightarrow \{i1, i2\}$, confidence = $2/2 = 100\%$
- if the minimum confidence threshold is 70%, then only the second, third and last rules are strong

Association Rule Mining

- ① find all frequent itemsets
 - by definition, all these itemsets satisfy min_sup
- ② generate strong association rules
 - analyze the frequent itemsets further to extract rules that also satisfy min_conf

Strong Rules are Not Necessarily Interesting

Example

- suppose we have 10,000 transactions

- 6,000 include {Games}
 - 7,500 include {Videos}
 - 4,000 include {Games, Videos}

$\{Games\} \rightarrow \{Videos\}$ [$support = 40\%$, $confidence = 66\%$]

- suppose $min_sup=30\%$ and $min_conf=60\%$, is this rule strong? yes

- is this rule useful?

- the probability one buys videos is $\frac{7,500}{10,000} = 75\% > 66\%$
 - knowing that one buys games actually decreases her probability of buying videos

- one could take unwise business decisions based on the above rule

Summary

We introduced **basic concepts** on **association analysis**, such as

- association rules
- rule support, confidence

Apriori Algorithm

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Fall 2017

Association Rule Mining

- ① find all frequent itemsets
 - by definition, all these itemsets satisfy min_sup
- ② generate strong association rules
 - analyze the frequent itemsets further to extract rules that also satisfy min_conf
 - the problem of association rule mining reduces to finding the frequent itemsets

how to discover frequent itemsets in large transactional databases?

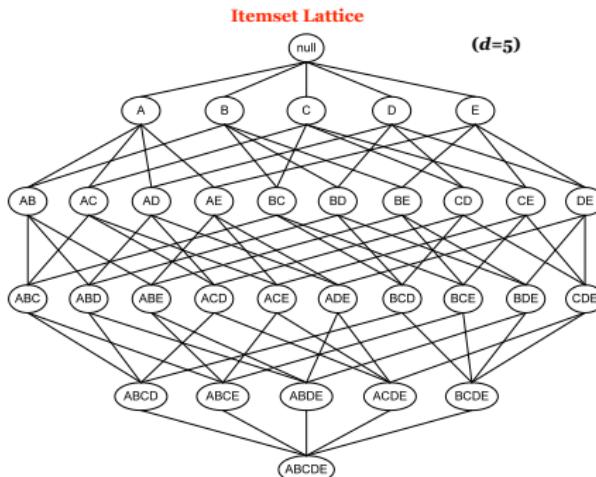
brute-force approach

- ① enumerate **all** candidate itemsets
- ② find the **support count** of each candidate

Brute-Force: Step 1

enumerate all candidate itemsets

- by creating an itemset lattice



- for a dataset that contains d items, the total number of (nonempty) candidate itemsets is $2^d - 1$

Brute-Force: Step 2

find the support count of each candidate

- by scanning the database and matching each transaction against the candidate

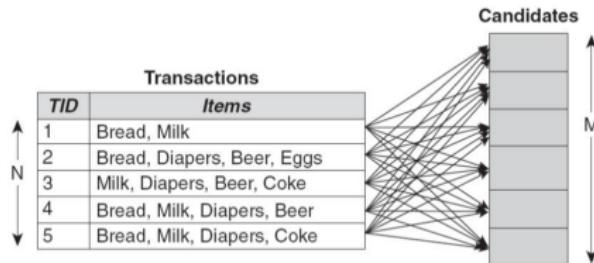


Figure 6.2. Counting the support of candidate itemsets.

```
for each transaction t in database do
    for each candidate contained in t do
        increment the support count;
    end
end
```

potential **problems**

- the total number of candidates can be very huge
- one transaction may contain many candidates

How to Reduce Computational Complexity?

- ① reduce the number of candidate itemsets
 - use (support-based) pruning techniques
- ② obtain the support counts more efficiently
 - use efficient data structures to store the candidates

Algorithms for efficient frequent itemset mining

- **Apriori** (covered in this lecture)
- **FP-growth** (covered in the next lecture)

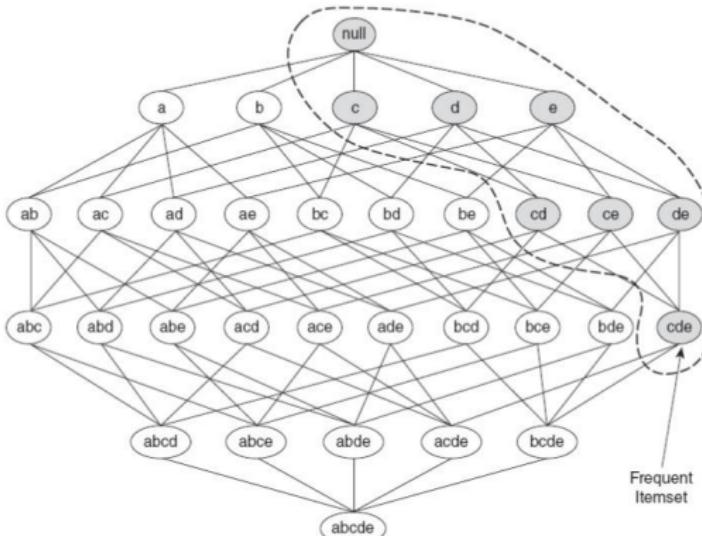
Apriori Principle

Example

if {beer, diaper, nuts} is frequent, so is {beer, diaper}

- every transaction having {beer, diaper, nuts} also contains {beer, diaper}

Any subset of a frequent itemset must be frequent

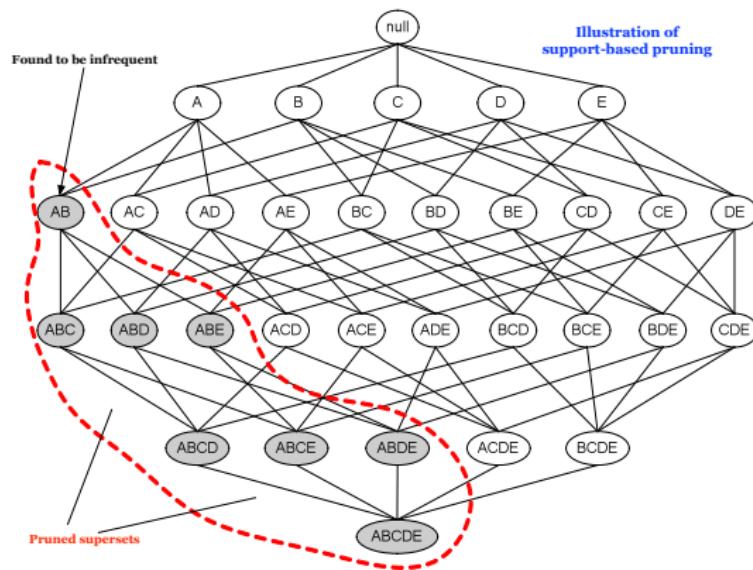


Support-Based Pruning

Antimonotone property of the support measure

\forall itemsets $X, Y, X \subseteq Y$ implies
 $\text{support_count}(X) \geq \text{support_count}(Y)$

- if there is any itemset which is infrequent, its superset should not be generated/tested!



Example

Candidate 1-itemsets

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Minimum support count = 3

Candidate 2-itemsets are produced from the unpruned 1-itemsets

Candidate 2-itemsets

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

{Coke} and {Eggs} are pruned as infrequent

{Bread,Beer} and {Milk,Beer} are pruned as infrequent

Candidate 3-itemsets are produced from the unpruned 2-itemsets

Candidate 3-itemsets

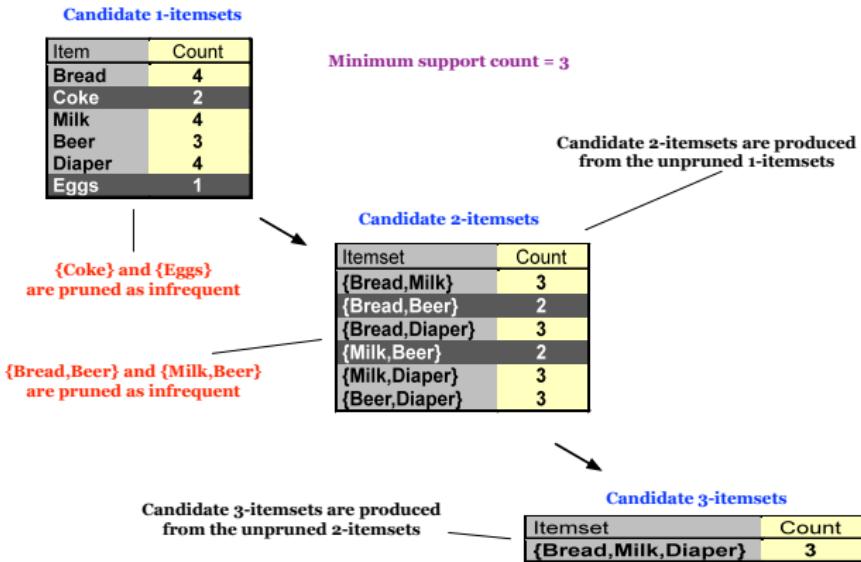
Itemset	Count
{Bread,Milk,Diaper}	3

Apriori Algorithm: Basic Idea

- Step 1:
 - enumerate the candidate **1-itemsets**
 - remove the candidate 1-itemsets that are not frequent by measuring their support counts
- Step 2:
 - generate the candidate **2-itemsets**, based on the remaining **(frequent) 1-itemsets**
 - remove the candidate 2-itemsets that are not frequent by measuring their support counts
- ... [continue the same process until no other itemsets can be generated] ...

The final set of frequent itemsets is the **union** of the itemsets that remained in every step

Effectiveness: Example



Brute-force

- enumerates all itemsets
- produce $\binom{6}{1} + \binom{6}{2} + \binom{6}{3} = 41$ candidates of size up to 3

Apriori

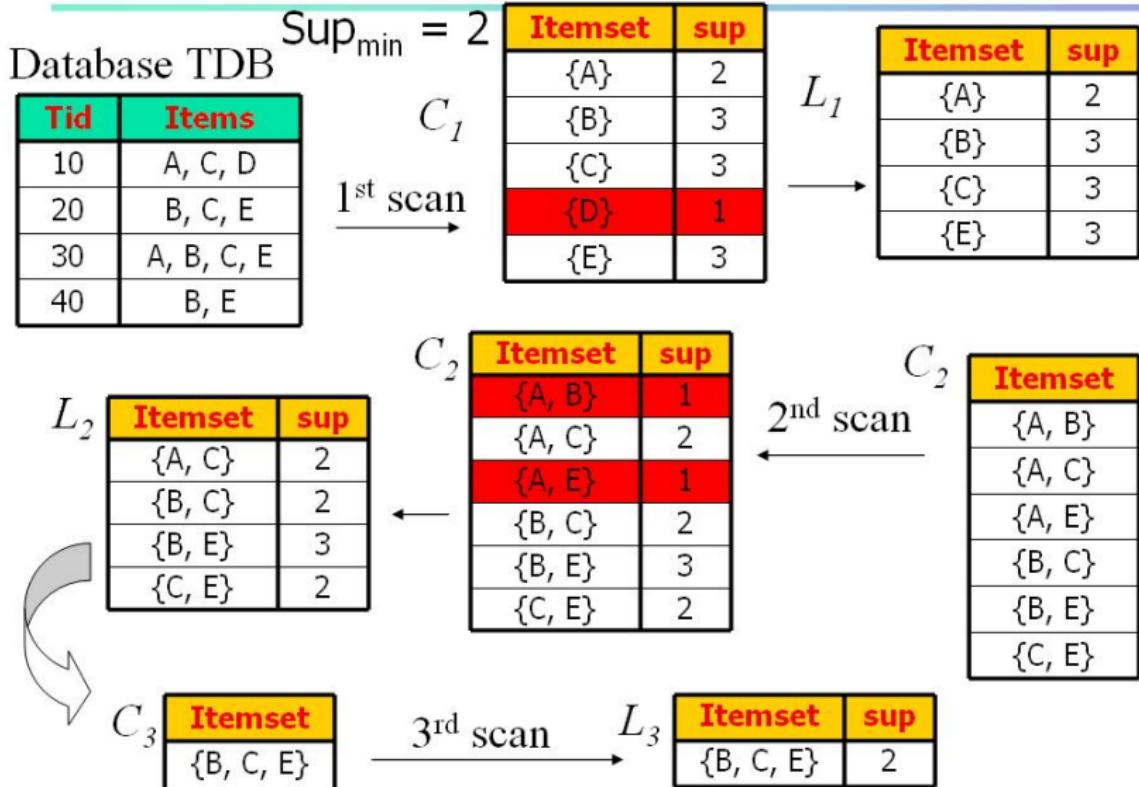
- produces $\binom{6}{1} + \binom{4}{2} + 1 = 13$ candidates of size up to 3

Pseudo-Code

- C_k : candidate itemset of size k
- L_k : frequent itemset of size k

```
 $L_1 = \text{frequent items};$ 
for ( $k = 1; L_k \neq \emptyset; k++$ ) do
     $C_{k+1} = \text{candidates generated from } L_k;$ 
    for each transaction  $t$  in database do
        for each candidate in  $C_{k+1}$  that are contained in  $t$  do
            | increment the support count;
        end
    end
     $L_{k+1} = \text{candidates in } C_{k+1} \text{ above min\_support};$ 
end
return  $\cup_k L_k$ ;
```

Example



Generation of Candidates (Simple Method)

recall that

- based on the frequent $(k - 1)$ -itemsets found in the previous iteration, we generate new candidate k -itemsets

How?

- extend every frequent $(k - 1)$ -itemset with other frequent items

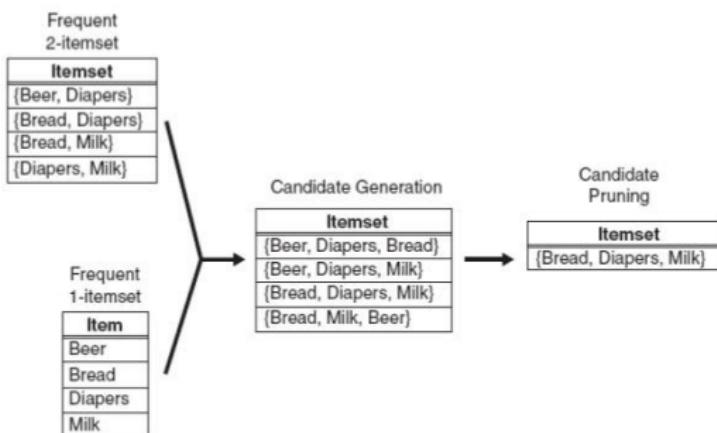


Figure 6.7. Generating and pruning candidate k -itemsets by merging a frequent $(k - 1)$ -itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

Example

$\{bread, diapers, milk\}$ can be generated by merging

- $\{bread, diapers\}$ with $\{milk\}$
- $\{bread, milk\}$ with $\{diapers\}$
- $\{diapers, milk\}$ with $\{bread\}$

How to avoid duplicates?

Avoiding Duplicates

- sort the items in lexicographic order

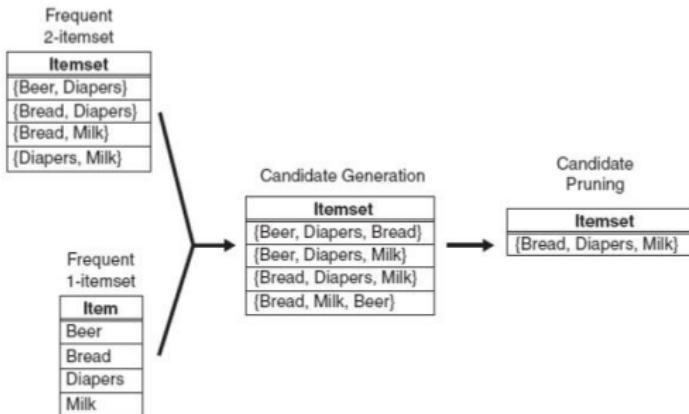


Figure 6.7. Generating and pruning candidate k -itemsets by merging a frequent $(k - 1)$ -itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

- itemset X is extended only by frequent items that are lexicographically larger than the items in X
 - e.g., $\{bread, diapers\}$ can be augmented with $\{milk\}$, but $\{diapers, milk\}$ cannot be augmented with $\{bread\}$

Will this leave out some frequent itemsets?

Candidate Pruning

- may still produce a large number of unnecessary candidates

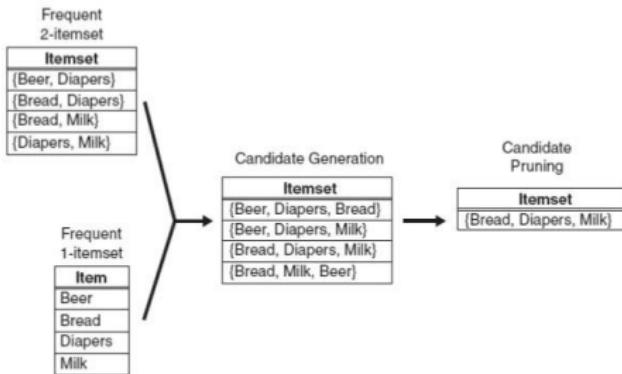


Figure 6.7. Generating and pruning candidate k -itemsets by merging a frequent $(k - 1)$ -itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

from support-based pruning

- for $\{\text{beer, diapers, milk}\}$ to be a candidate 3-itemset, $\{\text{beer, diapers}\}$, $\{\text{diapers, milk}\}$ and $\{\text{beer, milk}\}$ should be frequent 2-itemsets
- $\{\text{beer, diapers, milk}\}$ cannot be a frequent 3-itemset because $\{\text{beer, milk}\}$ is infrequent

Generation of Candidates in Apriori

(self-join) Generate candidate k -itemsets by merging frequent $(k - 1)$ -itemsets

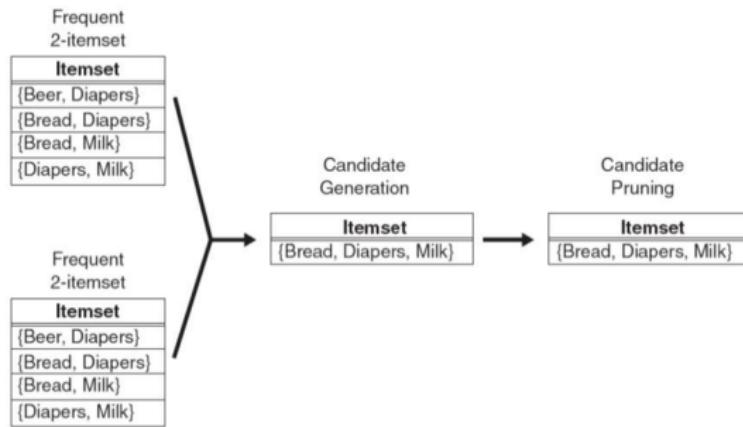


Figure 6.8. Generating and pruning candidate k -itemsets by merging pairs of frequent $(k - 1)$ -itemsets.

- A, B : any pair of $(k - 1)$ -frequent itemsets
 - $A = \{a_1, a_2, \dots, a_{k-1}\}$ and $B = \{b_1, b_2, \dots, b_{k-1}\}$
- merge A and B if and only if their first $(k - 2)$ items are identical
 - i.e., if $a_i = b_i$ for $i = 1, 2, \dots, k - 2$ and $a_{k-1} < b_{k-1}$ (avoid duplicates), then form $\{a_1, a_2, \dots, a_{k-1}, b_{k-1}\}$

Example ($\{beer, diapers, milk\}$)

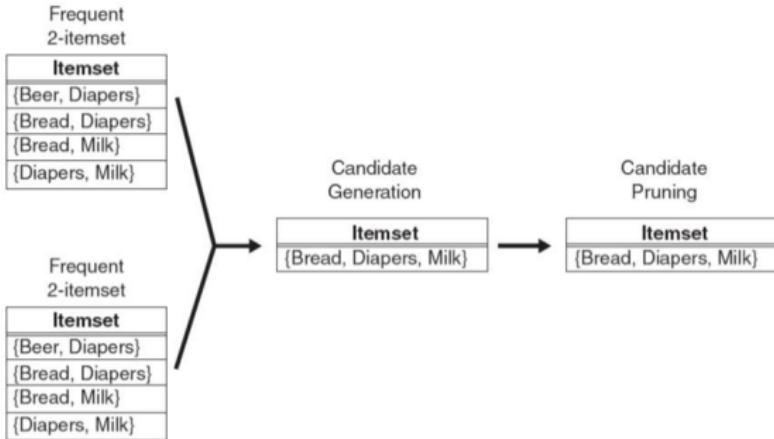


Figure 6.8. Generating and pruning candidate k -itemsets by merging pairs of frequent $(k-1)$ -itemsets.

- does not have to merge $\{beer, diapers\}$ with $\{diapers, milk\}$ because the first item in both itemsets are different
- if $\{beer, diapers, milk\}$ is a frequent itemset, it would have been obtained by merging $\{beer, diapers\}$ with $\{beer, milk\}$ instead

Candidate Pruning

- remove an itemset if it contains a $(k - 1)$ -itemset that is not frequent

Example

- frequent 3-itemsets: $\{abc, abd, acd, ace, bcd\}$
- candidate 4-itemset: $\{abcd, acde\}$
 - $abcd$ from abc and abd
 - $acde$ from acd and ace
- candidate pruning $\rightarrow \{abcd\}$
 - $acde$ is removed because ade is not a frequent 3-itemset

Do we need to check if the candidate contains a

- $(k - 2)$ -itemset,
- $(k - 3)$ -itemset, ...

that is not frequent?

Summary

We introduced **basic concepts** on **association analysis**, such as

- frequent itemsets
- rule extraction from frequent itemsets

FP-Growth Algorithm

Yu Zhang

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
stavrosp@cse.ust.hk

Fall 2017

Drawbacks of Apriori Algorithm

- ① may need to generate a **huge number of candidate itemsets**

Example

If there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate **candidate 2-itemsets**

- ② may need to **repeatedly scan the database**
 - need to go over each transaction in the database in order to compute the support of each candidate itemset

Drawbacks of Apriori Algorithm

- ① may need to generate a **huge number of candidate itemsets**

Example

If there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 candidate 2-itemsets

- ② may need to **repeatedly scan the database**
 - need to go over each transaction in the database in order to compute the support of each candidate itemset

FP-Growth (Frequent Pattern Growth)

Basic Idea: does not generate candidate itemsets

- uses a frequent-pattern tree (FP-tree)
 - a tree that compactly represents the entire transactional database
 - can be much smaller than the original database
 - since the FP-tree is much smaller than the database, mining frequent itemsets directly from the the FP-tree can be much faster
- a divide-and-conquer method for extracting frequent itemsets directly from the FP-tree
 - the entire problem is decomposed into smaller subproblems, which are recursively decomposed into even smaller subproblems
 - eventually, the solutions of the various subproblems are combined together to give the final solution (i.e., the final set of frequent itemsets)

Example

Initial Database

TID	Items
1	{f, a, c, d, g, i, m, p}
2	{a, b, c, f, l, m, o}
3	{b, f, h, j, o}
4	{b, c, k, s, p}
5	{a, f, c, e, l, p, m, n}

- the set of all items: {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p}
- assume that $\text{min_sup}=3$

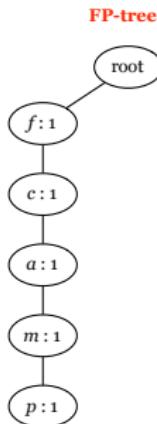
Initialization

Initial Database		Modified Database	
TID	Items	TID	Items
1	{f, a, c, d, g, i, m, p}	1	{f, c, a, m, p}
2	{a, b, c, f, l, m, o}	2	{f, c, a, b, m}
3	{b, f, h, j, o}	3	{f, b}
4	{b, c, k, s, p}	4	{c, b, p}
5	{a, f, c, e, l, p, m, n}	5	{f, c, a, m, p}

- scan the database and derive the list of frequent items in **descending support count** order
 - $\langle (f : 4), (c : 4), (a : 3), (b : 3), (m : 3), (p : 3) \rangle$ where the number after ':' is the item support count
- modified database:**
 - remove the infrequent items
 - sort the remaining items in every transaction according to the above support count order

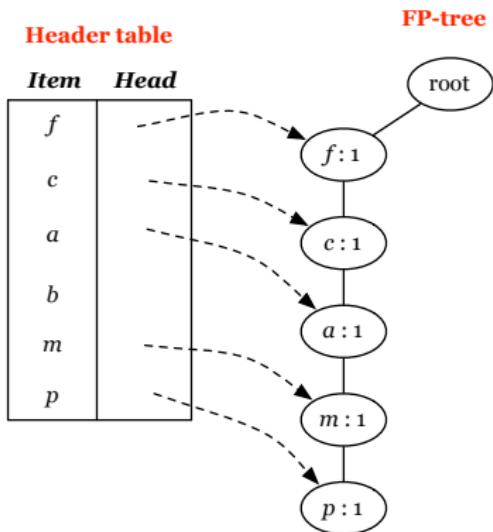
Retrieve the First Transaction

- create a root node
- retrieve the first transaction from the modified database
 - i.e., $\{f, c, a, m, p\}$
- create a **branch** from the root



- every node stores an item along with an initial support count equal to 1
- the items in the branch follow the support count order (i.e., the most frequent item is closer to the root)

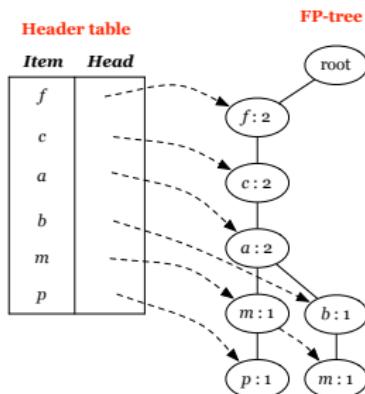
Produce a Header Table



- create an entry for every **frequent** item
- every entry contains an item and a pointer that points to a node storing this item in the tree (if any)
- the entries are sorted based on the descending support count order

Retrieve the Second Transaction ($\{f, c, a, b, m\}$)

- this transaction **shares** the same **prefix** fca with $\{f, c, a, m, p\}$
- instead of creating a new branch from the root for $\{f, c, a, b, m\}$, we create branch bm from the node storing a

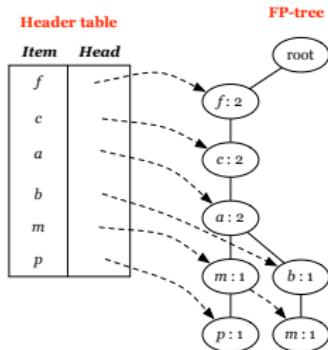


- increment the count of all nodes in prefix fca by one
- create a pointer **from the header table** pointing to the newly created node for b
- create a pointer **from the lastly created node** containing m to the new node containing m
 - facilitates accessing the nodes associated with m

FP-Tree

Modified Database

TID	Items
1	{f, c, a, m, p}
2	{f, c, a, b, m}
3	{f, b}
4	{c, b, p}
5	{f, c, a, m, p}

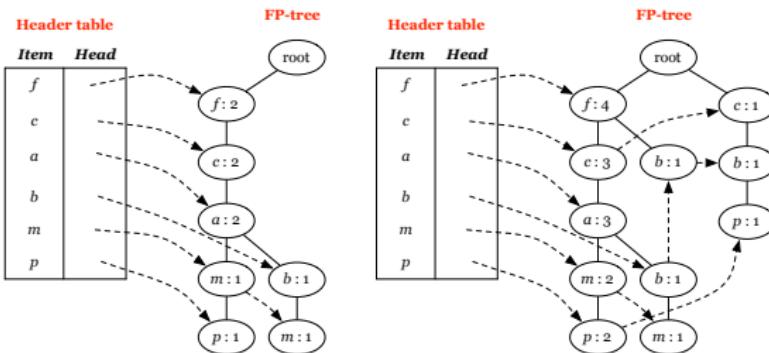


- the FP-tree **compactly** represents $\{f, c, a, m, p\}$ and $\{f, c, a, b, m\}$
 - because the FP-tree stores the **common prefixes** only once
 - sorting items based on descending support count order increases the chance that more prefix itemsets can be shared
- we can **fully recover both transactions** from the tree
 - bottom-up** from the leaves to the root

The problem of mining frequent patterns in databases is transformed into that of **mining the FP-tree**

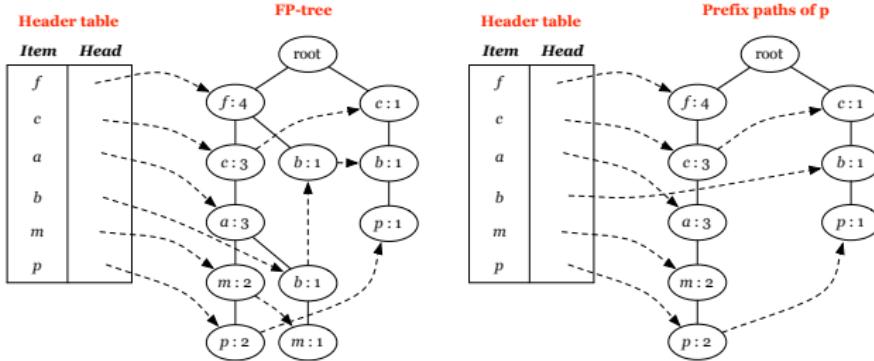
Insert the Transactions One by One

Modified Database	
TID	Items
1	{f, c, a, m, p}
2	{f, c, a, b, m}
3	{f, b}
4	{c, b, p}
5	{f, c, a, m, p}



- exploiting the common prefixes wherever possible
- increment the counts by 1 of the nodes you visit
- initialize the counts of new nodes to 1
- properly update the linked list pointers of all items

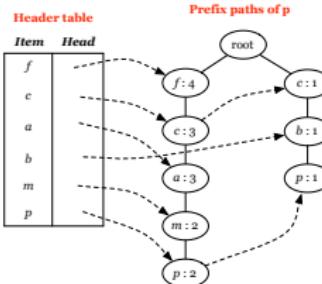
Focus on the Last Item in the Header Table (item p)



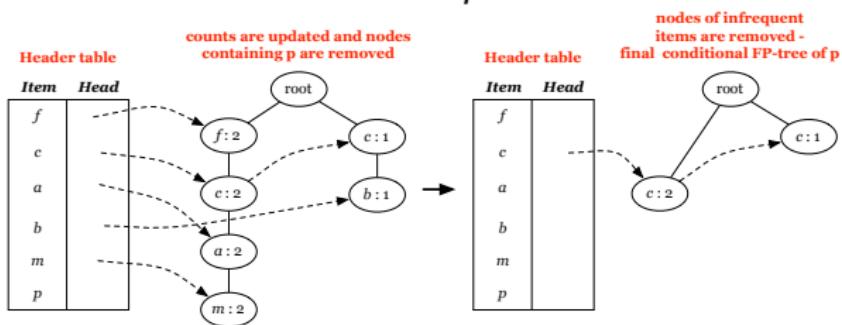
Goal: Produce frequent itemsets that have suffix p

- retrieve the paths (**prefix paths**) in the FP-tree containing p
- see whether p is a frequent item
 - visit all nodes in the tree containing p (by following the **linked-list** pointers starting from the header table)
 - measure the total support count of p
 - if min_sup is not satisfied, then stop
 - since the support of p is $3 = min_sup$, report p as frequent and continue to the next step

Create the Conditional FP-Tree of p

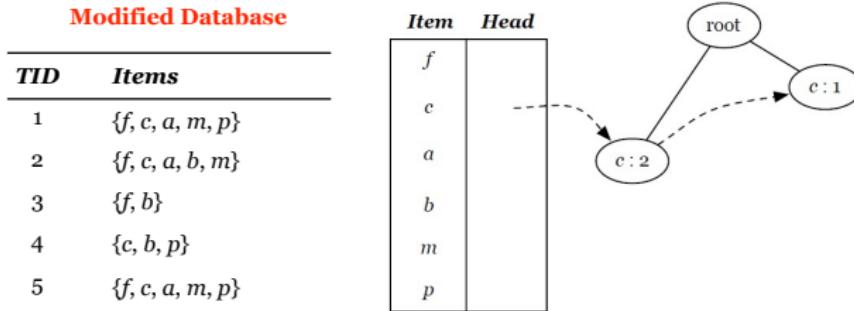


- create the conditional FP-tree of p



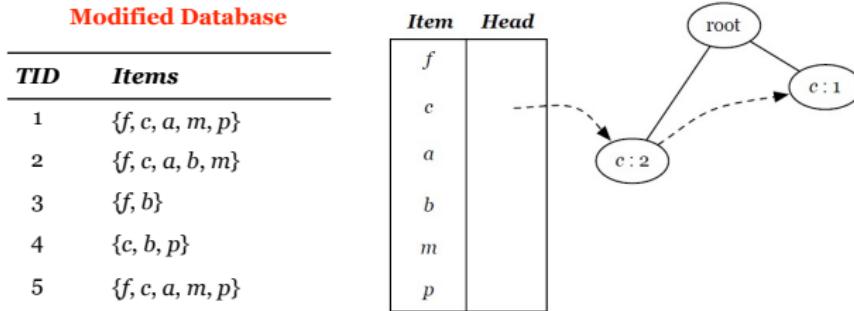
- update the counts in the prefix paths of p in order to reflect the counts only of the transactions containing p
- remove nodes storing p
- remove nodes for items that are no longer frequent in the resulting tree (i.e., their total count does not satisfy min_sup)

Conditional FP-Tree: Remarks



- the conditional FP-tree of p contains only items that can **frequently co-occur** with p
 - FP-growth reports c as a frequent itemset
- call the FP-growth procedure **recursively**, focusing only on the conditional FP-tree of p (**divide-and-conquer**)
 - this will create an conditional FP-tree for c and recursion stops
- since p is present in the same transactions as any itemset encoded by the tree, $\{c\} \cup \{p\}$ will also be frequent

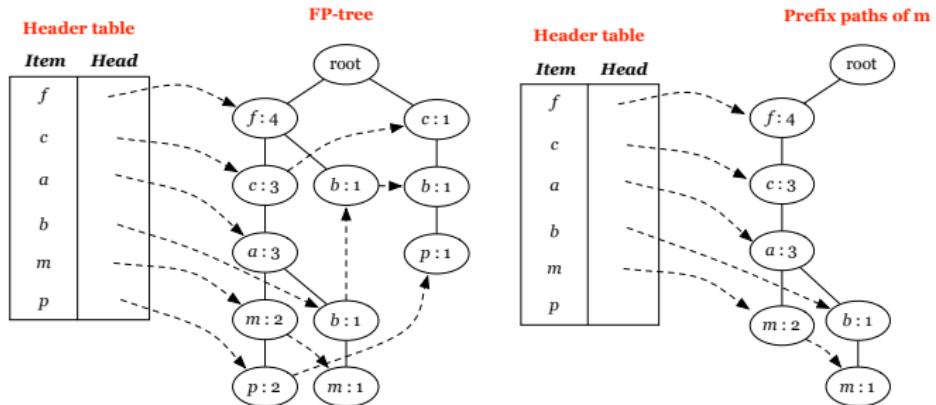
Conditional FP-Tree: Remarks



- the conditional FP-tree of p contains only items that can **frequently co-occur** with p
 - FP-growth reports c as a frequent itemset
- call the FP-growth procedure **recursively**, focusing only on the conditional FP-tree of p (**divide-and-conquer**)
 - this will create an empty conditional FP-tree for c and recursion stops
- since p is present in the same transactions as any itemset encoded by the tree, $\{c\} \cup \{p\}$ will also be frequent

Focus on the Next Item in the Header Table (item m)

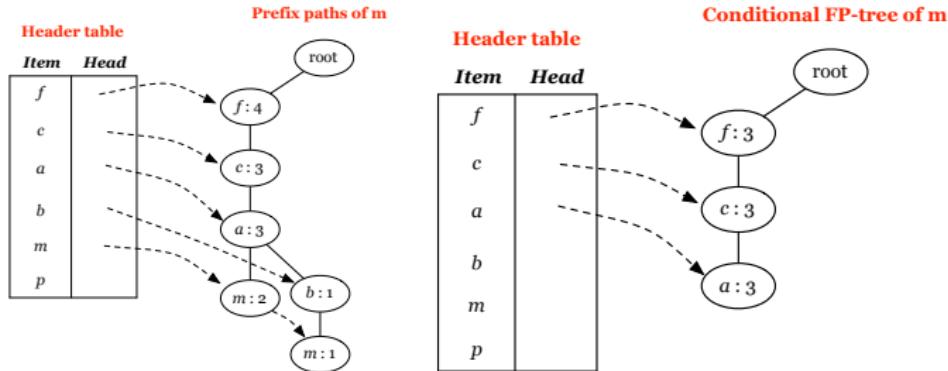
- so far we have covered the case of p
- we must run the same procedure for every other entry of the header table of the initial FP-tree



- let us focus on the next item, namely m
 - we will produce all the frequent itemsets that have m as suffix
- get its prefix paths
 - measure its support count and report it as frequent

Focus on the Next Item in the Header Table (item m)...

- ③ create the conditional FP-tree of m



- ④ recursively call FP-growth on the conditional FP-tree of m
- will return the frequent itemsets that have as suffix f , c and a
 - indeed, these are f , c , fc , a , ca , fa , fca
- ⑤ append m to these reported itemsets, in order to produce the frequent itemsets that have m as a suffix (in addition to the already reported singleton itemset m)
- i.e., fm , cm , fcm , am , cam , fam , $fcam$

Try out the remaining cases of b , a , c and f , in this order

FP-growth Algorithm: Pseudo-Code

The FP-growth Algorithm

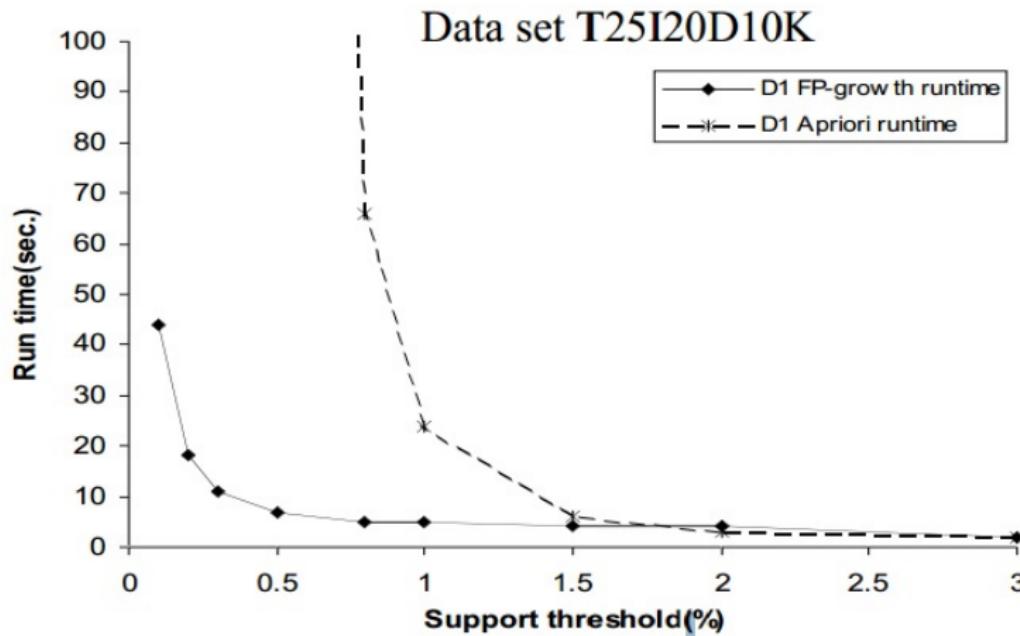
Input: FP-tree Tree

Output: Set of frequent itemsets F

1. $F = \emptyset$
2. **For each** item i in the header table of Tree
3. find the prefix paths of i on Tree
4. **If** i 's support satisfies min_sup , $F = F \cup \{i\}$
5. build the conditional FP-tree Tree_i of i on the prefix paths
6. **If** $\text{Tree}_i \neq \emptyset$, $F_i = \text{FP-growth}(\text{Tree}_i)$
7. append i to every $S \in F_i$, and add the resulting sets to F
8. **Return** F

Experiments: FP-Growth vs. Apriori

- on many real datasets, FP-growth is significantly faster than the Apriori algorithm



Summary

- This lecture covered the **FP-growth** algorithm for **frequent itemset mining**

SOCIAL NETWORKS MINING

Yu Zhang

COMP4331: Data Mining

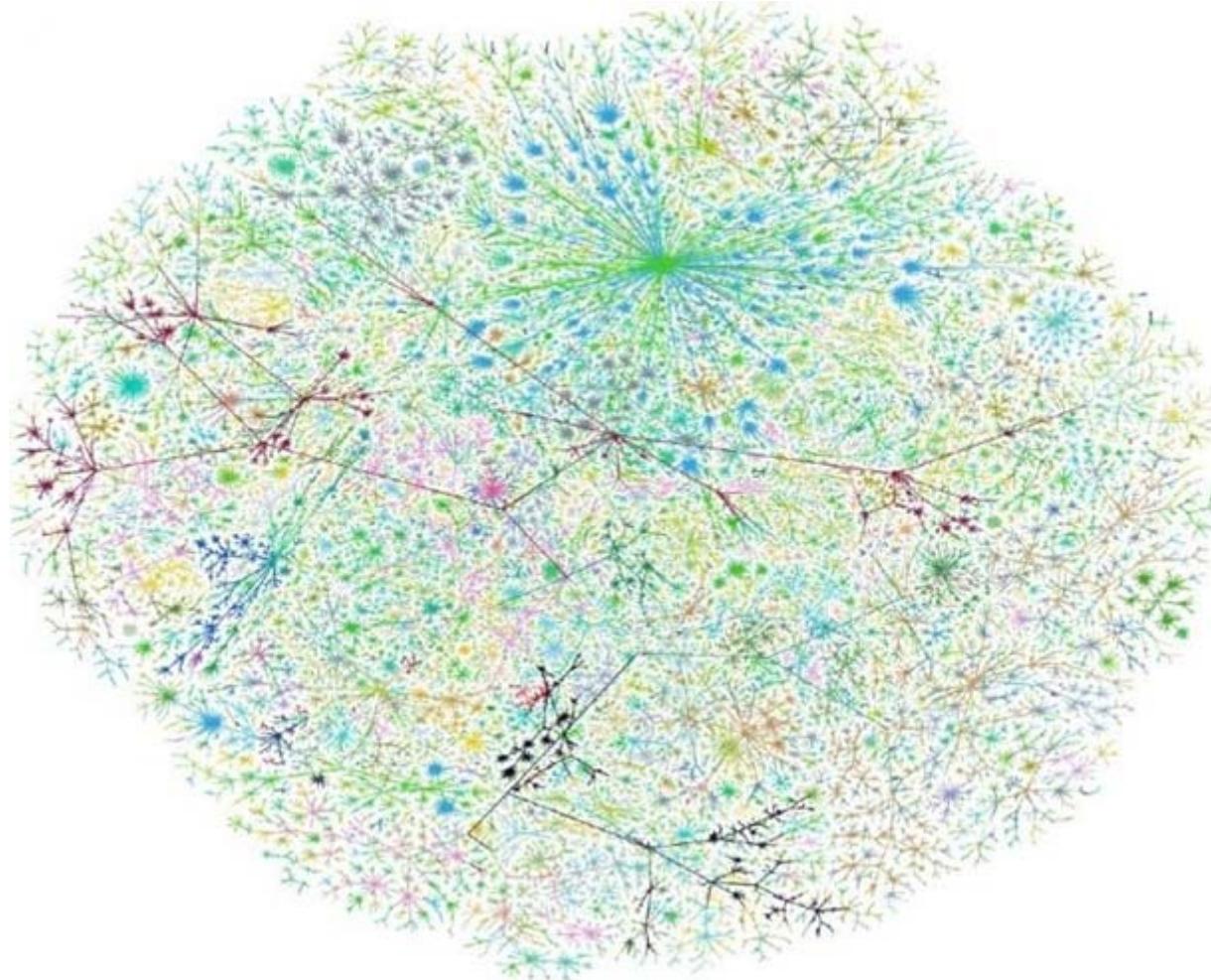
Fall 2017



Networks & Complex Systems

- **Hopelessly complex systems are around us:**
 - **Society** is a collection of six billion individuals
 - **Communication systems** link electronic devices
 - **Information** and **knowledge** is organized and linked
 - Thousands of **genes** in our cells work together in a seamless fashion
 - Our **thoughts** are hidden in the connections between billions of neurons in our brain

**What do these systems have in common?
How can we represent them?**



The Network!

Networks!!



Behind each such system there is an intricate wiring diagram, **a network**, that defines the **interactions** between the components

We will never understand these systems unless we understand the networks behind it

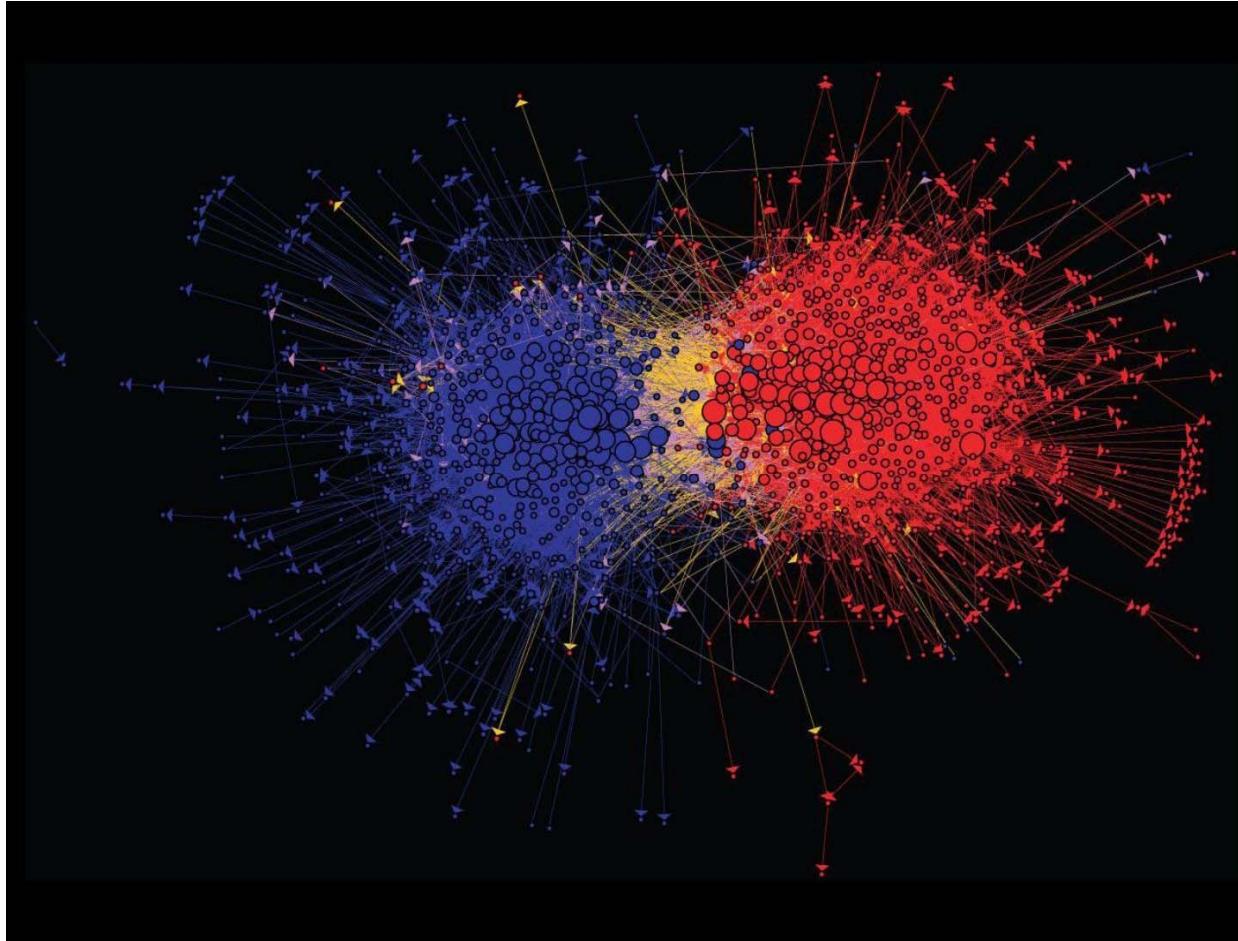
Networks: Social



Facebook social graph

4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]

Networks: Media



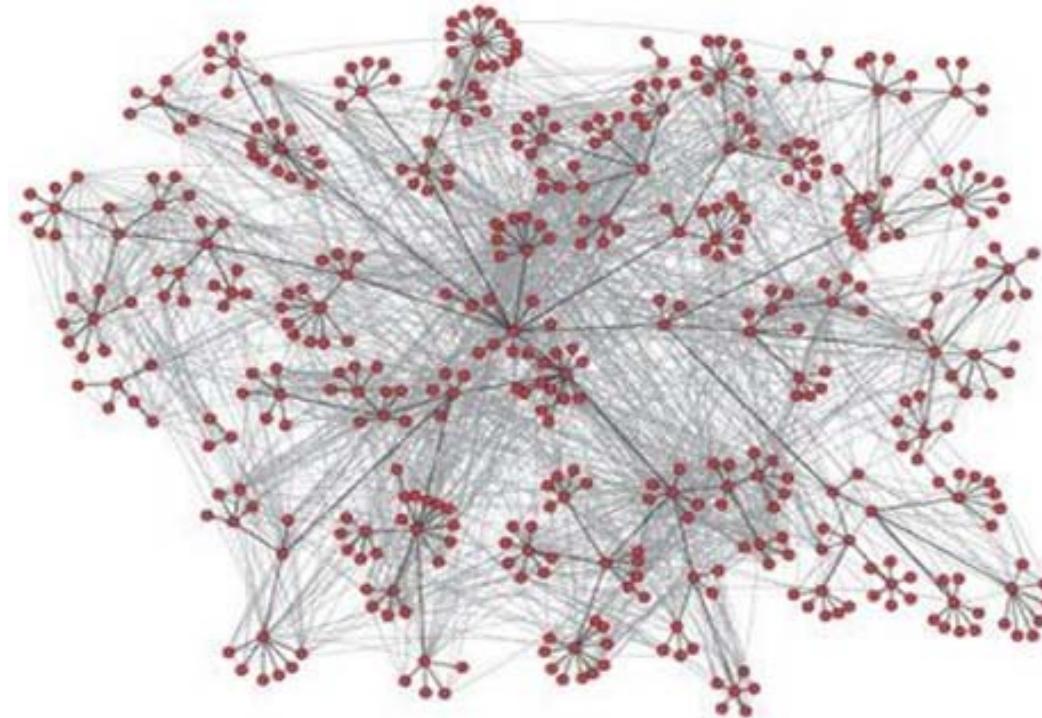
Connections between political blogs
Polarization of the network [Adamic-Glance, 2005]

Networks: Size Matters

□ Network data: Orders of magnitude

- **436-node** network of email exchange at a corporate research lab [Adamic-Adar, SocNets '03]
- **43,553-node** network of email exchange at an university [Kossinets-Watts, Science '06]
- **4.4-million-node** network of declared friendships on a blogging community [Liben-Nowell et al., PNAS '05]
- **240-million-node** network of communication on Microsoft Messenger [Leskovec-Horvitz, WWW '08]
- **800-million-node** Facebook network [Backstrom et al. '11]

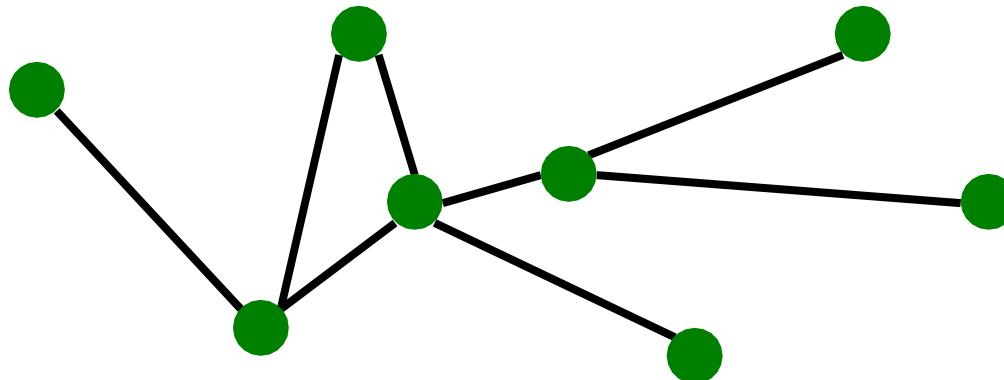
Structure of Networks?



Network is a collection of objects where some pairs of objects are connected by links

What is the structure of the network?

Components of a Network



- **Objects:** nodes, vertices N
- **Interactions:** links, edges E
- **System:** network, graph $G(N,E)$

Networks or Graphs?

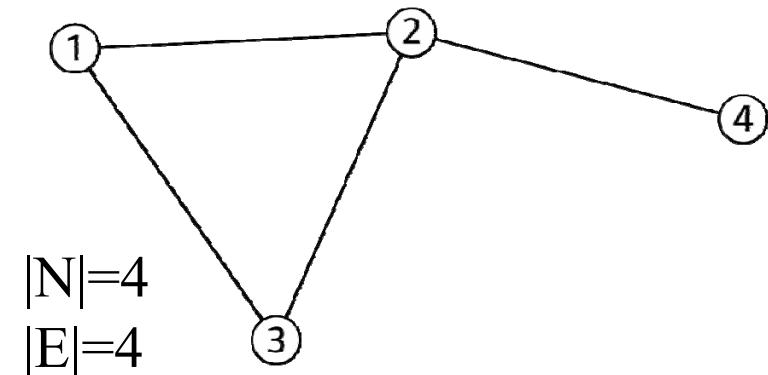
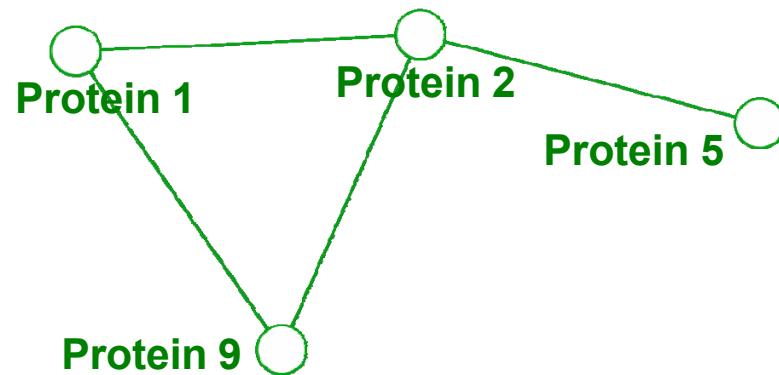
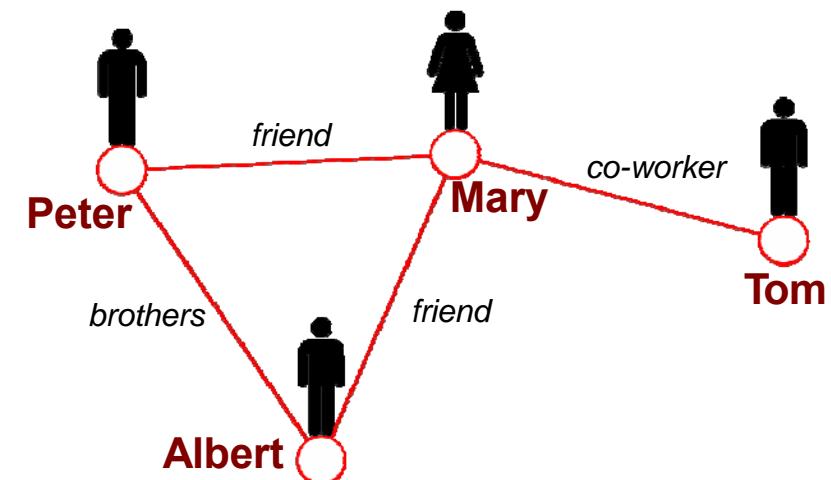
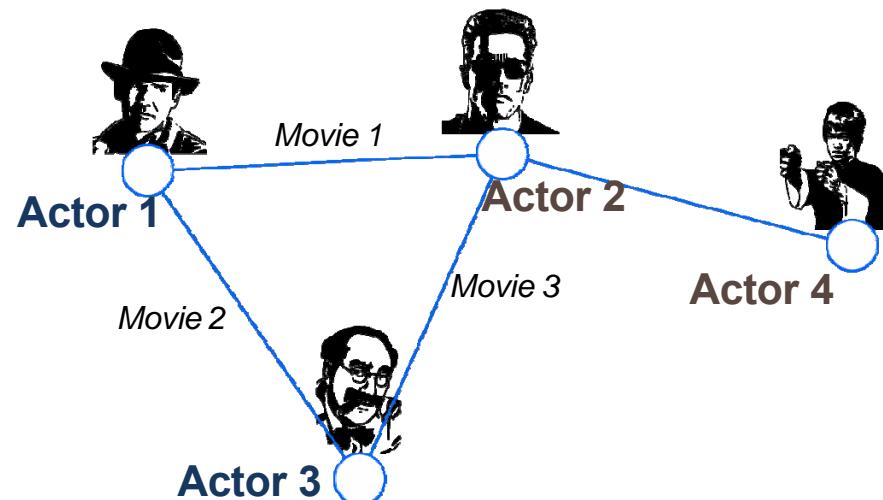
- **Network** often refers to real systems
 - Web, Social network, Metabolic network

Language: Network, node, link
- **Graph:** mathematical representation of a network
 - Web graph, Social graph (a Facebook term)

Language: Graph, vertex, edge

We will try to make this distinction whenever it is appropriate, but in most cases we will use the two terms interchangeably

Networks: Common Language

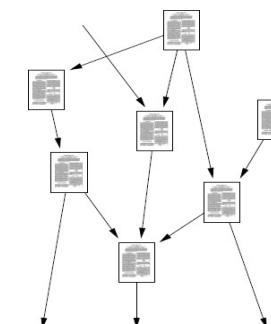
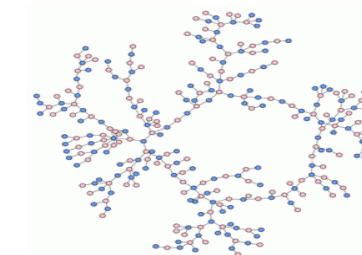


Choosing Proper Representation

- 
- Choice of the proper network representation determines our ability to use networks successfully:
 - In some cases there is a unique, unambiguous representation
 - In other cases, the representation is by no means unique
 - The way you assign links will determine the nature of the question you can study

Choosing Proper Representation

- If you connect individuals that work with each other, you will explore a **professional network**
- If you connect those that have a sexual relationship, you will be exploring **sexual networks**
- If you connect scientific papers that cite each other, you will be studying the **citation network**



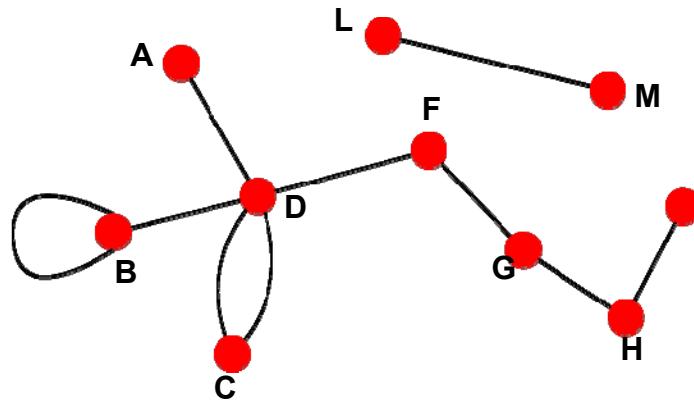
NETWORK PROPERTIES: HOW TO CHARACTERIZE A NETWORK?



Undirected vs. Directed Networks

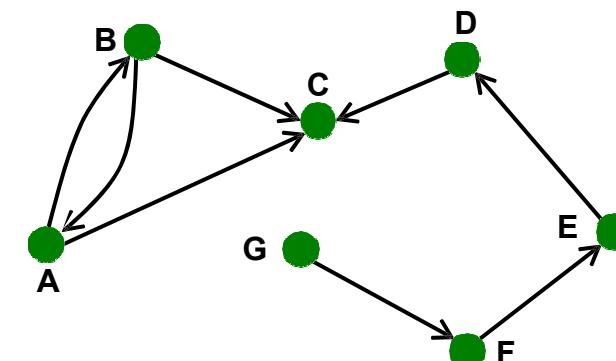
Undirected

- Links: undirected (symmetrical)



Directed

- Links: directed (arcs)



- Examples:

- Collaborations
- Friendship on Facebook

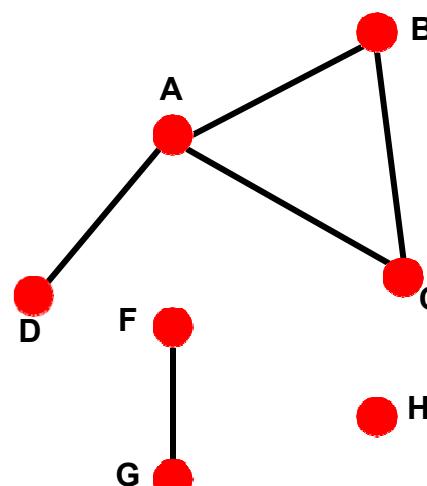
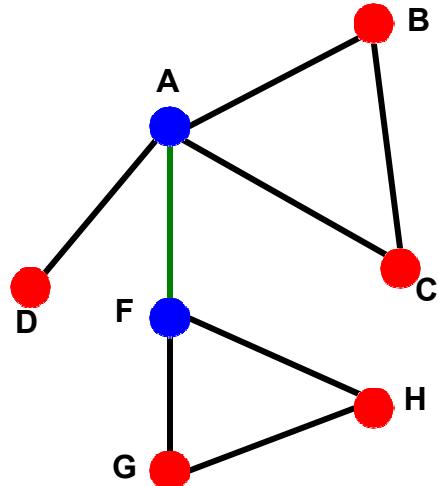
- Examples:

- Phone calls
- Following on Twitter

Connectivity of Graphs

□ Connected (undirected) graph:

- Any two vertices can be joined by a path.
- A disconnected graph is made up by two or more connected components



Largest Component:
Giant Component

Isolated node (node H)

Bridge edge: If we erase it, the graph becomes disconnected.

Articulation point: If we erase it, the graph becomes disconnected.

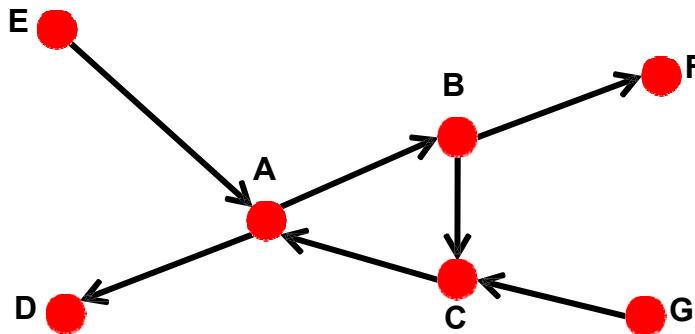
Connectivity of Directed Graphs

- **Strongly connected directed graph**

- has a path from each node to every other node and vice versa (e.g., A-B path and B-A path)

- **Weakly connected directed graph**

- is connected if we disregard the edge directions



Graph on the left is connected but not strongly connected (e.g., there is no way to get from F to G by following the edge directions).

Directed Graphs

- Two types of directed graphs:

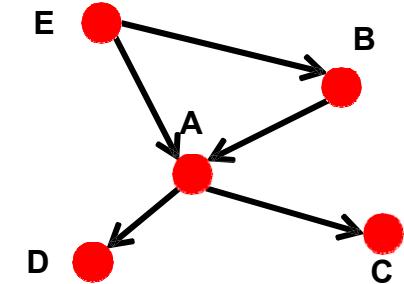
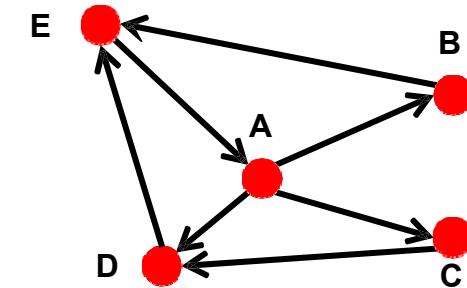
- Strongly connected:

- Any node can reach any node via a directed path

- DAG – Directed Acyclic Graph:

- Has no cycles: if u can reach v , then v can not reach u

- Any directed graph can be expressed in terms of these two types!

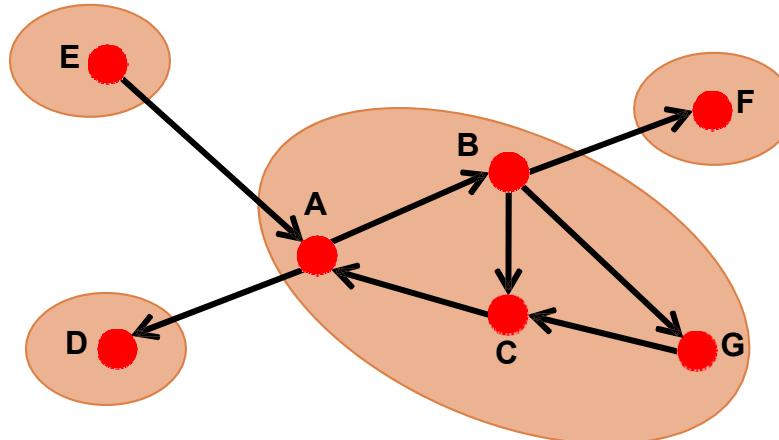


Strongly Connected Component

□ Strongly connected component (SCC)

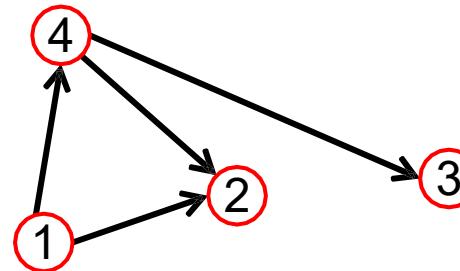
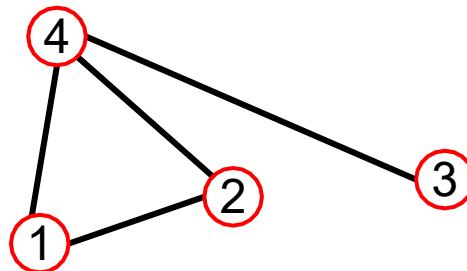
is a set of nodes S so that:

- Every pair of nodes in S can reach each other
- There is no larger set containing S with this property



Strongly connected
components of the graph:
 $\{A,B,C,G\}$, $\{D\}$, $\{E\}$, $\{F\}$

Adjacency Matrix



$A_{ij} = 1$ if there is a link from node i to node j

$A_{ij} = 0$ otherwise

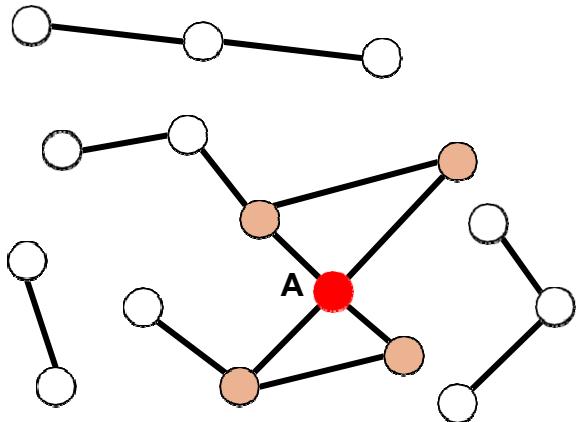
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

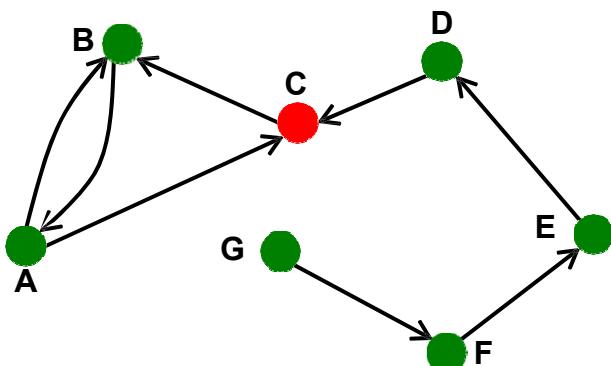
Note that for a directed graph (right) the matrix is not symmetric.

Node Degrees

Undirected



Directed



Source: node with $k^{in} = 0$

Sink: node with $k^{out} = 0$

Node degree, k_i : the number of edges adjacent to node i

$$k_A = 4$$

Avg. degree: $\bar{k} = \langle k \rangle = \frac{1}{N} \sum_{i=1}^N k_i = \frac{2E}{N}$

In directed networks we define an **in-degree** and **out-degree**. The (total) degree of a node is the sum of in- and out-degrees.

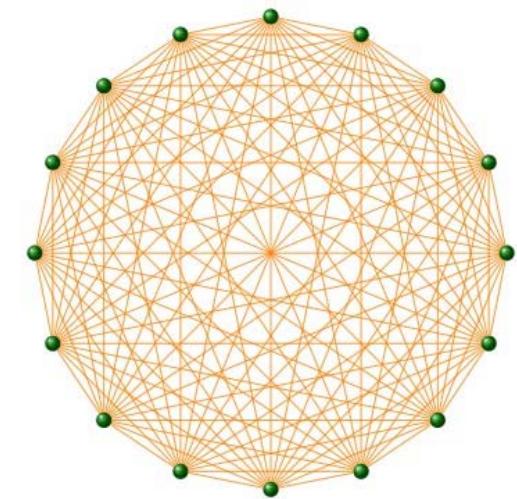
$$k_C^{in} = 2 \quad k_C^{out} = 1 \quad k_C = 3$$

$$\bar{k} = \frac{E}{N} \quad \overline{k^{in}} = \overline{k^{out}}$$

Complete Graph

The **maximum number of edges** in an undirected graph on N nodes is

$$E_{\max} = \binom{N}{2} = \frac{N(N-1)}{2}$$



A graph with the number of edges $E = E_{\max}$ is a **complete graph**, and its average degree is $N-1$

Networks are Sparse Graphs

Most real-world networks are **sparse**

$$E \ll E_{\max} \text{ (or } k \ll N-1)$$

WWW (Stanford-Berkeley):	$N=319,717$	$\langle k \rangle=9.65$
Social networks (LinkedIn):	$N=6,946,668$	$\langle k \rangle=8.87$
Communication (MSN IM):	$N=242,720,596$	$\langle k \rangle=11.1$
Coauthorships (DBLP):	$N=317,080$	$\langle k \rangle=6.62$
Internet (AS-Skitter):	$N=1,719,037$	$\langle k \rangle=14.91$
Roads (California):	$N=1,957,027$	$\langle k \rangle=2.82$
Protein (S. Cerevisiae):	$N=1,870$	$\langle k \rangle=2.39$

(Source: Leskovec et al., *Internet Mathematics*, 2009)

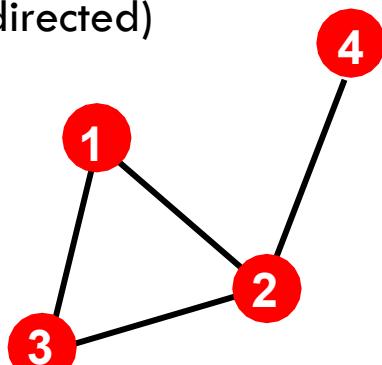
Consequence: Adjacency matrix is filled with zeros!

(**Density (E/N^2):** WWW= 1.51×10^{-5} , MSN IM = 2.27×10^{-8})

More Types of Graphs:

□ Unweighted

(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0$$

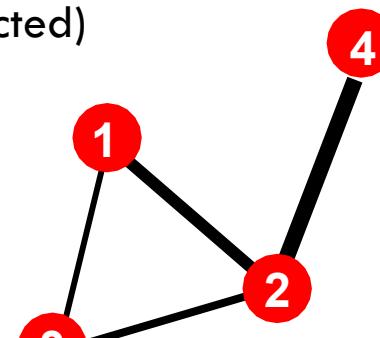
$$A_{ij} = A_{ji}$$

$$E = \frac{1}{2} \sum_{i,j=1}^N A_{ij} \quad k = \frac{2E}{N}$$

Examples: Friendship, Sex

□ Weighted

(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0$$

$$A_{ij} = A_{ji}$$

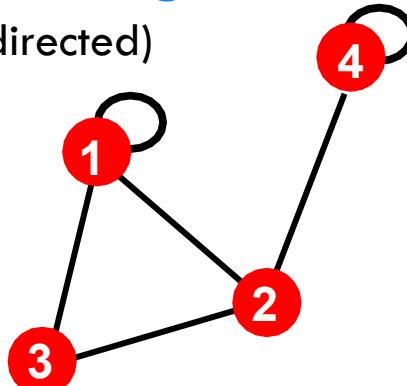
$$E = \frac{1}{2} \sum_{i,j=1}^N \text{nonzero}(A_{ij}) \quad \bar{k} = \frac{2E}{N}$$

Examples: Collaboration, Internet, Roads

More Types of Graphs:

□ Self-edges (self-loops)

(undirected)



$$A_{ij} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$$A_{ii} \neq 0$$

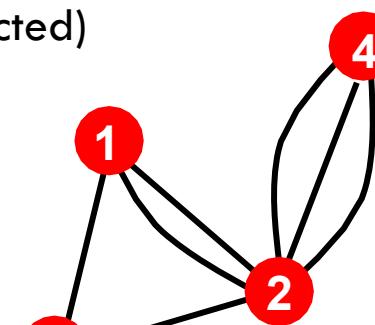
$$A_{ij} = A_{ji}$$

$$E = \frac{1}{2} \sum_{i,j=1, i \neq j}^N A_{ij} + \sum_{i=1}^N A_{ii}$$

Examples: Proteins, Hyperlinks

□ Multigraph

(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0$$

$$A_{ij} = A_{ji}$$

$$E = \frac{1}{2} \sum_{i,j=1}^N \text{nonzero}(A_{ij}) \quad \bar{k} = \frac{2E}{N}$$

Examples: Communication, Collaboration

Network Representations



WWW >> directed multigraph with self-interactions

Facebook friendships >> undirected, unweighted

Citation networks >> unweighted, directed, acyclic

Collaboration networks >> undirected multigraph or weighted graph

Mobile phone calls >> directed, (weighted?) multigraph

Protein Interactions >> undirected, unweighted with self-interactions

Bipartite Graph

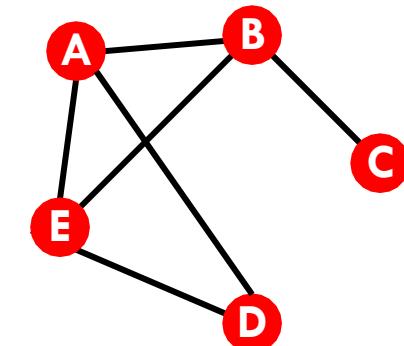
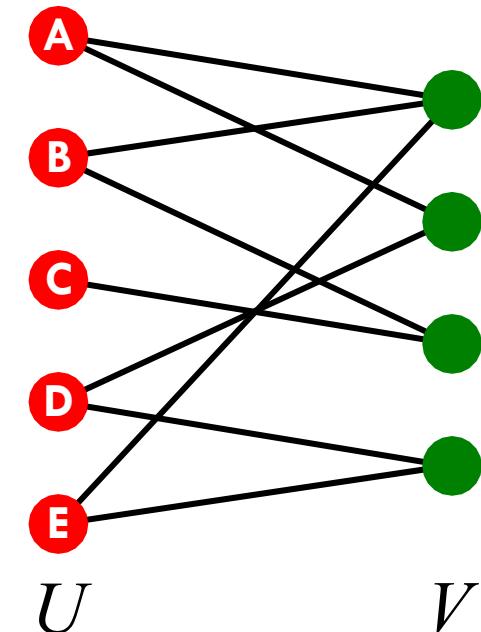
- **Bipartite graph** is a graph whose nodes can be divided into two disjoint sets U and V such that every link connects a node in U to one in V ; that is, U and V are independent sets.

- **Examples:**

- Authors-to-papers (they authored)
- Actors-to-Movies (they appeared in)
- Users-to-Movies (they rated)

- **“Folded” networks:**

- Author collaboration networks
- Movie co-rating networks



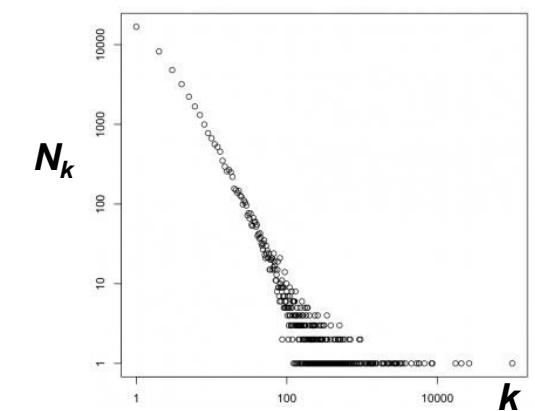
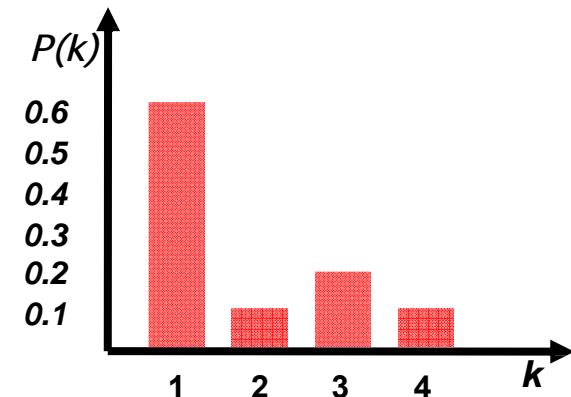
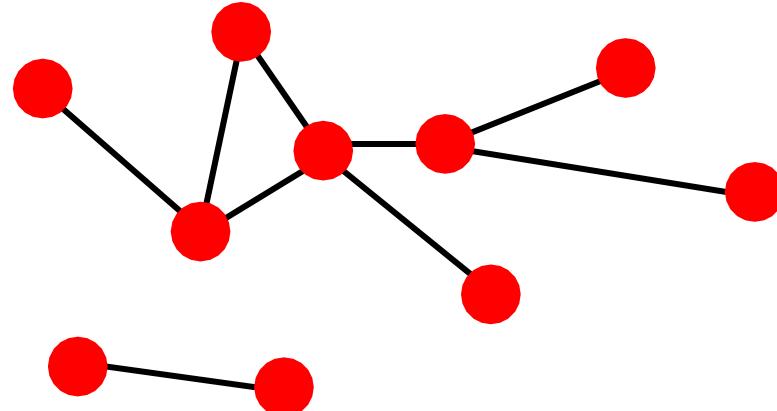
Degree Distribution

- **Degree distribution $P(k)$:** Probability that a randomly chosen node has degree k

$$N_k = \# \text{ nodes with degree } k$$

- Normalized histogram:

$$P(k) = N_k / N \quad \rightarrow \quad \text{plot}$$

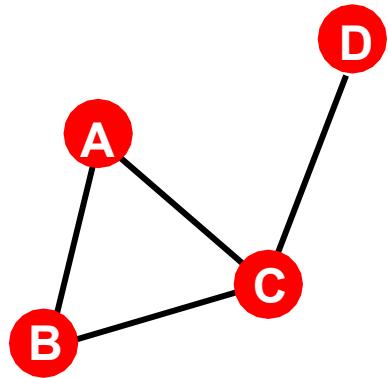


Distance in a Graph

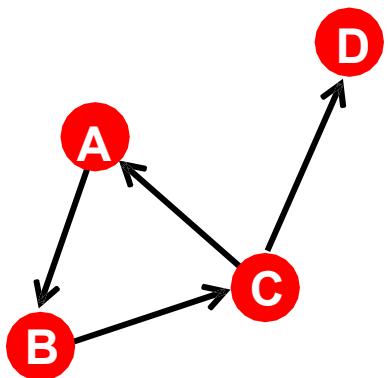
□ **Distance (shortest path, geodesic)**

between a pair of nodes is defined as the number of edges along the shortest path connecting the nodes

- *If the two nodes are disconnected, the distance is usually defined as infinite



$$h_{B,D} = 2$$



$$h_{B,C} = 1, h_{C,B} = 2$$

□ In **directed graphs** paths need to follow the direction of the arrows

- Consequence: Distance is not symmetric:
 $h_{A,C} \neq h_{C,A}$

Network Diameter

- **Diameter:** the maximum (shortest path) distance between any pair of nodes in a graph
- **Average path length** for a connected graph (component) or a strongly connected (component of a) directed graph

$$\bar{h} = \frac{1}{2E_{\max}} \sum_{i,j \neq i} h_{ij} \quad \text{where } h_{ij} \text{ is the distance from node } i \text{ to node } j$$

- Many times we compute the average only over the connected pairs of nodes (we ignore “infinite” length paths)

Clustering Coefficient

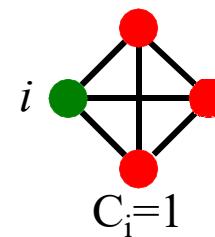
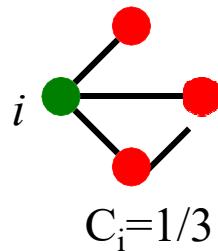
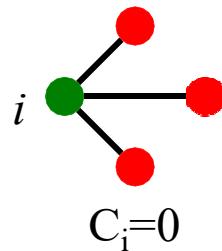
□ Clustering coefficient:

- What portion of i 's neighbors are connected?
- Node i with degree k_i
- $C_i \in [0,1]$



$$C_i = \frac{2e_i}{k_i(k_i - 1)}$$

where e_i is the number of edges between the neighbors of node i



□ Average Clustering Coefficient:

$$C = \frac{1}{N} \sum_i^N C_i$$

Key Network Properties



Degree distribution: $P(k)$

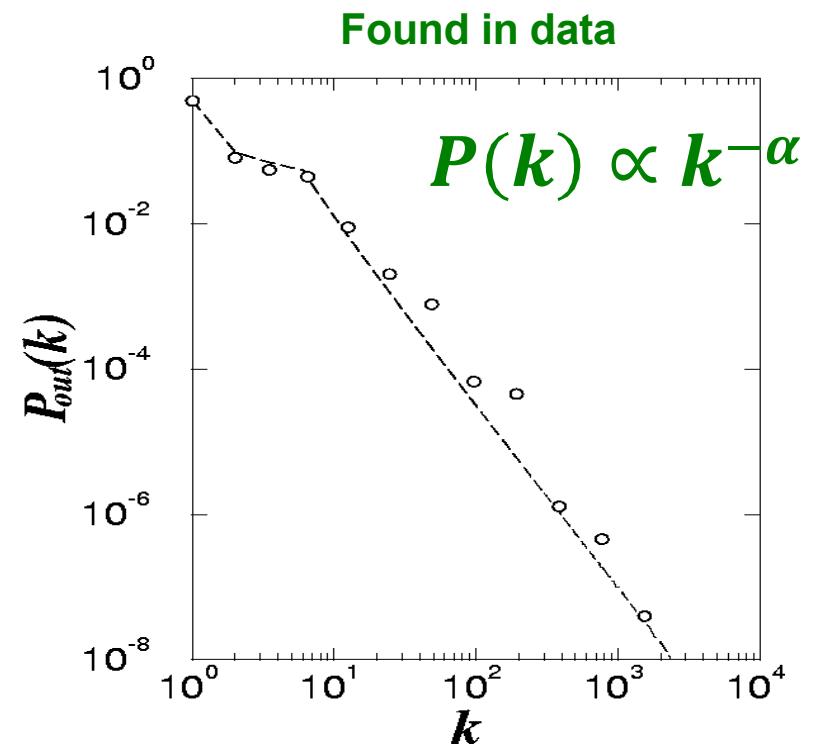
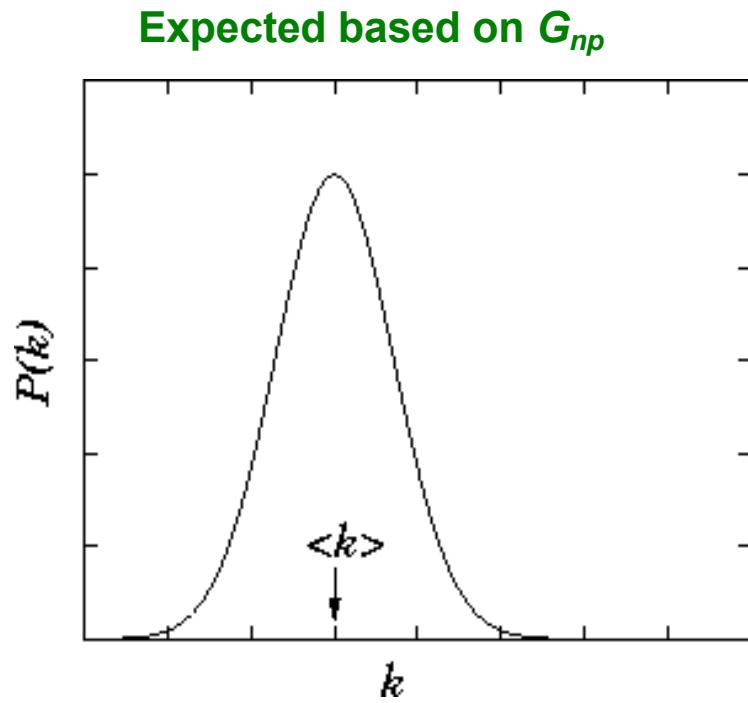
Path length: h

Clustering coefficient: C

POWER-LAW DISTRIBUTION

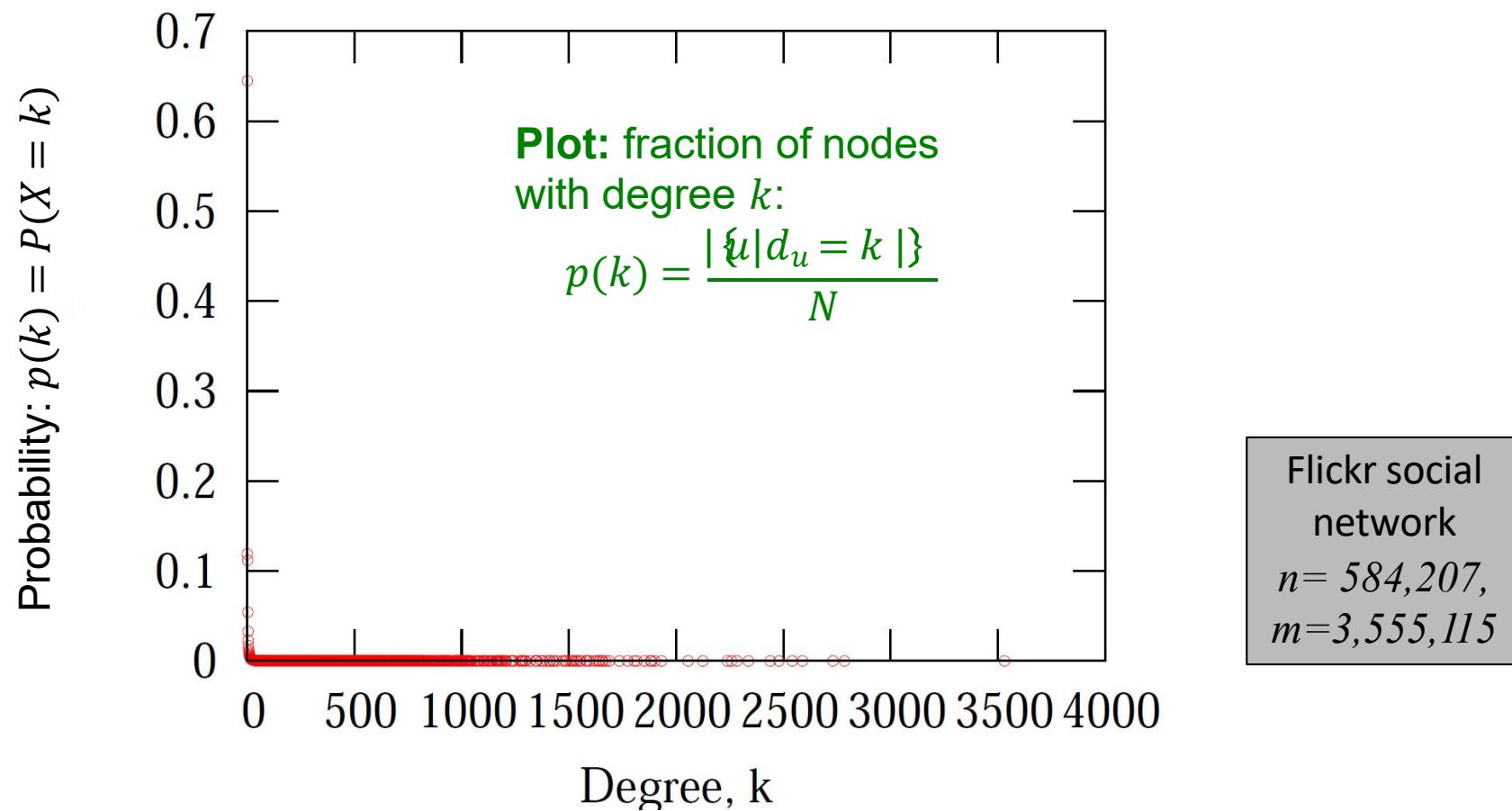


Degree Distributions



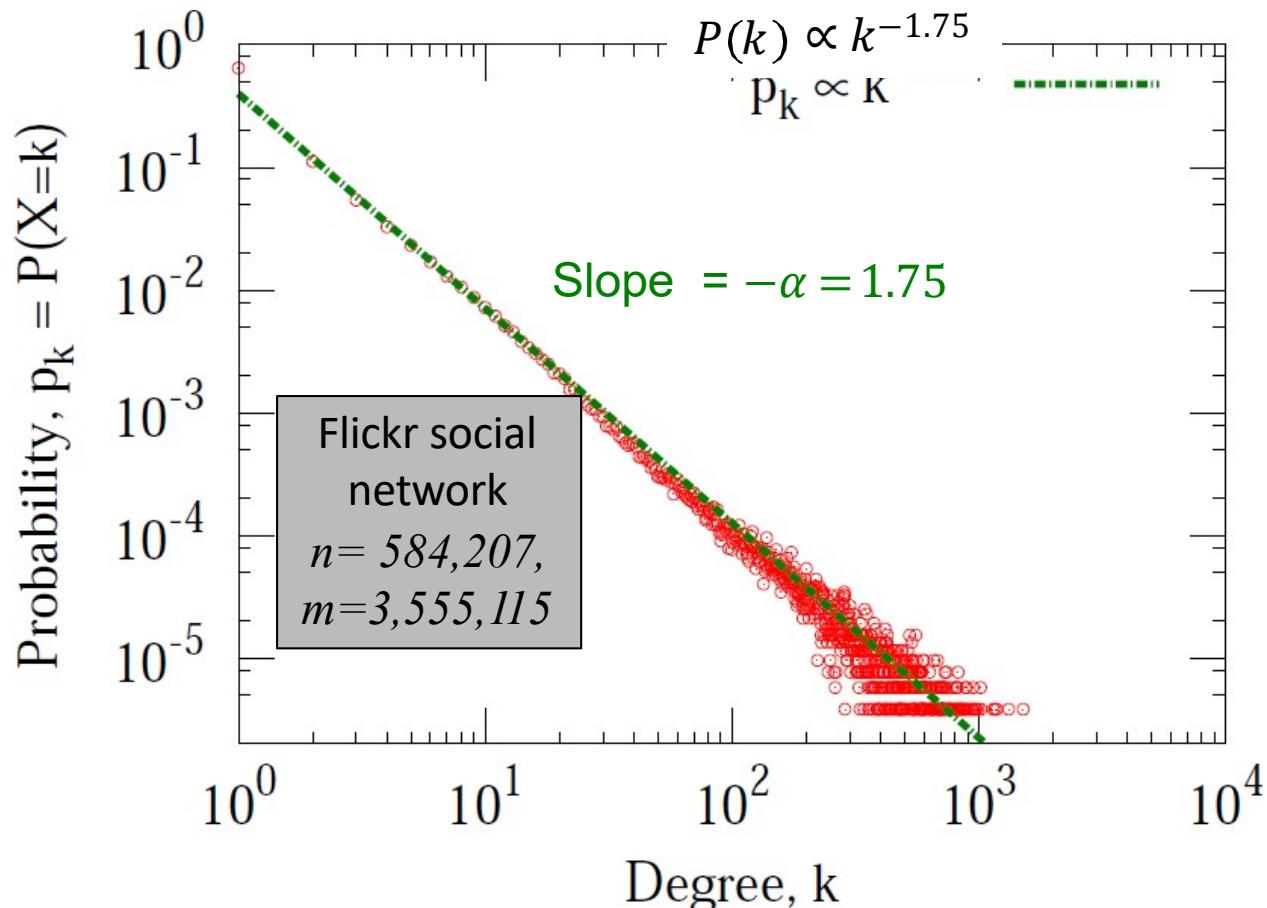
Node Degrees in Networks

- Take a network, plot a histogram of $P(k)$ vs. k



Node Degrees in Networks

- Plot the same data on *log-log scale*:

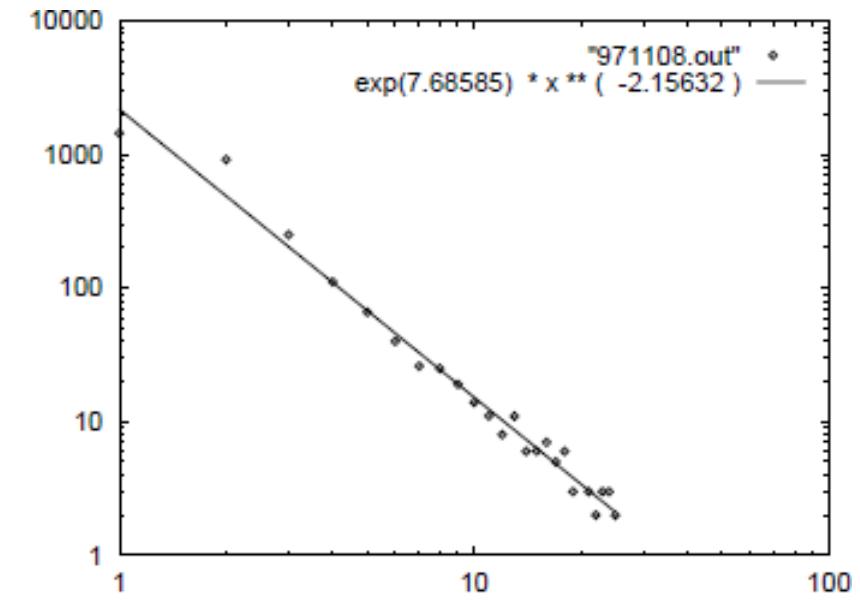
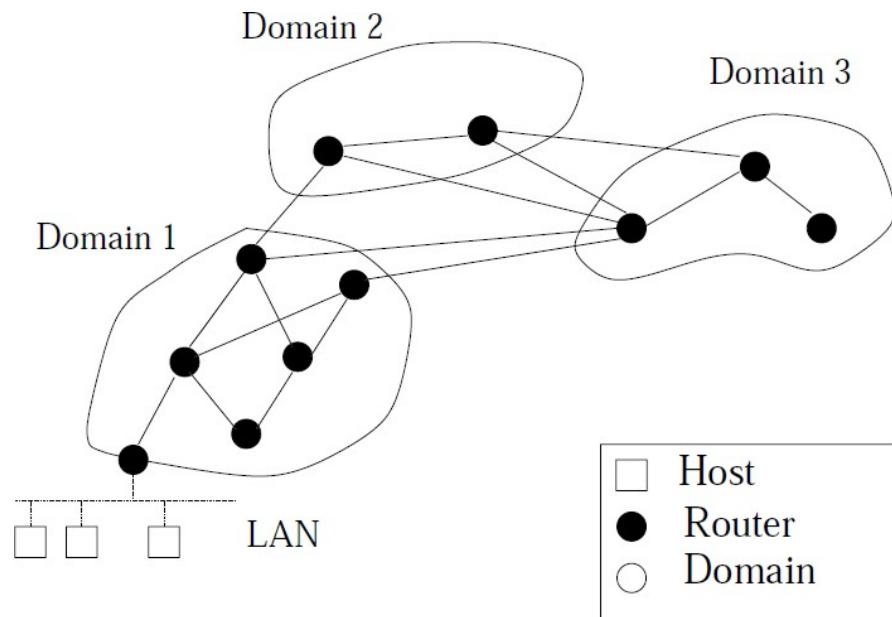


How to distinguish:
 $P(k) \propto \exp(-k)$ vs.
 $P(k) \propto k^{-\alpha}$?

Take logarithms:
if $y = f(x) = e^{-x}$ then
 $\log(y) = -x$
If $y = x^{-\alpha}$ then
 $\log(y) = -\alpha \log(x)$
So, on log-log axis
power-law looks like
a straight line of
slope $-\alpha$!

Node Degrees: Faloutsos³

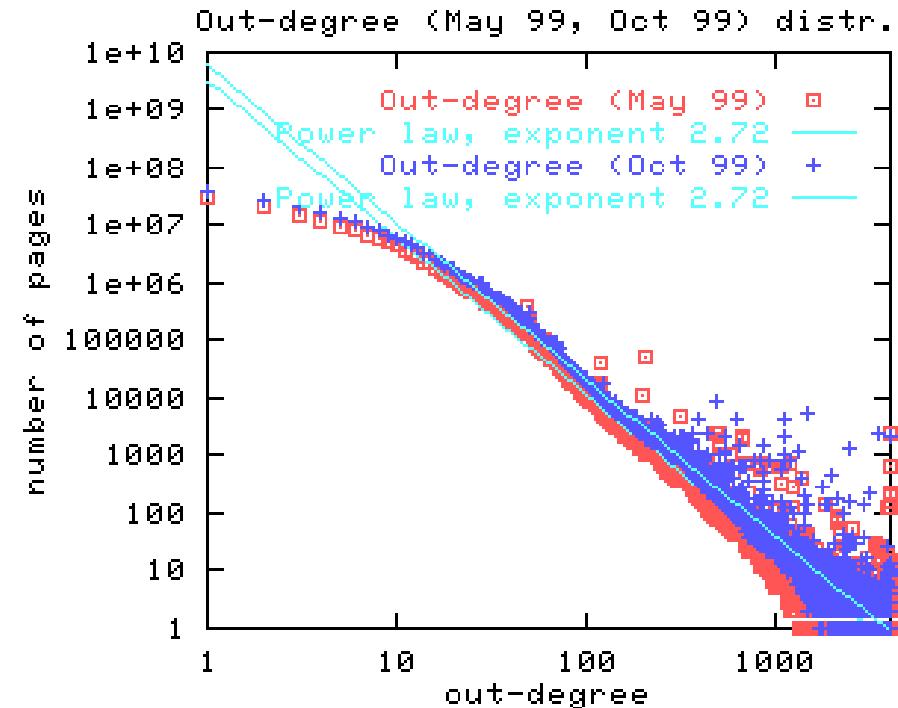
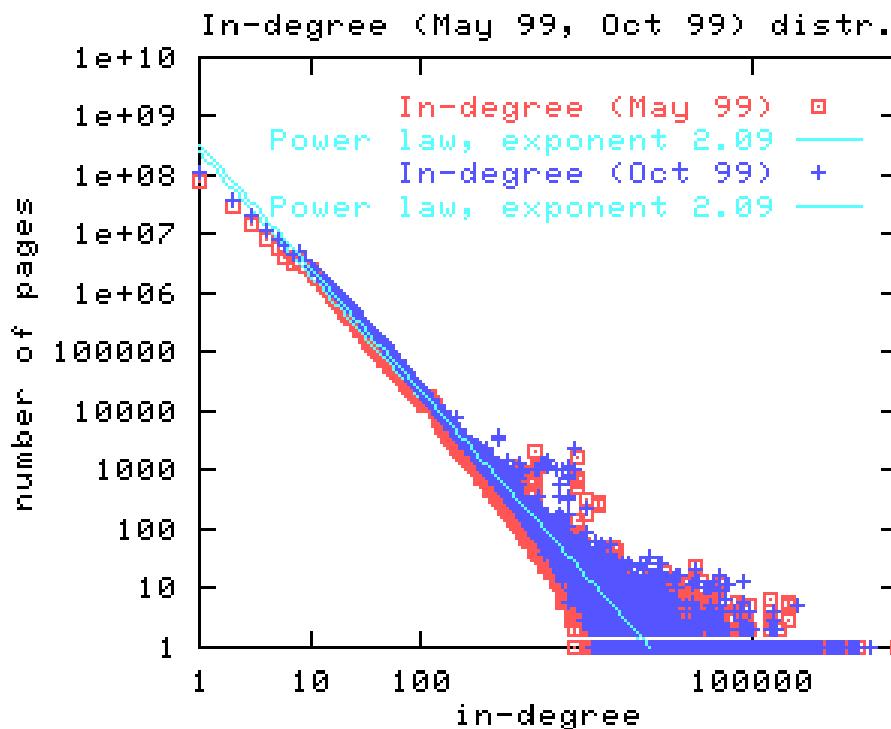
- Internet Autonomous Systems
[Faloutsos, Faloutsos and Faloutsos, 1999]



Internet domain topology

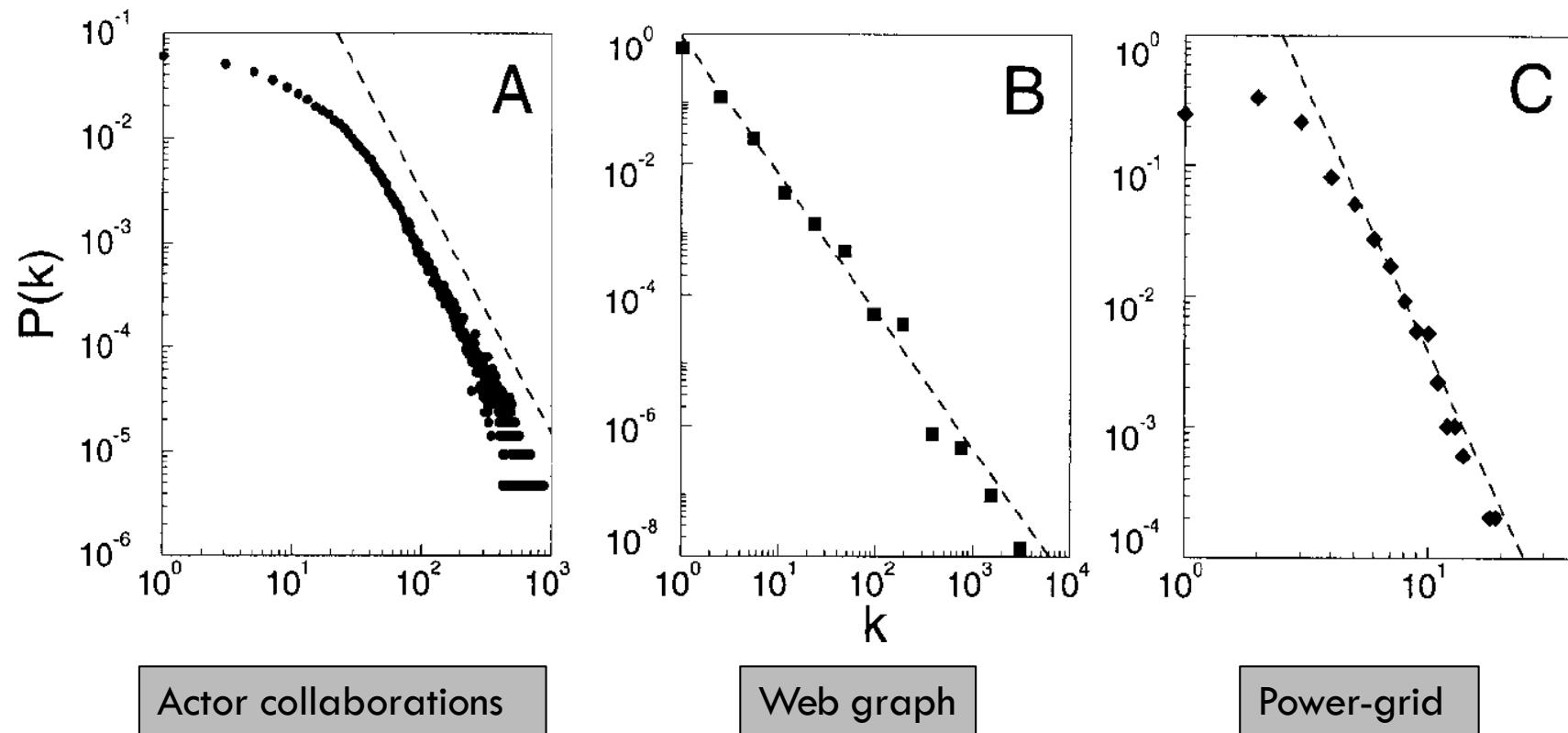
Node Degrees: Web

□ The World Wide Web [Broder et al., 2000]



Node Degrees: Barabasi&Albert

□ Other Networks [Barabasi-Albert, 1999]



Power-Law Degree Exponents

- Power-law degree exponent is typically $2 < \alpha < 3$

- Web graph:
 - $\alpha_{\text{in}} = 2.1, \alpha_{\text{out}} = 2.4$ [Broder et al. 00]

- Autonomous systems:

- $\alpha = 2.4$ [Faloutsos³, 99]

- Actor-collaborations:

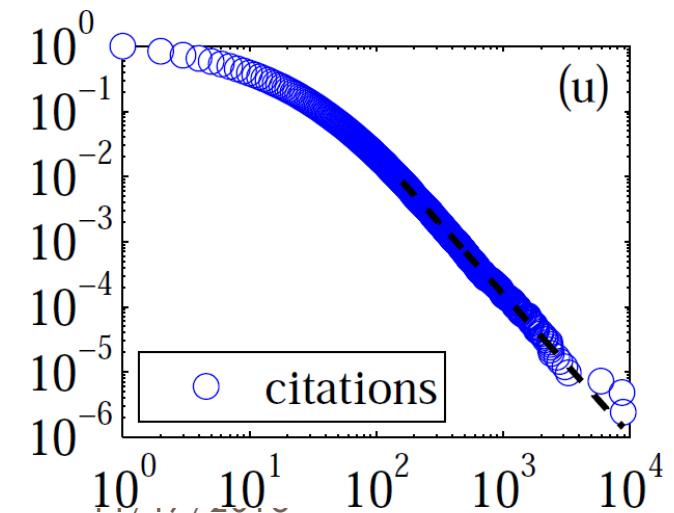
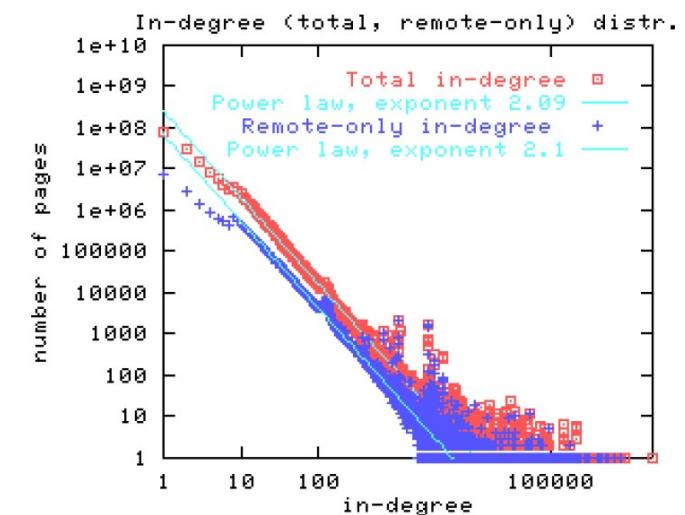
- $\alpha = 2.3$ [Barabasi-Albert 00]

- Citations to papers:

- $\alpha \approx 3$ [Redner 98]

- Online social networks:

- $\alpha \approx 2$ [Leskovec et al. 07]

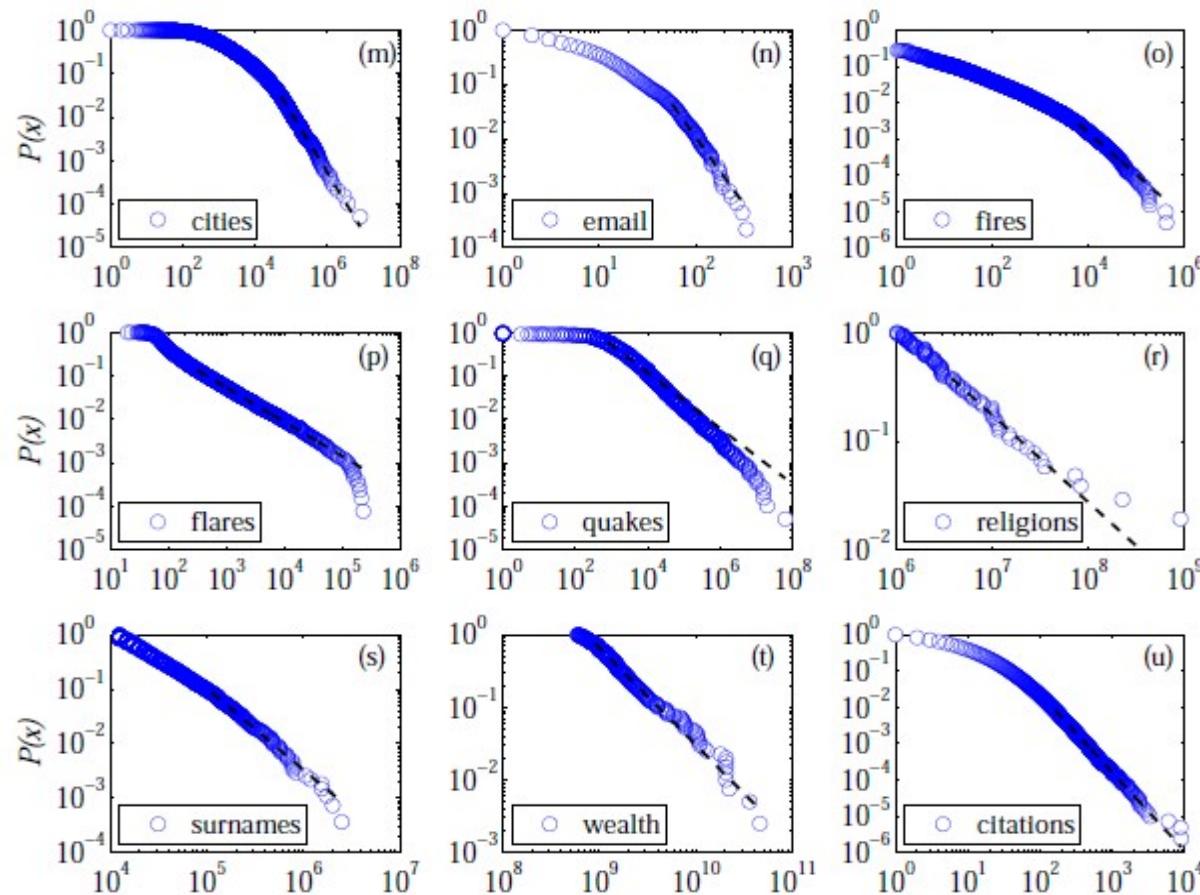


Scale-Free Networks

- **Definition:**
Networks with a power law tail in their degree distribution are called “scale-free networks”

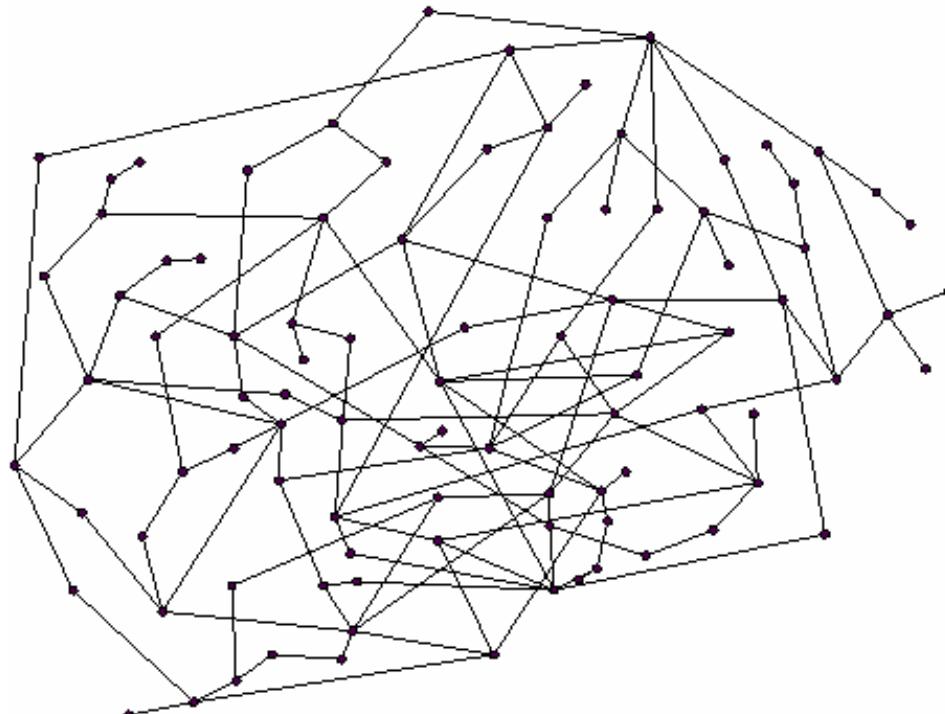
- **Where does the name come from?**
 - **Scale invariance:** There is no characteristic scale
 - **Scale-free function:** $f(ax) = a^\lambda f(x)$
 - Power-law function: $f(ax) = a^\lambda x^\lambda = a^\lambda f(x)$

Power-Laws are Everywhere

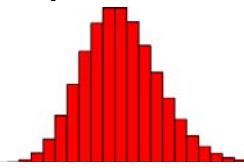


Many other quantities follow heavy-tailed distributions

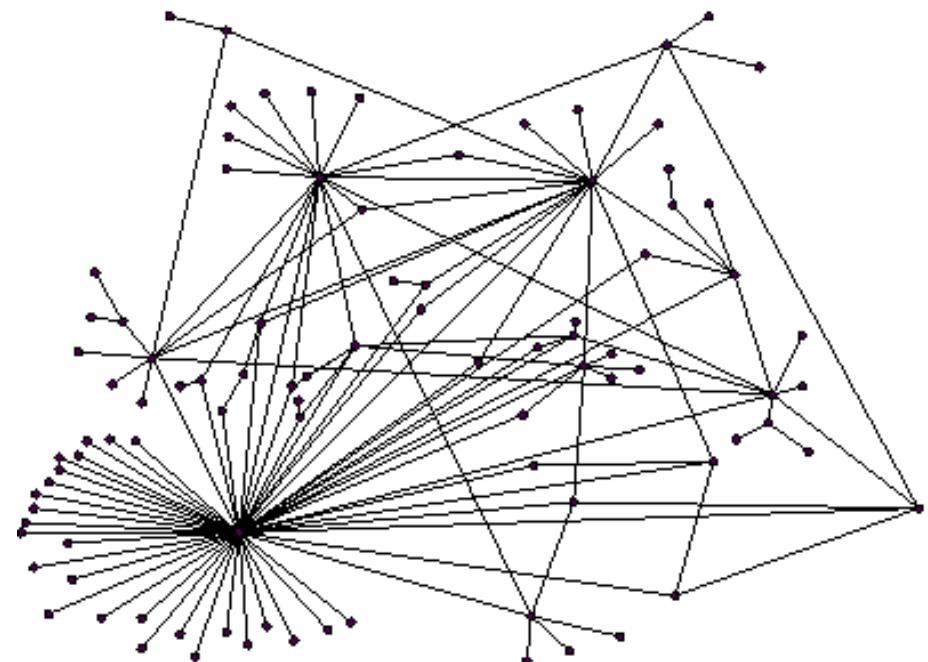
Random vs. Scale-free network



Random network
(Erdos-Renyi random graph)



Degree distribution is Binomial



Scale-free (power-law) network



Degree
distribution is
Power-law

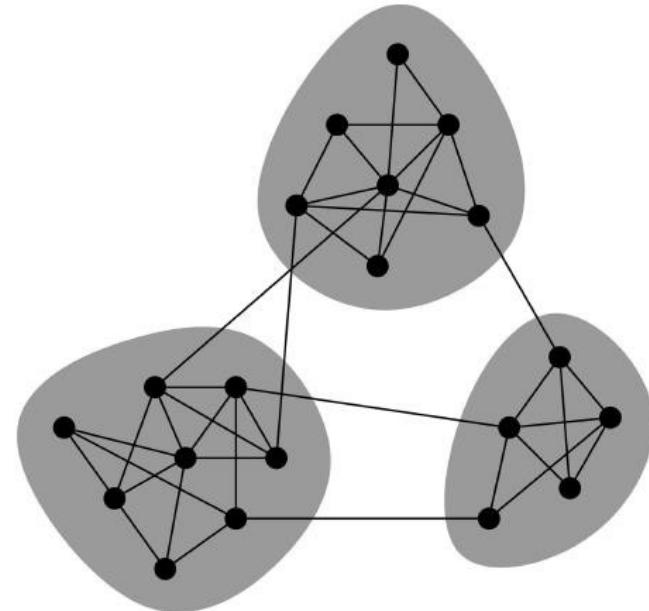
NETWORK COMMUNITIES



Network Communities

- Networks are composed of
tightly connected
sets of nodes

- **Network communities:**
 - Sets of nodes with **lots** of connections **inside** and **few** to
outside (the rest of the network)

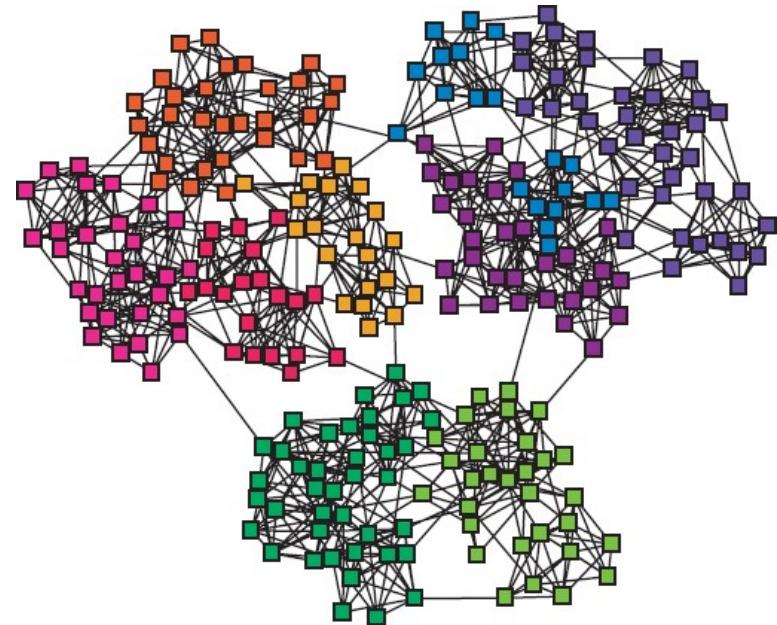


Finding Network Communities

- How to automatically find such densely connected groups of nodes?

- Ideally such automatically detected clusters would then correspond to real groups

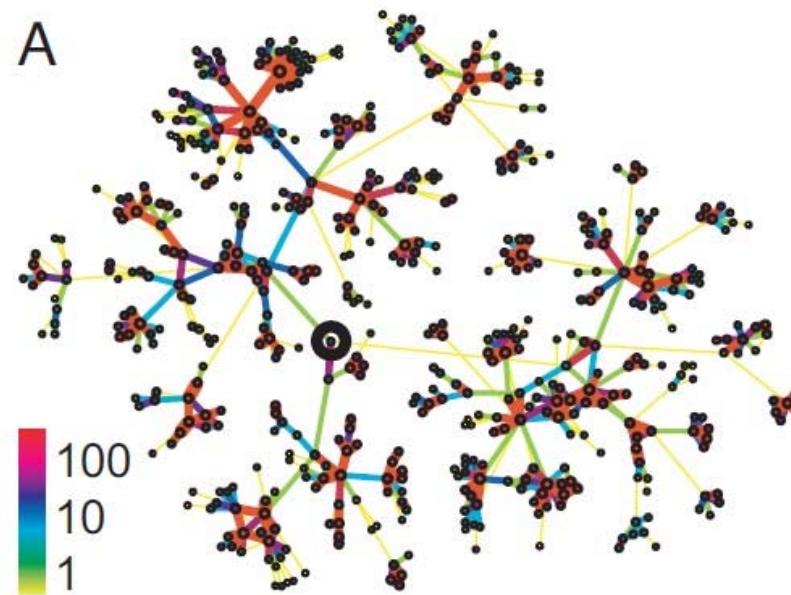
- For example:



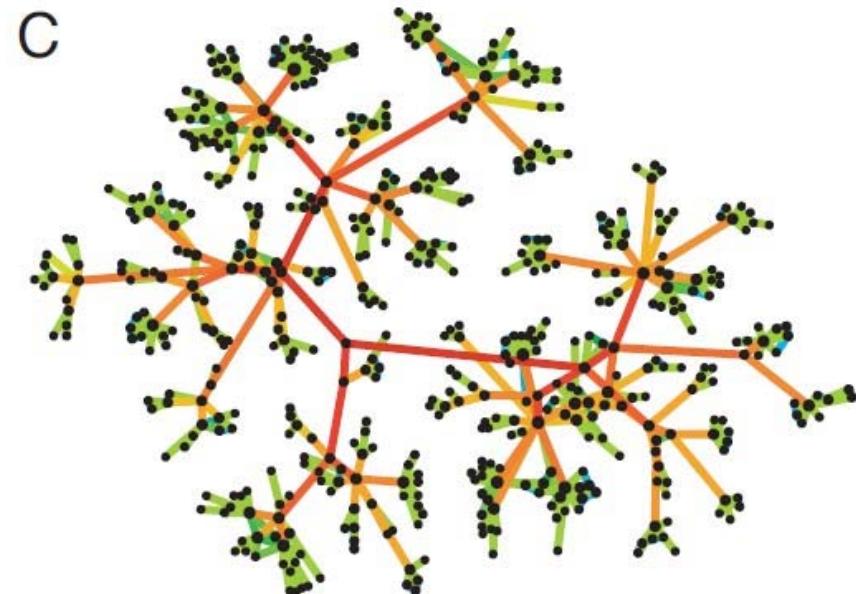
Communities, clusters,
groups, modules

Strength of Weak Ties

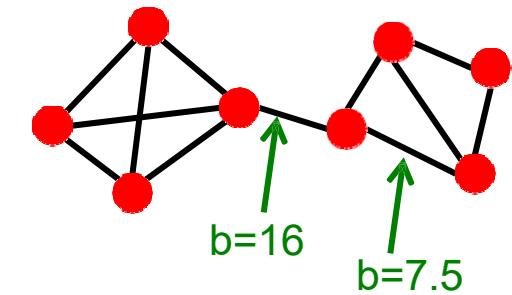
- **Edge betweenness:** Number of shortest paths passing over the edge
- **Intuition:**



Edge strengths (call volume)
in real network



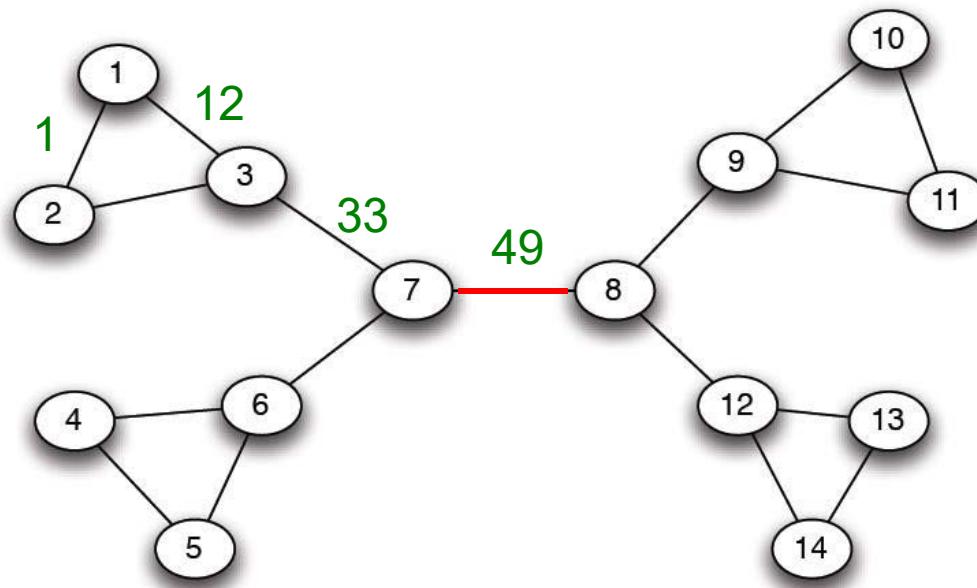
Edge betweenness
in real network



Girvan-Newman Method

- 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

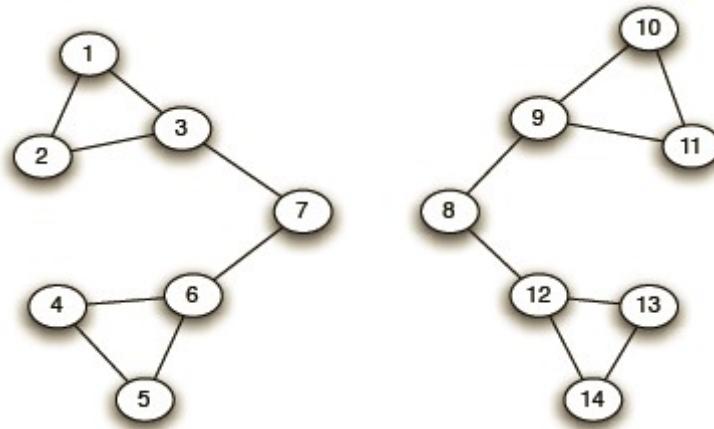
Girvan-Newman: Example



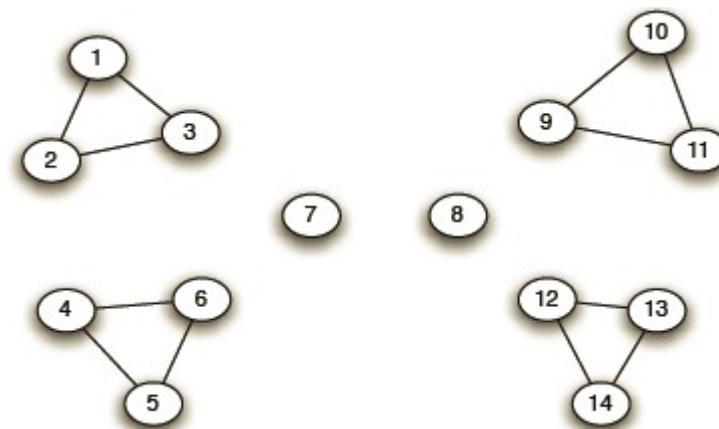
Need to re-compute
betweenness at
every step

Girvan-Newman: Example

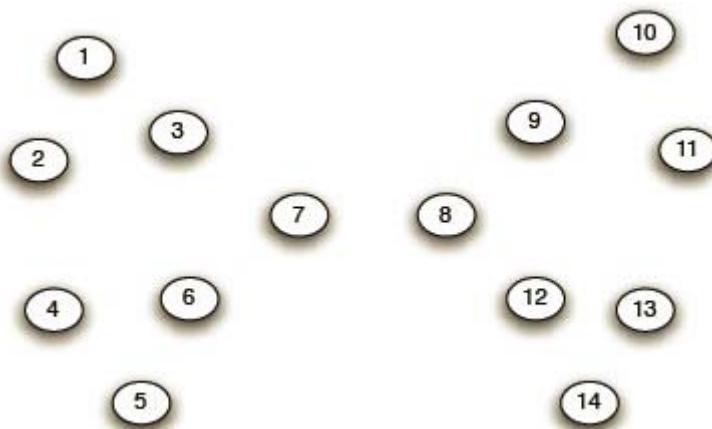
Step 1:



Step 2:



Step 3:



Hierarchical network decomposition:

