

# Assessment

- Final exam (50%) + Assignments (50%)
  - Closed book exam
- Assignments *End of week 8  
13 Oct 2021!*
  - 2 group-based projects: (25% + 25%)
    - Project 1 will be announced in Week 2
    - Project 2 will be announced in the 2<sup>nd</sup> half

A simple example for collaborative filtering.  
Can be formulated as a missing value prediction problem:

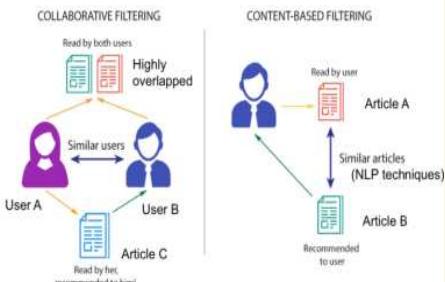
	Movie A	Movie B	Movie C	Movie D
User 1	5	4	1	1
User 2	2	3	2	4
User 3	2	4	?	1
User 4	2	3	1	?

Fill in missing value (User 4, Movie D) in the table

## Topics (Tentative)

- The 1<sup>st</sup> half (Week 1 to Week 6)
  - Introduction
  - Clustering – basic methods
  - Link analysis – PageRank
  - Graph neural network
  - Similarity search
  - Clustering – advanced methods
  - Graph community detection
- The 2<sup>nd</sup> half (Week 7 to Week 12)
  - Association Rule Mining
  - Classification
  - Recommendation Systems
  - Data Processing, Data Cleaning and integration others

## Recommender system



Focus on the similarity between users      Focus on the similarity between items

<https://towardsdatascience.com/brief-on-recommender-systems-b86a1068a4dd>

- Please start to form your group now
  - please edit the online form to create your group.
    - [https://docs.google.com/spreadsheets/d/1YMrw326R1Cjk\\_C9DlkqumhtLcQMhNEQ-DnGeKROs20/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1YMrw326R1Cjk_C9DlkqumhtLcQMhNEQ-DnGeKROs20/edit?usp=sharing)
  - Each group is limited to 4 members.
    - The first person for each group in the form is the coordinator and contact person.
    - You can form a group of less than 4 members.
  - Grouping will be finalized in week 3

## Many other applications



<https://www.geeksforgeeks.org/applications-of-data-mining/>

A simple example for collaborative filtering.

	Movie A	Movie B	Movie C	Movie D
User 1	5	4	1	1
User 2	2	3	2	4
User 3	2	4	?	1
User 4	2	3	1	?

Fill in missing value (User 4, Movie D) in the table

	Movie A	Movie B	Movie C	Movie D
User 1	5	4	1	1
User 2	2	3	2	4
User 3	2	4	?	1
User 4	2	3	1	?

Predict the missing value (User 4, Movie D) in the table

Collaborative filtering:

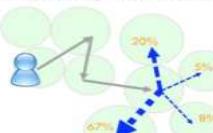
User 4 is similar to user 2, so we can predict 4 for (User 4, Movie D)

Un-personalized prediction (prior):

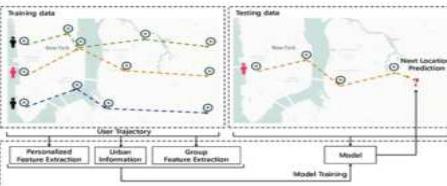
Use the average score as the prediction:  $(1+4+1)/3 = 2$

## Application: information retrieval

- Urban computing, smart city
  - Next location prediction/ future position prediction
    - Spatial-temporal trajectory data (e.g., GPS data + time)
    - Predict traffic congestion, improve traffic management



Source: <http://www.iitk-cnr.iitk.ac.in/research/mobile-y-data-mining-science-cities>



Source: PGNet: Personalized and Group Preferences Guided Network for Next Place Prediction

## Application

- Urban computing, smart city
  - Urban planning
    - Choose the address for a hospital/bus stop?
- Social computing
  - Analyse human behavior based on social data
    - Examples of social data: social media, blogs, social networks
    - Social media platforms: Twitter, Facebook, etc.
  - Social Media Analysis
    - Understand user behavior and activities
  - Social Network Analysis
    - Community detection

# Data basics

## Outline

- Types of datasets
- Data objects and attributes
- Distance and similarity
- Data normalization

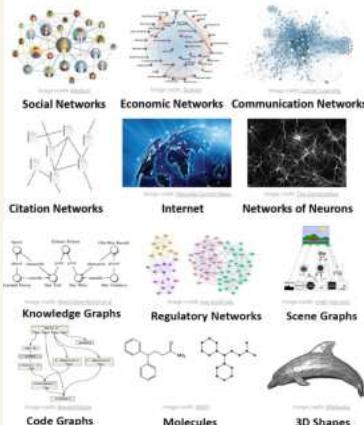
## Types of Datasets

### 1. Record Data

records in a relational database



### 2. Graphs and networks



### 3. Multimedia data



### 4. Text data

- Twitter/Facebook posts
- News
- Wikipedia texts
- Shopping item comments
- Books
- Transcripts
- Emails
- Documents

## Data objects and attributes

Example:  
1 table

- Data objects are also called samples, examples, instances, data points, records, tuples, ...
- Data attributes are also called dimensions, features, variables, channels, ...
- Data sets are made up of data objects/samples
- Data objects are described by attributes/features

Car:

Car_ID	Model	Year	Value	Pers_ID
100	BMW	1975	1000000	0
102	Rolls Royce	1965	3300000	0
103	Porsche	1993	500	1
104	Ferrari	2005	1500000	4
105	Renault	1998	20000	3
106	Renault	2001	70000	3
107	Smart	1999	2000	2

Each row represents a data sample;  
Each column represents a data attribute.

## Attribute Types

### Numeric

- real numbers (continuous values)
  - Prices, \$500, \$200; Image pixel intensity: [0 255]
  - temperature, height, or weight

### Nominal

- (Categorical) categories, states, or "names of things" (discrete values)

- hair\_color = [black, blond, brown, grey, red, white]

- marital status, occupation, ID numbers, zip codes

! nominal  
cannot compare

### Binary

- Nominal attribute with only 2 states (0 and 1)
- [true, false],

- [1, 0] indicates one item exists or not

### Ordinal

- Values have a meaningful order (ranking) but magnitude between successive values is not known

- Size = [small, medium, large], grades=[A, B, C, D]

### Convert raw data into numeric values

- For many data mining (machine learning) tasks, e.g., clustering, classification, regression,

- Nominal->numeric: use one-hot encoding

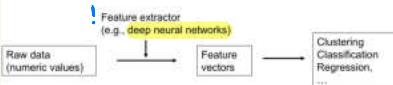
- Ordinal->numeric: use numbers to indicate ranking {1,2,3,...}

- Binary->numeric: convert to {0, 1}

Example: convert nominal values to numeric values using one-hot encoding

id	color	one-hot encoding	id	color	one-hot encoding
1	red	[1, 0, 0, 0]	5	red	[1, 0, 0, 0]
2	blue	[0, 1, 0, 0]	6	blue	[0, 1, 0, 0]
3	green	[0, 0, 1, 0]	7	green	[0, 0, 1, 0]
4	blue	[0, 1, 0, 0]	8	blue	[0, 1, 0, 0]

A typical pipeline for applying data mining techniques:



Feature vectors: one data sample is represented by one feature vector.  
e.g.,  $\mathbf{x} = [x_1, x_2, x_3, \dots]$

### Vector norm

- also called vector magnitude, the length of the vector

#### 1. Lp-norm (general):

Let  $p \geq 1$  be a real number. The  $p$ -norm (also called  $\ell_p$ -norm) of vector  $\mathbf{x} = (x_1, \dots, x_n)$  is

$$\|\mathbf{x}\|_p := \left( \sum_{i=1}^n |x_i|^p \right)^{1/p}.$$

#### 2. L1-norm:

$$\|\mathbf{x}\|_1 := \sum_{i=1}^n |x_i|.$$

#### 3. L2-norm:

$$\|\mathbf{x}\|_2 := \sqrt{x_1^2 + \dots + x_n^2}.$$

Also called Euclidean norm,  
vector length

## Distance

Given two vectors:  $\mathbf{x}_i$  and  $\mathbf{x}_j$  (they have  $d$  dimensions)

1. Calculate the vector difference (residual vector):  $\mathbf{r} = \mathbf{x}_i - \mathbf{x}_j$

2. apply vector norm on the vector difference:

$$d_p(i, j) = \|\mathbf{r}\|_p = \|\mathbf{x}_i - \mathbf{x}_j\|_p$$

### $p = 1$ : (L<sub>1</sub> norm) Manhattan distance (L<sub>1</sub> distance)

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{id} - x_{jd}|$$

### $p = 2$ : (L<sub>2</sub> norm) Euclidean distance (L<sub>2</sub> distance)

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{id} - x_{jd}|^2}$$

Metric	Formula	Interpretation
Euclidean distance	$d = \sqrt{\sum_i (x_i - y_i)^2}$	
Manhattan distance	$d = \sum_i  x_i - y_i $	
Minkowski distance (defined by vector norm):	$d(i, j) = \sqrt[p]{ x_{i1} - x_{j1} ^p +  x_{i2} - x_{j2} ^p + \dots +  x_{id} - x_{jd} ^p}$	

where  $i = (x_{i1}, x_{i2}, \dots, x_{id})$  and  $j = (x_{j1}, x_{j2}, \dots, x_{jd})$  are two  $d$ -dimensional data objects, and  $p$  is the order (defined based on L<sub>p</sub> norm)

$p = \text{parameter } (> 0)$

It has the properties:

- $d(i, j) > 0$  if  $i \neq j$ , and  $d(i, i) = 0$  (Positivity)

- $d(i, j) = d(j, i)$  (Symmetry)

- $d(i, j) \leq d(i, k) + d(k, j)$  (Triangle Inequality)

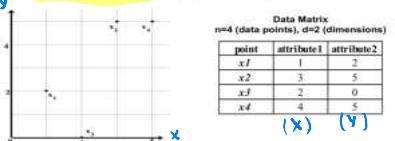
A distance that satisfies these properties is a metric

### Data matrix

- Describe a data set (data samples)

- E.g., a data matrix of  $n$  data points with  $d$  dimensions

- Each row indicates a feature vector



## Distance example

### Manhattan distance (L<sub>1</sub>)

1	2	3	4
x1	1	2	3
x2	3	0	1
x3	2	0	0
x4	0	1	2

### Euclidean distance (L<sub>2</sub>)

1	2	3	4
x1	1	2	3
x2	3	0	1
x3	2	0	0
x4	0	1	2

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (0-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (2-1)^2} = \sqrt{1+1+9+1} = \sqrt{12} = 3.46$$

$$\sqrt{(1-0)^2 + (2-1)^2 + (3-0)^2 + (1-2)^2} = \sqrt{1+1$$

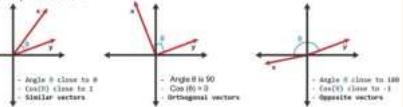
# Similarity

## Cosine similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Cosine distance = 1 - cosine similarity

Geometry illustration:



## Similarity example

- $D1 = [1, 1, 1, 1, 1, 0, 0]$
- $D2 = [0, 0, 1, 1, 0, 1, 1]$

First, we calculate the dot product of the vectors:

$$D1 \cdot D2 = 1 \times 0 + 1 \times 0 + 1 \times 1 + 1 \times 1 + 1 \times 0 + 0 \times 1 + 0 \times 1 = 2$$

Second, we calculate the magnitude (L2 norm) of the vectors:

$$\|D1\| = \sqrt{1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2} = \sqrt{5}$$

$$\|D2\| = \sqrt{0^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 1^2} = \sqrt{7}$$

$$\text{Finally: } \text{similarity}(D1, D2) = \frac{D1 \cdot D2}{\|D1\| \|D2\|} = \frac{2}{\sqrt{5} \sqrt{7}} = \frac{2}{\sqrt{35}} = 0.44721$$

We can further calculate the angle between the vectors:

$$\cos(\theta) = 0.44721$$

$$\theta = \arccos(0.44721) = 63.435$$

## Data normalization

### Data normalization

- The goal of normalization is to transform attributes/features to be on a similar scale.
- Algorithms may bias to the features which have a larger magnitude.
- E.g., L2-distance will be dominated by large attributes

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{id} - x_{jd}|^2}$$

Examples:

$$\begin{aligned} x1 &= [100, 0.1] \\ x2 &= [120, 0.01] \\ x3 &= [120, 0.1] \end{aligned} \longrightarrow d(x1, x2) = d(x1, x3)$$

#### 1. Max-Min Normalization

= rescale to a value in [0, 1]

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

#### 2. Z-score normalization

- Also called standardization
- rescale to ensure the mean and the standard deviation to be 0 and 1
- More robust to outlier

$$x_{stand} = \frac{x - \text{mean}(x)}{\text{standard deviation}(x)}$$

<https://www.kdnuggets.com/2020/01/data-transformation-standardization-normalization.html>  
<https://www.neurdatalab.com/std-deviation/>

## Example

### Input dataset

user	Age	Salary	Age Mean	15.4
			Age Std	7.829432
1	40	100000	Salary Mean	68800
2	32	80000	Salary Std	22718.85
3	21	43000	Age min	21
4	24	51000	Age max	40
5	35	70000	Salary min	43000
			Salary max	100000

### Z-score normalization

user	Age	Salary	user	Age	Salary
1	1.2261426	1.367291	1	1	1
2	0.2043571	0.490822	2	0.5789474	0.649123
3	-1.200598	-1.13064	3	0	0
4	-0.817428	-0.78006	4	0.1578947	0.140351
5	0.5875267	0.052588	5	0.7368421	0.473684

## Clustering

- K-means
- Hierarchical clustering

## Outline

### Basic methods for clustering:

#### K-means

- Algorithm
- Extension: K-means++

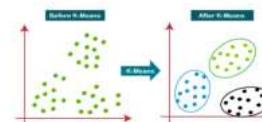
- K-means with clustroid (non-examinable)
- Optimization problem (non-examinable)
- (non-examinable content may be useful for your coursework projects)

#### Hierarchical Clustering

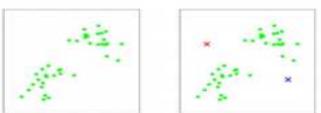
### Application:

- Image clustering  
↳ Cluster image pixels to generate superpixels

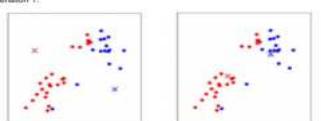
## K-means



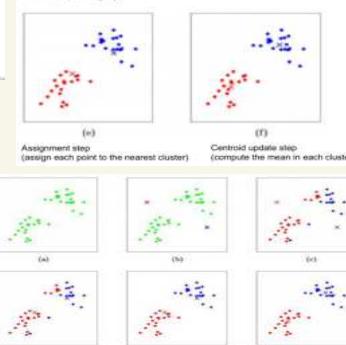
### A simple example



Iteration 1:



Iteration 2: (converged)



(a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations of k-means.

<https://stanford.edu/~cipech/cs229/handouts/kmeans.html>

# K-means

## Algorithm 1 k-means algorithm

- 1: Specify the number  $k$  of clusters to assign.
- 2: Randomly initialize  $k$  centroids.
- 3: repeat
  - a: expectation: Assign each point to its closest centroid.
  - b: maximization: Compute the new centroid (mean) of each cluster.
  - c: until The centroid positions do not change.

Assignment step  
Centroid update step

<https://realpython.com/k-means-clustering-python/>

## Example

- Given the following data vectors and initial centroids, perform K-means for 1 iteration.

Data points:

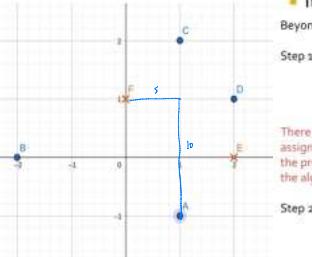
A	[ $-1, -1$ ]
B	[ $-2, 0$ ]
C	[ $1, 2$ ]
D	[ $2, 1$ ]

Initial centroids  
(2 clusters):

E (Cluster 1)	[ $2, 0$ ]
F (Cluster 2)	[ $0, 1$ ]

Data points:

A	[ $1, -1$ ]
B	[ $-2, 0$ ]
C	[ $1, 2$ ]
D	[ $2, 1$ ]



Initial centroids  
(2 clusters):

E (Cluster 1)	[ $2, 0$ ]
F (Cluster 2)	[ $0, 1$ ]

## Iteration 1

### Step 1: Cluster assignment

- 1) calculate squared L2 distance in Table 1

We can use squared L2 distance

as it will give the same comparison results as L2 distance.

E.g., SquaredL2(A, E) = ( $1-2$ ) $^2 + (-1-0)^2 = 1+1=2$ ;

SquaredL2(A, F) = ( $1-0$ ) $^2 + (-1-1)^2 = 1+4=5$ ;

Table 1 Squared L2 distance:

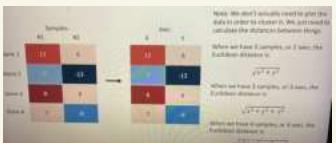
	E (2, 0) Cluster1	F (0, 1) Cluster2
A ( $1, -1$ )	2	5
B ( $-2, 0$ )	16	5
C ( $1, 2$ )	5	2
D ( $2, 1$ )	1	4

Squared L2 distance:

$$\|x - y\| = \sum_{i=1}^d (x_i - y_i)^2$$

L2 distance:

$$d_{L2}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$



## 2) Assign each point to its nearest centroid

Cluster 1: (A, D) Cluster 2 : (B, C)

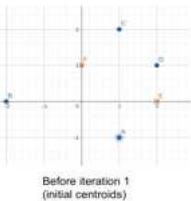
Step 2: Centroid update by compute the mean of each cluster

Cluster1 E=[1.5, 0], F=[-0.5, 1]

$$E_1 = \frac{(1+2)}{2} / 2 = 1.5, E_2 = \frac{(-1+1)}{2} / 2 = 0 \rightarrow E = [1.5, 0] \\ F_1 = \frac{(-2+1)}{2} / 2 = -0.5, F_2 = \frac{(0+2)}{2} / 2 = 1, \rightarrow F = [-0.5, 1]$$

Table 2 Squared L2 distance:

	E (2, 0) Cluster1	F (0, 1) Cluster2
A ( $1, -1$ )	2	5
B ( $-2, 0$ )	16	5
C ( $1, 2$ )	5	2
D ( $2, 1$ )	1	4



## Iteration 2

Beyond the question, we perform the 2<sup>nd</sup> iteration:

### Step 1: Cluster assignment

- 1) calculate squared L2 distance in Table 2

- 2) assign each point to its nearest centroid

Cluster C1: (A, D) Cluster C2: (B, C)

There is no changes of cluster assignments compared to the previous iteration, so the algorithm is converged.

### Step 2: Centroid update

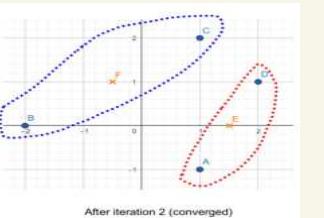
Cluster 1: (A, D)

Cluster 2: (B, C)

Converged

Table 3 Squared L2 distance

	E (1.5, 0) C1	F (-0.5, 1) C2
A ( $1, -1$ )	1.25	6.25
B ( $-2, 0$ )	12.25	3.25
C ( $1, 2$ )	4.25	3.25
D ( $2, 1$ )	1.25	6.25



## How to determine the K parameter?

- \* K is the number of clusters

How to select k?

- \* Try different k, looking at the change in the average distance to centroid as k increases

- \* Average distance to centroid falls rapidly at the beginning, then it changes slowly after a certain value of k

The elbow method



- Select a small k that can produce small average distance to centroid
  1. We prefer a low average distance to centroid (Centroid shift is good)
  2. We prefer a small k (to observe small groups)

The elbow method



Average distance to centroid: E

$$E = \frac{1}{N} \left( \sum_{p \in C_1} d(p_1, c_1) + \sum_{p \in C_2} d(p_2, c_2) + \dots + \sum_{p \in C_k} d(p_k, c_k) \right)$$

d: distance, e.g., L2 distance or Squared L2 distance

p: a data point

c: a cluster centroid, C: a cluster

N: the total number of data points

## Example



Table: Squared L2 distance

	E( $1.5, 0$ ) C1	F( $-0.5, 1$ ) C2
A ( $1, -1$ )	1.25	6.25
B ( $-2, 0$ )	12.25	3.25
C ( $1, 2$ )	4.25	3.25
D ( $2, 1$ )	1.25	6.25

k=2, use Squared L2 distance for d here:

$$Y_1 = d(A, C1) + d(D, C1) = 1.25 + 1.25 = 2.5$$

$$Y_2 = d(B, C2) + d(C, C2) = 12.25 + 3.25 = 15.5$$

Average distance to centroid:  
 $E = (y_1+y_2)/4 = 9/4 = 2.25$

### K-means

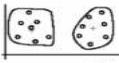
- \* distance functions
  - Euclidean distance (L2 distance)
  - most commonly used
  - L1 distance
  - Cosine distance
  - Other distance functions

## K-means

- \* Pros
  - Simple and fast, Easy to implement

### Cons

- \* Need to choose K
- \* Sensitive to outliers
- \* Cannot work well on data with non-spherical shape
- \* Sensitive to the initialization of the centroids



(A): Ideal clusters



(B): Non-spherical clusters

Limitation of K-means: cannot handle arbitrary shapes.

K-means is suitable for spherical or elliptical data.

K-means algorithm is sensitive to the initialization of the centroids

### K-means++

- \* Better initialization
- \* K-means++ is the standard K-means algorithm coupled with a smarter initialization of the centroids.

Ideas: select initial centroids that are far away from each other.

1. Sequentially initialize the centroids.

2. When selecting one centroid, we aim to select a data point that is far away from existing centroids.

### K-means++

let  $D(x)$  denote the shortest distance from a data point

$x$  to the existing centroids ( $C_1, C_2, \dots, C_k$ )

$$D(x) = \min_{i \in \{1, 2, \dots, k\}} d(x, c_i)$$

K-means++ (simplified version)

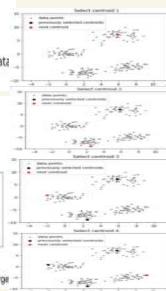
- Step 1: choose the data point with the highest  $D(x)$  as the new centroid ( $C_{k+1}$ ).

The selected data point will be far away from all existing centroids.

- Step 2: Repeat step 1 until we have k centroids.

1. If the data point  $x$  is close to any existing centroids,  $D(x)$  will be small

2. If the data point  $x$  is far away from any existing centroids,  $D(x)$  will be large



# Segmentation as Clustering

- Let's just use the pixel intensities!

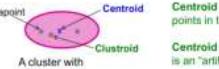
Segmentation: pixel grouping  
Assign pixels to clusters



More examples:  
<https://github.com/topics/slic>

- K-means with clustroid
  - Use clustroid to replace centroid
  - Clustroid: is an existing (data) point that is "closest" to all other points in the cluster.

**Non-examable**



Centroid is the avg. of all (data) points in the cluster.

Centroid is not an existing point. It is an 'artificial' point.

For a point  $p$ , calculate the average distance between  $p$  and all other members in the same cluster. Clustroid is the point that has the smallest average distance within the cluster.

## Optimization problem

**Non-examable**

- The optimization problem for K-means

The squared distance to centroid for one cluster:

$$S(C_i) = \sum_{x_j \in C_i} (c_i - x_j)^2$$

$S(C_i)$  : squared distance to centroid for cluster  $C_i$

$x_j \in C_i$  : a data point in cluster  $C_i$

$c_i$  : the cluster centroid of  $C_i$

Slides from Stanford CS131.

Goal: minimize average squared distance to centroid  
(or minimize within-cluster variance)

Objective function:  

$$c^*, \delta^* = \arg \min_{c, \delta} \frac{1}{N} \sum_{j=1}^N \sum_{i=1}^K \delta_{ij} (c_i - x_j)^2$$

Asterisk \* indicates the solutions to:  
 1. cluster centroids  
 2. and assignments

Whether  $x_j$  is assigned to  $C_i$   
 If  $x_j$  is a member of cluster  $C_i$ ,  $\delta_{ij} = 1$ ;  
 otherwise,  $\delta_{ij} = 0$ .

## K-means clustering

- Initialize ( $t = 0$ ): cluster centers  $c_1, \dots, c_K$
- Compute  $\delta^t$ : assign each point to the closest center  
 $\delta^t$  denotes the set of assignment for each  $x_j$  to cluster  $C_i$  at iteration  $t$   

$$\delta^t = \arg \min_{\delta^t} \frac{1}{N} \sum_{j=1}^N \sum_{i=1}^K \delta_{ij}^t (c_i^{t-1} - x_j)^2$$
 Freeze  $c$ , solve for  $\delta$
- Computer  $c^t$ : update cluster centers as the mean of the points  
 Cluster update step  

$$c^t = \arg \min_{c^t} \frac{1}{N} \sum_{j=1}^N \sum_{i=1}^K \delta_{ij}^t (c_i - x_j)^2$$
 Freeze  $\delta$ , solve for  $c$
- Update  $t = t + 1$ , Repeat Step 2-3 till stopped

- Hard to solve the optimization problem
- Hard to obtain global optimal solutions

- The iterative algorithm is also referred to as Lloyd's algorithm
  - Empirically works well, usually converges in a few iterations
- Further reading:
  - Mini-batch K-means for large-scale data
    - Handle a small number of data points at a time, using stochastic gradient descent to gradually update the cluster centre.
  - Further reading:
    - Online tool: [sklearn.cluster.MiniBatchKMeans](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html)

# Hierarchical clustering

## Hierarchical

### Agglomerative (bottom up)

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



### Divisive (top down)

- Start with one cluster and recursively split it

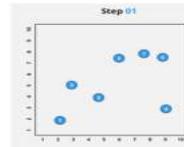
## Agglomerative clustering (or bottom-up hierarchical clustering)

### Simple algorithm

- Initialization:
  - Every point is its own cluster
- Repeat:
  - Find "most similar" pair of clusters
  - Merge into a parent cluster
- Until:
  - The desired number of clusters has been reached
  - There is only one cluster

First, make each data point a cluster, which forms  $N$  clusters.

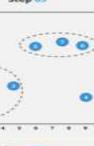
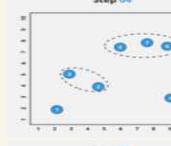
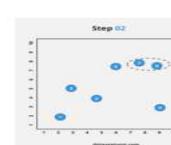
### Example 1:



Take the next two nearest clusters and make them one cluster



Again, take the two nearest clusters and make them one cluster



## Example 2

### What is missing here?

### How to define the similarity of two clusters?

#### Single Linkage

$D(c_i, c_j) = \min D(x_i, x_j)$   
Minimum distance or distance between closest elements in clusters



#### Complete Linkage

$D(c_i, c_j) = \max D(x_i, x_j)$   
Maximum distance between elements in clusters



#### Average Linkage

$D(c_i, c_j) = \frac{1}{|c_i| |c_j|} \sum_{x_i \in c_i} \sum_{x_j \in c_j} D(x_i, x_j)$   
Average of the distances of all pairs



#### Centroid Method

Combining clusters with minimum distance between the centroids of the two clusters



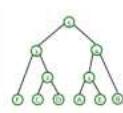
define cluster similarity

<https://dataaspirant.com/hierarchical-clustering-algorithm/>

- We will discuss the following calculation example in the tutorial class:

## Data points

A	[1, -1]
B	[2, 0]
C	[1, 2]
D	[2, 1]
E	[1, -1]



### When do we stop merging clusters?

- When some number  $k$  of clusters are found
- Keep merging until there is only 1 cluster

# Hierarchical Clustering

### Pros

- No need to set  $K$ : the number of clusters
- No assumption on the shape of the cluster
- Simple

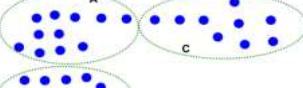
### Cons

- The algorithm can never undo the grouping. The data points may be incorrectly grouped at an earlier stage.

- Different distance metrics may produce very different results.
  - Single Linkage methods are sensitive to noise and outliers.
  - Complete linkage methods tend to break large clusters.

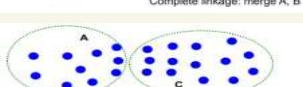
<https://dataaspirant.com/hierarchical-clustering-algorithm/>

## Single linkage: merge A, C



Single linkage: merge A, C

## Complete linkage: merge A, B



Single linkage: merge A, C

Complete linkage: merge A, B

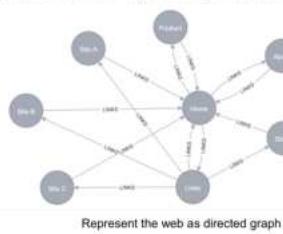
# Link Analysis: PageRank

## Outline

- PageRank Algorithm
  - Page rank
  - Page rank extensions
    - Potential problems
    - Teleportation
    - Page rank – Google Matrix

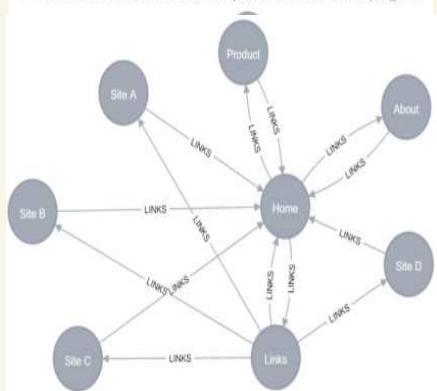
Many slides are from:  
<http://web.stanford.edu/class/cs224w/slides/04-pagerank.pdf>

- Web as a directed graph:
- Nodes = web pages
- Edges with directions = hyperlinks (in-links and out-links)



Represent the web as directed graph

- Link analysis is to discover useful information from relationships/connections between data objects. (mining the graph data)
- Use graphs to describe connections.
  - Vertices represent objects/data points
  - Links describe relationships among objects.
- PageRank is an algorithm to rank web pages.
  - An algorithm for link analysis and web search.
  - proposed by Sergey Brin and Larry Page,
  - students at Stanford University and the founders of Google.
- All web pages are not equally "important"
  - <https://guosheng.github.io/>
  - <https://www.ntu.edu.sg/> (More important)
- How to measure the importance of web pages?



- How to measure the importance of web pages?
- We can look at the links.
- Q1: In-coming links vs out-going links?
- Q2: Are all links equally important?

## Idea: Links as votes

- Page is more important if it has more links
  - In-coming links vs Out-going links
    - In-coming links are more important
- Examples:
  - www.stanford.edu has 23,400 in-links
  - thispersondoesnotexist.com has 1 in-link
- Are all in-links equal?
  - In-links from important pages count more

## Summary:

- Two key factors affecting the rank score:
  - 1. The number of in-links.
  - 2. The source nodes of the in-links

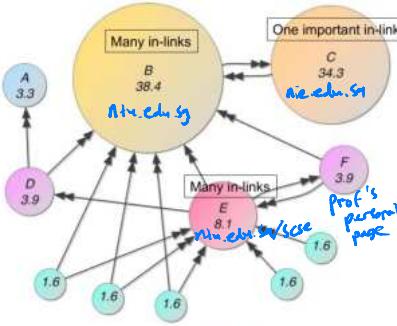
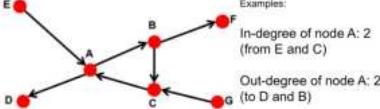


Image credit: Wikipedia

- In-links of a node (or backlinks)
  - the links pointing in (from other nodes), incoming links
  - in-degree: the number of in-links
- Out-links of a node
  - the links pointing out (to other nodes), outgoing links
  - out-degree: the number of out-links

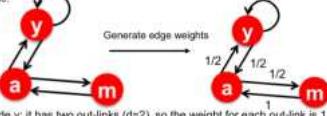


Examples:

## PageRank

- Define the edge (link) weights as  $w_{ij} = \frac{1}{d_i}$   
 $d_i$ : out-degree of the source node  $i$  (the number of out-links)

Example:



$$\text{Node } y: \text{it has two out-links } (d=2), \text{ so the weight for each out-link is } \frac{1}{2}.$$

$$\text{Node } a: \text{it has two out-links } (d=2), \text{ so the weight for each out-link is } \frac{1}{2}.$$

$$\text{Node } m: \text{it has one out-link } (d=1), \text{ so the weight for each out-link is } 1/1.$$

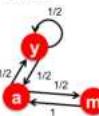
$$\text{Node } a: \text{it has two out-links } (d=2), \text{ so the weight for each out-link is } 1/2.$$

$$\text{Node } m: \text{it has one out-link } (d=1), \text{ so the weight for each out-link is } 1/1.$$

- Define the edge (link) weights as  $w_{ij} = \frac{1}{d_i}$   
 $d_i$ : out-degree of the source node  $i$  (the number of out-links)

use the source node of the link to define the weight.

Example:



The edge weights indicate the "votes" from the source node to the target node.

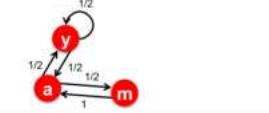
- Define the edge weights as  $w_{ij} = \frac{1}{d_i}$

All the out-links of one node share the same weight

Node  $y$ : it has two out-links ( $d=2$ ), so the weight for each out-link is  $1/2$

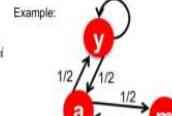
Node  $a$ : it has two out-links ( $d=2$ ), so the weight for each out-link is  $1/2$

Node  $m$ : it has one out-link ( $d=1$ ), so the weight for each out-link is  $1/1$



## PageRank: assign a rank score to each node

For a node  $j$ , we define the rank score  $r_j$  as  
 (flow equations):



$$r_y = r_y/2 + r_a/2$$

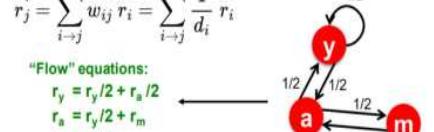
One node rank score = weighted sum of the in-linked node scores;

We also require:  
 (The sum of all node scores equals 1):

$$\sum_i r_i = 1$$

Flow equation:  
 flow-out value = flow-in value  
 $\downarrow$   
 $r_j = \sum_{i \rightarrow j} w_{ij} r_i = \sum_{i \rightarrow j} \frac{1}{d_i} r_i$

One node score = weighted sum of the in-linked node scores;



1.  $y$  has two in-links:  $y \rightarrow y$  and  $a \rightarrow y$ ; two in-linked nodes:  $y$  and  $a$ .
2.  $a$  has two in-links:  $y \rightarrow a$  and  $m \rightarrow a$ ; two in-linked nodes:  $y$  and  $m$ .
3.  $m$  has one in-link:  $a \rightarrow m$ ; one in-linked node:  $a$ .

How to solve for the rank scores?

Solution 1: directly solve the linear equations

Solution 2: power iteration method

### Solution 1:

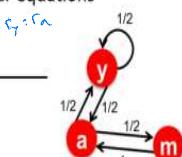
Directly solve the linear equations

"Flow" equations:

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$



Adding this equation:

$$\sum_i r_i = 1$$

We can solve these 4 linear equations to get the solutions:  
 $r_y = 0.4; r_a = 0.4; r_m = 0.2$

(The sum of all rank scores equals 1)

Example using elimination methods to solve linear systems:

EQ1:  $y = y/2 + a/2$

EQ2:  $a = y/2 + m$

EQ3:  $m = a/2$

EQ4:  $a + y + m = 1$

$$\begin{aligned} r_y + r_a + r_m &= 1 \\ r_y + \frac{r_a}{2} + r_m &= 1 \\ r_y + \frac{r_a}{2} + \frac{r_a}{2} &= 1 \\ r_y + r_a &= 1 \\ r_y + r_a + \frac{r_a}{2} &= 1 \\ r_y + \frac{3r_a}{2} &= 1 \\ r_y &= \frac{1}{2} - \frac{3r_a}{2} \end{aligned}$$

Subtract  
from  
y = 1/2 - 3r\_a/2

With EQ2, EQ3  $\rightarrow a = y/2 + a/2$  (eliminate y)

$\rightarrow$  EQ5:  $a = y$

With EQ4, EQ3, EQ5  $\rightarrow a + a + a/2 = 1$  (eliminate y, m)

$\rightarrow a = 2/5 = 0.4$

$\rightarrow y = 0.4, m > 0.2$

### Solution 2:

power iteration method

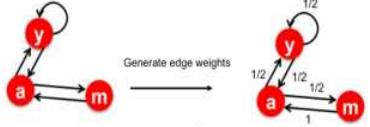
Construct transition matrix: M

- Matrix size:  $n \times n$  ( $n$ : the number of nodes)
- M is constructed by the edge weights.

If there is a link from i to j ( $i \rightarrow j$ ),

$$M_{j,i} = \frac{1}{d_i}$$

$d_i$  is the out-degree of node i



If there is a link from i to j ( $i \rightarrow j$ ),  $M_{j,i} = \frac{1}{d_i}$   
 $d_i$  is the out-degree of node i

Construct M column by column or row by row:  
1. column by column:  
Each column in M indicates out-links for one node  
2. row by row:  
Each row in M indicates the in-links for one node

Transition matrix M

Convert flow equations into matrix multiplication

The flow equation of one node ( $j=1, 2, \dots, n$ ):

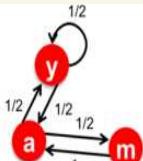
$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

Matrix expression of all flow equations:

$$r = M \cdot r$$

Rank vector  $r$ : a column vector of all rank scores.

$r = [r_1, r_2, r_3, \dots, r_n]^T$ .  $n$ : the total number of nodes.



	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	0
$r_a$	$\frac{1}{2}$	0	1
$r_m$	0	$\frac{1}{2}$	0

Transition matrix M

$$\begin{aligned} r_y &= r_y/2 + r_a/2 \\ r_a &= r_y/2 + r_m \\ r_m &= r_a/2 \end{aligned}$$

Matrix expression

	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	0
$r_a$	$\frac{1}{2}$	0	1
$r_m$	0	$\frac{1}{2}$	0

$r$       M       $r$

### Solution 2: Power iteration method

1. assign initial values to rank scores :  $r = 1/N$ ;

N is the total number of nodes

Uniform distribution

2. repeat the following until converge:

Calculate the page rank:  $r^{(t+1)} = M \cdot r^{(t)}$

Or equivalently update each node using the flow equation:  $r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$

Converge criteria:  $(\sum_i |r_i^{(t+1)} - r_i^{(t)}| < \epsilon)$  Or converge when reach the maximum number of iterations  $E$  :  $E$  is a pre-defined small value, e.g., 0.0001

### Power Iteration:

Set  $r_j \leftarrow 1/N$

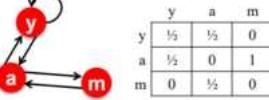
$$1: r'_j \leftarrow \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

2: If  $|r - r'| > \epsilon$ :

$r \leftarrow r'$

3: go to 1

$r'$  is the updated score vector  $r^{(t+1)}$



$$\begin{aligned} r_y &= r_y/2 + r_a/2 \\ r_a &= r_y/2 + r_m \\ r_m &= r_a/2 \end{aligned}$$

Initialization (uniform distribution):  $r_y = 1/3, r_a = 1/3, r_m = 1/3$  ;

Iteration 1: (Flow equation update)

$$r'_y = r_y/2 + r_a/2 = 1/3 * 1/2 + 1/3 * 1/2 = 1/3;$$

$$r'_a = r_y/2 + r_m = 1/3 * 1/2 + 1/3 = 3/6;$$

$$r'_m = r_a/2 = 1/3 * 1/2 = 1/6;$$

Update r:  $r_y = 1/3, r_a = 3/6, r_m = 1/6$  ;

Or we can use matrix update:

$$r^{(t+1)} = M \cdot r^{(t)}$$

	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	0
$r_a$	$\frac{1}{2}$	0	1
$r_m$	0	$\frac{1}{2}$	0

Transition matrix M

### Iteration 1:

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

Matrix update:

$$r^{(t+1)} = M \cdot r^{(t)}$$

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

Iteration 1:

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

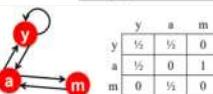
$M$        $r_0$        $r_1$

Initialization:  $r_y = 1/3, r_a = 1/3, r_m = 1/3$  ;

$$\begin{aligned} r'_y &= r_y/2 + r_a/2 = 1/3 * 1/2 + 1/3 * 1/2 = 1/3; \\ r'_a &= r_y/2 + r_m = 1/3 * 1/2 + 1/3 = 3/6; \\ r'_m &= r_a/2 = 1/3 * 1/2 = 1/6; \end{aligned}$$

Update r:  $r_y = 1/3, r_a = 3/6, r_m = 1/6$  ;

They give the same result



$$\begin{aligned} r_y &= r_y/2 + r_a/2 \\ r_a &= r_y/2 + r_m \\ r_m &= r_a/2 \end{aligned}$$

The node scores after iteration 1:  $r_y = 1/3, r_a = 3/6, r_m = 1/6$  ;

### Iteration 2:

$$r'_y = r_y/2 + r_a/2 = 1/3 * 1/2 + 3/6 * 1/2 = 5/12;$$

$$r'_a = r_y/2 + r_m = 5/12 * 1/2 + 3/12 = 11/24;$$

$$r'_m = r_a/2 = 3/6 * 1/2 = 3/12;$$

Update r:  $r_y = 5/12, r_a = 1/3, r_m = 3/12$  ;

The node scores after iteration 2:  $r_y = 5/12, r_a = 1/3, r_m = 3/12$  ;

### Iteration 3:

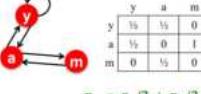
$$r''_y = r_y/2 + r_a/2 = 5/12 * 1/2 + 1/3 * 1/2 = 9/24;$$

$$r''_a = r_y/2 + r_m = 5/12 * 1/2 + 3/12 = 11/24;$$

$$r''_m = r_a/2 = 1/3 * 1/2 = 1/6;$$

Update r:  $r_y = 9/24, r_a = 11/24, r_m = 1/6$  ;

Iteration 4, 5, 7, ..., until converged



$$\begin{aligned} r_y &= r_y/2 + r_a/2 \\ r_a &= r_y/2 + r_m \\ r_m &= r_a/2 \end{aligned}$$

Summarized the results for solution2:

	$r_y$	$r_a$	$r_m$
Initial value:	$1/3 (N=3)$	$1/3$	$1/3$
iter1	$1/3$	$3/6$	$1/6$
iter2	$5/12$	$1/3$	$3/12$
iter3	$9/24$	$11/24$	$1/6$
... Converged	$6/15$	$6/15$	$3/15$

### Verify your result:

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 6/15 \\ 6/15 \\ 3/15 \end{bmatrix} \quad \begin{aligned} r_y &= r_y/2 + r_a/2 \\ r_a &= r_y/2 + r_m \\ r_m &= r_a/2 \end{aligned}$$

Substitute your result into the flow equations to verify whether the equations hold or not

### Compare solution 1 an 2 (same results):

Solution 1: directly solve the linear equations

$$\begin{aligned} \text{"Flow" equations:} \\ r_y &= r_y/2 + r_a/2 \\ r_a &= r_y/2 + r_m \\ r_m &= r_a/2 \\ \Sigma r_i &= 1 \end{aligned}$$

We can solve these 4 linear equations to get the solutions:

$$\begin{aligned} r_y &= 0.4; r_a = 0.4; r_m = 0.2 \\ r_y &= 0.4; r_a = 0.4; r_m = 0.2 \end{aligned}$$

Solution 2: power iteration method

	$r_y$	$r_a$	$r_m$
Initial value:	$1/3 (N=3)$	$1/3$	$1/3$
iter1	$1/3$	$3/6$	$1/6$
iter2	$5/12$	$11/24$	$1/6$
iter3	$6/15$	$6/15$	$3/15$
Converged	$0.4$	$0.4$	$0.2$

## Discussion

- Power iteration method may need many iterations to converge.
- Why we still need power iteration method?

## Discussion

- Why we still need power iteration method?
- It can handle large scale problems.
- In practice, we have a huge transition matrix. Directly solving a large number of linear equations is computational expensive or maybe even intractable.

In power iteration method, we can control the number of iterations to get an approximate solution. Usually, the approximated solution will be good enough.

- Properties of the transition matrix: M
  - Also call stochastic adjacency matrix
  - M should be a column stochastic matrix:

Each column sums to 1; all elements >= 0



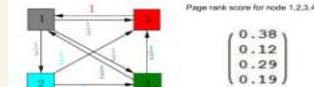
Transition matrix M

	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$r_a$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$r_m$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

The edge weight is defined by the number of out-links ( $d_i$ )  $\frac{1}{d_i}$

All the out-links of one node share the same weight.  
→ non-zero cell will take the same value for each column  
→ the sum of a column is:  $d \cdot (\frac{1}{d}) = 1$ .

More examples will be discussed in the tutorial class



## PageRank extensions

- Dead-end problem

- Spider-trap problem

## The Google matrix

## Two problems:

- 1) the dead-end problem
  - Dead-end nodes

## 2) the spider trap problem

- Spider trap groups

### 1. The dead-end problem: some nodes are dead ends (have no out-links)



	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	$0$
$r_a$	$0$	$0$	$0$

Node b is a dead-end node

Use the power iteration method:

	init	Iter 1	Iter 2
Node a	0.5	0	0
Node b	0.5	0.5	0

$$r^{(t+1)} = M \cdot r^{(t)}$$

The sum of the all node scores for each iteration is reducing (the sum should be 1 if there is no score leaking)



	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	$0$
$r_a$	$0$	$0$	$0$
$r_m$	$0$	$0$	$0$

Node m is a dead-end node

$$r^{(t+1)} = M \cdot r^{(t)}$$

	init	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	...
Node y	1/3	2/6	3/12	5/24	8/48	13/96	...
Node a	1/3	1/6	2/12	3/24	5/48	8/96	...
Node m	1/3	1/6	1/12	1/24	3/48	5/96	...

Score leaking: the sum of each column is reducing

	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	$0$
$r_a$	$\frac{1}{2}$	$0$	$0$
$r_m$	$0$	$0$	$0$

Flow equations:

$$r_y = \frac{1}{2}r_y + \frac{1}{2}r_a$$

$$r_a = \frac{1}{2}r_y$$

$$r_m = \frac{1}{2}r_a$$

$$r_y + r_a + r_m = 1$$

There is no solution

If only consider the top 3 equations, the solution is 0 for each node.

$r_y = r_a = r_m = 0$

## The dead-end problem:



a	b
a	0
b	1

Transition Matrix M

The problem of M:  
It's not a column stochastic matrix, so it cannot meet the assumption of the PageRank algorithm.

A column stochastic matrix should satisfy:  
each column sums to 1; all elements >= 0  
(stochastic here means probability)

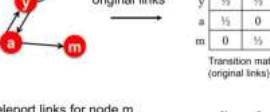
## Solution to Dead End:

- Add teleport links for the dead-end nodes
  - teleport links: the links from the dead-end node to all other nodes
- Fix the transition matrix by adding the teleport links

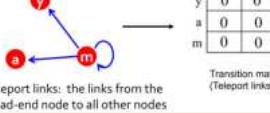
original links Teleport links for node m



original links



Teleport links for node m



teleport links: the links from the dead-end node to all other nodes

## Adding the teleport links to the original graph

- Update the transition matrix

y	a	m
$\frac{1}{3}$	$\frac{1}{3}$	$0$
$\frac{1}{3}$	$0$	$0$
$\frac{1}{3}$	$0$	$0$

Transition matrix M (original links)

y	a	m
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Transition matrix Q (Teleport links)

y	a	m
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

New transition matrix

y	a	m
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Now the sum of column m is 1

Update column m in the original matrix

y	a	m
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Transition matrix Q (Teleport links)

y	a	m
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Transition matrix M (original links)

Transition matrix Q (Teleport links)

$r^{(t+1)} = M \cdot r^{(t)}$

Use the power iteration method:

Init	Iter 1	Iter 2
Node a	0.5	0
Node b	0.5	1

$r^{(t+1)} = M \cdot r^{(t)}$



a spider trap

The problem of M:

It's not a column stochastic matrix, so it cannot meet the assumption of the PageRank algorithm.

A column stochastic matrix should satisfy:

each column sums to 1; all elements >= 0

(stochastic here means probability)

Spider trap issue: eventually, the spider traps absorb all the importance

(high rank scores)

Values remain not changing

Use the power iteration method:

Init	Iter 1	Iter 2
Node a	0.5	0
Node b	0.5	1

$r^{(t+1)} = M \cdot r^{(t)}$

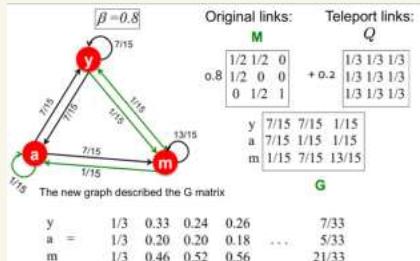
Solution:

Add special teleport links for every node

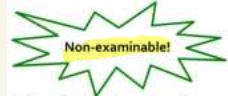
The teleport links will be separately handled

Teleport links to all other points

Teleport



## Further discussions

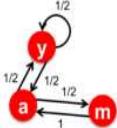


(The following topics are not examinable!)

- PageRank and random walk algorithm
- PageRank and eigenvector

## Random walk

One random walk step:

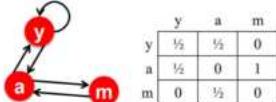


- A random walker on the graph:

▪ At time  $t$ , the walker is at node  $i$

- At time  $t+1$ , the walker follows the probabilities in the transition matrix to jump to another node.

$$\text{Transition matrix } M = \begin{bmatrix} r_y & r_a & r_m \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix}$$



$$\begin{aligned} r_y &= r_y/2 + r_a/2 \\ r_a &= r_y/2 + r_m \\ r_m &= r_a/2 \end{aligned}$$

A random walk problem:

Q: if a walker start at node m, after 3 random-walk steps, what is the location of the walker?  
(calculate the probability of landing for each node)

Solution:

We can change the initialization and use the power iteration method.

Starting from node m:

$$\begin{aligned} \text{1st step: } \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} &= \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \\ \text{2nd step: } \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} &= \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} \\ \text{3rd step: } \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} &= \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/4 \\ 3/4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.75 \\ 0 \end{bmatrix} \end{aligned}$$

The  $r$  vector gives the probability of each node that the walker will visit after 3 random walk steps

## Random walk and PageRank

Non-examinable

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \begin{bmatrix} 1/3 \\ 3/6 \\ 1/6 \end{bmatrix}, \begin{bmatrix} 5/12 \\ 11/24 \\ 1/6 \end{bmatrix}, \dots, \begin{bmatrix} 6/15 \\ 6/15 \\ 3/15 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.4 \\ 0.2 \end{bmatrix}$$

Initial value:  $1/N (N=3)$

Converged

PageRank is a random walk algorithm on a graph:

A walker randomly start at one node (equal probability for each node);  
in each time step, the walker follow the probabilities in the transition matrix to jump to the next node.

PageRank is a random walk algorithm on a graph:

Q: What is the probability of the walker's location after  $m$  time steps?

Perform the power iteration method with random initialization ( $1/n$ ) for  $m$  steps and the PageRank score vector gives the probability for each node.

## PageRank and Eigenvector

- The flow equation:

$$1 \cdot r = M \cdot r$$



$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

the rank vector  $r$  is an eigenvector of the transition matrix  $M$  (with eigenvalue 1)

With the requirement: all rank score sum equals to 1

The converged rank vector  $r$  is called the stationary distribution

Definition of eigenvector:

$$\lambda c = Ac \longrightarrow c: \text{eigenvector}; \lambda: \text{eigenvalue}$$

- Further readings:

- Stanford course CS224w

▪ <http://web.stanford.edu/class/cs224w/slides/04-pagerank.pdf>

- Lecture from Cornell

▪ <http://pi.math.cornell.edu/~mec/Winter2009/RalucaReimus/Lecture3/lecture3.html>

# Graph Neural Network

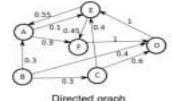
## Outline

- Introduction to graphs
- Graph convolutional network (GCN)
  - Network prediction (forward pass)
  - GCN applications/tasks
- Network training
  - Loss functions
  - Backward pass

## Introduction to graphs

### A graph

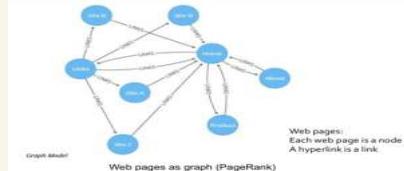
- Nodes and links (edges). May have weights on links.
- Weights indicate the strength of links
- Links can be directed or undirected.
- One undirected link can be treated as two directed links (forms bidirectional connections).



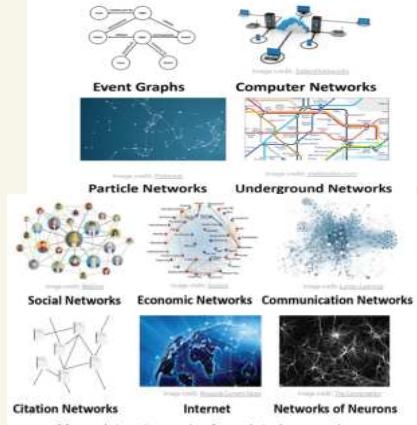
[http://www.mathworks.com/help/matlab/creating\\_gm.html#examples](http://www.mathworks.com/help/matlab/creating_gm.html#examples)  
[https://www.mathworks.com/help/matlab/creating\\_gm.html#examples](https://www.mathworks.com/help/matlab/creating_gm.html#examples)



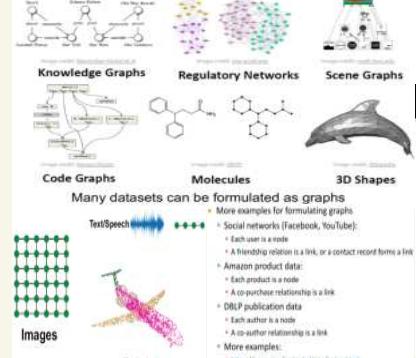
## Graph examples



Many datasets can be formulated as graphs



Many datasets can be formulated as graphs



# Graph convolutional network

## Graph neural network (GNN)

- Neural networks for graph data

## Graph convolutional network (GCN)

- GCN is a type of GNN
- Main idea:  
Aggregate the information from neighbourhoods to generate node features.

## Node classification task

### GCN output for node classification

- Predict a class label for each node
  - In practice, we predict a **class score vector** for each node. Each score indicates the confident for one class.
- In this class, we focus on the node classification task using Graph convolutional network (GCN).

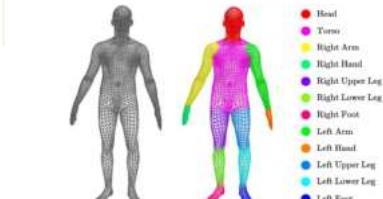
Example: Assume we have 4 classes in total. We aim to predict a 4-dimensional score vector for each node. One dimension corresponds to one class

For node A, suppose the GCN output is the following score vector: [0.1, 0.1, 0.6, 0.2]

As the 3<sup>rd</sup> element has the largest value, we predict the class label as 3 for node A.

## An application example

### GCN for node classification on 3D meshes:



Graph segmentation task: predict a body-part class label for each vertex.  
 Node features: (x,y,z) coordinates of each vertex  
<https://medium.com/state-of-deep-learning-on-3d-meshes-9608a3b13c98>

## GCN learning tasks

### GCN for other learning tasks

- Example: graph-level classification
  - Classification for the whole graph rather than each node.
- Example: node regression problem
  - Predict a real number for each node.

## GCN

### Graph convolutional network (GCN)

- Input: node features and graphs (affinity matrix)
- Forward pass for prediction
- Backward pass for training

## GCN input

### Input of GCN:

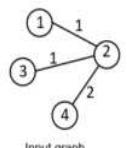
- The input (initial) feature vector for each node.
  - We assume the feature vector for each node is available.
  - The feature vector encode the information for each individual node
- The generation of the input feature vectors for each node depends on the applications.
  - E.g., in the 3D point segmentation example, the feature vector for each node is the (x, y, z) coordinates of the vertex
- The graph affinity matrix A.
  - describing the connections in a graph

### The graph affinity matrix A.

- In practice, we use row normalized affinity matrix A'.

$a_{i,j}$  : The element  $(i,j)$  in the affinity matrix A.

It is the edge weight of the edge between node i and node j. If there is no link between  $(i, j)$ ,  $a_{i,j} = 0$ ;



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

affinity matrix

### Normalized affinity matrix A'.

Row normalized affinity matrix A': the sum of each row in A' is 1.

$$a'_{ij} = \frac{a_{ij}}{\sum_{k=1}^N a_{ik}}$$

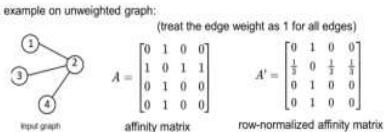
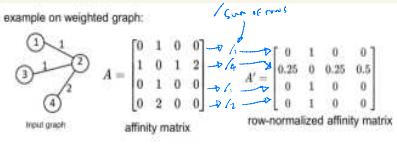
$\sum_{k=1}^N a_{ik}$  : is the sum of the i-th row in the original Affinity matrix A for node i

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

$$A' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.25 & 0 & 0.25 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

row-normalized affinity matrix

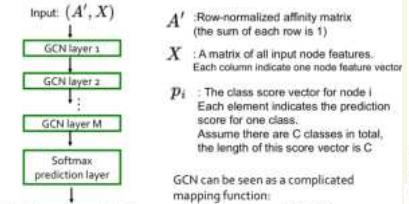
After normalization, the matrix is not symmetric



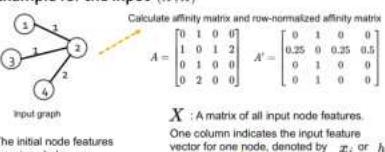
## GCN forward pass

GCN is a stack of multiple GCN layers

An example for node classification tasks.



Example for the input  $(A', X)$



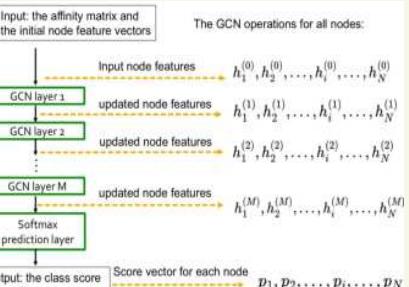
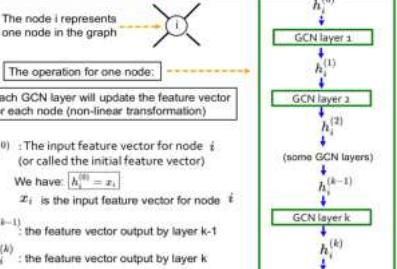
The initial node features are given below as:

$$\begin{aligned} x_1 &= [1, -1]^T, \\ x_2 &= [0, 1]^T, \\ x_3 &= [-1, 0]^T, \\ x_4 &= [1, 0]^T. \end{aligned}$$

$$h_i^{(0)} = x_i$$

$$\begin{bmatrix} 1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 \\ h_1^{(0)} & h_2^{(0)} & h_3^{(0)} & h_4^{(0)} \end{bmatrix}$$

## GCN is applied to every node



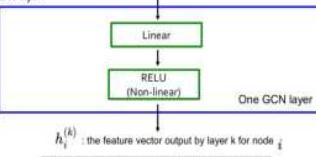
## One GCN layer

The calculation for one GCN layer:  
Each GCN layer consist of two blocks (operations):

1. linear block; 2. Non-linear block

$h_i^{(k-1)}$  : the feature vector output by layer  $k-1$  for node  $i$

The  $k$ -th GCN layer:

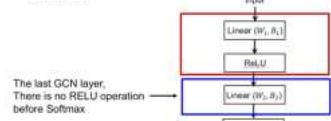


The output node feature vector is also called: node representation or node embedding

GCN layer exception:

In the last GCN layer, usually there is no non-linearity block.

Example:



<https://stats.stackexchange.com/questions/167099/how-to-add-final-softmax-layer-in-a-convolutional-neural-network>

Linear transform:  $\hat{h}_i^{(k)} = W_k \sum_{j \in \mathcal{N}_i} a_{ij}^{(k)} h_j^{(k-1)} + B_k h_i^{(k-1)}$

RELU (Non-linear transform):  $h_i^{(k)} = \max(0, \hat{h}_i^{(k)})$

$a_{ij}^{(k)}$  : element  $(i,j)$  in the row-normalized affinity matrix  $A'$ ;  $\mathcal{N}_i$ : the set of neighbors of node  $i$  (connected nodes)

$h_i^{(k-1)}$  : input feature vector of node  $i$  for layer  $k-1$

$h_i^{(k)}$  : output feature vector of node  $i$  for layer  $k$

$W_k$  : weight matrix for neighbourhood aggregation.

$B_k$  : weight matrix for self transformation

Linear transform:  $\hat{h}_i^{(k)} = W_k \sum_{j \in \mathcal{N}_i} a_{ij}^{(k)} h_j^{(k-1)} + B_k h_i^{(k-1)}$

neighbourhood aggregation

self transformation

$a_{ij}^{(k)}$  : indicates the edge weight in the graph.

it is the element  $(i,j)$  in the row-normalized affinity matrix  $A'$ :

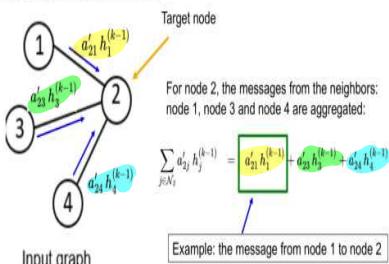
$W_k, B_k$  : are the GCN layer parameters. They are two matrixes. They are learnable parameters (or called layer weights).

In the exam or tutorial questions, the values of  $W_k, B_k$  will be given in the questions.

Network training: the training of GCN is to solve an optimization problem to obtain the values of these layer parameters

1. Neighbourhood aggregation : Aggregate the messages from the neighbours. Neighbours here means directly connected nodes.

Aggregation calculation example:



## Aggregation example

Target node

For node 2, the messages from the neighbors: node 1, node 3 and node 4 are aggregated:

$$\sum_{j \in \mathcal{N}_2} a_{2j}^{(k-1)} h_j^{(k-1)} = a_{21}^{(k-1)} h_1^{(k-1)} + a_{23}^{(k-1)} h_3^{(k-1)} + a_{24}^{(k-1)} h_4^{(k-1)}$$

Input graph

$a_{ij}^{(k)}$  is the weight on the edge, which can be obtained from the row-normalized affinity matrix

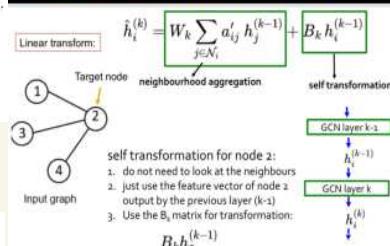
$$A' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

row-normalized affinity matrix

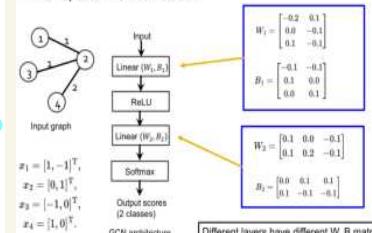
After aggregation, we perform transformation using  $W_k$ :

$$W_k \sum_{j \in \mathcal{N}_2} a_{2j}^{(k-1)} h_j^{(k-1)}$$

## Self transformation example



Examples of  $W$  and  $B$  matrix:



## Key idea of GCN:

- Aggregate the information (messages) from the neighbours to generate the features for one node.



<https://arxiv.org/pdf/1901.00596.pdf>

## One GCN layer

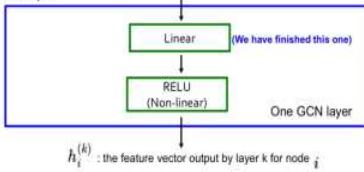
The calculation for one GCN layer:

Each GCN layer consist of two blocks (operations):

1. linear block; 2. Non-linear block

$$h_i^{(k-1)} : \text{the feature vector output by layer } k-1 \text{ for node } i$$

The k-th GCN layer:



The output node feature vector is also called: **node representation** or **node embedding**.

### Non-linearity functions

- Also called activation functions
- Like a "gate" or a tiny classifier: open (>0) or close (<=0)

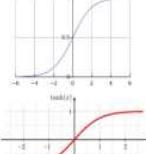
#### 1. Rectified linear unit (ReLU)

$$\text{ReLU}(x) = \max(x, 0)$$



#### 2. Sigmoid

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



#### 3. Hyperbolic tangent

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

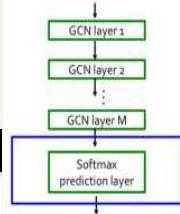


## GCN forward pass

- GCN is a stack of multiple GCN layers

An example for node classification tasks.

Input:  $(A', X)$



$p_i$ : The class score vector for node  $i$ . Each element indicates the prediction score for one class. Assume there are  $C$  classes in total, the length of this score vector is  $C$

Example for 3-class classification:

$$p_1 = [0.1 \ 0.1 \ 0.8]$$

$$p_2 = [0.2 \ 0.3 \ 0.5]$$

....

(We now come to this one)

Output for each node  $i$ :  $p_i$

### Softmax layer (prediction layer)

- The output values lie in [0 1]

\* Can be used as a normalization layer

- Turn the input vector into a probability output vector

\* Sum of all elements is 1. Each element  $\geq 0$ .

\* The output is class probability scores

Recall: GCN for classification:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^C e^{x_j}}$$

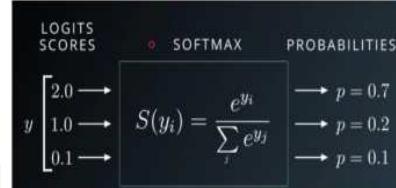
Example (3 classes):

The output is a 3-dimensional score vector for each node

$$\begin{bmatrix} 1.2 \\ 0.9 \\ 0.4 \end{bmatrix} \xrightarrow{\text{Softmax}} \begin{bmatrix} 0.46 \\ 0.34 \\ 0.20 \end{bmatrix}$$

Another example:

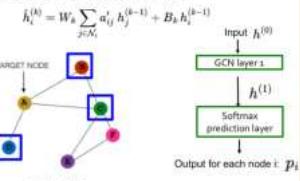
The input vector before using the Softmax function for class prediction is called logits.



<https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-3fa59641e86d>

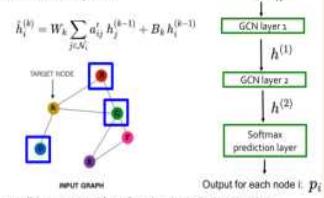
## Discussion

If we only have 1 layer of GCN, we only perform local message passing



<http://web.stanford.edu/class/cs224n/slides/06-GNNs.pdf>

How about 2 GCN layers?



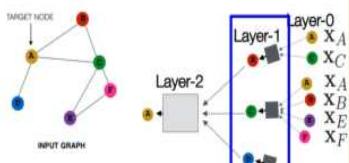
<http://web.stanford.edu/class/cs224n/slides/06-GNNs.pdf>

Messages will be propagated from distant nodes to the target node by multi-layer neighbourhood propagation.

$h^{(2)}$  Will encode information from distant nodes

**Message passing is not local in a multi-layer GCN.**

Messages will be propagated from distant nodes to the target node by multi-layer neighbourhood propagation.

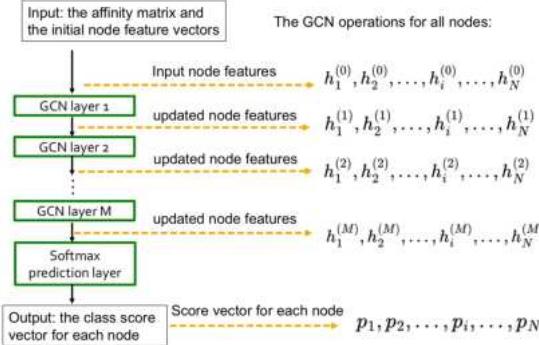


Example:

This two-layer GCN allows the message propagation from any other nodes in the graph to the target node A

<http://web.stanford.edu/class/cs224n/slides/06-GNNs.pdf>

## Recap of GCN for node classification



## GCN Forward Pass example

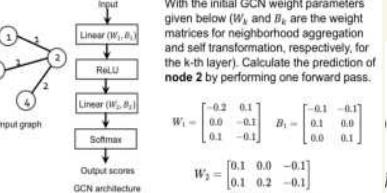
### Question:

Given a graph below, the task is to do node-wise classification using a 2-layer graph convolutional network (GCN) and a cross-entropy loss, with network architecture shown. The initial node features are given below as:



$$x_1 = [1, -1]^T, x_2 = [0, 1]^T, \\ x_3 = [-1, 0]^T, x_4 = [1, 0]^T.$$

### Question: (continued from last page)



## 1<sup>st</sup> GCN layer

### GCN Forward Pass Example

#### Solution:

Calculate affinity matrix and row-normalized affinity matrix

$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$     $A' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.25 & 0 & 0.25 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

Linear transform:  $\hat{h}_i^{(0)} = W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)} + B_k h_i^{(k-1)}$

Non-linear transform (ReLU):  $h_i^{(k)} = \max\{0, \hat{h}_i^{(k)}\}$

### Forward pass for node 1

J process node

We have  $h_i^{(0)} = x_i$

Linear transform:

$$\sum_{j \in \mathcal{N}_1} a'_{1j} h_j^{(0)} = a'_{12} h_2^{(0)} = 1 \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$W_1 \sum_{j \in \mathcal{N}_1} a'_{1j} h_j^{(0)} = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix}$$

$$B_1 h_1^{(0)} = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1 \\ -0.1 \end{bmatrix}$$

Linear transform:

$$\hat{h}_1^{(1)} = W_1 \sum_{j \in \mathcal{N}_1} a'_{1j} h_j^{(0)} + B_1 h_1^{(0)} = \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.0 \\ -0.2 \end{bmatrix}$$

Non-linear transform (ReLU):

$$h_1^{(1)} = \max\{0, \hat{h}_1^{(1)}\} = \begin{bmatrix} 0.1 \\ 0.0 \\ 0.0 \end{bmatrix}$$

From which node  
↑  
(?)

$h \times$

↓  
Current nodes

### Forward pass for node 2

We have  $h_i^{(0)} = x_i$

$$\sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(0)} = a'_{21} h_1^{(0)} + a'_{23} h_3^{(0)} + a'_{24} h_4^{(0)} \\ = 0.25 \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} + 0.25 \times \begin{bmatrix} -1 \\ 0 \end{bmatrix} + 0.5 \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ = \begin{bmatrix} 0.5 \\ -0.25 \end{bmatrix}$$

$$W_1 \sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(0)} = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \\ 0.1 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0.5 \\ -0.25 \end{bmatrix} = \begin{bmatrix} -0.125 \\ 0.025 \\ 0.075 \end{bmatrix}$$

$$B_1 h_2^{(0)} = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0.0 \\ 0.1 \end{bmatrix}$$

Linear transform:

$$\hat{h}_2^{(1)} = W_1 \sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(0)} + B_1 h_2^{(0)} = \begin{bmatrix} -0.125 \\ 0.025 \\ 0.075 \end{bmatrix} + \begin{bmatrix} -0.1 \\ 0.0 \\ 0.1 \end{bmatrix} = \begin{bmatrix} -0.225 \\ 0.025 \\ 0.175 \end{bmatrix}$$

Non-linear transform (ReLU):

$$h_2^{(1)} = \max\{0, \hat{h}_2^{(1)}\} = \begin{bmatrix} 0.0 \\ 0.025 \\ 0.175 \end{bmatrix}$$

### Forward pass for node 3

We have  $h_i^{(0)} = x_i$

$$\sum_{j \in \mathcal{N}_3} a'_{3j} h_j^{(0)} = a'_{32} h_2^{(0)} = 1 \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$W_1 \sum_{j \in \mathcal{N}_3} a'_{3j} h_j^{(0)} = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \\ 0.1 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix}$$

$$B_1 h_3^{(0)} = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \times \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \\ 0.0 \end{bmatrix}$$

Linear transform:

$$\hat{h}_3^{(1)} = W_1 \sum_{j \in \mathcal{N}_3} a'_{3j} h_j^{(0)} + B_1 h_3^{(0)} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.1 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.2 \\ -0.2 \\ -0.1 \end{bmatrix}$$

Non-linear transform (ReLU):

$$h_3^{(1)} = \max\{0, \hat{h}_3^{(1)}\} = \begin{bmatrix} 0.2 \\ 0.0 \\ 0.0 \end{bmatrix}$$

### Forward pass for node 4

We have  $h_i^{(0)} = x_i$

$$\sum_{j \in \mathcal{N}_4} a'_{4j} h_j^{(0)} = a'_{42} h_2^{(0)} = 1 \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$W_1 \sum_{j \in \mathcal{N}_4} a'_{4j} h_j^{(0)} = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \\ 0.1 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix}$$

$$B_1 h_4^{(0)} = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0.1 \\ 0.0 \end{bmatrix}$$

Linear transform:

$$\hat{h}_4^{(1)} = W_1 \sum_{j \in \mathcal{N}_4} a'_{4j} h_j^{(0)} + B_1 h_4^{(0)} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.1 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.0 \\ -0.1 \end{bmatrix}$$

Non-linear transform (ReLU):

$$h_4^{(1)} = \max\{0, \hat{h}_4^{(1)}\} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

## Summary (1<sup>st</sup> GCN layer)

Node features output by the 1<sup>st</sup> GCN layer:

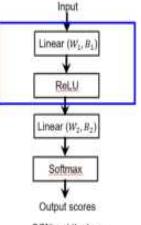
$$h_1^{(1)} = \max\{0, h_1^{(1)}\} = \begin{bmatrix} 0.1 \\ 0.0 \end{bmatrix}$$

$$h_2^{(1)} = \max\{0, h_2^{(1)}\} = \begin{bmatrix} 0.0 \\ 0.025 \end{bmatrix}$$

$$h_3^{(1)} = \max\{0, h_3^{(1)}\} = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$$

$$h_4^{(1)} = \max\{0, h_4^{(1)}\} = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$$

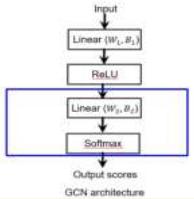
We have finished the 1<sup>st</sup> GCN layer



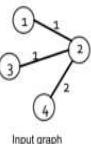
GCN architecture

## 2<sup>nd</sup> GCN layer

Next step: the 2<sup>nd</sup> GCN layer and Softmax prediction



### Forward pass (2<sup>nd</sup> GCN layer) for node 2



$$\begin{aligned}
 \sum_{j \in N_2} a'_{2j} h_j^{(1)} &= a'_{21} h_1^{(1)} + a'_{23} h_3^{(1)} + a'_{24} h_4^{(1)} \\
 &= 0.25 \times \begin{bmatrix} 0.1 \\ 0.0 \end{bmatrix} + 0.25 \times \begin{bmatrix} 0.2 \\ 0.0 \end{bmatrix} + 0.5 \times \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} \\
 &= \begin{bmatrix} 0.075 \\ 0.0 \end{bmatrix}
 \end{aligned}$$

$$\begin{aligned}
 W_2 \sum_{j \in N_2} a'_{2j} h_j^{(1)} &= \begin{bmatrix} 0.1 & 0.0 & -0.1 \\ 0.1 & 0.2 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0.075 \\ 0.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.0075 \\ 0.0075 \\ 0.0 \end{bmatrix} \\
 B_2 h_2^{(1)} &= \begin{bmatrix} 0.0 & 0.1 & 0.1 \\ 0.1 & -0.1 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0.0 \\ 0.025 \\ 0.175 \end{bmatrix} = \begin{bmatrix} 0.02 \\ -0.02 \end{bmatrix}
 \end{aligned}$$

Linear transform:

$$\hat{h}_2^{(2)} = W_2 \sum_{j \in N_2} a'_{2j} h_j^{(1)} + B_2 h_2^{(1)} = \begin{bmatrix} 0.0075 \\ 0.0075 \end{bmatrix} + \begin{bmatrix} 0.02 \\ -0.02 \end{bmatrix} = \begin{bmatrix} 0.0275 \\ -0.0125 \end{bmatrix}$$

Softmax:

$$h_2^{(2)} = \text{Softmax}(\hat{h}_2^{(2)}) = \text{Softmax}\left(\begin{bmatrix} 0.0275 \\ -0.0125 \end{bmatrix}\right) = \begin{bmatrix} 0.51 \\ 0.49 \end{bmatrix}$$

$$e^{-1} + e^{-2} = e^{-0.0275} + e^{-0.0125} = 1.0279 + 0.9876 = 2.0155$$

## GCN discussions

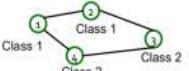
### Input feature vectors for GCN:

The input node feature vector depends on applications:

- Web: web page descriptions (extra text features);
- Social networks: User profile (text features) + User images (image features).
- 3D recognition: 3D coordinate (x, y, z) for each 3D point
- General case:  
Using one-hot feature vectors (or called indicator vectors).  
E.g., given 4 nodes in total, their on-hot feature vectors can be:  
 $x1=[1, 0, 0, 0]^T$ ;  
 $x2=[0, 1, 0, 0]^T$ ;  
 $x3=[0, 0, 1, 0]^T$ ;  
 $x4=[0, 0, 0, 1]^T$ .  
A one-hot vector: encode the identity of one input node

### Node classification

Predict a class label for each node in a graph



### Graph classification

Predict a class label for the whole graph



## GCN applications

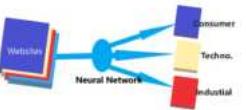
### Web page topic prediction

Input: Graph of web pages;

- Node: Web page;
- Edge: Hyper-link between web pages;
- Input node features: extra text features from the webpage

Task: Predict the topic of the webpage

- Classify each node

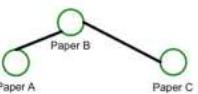


<https://www.datascience.com/introduction-classification-websites-machine-learning-with-hands-on-python-3rd-edition.pdf>

### Paper research topic classification

Datasets: publication datasets

- Node: each paper;
- Edge: citation relation;



Predict the topic of the paper as a classification problem

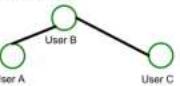
Topics: (action recognition, semantic segmentation, instance segmentation ...)

### User attribute (tag) prediction

Attributes can be occupations, professionals, interests, ...

Datasets: social network datasets

- Node: each user;
- Edge: friendship relation; contact record; subscription, following relationship.



Predict the attributes of a user as a binary classification problem for each attribute:

Interests: (dancing, reading, traveling, cycling, swimming, PC gaming, ...)



User: User interactions can be modeled as a bipartite undirected graph.

### GCN for Recommender Systems

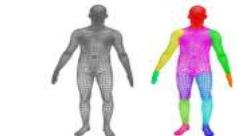
Classify positive and negative pairs.  
Positive (u, v): connected pairs.

Negative (u, v): not connected pairs.

Use dot-product to calculate the similarity of a user-item pair



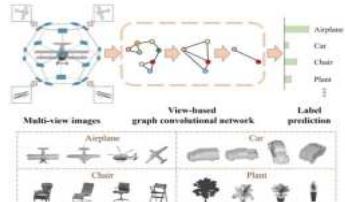
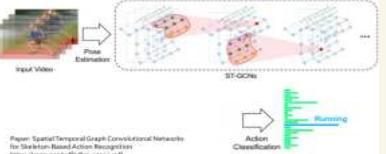
### GCN for node classification on 3D meshes



Graph segmentation task: each vertex in the mesh is assigned to one of twelve body-parts.



### GCN for skeleton based action recognition



### GCN for 3D object classification

Paper: View-GCN: View-based Graph Convolutional Network for 3D Shape Analysis

#### GCN for scene graph generation and captioning



Image captioning

The woman in shorts is standing behind the man who is jumping over fire hydrant.

<https://arxiv.org/pdf/2203.00443.pdf>

## More applications

### A collection of graph Learning tutorials/applications

- <https://medium.com/stanford-cs224w>
- [https://github.com/cvvc-team/pytorch\\_geometric\\_implemented\\_gnn\\_models](https://github.com/cvvc-team/pytorch_geometric_implemented_gnn_models)

### other examples:

- <https://paperswithcode.com/task/node-classification>

### Other deep learning methods for graphs:

- Transformer
  - Another popular method that can be used on graphs
    - <https://arxiv.org/pdf/2302.03937.pdf>
    - <https://www.stanford.edu/cs224w/>

# GCN training

## GCN training

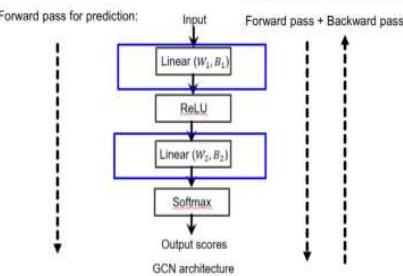
### Loss functions

- Cross-entropy loss
- L2 loss

### Backward pass

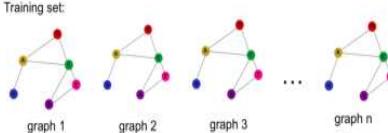
- gradient descent methods

Use gradient descent for training



Training: find solutions to the network parameters in each layer!

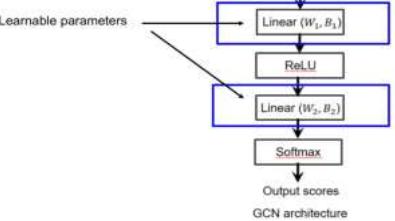
## Training data



+  
Class label for each node  
(ground-truth labels)

Train a GCN model  
↓  
GCN layer  
↓  
GCN layer

Example:



## GCN loss functions

1. minimize it

### 1. Cross-entropy loss for classification:

$$L(f(x)) = - \sum_i y_i \cdot \log z_i(\theta)$$

$$z_{ik}(\theta) = \text{Softmax}(h_{ik}) = \frac{e^{h_{ik}(\theta)}}{\sum_{j=1}^C e^{h_{ij}(\theta)}}$$

### 2. For node regression problems, you may use L2 loss: E.g., predict real numbers for each node

$$L(f(x)) = \sum_i \|y_i - h_i(x)\|^2$$

## GCN Loss functions

### \* Loss function for node classification

- Cross-entropy loss (vector expression, also called softmax loss)

$$L(f(x)) = - \sum_i y_i \cdot \log z_i(\theta)$$

$$z_{ik}(\theta) = \text{Softmax}(h_{ik}) = \frac{e^{h_{ik}(\theta)}}{\sum_{j=1}^C e^{h_{ij}(\theta)}}$$

$\mathbf{z}_{ik}$ : The score vector output by Softmax for the node i

$z_{ik}$ : The k-th element the score vector  $z_i$   
 $h_{ik}$ : The k-th element of the logit vector output by the network, corresponding to the k-th class.

$\mathbf{y}_i$ : The one-hot vector indicating the ground-truth class label

Cross-entropy measures the distance between the ground-truth vector and the prediction vector.

One-hot vector example:

One-hot vector is the ground-truth class score vector

Groundtruth class label:

Represented by one-hot vector:

$$\text{Class 1} \rightarrow [1, 0, 0, 0]^T$$

$$\text{Class 2} \rightarrow [0, 1, 0, 0]^T$$

$$\text{Class 3} \rightarrow [0, 0, 1, 0]^T$$

$$\text{Class 4} \rightarrow [0, 0, 0, 1]^T$$

For the c-th class, the c-th element in the one-hot vector is set to 1

Cross-entropy loss:

$$\min_{\theta} L(Y, Z) = - \sum_i y_i \cdot \log z_i(\theta)$$

If the score vector  $z_i$  is similar to the groundtruth vector  $y_i$ , the loss is small.

$\theta$  is the learnable parameters of all GCN layers:

$$\theta = \{W_1, B_1; W_2, B_2; \dots; W_K, B_K\}$$

GCN training:

We need to solve the optimization problem to obtain the solution of the learnable parameters.

$$L(f(x)) = - \sum_i y_i \cdot \log z_i(\theta)$$

► Cross-entropy loss can be also written as:

$$L(f(x)) = - \sum_i \sum_{k=1}^C y_{ik} \log z_{ik}(\theta)$$

$$z_{ik}(\theta) = \text{Softmax}(h_{ik}) = \frac{e^{h_{ik}(\theta)}}{\sum_{j=1}^C e^{h_{ij}(\theta)}}$$

k indicates the k-th class. C is the total number of classes

$y_{ik}$ : is the k-th element in the groundtruth one-hot vector  $\mathbf{y}_i$

Cross-entropy between two distributions: measure the distance (similarity) between two distributions:

Example: we have 3 classes in total.

y: ground-truth distribution for one sample

z: predicted distribution for one sample

Ground-truth	Prediction 1	Prediction 2
$\mathbf{y}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	$\mathbf{z}_1 = \begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix}$	$\mathbf{z}'_1 = \begin{bmatrix} 0.1 \\ 0.9 \\ 0.0 \end{bmatrix}$

Cross-entropy for one sample:  $L_i(\mathbf{y}_1, \mathbf{z}_1) = - \sum_{k=1}^C y_{ik} \log z_{ik}$

$$L(\mathbf{y}_1, \mathbf{z}_1) = - \log(0.5) = 0.3010$$

Large distance

$$L(\mathbf{y}_1, \mathbf{z}'_1) = - \log(0.9) = 0.0458$$

Small distance

We need to solve this optimization problem to obtain the network parameters (weights)

Cross-entropy loss:

$$\min_{\theta} L(Y, Z) = - \sum_i y_i \cdot \log z_i(\theta)$$

GCN training:

We need to solve the optimization problem to obtain the solution of the learnable parameters

## \* GCN training

### \* Backward pass (non-examinable)

- gradient descent methods



## \* Use gradient decent to solve the optimization

Iterative algorithm: repeatedly update weights in the (opposite) direction of gradients until convergence.

$$\theta(t+1) = \theta(t) + \Delta\theta(t);$$

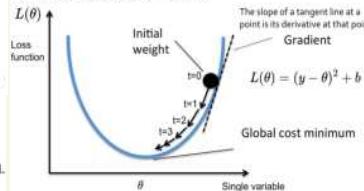
$$\Delta\theta(t) = -\eta \nabla_{\theta} L;$$

$\Delta\theta(t)$ : The update vector, a small change to the solution.

$\eta$ : Learning rate (LR), the step size for gradient update: Hyperparameter that controls the size of gradient step Can vary over the course of training (LR scheduling)

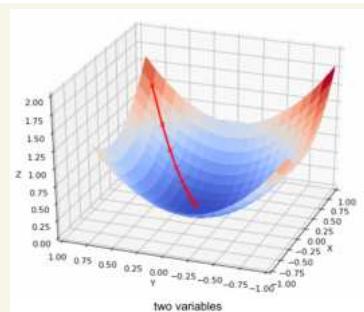
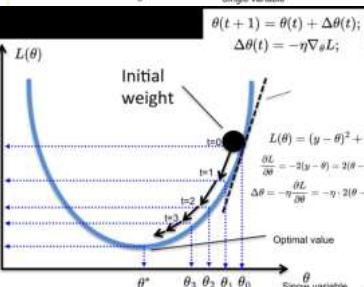
Gradient vector: a vector of partial derivatives of all variables.

Two key factors for gradient descent:  
1 update direction (gradients),  
2 update step size (learning rate)



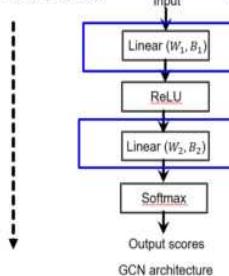
$$\theta(t+1) = \theta(t) + \Delta\theta(t);$$

$$\Delta\theta(t) = -\eta \nabla_{\theta} L;$$

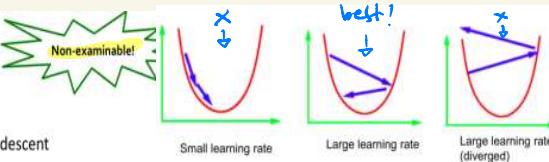


# GCN training

Forward pass for prediction:  
Input  
Use gradient descent for training  
Each training iteration:  
Forward pass + Backward pass

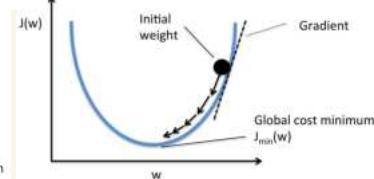


Use gradient descent for training  
Each training iteration:  
Forward pass + Backward pass

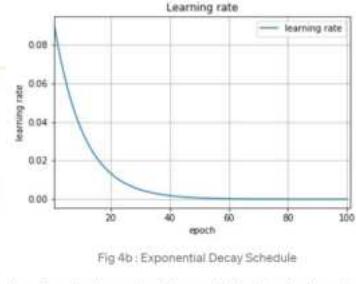


We need a good setting of learning rate

Learning rate needs to be gradually decreased  
Learning rate (LR): the step size for gradient update

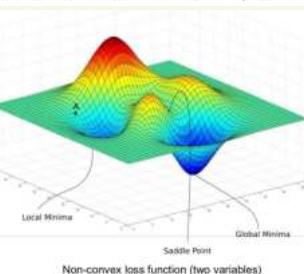
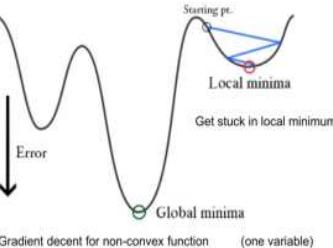


learning rate decay strategies:  
how to gradually decrease the learning rate



Learning rate decay scheduler: gradually reduce the learning rate

## Limitations of Gradient decent



## GCN training

### Backward pass example on a simplified network (2 layers, no RELU)

Forward:  
 $h(x) = W_1 \cdot x$      $g(h) = W_2 \cdot h$      $L = (g - y)^2$   
vector  $x$      $h(1^{\text{st}} \text{ layer}, W_1)$      $g(2^{\text{nd}} \text{ layer}, W_2)$      $L(\text{loss})$

Backward:  
 $\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial W_1}$      $\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial W_2}$

Using chain rule for gradient calculation

### Backward pass example on a simplified network (2 layers, no RELU)

Forward:  
 $h(x) = W_1 \cdot x$      $g(h) = W_2 \cdot h$      $L = (g - y)^2$   
vector  $x$      $h(1^{\text{st}} \text{ layer}, W_1)$      $g(2^{\text{nd}} \text{ layer}, W_2)$      $L(\text{loss})$

Backward:  
 $\frac{\partial L}{\partial W_1} = \nabla L_h \cdot \frac{\partial h}{\partial W_1}$      $\frac{\partial L}{\partial W_2} = \nabla L_g \cdot \frac{\partial g}{\partial W_2}$      $\nabla L_g = \frac{\partial L}{\partial g}$

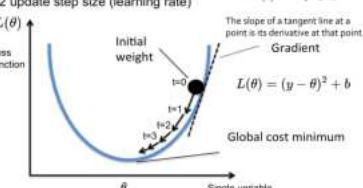
$\nabla L_h = \frac{\partial L}{\partial h} = \nabla L_g \cdot \frac{\partial g}{\partial h}$

Using chain rule for gradient calculation

### Some concepts for minibatch SGD:

- Batch size: the number of data points in a minibatch e.g., number of nodes for node classification task
- Iteration: 1 step of SGD on a minibatch
- Epoch: one full pass over the dataset (# iterations is equal to ratio of dataset size and batch size)

Two key factors for gradient descent:  
1 update direction (gradients),  
2 update step size (learning rate)



### Other details

- In practice, we optimize this loss function:  
Cross-entropy loss + L2 regularization (weight decay)

$$\min_{\theta} L(Y, Z) = - \sum_i y_i \cdot \log z_i(\theta) + \frac{1}{2} \lambda \|\theta\|^2$$

L2 regularization: encourage small values for the solution

- Use momentum mechanism for gradient update (Momentum SGD)

SGD for solution update:

$$\theta(t+1) = \theta(t) + \Delta\theta(t)$$

$$\text{Standard SGD: } \Delta\theta(t) = -\eta \Delta_\theta L$$

$$\text{Momentum SGD: } \Delta\theta(t) = -\eta \Delta_\theta L + \alpha \Delta\theta(t-1)$$

Momentum

## Momentum:

1. Encourage the update directions of two consecutive iterations to be similar (more consistent)

### 2. Noise Smoothing effect

Momentum averages gradients of recent iterations.

It is a running average of gradients that we use for each update step.

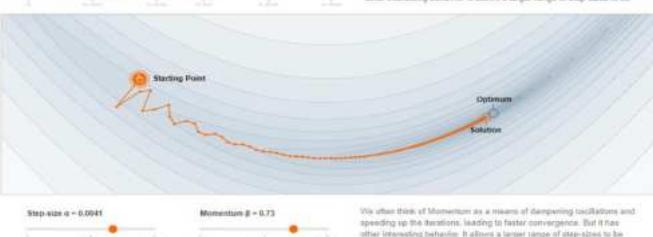
$$\text{Momentum SGD: } \Delta\theta(t) = -\eta\Delta_\theta L + \alpha\Delta\theta(t-1)$$

Momentum



With momentum, two consecutive updates are smooth.

<https://distill.pub/2017/momentum/>



### Other details

- More advanced gradient methods (adaptive methods):

- ADAM
  - <https://arxiv.org/abs/1412.6980>

### LARS

- <https://arxiv.org/abs/1708.03888>

## GCN training

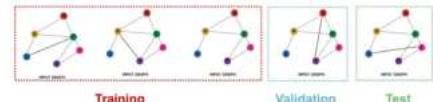
### Determine hyper-parameters:

#### Hyper-parameters:

- network architectures (e.g., #layers),
- network training iterations,
- learning rate, etc.

### Use a validation set to determine hyper-parameters.

- Test the model with different hyper-parameters on the validation set, choose the setting with the best performance



### Further readings:

#### Deep learning basics

- ConvNet, Batch normalization, residual/skip connections, optimizer (ADAM, momentum SGD), up/down-sampling, pooling, ...
- <http://cs231n.stanford.edu/>
- <https://atcold.github.io/pytorch-Deep-Learning/>

#### Transformers

- Another popular network architecture that can be used on graphs
- <http://web.stanford.edu/class/cs224w/>

#### Other advanced topics on GNN

- <https://github.com/AntonioLonga/PytorchGeometricTutorial>
- <http://web.stanford.edu/class/cs224w/>
- <https://medium.com/stanford-cs224w/>
- <http://web.stanford.edu/class/cs224w/>
- GraphSAGE, Graph attention network

### Online implementation

#### PyTorch Geometric

- <https://www.pyg.org/>
- PyG is a library built upon PyTorch to easily write and train Graph Neural Networks for a wide range of applications
- <https://pytorch-geometric.readthedocs.io/en/latest/notes/resources.html>
- <https://pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html>

#### Deep graph library

- <https://github.com/dmlc/dgl>

### Graph classification datasets:

- <https://pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html>

- <https://chrsmrrs.github.io/datasets/docs/datasets/>

- <https://paperswithcode.com/task/node-classification>

# Similarity Search - LSH

## Outline

- Nearest Neighbour Search (NNS) Problems
- Local Sensitive Hashing (LSH) – Random Projection
- Product Quantization (PQ)
- Inverted File Index (non-examinable!!)

## Similarity Search Applications

Visually similar results



<https://web.stanford.edu/class/cs246/>



## How does it work?

Query image



Search similar images from the database

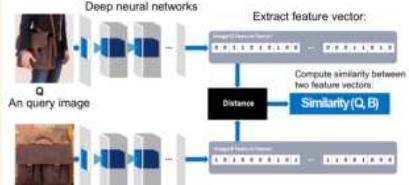


simple method (linear search):

Compute similarity between the query image and all images in the database.

Return the top-k similar images in the database.

Computational expensive for large scale database (e.g., 1 billion images in the database)



An image from the database

Similarity calculation for two images

## Nearest Neighbour Search (NNS)

- k-nearest neighbour search
  - Given a query, identify the top-k nearest neighbours (top-k most similar items) in the database
  - The result is based on a certain type of similarity metric. (e.g., L2 distance)
- Similarity metrics
  - Euclidean distance (L2 distance)
  - Cosine similarity
  - Hamming distance (for binary data)
  - Manhattan distance (L1 distance)
  - Inner product
  - ...

## Similarity

### 1. Euclidean distance (L2 distance):

$$d_{L2}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

### 2. Inner product (dot product):

The dot product of two vectors  $a = [a_1, a_2, \dots, a_n]$  and  $b = [b_1, b_2, \dots, b_n]$  is defined as:

$$a \cdot b = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

Geometric definition:

$$a \cdot b = \|a\| \|b\| \cos \theta,$$

where  $\theta$  is the angle between  $a$  and  $b$ .



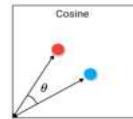
### 3. Cosine similarity

Given two vectors of attributes,  $A$  and  $B$ , the cosine similarity,  $\text{cos}(0)$ , is represented using a dot product and magnitude as:

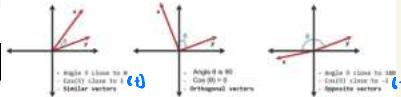
$$\text{similarity} = \text{cos}(\theta) = \frac{a \cdot b}{\|a\| \|b\|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}}$$

where  $a_i$  and  $b_i$  are components of vector  $A$  and  $B$  respectively.

Cosine Distance = 1 - Cosine Similarity



$$\text{similarity} = \text{cos}(\theta) = \frac{a \cdot b}{\|a\| \|b\|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}},$$



### 4. Hamming distance: $0 \times \infty$

- The number of positions with different values.
- Binary data: the number of different bits in two binary vectors.
- A good option for similarity calculation of binary data

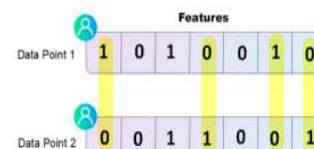
$x_1$	$x_2$	$x_3$	$x_4$
0	$\neq$ 1		
1	$\neq$ 0		
1	$\neq$ 0		
1	$\neq$ 0		

Hamming distance

$D(x_1, x_2)$	$D(x_3, x_4)$
4	
	1
	1
	0

Hamming distance == number of mismatches

Hamming distance on binary vectors:



$$\text{Hamming Distance} = 1 + 0 + 0 + 1 + 0 + 1 + 1 = 4$$

## Nearest Neighbour Search (NNS)

### Nearest Neighbour Search (NNS)

- Linear search
  - Also called linear scan, exhaustive search
  - Produce exact search results
  - Slow

### Approximate Nearest Neighbour Search (ANN)

- Return approximate search results
  - Local Sensitive Hashing (LSH)
  - Product Quantization (PQ)
  - Inverted File Index (IVF)
  - ...
- A more comprehensive list:
  - <https://github.com/erikbern/ann-benchmarks>

## Linear search

### Linear search (exhaustive search)

- Given a query vector, compute the similarity with all samples in the database, return the top-k most similar samples

Example:  
Question: given the query sample and the samples in the database below, find the top 2 nearest neighbours in the database.

Query sample:  
A [0, -1]

Database samples (5):

B [-2, 0]
C [1, 2]
D [2, 1]
E [1, -1]
F [-1, 2]

Example:  
Question: given the query sample and the samples in the database below, find the top 2 nearest neighbours in the database.

Query sample:  
A [0, -1]

Database samples (5):

B [-2, 0]
C [1, 2]
D [2, 1]
E [1, -1]
F [-1, 2]

Squared L2 distance:  $\|x - y\|^2 = \sum_{i=1}^d (x_i - y_i)^2$

B (-2, 0)	C (1, 2)	D (2, 1)	E (1, -1)	F (-1, 2)
5	10	8	1	10

Search result: the top 2 similar items are E and B.  
We need 5 distance calculations (we have 5 items in the database)

### Linear search (exhaustive search)

- Given a query vector, compute the similarity with all samples in the database, return the top-k most similar samples

For one query, search complexity:  $O(n)$ , n: the number of samples in the database

- Not efficient for large databases and high dimensional data
- Imagine a dataset containing millions or even billions of samples
- Generally, one similarity calculation for high dimensional data is expensive

## Outline

- Nearest Neighbour Search (NNS) Problems
- Local Sensitive Hashing (LSH) – Random Projection
- Product Quantization (PQ)
- Inverted File Index (non-examinable!!)

## LSH

- Local Sensitive Hashing – Random Projection
  - LSH hash functions
  - LSH + linear search for top-k retrieval (Fast)
  - LSH + hash tables for top-k retrieval

Extended discussion

Geometry view of LSH (non-examinable!)

← see this related to approximate

# Locality Sensitive Hashing (LSH)

- LSH is a family of hashing methods

- LSH-Random Projection
  - Random hyperplane hashing functions
  - Preserve cosine similarity

- Many other LSH methods
  - use different types of hashing function
  - preserve different types of similarity
  - E.g., LSH-MinHashing

## LSH-Random Projection

### LSH Random projection method

- Generate K hashing functions.
  - Randomly generate K vectors
    - (using standard gaussian distribution)
  - Each vector constructs one linear hash function.
- One hashing function transform the input vector into one binary value (1-bit)
- K hashing functions transform the input vector into a K-bit binary vector (hash vector, binary code).

One hashing function:  $h(x) = \begin{cases} 1 & : w^T x > 0 \\ 0 & : w^T x \leq 0 \end{cases}$

$x$ : the input vector

$w$ : the parameters for the hashing function  
(the randomly generated vector)

### Example (LSH + linear search):

Question: given the query sample and the samples in the database below, find the top 2 nearest neighbours in the database. Use LSH-random projection.

Query sample:	Database samples (5):
A [0, -1]	B [-2, 0] C [1, 2] D [2, 1] E [1, -1] F [-1, 2]

4 hashing functions are given:

$$\begin{aligned} w_1 &= [-1, 1]^T; \\ w_2 &= [-1, 0]^T; \\ w_3 &= [0, 1]^T; \\ w_4 &= [1, -1]^T; \end{aligned}$$

Solution:

Step1: convert the database items and the query item into binary vectors.

Query sample:	Database samples (5):
A [0, -1]	B [-2, 0] C [1, 2] D [2, 1] E [1, -1] F [-1, 2]

4 hashing functions are given:

$$\begin{aligned} w_1 &= [-1, 1]^T; \\ w_2 &= [-1, 0]^T; \\ w_3 &= [0, 1]^T; \\ w_4 &= [1, -1]^T; \end{aligned}$$

$$\begin{aligned} h_1: w_1^T x_A &= -1 * 0 + 1 * (-1) = -1 < 0 \rightarrow h_1(x_A) = 0 \\ h_2: w_2^T x_A &= -1 * 0 + 0 * (-1) = 0 < 0 \rightarrow h_2(x_A) = 0 \\ h_3: w_3^T x_A &= 0 * 0 + 1 * (-1) = -1 < 0 \rightarrow h_3(x_A) = 0 \\ h_4: w_4^T x_A &= 1 * 0 + (-1) * (-1) = 1 > 0 \rightarrow h_4(x_A) = 1 \end{aligned}$$

The binary vector for the query sample A is [0 0 0 1]

4 hashing functions are given:	Database samples (5):
$w_1 = [-1, 1]^T$	B [-2, 0] C [1, 2] D [2, 1] E [1, -1] F [-1, 2]
$w_2 = [-1, 0]^T$	
$w_3 = [0, 1]^T$	
$w_4 = [1, -1]^T$	

$$\begin{aligned} h_1: w_1^T x_B &= -1 * (-2) + 1 * 0 = 2 > 0 \rightarrow h_1(x_B) = 1 \\ h_2: w_2^T x_B &= -1 * (-2) + 0 * 0 = 2 > 0 \rightarrow h_2(x_B) = 1 \\ h_3: w_3^T x_B &= 0 * (-2) + 1 * 0 = 0 < 0 \rightarrow h_3(x_B) = 0 \\ h_4: w_4^T x_B &= 1 * (-2) + (-1) * 0 = -2 < 0 \rightarrow h_4(x_B) = 0 \end{aligned}$$

The binary vector for the database sample B is [1 1 0 0]

4 hashing functions are given:

$$\begin{aligned} w_1 &= [-1, 1]^T; \\ w_2 &= [-1, 0]^T; \\ w_3 &= [0, 1]^T; \\ w_4 &= [1, -1]^T; \end{aligned}$$

$$\begin{aligned} w_1^T x_C &= -1 * 1 + 1 * 1 = 1 > 0 \rightarrow h_1(x_C) = 1 \\ w_2^T x_C &= -1 * 1 + 0 * 2 = -1 < 0 \rightarrow h_2(x_C) = 0 \\ w_3^T x_C &= 0 * 1 + 1 * 2 = 2 > 0 \rightarrow h_3(x_C) = 1 \\ w_4^T x_C &= 1 * 1 + (-1) * 2 = -1 < 0 \rightarrow h_4(x_C) = 0 \end{aligned}$$

The binary vector for the database sample C is [1 0 1 0]

$$\begin{aligned} w_1^T x_D &= -1 * 2 + 1 * 1 = -1 < 0 \rightarrow h_1(x_D) = 0 \\ w_2^T x_D &= -1 * 2 + 0 * 1 = -2 < 0 \rightarrow h_2(x_D) = 0 \\ w_3^T x_D &= 0 * 2 + 1 * 1 = 1 > 0 \rightarrow h_3(x_D) = 1 \\ w_4^T x_D &= 1 * 2 + (-1) * 1 = 1 > 0 \rightarrow h_4(x_D) = 1 \end{aligned}$$

The binary vector for the database sample D is [0 0 1 1]

$$\begin{aligned} 4 \text{ hashing functions are given:} \\ w_1 &= [-1, 1]^T; \\ w_2 &= [-1, 0]^T; \\ w_3 &= [0, 1]^T; \\ w_4 &= [1, -1]^T; \end{aligned}$$

$$\begin{aligned} w_1^T x_E &= -1 * 1 + 1 * (-1) = -2 < 0 \rightarrow h_1(x_E) = 0 \\ w_2^T x_E &= -1 * 1 + 0 * (-1) = -1 < 0 \rightarrow h_2(x_E) = 0 \\ w_3^T x_E &= 0 * 1 + 1 * (-1) = -1 < 0 \rightarrow h_3(x_E) = 0 \\ w_4^T x_E &= 1 * 1 + (-1) * (-1) = 2 > 0 \rightarrow h_4(x_E) = 1 \end{aligned}$$

The binary vector for the database sample E is [0 0 0 1]

$$\begin{aligned} w_1^T x_F &= -1 * (-1) + 1 * 2 = 3 > 0 \rightarrow h_1(x_F) = 1 \\ w_2^T x_F &= -1 * (-1) + 0 * 2 = 1 > 0 \rightarrow h_2(x_F) = 1 \\ w_3^T x_F &= 0 * (-1) + 1 * 2 = 2 > 0 \rightarrow h_3(x_F) = 1 \\ w_4^T x_F &= 1 * (-1) + (-1) * 2 = -3 < 0 \rightarrow h_4(x_F) = 0 \end{aligned}$$

The binary vector for the database sample F is [1 1 1 0]

Query sample:  
A [0, -1]

Query sample:  
A [0 0 0 1]

Database samples (5):

B [-2, 0]
C [1, 2]
D [2, 1]
E [1, -1]
F [-1, 2]

After hashing functions

B [1 1 0 0]
C [1 0 1 0]
D [0 0 1 1]
E [0 0 0 1]
F [1 1 1 0]

Query sample:  
A [0 0 0 1]

Database samples (5):

B [1 1 0 0]
C [1 0 1 0]
D [0 0 1 1]
E [0 0 0 1]
F [1 1 1 0]

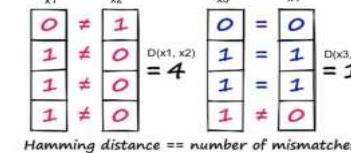
Step2: perform linear search on binary vectors using hamming distance  $D_h$ .

Search result:  
Given the query A,  
the top 2 nearest neighbours are: E and D

## Similarity (recap)

### 4. Hamming distance:

- Generally, counts the number of positions where two symbols are different.
- For binary data: the number of bits that are different in two binary vectors.



### L2 distance based linear search VS LSH + linear search

#### L2 distances

	B [-2, 0]	C [1, 2]	D [2, 1]	E [1, -1]	F [-1, 2]
A [0, -1]	5	10	8	1	10

Search result: the top 2 similar items are E and B

#### LSH and Hamming distances

	B	C	D	E	F
A	3	3	1	0	4

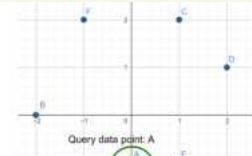
Search result: the top 2 similar items are E and D

Database samples (5):

B [-2, 0]
C [1, 2]
D [2, 1]
E [1, -1]
F [-1, 2]

Data points:

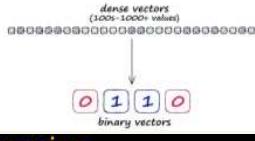
A (0, -1)
B (-2, 0)
C (1, 2)
D (2, 1)
E (1, -1)



## LSH for similarity search

### With LSH, we can

- generate low-dimensional binary features from high-dimensional input
- Preserve the similarity after binary mapping.
- If  $x$  is similar to  $y$  in the original space, after binary mapping, their binary vectors are also similar in the binary space.



## Discussion

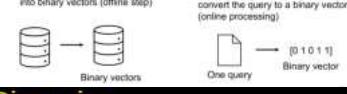
### Why LSH + linear search is more efficient than L2 distance + linear search?

- We can use the efficient hamming distance on the binary vectors.

- Hamming distance calculation is very efficient
  - XOR operation on the bits (counting mismatched bits)
  - Much more efficient than L2 distance on float values
  - Especially useful for high-dimensional data

### LSH requires extra calculation for binary transform

- Database binary transform with LSH is done offline.
  - The whole database are transformed to binary vectors for only one time (offline pre-processing)
  - The database binary vectors will be used for all queries.
  - For one query, we only need to transform the query input vector to binary vector.



Binary vectors

One query

Binary vector

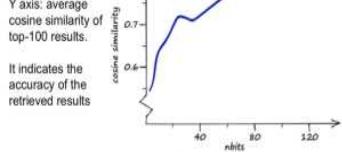
### Benefits of using binary vectors

- Binary vectors are efficient for storage, transmission and similarity calculation (Hamming distance).
  - A 128-dimensional float-value vector:  $128 \times 4 \times 4 = 512$  bytes
    - 1 float value requires 4 bytes (32 bits)
  - A 128 bit binary vector = 16 bytes
    - 1 byte = 8 bits,
- More hashing functions will increase the accuracy on top-k nearest neighbour search.
  - More hashing functions lead to longer binary vectors (we have more bits)
- With more bits, the top-k ranking results of LSH will be more similar to the cosine distance or L2 distance based top-k results.

## Analysis on the number of bits

Number of bits (nbits): the number of binary values in the binary vector. It is equal to the number of hash functions.

With more bits, the top-k retrieval results become more accurate.



It indicates the accuracy of the retrieved results

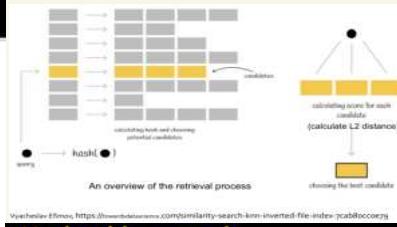
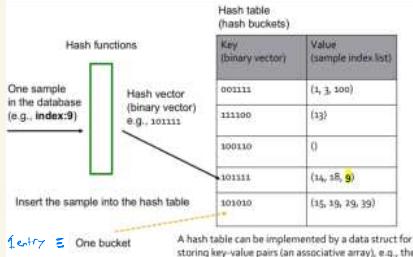
As one increases vector resolution with nbits, the results will become more precise — here, we can see that a larger nbits value results in higher cosine similarity in our results.

Sift3M dataset, 1M samples, 128 dimensions

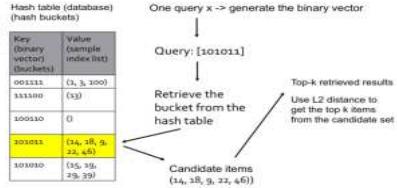
Facebook Faiss implementation, top 100 retrieved

<https://www.pinecone.io/learn/locality-sensitive-hashing-random-projection/>

## Create hash table



## Hash table example



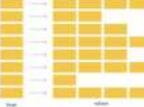
## More hashing functions

- We can modify the LSH example to generate more hashing functions and observe the difference.

Query sample:	Database samples (5):
A [0, -1]	B [-2, 0]
Hashing functions:	C [1, 2]
w <sub>1</sub> = [1 1 1];	D [2, 1]
w <sub>2</sub> = [-1 0 1];	E [1, -1]
w <sub>3</sub> = [0 1 1];	F [-1, 2]
Add more hash functions:	
w <sub>4</sub> = [1 0 1];	
w <sub>5</sub> = [-1 -1 1];	

We will explore this question in the tutorial class

A hash table aims to group similar data into the same bucket: similar data are likely to have the same key (binary vector)



- Each entry is a key-value pair: <h, sample\_idx\_list>
  - h is a hash vector, sample\_idx\_list is a list of the indexes of the data samples in the database
- If two hash vectors (hash codes) are the same, they will be mapped to the same bucket in the hash table.
  - The entries in the same bucket will have the same hash vector (hash code).

Two examples in the database have the same hash code (binary vector) → Two examples are mapped to the same bucket in a hash table

### Method 1: exhaustive search of all buckets

- Step 1: Build the hash table for the database (done)
- Step 2: process a query
  - Compute the hash vector for the query sample.
    - E.g. input query x → [101010]
  - Construct a candidate set for the retrieval
    - Compare the query hash vector with all buckets in the hash table using Hamming distance
    - Rank all buckets by hamming distance
    - Go through the top-ranked buckets to build a candidate set (the number of candidate items >> k items for top-k retrieval).
  - Retrieved the candidate items which have the same binary vector as the query.
  - Return top-k items from the candidate set using L2 distance.

## Using hash tables for LSH

Non examinable

### 2. Similarity search using hash tables

#### Method1: Using one hash table

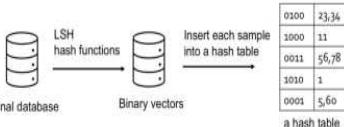
#### Method2: Using multiple hash tables

## Using hash tables for LSH

### Method 1: using one hash table

#### Step 1: build the hash table for the database

- Generate the hash vector (h) for each data sample in the database and insert <h, sample\_idx> into a hash table.



Task: top-5 retrieval, given the query x

Key (binary vector) (buckets)	Value (sample index list)
000111	(1, 3, 200)
111100	(13)
100110	()
101111	(14, 18, 9)
101010	(15, 19, 29, 39)

Hash table (database)  
(hash buckets)

Generate the binary vector of  
the query x  
Query x: [101011]

#### More hashing functions

- => leads to longer binary vectors
  - => more bits
  - => more buckets
- #bits: 2: maximum #buckets:  $2^2 = 4$   
#bits: 4: maximum #buckets:  $2^4 = 16$   
#bits: 8: maximum #buckets:  $2^8 = 256$   
#bits: 16: maximum #buckets:  $2^{16} = 65536$

#### Efficiency of using LSH + hash table: (compared to linear search)

- 1. the number of distance calculation is greatly reduced
  - Only calculate the L2 distance in the candidate set (the retrieved bucket)
  - Do not need to perform a linear scan to calculate distance for every item in the database.

#### 2. The binary vector generation is efficient

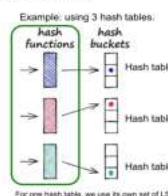
- In the query stage, only need to map the input query into a binary vector by applying hash functions

#### Method 2: using multiple hash tables

##### 1. Construct multiple hash tables

##### 2. Different tables use different sets of hash functions.

Generate M different sets of LSH hash functions. One set of hash functions is for one hash table.



#### Query process:

1. Use the query hash vector to retrieve one bucket in each hash table.
2. Combine the retrieved results from all hash tables as candidate items
3. Perform accurate linear search on the candidate items to output top-k results

Example: using 3 hash tables. We use different hash functions for each table.

#### Hash table 1

#### Hash table 2

#### Hash table 3

Key	Value (sample_index)
001111	(1, 3, 200)
111100	(13)
100110	()
101111	(14, 18, 9)
101010	(15, 19, 29, 39)

Key	Value (sample_index)
001111	(1, 3, 200)
111100	(13, 2, 5)
100110	()
101111	(14, 18, 9)
101010	(15, 19)

Key	Value (sample_index)
001111	(1, 3, 200)
111100	(9, 21)
100110	()
101111	(9, 10)
101010	(40)

Query X->[111100]

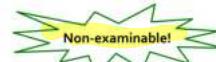
Query X->[101010]

Query X->[101010]

1. For one query X, we will generate 3 hash vectors.
2. Retrieved one bucket from each hash table.
3. Combined the retrieved buckets from each table: candidate set: (13, 2, 5, 40)
4. Linear search on the candidate set using L2 distance to return the top-k results

#### Extended discussion (non-examinable!)

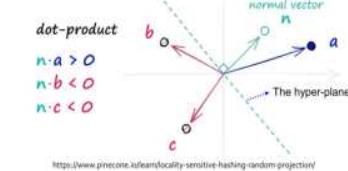
- Geometry view of LSH-random projection



## Geometry view of LSH

n: a randomly generated vector, represents one hashing function

1. Given the vector n, we can find a hyper-plane that
  - a) passes through the origin and,
  - b) the vector n is orthogonal to the plane (n is a normal vector to the hyper-plane).
2. The hyper-plane to partition the space into positive and negative regions



## Similarity (recap)

#### Inner product (dot product):

The dot product of two vectors  $\mathbf{a} = [a_1, a_2, \dots, a_n]$  and  $\mathbf{b} = [b_1, b_2, \dots, b_n]$  is defined as  $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$ .

Geometric definition:

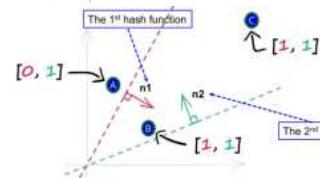
$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| |\mathbf{b}| \cos \theta, \text{ where } \theta \text{ is the angle between a and b.}$$

$-90^\circ < \theta < 90^\circ \rightarrow \cos(\theta) > 0$



The output of the hashing function is determined by  $\cos(\theta)$ :

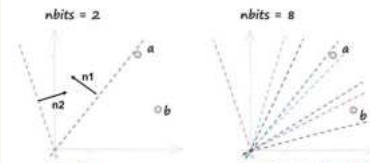
$$h(\mathbf{x}) = \begin{cases} 1 & : \mathbf{w}^T \mathbf{x} > 0 \\ 0 & : \mathbf{w}^T \mathbf{x} \leq 0 \end{cases}$$



Indicates whether the data points lies in the same side of the hyper-plane or not.

If they lie in the same side -> they are similar.

Otherwise, they are not similar.



## Online resources

#### FAISS

- FAISS is a library for efficient similarity search and clustering of dense vectors.

<https://github.com/facebookresearch/faiss/wiki>

# Similarity Search - PQ

## Outline

- Nearest Neighbour Search (NNS) Problems
- Local Sensitive Hashing (LSH) – Random Projection
- Product Quantization (PQ)**
- Inverted File Index (non-examinable!!)

## Product Quantization (PQ)

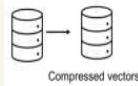
- Product Quantization (PQ)**
- A clustering based method
- Compress the data
- Speed up searching by using precomputed distances

Most materials are from  
<https://www.pinecone.io/learn/product-quantization/>

Paper: Product quantization for nearest neighbor search, 2011

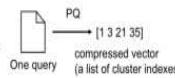
### Overview of PQ for similarity search

Step1: convert the database into compressed vectors (offline step)



Step 2: process query requests  
 Use efficient approximate distance (symmetric or asymmetric) calculation to get top-k retrieved results

When using symmetric distance:  
 For each query, we need to convert the query to a compressed vector



## The process of PQ for compression

### Pre-processing:

- Define the number of subspaces (subvectors).
- For each subspace, generate k cluster centroids using the samples in the database.

### Generating compressed vectors using PQ

Input: the input feature vector (high-dimensional)

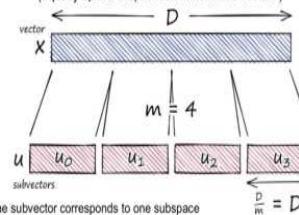
#### Steps:

- Equally split the input into M sub-vectors.  
 Each sub-vector corresponds to one subspace.
- Assigning each of these sub-vectors to its nearest cluster centroid (also called reproduction/reconstruction values)
- Use the assigned cluster IDs (indexes) of all sub-vectors to construct the compressed vector (or called quantized vector).

## Example

We define M=4 subspaces.

- We generate 4 sub-vectors for each input.
- (equally split the input vector into M sub-vectors)

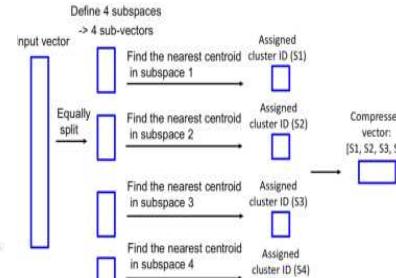


One subvector corresponds to one subspace

E.g., if the input vector dimension  $D=100$ , the length of each sub-vector is  $100/4 = 25$

Sub-vector generation example

## Compressed vector



## Example

### Product Quantization example

Define 2 Subspaces  
 Each subspace has 4 centroids.

Input vectors:  
 A: [1.82, 5.08, 2.21, 4.21]  
 B: [4.96, 4.46, 4.1, 1.3]

Subspace 1, We have 4 centroids: C1 to C4.

A1, B1: the sub-vectors for datapoint A and B, respectively

A1 = (1.82, 5.08)

B1 = (4.96, 4.46)

C1 = (1.8, 4.2)

C2 = (5.08, 5.16)

C3 = (3.24, 2.2)

C4 = (6.4, 3.06)

Subspace 2, We have 4 centroids: C1 to C4.

A2, B2: the sub-vectors for datapoint A and B, respectively

A2 = (2.01, 4.21)

B2 = (4.1, 1.3)

C1 = (1.9, 1.3)

C2 = (2.02, 3.3)

C3 = (3.92, 1.77)

C4 = (3.87, 3.98)

## Example

• Q: generate the compressed vector for datapoint A and B

Answer:

In subspace 1, A is assigned to the centroid C1  
 (1st dimension of the compressed vector)

In subspace 2, A is assigned to the centroid C2  
 (2nd dimension of the compressed vector)

→ A: [1, 2]

In subspace 1, B is assigned to the centroid C2

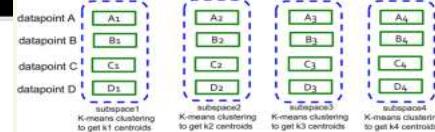
In subspace 2, B is assigned to the centroid C3

→ B: [2, 3]

### How to generate cluster centroids?

- Generate sub-vectors for all data examples in the database
- Define M subspaces → Generate M sub-vectors for one input.
- Perform K-means in each subspace to generate k centroids.
- For example, we can generate k=256 centroids for each sub space

Use datapoints in the database to generate centroids for each subspace



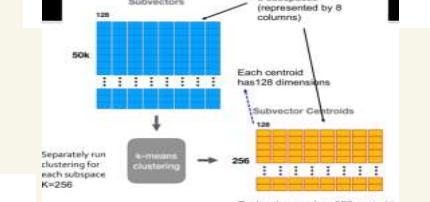
## Example: Subspace clustering

Database: 50K samples, each has 1024 dimensions: 1024 × 8 × 8



Each row indicates one data sample.

Define 8 subspaces. Each vector is splitted into 8 sub-vectors, each of length 128 (8 sub vectors × 128 dimensions = 1,024 dimensions).



### Compression analysis

3 float value requires a bytes (4×8=32 bits);  
 1 integer value requires 1 bytes (here we use 8-bit integer, value range: 0~255)

original vector: 1024 float → 3024 × (4 bytes)=4096 bytes

8 sub-vectors → 8 sub-vectors × 128 float → 8 × 128 × 4 bytes = 4096 bytes

Subvector Centroids → 8 sub-vectors × 128 integer → 8 × 128 × 1 bytes = 1024 bytes

The final compression is 532 times!

One centroid ID is an integer value. In this example, we only have 256 centroids, so we can use one 8-bit integer ( $2^{8-1}$ ) to store one cluster ID.

Credit: Vyachaslav Efremov, <https://charmeddatascience.com/similarity-search-product-quantization-tutorial/>

# Product Quantization

- Codebook based compression
  - the codebook is the centroids
  - Separate codebook (centroids) for each subspace
- Compress high-dimensional vector to a tiny vector of IDs that only requires very little memory/storage space.
- Product Quantization (PQ)

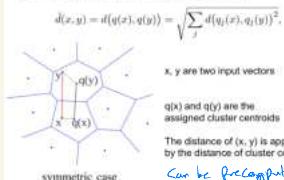
- A data dependent method
  - Need to use the data in the database to determine the parameters (cluster centres) of the algorithm.
- LSH is a data independent method
  - The hash functions are randomly generated, not using the data in the database.

## PQ for similarity search

- PQ for Similarity Search
  - Use liner search (exhaustive search)
  - Use approximate distance to speed up distance calculation
- Calculating approximate distance
  - Symmetric distance
    - Use pre-computed distance look-up table to speed up the calculations.
  - Asymmetric distance

## Symmetric distance

### Approximate distance: Symmetric distance



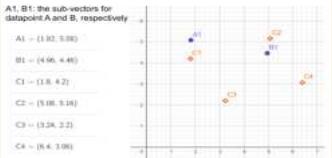
## Recap

### Overview of PQ for similarity search

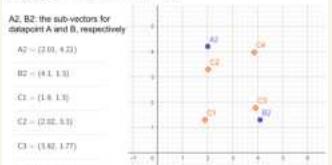
- Step1: convert the database into compressed vectors (offline step)
- Step 2: process query requests  
Use efficient approximate distance (metric space) pre-computation to calculate to get top-k retrieved results
- When using symmetric distance:  
For each query, we need to convert the query to a compressed vector
- Compressed vectors
- When using symmetric distance:  
Precompute the centroid distances tables for each subspace
- PQ  
One query [1, 3, 2, 3]  
compressed vector (a list of cluster indexes)

## Symmetric distance example

### Subspace 1, We have 4 centroids: C1 to C4.



### Subspace 2, We have 4 centroids: C1 to C4.



## Symmetric distance example

Define 2 subspaces. The centroids and datapoints are given in the above slides.

### Question:

- Construct centroid distance look up tables and
- compute the approximate Squared L2 distance (symmetric case) between A and B.

### Solution to Q1: Subspace 1

	Centroid 1	Centroid 2	Centroid 3	Centroid 4
Centroid 1	0	11.68	6.07	22.46
Centroid 2	11.68	0	12.15	6.35
Centroid 3	6.07	12.15	0	10.73
Centroid 4	22.46	6.35	10.73	0

Table 1: squared L2 distance table for Subspace 1 ( $D_1$ )

Distance calculation example:  
 $C1=[1.8, 4.2], C2=[5.08, 5.16]$ ,  
 $D_1(C1, C2) = (1.8-5.08)^2 + (4.2-5.16)^2 = 10.76 + 0.92 = 11.68$

Subspace 1, We have 4 centroids: C1 to C4.

A1, B1: the sub-vectors for datapoint A and B, respectively

A1 = (1.82, 5.08)

B1 = (4.96, 4.46)

C1 = (1.8, 4.2)

C2 = (5.08, 5.18)

C3 = (3.24, 2.2)

C4 = (6.4, 3.06)

### Subspace 2

	Centroid 1	Centroid 2	Centroid 3	Centroid 4
Centroid 1	0	4.03	4.30	11.06
Centroid 2	4.03	0	5.95	3.88
Centroid 3	4.30	5.95	0	4.89
Centroid 4	11.06	3.88	4.89	0

Table 2: squared L2 distance table for Subspace 2 ( $D_2$ )

Subspace 2, We have 4 centroids: C1 to C4.

A2, B2: the sub-vectors for datapoint A and B, respectively

A2 = (2.01, 4.21)

B2 = (4.1, 1.3)

C1 = (1.8, 1.3)

C2 = (2.02, 3.3)

C3 = (3.92, 1.77)

C4 = (3.87, 3.98)

### Question:

- compute the approximate Squared L2 distance (symmetric case) between A and B.

### Solution to Q2:

Compressed vectors: A: [1, 2]; B: [2, 3]

Look up the distance values in table D1 and D2 using the cluster IDs:

$$D(A_1, B_1) = D_1(\text{Centroid 1}, \text{Centroid 2}) = 11.68$$

$$D(A_2, B_2) = D_2(\text{Centroid 2}, \text{Centroid 3}) = 5.95$$

$$D(A, B) \rightarrow D(A_1, B_1) + D(A_2, B_2) = 11.68 + 5.95 = 17.63$$

### Subspace 1

	Centroid 1	Centroid 2	Centroid 3	Centroid 4
Centroid 1	0	11.68	6.07	22.46
Centroid 2	11.68	0	12.15	6.35
Centroid 3	6.07	12.15	0	10.73
Centroid 4	22.46	6.35	10.73	0

### Subspace 2

	Centroid 1	Centroid 2	Centroid 3	Centroid 4
Centroid 1	0	4.03	4.30	11.06
Centroid 2	4.03	0	5.95	3.88
Centroid 3	4.30	5.95	0	4.89
Centroid 4	11.06	3.88	4.89	0

Compressed vectors: A: [1, 2]; B: [2, 3]

$$D(A_1, B_1) = D_1(\text{Centroid 1}, \text{Centroid 2}) = 11.68$$

$$D(A_2, B_2) = D_2(\text{Centroid 2}, \text{Centroid 3}) = 5.95$$

$$D(A, B) \rightarrow D(A_1, B_1) + D(A_2, B_2) = 11.68 + 5.95 = 17.63$$

## Symmetric distance

### Approximate distance calculation

- we can calculate the distances much more efficiently using just table look-ups and some addition.

### Using centroid pre-computed distances.

Pre-compute and the distances between centroids for each subspace, stored the distances in a matrix or lookup table, use the centroid index pair to retrieve the partial distances for each subspace.

### Symmetric distance

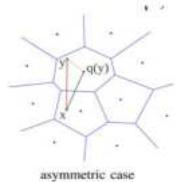
#### Asymmetric distance

More accurate for distance calculation, as we only quantize one input.  
(compared to symmetric distance)

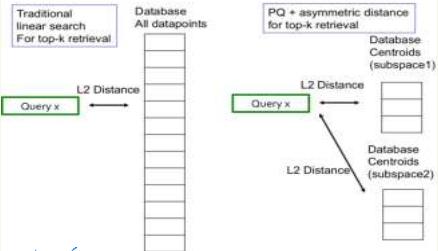
$$d(x, y) = d(x, q(y))$$

x: a query example,  
y: an example in the database.

Use q(y) to replace y for distance calculation



## Example for asymmetric distance



$$\begin{matrix} S_1 & S_2 \\ A & 1 \quad 2 \\ B & 2 \quad 3 \end{matrix}$$

# Example (asymmetric distance)

Data points in the database:

A	[4, 6, 14, 4]
B	[6, 2, 8, 2]
D	[6, 8, 10, 2]
E	[4, 4, 6, 4]

Query data point:

Q	[16, 4, 4, 8]
---	---------------

Question: Calculate the approximate Squared L2 distance using Asymmetric distance between the query Q and all data points in the database (A, B, D, E)

Solution:

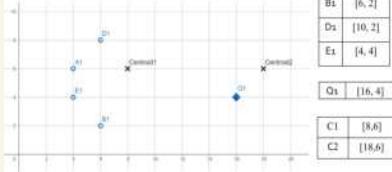
Step1: Compress the data points in the database  
(In each subspace, identify the nearest centroid for each data point)

Step 2: Calculate Asymmetric distances

Subspace 1:  
(A1, B1, D1, E1, Q1 are sub-vectors in Subspace1)

A1	[4, 6]
B1	[6, 2]
D1	[10, 2]
E1	[4, 4]
Q1	[16, 4]

C1	[8, 6]
C2	[18, 6]



Calculate the squared L2 distances between the data points and each centroid to identify the nearest centroid

Subspace 2:  
(A2, B2, D2, E2, Q2 are sub-vectors in Subspace2)

A2	[14, 4]
B2	[8, 2]
D2	[10, 2]
E2	[6, 4]
Q2	[4, 8]

C1	[10, 12]
C2	[10, 4]



Calculate the squared L2 distances between the data points and each centroid to identify the nearest centroid

Step1: Compress the data points in the database  
(In each subspace, identify the nearest centroid for each data point)

compression results:

A -> [1, 2]
B -> [1, 2]
D -> [1, 2]
E -> [1, 2]

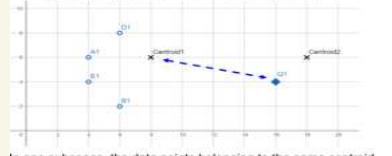
Step2: calculate Asymmetric distances

1) Generally, for the query Q, we need to calculate the distances between Q and all the centroids in each subspace.

2) For this question, as all data points are compressed as the same vector [1, 2], we only need to calculate the following distances:

Subspace 1: D (Q1, Centroid 1)  
Subspace 2: D (Q2, Centroid 2)

Subspace 1:  
(A1, B1, D1, E1, Q1 are sub-vectors in Subspace1)



In one subspace, the data points belonging to the same centroid will have equal approximate distances to the query.

$$D(A1, Q1) = D(B1, Q1) = D(D1, Q1) = D(E1, Q1)$$

Subspace 2:  
(A2, B2, D2, E2, Q2 are sub-vectors in Subspace2)



In one subspace, the data points belonging to the same centroid will have equal approximate distances to the query.

$$D(A2, Q2) = D(B2, Q2) = D(D2, Q2) = D(E2, Q2)$$

### Compression results

A -> [1, 2]
B -> [1, 2]
D -> [1, 2]
E -> [1, 2]
Q -> [1, 2]

As they are all compressed to the same vector, we have Approximate Squared L2 Distances (Asymmetric):

$$D(Q, A) = D(Q, B) = D(Q, D) = D(Q, E)$$

Subspace 1:	C1	[8, 6]
	C2	[18, 6]

Subspace 2:	C1	[10, 12]
	C2	[10, 4]

Subspace 1:  
 $D_1(Q_1, \text{Centroid 1}) = 8^2 + 2^2 = 68$

Subspace 2:  
 $D_2(Q_2, \text{Centroid 2}) = 6^2 + 4^2 = 52$

Approximate Squared L2 Distances (Asymmetric):

$$D(Q, A) = D(Q, B) = D(Q, D) = D(Q, E) = D_1 + D_2 = 120$$

### Asymmetric distance calculation is efficient

#### General case:

For one subspace, we only need to calculate K approximate distances. K is the number of centroids in one subspace.

K is much less than the total number of data points in the database. E.g., K=128 centroids, data points: 10 million

# Asymmetric distance

- Given a query in a top-k retrieval task, we don't need to compute the distance between the query and all the samples in the database.

We only need to calculate the distance between the query and all the centroids in each subspace.

- The number of centroids are much less than the number of samples in the database, so we can speed up the retrieval process.

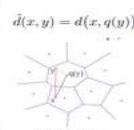
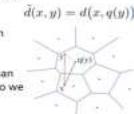
- We don't need to quantize x, which reduces computation compared to the symmetric case.

For each subspace, we compute the squared L2 distances between the x and all centroids.

This bring extra computation compared to the symmetric case.

- The above extra computation cost is similar with the quantization cost for x. (cluster assignment requires distance calculation between x and all centroids)

That means symmetric and asymmetric distance have the same computation complexity.



### Comparison

$$d(x, y) = d(x, q(y)) = \sqrt{\sum_i (x_i - q(y)_i)^2}$$

$$\hat{d}(x, y) = d(x, q(y))$$

Compressed vectors:

A -> [1, 2]
B -> [1, 2]
D -> [1, 2]
E -> [1, 2]

Subspace 1:	C1	[8, 6]
	C2	[18, 6]

Subspace 2:	C1	[10, 12]
	C2	[10, 4]

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

asymmetric case
-----------------

symmetric case
----------------

## Outline

- Nearest Neighbour Search (NNS) Problems
- Local Sensitive Hashing (LSH) – Random Projection
- Product Quantization (PQ)
- Inverted File Index (non-examinable!!)

### Inverted File Index



- Inverted File Index
  - Clustering based method
  - Data dependent
  - Reduce the search scope



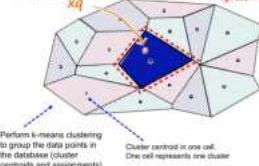
### Inverted File Index (IVT)

- An inverted index here means a mapping (a lookup table) from cluster centroids (words) to the cluster members.

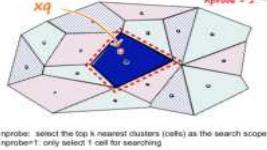
#### Steps:

1. Clustering of the samples in the database, generate the centroids and cells
2. Given a query vector, identify the cells for search scope
3. Use linear search (exhaustive search) to retrieve results within the selected cells

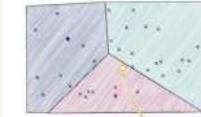
$xq$ : the query vector  
search scope:  $nprobe = 1$



$xq$ : the query vector  
search scope:  $nprobe = 1$



There are 2 parameters in IVT:  
1. nprobe: the number of cells to search  
2. The number of cells (clusters) to create.

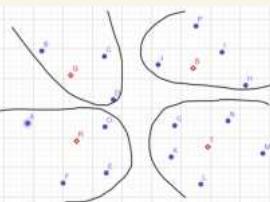


Our query vector  $xq$  falls on the edge of the response cell. During testing phase if  $xq$  falls in the red cell, we will not compare these. If  $nprobe = 1$  – in this reason we will never search scope in the response cell only.



#### Inverted File Index Example

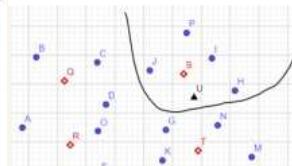
- Q(a): assign the samples to the cells defined by the centroids.



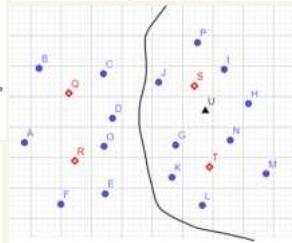
one data point is assigned to its nearest centroid based on L2 distance



Q(b): Given the query sample  $U$ , list the candidate samples in the database for similar search when nProbe=1 and nProbe=2, respectively



nProbe=1: only search in 1 cell.

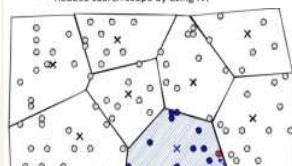


nProbe=2: search in 2 cells (the top 2 closest cells)

### PQ+IVF



- PQ+IVF
  - Product Quantization + Inverted File Index
  - A hierarchical solution
  - Much better performance in searching
  - Reduce search scope by using IVF



Step1: use IVF to identify search scope (cells). IVF allows us to restrict our search in the target cells. PQ generates compressed vectors and uses fast distance calculation

Step2: perform PQ based linear search in the target cells. PQ generates compressed vectors and uses fast distance calculation

Experiment:  
Sift1M dataset, 2048 centroids for each subspace

L2 distance base linear search

	FlatL2	PQ	IVFPQ
Recall (%)	100	50	52
Speed (ms)	8.26	1.49	0.09
Memory (MB)	256	6.5	9.2

<https://www.pinecone.io/learn/product-quantization/>

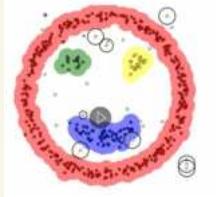
## Online resources

### FAISS

- Faiss is a library for efficient similarity search and clustering of dense vectors.

▪ <https://github.com/facebookresearch/faiss/wiki>





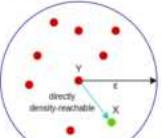
## Discussion:

Construct clusters in DBSCAN:

- A cluster is constructed by merging reachable core points and their border points.
- A cluster consists of core points that are reachable from one another and all the border points of these core points.
- The requirement to form a cluster is to have at least one core point.

<https://www.datasciencecentral.com/dbscan-algorithm-complete-guide-and-application-with-python-wikibook-dbscan.pdf>

"directly reachable" is also called "directly density-reachable".  
1. A point X is directly density-reachable (or directly reachable) from point Y w.r.t. epsilon, minPoints if,  
1) X belongs to the epsilon-neighborhood of Y, i.e. dist(X, Y) <= epsilon  
It can be a border point or core point  
2) Y is a core point



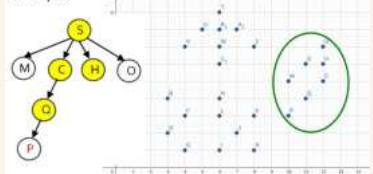
<https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/>

2. A point X is density-reachable (or reachable) from point Y w.r.t. epsilon, minPoints if there is a chain of points  $p_0, p_1, p_2, \dots, p_n = X$  such that  $p_{i-1}$  is directly density-reachable from  $p_i$ .



If X is reachable from Y (source point)  
we can find a path connecting points s and u, where each point in the path is directly reachable from the previous one. (The path is constructed by "directly reachable" core points.)

## Example:



A tree to describe the process of cluster growth in DBSCAN

A parent node: represents a core point

A child node: a neighbour of the parent node

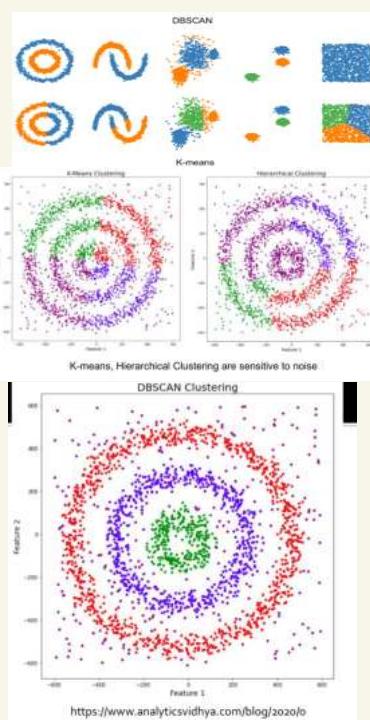
A connection from a parent node to its child indicates directly reachable  
One point is reachable from any core points in the cluster (can find a path in the tree)

## Discussion

- DBSCAN**
  - All core points are equally important to determine the shape of one cluster, so it can work for clusters with arbitrary shapes
- K-Means:**
  - the centroid is important.
  - the shape of the cluster is determined by only one point
  - only works well for clusters with spherical shapes
- DBSCAN is density-based clustering**
  - defines clusters as dense regions separated by low-density regions.

<https://github.com/NSHipster/DBSCAN>

## Extended discussion

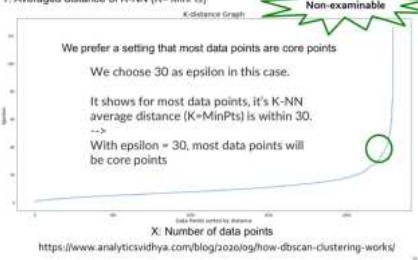


<https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/>

- Two parameters:**
  - MinPts and epsilon
  - Select a value for MinPts and then search for epsilon
- How to choose epsilon? (Given MinPts)**
  - Use K-distance graph
  - Step 1: Calculate the average distance between each point in the data set and its K nearest neighbors (set K as the MinPts value).
  - Step 2: Sort distance values by ascending value and plot the K-distance graph
  - Step 3: find the elbow point in the graph and use the corresponding distance as Epsilon

<https://medium.com/tararammulin/dbscan-parameter-estimation-f8330e3a3bd>

Y: Averaged distance of K-NN (K=MinPts)



<https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/>

We prefer a setting that most data points are core points

We choose 30 as epsilon in this case.

It shows for most data points, it's K-NN average distance (K=MinPts) is within 30.  
-->

With epsilon = 30, most data points will be core points

<https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/>

## Discussion

### Pros and Cons of DBSCAN

#### Pros:

- Does not require to specify number of clusters beforehand.
- Performs well with arbitrary shapes clusters.
- DBSCAN is robust to outliers and able to detect the outliers.

#### Cons

- It is not very effective when you have clusters of varying densities.
  - if there are different density levels, it is difficult to choose a setting of the neighbourhood distance threshold (epsilon) and MinPts that can work well for all density levels.
- If you have high dimensional data, the determining of the distance threshold  $\epsilon$  becomes a challenging task.

<https://www.digitalvidya.com/blog/the-top-5-clustering-algorithms-data-scientists-should-know/>

DBSCAN is not very effective for clusters with varying density

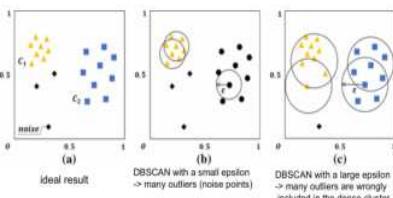


Figure credit: AA-DBSCAN: an approximate adaptive DBSCAN for finding clusters with varying densities

# Graph community detection

## Outline

- Louvain Algorithm
- Single pass

only look for connection

- Multi-pass Louvain Algorithm
- (non-examinable!)

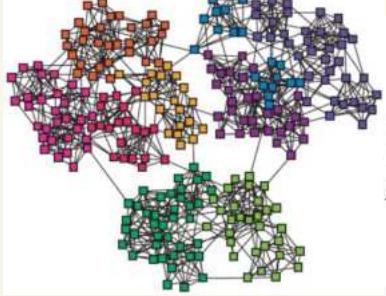
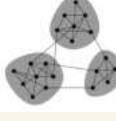
Many slides are from:  
<http://snap.stanford.edu/~mcauliffe/2010louvain-communities.pdf>

One community in a graph:

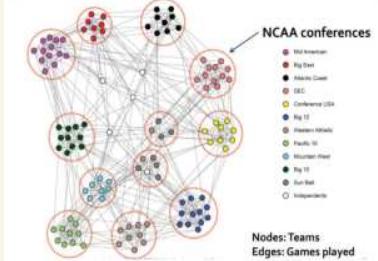
- A cluster of nodes
- a group of tightly (densely) connected nodes
- many internal connections and few external connections** (to the rest of the network)
- A community is also called: A cluster, A group, A module

Community detection

- A graph clustering task
- Automatically finds densely connected groups/clusters



## NCAA football network



## Louvain Algorithm

- Community detection on graphs
- Greedy algorithm
  - Make locally optimal decision at each step
- Supports weighted graphs
- Provides hierarchical communities
- Widely utilized to study large networks
  - Fast, rapid convergence
  - Produce high-quality results

## Modularity score

Use this equation to calculate the modularity score for a community:

$$Q(C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2$$

- Modularity score: measure the quality of a community
- We aim to maximize the total Modularity score of all communities.

$$\Sigma_{in} \equiv \sum_{i,j \in C} A_{ij}$$

: Sum of the weights of internal edges in the community C. (double count each edge) (high value indicates strong internal connections)

$$\Sigma_{tot} \equiv \sum_{i \in C} k_i$$

: Sum of the weighted degrees of all nodes in the community C

$m$  : the sum of all edge weights in the undirected graph

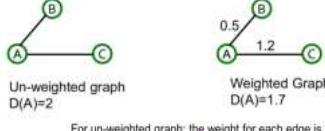
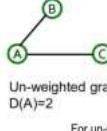
### Node degree

#### Un-weighted graph:

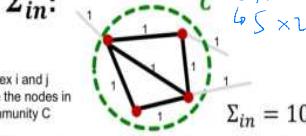
- Node degree: the number of connected edges.
- all edge weights equal to 1

#### Weighed graph:

- Node degree (weighted):  
the sum of the weights of connected edges.



For un-weighted graph: the weight for each edge is 1



$$\Sigma_{in} \equiv \sum_{i,j \in C} A_{ij}$$

: Sum of all internal edge weights of the community (each internal edge will be double counted in the summation)

$A_{i,j}$  is the edge weight for the edge connecting the nodes  $(i, j)$

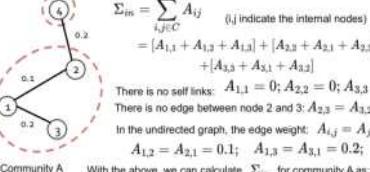
In the undirected graph, the edge weight:  $A_{i,j} = A_{j,i}$

For the example here, the internal edges will be double counted in the summation:

$$(1+1+1+1) \times 2 = 10$$

### Another example

For the community A, there are three internal nodes {1,2,3}. We calculate:  $\Sigma_{in}$  for community A:



There is no self-links:  $A_{1,1} = 0; A_{2,2} = 0; A_{3,3} = 0$

There is no edge between node 2 and node 3:  $A_{2,3} = A_{3,2} = 0$

In the undirected graph, the edge weight:  $A_{i,j} = A_{j,i}$

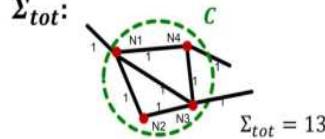
$$A_{1,2} = A_{2,1} = 0.1; A_{1,3} = A_{3,1} = 0.2;$$

With the above, we can calculate:  $\Sigma_{in}$  for community A as:

$$\Sigma_{in} = \sum_{i,j \in C} A_{ij} = (A_{1,2} + A_{1,3}) \times 2 = (0.1 + 0.2) \times 2 = 0.6$$

(sum the internal edge weights and multiply 2)

For un-weighted graph: the weight for each edge is 1



$$\Sigma_{tot} \equiv \sum_{i \in C} k_i$$

: Sum of the degrees of all internal nodes in the community C.

Here  $k_i$  is the node degree of node i.

Sum of the node degrees of all internal nodes in C:

$$4 + 2 + 4 + 3 = 13$$

(for node N1, N2, N3, N4, respectively)

## Example: an extreme case

- An extreme case: all nodes belong to 1 community

This trivial clustering strategy provides a baseline score

$$Q(C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 = \frac{2m}{2m} - \left(\frac{2m}{2m}\right)^2 = 0$$

We have:  $\Sigma_{in} = 2m$

$$\Sigma_{tot} = 2m$$

Example for the left graph:



$m$  : the sum of all edge weights in the undirected graph

$$m = 0.1 + 0.2 + 0.2 = 0.5$$

$$\Sigma_{in} = (0.1 + 0.2 + 0.2) \times 2 = 1$$

$$\Sigma_{tot} = (0.1 + 0.2) + (0.1 + 0.2) + (0.1 + 0.2) = 1$$

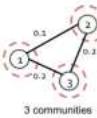
$$Q(C) = 0$$

- Another extreme case: one node forms one community

$$Q(C_i) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 = \frac{0}{2m} - \left(\frac{k_i}{2m}\right)^2 = -\left(\frac{k_i}{2m}\right)^2$$

We have:  $\Sigma_{in} = 0$   
 $k_i$  : the node degree of node i

Example for the left graph:



$$m = 0.1 + 0.2 + 0.2 = 0.5$$

$$Q(\{1\}) = -\left(\frac{k_1}{2m}\right)^2 = -\left(\frac{0.1 + 0.2}{2m}\right)^2 = -0.09$$

$$Q(\{2\}) = -\left(\frac{k_2}{2m}\right)^2 = -\left(\frac{0.1 + 0.2}{2m}\right)^2 = -0.09$$

$$Q(\{3\}) = -\left(\frac{k_3}{2m}\right)^2 = -\left(\frac{0.2 + 0.2}{2m}\right)^2 = -0.16$$

- Another extreme case: one node forms one community

The modularity score for the community formed by node i:

$$Q(C_i) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 = \frac{0}{2m} - \left(\frac{k_i}{2m}\right)^2 = -\left(\frac{k_i}{2m}\right)^2$$

We have:  $\Sigma_{in} = 0$   
 $\Sigma_{tot} = k_i$

$k_i$  : the node degree of node i

The total modularity score of all communities:  
(N indicates the total number of communities)

$$\sum_{i=1}^N Q(C_i) = -\sum_{i=1}^N \left(\frac{k_i}{2m}\right)^2$$

The total score is less than 0

## Modularity score (discussion)

What is a good community?

The nodes in the community have lots of internal connections and very few external connections (to the rest of the network)

Modularity score: measure the quality of a community

A large value indicates a lot of internal connections

$$Q(C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2$$

$\frac{\Sigma_{in}}{2m} \leq 1$

$\frac{\Sigma_{tot}}{2m} \leq 1$

1. A large value indicates a large community or large node degrees.
2. This term will penalize large community and the external connections.
  - The node degree includes the internal and external links of the node.
  - A large number of external links will lead to a large node degree.
3. Intuitively, the square operation is to downgrade the impact of the second term: e.g.  $(0.1)^2 < 0.1$

### Louvain Algorithm

- aims to greedily maximize the Modularity score.
- Modularity score: a metric to measure the quality of a community

#### Single pass Louvain Algorithm

- Community generation
  - Also called modularity optimization

# Louvain Algorithm: community generation

- Initialization: each node forms a distinct community
- A: generate a random list of nodes (start one scan)
- B: (one scan) sequentially process each node in the list for community update
- Repeat A, B until converge
- Converge:  
there is no update of the community in the last scan

- B: sequentially process each node for community update
- 1) Node movement step.
  - Identify possible movements by finding neighboring communities (directly connected external communities)
  - For each movement: compute the modularity gain ( $\Delta Q$ ) as the movement score.

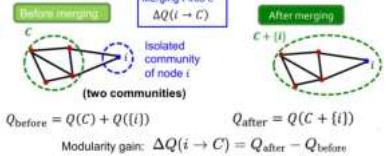
$$\Delta Q = Q_{\text{after}} - Q_{\text{before}}$$

- 2) Community update step.  
Move the node to a community that yields the largest positive score ( $\Delta Q$ ). If the largest score is not positive, there is no movement of the node.

## Modularity gain (movement score)

Directly use this equation to calculate the modularity score for each community:

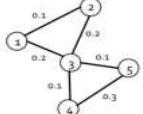
$$Q(C) = \frac{\sum_{i,j} a_{ij}}{2m} - \left(\frac{\sum_i d_i}{2m}\right)^2$$

$$\Delta Q = Q_{\text{after}} - Q_{\text{before}}$$


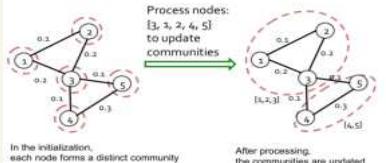
## Example: community update

Q: A graph is given below.

In the initialization, each node forms a distinct community. Use Louvain algorithm to sequentially process the nodes in the order (3, 1, 2, 4, 5) to update the communities given in the initialization.



## Result



## Intermediate steps

Sequentially process the node list {3, 1, 2, 4, 5}

### processing node 3

- Identify neighbouring communities of node 3: {1}, {2}, {4} and {5}
- Calculate the movement scores (modularity gains) for the following 4 movements:

$$\begin{aligned} \Delta Q(3 \rightarrow \{1\}) &= \\ \Delta Q(3 \rightarrow \{2\}) &= \\ \Delta Q(3 \rightarrow \{4\}) &= \\ \Delta Q(3 \rightarrow \{5\}) &= \end{aligned}$$

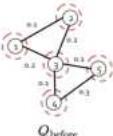
## Intermediate steps

Sequentially process the node list {3, 1, 2, 4, 5}

### processing node 3

Movement score (Modularity gain) of moving node 3 to community {1}:

$$\Delta Q(3 \rightarrow \{1\}) = Q_{\text{after}} - Q_{\text{before}} = Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})]$$



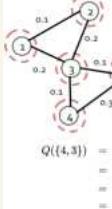
$Q(C) = \frac{\sum_{i,j} a_{ij}}{2m} - \left(\frac{\sum_i d_i}{2m}\right)^2$

Sequentially process the node list {3, 1, 2, 4, 5}

### processing node 3

The movement score of moving node 3 to {4}:  $\Delta Q(3 \rightarrow \{4\})$

$$\begin{aligned} \Delta Q(3 \rightarrow \{4\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{4\} + \{3\}) - [Q(\{4\}) + Q(\{3\})] \\ &= Q(\{4,3\}) - [Q(\{4\}) + Q(\{3\})] \\ &= -0.15 - (-0.04 - 0.09) \\ &= -0.02 \end{aligned}$$

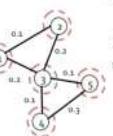


$$\begin{aligned} Q(\{4,3\}) &= \frac{\sum_{i,j} a_{ij}}{2m} - \left(\frac{\sum_i d_i}{2m}\right)^2 \\ &= \frac{2.5*6}{2*10} - \left(\frac{2.5*3}{2*5}\right)^2 \\ &= \frac{2.5*6}{2*10} - \left(\frac{2.5*0.6}{2*5}\right)^2 \\ &= -0.15 \end{aligned}$$

$$Q(\{4\}) = \left[0 - \left(\frac{1.5}{2m}\right)^2\right] = -0.04$$

### processing node 3

$\Delta Q(3 \rightarrow \{1\}) = Q_{\text{after}} - Q_{\text{before}} = Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})]$

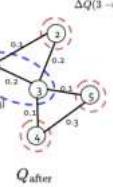


$$m = 0.1 + 0.2 + 0.2 + 0.1 + 0.1 + 0.3 = 1$$

$$\begin{aligned} Q(\{1\}) &= \left[0 - \left(\frac{1.5}{2m}\right)^2\right] \\ &= -0.0225 \\ Q(\{3\}) &= \left[0 - \left(\frac{1.5}{2m}\right)^2\right] \\ &= -0.09 \end{aligned}$$

### processing node 3

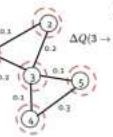
$\Delta Q(3 \rightarrow \{1\}) = Q_{\text{after}} - Q_{\text{before}} = Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})]$



$$\begin{aligned} Q(\{1,3\}) &= \frac{\sum_{i,j} a_{ij}}{2m} - \left(\frac{\sum_i d_i}{2m}\right)^2 \\ &= \frac{2.5*6}{2*10} - \left(\frac{2.5*1.5}{2*5}\right)^2 \\ &= \frac{2*0.2}{2} - \left(\frac{0.3*0.6}{2}\right)^2 \\ &= -0.0025 \end{aligned}$$

### processing node 3

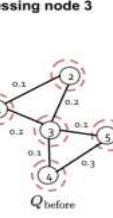
Movement score (Modularity gain) of moving node 3 to community {2}:



$$\begin{aligned} \Delta Q(3 \rightarrow \{2\}) &= Q_{\text{after}} - Q_{\text{before}} = Q(\{2\} + \{3\}) - [Q(\{2\}) + Q(\{3\})] \\ &= Q(\{1,3\}) - [Q(\{1\}) + Q(\{3\})] \\ &= -0.0025 - (-0.0225 - 0.09) \\ &= 0.11 \end{aligned}$$

### processing node 3

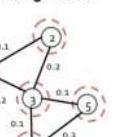
Sequentially process the node list {3, 1, 2, 4, 5}



$$\begin{aligned} \Delta Q(3 \rightarrow \{2\}) &= Q_{\text{after}} - Q_{\text{before}} = Q(\{2\} + \{3\}) - [Q(\{2\}) + Q(\{3\})] \\ &= Q(\{1,3\}) - [Q(\{1\}) + Q(\{3\})] \\ &= -0.0025 - (-0.0225 - 0.09) \\ &= 0.11 \end{aligned}$$

### processing node 3

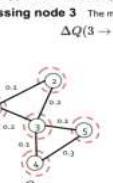
Movement score (Modularity gain) of moving node 3 to community {2}:



$$\begin{aligned} \Delta Q(3 \rightarrow \{2\}) &= Q_{\text{after}} - Q_{\text{before}} = Q(\{2\} + \{3\}) - [Q(\{2\}) + Q(\{3\})] \\ &= Q(\{1,3\}) - [Q(\{1\}) + Q(\{3\})] \\ &= -0.0025 - (-0.0225 - 0.09) \\ &= 0.11 \end{aligned}$$

### processing node 3

The movement score of moving node 3 to {4}:  $\Delta Q(3 \rightarrow \{4\})$

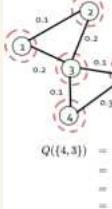


$$\begin{aligned} \Delta Q(3 \rightarrow \{4\}) &= Q_{\text{after}} - Q_{\text{before}} = Q(\{4\} + \{3\}) - [Q(\{4\}) + Q(\{3\})] \\ &= Q(\{1,3\}) - [Q(\{1\}) + Q(\{3\})] \\ &= -0.0025 - (-0.0225 - 0.09) \\ &= 0.11 \end{aligned}$$

Sequentially process the node list {3, 1, 2, 4, 5}

### processing node 3

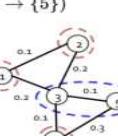
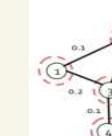
The movement score of moving node 3 to {5}:  $\Delta Q(3 \rightarrow \{5\})$



$$\begin{aligned} Q(\{4,3\}) &= \frac{\sum_{i,j} a_{ij}}{2m} - \left(\frac{\sum_i d_i}{2m}\right)^2 \\ &= \frac{2.5*6}{2*10} - \left(\frac{2.5*1.5}{2*5}\right)^2 \\ &= -0.15 \end{aligned}$$

$$Q(\{4\}) = \left[0 - \left(\frac{1.5}{2m}\right)^2\right] = -0.04$$

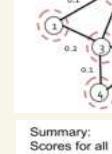
The movement score of moving node 3 to {5}:  $\Delta Q(3 \rightarrow \{5\})$



Sequentially process the node list {3, 1, 2, 4, 5}

### processing node 3

The gain of moving node 3 to {5} is the same as:  $\Delta Q(3 \rightarrow \{4\})$



$$\Delta Q(3 \rightarrow \{5\}) = -0.02$$

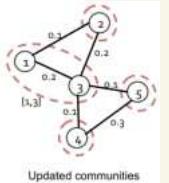
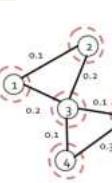
Summary: Scores for all possible movements:

$$\Delta Q(3 \rightarrow \{1\}) = \Delta Q(3 \rightarrow \{2\}) = -0.11$$

$$\Delta Q(3 \rightarrow \{4\}) = \Delta Q(3 \rightarrow \{5\}) = -0.02$$

There is a tie for 3->{1} and 3->{2}. Randomly choose one community to proceed.

Here we choose {1}. Move node #3 to community {1}, now we have {1,3}.



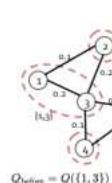
After processing node 3, we move node 3 to {1}.

Sequentially process the node list {3, 1, 2, 4, 5}

### processing node 1

The neighbouring (directly connected) communities of node 1: Community {2}

Only one possible movement:



$$Q_{\text{after}} = Q(\{1\}) + Q(\{2\})$$

$$\begin{aligned} Q(\{1\}) &= \frac{\sum_{i,j} a_{ij}}{2m} - \left(\frac{\sum_i d_i}{2m}\right)^2 \\ &= \frac{2.5*6}{2*10} - \left(\frac{2.5*1.5}{2*5}\right)^2 \\ &= -0.1 \end{aligned}$$

$$\begin{aligned} Q(\{2\}) &= \frac{\sum_{i,j} a_{ij}}{2m} - \left(\frac{\sum_i d_i}{2m}\right)^2 \\ &= \frac{2.5*6}{2*10} - \left(\frac{2.5*1.5}{2*5}\right)^2 \\ &= -0.1 \end{aligned}$$

$$Q(\{1,2\}) = \left[0 - \left(\frac{1.5}{2m}\right)^2\right] = -0.09$$

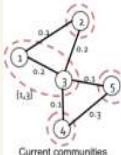
$$Q(\{1\}) = \left[0 - \left(\frac{1.5}{2m}\right)^2\right] = -0.09$$

$$Q(\{2\}) = \left[0 - \left(\frac{1.5}{2m}\right)^2\right] = -0.09$$

$$Q(\{1,2\}) = \left[0 - \left(\frac{1.5}{2m}\right)^2\right] = -0.09$$

Sequentially process the node list [3, 1, 2, 4, 5]

### processing node 1



$$\begin{aligned} &\text{Move node 1 out from } \{1, 3\} \text{ and add node 1 to } \{2\}: \\ \Delta Q(1, \{3\} \rightarrow \{2\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{3\}) + Q(\{1, 2\}) - (Q(\{1, 3\}) + Q(\{2\})) \\ &= -0.09 + 0.01 - (-0.0025 - 0.0225) \\ &= -0.055 \end{aligned}$$

Current communities

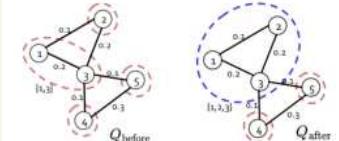
The largest gain is a negative value (there is only one neighboring community), do not move node 1 to [2].

### processing node 2

The neighbouring communities of node 2: Community {1, 3}. Only one possible movement:

$$\Delta Q(2 \rightarrow \{1, 3\}) = Q_{\text{after}} - Q_{\text{before}}$$

$$= Q(\{1, 2, 3\}) - Q(\{1, 3\}) - Q(\{2\})$$



### processing node 2

The neighbouring communities of node 2: Community {1, 3}. Only one possible movement:

$$\begin{aligned} \Delta Q(2 \rightarrow \{1, 3\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{1, 3\} + \{2\}) - [Q(\{1, 3\}) + Q(\{2\})] \\ &= Q(\{1, 2, 3\}) - Q(\{1, 3\}) - Q(\{2\}) \end{aligned}$$

$$\begin{aligned} Q(\{1, 2, 3\}) &= \frac{5}{2m} - (\frac{k_1}{2m})^2 \\ &= \frac{2x(k_1+k_2+k_3)}{2m} - (\frac{k_1+k_2+k_3}{2m})^2 \\ &= 0.14 \end{aligned}$$

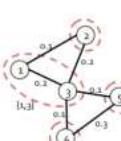
$$\begin{aligned} Q(\{2\}) &= -(\frac{k_2}{2m})^2 \\ &= -\frac{(k_2)^2}{2m} \\ &= -0.0225 \\ &= -0.0025 \end{aligned}$$

$$\begin{aligned} \Delta Q(2 \rightarrow \{1, 3\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{1, 3\} + \{2\}) - [Q(\{1, 3\}) + Q(\{2\})] \\ &= Q(\{1, 2, 3\}) - Q(\{1, 3\}) - Q(\{2\}) \\ &= 0.165 \end{aligned}$$

Current communities

Updated communities after processing node 2

The gain is a positive value, and there is only 2 neighboring community, so it's the largest positive gain. We move 2 to [1,3].



After processing node 2, we move node 2 to [1,3].

### processing node 4

The neighbouring communities of node 4: 1. Community {1, 2, 3} 2. Community {5}

There are two possible movements. We need to calculate the following two modularity gains:

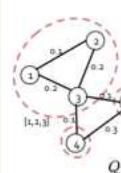
$$\Delta Q(4 \rightarrow \{1, 2, 3\})$$

$$\Delta Q(4 \rightarrow \{5\})$$

Current communities

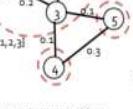
### processing node 4

$$\Delta Q(4 \rightarrow \{1, 2, 3\}) = Q_{\text{after}} - Q_{\text{before}}$$



### processing node 4

$$\begin{aligned} \Delta Q(4 \rightarrow \{1, 2, 3\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{4\} + \{1, 2, 3\}) - [Q(\{4\}) + Q(\{1, 2, 3\})] \\ &= Q(\{1, 2, 3, 4\}) - Q(\{4\}) - Q(\{1, 2, 3\}) \\ &= (-0.04) - (-0.04) - 0.14 \\ &= -0.14 \end{aligned}$$



Current communities

$$k_4 = 0.1 + 0.3 = 0.4$$

$$Q(\{4\}) = -(\frac{k_4}{2m})^2 = -0.04$$

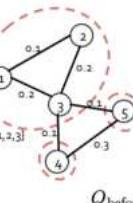
$$Q(\{1, 2, 3, 4\}) = \frac{5}{2m} - (\frac{k_1}{2m})^2$$

$$= \frac{2x(k_1+k_2+k_3+k_4)}{2m} - (\frac{k_1+k_2+k_3+k_4}{2m})^2$$

$$= \frac{2x(0.1+0.2+0.3+0.4)}{2m} - (\frac{0.3+0.3+0.6+0.4}{2m})^2$$

$$= -0.04$$

$$\Delta Q(4 \rightarrow \{5\}) = Q_{\text{after}} - Q_{\text{before}}$$



$Q_{\text{before}}$

$Q_{\text{after}}$

$$\Delta Q(4 \rightarrow \{5\}) = Q_{\text{after}} - Q_{\text{before}}$$

$$\begin{aligned} &= Q(\{4\} + \{5\}) - [Q(\{4\}) + Q(\{5\})] \\ &= Q(\{4, 5\}) - Q(\{4\}) - Q(\{5\}) \\ &= 0.14 - (-0.04) - (-0.04) \\ &= 0.22 \end{aligned}$$

$$Q(\{4\}) = -(\frac{k_4}{2m})^2 = -0.04$$

$$k_5 = 0.1 + 0.3 = 0.4$$

$$Q(\{5\}) = -(\frac{k_5}{2m})^2 = -0.04$$

$$\begin{aligned} \Delta Q(4 \rightarrow \{5\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{4, 5\}) - [Q(\{4\}) + Q(\{5\})] \\ &= Q(\{4, 5\}) - Q(\{4\}) - Q(\{5\}) \\ &= 0.14 - (-0.04) - (-0.04) \\ &= 0.22 \end{aligned}$$

$Q_{\text{before}}$

$Q_{\text{after}}$

$$\Delta Q(4 \rightarrow \{5\}) = Q_{\text{after}} - Q_{\text{before}}$$

$$= Q(\{4\}) + Q(\{1, 2, 3, 5\}) - [Q(\{4, 5\}) + Q(\{1, 2, 3\})]$$

$$= -0.04 - 0.04 - (0.14 + 0.14) = -0.36$$

The largest gain is a negative value (there is one neighboring community). Do not move node 5.

Summary:  
Scores for the 2 possible movements

$$\Delta Q(4 \rightarrow \{1, 2, 3\}) = -0.14$$

$$\Delta Q(4 \rightarrow \{5\}) = 0.22$$

Current communities

$\Delta Q(4 \rightarrow \{5\})$  has the largest positive gain.

We move node 4 to [5].



updated communities

After processing node 4, we move node 4 to [5].



updated communities

### processing node 5

The neighbouring communities of node 5: Community {1, 2, 3}

There is only one neighbouring community for node 5, so there is only one possible movement.

We need to calculate the following modularity gain:

$$\Delta Q(\{4, 5\} \rightarrow \{1, 2, 3\}) = Q_{\text{after}} - Q_{\text{before}}$$

$$= Q(\{4\}) + Q(\{1, 2, 3, 5\}) - [Q(\{4, 5\}) + Q(\{1, 2, 3\})]$$

$$= Q(\{1, 2, 3, 4, 5\}) - Q(\{4, 5\}) - Q(\{1, 2, 3\})$$

$$= \frac{5}{2m} - (\frac{k_5}{2m})^2 - (\frac{k_4+k_5}{2m})^2$$

$$= \frac{2x(k_1+k_2+k_3+k_4+k_5)}{2m} - (\frac{k_1+k_2+k_3+k_4+k_5}{2m})^2$$

$$= \frac{2x(0.1+0.2+0.3+0.4+0.5)}{2m} - (\frac{1.5+1.5+0.6+0.4}{2m})^2$$

$$= -0.04$$

The below values can be obtained from previous steps:

$$Q(\{4\}) = -(\frac{k_4}{2m})^2 = -0.04$$

$$Q(\{4, 5\}) = 0.14$$

$$Q(\{1, 2, 3\}) = 0.14$$

$$\Delta Q(\{4, 5\} \rightarrow \{1, 2, 3\}) = Q_{\text{after}} - Q_{\text{before}}$$

$$= Q(\{4\}) + Q(\{1, 2, 3, 5\}) - [Q(\{4, 5\}) + Q(\{1, 2, 3\})]$$

$$= -0.04 + 0.14 - (0.14 + 0.14) = 0.16$$

$$\Delta Q(\{4, 5\} \rightarrow \{1, 2, 3\}) = Q_{\text{after}} - Q_{\text{before}}$$

$$= Q(\{4\}) + Q(\{1, 2, 3, 5\}) - [Q(\{4, 5\}) + Q(\{1, 2, 3\})]$$

$$= -0.04 - 0.04 - (0.14 + 0.14) = -0.36$$

The largest gain is a negative value (there is one neighboring community). Do not move node 5.

Current communities

Result

After processing the node list {3, 1, 2, 4, 5}, we have two communities: {1, 2, 3} and {4, 5}.

Additional discussion: (beyond the question)

We have finished the 1<sup>st</sup> scan after processing the node list.

To fully complete community generation, continue to do the 2<sup>nd</sup> scan, 3<sup>rd</sup> scan, ...

In each scan, we randomly generate a node list for processing.

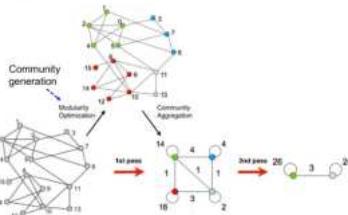
Converge:

If there is no node movement in one scan, the community generation step converges.

Non-examitable!

#### • Multi-pass Louvain Algorithm

- Multiple passes:
  - Produce hierarchical results
  - One pass corresponds to one hierarchical level
- Each pass is made of 2 phases:
  - Phase 1: community generation (we have learnt)
  - Phase 2: community aggregation

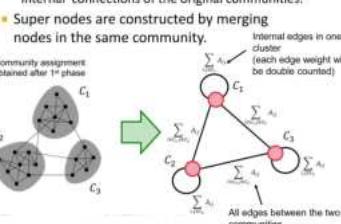


Multi-pass Louvain Algorithm high-level illustration

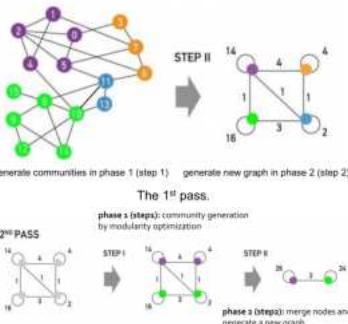
For un-weighted graph: the weight for each edge is 1

#### • Phase 2: community aggregation

- Generate a new graph based on the communities
- Create one node (super node) for one community in the new graph
- Create edges of the super nodes if the original communities are connected by at least one edge
  - The weight of the edge between the two super nodes is the sum of the edge weights between the original communities
- Create self-connections of the super nodes based on the internal connections of the original communities.



## Example



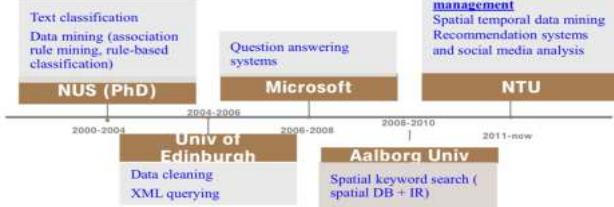
#### • Further reading

- BigCLAM
  - For overlapping community detection
- Spectral clustering
  - A well-known clustering method on graph
  - <http://snap.stanford.edu/class/cs224w-2019/>
  - [https://en.wikipedia.org/wiki/Spectral\\_clustering](https://en.wikipedia.org/wiki/Spectral_clustering)

# Introduction to second half of SC4020-CE4032-CZ4032 Data Analytics and Mining

- Instructor: Prof Cong Gao
- Specialized in data science (data management, data mining)
- Email: [gaocong@ntu.edu.sg](mailto:gaocong@ntu.edu.sg)
- Homepage: <https://personal.ntu.edu.sg/gaocong/>
- Office: N4-2c-103
- Emails: Pls put [SC4020] as email title
- You are encouraged to use Discussion Board in the course website so that others can read/answer your questions as well

## My Research Areas



## Topics of second half/ tutorial

- Lecture:
  - Association Rule Mining (week 7–8)
  - Sequential pattern mining, Classification, and overfitting (week 9–10)
  - Recommendation Systems (Week 11)
  - Data Processing, Data Cleaning and integration (Week 12)
  - Advanced topic (Week 12)
- Tutorial will start from Week 7. No tutorial in Week 11**

## Assessment (some content is from first lecture)

- Final exam (50%) + Assignments (50%)
  - Closed book exam
- Group-based Assignment for 2nd half: 25%
  - Will release toward the end of Week 8
  - Due before the start of final exam (21 Nov, Thursday)
- Individual Contribution Claim** (agreed by all)
- If there is a large difference, get different scores
- Plagiarism → 0 marks.

## Two Additional Reference Books

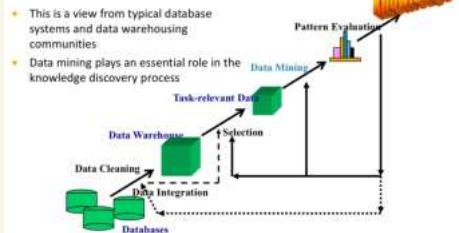


Data Mining: Concepts and Techniques (3rd ed.),  
Morgan Kaufmann, 2011  
• Jiawei Han, Micheline Kamber and Jian Pei



Introduction to Data Mining (Second Edition)  
• by Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar

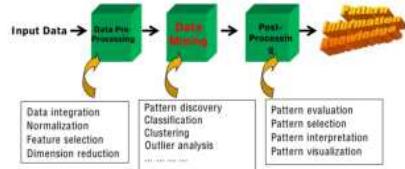
## Knowledge Discovery (KDD) Process



## Data Mining in Business Intelligence



## KDD Process: A View from ML and Statistics



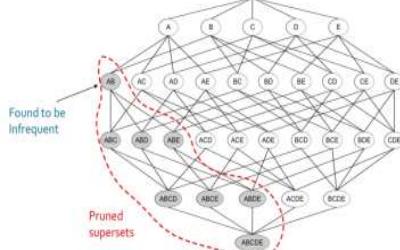
This is a view from typical machine learning and statistics communities

## A Brief History of Data Mining Society (for your reference)

- 1989 IJCAI Workshop on Knowledge Discovery in Databases
  - Knowledge Discovery in Databases (G. Piatetsky-Shapiro and W. Frawley, 1991)
- 1991-1994 Workshops on Knowledge Discovery in Databases
  - Advances in Knowledge Discovery and Data Mining (U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, 1996)
- 1995-1998 International Conferences on Knowledge Discovery in Databases and Data Mining (KDD'95-98)
  - Journal of Data Mining and Knowledge Discovery (1997)
  - ACM SIGKDD conferences since 1998 and SIGKDD Explorations
  - More conferences on data mining
    - PAKDD (1997), PKDD (1997), SIAM-Data Mining (2001), (IEEE) ICDM (2001), WSDM (2008), etc.
  - ACM Transactions on KDD (2007)

Part 2

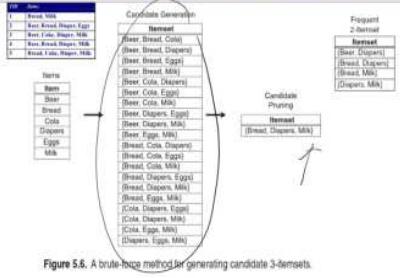




## Apriori Algorithm

- $F_k$ : frequent k-itemsets (containing k items)
  - $C_k$ : candidate k-itemsets
- Algorithm**
    - Let  $k=1$
    - Generate  $F_1 = \{\text{frequent 1-itemsets}\}$
    - Repeat until  $F_k$  is empty
      - Candidate Generation:** Generate  $C_{k+1}$  from  $F_k$
      - Candidate Pruning:** Prune candidate itemsets in  $C_{k+1}$  containing subsets of length  $k$  that are infrequent
      - Support Counting:** Count the support of each candidate in  $C_{k+1}$  by scanning the DB
      - Candidate Elimination:** Eliminate candidates in  $C_{k+1}$  that are infrequent, leaving only those that are frequent  $\Rightarrow F_{k+1}$

### Candidate Generation: Brute-force method



### Candidate Generation: $F_{k-1} \times F_{k-1}$ Method

- To generate  $C_{k+1}$  from  $F_k$ : Merge two frequent (k)-itemsets if their first (k-1) items are identical
- $F_3 = \{\text{ABC, ABD, ABE, ACD, BCD, BDE, CDE}\}$
- Merge(ABC, ABD) = ABCD
- Merge(ABC, ABE) = ABCE
- Merge(ABD, ABE) = ABDE
- Do not merge(ABD, ACD) because they share only prefix of length 1 instead of length 2

## Candidate Pruning

- Let  $F_3 = \{\text{ABC, ABD, ABE, ACD, BCD, BDE, CDE}\}$  be the set of frequent 3-itemsets
- $C_4 = \{\text{ABCD, ABCE, ABDE}\}$  is the set of candidate 4-itemsets generated (from previous slide)
- Candidate pruning**
  - Prune ABCE because ACE and BCE are infrequent
  - Prune ABDE because ADE is infrequent
- After candidate pruning:  $C_4 = \{\text{ABCD}\}$

### Another example:

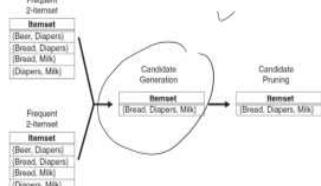
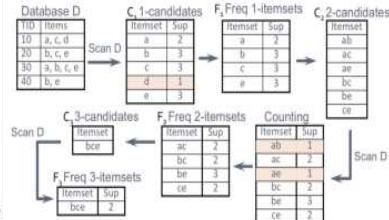


Figure 5.8: Generating and pruning candidate  $k$ -itemsets by merging pairs of frequent  $(k-1)$ -itemsets.

## Example: Apriori-based Mining

Minsup=2



## Step2: Rule generation

- Step 1:** Find all frequent itemsets  $I$ 
  - Generate all itemsets whose support  $\geq \text{minsup}$
- Step 2: Rule generation**
  - Given a frequent itemset  $I$ , find all non-empty subsets  $A \subset I$  such that  $A \rightarrow I - A$  satisfies the **minimum confidence requirement minconf**
    - If  $\{A, B, C\}$  is a frequent itemset, candidate rules:
 
$$\begin{array}{ll} A \rightarrow D & ABD \rightarrow C \\ A \rightarrow CD & B \rightarrow ACD \\ AB \rightarrow CD & AC \rightarrow BD \\ BD \rightarrow AC & AD \rightarrow BC \end{array}$$
    - An example to compute the rule confidence
 
$$\text{confidence}(A \rightarrow B) = \text{support}(A, B) / \text{support}(A, B)$$
  - Do the above for every frequent itemset, and output all the rules above the confidence threshold

## Example

$B_1 = \{m, c, b\}$	$B_2 = \{m, p, j\}$	Support threshold
$B_3 = \{m, c, b, n\}$	$B_4 = \{c, j\}$	$\text{minsup} = 3$
$B_5 = \{m, p, b\}$	$B_6 = \{m, c, b\}$	confidence
$B_7 = \{c, b, j\}$	$B_8 = \{b, c\}$	$\text{minconf} = 0.75$

### 1) Frequent itemsets:

$$\{(b, m); s = 4 / (b, c); 4 / (c, m); 3 / (c, j); 3 / (m, c, b); 3 / (m, p, j)\}$$

### 2) Generate rules:

$$\{(b, m); b \rightarrow m; s = 4 / 6; m \rightarrow b; c \rightarrow 4 / 5\}$$

$$\{(b, c); b \rightarrow c; s = 5 / 6; c \rightarrow b; c \rightarrow 5 / 6\}$$

$$\dots$$

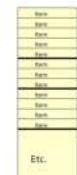
$$\{(m, c, b); b \rightarrow c; s = 3 / 5; b, m \rightarrow c; s = 3 / 4; m, c \rightarrow b; s = 3 / 3\}$$

$$\dots$$

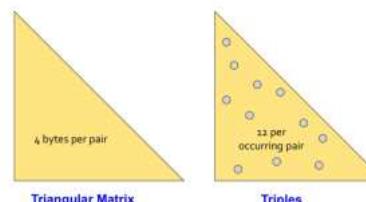
$$\dots$$
</div

## Itemsets: Computation Model

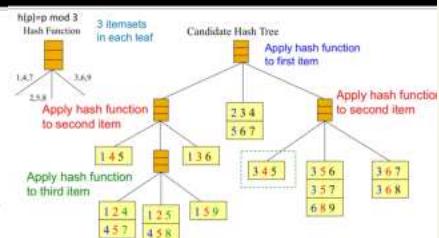
- Back to finding frequent itemsets
- Typically, data is kept in flat files rather than in a database system:
  - Stored on disk
  - Stored basket-by-basket
  - Baskets are small but we have many baskets and many items



## Comparing the 2 Approaches



## Building a Hash Tree of Itemsets



## Comparing the two approaches

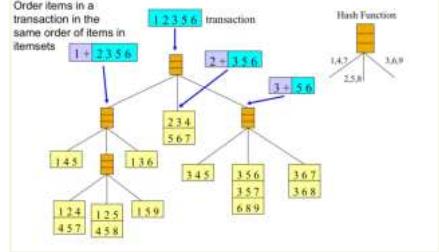
### Approach 1: Triangular Matrix

- $n = \text{total number items}$
- Count pair of items  $\{i, j\}$  only if  $i < j$
- Keep pair counts in lexicographic order:
  - $\{1, 2\}, \{1, 3\}, \dots, \{1, n\}, \{2, 3\}, \{2, 4\}, \dots, \{2, n\}, \{3, 4\}, \dots$
  - Pair  $\{i, j\}$  is at position  $(i-1)(n-i)/2 + j - i$
  - Total number of pairs  $n(n-1)/2$ ; total bytes =  $2n^2$
- Triangular Matrix requires 4 bytes per pair
- Approach 2 uses 12 bytes per occurring pair (but only for pairs with count > 0)
- Beats Approach 1 if less than 1/3 of possible pairs actually occur

Problem is if we have too many items so the pairs do not fit into memory.

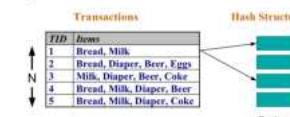
Can we do better?

## Support Counting Using a Hash Tree



## Support Counting of Candidate Itemsets

- When itemsets are longer, matching every candidate itemset against every transaction is an expensive operation
- To reduce number of comparisons, store the candidate itemsets in a hash structure
- Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets



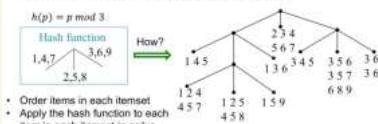
## Support Counting Using a Hash Tree (the next 5 slides for your information only, not examinable)

Suppose you have 15 candidate itemsets of length 3:

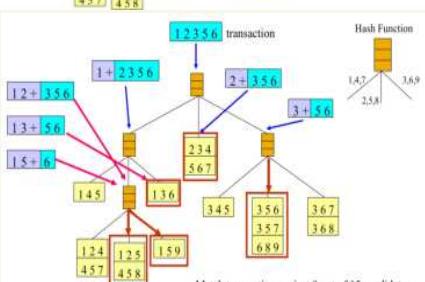
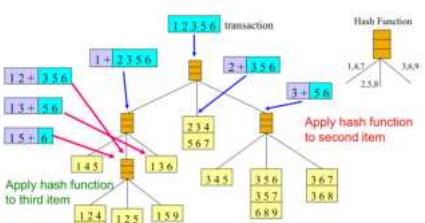
$\{1, 4, 5\}, \{1, 2, 4\}, \{4, 5, 7\}, \{1, 2, 5\}, \{4, 5, 8\}, \{1, 5, 9\}, \{1, 3, 6\}, \{2, 3, 4\}, \{5, 6, 7\}, \{3, 4, 5\}, \{3, 5, 6\}, \{3, 5, 7\}, \{6, 8, 9\}, \{3, 6, 7\}, \{3, 6, 8\}$

Use a data structure to organize these itemsets to reduce computation:

- Hash function
- Max leaf size: max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, split the node)



- Order items in each itemset
- Apply the hash function to each item in each itemset in order



Match transaction against 9 out of 15 candidates

## Computation Model

- The dominating cost of mining disk-resident data is usually the **number of disk I/Os**
- In practice, association-rule algorithms read the data in **passes** – all baskets read in turn
- We measure the cost by the **number of passes** an algorithm makes over the data

## Main-Memory Bottleneck

- Itemsets:** To find frequent itemsets, we have to count them. To count them, we have to generate them, and usually store them in **memory**.
- For many frequent-itemset algorithms, **main-memory** is the critical resource
  - As we read baskets, we need to count something, e.g., occurrences of pairs of items
  - The number of different things we can count is limited by main memory
  - Swapping counts in/out is a disaster (**why?**)

## Naïve Algorithm for finding frequent pairs

- Let us consider finding frequent pairs
- Read file once, counting in main memory the occurrences of each pair:
  - From each basket of  $n$  items, generate its  $n(n-1)/2$  pairs by two nested loops
- Fails if #itemsets $^2$  exceeds main memory
  - Remember: #items can be 100K (Wal-Mart) or 10B (Web pages)
  - Suppose 10<sup>5</sup> items, counts are 4-byte integers
  - Number of pairs of items:  $10^5(10^5-1)/2 = 5 \times 10^9$
  - Therefore,  $2 \times 10^{10}$  (20 gigabytes) of memory needed

## Counting Pairs in Memory

### Two approaches:

- Approach 1: Count all pairs using a matrix
- Approach 2: Keep a table of triples  $[i, j, c] =$  "the count of the pair of items  $\{i, j\}$  is  $c$ "
- If integers and item sets are 4 bytes, we need approximately 12 bytes for pairs with count > 0
- Plus some additional overhead for the hashtable

### Note:

- Approach 1 only requires 4 bytes per pair
- Approach 2 uses 12 bytes per pair (but only for pairs with count > 0)

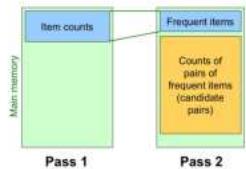
## Implementation of A-Priori Algorithm

- Pass 1:** Read baskets and count in main memory the occurrences of each **individual item**
  - Requires only memory proportional to #items

Items that appear  $\geq s$  times are the **frequent items**

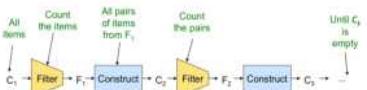
- Pass 2:** Read baskets again and count in main memory **only** those pairs where both elements are frequent (from Pass 1)
  - Requires memory proportional to square of **frequent** items only (for counts)
  - Plus a list of the frequent items (so you know what must be counted)

## Main-Memory: Picture of A-Priori



## Frequent Triples, Etc.

- Other passes: For each  $k$ , we construct two sets of ***k-itemsets*** (sets of size  $k$ ):
  - $C_k$  = **candidate *k-itemsets***
  - $F_k$  = the set of truly frequent  $k$ -itemsets
- For each  $k$ , need room in main memory to count each candidate, and need to make a pass of the data to count the occurrences for itemsets in  $C_k$



## PCY (Park-Chen-Yu) Algorithm

## Observation:

- In pass 1 of A-Priori, most memory is idle
- We store only individual item counts
- Can we use the idle memory to reduce memory required in pass 2?**
- Pass 1 of PCY:** In addition to item counts, maintain a hash table with as many buckets as fit in memory
  - Keep a **count** for each bucket into which pairs of items are hashed
    - For each bucket just keep the **count**, not the actual pairs that hash to the bucket!

## PCY Algorithm – First Pass

```

FOR (each basket) :
  FOR (each item in the basket) :
    add 1 to its count;
  FOR (each pair of items) :
    hash the pair to a bucket;
    add 1 to the count for that bucket;
  New in PCY
  
```

## Few things to note:

- Pairs of items need to be generated from the input file; they are not present in the file
- We are not just interested in the presence of a pair, but we need to see whether it is present at least  $s$  (support) times

## Observations about Buckets

- Observation:** If a bucket contains a **frequent pair**, then the bucket is surely **frequent**
- However, even without any frequent pair, a bucket can still be frequent  $\heartsuit$
- So, we cannot use the hash to eliminate any member (pair) of a "frequent" bucket
- But, for a bucket with total count less than  $s$ , none of its pairs can be frequent  $\heartsuit$**
- Pairs that hash to this bucket can be eliminated as candidates (even if the pair consists of 2 frequent items)

## Pass 2:

Only count pairs that hash to frequent buckets

## Example: PCY Algorithm First Pass

## Items

## =

{milk, coke, beer, pepsi, juice}

## Baskets:

$B_1 = \{m, c, b\}$	$B_2 = \{m, p, j\}$
$B_3 = \{m, b\}$	$B_4 = \{c, j\}$
$B_5 = \{m, p, b\}$	$B_6 = \{m, c, b, j\}$
$B_7 = \{c, b, j\}$	$B_8 = \{b, c\}$

- By assigning  $milk = 1$ ,  $coke = 2$ ,  $beer = 3$ ,  $pepsi = 4$ ,  $juice = 5$  then **Baskets:**

$B_1 = \{1, 2, 3\}$	$B_2 = \{1, 4, 5\}$
$B_3 = \{1, 3\}$	$B_4 = \{2, 5\}$
$B_5 = \{1, 3, 4\}$	$B_6 = \{1, 2, 3, 5\}$
$B_7 = \{2, 3, 5\}$	$B_8 = \{2, 3\}$

- Define a hash function (Hashing pair  $(i, j)$  to bucket  $K$ ):

$$h(i, j) = (i + j) \% 5 = K$$

## Example:

$$h(1,3) = (1 + 3) \% 5 = 4 \rightarrow \text{Bucket } 4$$

## PCY Algorithm – Between Passes

## Optimization: Replace the buckets by a bit-vector:

- 1 means the bucket count exceeded the support  $s$  (call it a **frequent bucket**); 0 means it did not
- 4-byte integer counts are replaced by bits, so the bit-vector requires  $1/32$  of memory
- Also, decide which items are frequent and list them for the second pass

## PCY Algorithm – Pass 2

- Count all pairs  $(i, j)$  that meet the conditions for being a **candidate pair**:

- Both  $i$  and  $j$  are frequent items
- The pair  $(i, j)$  hashes to a bucket whose bit in the bit vector is 1 (i.e., a **frequent bucket**)

## Both conditions are necessary for the pair to have a chance of being frequent

- Implementation details: On second pass, a table of **(item, item, count)** triples is essential (we cannot use triangular matrix approach, **why?**)
    - Thus, hash table must eliminate approx.  $2/3$  of the candidate pairs for PCY to beat A-Priori
- Main-Memory: Picture of PCY
- 
- For support threshold  $s = 3$
- Item Count
- | Item # | Count |
|--------|-------|
| 1      | 5     |
| 2      | 5     |
| 3      | 5     |
| 4      | 2     |
| 5      | 4     |
- Items in each bucket
- | Bucket #         | Count |
|------------------|-------|
| (1,4), (2,3) → 0 | 0     |
| (1,5), (2,4) → 1 | 1     |
| (2,5), (3,4) → 2 | 2     |
| (1,2), (3,5) → 3 | 3     |
| (1,3), (4,5) → 4 | 4     |
- For  $s = 3$ ,  $L_1 = \{1, 2, 3, 5\}$ , and Bitmap  $\{1, 0, 1, 1, 1\}$
- First Pass:
- ```

FOR (each basket) :
  FOR (each item in the basket) :
    add 1 to its count;
  FOR (each pair of items) :
    hash the pair to a bucket;
    add 1 to the count for that bucket;
  
```
- Baskets:
- |                        | Item # | Count |
|------------------------|--------|-------|
| $B_1 = \{1, 2, 3\}$    | 1      | 5     |
| $B_2 = \{1, 3, 4\}$    | 2      | 5     |
| $B_3 = \{1, 2, 3, 5\}$ | 3      | 5     |
| $B_4 = \{2, 3, 5\}$    | 4      | 2     |
| $B_5 = \{2, 3\}$       | 5      | 4     |
- First Pass:
- ```

FOR (each basket) :
  FOR (each item in the basket) :
    add 1 to its count;
  FOR (each pair of items) :
    hash the pair to a bucket;
    add 1 to the count for that bucket;
  
```
- |                  | Bucket # | Count |
|------------------|----------|-------|
| (1,4), (2,3) → 0 | 0        | 6     |
| (1,5), (2,4) → 1 | 1        | 2     |
| (2,5), (3,4) → 2 | 2        | 4     |
| (1,2), (3,5) → 3 | 3        | 4     |
| (1,3), (4,5) → 4 | 4        | 5     |
- Hash Table:
- |                        | Bucket # | Count |
|------------------------|----------|-------|
| $B_1 = \{1, 2, 3\}$    | 0        | 6     |
| $B_2 = \{1, 3, 4\}$    | 1        | 2     |
| $B_3 = \{1, 2, 3, 5\}$ | 2        | 4     |
| $B_4 = \{2, 3, 5\}$    | 3        | 4     |
| $B_5 = \{2, 3\}$       | 4        | 5     |
- $h(1,2) = 3 \rightarrow \text{Bucket } 3$



# Frequent Itemset Mining & Association Rules (Part 2)

Some slides adapted from UIUC data mining course, and the data mining book by Kumar etc.

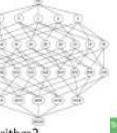
## Outline

- A different frequent itemset/pattern mining algorithm
- Sequential pattern mining

### Why Mining Frequent Patterns by Pattern Growth?

#### Apriori: A breadth-first search mining algorithm

- First find the complete set of frequent k-itemsets
- Then derive frequent (k+1)-itemset candidates
- Scan DB again to find true frequent (k+1)-itemsets
- Motivation for a different mining methodology
- Can we develop a depth-first search mining algorithm?
- For a frequent itemset p, can subsequent search be confined to only those transactions that contain p?
- Such thinking leads to a frequent pattern growth approach:
- FP Growth (Han et al "Mining Frequent Patterns without Candidate Generation," SIGMOD 2000)



### Example: Frequent Pattern Growth Algorithm

#### Mining FP w/o candidate generation.

#### Steps:

- Find the frequency of 1-itemset.
- Construct Ordered-Item set.
- Construct FP-tree (i.e., inserting ordered-item set).
- Reursively mine FP-tree and grow frequent patterns obtained so far
  - Construct Conditional database.
  - Construct conditional FP-tree and generate Frequent Patterns. Repeat the process on each newly created Conditional FP-tree for mining (longer) frequent patterns. (...until the resulting FP-tree is empty, or it contains only one path.)

### Example: Frequent Pattern Growth Algorithm

#### Transaction DB

#### 1-Itemset Frequency

Transaction ID	Name	Transaction ID	Name	Items	Frequency
T1	{f, a, c, d, g, l, m, p}	T5	{a, f, c, e, l, p, m, n}	a	4
T2	{a, b, c, f, l, m, o}	T6	{c, j, m, b, n}	b	4
T3	{b, f, h, j, o}	T7	{d, e, f, h}	c	5
T4	{b, c, k, s, p}	T8	{a, g, i, k, x, f}	f	6
				m	4
				p	3
				e	2
				g	1
				d	1
				l	1
				j	1
				n	1
				o	1
				s	1
				x	1

#### Step 1:

Find the frequency of 1-itemset.

#### Transaction DB

#### 1-Itemset Frequency

Transaction ID	Name	Transaction ID	Name	Items	Frequency
T1	{f, a, c, d, g, l, m, p}	T5	{a, f, c, e, l, p, m, n}	a	4
T2	{a, b, c, f, l, m, o}	T6	{c, j, m, b, n}	b	4
T3	{b, f, h, j, o}	T7	{d, e, f, h}	c	5
T4	{b, c, k, s, p}	T8	{a, g, i, k, x, f}	f	6
				m	4
				p	3
				e	2
				g	1
				d	1
				l	1
				j	1
				n	1
				o	1
				s	1
				x	1

#### Step 2:

Construct Ordered-Item set.

- (Let the minimum support threshold  $s = 3$ )
- Frequent Items: (a, b, c, f, m, p)

#### Transaction DB

Transaction ID	Items	Transaction ID	Items
T1	{f, a, c, d, g, l, m, p}	T5	{a, f, c, e, l, p, m, n}
T2	{a, b, c, f, l, m, o}	T6	{c, j, m, b, n}
T3	{b, f, h, j, o}	T7	{d, e, f, h}
T4	{b, c, k, s, p}	T8	{a, g, i, k, x, f}

Item	Frequency	Header
a	4	
b	4	
c	5	
f	6	
m	4	
p	3	

Header	Frequency	Header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	

#### Step 2:

Construct Ordered-Item set.

- Sort the frequent items in a descending order of their respective frequencies to form a frequent item set and header table.

Frequent Item set:  
 $L = \{f, b, c, 5, a, 4, b, 4, m, 4, p, 3\}$

Header Table references the occurrences of the frequent items in the FP-tree

#### Step 3:

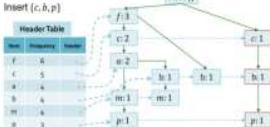
Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

#### Header Table

ID	Name	Ordered-Item Set
T1	{f, a, c, d, g, l, m, p}	{f, c, a, m, p}
T2	{a, b, c, f, l, m, o}	{c, a, b, m, n}
T3	{b, f, h, j, o}	{f, b}
T4	{b, c, k, s, p}	{c, b, p}

Root (1)



#### Step 3:

Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

#### Header Table

ID	Name	Ordered-Item Set
T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T6	{c, j, m, b, n}	{c, b, m}
T7	{d, e, f, h}	{f}
T8	{a, g, i, k, x, f}	{f, a}

Root (1)



#### Step 3:

Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

#### Header Table

ID	Name	Ordered-Item Set
T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T6	{c, j, m, b, n}	{c, b, m}
T7	{d, e, f, h}	{f}
T8	{a, g, i, k, x, f}	{f, a}

Root (1)



#### Step 3:

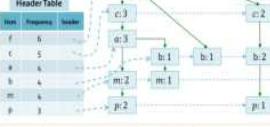
Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

#### Header Table

ID	Name	Ordered-Item Set
T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T6	{c, j, m, b, n}	{c, b, m}
T7	{d, e, f, h}	{f}
T8	{a, g, i, k, x, f}	{f, a}

Root (1)



#### Step 3:

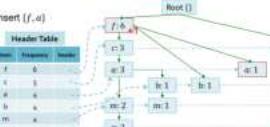
Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

#### Header Table

ID	Name	Ordered-Item Set
T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T6	{c, j, m, b, n}	{c, b, m}
T7	{d, e, f, h}	{f}
T8	{a, g, i, k, x, f}	{f, a}

Root (1)



#### Step 3:

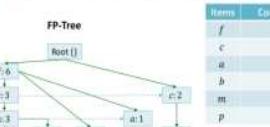
Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

#### Header Table

ID	Name	Ordered-Item Set
T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T6	{c, j, m, b, n}	{c, b, m}
T7	{d, e, f, h}	{f}
T8	{a, g, i, k, x, f}	{f, a}

Root (1)



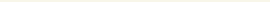
#### Step 4.a:

Construct Conditional Data Base for each frequent item. (Mining FP-Tree)

- Finding all prefix paths to the node (i.e., item).

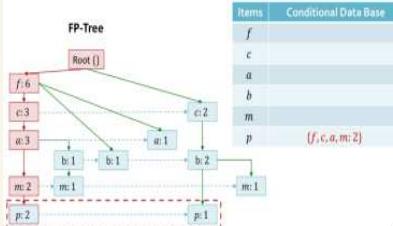
#### FP-Tree

Items	Conditional Data Base
f	
c	
a	
b	
m	
p	



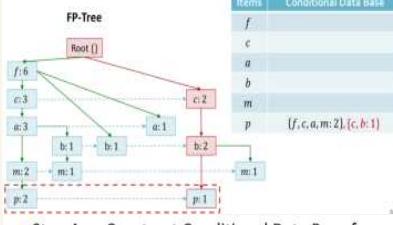
## Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)

- Finding all prefix paths to the node (i.e., item).



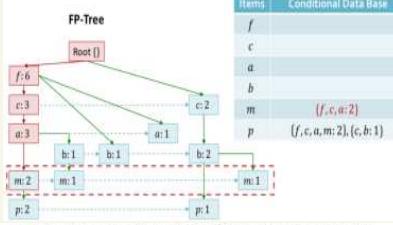
## Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)

- Finding all prefix paths to the node (i.e., item).



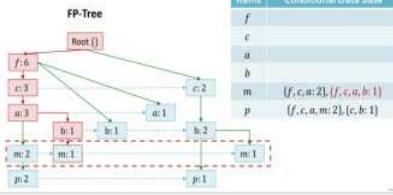
## Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)

- Finding all prefix paths to the node (i.e., item).



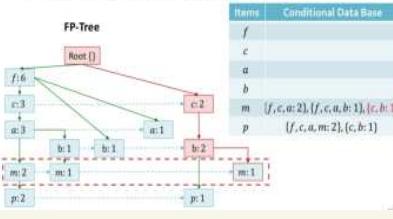
## Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)

- Finding all prefix paths to the node (i.e., item).



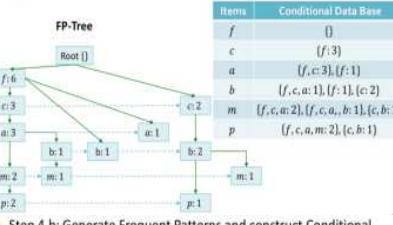
## Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)

- Finding all prefix paths to the node (i.e., item).



## Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)

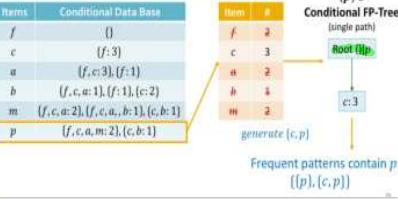
- Finding all prefix paths to the node (i.e., item).



## Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns

- Find frequent items from each (X)'s conditional DB. Each frequent item i and X will form a new frequent pattern (i, X)

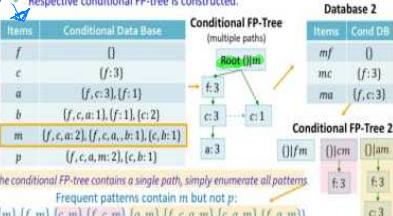
Respective conditional FP-tree is constructed.



## Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns

- Find frequent items from each (X)'s conditional DB. Each frequent item i and X will form a new frequent pattern (i, X)

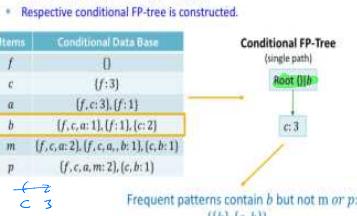
Respective conditional FP-tree is constructed.



## Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns

- Find frequent items from each (X)'s conditional DB. Each frequent item i and X will form a new frequent pattern (i, X)

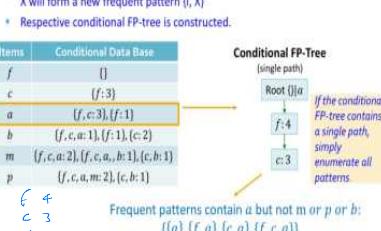
Respective conditional FP-tree is constructed.



## Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns

- Find frequent items from each (X)'s conditional DB. Each frequent item i and X will form a new frequent pattern (i, X)

Respective conditional FP-tree is constructed.



## Final results

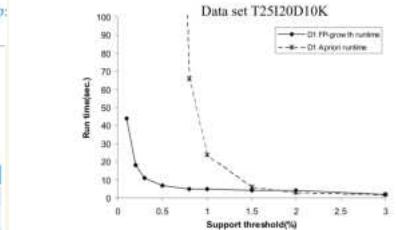
Frequent Pattern Subsets	Frequent Patterns
Patterns contain p.	{(p), {c, p}}
Patterns contain m but not p	{(m), (f, m), (c, m), (f, c, m), (a, m), (f, a, m), (c, a, m), (f, a, m)}
Patterns contain b but not p or m	{(b), (f, b), (c, b)}
Patterns contain a but not p or m or b	{(a), (f, a), (c, a), (f, c, a)}
Patterns contain c but not p, m, b, a	{(c), (f, c)}
Patterns contain f but not p, m, b, a, c	{(f)}

## Summary of Mining Frequent Patterns Using FP-tree

### General idea (divide-and-conquer)

- Recursively grow frequent patterns using FP-tree
- Frequent patterns can be partitioned into subsets according to L-order
- L-order = f: 6, c: 5, a: 4, b: 4, m: 4, p: 3
- Patterns containing p
- Patterns having m but no p
- Patterns having b but no m or p
- ...
- Patterns having c but no a nor b, m, p
- Pattern f

## FP-growth vs. Apriori: Scalability With the Support Threshold



## Why Is Frequent Pattern Growth Fast?

### Performance study shows

- FP-growth can be an order of magnitude faster than Apriori

### Reasons

- No candidate generation, no candidate test
- Use compact data structure
- Eliminate repeated database scan
- Basic operations are counting and FP-tree building

### Challenges

- When FP-tree cannot fit in memory

## Outline

### A different frequent itemset/pattern mining algorithm

### Sequential pattern mining

## Examples of Sequence

- Sequence of different transactions by a customer at an online store:  
 $\langle$  [Digital Camera, iPad] [memory card] [headphone, iPad cover]  $\rangle$
- Sequence of initiating events causing the nuclear accident at 3-mile Island:  
 $\langle$  [clogged resin] [outlet valve closure] [loss of feedwater] [condenser polisher outlet valve shut] [booster pumps trip] [main water pump trips] [main turbine trips] [reactor pressure increases]  $\rangle$
- Sequence of books checked out at a library:  
 $\langle$  [Fellowship of the Ring] [The Two Towers] [Return of the King]  $\rangle$

## Sequential Pattern Discovery: Examples

- In point-of-sale transaction sequences,
  - Computer Bookstore:  
 $\langle$  Intro\_To\_Visual\_C, C++\_Primer  $\rangle \rightarrow$   
 $\langle$  Perl\_for\_dummies, Tcl\_Tk  $\rangle$
  - Athletic Apparel Store:  
 $\langle$  Shoes, Racket, Racketball  $\rangle \rightarrow$  (Sports\_Jacket)
- Detecting Erroneous Sentences using Automatically Mined Sequential Patterns [1]
  - $\langle$  the, more, the, JJ [base form of adjective]  $\rangle$  in erroneous sentences

[4] Guohua Sun, Jianxin Liu, Bin Liang, Ming Zhou, Zhengping Xiong, John Lee, Chen-Yue Lin: Detecting Erroneous Sentences Using Automatically Mined Sequential Patterns. IJCS 2007.

## Examples of Sequence Data

Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc.
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc.
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequence	DNA sequences of a particular species	An element of the DNA sequence	Bases A, T, G, C

Element (Transaction)  Event (Item) 

## Sequence Data vs. Market-basket Data

Sequence Database:			Market- basket Data		
Customer	Date	Items bought	Events		
A	10	2, 3, 5	2, 3, 5		
A	20	1, 6	1, 6		
A	23	1	1		
B	11	4, 5, 6	4, 5, 6		
B	17	2	2		
B	21	1, 2, 7, 8	1, 2, 7, 8		
B	28	1, 6	1, 6		
C	14	1, 7, 8	1, 7, 8		

## Formal Definition of a Sequence

- A sequence is an ordered list of elements  
 $S = \langle e_1, e_2, e_3, \dots \rangle$
- Each element contains a collection of events (items)  
 $e_i = \{i_1, i_2, \dots, i_k\}$
- Length of a sequence,  $|S|$ , is given by the number of elements in the sequence
- A k-sequence is a sequence that contains k events (items)
  - $\langle a, b | a \rangle$  has a length of 2 and it is a 3-sequence

## Formal Definition of a Subsequence

- A sequence  $t: \langle a_1, a_2, \dots, a_n \rangle$  is contained in another sequence  $s: \langle b_1, b_2, \dots, b_m \rangle$  if there exist integers  $i_1 < i_2 < \dots < i_n$  such that  $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_n \subseteq b_{i_n}$
- Illustrative Example:  
 $s: \langle b_1, b_2, b_3, b_4, b_5 \rangle$   
 $t: \langle a_1, a_2, a_3 \rangle$   
 $t$  is a subsequence of  $s$  if  $a_1 \subseteq b_1, a_2 \subseteq b_2, a_3 \subseteq b_3$

Data sequence	Subsequence	Contain?
$\langle 2, 4   (3, 5, 6) \rangle$	$\langle 2   (3, 8) \rangle$	Yes
$\langle (1, 2)   3, 4 \rangle$	$\langle (1)   (2) \rangle$	No
$\langle (2, 4)   2, 5   4 \rangle$	$\langle (2)   (4) \rangle$	Yes
$\langle (2, 4)   2, 5   4   5 \rangle$	$\langle (2)   (4)   5 \rangle$	No
$\langle (2, 4)   2, 5   4   5 \rangle$	$\langle (2)   (5)   5 \rangle$	Yes
$\langle (2, 4)   2, 5   4   5 \rangle$	$\langle (2, 4, 5) \rangle$	No

## Sequential Pattern Mining: Definition

- The support of a subsequence w is defined as the fraction (or number) of data sequences that contain w. We use fraction in the rest slides
- A sequential pattern is a frequent subsequence (i.e., a subsequence whose support is  $\geq \text{minsup}$ )
- Given:
  - a database of sequences
  - a user-specified minimum support threshold,  $\text{minsup}$
- Task:
  - Find all subsequences with support  $\geq \text{minsup}$

## Sequential Pattern Mining: Example

Object	Timestamp	Events
A	1	1, 2, 4
A	2	2, 3
A	3	5
B	1	1, 2
B	2	2, 3, 4
C	1	1, 2
C	2	2, 3, 4
C	3	2, 4, 5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

minsup = 3

Examples of Frequent Subsequences:

- $\langle (1, 2) \rangle$   $\leq s=3$
- $\langle (2, 3) \rangle$   $\leq s=3$
- $\langle (2, 4) \rangle$   $\leq s=4$
- $\langle (3) | 5 \rangle$   $\leq s=4$
- $\langle (1) | (2) \rangle$   $\leq s=4$
- $\langle (2) | (2) \rangle$   $\leq s=3$
- $\langle (1) | (2, 3) \rangle$   $\leq s=3$
- $\langle (2) | (2, 3) \rangle$   $\leq s=3$
- $\langle (1, 2) | (2, 3) \rangle$   $\leq s=3$

## Extracting Sequential Patterns: Simple example

- Given 2 events: a, b
- Candidate 1-subsequences:  
 $\langle a | b \rangle, \langle b | a \rangle$
- Candidate 2-subsequences:  
 $\langle a | a \rangle, \langle a | b \rangle, \langle b | a \rangle, \langle b | b \rangle, \langle a, b \rangle$
- Candidate 3-subsequences:  
 $\langle a | a | a \rangle, \langle a | a | b \rangle, \langle a | b | a \rangle, \langle a | b | b \rangle, \langle b | a | a \rangle, \langle b | a | b \rangle, \langle b | b | a \rangle, \langle b | b | b \rangle, \langle a, b | a \rangle, \langle a, b | b \rangle, \langle a | a, b \rangle, \langle a | b, b \rangle$



## Generalized Sequential Pattern (GSP)

- Step 1: Make the first pass over the sequence database D to yield all the 1-element frequent sequences
- Step 2: Repeat until no new frequent sequences are found
  - Candidate Generation:
    - Merge all frequent subsequences found in the (k-1)th pass to generate candidate sequences that contain k-items
  - Candidate Pruning (Apriori):
    - Prune candidate k-sequences that contain infrequent (k-1)-subsequences
  - Support Counting:
    - Make a new pass over the sequence database D to find the support for these candidate sequences
  - Candidate Elimination:
    - Eliminate candidate k-sequences whose actual support is less than minsup
- General case (k>2):
- Merging two frequent 1-sequences  $\langle i_1 \rangle$  and  $\langle i_2 \rangle$  will produce the following candidate 2-sequences:  $\langle \{i_1\} | \{i_2\} \rangle, \langle \{i_1\} | \{i_2\} \rangle, \langle \{i_2\} | \{i_1\} \rangle$  and  $\langle \{i_1, i_2\} \rangle$ . (Note:  $\langle i_1 \rangle$  can be merged with itself to produce:  $\langle \{i_1\} | \{i_1\} \rangle$ )
- Base case (k=2):
- Merging two frequent 1-sequences  $\langle i_1 \rangle$  and  $\langle i_2 \rangle$  will produce the following candidate 2-sequences:  $\langle \{i_1\} | \{i_2\} \rangle, \langle \{i_1\} | \{i_2\} \rangle, \langle \{i_2\} | \{i_1\} \rangle$  and  $\langle \{i_1, i_2\} \rangle$ .

## Candidate Generation

### GSP: Apriori-Based Sequential Pattern Mining

- Initial candidates: All 0-singleton sequences
- Scan DB once: count support for each candidate
- Base case (k=2): Generate candidate 2-subsequences

minsup = 2				
$\langle a \rangle$	$\langle ab \rangle$	$\langle ba \rangle$	$\langle abc \rangle$	$\langle acb \rangle$
$\langle b \rangle$	$\langle ab \rangle$	$\langle ba \rangle$	$\langle bac \rangle$	$\langle bac \rangle$
$\langle c \rangle$	$\langle abc \rangle$	$\langle acb \rangle$	$\langle bac \rangle$	$\langle cab \rangle$
$\langle ab \rangle$	$\langle abc \rangle$	$\langle acb \rangle$	$\langle bac \rangle$	$\langle cab \rangle$
$\langle ba \rangle$	$\langle abc \rangle$	$\langle acb \rangle$	$\langle bac \rangle$	$\langle cab \rangle$
$\langle abc \rangle$				
$\langle ab   c \rangle$				
$\langle ba   c \rangle$				
$\langle abc   \cdot \rangle$				
$\langle ab   bc \rangle$				
$\langle ba   cb \rangle$				
$\langle abc   bc \rangle$				
$\langle ab   bc   c \rangle$				

- Without Apriori pruning: (8 singletons)  $8^2 \times 8^2 / 2 = 92$  candidates
- With pruning: candidates:  $36 + 15 = 51$

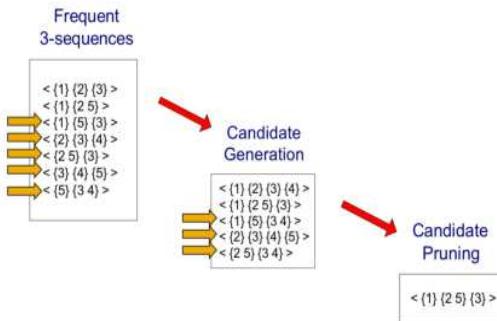
## Extracting Sequential Patterns

- Given n events:  $i_1, i_2, i_3, \dots, i_n$
- Candidate 1-subsequences:  
 $\langle i_1 \rangle, \langle i_2 \rangle, \langle i_3 \rangle, \dots, \langle i_n \rangle$
- Candidate 2-subsequences:  
 $\langle i_1, i_2 \rangle, \langle i_1, i_3 \rangle, \dots, \langle i_1, i_n \rangle$   
 $\langle i_2, i_3 \rangle, \langle i_2, i_4 \rangle, \dots, \langle i_2, i_n \rangle$   
 $\langle i_3, i_4 \rangle, \langle i_3, i_5 \rangle, \dots, \langle i_3, i_n \rangle$   
 $\dots$
- Candidate 3-subsequences:  
 $\langle i_1, i_2, i_3 \rangle, \langle i_1, i_2, i_4 \rangle, \dots$   
 $\langle i_1, i_2, i_3 \rangle, \langle i_1, i_2, i_4 \rangle, \dots$   
 $\langle i_1, i_2, i_3 \rangle, \langle i_1, i_2, i_4 \rangle, \dots$   
 $\langle i_1, i_2, i_3 \rangle, \langle i_1, i_2, i_4 \rangle, \dots$

## Candidate Generation Examples

- Merging  $w_1 = \langle \{1\} \{2\} \{3\} \{4\} \{6\} \rangle$  and  $w_2 = \langle \{2\} \{3\} \{4\} \{6\} \{5\} \rangle$  produces the **candidate sequence**  $\langle \{1\} \{2\} \{3\} \{4\} \{5\} \rangle$  because the last element of  $w_2$  has only one event
- Merging  $w_1 = \langle \{1\} \{2\} \{3\} \{4\} \rangle$  and  $w_2 = \langle \{2\} \{3\} \{4\} \{5\} \rangle$  produces the **candidate sequence**  $\langle \{1\} \{2\} \{3\} \{4\} \{5\} \rangle$  because the last element in  $w_2$  has more than one event
- Merging  $w_1 = \langle \{1\} \{2\} \{3\} \rangle$  and  $w_2 = \langle \{2\} \{3\} \{4\} \rangle$  produces the **candidate sequence**  $\langle \{1\} \{2\} \{3\} \{4\} \rangle$  because the last element in  $w_2$  has more than one event
- We do **not** merge the sequences  $w_1 = \langle \{1\} \{2\} \{6\} \{4\} \rangle$  and  $w_2 = \langle \{1\} \{2\} \{4\} \{5\} \rangle$  to produce the candidate  $\langle \{1\} \{2\} \{6\} \{4\} \{5\} \rangle$  because if the latter is a viable candidate, then it can be obtained by merging  $w_1$  with  $\langle \{2\} \{6\} \{4\} \{5\} \rangle$

## GSP Example



## Constraints on sequence patterns

- We can impose different constraints for two items in a sequence pattern. For example
  - a max-gap for two items
  - A min-gap for two items (e.g., contiguous)
- Need to check these constraints when deciding if a pattern is contained by a data sequence
- One solution: Mine sequential patterns without constraints
  - Postprocess the discovered patterns

## Summary

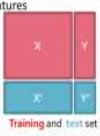
- FP-tree algorithms
  - Understand the algorithm
- Sequential pattern mining
  - Understand the algorithm

## Classification (Part 1)

### Supervised Learning

#### Data is labeled:

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



#### Where you can be:

- Real number:** Regression
- Categorical:** Classification
- Complex object:  
Ranking of items, etc.

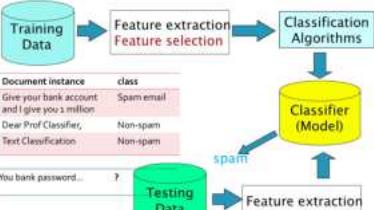
#### Task of prediction:

estimate a function  $f(x)$  so that  $y = f(x)$   
also works on unseen  $X, Y$

### Examples of Classification Task

Task	Attribute set, $x$	Class label, $y$
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from x-rays or MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

### Classification Framework



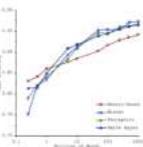
### Classification Methods

- Decision Tree based Methods
- Rule-based Methods
- Memory-based or Instance-based Methods
- Naïve Bayes Classifiers
- Support Vector Machines
- Neural Networks , Deep Neural Nets  
(<https://www.deeplearningbook.org/> for your reference)
- Ensemble Classification

### Why large scale ML?

#### Brawn or Brains?

- In 2001, Microsoft researchers ran a test to evaluate 4 of different approaches to ML-based language translation



#### Findings:

- Size of the dataset** used to train the model **mattered more** than the model itself
- As the dataset grew large, performance difference between the models became small

Bainbridge, M. and Brill, E. (2001). "Scaling to Very Very Large Corpora for Natural Language Disambiguation".

## Why large scale ML?

### The Unreasonable Effectiveness of Data

- In 2017, Google revisited the same type of experiment with the latest Deep Learning models in computer vision

### Findings:

- Performance increases logarithmically based on volume of training data
- Complexity of modern ML models (i.e., deep neural nets) allows for even further performance gains

### Large datasets + large ML models => amazing results!!

"Reinventing Unreasonable Effectiveness of Data in Deep Learning Era". <https://arxiv.org/pdf/1705.07115.pdf>

## Why Worry About Non-Deep Models?

### A few reasons why this is important:

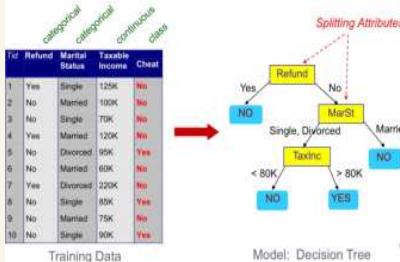
- Classical tasks in NLP and Vision are getting commoditized (you take pretrained model and fine tune it), but there are many other unique ML tasks.
- Deep models are often hard to scale and require lots and lots of data. Traditional models allow you to encode prior knowledge better and give you more control.
- Personally, if I am working on a well understood problem I'd use deep learning, but if I am the first person to work on a new problem/classifier I'd use techniques we'll discuss here.

### Outline

#### Classification Techniques

- Decision Tree
- Classification based on association rules (CBA)
- Ensemble classifier
- Overfitting
- Classification evaluation

## Example of a Decision Tree



## Another Example of Decision Tree

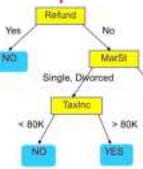


## Apply Model to Test Data

Start from the root of tree.

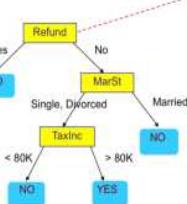
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



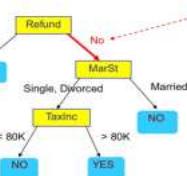
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



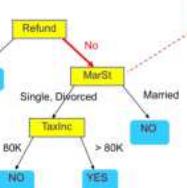
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



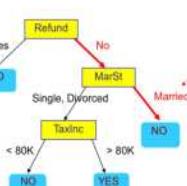
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



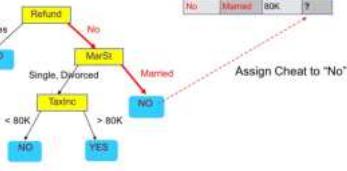
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

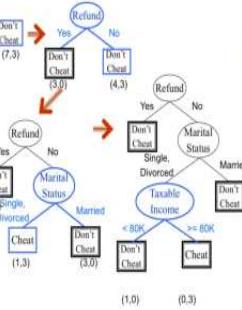
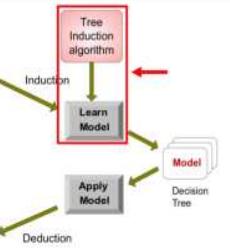


Assign Cheat to "No"

# Decision Tree Classification Task

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

Tid	Refund	Age	Marital Status	Income	Cheat
11	No	Young	Divorced	120K	No
12	No	Medium	Divorced	80K	No
13	Yes	Large	Married	110K	Y
14	No	Small	Divorced	85K	No
15	No	Large	Divorced	80K	N



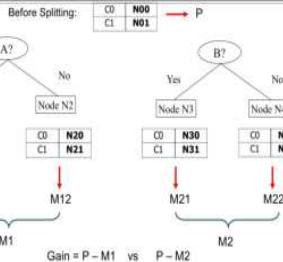
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Finding the Best Split

- Compute impurity measure ( $P$ ) before splitting
- Compute impurity measure ( $M$ ) after splitting
  - Compute impurity measure of each child node
  - $M$  is the weighted impurity of child nodes
- Choose the attribute and the split that produces the highest gain. Note an attribute may have multiple ways to split

$$\text{Gain} = P - M$$

or equivalently, lowest impurity measure after splitting ( $M$ )



## Design Issues of Decision Tree Induction

- The idea is to use greedy strategy.
- Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - Which attribute to split and how to determine the best split?
  - Determine when to stop splitting

## Decision Tree Induction

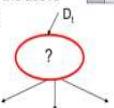
- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT

## General Structure of Hunt's Algorithm

Let  $D_t$  be the set of training records that reach a node  $t$ .

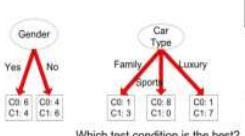
General Procedure:

- If  $D_t$  contains records that belong to the same class  $y_t$ , then  $t$  is a leaf node labeled as  $y_t$ .
- If  $D_t$  contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the above procedure to each subset.



## How to determine the Best Split

Before Splitting: 10 records of class 0, 10 records of class 1



Which test condition is the best?

### Greedy approach:

- Nodes with homogeneous (purer) class distribution are preferred
- Need a measure of node impurity:

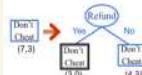
C0: 5  
C1: 5

Non-homogeneous,  
High degree of impurity

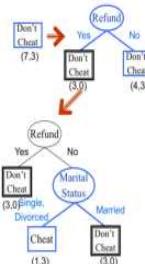
C0: 9  
C1: 1

Homogeneous,  
Low degree of impurity

## Hunt's Algorithm



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	80K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



## Computing GINI Measure

### Gini Index

$$\text{Gini Index} = 1 - \sum_{i=0}^{c-1} p_i(t)^2 \quad \text{Where } p_i(t) \text{ is the probability of class } i \text{ at node } t, \text{ and } c \text{ is the total number of classes}$$

### Entropy

$$\text{Entropy} = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

### Misclassification error

$$\text{Classification error} = 1 - \max[p_i(t)]$$

## Computing GINI Measure

### Gini Index for a given node $t$

$$\text{Gini Index} = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where  $p_i(t)$  is the probability of class  $i$  at node  $t$ , and  $c$  is the total number of classes

- Maximum of  $1 - 1/c$  when records are equally distributed among all classes, implying the least beneficial situation for classification
- Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification

C1	0	Gini = 0.000
C1	1	Gini = 0.278

C1	2	Gini = 0.500
C1	3	Gini = 0.500

## Computing Gini Index of a Single Node

$$\text{Gini Index} = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

For 2-class problem

C1	0	P <sub>C1 t</sub> (t) = 0.8 = 0	P <sub>C2 t</sub> (t) = 0.2 = 1
C2	6	Gini = 1 - P <sub>C1 t</sub> (t) <sup>2</sup> - P <sub>C2 t</sub> (t) <sup>2</sup> = 1 - 0 - 1 = 0	

C1	1	P <sub>C1 t</sub> (t) = 1/6	P <sub>C2 t</sub> (t) = 5/6
C2	5	Gini = 1 - (1/6) <sup>2</sup> - (5/6) <sup>2</sup> = 0.278	

C1	2	P <sub>C1 t</sub> (t) = 2/6	P <sub>C2 t</sub> (t) = 4/6
C2	4	Gini = 1 - (2/6) <sup>2</sup> - (4/6) <sup>2</sup> = 0.444	

## Computing Gini Index for a Collection of Nodes

- In general, when a node  $p$  is split into  $k$  partitions (children)

$$G_{\text{int},\text{split}} = \sum_{i=1}^k \frac{n_i}{n} G_{\text{int}(i)}$$

where,  $n_i$  = number of records at child  $i$ ,  
 $n$  = number of records at parent node  $p$ .

## Binary Attributes: Computing GINI Index

- Binary Attributes:** Splits into two partitions (child nodes)
- Effect of Weighing partitions:**
  - Larger and purer partitions are sought



Parent	
C1	7
C2	5
Gini =	0.486

Gini(N1) =  $1 - (5/6)^2 - (1/6)^2 = 0.278$   
Gini(N2) =  $1 - (2/6)^2 - (4/6)^2 = 0.361$   
Gini =  $0.486 - 0.361 = 0.125$

N1	N2
C1 5	2
C2 1	4
Gini=0.361	

## Categorical Attributes: Computing GINI Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multis-way split

CarType		
Family	Sports	Luxury
C1 1	1	1
C2 3	0	7
Gini 0.163		

Two-way split  
(find best partition of values)

CarType		
(Sports)	Luxury	Family
C1 9	1	1
C2 0	2	20
Gini 0.466		

Which of these is the best?

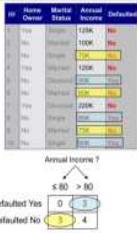
## Continuous attributes: Computing GINI Index

### Two ways of handling continuous attributes

- Discretization** to form an (ordinal) categorical attribute, then use the method in last slide
- Discretization will be introduced in later week
- Binary Decision:**  $(A < v) \text{ or } (A \geq v)$ 
  - consider all possible splits and finds the best cut
  - can be more computationally intensive

### Use Binary Decisions based on one value

- Choices for the splitting value
  - Each distinct value is a possible splitting value
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions,  $A \leq v$  and  $A > v$
- Simple method to choose best  $v$ 
  - For each  $v$ , scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.
- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index



# Considerations in CAR mining

## Set different minimum class supports

- Deal with imbalanced class distribution, e.g., some class is rare, 98% negative and 2% positive.
- We can set the minsup(positive) = 0.2% and minsup(negative) = 2%.
- Rule pruning** may be performed.
- Pruning off less useful rules.
- E.g., A rule  $a, b, c \rightarrow \text{class1}$  is pruned if its confidence is 60%, but a subset rule  $a, b \rightarrow \text{class1}$  has a confidence 70%.
- $a, b \rightarrow \text{class1}$  must have a higher support than  $a, b, c \rightarrow \text{class1}$

## Example of pruning rules

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

$r_1: ((A, e), (B, p), 3), ((C, y), 2)$   
Confidence = 2/3 = 67%  
Error rate of 33%

$r_1': ((A, e), 4), ((C, y), 3)$   
Confidence = 1/4 = 25%  
Error rate of 25%

$r_1'$  is a subset of  $r_1$  and  
 $r_1'$  has a higher confidence (lower Error rate) than  $r_1$

=  
Prune  $r_1$

## Classification and Association Rule Mining

- There are many ways to build classifiers using CARs. Several existing systems available.
- Strongest rules:** After CARs are mined, do nothing.
  - For each test case, we simply choose the most confident rule that covers the test case to classify it.
  - Or, using a combination of rules.
- Selecting a subset of Rules to build CBA**
  - To be explained .
  - Reference: Bing Liu, Wynne Hsu, Yiming Ma: Integrating Classification and Association Rule Mining. KDD 1998: 80-86

## CBA Algorithm

A CBA classifier  $L$  is of the form:  
 $L = \langle r_1, r_2, \dots, r_k, \text{default-class} \rangle$

CBA algorithm consists of two parts:

- Rule Generator (CBA-RG):
  - Generate CARs for each class.
  - A simple extension of Apriori algorithm. We have a tutorial question on it.
- Classifier builder (CBA-CB): select a subset of rules

## CBA: Rules are sorted first

**Definition:** Given two rules,  $r_i$  and  $r_j$ ,  $r_i > r_j$  (also called  $r_i$  precedes  $r_j$  or  $r_i$  has a higher precedence than  $r_j$ ) if
 

- the confidence of  $r_i$  is greater than that of  $r_j$ ,
- their confidences are the same, but the support of  $r_i$  is greater than that of  $r_j$ .

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	m
g	w	n
g	w	n
e	p	n
f	q	n

## Example: CBA-CB (M1)

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

### CARs after pruning:

$r_1: A = e \rightarrow y$	sup = 30%	conf = 75%
$r_2: A = g \rightarrow n$	sup = 30%	conf = 60%
$r_3: B = q \rightarrow y$	sup = 20%	conf = 66%
$r_4: B = q \rightarrow y$	sup = 30%	conf = 60%
$r_5: B = w \rightarrow n$	sup = 20%	conf = 100%
$r_6: A = g, B = q \rightarrow y$	sup = 20%	conf = 66%

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

### CARs after pruning:

$r_1: A = e \rightarrow y$	sup = 30%	conf = 75%
$r_2: A = g \rightarrow n$	sup = 30%	conf = 60%
$r_3: B = p \rightarrow y$	sup = 20%	conf = 66%
$r_4: B = p \rightarrow y$	sup = 30%	conf = 60%
$r_5: B = w \rightarrow n$	sup = 20%	conf = 100%
$r_6: A = g, B = q \rightarrow y$	sup = 20%	conf = 66%

### Sort CARs (>Precedence):

= [http://www.cs.psu.edu/~mehrotra/courses/CS5110/Notes/Ch10.pdf#page=10](#)

$r_5 > r_1 > r_3 > r_6 > r_2 > r_4$

## CBA-CB: M1 Algorithm

### CBA (R,D) Algorithm: R CARs, D training data

- $R = \text{sort}(R)$ ; // sorting is done based on precedence definition
- $\text{RuleList} = \emptyset$
- for each rule  $r$  in  $R$  in sequence do
  - If  $D$  is not  $\emptyset$  AND  $r$  classifies at least one tuple in  $D$  correctly then
    - delete all the training tuples covered by  $r$  from  $D$ ;
    - Add  $r$  at the end of  $\text{RuleList}$ ;
    - Add the majority class in the uncovered training data as the default class at the end of  $\text{RuleList}$
    - compute the total number of errors of  $\text{RuleList}$ ;
  - Find the first rule  $p$  in  $\text{RuleList}$  with the lowest total number of errors and drop all the rules after  $p$  in  $\text{RuleList}$ ; add default class;

### Example: CBA-CB (M1)

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

### Sorted CARs (>Precedence):

$r_5 > r_1 > r_3 > r_6 > r_2 > r_4$

### Algorithm Line 3-8:

$r_5: B = w \rightarrow n$

$r_1: A = e \rightarrow y$

$r_3: B = p \rightarrow y$

$r_6: A = g \rightarrow n$

$r_2: A = e \rightarrow y$

$r_4: B = q \rightarrow y$

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

### Sorted CARs (>Precedence):

$r_5 > r_1 > r_3 > r_6 > r_2 > r_4$

### Algorithm Line 3-8 :

$r_5: B = w \rightarrow n$

$r_1: A = e \rightarrow y$

$r_3: B = p \rightarrow y$

$r_6: A = g \rightarrow n$

$r_2: A = e \rightarrow y$

$r_4: B = q \rightarrow y$

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

### Sorted CARs (>Precedence):

$r_5 > r_1 > r_3 > r_6 > r_2 > r_4$

### Algorithm Line 3-8 :

$r_5: B = w \rightarrow n$

$r_1: A = e \rightarrow y$

$r_3: B = p \rightarrow y$

$r_6: A = g \rightarrow n$

$r_2: A = e \rightarrow y$

$r_4: B = q \rightarrow y$

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

### Sorted CARs (>Precedence):

$r_5 > r_1 > r_3 > r_6 > r_2 > r_4$

### Algorithm Line 3-8 :

$r_5: B = w \rightarrow n$

$r_1: A = e \rightarrow y$

$r_3: B = p \rightarrow y$

$r_6: A = g \rightarrow n$

$r_2: A = e \rightarrow y$

$r_4: B = q \rightarrow y$

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

### Sorted CARs (>Precedence):

$r_5 > r_1 > r_3 > r_6 > r_2 > r_4$

### Algorithm Line 3-8 :

$r_5: B = w \rightarrow n$

$r_1: A = e \rightarrow y$

$r_3: B = p \rightarrow y$

$r_6: A = g \rightarrow n$

$r_2: A = e \rightarrow y$

$r_4: B = q \rightarrow y$

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

### Sorted CARs (>Precedence):

$r_5 > r_1 > r_3 > r_6 > r_2 > r_4$

### Algorithm Line 3-8 :

$r_5: B = w \rightarrow n$

$r_1: A = e \rightarrow y$

$r_3: B = p \rightarrow y$

$r_6: A = g \rightarrow n$

$r_2: A = e \rightarrow y$

$r_4: B = q \rightarrow y$

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

### Sorted CARs (>Precedence):

$r_5 > r_1 > r_3 > r_6 > r_2 > r_4$

### Algorithm Line 3-8 :

$r_5: B = w \rightarrow n$

$r_1: A = e \rightarrow y$

$r_3: B = p \rightarrow y$

$r_6: A = g \rightarrow n$

$r_2: A = e \rightarrow y$

$r_4: B = q \rightarrow y$

### Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e</td		

## Classification (Part 2)

### Outline

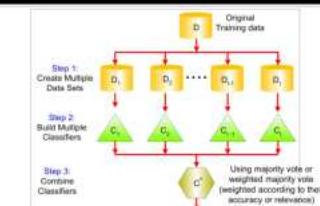
#### Classification Techniques

- Decision Tree
- Classification based on association rules
- Ensemble classifier
- Oversampling
- Classification evaluation

### Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers, such as majority of voting
- Assumption:
  - Individual classifiers (voters) could be lousy (stupid), but the aggregate (electorate) can usually classify (decide) correctly.

### General Idea



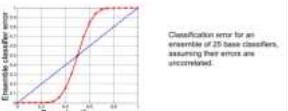
### Example: Why Do Ensemble Methods Work?

- Suppose there are 25 base classifiers
- Each classifier has error rate,  $\epsilon = 0.35$
- Majority vote of classifiers used for classification
- If all classifiers are identical:
  - Error rate of ensemble =  $\epsilon (0.35)$
- If all classifiers are independent (errors are uncorrelated):
  - Error rate of ensemble = probability of having more than half of base classifiers being wrong

$$e_{\text{ensemble}} = \sum_{i=1}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06$$

### Necessary Conditions for Ensemble Methods

- Ensemble Methods work better than a single base classifier if:
  - All base classifiers are independent of each other
  - All base classifiers perform better than random guessing (error < 0.5 for binary classification)



### Examples of Ensemble Methods

#### How to generate an ensemble of classifiers?

- Bagging
- Boosting

### Bagging (Bootstrap AGGREGATING)

- Bootstrap sampling: sampling with replacement

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

- Build classifier on each bootstrap sample

## Bagging Algorithm

### Algorithm 4.5 Bagging algorithm.

- Let  $k$  be the number of bootstrap samples.
- for  $i = 1$  to  $k$  do
  - Create a bootstrap sample of size  $N$ ,  $D_i$ .
  - Train a base classifier  $C_i$  on the bootstrap sample  $D_i$ .
- end for
- $C^*(x) = \operatorname{argmax}_y \delta(C_i(x) = y)$ .
- { $\delta(\cdot) = 1$  if its argument is true and 0 otherwise.}

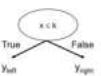
### Bagging Example

- Consider 1-dimensional data set:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	1	-1	-1	-1	-1	1	1

- Classifier is a decision stump (decision tree of size 1)

- Decision rule:  $x \leq k$  versus  $x > k$
- Split point  $k$  is chosen based on entropy



### Bagging Example

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	1	-1	-1	-1	-1	1	1

Bagging Round 1:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.5 \rightarrow y = 1$   
 $x > 0.5 \rightarrow y = -1$

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	1	1	1	1

$x \leq 0.7 \rightarrow y = 1$   
 $x > 0.7 \rightarrow y = -1$

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	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 4:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
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	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	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	0.8	0.9	1
y	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	0.2	0.3	0.4	0.5	0.6	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 8:

x	0.1	0.2	0.3	0.4	0.5	0.6	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	0.2	0.3	0.4	0.5	0.6	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 10:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	1	1	1	1

$x \leq 0.5 \rightarrow y = 1$   
 $x > 0.5 \rightarrow y = -1$

### Summary of Trained Decision Stumps:

Round	Split Point	Left Class	Right Class
1	0.35	-1	1
2	0.7	1	-1
3	0.35	-1	1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

- Use majority vote (sign of sum of predictions) to determine class of ensemble classifier

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	2	2	2	2
Sign	1	1	1	-1	-1	-1	1	1	1	1

## Improvement of Bagging: Random Forest

### Train a Bagged Decision Tree

- But use a modified tree learning algorithm that selects (at each candidate split) a **random subset of features**
  - If we have  $d$  features, consider  $\sqrt{d}$  random features

### This is called: Feature bagging

- Benefit:** Breaks correlation between trees
  - If one feature is very strong predictor, then every tree will select it, causing trees to be correlated.

### Random Forests achieve state-of-the-art results in many classification problems!

## Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records

- Initially, all N records are assigned equal weights

- Unlike bagging, sampling weights may change at the end of boosting round

### Successful algorithms

- Example 1: adaBoost (some slides for your information or self-study)

- Example 2: Gradient Boosted Decision Trees

### Records that are wrongly classified will have their weights increased

### Records that are correctly classified will have their weights decreased

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

- Example 4 is hard to classify

- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

## Build Decision Trees with AdaBoost

Suppose we have training data  $\{(x_i, y_i)\}_{i=1}^N$ ,  $y_i \in \{1, -1\}$

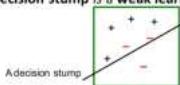
- Initialize equal weights for all observations

- At each iteration t:
  - Train a stump  $G_t$  using data weighted by  $w_t$
  - Compute the misclassification error adjusted by  $w_t$
  - Compute the weight of the current tree
  - Reweight each observation based on prediction accuracy

# AdaBoost: Weak learner

## 1. build Decision "stumps":

- 1-level decision tree
- A decision boundary based on one feature
  - E.g.: If someone is not a smoker, then predict them to live past 80 years old
- Building blocks of AdaBoost algorithm
- Decision stump is a weak learner**



**Boosting theory:**  
Weak learners have >50% accuracy then we can learn a perfect classifier.

## Update Step

### 2. Calculate the weighted misclassification error

$$err_t = \frac{\sum_{i=1}^N w_i I(y_i \neq G_t(x_i))}{\sum_{i=1}^N w_i}$$

### 3. Use the error score to weight the current tree in the final classifier:

$$\alpha_t = \log \left( \frac{1 - err_t}{err_t} \right)$$

A classifier with 50% accuracy is given a weight of zero;

### 4. Use misclassification error and tree weight to reweight the training data:

$$w_i \leftarrow w_i \exp[\alpha_t (y_i - G_t(x_i))]$$

Harder to classify training instances get higher weight

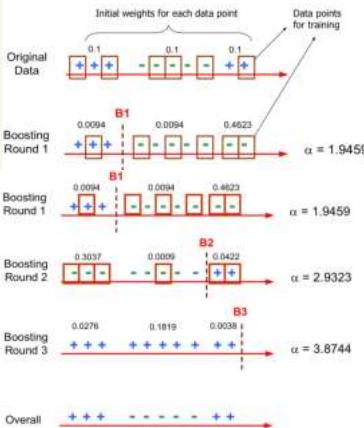
## Final Prediction

- Final prediction is a weighted sum of the predictions from each stump:

$$G(x) = \text{sign} \left[ \sum_{t=1}^T \alpha_t G_t(x) \right]$$

- More accurate trees are weighted higher in the final model

## Illustrating AdaBoost



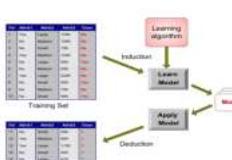
## Outline

### Classification Techniques

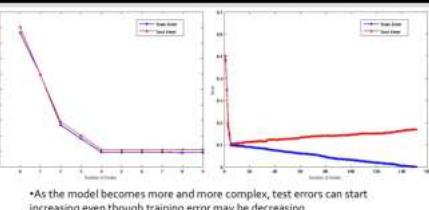
- Decision Tree
- Classification based on association rules
- Ensemble classifier
- Overfitting
- Classification evaluation

# Classification Errors

- Training errors:** Errors committed on the training set
- Test errors:** Errors committed on the test set
- Generalization errors:** Expected error of a model over random selection of records from same distribution



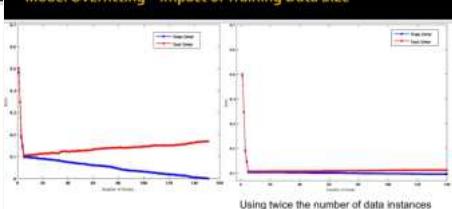
# Model Underfitting and Overfitting



**Underfitting:** when model is too simple, both training and test errors are large

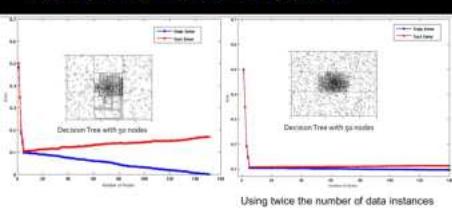
**Overfitting:** when model is too complex, training error is small but test error is large

### Model Overfitting – Impact of Training Data Size



- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

### Model Overfitting – Impact of Training Data Size



- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

### Reasons for Model Overfitting

- Not enough training data
- High model complexity
  - Multiple Comparison Procedure

### Notes on Overfitting

Use decision trees as an example:

- Overfitting results in decision trees that are **more complex** than necessary
- Training error does not provide a good estimate of how well the tree will perform on previously unseen records
- Need ways for estimating generalization errors

### Model Selection

- Performed during model building
- Purpose is to ensure that model is not overly complex (to avoid overfitting)
- Need to estimate generalization error
  - Using Validation Set
  - Incorporating Model Complexity (e.g., number of leaf nodes in decision tree) in model training
    - Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

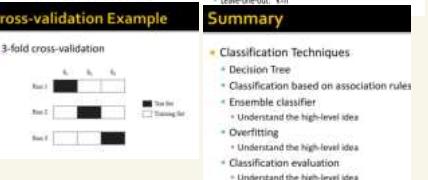
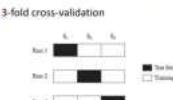
### Using Validation Set

- Divide training data into two parts:

- training set:
  - use for model building
- validation set:
  - use for estimating generalization error
  - Note: validation set is not the same as test set

### Cross-validation Example

- 3-fold cross-validation



### Classification Model Evaluation

- Purpose:
  - To estimate performance of classifier on previously unseen data (test set)
- Holdout:
  - Reserve k% for training and (100-k)% for testing
  - Random subsampling: repeated holdout
  - Cross-validation
    - Partition data into k disjoint subsets
    - k-fold: train on k-1 partitions, test on the remaining one (Leave-one-out, k-fold)
- Summary

- Classification Techniques
  - Decision Tree
  - Classification based on association rules
  - Ensemble classifier
    - Understand the high-level idea
  - Overfitting
    - Understand the high-level idea
  - Classification evaluation
    - Understand the high-level idea

# Introduction to Recommendation Systems

## Outline

- ▷ WHERE can we find recommendation systems?
- ▷ WHY do we need recommendation systems?
- ▷ WHAT is recommendation?
- ▷ HOW do we make recommendations?
- ▷ Different types of recommendation systems
- ▷ Challenges encountered by recommendation systems

### 1.

## WHERE can we find recommendation systems?

### Recommendation Systems – WHERE?



#### ▷ E.g., Netflix

- Netflix might try to recommend a show which is very popular based on your current location (E.g., No. 10 in Singapore Today)
- ▷ E.g., Netflix
  - Netflix might try to recommend the Top 10 shows based on your current location
  - However, note that these are **not** personalized recommendations
    - Based on other users located in Singapore
    - Everyone in Singapore gets the same recommendations

#### ▷ E.g., Google Maps

- Given your current location, Google Maps might recommend popular locations *near* you

#### ▷ E.g., Netflix

- Shows are often recommended based on your viewing history (i.e., shows that you have watched previously)

#### ▷ E.g., Amazon

- Let's assume that we are viewing some book in Amazon (e.g., Artificial Intelligence: A Modern Approach)

#### ▷ E.g., Amazon

- On the same page, Amazon would try to recommend other items (i.e., books) that are frequently bought together  
*that were bought by similar customer*

#### ▷ Recommendation systems can be found everywhere!

- Movies, Books, Music, Points of Interests, ...

#### ▷ (Literally) an indispensable part of our daily lives

### 2.

## WHY do we need recommendation systems?

1. Large catalogue
  - w/ up to millions of items
2. Beneficial to both the end users & the businesses

### Large Catalogue

▷ "Amazon has an inventory of about 12 million items across all its categories and services!"

▷ Let's say... I would like to get a new chair (but I am not entirely sure what kind of a chair would be good)

- There could be thousands of chairs being sold
- Different types of chairs: Office, Dining, ...
- Different colours, sizes, prices, ...
- It could be very challenging and tedious to find something ideal!
- Recommender systems can help to narrow down the options effectively ☺

## Large (and expanding) Catalogue ☺

- ▷ Platforms such as YouTube, TikTok, etc. are based on User Generated Content (UGC)
- ▷ In contrast to platforms such as Netflix, the catalogue is **expanding rapidly!**



### Beneficial to End Users ☺



#### ▷ E.g., Google Maps

- The 'explore' feature assists users in finding **nearby** points of interests
  - Users might not be aware of these locations before this!

#### ▷ E.g., Netflix

- By leveraging your viewing **history**, Netflix could recommend other shows which you are most likely to watch...
- Similarly, you might not be aware of such shows prior to this!

#### ▷ A good recommender system would be akin to having a friend who **knows you even better than yourself**

- Assists users in finding **items of interest**
- Introduces new items to users (*Novelty*)
- Presents unexpected items to users (*Serendipity*)

### Beneficial to Businesses ☺

▷ Helps item providers (e.g., sellers, content creators, etc.) deliver their products to its intended audience

▷ Improves customer satisfaction

- Less time spent on browsing / searching
- Caters to the preferences of different users

▷ The underlying recommendation system could be the **key difference** between...

- 2 platforms having the same catalogue
- 2 platforms with the same type of products

## 3. WHAT is recommendation?

#### ▷ Where?

- Everywhere!
- Part and parcel of our daily lives

#### ▷ Why?

- Large (and perhaps expanding) catalogues
- Good for both end users & businesses

#### ▷ What? → Formalizing the recommendation problem

- What are the inputs? Outputs?
- Are we making good (or bad) recommendations?

### A (Virtuous) Cycle

▷ Recommendation systems are driven by user interactions



- More interactions → Better awareness of user preferences
- Better awareness of user preferences → Better recommendations
- Better recommendations → More interactions

## Recommendations & Feedback

▷ Recommendation systems

- Outputs: Items of interest (e.g., Movies, Books, etc.)
- Inputs: User feedback (Explicit & Implicit)



### Explicit Feedback

▷ For example, users can provide a **rating** for an item

#### Thomas J Ballatore

★★★★★  
17 December 2020 - Published on Amazon.com  
Verified Purchase

#### Adam Zhang

★★★★★  
2 November 2020 - Published on Amazon.com  
Verified Purchase

#### ★★★★★

Verified Purchase

#### ★★★★★

Verified Purchase

#### ★★★★★

Solid MI

By Zimmer | on September 2, 2010

Format: Movie

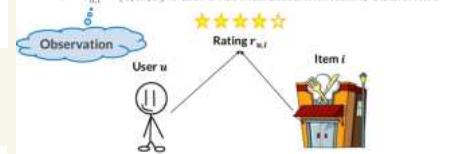
No doubt one of if not the best movie released this year and, just my IMO, in the top 3 Mission films. However I'm not sure it's quite deserving of the high RT rating it received. It does drag a bit in the second act when Solomon Lane is introduced again. The film needed a truly great score stealing villain IMO to compete with the great action and Cruiser's stunts, and Lane just isn't that interesting. Much has been said of Carroll and his amazing moustache and he's decent but a bit wooden. Great physical presence though, Cruise is used as usual. Really really enjoyed the first act and the action scenes toward the end were great. The score by Lame Raffe might just be the best he's done yet. Should have cut the running time a bit tho.

### Explicit Feedback – Rating Prediction

For each user  $u$ , we would like to estimate the rating  $\hat{r}_{u,i}$  for any new item  $i$

#### ▷ (Input) Matrix: $R \in \mathbb{R}^{N \times M}$

- $N$  users,  $M$  items
- $r_{u,i} = [1, \dots, 5]$  if user  $u$  has interacted with item  $i$ , 0 otherwise



▷ Recommend new items that the user would rate highly

▷ Given the observed entries, recover the missing entries

- $\{r_{u,i}\}$ : The set of observed entries
- $\{\hat{r}_{u,i}\}$ : The set of missing entries (to be recovered)

5	?	4	?
?	3	?	1
4	2	?	?
?	?	3	5

## Rating Prediction – Evaluation

- Are we making good or bad recommendations?
  - Compare predicted ratings against actual ratings
- Evaluation (for rating prediction) are usually done using pointwise metrics
  - Mean Absolute Error (MAE)
  - Root Mean Squared Error (RMSE)

## A Simple Example



## A Simple Example – MAE & RMSE



$$\text{MAE} = \frac{(|2-3| + |5-2| + |3-4|)}{3} \approx 1.66$$

$$\text{RMSE} = \sqrt{((2-3)^2 + (5-2)^2 + (3-4)^2)} / 3 \approx 1.91$$

As compared to MAE, RMSE penalizes large prediction errors more (Errors are squared).

## Implicit Feedback

- Explicit Feedback (e.g. ratings, reviews, ...)
- Users need to provide them consciously
- Might not always be readily available

## Alternative: Implicit Feedback

- Collected in the background
- Examples:
  - User has clicked on a link
  - User has watched a movie
  - User has viewed a product
  - User has purchased a product

## Implicit Feedback – Top-K Recommendation

- Given the observed entries, predict the likelihoods for the missing entries
- $I_{ui}$ : The set of observed entries
- $\hat{I}_{ui}$ : The set of missing entries (to be predicted)



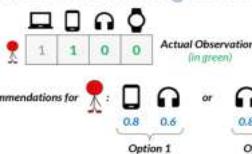
## A Simple Example



## Top-K Recommendation – Evaluation

- Are we making good or bad recommendations?
  - Compare actual observations against the list of top-K recommended items
- Evaluation are usually done using classification metrics and/or ranking-based metrics
  - Classification: Precision, Recall, ...
  - Ranking-based:
    - Normalized Discounted Cumulative Gain (nDCG)
    - Mean Reciprocal Rank (MRR)
    - ...

## Classification vs Ranking-based



- For classification-based metrics, Options 1 & 2 are equally good...
  - The phone (actual purchase) is in both top-2 lists
  - Order does not matter
- For ranking-based metrics, Option 1 is better than Option 2!
  - The phone (actual purchase) is ranked above the headphones for Option 1

## Implicit Feedback – Caveat

- E.g., User did not watch a movie
  - Possibility 1: User dislikes that movie
  - Possibility 2: User is not aware of such a movie
- E.g., User has watched a movie
  - Does the user like or dislike the movie?
- E.g., User purchased a product
  - Does the user like the product?
  - What if it was bought as a gift for someone else?
- For implicit feedback, observations (or the lack thereof) do not necessarily reflect preferences

## 4. HOW do we make recommendations?

### Making Recommendations

- Non-personalized
  - Random
  - Most Popular (e.g., Top 10 movies)
  - Most Recent (e.g., Latest uploads on YouTube)
- Personalized
  - Adapts to the preferences of each user
  - In most cases, provides much better recommendations than non-personalized methods

### Personalized Recommendations

- Collaborative Filtering (Concept)
- Memory-based Approach (Methodology)
- Model-based Approach (Methodology)

## What is Collaborative Filtering?

- Key Idea: People often get the best recommendations from someone with similar tastes as themselves!



- Friend A has similar preferences as the User
- Intuitively, Friend A's recommendations should be better than Friend B

- Relyes (mostly) on the historical interactions
  - Of the target user
  - As well as those of other similar users!

## Not something new

- Has been proposed a long time ago
- However, it's still a very powerful tool
- Most (if not all) recommendation systems are still based on this concept!

- The filtering (recommending a subset of items) is performed with the help of other similar users (i.e., in a collaborative manner)

## Memory-based Approaches

### Memory-based Approach (Methodology)

- Based on similarities between users (or items)
- Leverages upon the historical interactions of the most similar users (or items) to perform recommendations

### Similarity Measures

- Jaccard Similarity
- Cosine Similarity
- Pearson Correlation
- ...

### Examples

- User-based k-Nearest Neighbour (UserKNN)
- Item-based k-Nearest Neighbour (ItemKNN)

### Benefits

- Intuitive & Explainable
- We are recommending Movie X to you because other similar users (who have also watched Movies Y and Z) liked Movie X as well!

### Drawbacks

- Cannot handle sparse data well
  - E.g., If most users only interact with small number of items, it would be challenging (or maybe impossible) to find similar users
- Scalability
  - Less computational efficient when the number of users and items grow

### Model-based Approach (Methodology)

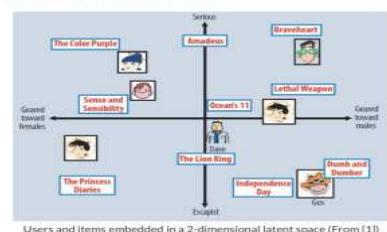
- Models are built using different data mining (and more recently, machine / deep learning) techniques

### Examples

- Bayesian Networks
- Clustering-based Models
- Latent Factor Models (LFMs)
  - Matrix Factorization (MF)



### Latent Factor Models



### Shared Latent Space

- Each user is represented by a vector (of latent factors)
- Each item is represented by a vector (of latent factors)

### Latent Factors

- Learned automatically from historical interactions
- These could be movie genres, item attributes, ...
  - But they are latent, so we do not really know their true meaning

- Similar users are embedded close to each other

- Similar items are embedded close to each other



## Latent Factor Models

▷ How to compute the user-item rating (or likelihood) using latent factors?

- Simple method: Dot product

$$a \cdot b = \sum_{i=1}^n a_i b_i$$

- $a = 1$  vector
- $b = 2$  vector
- $n = \text{dimension of the vector space}$
- $a_i = \text{component of vector } a$
- $b_i = \text{component of vector } b$

- Alternatives: Learning the user-item interaction function using neural networks (e.g. [1]), ...

## Model-based Approaches

▷ Benefits

- Works well even with sparse data
- Efficient and scalable
- # of latent factors << # of users & # of items
  - User-item matrix is compressed into a low-dimensional latent space
  - Dot product between two vectors is very fast

▷ Drawbacks

- Interpretability
  - Black box
- What is the true meaning of the latent factors? ☺

## Hybrid Models

▷ E.g., Combination of memory-based & model-based

- Overcomes the limitations of individual approaches
- Could end up with increased complexity

▷ E.g., Training multiple models and combining their predictions



## 5. Different types of recommendation systems

### Different Types?

▷ Due to the different item types

- E.g., Movies, News, Points of Interests (POIs), ...



▷ Due to the available information

- Content: Text, Images, User demographics, Item attributes, ...
- Context: Time, Location, ...



### POI Recommendation Systems

▷ In contrast to movies, books, etc., POIs are 'physical items'

▷ When users interact with POIs, they are limited by physical constraints

- Distance
  - Recommendations cannot be too far away
- Time
  - Travel time required to reach recommended location

▷ As a result, bad recommendations could be somewhat more 'costly'

▷ Context-sensitive

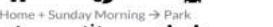
- Distance
  - Current location (e.g., Home vs Workplace)
- Time
  - Weekday vs Weekend
  - Time of the day (Morning/Afternoon/Night)



▷ E.g., Workplace + Friday Evening → Restaurant / Bar



▷ E.g., Home + Saturday Evening → Cinema



▷ E.g., Home + Sunday Morning → Park



▷ Locality

- People's activities in urban environments are usually concentrated in certain geographical regions

• E.g., near our home or workplace

▷ Challenges for POI recommendation

- Very sparse data [1]
- Density for Netflix (Movies): 1.2%
- Density for Yelp & Foursquare (POIs): 0.1%

• Incorporating the rich context information adequately

## Group Recommendation Systems

▷ Recommending items to a group of users

- A group of co-workers
- A group of friends
- A couple
- A family
- ...

▷ Users in a group tend to have...

- Different preferences
- Different levels of influence (Leaders vs Followers)
- ...

▷ To be covered later... ☺



## 6.

### Challenges encountered by recommendation systems

#### Challenges

- ▷ Cold-start Problem
- ▷ Popularity Bias
- ▷ Shilling Attacks
- ▷ ...

#### Cold-start Problem

▷ Recommendation systems are driven by user interactions



- How about new users or items w/ no historical interactions?
- How about users or items w/ little historical interactions?
- Difficult to calculate similarity and/or derive a good representation

- ▷ In some cases, we could utilize the content information
  - E.g., User demographics (Age, Gender, Occupation, ...)
  - E.g., Item attributes (Genre, Year, Actors, ...)



▷ Possible solutions

- Content-based Filtering
- Hybrid Model (Collaborative Filtering + Content)

#### Popularity Bias

▷ Most recommendation systems suffer from popularity bias

- Popular items tend to get recommended over and over again
- Less popular (or niche) products are less likely to be discovered



#### Shilling Attacks

▷ Collaborative Filtering

- "Users who shared similar preferences in the past will likely agree in the future as well"

- Based on "wisdom of the crowd"
  - i.e., what others think of an item or a group of items to compute recommendations

- Shortcoming: Unable to distinguish genuine user profiles from fake ones

▷ Characterised by...

- Several fake user profiles
- Often by an adversarial party
- Harvest recommendation outcomes towards a malicious desire

▷ "Malicious Desire"

- Personal gain, market penetration, causing mischief, etc.

## Robustness Against Shilling Attacks

▷ Existing work tends to cover 3 main directions

1. Attack Designs
2. Detection Algorithms
3. Defence Strategies

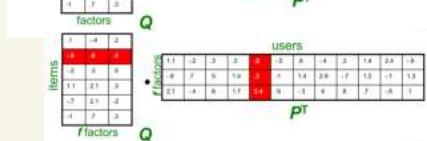
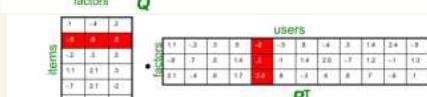


## Ratings as Products of Factors

- How to estimate the missing rating of user  $x$  for item  $i$ ?

$$\hat{r}_{xi} = q_i \cdot p_x \\ = \sum_{f} q_{if} \cdot p_{xf}$$

$q_i$  = row  $i$  of  $Q$   
 $p_x$  = column  $x$  of  $P^T$



$$M_1 = 3.6 \quad M_2 = 4.57 \quad M_3 = 3$$

$$1 - 2.6 \quad 2$$

$$3 - 0.6 \quad 4$$

$$5 \quad 6 \quad 7$$

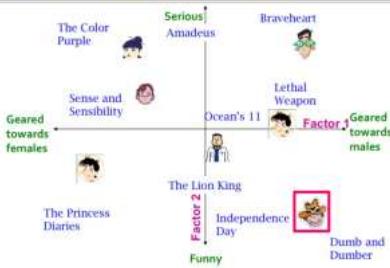
$$8 - 1.4 \quad 9$$

$$10 - 1.4 \quad 11$$

$$12 - 0.4 \quad 13$$

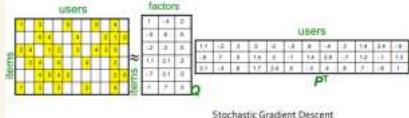
52

## Latent Factor Models



- Our goal is to find  $P$  and  $Q$  such that:

$$\min_{P,Q} \sum_{(i,x) \in R} (r_{xi} - q_i \cdot p_x)^2$$



## Back to Our Problem

- Want to minimize Sum of Squared Errors (SSE) for unseen test data

- Idea: Minimize SSE on training data

- Want large  $k$  (# of factors) to capture all the signals
- But, SSE on test data begins to rise for  $k > 2$



- This is a classical example of overfitting:

- With too much freedom (too many free parameters) the model starts fitting noise
  - That is it fits too well the training data and thus **not generalizing** well to unseen test data

## Dealing with Missing Entries



- To solve overfitting we introduce regularization:

- Allow rich model where there are sufficient data
- Shrink aggressively where data are scarce

$$\min_{P,Q} \underbrace{\sum_{\text{training}} (r_{xi} - q_i \cdot p_x)^2}_{\text{"error"}} + \left[ \underbrace{\lambda_1 \sum_x \|p_x\|^2}_{\text{"length"}} + \underbrace{\lambda_2 \sum_i \|q_i\|^2}_{\text{"length"}} \right]$$

$\lambda_1, \lambda_2 \dots$  user set regularization parameters

Note: We do not care about the "raw" value of the objective function, but we care in  $P, Q$  that achieve the minimum of the objective

## Summary

- User CF and item CF
  - Understand how to compute
- Latent factor approach—Matrix Factorization
  - Understand the high-level idea



# Data basics

Lin Guosheng  
School of Computer Science and Engineering  
Nanyang Technological University

# Outline

- Types of datasets
- Data objects and attributes
- Distance and similarity
- Data normalization

# Types of Datasets

## ■ 1. Record Data

- records in a relational database

Person:

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rom

no relation

Car:

Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

# Types of Datasets

## ■ 2. Graphs and networks



Image credit: [Medium](#)

**Social Networks**

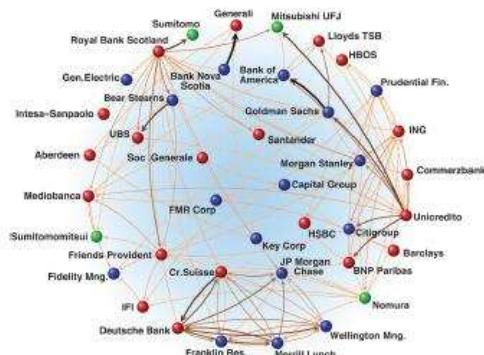


Image credit: [Science](#)

**Economic Networks**

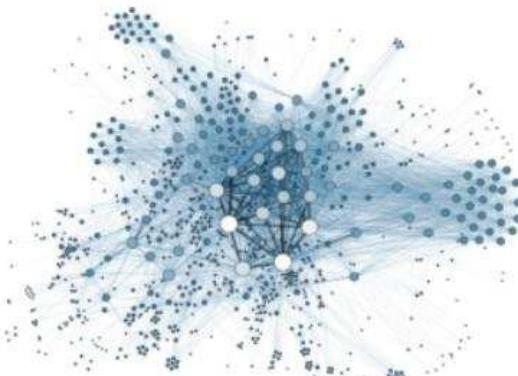


Image credit: [Lumen Learning](#)

**Communication Networks**

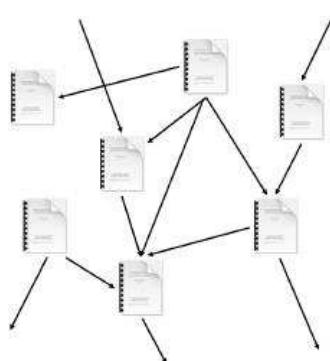


Image credit: [Missoula Current News](#)

**Citation Networks**

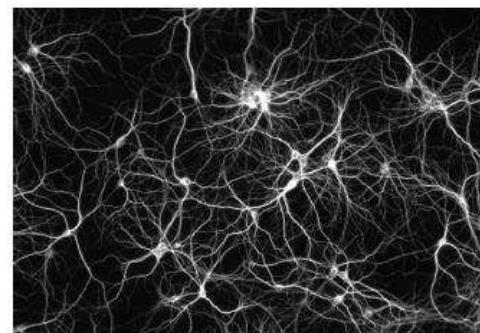


Image credit: [The Conversation](#)

**Internet**

**Networks of Neurons**

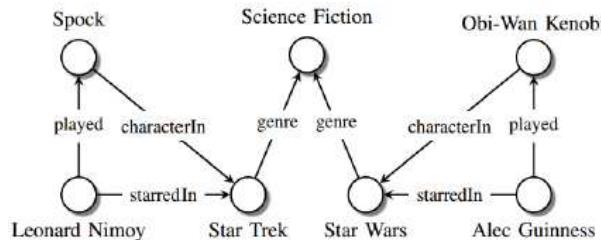


Image credit: [Maximilian Nickel et al](#)

## Knowledge Graphs

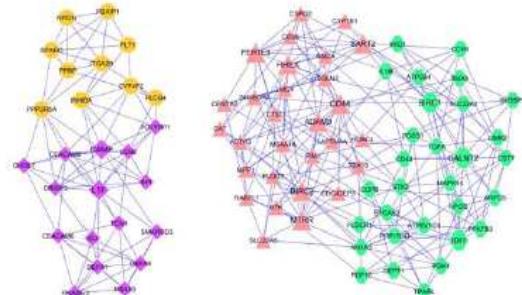


Image credit: [ese.wustl.edu](#)

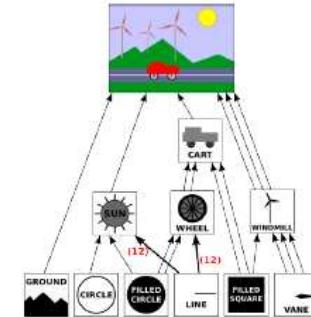


Image credit: [math.hws.edu](#)

## Scene Graphs

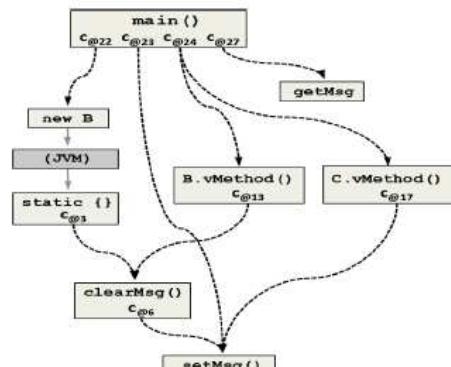


Image credit: [ResearchGate](#)

## Code Graphs

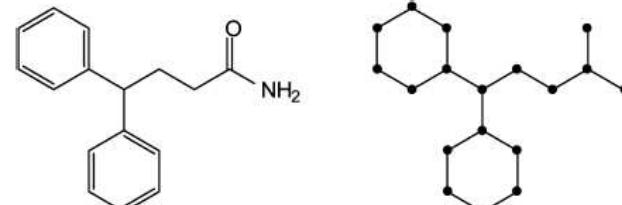


Image credit: [MDPI](#)

## Molecules

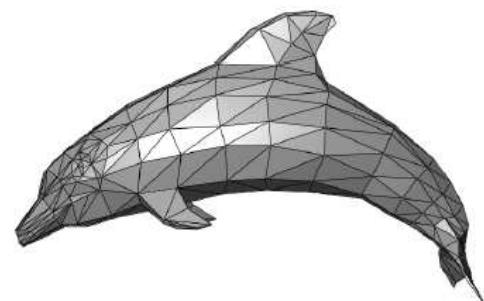


Image credit: [Wikipedia](#)

## 3D Shapes

# Types of Datasets

## ■ 3. Multimedia data



Images

Class: dribble



Class: kick\_ball



Videos / image sequences

# Types of Datasets

## ■ 4. Text data

- Twitter/Facebook posts
- News
- Wikipedia texts
- Shopping item comments
- Books
- Transcripts
- Emails
- Documents
- ...

**The Daily News**

21<sup>st</sup> July 1969      THE WORLD'S FAVOURITE NEWSPAPER      · Since 1879

### The Eagle Has Landed!



The American flag is placed on the moon.

American astronaut Neil Armstrong has become the first person to step on the surface of the moon. Armstrong used the phrase 'the eagle has landed' to let Houston know that the lunar module had actually landed on the moon. As Armstrong put his foot on the moon he declared: "That's one small step for man, one giant leap for mankind." The craft landed on an area of the moon called 'the sea of tranquility.'

Armstrong was joined on the moon by Edwin 'Buzz' Aldrin. A third astronaut, called Michael Collins, stayed aboard the Columbia mothership and orbited the moon. The two moon bound astronauts stayed on the moon for a total of around 21 hours. They spent most of this time inside the lunar module.

*Story continued on the next page.*

Elon Musk - it's fanpage Retweeted

NASA's Perseverance Mars Rover  @NASAPersevere · Aug 2

Exciting news: Not only did I recently grab a new rock core (#11), but plans are coming together to bring these samples back to Earth. A new group of robots (including next-gen helicopters!) could join me for an unprecedented team-up.

More: [go.nasa.gov/3cUWwOU](http://go.nasa.gov/3cUWwOU) #SamplingMars



# Data objects and attributes

- Data objects are also called samples, examples, instances, data points, records, tuples, ....
- Data attributes are also called dimensions, features, variables, channels, ...
- Data sets are made up of data objects/samples
- Data objects are described by attributes/features

Car:

Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

Each row represents a data sample;

Each column represents a data attribute.

# Attribute Types

- **Numeric:** real numbers (continuous values)
  - *Prices, \$500, \$200; Image pixel intensity: [0 255]*
  - *temperature, height, or weight*
- **Nominal:** (Categorical) categories, states, or “names of things” (discrete values)
  - *Hair\_color = {black, blond, brown, grey, red, white}*
  - marital status, occupation, ID numbers, zip codes
- **Binary** (discrete values)
  - Nominal attribute with only 2 states (0 and 1)
  - {true, false},
  - {1, 0} indicates one item exists or not
- **Ordinal** (discrete values)
  - Values have a meaningful order (ranking)  
but magnitude between successive values is not known
  - *Size = {small, medium, large}, grades={A, B, C, D},*

## ■ Convert raw data into numeric values

- For many data mining (machine learning) tasks, e.g., clustering, classification, regression.
- Nominal ->numeric: use one-hot encoding
- Ordinal->numeric: use numbers to indicate ranking (1,2,3,...)
- Binary -> numeric: convert to {0, 1}

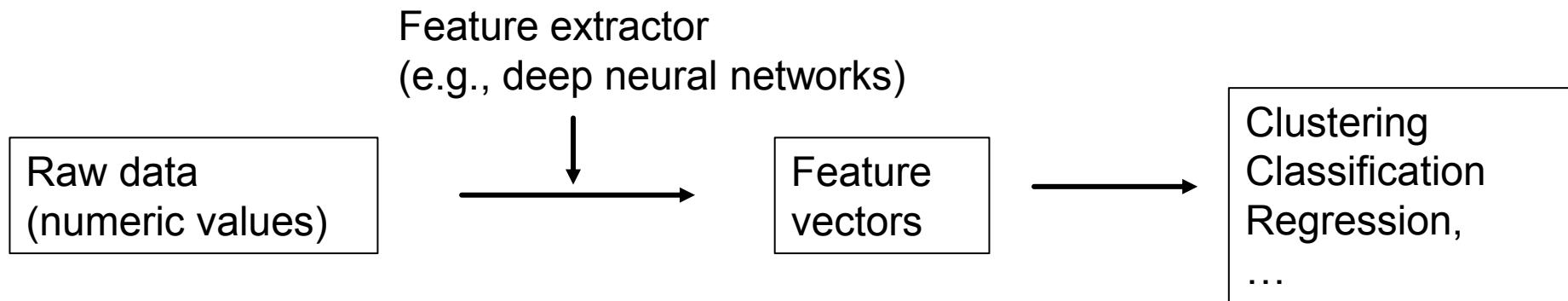
Example: convert nominal values to numeric values using one-hot encoding

id	color
1	red
2	blue
3	green
4	blue



id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

A typical pipeline for applying data mining techniques:



Feature vectors: one data sample is represented by one feature vector.  
e.g.,  $\mathbf{x} = [x_1, x_2, x_3, x_4, \dots]$

## ■ Vector norm

- also called vector magnitude, the length of the vector

### 1. L<sub>p</sub>-norm (general):

Let  $p \geq 1$  be a real number. The  $p$ -norm (also called  $\ell_p$ -norm) of vector  $\mathbf{x} = (x_1, \dots, x_n)$  is<sup>[9]</sup>

$$\|\mathbf{x}\|_p := \left( \sum_{i=1}^n |x_i|^p \right)^{1/p}.$$

### 2. L1-norm:

$$\|\mathbf{x}\|_1 := \sum_{i=1}^n |x_i|.$$

### 3. L2-norm:

$$\|\mathbf{x}\|_2 := \sqrt{x_1^2 + \dots + x_n^2}.$$

Also called Euclidean norm,  
vector length

# Distance

Given two vectors:  $\mathbf{x}_i$  and  $\mathbf{x}_j$  (they have  $l$  dimensions)

1. Calculate the vector difference (residual vector):  $\mathbf{r} = \mathbf{x}_i - \mathbf{x}_j$
2. apply vector norm on the vector difference:

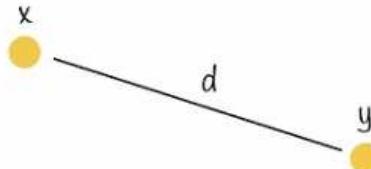
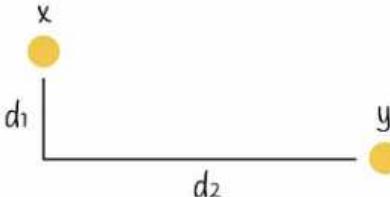
$$d_p(i, j) = \|\mathbf{r}\|_p = \|\mathbf{x}_i - \mathbf{x}_j\|_p$$

- $p = 1$ : ( $L_1$  norm) **Manhattan distance (L1 distance)**

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{il} - x_{jl}|$$

- $p = 2$ : ( $L_2$  norm) **Euclidean distance (L2 distance)**

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{il} - x_{jl}|^2}$$

Metric	Formula	Interpretation
Euclidean distance	$d = \sqrt{\sum_i^n (x_i - y_i)^2}$	 A diagram showing two yellow circular points labeled 'x' and 'y'. A straight line segment connects them, labeled 'd', representing the Euclidean distance between the two points.
Manhattan distance	$d = \sum_i^n  x_i - y_i $	 A diagram showing two yellow circular points labeled 'x' and 'y'. They are connected by a path consisting of a vertical segment of length $d_1$ and a horizontal segment of length $d_2$ , forming an L-shape. Below the diagram, the formula $d = d_1 + d_2$ is written.

<https://towardsdatascience.com/similarity-search-knn-inverted-file-index-7cab8occoe79>

# Distance

- Minkowski distance (defined by vector norm):

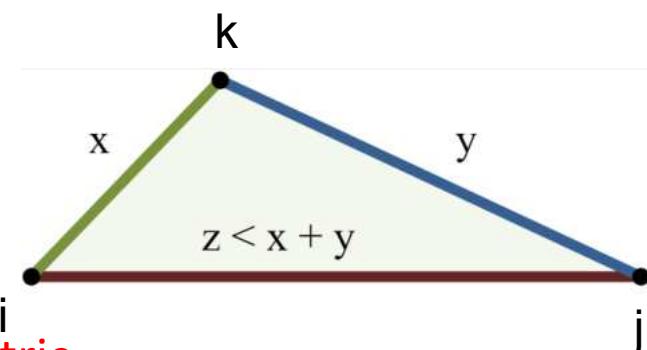
$$d(i, j) = \sqrt[p]{|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \dots + |x_{il} - x_{jl}|^p}$$

where  $i = (x_{i1}, x_{i2}, \dots, x_{il})$  and  $j = (x_{j1}, x_{j2}, \dots, x_{jl})$  are two  $l$ -dimensional data objects, and  $p$  is the order  
( defined based on L- $p$  norm)

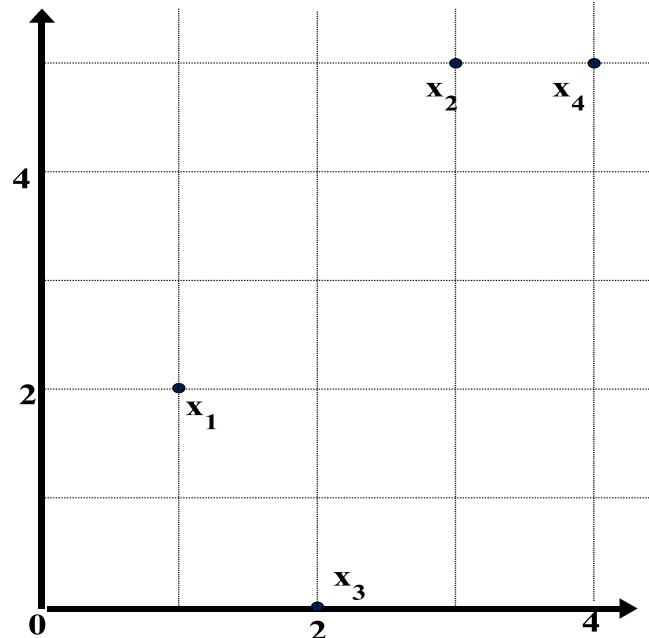
- It has the properties:

- $d(i, j) > 0$  if  $i \neq j$ , and  $d(i, i) = 0$  (Positivity)
- $d(i, j) = d(j, i)$  (Symmetry)
- $d(i, j) \leq d(i, k) + d(k, j)$  (Triangle Inequality)

- A distance that satisfies these properties is a **metric**



- Data matrix
  - Describe a data set (data samples)
  - E.g, a data matrix of  $n$  data points with  $d$  dimensions
  - Each row indicates a feature vector

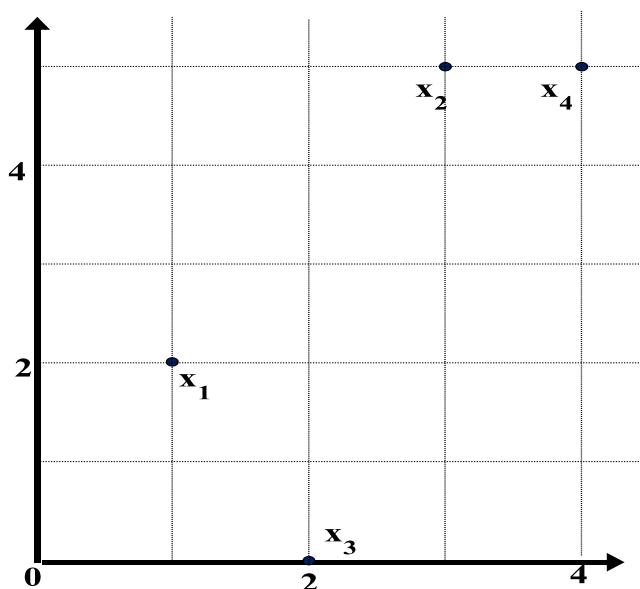


**Data Matrix**  
 $n=4$  (data points),  $d=2$  (dimensions)

point	attribute1	attribute2
$x1$	1	2
$x2$	3	5
$x3$	2	0
$x4$	4	5

# Distance example

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Manhattan distance ( $L_1$ )

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean distance ( $L_2$ )

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

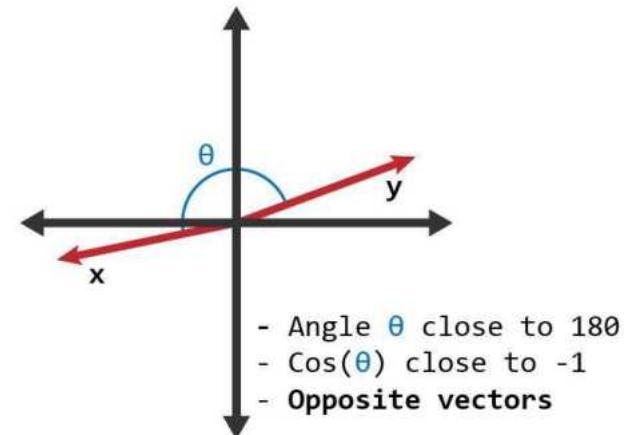
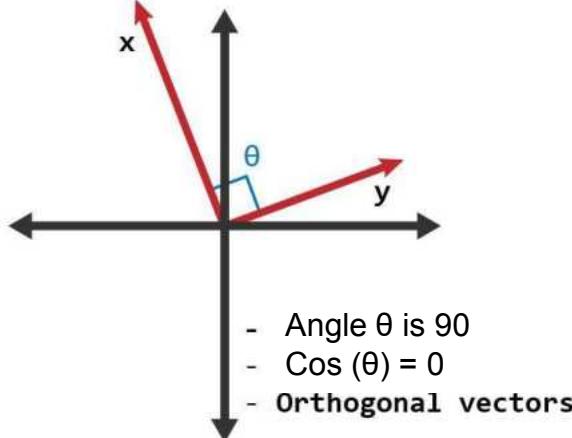
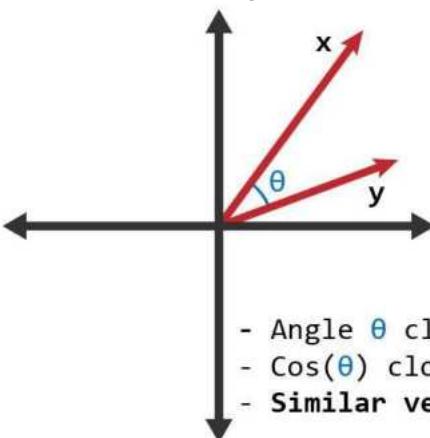
# Similarity

## Cosine similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

Cosine distance = 1 - cosine similarity

Geometry illustration:



<https://www.learndatasci.com/glossary/cosine-similarity/>

# Similarity example

- $D1 = [1, 1, 1, 1, 1, 0, 0]$
- $D2 = [0, 0, 1, 1, 0, 1, 1]$

First, we calculate the dot product of the vectors:

$$D1 \cdot D2 = 1 \times 0 + 1 \times 0 + 1 \times 1 + 1 \times 1 + 1 \times 0 + 0 \times 1 + 0 \times 1 = 2$$

Second, we calculate the magnitude (L2 norm) of the vectors:

$$\|D1\| = \sqrt{1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2} = \sqrt{5}$$

$$\|D2\| = \sqrt{0^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 1^2} = \sqrt{4}$$

Finally:  $similarity(D1, D2) = \frac{D1 \cdot D2}{\|D1\| \|D2\|} = \frac{2}{\sqrt{5}\sqrt{4}} = \frac{2}{\sqrt{20}} = 0.44721$

We can further calculate the angle between the vectors:

$$\cos(\theta) = 0.44721$$

$$\theta = \arccos(0.44721) = 63.435$$

# Data normalization

- Data normalization
  - The goal of normalization is to transform attributes/features to be on a similar scale.
    - Algorithms may bias to the features which have a larger magnitude.
    - E.g., L2-distance will be dominated by large attributes

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{il} - x_{jl}|^2}$$

Examples:

$$\begin{array}{l} x1= [ 100, 0.1 ] \\ x2= [ 120, 0.01 ] \\ x3= [ 120, 0.1 ] \end{array} \longrightarrow d(x1, x2) \approx d(x1, x3)$$

- 1. Max-Min Normalization
  - rescale to a value in [0, 1]

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- 2. Z-score normalization
  - Also called standardization
  - rescale to ensure the mean and the standard deviation to be 0 and 1
  - More robust to outlier

**Population Standard Deviation**

$$\sigma = \sqrt{\frac{\sum(X_i - \mu)^2}{N}}$$

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

$\sigma$  = population standard deviation

$N$  = the size of the population

$X_i$  = each value from the population

$\mu$  = the population mean

# Example

Input dataset

user	Age	Salary
1	40	100000
2	32	80000
3	21	43000
4	24	51000
5	35	70000

Age Mean	30.4
Age Std	7.829432
Salary mean	68800
Salary std	22818.85
Age min	21
Age max	40
Salary min	43000
Salary max	100000

Z-score normalization

user	Age	Salary
1	1.2261426	1.367291
2	0.2043571	0.490822
3	-1.200598	-1.13064
4	-0.817428	-0.78006
5	0.5875267	0.052588

Max-Min Normalization

user	Age	Salary
1	1	1
2	0.5789474	0.649123
3	0	0
4	0.1578947	0.140351
5	0.7368421	0.473684

# **Clustering**

**-K-means**

**-Hierarchical clustering**

Lin Guosheng

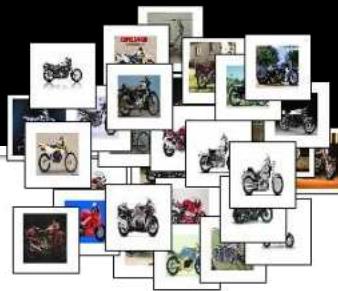
School of Computer Science and Engineering  
Nanyang Technological University

# Outline

- Basic methods for clustering:
  - K-means
    - Algorithm
    - Extension: K-means++
    - K-means with clustroid (**non-examinable**)
    - Optimization problem (**non-examinable**)  
**(non-examinable content may be useful for your coursework projects)**
  - Hierarchical Clustering

# Application

15 -



10 -

5 -

0 -

-5 -

-10 -

-15 -

-15

-10

-5

0

5

10



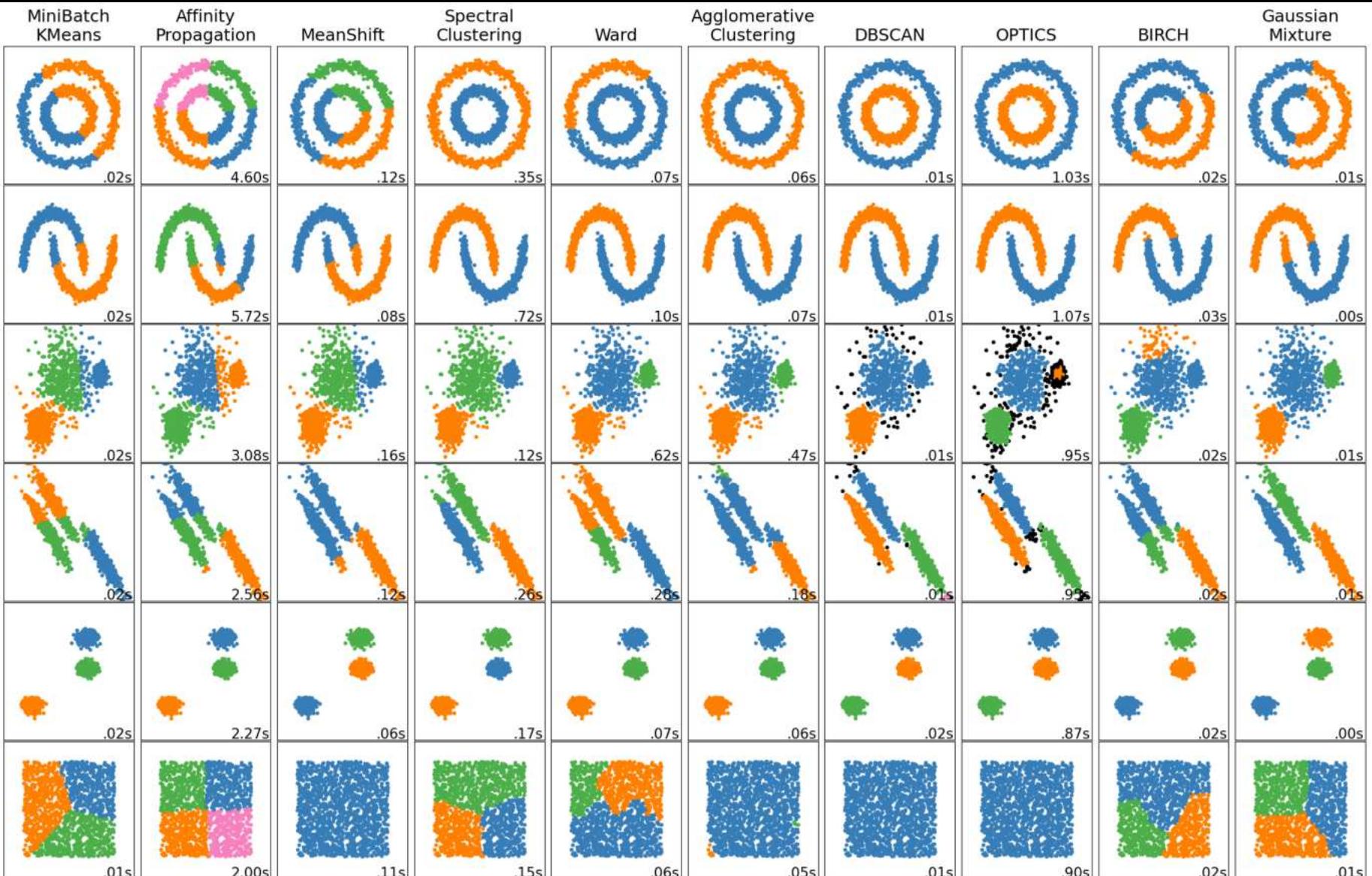
Image clustering

<https://pythonawesome.com/image-classification-feature-visualization-and-transfer-learning-with-keras/>



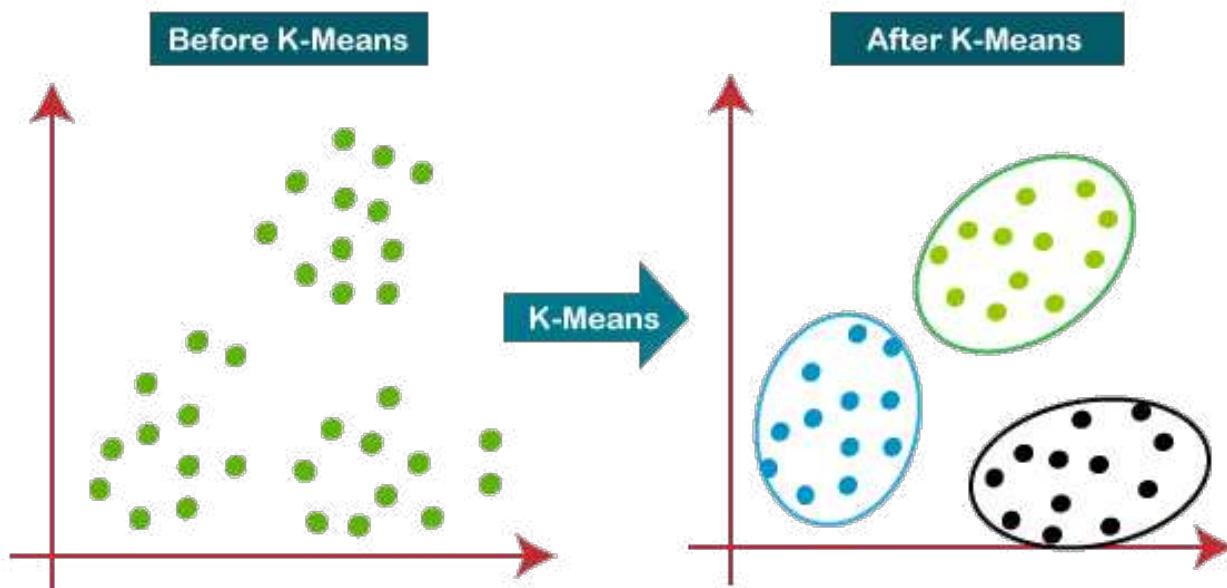
Cluster image pixels to generate superpixels

<https://www.epfl.ch/labs/ivrl/research/slic-superpixels/>



[https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_cluster\\_comparison.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html)

# K-means

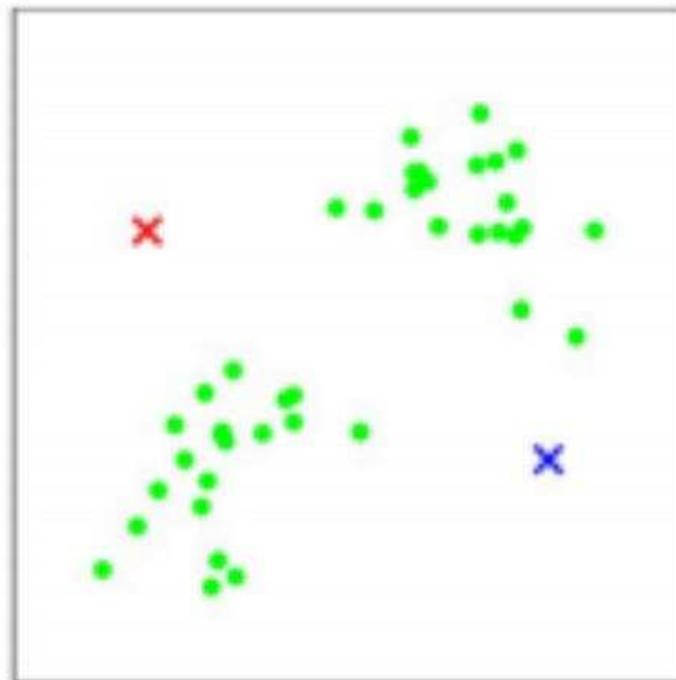


# A simple example



(a)

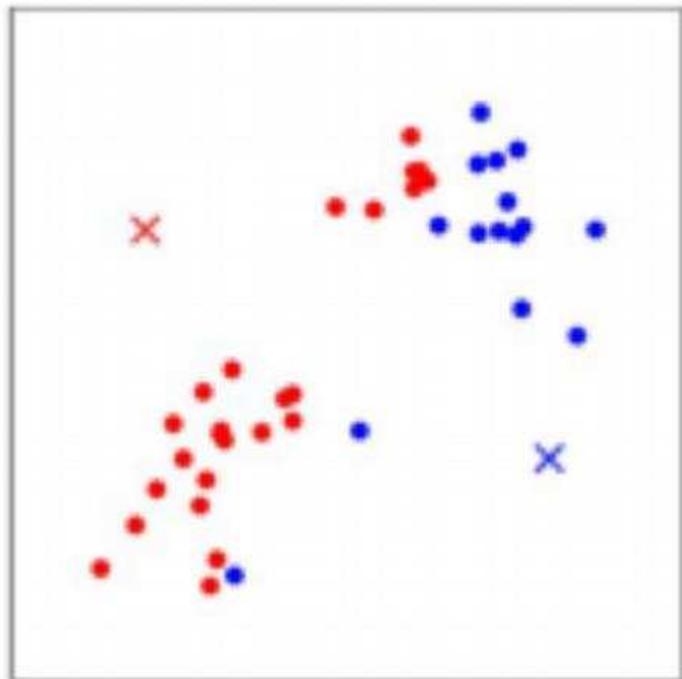
Input data points  
(one point indicates one example,  
feature dimension = 2)



(b)

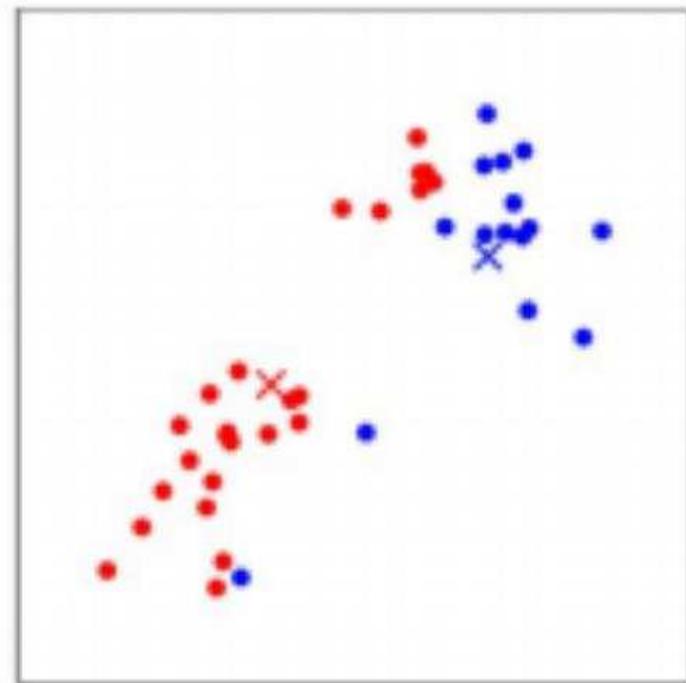
Random initialized cluster centroids  
( $K=2$ )

Iteration 1:



(c)

Assignment step  
(assign each point to the nearest cluster)



(d)

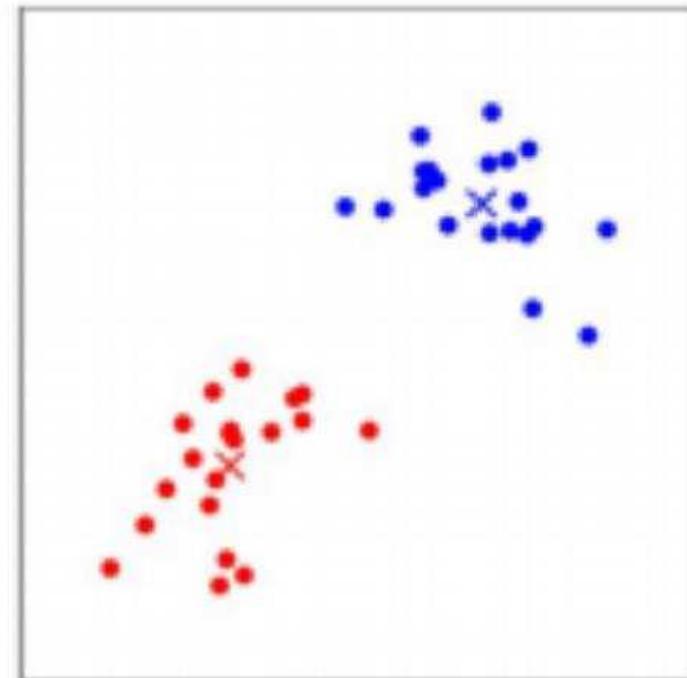
Centroid update step  
(compute the mean in each cluster)

Iteration 2: (converged)



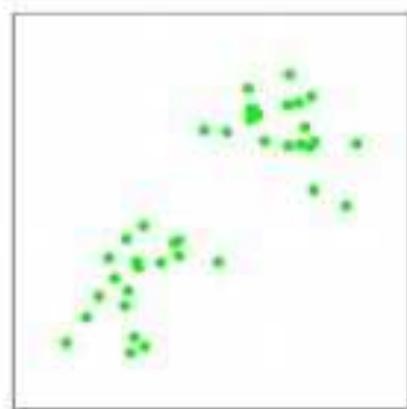
(e)

Assignment step  
(assign each point to the nearest cluster)

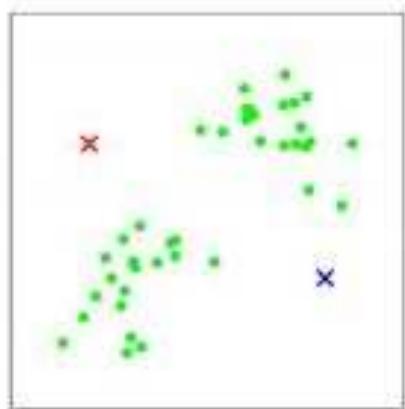


(f)

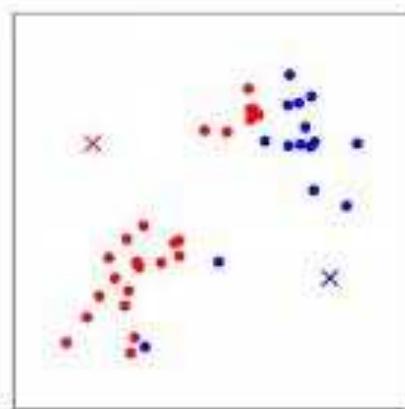
Centroid update step  
(compute the mean in each cluster)



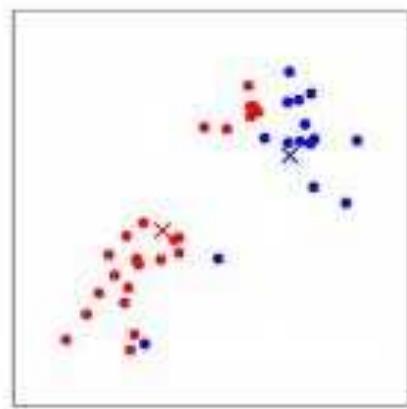
(a)



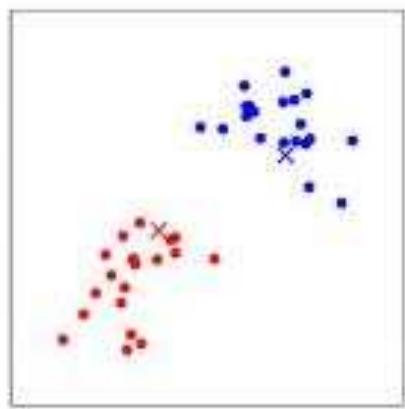
(b)



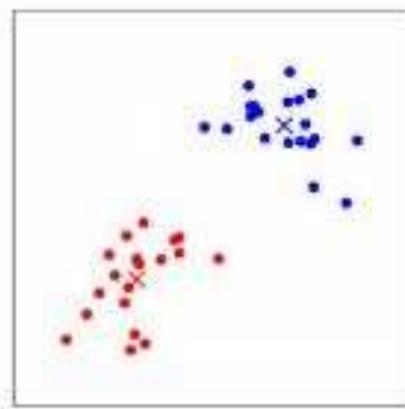
(c)



(d)



(e)



(f)

(a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations of k-means.

<https://stanford.edu/~cziegler/cs221/handouts/kmeans.html>

## ■ K-means

---

### Algorithm 1 $k$ -means algorithm

- 1: Specify the number  $k$  of clusters to assign.
- 2: Randomly initialize  $k$  centroids.
- 3: **repeat**
- 4:   **expectation:** Assign each point to its closest centroid.
- 5:   **maximization:** Compute the new centroid (mean) of each cluster.
- 6: **until** The centroid positions do not change.

Assignment step



Centroid update step



## ■ Example

- Given the following data vectors and initial centroids, perform K-means for 1 iteration.

Data points:

A	$[-1, -1]$
B	$[-2, 0]$
C	$[1, 2]$
D	$[2, 1]$

Initial centroids

(2 clusters):

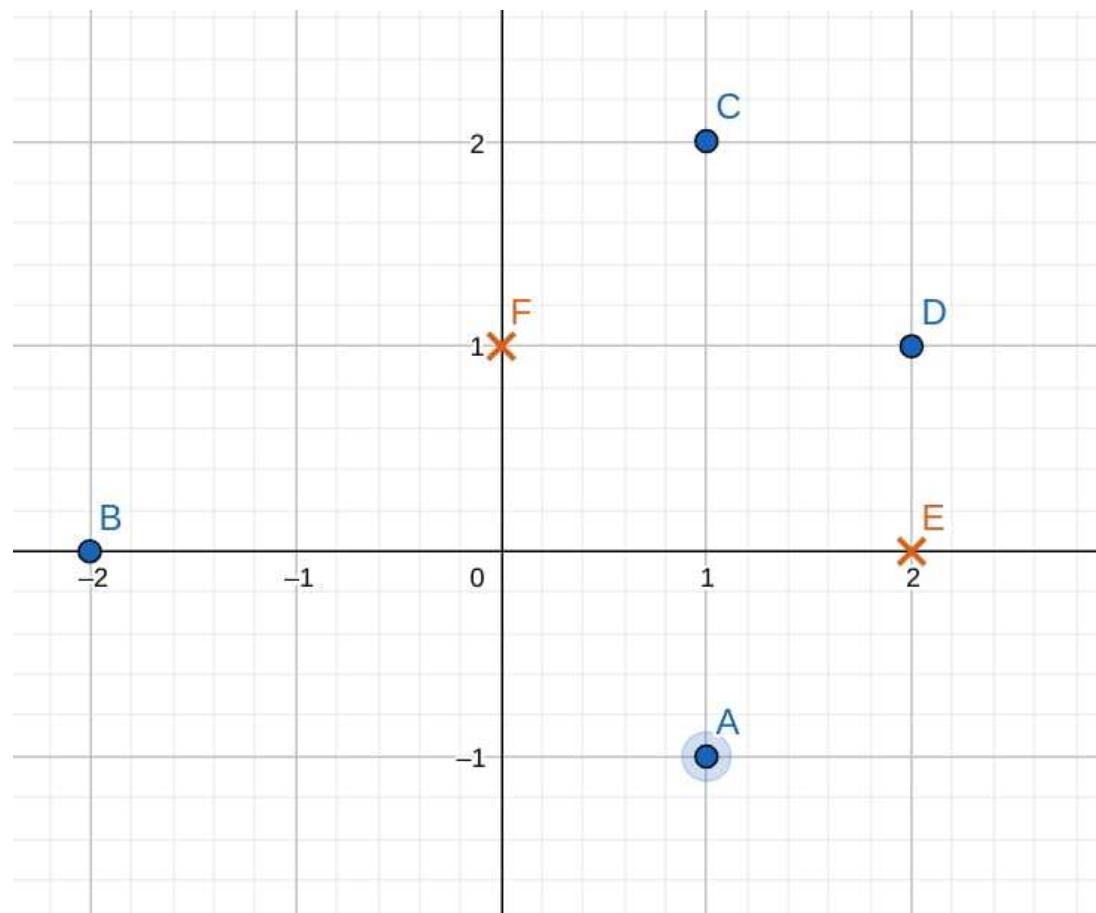
E (Cluster 1)	$[2, 0]$
F (Cluster 2)	$[0, 1]$

Data points:

A	[1, -1]
B	[-2, 0]
C	[1, 2]
D	[2, 1]

Initial centroids  
(2 clusters):

E (Cluster 1)	[2, 0]
F (Cluster 2)	[0, 1]



# Iteration 1

Step 1: Cluster assignment

1) calculate squared L2 distance in Table 1

We can use squared L2 distance

as it will give the same comparison results as L2 distance.

E.g.,  $\text{SquaredL2}(A, E) = (1-2)^2 + (-1-0)^2 = 1 + 1 = 2;$

$\text{SquaredL2}(A, F) = (1-0)^2 + (-1-1)^2 = 1 + 4 = 5;$

Table 1 Squared L2 distance:

	E (2, 0) Cluster1	F (0, 1) Cluster2
A (1, -1)	2	5
B (-2, 0)	16	5
C (1, 2)	5	2
D (2, 1)	1	4

Squared L2 distance:

$$\|x - y\| = \sum_{i=1}^d (x_i - y_i)^2$$

L2 distance:

$$d_{L2}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

2) Assign each point to its nearest centroid

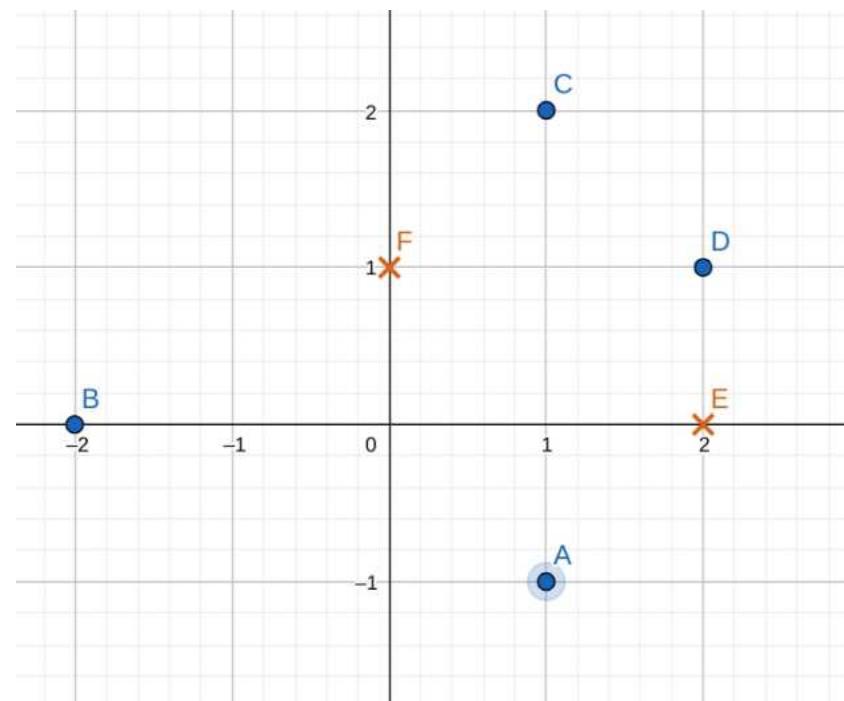
Cluster1: (A, D) Cluster2 : (B, C)

Step 2: Centroid update by compute the mean of each cluster  
Cluster1 E=[1.5, 0], F=[-0.5, 1]

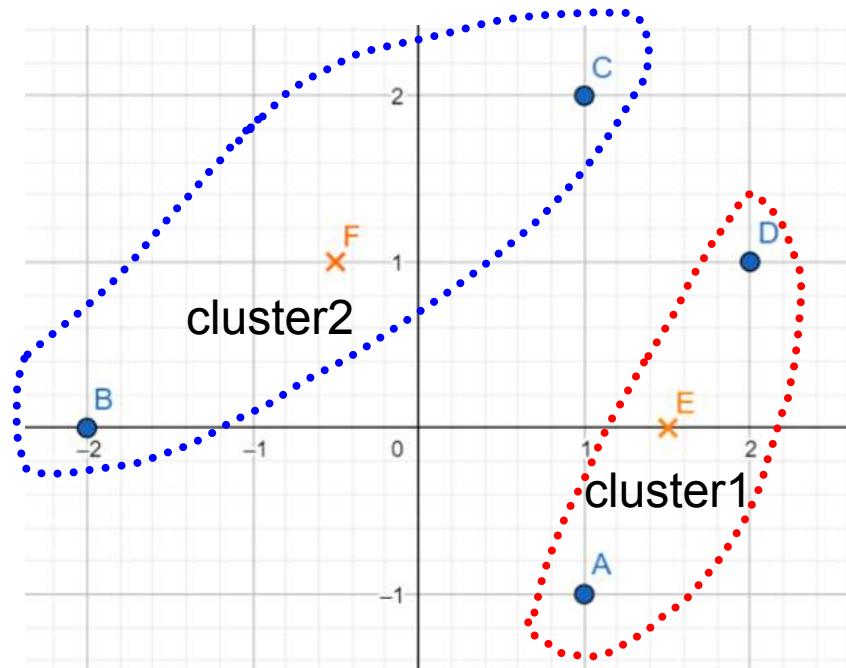
$$E_1 = (1+2)/2 = 1.5, E_2 = (-1 + 1)/2 = 0, \rightarrow E = [1.5, 0] \\ F_1 = (-2 + 1)/2 = -0.5, F_2 = (0 + 2)/2 = 1, \rightarrow F = [-0.5, 1]$$

Table 1 Squared L2 distance:

	E (2, 0) Cluster1	F (0, 1) Cluster2
A (1, -1)	2	5
B (-2, 0)	16	5
C (1, 2)	5	2
D (2, 1)	1	4



Before iteration 1  
(initial centroids)



After iteration 1

## ■ Iteration 2

Beyond the question, we perform the 2<sup>nd</sup> iteration:

Step 1: Cluster assignment

- 1) calculate squared L<sub>2</sub> distance in Table 2
  - 2) assign each point to its nearest centroid
- Cluster C<sub>1</sub>: (A, D)   Cluster C<sub>2</sub>: (B, C)

There is no changes of cluster assignments compared to the previous iteration, the algorithm is converged.

Step 2: Centroid update

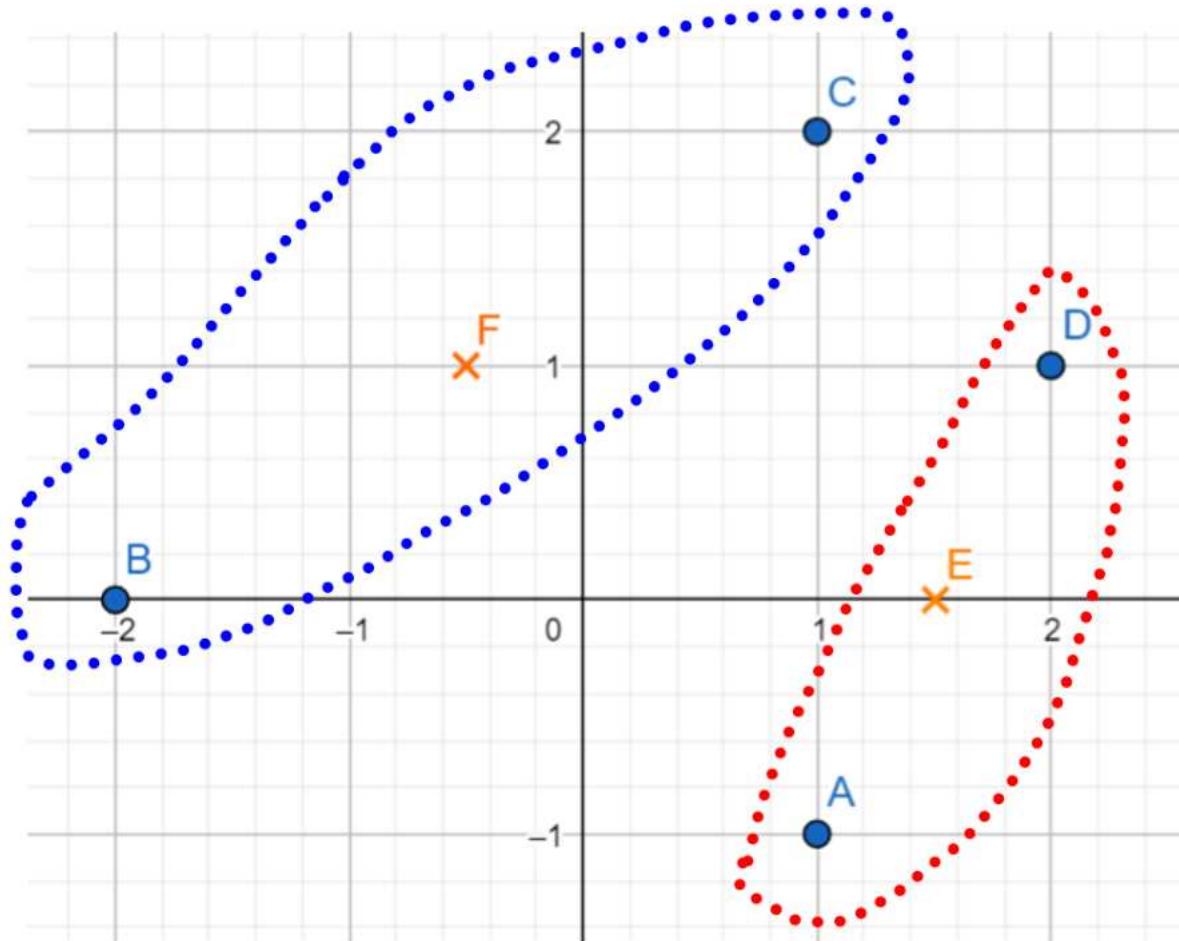
Cluster 1 (E): (1.5, 0)

Cluster 2 (F): (-0.5, 1)

Converged

Table: Squared L<sub>2</sub> distance

	E(1.5, 0) C <sub>1</sub>	F (-0.5, 1) C <sub>2</sub>
A (1, -1)	<b>1.25</b>	6.25
B (-2, 0)	12.25	<b>3.25</b>
C (1, 2)	4.25	<b>3.25</b>
D (2, 1)	<b>1.25</b>	6.25



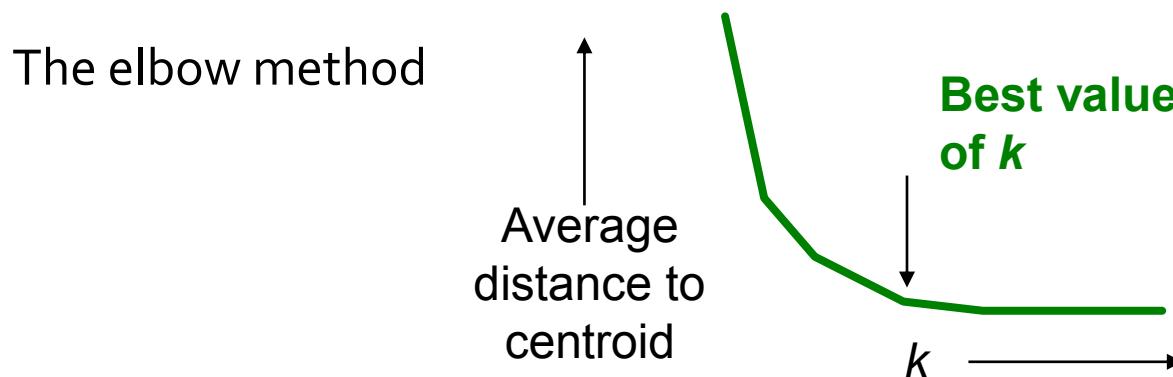
After iteration 2 (converged)

- How to determine the K parameter?
  - K is the number of clusters

[https://en.wikipedia.org/wiki/K-means\\_clustering](https://en.wikipedia.org/wiki/K-means_clustering)

## How to select k?

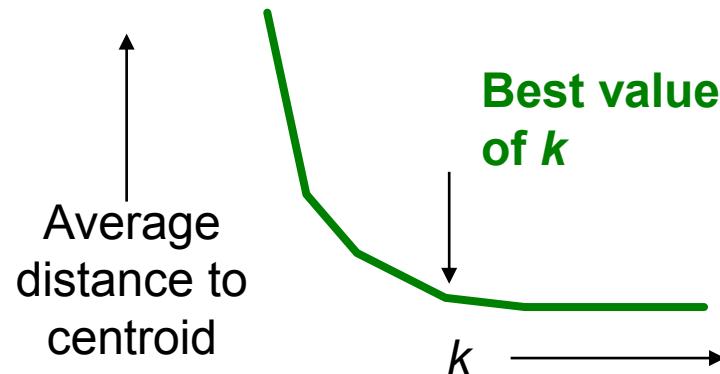
- Try different k, looking at the change in the average distance to centroid as k increases
- Average distance to centroid falls rapidly at the beginning, then it changes slowly after a certain value of k



Select a small k that can produce small average distance to centroid

1. We prefer a low average distance to centroid
2. We prefer a small k

The elbow method



Average distance to centroid: E

$$E = \frac{1}{N} \left( \sum_{p_i \in C_1} d(p_i, c_1) + \sum_{p_i \in C_2} d(p_i, c_2) + \dots + \sum_{p_i \in C_k} d(p_i, c_k) \right)$$

d: distance, e.g., L2 distance or Squared L2 distance

p: a data point

c: a cluster centroid, C: a cluster

N: the total number of data points

# Example

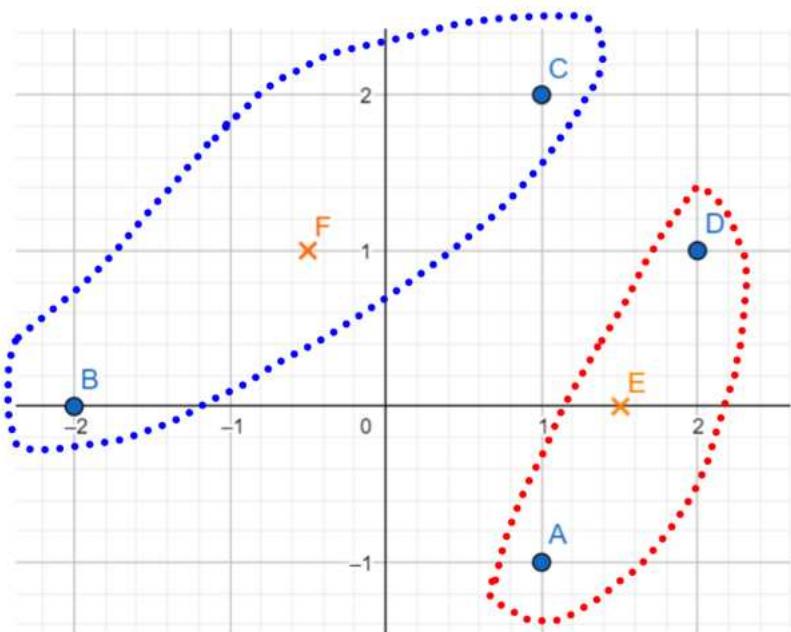


Table: Squared L<sub>2</sub> distance

	E(1.5, 0) C <sub>1</sub>	F (-0.5, 1) C <sub>2</sub>
A (1, -1)	<b>1.25</b>	6.25
B (-2, 0)	12.25	<b>3.25</b>
C (1, 2)	4.25	<b>3.25</b>
D (2, 1)	<b>1.25</b>	6.25

k=2, use Squared L<sub>2</sub> distance for d here:

$$y_1 = d(A, C_1) + d(D, C_1) = 1.25 + 1.25 = 2.5$$

$$y_2 = d(B, C_2) + d(C, C_2) = 3.25 + 3.25 = 6.5$$

Average distance to centroid:

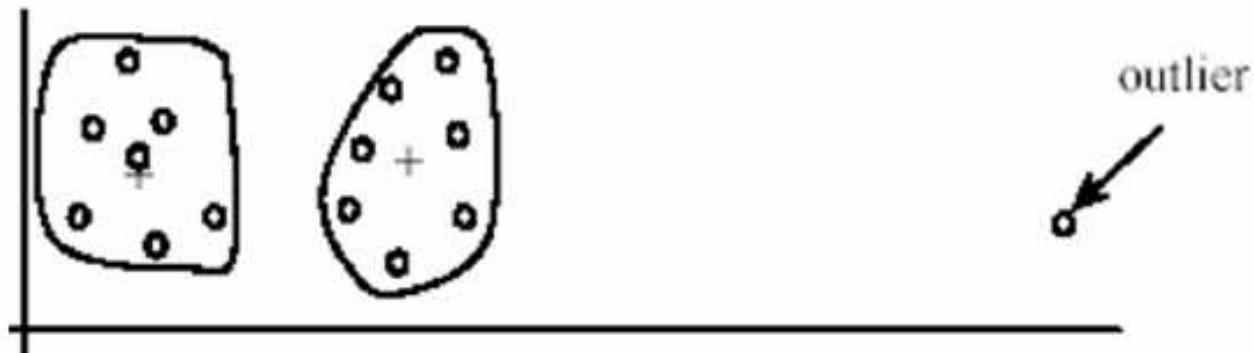
$$E = (y_1+y_2)/4 = 9/4 = 2.25$$

## ■ K-means

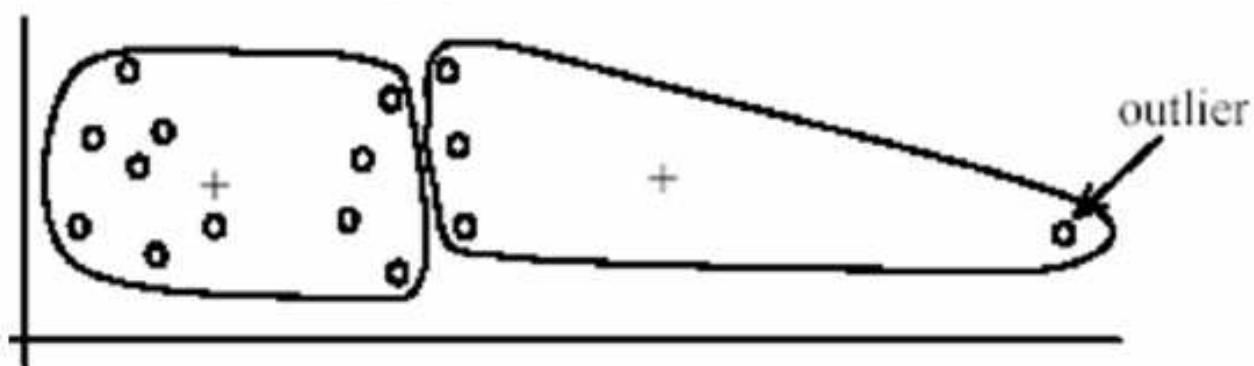
- distance functions
  - Euclidean distance (L2 distance)
    - most commonly used
  - L1 distance
  - Cosine distance
  - Other distance functions

# K-means

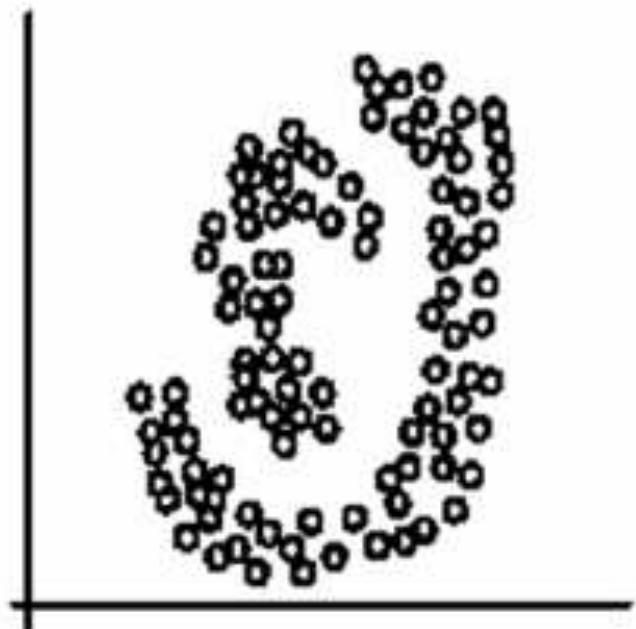
- Pros
  - Simple and fast, Easy to implement
- Cons
  - Need to choose K
  - Sensitive to outliers
  - Cannot work well on data with non-spherical shape
  - Sensitive to the initialization of the centroids



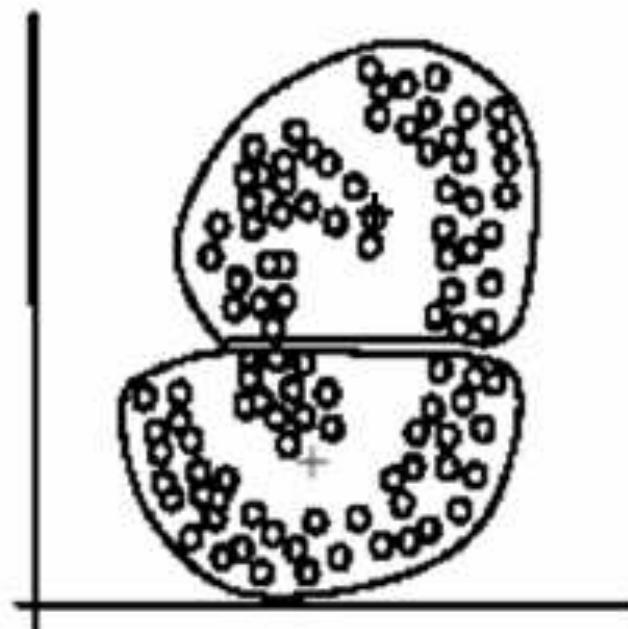
(B): Ideal clusters



Limitation of K-means: sensitive to outliers



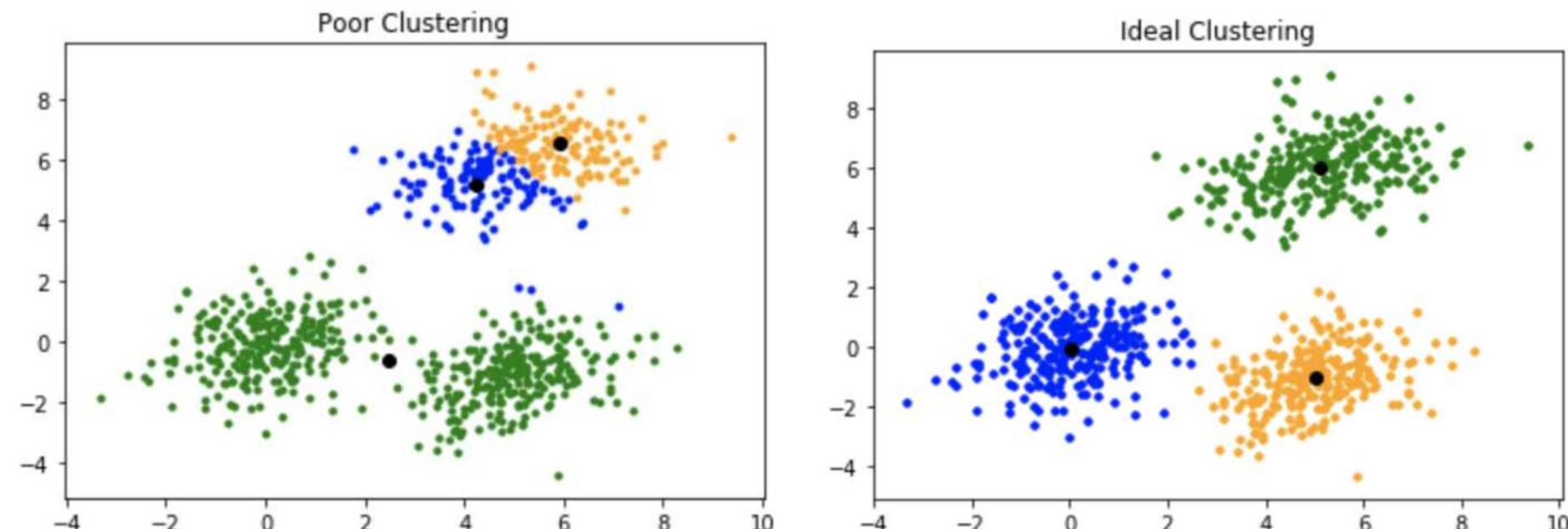
(A): Two natural clusters



(B):  $k$ -means clusters

Limitation of K-means: cannot handle arbitrary shapes.  
K-means is suitable for spherical or elliptical data.

- K-means algorithm is sensitive to the initialization of the centroids



<https://www.geeksforgeeks.org/ml-k-means-algorithm/>

- K-mean++:
  - Better initialization
  - K-means++ is the standard K-means algorithm coupled with a smarter initialization of the centroids.

## **K-mean++:**

Idea: select initial centroids that are far away from each other.

1. Sequentially initialize the centroids.
2. When selecting one centroid, we aim to select a data point that is far away from existing centroids.

<http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

## K-mean++:

let  $D(x)$  denote the shortest distance from a data point  $x$  to the existing centroids ( $c_1, c_2, c_3, \dots, c_n$ ).

$$D(x) = \min_i \text{distance}(x, c_i)$$

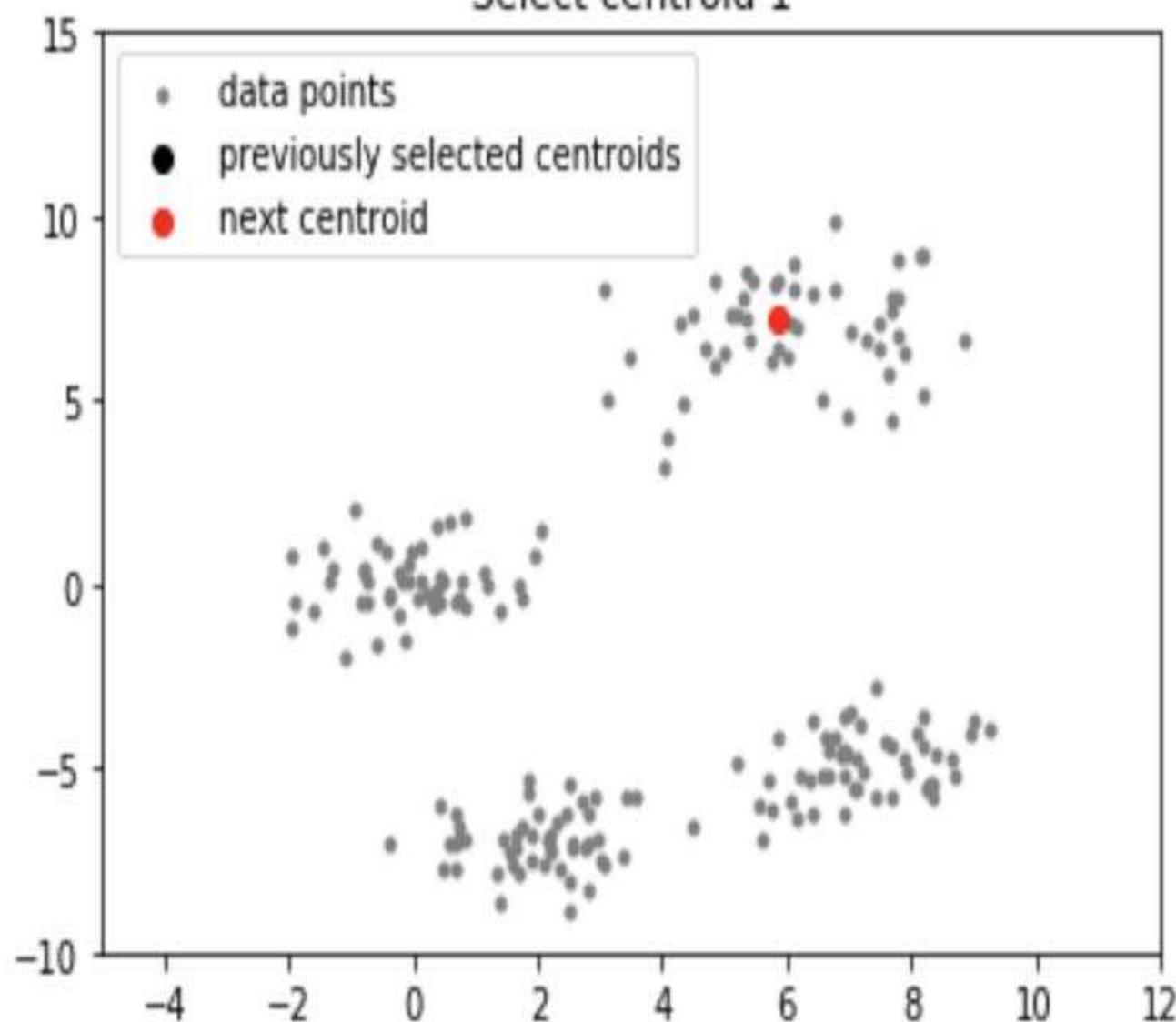
K-mean ++ (simplified version):

Step 1. choose the data point with the highest  $D(x)$  as the new centroid ( $C_{n+1}$ ).  
The selected data point will be far away from all existing centroids.

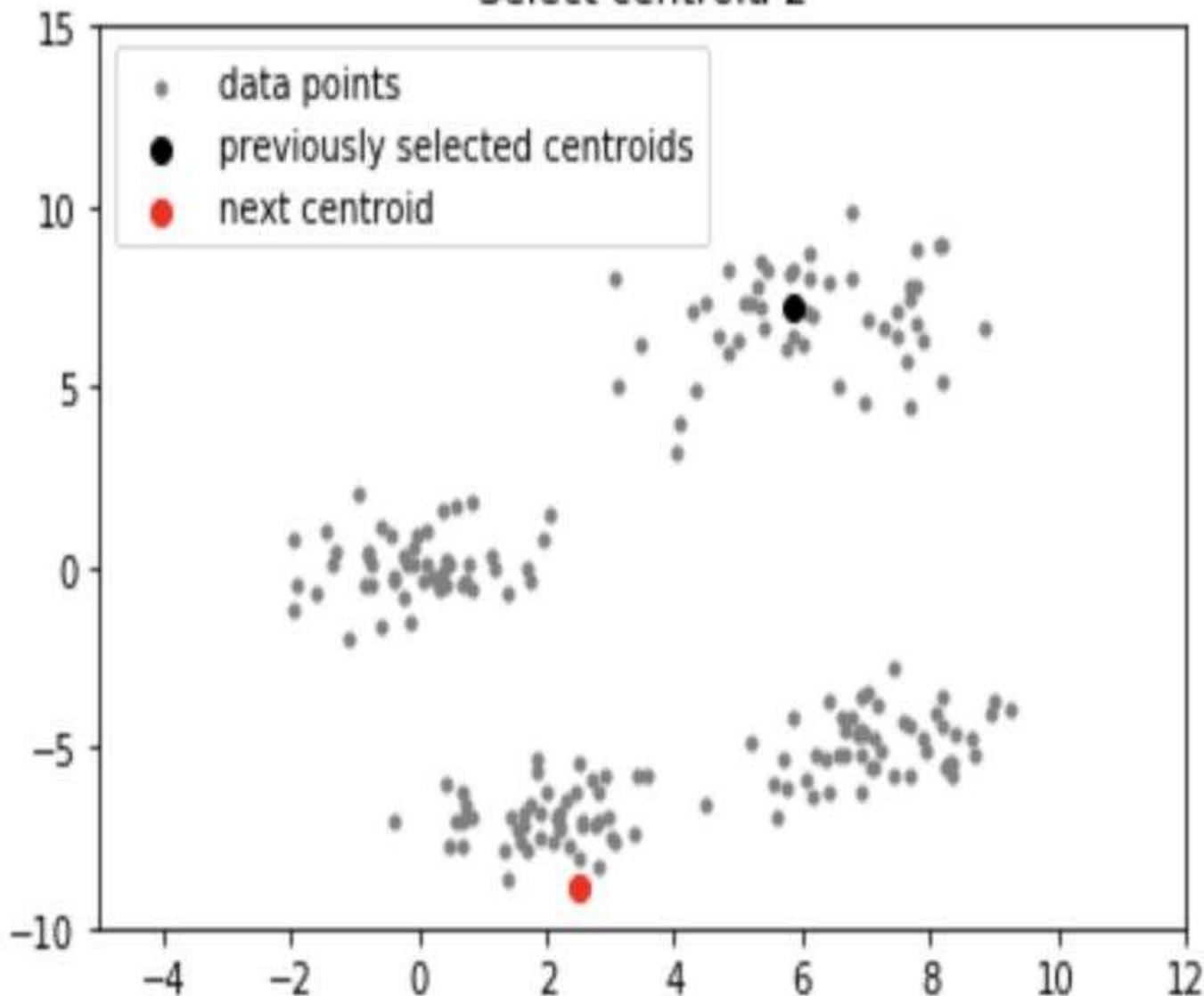
Step 2. Repeat step 1 until we have  $k$  centroids.

1. If the data point  $x$  is close to any existing centroids,  $D(x)$  will be small
2. If the data point  $x$  is far away from any existing centroids,  $D(x)$  will be large

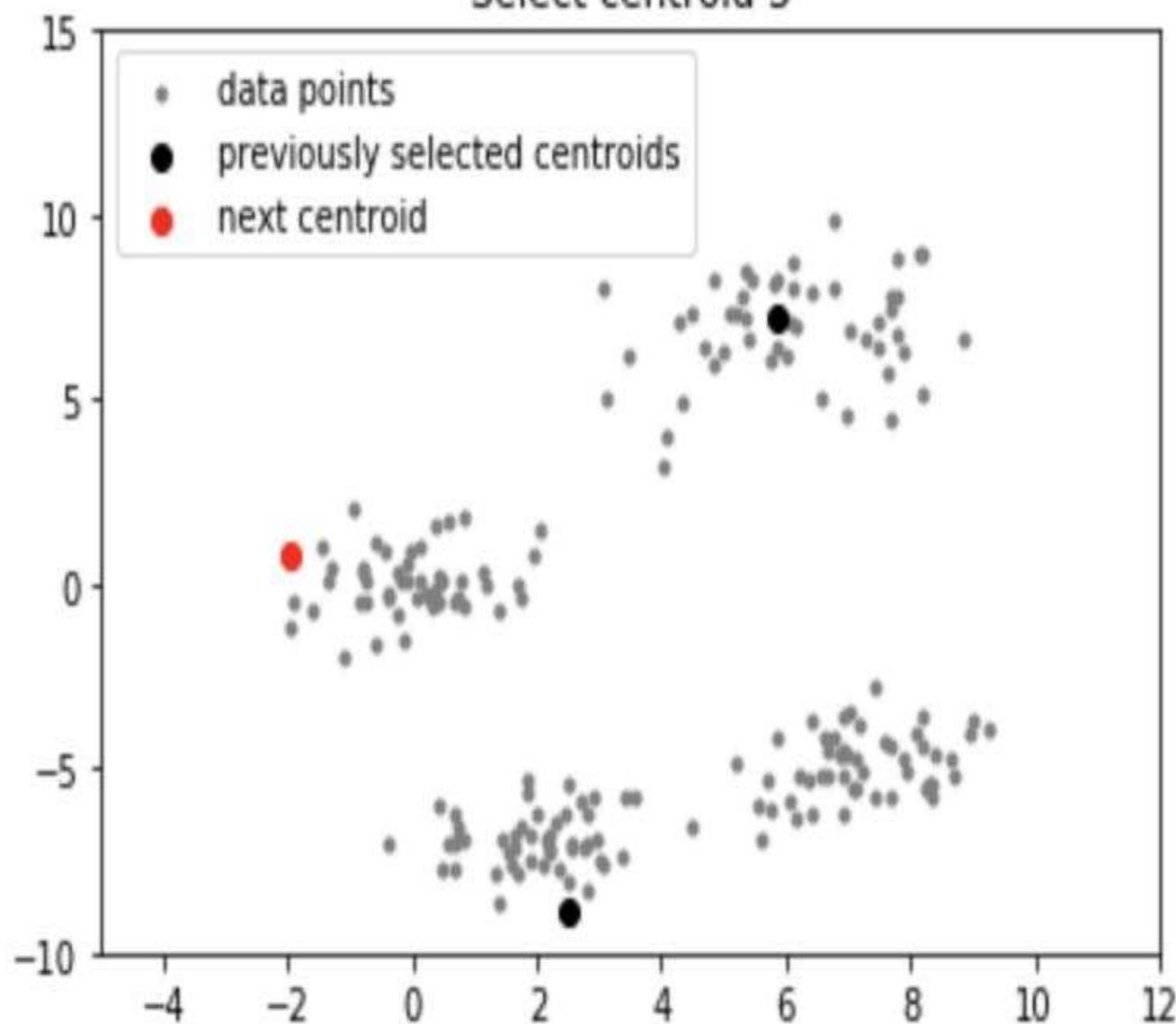
### Select centroid 1



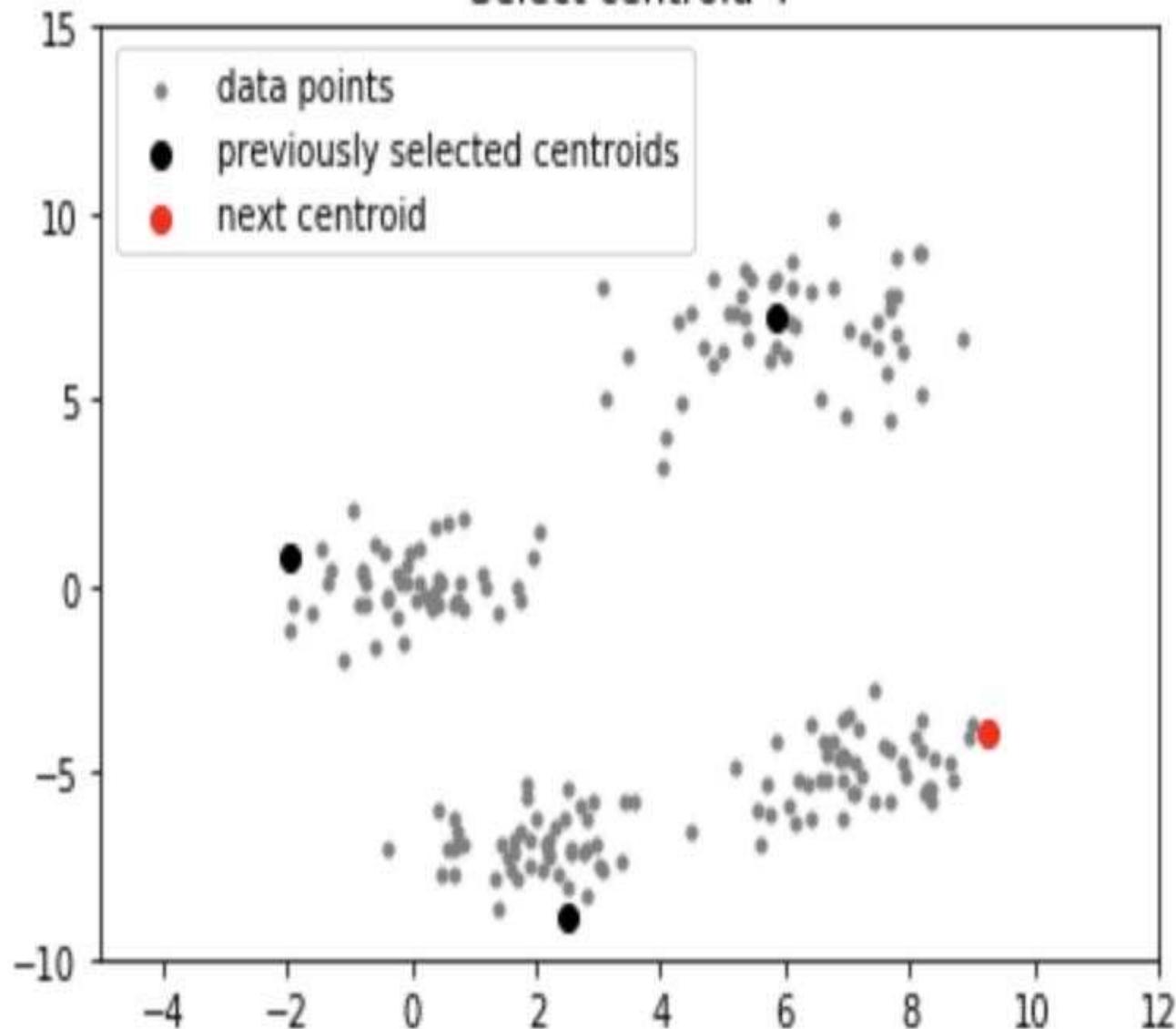
## Select centroid 2



### Select centroid 3



### Select centroid 4



# Segmentation as Clustering

- Let's just use the pixel intensities!

Segmentation: pixel grouping  
Assign pixels to clusters



$k = 2$



$k = 3$



More examples:

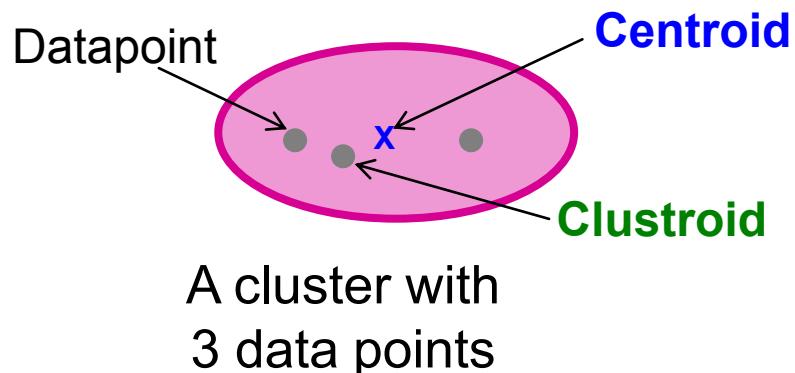
<https://github.com/topics/slic>



More examples: SLIC for super-pixel segmentation  
<https://github.com/topics/slic>

## ■ K-means with clustroid

- Use clustroid to replace centroid
  - Clustroid: is an existing (data) point that is “closest” to all other points in the cluster.



**Centroid** is the avg. of all (data) points in the cluster.

**Centroid** is not an existing point. It is an “artificial” point.

For a point  $p$ , calculate the average distance between  $p$  and all other members in the same cluster. Clustroid is the point that has the smallest average distance within the cluster.



# Optimization problem



## ■ The optimization problem for K-means

The squared distance to centroid for one cluster:

$$S(C_i) = \sum_{x_j \in C_i} (c_i - x_j)^2$$

$S(C_i)$  : squared distance to centroid for cluster  $C_i$

$x_j \in C_i$  : a data point in cluster  $C_i$

$c_i \in C_i$  : the cluster centroid of  $C_i$

# Optimization problem

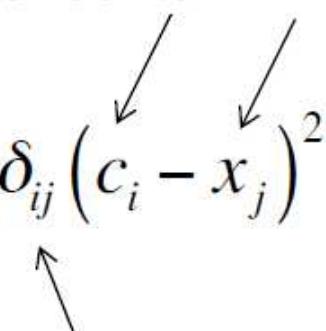
Goal: minimize average squared distance to centroid  
(or minimize within-cluster variance)

Objective function:

$$c^*, \delta^* = \arg \min_{c, \delta} \frac{1}{N} \sum_j^K \sum_i \delta_{ij} (c_i - x_j)^2$$

Asterisk \* indicates the solutions to

1. cluster centroids
2. and assignments

Cluster center      Data  
  
Whether  $x_j$  is assigned to  $c_i$   
if  $x_j$  is a member of cluster  $c_i$ ,  $\delta_{ij} = 1$ ;  
otherwise,  $\delta_{ij} = 0$ .

# Optimization problem

## K-means clustering

1. Initialize ( $t = 0$ ): cluster centers  $c_1, \dots, c_K$
2. Compute  $\delta^t$ : assign each point to the closest center
  - $\delta^t$  denotes the set of assignment for each  $x_j$  to cluster  $c_i$  at iteration  $t$

Assignment step

$$\delta^t = \arg \min_{\delta} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij} (c_i^{t-1} - x_j)^2 \quad \text{Freeze } c, \text{ solve for } \delta$$

3. Computer  $c'$ : update cluster centers as the mean of the points

Centroid update step

$$\mathbf{c}^t = \arg \min_{\mathbf{c}} \frac{1}{N} \sum_j^N \sum_i^K \delta_{ij}^t (c_i - x_j)^2 \quad \text{Freeze } \delta, \text{ solve for } c$$

4. Update  $t = t + 1$ , Repeat Step 2-3 till stopped

# Optimization problem

- Hard to solve the optimization problem
  - Hard to obtain global optimal solutions
- The iterative algorithm is also referred to as Lloyd's algorithm
  - Empirically works well, usually converges in a few iterations

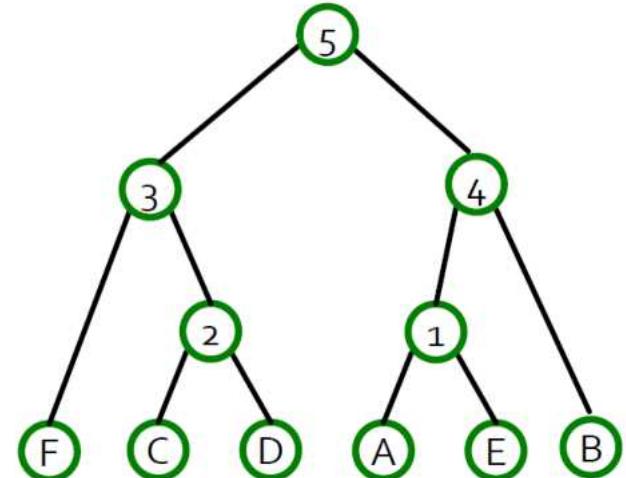
[https://en.wikipedia.org/wiki/K-means\\_clustering](https://en.wikipedia.org/wiki/K-means_clustering)

- Further reading:
  - Mini-batch K-means for large-scale data
    - Handle a small number of data points at a time, using stochastic gradient decent to gradually update the cluster centre .
    - Further reading:
      - Online tool: `sklearn.cluster.MiniBatchKMeans`

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html>

# Hierarchical clustering

- Hierarchical
  - Agglomerative (bottom up)
    - Initially, each point is a cluster
    - Repeatedly combine the two nearest clusters into one
  - Divisive (top down)
    - Start with one cluster and recursively split it



# Hierarchical clustering

## Agglomerative clustering

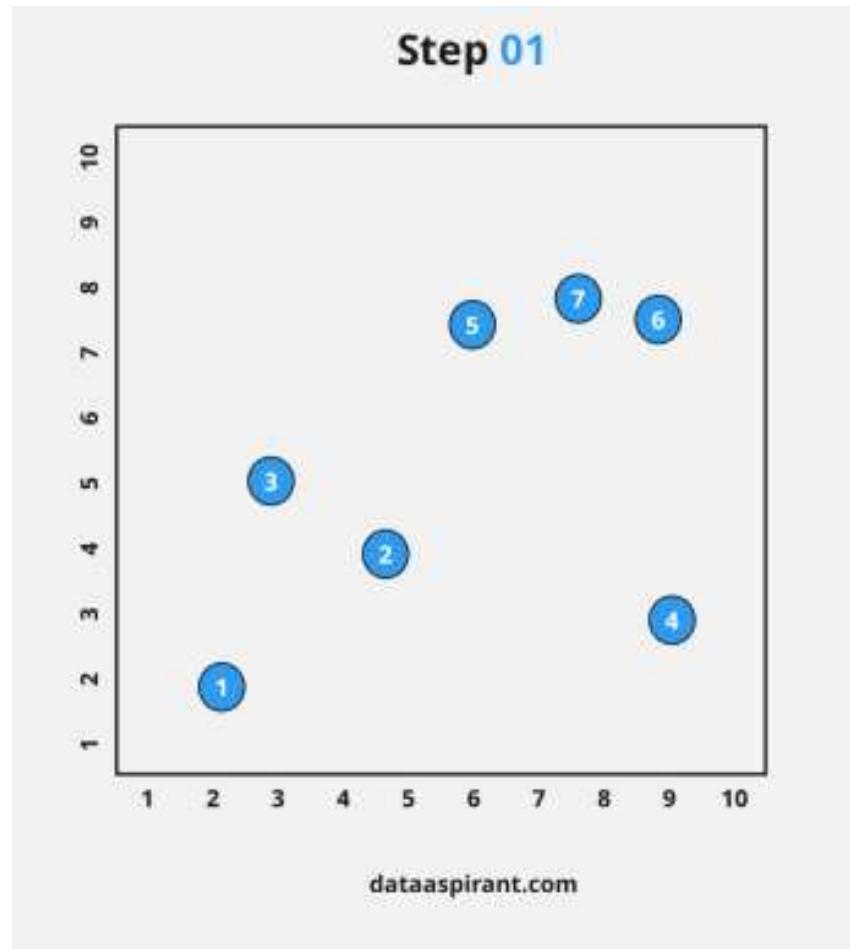
(or bottom-up hierarchical clustering)

### Simple algorithm

- Initialization:
  - Every point is its own cluster
- Repeat:
  - Find “most similar” pair of clusters
  - Merge into a parent cluster
- Until:
  - The desired number of clusters has been reached
  - There is only one cluster

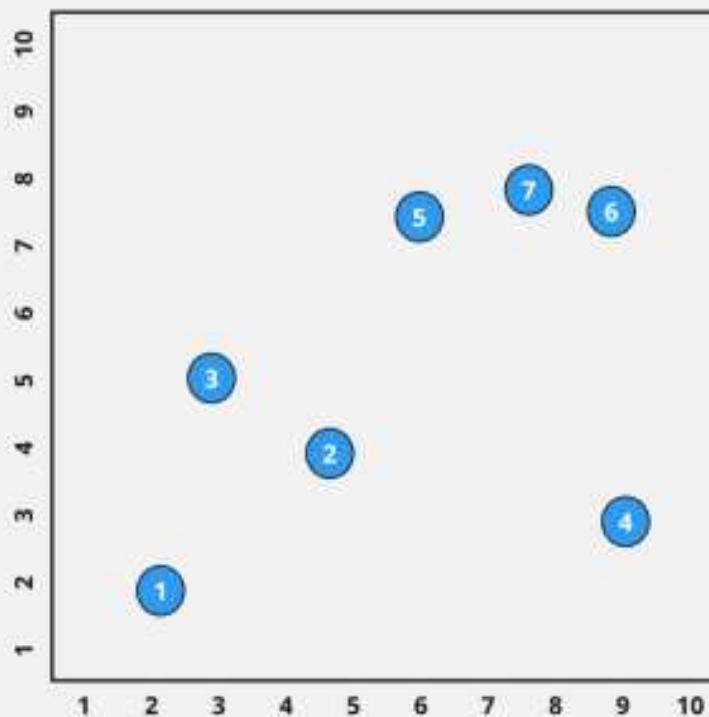
First, make each data point a cluster, which forms N clusters.

## Example 1:

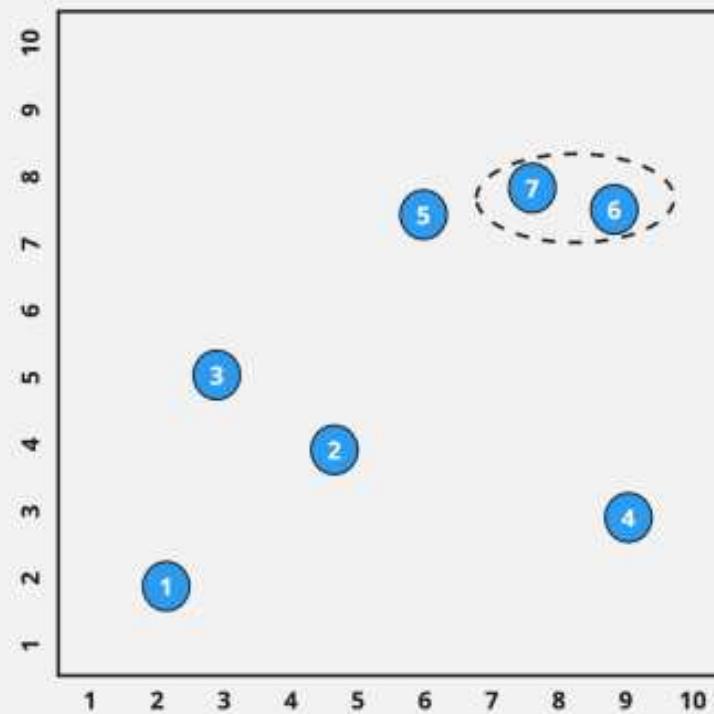


Take the next two nearest clusters and make them one cluster

Step 01

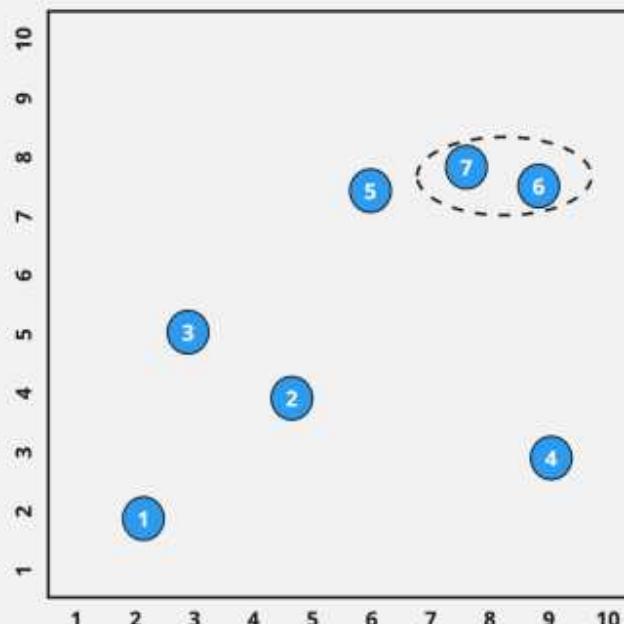


Step 02

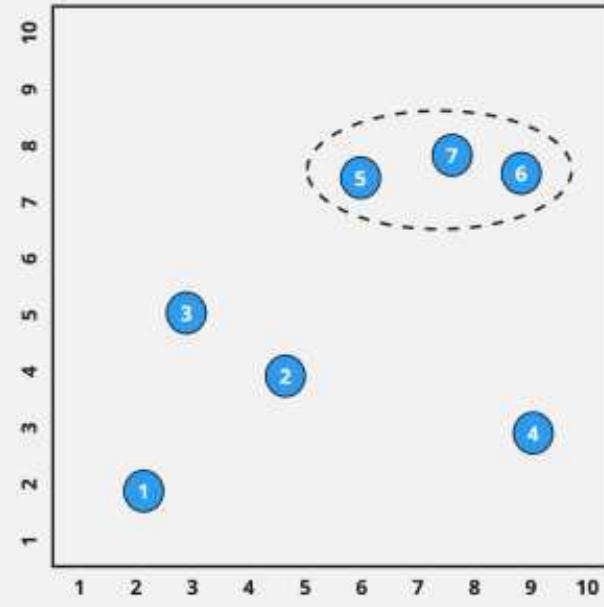


Again, take the two nearest clusters and make them one cluster

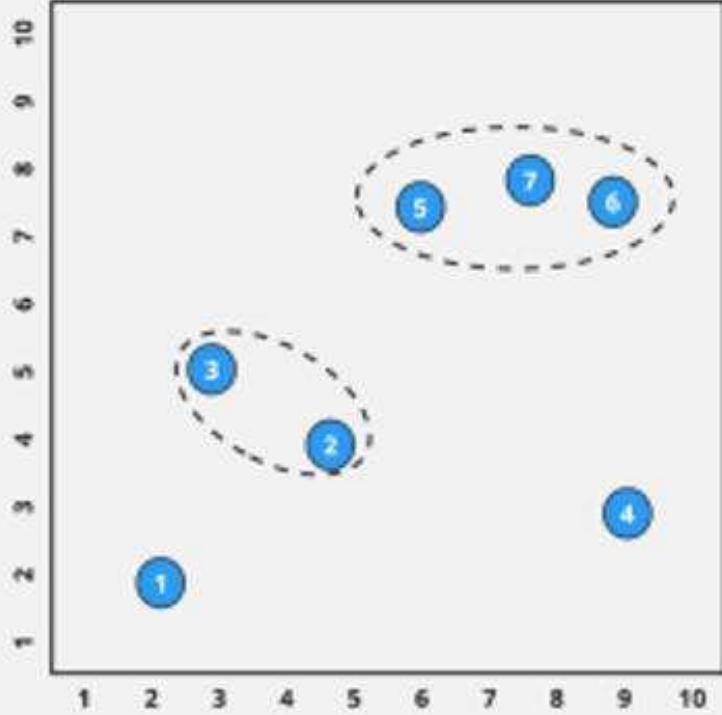
Step 02



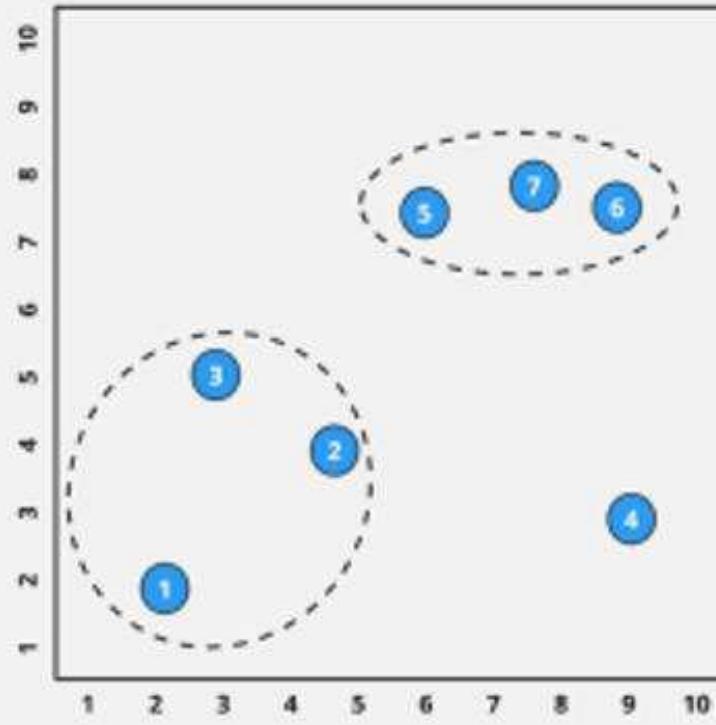
Step 03



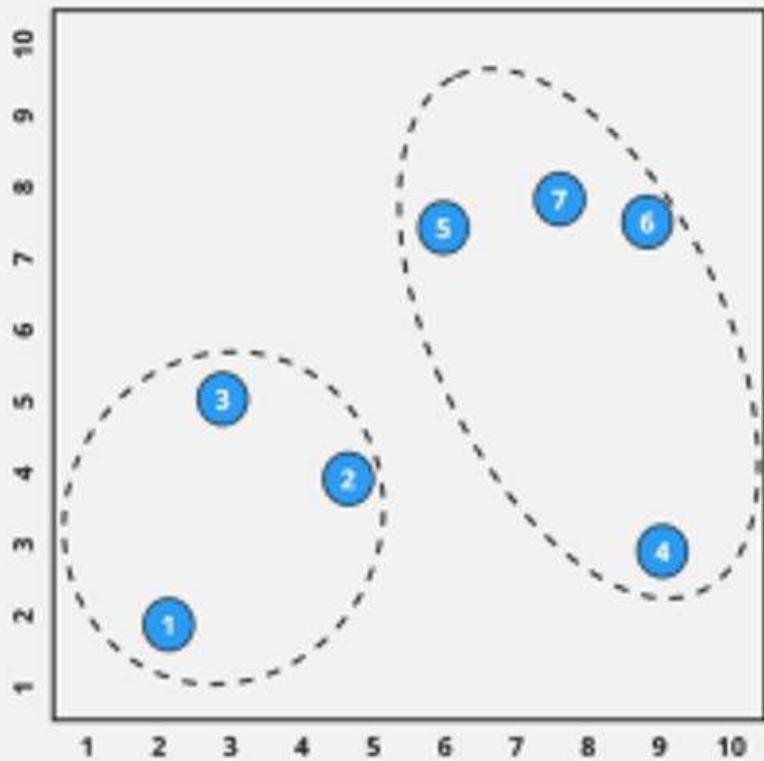
**Step 04**



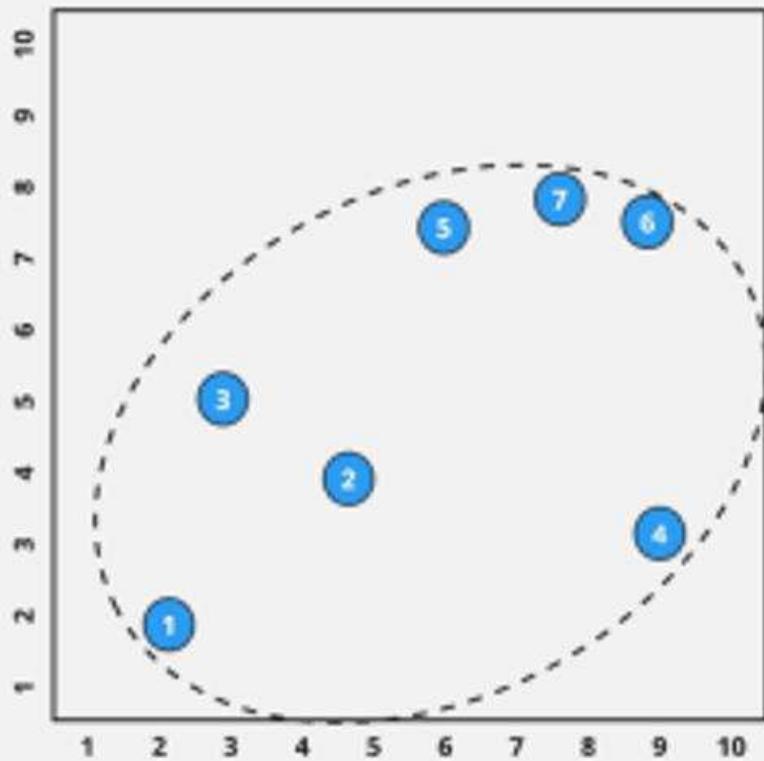
**Step 05**



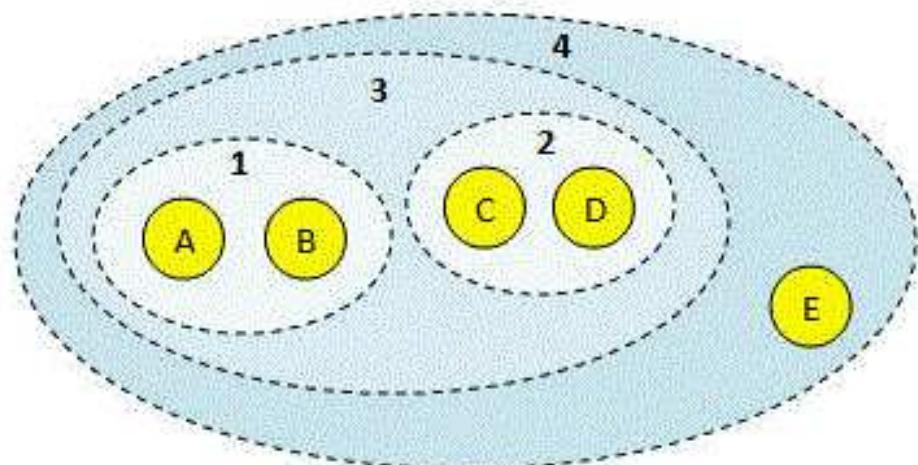
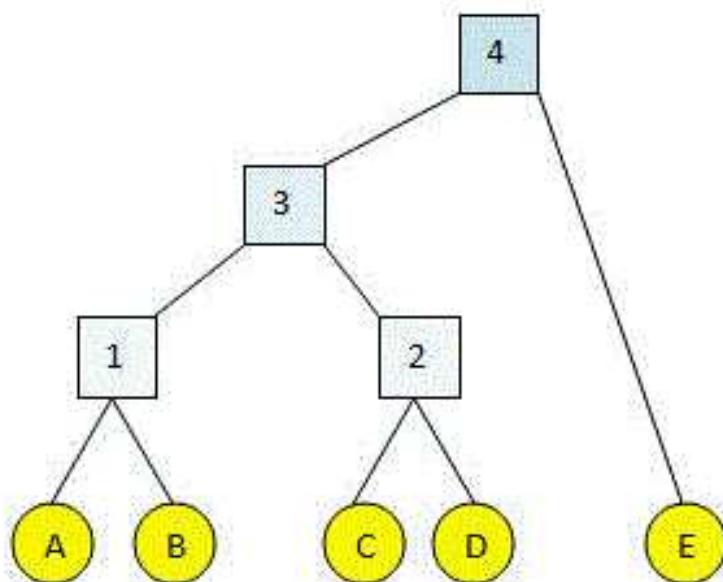
**Step 06**



**Step 07**



## Example 2

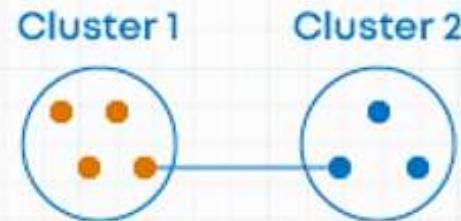


- What is missing here?
- How to define the similarity of two clusters?

- **Single Linkage**

$$D(c_1, c_2) = \min D(x_i, x_j)$$

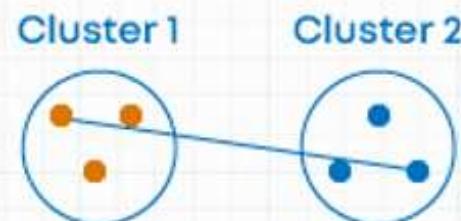
Minimum distance or distance between closest elements in clusters



- **Complete Linkage**

$$D(c_1, c_2) = \max D(x_i, x_j)$$

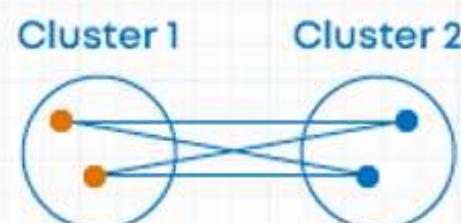
Maximum distance between elements in clusters



- **Average Linkage**

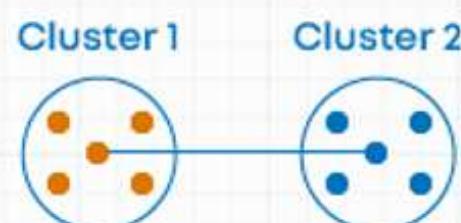
$$D(c_1, c_2) = \frac{1}{|c_1|} \frac{1}{|c_2|} \sum \sum D(x_i, x_j)$$

Average of the distances of all pairs



- **Centroid Method**

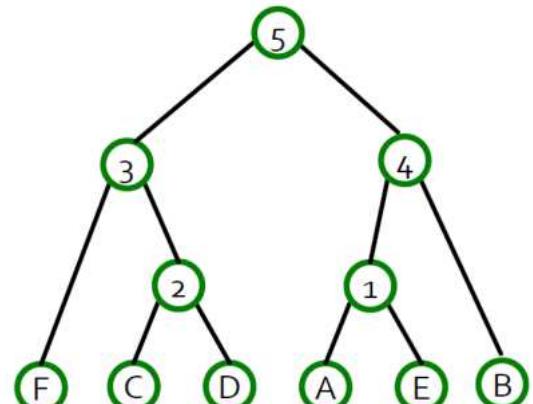
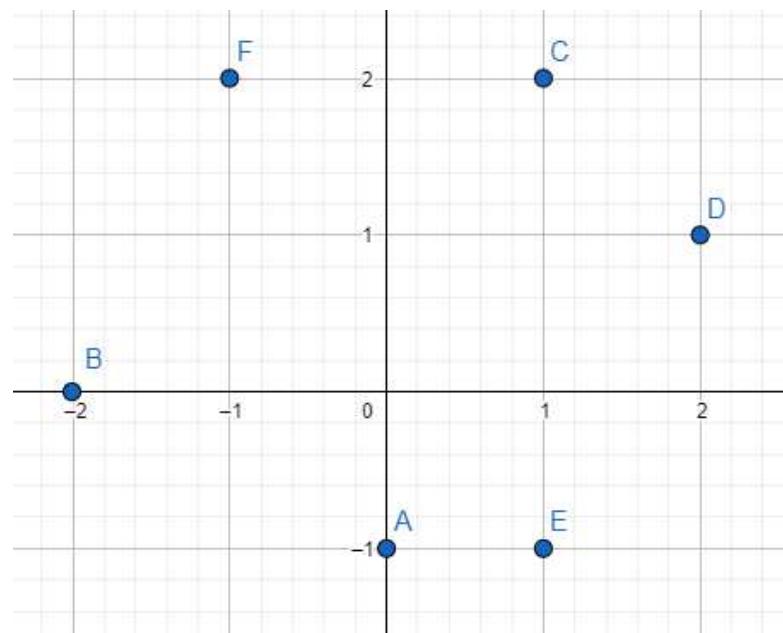
Combining clusters with minimum distance between the centroids of the two clusters



- We will discuss the following calculation example in the tutorial class:

Data points

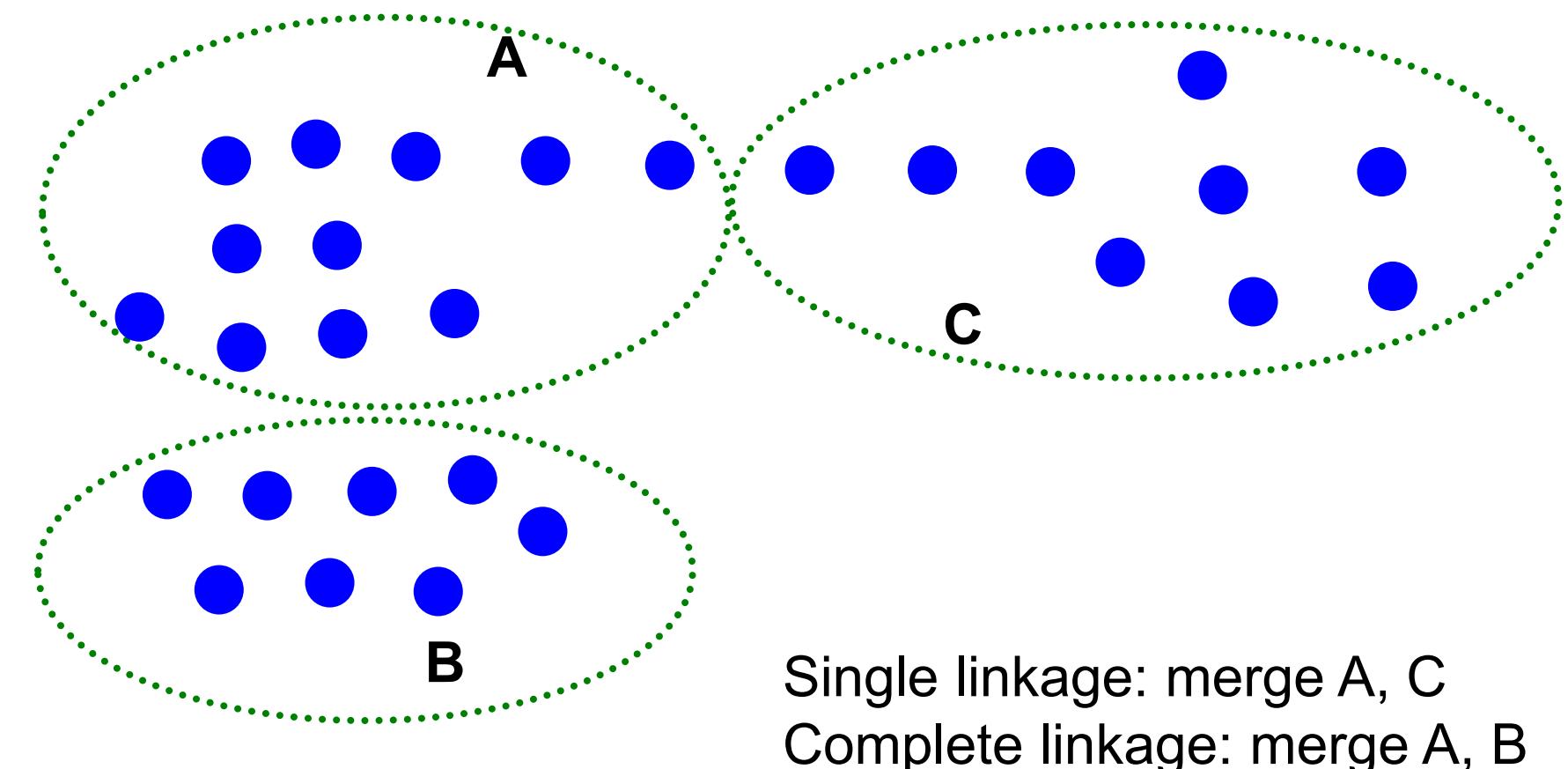
A	$[0, -1]$
B	$[-2, 0]$
C	$[1, 2]$
D	$[2, 1]$
E	$[1, -1]$
F	$[-1, 2]$

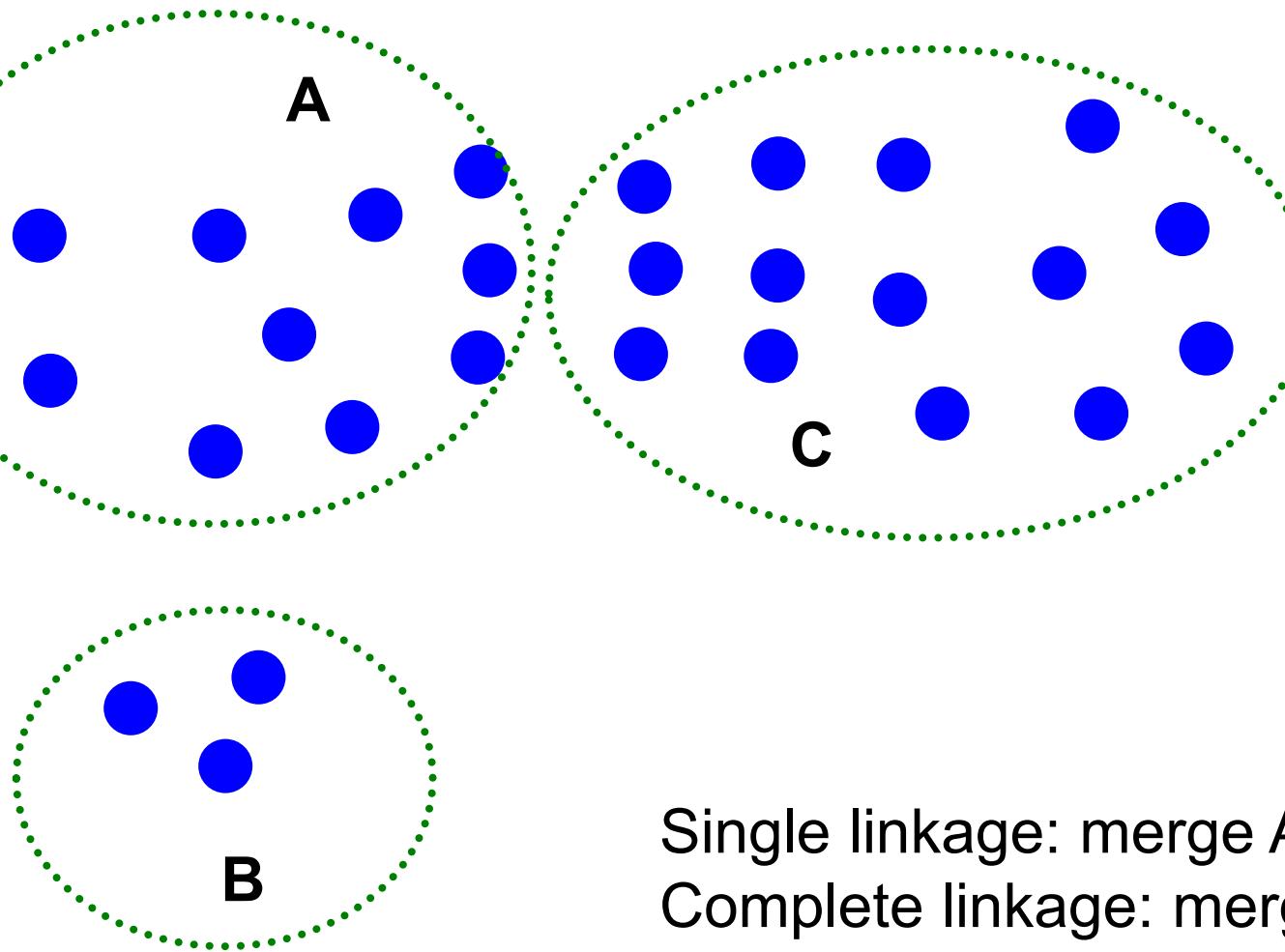


- When do we stop merging clusters?
  - When some number  $k$  of clusters are found
  - Keep merging until there is only 1 cluster

# Hierarchical Clustering

- Pros
  - No need to set K: the number of clusters
  - No assumption on the shape of the cluster
  - Simple
  
- Cons
  - The algorithm can never undo the grouping.  
The data points may be incorrectly grouped at an earlier stage.
  
  - Different distance metrics may produce very different results.
    - Simple Linkage methods are sensitive to noise and outliers.
    - Complete linkage methods tend to break large clusters.





# Link Analysis: PageRank

Lin Guosheng  
School of Computer Science and Engineering  
Nanyang Technological University

# Outline

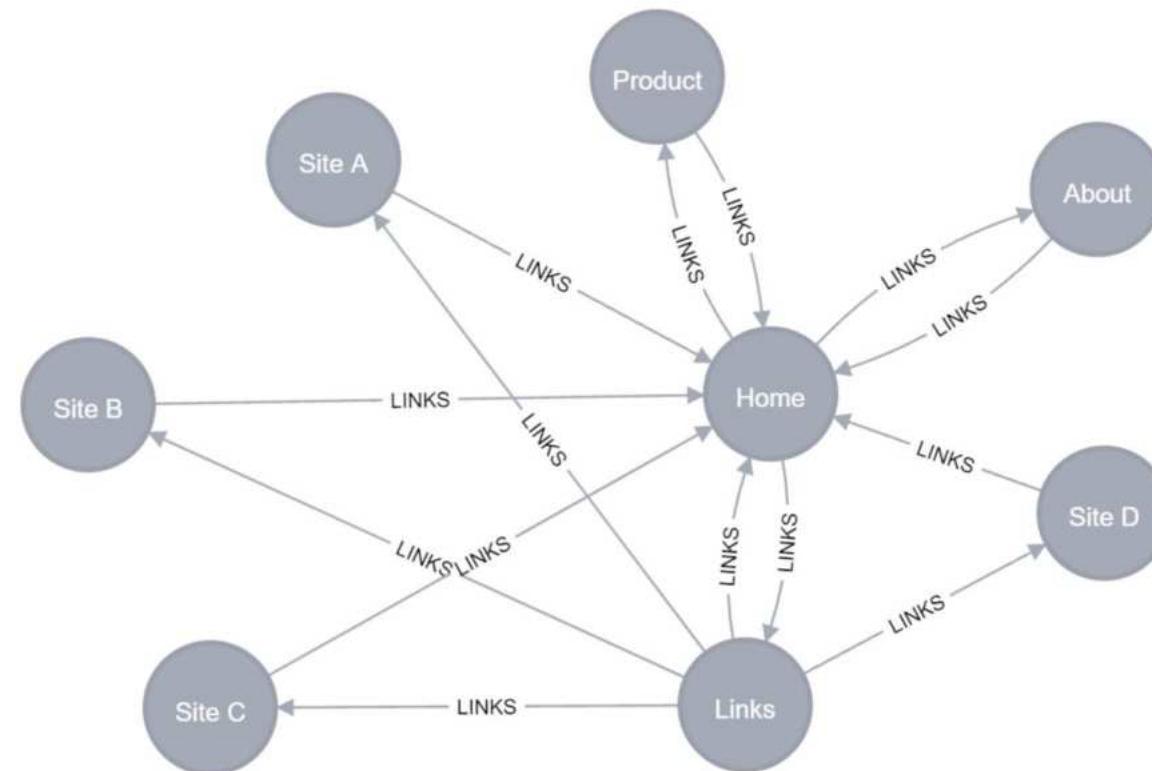
- PageRank Algorithm
  - Page rank
  - Page rank extensions
    - Potential problems
    - Teleportation
    - Page rank – Google Matrix

Many slides are from:

<http://web.stanford.edu/class/cs224w/slides/o4-pagerank.pdf>

Web as a directed graph:

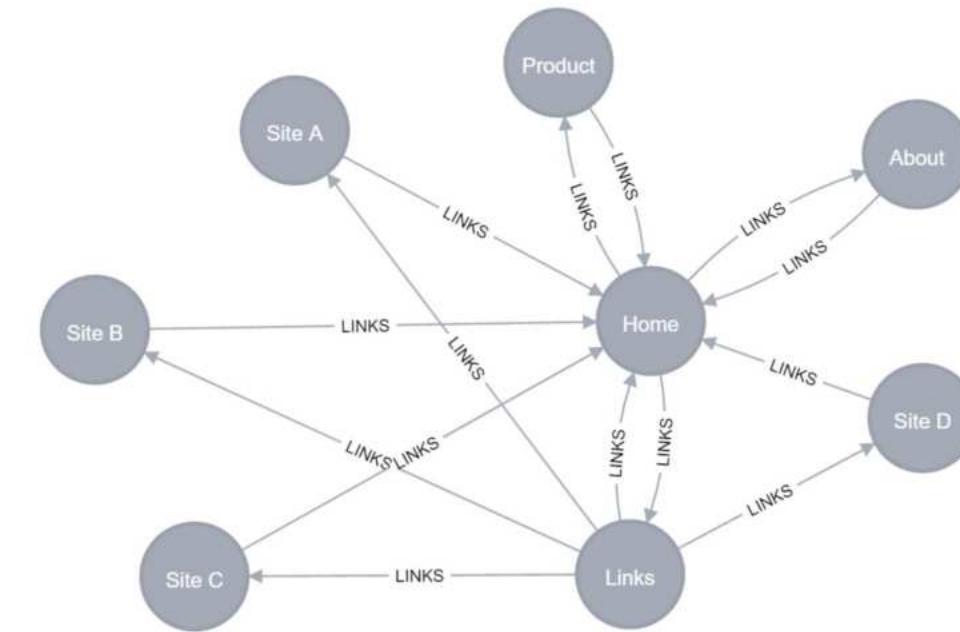
- Nodes = web pages
- Edges with directions = hyperlinks (in-links and out-links)



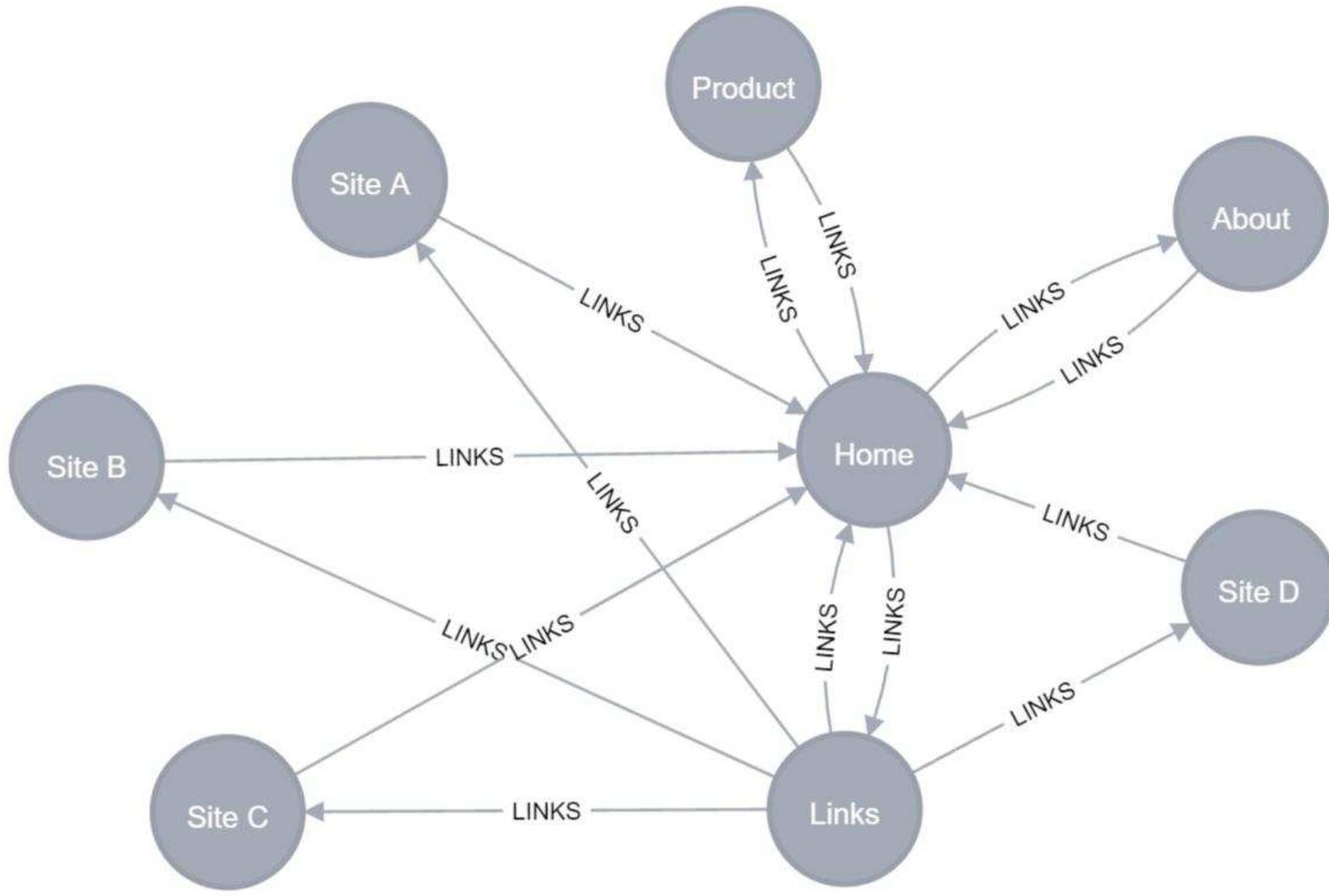
Represent the web as directed graph

- Link analysis is to discover useful information from relationships/connections between data objects. (mining the graph data)
- Use graphs to describe connections.
  - Vertices represent objects/data points
  - Links describe relationships among objects.
- PageRank is an algorithm to rank web pages.
  - An algorithm for link analysis and web search.
  - proposed by Sergey Brin and Larry Page,
    - students at Stanford University and the founders of Google.

- All web pages are not equally “important”
  - <https://guosheng.github.io/>
  - <https://www.ntu.edu.sg/> (More important)
- How to measure the importance of web pages?



- How to measure the importance of web pages?
  - We can look at the links.
    - Q1: In-coming links vs out-going links?
    - Q2: Are all links equally important?



## Idea: Links as votes

- Page is more important if it has more links
  - In-coming links vs Out-going links
    - In-coming links are more important
  - Examples:
    - www.stanford.edu has 23,400 in-links
    - thispersondoesnotexist.com has 1 in-link

## Idea: Links as votes

- Are all in-links equal?
  - In-links from important pages count more
- Summary:
  - Two key factors affecting the rank score:
    - 1. The number of in-links.
    - 2. The source nodes of the in-links

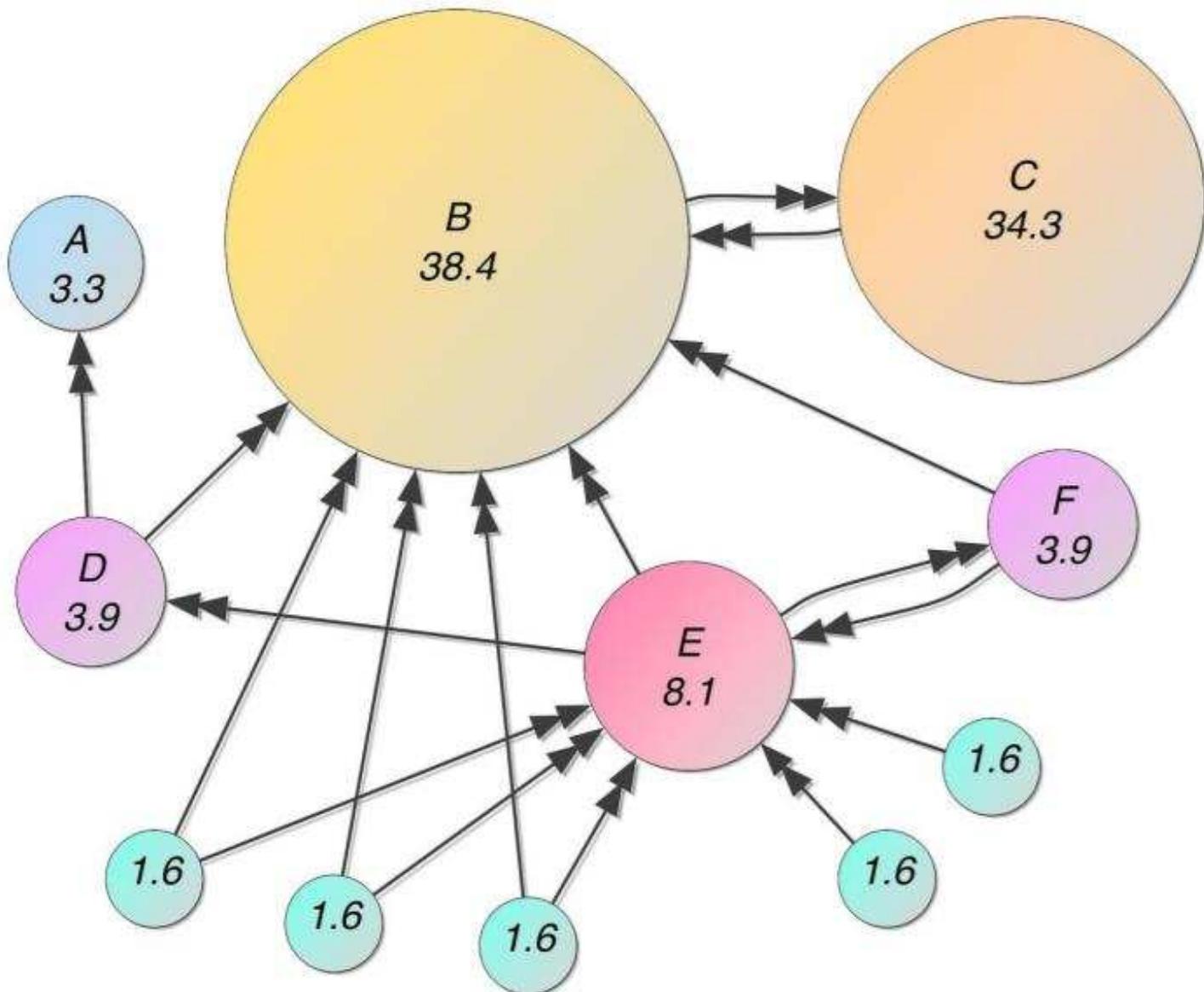


Image credit: [Wikipedia](#)

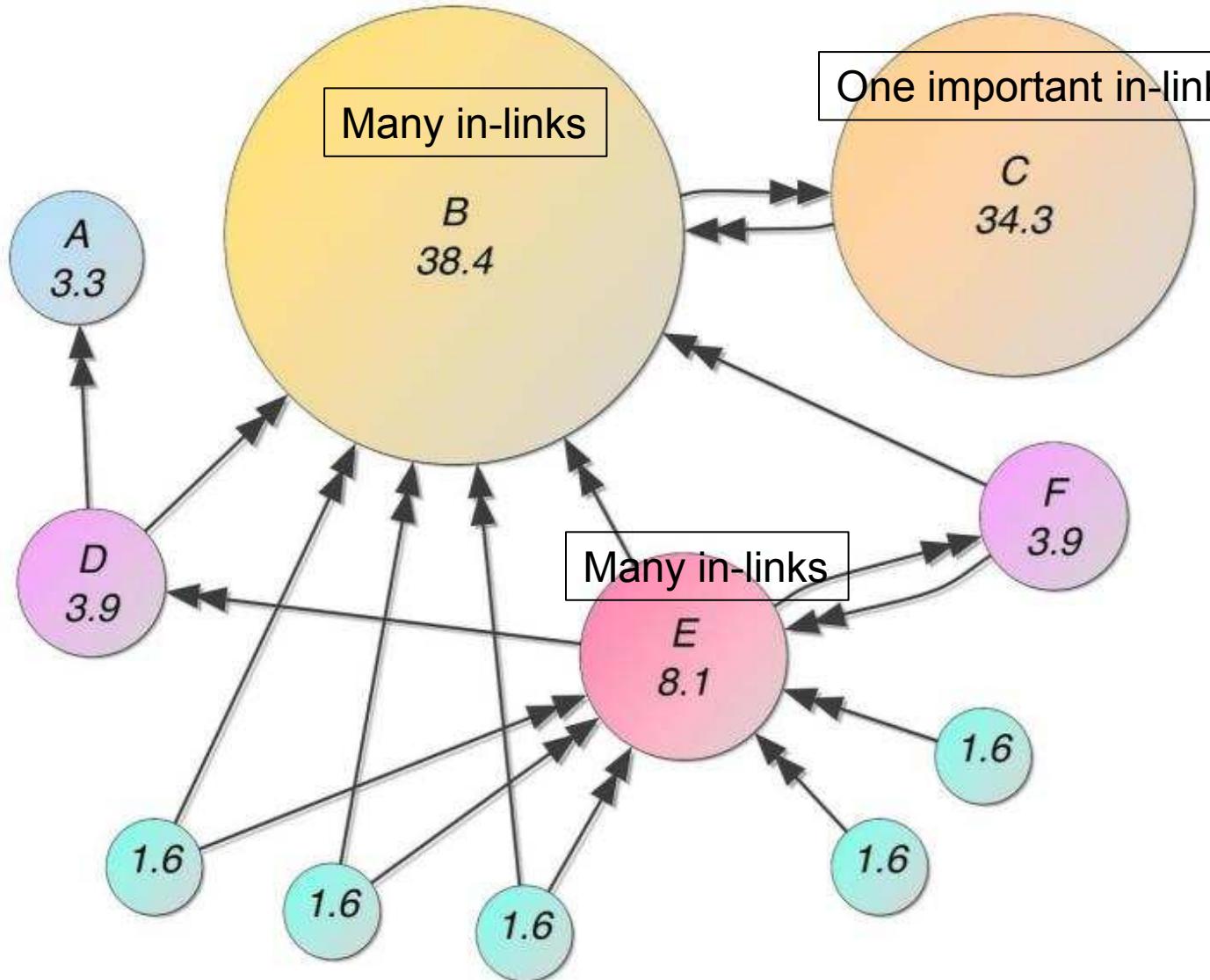


Image credit: [Wikipedia](#)

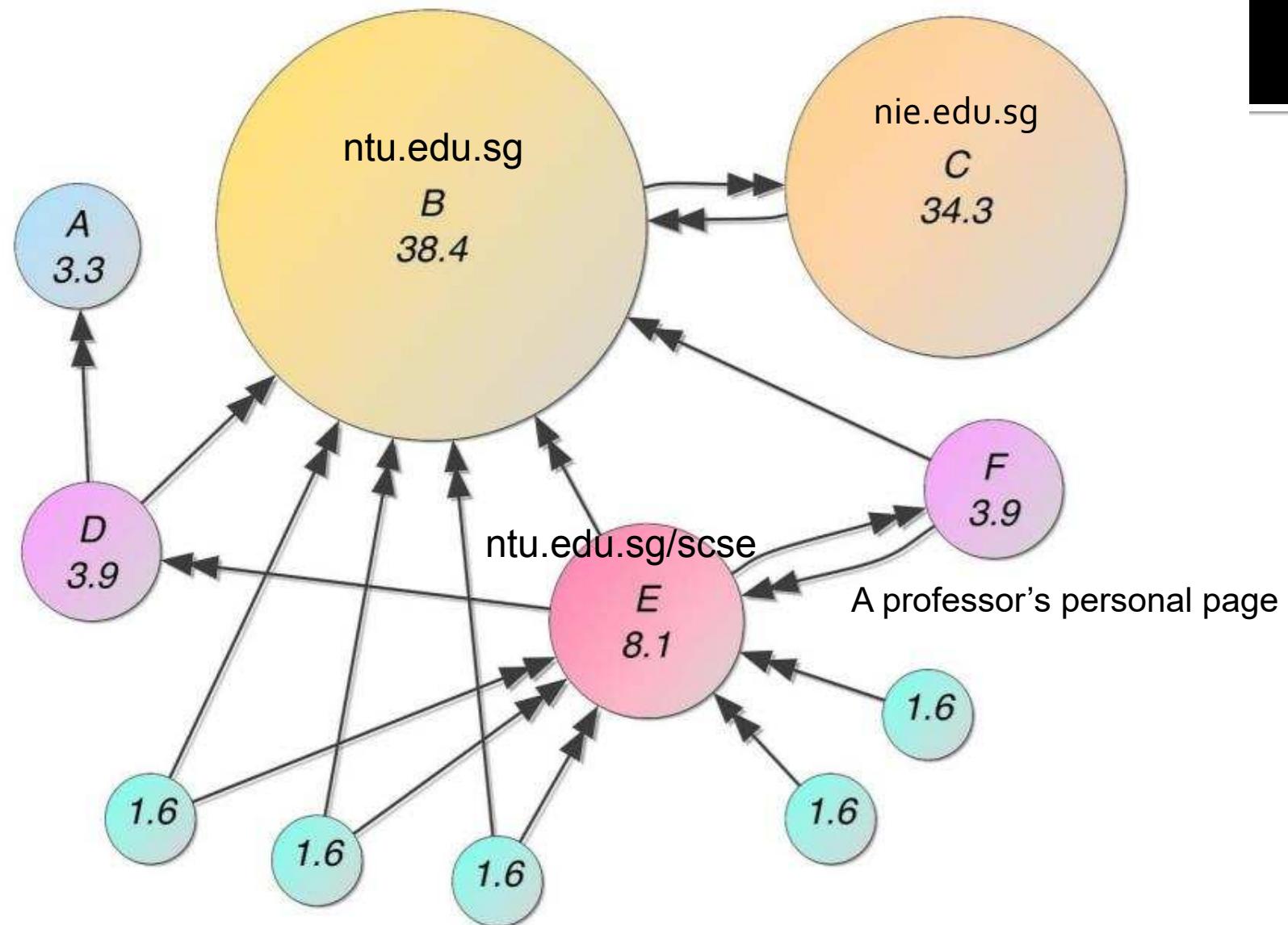
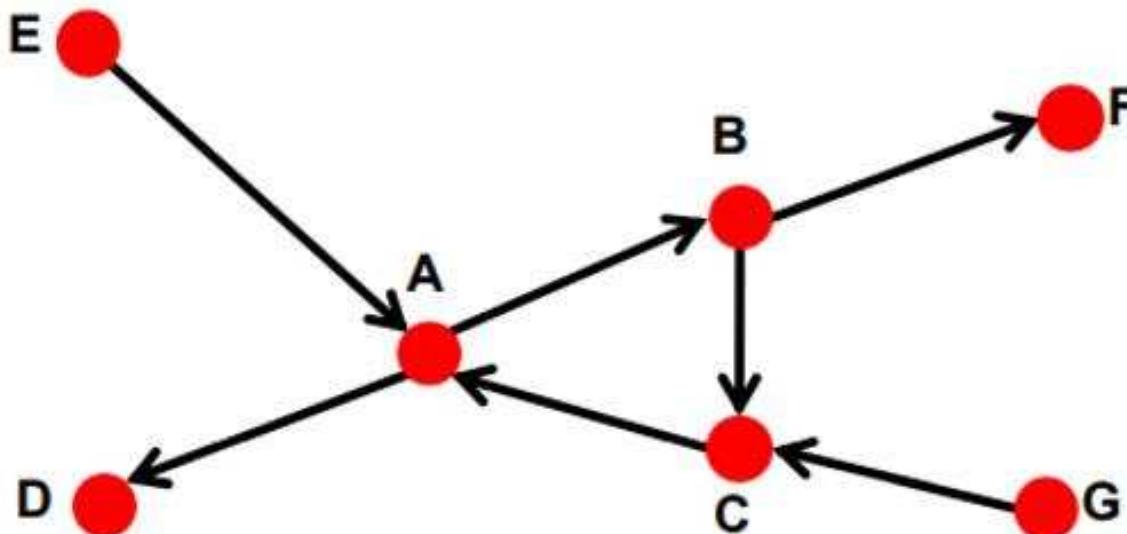


Image credit: [Wikipedia](#)

- In-links of a node (or backlinks)
  - the links pointing in (from other nodes), incoming links
  - in-degree: the number of in-links
- Out-links of a node
  - The links pointing out (to other nodes), outgoing links
  - out-degree: the number of out-links



Examples:

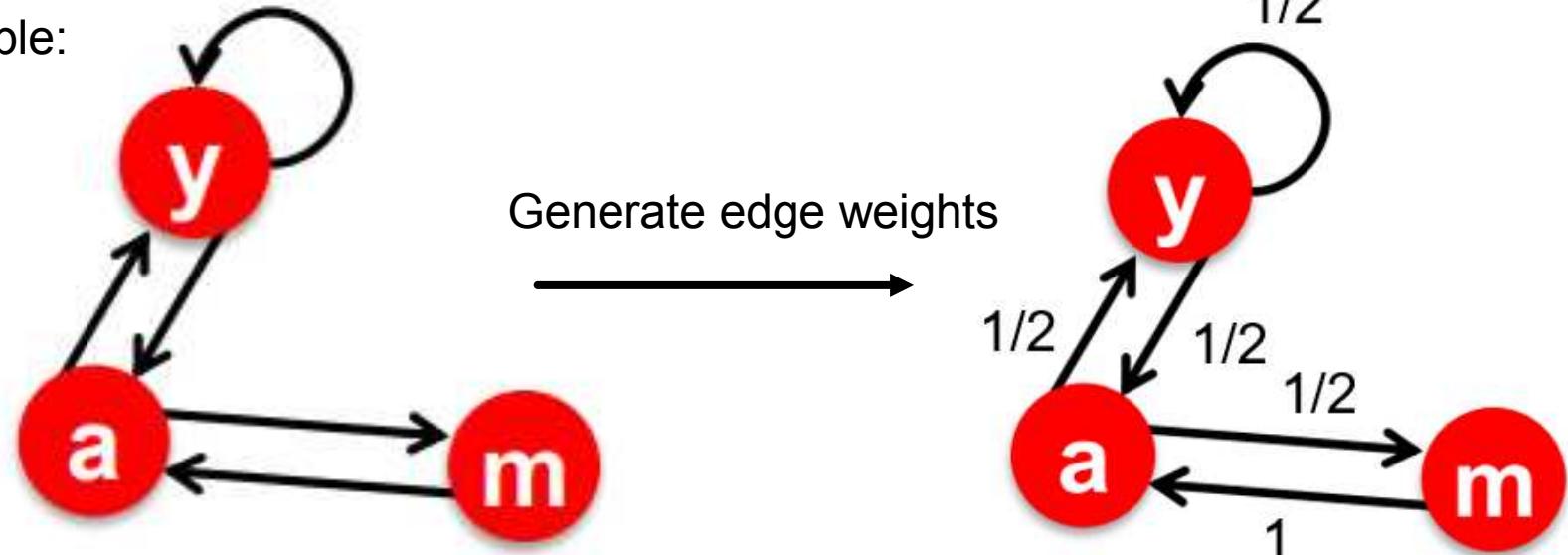
In-degree of node A: 2  
(from E and C)

Out-degree of node A: 2  
(to D and B)

# PageRank

- Define the edge (link) weights as  $w_{ij} = \frac{1}{d_i}$   
 $d_i$ : out-degree of the source node i (the number of out-links)  
use the source node of the link to define the weight.

Example:



Node y: it has two out-links ( $d=2$ ), so the weight for each out-link is  $1/2$

Node a: it has two out-links ( $d=2$ ), so the weight for each out-link is  $1/2$

Node m: it has one out-link ( $d=1$ ), so the weight for each out-link is  $1/1$

# PageRank

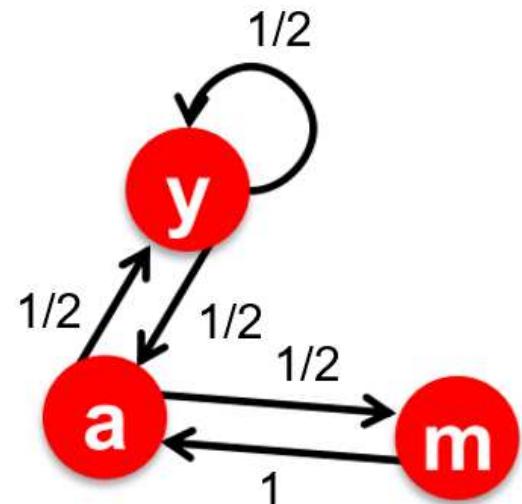
- Define the edge (link) weights as  $w_{ij} = \frac{1}{d_i}$

$d_i$ : out-degree of the source node  $i$  (the number of out-links)  
use the source node of the link to define the weight.

The edge weights will be used to calculate the node importance score (page rank score)

The edge weights indicate the “votes” from the source node to the target node.

Example:



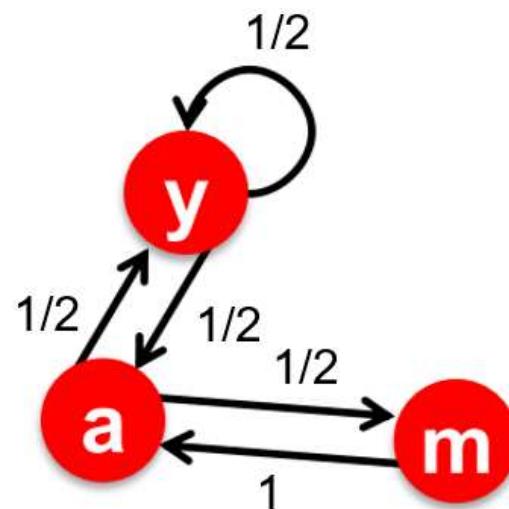
- Define the edge weights as  $w_{ij} = \frac{1}{d_i}$

All the out-links of one node share the same weight

Node y: it has two out-links ( $d=2$ ), so the weight for each out-link is  $1/2$

Node a: it has two out-links ( $d=2$ ), so the weight for each out-link is  $1/2$

Node m: it has one out-link ( $d=1$ ), so the weight for each out-link is  $1/1$



## ■ PageRank: assign a rank score to each node

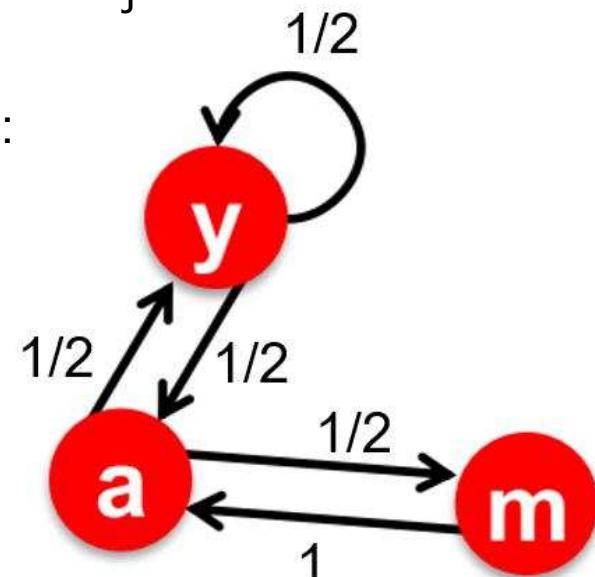
For a node  $j$ , we define the rank score  $r_j$  as  
(flow equations):

$$r_j = \sum_{i \rightarrow j} w_{ij} r_i = \sum_{i \rightarrow j} \frac{1}{d_i} r_i$$

the edge weight:  
(the link from  $i$  to  $j$ )  $w_{ij} = \frac{1}{d_i}$

One node rank score  
= weighted sum of the  
in-linked node scores;

Example:



$$r_y = r_y/2 + r_a/2$$

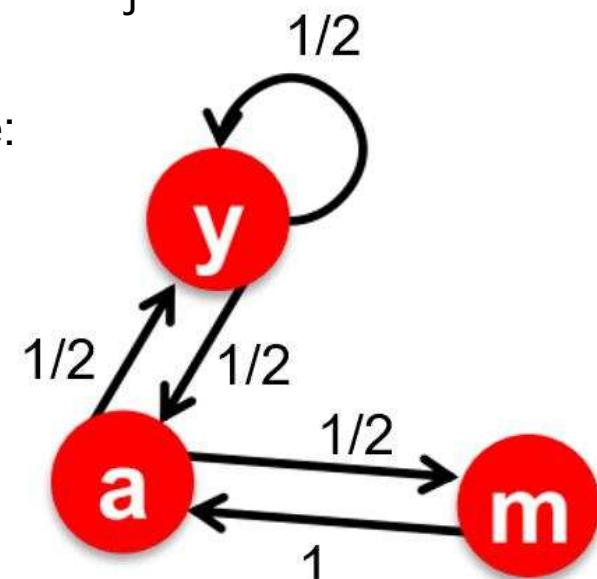
y has two in-links:  
 $a \rightarrow y$  and  $y \rightarrow y$

- PageRank: assign a rank score to each node

For a node  $j$ , we define the rank score  $r_j$  as  
(flow equations):

$$r_j = \sum_{i \rightarrow j} w_{ij} r_i = \sum_{i \rightarrow j} \frac{1}{d_i} r_i$$

Example:



We also require:

(The sum of all node scores equals 1):

$$\sum_i r_i = 1$$

$$r_y = r_y/2 + r_a/2$$

y has two in-links:  
a->y and y->y

Flow equation:

flow-out value = flow-in value

$$r_j = \sum_{i \rightarrow j} w_{ij} r_i = \sum_{i \rightarrow j} \frac{1}{d_i} r_i$$

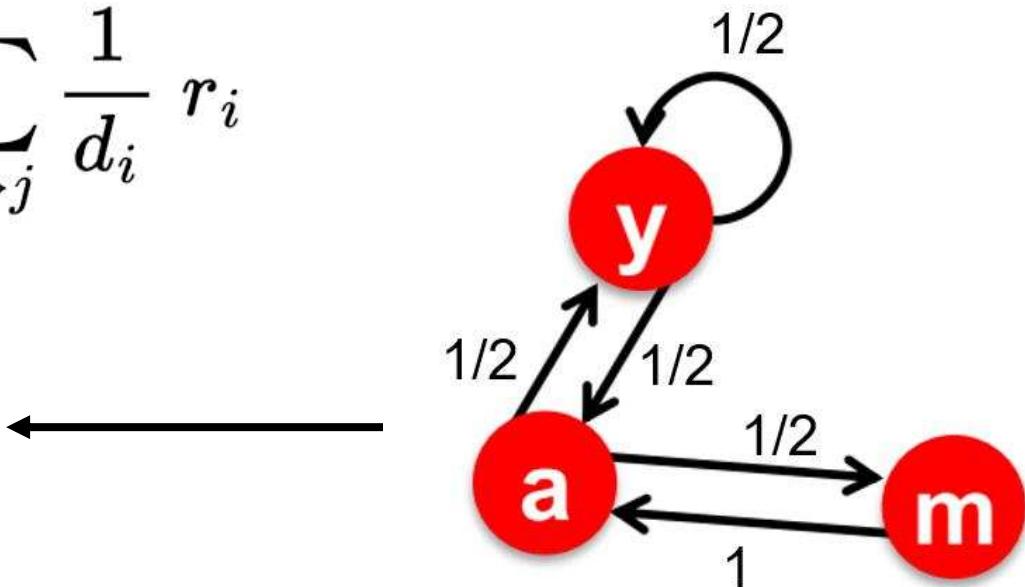
One node score = weighted sum  
of the in-linked node scores;

“Flow” equations:

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$



1. y has two in-links: y->y and a->y; two in-linked nodes: y and a.
2. a has two in-links: y->a and m->a; two in-linked nodes: y and m.
3. m has one in-link: a->m; one in-linked node: a.

- How to solve for the rank scores?

- Solution 1: directly solve the linear equations
- Solution 2: power iteration method

## ■ Solution 1:

Directly solve the linear equations

“Flow” equations:

$$r_y = r_y/2 + r_a/2$$

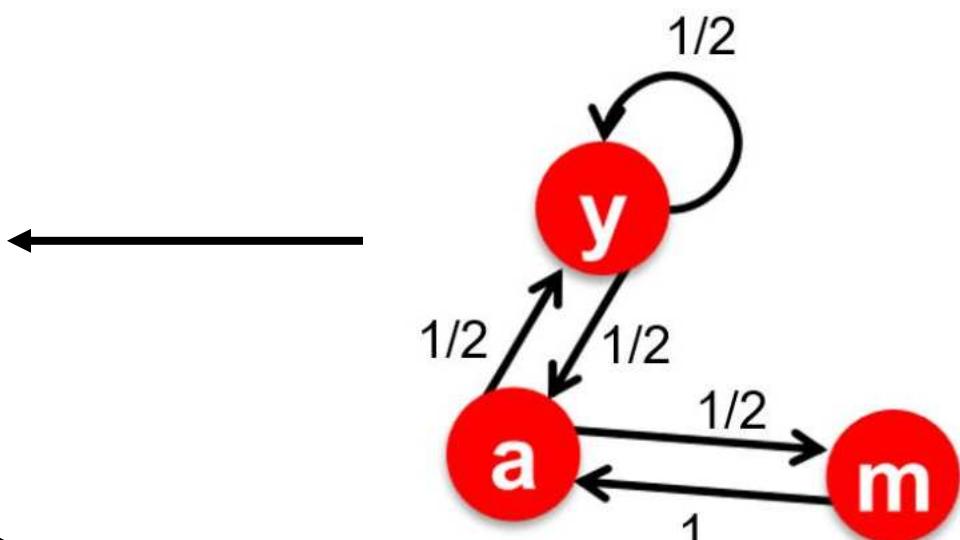
$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

Adding this equation:

$$\sum_i r_i = 1$$

(The sum of all rank scores equals 1)



We can solve these 4 linear equations to get the solutions:  
 $r_y = 0.4$ ;  $r_a = 0.4$ ;  $r_m = 0.2$

## Example using elimination methods to solve linear systems:

- EQ1:  $y = y/2 + a/2$
- EQ2:  $a = y/2 + m$
- EQ3:  $m = a/2$
- EQ4:  $a + y + m = 1$
  
- With EQ2, EQ3  $\rightarrow a = y/2 + a/2$  (eliminate y)  
 $\rightarrow$  EQ5:  $a = y$
  
- With EQ4, EQ3, EQ5  $\rightarrow a + a + a/2 = 1$  (eliminate y, m)
  
- $\rightarrow a = 2/5 = 0.4$
- $\rightarrow y = 0.4, m \rightarrow 0.2$

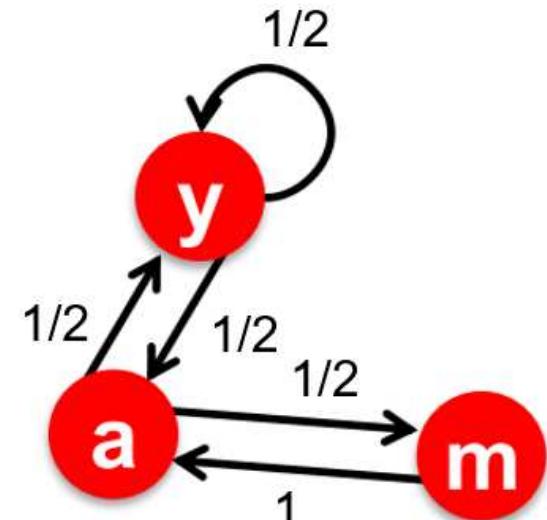
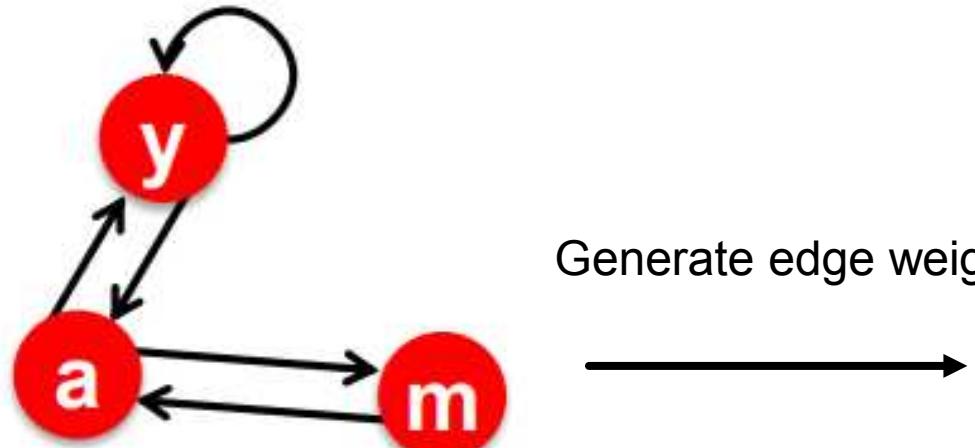
## ■ Solution 2: power iteration method

- Construct transition matrix:  $M$ 
  - Matrix size:  $n \times n$  ( $n$ : the number of nodes)
  - $M$  is constructed by the edge weights.

If there is a link from  $i$  to  $j$  ( $i \rightarrow j$ ),

$$M_{j,i} = \frac{1}{d_i}$$

$d_i$  is the out-degree of node  $i$



If there is a link from  $i$  to  $j$  ( $i \rightarrow j$ ),  $M_{j,i} = \frac{1}{d_i}$

$d_i$  is the out-degree of node  $i$

Construct M column by column or row by row:

1. column by column:

Each column in M indicates out-links for one node

2. row by row:

Each row in M indicates the in-links for one node

	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	0
$r_a$	$\frac{1}{2}$	0	1
$r_m$	0	$\frac{1}{2}$	0

Transition matrix M

- Convert flow equations into matrix multiplication

The flow equation of one node ( $j=1, 2, \dots, n$ ):

$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

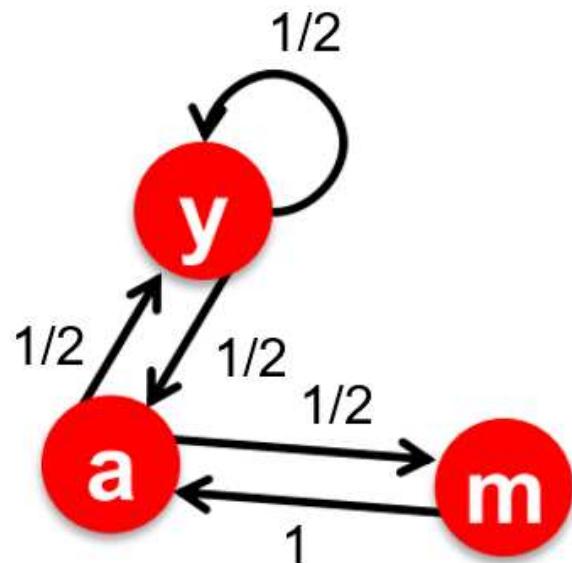
Matrix expression  
of all flow equations:



$$\mathbf{r} = \mathbf{M} \cdot \mathbf{r}$$

Rank vector  $\mathbf{r}$ : a column vector of all rank scores.

$\mathbf{r} = [r_1, r_2, r_3, \dots, r_n]^T$ .     $n$ : the total number of nodes.



	$r_y$	$r_a$	$r_m$
$r_y$	1/2	1/2	0
$r_a$	1/2	0	1
$r_m$	0	1/2	0

Transition matrix M

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

Matrix expression

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

**$r$**        **$M$**        **$r$**

## ■ Solution 2: Power iteration method

- 1. assign initial values to rank scores :  $r_i = 1/N$ ;
  - N is the total number of nodes
  - Uniform distribution
- 2. repeat the following util converge:

Calculate the page rank:  $r^{(t+1)} = M \cdot r^{(t)}$

Or equivalently update each node  
using the flow equation:

$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

Converge criteria:  $(\sum_i |r_i^{t+1} - r_i^t| < \epsilon)$

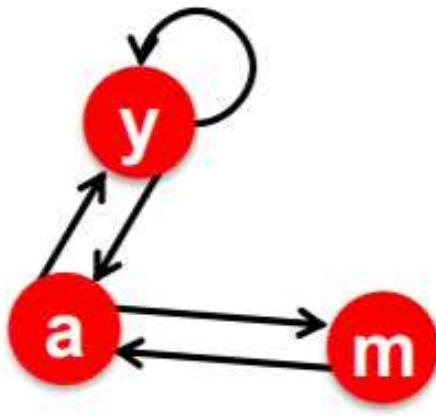
$\epsilon$  : is a pre-defined small value, e.g., 0.0001

Or converge when reach the  
maximum number of iterations

## ■ Power Iteration:

- Set  $\mathbf{r}_j \leftarrow 1/N$
- 1:  $\mathbf{r}'_j \leftarrow \sum_{i \rightarrow j} \frac{r_i}{d_i}$
- 2: If  $|\mathbf{r} - \mathbf{r}'| > \varepsilon$ :
  - $\mathbf{r} \leftarrow \mathbf{r}'$
- 3: go to 1

$\mathbf{r}'$  is the updated score vector  $\mathbf{r}^{(t+1)}$



	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

Initialization (uniform distribution):  $r_y=1/3$ ,  $r_a=1/3$ ,  $r_m=1/3$  ;

**Iteration 1:** (Flow equation update)

$$r'_y = r_y/2 + r_a/2 = 1/3 * 1/2 + 1/3 * 1/2 = 1/3;$$

$$r'_a = r_y/2 + r_m = 1/3 * 1/2 + 1/3 = 3/6;$$

$$r'_m = r_a/2 = 1/3 * 1/2 = 1/6;$$

Update  $r$ :  $r_y=1/3$ ,  $r_a=3/6$ ,  $r_m=1/6$  ;

Or we can use matrix update:

$$r^{(t+1)} = M \cdot r^{(t)}$$

	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	0
$r_a$	$\frac{1}{2}$	0	1
$r_m$	0	$\frac{1}{2}$	0

Transition matrix M

**Iteration 1:**

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

$M$                      $r_0$                      $r_1$

Matrix update:

$$r^{(t+1)} = M \cdot r^{(t)}$$

**Iteration 1:**

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

**M**                    **r<sub>0</sub>**                    **r<sub>1</sub>**

Flow equation update:

$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

Initialization:  $r_y=1/3, r_a=1/3, r_m=1/3$  ;

**Iteration 1:**

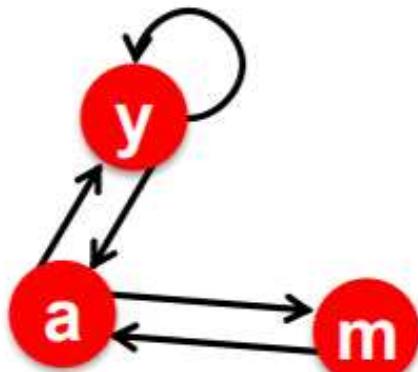
$$r'_y = r_y/2 + r_a/2 = 1/3 * 1/2 + 1/3 * 1/2 = 1/3;$$

$$r'_a = r_y/2 + r_m = 1/3 * 1/2 + 1/3 = 3/6;$$

$$r'_m = r_a/2 = 1/3 * 1/2 = 1/6;$$

Update r:  $r_y=1/3, r_a=3/6, r_m=1/6$  ;

They give the same result



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	1
m	0	$\frac{1}{2}$	0

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

The node scores after iteration 1:  $r_y=1/3$ ,  $r_a=3/6$ ,  $r_m=1/6$  ;

## Iteration 2:

$$r'_y = r_y/2 + r_a/2 = 1/3 * 1/2 + 3/6 * 1/2 = 5/12;$$

$$r'_a = r_y/2 + r_m = 1/3 * 1/2 + 1/6 = 1/3;$$

$$r'_m = r_a/2 = 3/6 * 1/2 = 3/12;$$

Update  $r$ :  $r_y=5/12$ ,  $r_a=1/3$ ,  $r_m=3/12$  ;

The node scores after iteration 2:  $r_y=5/12$ ,  $r_a=1/3$ ,  $r_m=3/12$  ;

### **Iteration 3:**

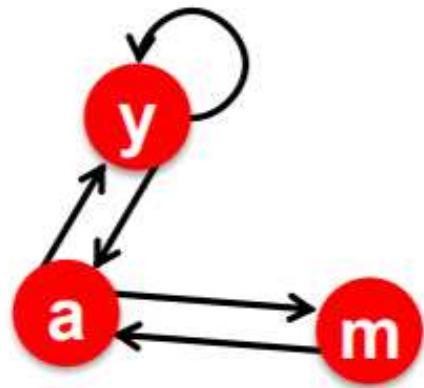
$$r'_y = r_y/2 + r_a/2 = 5/12 * 1/2 + 1/3 * 1/2 = 9/24;$$

$$r'_a = r_y/2 + r_m = 5/12 * 1/2 + 3/12 = 11/24;$$

$$r'_m = r_a/2 = 1/3 * 1/2 = 1/6;$$

Update  $r$ :  $r_y=9/24$ ,  $r_a=11/24$ ,  $r_m=1/6$  ;

**Iteration 4, 5, 7, ...., until converged**



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	1
m	0	$\frac{1}{2}$	0

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

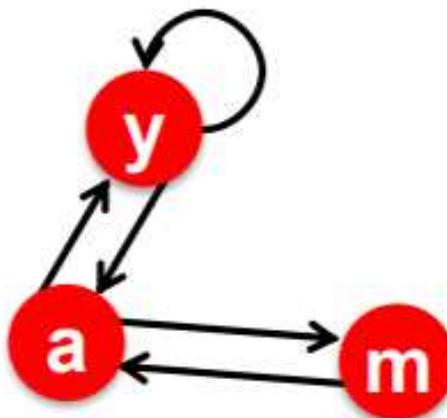
$$r_m = r_a/2$$

Summarized the results for solution2:

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \quad \begin{bmatrix} 1/3 \\ 3/6 \\ 1/6 \end{bmatrix} \quad \begin{bmatrix} 5/12 \\ 1/3 \\ 3/12 \end{bmatrix} \quad \begin{bmatrix} 9/24 \\ 11/24 \\ 1/6 \end{bmatrix} \quad \dots \quad \begin{bmatrix} 6/15 \\ 6/15 \\ 3/15 \end{bmatrix}$$

Initial value:  $1/N$  ( $N=3$ )      iter1      iter2      iter3      Converged

■ Verify your result:



	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 6/15 \\ 6/15 \\ 3/15 \end{bmatrix}$$



$$\begin{aligned} r_y &= r_y/2 + r_a/2 \\ r_a &= r_y/2 + r_m \\ r_m &= r_a/2 \end{aligned}$$

Substitute your result into the flow equations to verify whether the equations hold or not

## ■ Compare solution 1 an 2 (same results):

**Solution 1:** directly solve the linear equations

“Flow” equations:

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

$$\sum_i r_i = 1$$



We can solve these 4 linear equations to get the solutions:

$$r_y = 0.4; r_a = 0.4; r_m = 0.2$$

**Solution 2:** power iteration method

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \begin{bmatrix} 1/3 \\ 3/6 \\ 1/6 \end{bmatrix} \begin{bmatrix} 5/12 \\ 1/3 \\ 3/12 \end{bmatrix} \begin{bmatrix} 9/24 \\ 11/24 \\ 1/6 \end{bmatrix} \dots \begin{bmatrix} 6/15 \\ 6/15 \\ 3/15 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.4 \\ 0.2 \end{bmatrix}$$

Initial value:  $1/N (N=3)$       iter1      iter2      iter3      Converged

- Discussion
  - Power iteration method may need many iterations to converge.
  - Why we still need power iteration method?

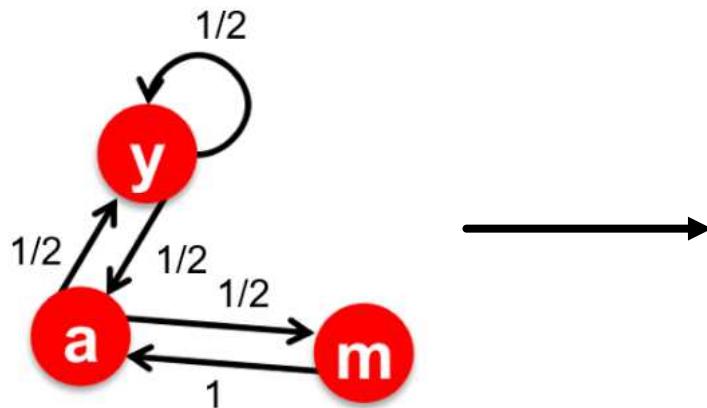
## ■ Discussion

- Why we still need power iteration method?
- It can handle large scale problems.
  - In practice, we have a huge transition matrix. Directly solving a large number of linear equations is computational expensive or maybe even intractable.
- In power iteration method, we can control the number of iterations to get an approximate solution. Usually, the approximated solution will be good enough.

- Properties of the transition matrix: M

- Also call stochastic adjacency matrix
- M should be a column stochastic matrix:
  - each column sums to 1; all elements  $\geq 0$

Each column in M indicates the weights of out-links for one node



	$r_y$	$r_a$	$r_m$
$r_y$	1/2	1/2	0
$r_a$	1/2	0	1
$r_m$	0	1/2	0

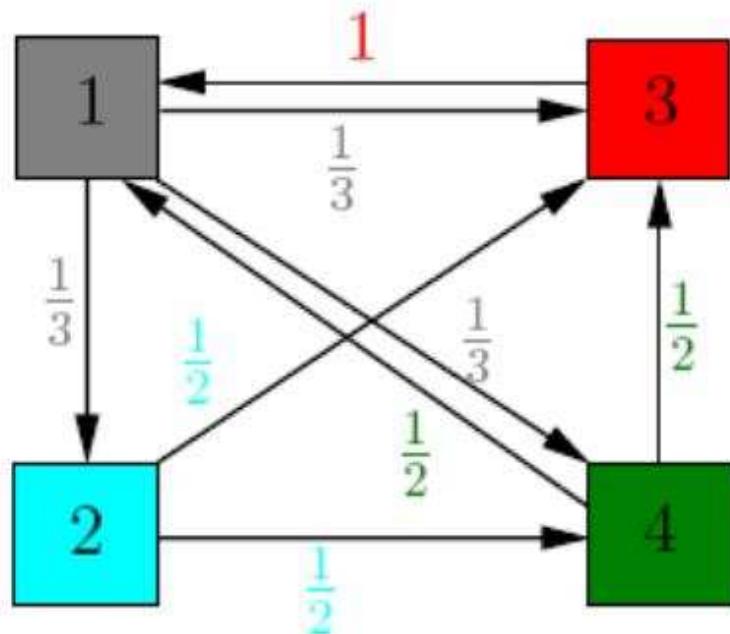
Transition matrix M

The edge weight is defined by the number of out-links (d)  $\frac{1}{d_i}$

All the out-links of one node share the same weight

- > non-zero cell will take the same value for each column
- > the sum of a column is:  $d \cdot (1/d) = 1$ .

- More examples will be discussed in the tutorial class



Page rank score for node 1,2,3,4:

$$\begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}$$

- PageRank extensions
  - “Dead-end” problem
  - “Spider-trap” problem
  - The Google matrix

- Two problems:
  - 1) the dead-end problem
    - Dead-end nodes
  - 2) the spider trap problem
    - Spider trap groups

1. The dead-end problem:  
some nodes are dead ends (have no out-links)



Node b is a dead-end node

Use the power iteration method:

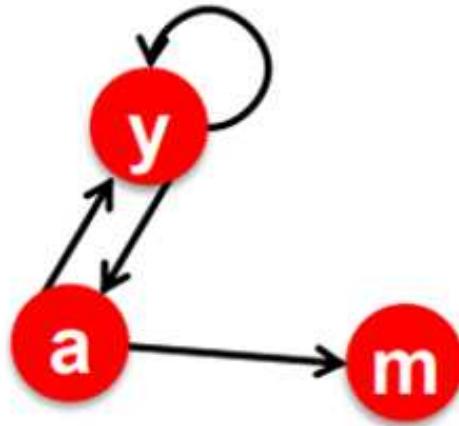
	init	Iter 1	Iter 2
Node a	0.5	0	0
Node b	0.5	0.5	0

Transition Matrix: M

	a	b
a	0	0
b	1	0

$$r^{(t+1)} = M \cdot r^{(t)}$$

The sum of the all node scores for each iteration is reducing  
(the sum should be 1 if there is no score leaking)



Transition Matrix: M

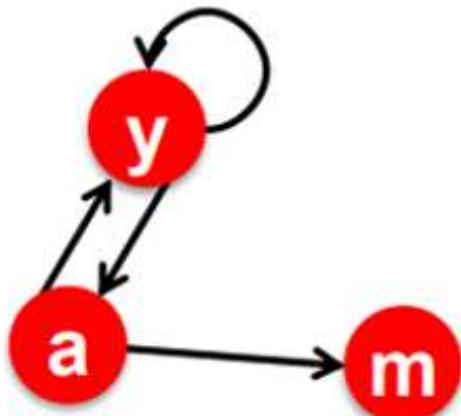
	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	0

Node m is a dead-end node

$$r^{(t+1)} = M \cdot r^{(t)}$$

	init	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	...	
Node y	1/3	2/6	3/12	5/24	8/48	13/96	...	0
Node a	1/3	1/6	2/12	3/24	5/48	8/96	...	0
Node m	1/3	1/6	1/12	1/24	3/48	5/96	...	0

Score leaking: the sum of each column is reducing.



Transition Matrix: M

	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	0

Node m is a dead-end node

$$r^{(t+1)} = M \cdot r^{(t)}$$

Flow equations:

$$r_y = \frac{1}{2}r_y + \frac{1}{2}r_a$$

$$r_a = \frac{1}{2}r_y$$

$$r_m = \frac{1}{2}r_a$$

$$r_y + r_a + r_m = 1$$

There is no solution

If only consider the top 3 equations,  
the solution is 0 for each node.

The dead-end problem:



	a	b
a	0	0
b	1	0

Transition Matrix: M

The problem of M:

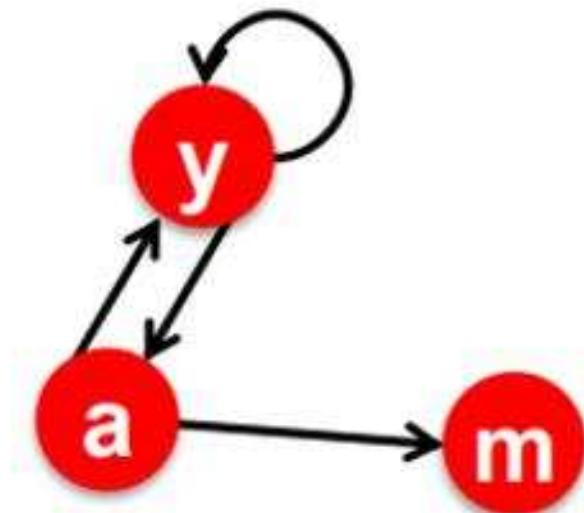
It's not a column stochastic matrix, so it cannot meet the assumption of the PageRank algorithm.

A column stochastic matrix should satisfy:  
each column sums to 1; all elements  $\geq 0$   
(stochastic here means probability)

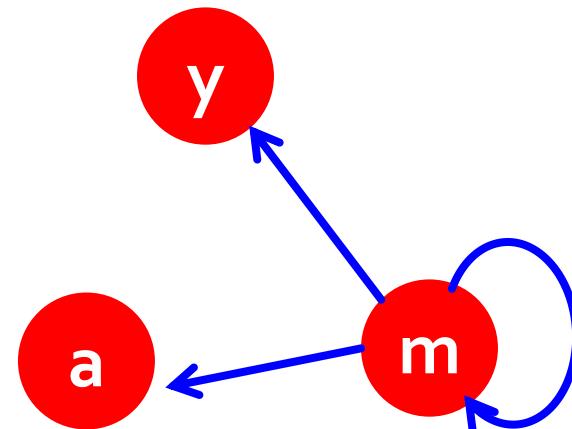
## ■ Solution to Dead End:

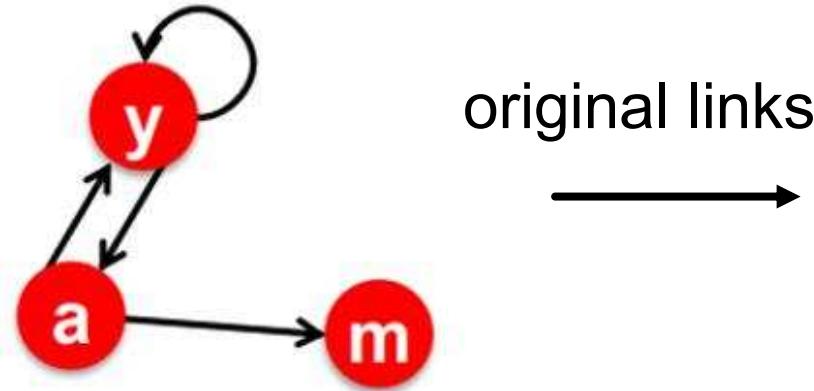
- Add teleport links for the dead-end nodes
  - teleport links:  
the links from the dead-end node to all other nodes
- Fix the transition matrix by adding the teleport links

original links



Teleport links for node m

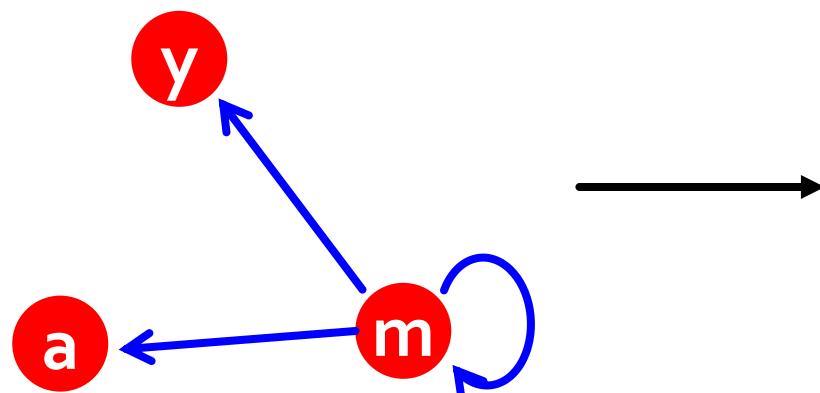




	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	0

Transition matrix M  
(original links)

Teleport links for node m



	y	a	m
y	0	0	$\frac{1}{3}$
a	0	0	$\frac{1}{3}$
m	0	0	$\frac{1}{3}$

Transition matrix Q  
(Teleport links)

teleport links: the links from the dead-end node to all other nodes

- Adding the teleport links to the original graph
  - Update the transition matrix

	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	0

Transition matrix M  
(original links)

New transition matrix

	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$
a	$\frac{1}{2}$	0	$\frac{1}{3}$
m	0	$\frac{1}{2}$	$\frac{1}{3}$

Update column m in  
the original matrix

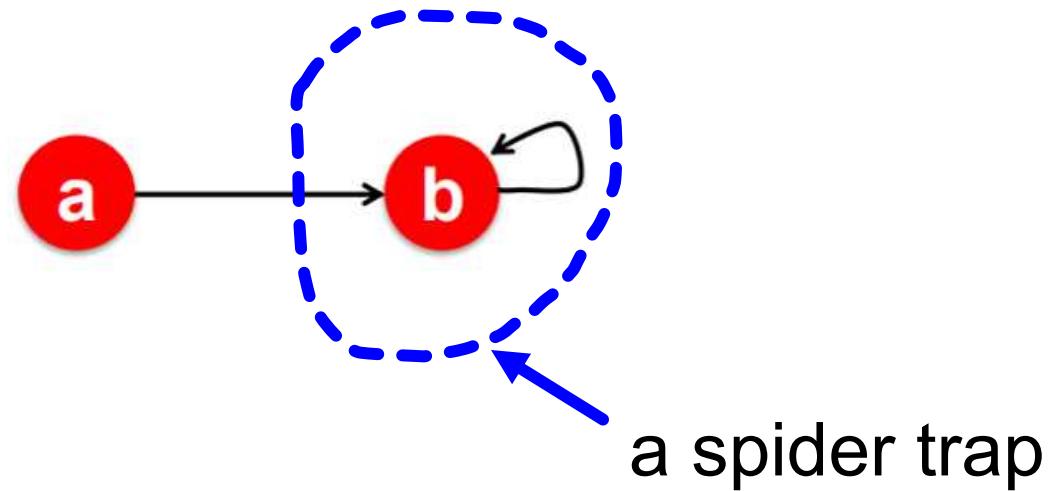


	y	a	m
y	0	0	$\frac{1}{3}$
a	0	0	$\frac{1}{3}$
m	0	0	$\frac{1}{3}$

Transition matrix Q  
(Teleport links)

Now the sum of  
column m is 1

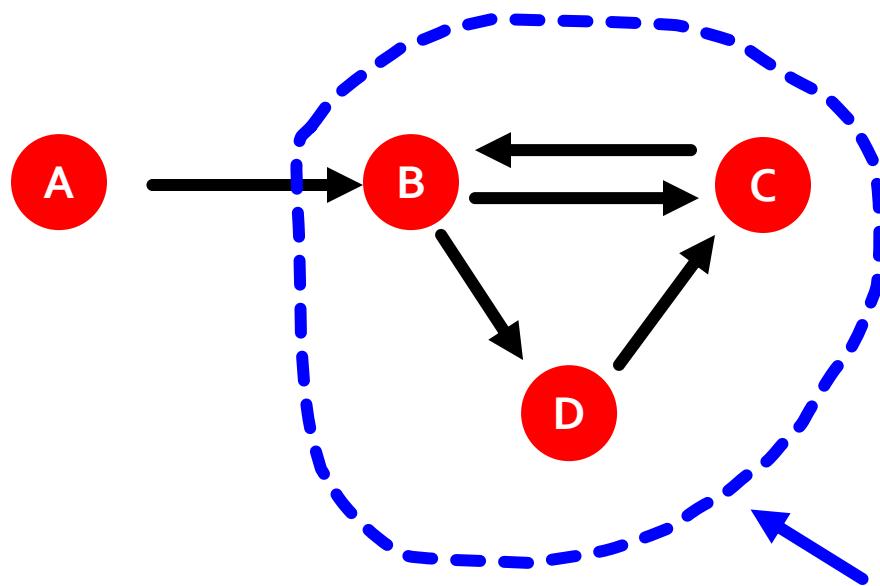
## 2. The spider trap problem:



A spider trap group: (a group of nodes)  
all out-links are within the group

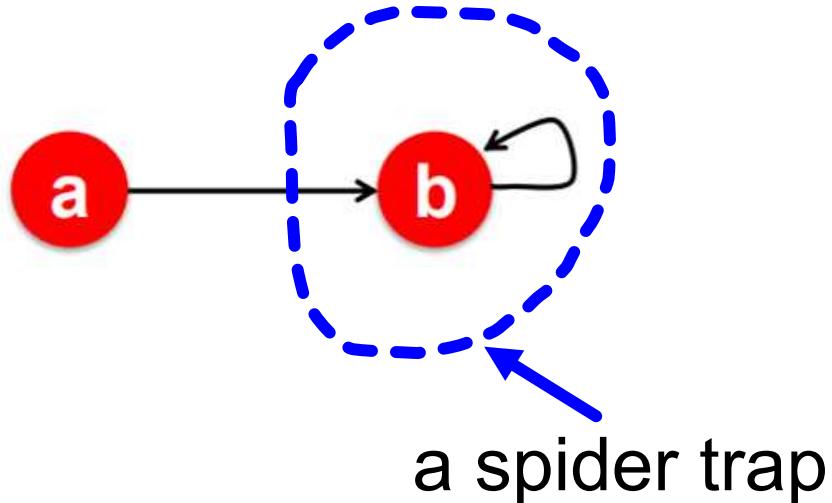
After entering the spider-trap group, the web user cannot go out of the group by navigating through the links

Another spider trap example:  
(all out-links are within the group)



the group forms a spider trap problem

After entering the spider-trap group, the web user cannot go out of the group by navigating through the links



	a	b
a	0	0
b	1	1

Transition Matrix: M

Spider trap issue: eventually, the spider traps absorb all the importance (high rank scores)

Use the power iteration method:

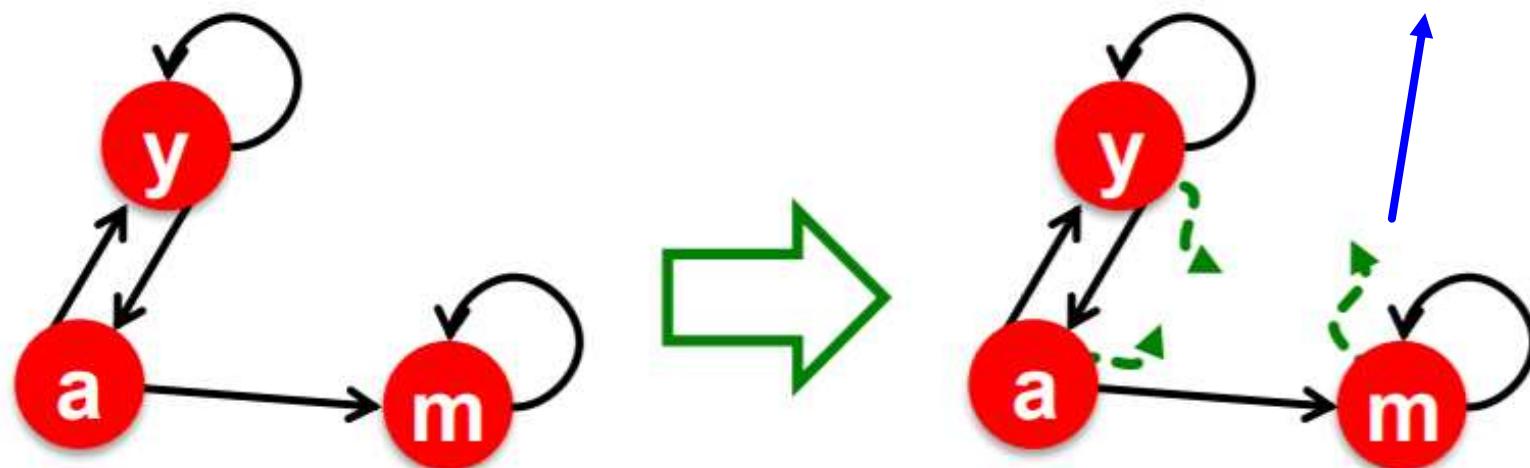
	init	Iter 1	Iter 2
Node a	0.5	0	0
Node b	0.5	1	1

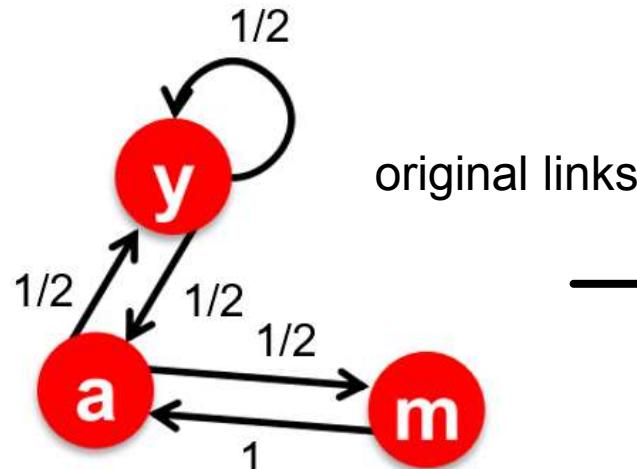
$$r^{(t+1)} = M \cdot r^{(t)}$$

Solution:

Add special teleport links for every node  
The teleport links will be separately handled

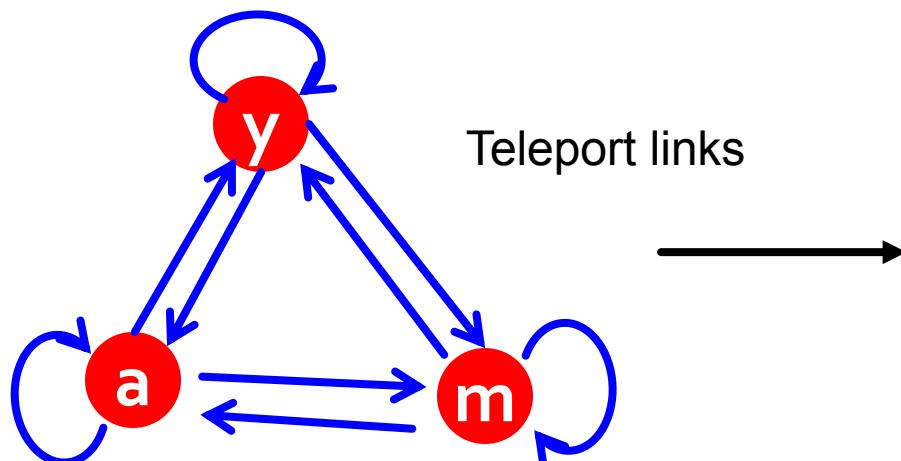
Teleport links to  
all other points





	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	0
$r_a$	$\frac{1}{2}$	0	1
$r_m$	0	$\frac{1}{2}$	0

Transition matrix M  
(original links)



	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$r_a$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$r_m$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Transition matrix Q  
(Teleport links)

# PageRank – Google Matrix

Using the Google matrix as the new transition matrix

The Google matrix is constructed by two transition matrixes:

$$G = \beta M + (1 - \beta) Q$$



1. the original matrix      2. teleport matrix

$\beta$  is a weighting parameter, in practice,  $\beta = 0.8$  or  $0.9$

$Q$  is an  $N$  by  $N$  matrix, where all elements are  $1/N$

$N$  is the number of nodes in the graph

Use the Google matrix in the power iteration method:

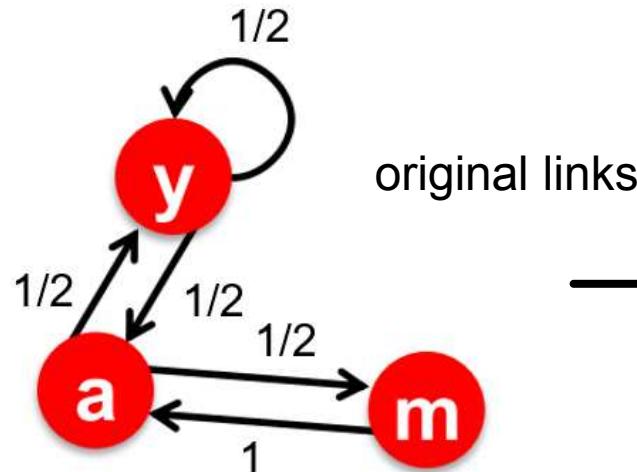
$$r^{t+1} = G \cdot r^t$$

# PageRank – Google Matrix

## PageRank – Google Matrix

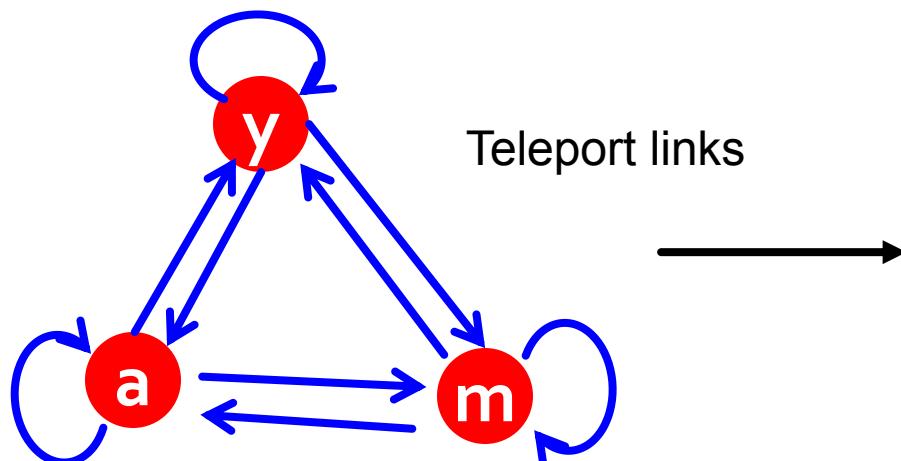
The flow equation:

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$



	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{2}$	$\frac{1}{2}$	0
$r_a$	$\frac{1}{2}$	0	1
$r_m$	0	$\frac{1}{2}$	0

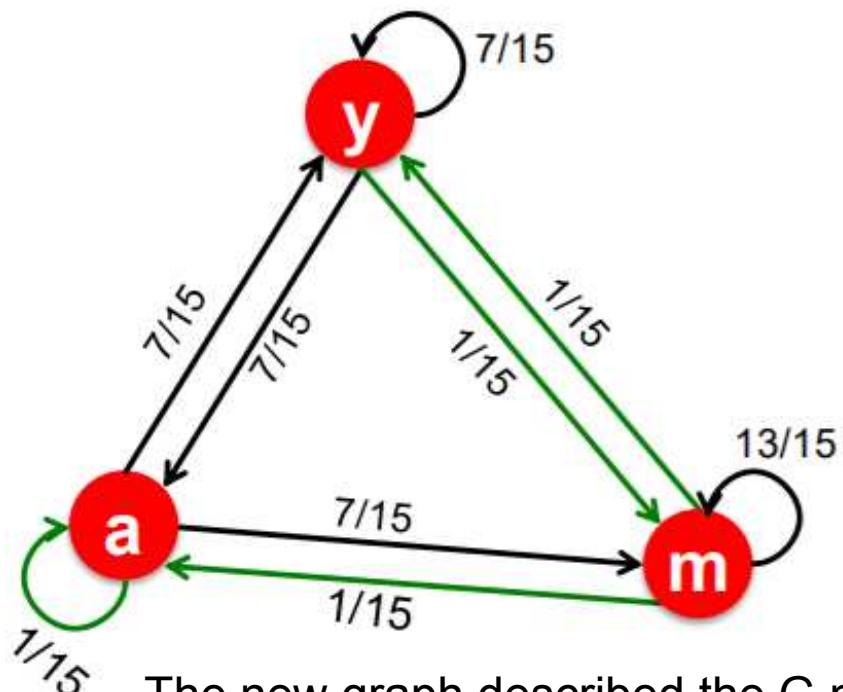
Transition matrix M  
(original links)



	$r_y$	$r_a$	$r_m$
$r_y$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$r_a$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$r_m$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Transition matrix Q  
(Teleport links)

$$\beta = 0.8$$



Original links:

**M**

$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}$$

Teleport links:

**Q**

$$+ 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$\begin{matrix} y & 7/15 & 7/15 & 1/15 \\ a & 7/15 & 1/15 & 1/15 \\ m & 1/15 & 7/15 & 13/15 \end{matrix}$$

**G**

The new graph described the G matrix

y	1/3	0.33	0.24	0.26	...	7/33
a	1/3	0.20	0.20	0.18	...	5/33
m	1/3	0.46	0.52	0.56		21/33

# Further discussions



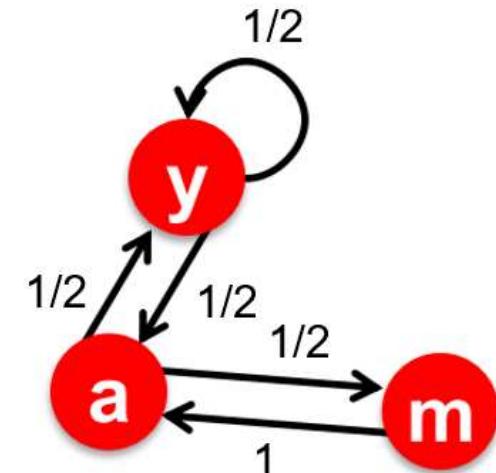
(The following topics are not examinable!)

- PageRank and random walk algorithm
- PageRank and eigenvector

# Random walk

One random walk step:

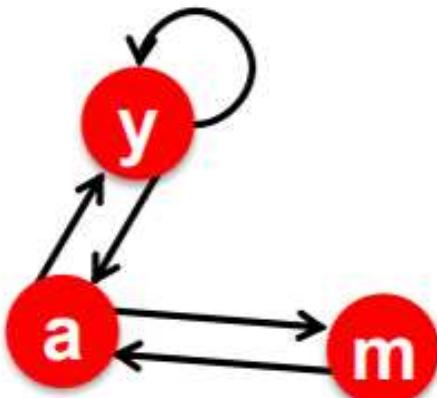
- A random walker on the graph:
  - At time  $t$ , the walker is at node  $i$
  - At time  $t + 1$ , the walker follows the probabilities in the transition matrix to jump to another node.



	$r_y$	$r_a$	$r_m$
$r_y$	1/2	1/2	0
$r_a$	1/2	0	1
$r_m$	0	1/2	0

Transition matrix M

# Random walk



	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$r_y = r_y/2 + r_a/2$$

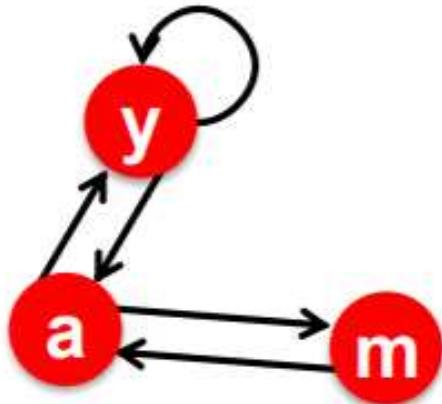
$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

A random walk problem:

Q: if a walker start at node **m**, after 3 random-walk steps,  
what is the location of the walker?  
(calculate the probability of landing for each node)

# Random walk



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	1
m	0	$\frac{1}{2}$	0

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

Solution:

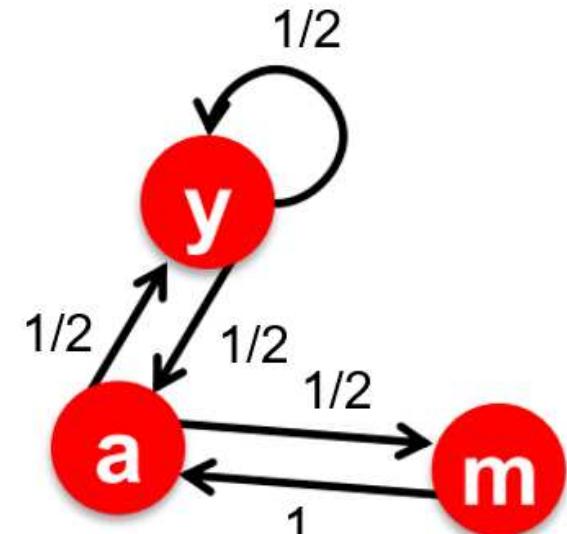
We can change the initialization and use the power iteration method.

Starting from node m:

1st step:  $\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

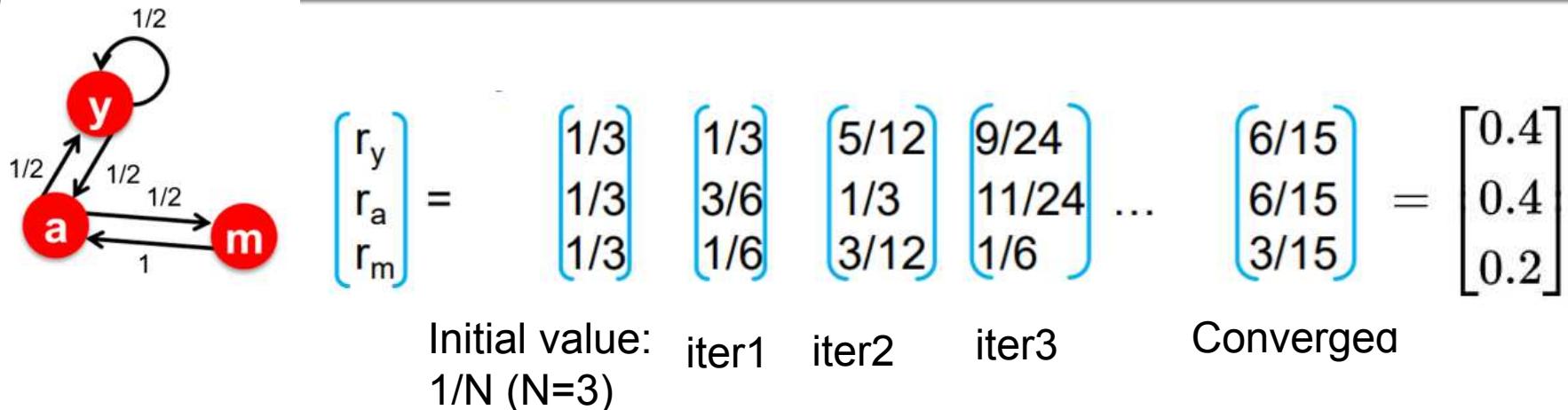
2nd step:  $\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix}$

3rd step:  $\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/4 \\ 3/4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.75 \\ 0 \end{bmatrix}$



The **r** vector gives the probability of each node that the walker will visit after 3 random walk steps

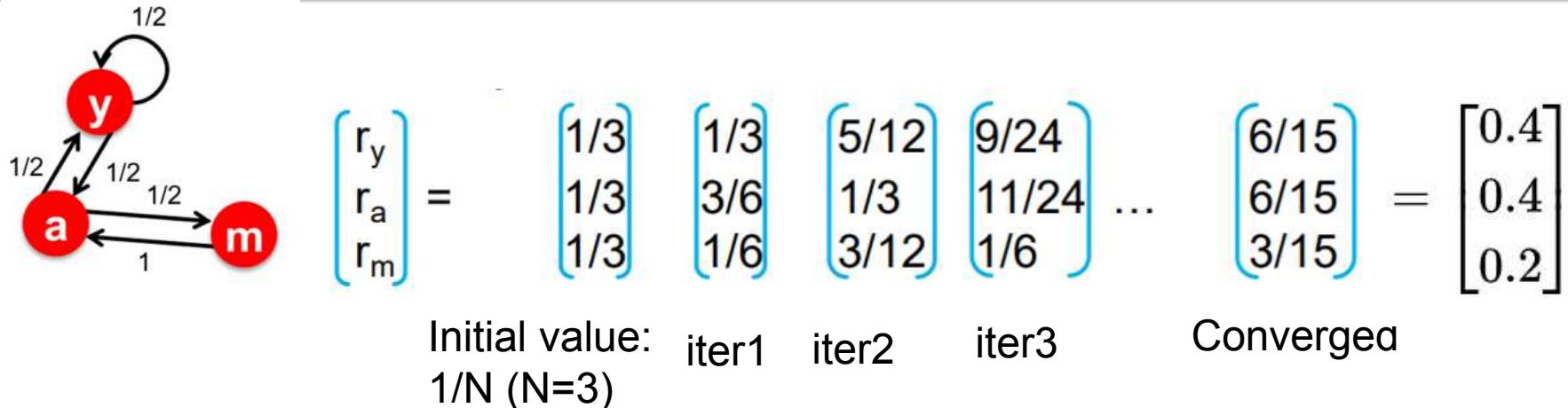
# Random walk and PageRank



**PageRank is a random walk algorithm on a graph:**

A walker randomly start at one node (equal probability for each node);  
in each time step, the walker follow the probabilities in the transition matrix to jump to the next node.

# Random walk and PageRank



**PageRank is a random walk algorithm on a graph:**

Q: What is the probability of the walker's location after  $m$  time steps?

Perform the power iteration method with random initialization ( $1/n$ ) for  $m$  steps and the PageRank score vector gives the probability for each node.

# PageRank and Eigenvector

- The flow equation:

$$1 \cdot \mathbf{r} = \mathbf{M} \cdot \mathbf{r}$$



$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

the rank vector  $\mathbf{r}$  is an eigenvector of the transition matrix  $\mathbf{M}$  (with eigenvalue 1)

With the requirement: all rank score sum equals to 1

The converged rank vector  $\mathbf{r}$  is called the stationary distribution

Definition of eigenvector:

$$\lambda \mathbf{c} = \mathbf{A}\mathbf{c} \longrightarrow \mathbf{c}: \text{eigenvector}; \lambda: \text{eigenvalue}$$

- Further readings:
  - Stanford course CS224w
    - <http://web.stanford.edu/class/cs224w/slides/04-pagerank.pdf>
  - Lecture from Cornell
    - <http://pi.math.cornell.edu/~mec/Winter2009/RalucaReMus/Lecture3/lecture3.html>

# Graph examples

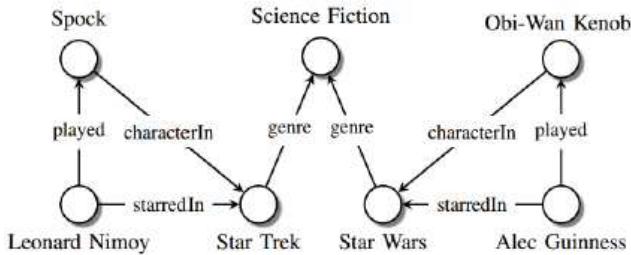


Image credit: [Maximilian Nickel et al](#)

## Knowledge Graphs

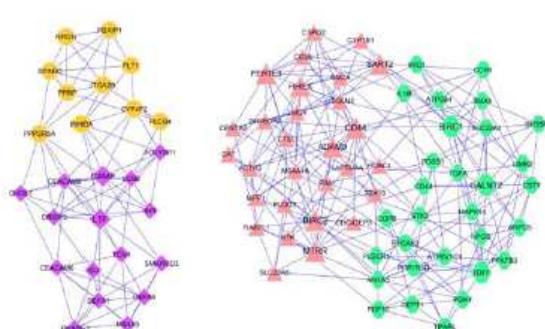


Image credit: [ese.wustl.edu](#)

## Regulatory Networks

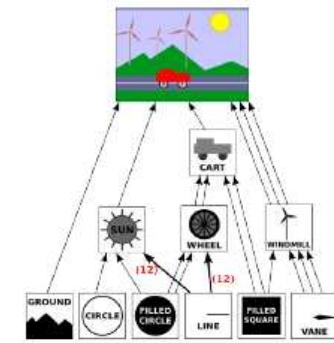


Image credit: [math.hws.edu](#)

## Scene Graphs

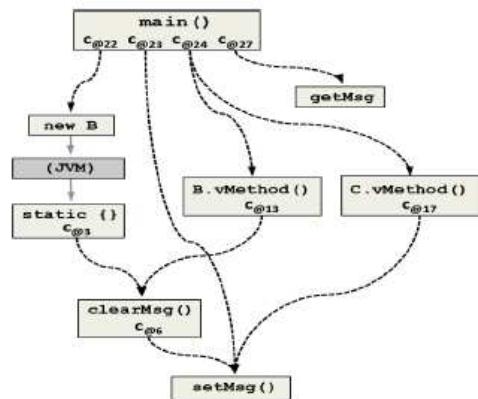


Image credit: [ResearchGate](#)

## Code Graphs

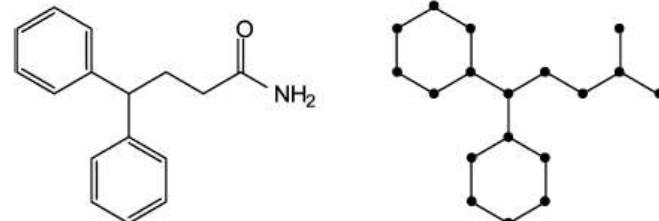


Image credit: [MDPI](#)

## Molecules

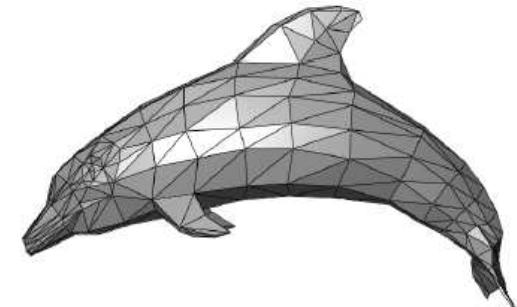


Image credit: [Wikipedia](#)

## 3D Shapes

Many datasets can be formulated as graphs

# Graph Neural Network

Lin Guosheng  
School of Computer Science and Engineering  
Nanyang Technological University

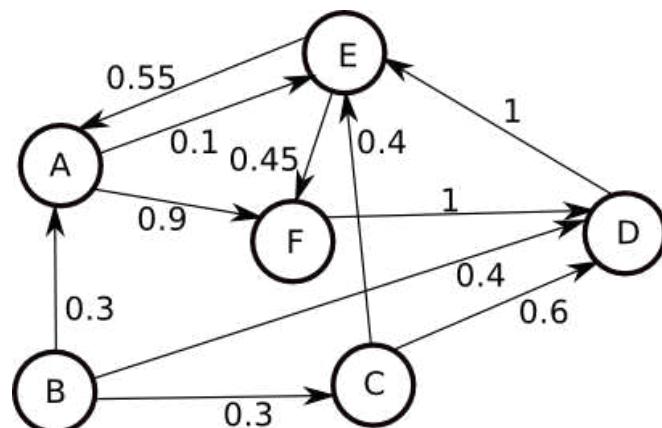
# Outline

- Introduction to graphs
- Graph convolutional network (GCN)
  - Network prediction (forward pass)
  - GCN applications/tasks
- Network training
  - Loss functions
  - Backward pass

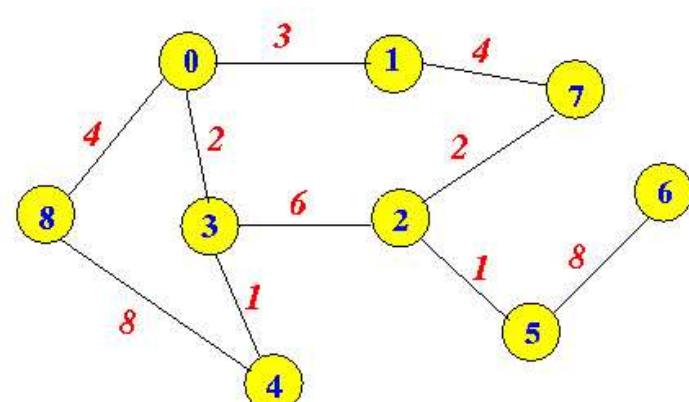
# Introduction to graphs

## ■ A graph

- Nodes and links (edges). May have weights on links.
  - Weights indicate the strength of links
- Links can be directed or undirected.
  - One undirected link can be treated as two directed links (forms bidirectional connections)



Directed graph

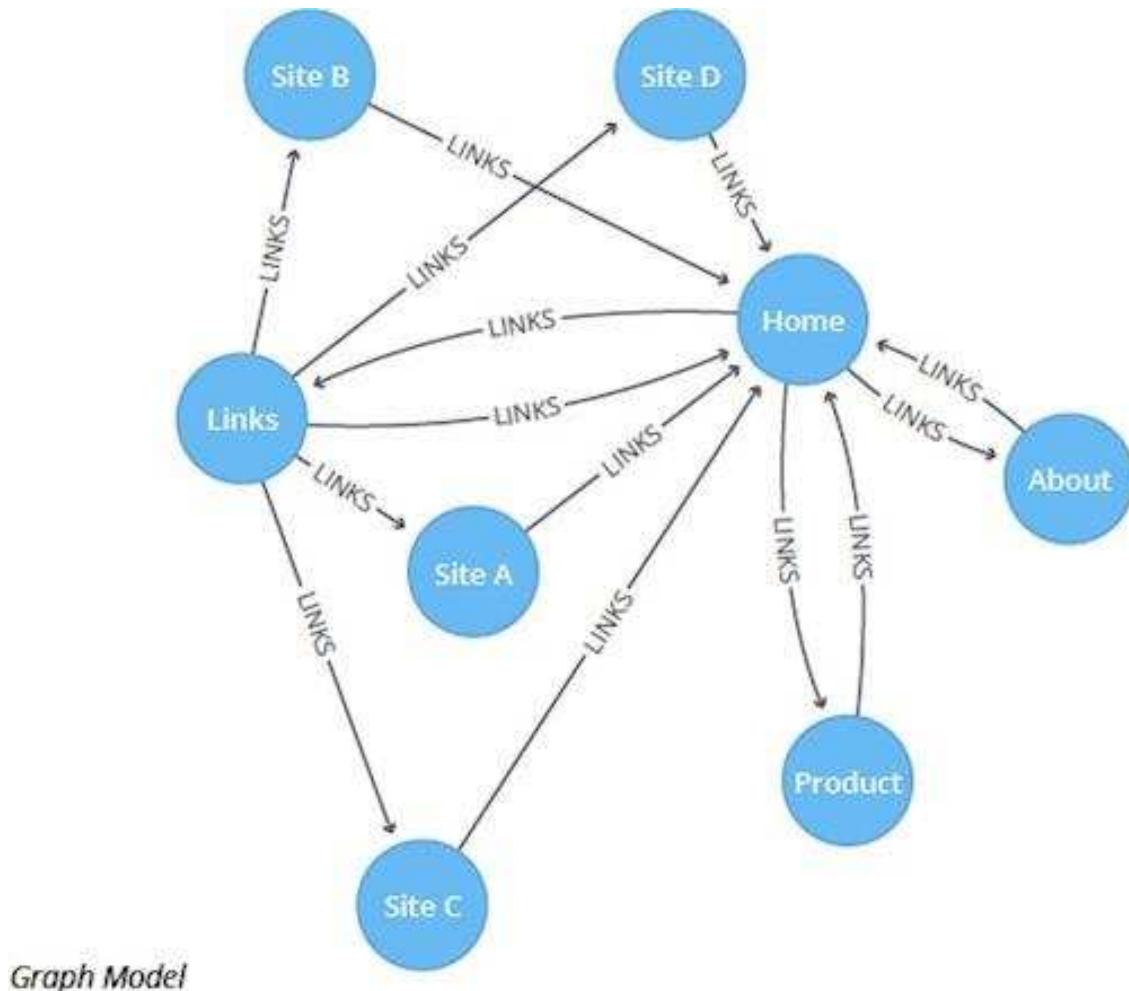


Un-directed graph

<http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/11-Graph/weighted.html>

<https://iwillgetthatjobatgoogle.tumblr.com/post/12156105945/weighted-graph-implementation>

# Graph examples



Web pages as graph (PageRank)

Web pages:  
Each web page is a node  
A hyperlink is a link

# Graph examples

Many datasets can be formulated as graphs



Image credit: [Medium](#)

Social Networks

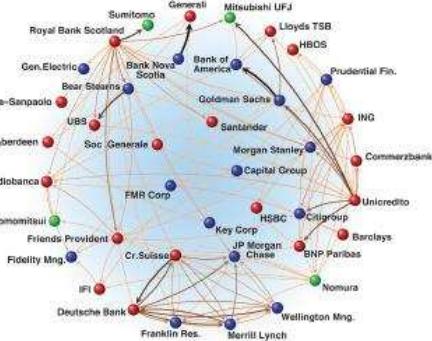


Image credit: [Science](#)

Economic Networks

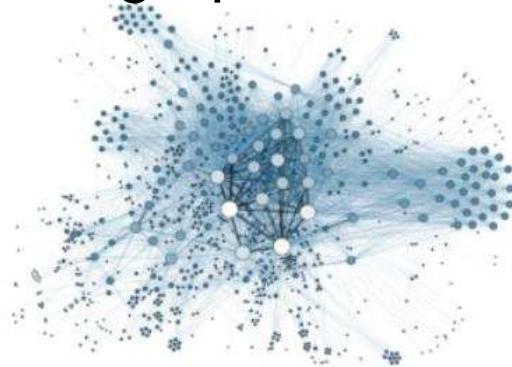


Image credit: [Lumen Learning](#)

Communication Networks

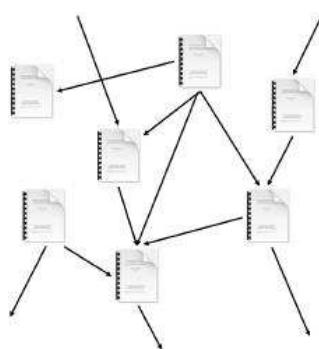


Image credit: [Missoula Current News](#)

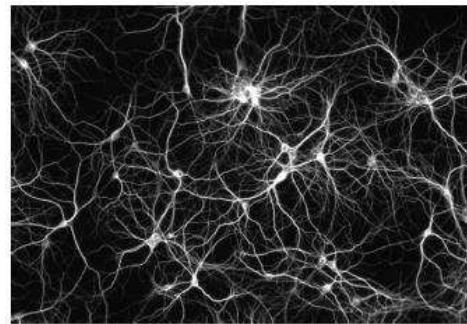


Image credit: [The Conversation](#)

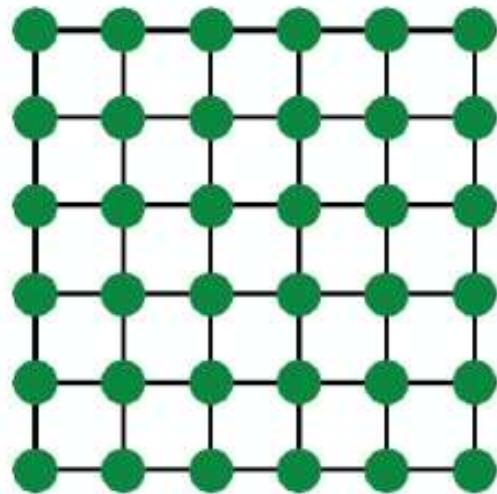
Citation Networks

Internet

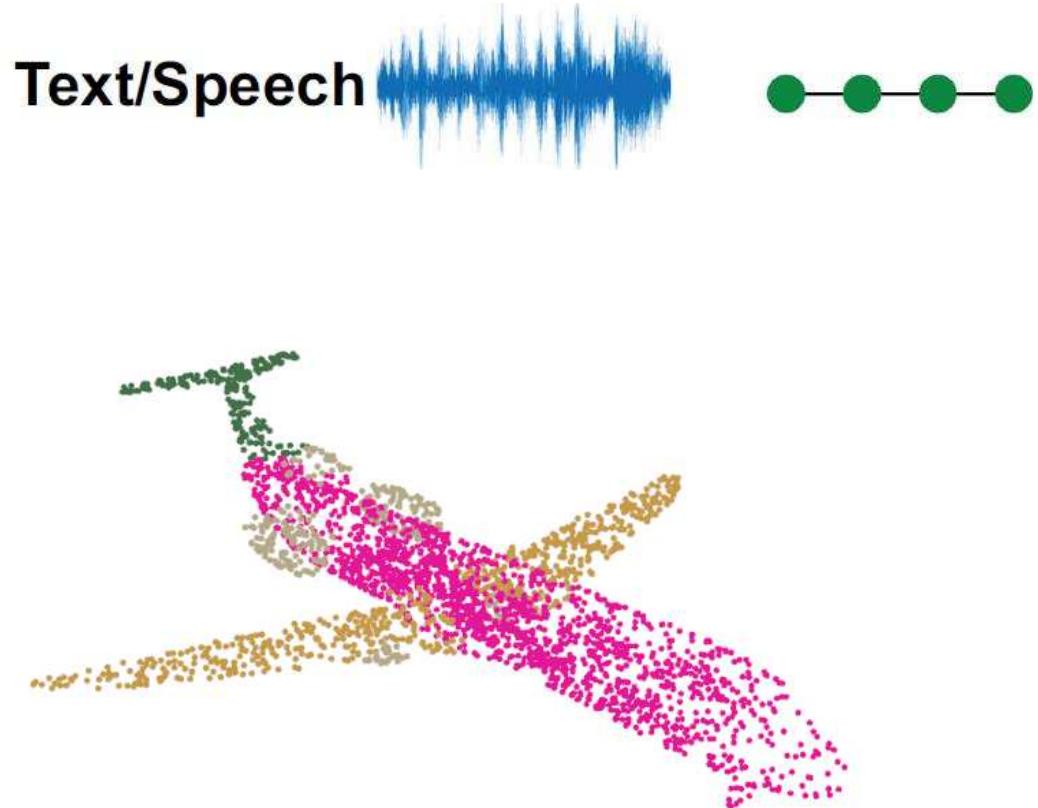
Networks of Neurons

Many datasets can be formulated as graphs

# Graph examples



**Images**



3D point clouds

Many datasets can be formulated as graphs

- More examples for formulating graphs
  - Social networks (Facebook, YouTube):
    - Each user is a node
    - A friendship relation is a link, or a contact record forms a link
  - Amazon product data:
    - Each product is a node
    - A co-purchase relationship is a link
  - DBLP publication data
    - Each author is a node
    - A co-author relationship is a link
  - More examples:
    - <https://snap.stanford.edu/data/index.html>

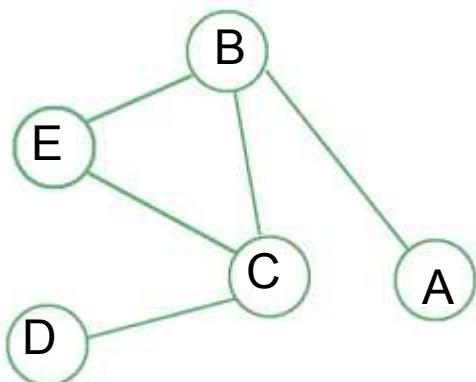
# Graph convolutional network

- Graph neural network (GNN)
  - Neural networks for graph data
- Graph convolutional network (GCN)
  - GCN is a type of GNN
  - Main idea:  
Aggregate the information from neighbourhoods  
to generate node features.

# Node classification task

- GCN output for node classification
  - Predict a class label for each node
    - In practice, we predict **a class score vector** for each node. Each score indicates the confidence for one class.
    - In this class, we focus on the node classification task using Graph convolutional network (GCN).

Example: Assume we have 4 classes in total.  
We aim to predict a 4-dimensional score vector for each node. One dimension corresponds to one class



For node A, suppose the GCN output is the following score vector: [0.1, 0.1, 0.6, 0.2]

As the 3<sup>rd</sup> element has the largest value, we predict the class label as 3 for node A.

# GCN

## ■ Graph convolutional network (GCN)

- Input: node features and graphs (affinity matrix)
- Forward pass for prediction
- Backward pass for training

# GCN input

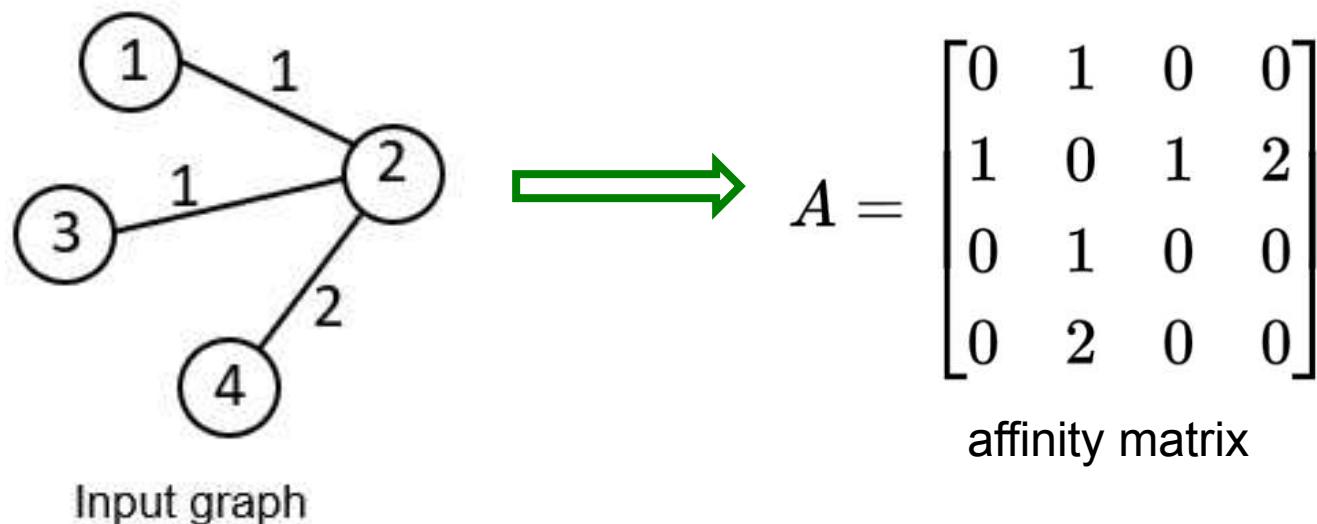
- Input of GCN:
  - 1. The input (initial) feature vector for each node.
    - We assume the feature vector for each node is available.
      - The feature vector encode the information for each individual node
    - The generation of the input feature vectors for each node depends on the applications.
      - E.g., in the 3D point segmentation example, the feature vector for each node is the (x, y, z) coordinates of the vertice
  - 2. The graph affinity matrix A.
    - describing the connections in a graph

# GCN input

- The graph affinity matrix A.
  - In practice, we use row normalized affinity matrix A'.

$a_{i,j}$  : The element  $(i,j)$  in the affinity matrix A.

It is the edge weight of the edge between node i and node j. If there is no link between  $(i, j)$ ,  $a_{i,j} = 0$ ;



- Normalized affinity matrix  $A'$ .

Row normalized affinity matrix  $A'$ : the sum of each row in  $A'$  is 1.

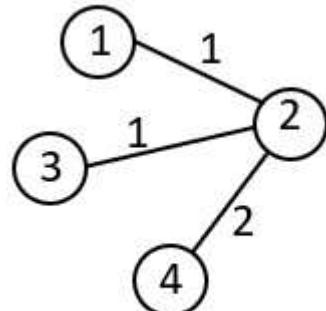
The  $(i,j)$  element in  $A'$  is :

$$a'_{ij} = \frac{a_{ij}}{\sum_{k=1}^N a_{ik}}$$

$\sum_{k=1}^N a_{ik}$  : is the sum of the i-th row in the original Affinity matrix A for node i

After normalization, the matrix is not symmetric

### 1. example on weighted graph:



Input graph

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

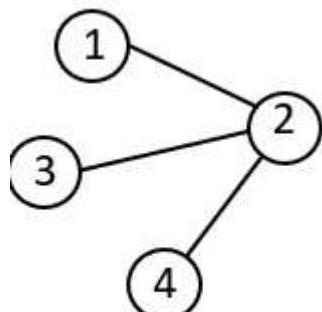
affinity matrix

$$A' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.25 & 0 & 0.25 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

row-normalized affinity matrix

### 2. example on unweighted graph:

(treat the edge weight as 1 for all edges)



Input graph

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

affinity matrix

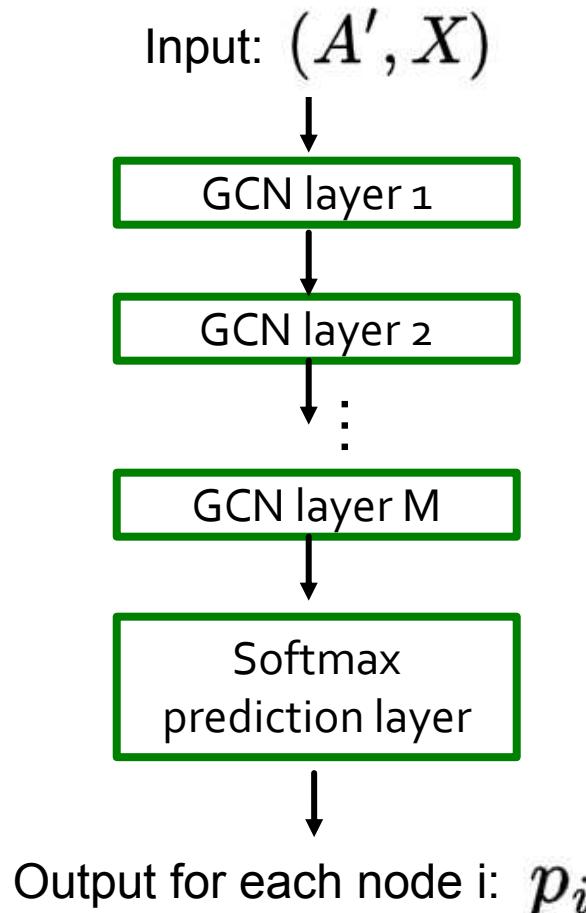
$$A' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

row-normalized affinity matrix

# GCN forward pass

- GCN is a stack of multiple GCN layers

An example for node classification tasks.

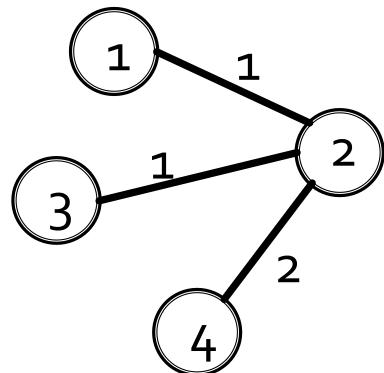


- $A'$  : Row-normalized affinity matrix  
(the sum of each row is 1)
- $X$  : A matrix of all input node features.  
Each column indicate one node feature vector
- $p_i$  : The class score vector for node  $i$   
Each element indicates the prediction score for one class.  
Assume there are  $C$  classes in total,  
the length of this score vector is  $C$

GCN can be seen as a complicated mapping function:

$$P = f_{GCN}(A', X)$$

## Example for the input $(A', X)$



Input graph

The initial node features are given below as:

$$x_1 = [1, -1]^T,$$

$$x_2 = [0, 1]^T,$$

$$x_3 = [-1, 0]^T,$$

$$x_4 = [1, 0]^T.$$

Calculate affinity matrix and row-normalized affinity matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \quad A' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.25 & 0 & 0.25 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$X$  : A matrix of all input node features.

One column indicates the input feature vector for one node, denoted by  $x_i$  or  $h_i^{(0)}$

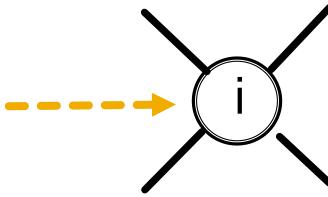
$$h_i^{(0)} = x_i$$

The matrix  $X$  (Matrix  $X$ ) is shown as a 4x4 matrix where each column represents the initial node features for a specific node. The columns are labeled  $h_1^{(0)}$ ,  $h_2^{(0)}$ ,  $h_3^{(0)}$ , and  $h_4^{(0)}$ . The values in the matrix are:

$$X = \begin{bmatrix} 1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 \end{bmatrix}$$

# GCN is applied to every node

The node  $i$  represents one node in the graph



The operation for one node:

Each GCN layer will update the feature vector for each node (non-linear transformation)

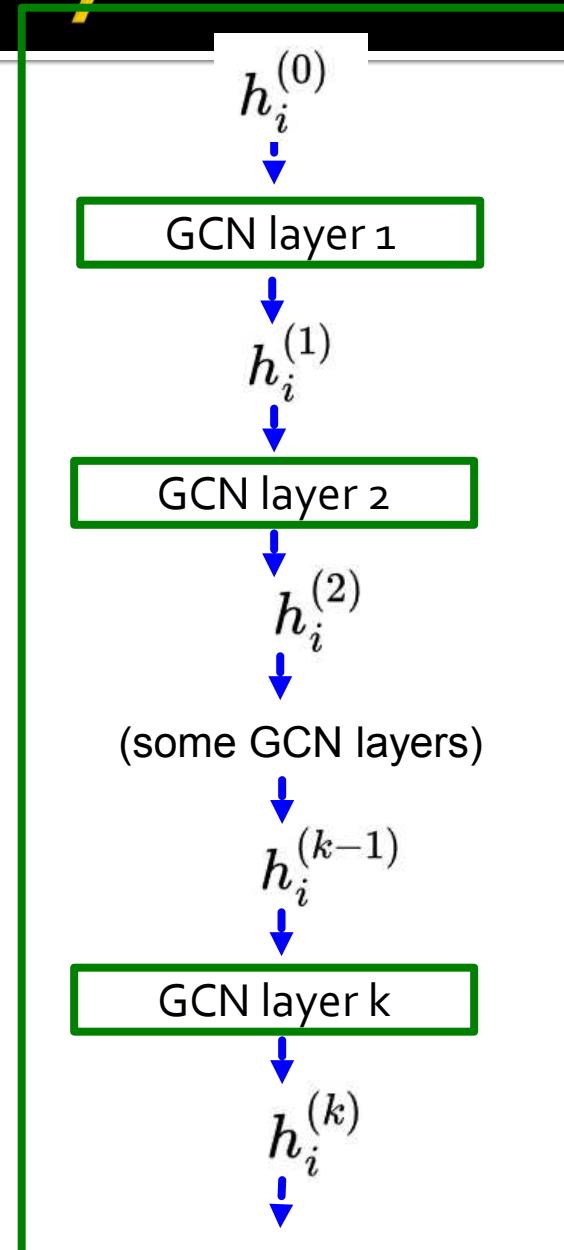
$h_i^{(0)}$  : The input feature vector for node  $i$  (or called the initial feature vector)

We have:  $h_i^{(0)} = x_i$

$x_i$  is the input feature vector for node  $i$

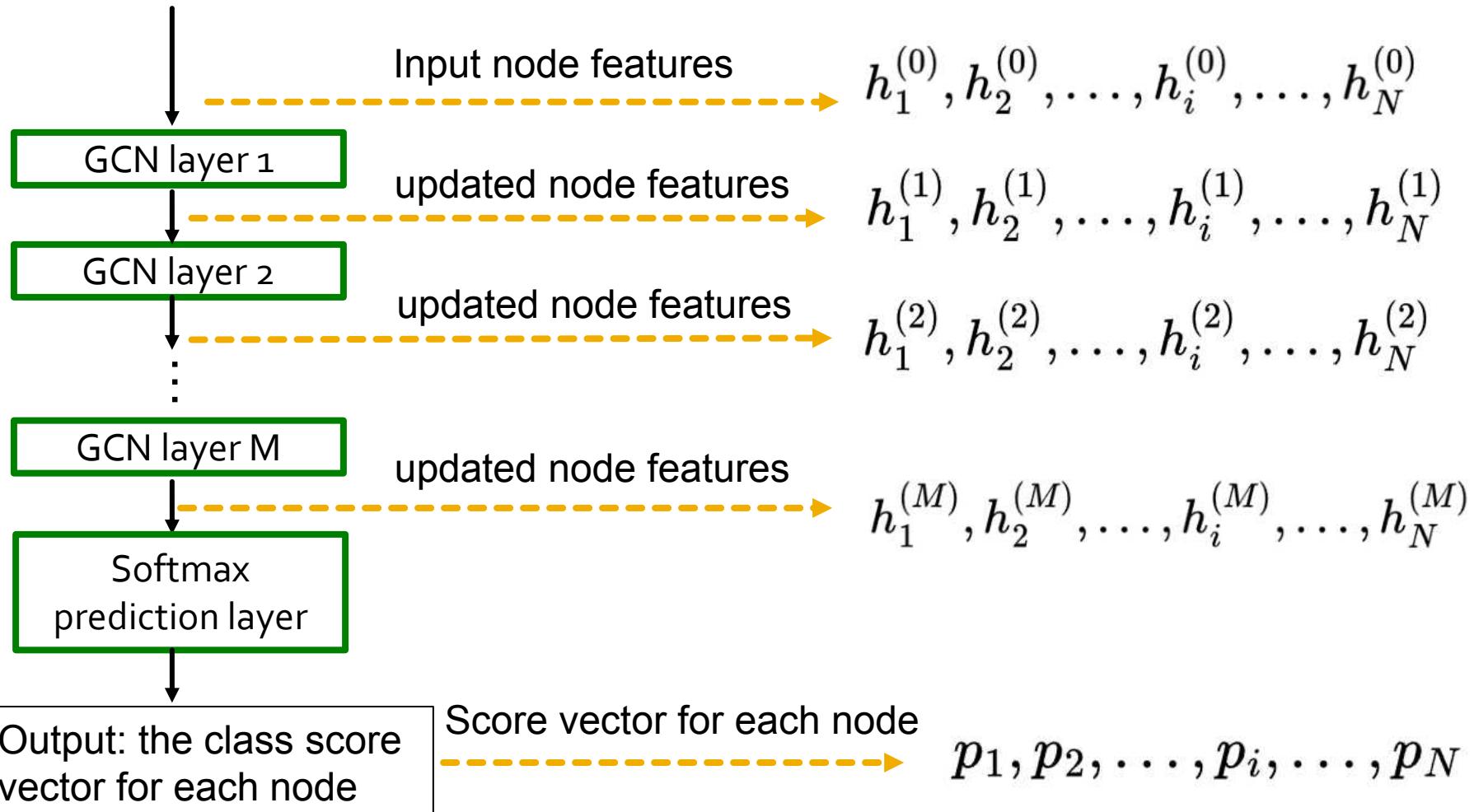
$h_i^{(k-1)}$  : the feature vector output by layer  $k-1$

$h_i^{(k)}$  : the feature vector output by layer  $k$



Input: the affinity matrix and  
the initial node feature vectors

The GCN operations for all nodes:



# One GCN layer

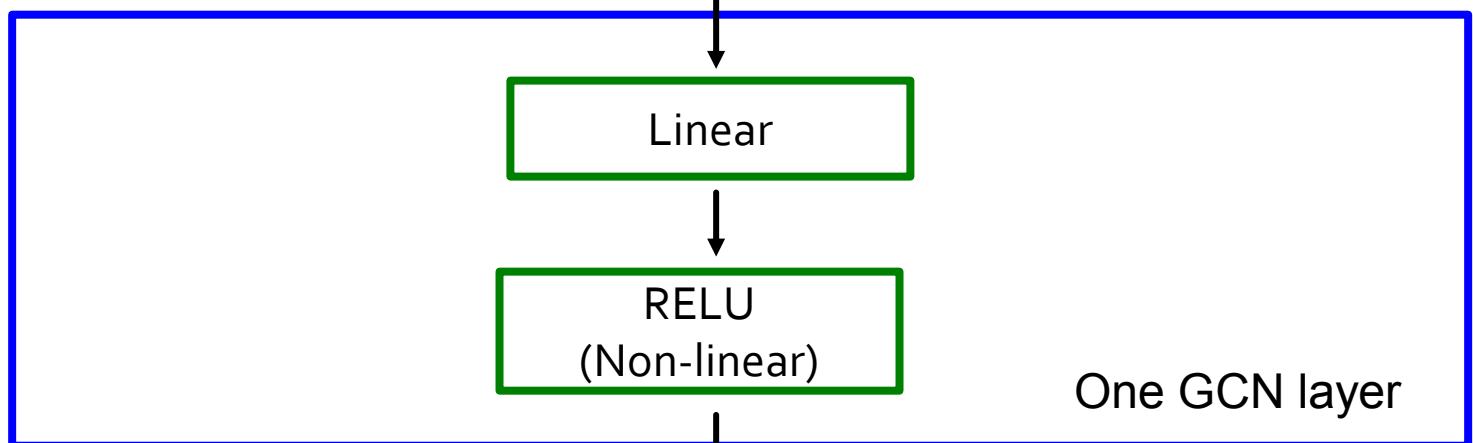
The calculation for one GCN layer:

Each GCN layer consist of two blocks (operations):

1. linear block; 2: Non-linear block

$h_i^{(k-1)}$  : the feature vector output by layer k-1 for node  $i$

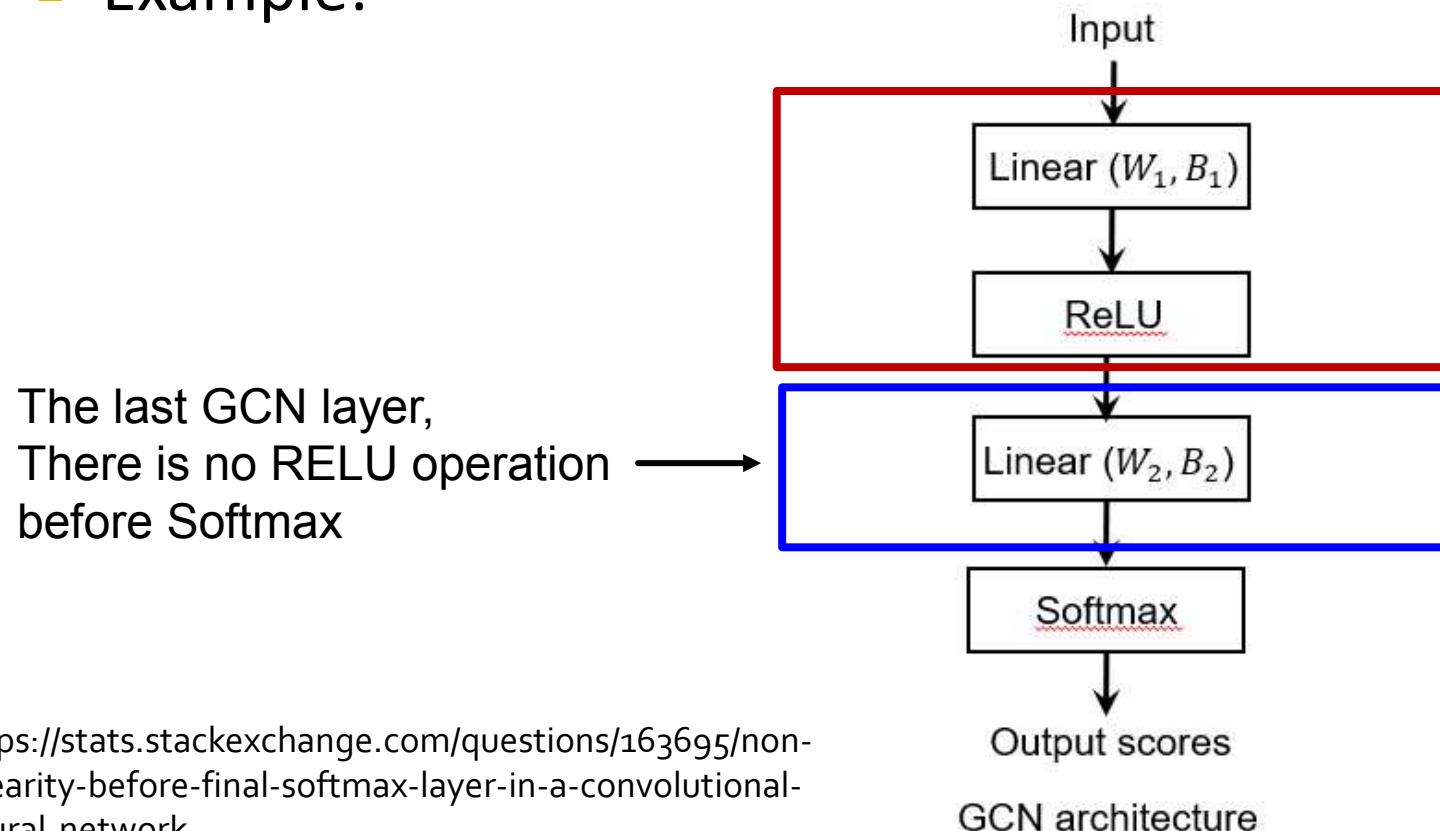
The k-th GCN layer:



$h_i^{(k)}$  : the feature vector output by layer k for node  $i$

The output node feature vector is also called:  
**node representation or node embedding.**

- GCN layer exception:  
In the last GCN layer, usually there is no non-linearity block.
- Example:



# One GCN layer

Linear transform:

$$\hat{h}_i^{(k)} = W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)} + B_k h_i^{(k-1)}$$

RELU (Non-linear transform):

$$h_i^{(k)} = \max\{0, \hat{h}_i^{(k)}\}$$

$a'_{ij}$ : the element (i,j) in the row-normalized affinity matrix A';

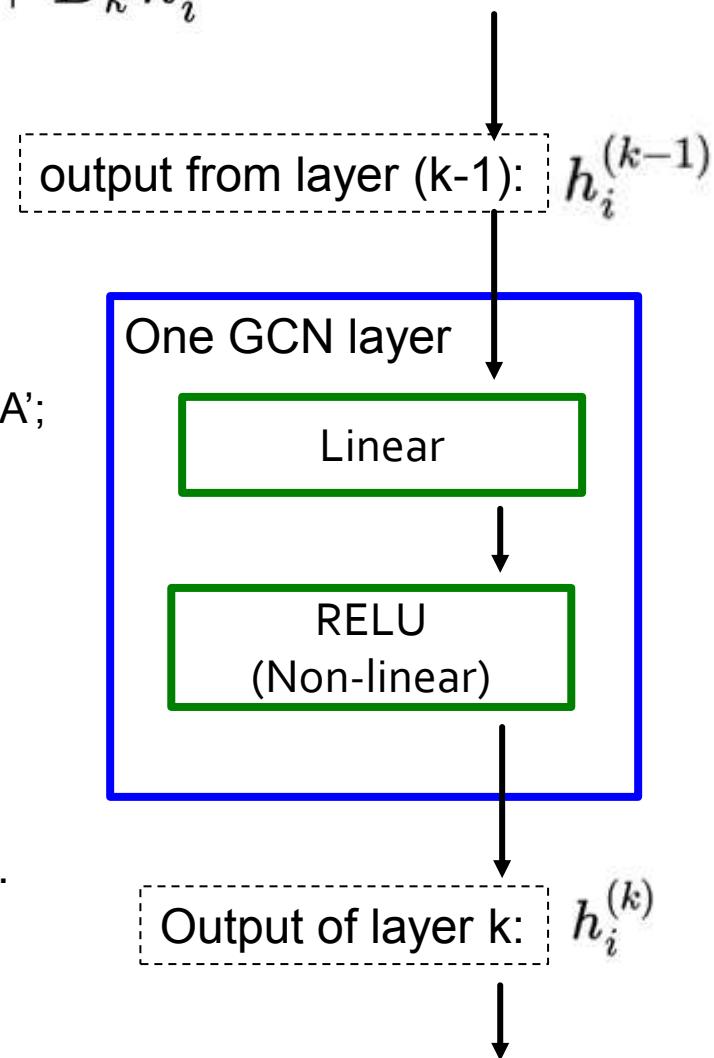
$\mathcal{N}_i$ : the set of neighbours of node i (connected nodes)

$h_i^{(k-1)}$ : input feature vector of node i for layer k

$h_i^{(k)}$ : output feature vector of node i for layer k

$W_k$ : Weight matrix for **neighbourhood aggregation**.

$B_k$ : Weight matrix for **self transformation**



Linear transform:

$$\hat{h}_i^{(k)} = \boxed{W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)}} + \boxed{B_k h_i^{(k-1)}}$$

neighbourhood aggregation

↓  
self transformation

$a'_{ij}$  : indicates the edge weight in the graph.  
it is the element (i,j) in the row-normalized affinity matrix A';

$W_k, B_k$  : are the GCN layer parameters. They are two matrixes.  
They are learnable parameters (or called layer weights).

In the exam or tutorial questions,  
the values of  $W_k, B_k$  will be given in the questions.

Network training: the training of GCN is to solve an optimization problem to obtain the values of these layer parameters

Linear transform:

$$\hat{h}_i^{(k)} = \boxed{W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)}} + \boxed{B_k h_i^{(k-1)}}$$

↓

neighbourhood aggregation      self transformation

## 1. Neighbourhood aggregation :

Aggregate the messages from the neighbours.

Neighbours here means directly connected nodes.

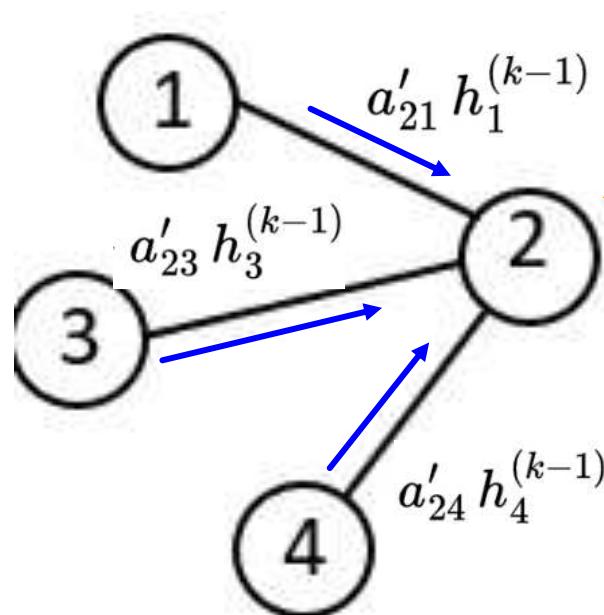
Linear transform:

$$\hat{h}_i^{(k)} = \boxed{W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)}} + \boxed{B_k h_i^{(k-1)}}$$

neighbourhood aggregation

self transformation

Aggregation calculation example:



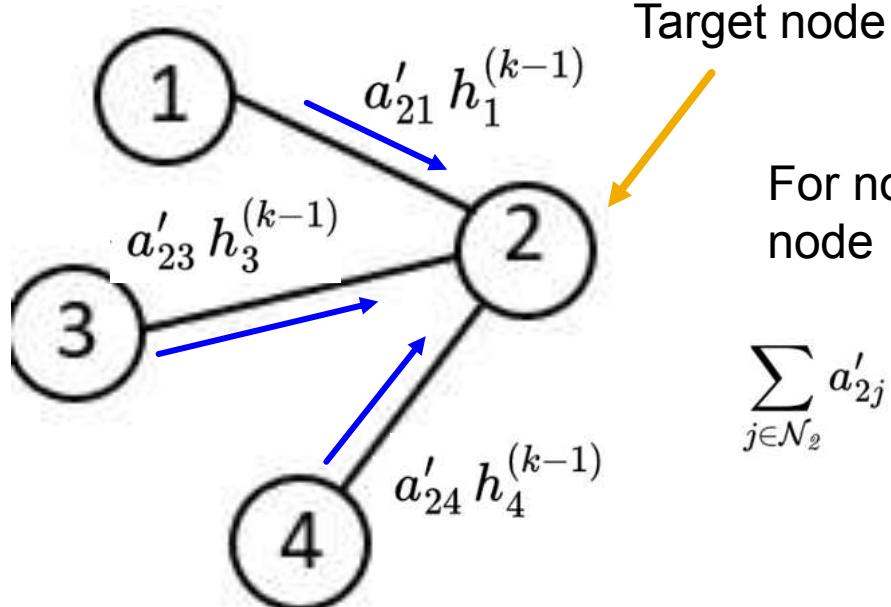
Input graph

For node 2, the messages from the neighbors: node 1, node 3 and node 4 are aggregated:

$$\sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(k-1)} = \boxed{a'_{21} h_1^{(k-1)}} + \boxed{a'_{23} h_3^{(k-1)}} + \boxed{a'_{24} h_4^{(k-1)}}$$

Example: the message from node 1 to node 2

# Aggregation example



Input graph

$a'_{ij}$  is the weight on the edge,  
which can be obtained from the  
row-normalized affinity matrix

For node 2, the messages from the neighbors:  
node 1, node 3 and node 4 are aggregated:

$$\sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(k-1)} = a'_{21} h_1^{(k-1)} + a'_{23} h_3^{(k-1)} + a'_{24} h_4^{(k-1)}$$

$$A' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

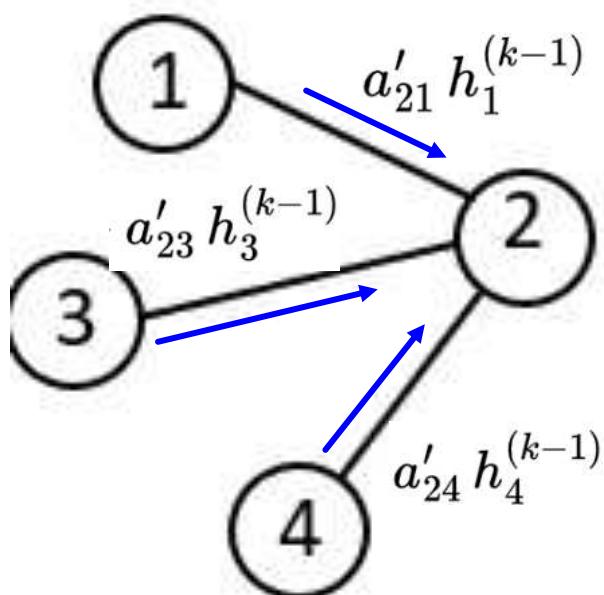
row-normalized affinity matrix

# Aggregation example

Linear transform:

$$\hat{h}_i^{(k)} = \boxed{W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)}} + \boxed{B_k h_i^{(k-1)}}$$

neighbourhood aggregation      self transformation



Input graph

For node 2, the messages from the neighbors:  
node 1, node 3 and node 4 are aggregated

$$\sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(k-1)} = a'_{21} h_1^{(k-1)} + a'_{23} h_3^{(k-1)} + a'_{24} h_4^{(k-1)}$$

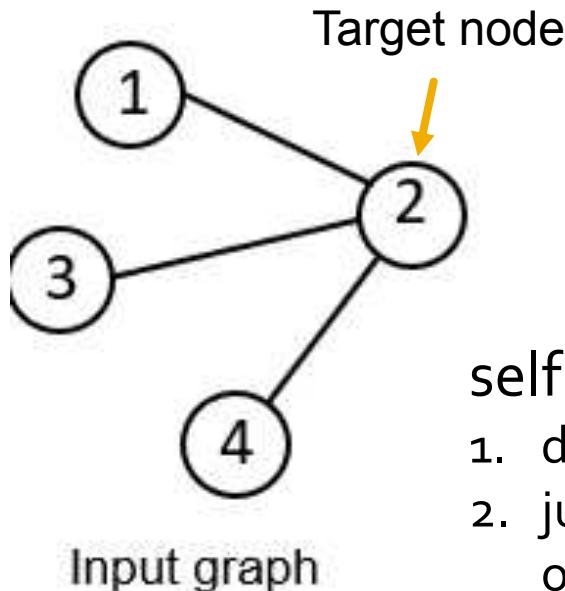
After aggregation, we perform transformation using  $W_k$ :

$$W_k \sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(k-1)}$$

# Self transformation example

Linear transform:

$$\hat{h}_i^{(k)} = W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)} + B_k h_i^{(k-1)}$$



neighbourhood aggregation

$$B_k h_i^{(k-1)}$$

self transformation

GCN layer k-1

$$h_i^{(k-1)}$$

GCN layer k

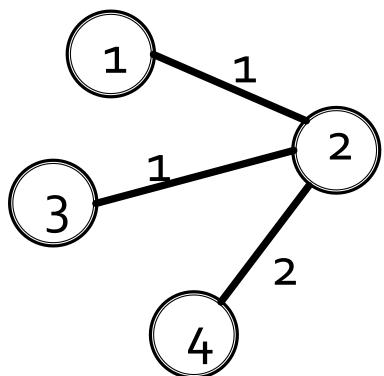
$$h_i^{(k)}$$

self transformation for node 2:

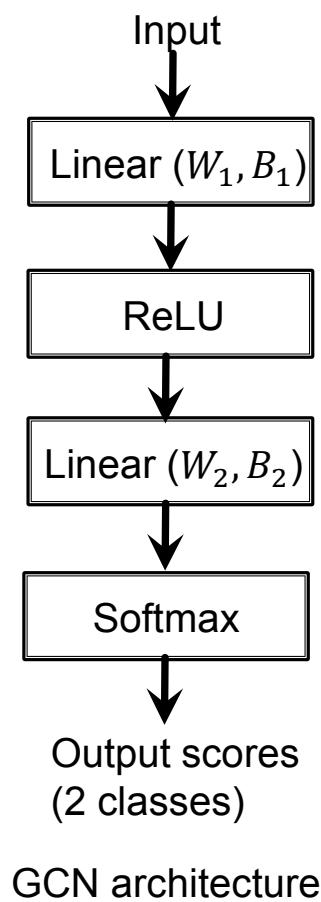
1. do not need to look at the neighbours
2. just use the feature vector of node 2 output by the previous layer (k-1)
3. Use the  $B_k$  matrix for transformation:

$$B_k h_2^{(k-1)}$$

- Examples of W and B matrix:



$$x_1 = [1, -1]^T, \\ x_2 = [0, 1]^T, \\ x_3 = [-1, 0]^T, \\ x_4 = [1, 0]^T.$$



$$W_1 = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \\ 0.1 & -0.1 \end{bmatrix}$$

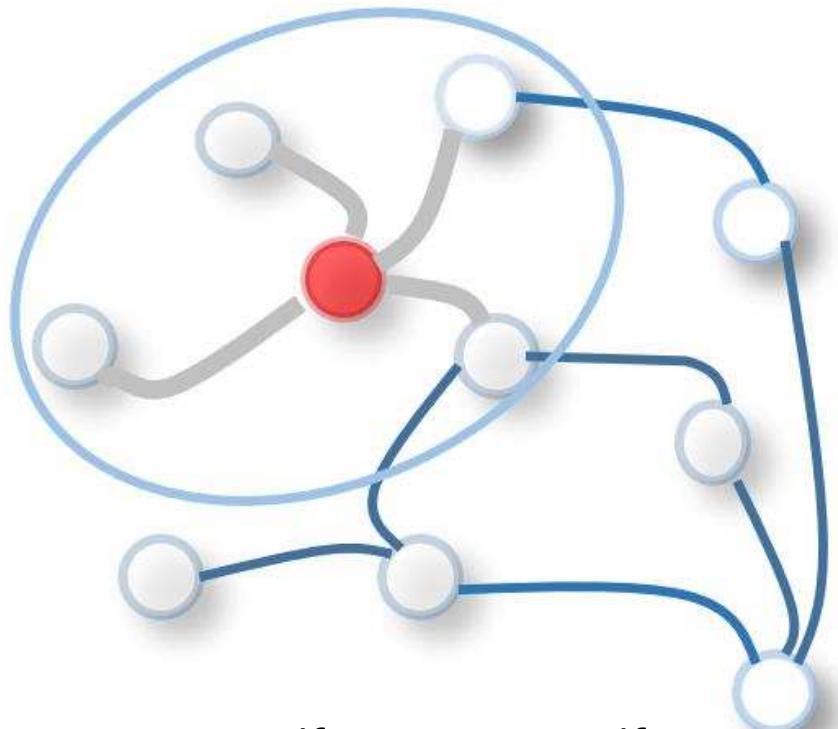
$$B_1 = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.1 & 0.0 & -0.1 \\ 0.1 & 0.2 & -0.1 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 0.0 & 0.1 & 0.1 \\ 0.1 & -0.1 & -0.1 \end{bmatrix}$$

Different layers have different W, B matrix

- Key idea of GCN:
  - Aggregate the information (messages) from the neighbours to generate the features for one node.



<https://arxiv.org/pdf/1901.00596.pdf>

# One GCN layer

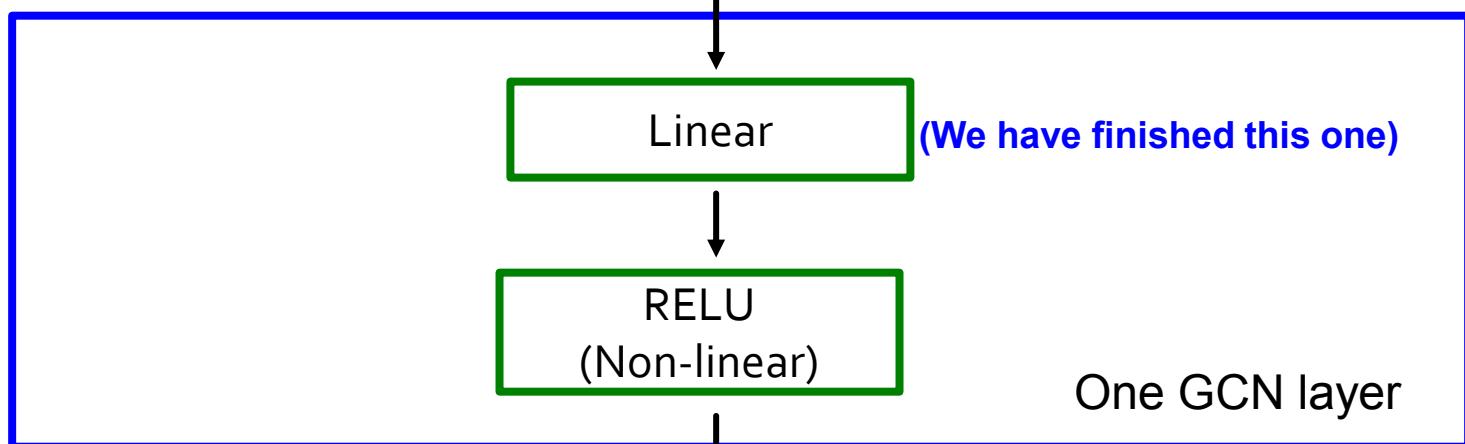
The calculation for one GCN layer:

Each GCN layer consist of two blocks (operations):

1. linear block; 2: Non-linear block

$h_i^{(k-1)}$ : the feature vector output by layer k-1 for node  $i$

The k-th GCN layer:



$h_i^{(k)}$  : the feature vector output by layer k for node  $i$

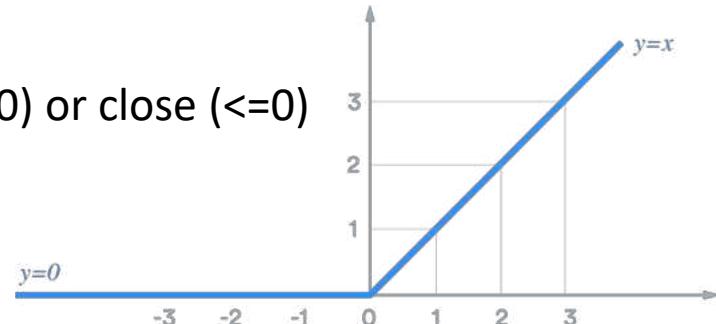
The output node feature vector is also called:  
**node representation** or **node embedding**.

## ■ Non-linearity functions

- Also called activation functions
- Like a “gate” or a tiny classifier: open ( $>0$ ) or close ( $\leq 0$ )

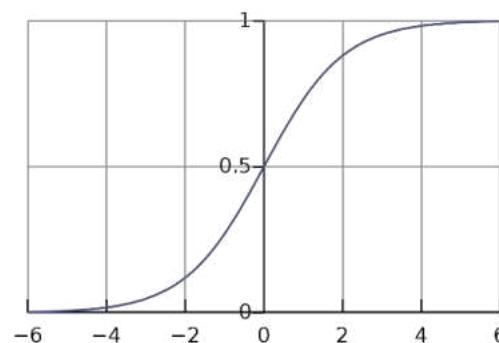
### 1. Rectified linear unit (ReLU)

$$ReLU(x) = \max(x, 0)$$



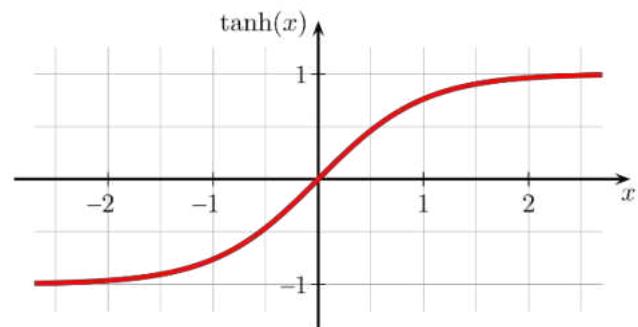
### 2. Sigmoid

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



### 3. Hyperbolic tangent

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

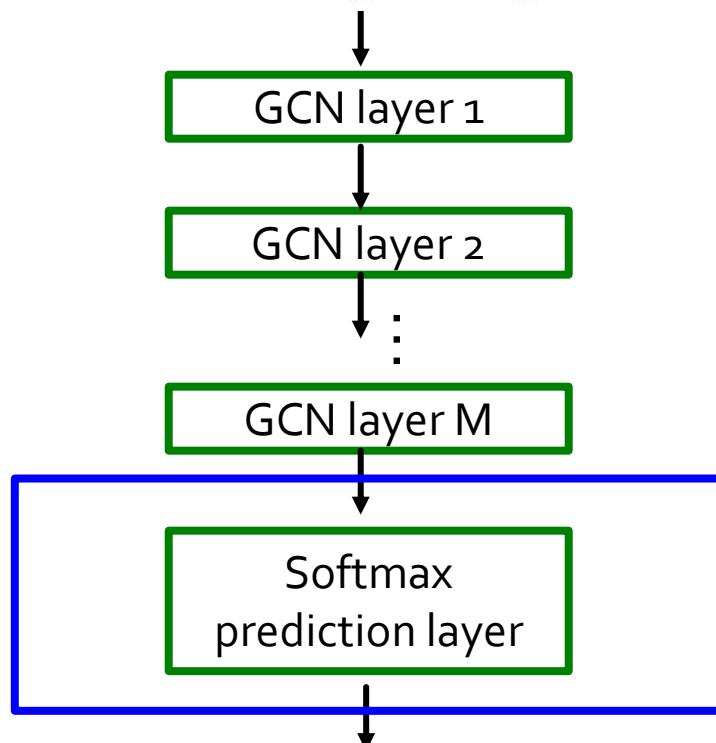


# GCN forward pass

- GCN is a stack of multiple GCN layers

An example for node classification tasks.

Input:  $(A', X)$



$p_i$  : The class score vector for node  $i$   
Each element indicates the prediction score for one class.  
Assume there are  $C$  classes in total,  
the length of this score vector is  $C$

Example for 3-class classification:  
 $p_1 = [0.1 \quad 0.1 \quad 0.8],$   
 $p_2 = [0.2 \quad 0.3 \quad 0.5],$   
....

(We now come to this one)

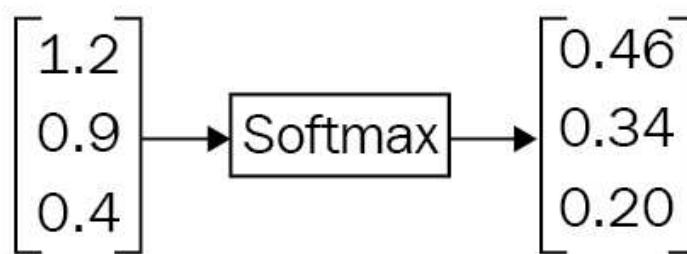
Output for each node  $i$ :  $p_i$

- Softmax layer (prediction layer)
  - The output values lie in [0 1]
    - Can be used as a normalization layer
  - Turn the input vector into a probability output vector
    - Sum of all elements is 1. Each element  $\geq 0$ .
    - The output is class probability scores

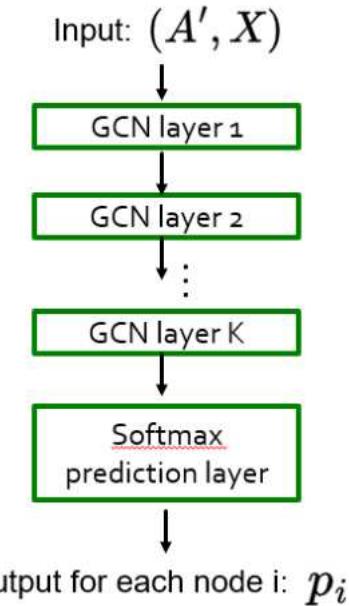
$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^C e^{x_j}}$$

Example (3 classes):

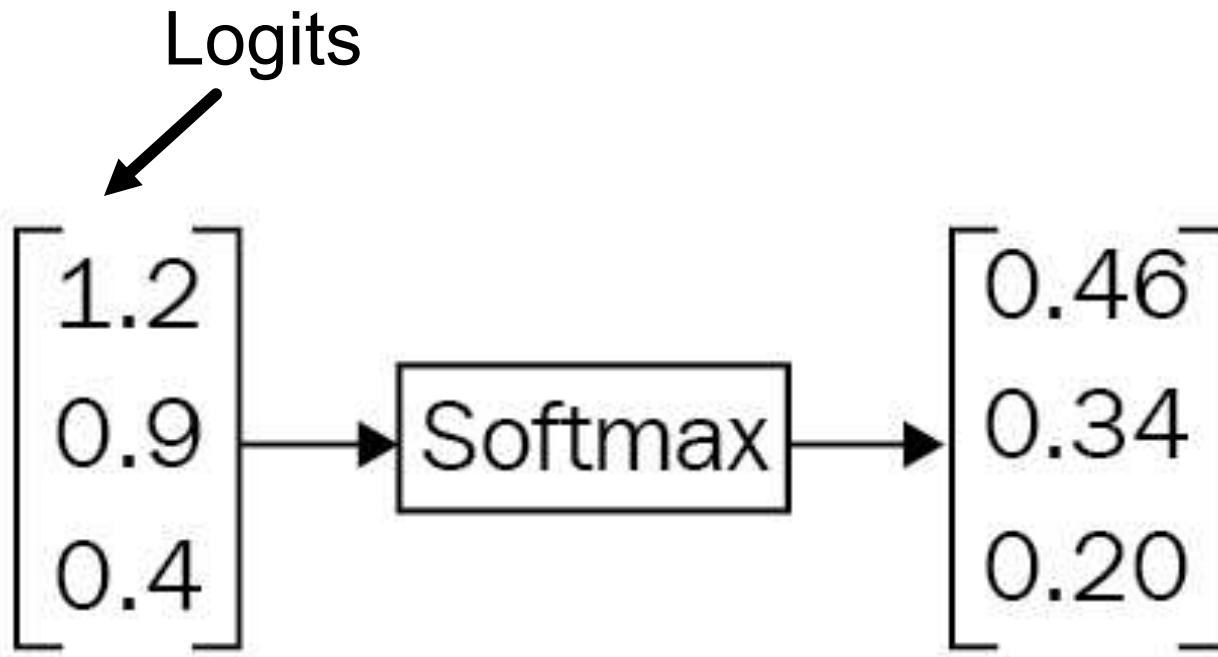
The output is a 3-dimensional score vector for each node



Recall: GCN for classification:

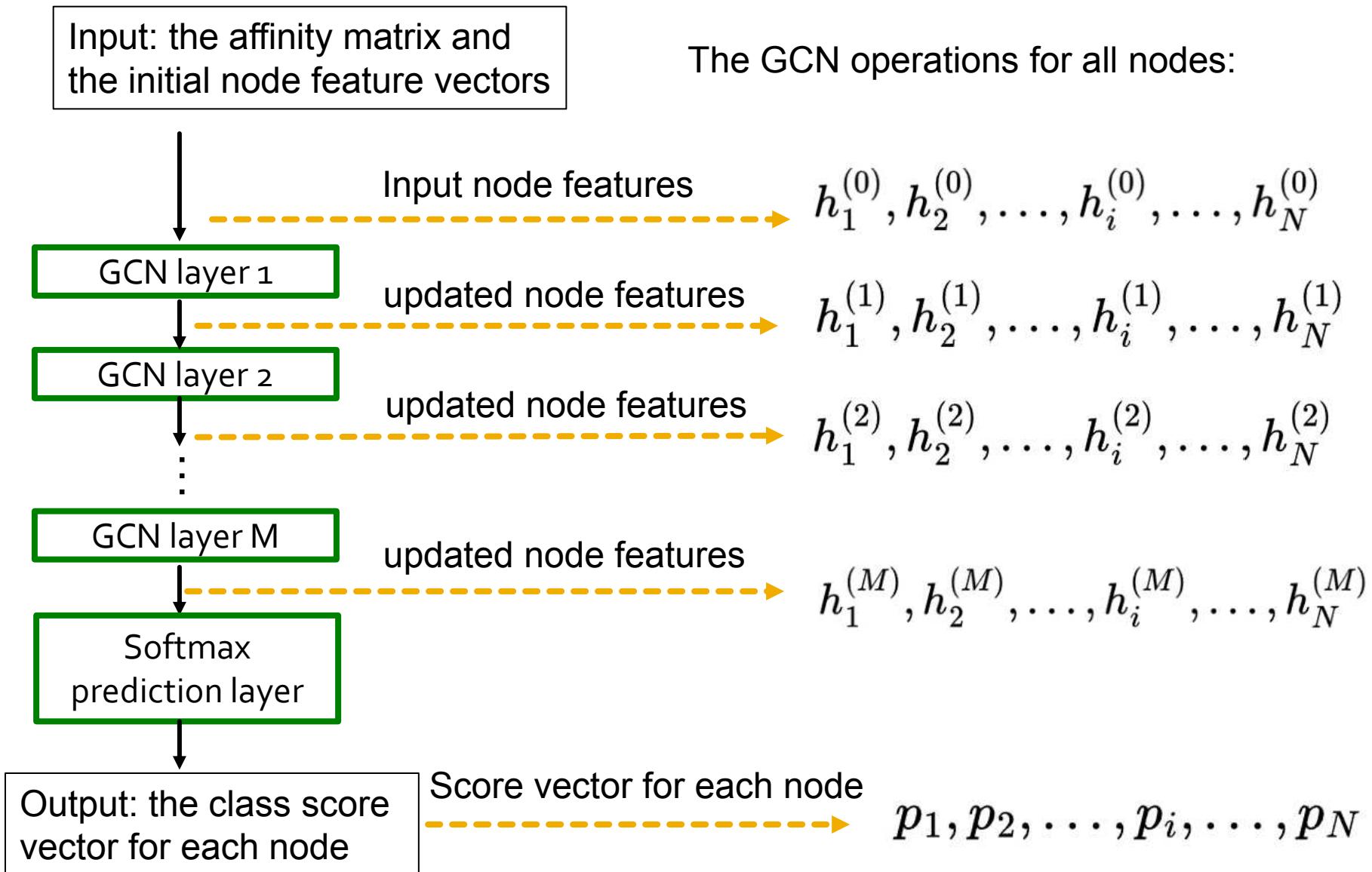


The input vector before using the Softmax function for class prediction is called logits.



<https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d>

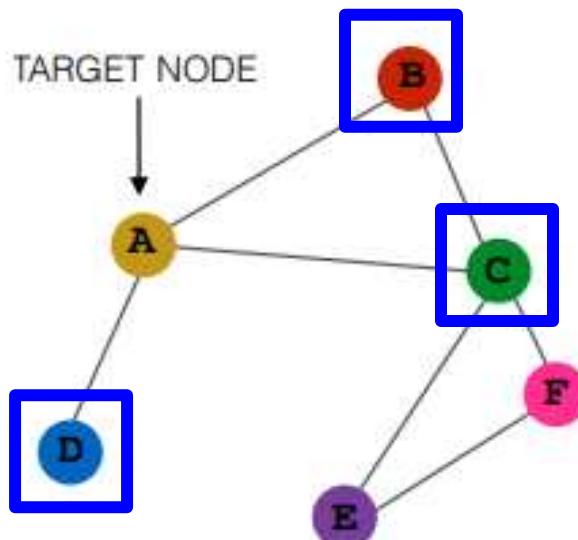
# Recap of GCN for node classification



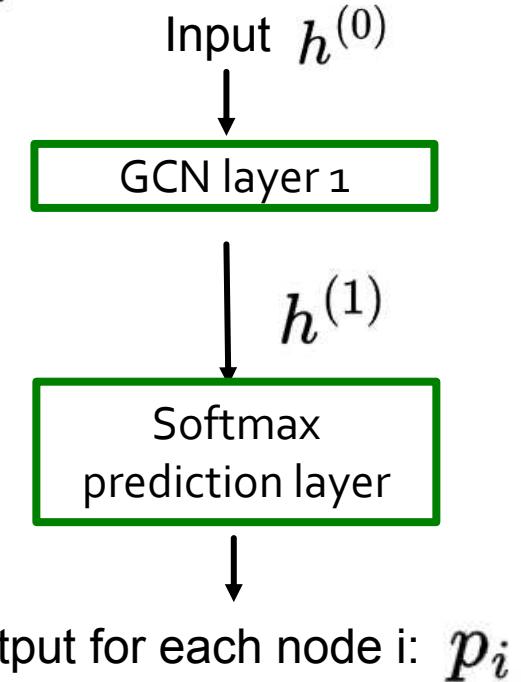
# Discussion

If we only have 1 layer of GCN,  
we only perform local message passing

$$\hat{h}_i^{(k)} = W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)} + B_k h_i^{(k-1)}$$



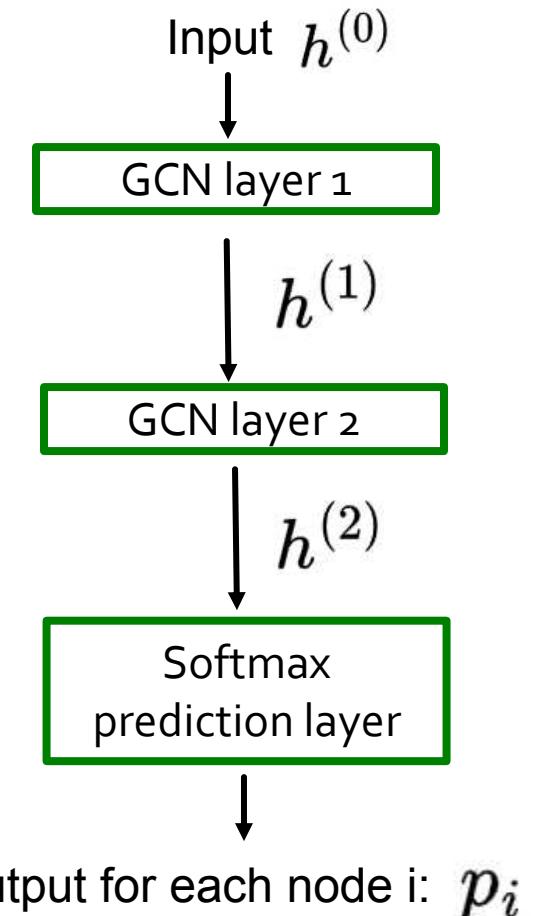
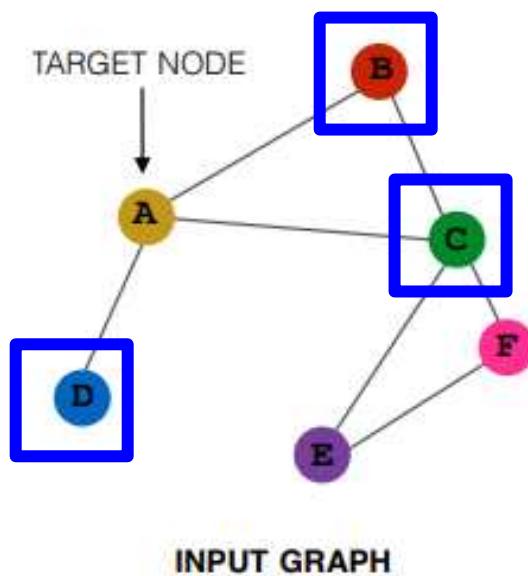
INPUT GRAPH



# Discussion

How about 2 GCN layers?

$$\hat{h}_i^{(k)} = W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)} + B_k h_i^{(k-1)}$$



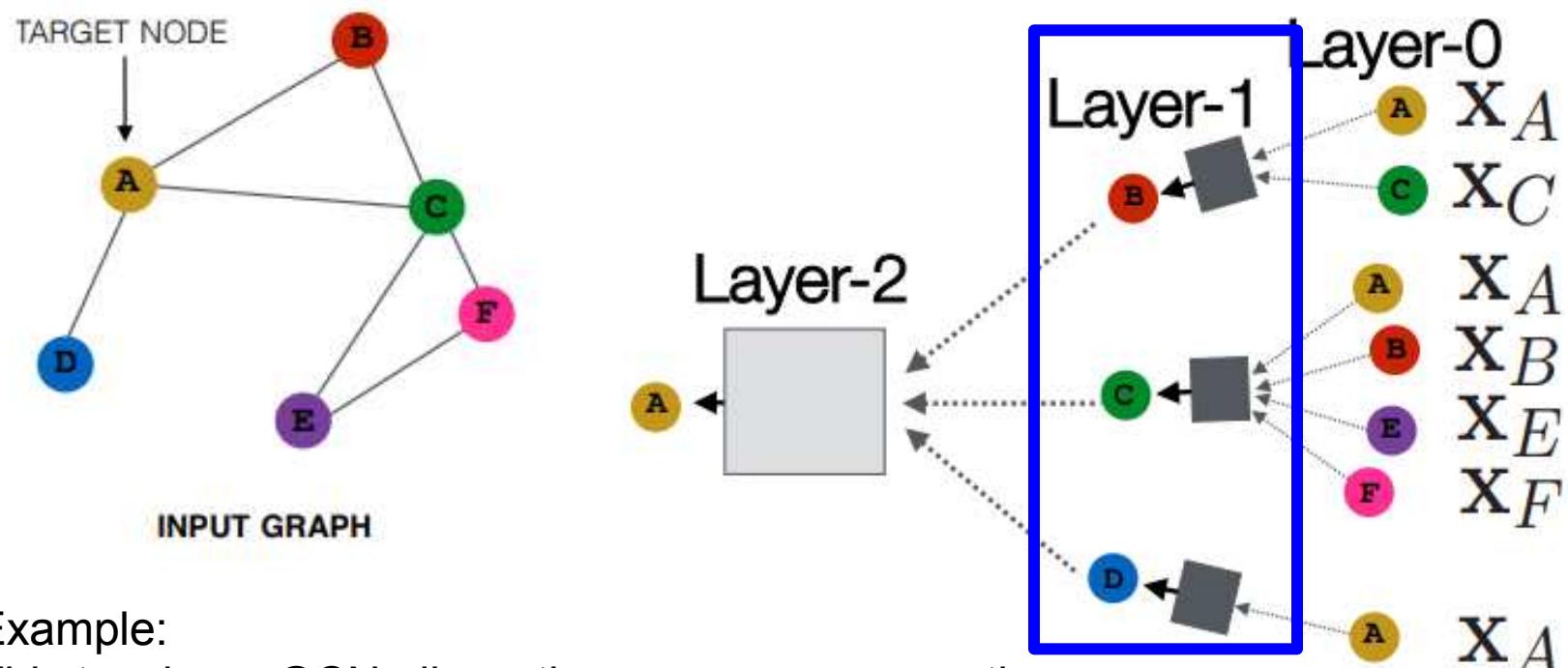
Messages will be propagated from distant nodes to the target node by multiple-layer neighbourhood propagation.

$h^{(2)}$  Will encode information from distant nodes

# Discussion

**Message passing is not local in a multi-layer GCN.**

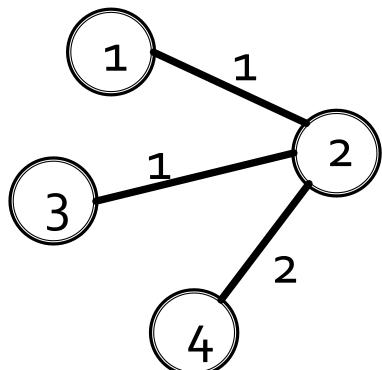
Messages will be propagated from distant nodes to the target node by multiple-layer neighbourhood propagation.



Example:

This two-layer GCN allows the message propagation from any other nodes in the graph to the target node A

# GCN Forward Pass example



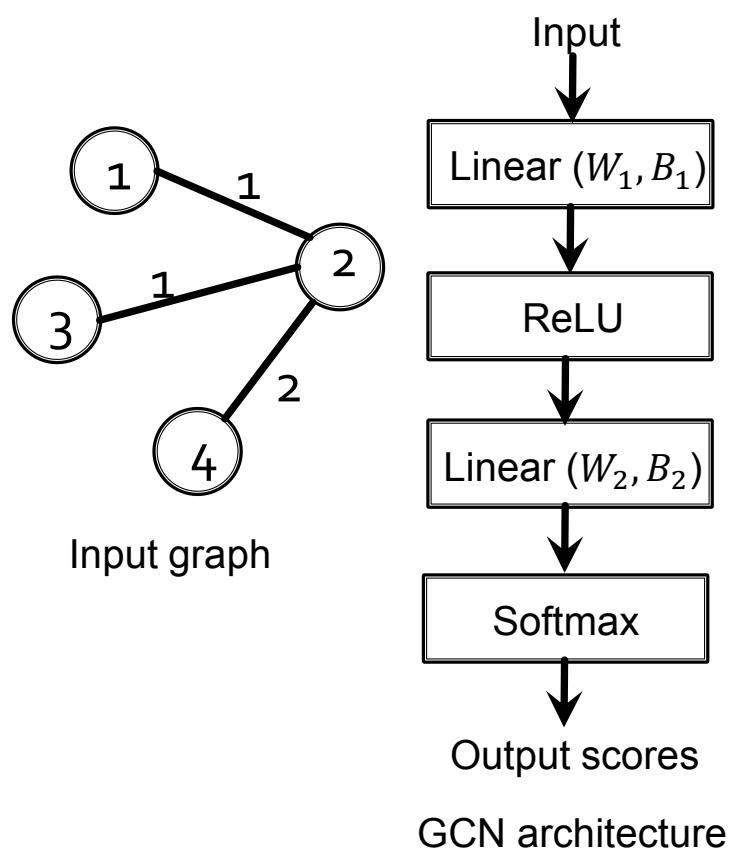
Input graph

## Question:

Given a graph below, the task is to do node-wise classification using a 2-layer graph convolutional network (GCN) and a cross-entropy loss, with network architecture shown. The initial node features are given below as:

$$x_1 = [1, -1]^T, x_2 = [0, 1]^T, \\ x_3 = [-1, 0]^T, x_4 = [1, 0]^T.$$

## Question: (continued from last page)



With the initial GCN weight parameters given below ( $W_k$  and  $B_k$  are the weight matrices for neighborhood aggregation and self transformation, respectively, for the  $k$ -th layer). Calculate the prediction of **node 2** by performing one forward pass.

$$W_1 = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \\ 0.1 & -0.1 \end{bmatrix} \quad B_1 = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.1 & 0.0 & -0.1 \\ 0.1 & 0.2 & -0.1 \end{bmatrix}$$

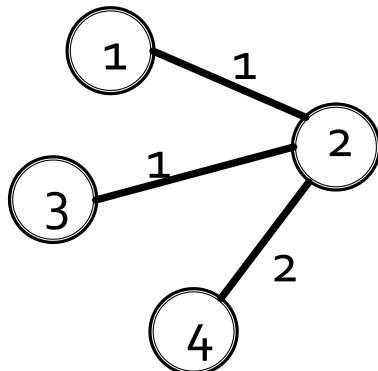
$$B_2 = \begin{bmatrix} 0.0 & 0.1 & 0.1 \\ 0.1 & -0.1 & -0.1 \end{bmatrix}$$

# 1<sup>st</sup> GCN layer

## GCN Forward Pass Example

**Solution:**

Calculate affinity matrix and row-normalized affinity matrix



Input graph

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \quad A' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.25 & 0 & 0.25 & 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Linear transform:

$$\hat{h}_i^{(k)} = W_k \sum_{j \in \mathcal{N}_i} a'_{ij} h_j^{(k-1)} + B_k h_i^{(k-1)}$$

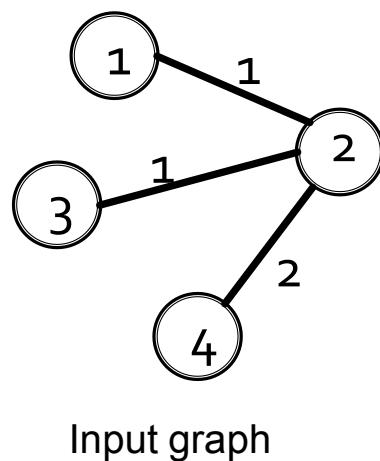
Non-linear transform (ReLU):

$$h_i^{(k)} = \max\{0, \hat{h}_i^{(k)}\}$$

## Forward pass for node 1

We have  $h_i^{(0)} = x_i$

Linear transform:



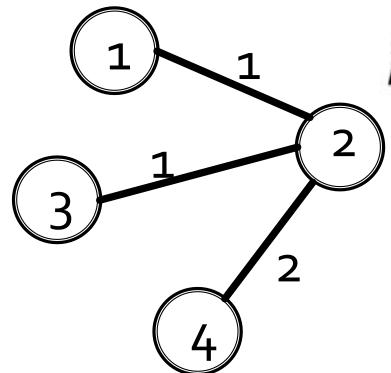
$$\sum_{j \in \mathcal{N}_1} a'_{1j} h_j^{(0)} = a'_{12} h_2^{(0)} = 1 \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$W_1 \sum_{j \in \mathcal{N}_1} a'_{1j} h_j^{(0)} = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \\ 0.1 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix}$$

$$B_1 h_1^{(0)} = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1 \\ -0.1 \end{bmatrix}$$

## Forward pass for node 1

Linear transform:



Input graph

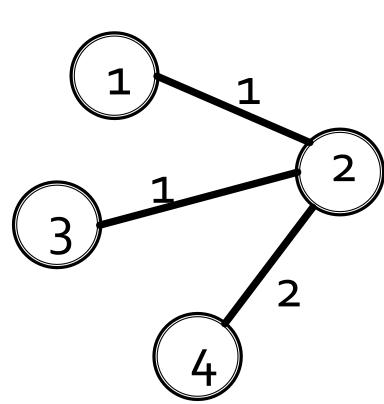
$$\hat{h}_1^{(1)} = W_1 \sum_{j \in \mathcal{N}_1} a'_{1j} h_j^{(0)} + B_1 h_1^{(0)} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.0 \\ -0.2 \end{bmatrix}$$

Non-linear transform (ReLU):

$$h_1^{(1)} = \max\{0, \hat{h}_1^{(1)}\} = \begin{bmatrix} 0.1 \\ 0.0 \\ 0.0 \end{bmatrix}$$

## Forward pass for node 2

We have  $h_i^{(0)} = x_i$

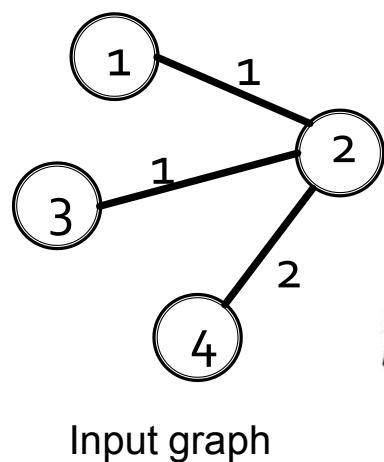


Input graph

$$\begin{aligned}
 \sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(0)} &= a'_{21} h_1^{(0)} + a'_{23} h_3^{(0)} + a'_{24} h_4^{(0)} \\
 &= 0.25 \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} + 0.25 \times \begin{bmatrix} -1 \\ 0 \end{bmatrix} + 0.5 \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
 &= \begin{bmatrix} 0.5 \\ -0.25 \end{bmatrix}
 \end{aligned}$$

$$W_1 \sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(0)} = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \\ 0.1 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0.5 \\ -0.25 \end{bmatrix} = \begin{bmatrix} -0.125 \\ 0.025 \\ 0.075 \end{bmatrix}$$

## Forward pass for node 2



$$B_1 h_2^{(0)} = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0.0 \\ 0.1 \end{bmatrix}$$

Linear transform:

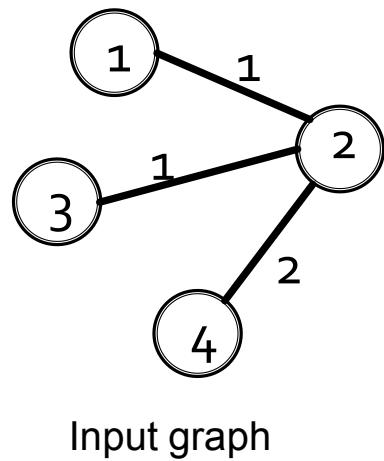
$$\hat{h}_2^{(1)} = W_1 \sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(0)} + B_1 h_2^{(0)} = \begin{bmatrix} -0.125 \\ 0.025 \\ 0.075 \end{bmatrix} + \begin{bmatrix} -0.1 \\ 0.0 \\ 0.1 \end{bmatrix} = \begin{bmatrix} -0.225 \\ 0.025 \\ 0.175 \end{bmatrix}$$

Non-linear transform (ReLU):

$$h_2^{(1)} = \max\{0, \hat{h}_2^{(1)}\} = \begin{bmatrix} 0.0 \\ 0.025 \\ 0.175 \end{bmatrix}$$

## Forward pass for node 3

We have  $h_i^{(0)} = x_i$



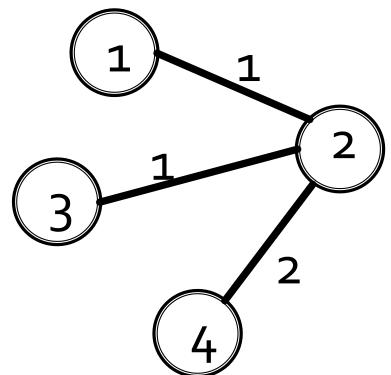
$$\sum_{j \in \mathcal{N}_3} a'_{3j} h_j^{(0)} = a'_{32} h_2^{(0)} = 1 \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$W_1 \sum_{j \in \mathcal{N}_3} a'_{3j} h_j^{(0)} = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \\ 0.1 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix}$$

$$B_1 h_3^{(0)} = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \times \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \\ 0.0 \end{bmatrix}$$

## Forward pass for node 3

Linear transform:



$$\hat{h}_3^{(1)} = W_1 \sum_{j \in \mathcal{N}_3} a'_{3j} h_j^{(0)} + B_1 h_3^{(0)} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.1 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.2 \\ -0.2 \\ -0.1 \end{bmatrix}$$

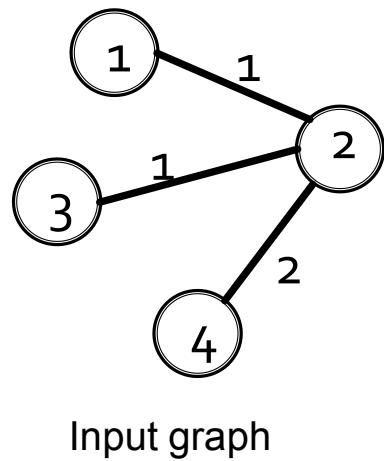
Non-linear transform (ReLU):

Input graph

$$h_3^{(1)} = \max\{0, \hat{h}_3^{(1)}\} = \begin{bmatrix} 0.2 \\ 0.0 \\ 0.0 \end{bmatrix}$$

## Forward pass for node 4

We have  $h_i^{(0)} = x_i$



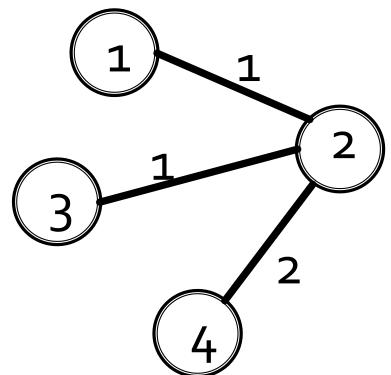
$$\sum_{j \in \mathcal{N}_4} a'_{4j} h_j^{(0)} = a'_{42} h_2^{(0)} = 1 \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$W_1 \sum_{j \in \mathcal{N}_4} a'_{4j} h_j^{(0)} = \begin{bmatrix} -0.2 & 0.1 \\ 0.0 & -0.1 \\ 0.1 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix}$$

$$B_1 h_4^{(0)} = \begin{bmatrix} -0.1 & -0.1 \\ 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0.1 \\ 0.0 \end{bmatrix}$$

## Forward pass for node 4

Linear transform:



$$\hat{h}_4^{(1)} = W_1 \sum_{j \in \mathcal{N}_4} a'_{4j} h_j^{(0)} + B_1 h_4^{(0)} = \begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \end{bmatrix} + \begin{bmatrix} -0.1 \\ 0.1 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.0 \\ -0.1 \end{bmatrix}$$

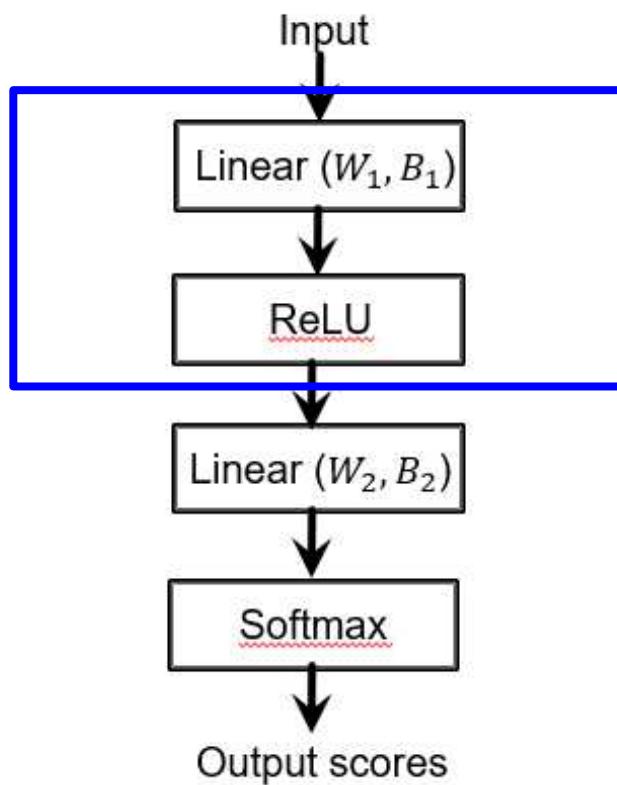
Non-linear transform (ReLU):

Input graph

$$h_4^{(1)} = \max\{0, \hat{h}_4^{(1)}\} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

## ■ Summary (1<sup>st</sup> GCN layer)

We have finished the 1<sup>st</sup> GCN layer



Node features output by the 1<sup>st</sup> GCN layer:

$$h_1^{(1)} = \max\{0, \hat{h}_1^{(1)}\} = \begin{bmatrix} 0.1 \\ 0.0 \\ 0.0 \end{bmatrix}$$

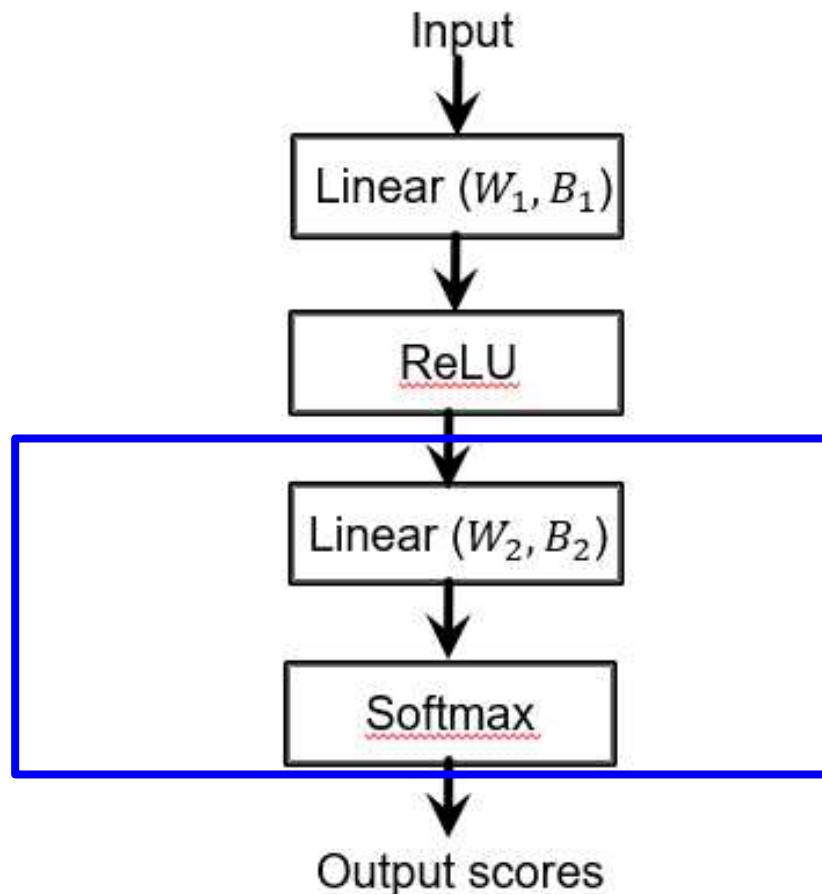
$$h_2^{(1)} = \max\{0, \hat{h}_2^{(1)}\} = \begin{bmatrix} 0.0 \\ 0.025 \\ 0.175 \end{bmatrix}$$

$$h_3^{(1)} = \max\{0, \hat{h}_3^{(1)}\} = \begin{bmatrix} 0.2 \\ 0.0 \\ 0.0 \end{bmatrix}$$

$$h_4^{(1)} = \max\{0, \hat{h}_4^{(1)}\} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

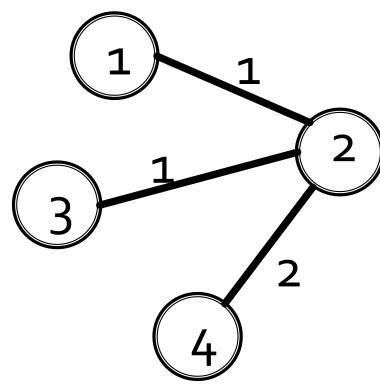
# 2<sup>nd</sup> GCN layer

Next step: the 2<sup>nd</sup> GCN layer and Softmax prediction



GCN architecture

## Forward pass (2<sup>nd</sup> GCN layer) for node 2

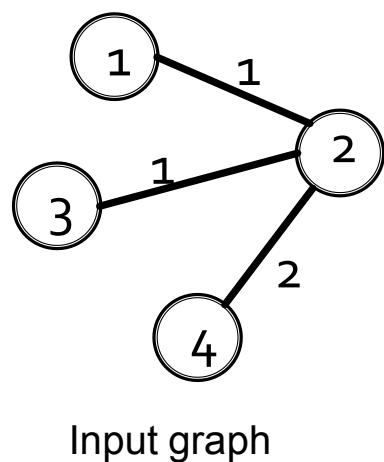


Input graph

$$\begin{aligned}
 \sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(1)} &= a'_{21} h_1^{(1)} + a'_{23} h_3^{(1)} + a'_{24} h_4^{(1)} \\
 &= 0.25 \times \begin{bmatrix} 0.1 \\ 0.0 \\ 0.0 \end{bmatrix} + 0.25 \times \begin{bmatrix} 0.2 \\ 0.0 \\ 0.0 \end{bmatrix} + 0.5 \times \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{bmatrix} \\
 &= \begin{bmatrix} 0.075 \\ 0.0 \\ 0.0 \end{bmatrix}
 \end{aligned}$$

$$W_2 \sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(1)} = \begin{bmatrix} 0.1 & 0.0 & -0.1 \\ 0.1 & 0.2 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0.075 \\ 0.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.0075 \\ 0.0075 \end{bmatrix}$$

## Forward pass (2<sup>nd</sup> GCN Layer) for node 2



$$B_2 h_2^{(1)} = \begin{bmatrix} 0.0 & 0.1 & 0.1 \\ 0.1 & -0.1 & -0.1 \end{bmatrix} \times \begin{bmatrix} 0.0 \\ 0.025 \\ 0.175 \end{bmatrix} = \begin{bmatrix} 0.02 \\ -0.02 \end{bmatrix}$$

Linear transform:

$$\hat{h}_2^{(2)} = W_2 \sum_{j \in \mathcal{N}_2} a'_{2j} h_j^{(1)} + B_2 h_2^{(1)} = \begin{bmatrix} 0.0075 \\ 0.0075 \end{bmatrix} + \begin{bmatrix} 0.02 \\ -0.02 \end{bmatrix} = \begin{bmatrix} 0.0275 \\ -0.0125 \end{bmatrix}$$

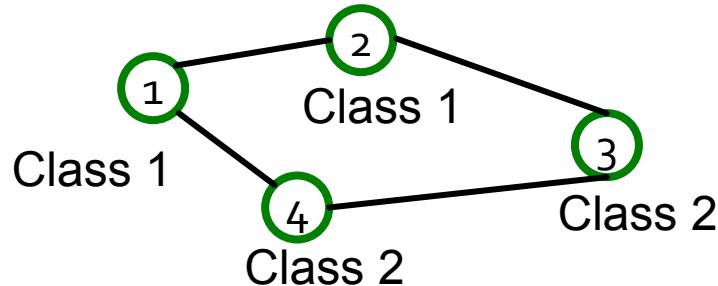
Softmax:

$$h_2^{(2)} = \text{Softmax}(\hat{h}_2^{(2)}) = \text{Softmax}\left(\begin{bmatrix} 0.0275 \\ -0.0125 \end{bmatrix}\right) = \begin{bmatrix} 0.51 \\ 0.49 \end{bmatrix}$$

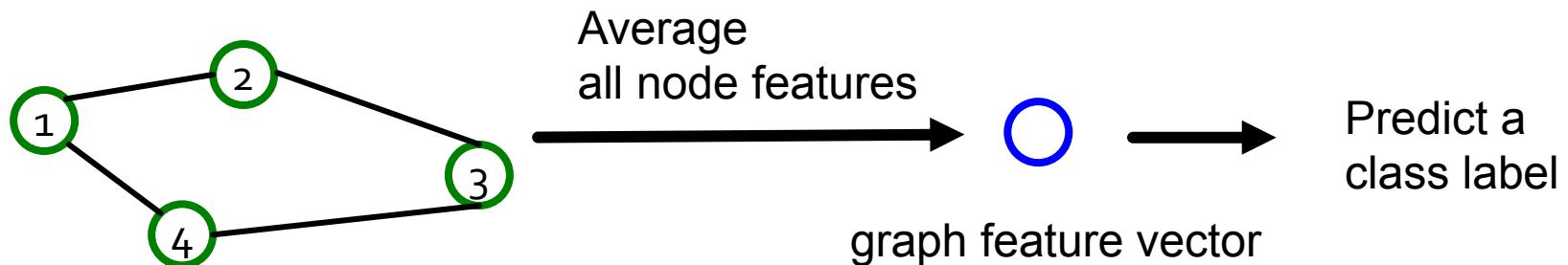
$$e^{z1} + e^{z2} = e^{0.0275} + e^{-0.0125} = 1.0279 + 0.9876 = 2.0155$$

# Graph classification tasks

- Node classification
  - Predict a class label for each node in a graph



- Graph classification
  - Predict a class label for the whole graph



# GCN discussions

- Input feature vectors for GCN:
  - The input node feature vector depends on applications:
    - Examples:
      - Social networks: use one-hot feature vectors.
      - 3D segmentation: 3D coordinate (x, y, z) for each 3D point
    - General case:

Using one-hot feature vectors (or called indicator vectors).

      - E.g., given 4 nodes in total, the one-hot vectors can be:

$x_1 = [1, 0, 0, 0]^T;$

$x_2 = [0, 1, 0, 0]^T;$

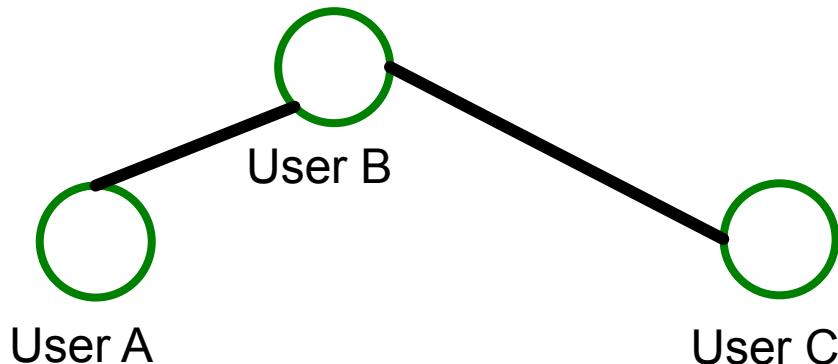
$x_3 = [0, 0, 1, 0]^T;$

$x_4 = [0, 0, 0, 1]^T;$

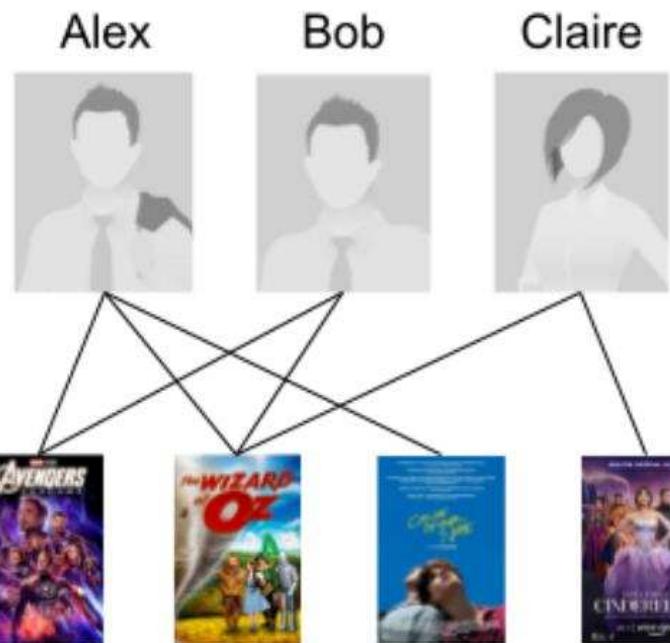
A one-hot vector:  
encode the identity of one  
input node

# GCN application examples

- User attribute (tag) prediction
  - Attributes can be occupations, professionals, interests, ...
  - Datasets: social network datasets
  - Node: each user;  
Edge: friendship relation; contact record; subscription, following relationship.



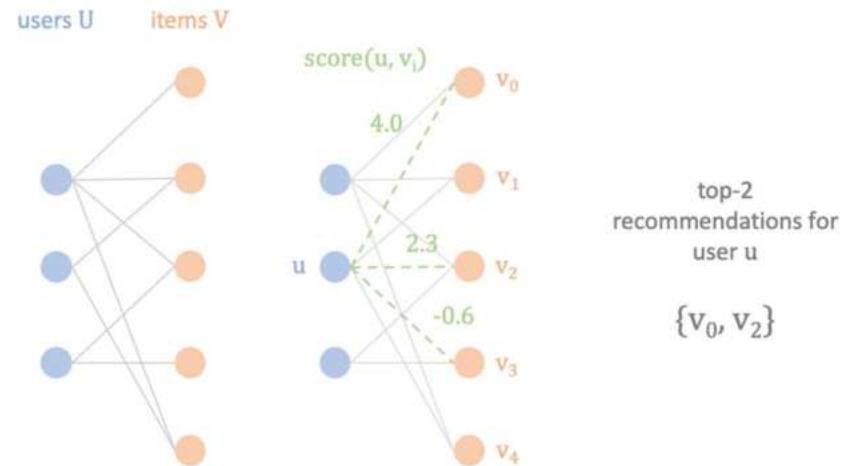
Predict the attributes of a user as a binary classification problem for each attribute:  
Interests: {dancing, reading, traveling, cycling, swimming, PC gaming, ....}



User-Item interactions can be modeled as a bipartite undirected graph.

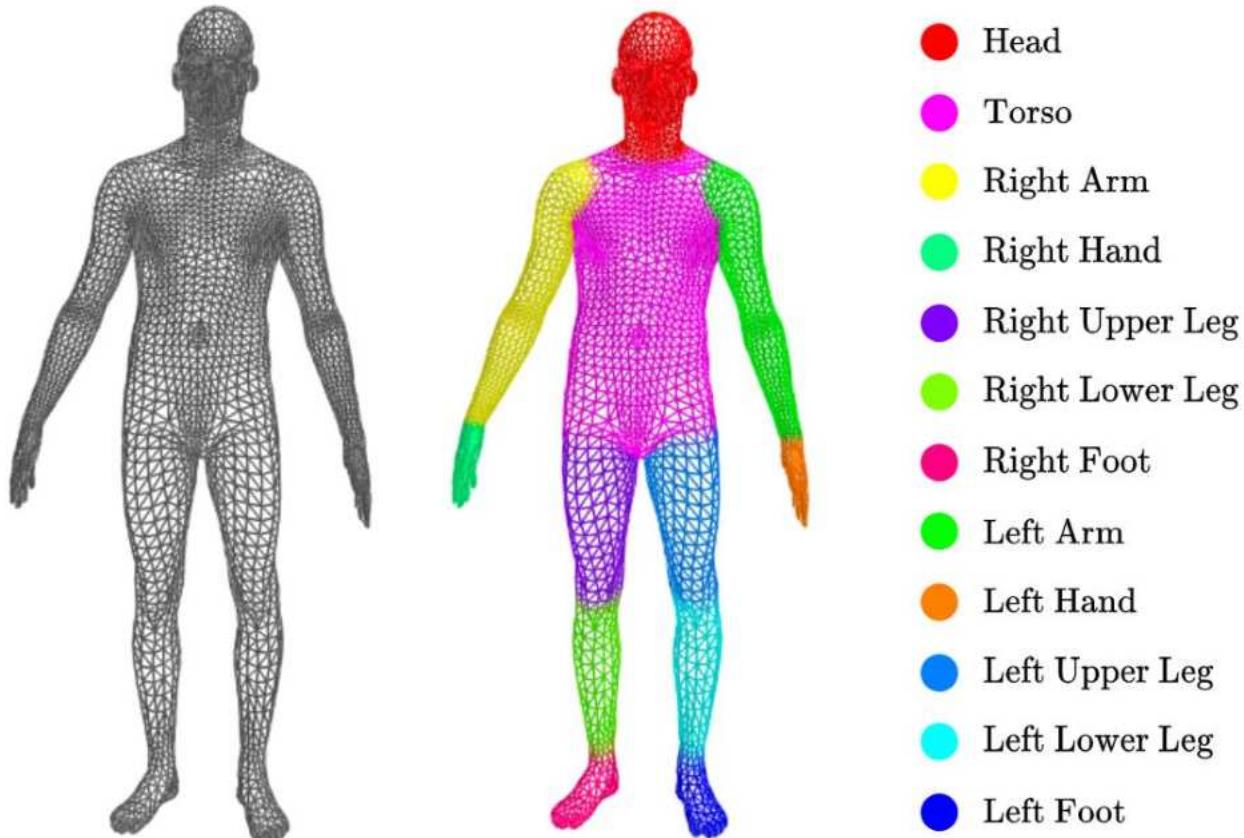
## GCN for Recommender Systems

Classify positive and negative pairs  
Positive  $(u, v)$ : connected pairs  
Negative  $(u, v)$ : not connected pairs  
Use dot-product to calculate the similarity of a user-item pair



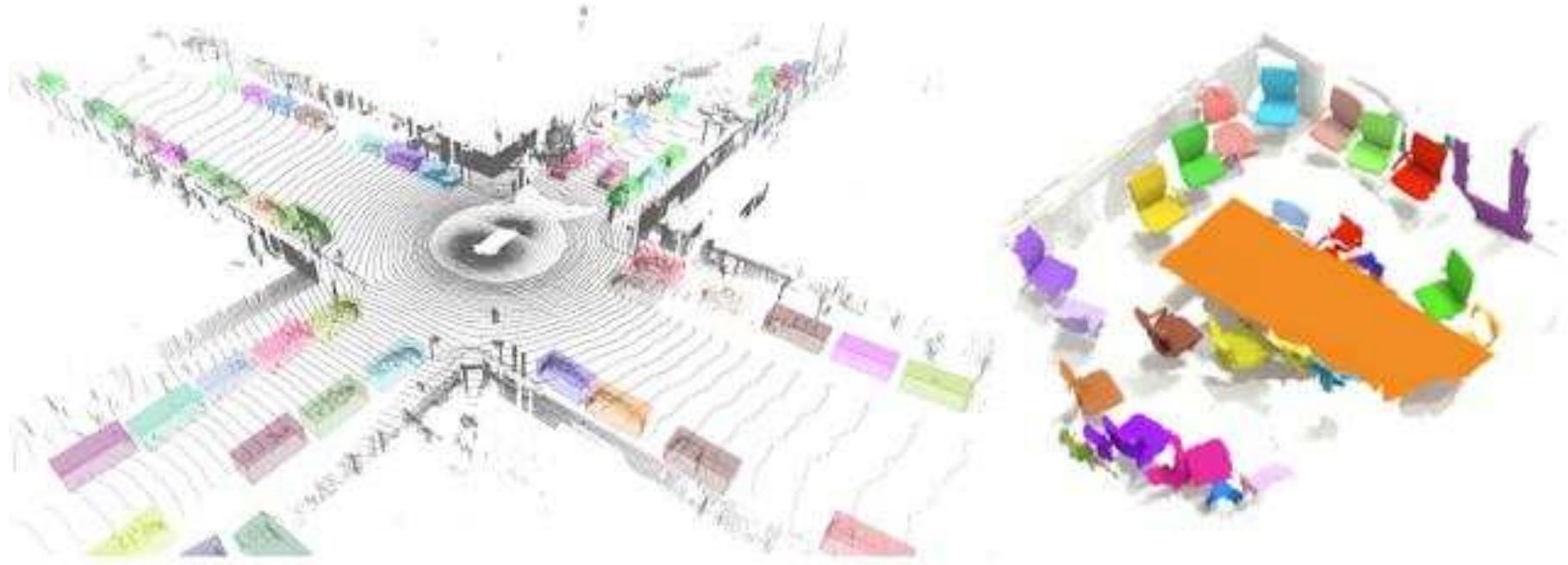
# GCN applications

GCN for node classification on 3D meshes:



Graph segmentation task: each vertex in the mesh is assigned to one of twelve body-parts.  
Node features: (x,y,z) coordinates of each vertice

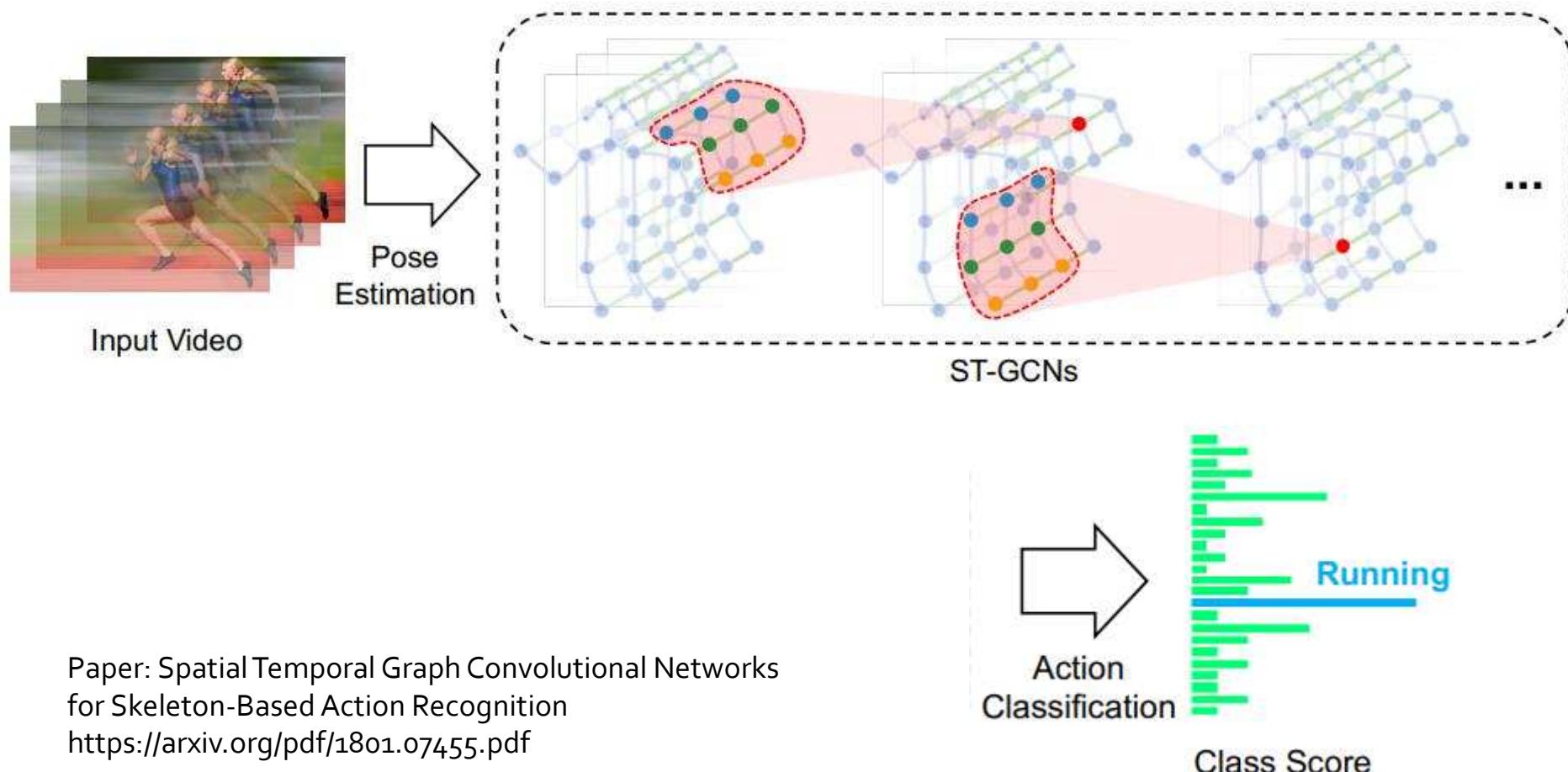
<https://medium.com/stanford-cs224w/deep-learning-on-3d-meshes-9608a5b33c98>



3D point cloud segmentation

<https://ai.googleblog.com/2021/02/3d-scene-understanding-with-tensorflow.html>

## ■ GCN for skeleton based action recognition



## ■ GCN for scene graph generation and captioning

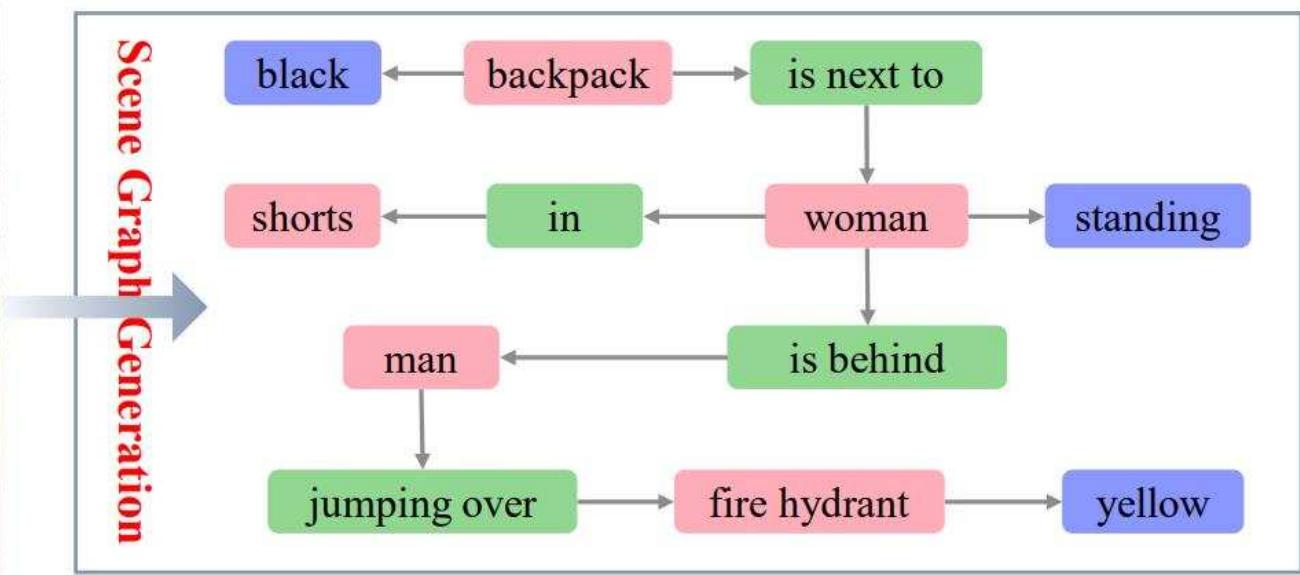
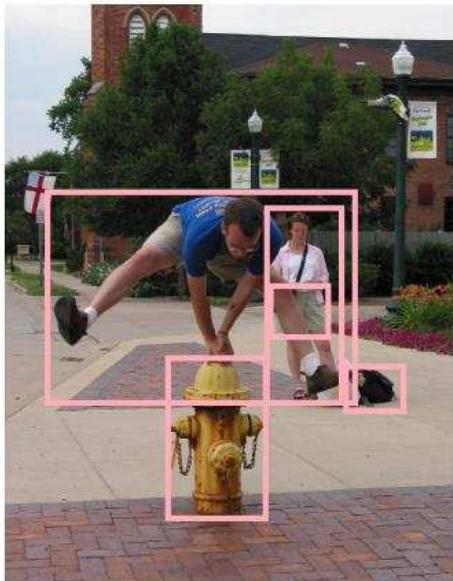


Image captioning

The woman in shorts is standing behind the man who is jumping over fire hydrant.

<https://arxiv.org/pdf/2201.00443.pdf>

# More applications

- A collection of graph Learning tutorials/applications
  - <https://medium.com/stanford-cs224w>
  - [https://github.com/pyg-team/pytorch\\_geometric#implemented-gnn-models](https://github.com/pyg-team/pytorch_geometric#implemented-gnn-models)
- other examples:
  - <https://paperswithcode.com/task/node-classification>

# GCN discussions

- Other deep learning methods for graphs:
  - Transformer
    - Another popular method that can be used on graphs
      - <http://papers.neurips.cc/paper/9367-graph-transformer-networks.pdf>
      - <http://web.stanford.edu/class/cs224w/>



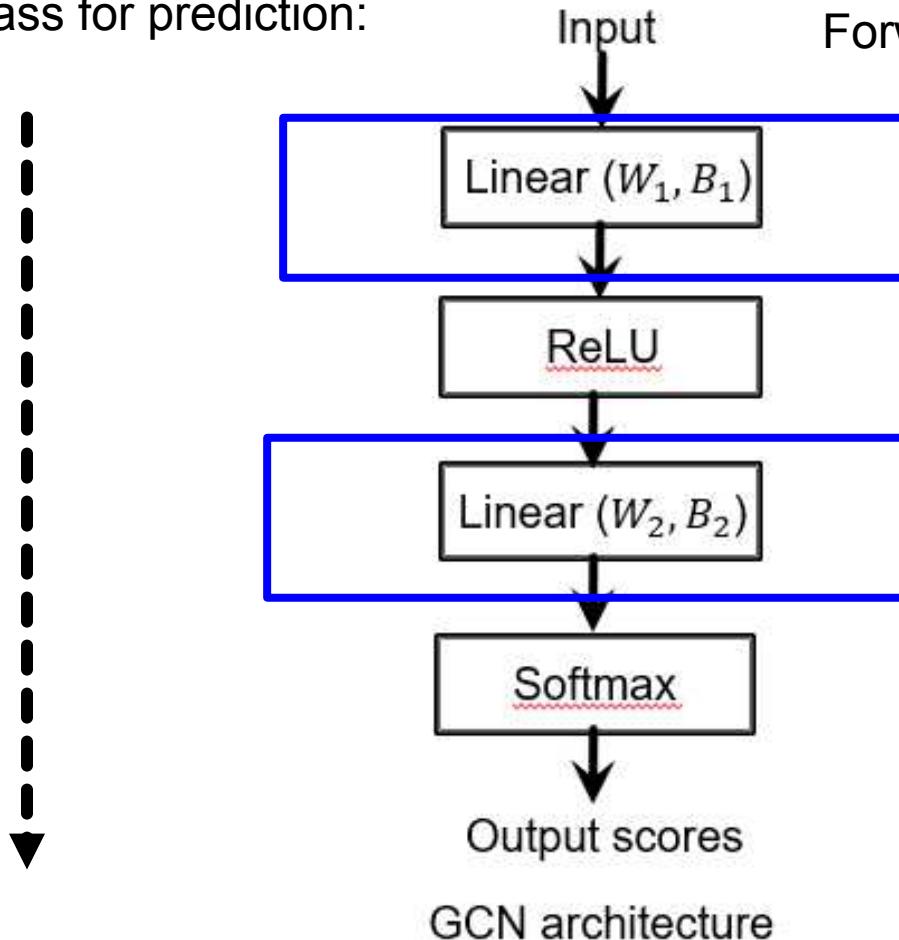
# GCN training

- GCN training
  - Loss functions
    - Cross-entropy loss
    - L2 loss
  - Backward pass
    - gradient descent methods

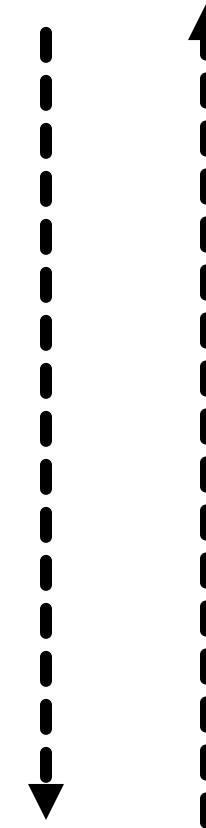
# GCN training

Use gradient descent for training

Forward pass for prediction:



Forward pass + Backward pass

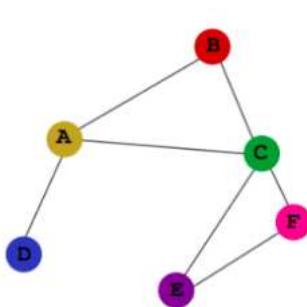


Training: find solutions to the network parameters in each layer!

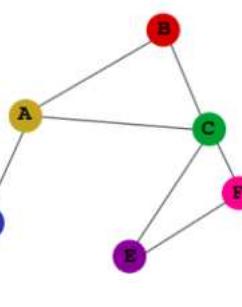
# GCN training

## ■ Training data

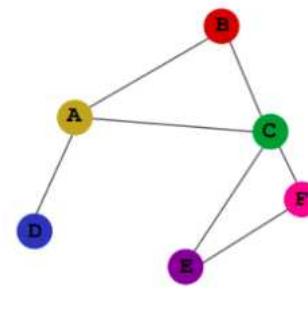
Training set:



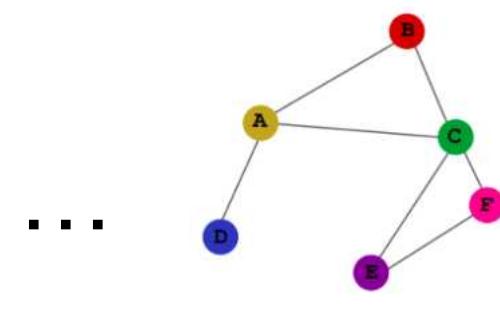
graph 1



graph 2



graph 3

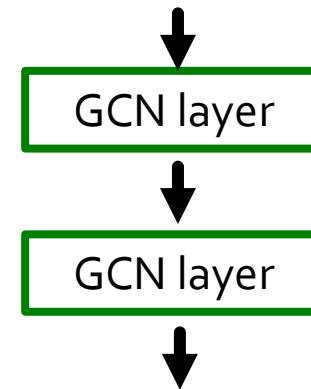


graph n



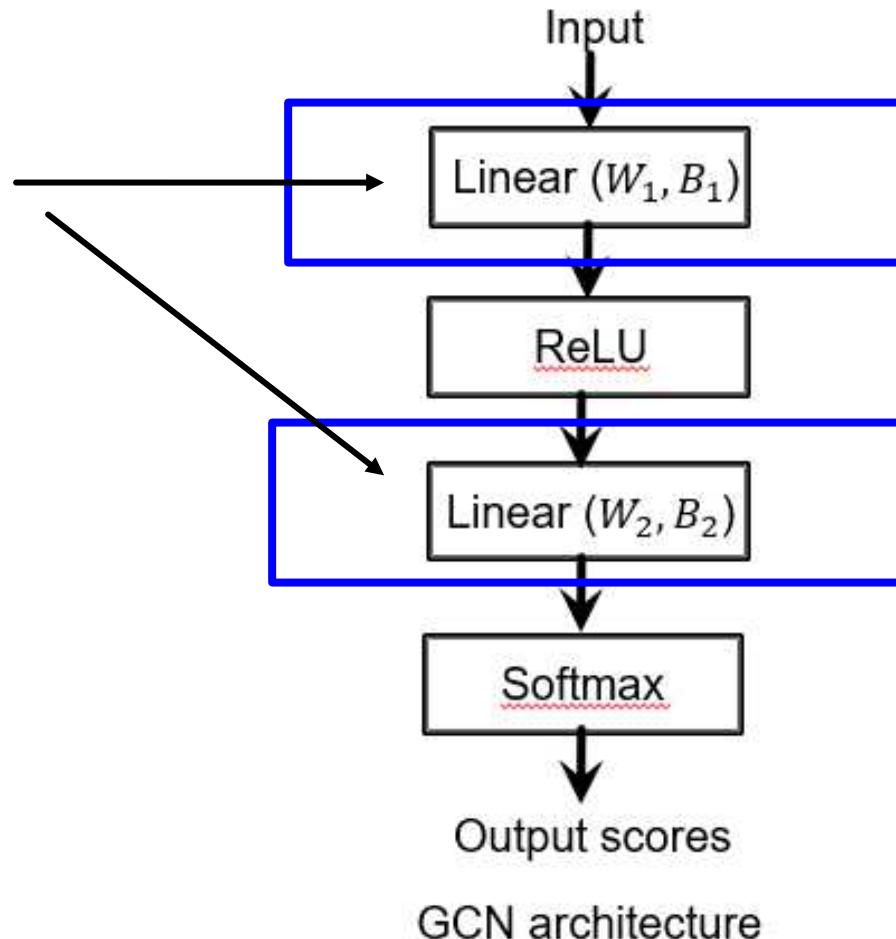
Class label for each node  
(ground-truth labels)

Train a GCN model



Example:

Learnable parameters



## ■ GCN loss functions

1. Cross-entropy loss for classification:

$$L(f(x)) = - \sum_i \mathbf{y}_i \cdot \log \mathbf{z}_i(\theta)$$

$$z_{ik}(\theta) = \text{Softmax}(h_{ik}) = \frac{e^{h_{ik}(\theta)}}{\sum_{j=1}^C e^{h_{ij}(\theta)}}$$

2. For node regression problems, you may use L<sub>2</sub> loss:  
E.g., predict real numbers for each node

$$L(f(\mathbf{x})) = \sum_{\mathbf{i}} \|\mathbf{y}_{\mathbf{i}} - \mathbf{h}_{\mathbf{i}}(\mathbf{x})\|^2$$

# GCN Loss functions

- Loss function for node classification
  - Cross-entropy loss (vector expression, also called softmax loss)

$$L(f(x)) = - \sum_i \mathbf{y}_i \cdot \log \mathbf{z}_i(\theta)$$

$$z_{ik}(\theta) = \text{Softmax}(h_{ik}) = \frac{e^{h_{ik}(\theta)}}{\sum_{j=1}^C e^{h_{ij}(\theta)}}$$

$\mathbf{z}_i$  : The score vector output by Softmax for the node i

$z_{ik}$  : The k-th element the score vector  $\mathbf{z}_i$

$h_{ik}$  : The k-th element of the logit vector output by the network, corresponding to the k-th class.

$\mathbf{y}_i$  : The one-hot vector indicating the ground-truth class label

Cross-entropy measures the distance between the ground-truth vector and the prediction vector.

One-hot vector example:

One-hot vector is the ground-truth class score vector

Groudtruth class label:

Represented by one-hot vector:

$$\text{Class 1} \longrightarrow [ 1, 0, 0, 0 ]^T$$

$$\text{Class 2} \longrightarrow [ 0, 1, 0, 0 ]^T$$

$$\text{Class 3} \longrightarrow [ 0, 0, 1, 0 ]^T$$

$$\text{Class 4} \longrightarrow [ 0, 0, 0, 1 ]^T$$

For the c-th class, the c-th element in the one-hot vector is set to 1

Cross-entropy loss:

$$\min_{\theta} L(Y, Z) = - \sum_i y_i \cdot \log z_i(\theta)$$

If the score vector  $z_i$  is similar to the groundtruth vector  $y_i$ , the loss is small.

$\theta$  is the learnable parameters of all GCN layers:

$$\theta = \{W_1, B_1; W_2, B_2; \dots; W_K, B_K; \}$$

GCN training:

We need to solve the optimization problem to obtain the solution of the learnable parameters

$$L(f(x)) = - \sum_i \mathbf{y}_i \cdot \log \mathbf{z}_i(\theta)$$



- Cross-entropy loss can be also written as:

$$L(f(x)) = - \sum_i \sum_{k=1}^C y_{ik} \log z_{ik}(\theta)$$

$$z_{ik}(\theta) = \text{Softmax}(h_{ik}) = \frac{e^{h_{ik}(\theta)}}{\sum_{j=1}^C e^{h_{ij}(\theta)}}$$

k indicates the k-th class. C is the total number of classes

$y_{ik}$  : is the k-th element in the groundtruth one-hot vector  $\mathbf{y}_i$

Cross-entropy between two distributions: measure the distance (similarity) between two distributions:

Example: we have 3 classes in total.

$\mathbf{y}$ : ground-truth distribution for one sample

$\mathbf{z}$ : predicted distribution for one sample

Ground-truth	Prediction 1	Prediction 2
$\mathbf{y}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$	$\mathbf{z}_1 = \begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix}$	$\mathbf{z}'_1 = \begin{bmatrix} 0.1 \\ 0.9 \\ 0.0 \end{bmatrix}$

Cross-entropy for one sample:  $L_i(\mathbf{y}_i, \mathbf{z}_i) = - \sum_{k=1}^C y_{ik} \log z_{ik}$

$$L(\mathbf{y}_1, \mathbf{z}_1) = -\log(0.5) = 0.3010 \quad \text{Large distance}$$

$$L(\mathbf{y}_1, \mathbf{z}'_1) = -\log(0.9) = 0.0458 \quad \text{Small distance}$$

We need to solve this optimization problem to obtain the network parameters (weights)

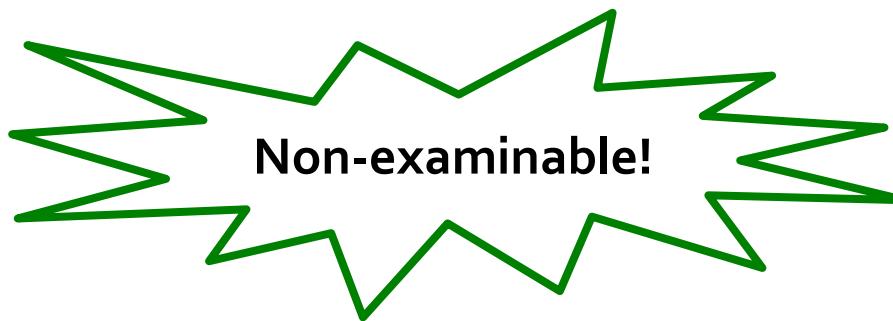
Cross-entropy loss:

$$\min_{\theta} L(Y, Z) = - \sum_i y_i \cdot \log z_i(\theta)$$

GCN training:

We need to solve the optimization problem to obtain the solution of the learnable parameters

- GCN training
  - Backward pass
    - gradient descent methods



## ■ Use gradient decent to solve the optimization

Iterative algorithm: repeatedly update weights in the (opposite) direction of gradients until convergence

$$\theta(t + 1) = \theta(t) + \Delta\theta(t);$$

$$\Delta\theta(t) = -\eta \nabla_{\theta} L;$$

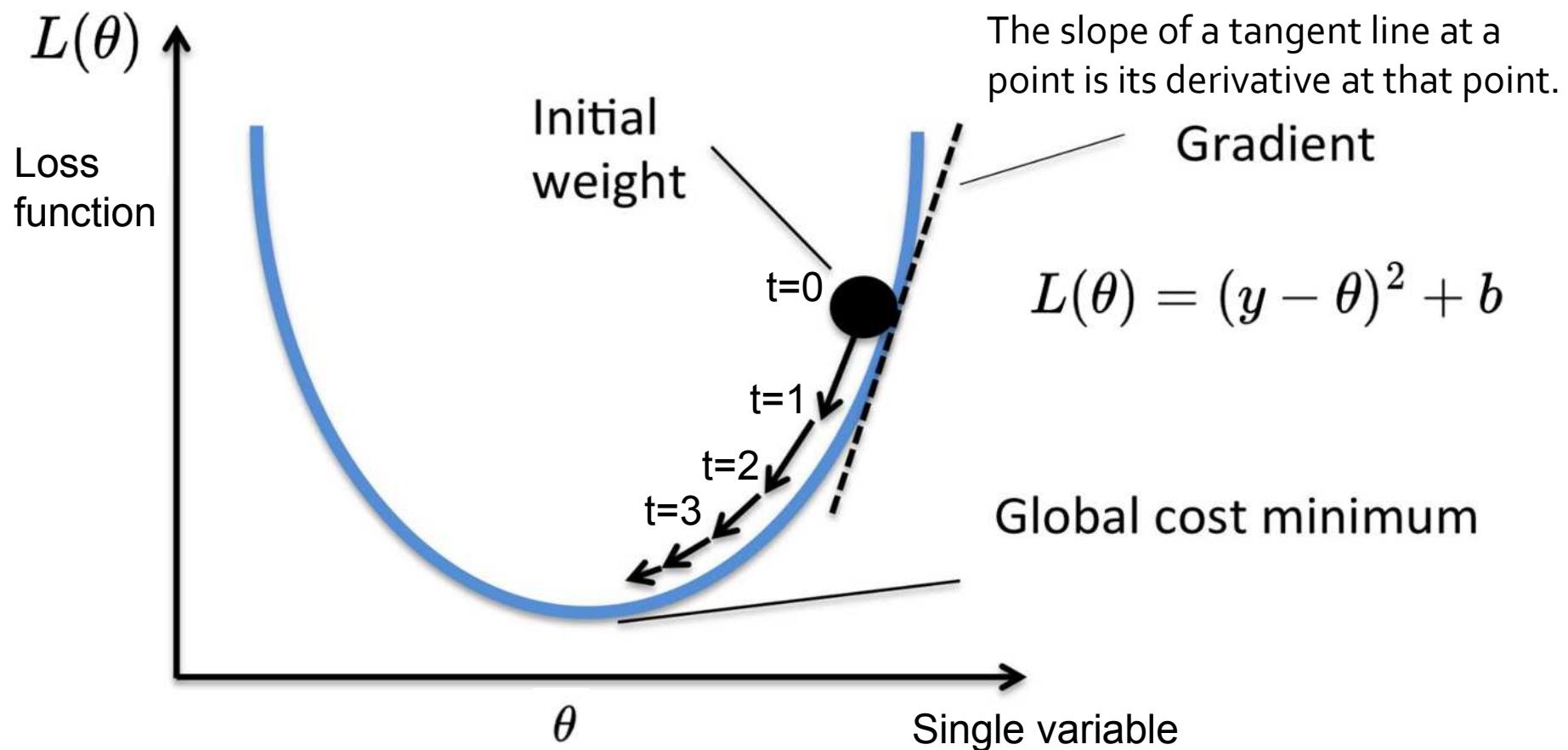
$\Delta\theta(t)$  : The update vector, a small change to the solution.

$\eta$  : Learning rate (LR), the step size for gradient update:  
Hyperparameter that controls the size of gradient step  
Can vary over the course of training (LR scheduling)

Gradient vector: a vector of partial derivatives of all variables.

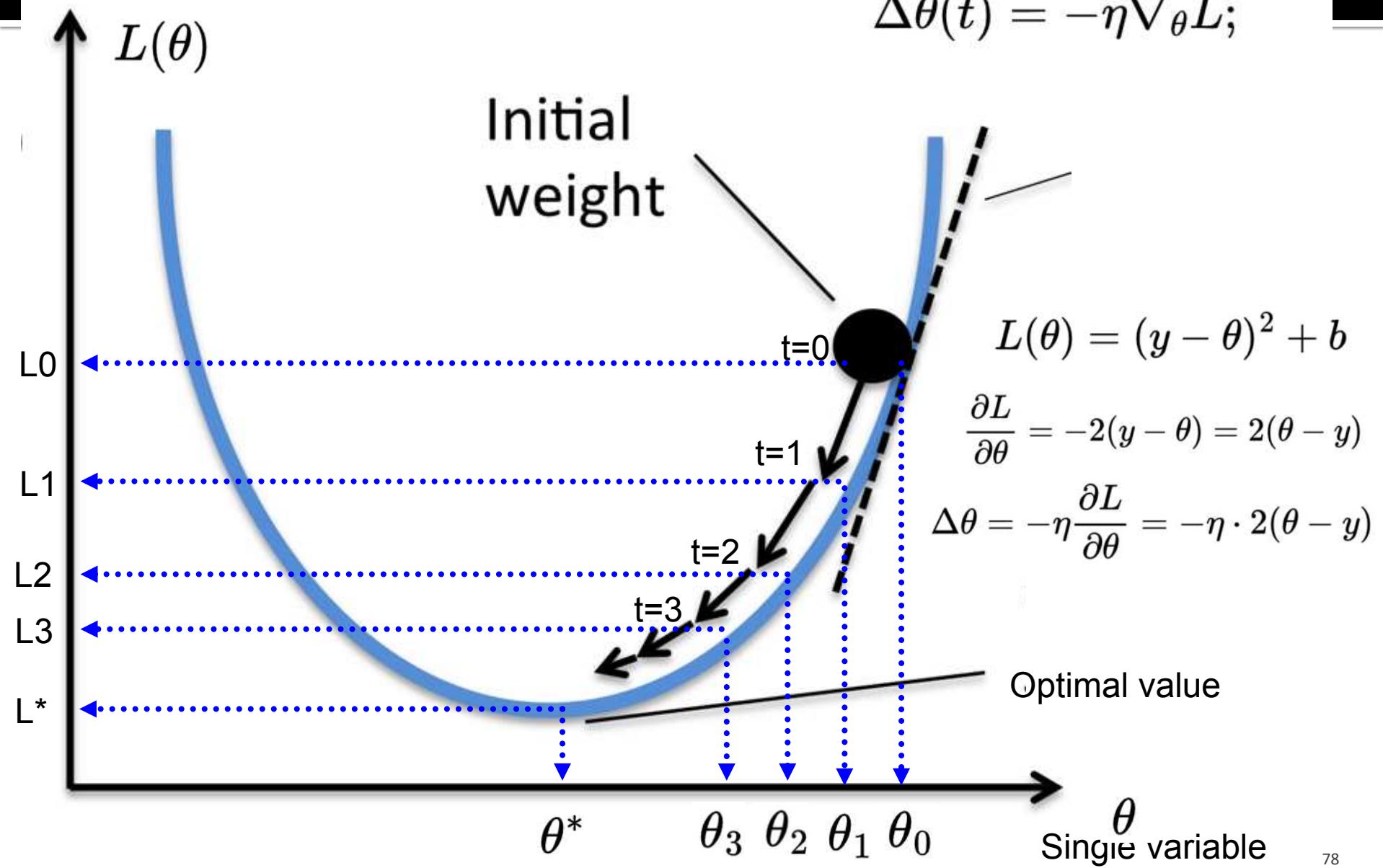
Two key factors for gradient descent:  
1 update direction (gradients),  
2 update step size (learning rate)

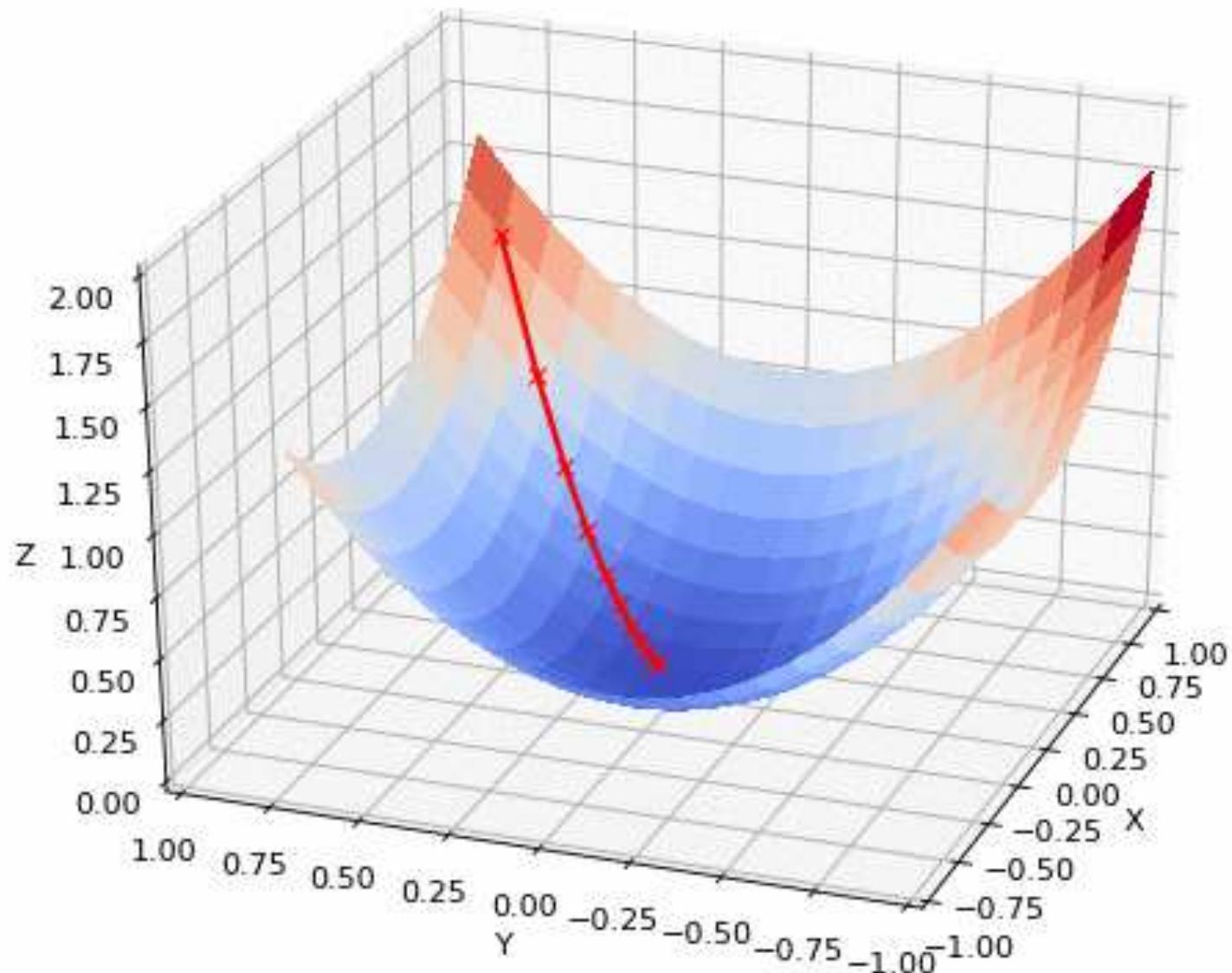
$$\theta(t + 1) = \theta(t) + \Delta\theta(t);$$
$$\Delta\theta(t) = -\eta \nabla_{\theta} L;$$



$$\theta(t+1) = \theta(t) + \Delta\theta(t);$$

$$\Delta\theta(t) = -\eta \nabla_{\theta} L;$$



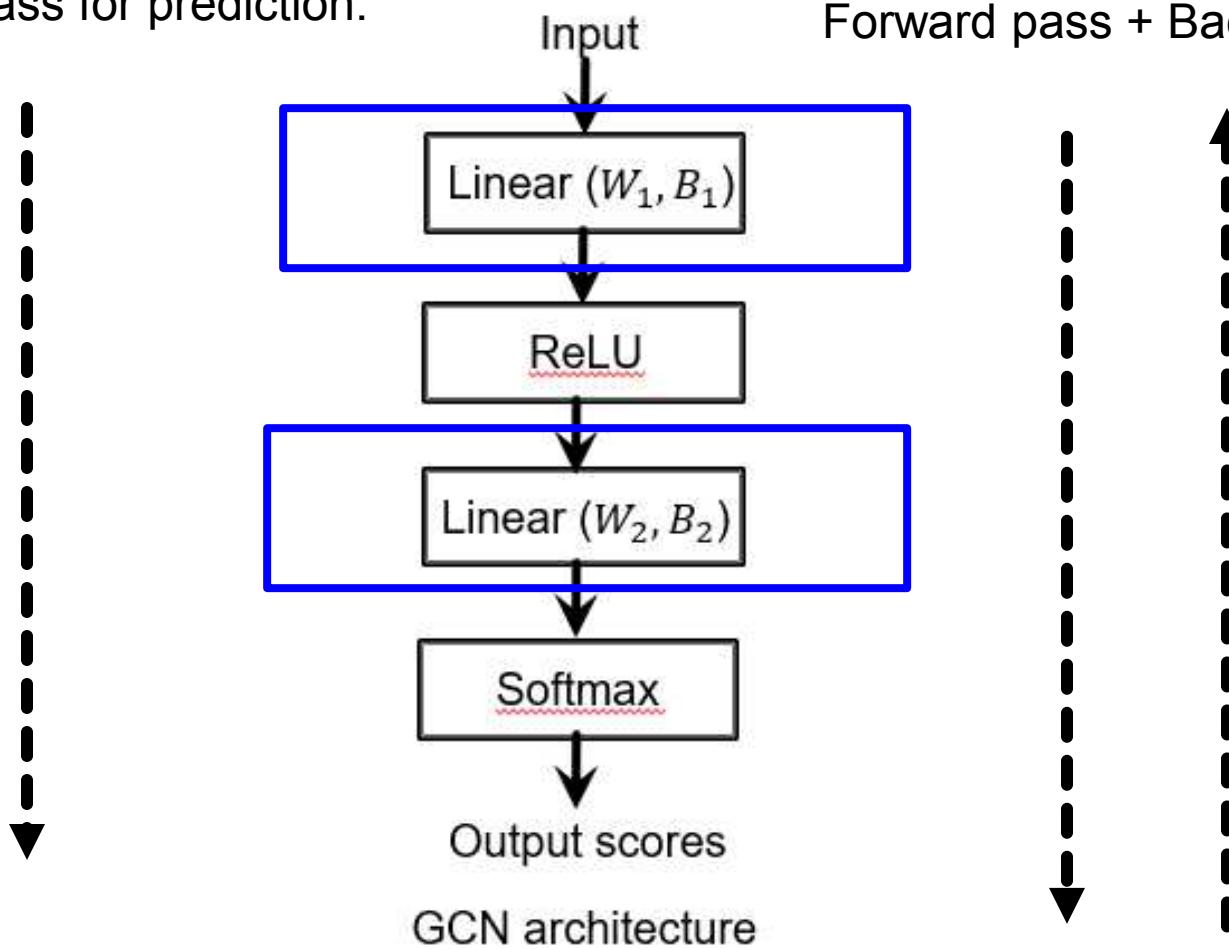


two variables

<https://thinkingandcomputing.com/posts/genetic-algorithms-neural-networks.html>

# GCN training

Forward pass for prediction:



- Backward pass:
  - Apply back-propagation to calculate gradients for all layers and update the parameters.
  - Back-propagation: Use of chain rule to calculate gradients for all layers

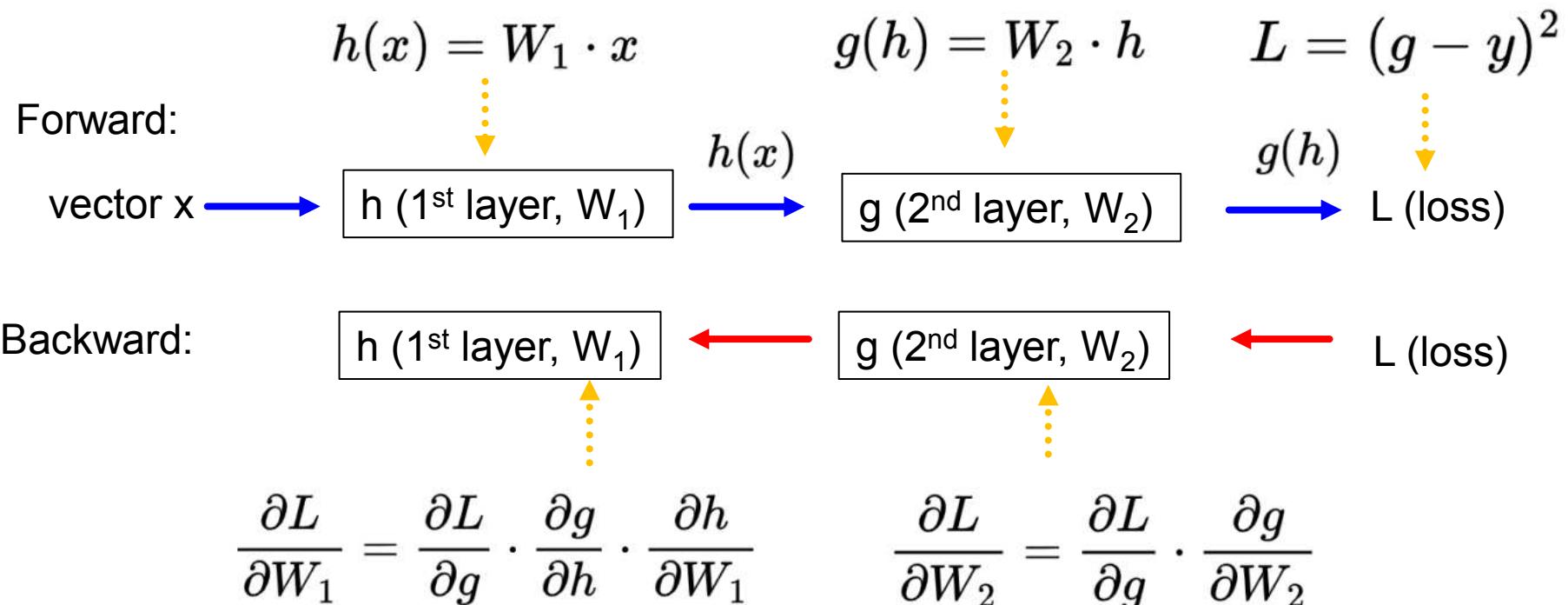
chain rule for gradient calculation for composition function:

$$\frac{d}{dt} f(g(t)) = f'(g(t))g'(t) = \frac{df}{dg} \cdot \frac{dg}{dt}$$

<https://pl.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/differentiating-vector-valued-functions/a/multivariable-chain-rule-simple-version>

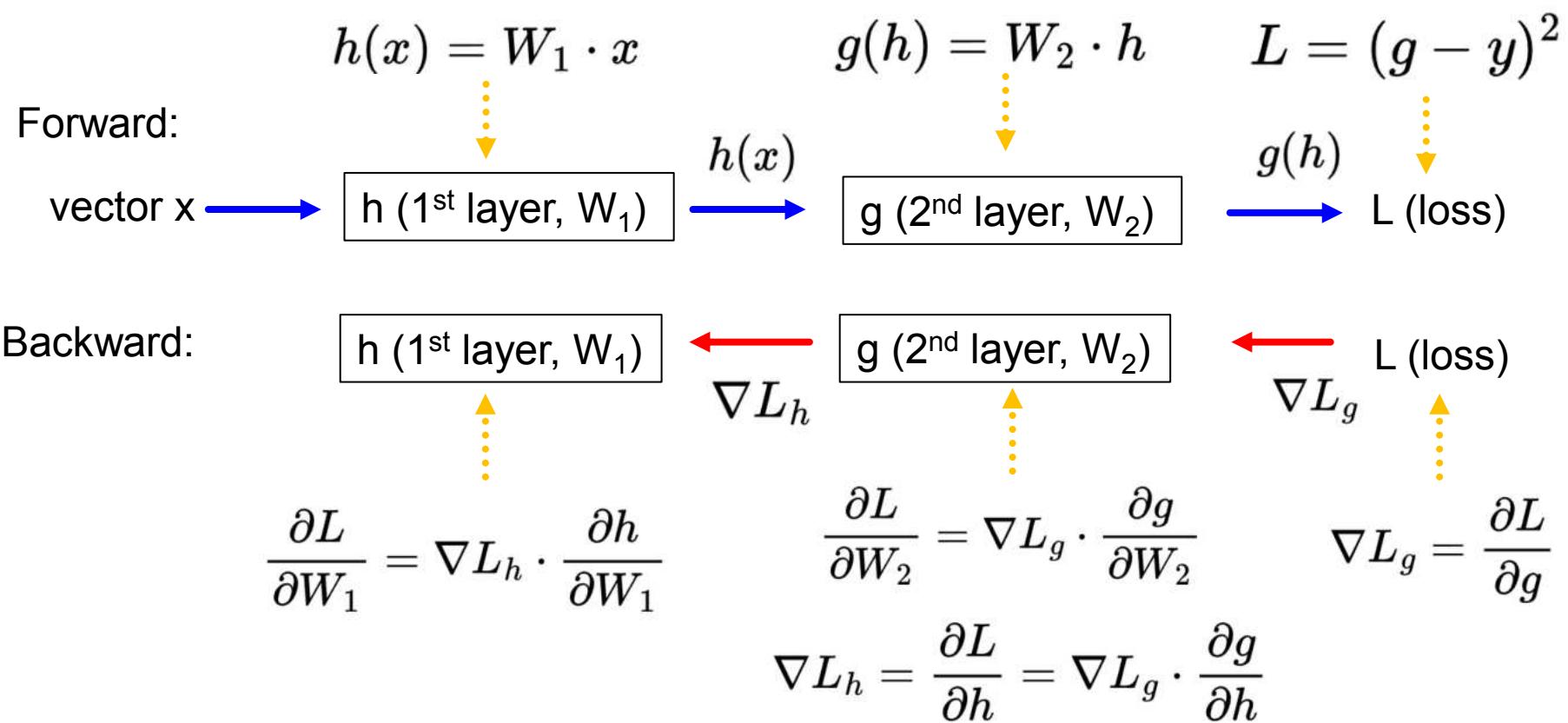
- Backward pass example on a simplified network (2 layers, no RELU)

$L$  is a loss function defined on  $f(x)$ , For example,  $L$  is an L2 loss for regression

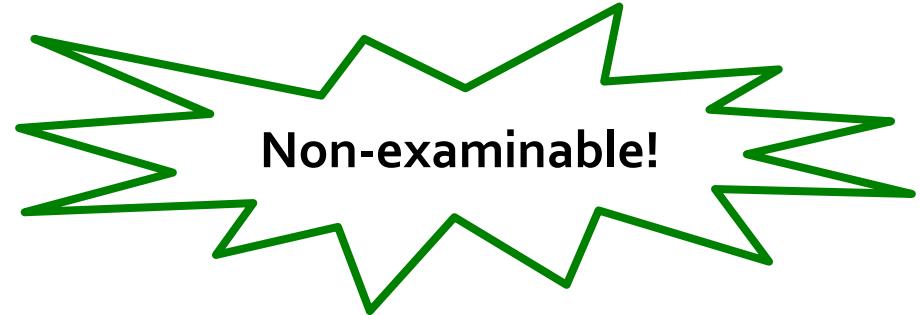


Using chain rule for gradient calculation

- Backward pass example on a simplified network (2 layers, no RELU)

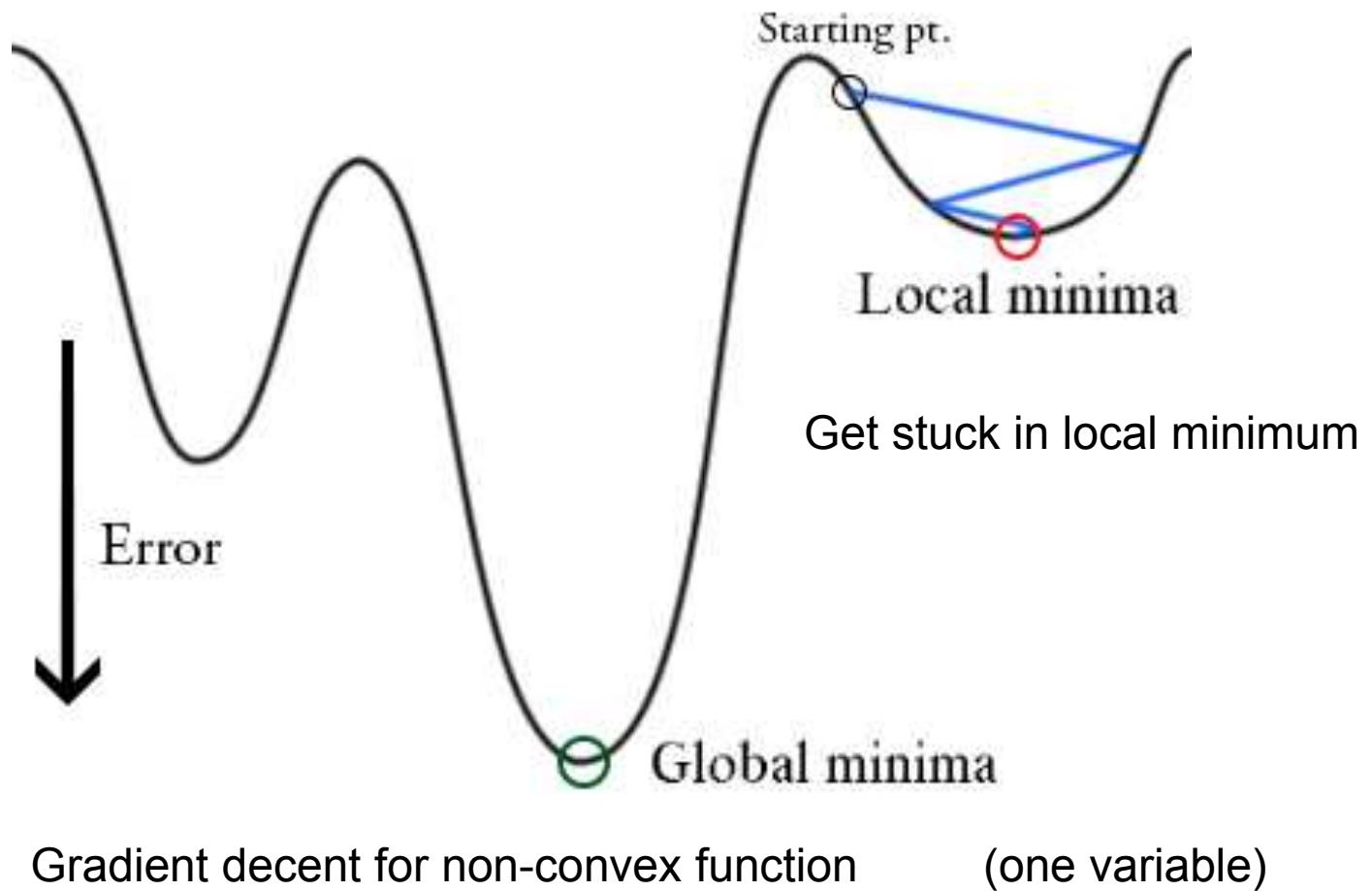


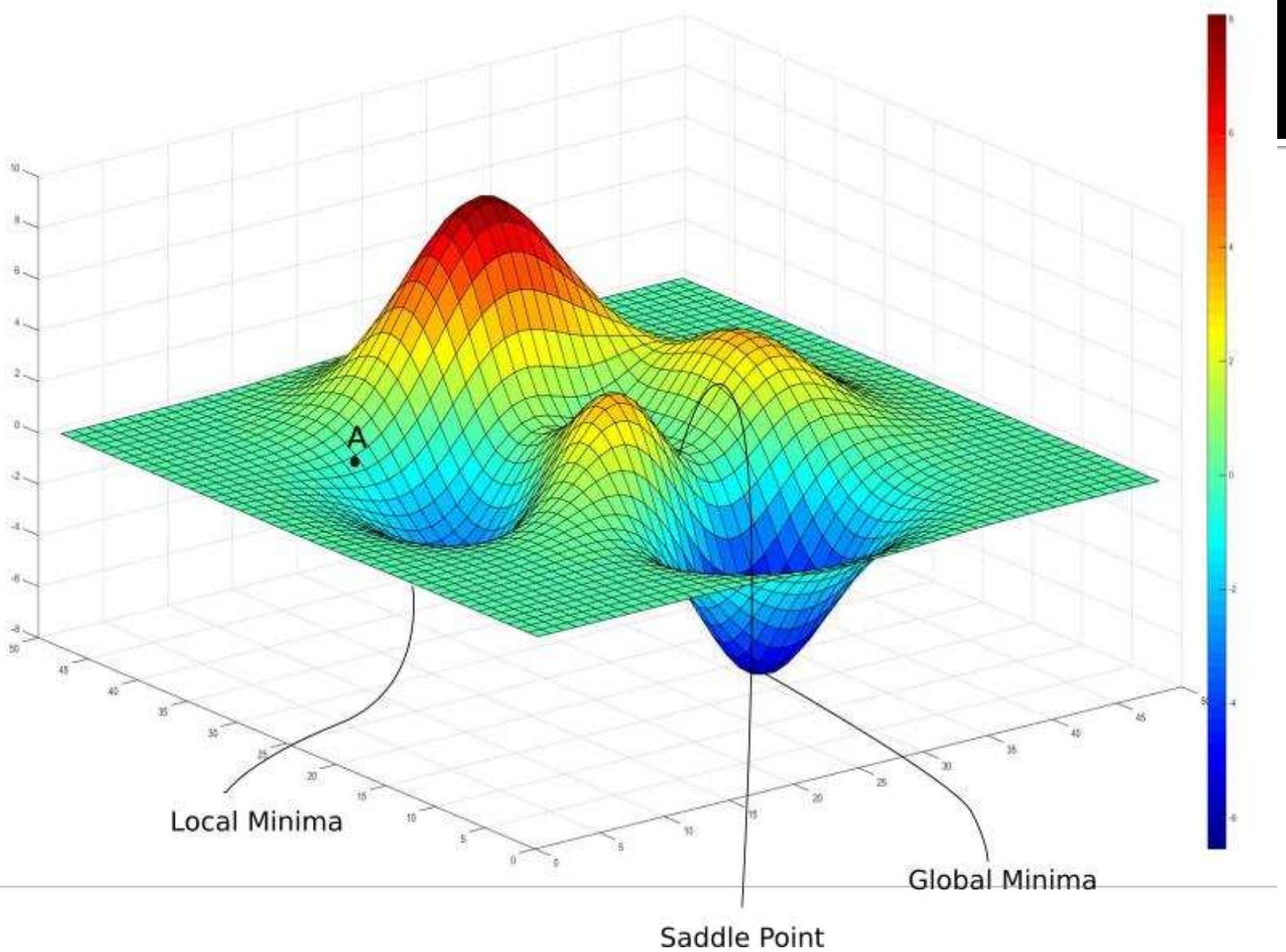
Backward pass for gradient calculation



- Further discussion
  - Limitation of gradient descent
  - Minibatch SGD
  - How to set learning rate?
  - Comparisons of activation functions
  - Weight decay (L2 regularization)
  - Momentum SGD

# Limitations of Gradient decent





Non-convex loss function (two variables)

<https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/>

# GCN training

- Minibatch Stochastic Gradient Decent (SGD)
  - Minibatch SGD
  - For each training iteration, randomly select a small set of data points (e.g., nodes) to perform gradient decent update.
  - Advantages:
    - Can be applied for large scale problems
      - Cannot calculate the gradients for all data points
    - The random sampling Introduces noises into the optimization, help to regularize the training and jump out of local minimum.

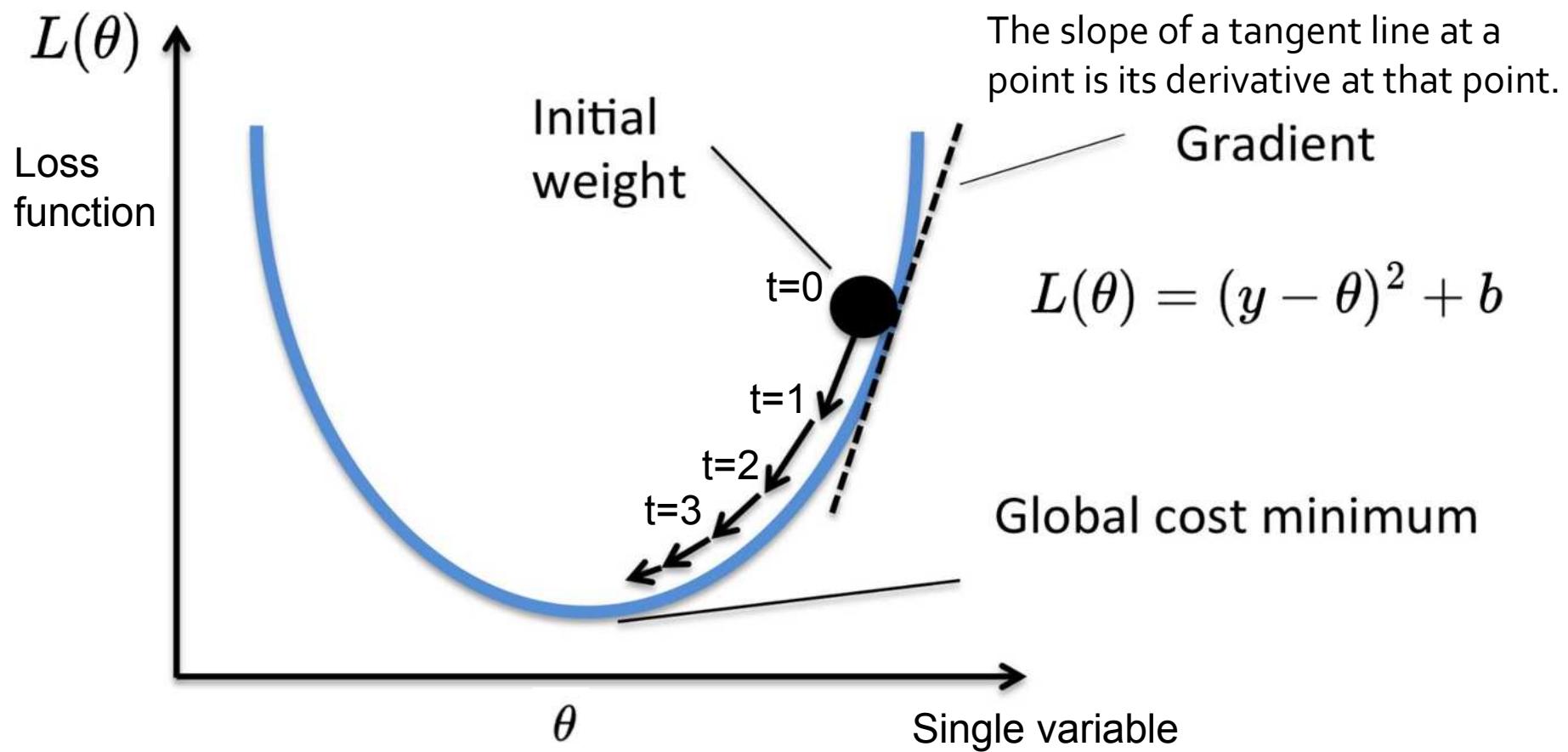
## ■ Minibatch Stochastic Gradient Decent (SGD)

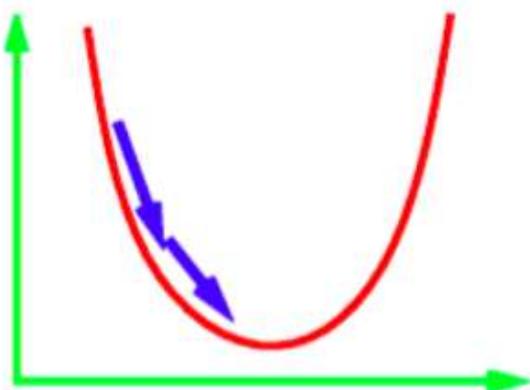
Some concepts for minibatch SGD:

- Batch size: the number of data points in a minibatch e.g., number of nodes for node classification task
- Iteration: 1 step of SGD on a minibatch
- Epoch: one full pass over the dataset (# iterations is equal to ratio of dataset size and batch size)

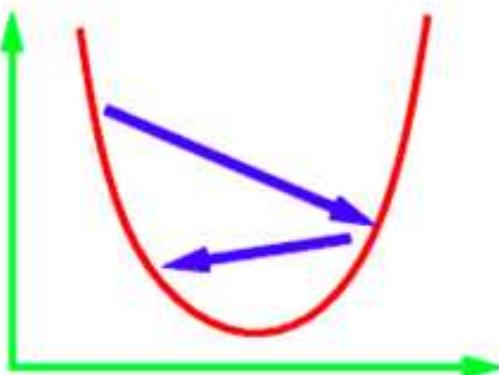
Two key factors for gradient descent:  
1 update direction (gradients),  
2 update step size (learning rate)

$$\theta(t+1) = \theta(t) + \Delta\theta(t);$$
$$\Delta\theta(t) = -\eta \nabla_{\theta} L;$$

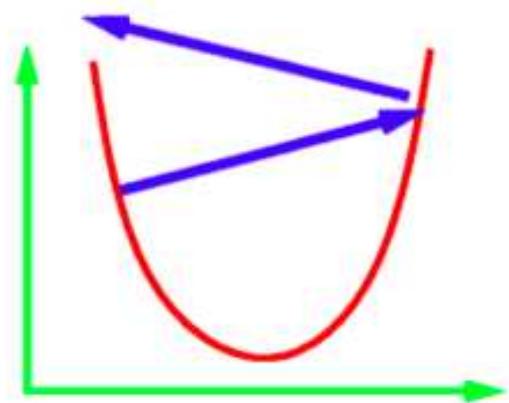




Small learning rate



Large learning rate

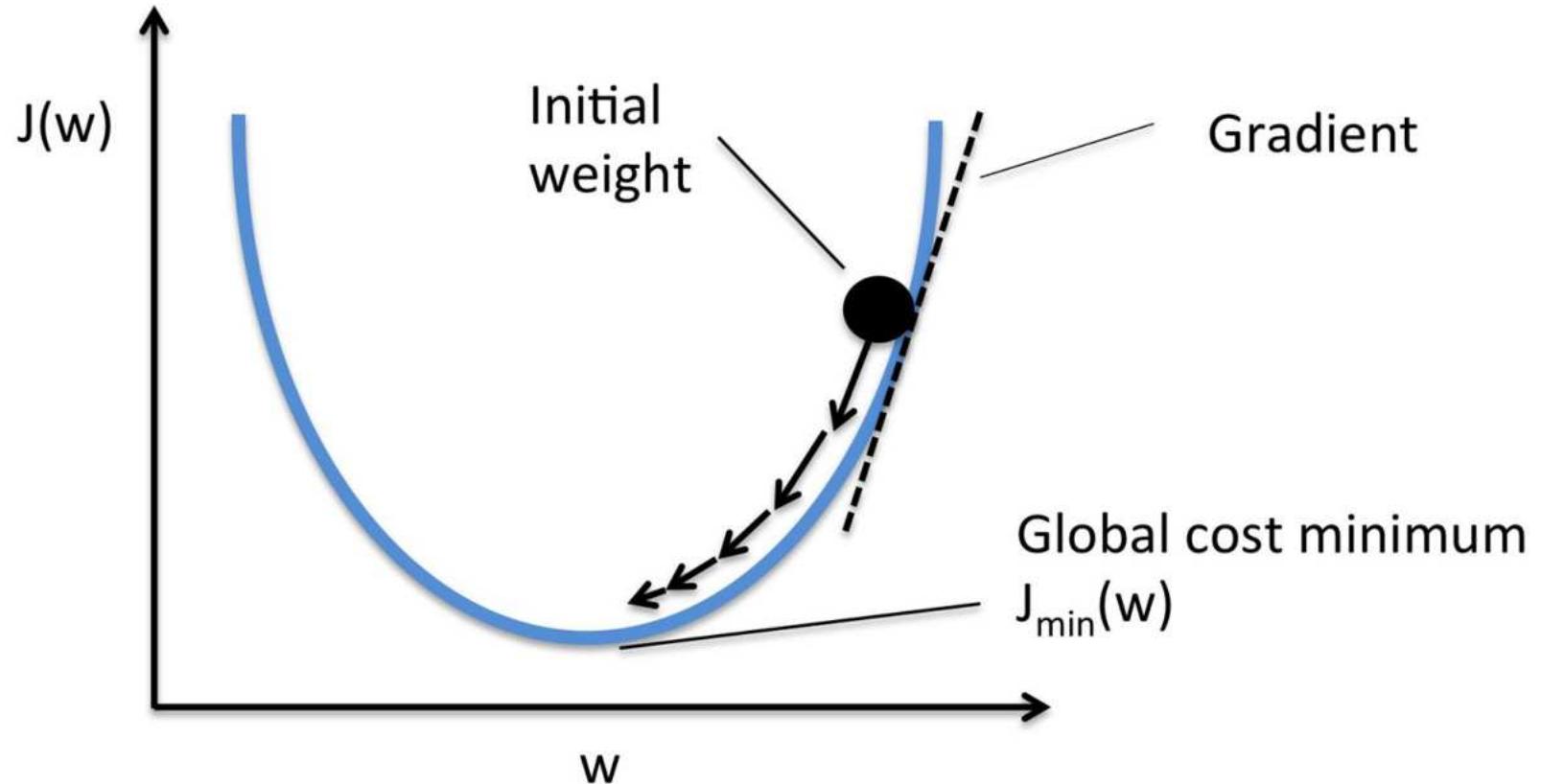


Large learning rate  
(diverged)

We need a good setting of learning rate

Learning rate needs to be gradually decreased

Learning rate (LR): the step size for gradient update



learning rate decay strategies:  
how to gradually decrease the learning rate

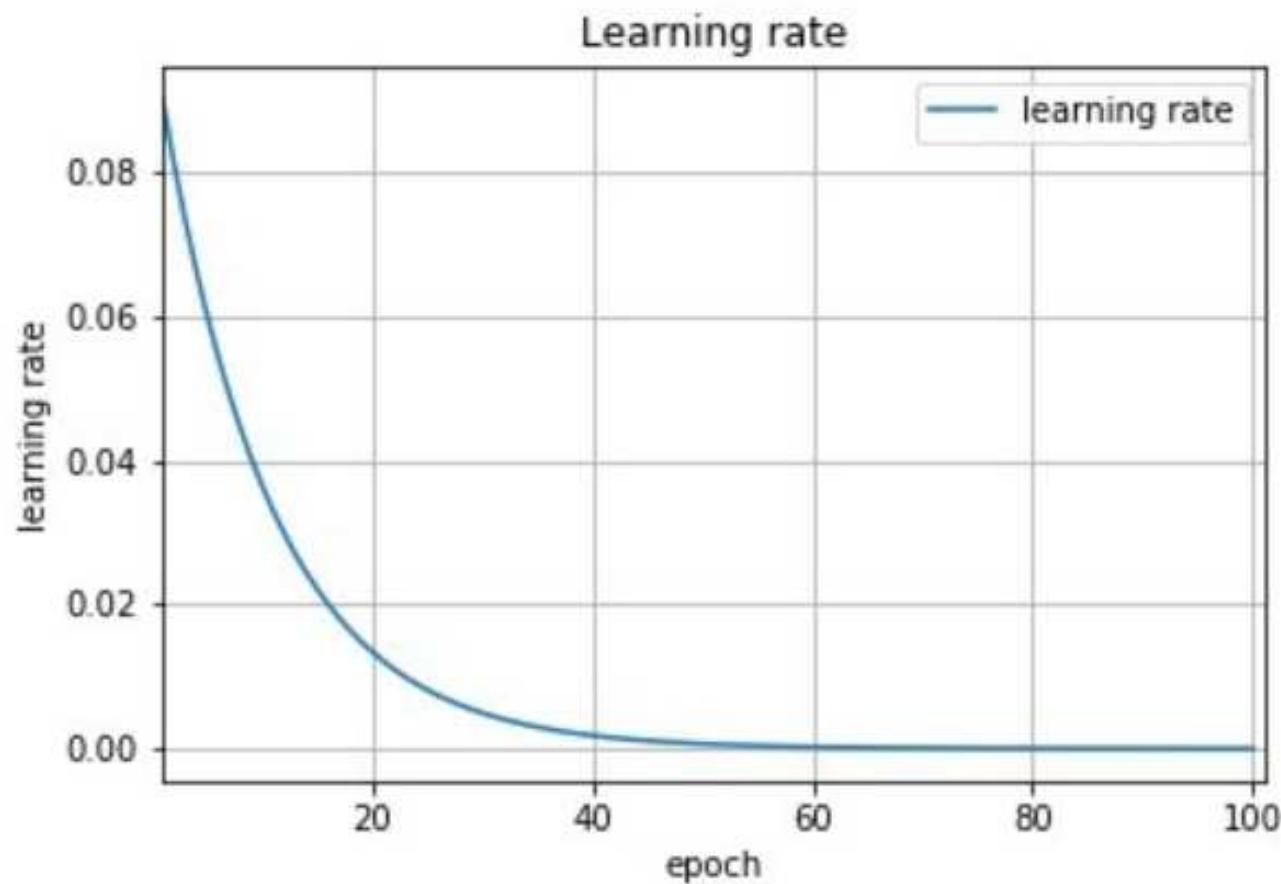


Fig 4b : Exponential Decay Schedule

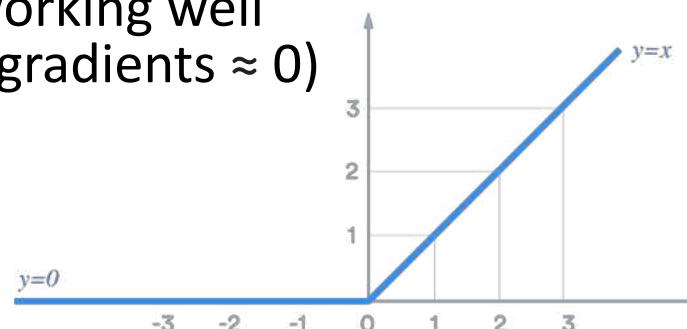
Learning rate decay scheduler: gradually reduce the learning rate

## ■ Revisit activation function

- Sigmoid and Hyperbolic tangent are not working well with gradient descent (saturated regions: gradients  $\approx 0$ )

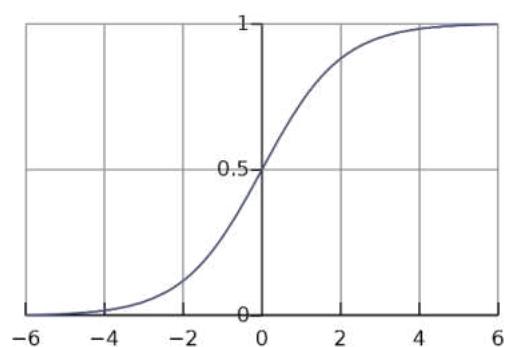
1. Rectified linear unit (ReLU)

$$ReLU(x) = \max(x, 0)$$



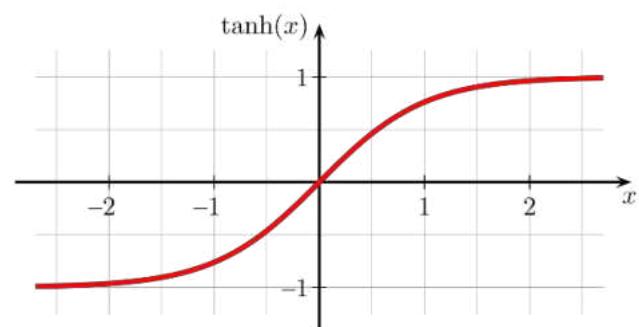
2. Sigmoid

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



3. Hyperbolic tangent

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$



## ■ Other details

1. In practice, we optimize this loss function:  
Cross-entropy loss + L2 regularization (weight decay)

$$\min_{\theta} L(Y, Z) = - \sum_i y_i \cdot \log z_i(\theta) + \frac{1}{2} \lambda \|\theta\|^2$$

L2 regularization: encourage small values for the solution

## 2. Use momentum mechanism for gradient update (Momentum SGD)

SGD for solution update:

$$\theta(t + 1) = \theta(t) + \Delta\theta(t)$$

Standard SGD:  $\Delta\theta(t) = -\eta\Delta_\theta L$

Momentum SGD:  $\Delta\theta(t) = -\eta\Delta_\theta L + \alpha\Delta\theta(t - 1)$

Momentum

## Momentum:

1. Encourage the update directions of two consecutive iterations to be similar (more consistent)
2. Noise Smoothing effect

Momentum averages gradients of recent iterations.

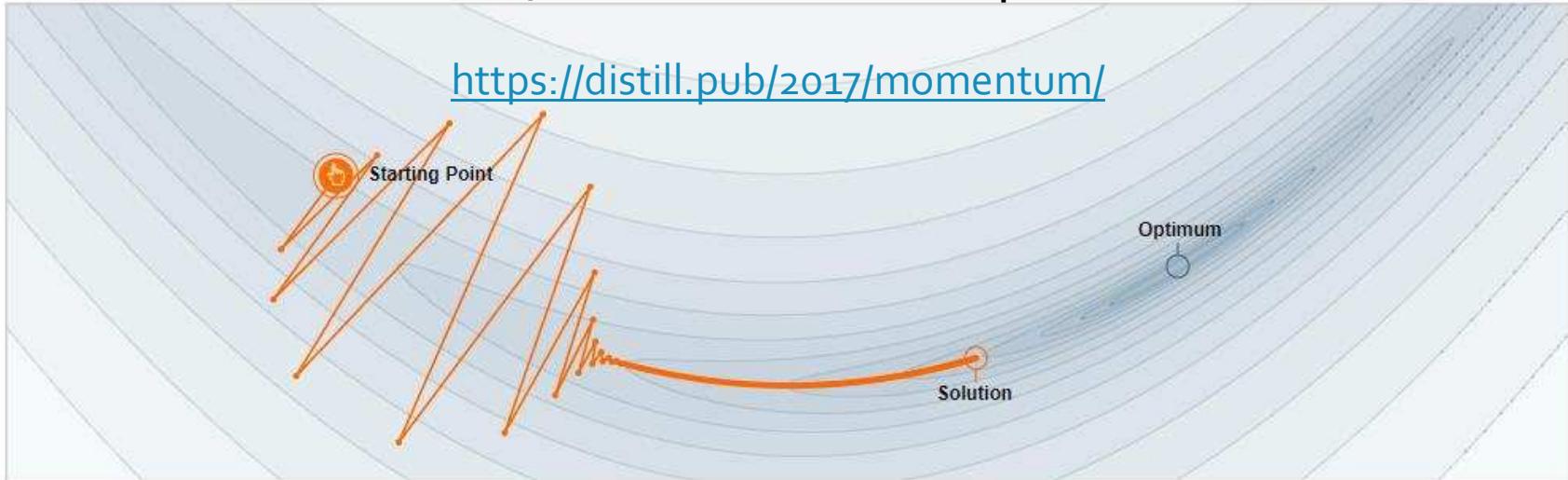
It is a running average of gradients that we use for each update step.

$$\text{Momentum SGD: } \Delta\theta(t) = -\eta\Delta_\theta L + \alpha\Delta\theta(t - 1)$$

Momentum

# With momentum, two consecutive updates are smooth.

<https://distill.pub/2017/momentum/>



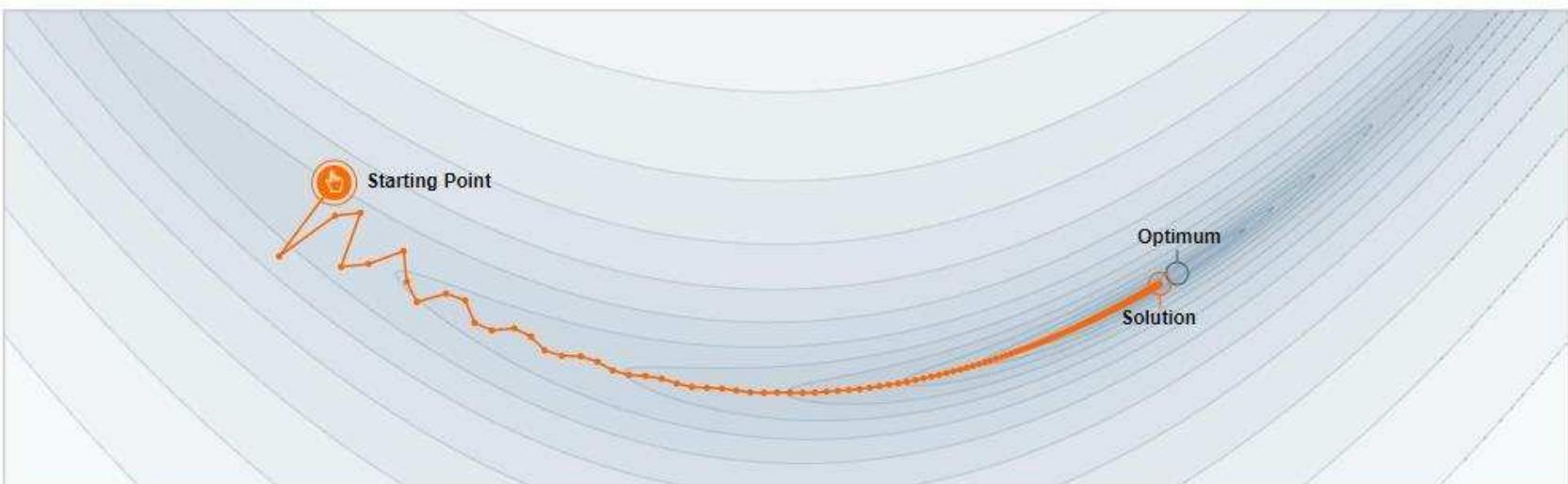
Step-size  $\alpha = 0.0041$



Momentum  $\beta = 0.0$



We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?



Step-size  $\alpha = 0.0041$



Momentum  $\beta = 0.73$



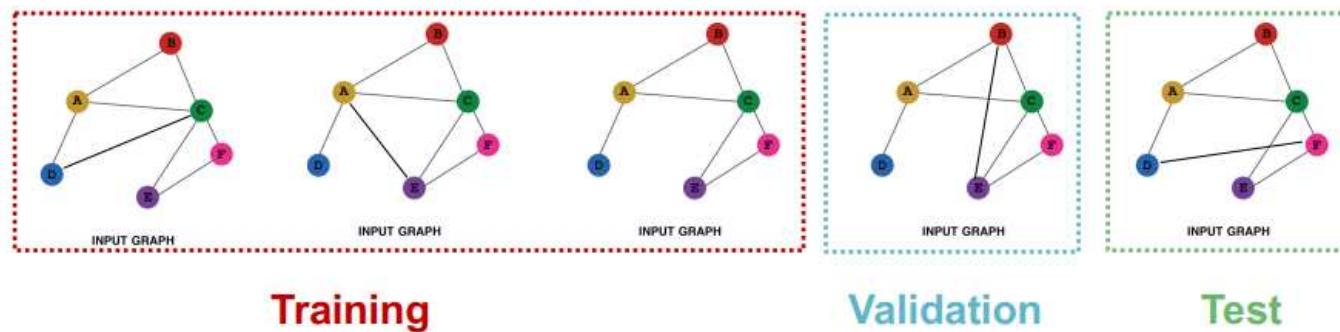
We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

## ■ Other details

- More advanced gradient methods  
(adaptive methods):
  - ADAM
    - <https://arxiv.org/abs/1412.6980>
  - LARS
    - <https://arxiv.org/abs/1708.03888>

# GCN training

- Determine hyper-parameters:
  - Hyper-parameters:
    - network architectures (e.g, #layers),
    - network training iterations,
    - learning rate, etc.
- Use a validation set to determine hyper-parameters.
  - Test the model with different hyper-parameters on the validation set, choose the setting with the best performance



- Further readings:
  - Deep learning basics
    - ConvNet, Batch normalization, residual/skip connections, optimizer (ADAM, momentum SGD), up/down-sampling, pooling, ...
    - <http://cs231n.stanford.edu/>
    - <https://atcold.github.io/pytorch-Deep-Learning/>
  - Transformers
    - Another popular network architecture that can be used on graphs
    - <http://web.stanford.edu/class/cs224w/>
  - Other advanced topics on GNN
    - <https://github.com/AntonioLonga/PytorchGeometricTutorial>
    - <http://web.stanford.edu/class/cs224w/>
    - <https://medium.com/stanford-cs224w>
    - <http://web.stanford.edu/class/cs224w/>
      - GraphSAGE, Graph attention network

- Online implementation
  - PyTorch Geometric
    - <https://www.pyg.org/>
    - PyG is a library built upon PyTorch to easily write and train Graph Neural Networks for a wide range of applications
    - <https://pytorch-geometric.readthedocs.io/en/latest/notes/resources.html>
    - <https://pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html>
  - Deep graph library
    - <https://github.com/dmlc/dgl>

## ■ Graph classification datasets:

- <https://pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html>
- <https://chrsmrrs.github.io/datasets/docs/datasets/>
- <https://paperswithcode.com/task/node-classification>

# Similarity Search - LSH

Lin Guosheng  
School of Computer Science and Engineering  
Nanyang Technological University

# Outline

- Nearest Neighbour Search (NNS) Problems
- Local Sensitive Hashing (LSH) – Random Projection
- Product Quantization (PQ)
- Inverted File Index (**non-examinable!!**)

# Similarity Search Applications

## Visually similar results



Q

A grid of nine images showing various pairs of sneakers, likely generated by a visual search algorithm. Each image includes a caption and a user profile icon.

- Top row:
  - Nike (by kasia fashion)
  - V (by Gabriela Sg)
  - "zizi repetto" (by Bonnie & Jane Look)
- Middle row:
  - kris van assche sneakers (by Natalia Bilecka lust)
  - This COS top from the men's section ticks all the right... (by Carlo Bevelander Low Top)
  - stan smith outfits - Buscar con Google (by Denys Finch-Hatton Sneakers)
- Bottom row:
  - Ginnies Ladies (by Ginnies Ladies)
  - 3 (by 3)

<https://web.stanford.edu/class/cs246/>

# How does it work?

Query image



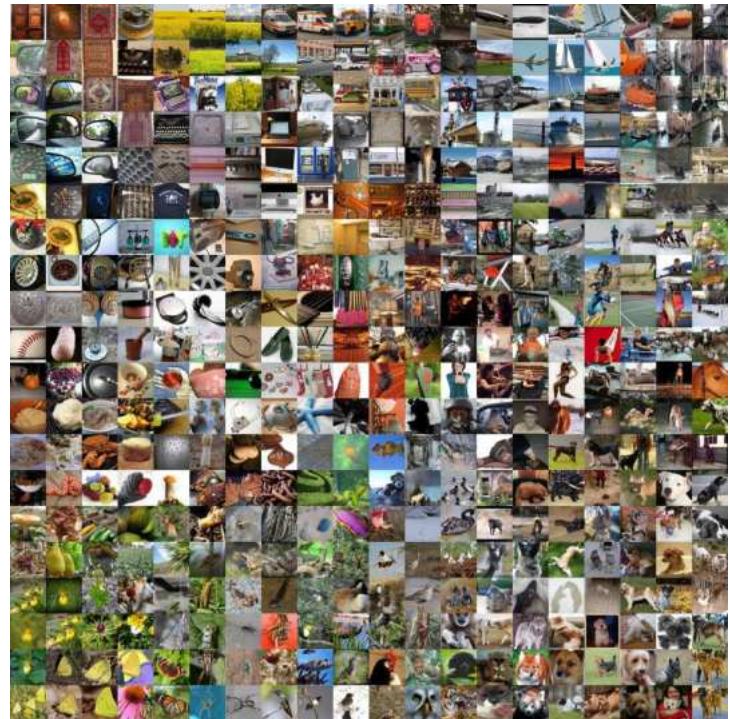
Search similar images  
from the database

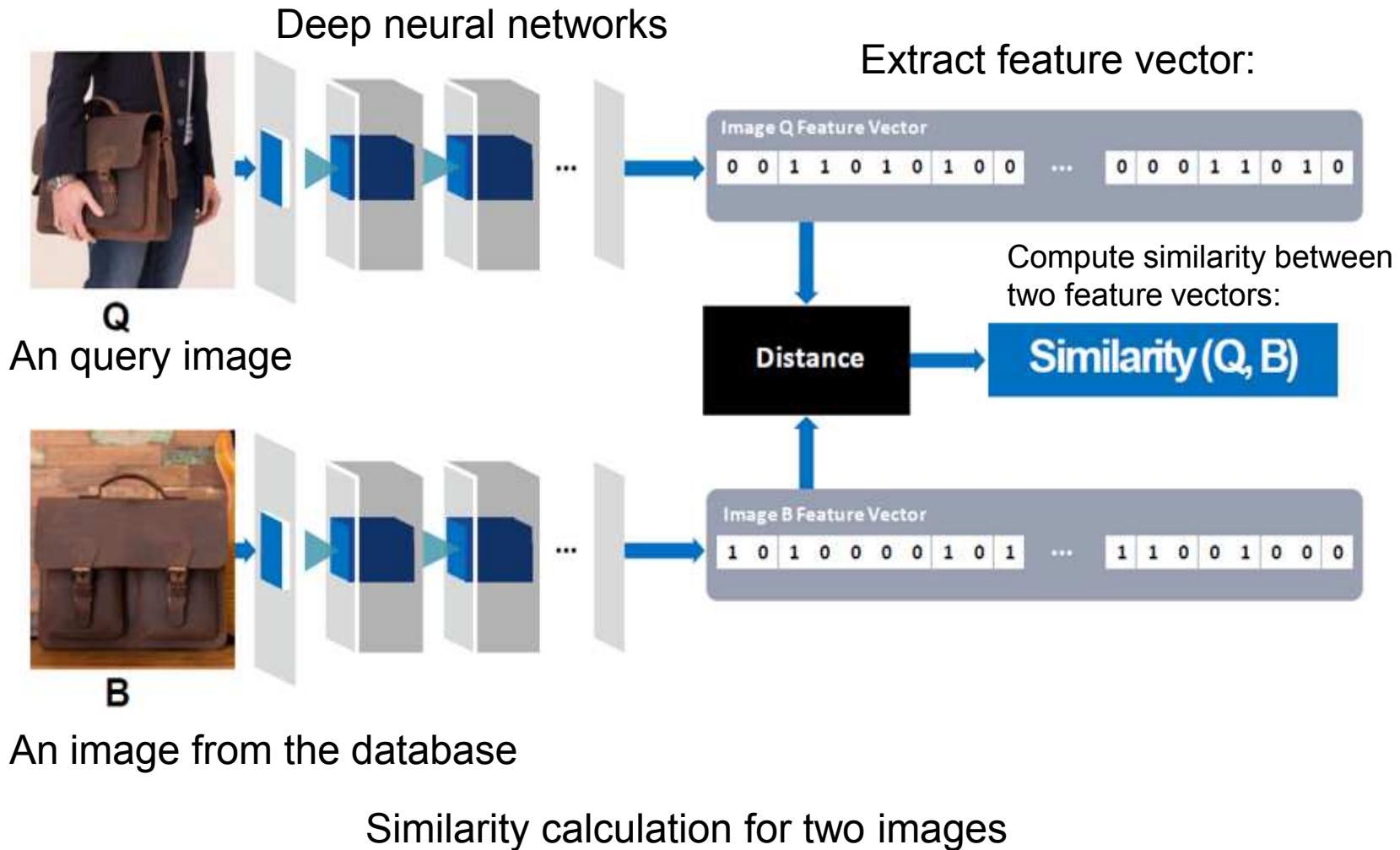


A simple method (linear search):

1. Compute similarity between the query image and all images in the database.
2. Return the top-k similar images in the database.
3. Computational expensive for large scale database (e.g., 1 billion images in the database)

Image database  
(e.g, billions of images)





# Nearest Neighbour Search (NNS)

- k-nearest neighbour search
  - Given a query, identify the top-k nearest neighbours (top-k most similar items) in the database
  - The result is based on a certain type of similarity metric. (e.g., L2 distance)
- Similarity metrics
  - Euclidean distance (L2 distance)
  - Cosine similarity
  - Hamming distance (for binary data)
  - Manhattan distance (L1 distance)
  - Inner product
  - ...

# Similarity

## 1. Euclidean distance (L<sub>2</sub> distance) :

$$d_{L_2}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

## 2. Inner product (dot product):

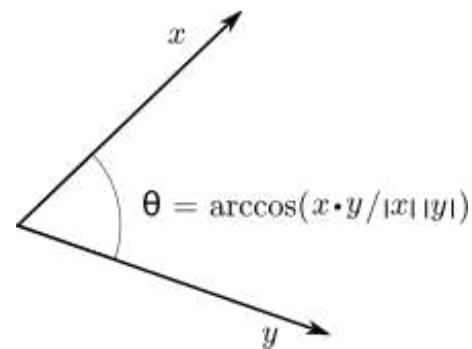
The dot product of two vectors  $\mathbf{a} = [a_1, a_2, \dots, a_n]$  and  $\mathbf{b} = [b_1, b_2, \dots, b_n]$  is defined as:<sup>[3]</sup>

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Geometric definition:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta,$$

where  $\theta$  is the angle between  $\mathbf{a}$  and  $\mathbf{b}$ .



# Similarity

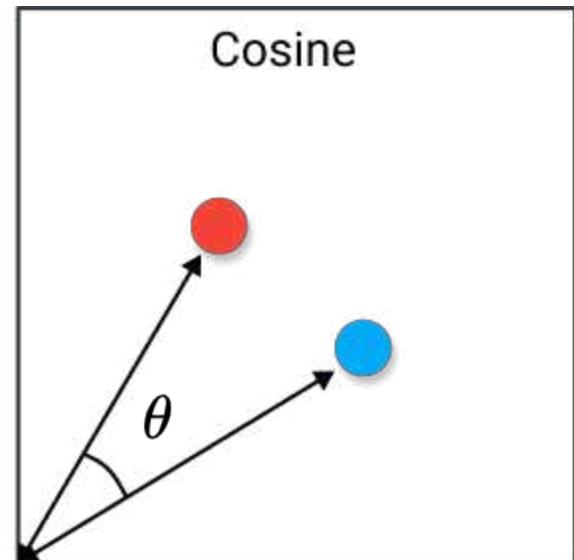
## 3. Cosine similarity

Given two vectors of attributes,  $A$  and  $B$ , the cosine similarity,  $\cos(\theta)$ , is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

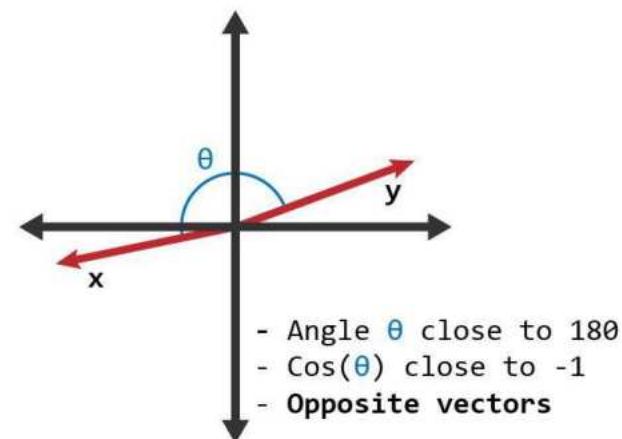
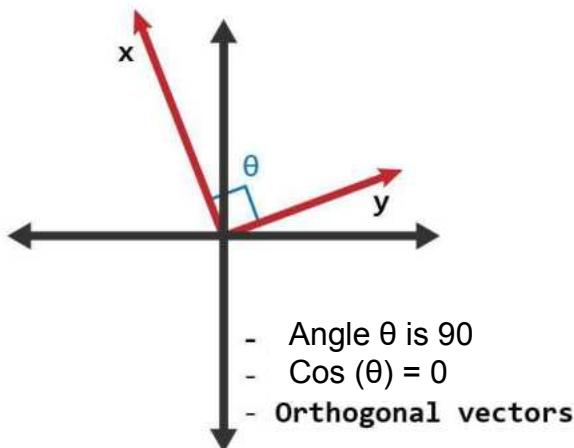
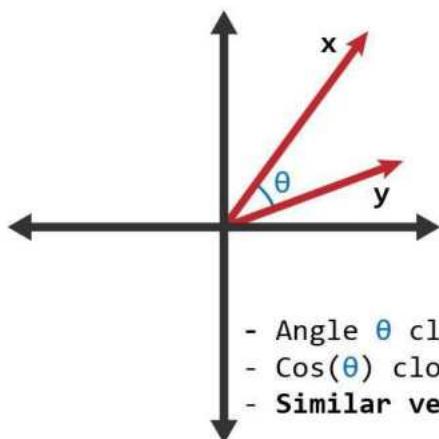
where  $A_i$  and  $B_i$  are components of vector  $A$  and  $B$  respectively.

Cosine Distance = 1 - Cosine Similarity



# Similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

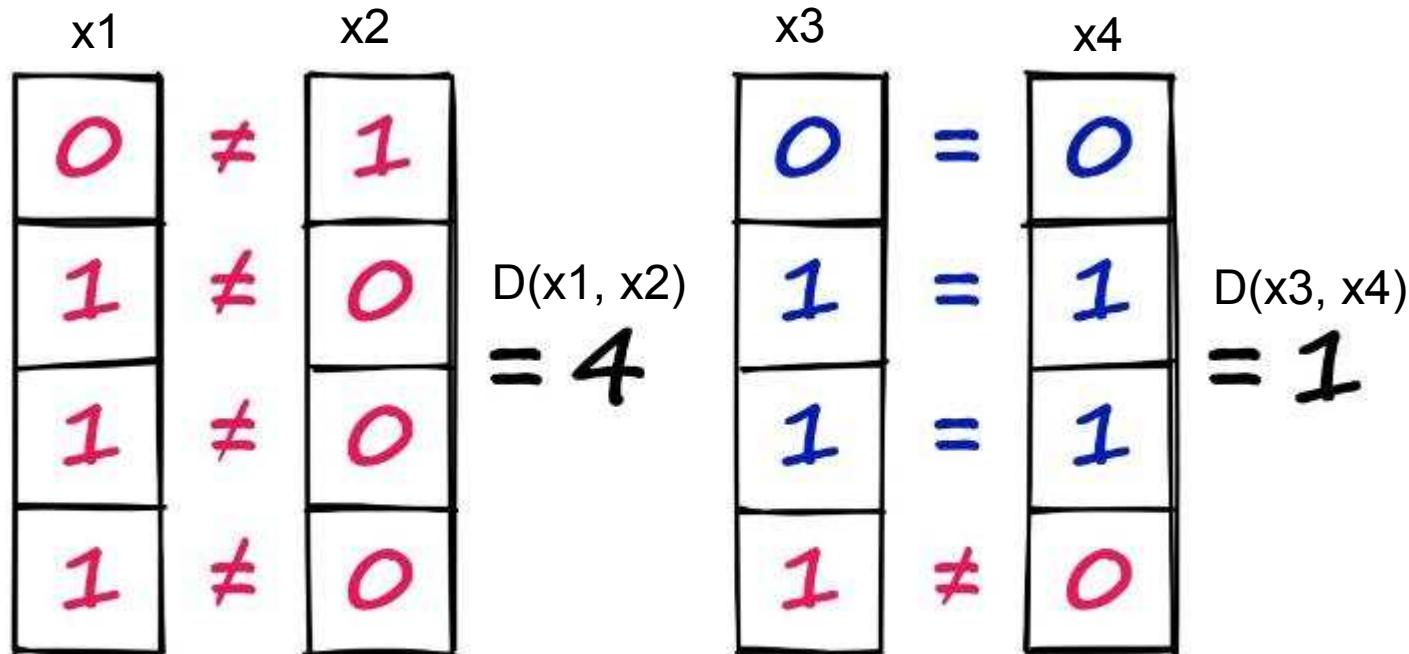


<https://www.learndatasci.com/glossary/cosine-similarity/>

# Similarity

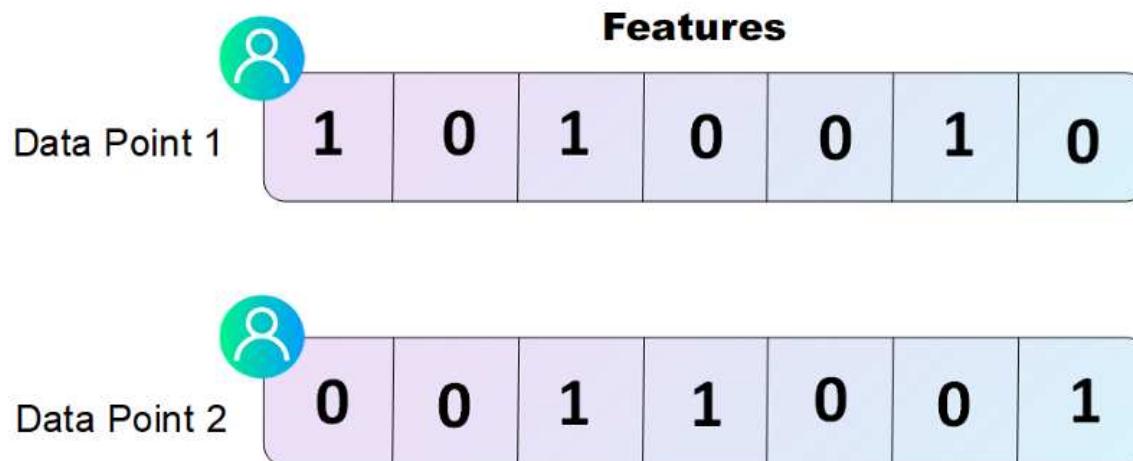
## 4. Hamming distance:

- The number of positions with different values.
- Binary data: the number of different bits in two binary vectors.
- A good option for similarity calculation of binary data



# Similarity

Hamming distance on binary vectors:



$$\text{Hamming Distance} = 1 + 0 + 0 + 1 + 0 + 1 + 1 = 4$$

<https://aigents.co/data-science-blog/publication/distance-metrics-for-machine-learning>

# Nearest Neighbour Search (NNS)

## ■ Nearest Neighbour Search (NNS)

- Linear search
  - Compute the distances between the query and each sample in the database.
  - Also called linear scan, exhaustive search
  - Produce extract NNS search results
  - Slow
- Approximate Nearest Neighbour Search (ANN)
  - Return approximate search results
    - Local Sensitive Hashing (LSH)
    - Product Quantization (PQ)
    - Inverted File Index (IVF)
    - ....
  - A more comprehensive list:
    - <https://github.com/erikbern/ann-benchmarks>

# Linear search

- Linear search (exhaustive search)
  - Given a query vector, compute the similarity with all samples in the database, return the top-k most similar samples

Example:

Question: given the query sample and the samples in the database below, find the top 2 nearest neighbours in the database.

Query sample:

A	[ 0, -1]
---	----------

Database samples (5):

B	[ -2, 0]
C	[ 1, 2]
D	[ 2, 1]
E	[ 1, -1]
F	[ -1, 2]

# Linear search

Example:

Question: given the query sample and the samples in the database below, find the top 2 nearest neighbours in the database.

Query sample:

A	[ 0, -1]
---	----------

Database samples (5):

B	[ -2, 0]
C	[ 1, 2]
D	[ 2, 1]
E	[ 1, -1]
F	[ -1, 2]

Squared L2 distance:

$$\|x - y\|^2 = \sum_{i=1}^d (x_i - y_i)^2$$

	B (-2, 0)	C (1, 2)	D (2, 1)	E (1, -1)	F (-1, 2)
A (0, -1)	5	10	8	1	10

Search result: the top 2 similar items are E and B

We need 5 distance calculations (we have 5 items in the database)

# Linear search

- Linear search (exhaustive search)
  - Given a query vector, compute the similarity with all samples in the database, return the top-k most similar samples
    - For one query, search complexity:  $O(n)$ ,  
 $n$ : the number of samples in the database
  - Not efficient for large databases and high dimensional data
    - Imagine a dataset containing millions or even billions of samples
    - Generally, one similarity calculation for high dimensional data is expensive

# Outline

- Nearest Neighbour Search (NNS) Problems
- **Local Sensitive Hashing (LSH) – Random Projection**
- Product Quantization (PQ)
- Inverted File Index (**non-examinable!!**)

# LSH

- Local Sensitive Hashing – Random Projection
  - LSH hash functions
  - LSH + linear search for top-k retrieval
  - LSH + hash tables for top-k retrieval
- Extended discussion
  - Geometry view of LSH (**non-examinable!**)

# Locality Sensitive Hashing (LSH)

- LSH is a family of hashing methods
  - LSH-Random Projection
    - Random hyperplane hashing functions
    - Preserve cosine similarity
  - Many other LSH methods
    - use different types of hashing function
    - preserve different types of similarity
    - E.g., LSH-MinHashing

Many materials are from:

<https://www.pinecone.io/learn/locality-sensitive-hashing-random-projection/>

# LSH-Random Projection

- LSH Random projection method
  - Generate K hashing functions.
    - Randomly generate K vectors
      - (using standard gaussian distribution)
    - Each vector constructs one linear hash function.
  - One hashing function transform the input vector into one binary value (1-bit)
  - K hashing functions transform the input vector into a K-bit binary vector (hash vector, binary code).

One hashing function:

$$h(x) = \begin{cases} 1 & : w^T x > 0 \\ 0 & : w^T x \leq 0 \end{cases}$$

$x$  : the input vector

$w$  : the parameters for the hashing function  
(the randomly generated vector)

# LSH-Random Projection

## Example (LSH + linear search):

Question: given the query sample and the samples in the database below, find the top 2 nearest neighbours in the database. Use LSH-random projection.

Query sample:

A	[ 0, -1]
---	----------

Database samples (5):

B	[ -2, 0]
C	[ 1, 2]
D	[ 2, 1]
E	[ 1, -1]
F	[ -1, 2]

4 hashing functions are given:

$$w_1 = [ -1 \ 1];$$

$$w_2 = [ -1 \ 0];$$

$$w_3 = [ \ 0 \ 1];$$

$$w_4 = [ \ 1 \ -1];$$

# LSH-Random Projection

Solution:

Step1: convert the database items and the query item into binary vectors.

Query sample:

A	[ 0, -1]
---	----------

Database samples (5):

B	[ -2, 0]
C	[ 1, 2]
D	[ 2, 1]
E	[ 1, -1]
F	[ -1, 2]

4 hashing functions are given:

$$w_1 = [-1 \ 1]^T;$$

$$w_2 = [-1 \ 0]^T;$$

$$w_3 = [ 0 \ 1]^T;$$

$$w_4 = [ 1 \ -1]^T;$$

$$h_1 : w_1^T x_A = -1 * 0 + 1 * (-1) = -1 \leq 0 \rightarrow h_1(x_A) = 0$$

$$h_2 : w_2^T x_A = -1 * 0 + 0 * (-1) = 0 \leq 0 \rightarrow h_2(x_A) = 0$$

$$h_3 : w_3^T x_A = 0 * 0 + 1 * (-1) = -1 \leq 0 \rightarrow h_3(x_A) = 0$$

$$h_4 : w_4^T x_A = 1 * 0 + (-1) * (-1) = 1 > 0 \rightarrow h_4(x_A) = 1$$

The binary vector for the query sample A is [ 0 0 0 1 ]

# LSH-Random Projection

4 hashing functions are given:

$$w_1 = [-1 \ 1]^T;$$

$$w_2 = [-1 \ 0]^T;$$

$$w_3 = [0 \ 1]^T;$$

$$w_4 = [1 \ -1]^T;$$

Database samples (5):

B	[-2, 0]
C	[ 1, 2]
D	[ 2, 1]
E	[ 1, -1]
F	[-1, 2]

$$h_1 : w_1^T x_B = -1 * (-2) + 1 * 0 = 2 > 0 \rightarrow h_1(x_B) = 1$$

$$h_2 : w_2^T x_B = -1 * (-2) + 0 * 0 = 2 > 0 \rightarrow h_2(x_B) = 1$$

$$h_3 : w_3^T x_B = 0 * (-2) + 1 * 0 = 0 \leq 0 \rightarrow h_3(x_B) = 0$$

$$h_4 : w_4^T x_B = 1 * (-2) + (-1) * 0 = -2 \leq 0 \rightarrow h_4(x_B) = 0$$

The binary vector for the database sample B is [ 1 1 0 0]

# LSH-Random Projection

4 hashing functions are given:

$$w_1 = [-1 \ 1]^T;$$

$$w_2 = [-1 \ 0]^T;$$

$$w_3 = [0 \ 1]^T;$$

$$w_4 = [1 \ -1]^T;$$

$$w_1^T x_C = -1 * 1 + 1 * 2 = 1 > 0 \rightarrow h_1(x_C) = 1$$

$$w_2^T x_C = -1 * 1 + 0 * 2 = -1 \leq 0 \rightarrow h_2(x_C) = 0$$

$$w_3^T x_C = 0 * 1 + 1 * 2 = 2 > 0 \rightarrow h_3(x_C) = 1$$

$$w_4^T x_C = 1 * 1 + (-1) * 2 = -1 \leq 0 \rightarrow h_4(x_C) = 0$$

Database samples (5):

B	[-2, 0]
C	[1, 2]
D	[2, 1]
E	[1, -1]
F	[-1, 2]

The binary vector for the database sample C is [1 0 1 0]

$$w_1^T x_D = -1 * 2 + 1 * 1 = -1 \leq 0 \rightarrow h_1(x_D) = 0$$

$$w_2^T x_D = -1 * 2 + 0 * 1 = -2 \leq 0 \rightarrow h_2(x_D) = 0$$

$$w_3^T x_D = 0 * 2 + 1 * 1 = 1 > 0 \rightarrow h_3(x_D) = 1$$

$$w_4^T x_D = 1 * 2 + (-1) * 1 = 1 > 0 \rightarrow h_4(x_D) = 1$$

The binary vector for the database sample D is [0 0 1 1]

# LSH-Random Projection

4 hashing functions are given:

$$w_1 = [-1 \ 1]^T;$$

$$w_2 = [-1 \ 0]^T;$$

$$w_3 = [0 \ 1]^T;$$

$$w_4 = [1 \ -1]^T;$$

$$w_1^T x_E = -1 * 1 + 1 * (-1) = -2 \leq 0 \rightarrow h_1(x_E) = 0$$

$$w_2^T x_E = -1 * 1 + 0 * (-1) = -1 \leq 0 \rightarrow h_2(x_E) = 0$$

$$w_3^T x_E = 0 * 1 + 1 * (-1) = -1 \leq 0 \rightarrow h_3(x_E) = 0$$

$$w_4^T x_E = 1 * 1 + (-1) * (-1) = 2 > 0 \rightarrow h_4(x_E) = 1$$

Database samples (5):

B	[-2, 0]
C	[1, 2]
D	[2, 1]
E	[1, -1]
F	[-1, 2]

The binary vector for the database sample E is [0 0 0 1]

$$w_1^T x_F = -1 * (-1) + 1 * 2 = 3 > 0 \rightarrow h_1(x_F) = 1$$

$$w_2^T x_F = -1 * (-1) + 0 * 2 = 1 > 0 \rightarrow h_2(x_F) = 1$$

$$w_3^T x_F = 0 * (-1) + 1 * 2 = 2 > 0 \rightarrow h_3(x_F) = 1$$

$$w_4^T x_F = 1 * (-1) + (-1) * 2 = -3 \leq 0 \rightarrow h_4(x_F) = 0$$

The binary vector for the database sample F is [1 1 1 0]

Query sample:

A	[ 0, -1]
---	----------

Query sample:

A	[ 0 0 0 1]
---	------------

Database samples (5):

B	[ -2, 0 ]
C	[ 1, 2 ]
D	[ 2, 1 ]
E	[ 1, -1 ]
F	[ -1, 2 ]

After hashing functions



Database samples (5):

B	[ 1 1 0 0 ]
C	[ 1 0 1 0 ]
D	[ 0 0 1 1 ]
E	[ 0 0 0 1 ]
F	[ 1 1 1 0 ]

Query sample:

A	[ 0 0 0 1 ]
---	-------------

Step2: perform linear search on binary vectors using hamming distance  $D_H$

Database samples (5):

B	[ 1 1 0 0 ]
C	[ 1 0 1 0 ]
D	[ 0 0 1 1 ]
E	[ 0 0 0 1 ]
F	[ 1 1 1 0 ]

$$D_H(A, B) = 3 \text{ (number of mismatched bits)}$$

$$D_H(A, C) = 3$$

$$D_H(A, D) = 1$$

$$D_H(A, E) = 0$$

$$D_H(A, F) = 4$$

Search result:

Given the query A,  
the top 2 nearest neighbours are: E and D

# Similarity (recap)

## 4. Hamming distance:

- Generally, counts the number of positions where two symbols are different.
- For binary data: the number of bits that are different in two binary vectors.

x1	x2	x3	x4
0	≠ 1		
1	≠ 0		
1	≠ 0		
1	≠ 0		

D(x1, x2) = 4

D(x3, x4) = 1

Hamming distance == number of mismatches

- L2 distance based linear search VS LSH + linear search

L2 distances

	B (-2, 0)	C (1, 2)	D (2, 1)	E (1, -1)	F (-1, 2)
A (0, -1)	5	10	8	1	10

Search result: the top 2 similar items are E and B

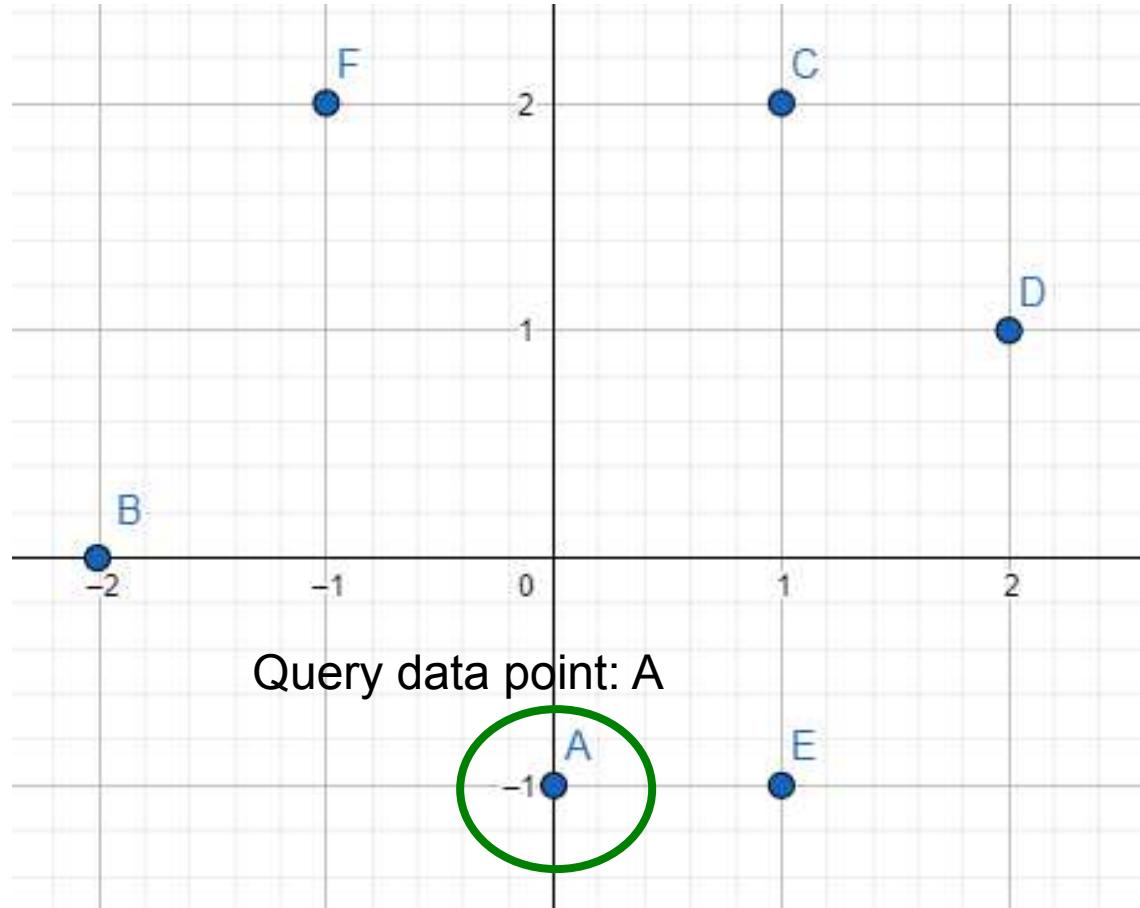
LSH and Hamming distances

	B	C	D	E	F
A	3	3	1	0	4

Search result: the top 2 similar items are E and D

Data points:

A	(0, -1)
B	(-2, 0)
C	(1, 2)
D	(2, 1)
E	(1, -1)
F	(-1, 2)



LSH and Hamming distances

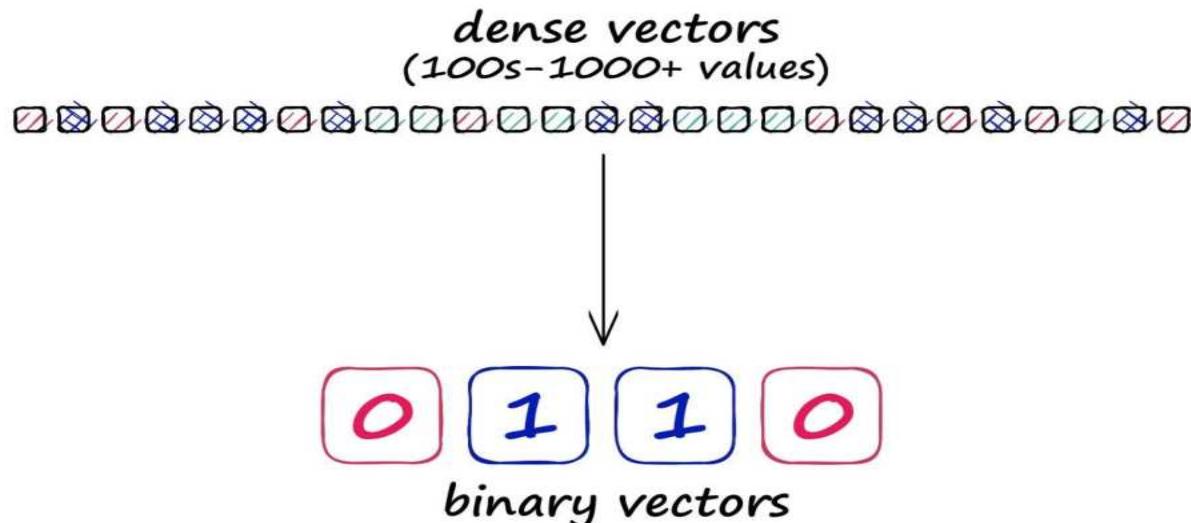
	B	C	D	E	F
A	3	3	1	0	4

L2 distances on the original features

	B	C	D	E	F
A	5	10	8	1	10

# LSH for similarity search

- With LSH, we can
  - generate low-dimensional binary features from high-dimensional input
  - Preserve the similarity after binary mapping.
    - If  $x$  is similar to  $y$  in the original space, after binary mapping, their binary vectors are also similar in the binary space.



# Discussion

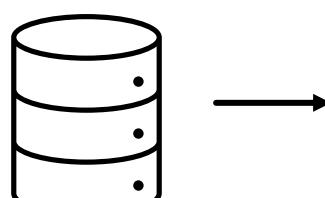
- Why LSH + linear search is more efficient than L2 distance + linear search?

# Discussion

- Why LSH + linear search is more efficient than L2 distance + linear search?
  - We can use the efficient hamming distance on the binary vectors.
  - Hamming distance calculation is very efficient
    - XOR operation on the bits (counting mismatched bits)
    - Much more efficient than L2 distance on float values
    - Especially useful for high-dimensional data

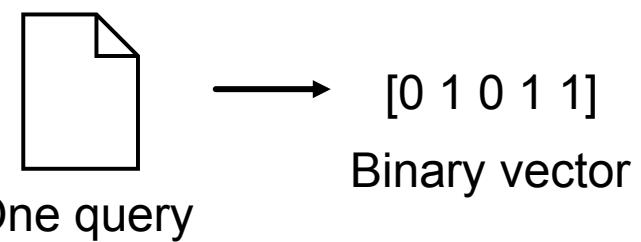
- LSH requires extra calculation for binary transform
  - Database binary transform with LSH is done offline.
    - The whole database are transformed to binary vectors for only one time (offline pre-processing)
    - The database binary vectors will be used for all queries.
    - For one query, we only need to transform the query input vector to binary vector.

Step1: convert the database  
into binary vectors (offline step)



Binary vectors

Step 2: process query requests  
For each query, we only need to  
convert the query to a binary vector  
(online processing)



One query

# Discussion

- Benefits of using binary vectors
  - Binary vectors are efficient for storage, transmission and similarity calculation (Hamming distance).
    - A 128-dimensional float-value vector:  
 $128 \times 4 \text{ byte} = 512 \text{ bytes}$ 
      - 1 float value requires 4 bytes (32 bits)
    - A 128 bit binary vector = 16 bytes
      - 1 byte = 8 bits,

# Discussion

- More hashing functions will increase the accuracy on top-k nearest neighbour search.
  - More hashing functions lead to longer binary vectors (we have more bits)
  - With more bits, the top-k ranking results of LSH will be more similar to the cosine distance or L2 distance based top-k results.

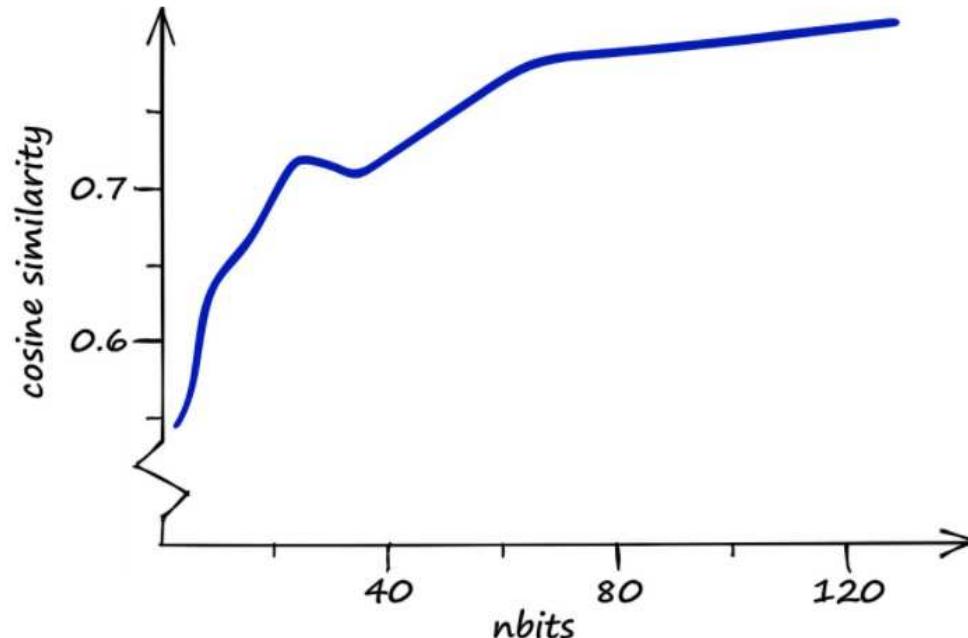
# Analysis on the number of bits

Number of bits (**nbits**): the number of binary values in the binary vector.  
It is equal to the number of hash functions.

With more bits, the top-k retrieval results become more accurate.

Y axis: average cosine similarity of top-100 results.

It indicates the accuracy of the retrieved results



As we increase vector resolution with **nbits**, our results will become more precise — here, we can see that a larger **nbits** value results in higher cosine similarity in our results.

Sift1M dataset, 1M samples, 128 dimensions  
Facebook Faiss implementation, top 100 retrieved

<https://www.pinecone.io/learn/locality-sensitive-hashing-random-projection/>

# More hashing functions

- We can modify the LSH example to generate more hashing functions and observe the difference.

Query sample:

A	[ 0, -1]
---	----------

Hashing functions:

$$w_1 = [-1 \ 1]^T;$$

$$w_2 = [-1 \ 0]^T;$$

$$w_3 = [ 0 \ 1]^T;$$

$$w_4 = [ 1 \ -1]^T;$$

Add more hash functions:

$$w_5 = [ 1 \ 0]^T;$$

$$w_6 = [-1 \ -1]^T;$$

Database samples (5):

B	[ -2, 0]
C	[ 1, 2]
D	[ 2, 1]
E	[ 1, -1]
F	[ -1, 2]

We will explore this question in the tutorial class

# LSH for similarity search

- LSH for similarity search

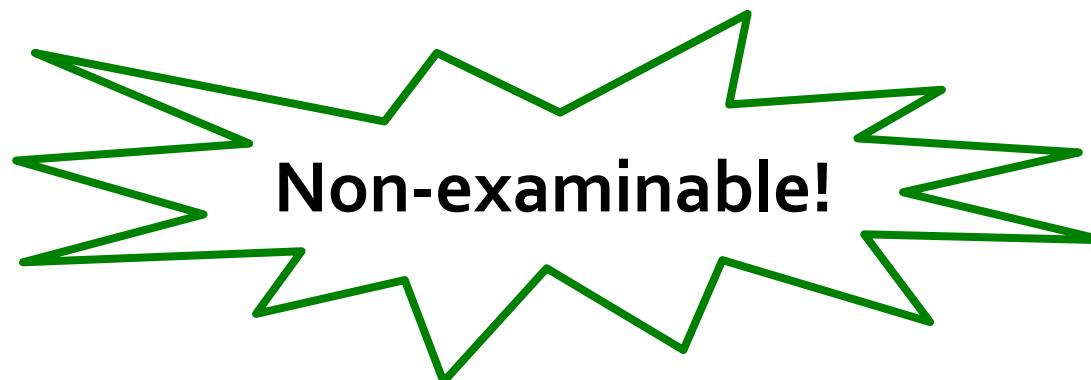
Two different approaches

- 1. Using linear search on the binary vectors
  - As shown in the previous example
- 2. Using hash tables for similarity search

# Using hash tables for LSH

## ■ 2. Similarity search using hash tables

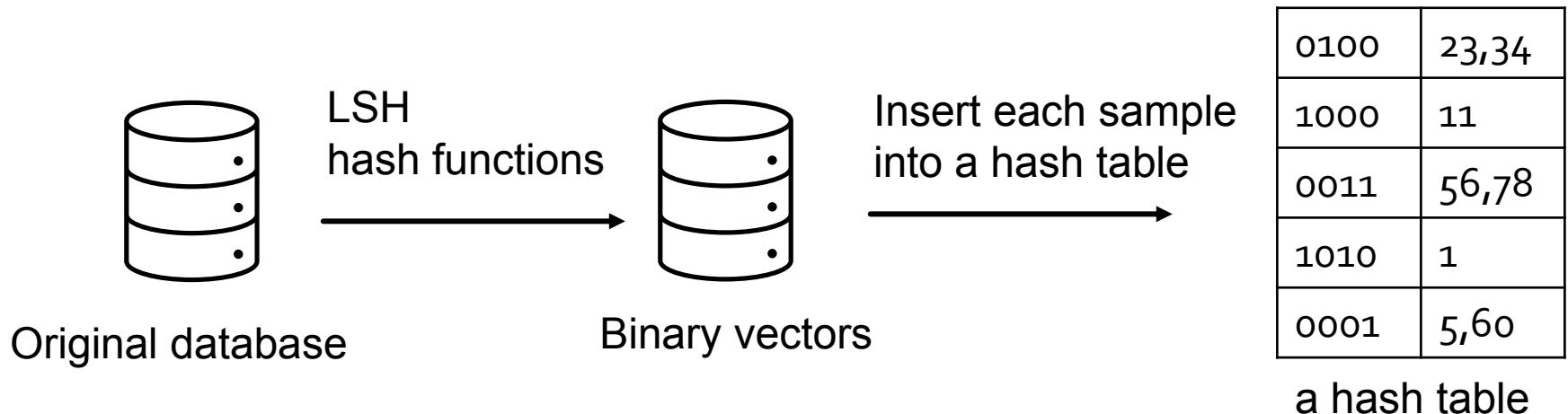
- Method1: Using one hash table
  - **(non-examinable)**
- Method2: Using multiple hash tables
  - **(non-examinable)**



# Using hash tables for LSH

## Method 1: using one hash table

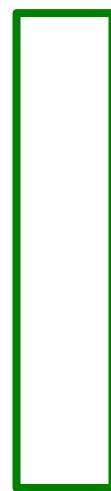
- Step 1: build the hash table for the database
  - Generate the hash vector ( $h$ ) for each data sample in the database and insert  $\langle h, \text{sample\_idx} \rangle$  into a hash table.



# Create hash table

Hash functions

One sample  
in the database  
(e.g., **index:9**)



Hash vector  
(binary vector)  
e.g., 101111

Insert the sample into the hash table

One bucket

Hash table  
(hash buckets)

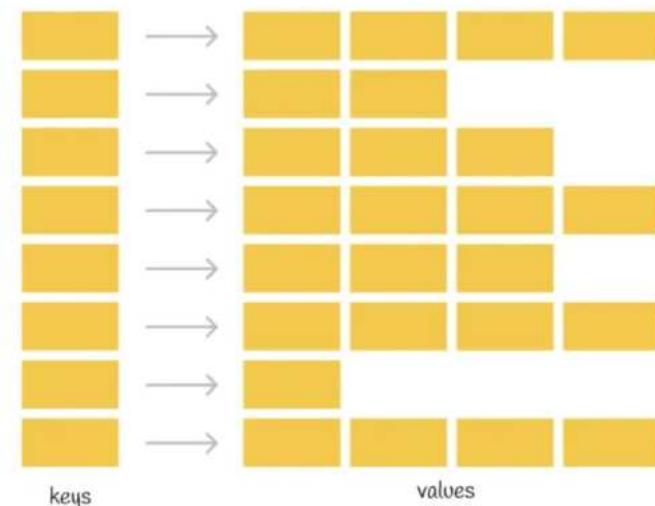
Key (binary vector)	Value (sample index list)
001111	(1, 3, 100)
111100	(13)
100110	()
101111	(14, 18, 9)
101010	(15, 19, 29, 39)

A hash table can be implemented by a data struct for storing key-value pairs (an associative array), e.g., the dictionary data type in Python

## ■ Hash table, bucket and hash vector

- A hash table is a key-value data structure
- can be implemented by an associative array for storing key-value entries
  - e.g., the dictionary data structure in Python for storing key-value items
- The entries in a hash table are called buckets

A hash table aims to group similar data into the same bucket: similar data are likely to have the same key (binary vector)



- Each entry is a key-value pair:  $\langle h, \text{sample\_idx\_list} \rangle$ 
  - $h$  is a binary vector (or called hash vector, hash code, binary code)
  - $\text{sample\_idx\_list}$  is a list of the indexes of the data samples in the database
- If two binary vectors (hash vectors) are the same, they will be mapped to the same bucket in the hash table.

Two examples in the database have the same hash vector

indicates

Two examples are mapped to the same bucket in a hash table

## Method 1: exhaustive search of all buckets

- Step 1: Build the hash table for the database (**done**)
- Step 2: process a query
  - Compute the hash vector for the query sample.
    - E.g input query  $x \rightarrow [101010]$
  - Construct a candidate set for the retrieval
    - Compare the query hash vector with all buckets in the hash table using Hamming distance
    - Rank all buckets by hamming distance
    - Go through the top-ranked buckets to build a candidate set (the number of candidate items  $\geq k$  items for top- $k$  retrieval).
  - Return top- $k$  items from the candidate set using L2 distance.

# Example for using a single hash table

Task: top-5 retrieval, given the query x

Key (binary vector) (buckets)	Value (sample index list)
001111	(1, 3, 100)
111100	(13)
100110	()
101111	(14, 18, 9)
101010	(15, 19, 29, 39)

Generate the binary vector of  
the query x

Query x: [101011]

Hash table (database)  
(hash buckets)

# Example for using a single hash table

**Task: top-5 retrieval, given the query x**

Key (binary vector) (buckets)	Value (sample index list)
001111	(1, 3, 100)
111100	(13)
100110	()
101111	(14, 18, 9)
101010	(15, 19, 29, 39)

Hash table (database)  
(hash buckets)

- 1) Calculate the Hamming distance between the query binary vector with all buckets in the table, and rank the buckets

Query: [101011]



buckets	Hamming Distance	Rank
001111	2	3
111100	4	5
100110	3	4
101111	1	1
101010	1	2

# Example for using a single hash table

Task: top-5 retrieval, given the query x

Key (binary vector) (buckets)	Value (sample index list)
001111	(1, 3, 100)
111100	(13)
100110	()
101111	(14, 18, 9)
101010	(15, 19, 29, 39)

Hash table (database)  
(hash buckets)

Query: [101011]

buckets	Hamming Distance	Rank
001111	2	3
111100	4	5
100110	3	4
101111	1	1
101010	1	2

2) Generate a candidate set  
for top-5 retrieval  
(should have  $\geq 5$  items)



Candidate buckets

101111	(14, 18, 9)
101010	(15, 19, 29, 39)

# Example for using a single hash table

**Task: top-5 retrieval, given the query x**

Candidate buckets (candidate set) for the query

101111	(14, 18, 9)
101010	(15, 19, 29, 39)



3) Use L2 distance to get the top 5 items from the candidate set

# Using hash tables for LSH

- Efficiency of using LSH + hash table:  
(compared to linear search)
  - 1. The number of distance calculations is reduced

As samples are grouped into buckets, the number of buckets are smaller than the number of samples in the database
  - 2. Hamming distance calculation is very efficient

# Using hash tables for LSH

- More hashing functions
  - --> Generate longer binary vectors (more bits)
  - --> More buckets in the hash table
    - #bits: 2: maximum #buckets:  $2^2=4$
    - #bits: 4, maximum #buckets:  $2^4=16$
    - #bits: 8, maximum #buckets:  $2^8=256$
    - #bits: 16, maximum #buckets:  $2^{16}=65536$

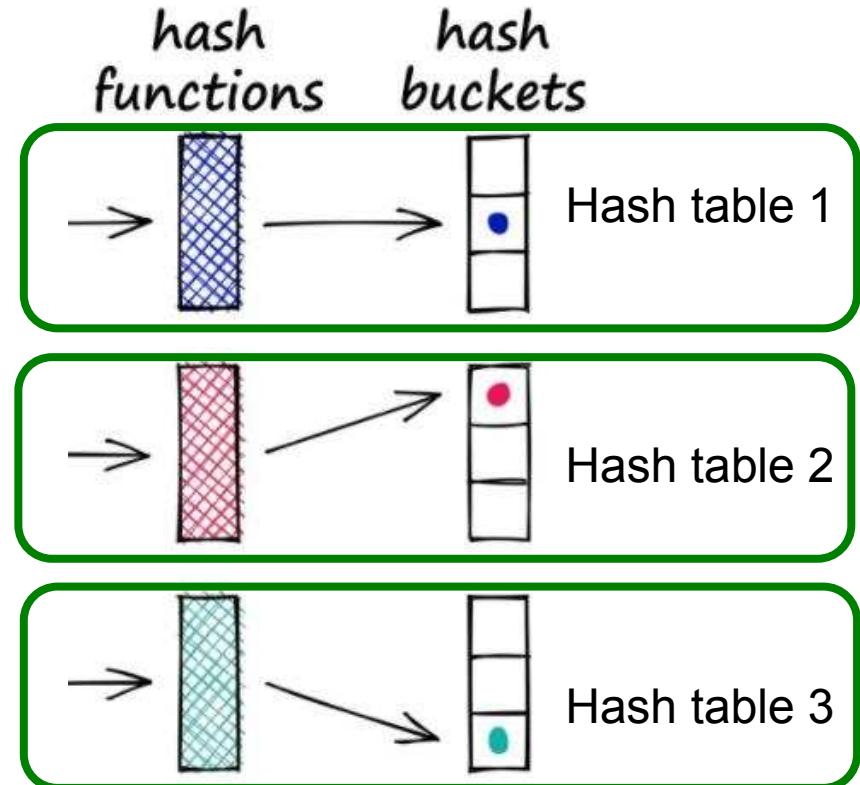
# Using hash tables for LSH

## Method 2: using multiple hash tables

1. Construct multiple hash tables
2. Different tables use different sets of hash functions.

Generate M different sets of LSH hash functions. One set of hash functions is for one hash table.

Example: using 3 hash tables.



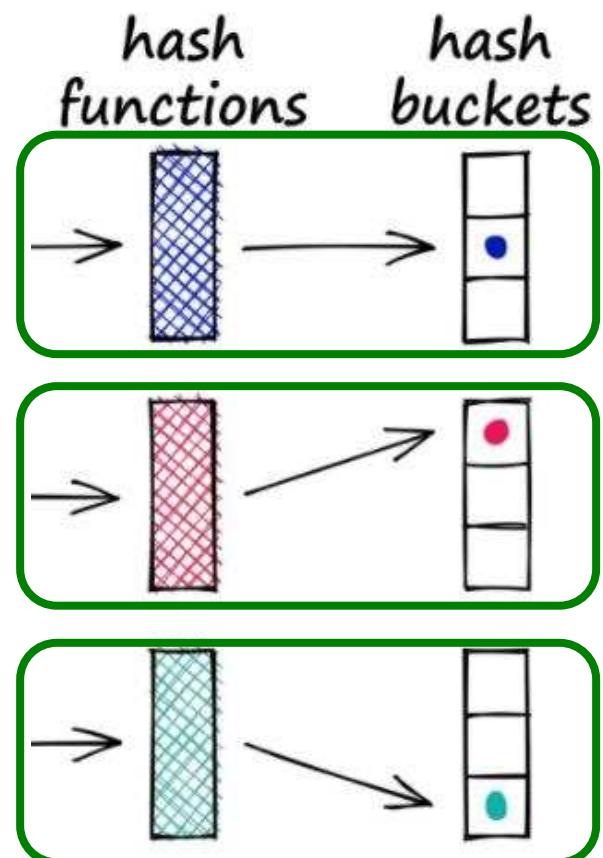
For one hash table, we use its own set of LSH hash functions to generate hash vectors.

# Using hash tables for LSH

## Method 2: using multiple hash tables (cont.)

Query process:

1. Use the query hash vector to retrieve one bucket in each hash table.
2. Combine the retrieved results from all hash tables as candidate items
3. Perform accurate linear search on the candidate items to output top-k results



Example: using 3 hash tables.

# Using hash tables for LSH

Example: use 3 hash tables. We use different hash functions for each table.

Hash table 1

Key	Value (sample_idxes)
001111	(1, 3, 100)
111100	(13, 2, 5)
100110	(15, 19, 29)
101111	(14, 18, 9)
101010	()

Hash table 2

Key	Value (sample_idxes)
001111	(100)
111100	()
100110	()
101111	(9, 10)
101010	(40)

Hash table 3

Key	Value (sample_idxes)
001111	(11, 23)
111100	(9, 21)
100110	()
101111	(30)
101010	(20)

Query X-> [111100]

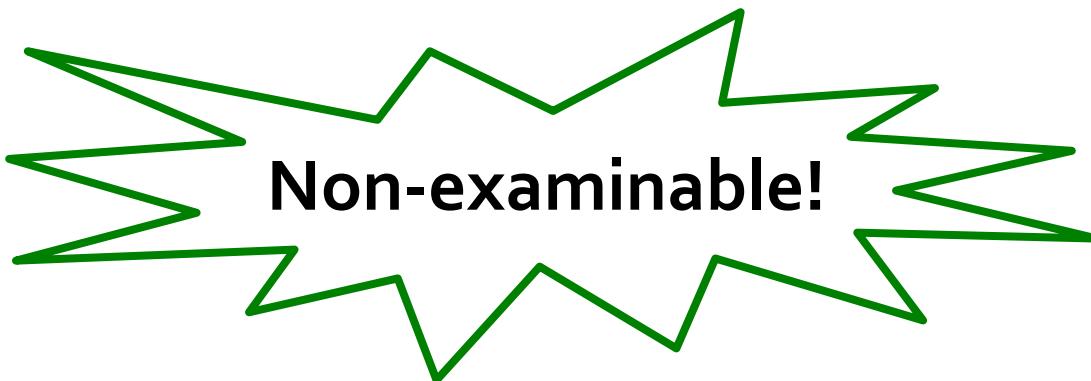
Query X-> [101010]

Query X-> [101010]

1. For one query X, we will generate 3 hash vectors.
2. Retrieved one bucket from each hash table
3. Combined the retrieved buckets from each table: candidate set: {13, 2, 5, 40}
4. Linear search on the candidate set using L2 distance to return the top-k results

# Extended discussion

- Extended discussion (non-examinable!)
  - Geometry view of LSH-random projection



# Geometry view of LSH

$\mathbf{n}$ : a randomly generated vector, represents one hashing function

- Given the vector  $\mathbf{n}$ , we can find a hyper-plane that

- passes through the origin and,
- the vector  $\mathbf{n}$  is orthogonal to the plane ( $\mathbf{n}$  is a normal vector to the hyper-plane).



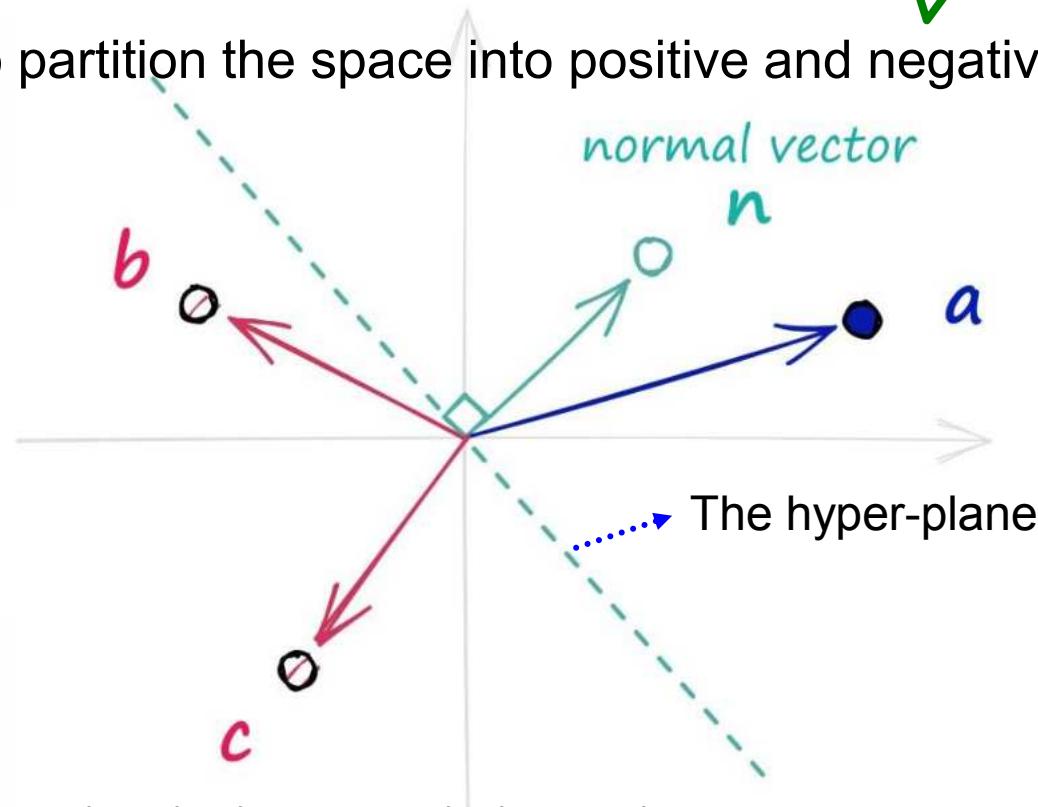
- The hyper-plane to partition the space into positive and negative regions

dot-product

$$\mathbf{n} \cdot \mathbf{a} > 0$$

$$\mathbf{n} \cdot \mathbf{b} < 0$$

$$\mathbf{n} \cdot \mathbf{c} < 0$$



# Similarity (recap)

## Inner product (dot product):

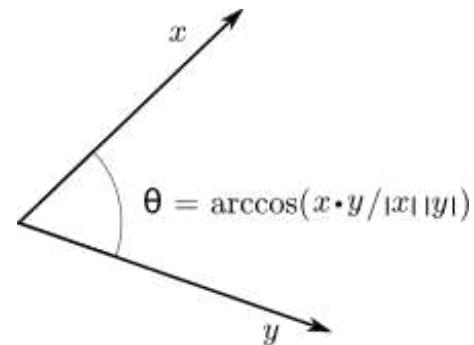
The dot product of two vectors  $\mathbf{a} = [a_1, a_2, \dots, a_n]$  and  $\mathbf{b} = [b_1, b_2, \dots, b_n]$  is defined as:<sup>[3]</sup>

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

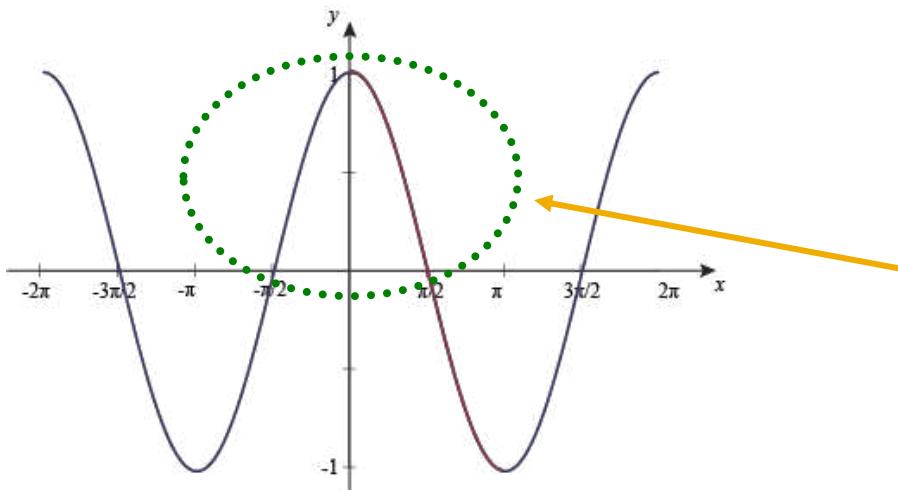
Geometric definition:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta,$$

where  $\theta$  is the angle between  $\mathbf{a}$  and  $\mathbf{b}$ .

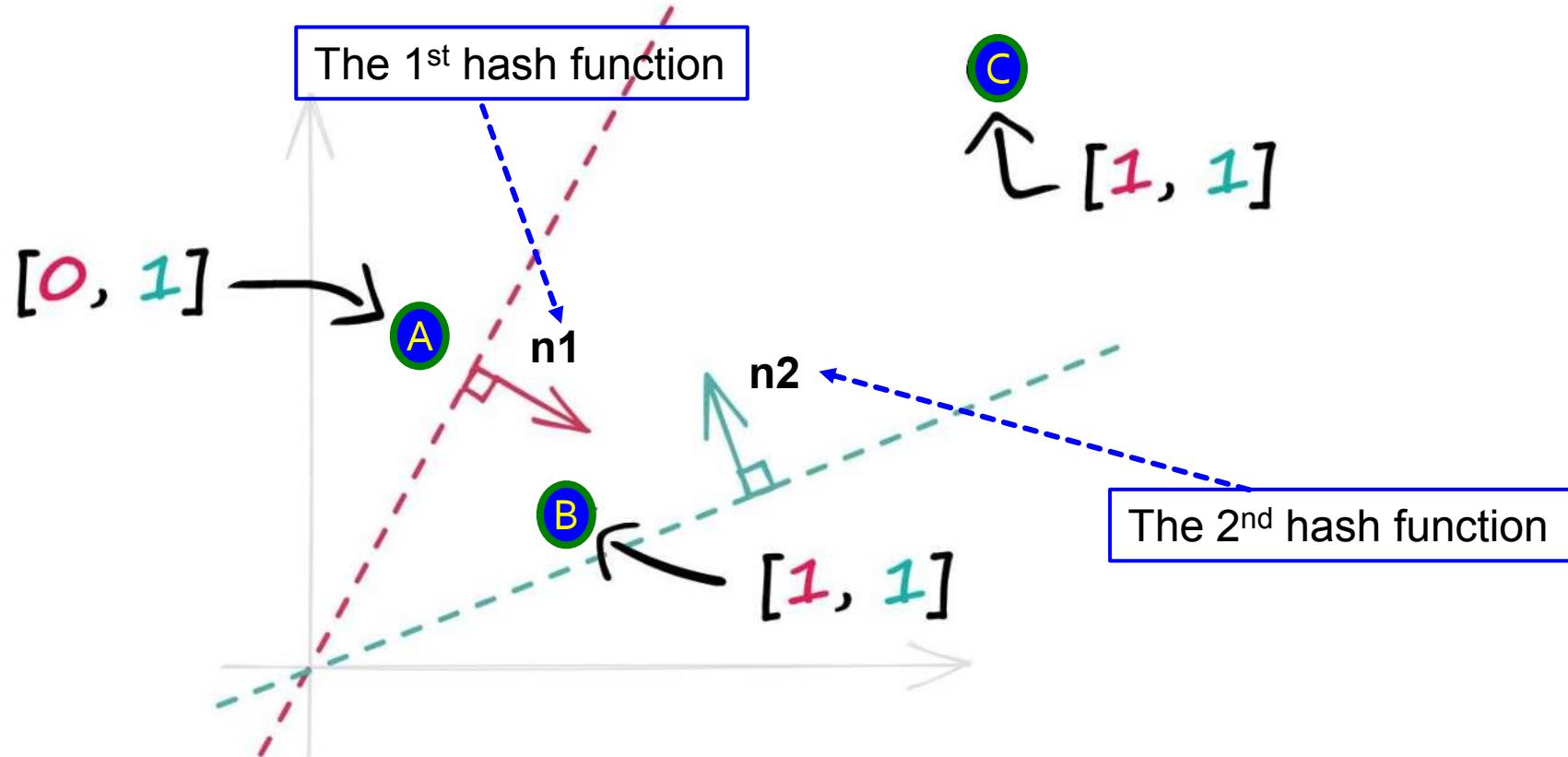


$$-90^\circ < \theta < 90^\circ \rightarrow \cos(\theta) > 0$$



The output of the hashing function is determined by  $\cos(\theta)$ :

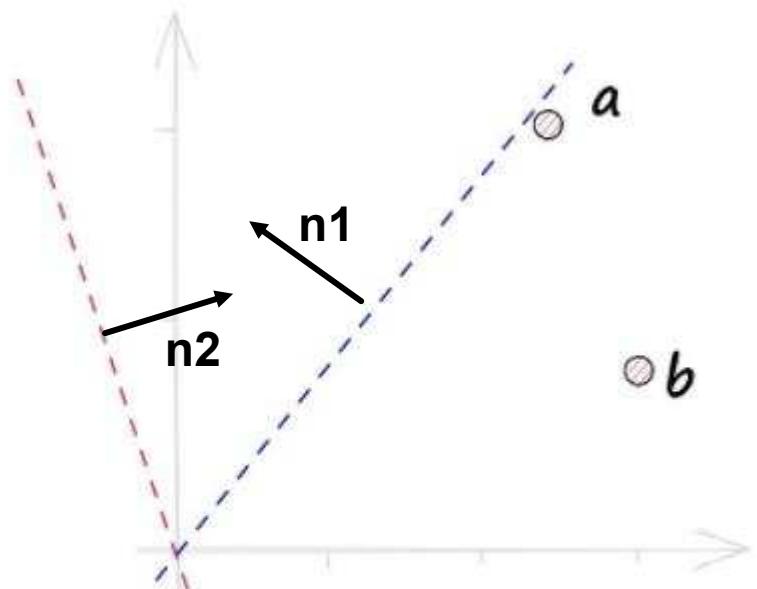
$$h(x) = \begin{cases} 1 & : w^T x > 0 \\ 0 & : w^T x \leq 0 \end{cases}$$



One hash function (1 bit) provides weak similarity information:

It indicates whether the data points lie in the same side of the hyper-plane or not.  
 If they lie in the same side -> they are similar.  
 Otherwise, they are not similar.

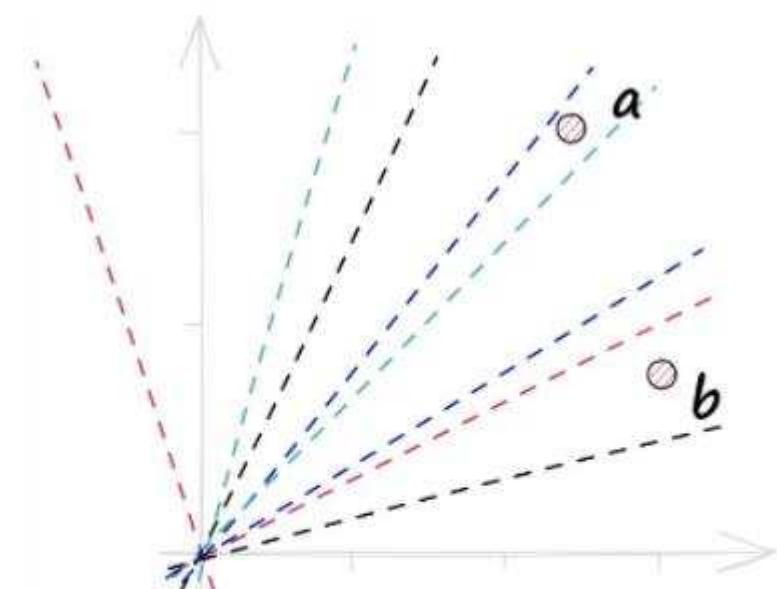
*nbits = 2*



$$a = 0 \text{ } 1$$

$$b = 0 \text{ } 1$$

*nbits = 8*



$$a = 0 \text{ } 1 \text{ } 1 \text{ } 1 \text{ } 0 \text{ } 1 \text{ } 0 \text{ } 0$$

$$b = 0 \text{ } 1 \text{ } 0 \text{ } 1 \text{ } 1 \text{ } 0 \text{ } 0 \text{ } 0$$

Increasing the **nbits** parameter increases the number of hyperplanes used to build the binary vector representations.

## Online resources

- FAISS
  - Faiss is a library for efficient similarity search and clustering of dense vectors.
  - <https://github.com/facebookresearch/faiss/wiki>

# Similarity Search - PQ

Lin Guosheng  
School of Computer Science and Engineering  
Nanyang Technological University

# Outline

- Nearest Neighbour Search (NNS) Problems
- Local Sensitive Hashing (LSH) – Random Projection
- **Product Quantization (PQ)**
- Inverted File Index (**non-examinable!!**)

# Product Quantization (PQ)

- Product Quantization (PQ)
  - A clustering based method
  - Compress the data
  - Speed up searching by using precomputed distances

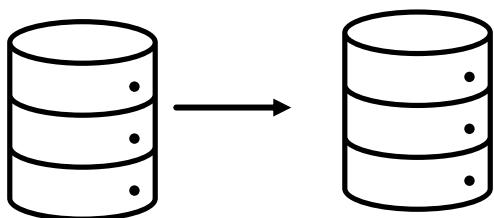
Most materials are from

<https://www.pinecone.io/learn/product-quantization/>

Paper: Product quantization for nearest neighbor search, 2011

# Overview of PQ for similarity search

Step1: convert the database  
into compressed vectors (offline step)

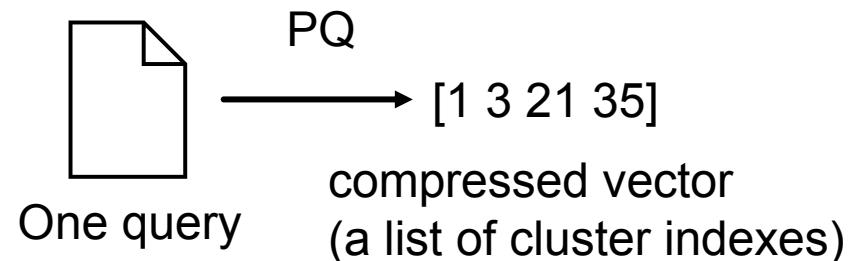


Compressed vectors

When using symmetric distance:  
Precompute the  
centroid distances tables  
for each subspace

Step 2: process query requests  
Use efficient approximate distance  
(symmetric or asymmetric)  
calculation to get top-k retrieved results

When using symmetric distance:  
For each query, we need to  
convert the query to a compressed vector



# The process of PQ for compression

## Pre-processing:

1. Define the number of subspaces (subvectors).
2. For each subspace, generate k cluster centroids using the samples in the database.

## Generating compressed vectors using PQ

Input: the input feature vector (high-dimensional)

Steps:

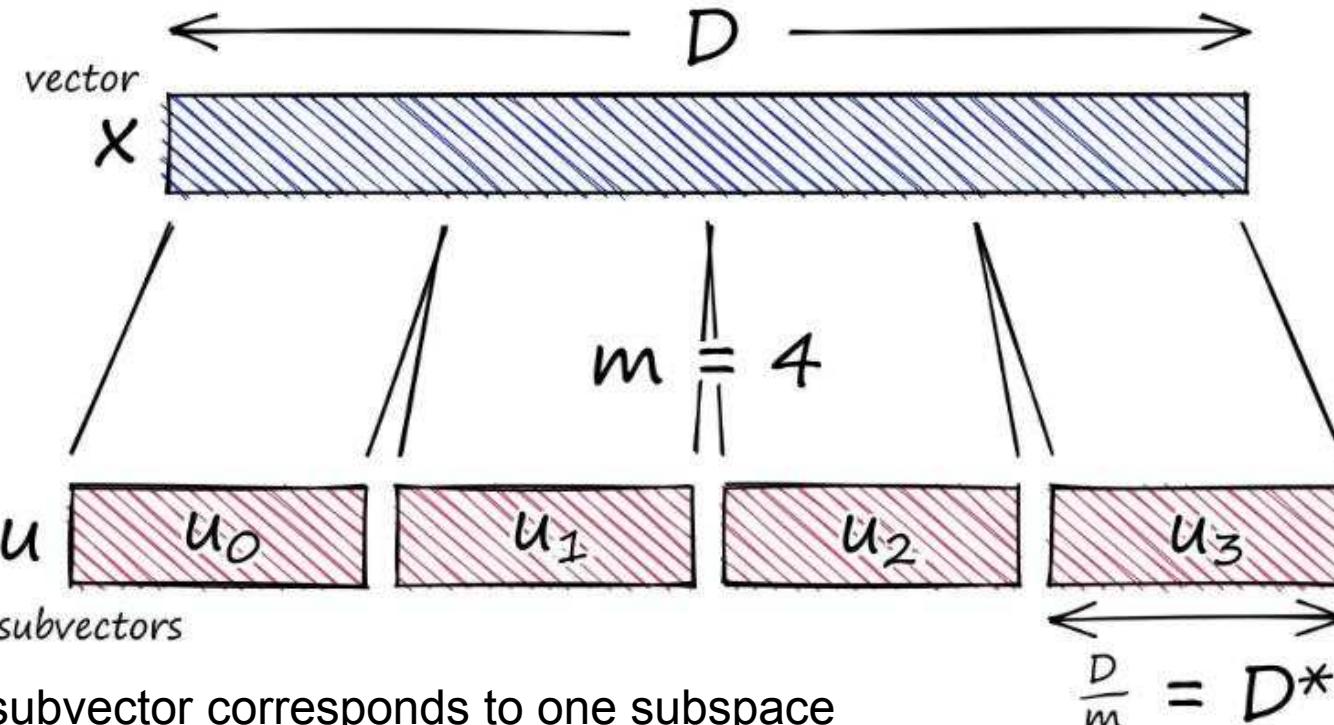
1. Equally split the input into M sub-vectors.  
Each sub-vector corresponds to one subspace.
2. Assigning each of these sub-vectors to its nearest cluster centroid  
(also called reproduction/reconstruction values)
3. Use the assigned cluster IDs (indexes) of all sub-vectors to construct the compressed vector (or called quantized vector).

# Example

We define  $M=4$  subspaces.

-> We generate 4 sub-vectors for each input.

(equally split the input vector into  $M$  sub-vectors )

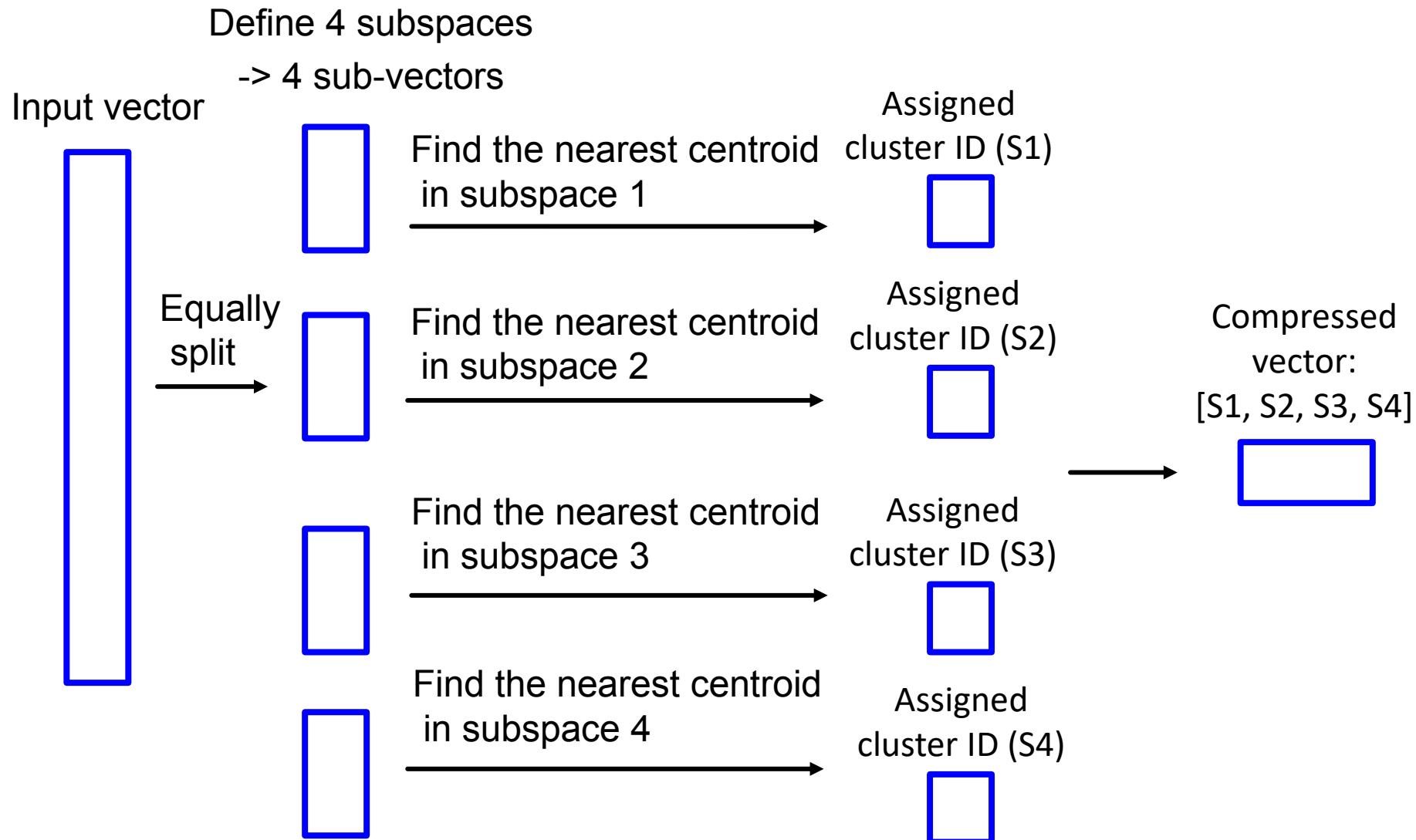


One subvector corresponds to one subspace

E.g., if the input vector dimension  $D=100$ ,  
the length of each sub-vector is  $100/4 = 25$

Sub-vector generation example

# Compressed vector



# Example

- Product Quantization example

Define 2 Subspaces

Each subspace has 4 centroids.

Input vectors:

A: [1.82, 5.08, 2.21, 4.21]

B: [4.96, 4.46, 4.1, 1.3]

# Example

**Subspace 1**, We have 4 centroids: C1 to C4.

A1, B1: the sub-vectors for datapoint A and B, respectively

$$A1 = (1.82, 5.08)$$

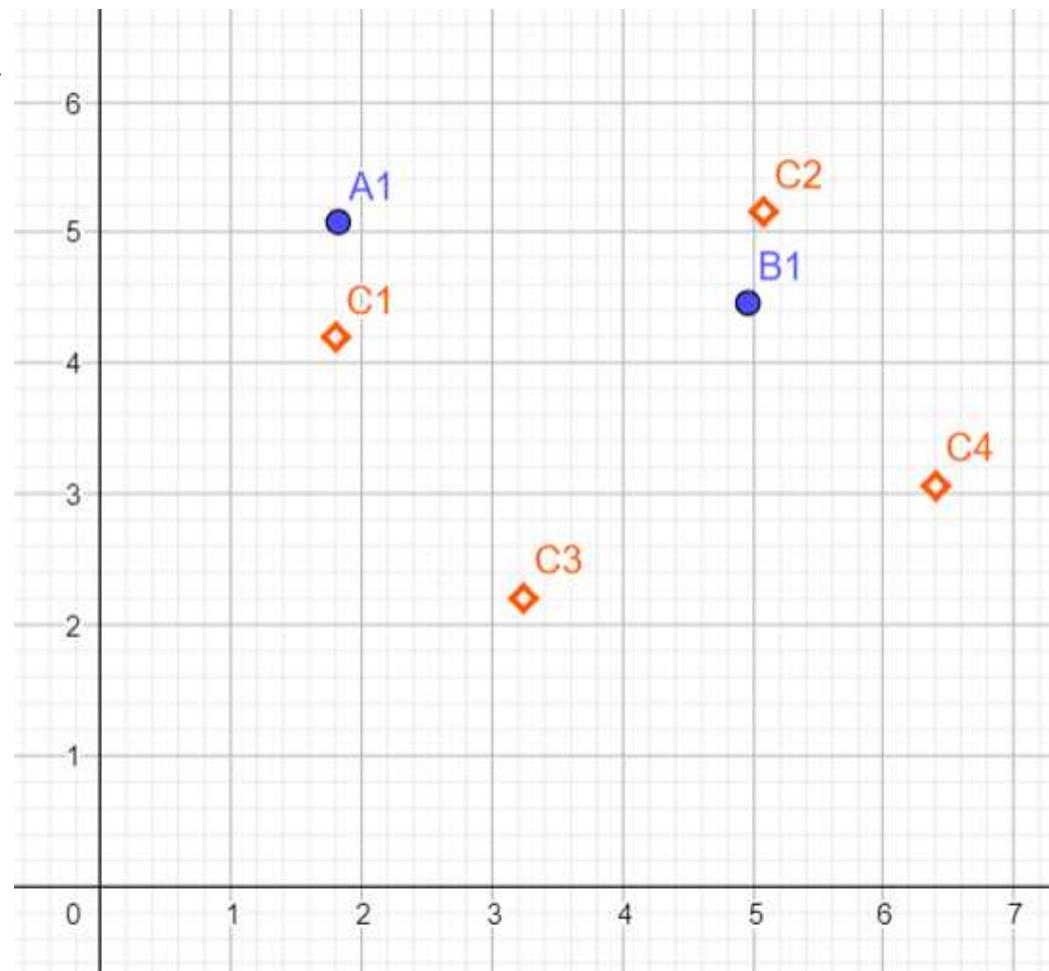
$$B1 = (4.96, 4.46)$$

$$C1 = (1.8, 4.2)$$

$$C2 = (5.08, 5.16)$$

$$C3 = (3.24, 2.2)$$

$$C4 = (6.4, 3.06)$$



# Example

**Subspace 2,** We have 4 centroids: C1 to C4.

A2, B2: the sub-vectors for datapoint A and B, respectively

$$A2 = (2.01, 4.21)$$

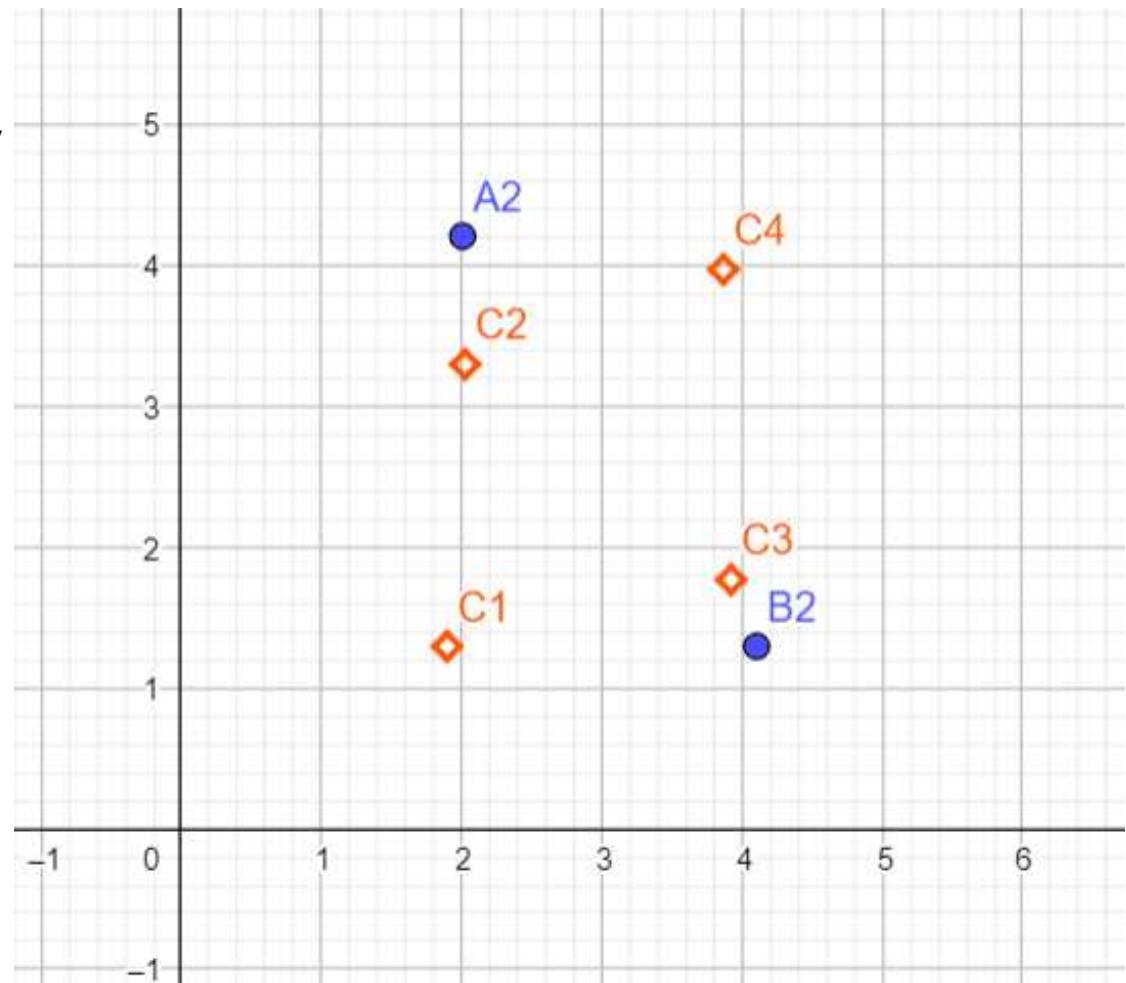
$$B2 = (4.1, 1.3)$$

$$C1 = (1.9, 1.3)$$

$$C2 = (2.02, 3.3)$$

$$C3 = (3.92, 1.77)$$

$$C4 = (3.87, 3.98)$$



# Example

- Q: generate the compressed vector for datapoint A and B:

Answer:

In subspace 1, A is assigned to the centroid C1  
**(1st dimension of the compressed vector)**

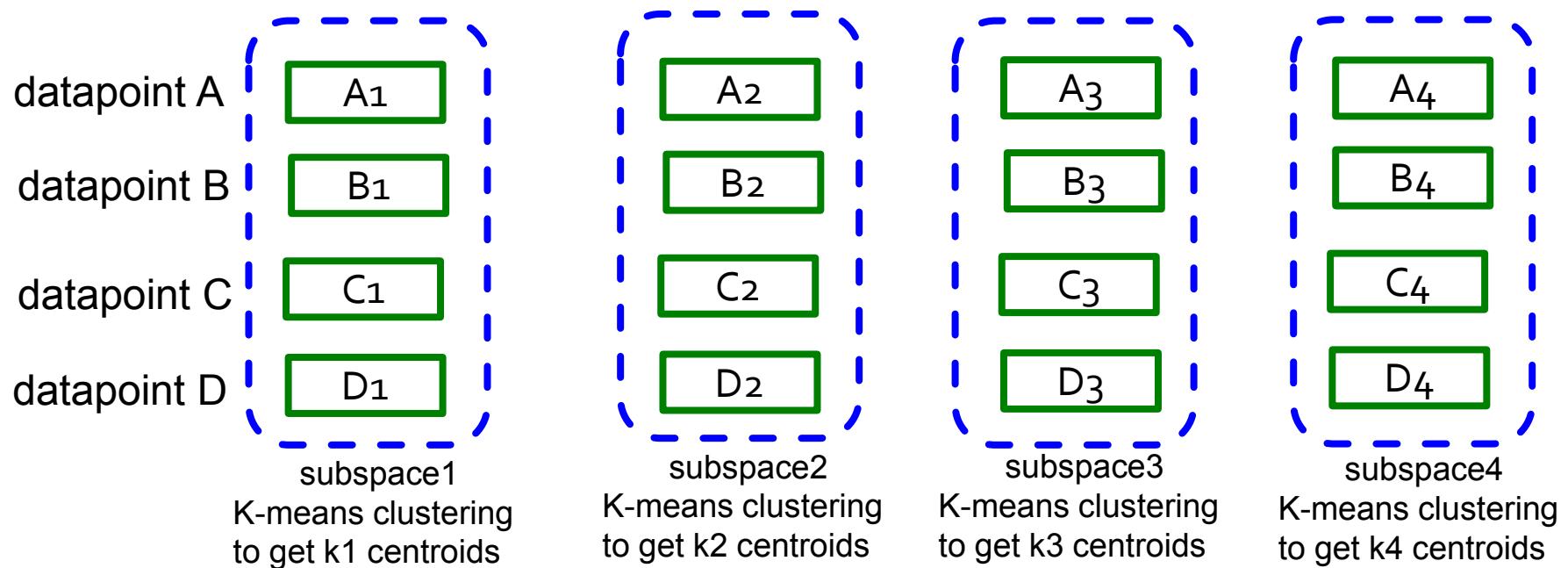
In subspace 2, A is assigned to the centroid C2  
**(2nd dimension of the compressed vector)**

-> A: [1, 2]

In subspace 1, B is assigned to the centroid C2  
In subspace 2, B is assigned to the centroid C3  
■ -> B: [2, 3]

- How to generate cluster centroids?
  - Generate sub-vectors for all data examples in the database
    - Define M subspaces -> Generate M sub-vectors for one input.
  - Perform K-means in each subspace to generate k centroids.
    - For example, we can generate k=256 centroids for each sub space

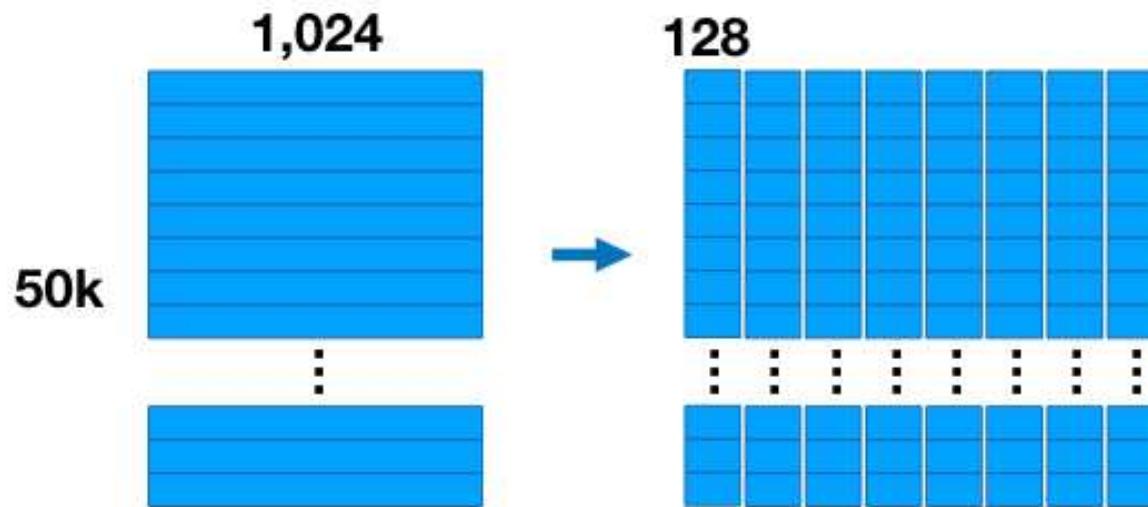
Use datapoints in the database to generate centroids for each subspace



# Example: Subspace clustering

Database: 50K samples,  
each has 1024 dimensions:

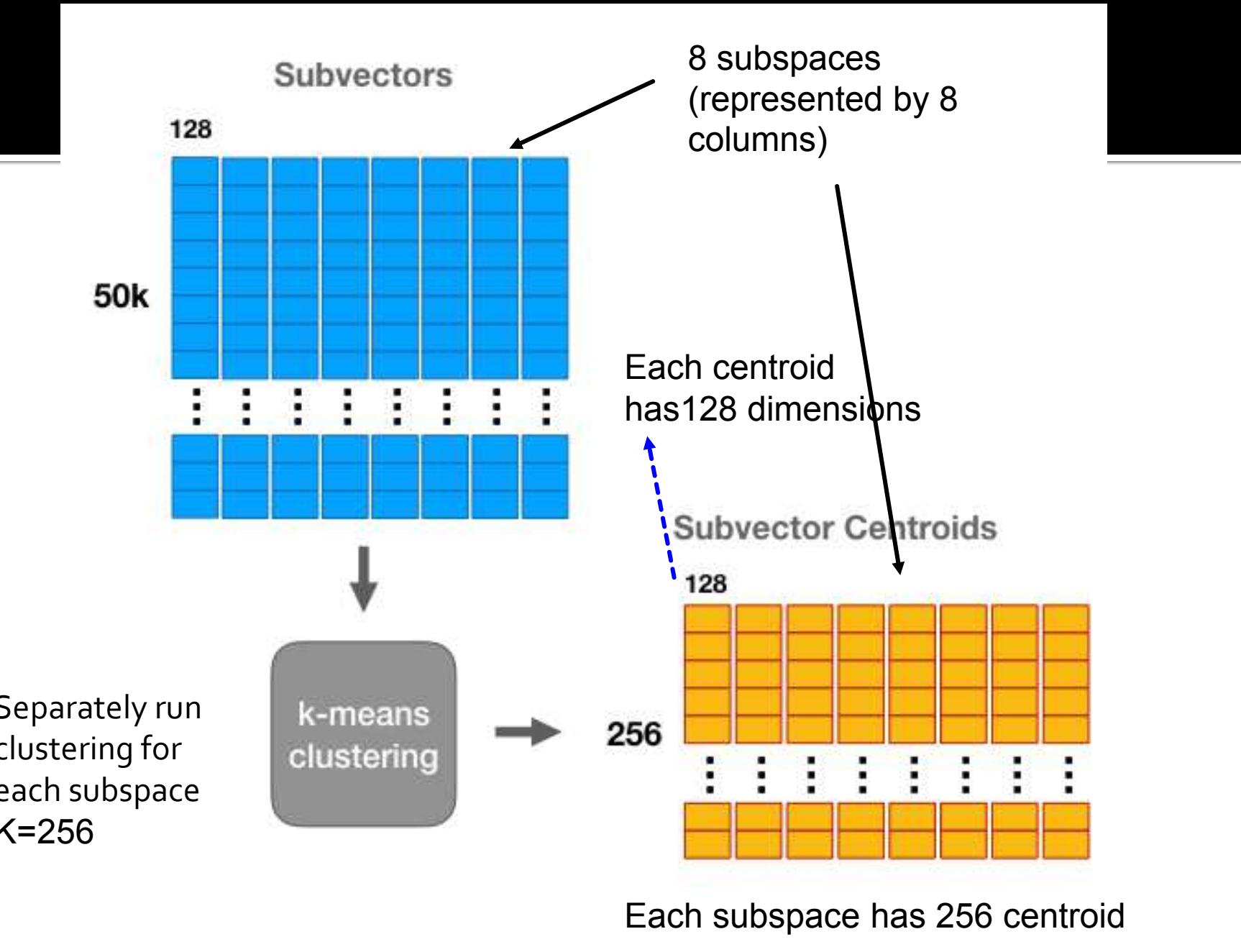
8 sub-vectors for each sample:  
(each sub-vector's dimension: 128)



Each row indicates one data sample.

Define 8 subspaces. Each vector is spitted into 8 sub-vectors, each of length 128 (8 sub vectors x 128 dimensions = 1,024 dimensions).

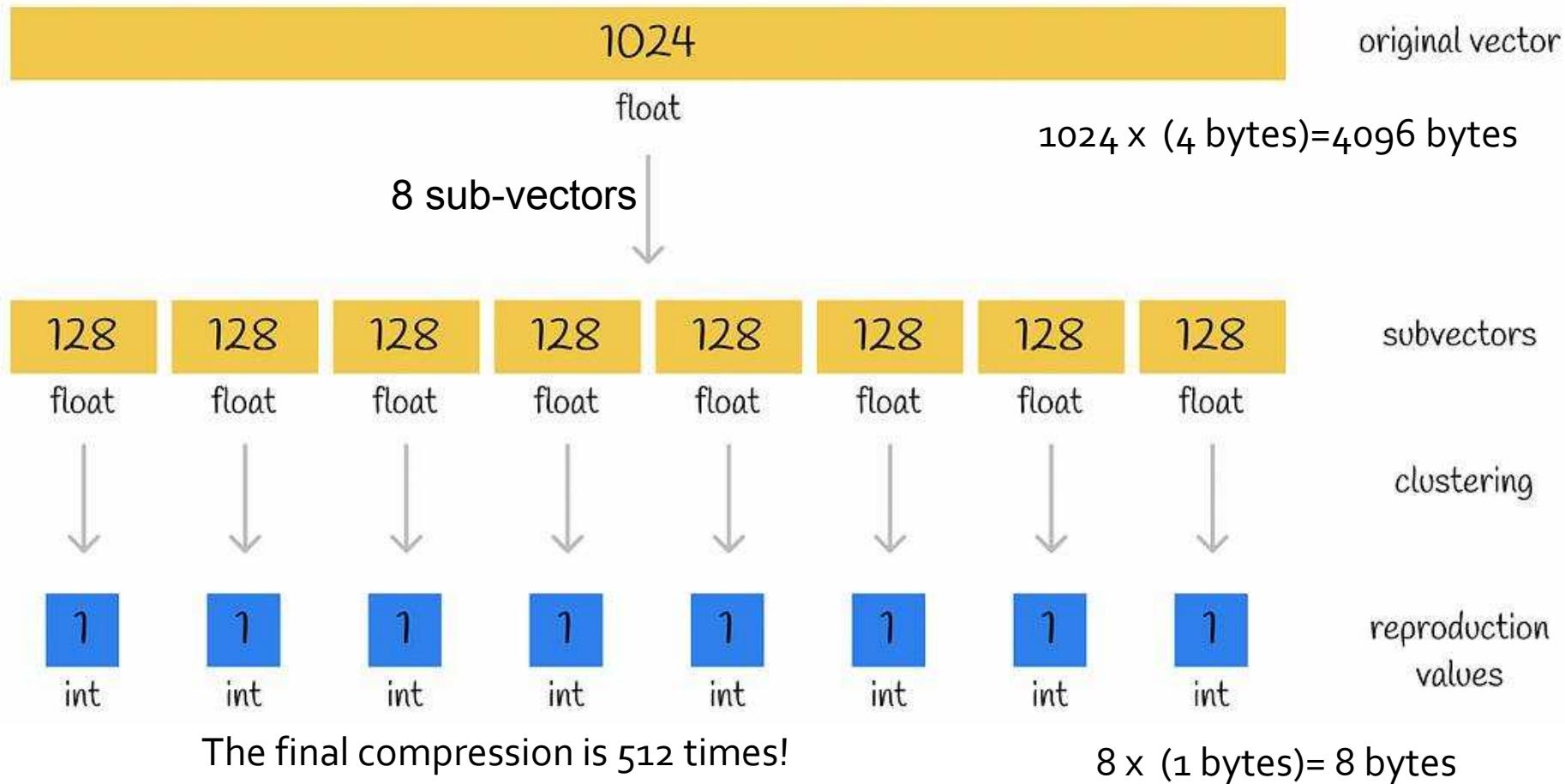
<https://mccormickml.com/2017/10/13/product-quantizer-tutorial-part-1/#nearest-neighbor-search>



# Compression analysis

1 float value requires 4 bytes ( $4 \times 8 = 32$  bits);

1 integer value requires 1 byte (here we use 8-bit integer, value range: 0-255)



One centroid ID is an integer value. In this example, we only have 256 centroids, so we can use one 8-bit integer ( $2^{8=256}$ ) to store one cluster ID.

# Product Quantization

- Codebook based compression
  - the codebook is the centroids
  - Separate codebook (centroids) for each subspace
- Compress high-dimensional vector to a tiny vector of IDs that only requires very little memory/storage space.

# Product Quantization

- Product Quantization (PQ)
  - A data dependent method
    - Need to use the data in the database to determine the parameters (cluster centres) of the algorithm.
  - LSH is a data independent method
    - The hash functions are randomly generated, not using the data in the database.

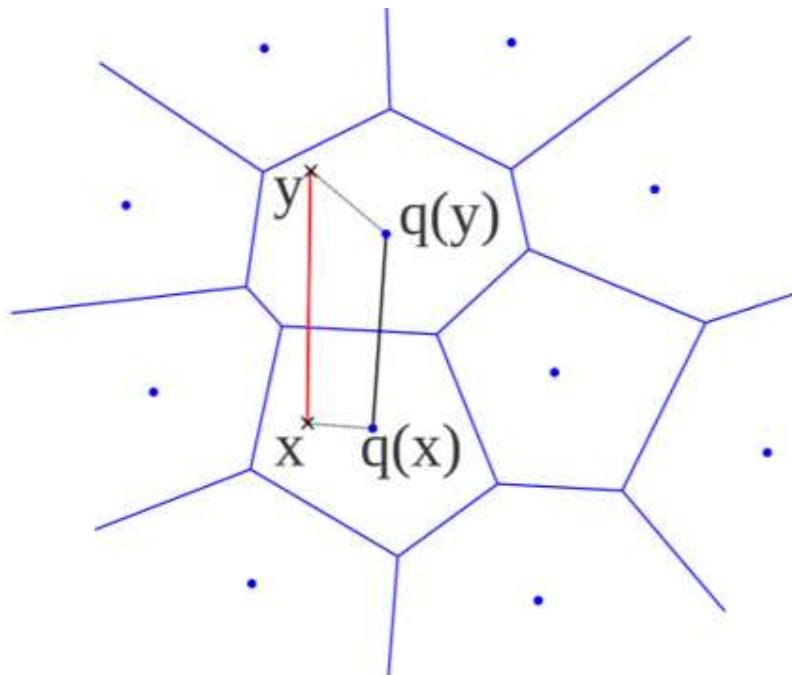
# PQ for similarity search

- PQ for Similarity Search
  - Use liner search (exhaustive search )
    - Use approximate distance to speed up distance calculation
  - Calculating approximate distance
    - Symmetric distance
      - Use pre-computed distance look-up table to speed up the calculation.
    - Asymmetric distance

# Symmetric distance

- Approximate distance: Symmetric distance

$$\hat{d}(x, y) = d(q(x), q(y)) = \sqrt{\sum_j d(q_j(x), q_j(y))^2},$$



symmetric case

$x, y$  are two input vectors

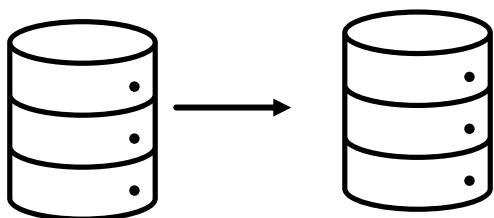
$q(x)$  and  $q(y)$  are the assigned cluster centroids

The distance of  $(x, y)$  is approximated by the distance of cluster centroids

# Recap

## Overview of PQ for similarity search

Step1: convert the database into compressed vectors (offline step)

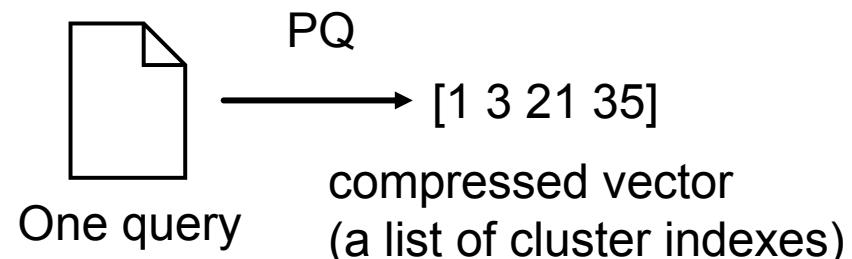


Compressed vectors

When using symmetric distance:  
Precompute the centroid distances tables for each subspace

Step 2: process query requests  
Use efficient approximate distance (symmetric or asymmetric) calculation to get top-k retrieved results

When using symmetric distance:  
For each query, we need to convert the query to a compressed vector



# Symmetric distance example

**Subspace 1,** We have 4 centroids: C1 to C4.

A1, B1: the sub-vectors for datapoint A and B, respectively

$$A1 = (1.82, 5.08)$$

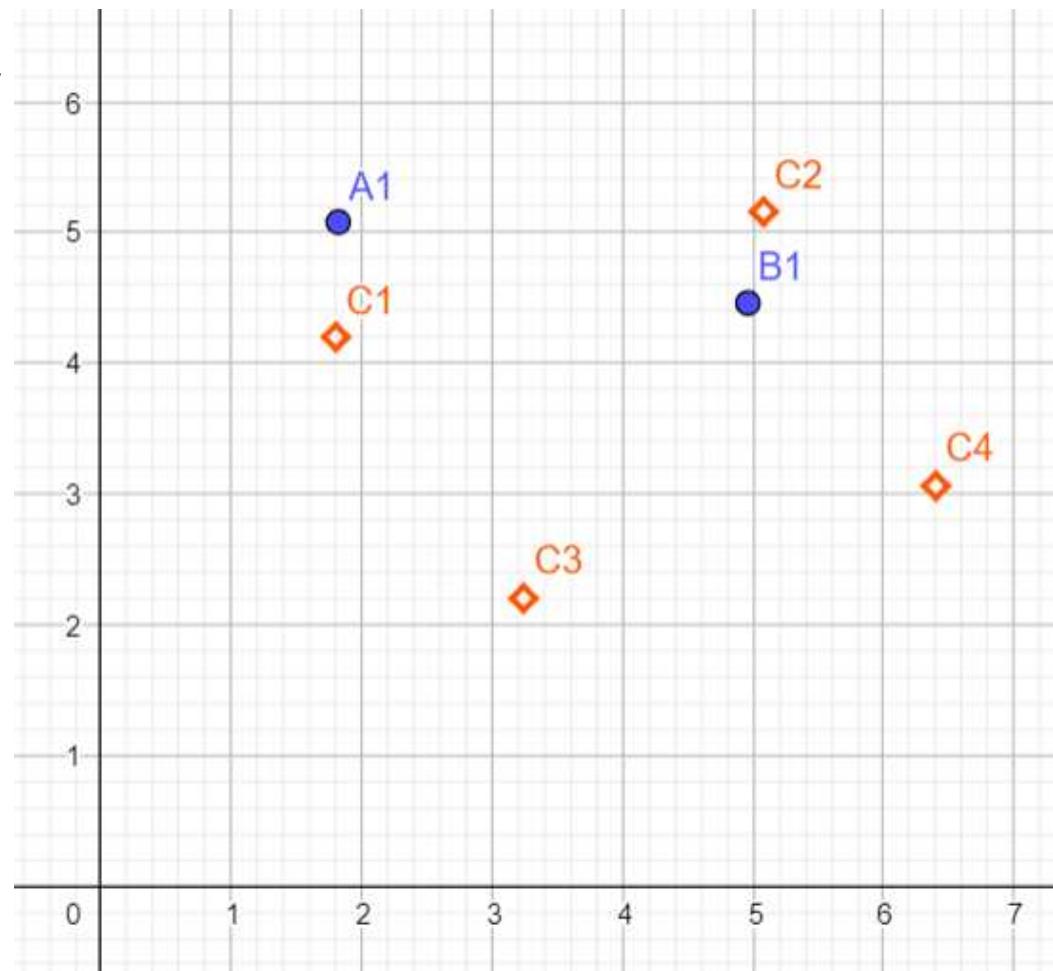
$$B1 = (4.96, 4.46)$$

$$C1 = (1.8, 4.2)$$

$$C2 = (5.08, 5.16)$$

$$C3 = (3.24, 2.2)$$

$$C4 = (6.4, 3.06)$$



# Symmetric distance example

**Subspace 2,** We have 4 centroids: C1 to C4.

A2, B2: the sub-vectors for datapoint A and B, respectively

$$A2 = (2.01, 4.21)$$

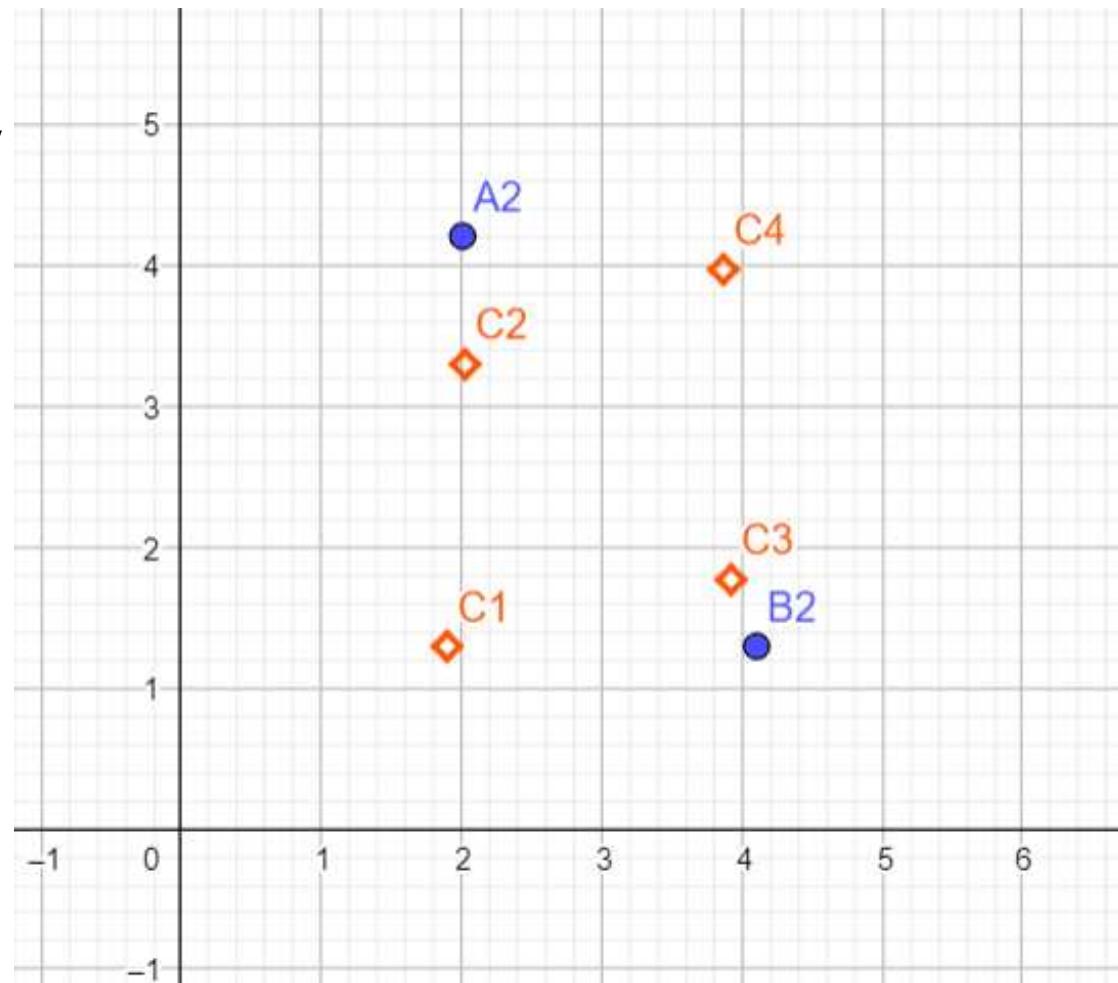
$$B2 = (4.1, 1.3)$$

$$C1 = (1.9, 1.3)$$

$$C2 = (2.02, 3.3)$$

$$C3 = (3.92, 1.77)$$

$$C4 = (3.87, 3.98)$$



# Symmetric distance example

Define 2 subspaces. The centroids and datapoints are given in the above slides.

**Question:**

1. Construct centroid distance look up tables and
2. compute the approximate Squared L2 distance (symmetric case) between A and B.

# Symmetric distance example

## ■ Solution to Q1: Subspace 1

	Centroid 1	Centroid 2	Centroid 3	Centroid 4
Centroid 1	0	11.68	6.07	22.46
Centroid 2	11.68	0	12.15	6.15
Centroid 3	6.07	12.15	0	10.73
Centroid 4	22.46	6.15	10.73	0

Table 1: squared L2 distance table for Subspace 1 (D1)

Distance calculation example:

$$C1=[1.8, 4.2], C2=[5.08, 5.16],$$

$$D1(C1, C2) = (1.8-5.08)^2 + (4.2-5.16)^2 = 10.76 + 0.92 = 11.68$$

**Subspace 1**, We have 4 centroids: C1 to C4.

A1, B1: the sub-vectors for datapoint A and B, respectively

$$A1 = (1.82, 5.08)$$

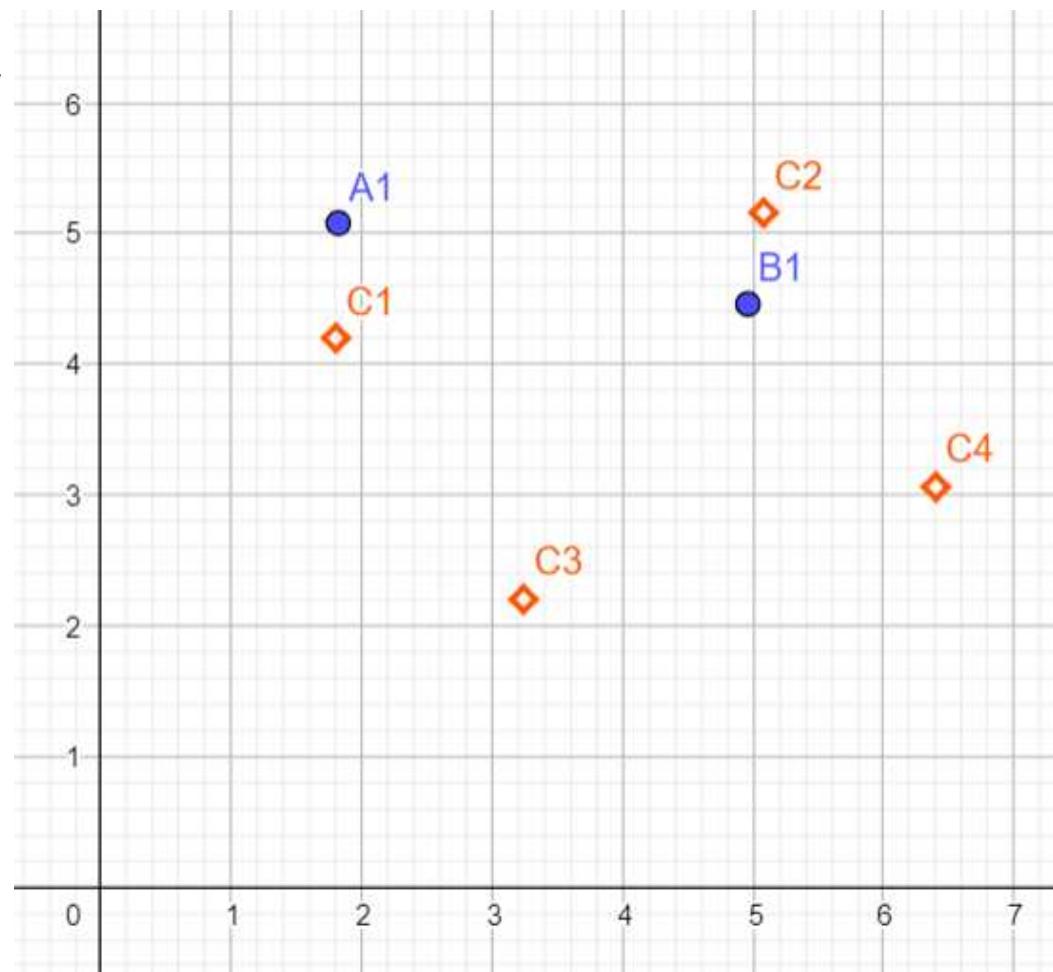
$$B1 = (4.96, 4.46)$$

$$C1 = (1.8, 4.2)$$

$$C2 = (5.08, 5.16)$$

$$C3 = (3.24, 2.2)$$

$$C4 = (6.4, 3.06)$$



## Subspace 2

	Centroid 1	Centroid 2	Centroid 3	Centroid 4
Centroid 1	0	4.01	4.30	11.06
Centroid 2	4.01	0	5.95	3.88
Centroid 3	4.3	5.95	0	4.89
Centroid 4	11.06	3.88	4.89	0

Table 2: squared L2 distance table for Subspace 2 (D2)

**Subspace 2,** We have 4 centroids: C1 to C4.

A2, B2: the sub-vectors for datapoint A and B, respectively

$$A2 = (2.01, 4.21)$$

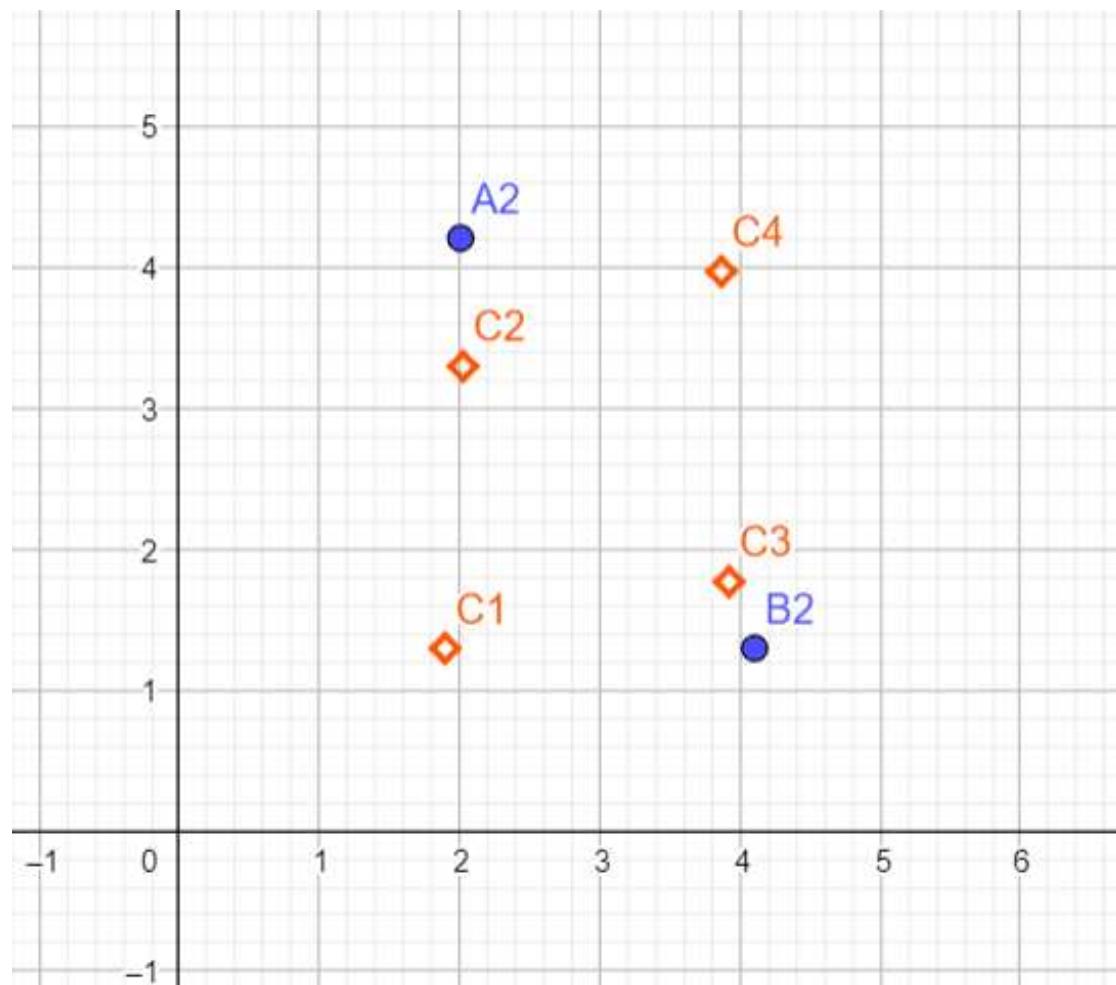
$$B2 = (4.1, 1.3)$$

$$C1 = (1.9, 1.3)$$

$$C2 = (2.02, 3.3)$$

$$C3 = (3.92, 1.77)$$

$$C4 = (3.87, 3.98)$$



**Question:**

2. compute the approximate Squared L<sub>2</sub> distance (symmetric case) between A and B.

■ **Solution to Q2:**

**Compressed vectors: A: [1, 2]; B: [2, 3]**

Look up the distance values in table D1 and D2 using the cluster IDs:

$$D(A_1, B_1) = D_1(\text{Centroid 1, Centroid 2}) = 11.68$$

$$D(A_2, B_2) = D_2(\text{Centroid 2, Centroid 3}) = 5.95$$

$$D(A, B) \approx D(A_1, B_1) + D(A_2, B_2) = 11.68 + 5.95 = 17.63$$

## Subspace 1

	Centroid 1	Centroid 2	Centroid 3	Centroid 4
Centroid 1	0	11.68	6.07	22.46
Centroid 2	11.68	0	12.15	6.15
Centroid 3	6.07	12.15	0	10.73
Centroid 4	22.46	6.15	10.73	0

## Subspace 2

	Centroid 1	Centroid 2	Centroid 3	Centroid 4
Centroid 1	0	4.01	4.30	11.06
Centroid 2	4.01	0	5.95	3.88
Centroid 3	4.3	5.95	0	4.89
Centroid 4	11.06	3.88	4.89	0

**Compressed vectors: A: [1, 2]; B: [2, 3]**

$$D(A_1, B_1) = D_1(\text{Centroid 1}, \text{Centroid 2}) = 11.68$$

$$D(A_2, B_2) = D_2(\text{Centroid 2}, \text{Centroid 3}) = 5.95$$

$$D(A, B) \approx D(A_1, B_1) + D(A_2, B_2) = 11.68 + 5.95 = 17.63$$

# Symmetric distance

- Approximate distance calculation
  - we can calculate the distances much more efficiently using just table look-ups.
  - Using centroid pre-computed distances.  
Pre-compute and the distances between centroids for each subspace, stored the distances in a matrix or lookup table, use the centroid index pair to retrieve the partial distances for each subspace.
  - Sum-up the partial distances from all subspaces as the final distance output

# Asymmetric distance

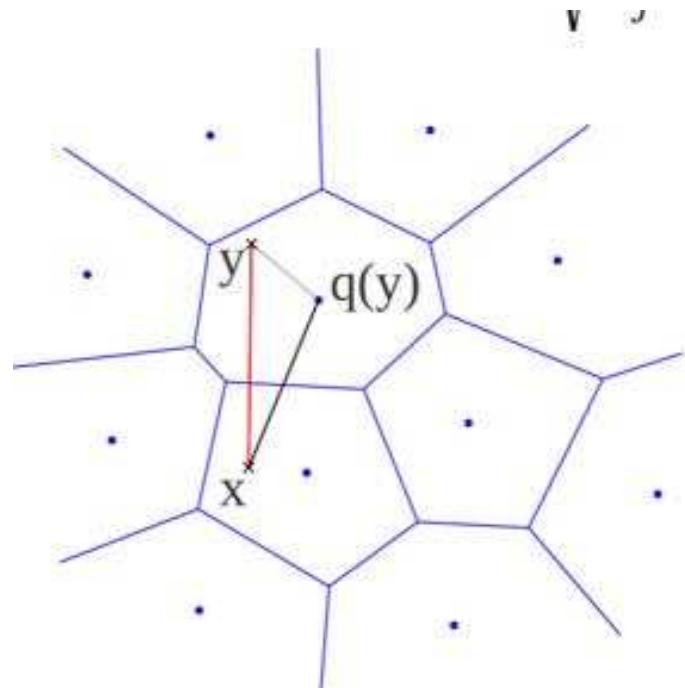
## Asymmetric distance

More accurate for distance calculation,  
as we only quantize one input.  
(compared to symmetric distance)

$$\tilde{d}(x, y) = d(x, q(y))$$

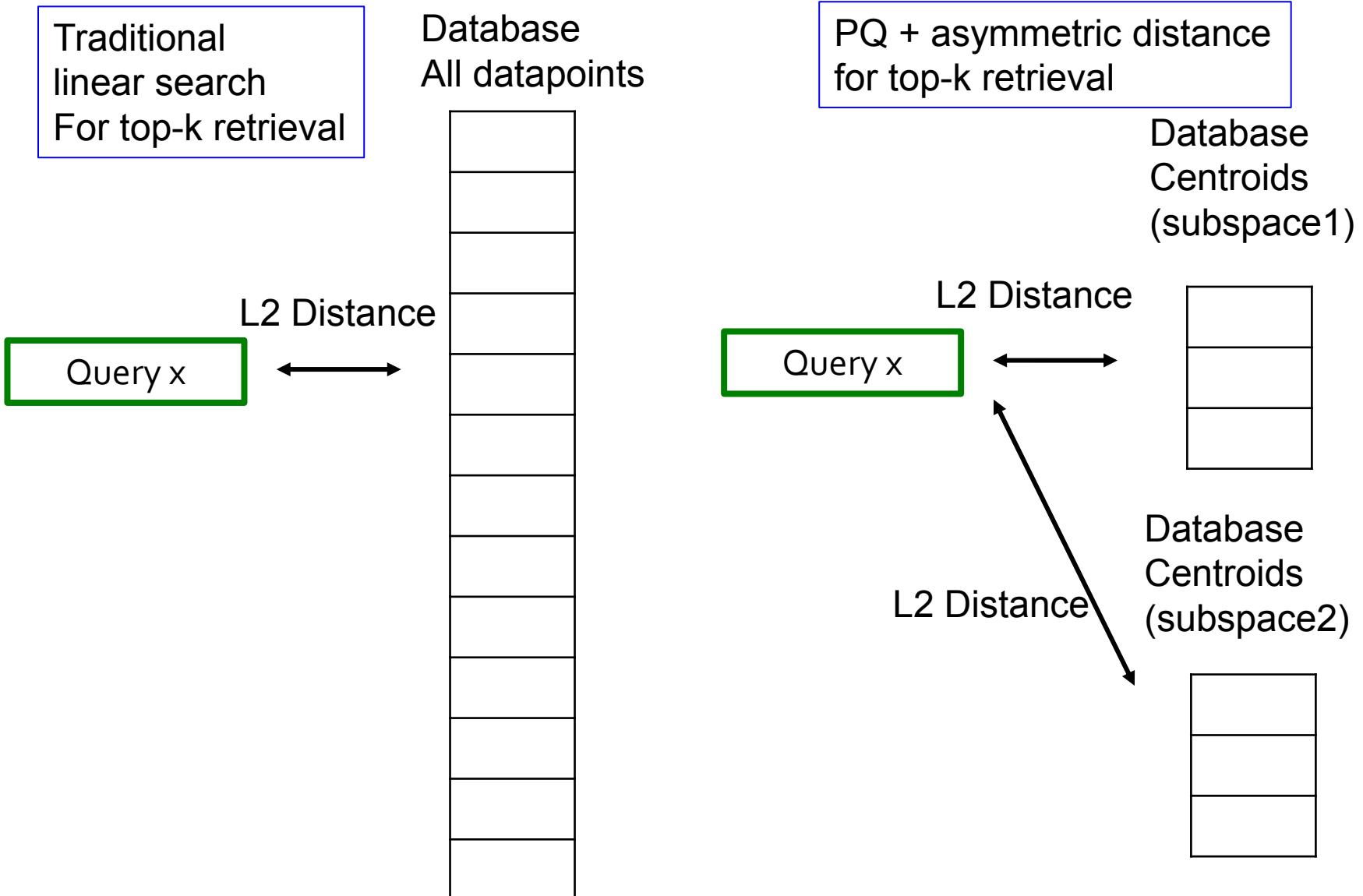
x: a query example,  
y: an example in the database.

Use  $q(y)$  to replace  $y$   
for distance calculation



asymmetric case

# Example for asymmetric distance



# Example (asymmetric distance)

Data points in the database:

A	[4, 6, 14, 4]
B	[6, 2, 8, 2]
D	[6, 8, 10, 2]
E	[4, 4, 6, 4]

Query data point:

Q	[16, 4, 4, 8]
---	---------------

Subspace 1

Centroid 1	[8,6]
Centroid 2	[18,6]

Subspace 2

Centroid 1	[10, 12]
Centroid 2	[10, 4]

Question: Calculate the approximate Squared L2 distance using **Asymmetric distance** between the query Q and all data points in the database (A, B, D, E)

# Example (asymmetric distance)

Solution:

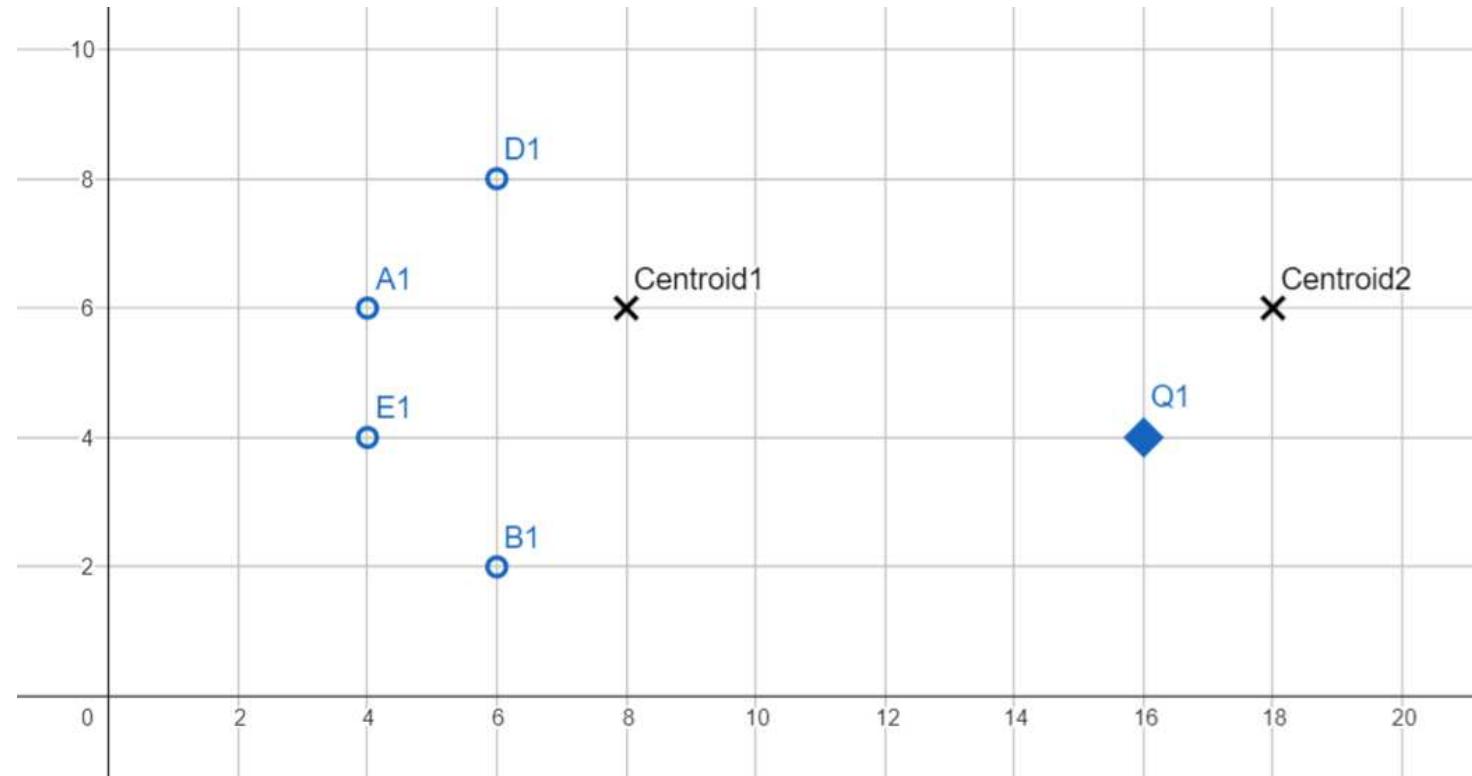
Step1: Compress the data points in the database  
(In each subspace, identify the nearest centroid for each data point)

Step 2: Calculate Asymmetric distances

# Example (asymmetric distance)

Subspace 1:

(A1, B1, D1, E1, Q1 are sub-vectors in Subspace1)



A1	[4, 6]
B1	[6, 2]
D1	[10, 2]
E1	[4, 4]

Q1	[16, 4]
----	---------

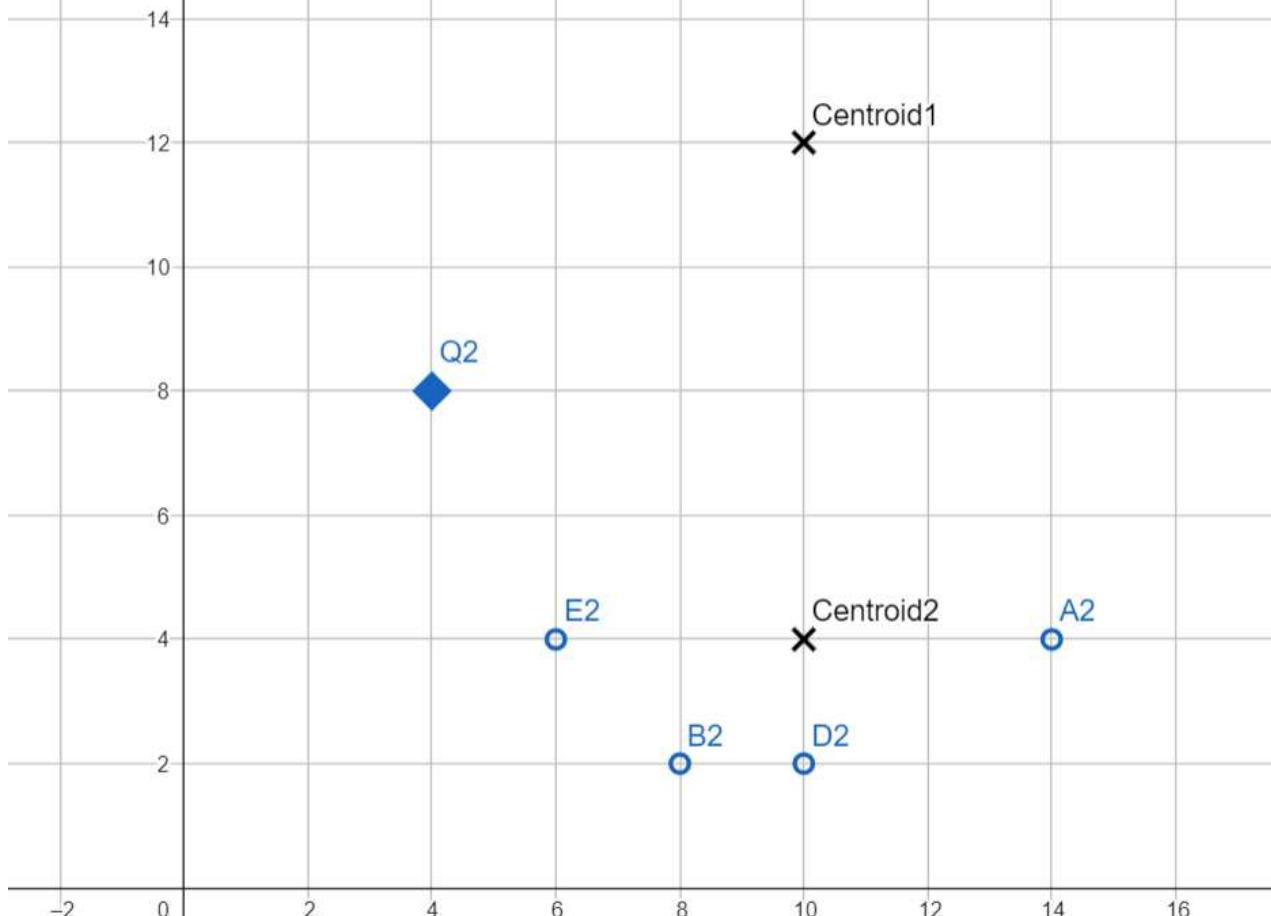
C1	[8, 6]
C2	[18, 6]

Calculate the squared L2 distances between the data points and each centroid to identify the nearest centroid

# Example (asymmetric distance)

Subspace 2:

(A2, B2, D2, E2, Q2 are sub-vectors in Subspace2)



A2	[14, 4]
B2	[8, 2]
D2	[10, 2]
E2	[6, 4]

Q2	[4, 8]
----	--------

C1	[10, 12]
C2	[10, 4]

Calculate the squared L2 distances between the data points and each centroid to identify the nearest centroid

# Example (asymmetric distance)

Step1: Compress the data points in the database  
(In each subspace, identify the nearest centroid for each data point)

compression results:

A -> [1, 2]

B -> [1, 2]

D -> [1, 2]

E -> [1, 2]

# Example (asymmetric distance)

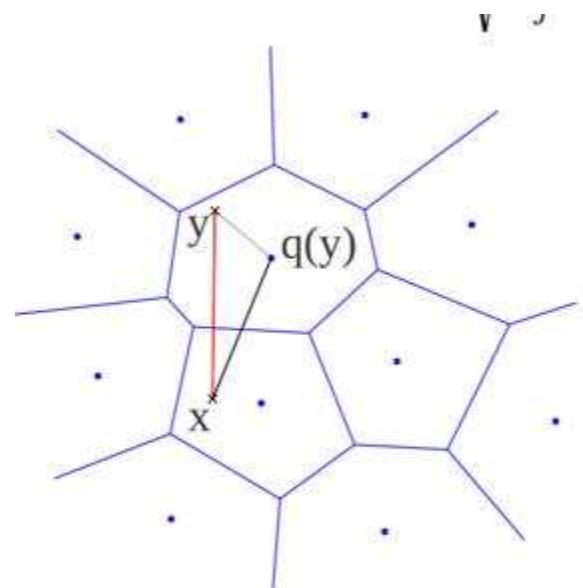
Step2: calculate Asymmetric distances

1) Generally, for the query Q, we need to calculate the distances between Q and all the centroids in each subspace.

2) For this question, as all data points are compressed as the same vector [1, 2], we only need to calculate the following distances:

Subspace 1: D (Q1, Centroid 1)  
Subspace 2: D (Q2, Centroid 2)

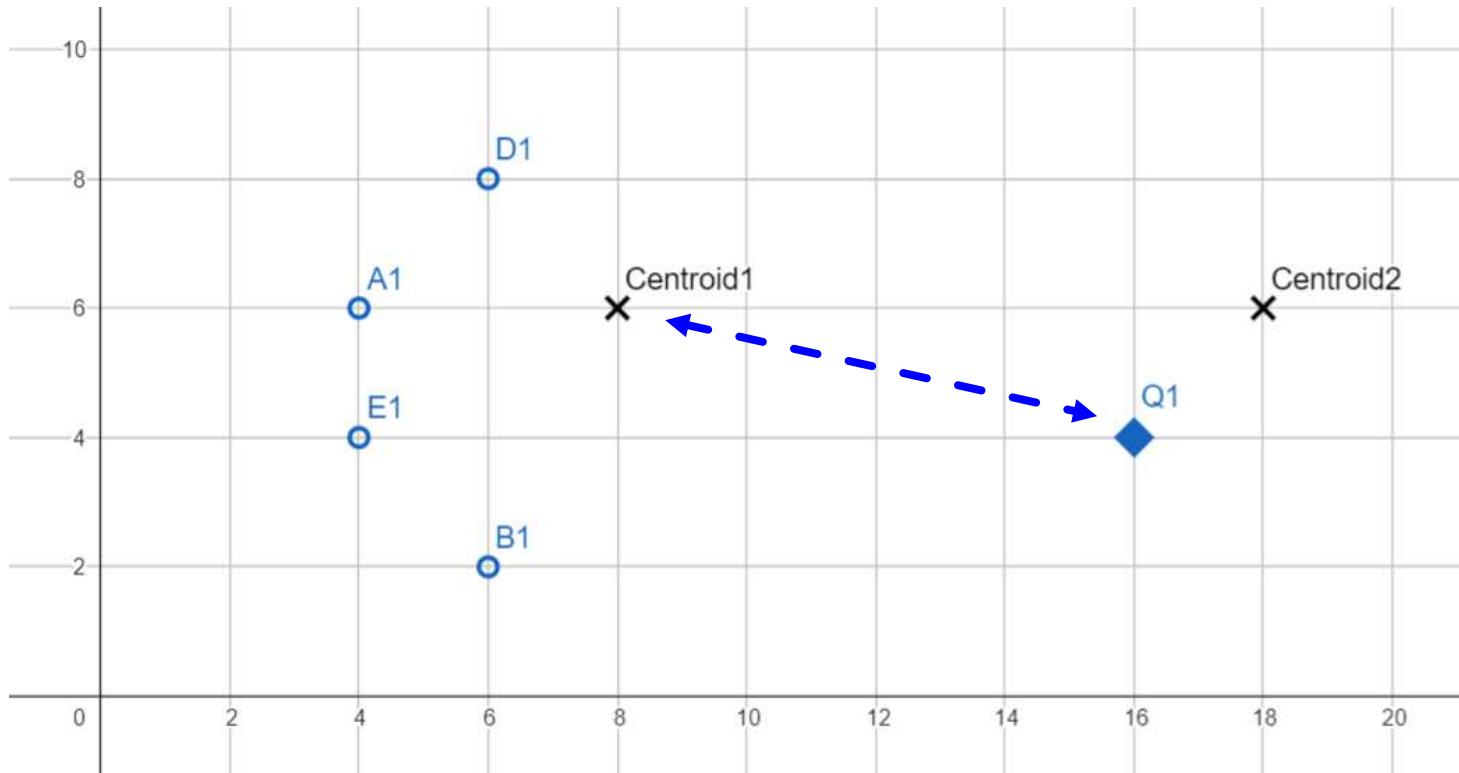
$$\tilde{d}(x, y) = d(x, q(y))$$



# Example (asymmetric distance)

Subspace 1:

(A1, B1, D1, E1, Q1 are sub-vectors in Subspace1)



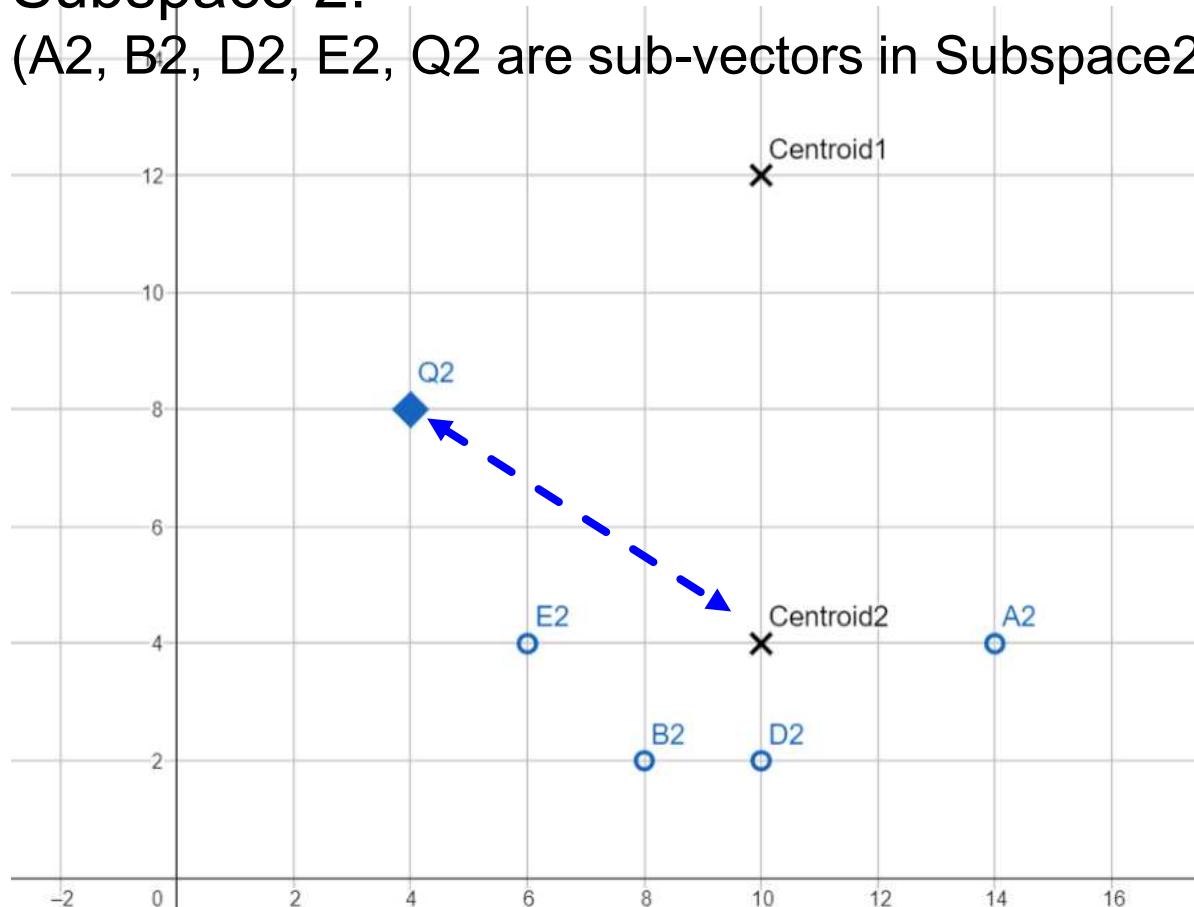
In one subspace, the data points belonging to the same centroid will have equal approximate distances to the query.

$$D(A1, Q1) = D(B1, Q1) = D(D1, Q1) = D(E1, Q1)$$

# Example (asymmetric distance)

Subspace 2:

(A2, B2, D2, E2, Q2 are sub-vectors in Subspace2)



In one subspace, the data points belonging to the same centroid will have equal approximate distances to the query.

$$D(A2, Q2) = D(B2, Q2) = D(D2, Q2) = D(E2, Q2)$$

# Example (asymmetric distance)

Compression results

A -> [1, 2]

B -> [1, 2]

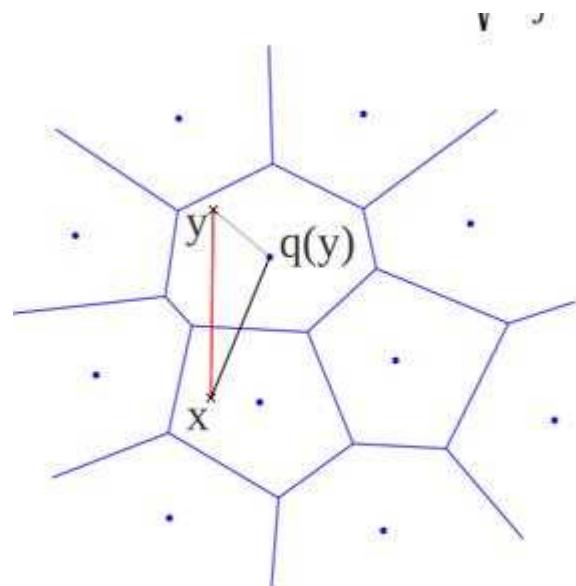
D -> [1, 2]

E -> [1, 2]

As they are all compressed to the same vector, we have Approximate Squared L<sub>2</sub> Distances (Asymmetric):

$$D(Q, A) = D(Q, B) = D(Q, D) = D(Q, E)$$

$$\tilde{d}(x, y) = d(x, q(y))$$



asymmetric case

# Example (asymmetric distance)

Subspace 1:

Q <sub>1</sub>	[16, 4]
----------------	---------

C1	[8,6]
C2	[18,6]

Subspace 2:

Q <sub>2</sub>	[4, 8]
----------------	--------

C1	[10, 12]
C2	[10, 4]

Compressed  
vectors:

$$A \rightarrow [1, 2]$$

$$B \rightarrow [1, 2]$$

$$D \rightarrow [1, 2]$$

$$E \rightarrow [1, 2]$$

Subspace 1:

$$D_1(Q_1, \text{Centroid 1}) = 8^2 + 2^2 = 68$$

Subspace 2:

$$D_2(Q_2, \text{Centroid 2}) = 6^2 + 4^2 = 52$$

Approximate Squared L<sub>2</sub> Distances (Asymmetric):

$$D(Q, A) = D(Q, B) = D(Q, D) = D(Q, E) = D_1 + D_2 = 120$$

# Example (asymmetric distance)

- Asymmetric distance calculation is efficient
- General case:
  - For one subspace, we only need to calculate K approximate distances. K is the number of centroids in one subspace.
  - K is much less than the total number of data points in the database. E.g., K=128 centroids, data points: 10 million

# Asymmetric distance

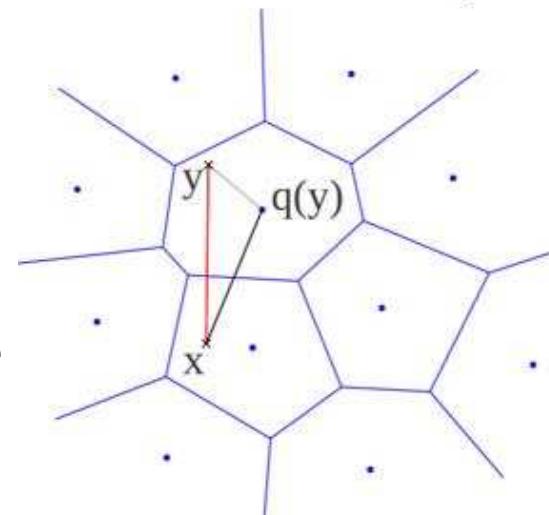
- Given a query in a top-k retrieval task, we don't need to compute the distance between the query and all the samples in the database.

We only need to calculate the distance between the query and all the centroids in each subspace.

- The number of centroids are much less than the number of samples in the database, so we can speed up the retrieval process.

- We don't need to quantize  $x$ , which reduces computation compared to the symmetric case.

$$\tilde{d}(x, y) = d(x, q(y))$$



asymmetric case

# Asymmetric distance

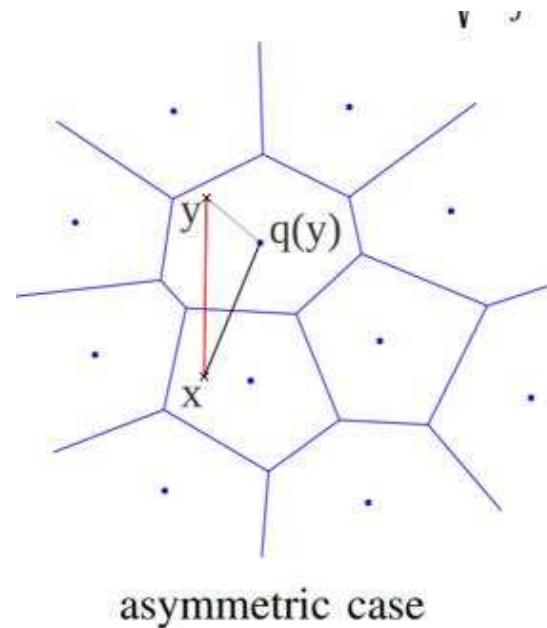
4. For each subspace, we compute the squared L<sub>2</sub> distances between the  $x$  and all centroids.

This bring extra computation compared to the symmetric case.

5. The above extra computation cost is similar with the quantization cost for  $x$ . (cluster assignment requires distance calculation between  $x$  and all centroids)

That means symmetric and asymmetric distance have the same computation complexity.

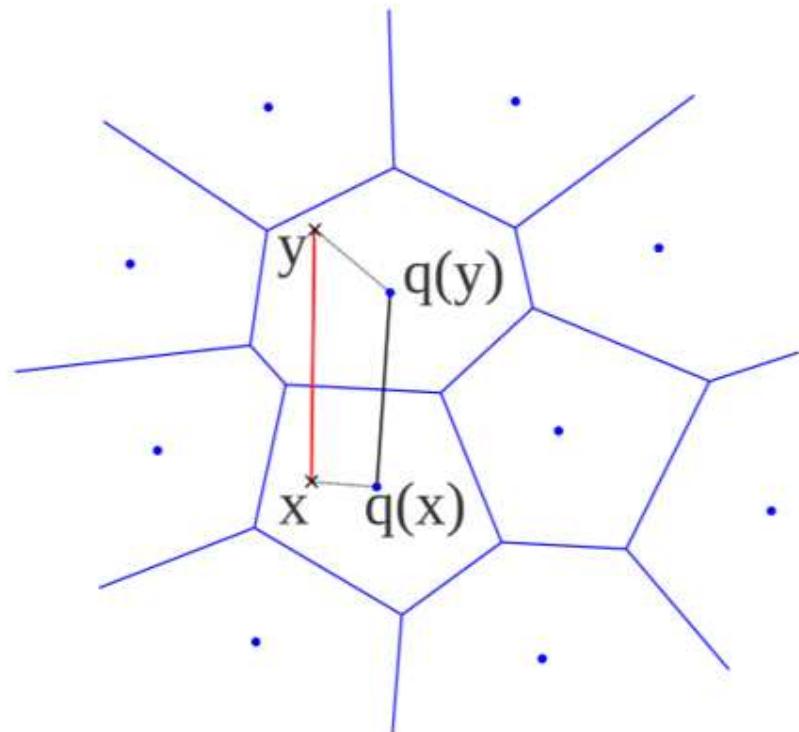
$$\tilde{d}(x, y) = d(x, q(y))$$



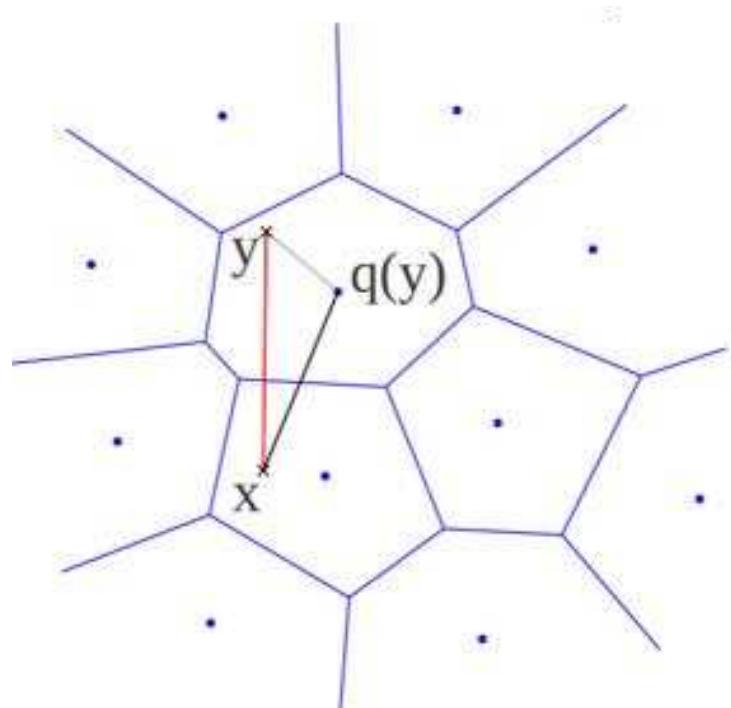
# Comparison

$$\hat{d}(x, y) = d(q(x), q(y)) = \sqrt{\sum_j d(q_j(x), q_j(y))^2}, \dots$$

$$\tilde{d}(x, y) = d(x, q(y))$$

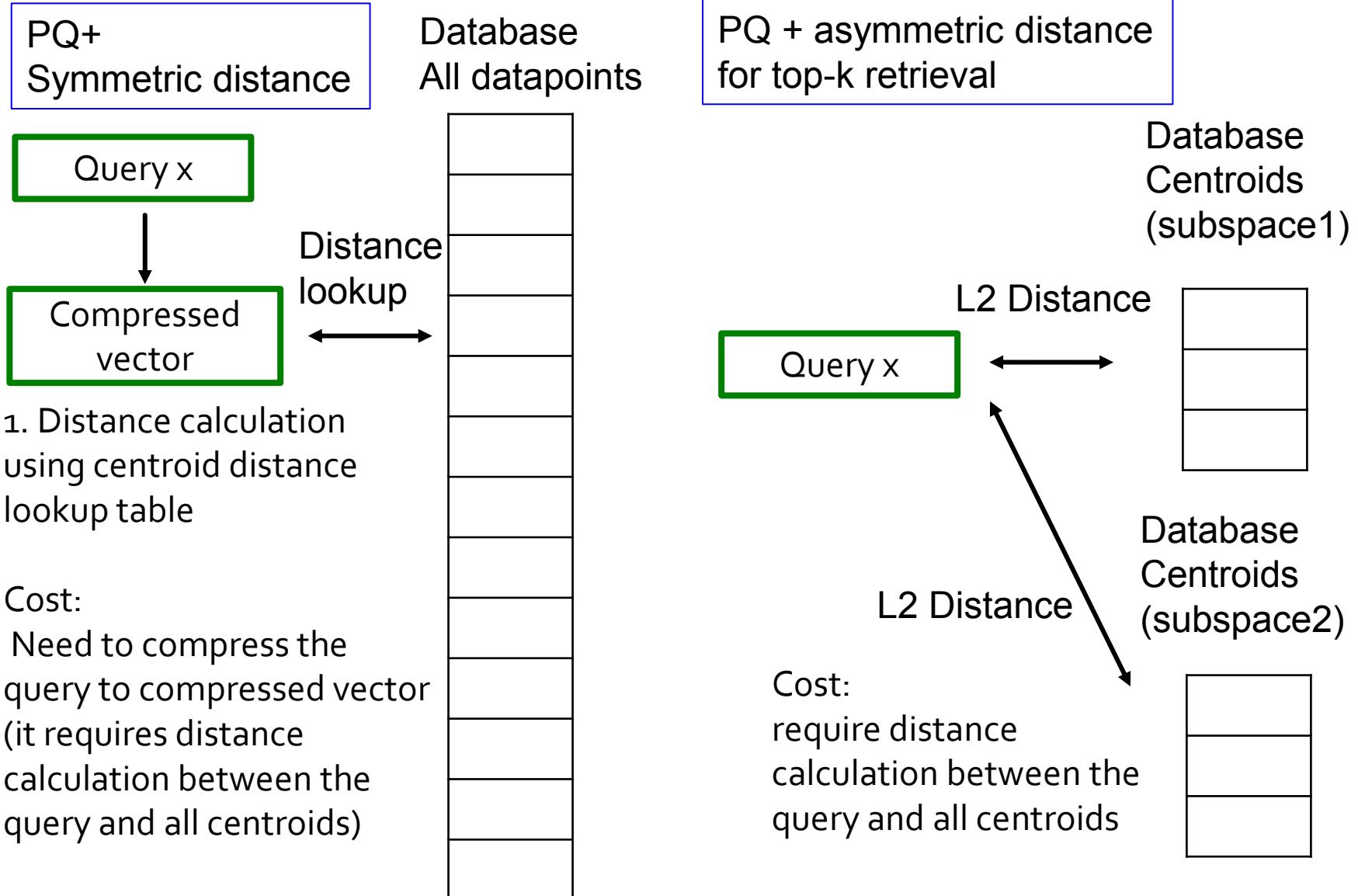


symmetric case



asymmetric case

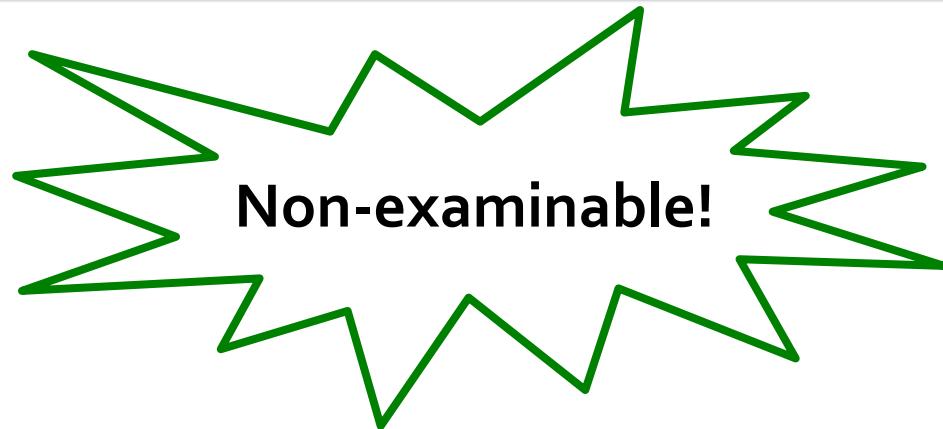
# Comparison for top-k search



# Outline

- Nearest Neighbour Search (NNS) Problems
- Local Sensitive Hashing (LSH) – Random Projection
- Product Quantization (PQ)
- **Inverted File Index (non-examinable!!)**

# Inverted File Index



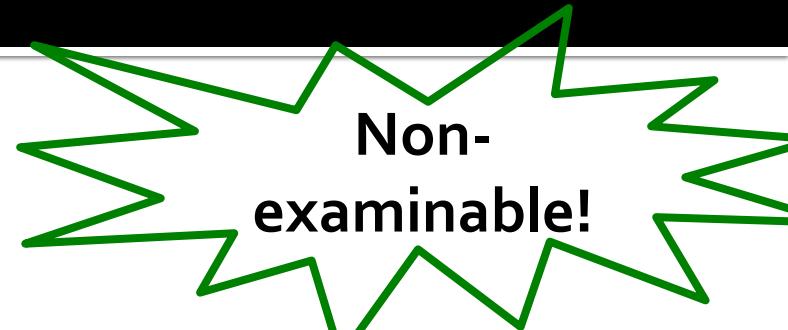
- Inverted File Index
  - Clustering based method
  - Data dependent
  - Reduce the search scope

## ■ Inverted File Index (IVT)

- An inverted index here means a mapping (a lookup table) from cluster centroids (words) to the cluster members.

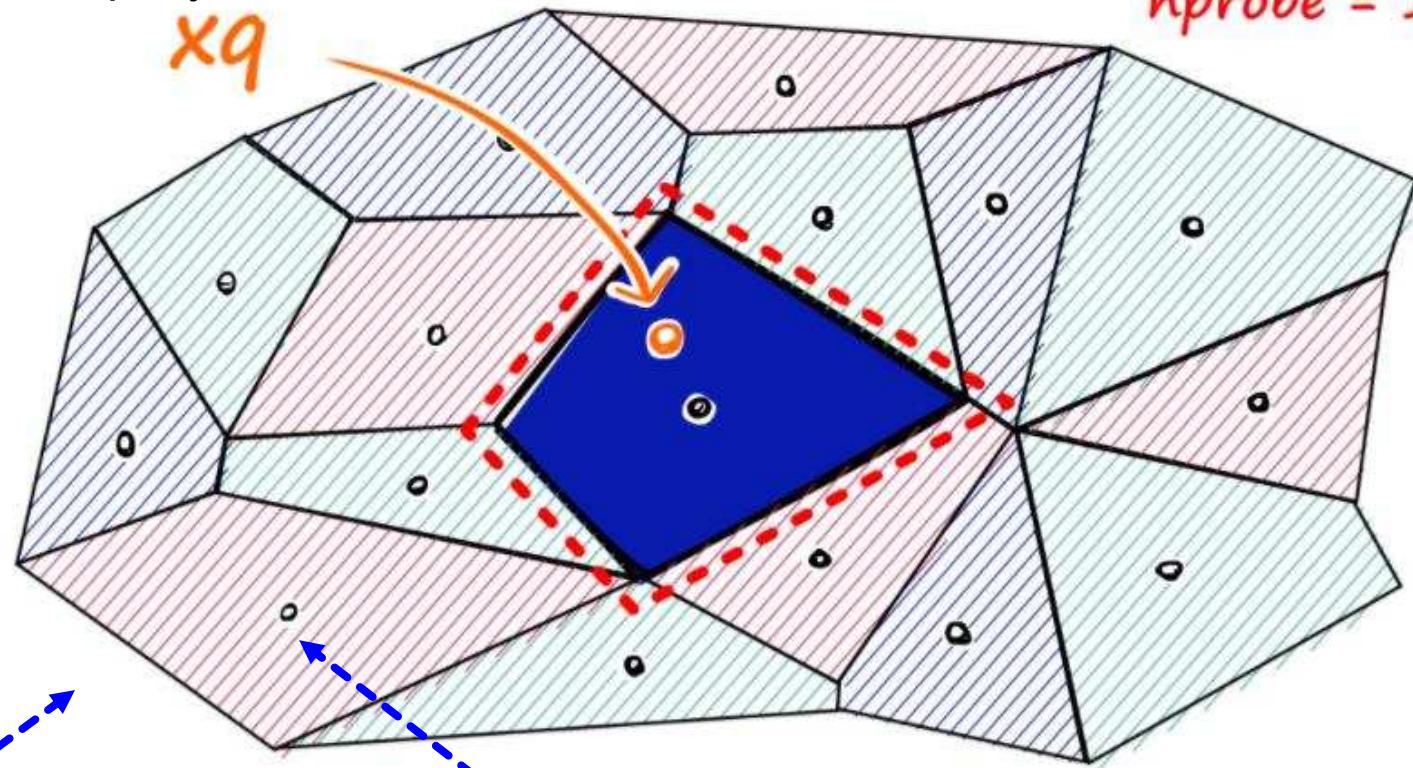
## ■ Steps:

- 1. Clustering of the samples in the database, generate the centroids and cells
- 2. Given a query vector, identify the cells for search scope
- 3. Use linear search (exhaustive search) to retrieve results within the selected cells



$xq$ : the query vector

search scope  
 $nprobe = 1$

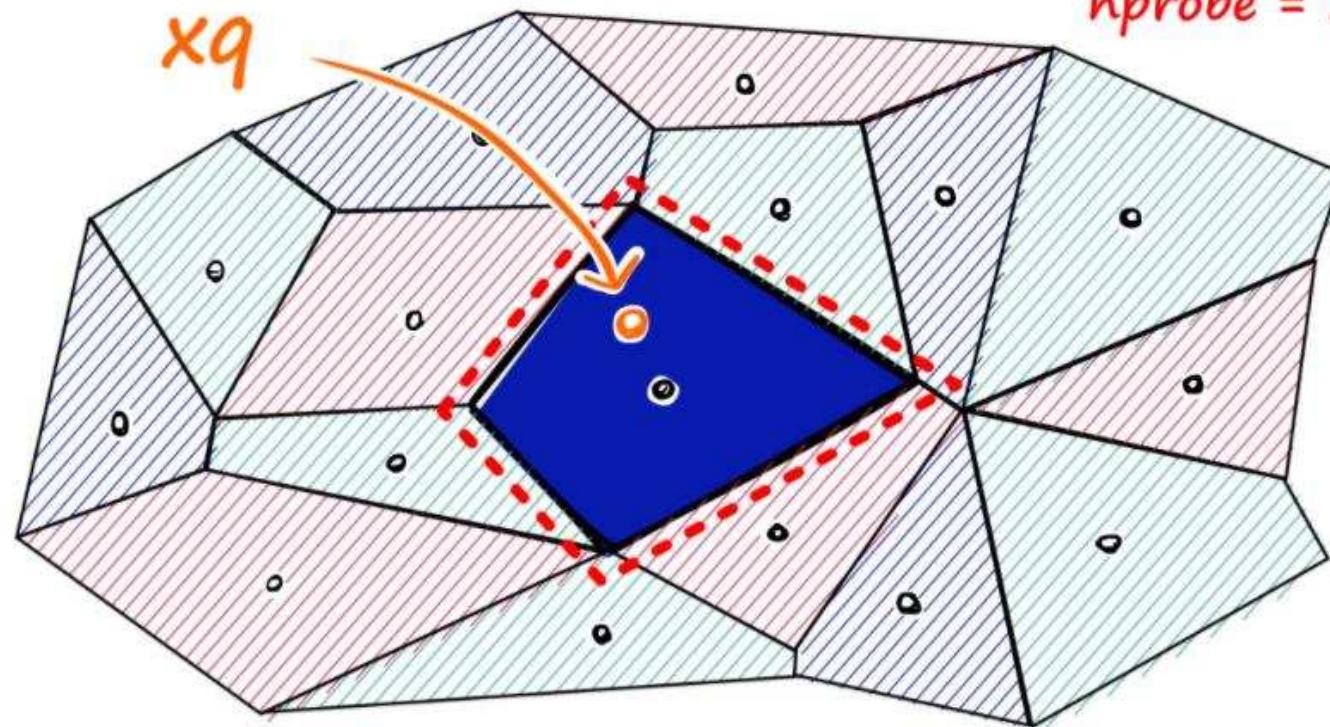


Perform k-means clustering  
to group the data points in  
the database (cluster  
centroids and assignments)

Cluster centroid in one cell.  
One cell represents one cluster

$xq$ : the query vector

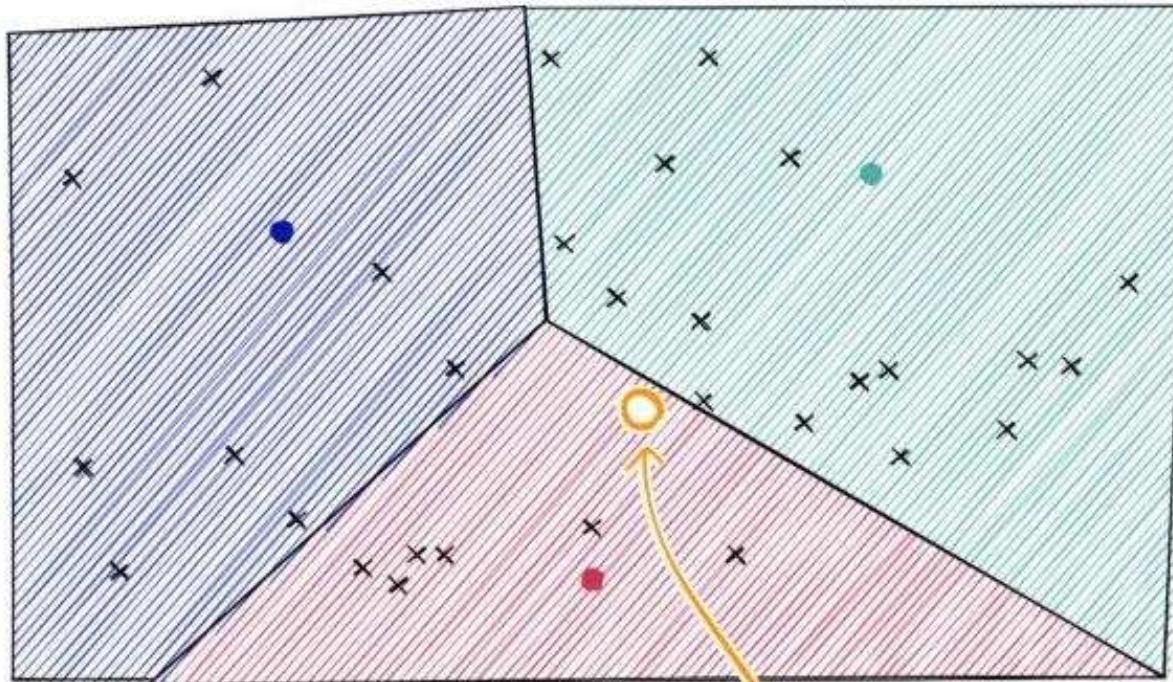
search scope  
 $nprobe = 1$



$nprobe$ : select the top  $k$  nearest clusters (cells) as the search scope  
 $nprobe=1$ : only select 1 cell for searching

There are 2 parameters in IVT:

1.  $nprobe$ : the number of cells to search
2. The number of cells (clusters) to create.



## Edge problem

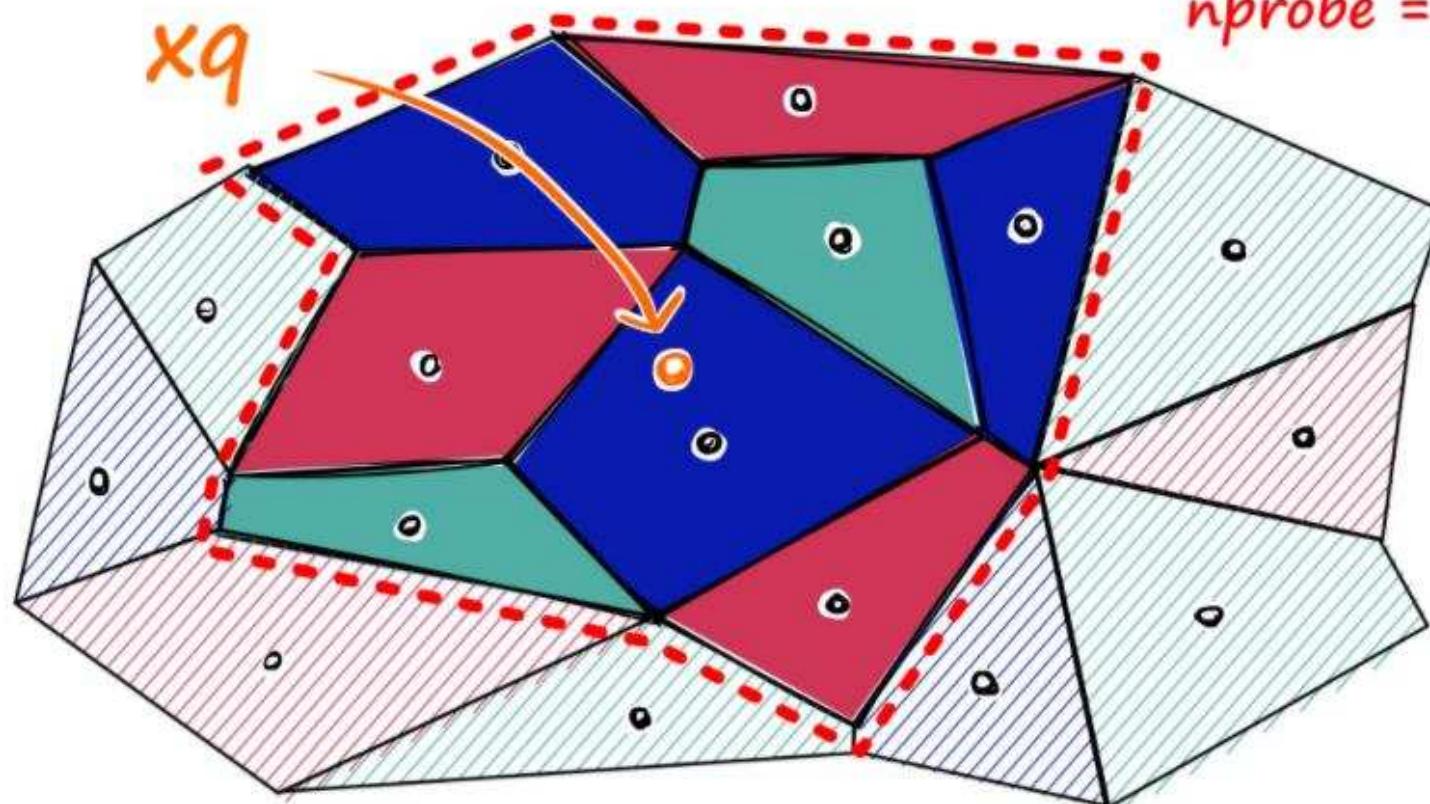
*query vector*

with `nprobe == 1`, scope restricted  
to magenta cell only

Our query vector `xq` lands on the edge of the magenta cell. Despite being closer to datapoints in the teal cell, we will not compare these if `nprobe == 1` — as this means we would restrict search scope to the magenta cell only.

$xq$ : the query vector

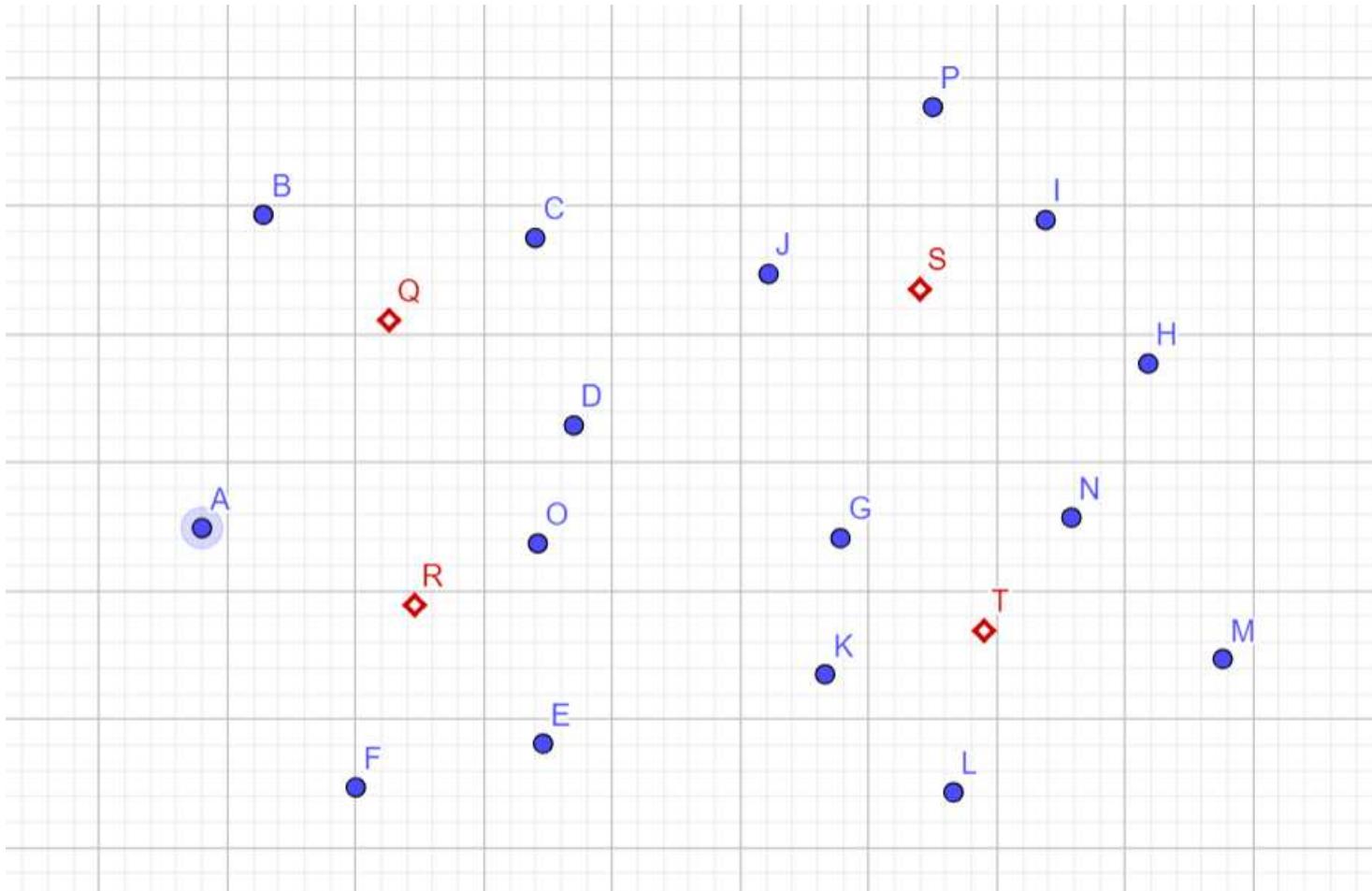
search scope  
 $nprobe = 8$



Increasing **nprobe** increases our search scope.

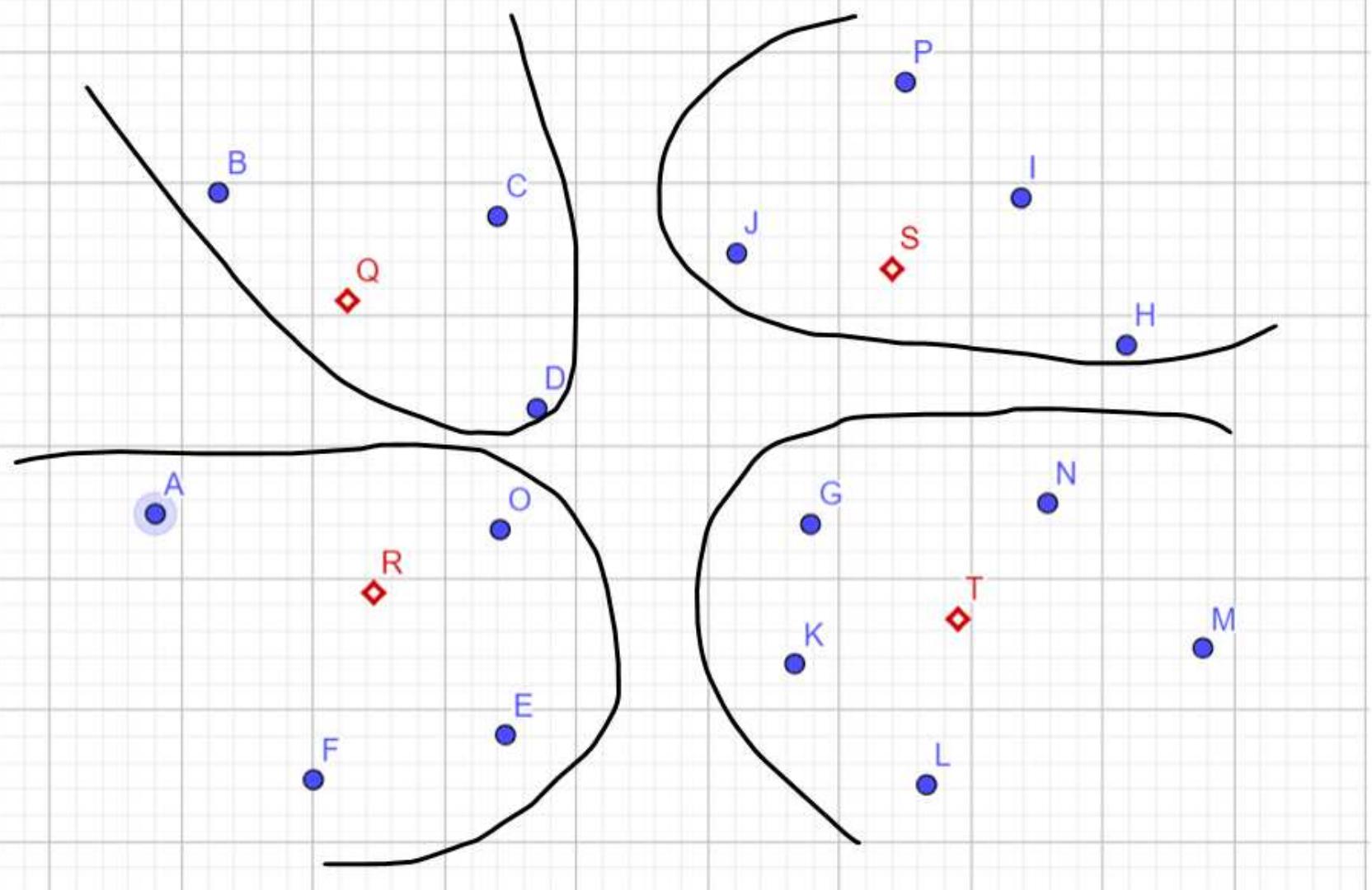
$nprobe=8$ : select the top 8 nearby cells for searching.  
(ranked by the distance between the query and the centroids)

## ■ Inverted File Index Example

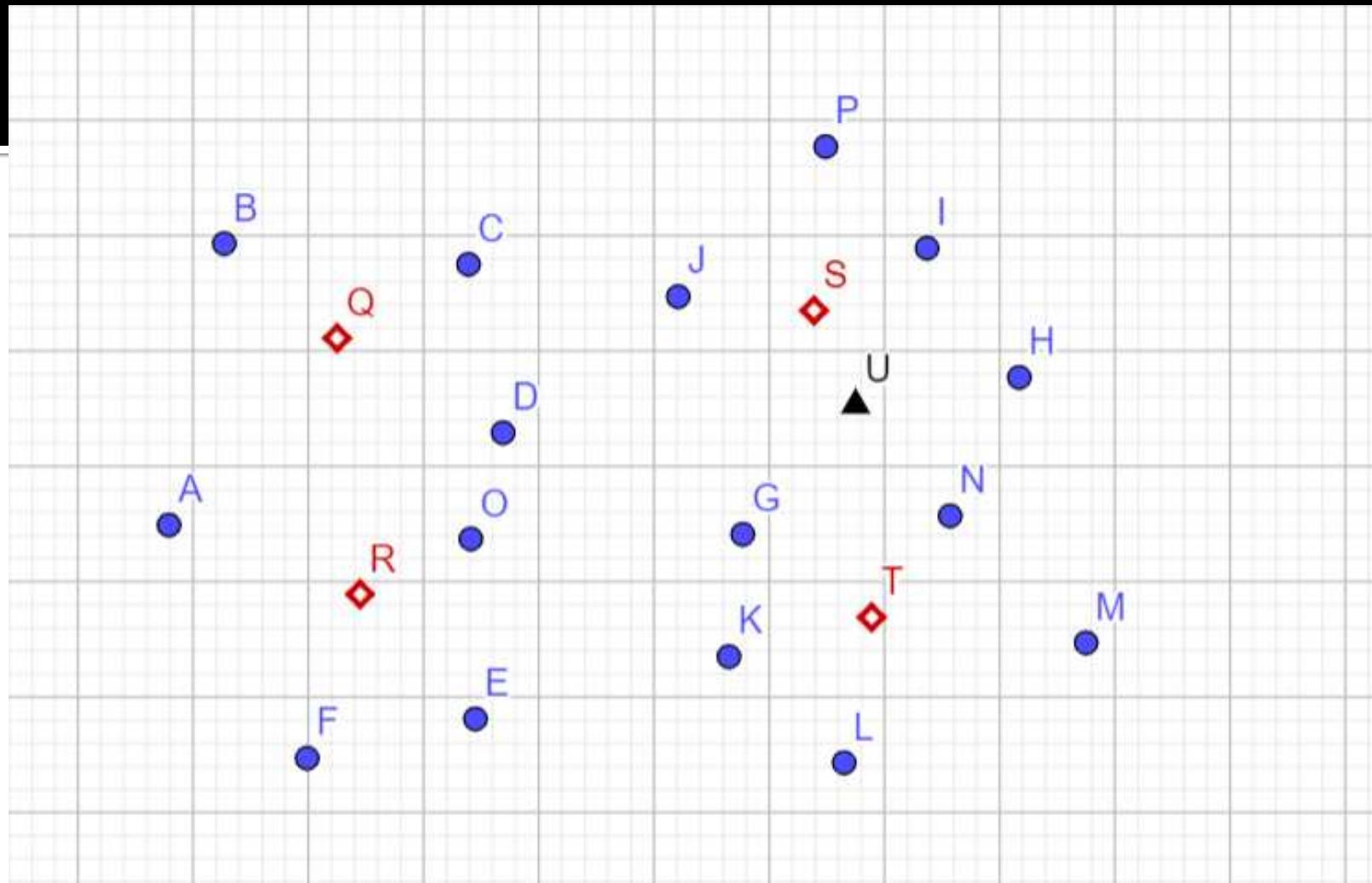


Q, R, S, T are centroids, all other points are the data samples in the database.

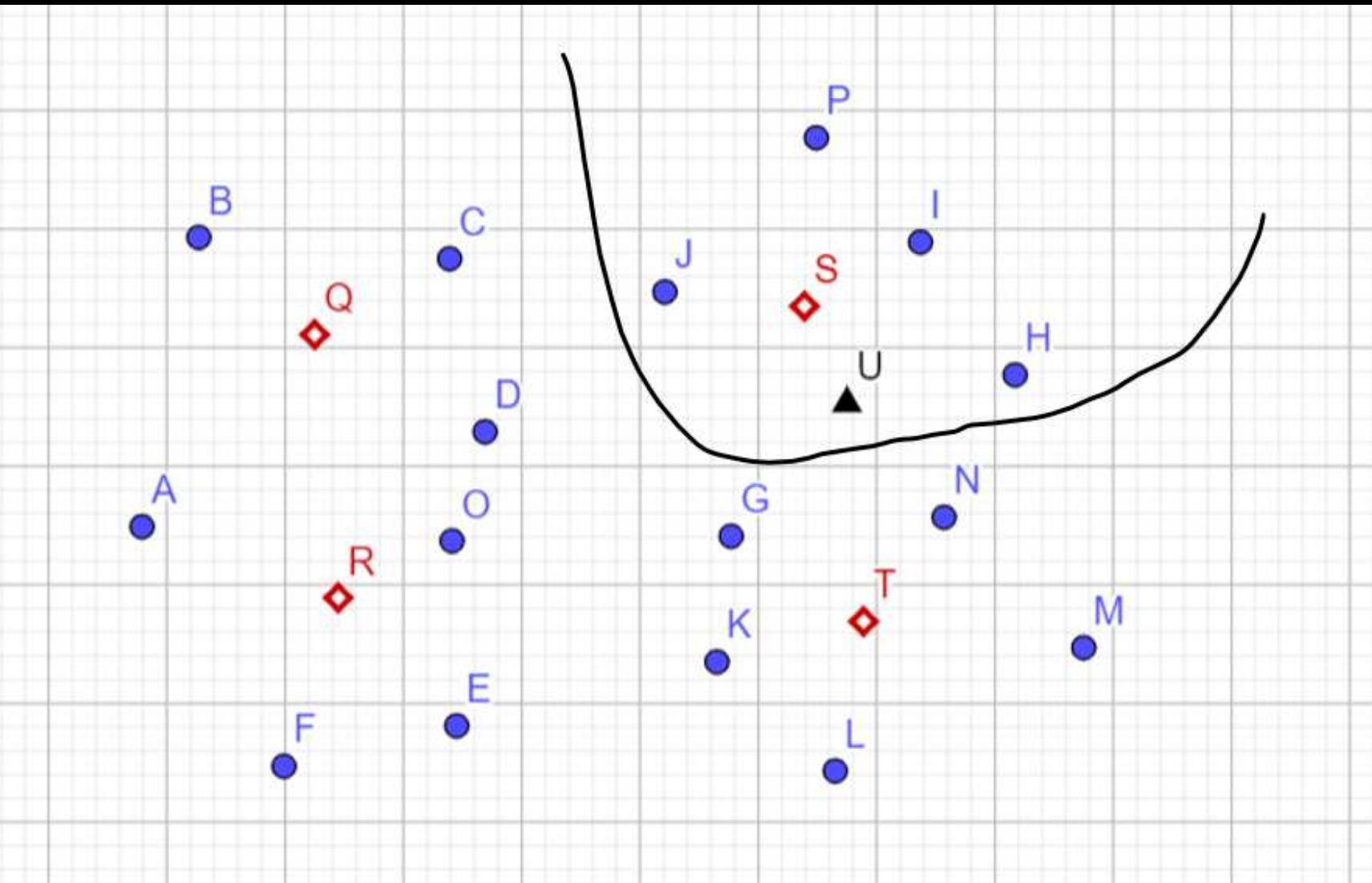
- Q(a): assign the samples to the cells defined by the centroids.



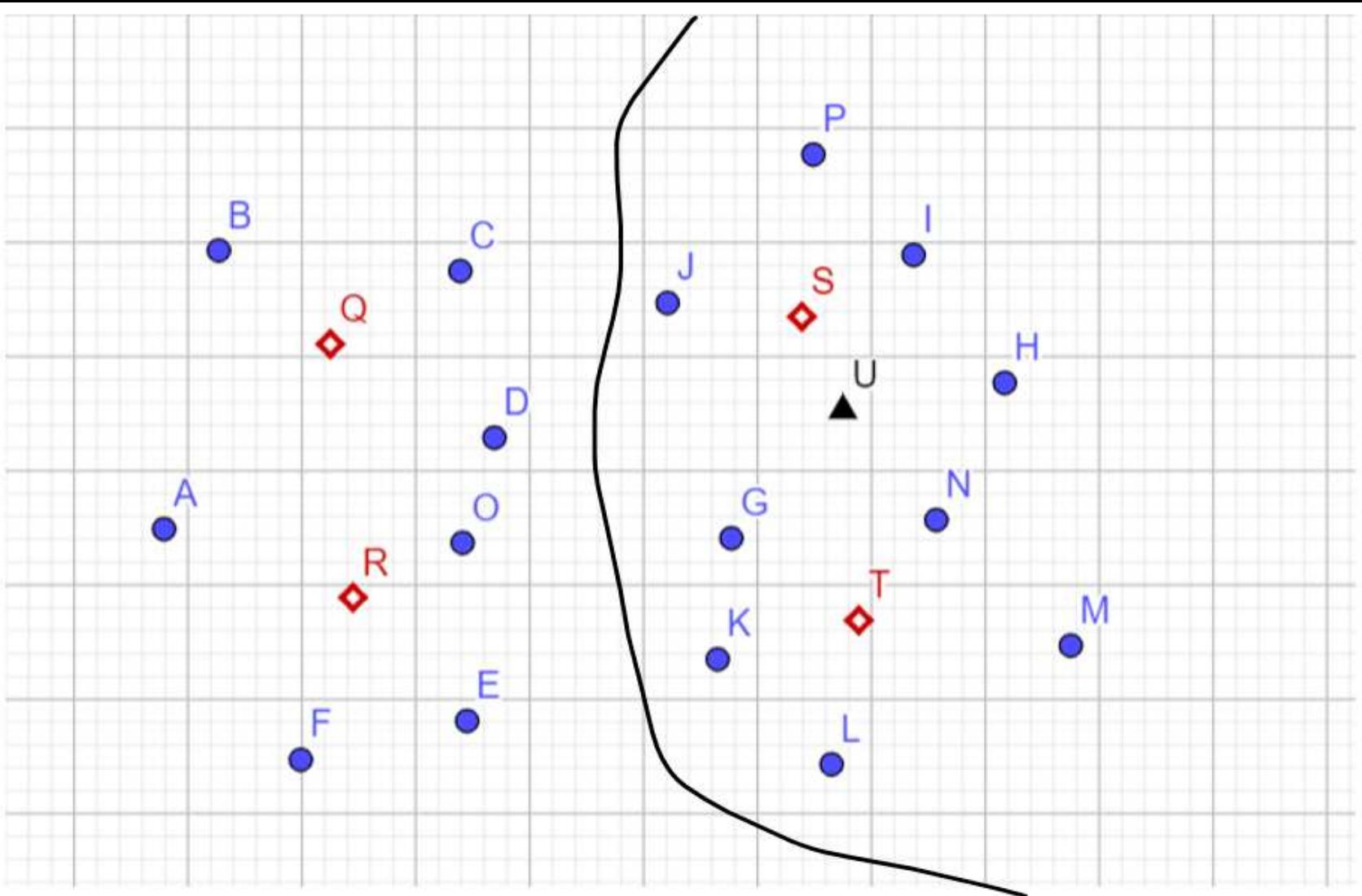
one data point is assigned to its nearest centroid based on L2 distance



Q(b) Given the query sample U,  
list the candidate samples in the database for similar search  
when nProbe=1 and nProbe=2, respectively



nProbe=1: only search in 1 cell

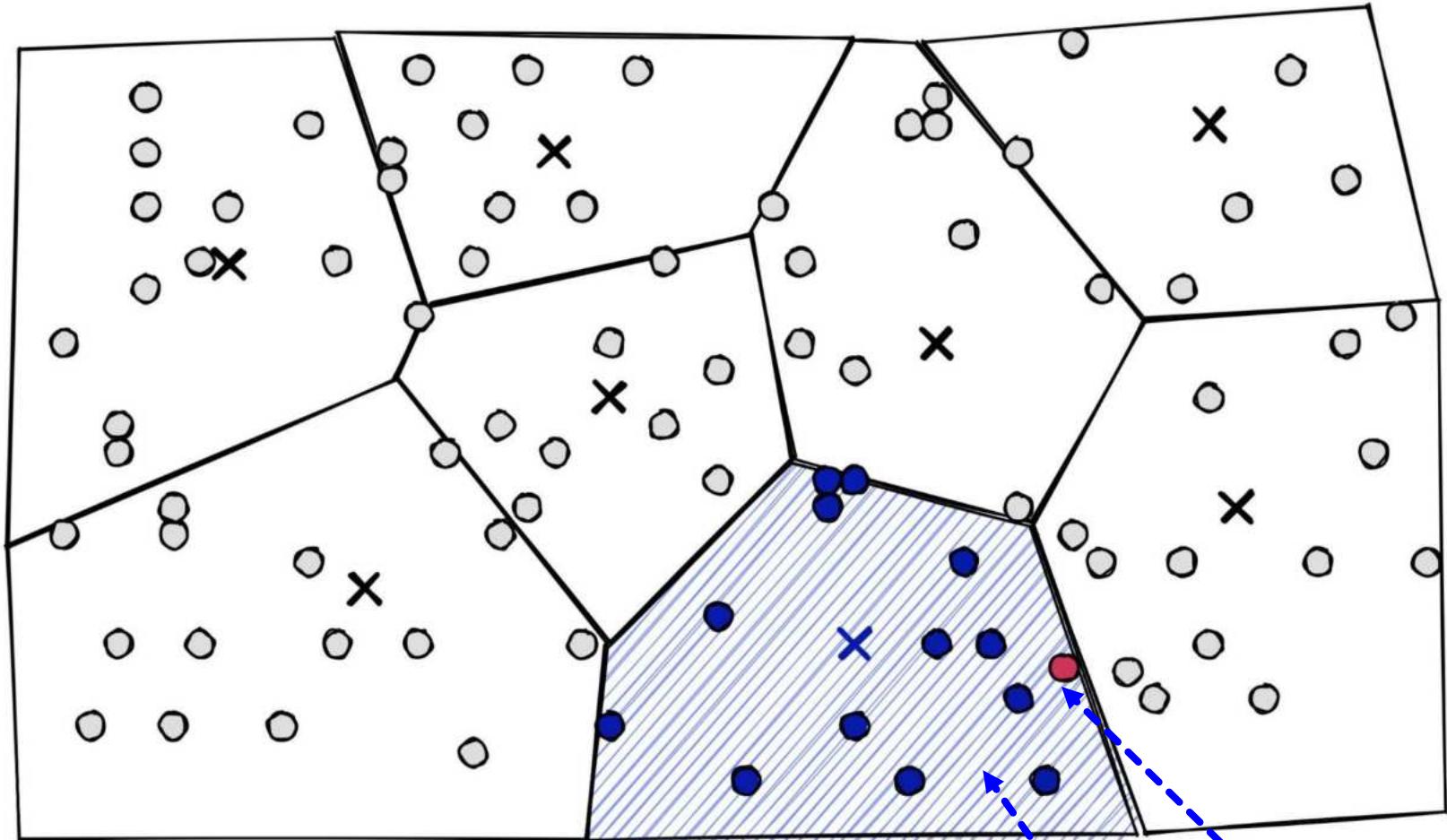


nProbe=2: search in 2 cells (the top 2 closest cells)

# PQ+IVF



- PQ+IVF
  - Product Quantization + Inverted File Index
  - A hierarchical solution
  - Much better performance in searching
  - Reduce search scope by using IVF



Step1: use IVF to identify search scope (cells).

IVF allows us to restrict our search in the target cells only.

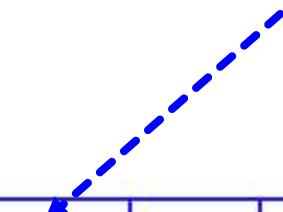
Target cell

Step2: perform PQ based linear search in the target cells. PQ generates compressed vectors and uses fast distance calculation

query

Experiment:  
Sift1M dataset, 2048 centroids for each subspace

L2 distance base linear search



	FlatL2	PQ	IVFPQ
<b>Recall (%)</b>	100	50	52
<b>Speed (ms)</b>	8.26	1.49	0.09
<b>Memory (MB)</b>	256	6.5	9.2

<https://www.pinecone.io/learn/product-quantization/>

# Online resources

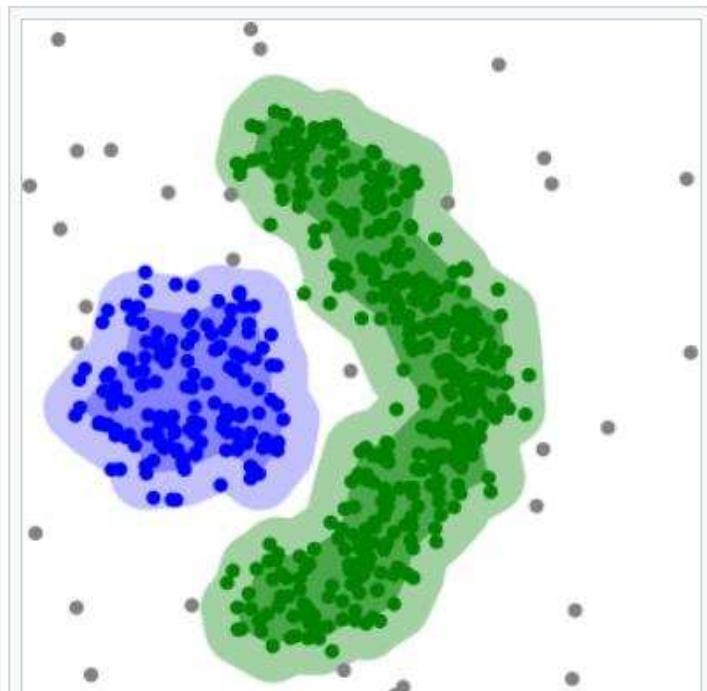
- FAISS
  - Faiss is a library for efficient similarity search and clustering of dense vectors.
  - <https://github.com/facebookresearch/faiss/wiki>

# Clustering - DBSCAN

Lin Guosheng  
School of Computer Science and Engineering  
Nanyang Technological University

# DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)



DBSCAN can find non-linearly  
separable clusters. This dataset cannot  
be adequately clustered with k-means  
or Gaussian Mixture EM clustering.

# DBSCAN

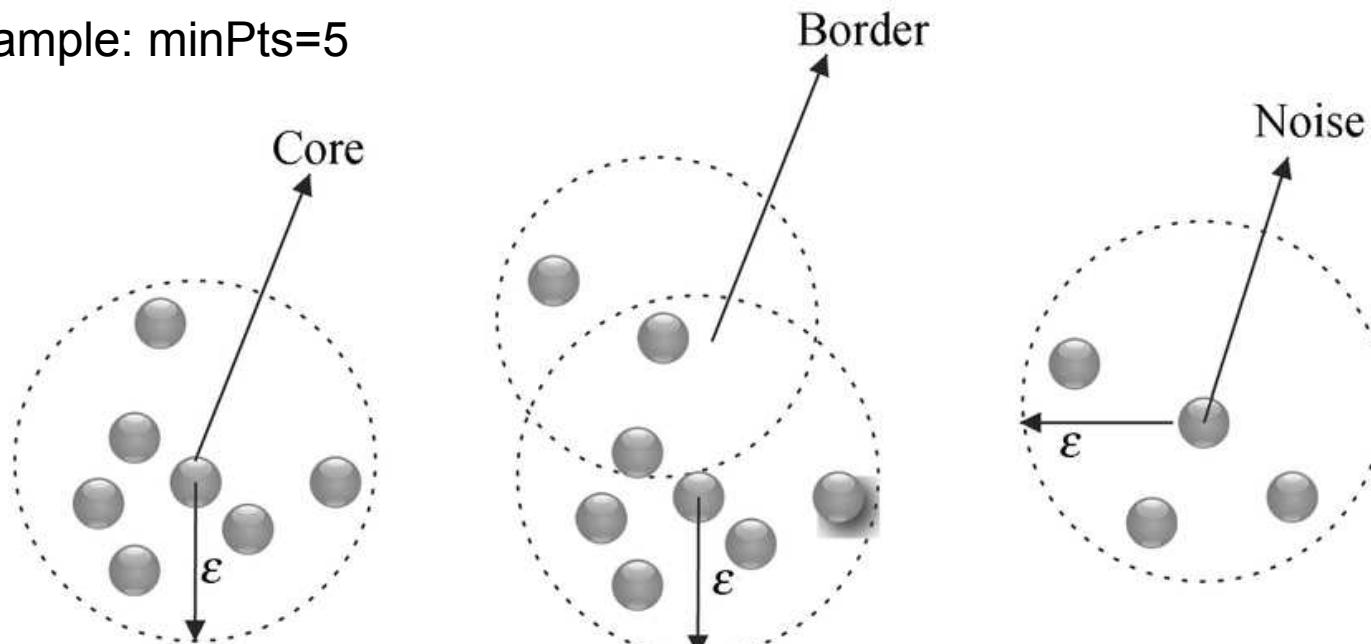
There are two key parameters of DBSCAN:

- `epsilon_radius` (`epsilon` or `eps`):  
This radius defines the epsilon-neighbourhoods. Two points are considered to be neighbours if the distance between them are less than or equal to `eps`.
- `minPts`: Minimum number of data points to define a cluster (or a core point).

<https://towardsdatascience.com/dbSCAN-clustering-explained-97556a2ad556>

- **Core point:** A point is a core point if there are at least `minPts` number of points (including the point itself) in its surrounding area with radius  $\text{eps}$  (within the epsilon-neighbourhood).
- **Border point:** A point is a border point if it is not a core point and it is directly reachable from a core point.
- **Outlier (noise point):** A point is an outlier if it is not a core point and not reachable from any core points.

Example: `minPts=5`



# Algorithm

- Step 1. Start a cluster.
  - An unvisited point  $x$  is selected at random. If  $x$  is a core point, construct a new cluster starting from  $x$  using Step 2. Otherwise, the point  $x$  is marked as a noise point; mark the point as visited; jump to step 1.
- Step 2. Construct a cluster (cluster expansion)
  - construct the cluster using breadth-first search
  - Mark all points in the cluster as visited.
- Stop until all points are visited.

# Cluster creation

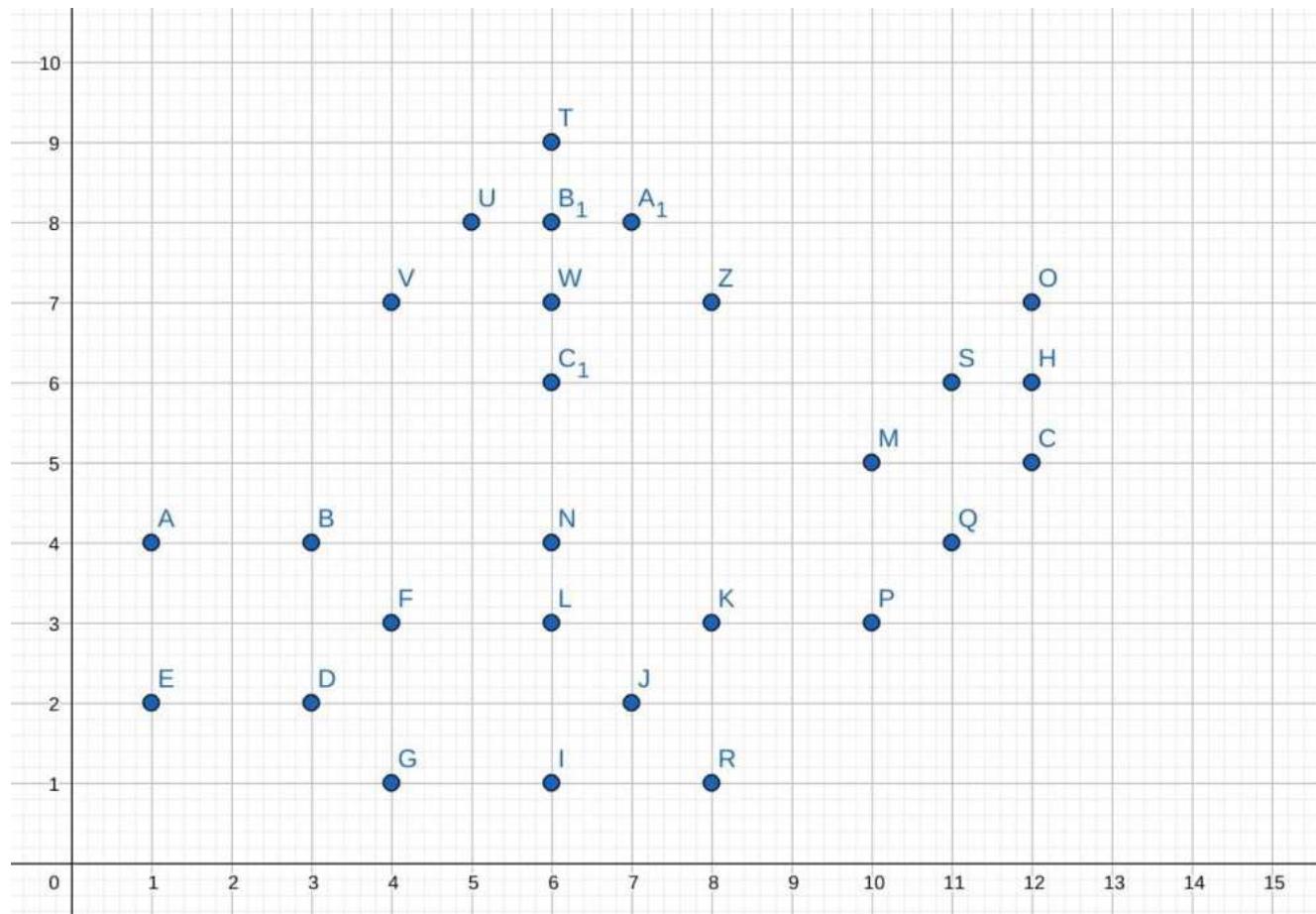
- Create the cluster as building a tree
  - 1) initialization:
    - The starting core point is the root node of the tree
    - Create an empty queue, add the root node into the queue
      - The queue is to implement breadth-first search (layer-wise traversal) of the tree. (walk through all nodes on the same level before moving on to the next level).
      - The queue is to maintain a list of TODO nodes.

# Cluster creation

- Create the cluster as building a tree
  - 2) dequeue to get the target node:
    - if the target node is a core point:
      - Identify all neighbouring points (directly reachable nodes).
      - If a neighbouring node is not in the tree
        - Tree update: add it as a child node under the target node.
        - Implementation: add it to the queue for future processing
  - 3) Goto step 2) to process the next node
    - Stop expanding the tree if the queue is empty.
  - All the nodes in the tree are the members of the cluster.

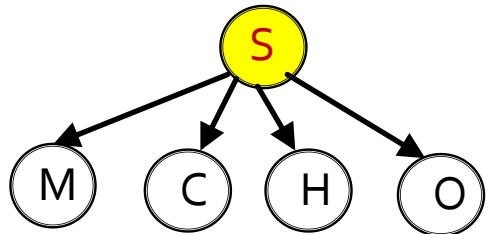
# DBSCAN Example

**Question:** given the data points in the figure and the DBSCAN parameters: **minPts=4, radius  $\text{eps}=1.5$** , illustrate the process of finding one cluster starting from point S. If cannot find a cluster starting from S, provide the discussion.

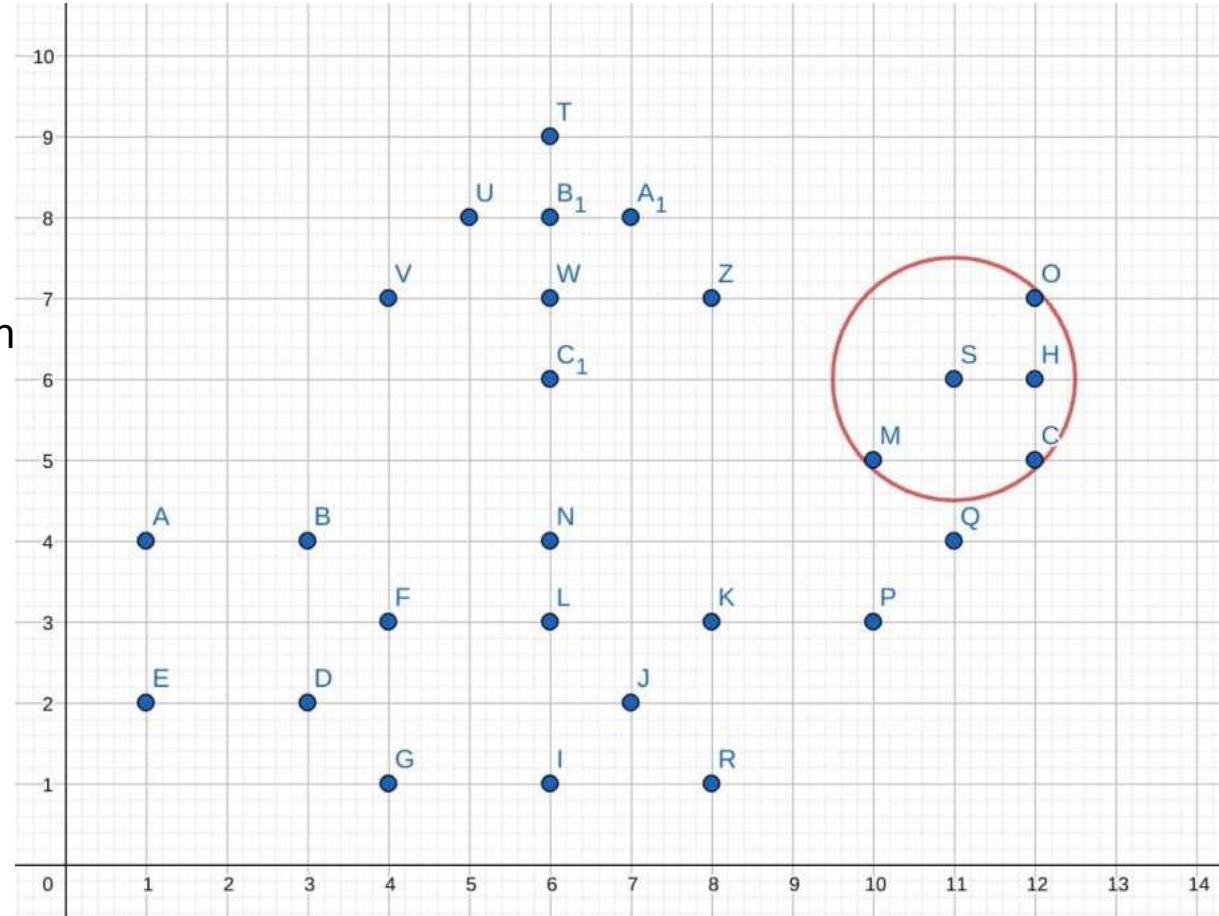


# Starting from S

1. check whether the point is a core point or not
2. If it is a core point, find out all neighbors. (directly reachable nodes). Add them to the tree as child nodes if they are not in the tree.



A tree describes a cluster  
Core points are highlighted



S is a core point,  
as the number of neighbors within `eps_radius`:  
 $5 \geq \text{minPts}$  (4) (including the point S itself!)

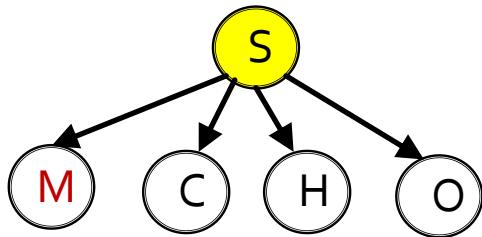
Queue: S

↓ Dequeue and add new nodes

Queue: M C H O

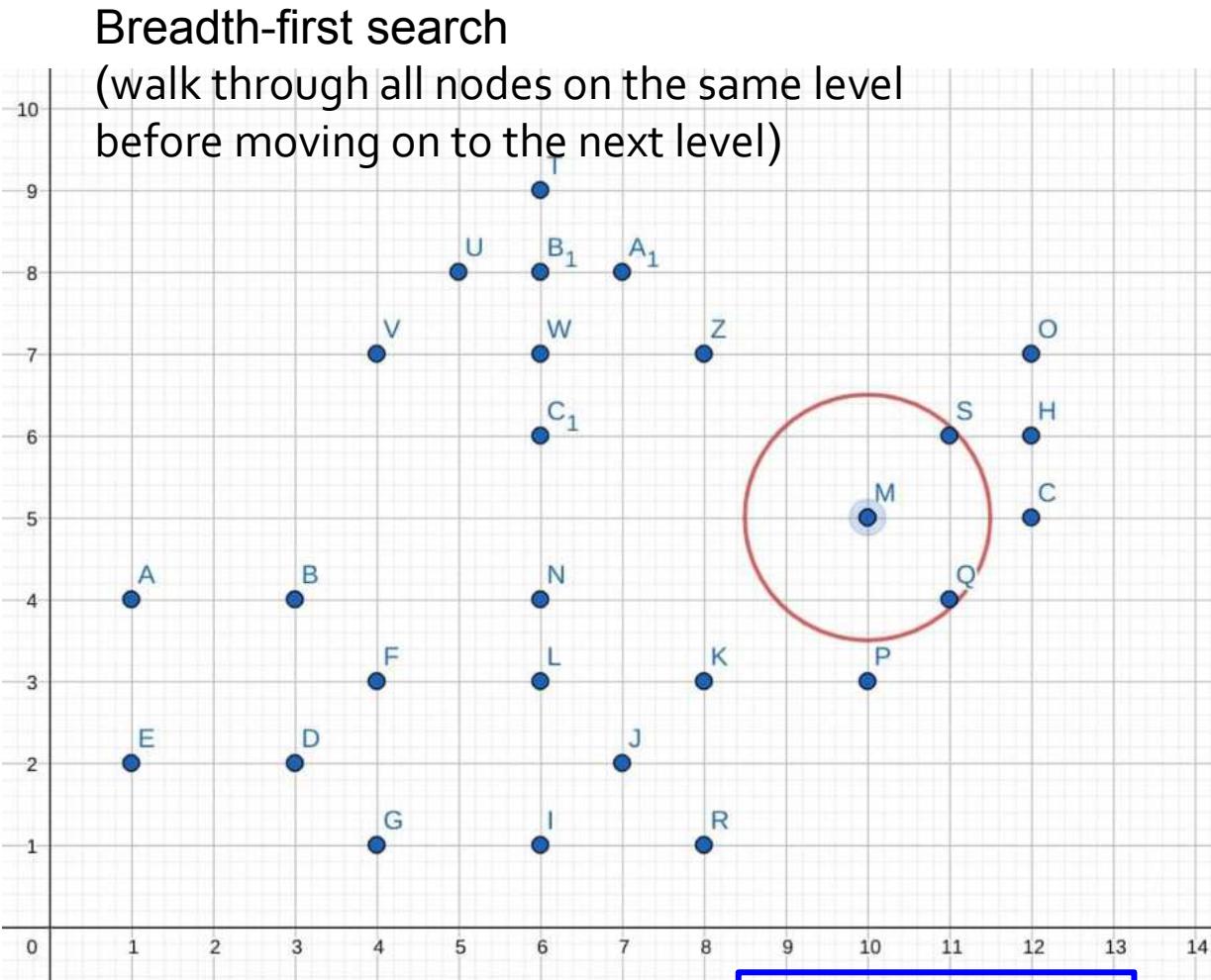
# Explore M

1. Determine whether the point is a core point or not



A tree describes a cluster

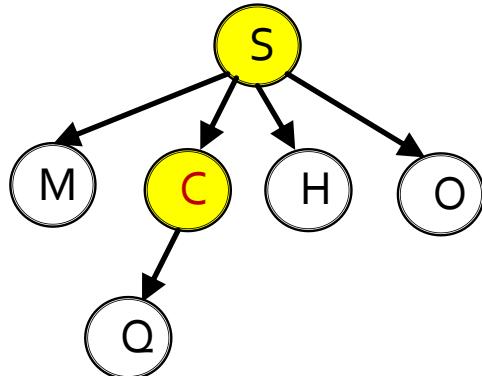
M is not a core point,  
as the number of neighbors within `eps_radius`:  $3 < \text{minPts}$  (4)



Queue: M C H O  
↓ Dequeue  
Queue: C H O

# Explore C

1. Determine whether the point is a core point or not
2. If it is a core point, find out all neighbors (directly reachable nodes). Add them to the tree as child nodes if they are not in the tree.

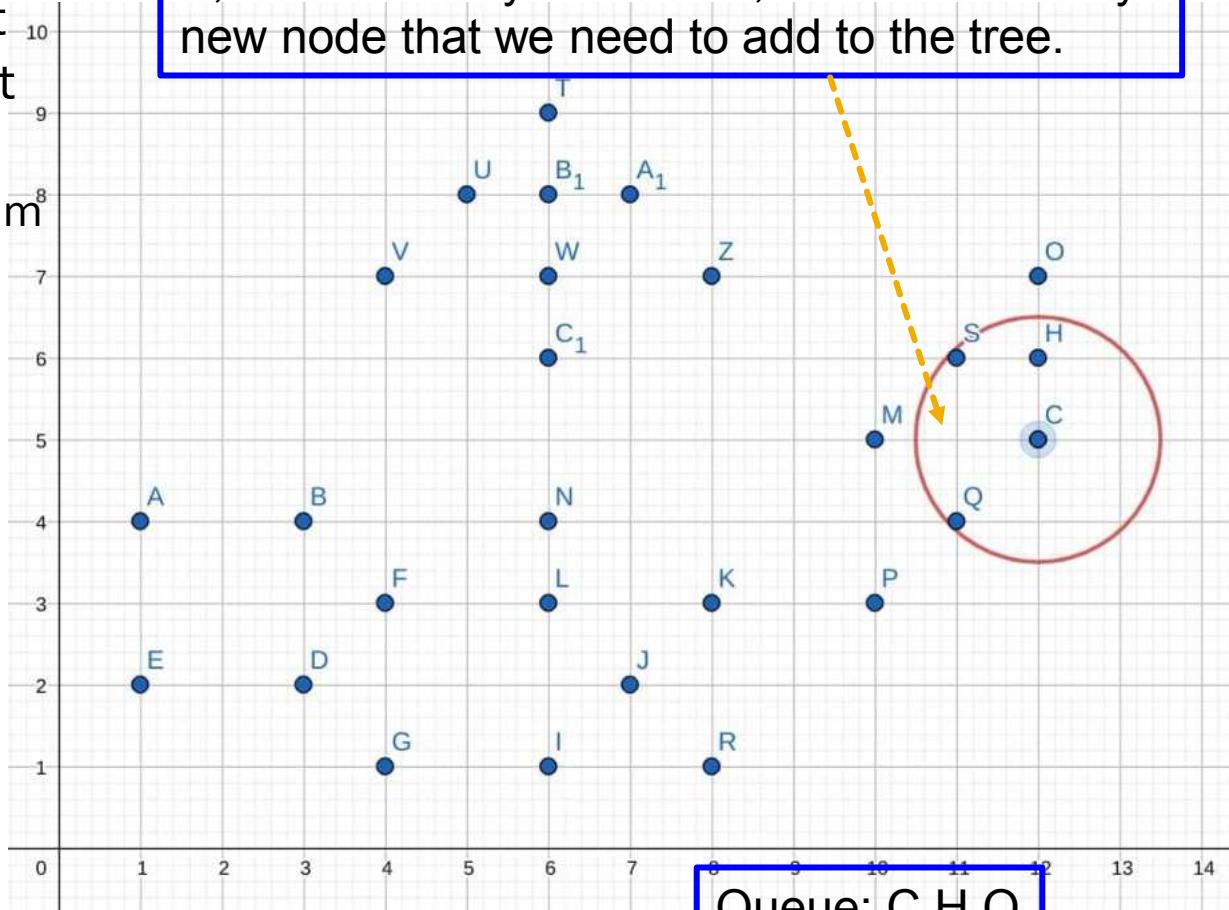


A tree describes a cluster

C is a core point, as the number of neighbors within `eps_radius: 4 >= minPts (4)` (including the point C itself!)

C has 3 neighbours: S, H, Q.

S, H are already in the tree, and Q is the only new node that we need to add to the tree.



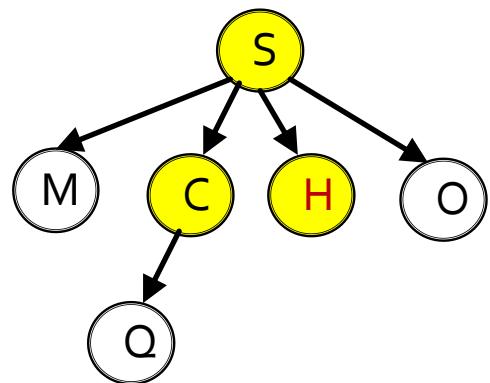
Queue: C H O

↓ Dequeue and add new nodes

Queue: H O Q

# Explore H

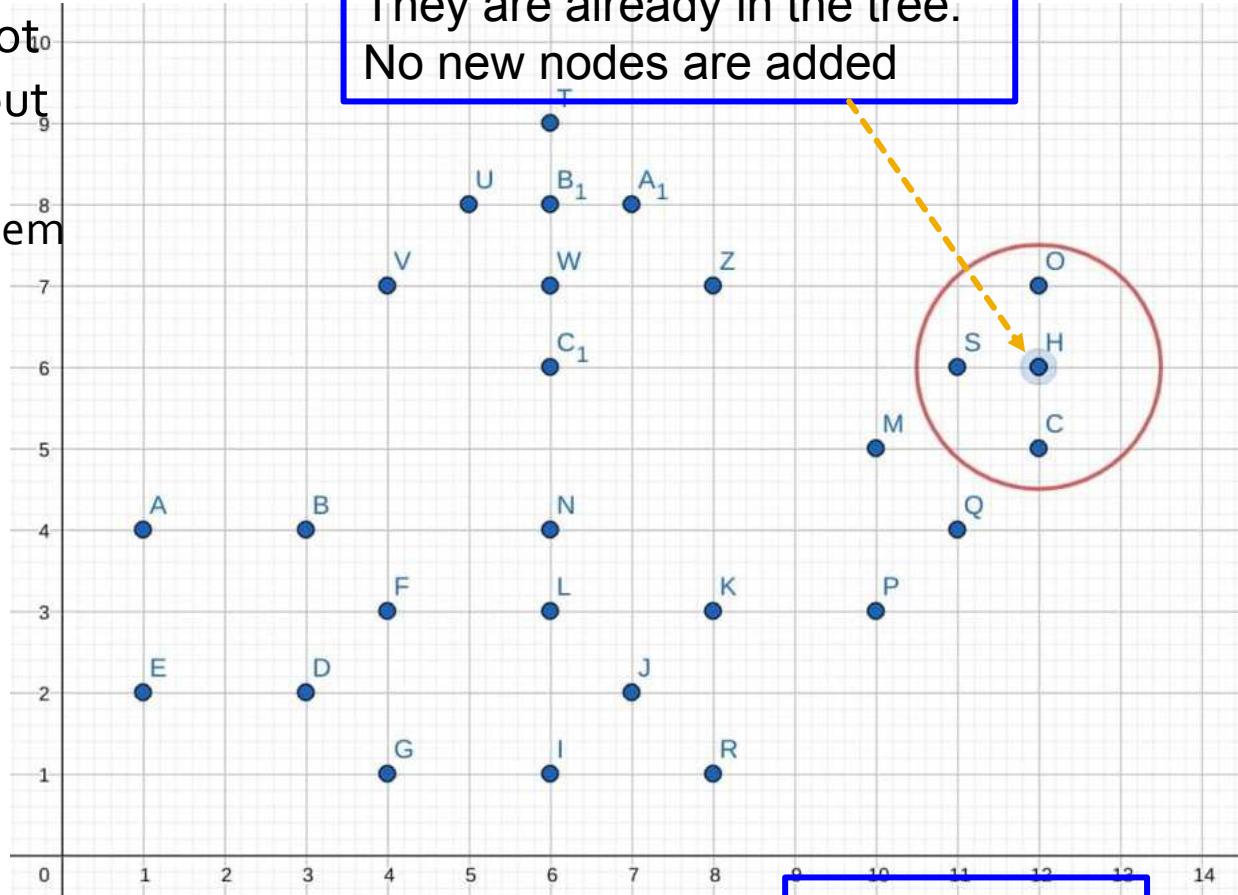
1. Determine whether the point is a core point or not
2. If it is a core point, find out all neighbors (directly reachable nodes). Add them to the tree as child nodes if they are not in the tree.



A tree describes a cluster

H is a core point,  
as the number of neighbors within  
eps\_radius:  $4 \geq \text{minPts}$  (4)

H has 3 neighbours: S, C, O.  
They are already in the tree.  
No new nodes are added



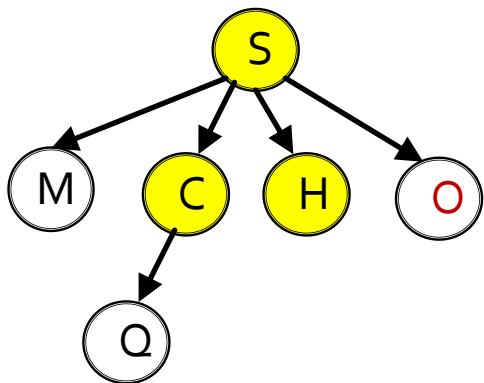
Queue: H O Q

↓ Dequeue

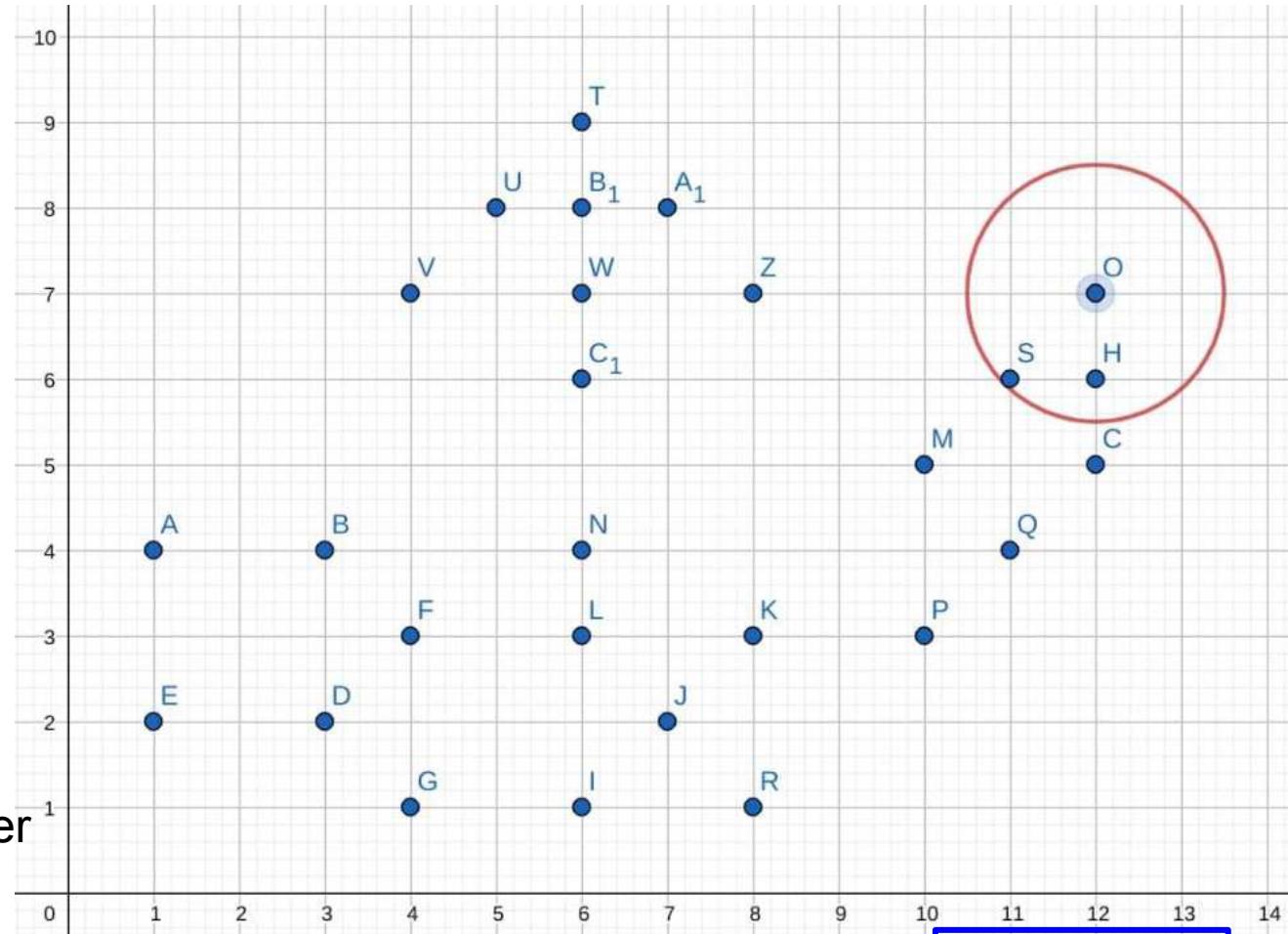
Queue: O Q

# Explore O

1. Determine whether the point is a core point or not



A tree describes a cluster



O is not a core point,  
as the number of neighbors within eps\_radius:  $3 < \text{minPts}$  (4)

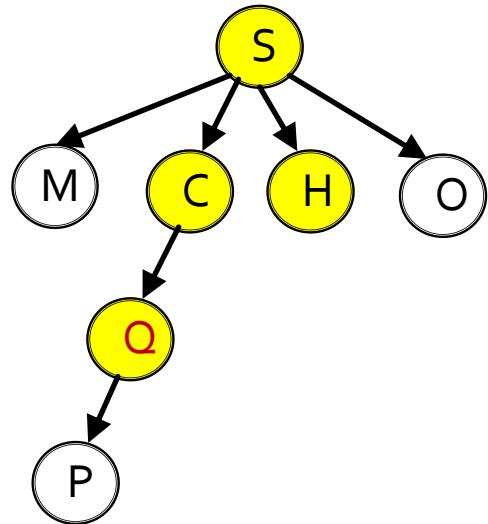
Queue: O Q

↓ Dequeue

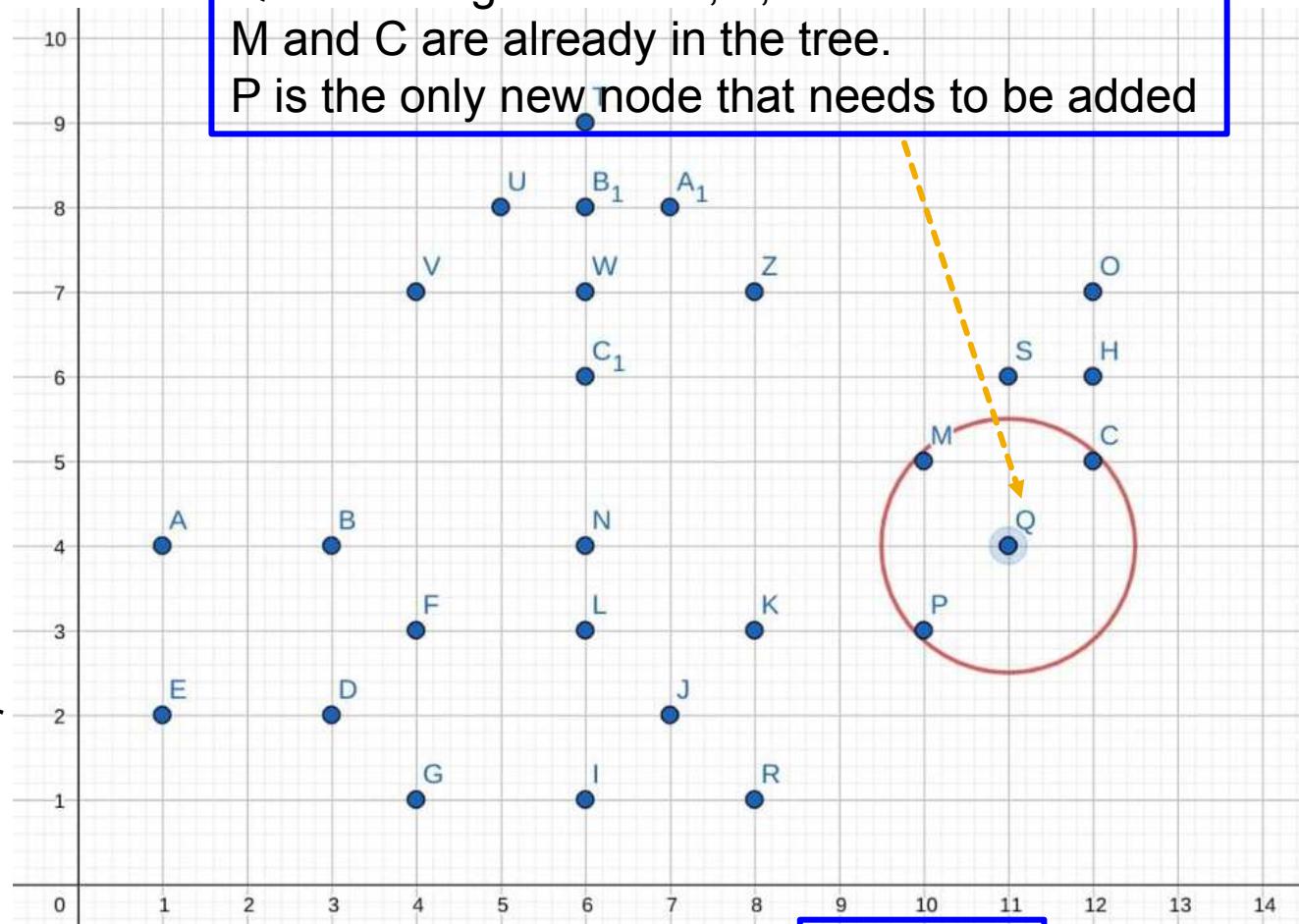
Queue: Q

# Explore Q

Proceed to the next layer: Q



A tree describes a cluster



Q is a core point,  
as the number of neighbors within  
eps\_radius: 4  $\geq$  minPts (4)

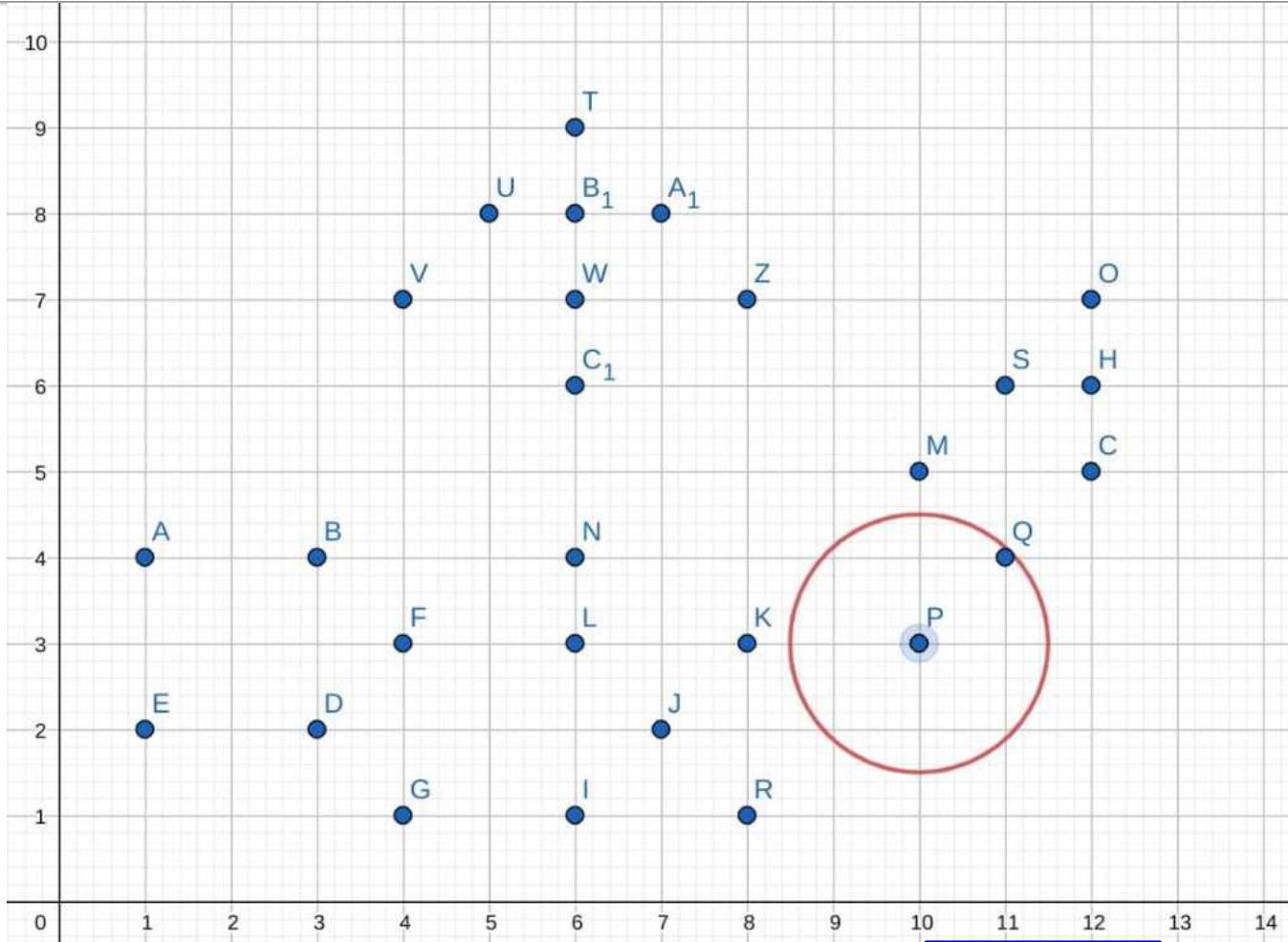
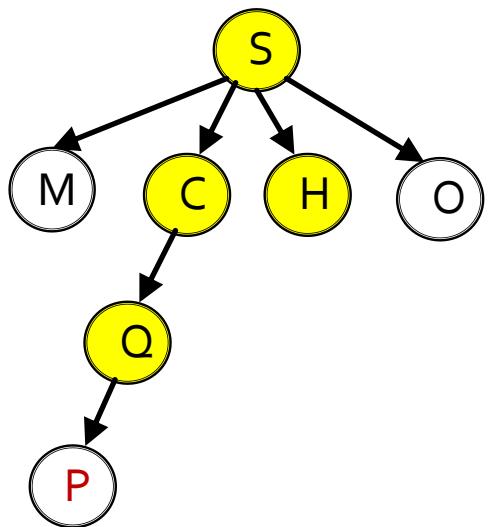
Queue: Q

↓ Dequeue and add new nodes

Queue: P

# Explore P

Proceed to the next layer: P



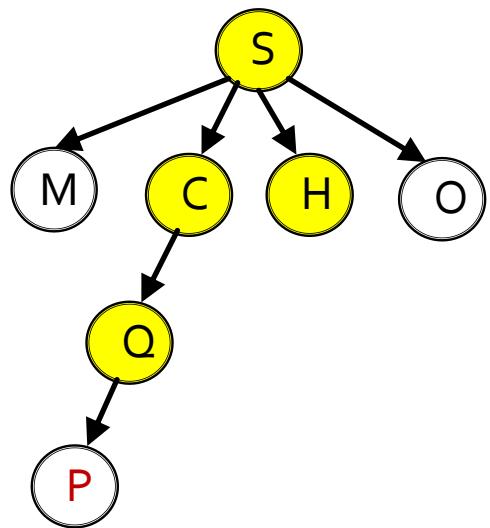
P is not a core point,  
as the number of neighbors within eps\_radius:  $2 < \text{minPts} (4)$

Queue: P

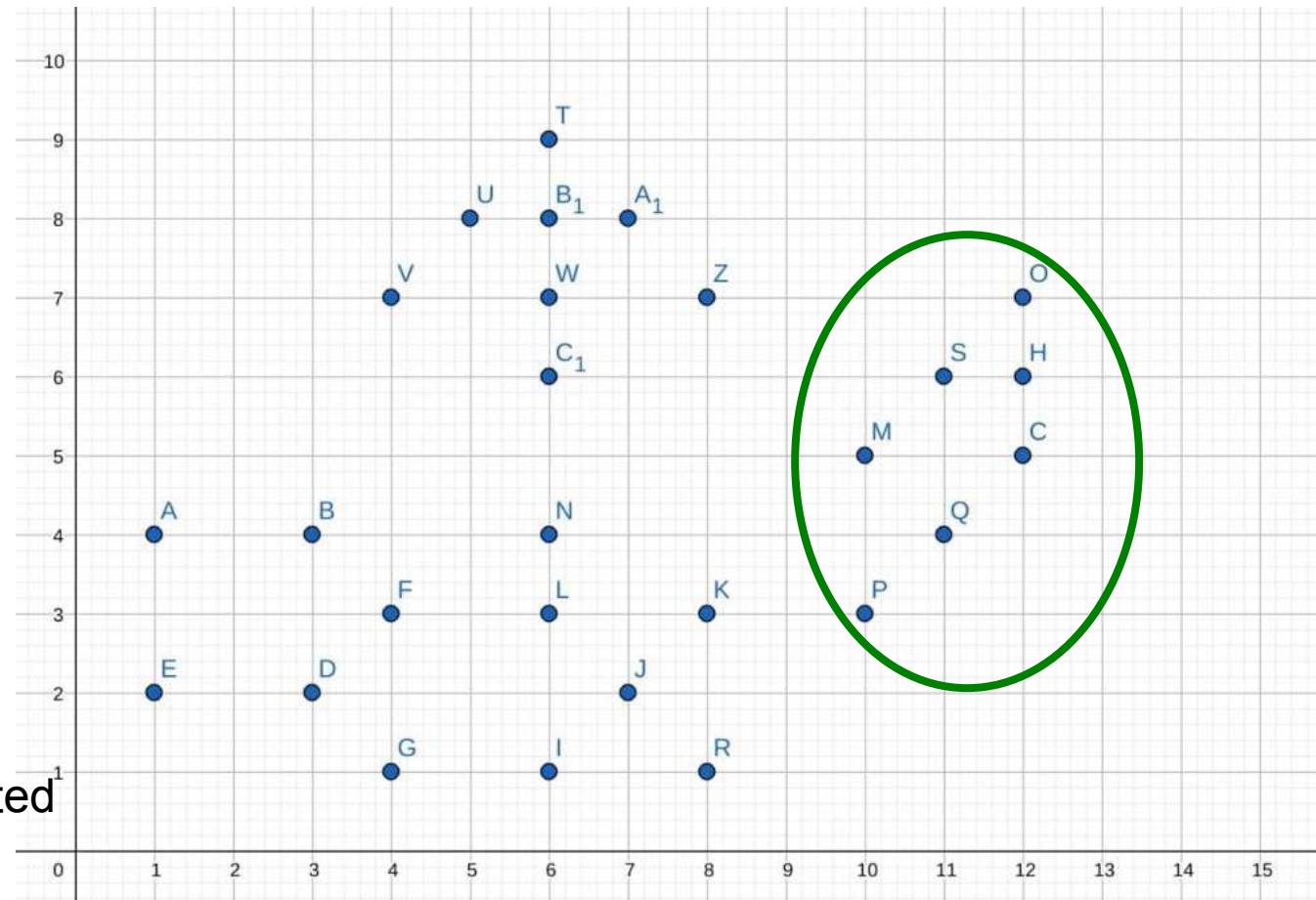
↓ Dequeue

Queue: <empty>

# Final result

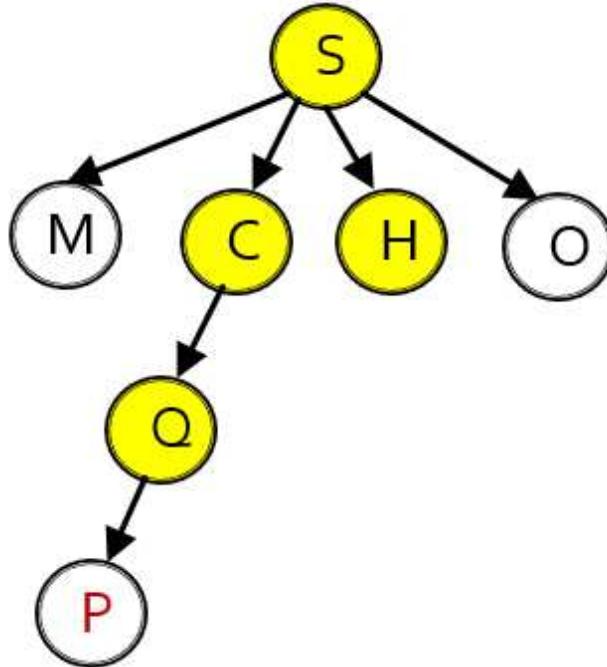


Core points are highlighted



Starting from data point S, we can construct the above cluster

## Discussion:



Use a tree to describe the process of cluster growth in DBSCAN

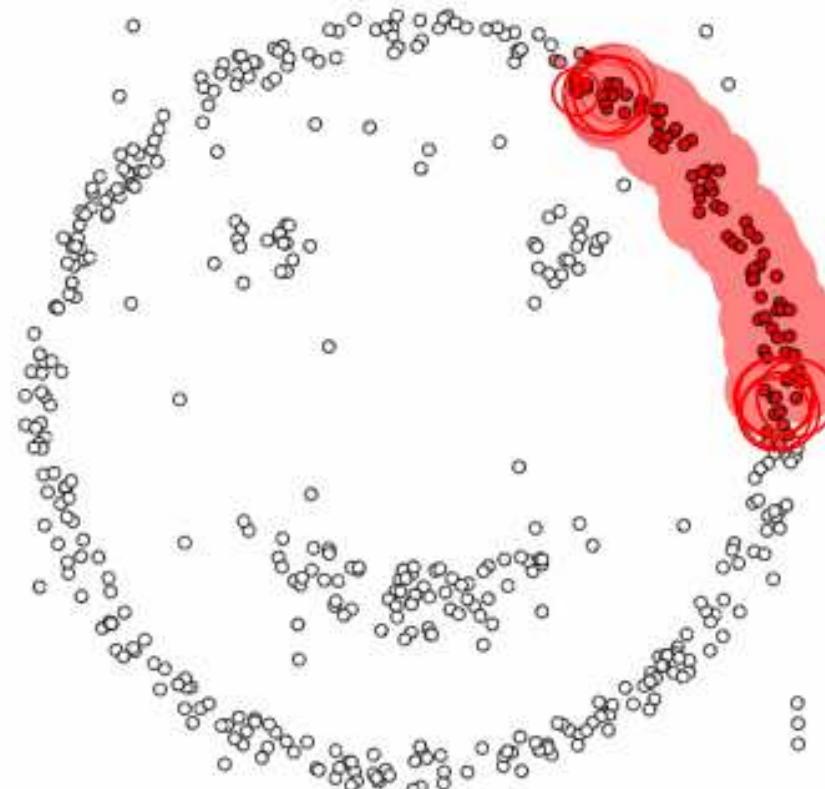
**A parent node:** represents a core point

**A child node:** a neighbour of the parent node

1. Use breadth-first search (layer-wise traversal) to expand the tree:  
walk through all nodes on the same level before moving on to the next level
2. The Leaf node can be a border point or core point

## Another example: illustration for cluster growth (animation)

epsilon = 1.00  
minPoints = 4



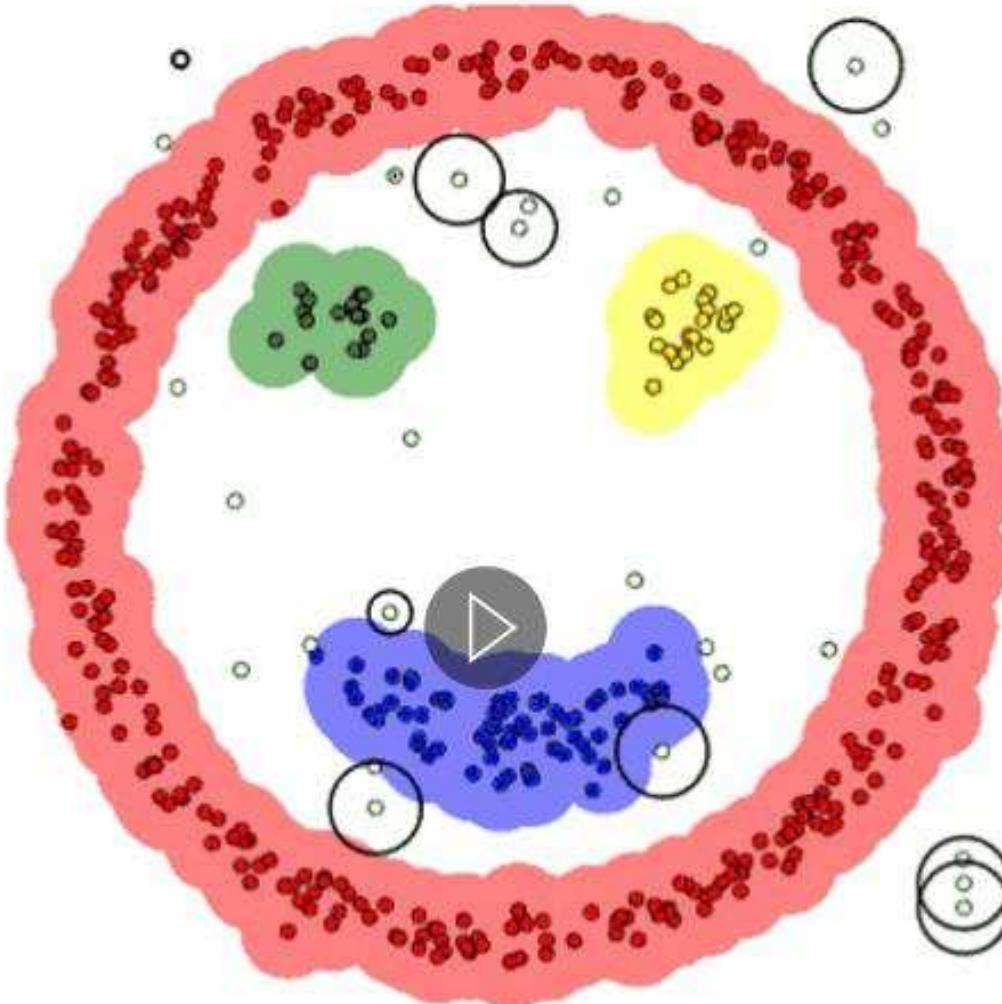
Restart



Pause

<https://www.digitalvidya.com/blog/the-top-5-clustering-algorithms-data-scientists-should-know/>

## Another example: illustration for cluster growth



## Discussion:

### Construct clusters in DBSCAN:

- A cluster is constructed by merging reachable core points and their border points.
  - A cluster consists of core points that are reachable from one another and all the border points of these core points.
- The requirement to form a cluster is to have at least one core point.

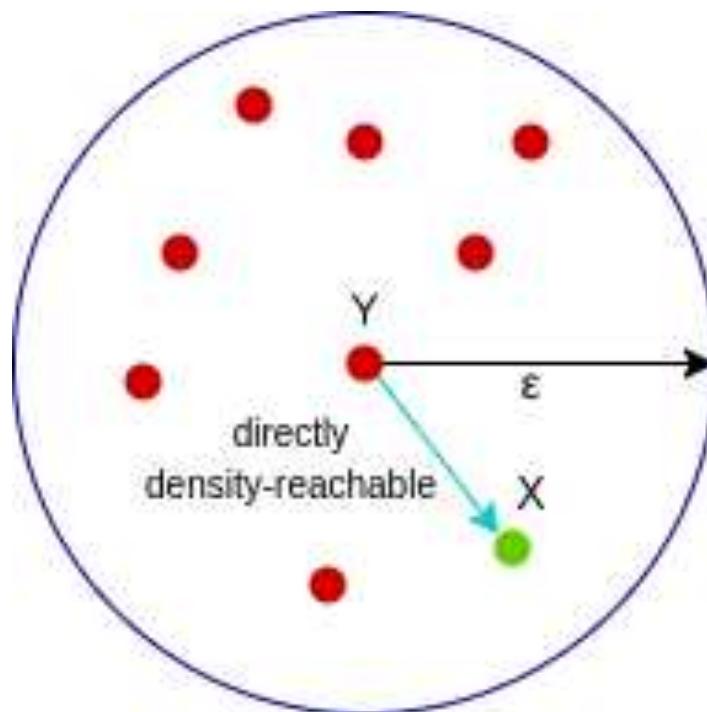
“directly reachable” is also called “directly density-reachable”

1. A point **X** is **directly density-reachable (or directly reachable)** from point **Y** w.r.t  $\text{epsilon}$ ,  $\text{minPoints}$  if,

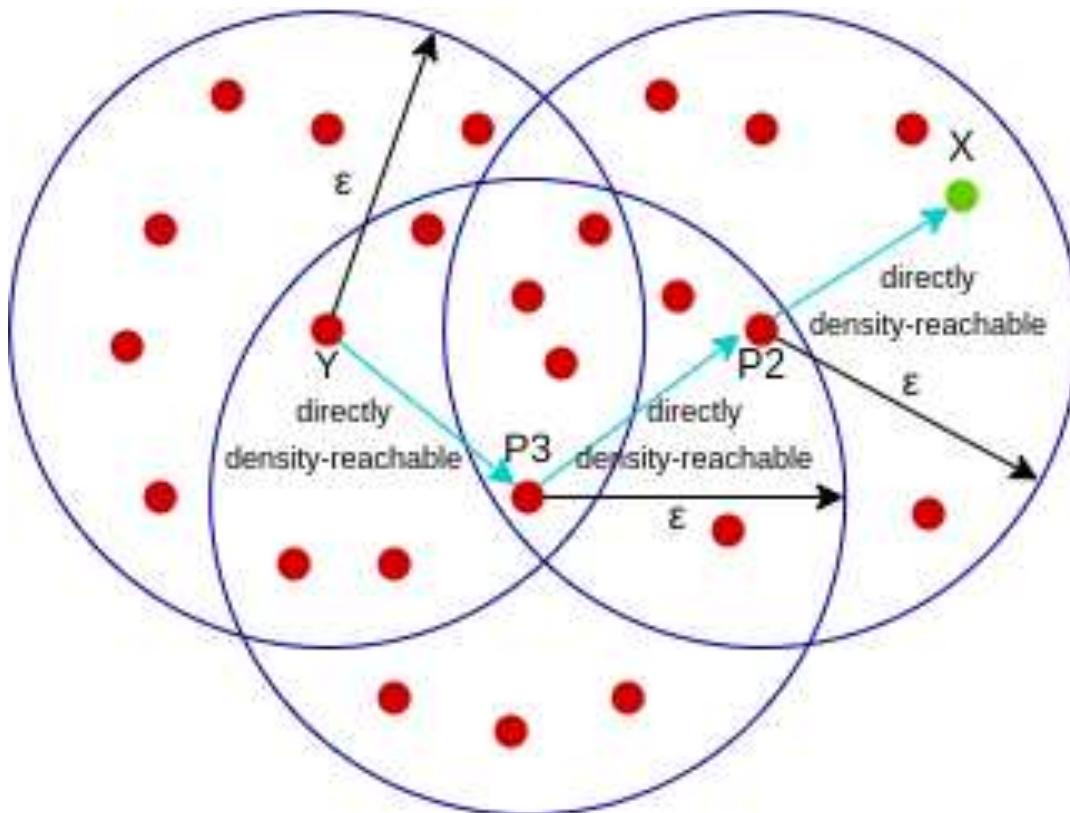
1) **X** belongs to the  $\text{epsilon}$ -neighborhood of **Y**, i.e,  $\text{dist}(X, Y) \leq \text{epsilon}$

It can be a border point or core point

2) **Y** is a core point



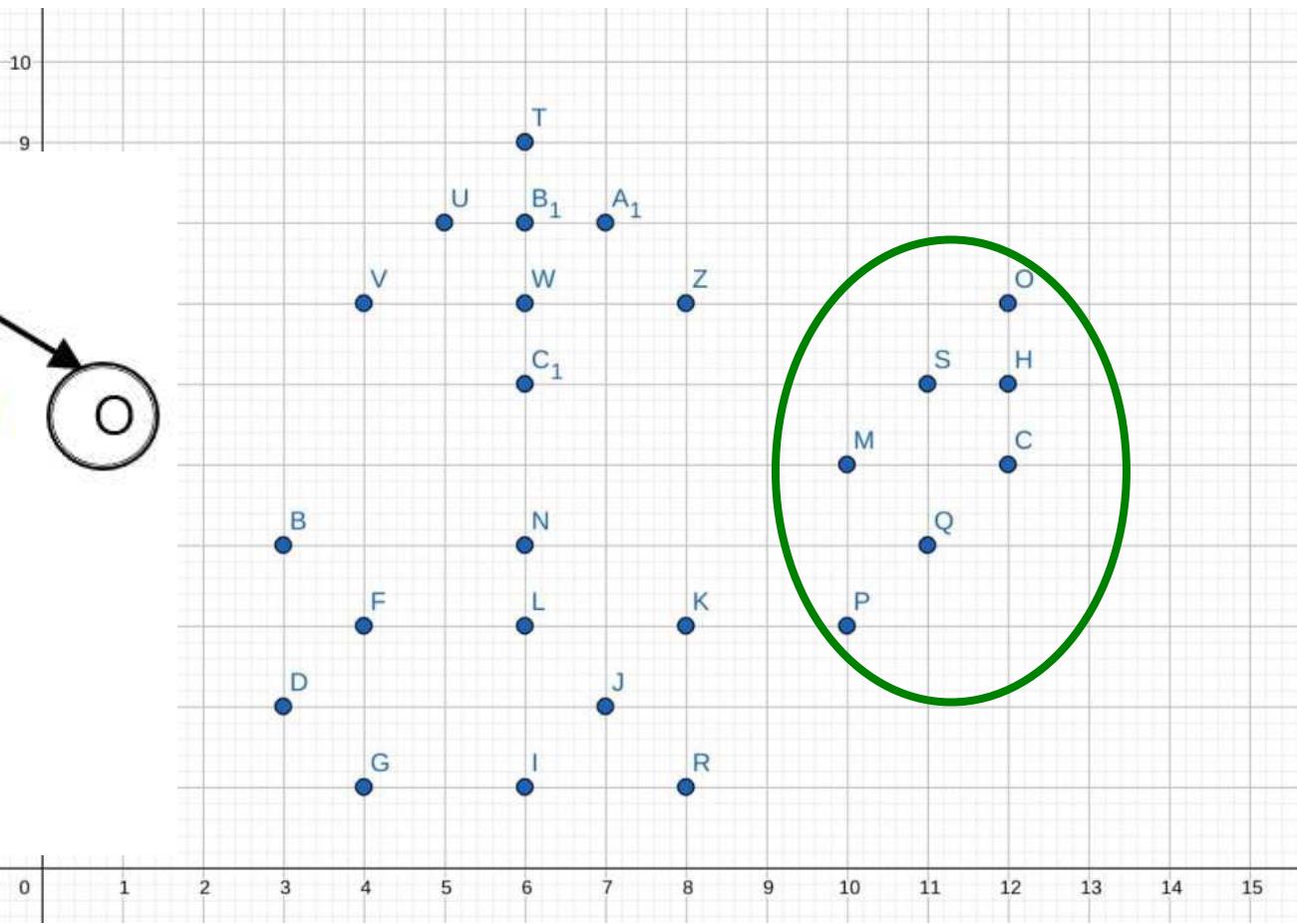
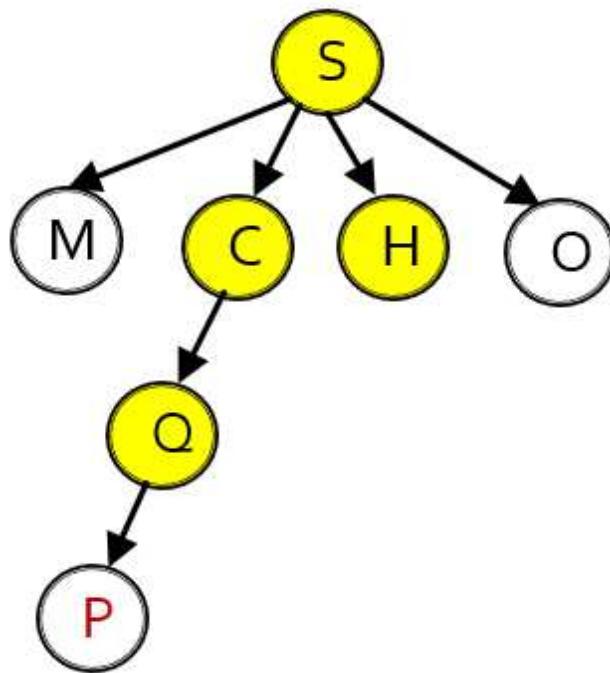
2. A point **X** is **density-reachable (or reachable)** from point **Y** w.r.t *epsilon*, *minPoints* if there is a chain of points  $p_1, p_2, p_3, \dots, p_n$  and  $p_1=X$  and  $p_n=Y$  such that  $p_{i+1}$  is directly density-reachable from  $p_i$ .



If X is reachable from Y (source point) :

we can find a path connecting points x and y, where each point in the path is directly reachable from the previous one. (The path is constructed by “directly reachable” core points)

## Example:



A tree to describe the process of cluster growth in DBSCAN

**A parent node:** represents a core point

**A child node:** a neighbour of the parent node

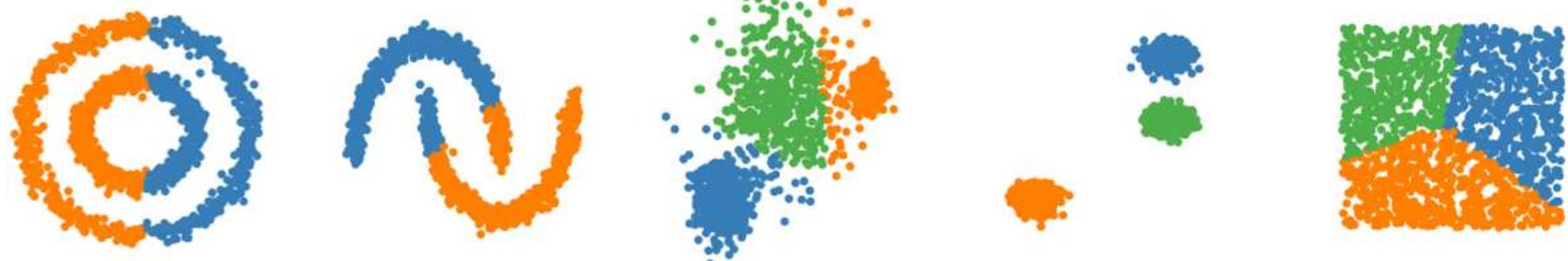
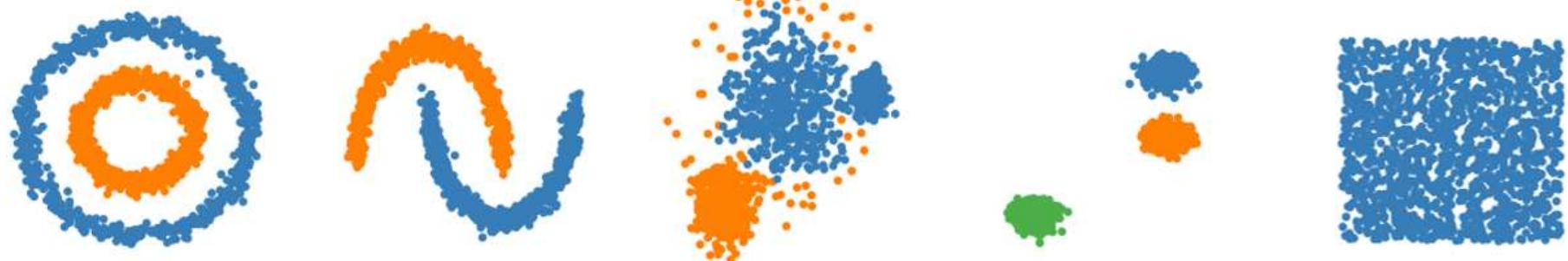
A connection from a parent node to its child indicates directly reachable

One point is reachable from any core points in the cluster (can find a path in the tree)

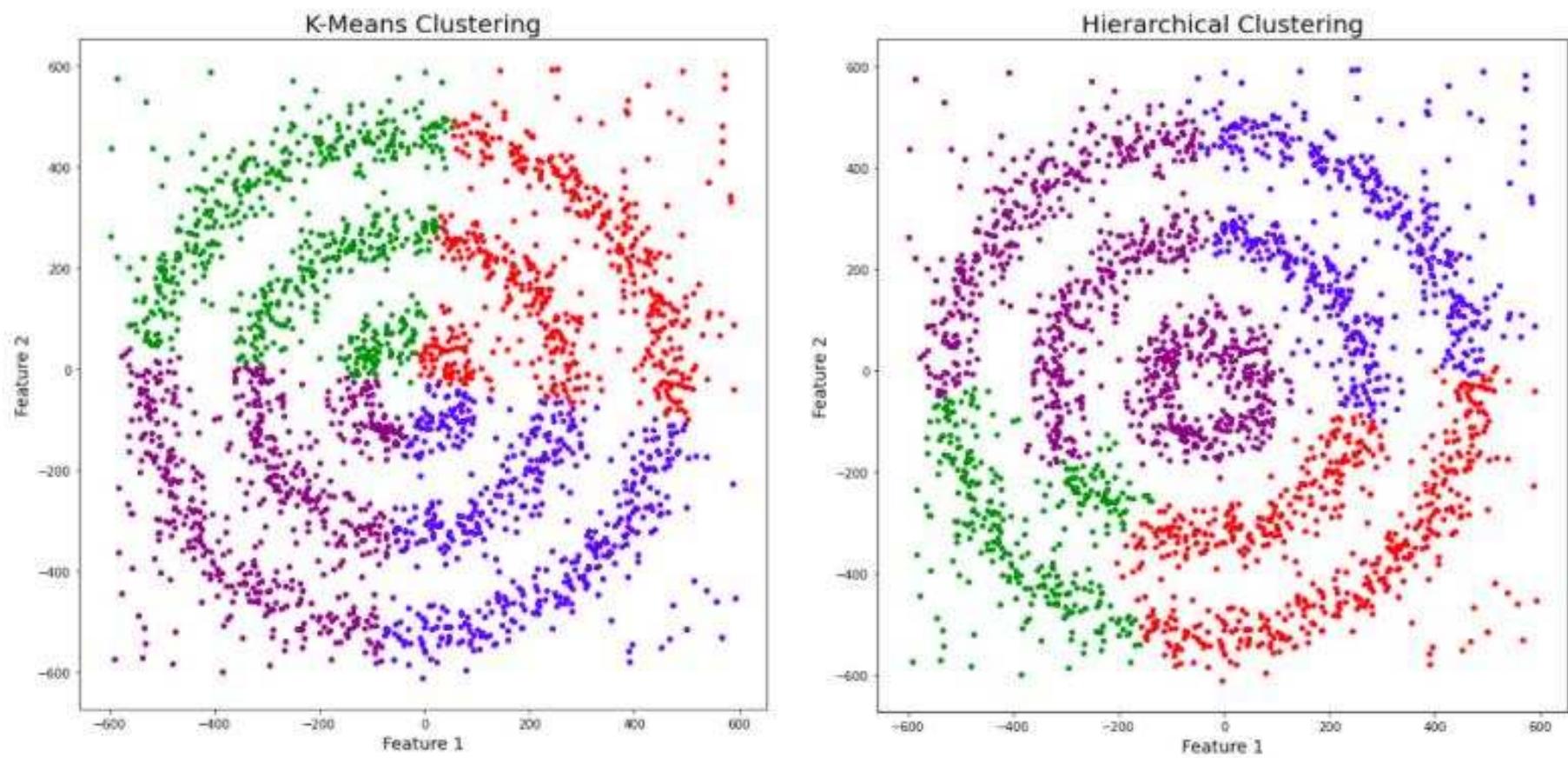
# Discussion

- DBSCAN
  - All core points are equally important to determine the shape of one cluster, so it can work for clusters with arbitrary shapes
- K-Means:
  - the centroid is important.
  - the shape of the cluster is determined by only one point
  - only works well for clusters with spherical shapes
- DBSCAN is density-based clustering
  - defines clusters as dense regions separated by low-density regions.

DBSCAN

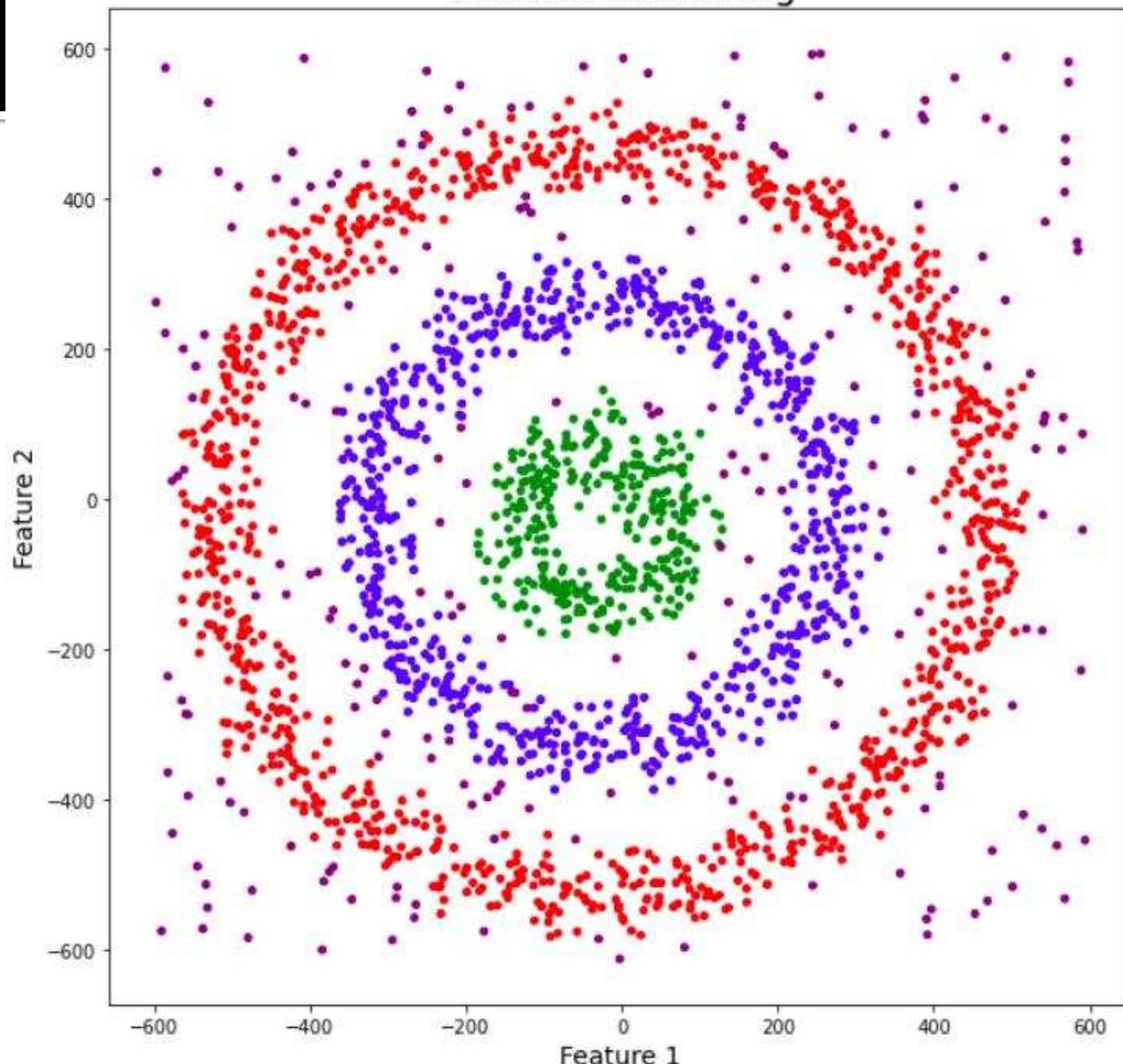


K-means



K-means, Hierarchical Clustering are sensitive to noise

## DBSCAN Clustering



<https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/>

# Discussion

## Pros and Cons of DBSCAN

### Pros:

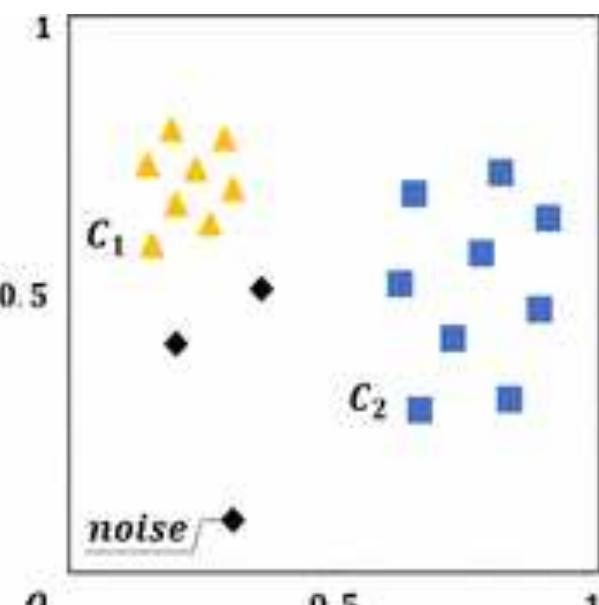
- Does not require to specify number of clusters beforehand.
- Performs well with arbitrary shapes clusters.
- DBSCAN is robust to outliers and able to detect the outliers.

# Discussion

- Cons
  - It is not very effective when you have clusters of varying densities.
    - if there are different density levels, it is difficult to choose a setting of the neighbourhood distance threshold ( $\text{epsilon}$ ) and  $\text{MinPts}$  that can work well for all density levels.
  - If you have high dimensional data, the determining of the distance threshold  $\mathcal{E}$  becomes a challenging task.

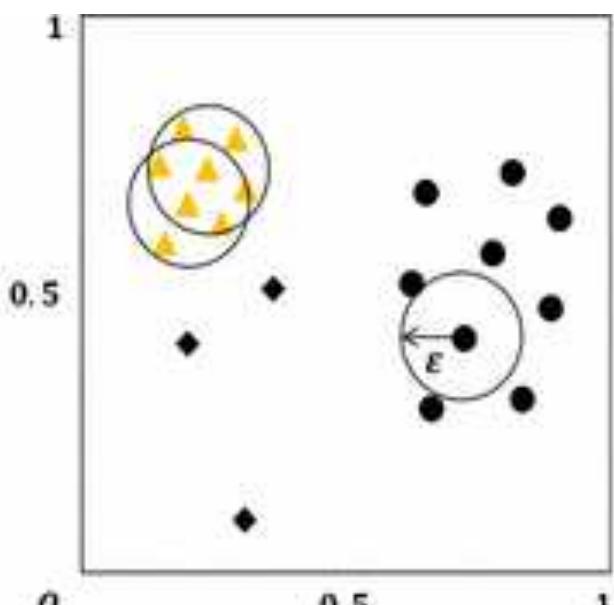
<https://www.digitalvidya.com/blog/the-top-5-clustering-algorithms-data-scientists-should-know/>

DBSCAN is not very effective for clusters with varying density



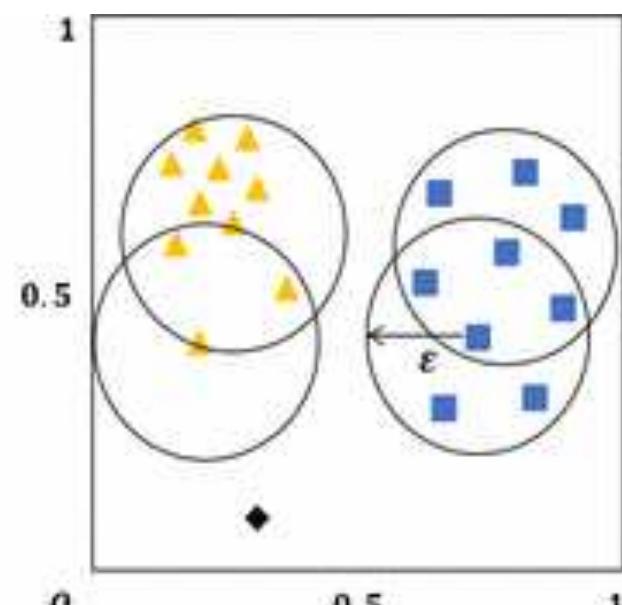
(a)

ideal result



(b)

DBSCAN with a small epsilon  
-> many outliers (noise points)



(c)

DBSCAN with a large epsilon  
-> many outliers are wrongly included in the dense cluster

# Extended discussion

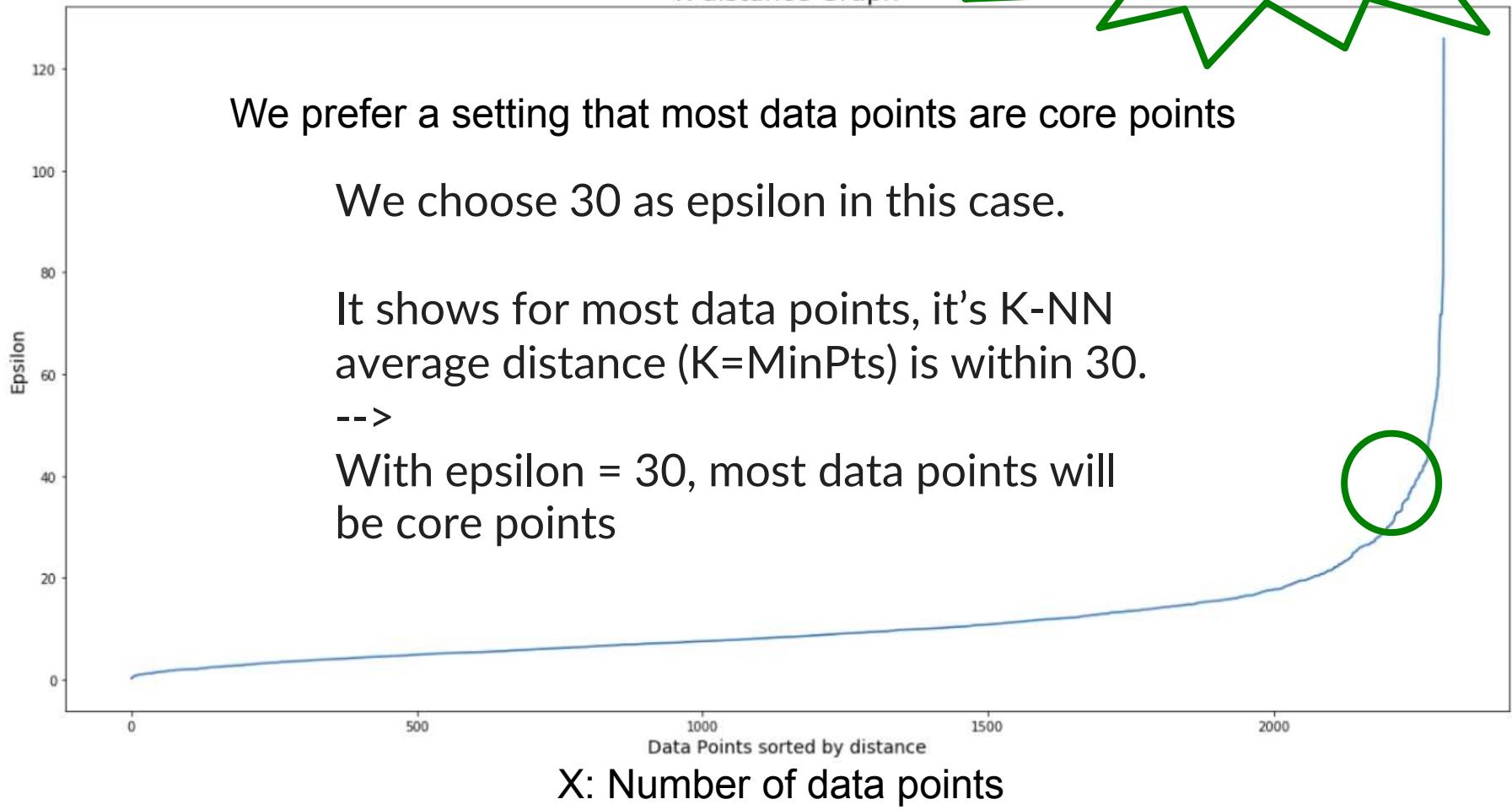
- Two parameters:
  - MinPts and epsilon
  - Select a value for MinPts and then search for epsilon
- How to choose epsilon? (Given MinPts)
  - Use K-distance graph
  - Step 1: Calculate the average distance between each point in the data set and its K nearest neighbors (set K as the MinPts value).
  - Step 2: Sort distance values by ascending value and plot the K-distance graph
  - Step 3: find the elbow point in the graph and use the corresponding distance as Epsilon



<https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd>

Y: Averaged distance of K-NN (K= MinPts)

K-distance Graph



# Graph community detection

Lin Guosheng  
School of Computer Science and Engineering  
Nanyang Technological University

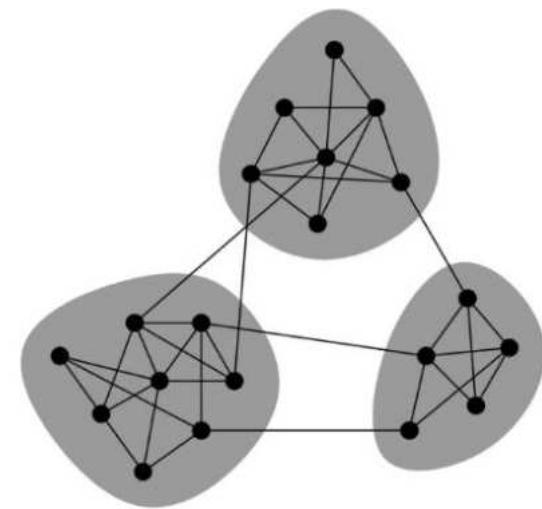
# Outline

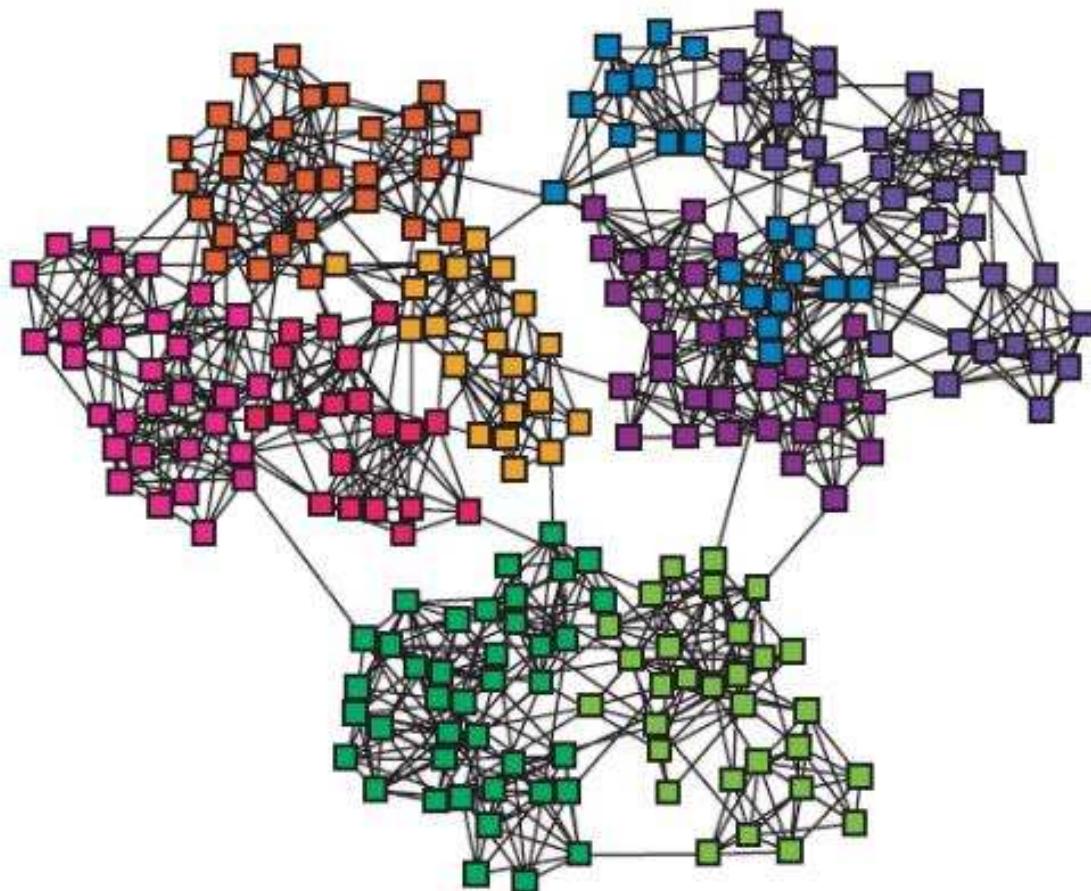
- Louvain Algorithm
  - Single pass
- Multi-pass Louvain Algorithm
  - **(non-examinable!)**

Many slides are from:

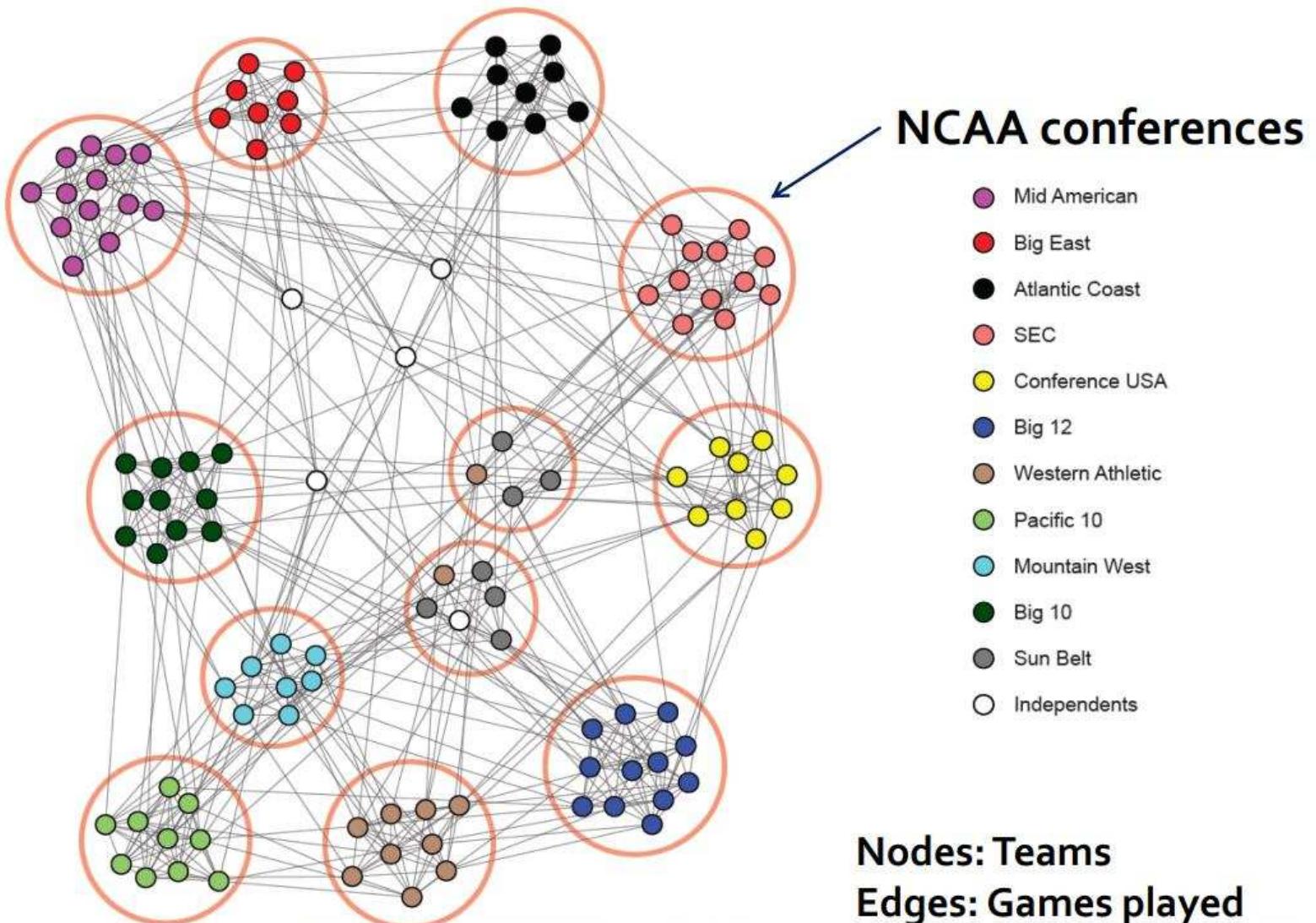
<http://snap.stanford.edu/class/cs224w-2020/slides/13-communities.pdf>

- One community in a graph:
  - A cluster of nodes
  - a group of tightly (densely) connected nodes
  - **many internal connections and few external connections** (to the rest of the network)
  - A community is also called: A cluster, A group, A module
- Community detection
  - A graph clustering task
  - Automatically find densely connected groups/clusters





# NCAA football network



# Louvain Algorithm

- Community detection on graphs
  - Greedy algorithm
    - Make locally optimal decision at each step
  - Supports weighted graphs
  - Provides hierarchical communities
  - Widely utilized to study large networks
    - Fast, rapid convergence
    - Produce high-quality results

# Modularity score

Use this equation to calculate the modularity score for a community:

$$Q(C) = \frac{\Sigma_{in}}{2m} - \left( \frac{\Sigma_{tot}}{2m} \right)^2$$

1. Modularity score: measure the quality of a community
2. We aim to maximize the total Modularity score of all communities.

$\Sigma_{in} \equiv \sum_{i,j \in C} A_{ij}$  : Sum of the weights of internal edges in the community C. (double count each edge)  
(high value indicates strong internal connections)

$\Sigma_{tot} \equiv \sum_{i \in C} k_i$  : Sum of the weighted degrees of all nodes in the community C

**m** : the sum of all edge weights in the undirected graph

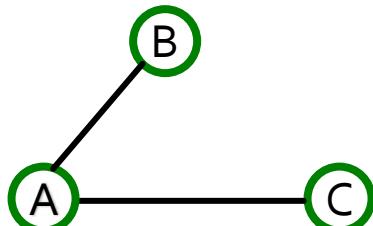
## ■ Node degree

- Un-weighted graph:

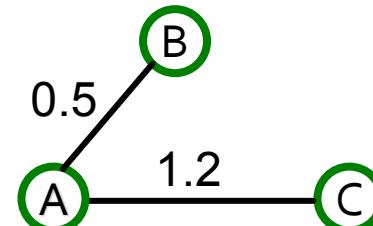
- Node degree: the number of connected edges.
  - all edge weights equal to 1

- Weighed graph:

- Node degree (weighted):  
the sum of the weights of connected edges.



Un-weighted graph  
 $D(A)=2$

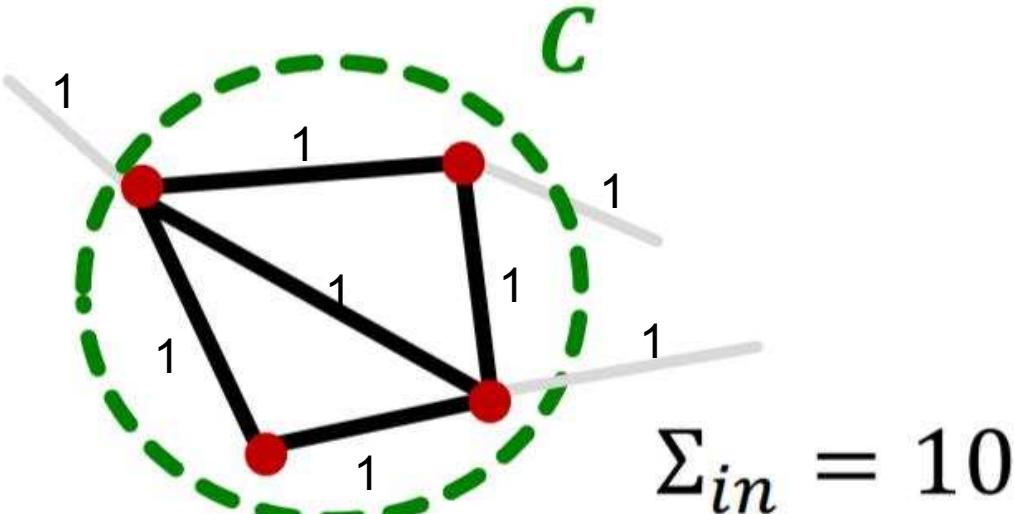


Weighted Graph  
 $D(A)=1.7$

$\Sigma_{in}:$

The index  $i$  and  $j$  indicate the nodes in the community  $C$

For un-weighted graph: the weight for each edge is 1



$$\Sigma_{in} \equiv \sum_{i,j \in C} A_{ij}$$

: Sum of all internal edge weights of the community (each internal edge will be double counted in the summation)

$A_{ij}$  is the edge weight for the edge connecting the nodes  $(i, j)$

In the undirected graph, the edge weight:  $A_{i,j} = A_{j,i}$

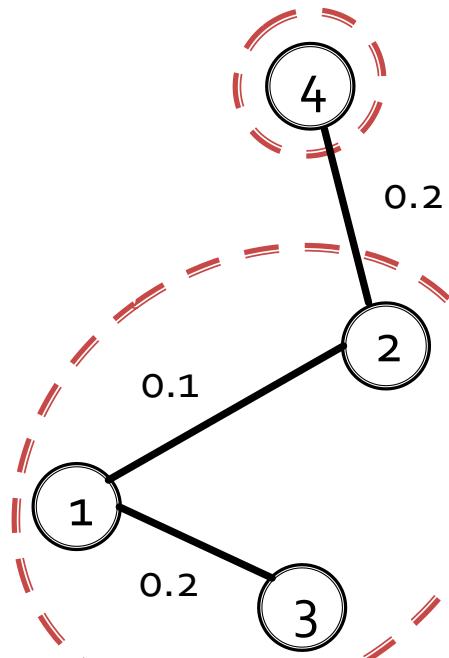
For the example here, the internal edges will be double counted in the summation:

$$(1+1+1+1+1) \times 2=10$$

# Another example

For the community A, there are three internal nodes {1,2,3}

Community B



We calculate  $\Sigma_{in}$  for community A:

$$\Sigma_{in} = \sum_{i,j \in C} A_{ij} \quad (i,j \text{ indicate the internal nodes})$$

$$= [A_{1,1} + A_{1,2} + A_{1,3}] + [A_{2,2} + A_{2,1} + A_{2,3}] \\ + [A_{3,3} + A_{3,1} + A_{3,2}]$$

There is no self links:  $A_{1,1} = 0; A_{2,2} = 0; A_{3,3} = 0$

There is no edge between node 2 and 3:  $A_{2,3} = A_{3,2} = 0$

In the undirected graph, the edge weight:  $A_{i,j} = A_{j,i}$

$$A_{1,2} = A_{2,1} = 0.1; \quad A_{1,3} = A_{3,1} = 0.2;$$

Community A

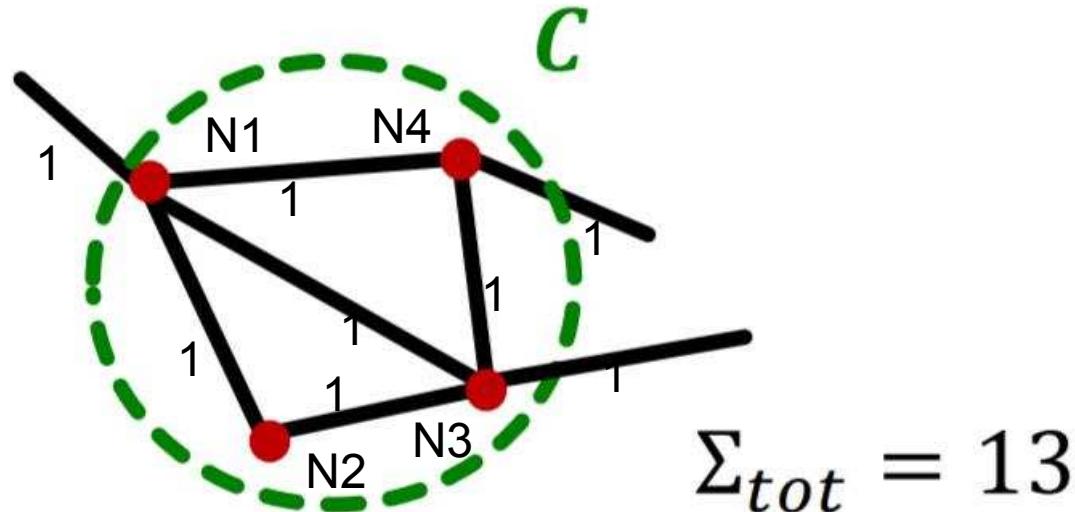
With the above, we can calculate  $\Sigma_{in}$  for community A as:

$$\Sigma_{in} = \sum_{i,j \in C} A_{ij} = (A_{1,2} + A_{1,3}) \times 2 = (0.1 + 0.2) \times 2 = 0.6$$

(sum the internal edge weights and multiply 2)

For un-weighted graph: the weight for each edge is 1

$\Sigma_{tot}:$



$\Sigma_{tot} \equiv \sum_{i \in C} k_i$  : Sum of the degrees of all internal nodes in the community C.  
Here  $k_i$  is the node degree of node i.

Sum of the node degrees of all internal nodes in C:

$$4 + 2 + 4 + 3 = 13$$

(for node N1, N2, N3, N4, respectively)

# Example: an extreme case

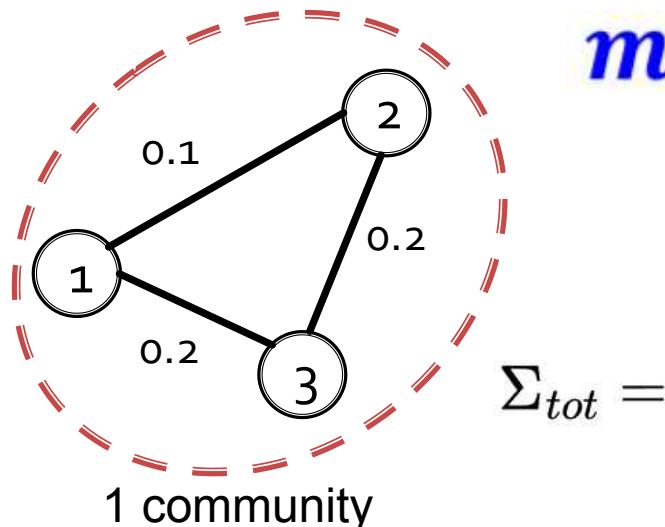
- An extreme case: all nodes belong to 1 community
  - This trivial clustering strategy provides a baseline score

$$Q(C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 = \frac{2m}{2m} - \left(\frac{2m}{2m}\right)^2 = 0$$

We have:  $\Sigma_{in} = 2m$

$$\Sigma_{tot} = 2m$$

Example for the left graph:



**m**

: the sum of all edge weights in the undirected graph

$$m = 0.1 + 0.2 + 0.2 = 0.5$$

$$\Sigma_{in} = (0.1 + 0.2) \times 2 = 1$$

$$\Sigma_{tot} = (0.1 + 0.2) + (0.1 + 0.2) + (0.2 + 0.2) = 1$$

$$Q(C) = 0$$

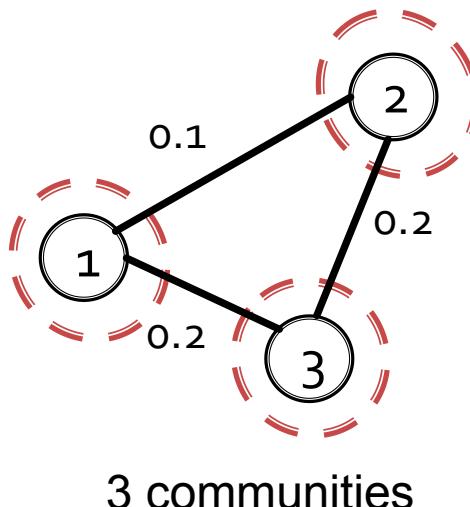
# Example: another extreme case

- Another extreme case: one node forms one community

$$Q(C_i) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 = \frac{0}{2m} - \left(\frac{k_i}{2m}\right)^2 = -\left(\frac{k_i}{2m}\right)^2$$

We have:  $\Sigma_{in} = 0$        $k_i$  : the node degree of node i  
 $\Sigma_{tot} = k_i$

Example for the left graph:



$$m = 0.1 + 0.2 + 0.2 = 0.5$$

$$Q(\{1\}) = -\left(\frac{k_1}{2m}\right)^2 = -\left(\frac{0.1 + 0.2}{1}\right)^2 = -0.09$$

$$Q(\{2\}) = -\left(\frac{k_2}{2m}\right)^2 = -\left(\frac{0.1 + 0.2}{1}\right)^2 = -0.09$$

$$Q(\{3\}) = -\left(\frac{k_3}{2m}\right)^2 = -\left(\frac{0.2 + 0.2}{1}\right)^2 = -0.16$$

# Example: another extreme case

- Another extreme case: one node forms one community

The modularity score for the community formed by node i:

$$Q(C_i) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 = \frac{0}{2m} - \left(\frac{k_i}{2m}\right)^2 = -\left(\frac{k_i}{2m}\right)^2$$

We have:  $\Sigma_{in} = 0$

$$\Sigma_{tot} = k_i$$

$k_i$  : the node degree of node i

The total modularity score of all communities:  
(N indicates the total number of communities)

$$\sum_{i=1}^N Q(C_i) = - \sum_{i=1}^N \left(\frac{k_i}{2m}\right)^2$$

The total score is less than 0

# Modularity score (discussion)

What is a good community:

The nodes in the community have lots of internal connections and very few external connections (to the rest of the network)

Modularity score: measure the quality of a community

A large value indicates a lot of internal connections

$$Q(C) = \frac{\Sigma_{in}}{2m} - \left( \frac{\Sigma_{tot}}{2m} \right)^2$$

$$\frac{\Sigma_{in}}{2m} \leq 1$$

$$\frac{\Sigma_{tot}}{2m} \leq 1$$

1. A large value indicates a large community or large node degrees.
2. This term will penalize large community and the external connections.
  - The node degree includes the internal and external links of the node.
  - A large number of external links will lead to a large node degree.
3. Intuitively, the square operation is to downgrade the impact of the second term: e.g.  $(0.1)^2 < 0.1$

## ■ Louvain Algorithm

- aims to greedily maximize the Modularity score.
- Modularity score: a metric to measure the quality of a community
- Single pass Louvain Algorithm
  - Community generation
    - Also called modularity optimization

# Louvain Algorithm: community generation

- Initialization: each node forms a distinct community
- A: generate a random list of nodes (start one scan)
- B: (one scan) sequentially process each node in the list for community update
- Repeat A, B until converge
  - Converge:  
there is no update of the community in the last scan

- B: sequentially process each node for community update
  - 1) Node movement step.
    - Identify possible movements by finding neighboring communities (directly connected external communities)
    - For each movement: compute the modularity gain ( $\Delta Q$ ) as the movement score.

$$\Delta Q = Q_{\text{after}} - Q_{\text{before}}$$

- 2) Community update step.
  - Move the node to a community that yields the largest positive score ( $\Delta Q$ ). If the largest score is not positive, there is no movement of the node.

# Modularity gain (movement score)

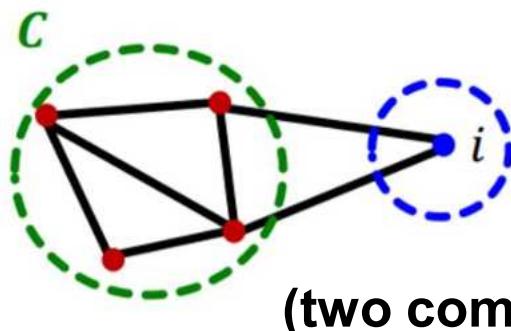
Directly use this equation to calculate the modularity score for each community:

$$Q(C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2$$

Modularity gain:

$$\Delta Q = Q_{\text{after}} - Q_{\text{before}}$$

Before merging

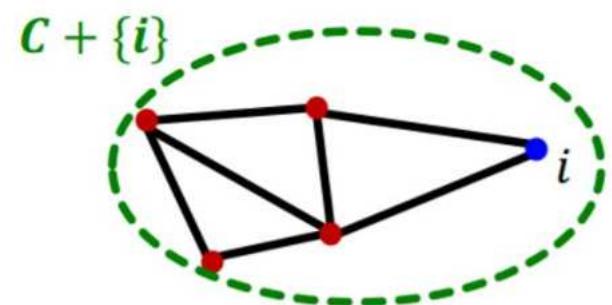


Merging  $i$  into  $C$

$$\Delta Q(i \rightarrow C)$$

Isolated  
community  
of node  $i$

After merging



$$Q_{\text{before}} = Q(C) + Q(\{i\})$$

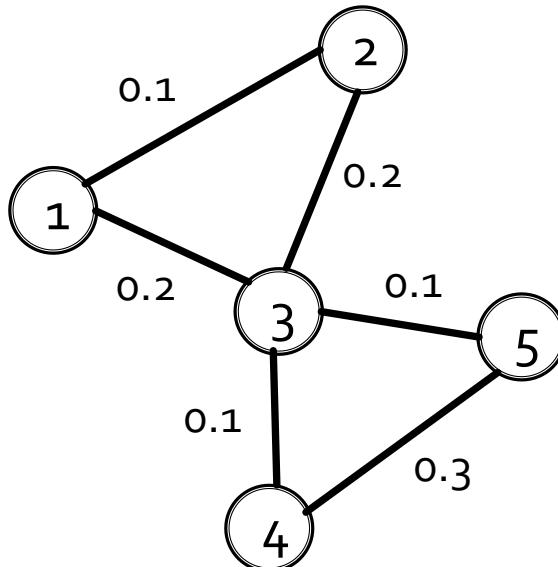
$$Q_{\text{after}} = Q(C + \{i\})$$

Modularity gain:  $\Delta Q(i \rightarrow C) = Q_{\text{after}} - Q_{\text{before}}$

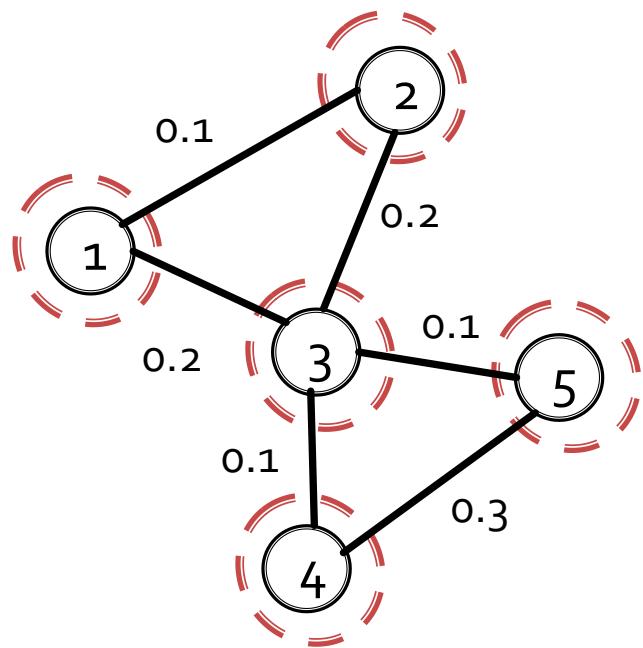
## Example: community update

Q: A graph is given below.

In the initialization, each node forms a distinct community.  
Use Louvain algorithm to sequentially process the nodes  
in the order {3, 1, 2, 4, 5} to update the communities given  
in the initialization.

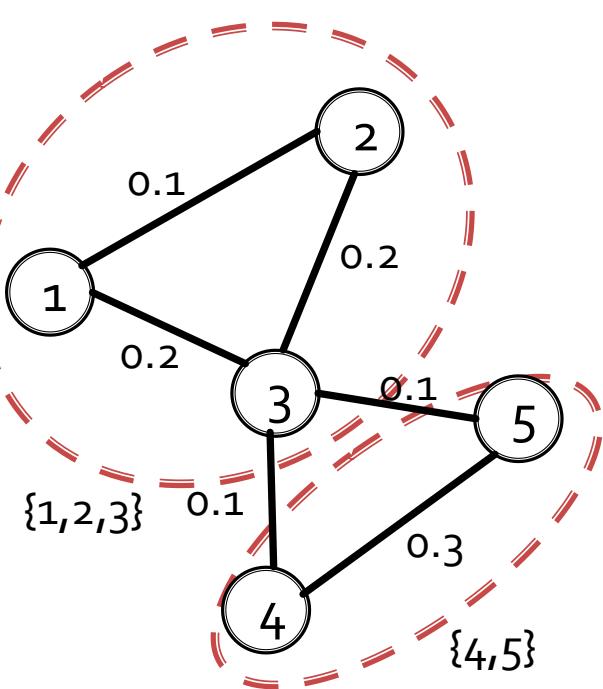


# Result



In the initialization,  
each node forms a distinct community

Process nodes:  
 $\{3, 1, 2, 4, 5\}$   
to update  
communities



After processing,  
the communities are updated

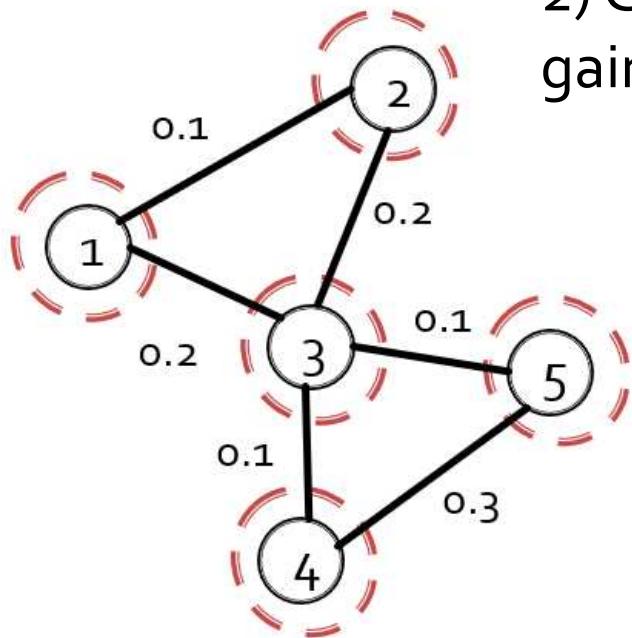
# Intermediate steps

Sequentially process the node list  $\{3, 1, 2, 4, 5\}$

## processing node 3

- 1) Identify neighbouring communities of node 3:  $\{1\}, \{2\}, \{4\}$  and  $\{5\}$

- 2) Calculate the movement scores (modularity gains) for the following 4 movements:



$$\Delta Q(3 \rightarrow \{1\})$$

$$\Delta Q(3 \rightarrow \{2\})$$

$$\Delta Q(3 \rightarrow \{4\})$$

$$\Delta Q(3 \rightarrow \{5\})$$

## Intermediate steps

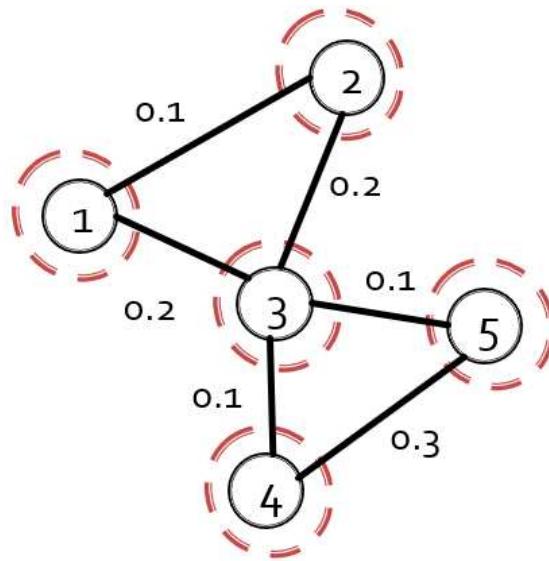
$$Q(C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2$$

Sequentially process the node list  $\{3, 1, 2, 4, 5\}$

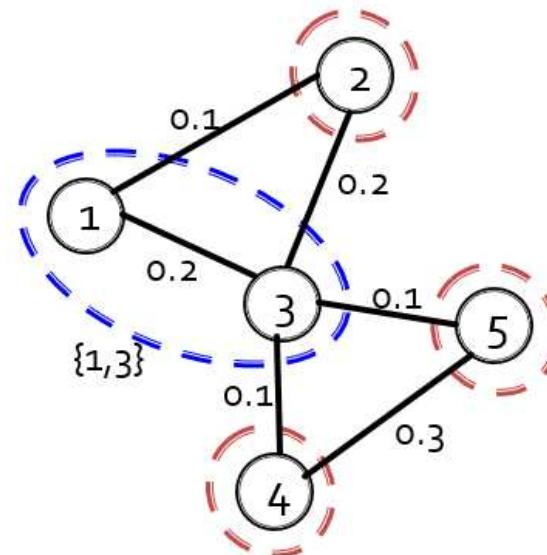
### processing node 3

Movement score (Modularity gain)  
of moving node 3 to community  $\{1\}$ :

$$\begin{aligned}\Delta Q(3 \rightarrow \{1\}) &= Q_{after} - Q_{before} \\ &= Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})]\end{aligned}$$



$Q_{before}$



$Q_{after}$

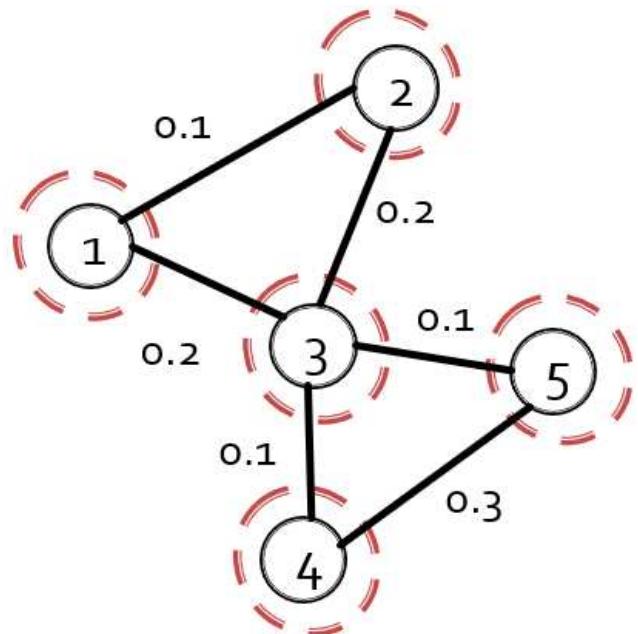
## Intermediate steps

$$Q(C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2$$

Sequentially process the node list  $\{3, 1, 2, 4, 5\}$

### processing node 3

$$\begin{aligned}\Delta Q(3 \rightarrow \{1\}) &= Q_{after} - Q_{before} \\ &= Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})]\end{aligned}$$



$Q_{before}$

$$\begin{aligned}k_1 &= 0.1 + 0.2 = 0.3; \quad k_3 = 0.2 + 0.2 + 0.1 + 0.1 = 0.6; \\ m &= 0.1 + 0.2 + 0.2 + 0.1 + 0.1 + 0.3 = 1\end{aligned}$$

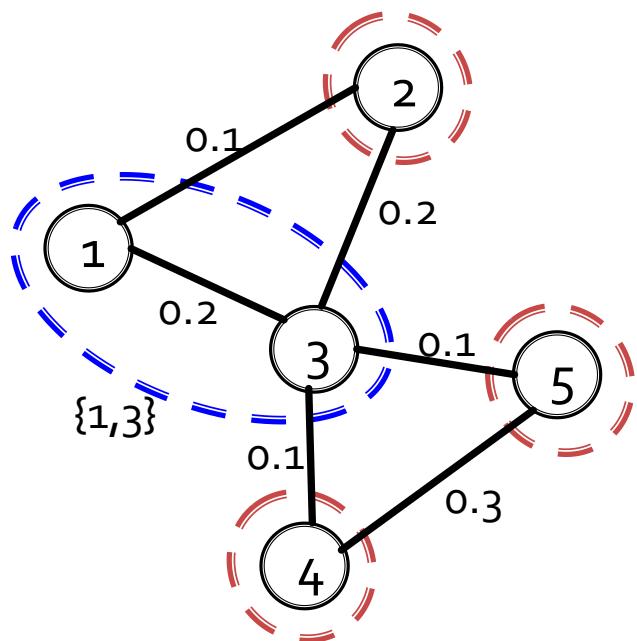
$$\begin{aligned}Q(\{1\}) &= \left[0 - \left(\frac{k_1}{2m}\right)^2\right] \\ &= -0.0225\end{aligned}$$

$$\begin{aligned}Q(\{3\}) &= \left[0 - \left(\frac{k_3}{2m}\right)^2\right] \\ &= -0.09\end{aligned}$$

## processing node 3

$$\Delta Q(3 \rightarrow \{1\}) = Q_{after} - Q_{before}$$

$$= Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})]$$

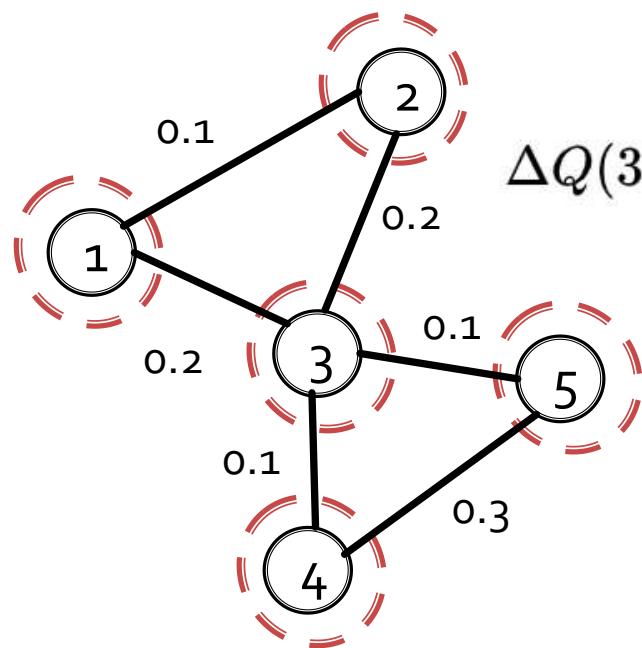


$$\begin{aligned}
 Q(\{1, 3\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\
 &= \frac{2 \times e_{31}}{2m} - \left(\frac{k_1+k_3}{2m}\right)^2 \\
 &= \frac{2 \times 0.2}{2} - \left(\frac{0.3+0.6}{2}\right)^2 \\
 &= -0.0025
 \end{aligned}$$

$Q_{\text{after}}$

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 3



Movement score (Modularity gain)  
of moving node 3 to community {1}

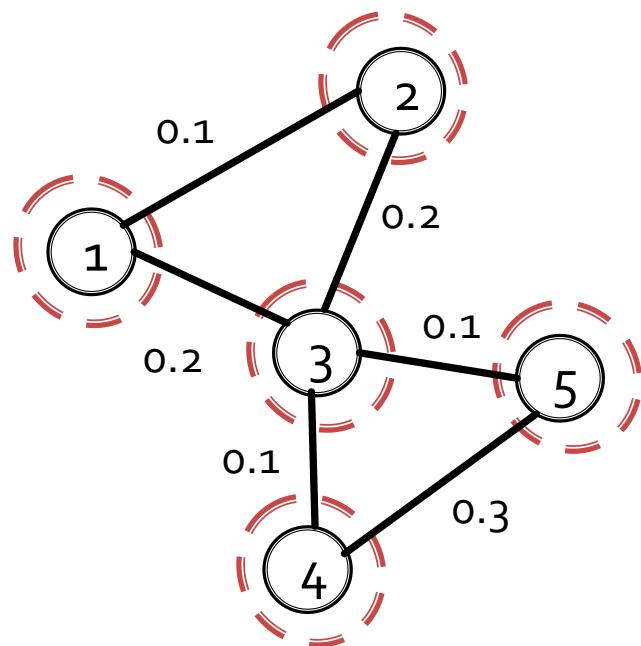
$$\begin{aligned}\Delta Q(3 \rightarrow \{1\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})] \\ &= Q(\{1, 3\}) - [Q(\{1\}) + Q(\{3\})] \\ &= -0.0025 - (-0.0225 - 0.09) \\ &= 0.11\end{aligned}$$

Sequentially process the node list {3, 1, 2, 4, 5}

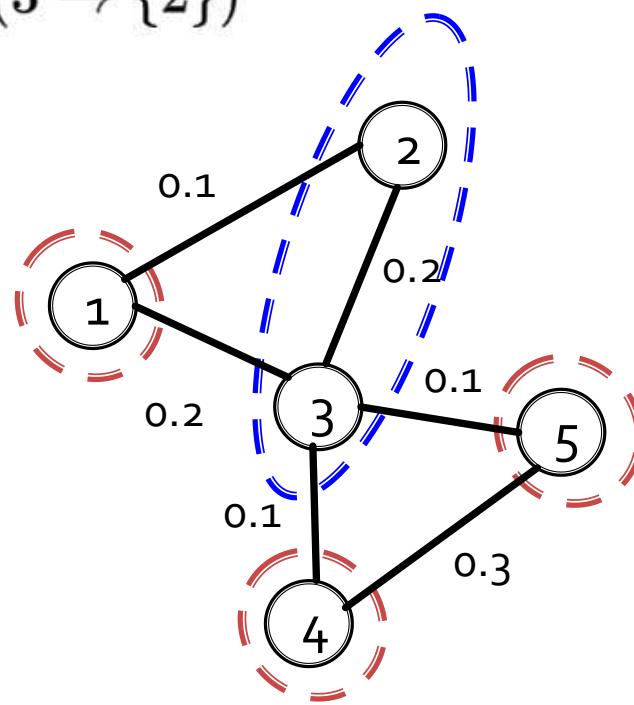
## processing node 3

Movement score (Modularity gain) of moving node 3 to community {2}

$$\Delta Q(3 \rightarrow \{2\})$$



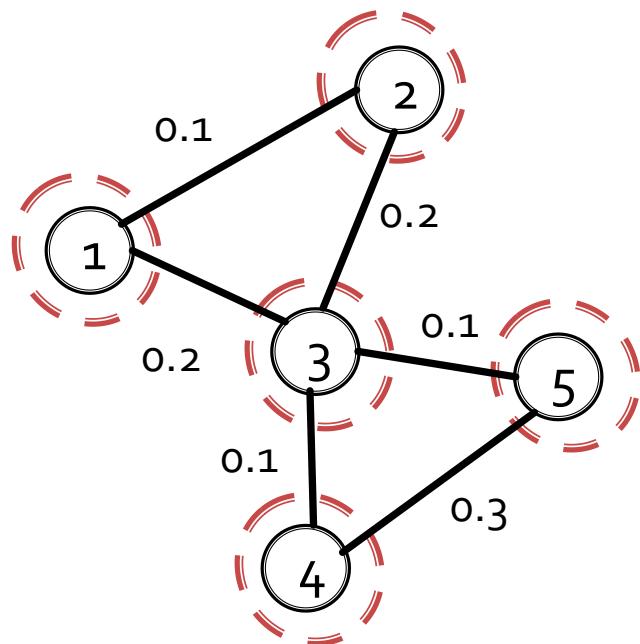
$Q_{\text{before}}$



$Q_{\text{after}}$

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 3



Movement score (Modularity gain) of moving node 3 to community {2}

The link (1,3) and (2,3) have the same weight.

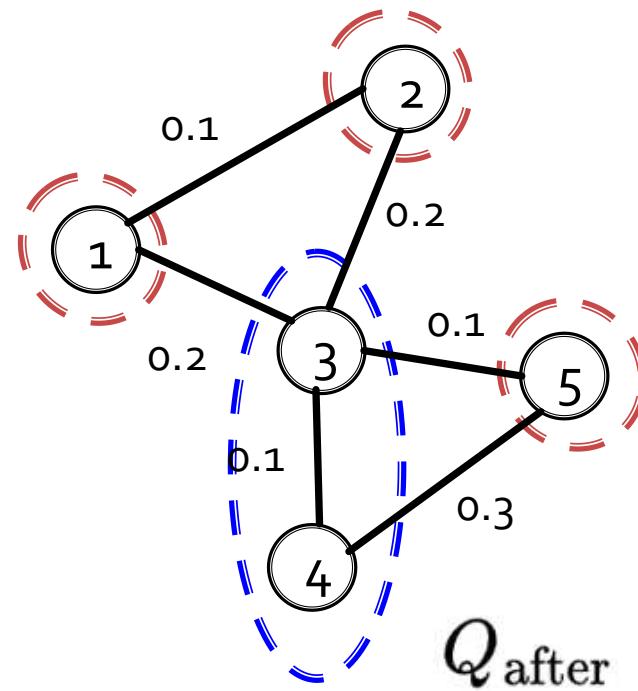
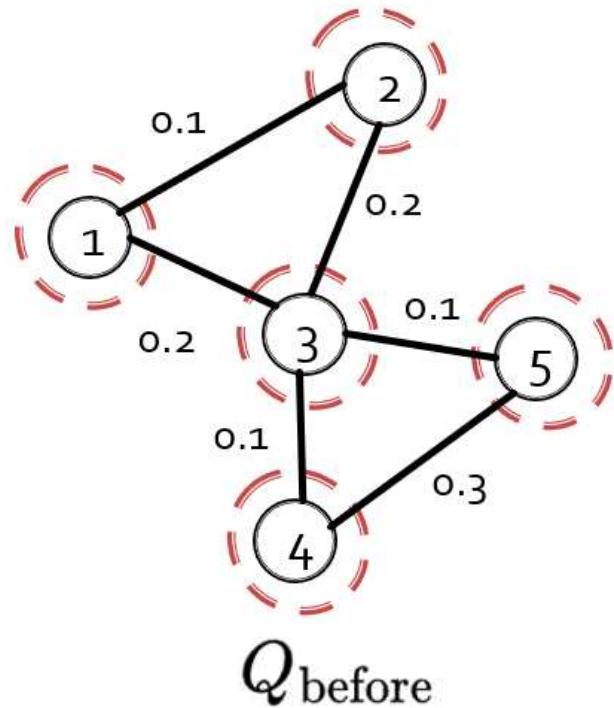
The gain of moving node 3 to {2} is the same as  $\Delta Q(3 \rightarrow \{1\})$

$$\Delta Q(3 \rightarrow \{2\}) = 0.11$$

Sequentially process the node list  $\{3, 1, 2, 4, 5\}$

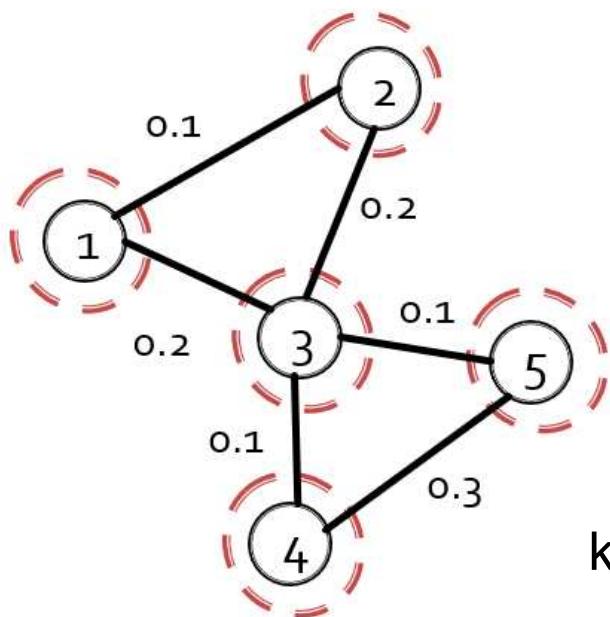
**processing node 3** The movement score of moving node 3 to  $\{4\}$ :

$$\begin{aligned}\Delta Q(3 \rightarrow \{4\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{4, 3\}) - [Q(\{4\}) + Q(\{3\})]\end{aligned}$$



Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 3



The movement score of moving node 3 to {4}:

$$\begin{aligned}
 \Delta Q(3 \rightarrow \{4\}) &= Q_{\text{after}} - Q_{\text{before}} \\
 &= Q(\{4\} + \{3\}) - [Q(\{4\}) + Q(\{3\})] \\
 &= Q(\{4, 3\}) - [Q(\{4\}) + Q(\{3\})] \\
 &= -0.15 - (-0.04 - 0.09) \\
 &= -0.02
 \end{aligned}$$

$$k_3 = 0.2 + 0.2 + 0.1 + 0.1 = 0.6; \quad k_4 = 0.1 + 0.3 = 0.4$$

$$\begin{aligned}
 Q(\{4, 3\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\
 &= \frac{2 \times e_{34}}{2m} - \left(\frac{k_4+k_3}{2m}\right)^2 \\
 &= \frac{2 \times 0.1}{2} - \left(\frac{0.4+0.6}{2}\right)^2 \\
 &= -0.15
 \end{aligned}$$

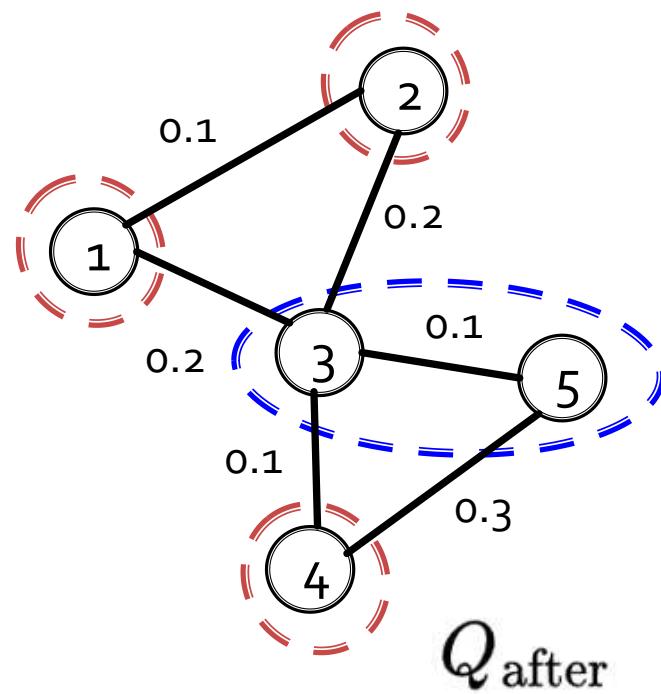
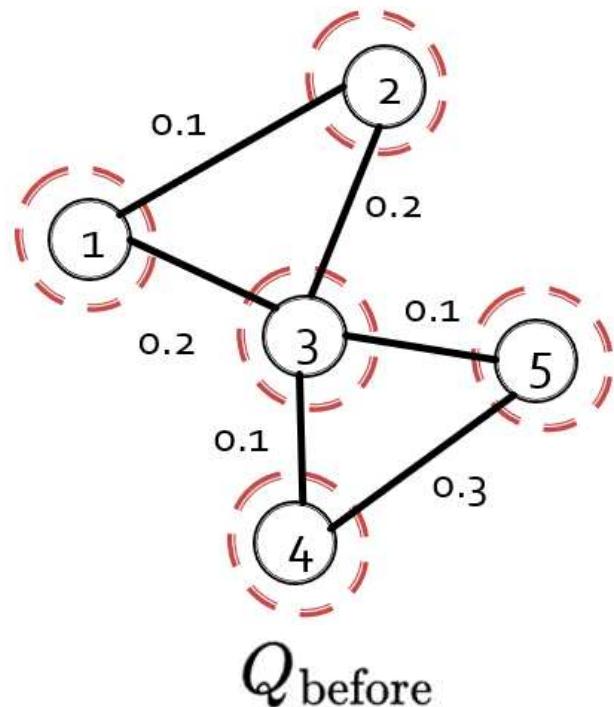
$$\begin{aligned}
 Q(\{3\}) &= \left[0 - \left(\frac{k_3}{2m}\right)^2\right] \\
 &= -0.09 \\
 Q(\{4\}) &= \left[0 - \left(\frac{k_4}{2m}\right)^2\right] \\
 &= -0.04
 \end{aligned}$$

Sequentially process the node list  $\{3, 1, 2, 4, 5\}$

## processing node 3

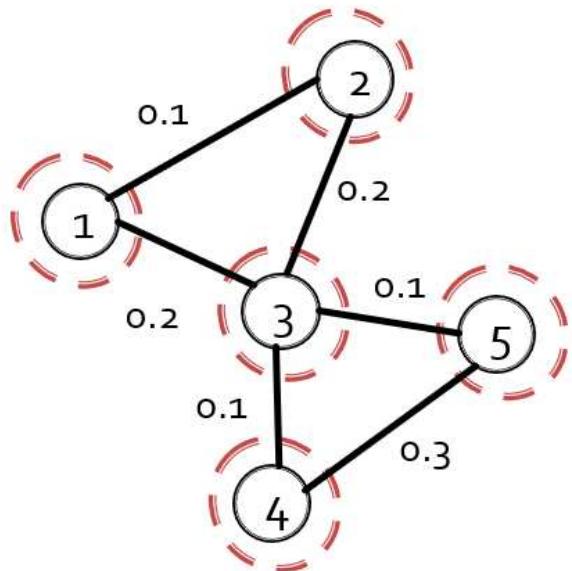
The movement score of moving node 3 to  $\{5\}$ :

$$\Delta Q(3 \rightarrow \{5\})$$



Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 3



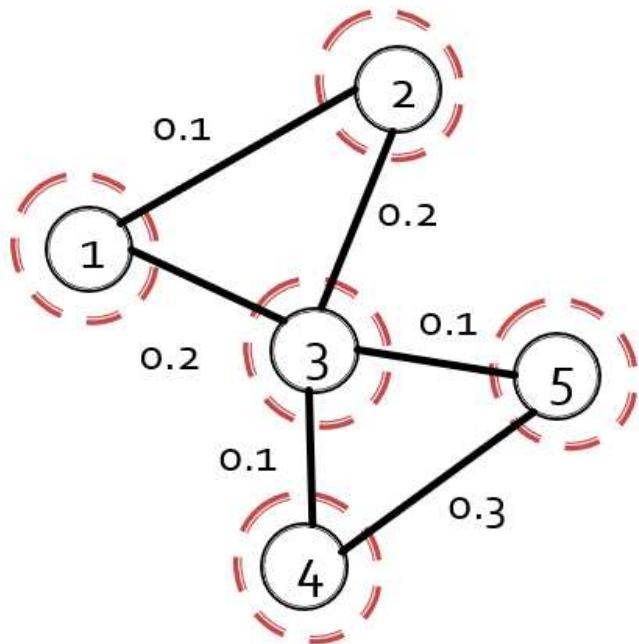
The movement score of moving node 3 to {5}:

The gain of moving node 3 to {5}  
is the same as:  $\Delta Q(3 \rightarrow \{4\})$

$$\Delta Q(3 \rightarrow \{5\}) = -0.02$$

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 3



Summary:

Scores for all possible movements:

$$\Delta Q(3 \rightarrow \{1\}) = \Delta Q(3 \rightarrow \{2\}) = 0.11$$

$$\Delta Q(3 \rightarrow \{4\}) = \Delta Q(3 \rightarrow \{5\}) = -0.02$$

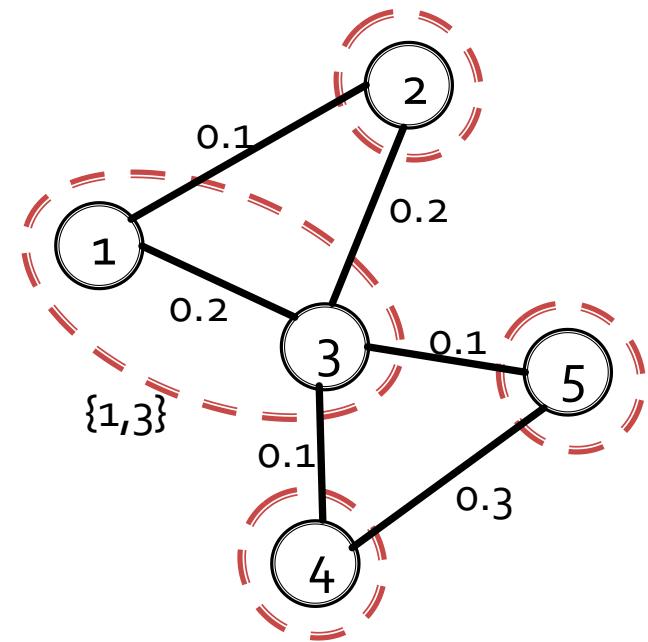
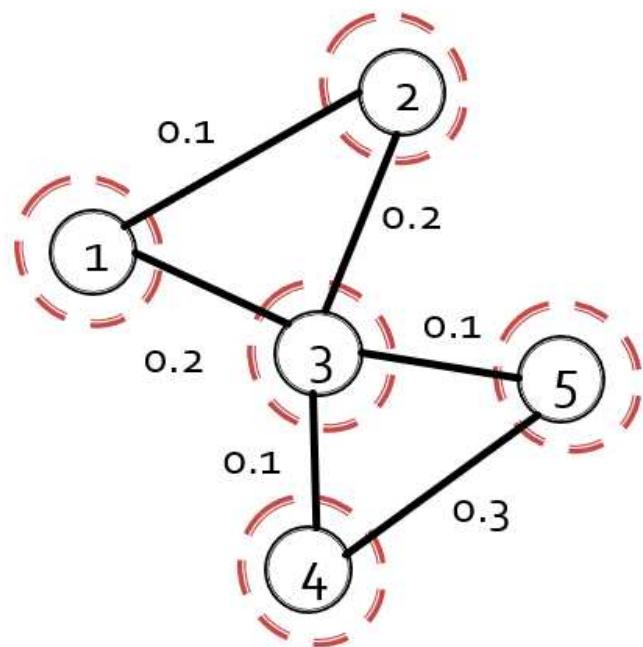
There is a tie for 3->{1} and 3->{2}.  
Randomly choose one community to proceed.

Here we choose {1}.

Move node #3 to community {1},  
now we have {1, 3}.

Sequentially process the node list  $\{3, 1, 2, 4, 5\}$

## processing node 3



Updated communities

After processing node 3, we move node 3 to  $\{1\}$ .

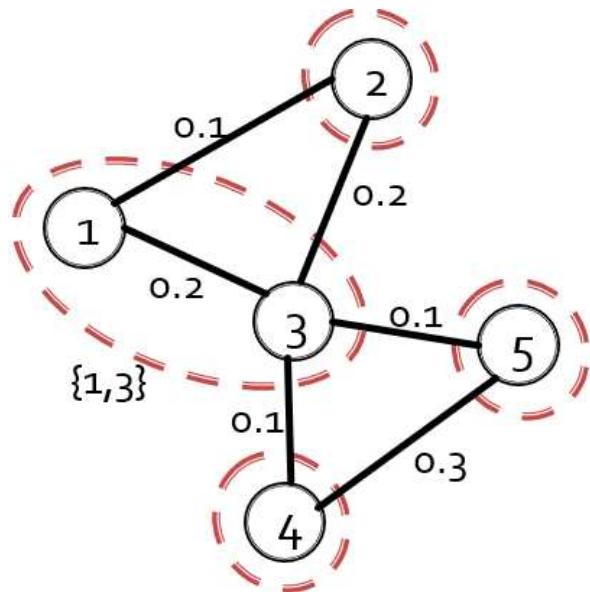
Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 1

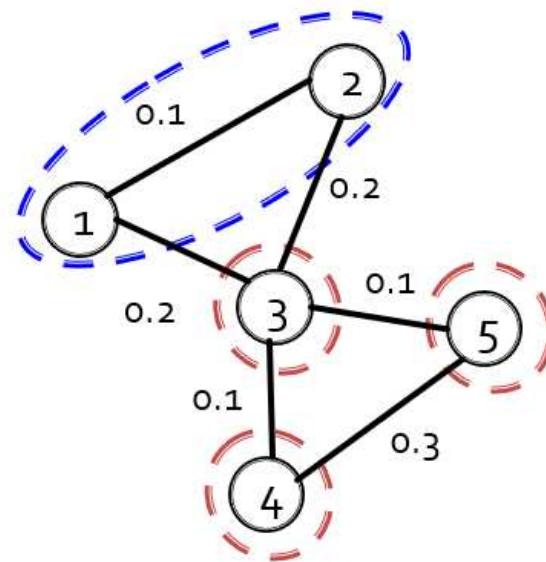
The neighbouring (directly connected) communities of node 1: Community {2}  
Only one possible movement:

Move node 1 out from {1, 3} and add node 1 to {2}:

$$\Delta Q(\{1, 3\} \rightarrow 1 \rightarrow \{2\}) = Q_{\text{after}} - Q_{\text{before}}$$



$$Q_{\text{before}} = Q(\{1, 3\}) + Q(\{2\})$$



$$Q_{\text{after}} = Q(\{3\}) + Q(\{1, 2\})$$

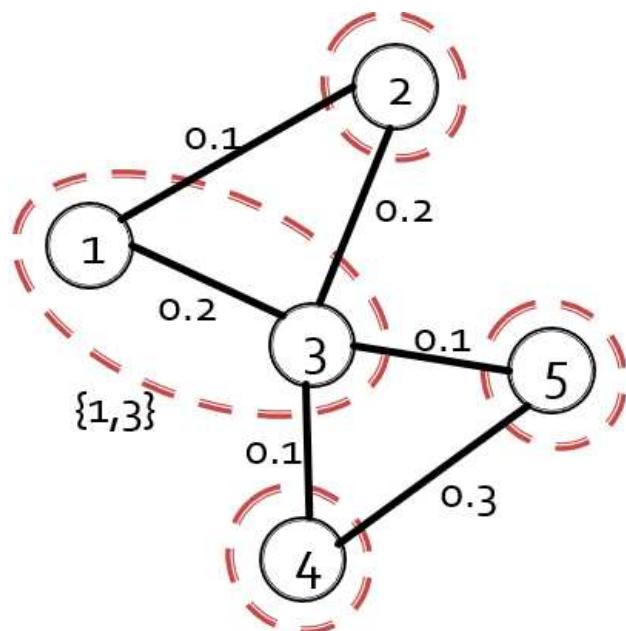
Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 1

The neighbouring (directly connected) communities of node 1: Community {2}  
Only one possible movement:

Move node 1 out from {1, 3} and add node 1 to {2}:

$$\Delta Q(\{1, 3\} \rightarrow 1 \rightarrow \{2\}) = Q_{\text{after}} - Q_{\text{before}}$$



Current communities

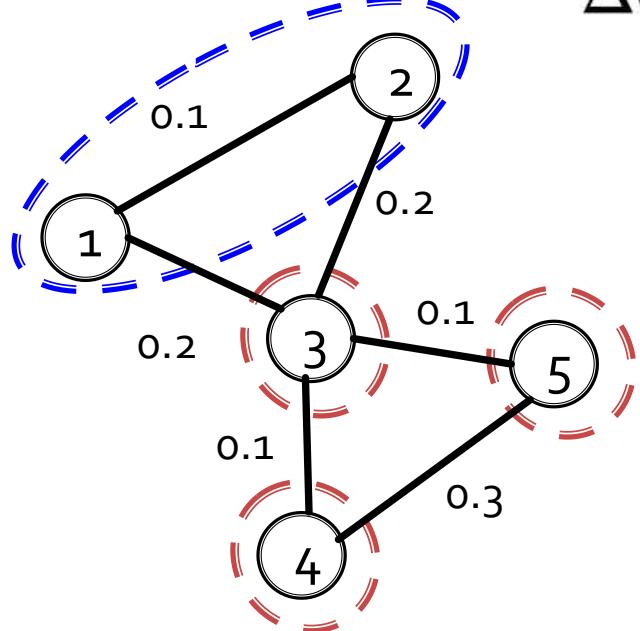
$$Q_{\text{before}} = Q(\{1, 3\}) + Q(\{2\})$$

From the previous steps,  
we already have the following values:

$$Q(\{1, 3\}) = -0.0025$$

$$Q(\{2\}) = Q(\{1\}) = -0.0225$$

## processing node 1



$$Q_{\text{after}} = Q(\{3\}) + Q(\{1, 2\})$$

Move node 1 out from  $\{1, 3\}$  and add node 1 to  $\{2\}$ :

$$\Delta Q(\{1, 3\} \rightarrow 1 \rightarrow \{2\}) = Q_{\text{after}} - Q_{\text{before}}$$

$$Q_{\text{after}} = Q(\{3\}) + Q(\{1, 2\})$$

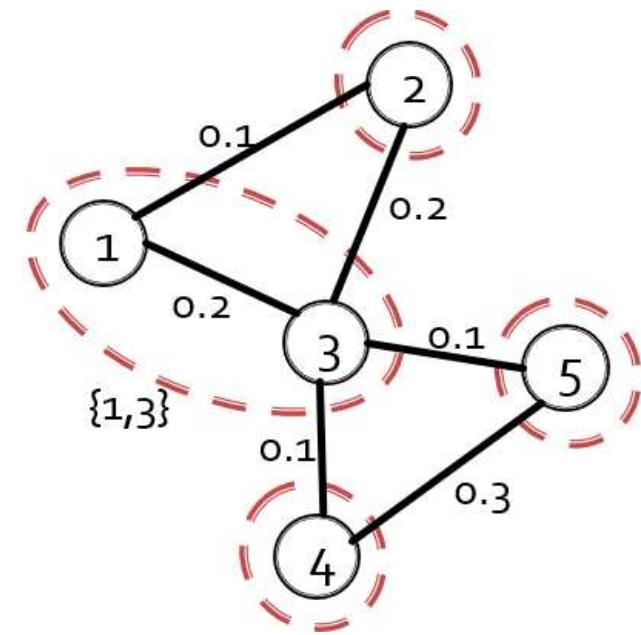
$$\begin{aligned} Q(\{1, 2\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\ &= \frac{2 \times e_{12}}{2m} - \left(\frac{k_2+k_1}{2m}\right)^2 \\ &= \frac{2 \times 0.1}{2} - \left(\frac{0.3+0.3}{2}\right)^2 \\ &= 0.01 \end{aligned}$$

From the previous steps,  
we already have the following values:

$$Q(\{3\}) = -0.09$$

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 1



Move node 1 out from {1, 3} and add node 1 to {2}:

$$\begin{aligned}\Delta Q(\{1, 3\} \rightarrow 1 \rightarrow \{2\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{3\}) + Q(\{1, 2\}) - (Q(\{1, 3\}) + Q(\{2\})) \\ &= -0.09 + 0.01 - (-0.0025 - 0.0225) \\ &= -0.055\end{aligned}$$

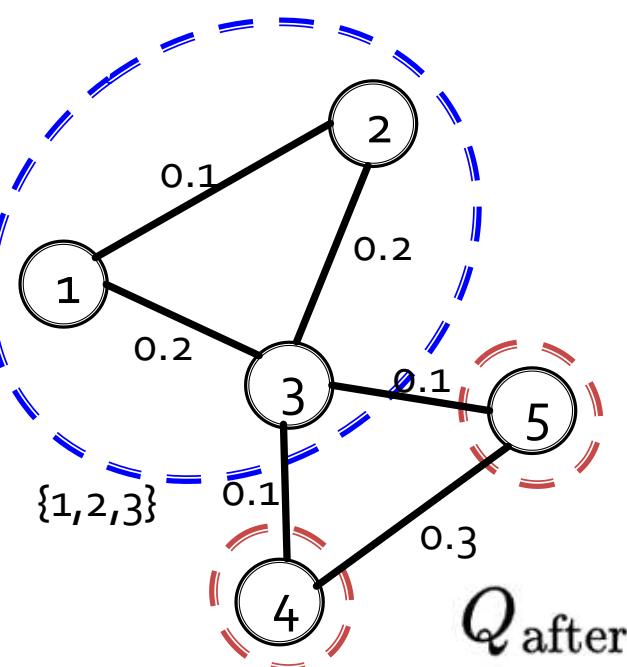
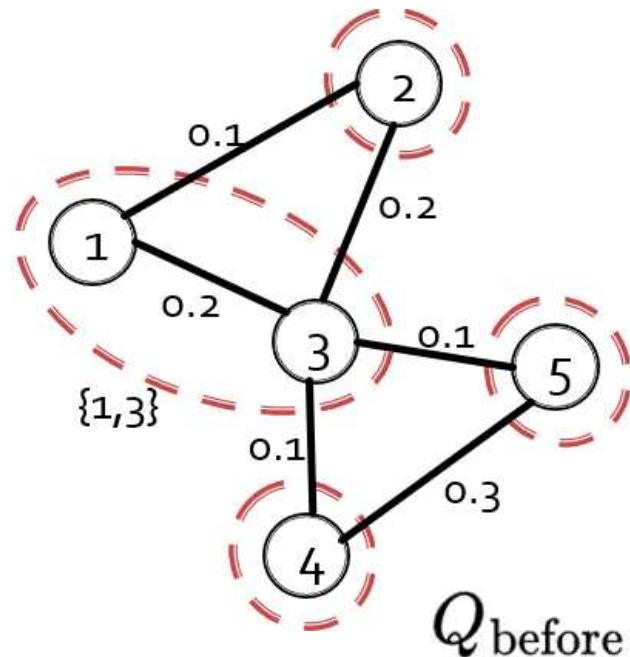
The largest gain is a negative value (there is only one neighboring community), do not move node 1 to {2}

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 2

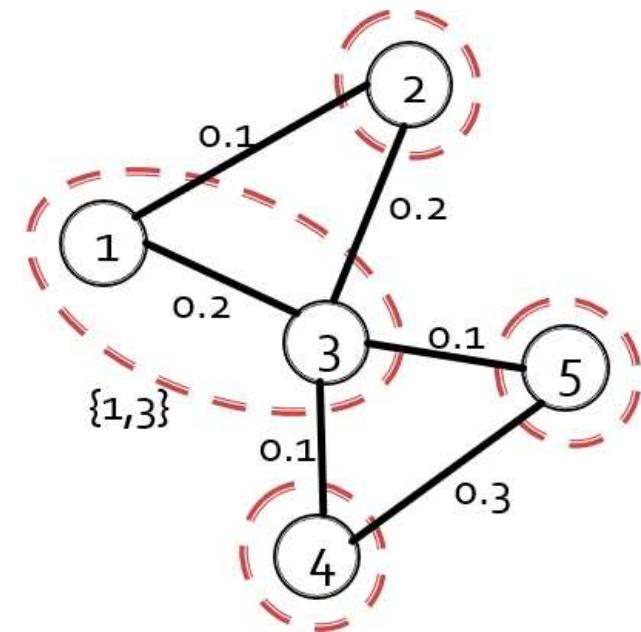
The neighbouring communities of node 2:  
Community {1, 3}. Only one possible movement:

$$\begin{aligned}\Delta Q(2 \rightarrow \{1, 3\}) &= Q_{after} - Q_{before} \\ &= Q(\{1, 2, 3\}) - Q(\{1, 3\}) - Q(\{2\})\end{aligned}$$



Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 2



Current communities

$$\begin{aligned} Q(\{2\}) &= -\left(\frac{k_2}{2m}\right)^2 \\ &= -0.0225 \end{aligned}$$

The neighbouring communities of node 2: Community {1, 3}  
Only one possible movement:

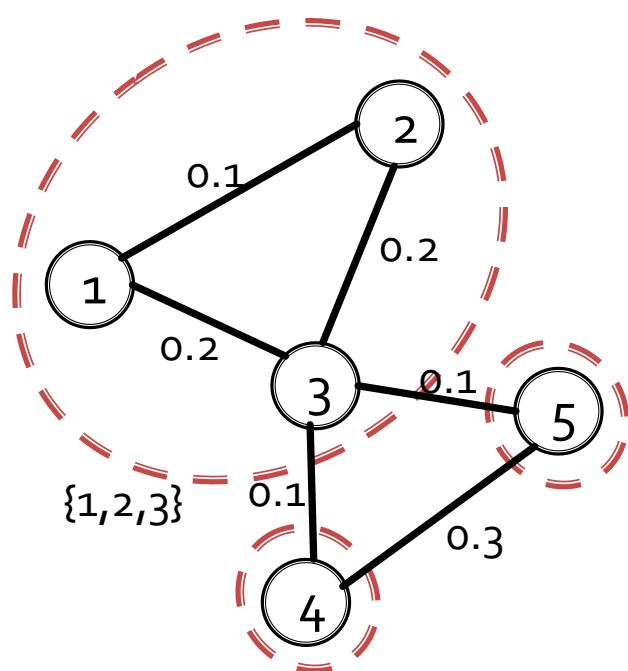
$$\begin{aligned} \Delta Q(2 \rightarrow \{1, 3\}) &= Q_{after} - Q_{before} \\ &= Q(\{1, 3\} + \{2\}) - [Q(\{1, 3\}) + Q(\{2\})] \\ &= Q(\{1, 2, 3\}) - Q(\{1, 3\}) - Q(\{2\}) \end{aligned}$$

$$\begin{aligned} Q(\{1, 2, 3\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\ &= \frac{2 \times (e_{13} + e_{21} + e_{23})}{2m} - \left(\frac{k_1 + k_2 + k_3}{2m}\right)^2 \\ &= 0.14 \end{aligned}$$

$$\begin{aligned} Q(\{1, 3\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\ &= \frac{2 \times e_{13}}{2m} - \left(\frac{k_1 + k_3}{2m}\right)^2 \\ &= -0.0025 \end{aligned}$$

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 2



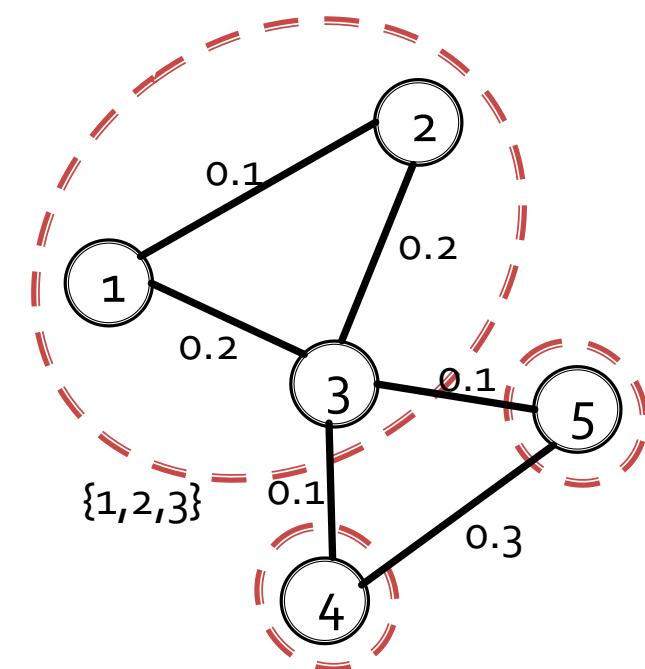
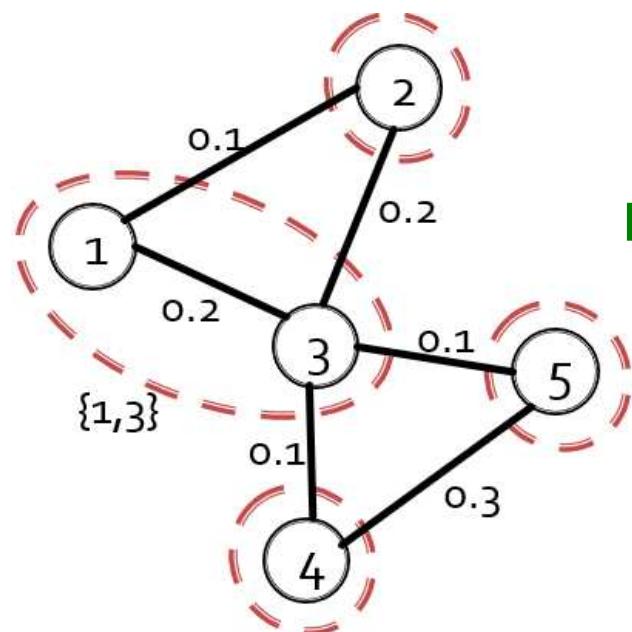
Updated communities  
after processing node 2

$$\begin{aligned}\Delta Q(2 \rightarrow \{1, 3\}) &= Q_{\text{after}} - Q_{\text{before}} \\ &= Q(\{1, 3\} + \{2\}) - [Q(\{1, 3\}) + Q(\{2\})] \\ &= Q(\{1, 2, 3\}) - Q(\{1, 3\}) - Q(\{2\}) \\ &= 0.165\end{aligned}$$

The gain is a positive value, and there is only 1 neighboring community, so it's the largest positive gain. We move 2 to {1,3}.

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 2

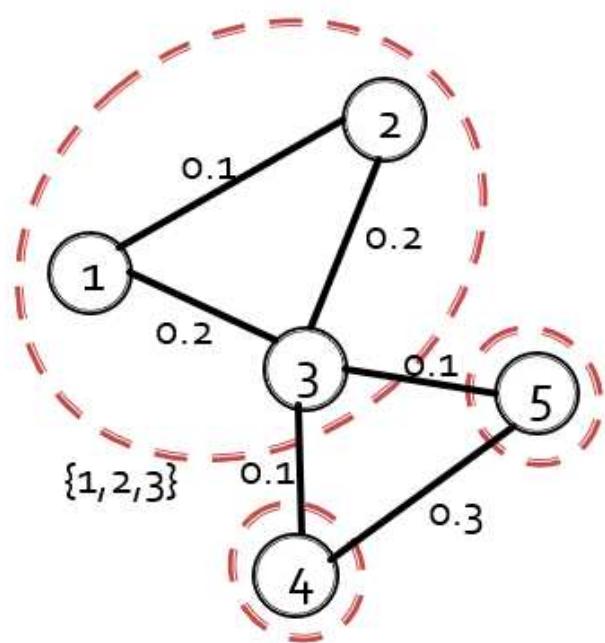


Updated communities

After processing node 2, we move node 2 to  $\{1, 3\}$ .

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 4



Current communities

The neighbouring communities of node 4:

1. Community {1, 2, 3}
2. Community {5}

There are 2 possible movements.  
We need to calculate the following two modularity gains:

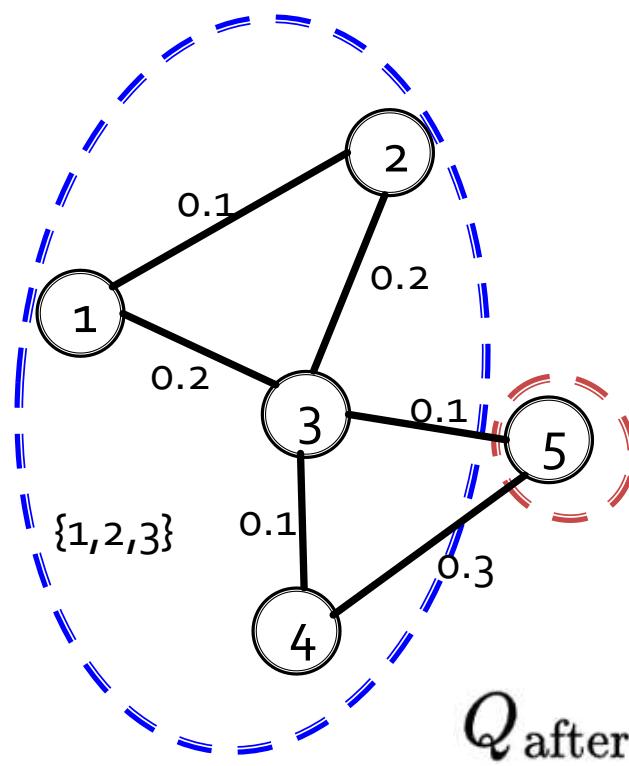
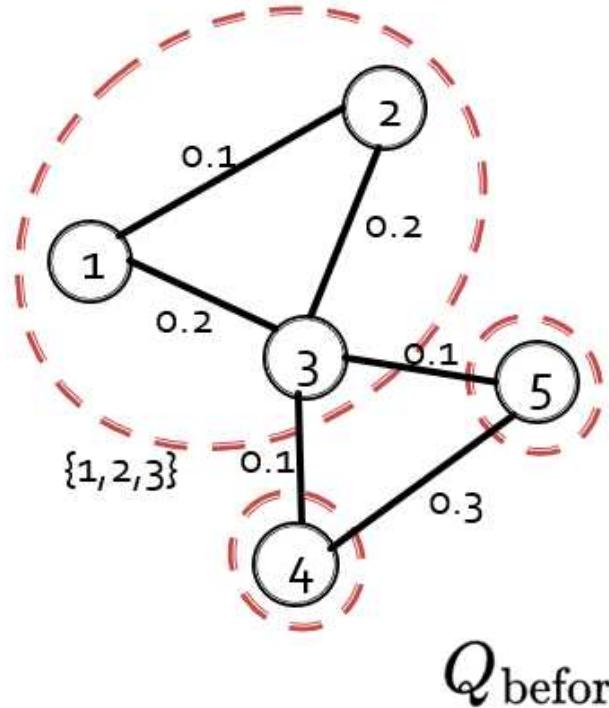
$$\Delta Q(4 \rightarrow \{1, 2, 3\})$$

$$\Delta Q(4 \rightarrow \{5\})$$

Sequentially process the node list {3, 1, 2, 4, 5}

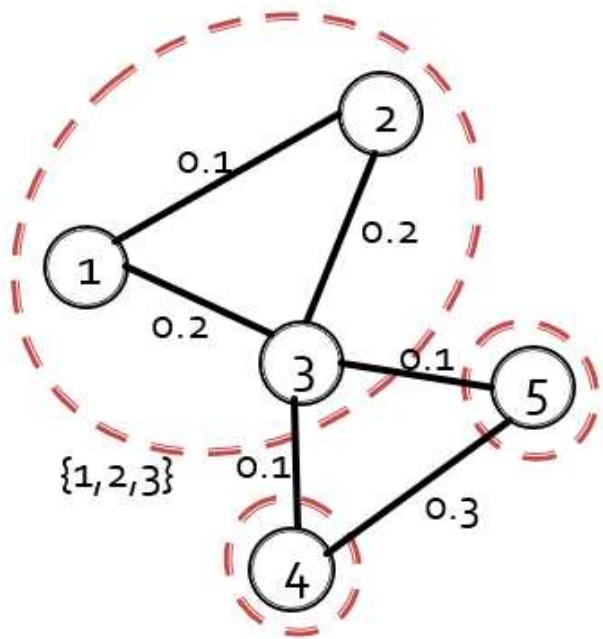
## processing node 4

$$\Delta Q(4 \rightarrow \{1, 2, 3\}) = Q_{\text{after}} - Q_{\text{before}}$$



Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 4



Current communities

$$\begin{aligned}
 \Delta Q(4 \rightarrow \{1, 2, 3\}) &= Q_{\text{after}} - Q_{\text{before}} \\
 &= Q(\{4\} + \{1, 2, 3\}) - [Q(\{4\}) + Q(\{1, 2, 3\})] \\
 &= Q(\{1, 2, 3, 4\}) - Q(\{4\}) - Q(\{1, 2, 3\}) \\
 &= (-0.04) - (-0.04) - 0.14 \\
 &= -0.14
 \end{aligned}$$

$$k_4 = 0.1 + 0.3 = 0.4$$

$$Q(\{4\}) = -\left(\frac{k_4}{2m}\right)^2 = -0.04$$

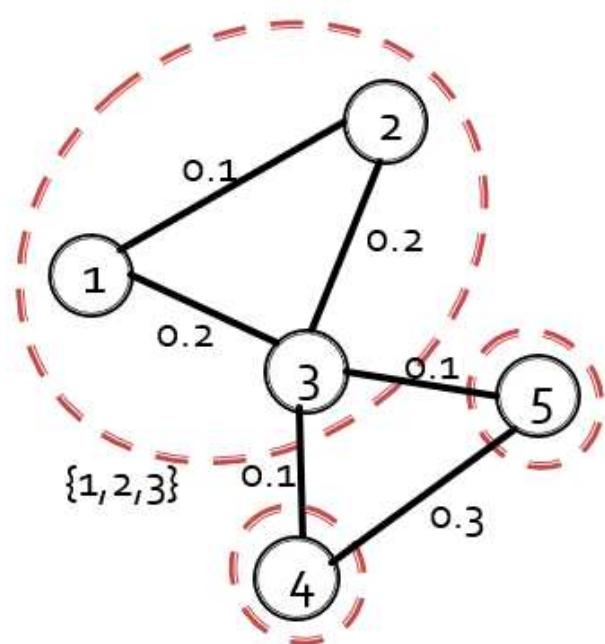
$$Q(\{1, 2, 3\}) = 0.14$$

$$\begin{aligned}
 Q(\{1, 2, 3, 4\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\
 &= \frac{2 \times (e_{12} + e_{13} + e_{23} + e_{34})}{2m} - \left(\frac{k_1 + k_2 + k_3 + k_4}{2m}\right)^2 \\
 &= \frac{2 \times (0.1 + 0.2 + 0.2 + 0.1)}{2} - \left(\frac{0.3 + 0.3 + 0.6 + 0.4}{2}\right)^2 \\
 &= -0.04
 \end{aligned}$$

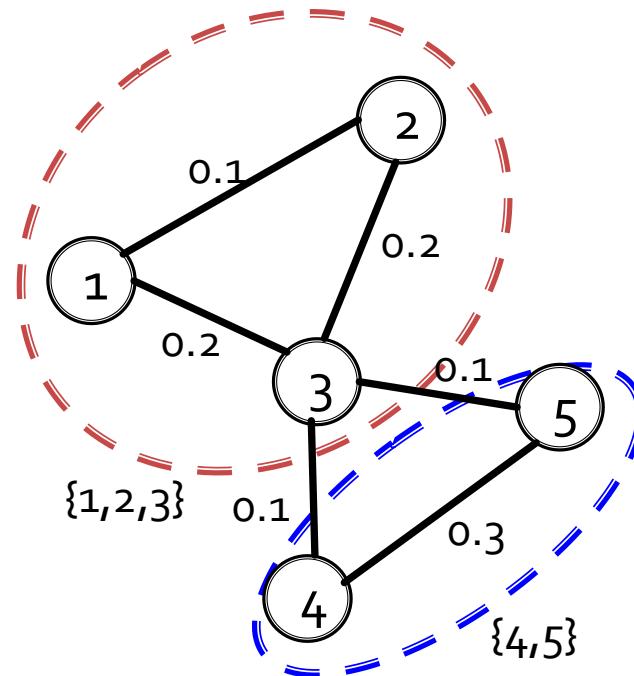
Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 4

$$\Delta Q(4 \rightarrow \{5\}) = Q_{after} - Q_{before}$$



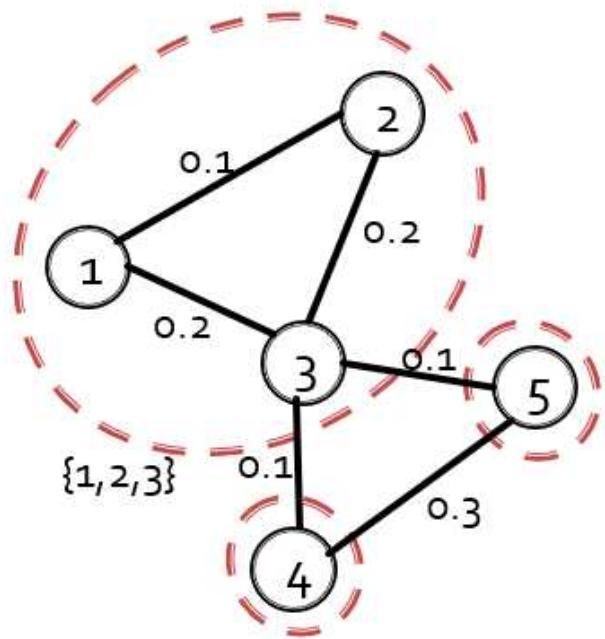
$Q_{before}$



$Q_{after}$

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 4



Current communities

$$\begin{aligned}
 \Delta Q(4 \rightarrow \{5\}) &= Q_{\text{after}} - Q_{\text{before}} \\
 &= Q(\{4\} + \{5\}) - [Q(\{4\}) + Q(\{5\})] \\
 &= Q(\{4, 5\}) - Q(\{4\}) - Q(\{5\}) \\
 &= 0.14 - (-0.04) - (-0.04) \\
 &= 0.22
 \end{aligned}$$

$$Q(\{4\}) = -\left(\frac{k_4}{2m}\right)^2 = -0.04$$

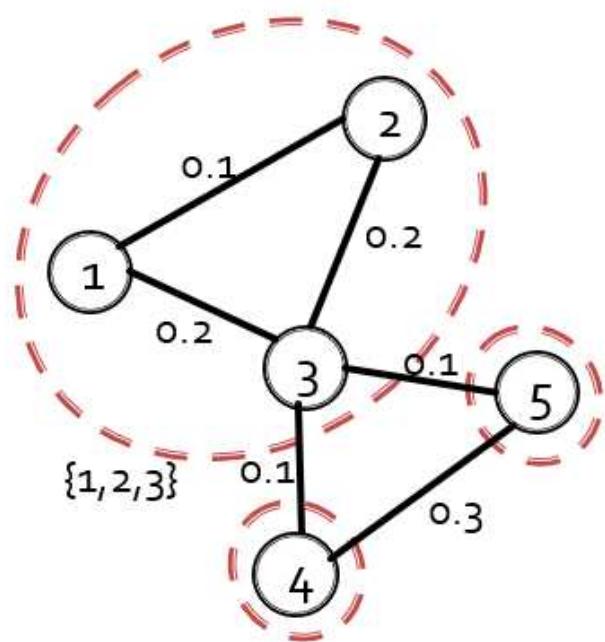
$$k_5 = 0.1 + 0.3 = 0.4$$

$$Q(\{5\}) = -\left(\frac{k_5}{2m}\right)^2 = -0.04$$

$$\begin{aligned}
 Q(\{4, 5\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\
 &= \frac{2 \times e_{45}}{2m} - \left(\frac{k_4 + k_5}{2m}\right)^2 \\
 &= 0.14
 \end{aligned}$$

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 4



Current communities

Summary:  
Scores for the 2 possible movements

$$\Delta Q(4 \rightarrow \{1, 2, 3\}) = -0.14$$

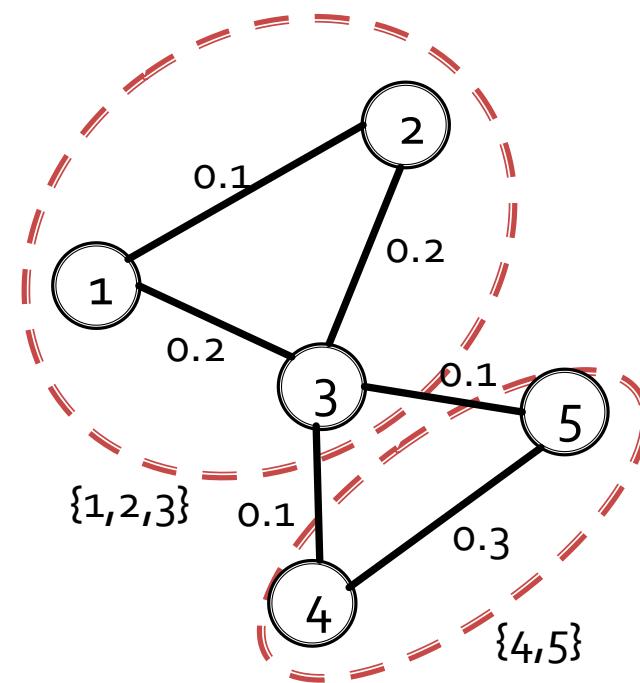
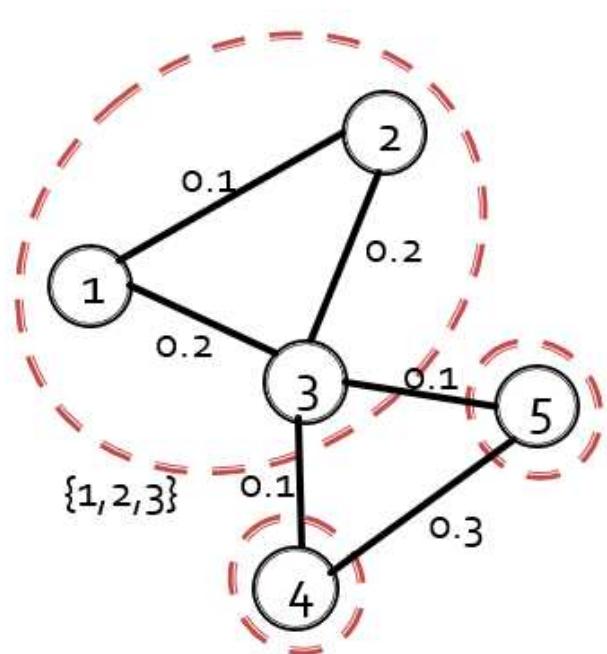
$$\Delta Q(4 \rightarrow \{5\}) = 0.22$$

$\Delta Q(4 \rightarrow \{5\})$  has the largest positive gain.

We move node 4 to  $\{5\}$ .

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 4

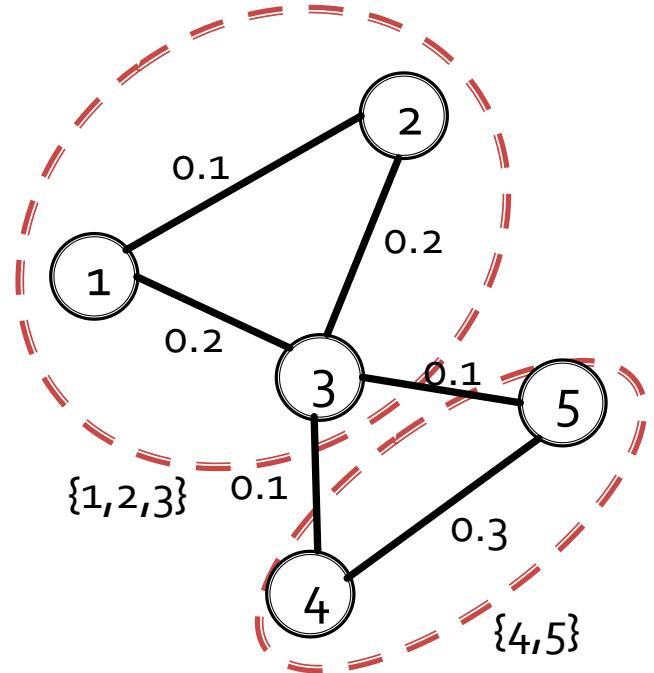


updated communities

After processing node 4, we move node 4 to  $\{5\}$ .

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 5



Current communities

The neighbouring communities of node 5: Community {1, 2, 3}

There is only one neighbouring community for node 5, so there is only one possible movement

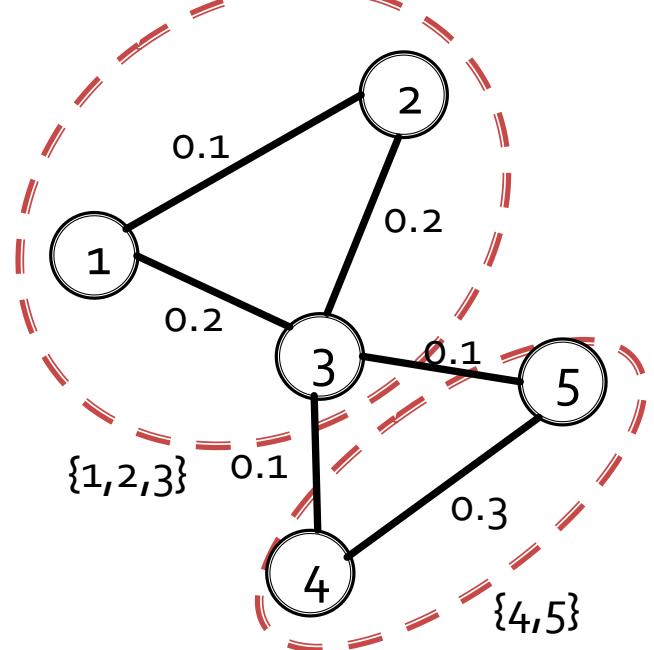
We need to calculate the following modularity gain:

$$\Delta Q(\{4, 5\} \rightarrow 5 \rightarrow \{1, 2, 3\})$$

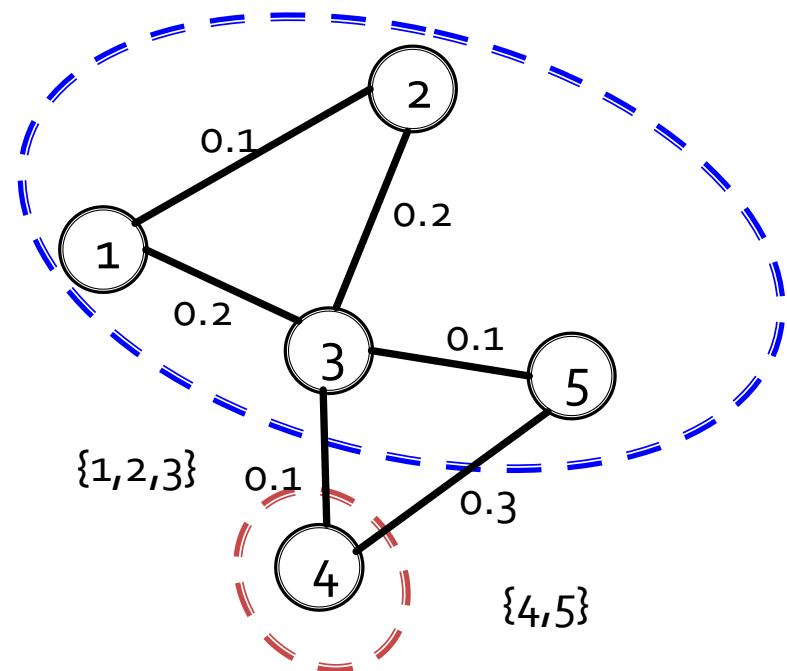
Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 5

$$\Delta Q(\{4, 5\} \rightarrow 5 \rightarrow \{1, 2, 3\})$$



$Q_{\text{before}}$



$Q_{\text{after}}$

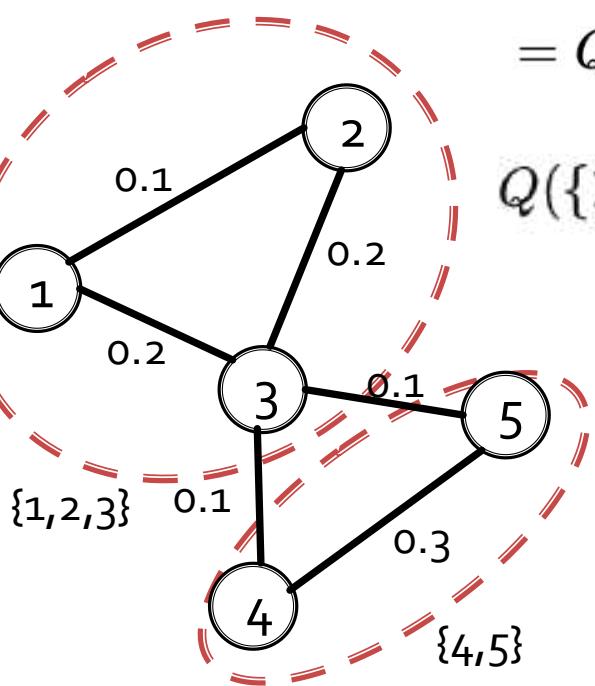
Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 5

$$\Delta Q(\{4, 5\} \rightarrow 5 \rightarrow \{1, 2, 3\}) = Q_{\text{after}} - Q_{\text{before}}$$

$$= Q(\{4\}) + Q(\{1, 2, 3, 5\}) - [Q(\{4, 5\}) + Q(\{1, 2, 3\})]$$

$$\begin{aligned} Q(\{1, 2, 3, 5\}) &= \frac{\Sigma_{\text{in}}}{2m} - \left(\frac{\Sigma_{\text{tot}}}{2m}\right)^2 \\ &= \frac{2 \times (e_{12} + e_{13} + e_{23} + e_{35})}{2m} - \left(\frac{k_1 + k_2 + k_3 + k_5}{2m}\right)^2 \\ &= \frac{2 \times (0.1 + 0.2 + 0.2 + 0.1)}{2} - \left(\frac{0.3 + 0.3 + 0.6 + 0.4}{2}\right)^2 \\ &= -0.04 \end{aligned}$$



Current communities

The below values can be obtained from previous steps:

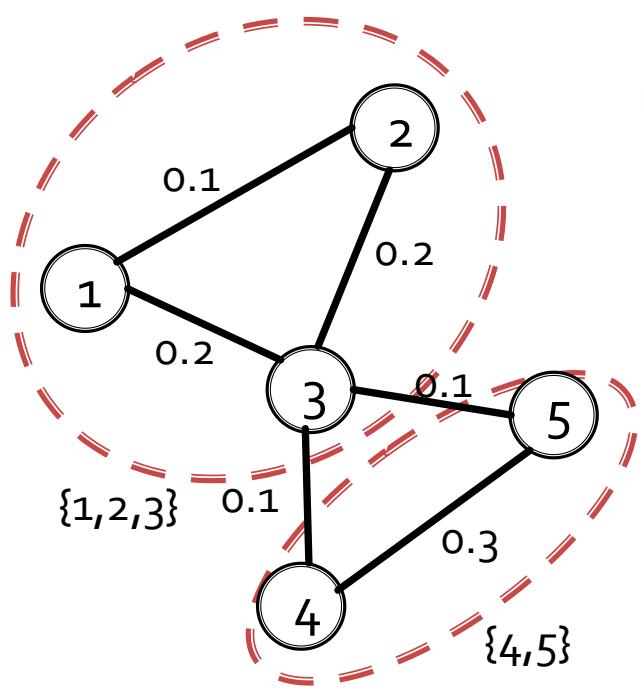
$$Q(\{4\}) = -\left(\frac{k_4}{2m}\right)^2 = -0.04$$

$$Q(\{4, 5\}) = 0.14$$

$$Q(\{1, 2, 3\}) = 0.14$$

Sequentially process the node list {3, 1, 2, 4, 5}

## processing node 5



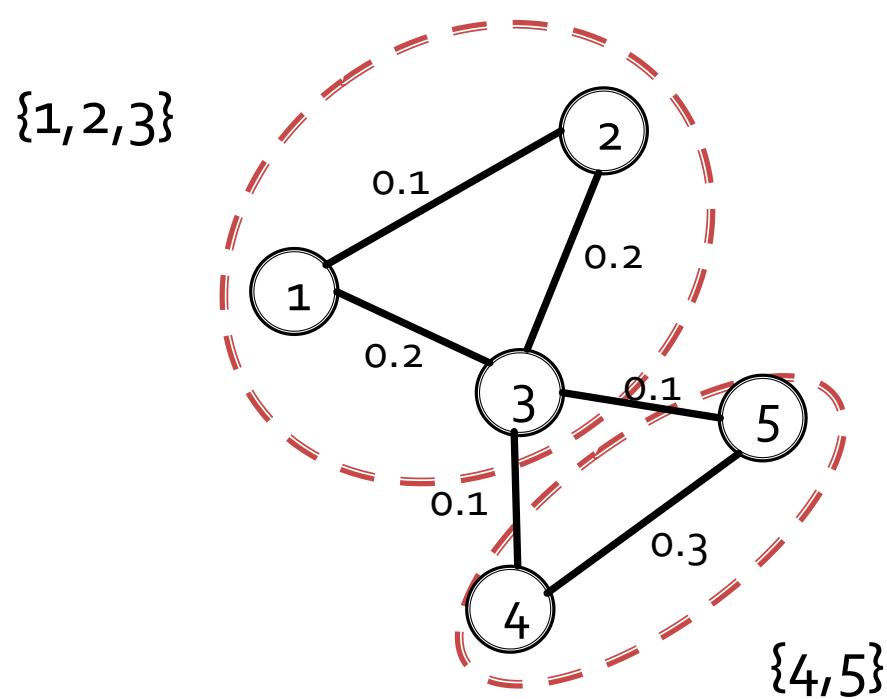
Current communities

$$\begin{aligned}\Delta Q(\{4, 5\} \rightarrow 5 \rightarrow \{1, 2, 3\}) \\ &= Q_{after} - Q_{before} \\ &= Q(\{4\}) + Q(\{1, 2, 3, 5\}) - [Q(\{4, 5\}) + Q(\{1, 2, 3\})] \\ &= -0.04 - 0.04 - (0.14 + 0.14) = -0.36\end{aligned}$$

The largest gain is a negative value  
(there is one neighboring community).  
Do not move node 5.

# Result

After processing the node list  $\{3, 1, 2, 4, 5\}$ , we have two communities:  $\{1, 2, 3\}$  and  $\{4, 5\}$ .



Additional discussion: (beyond the question)

We have finished the 1<sup>st</sup> scan after processing the node list.

To fully complete community generation, continue to do the 2<sup>nd</sup> scan, 3<sup>rd</sup> scan, ....

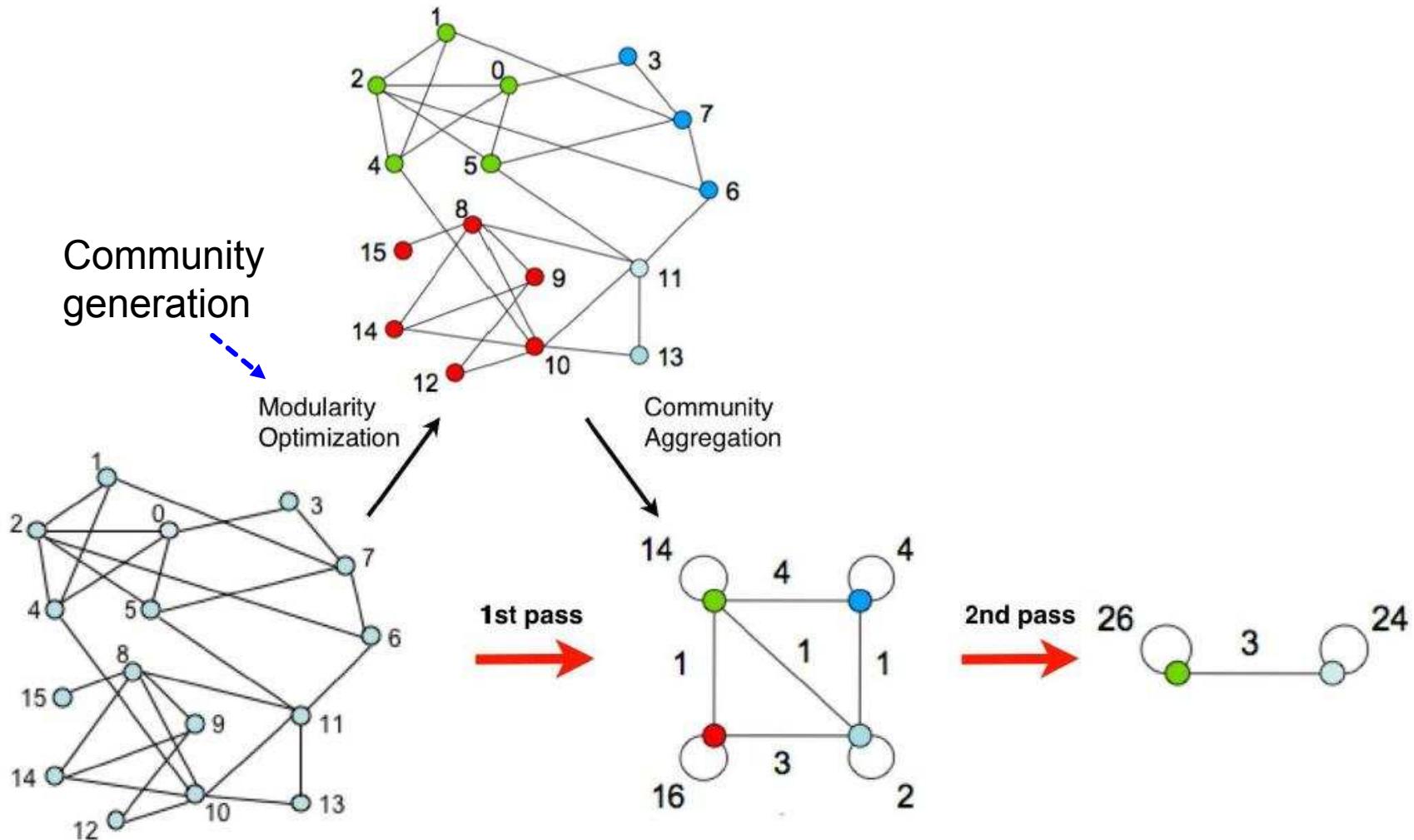
In each scan, we randomly generate a node list for processing.

Converge:

if there is no node movement in one scan, the community generation step converges.



- Multi-pass Louvain Algorithm
  - Multiple passes.
    - Produce hierarchical results
    - One pass corresponds to one hierarchical level
  - Each pass is made of 2 phases:
    - Phase 1: community generation (we have learnt)
    - Phase 2: community aggregation



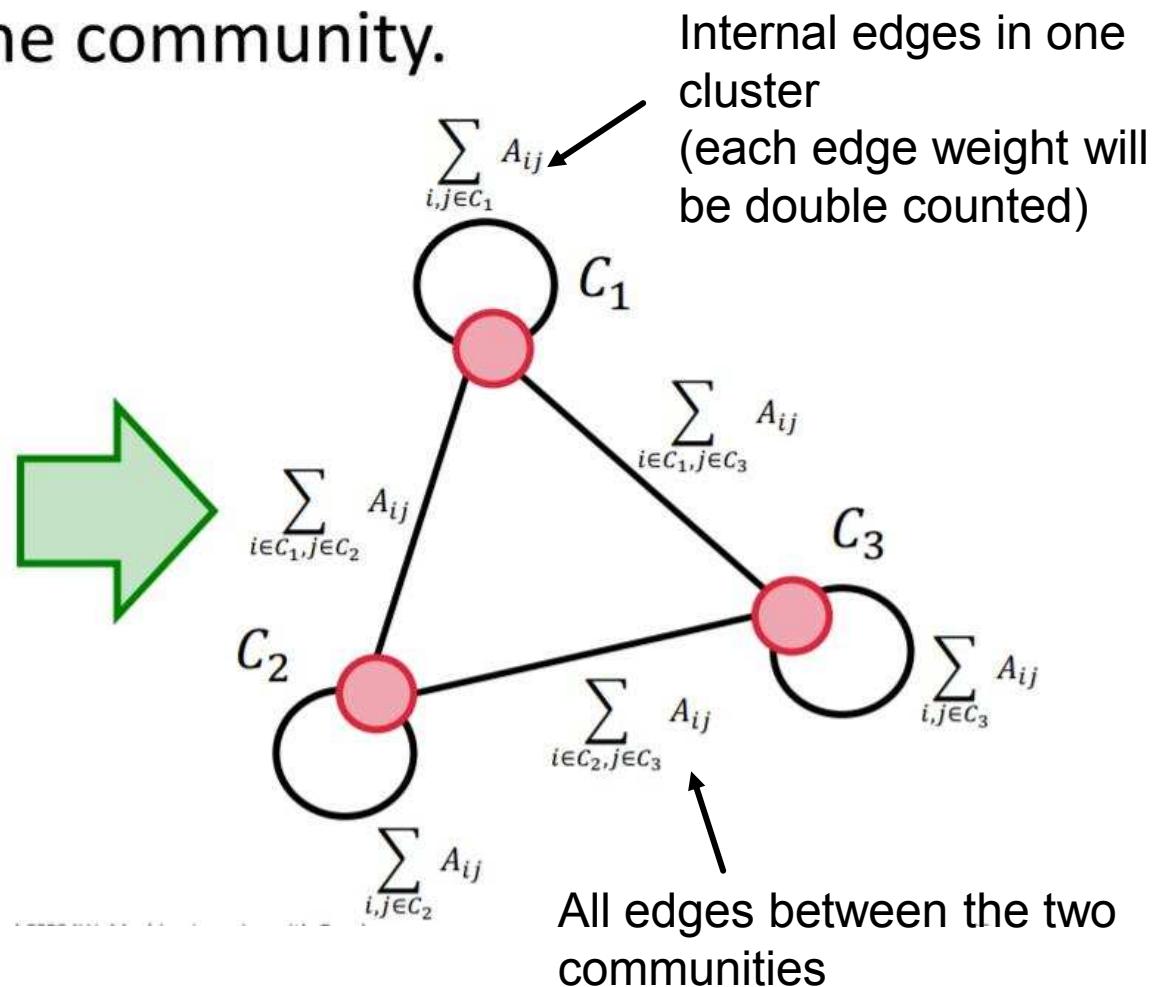
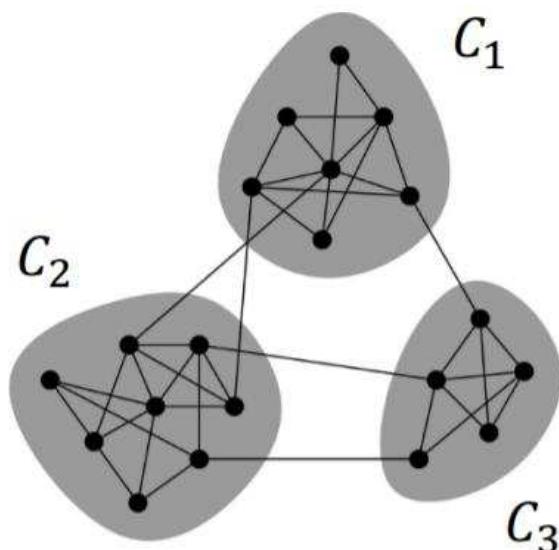
Multi-pass Louvain Algorithm high-level illustration

For un-weighted graph: the weight for each edge is 1

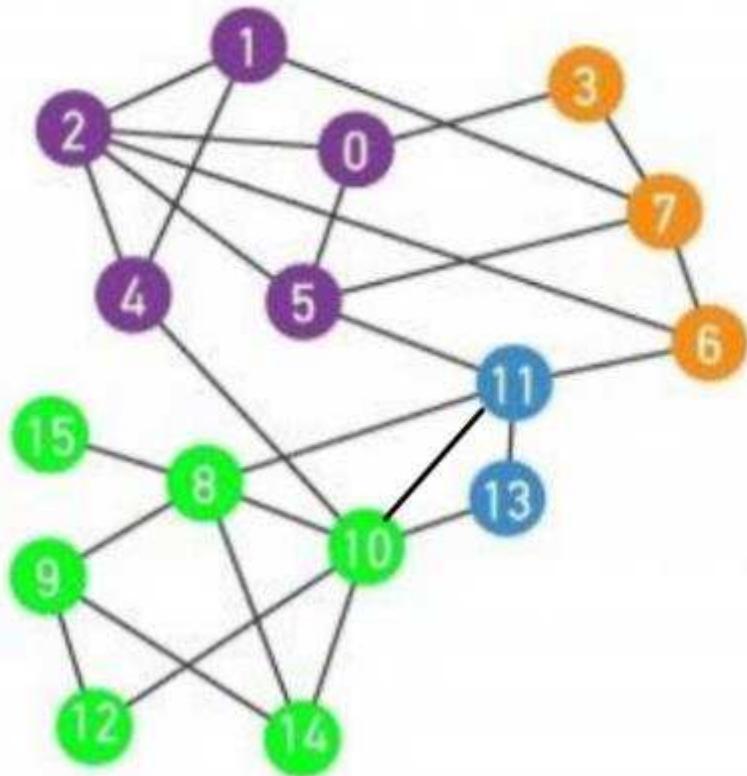
- Phase 2: community aggregation
  - Generate a new graph based on the communities
    - Create one node (super node) for one community in the new graph
    - Create edges of the super nodes if the original communities are connected by at least one edge
      - The weight of the edge between the two super nodes is the sum of the edge weights between the original communities
    - Create self-connections of the super nodes based on the internal connections of the original communities.

- Super nodes are constructed by merging nodes in the same community.

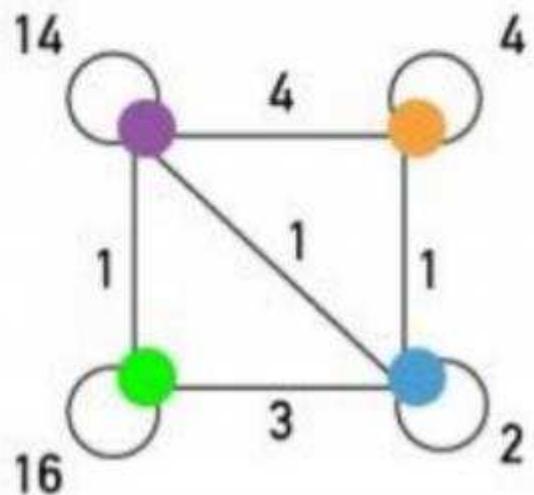
Community assignment obtained after 1<sup>st</sup> phase



# Example



STEP II



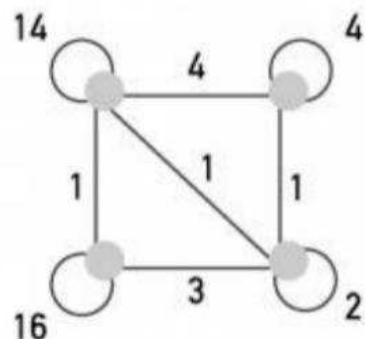
generate communities in phase 1 (step 1)

generate new graph in phase 2 (step 2)

The 1<sup>st</sup> pass.

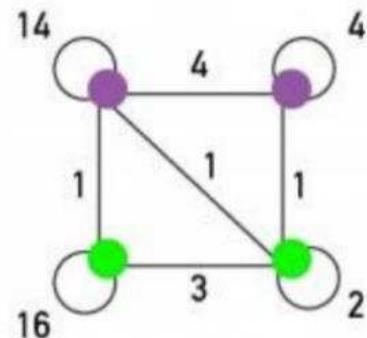
# Example

2<sup>ND</sup> PASS

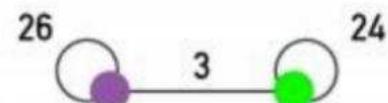


**phase 1 (step1):** community generation  
by modularity optimization

STEP I



STEP II



**phase 2 (step2):** merge nodes and  
generate a new graph

In the 2<sup>nd</sup> pass:

run phase 1 (step 1 in the figure) and phase 2 (step 2) again on the new graph

## ■ Further reading

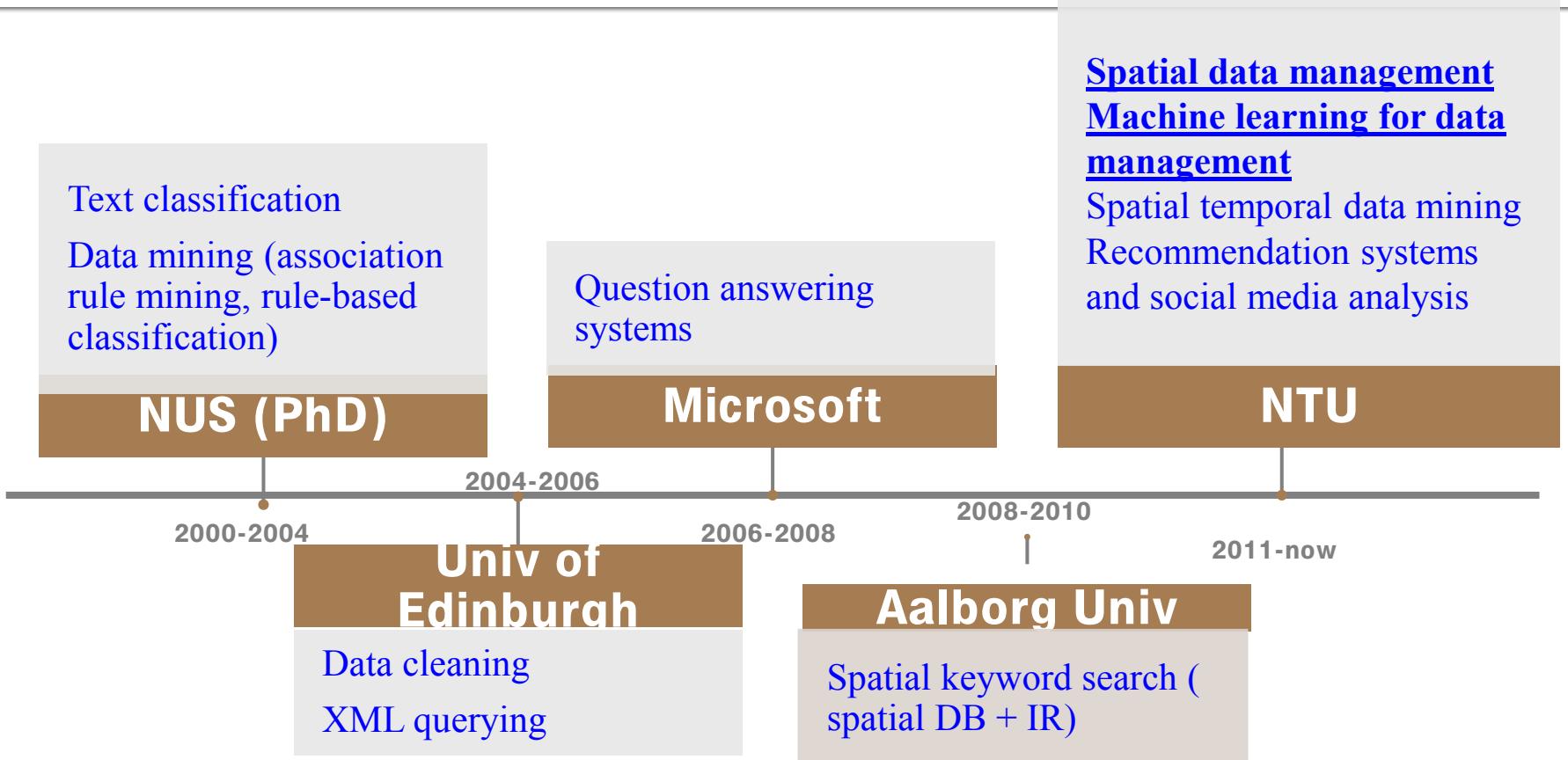
- BigCLAM
  - For overlapping community detection
- Spectral clustering
  - A well-known clustering method on graph
  - <http://snap.stanford.edu/class/cs224w-2019/>
  - [https://en.wikipedia.org/wiki/Spectral\\_clustering](https://en.wikipedia.org/wiki/Spectral_clustering)

# **Introduction to second half of SC4020-CE4032-CZ4032 Data Analytics and Mining**

# About me

- Instructor: Prof Cong Gao
- Specialized in data science (data management, data mining)
- Email: [gaocong@ntu.edu.sg](mailto:gaocong@ntu.edu.sg)
- Homepage:  
<https://personal.ntu.edu.sg/gaocong/>
- Office: N4-2c-103
- **Emails: Pls put [SC4020] as email title**
- **You are encouraged to use Discussion Board in the course website so that others can read/answer your questions as well**

# My Research Areas



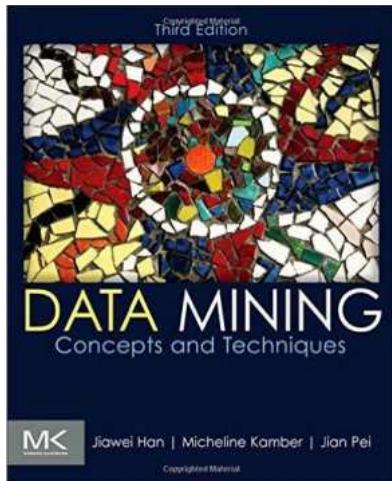
# Topics of second half/ tutorial

- Lecture:
  - Association Rule Mining (week 7—8)
  - Sequential pattern mining, Classification, and overfitting (week 9--10)
  - Recommendation Systems (Week 11)
  - Data Processing, Data Cleaning and integration (Week 12)
  - Advanced topic (Week 12)
- **Tutorial will start from Week 7. No tutorial in Week 11**

# Assessment (some content is from first lecture)

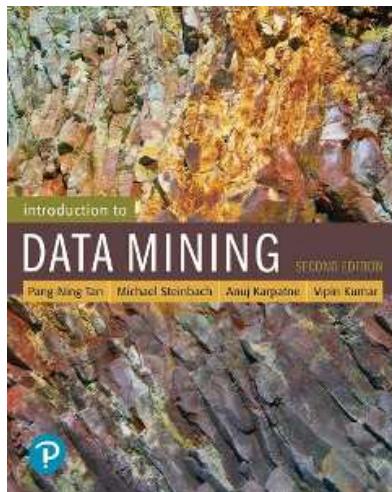
- Final exam (50%) + Assignments (50%)
  - Closed book exam
- Group-based Assignment for 2nd half: 25%
  - Will release toward the end of Week 8
  - Due before the start of final exam (21 Nov, Thursday)
  - Individual Contribution Claim (agreed by all)
  - If there is a large difference, get different scores
  - Plagiarism → 0 marks.

# Two Additional Reference Books



**Data Mining: Concepts and Techniques (3rd ed),  
Morgan Kaufmann, 2011**

- **Jiawei Han, Micheline Kamber and Jian Pei**



**Introduction to Data Mining (Second Edition)**

- **by Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar**

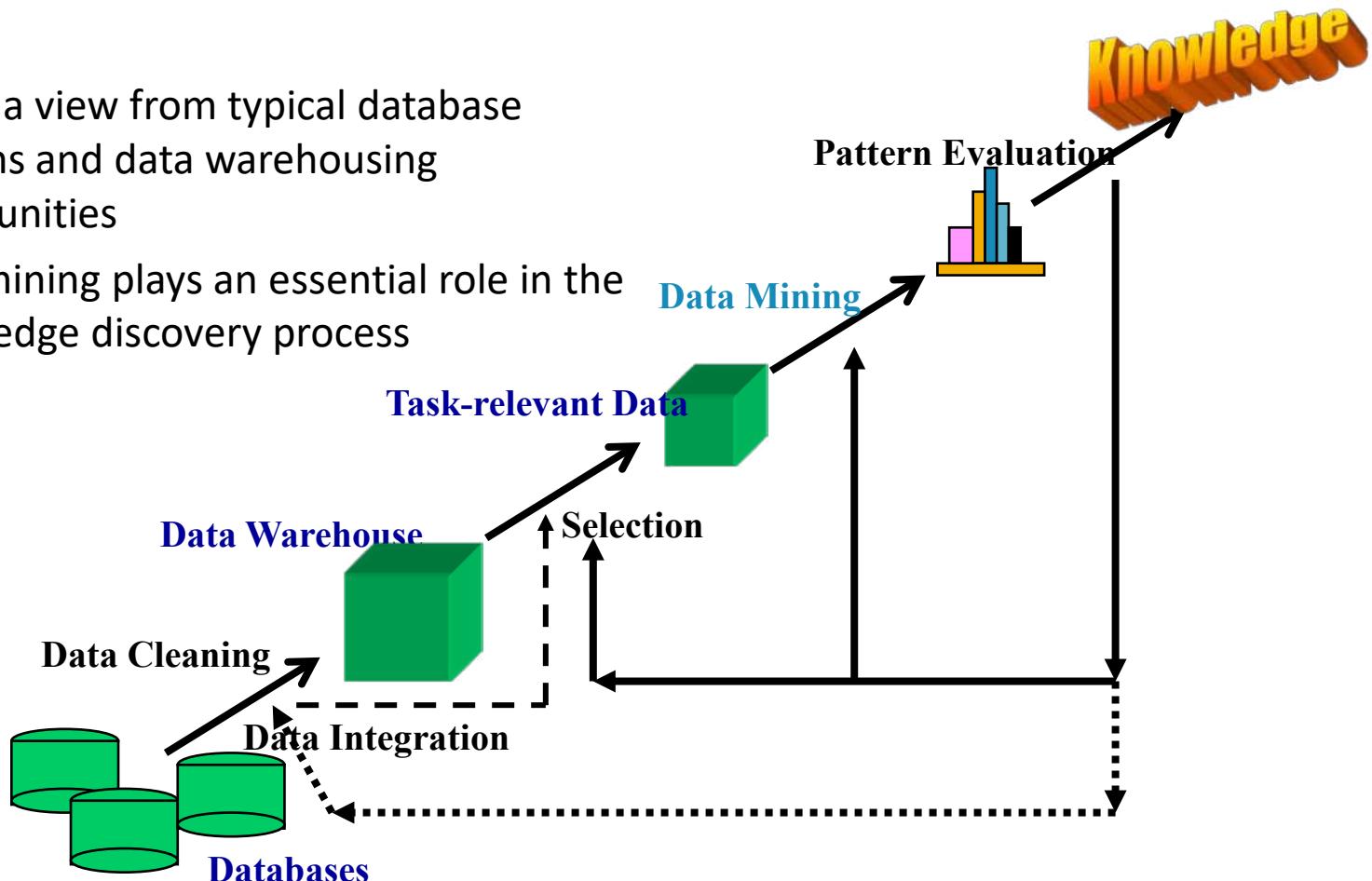
# Additional slides

---

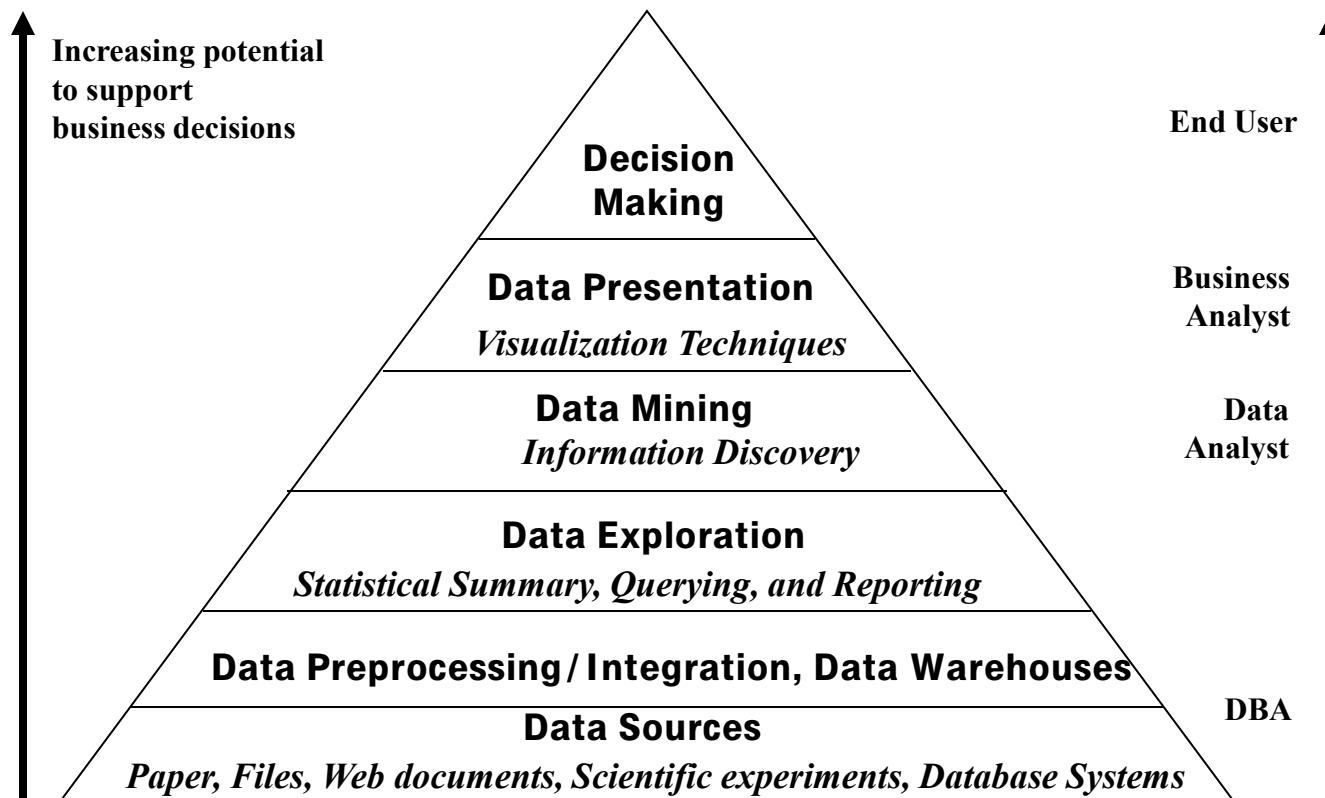
- The rest slides are just for your information

# Knowledge Discovery (KDD) Process

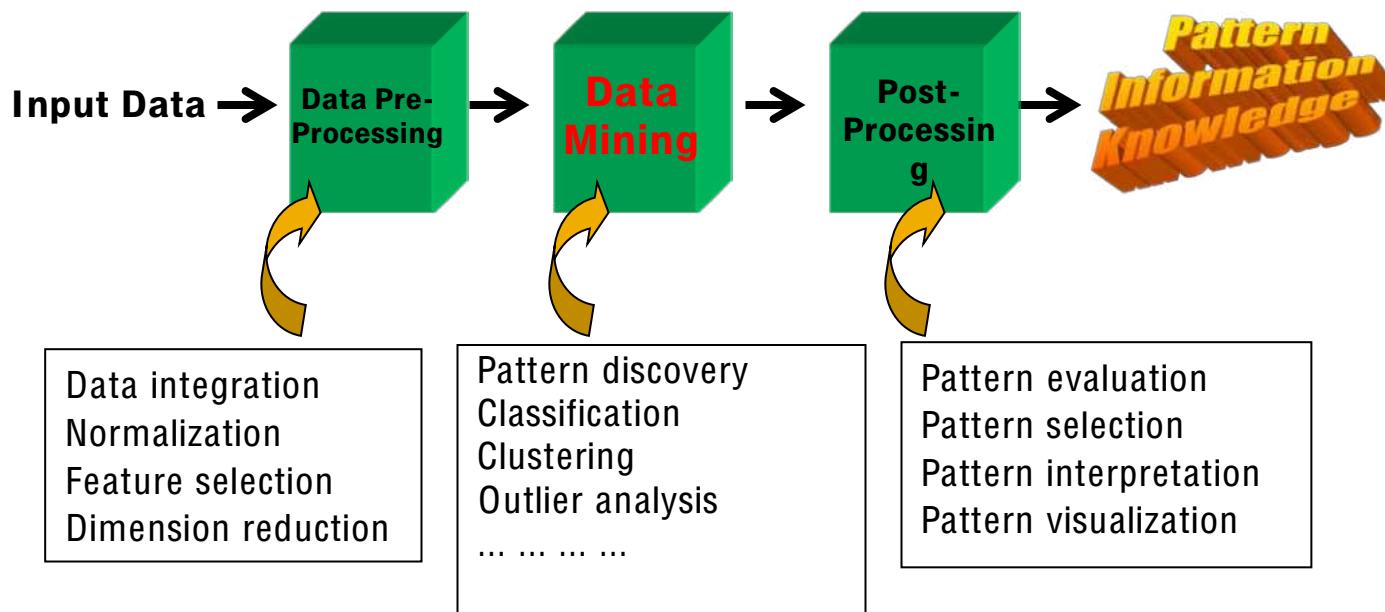
- This is a view from typical database systems and data warehousing communities
- Data mining plays an essential role in the knowledge discovery process



# Data Mining in Business Intelligence



# KDD Process: A View from ML and Statistics



- This is a view from typical machine learning and statistics communities

# A Brief History of Data Mining Society (for your reference)

- 1989 IJCAI Workshop on Knowledge Discovery in Databases
  - Knowledge Discovery in Databases (G. Piatetsky-Shapiro and W. Frawley, 1991)
- 1991-1994 Workshops on Knowledge Discovery in Databases
  - Advances in Knowledge Discovery and Data Mining (U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, 1996)
- 1995-1998 International Conferences on Knowledge Discovery in Databases and Data Mining (KDD'95-98)
  - Journal of Data Mining and Knowledge Discovery (1997)
- ACM SIGKDD conferences since 1998 and SIGKDD Explorations
- More conferences on data mining
  - PAKDD (1997), PKDD (1997), SIAM-Data Mining (2001), (IEEE) ICDM (2001), WSDM (2008), etc.
- ACM Transactions on KDD (2007)

# **Frequent Itemset Mining & Association Rules (Part 1)**

Slides adapted from Stanford , UIUC data mining course, and textbook slides from Kumar etc

# Outline

- **Basics**
  - Introduction & Application
  - Frequent itemsets & Association rules
- **Algorithms for finding frequent itemsets**
  - A-Priori algorithm
- **Implementation of Algorithms**
  - Finding frequent pairs
  - A-Priori algorithm
  - PCY algorithm
- **Interestingness of association rules, other types of itemsets**

# Intro to Association Rule Discovery

## Supermarket shelf management – Market-basket model:

- **Goal:** Identify items that are bought together by sufficiently many customers
- **Approach:** Process the sales data collected with barcode scanners to find dependencies among items
- **A classic rule:**
  - If someone buys diaper and milk, then he/she is likely to buy beer
  - Don't be surprised if you find beer next to diapers!

# The Market-Basket Model

- A large set of **items**

- e.g., things sold in a supermarket

- A **large set of baskets or transactions**

- Each basket/transaction is a **small subset of items**

- e.g., the things one customer buys on one day

- Want to discover **association rules**

- People who bought {x,y,z} tend to buy {v,w}
  - Amazon!

Input:

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

Output:

**Rules Discovered:**

{Milk} --> {Coke}

{Diaper, Milk} --> {Beer}

# Applications – (1)

- **Items** = products; **Baskets** = sets of products bought in one trip to the store
- **Real market baskets:** TBs of data about what customers buy together
  - Tells how typical customers navigate stores, lets them position tempting items
  - Suggests tie-in “tricks”, e.g., run sale on diapers and raise the price of beer

# Applications – (2)

- **Baskets** = sentences; **Items** = words containing in each sentences
  - Frequent set of items may represent phrase, which can be used for phase detection in natural language processing applications
- **Baskets** = patients; **Items** = genes and values
  - Frequent set of items may represent that a combination of different genes and their values, which may indicate some illness

# Other applications

- It can be used for other data mining tasks, such as
  - Classification
  - Clustering
  - Outlier detection
- Extend to other type of data, e.g.,
  - From a set of graphs, mine frequent subgraphs (with applications in chemical applications and cyber security)

# Frequent Itemsets/Patterns

- **Task:** Find sets of items that appear together “frequently” in baskets
- **Support** for itemset  $I$ : Number of transactions containing all items in  $I$ , denoted by  $support(I)$ 
  - (can also be expressed as a fraction of the total number of transactions)
- Given a **support threshold  $minsup$** , then sets of items that appear in at least  $minsup$  baskets are called **frequent itemsets (or patterns)**

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

Support of  
 $\{Beer, Bread\} = 2$

# Example: Frequent Itemsets

- Items = {milk, coke, pepsi, beer, juice}
- Support threshold = 3 baskets

$$B_1 = \{m, c, b\}$$

$$B_3 = \{m, b\}$$

$$B_5 = \{m, p, b\}$$

$$B_7 = \{c, b, j\}$$

$$B_2 = \{m, p, j\}$$

$$B_4 = \{c, j\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_8 = \{b, c\}$$

- ~~Frequent itemsets:~~  $\{\text{m}\}, \{\text{c}\}, \{\text{b}\}, \{\text{j}\}, \{\text{m}, \text{b}\}, \{\text{b}, \text{c}\}, \{\text{c}, \text{j}\}.$

# Association Rules

- **Association Rules:**
  - If-then rules about the contents of transactions
- $\{i_1, i_2, \dots, i_k\} \rightarrow j$  means: “if a transaction contains all of  $i_1, \dots, i_k$  then it is *likely* to contain  $j$ ”
- **In practice there are too many rules**
  - Not all the generated rules are interesting
  - **want to find significant/interesting ones!**
- **Confidence** of this association rule is the probability of  $j$  given  $I = \{i_1, \dots, i_k\}$

$$conf(I \rightarrow j) = \frac{support(I \cup j)}{support(I)}$$

# Example: Confidence

Data:

$$B_1 = \{m, c, b\}$$

$$B_3 = \{m, b\}$$

$$B_5 = \{m, p, b\}$$

$$B_7 = \{c, b, j\}$$

$$B_2 = \{m, p, j\}$$

$$B_4 = \{c, j\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_8 = \{b, c\}$$

$$conf(I \rightarrow j) = \frac{support(I \cup j)}{support(I)}$$

- Association rule:  $\{m\} \rightarrow b$ 
  - Confidence =  $4/5 = 0.8$
- Association rule:  $\{m, b\} \rightarrow c$ 
  - Confidence =  $2/4 = 0.5$

# Association Rule Mining Task

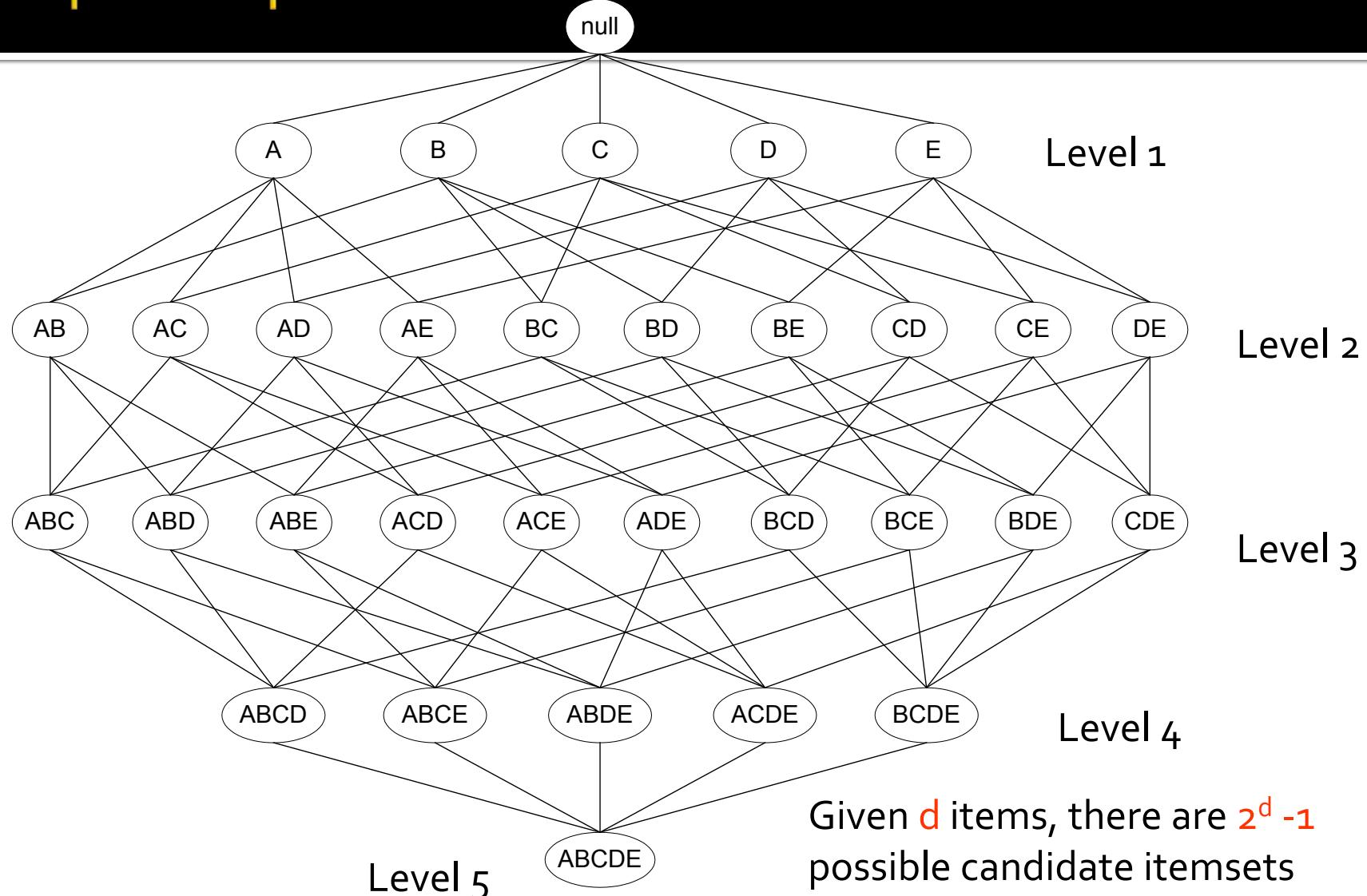
- **Problem:** Given a set of transactions T, the goal of association rule mining is to find all rules having
  - $\text{support} \geq \text{minsup}$  threshold
  - $\text{confidence} \geq \text{minconf}$  threshold
- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the  $\text{minsup}$  and  $\text{minconf}$  thresholds

⇒ Computationally prohibitive!

# Mining Association Rules

- **Step 1: Find all frequent itemsets  $I$** 
  - Generate all itemsets whose support  $\geq \text{minsup}$
  - Based on number of items  $l$  in an itemset, we group them into  $l$ -itemsets
    - E.g, 2-itemsets for an itemset containing 2 items
- **Step 2: Rule generation**
  - For each frequent itemset  $I$ , find all non-empty subsets  $A \subset I$  such that  $A \rightarrow I - A$  satisfies the **minimum confidence requirement ( $\text{minconf}$ )**

# Step 1: Frequent Itemset Generation: the Itemset Lattice



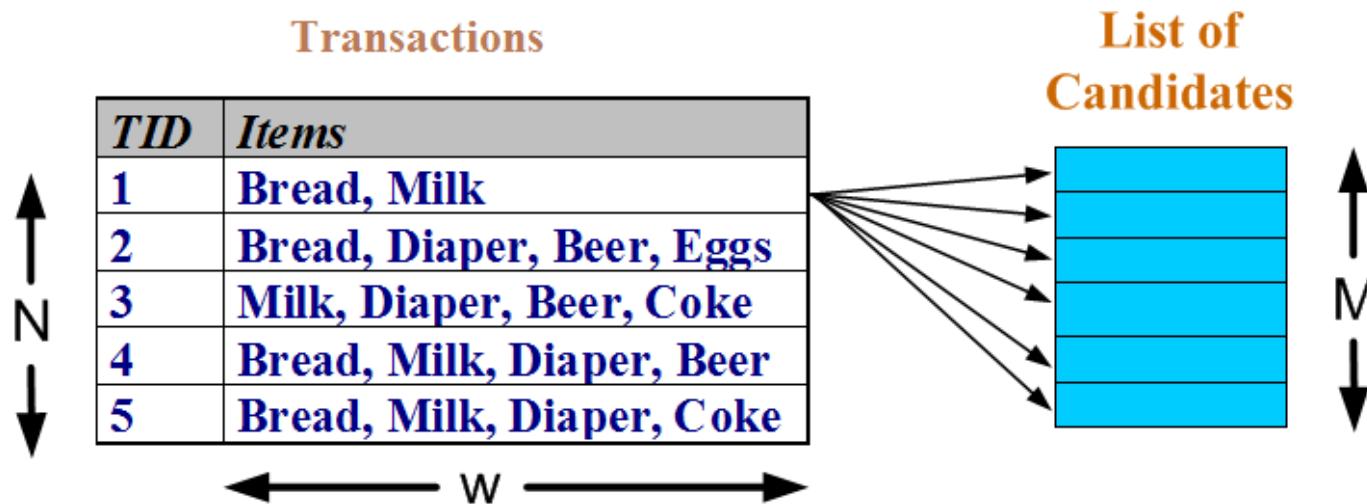
Given  $d$  items, there are  $2^d - 1$  possible candidate itemsets

Frequent itemset generation is computationally expensive!

# Frequent Itemset Generation

## Brute-force approach:

- Each itemset in the lattice is a **candidate** frequent itemset
- Count the support of each candidate by scanning the database
- Output the itemsets with a support  $\geq \text{minsup}$



- Match each transaction against every candidate
- Complexity  $\sim O(NMw)$ , *w* is the maximum transaction width  
=> **Expensive** since  $M = 2^d - 1!!!$

# Apriori Principle: Reducing Number of Candidates

## ■ Apriori Principle:

- If an itemset is frequent, then all of its subsets must also be frequent

## ■ Apriori principle holds due to the following property of the support measure:

$$\forall X, Y: (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- Support of an itemset never exceeds the support of its subsets
- This is known as the **anti-monotone** property of support

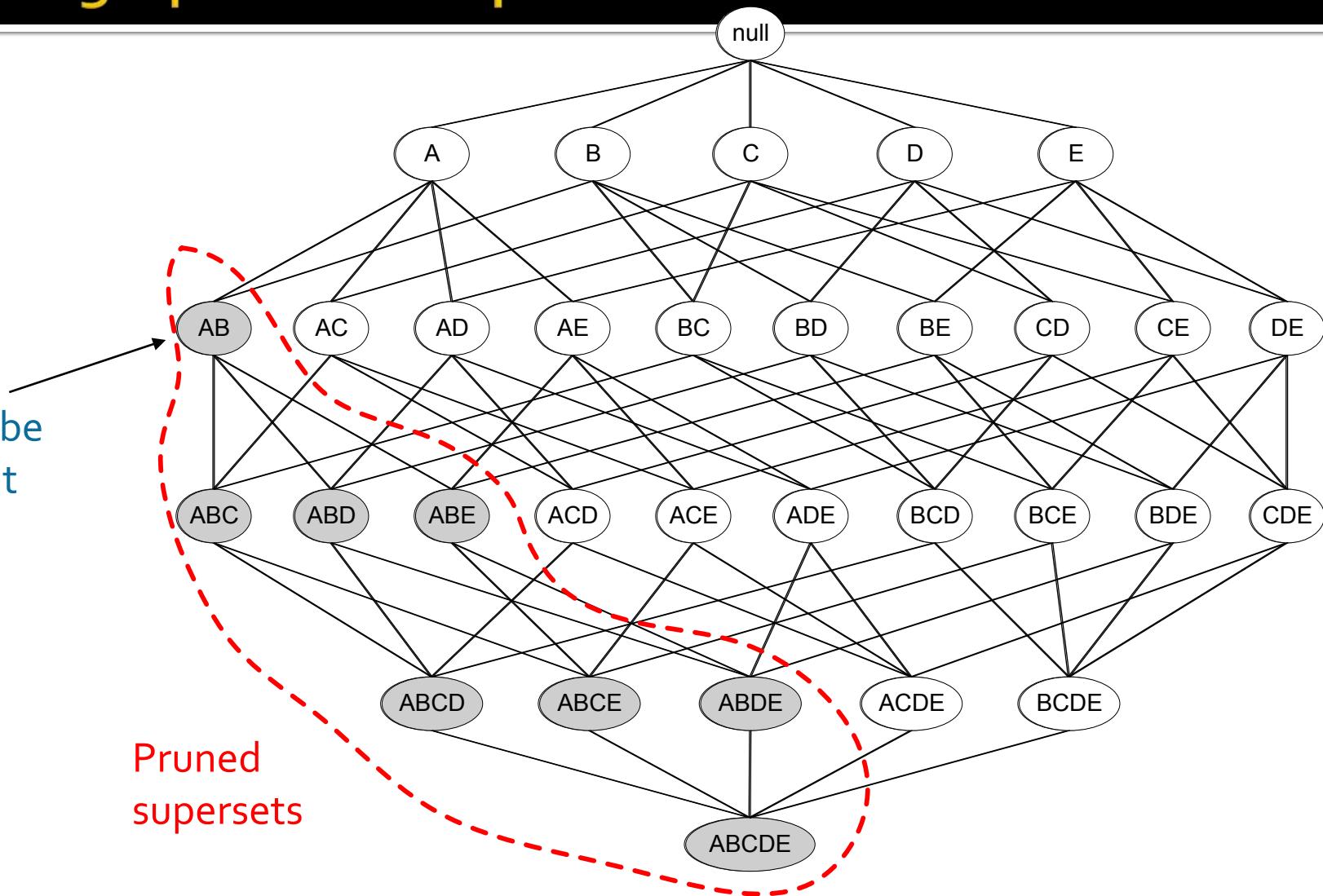
# Anti-Monotone Property

- Any subset of a **frequent** itemset must be also **frequent** — an anti-monotone property
  - Any transaction containing {beer, diaper, milk} also contains {beer, diaper}
  - {beer, diaper, milk} is frequent  $\rightarrow$  {beer, diaper} must also be frequent
- In other words, any **superset** of an **infrequent** itemset must also be **infrequent**
  - No superset of any infrequent itemset should be generated or tested
  - Many item combinations can be pruned!

# Illustrating Apriori Principle

Found to be  
Infrequent

Pruned  
supersets



# Apriori Algorithm

- $F_k$ : frequent k-itemsets (containing k items)
- $C_k$ : candidate k-itemsets
- Algorithm
  - Let  $k=1$
  - Generate  $F_1 = \{\text{frequent 1-itemsets}\}$
  - Repeat until  $F_k$  is empty
    - **Candidate Generation:** Generate  $C_{k+1}$  from  $F_k$
    - **Candidate Pruning:** Prune candidate itemsets in  $C_{k+1}$  containing subsets of length  $k$  that are infrequent
    - **Support Counting:** Count the support of each candidate in  $C_{k+1}$  by scanning the DB
    - **Candidate Elimination:** Eliminate candidates in  $C_{k+1}$  that are infrequent, leaving only those that are frequent  $\Rightarrow F_{k+1}$

# Candidate Generation: Brute-force method

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

Items

Item
Beer
Bread
Cola
Diapers
Eggs
Milk

Candidate Generation

Itemset
{Beer, Bread, Cola}
{Beer, Bread, Diapers}
{Beer, Bread, Eggs}
{Beer, Bread, Milk}
{Beer, Cola, Diapers}
{Beer, Cola, Eggs}
{Beer, Cola, Milk}
{Beer, Diapers, Eggs}
{Beer, Diapers, Milk}
{Beer, Eggs, Milk}
{Bread, Cola, Diapers}
{Bread, Cola, Eggs}
{Bread, Cola, Milk}
{Bread, Diapers, Eggs}
{Bread, Diapers, Milk}
{Bread, Eggs, Milk}
{Cola, Diapers, Eggs}
{Cola, Diapers, Milk}
{Cola, Eggs, Milk}
{Diapers, Eggs, Milk}

Frequent  
2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Candidate  
Pruning

Itemset
{Bread, Diapers, Milk}

Figure 5.6. A brute-force method for generating candidate 3-itemsets.

# Candidate Generation: $F_{k-1} \times F_{k-1}$ Method

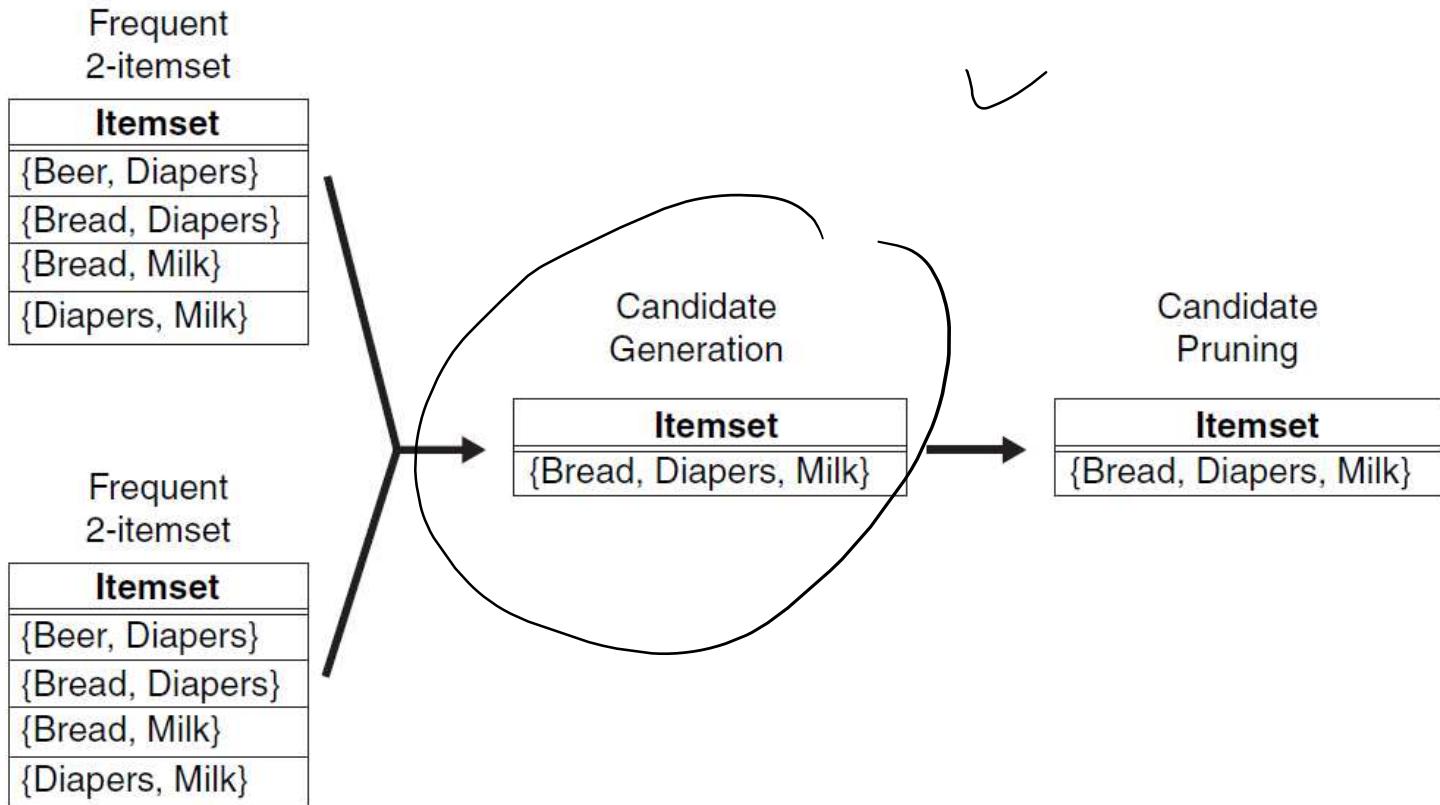
- To generate  $C_{k+1}$  from  $F_k$ : Merge two frequent (k)-itemsets if their first (k-1) items are identical
- $F_3 = \{ABC, ABD, ABE, ACD, BCD, BDE, CDE\}$ 
  - Merge(ABC, ABD) = ABCD
  - Merge(ABC, ABE) = ABCE
  - Merge(ABD, ABE) = ABDE
  - Do not merge(ABD, ACD) because they share only prefix of length 1 instead of length 2

# Candidate Pruning

- Let  $F_3 = \{ABC, ABD, ABE, ACD, BCD, BDE, CDE\}$  be the set of frequent 3-itemsets
- $C_4 = \{ABCD, ABCE, ABDE\}$  is the set of candidate 4-itemsets generated (from previous slide)
- Candidate pruning
  - Prune ABCE because ACE and BCE are infrequent
  - Prune ABDE because ADE is infrequent
- After candidate pruning:  $C_4 = \{ABCD\}$

# Candidate Generation: $F_{k-1} \times F_{k-1}$ Method

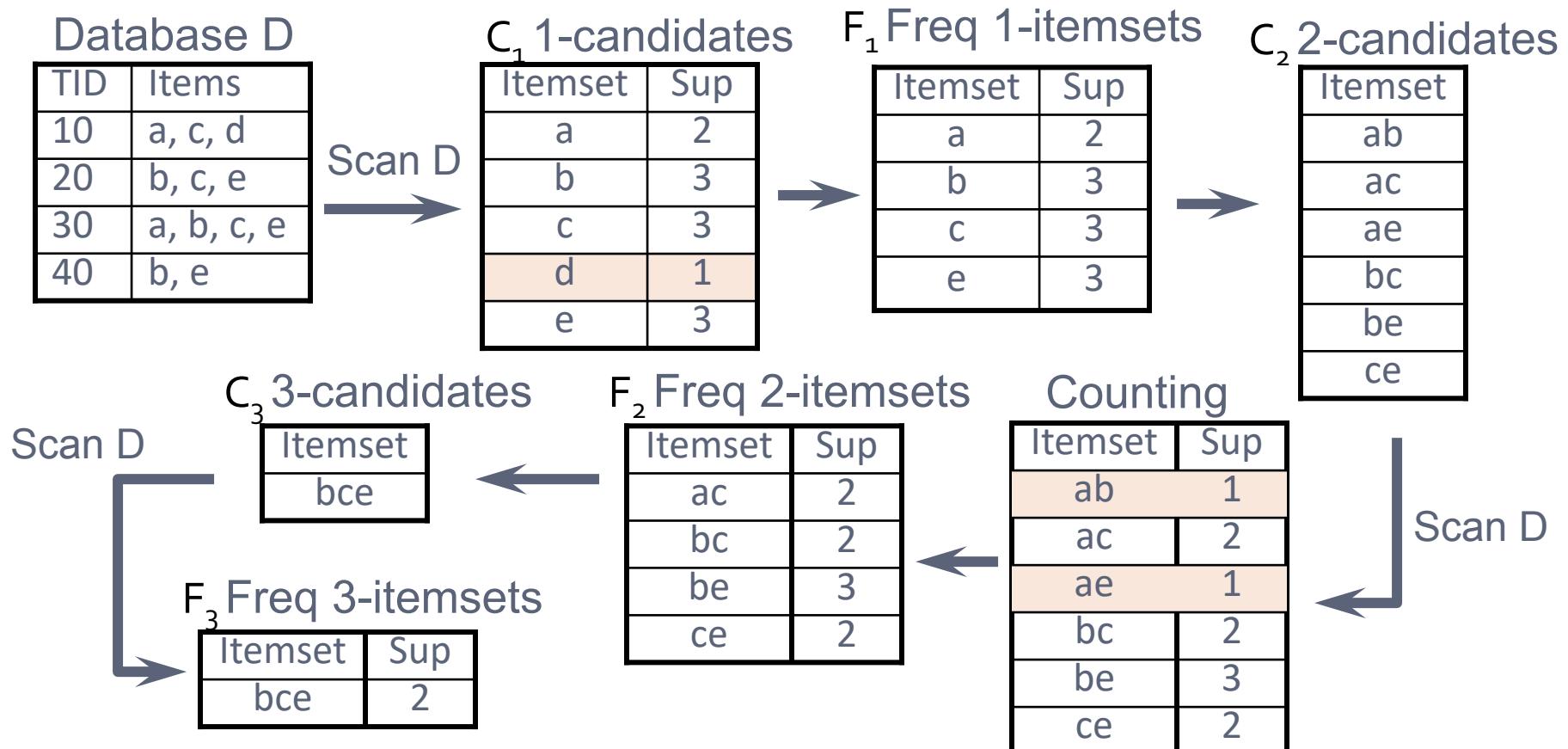
## Another example:



**Figure 5.8.** Generating and pruning candidate  $k$ -itemsets by merging pairs of frequent  $(k-1)$ -itemsets.

# Example: Apriori-based Mining

Minsup=2



# Step2: Rule generation

- **Step 1:** Find all frequent itemsets  $I$ 
  - Generate all itemsets whose support  $\geq \text{minsup}$
- **Step 2: Rule generation**
  - Given a frequent itemset  $I$ , find all non-empty subsets  $A \subset I$  such that  $A \rightarrow I - A$  satisfies the **minimum confidence requirement minconf**
    - If  $\{A, B, C, D\}$  is a frequent itemset, candidate rules:

$ABC \rightarrow D$ ,	$ABD \rightarrow C$ ,	$ACD \rightarrow B$ ,	$BCD \rightarrow A$ ,
$A \rightarrow BCD$ ,	$B \rightarrow ACD$ ,	$C \rightarrow ABD$ ,	$D \rightarrow ABC$
$AB \rightarrow CD$ ,	$AC \rightarrow BD$ ,	$AD \rightarrow BC$ ,	$BC \rightarrow AD$ ,
$BD \rightarrow AC$ ,	$CD \rightarrow AB$		
    - An example to compute the rule confidence
      - $\text{confidence}(A, B \rightarrow C, D) = \text{support}(A, B, C, D) / \text{support}(A, B)$
    - Do the above for every frequent itemset, and **output all the rules above the confidence threshold**

# Example

$$B_1 = \{m, c, b\}$$

$$B_3 = \{m, c, b, n\}$$

$$B_5 = \{m, p, b\}$$

$$B_7 = \{c, b, j\}$$

$$B_2 = \{m, p, j\}$$

$$B_4 = \{c, j\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_8 = \{b, c\}$$

Support threshold  
 $\text{minsup} = 3$ ,  
confidence  
 $\text{minconf} = 0.75$

## 1) Frequent itemsets:

- $\{b, m\}$ :  $s = 4$     $\{b, c\}$ : 4    $\{c, m\}$ : 3    $\{c, j\}$ : 3    $\{m, c, b\}$ : 3

## 2) Generate rules:

$\{b, m\}$ :  $b \rightarrow m$ :  $c=4/6$ ;    $m \rightarrow b$ :  $c=4/5$

$\{b, c\}$ :  $b \rightarrow c$ :  $c=5/6$ ;    $c \rightarrow b$ :  $c=5/6$

...

$\{m, c, b\}$ :  ~~$b, c \rightarrow m$ :  $c=3/5$~~ ;    $b, m \rightarrow c$ :  $c=3/4$ ;    $m, c \rightarrow b$ :  $c=3/3$

~~$b \rightarrow c, m$ :  $c=3/6$~~ ;    ~~$c \rightarrow b, m$ :  $c=3/6$~~ ;    ~~$m \rightarrow b, c$ :  $c=3/5$~~

# Rule Generation

- If  $|I| = k$ , then there are  $2^k - 2$  candidate association rules (ignoring  $I \rightarrow \emptyset$  and  $\emptyset \rightarrow I$ )

- For example  $I$  is  $\{A, B, C, D\}$ , 14 candidate rules:

$ABC \rightarrow D$ ,	$ABD \rightarrow C$ ,	$ACD \rightarrow B$ ,	$BCD \rightarrow A$ ,
$A \rightarrow BCD$ ,	$B \rightarrow ACD$ ,	$C \rightarrow ABD$ ,	$D \rightarrow ABC$
$AB \rightarrow CD$ ,	$AC \rightarrow BD$ ,	$AD \rightarrow BC$ ,	$BC \rightarrow AD$ ,
$BD \rightarrow AC$ ,	$CD \rightarrow AB$		

# Another example of Rule Generation

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

## Observations:

- All the above rules are binary partitions of the same itemset:  
 $\{\text{Milk, Diaper, Beer}\}$
- Rules originating from the same itemset have **identical support** but can have **different confidence**

## Example of Rules:

$\{\text{Milk,Diaper}\} \rightarrow \{\text{Beer}\}$  ( $s=2, c=2/3$ )  
 $\{\text{Milk,Beer}\} \rightarrow \{\text{Diaper}\}$  ( $s=2, c=2/2$ )  
 $\{\text{Diaper,Beer}\} \rightarrow \{\text{Milk}\}$  ( $s=2, c=2/3$ )  
 $\{\text{Beer}\} \rightarrow \{\text{Milk,Diaper}\}$  ( $s=2, c=2/3$ )  
 $\{\text{Diaper}\} \rightarrow \{\text{Milk,Beer}\}$  ( $s=2, c=2/4$ )  
 $\{\text{Milk}\} \rightarrow \{\text{Diaper,Beer}\}$  ( $s=2, c=2/4$ )

# Prunning in Rule Generation

- Confidence does not have an anti-monotone property  
 $c(ABC \rightarrow D)$  can be larger or smaller than  $c(AB \rightarrow D)$
- How to prune?
  - Confidence of rules generated from the same itemset has an anti-monotone property
  - E.g., Suppose  $\{A,B,C,D\}$  is a frequent 4-itemset:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

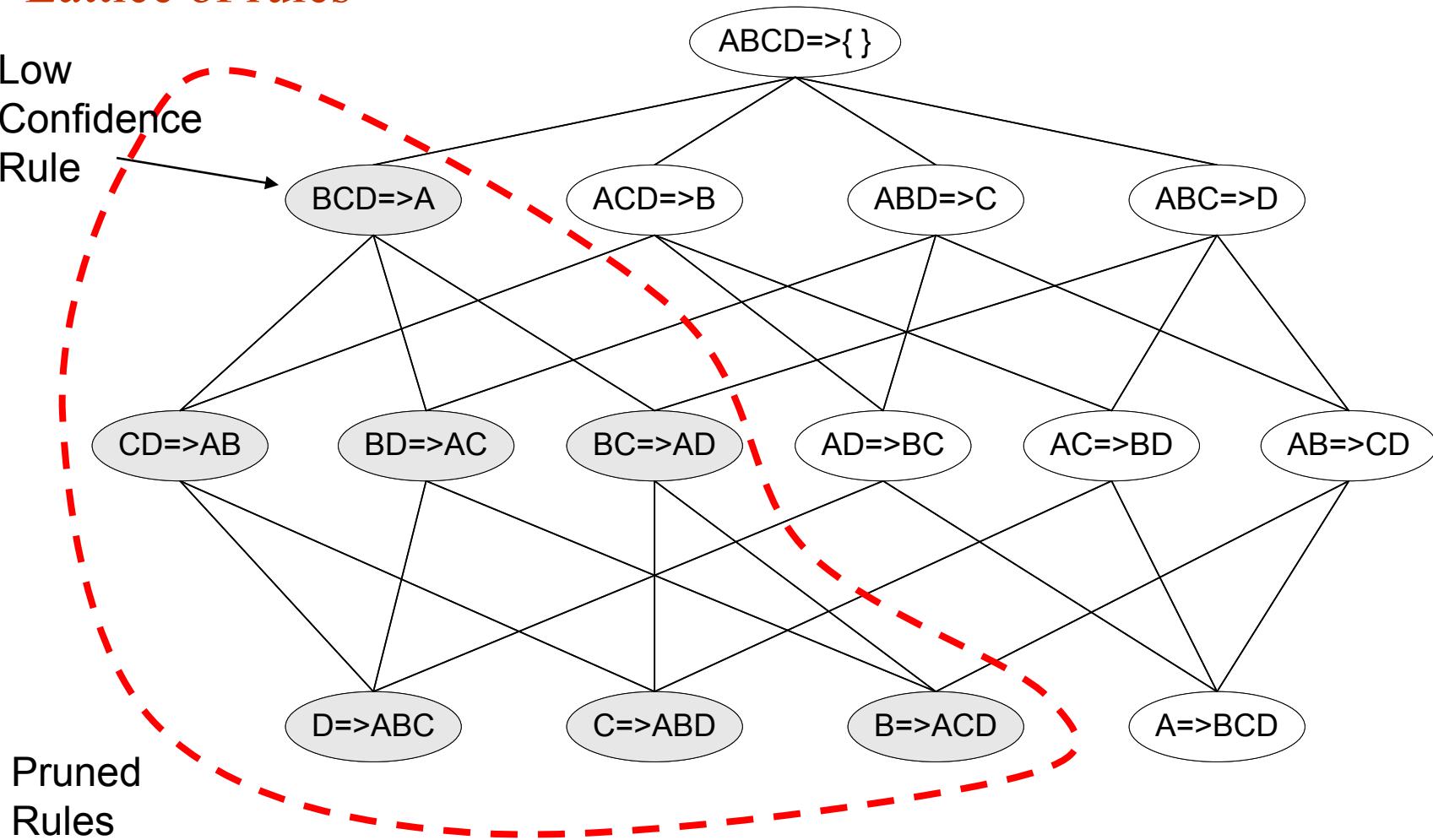
Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

- **Observation:** If  $A, B, C \rightarrow D$  is below confidence, so is  $A, B \rightarrow C, D$
- Can generate “bigger” rules from smaller ones (RHS)!

# Rule Generation for Apriori Algorithm

Lattice of rules

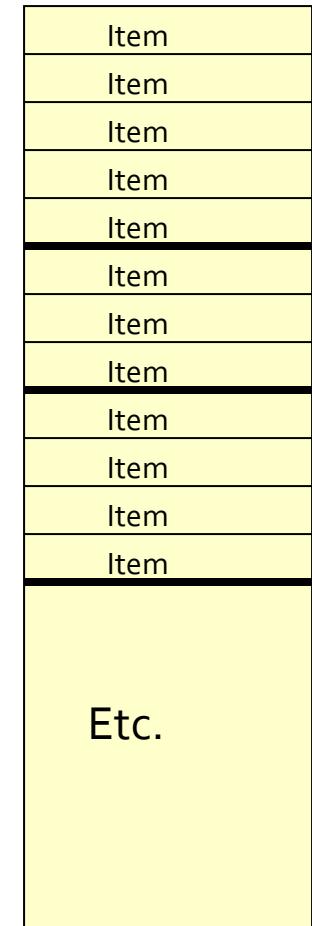
Low  
Confidence  
Rule



# **Implementation of algorithms for finding Frequent Itemsets on BIG data**

# Itemsets: Computation Model

- Back to finding frequent itemsets
- Typically, **data** is kept in flat files rather than in a database system:
  - Stored on disk
  - Stored basket-by-basket
  - Baskets are **small** but we have many baskets and many items



Items are positive integers,  
and boundaries between  
baskets are –1.

# Computation Model

- The dominating cost of mining disk-resident data is usually the **number of disk I/Os**
- In practice, association-rule algorithms read the data in ***passes*** – all baskets read in turn
- We measure the cost by the **number of *passes*** an algorithm makes over the data

# Main-Memory Bottleneck

- **Itemsets:** To find frequent itemsets, we have to count them. To count them, we have to generate them, and usually store them in **memory**.
- For many frequent-itemset algorithms, **main-memory is the critical resource**
  - As we read baskets, we need to count something, e.g., occurrences of pairs of items
  - The number of different things we can count is limited by main memory
  - Swapping counts in/out is a disaster (**why?**)

# Naïve Algorithm for finding frequent pairs

- Let us consider finding frequent pairs
- Read file once, counting in main memory the occurrences of each pair:
  - From each basket of  $n$  items, generate its  $n(n-1)/2$  pairs by two nested loops
- Fails if (#items) $^2$  exceeds main memory
  - Remember: #items can be 100K (Wal-Mart) or 10B (Web pages)
    - Suppose  $10^5$  items, counts are 4-byte integers
    - Number of pairs of items:  $10^5(10^5-1)/2 = 5*10^9$
    - Therefore,  $2*10^{10}$  (20 gigabytes) of memory needed

# Counting Pairs in Memory

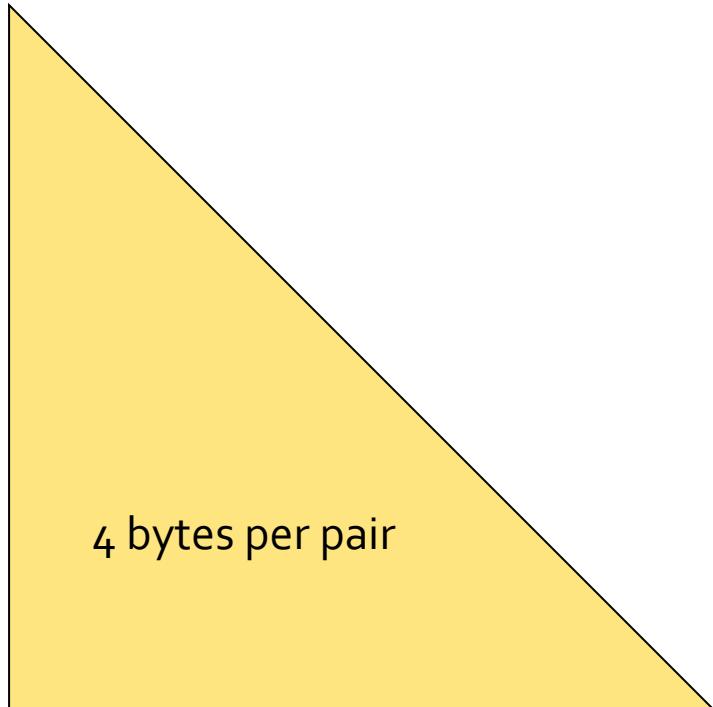
Two approaches:

- **Approach 1:** Count all pairs using a matrix
- **Approach 2:** Keep a table of triples  $[i, j, c] =$   
“the count of the pair of items  $\{i, j\}$  is  $c$ .”
  - If integers and item ids are 4 bytes, we need approximately 12 bytes for pairs with count  $> 0$
  - Plus some additional overhead for the hashtable

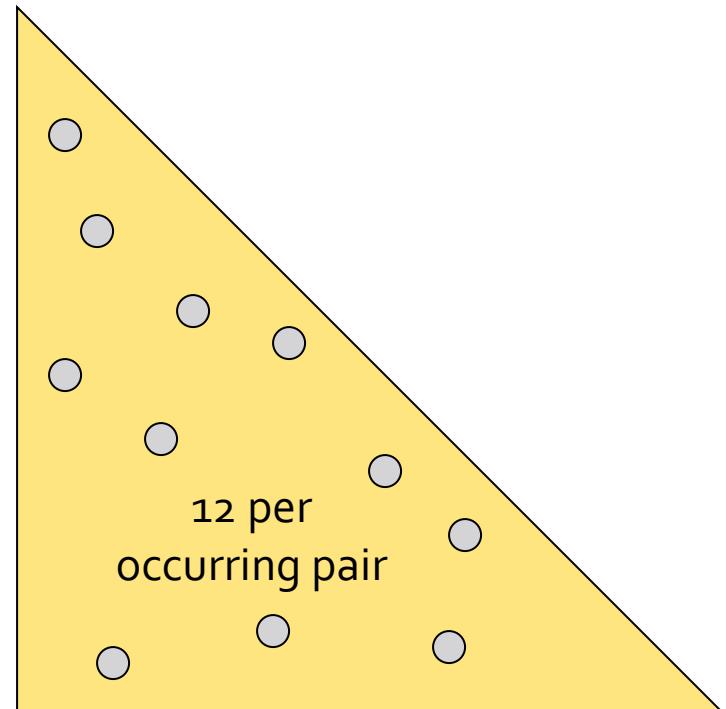
Note:

- **Approach 1** only requires 4 bytes per pair
- **Approach 2** uses 12 bytes per pair  
(but only for pairs with count  $> 0$ )

# Comparing the 2 Approaches



Triangular Matrix



Triples

# Comparing the two approaches

- **Approach 1: Triangular Matrix**
  - $n$  = total number items
  - Count pair of items  $\{i, j\}$  only if  $i < j$
  - Keep pair counts in lexicographic order:
    - $\{1,2\}, \{1,3\}, \dots, \{1,n\}, \{2,3\}, \{2,4\}, \dots, \{2,n\}, \{3,4\}, \dots$
  - Pair  $\{i, j\}$  is at position  $(i-1)(n-i)/2 + j - i$
  - Total number of pairs  $n(n-1)/2$ ; total bytes=  $2n^2$
  - **Triangular Matrix** requires 4 bytes per pair
- **Approach 2** uses **12 bytes** per occurring pair  
*(but only for pairs with count > 0)*
  - Beats Approach 1 if less than **1/3** of possible pairs actually occur

# Comparing the two approaches

## ■ Approach 1: Triangular Matrix

- $n$  = total number items
- Count pair of items  $\{i, j\}$  only if  $i < j$
- Keep pair counts in lexicographic order:
  - $\{1,2\}, \{1,3\}, \dots, \{1,n\}, \{2,3\}, \{2,4\}, \dots, \{2,n\}, \{3,4\}, \dots$
- Pair  $\{i, j\}$  is at position  $(i-1)(n-i)/2 + j - i$
- Total number of pairs  $n(n-1)/2$ ; total bytes =  $2n^2$

- **Triangular Matrix** requires 4 bytes per pair

## ■ Approach 2 uses 12 bytes per pair *(but only for pairs with count > 0)*

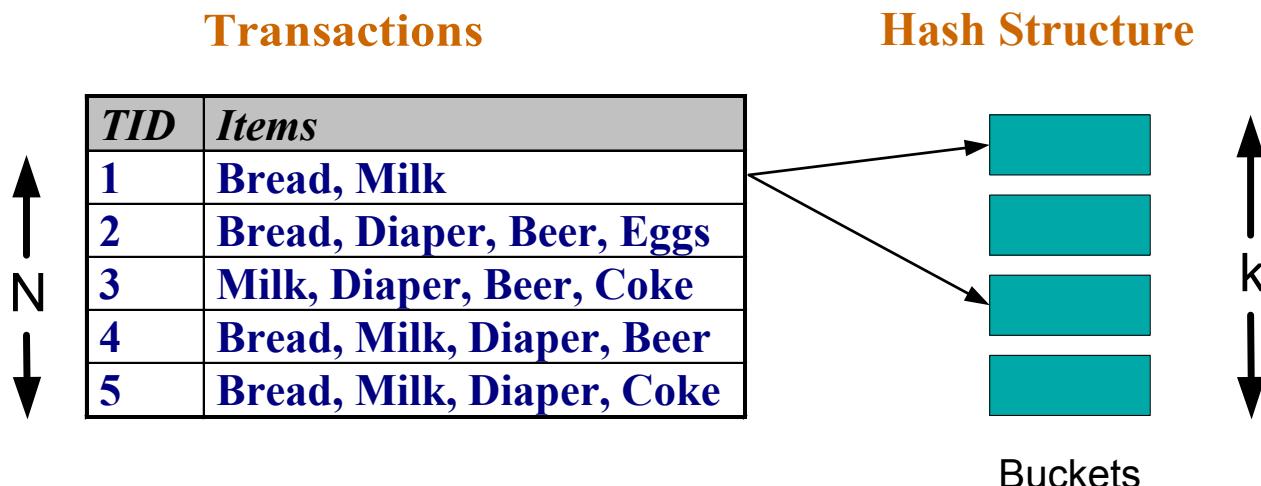
- Beats Approach 1 if less than 1/3 of possible pairs actually occur

Problem is if we have too many items so the pairs do not fit into memory.

Can we do better?

# Support Counting of Candidate Itemsets

- When itemsets are longer, matching every candidate itemset against every transaction is an expensive operation
- To reduce number of comparisons, store the candidate itemsets in a hash structure
  - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets



# Support Counting Using a Hash Tree (the next 5 slides for your information only, not examinable)

Suppose you have 15 candidate itemsets of length 3:

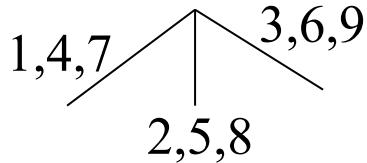
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

Use a data structure to organize these itemsets to reduce computation:

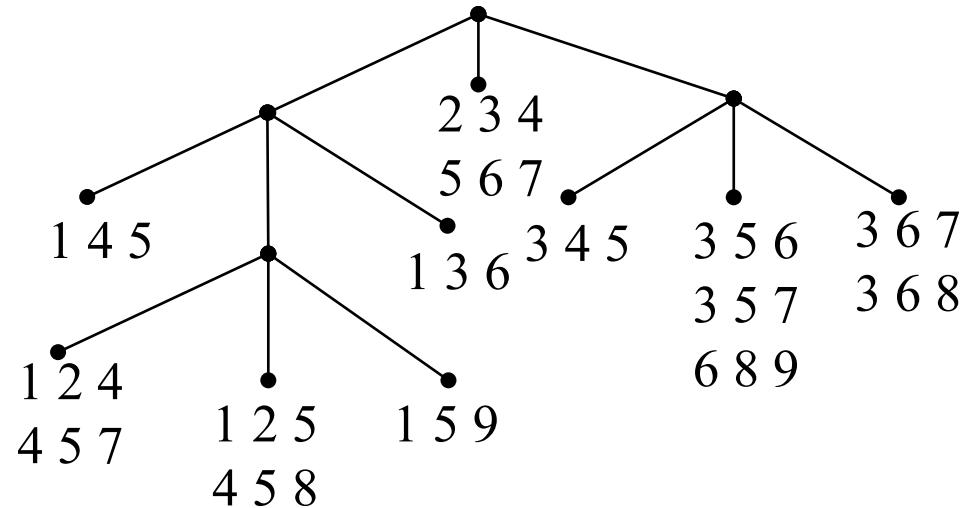
- Hash function
- Max leaf size: max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, **split the node**)

$$h(p) = p \bmod 3$$

Hash function



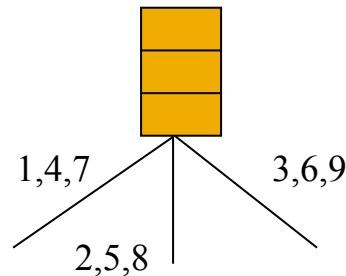
How?



- Order items in each itemset
- Apply the hash function to each item in each itemset in order

# Building a Hash Tree of Itemsets

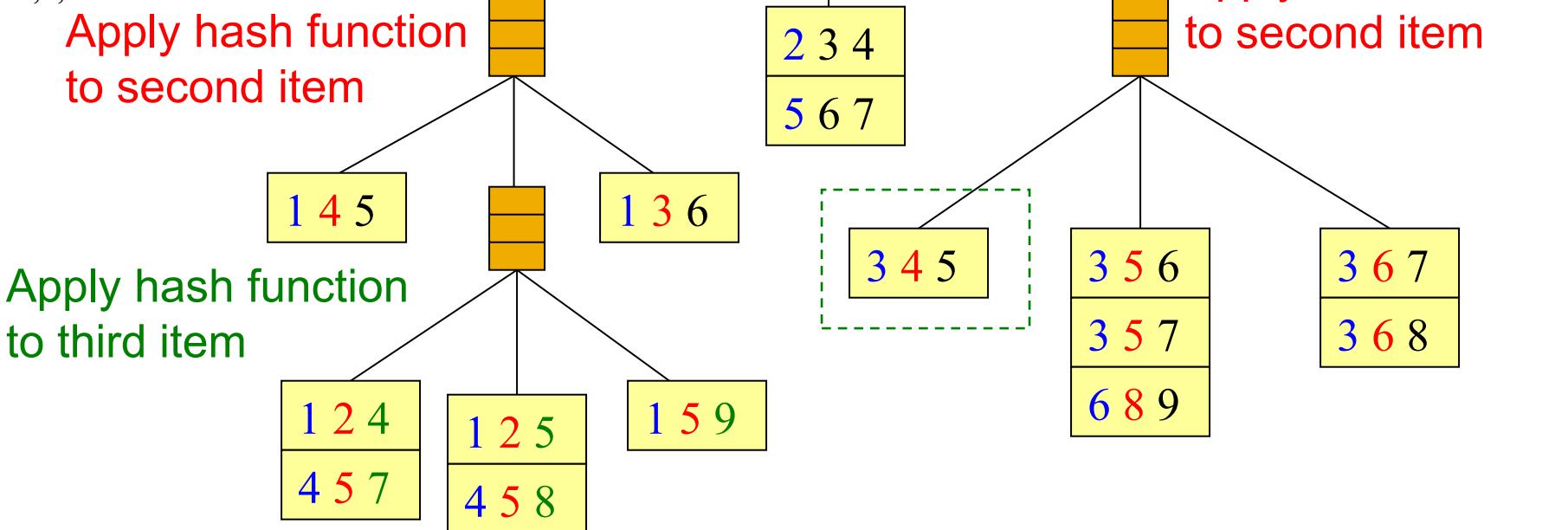
$h(p) = p \bmod 3$   
Hash Function



Apply hash function  
to second item

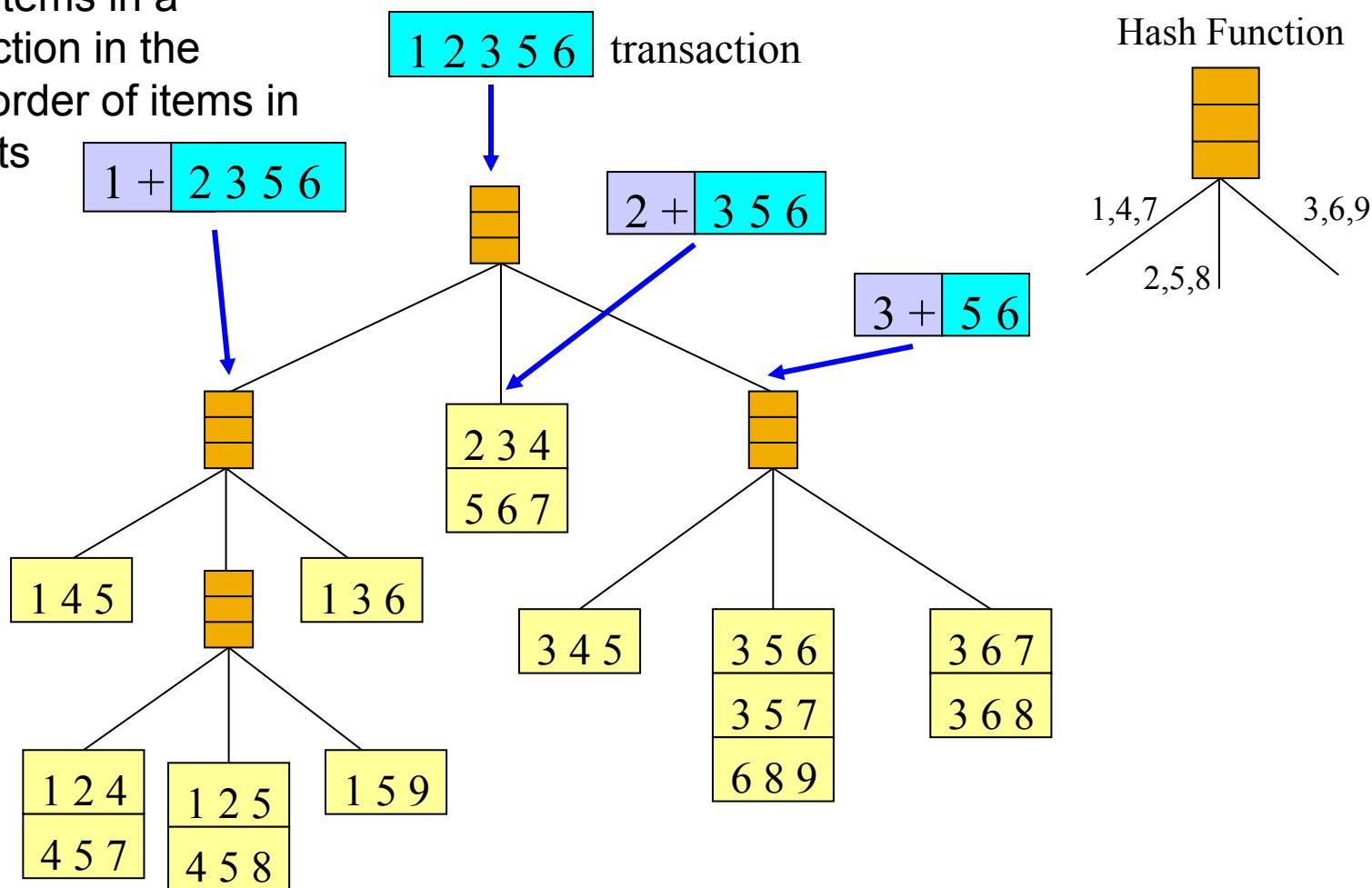
3 itemsets  
in each leaf

Candidate Hash Tree

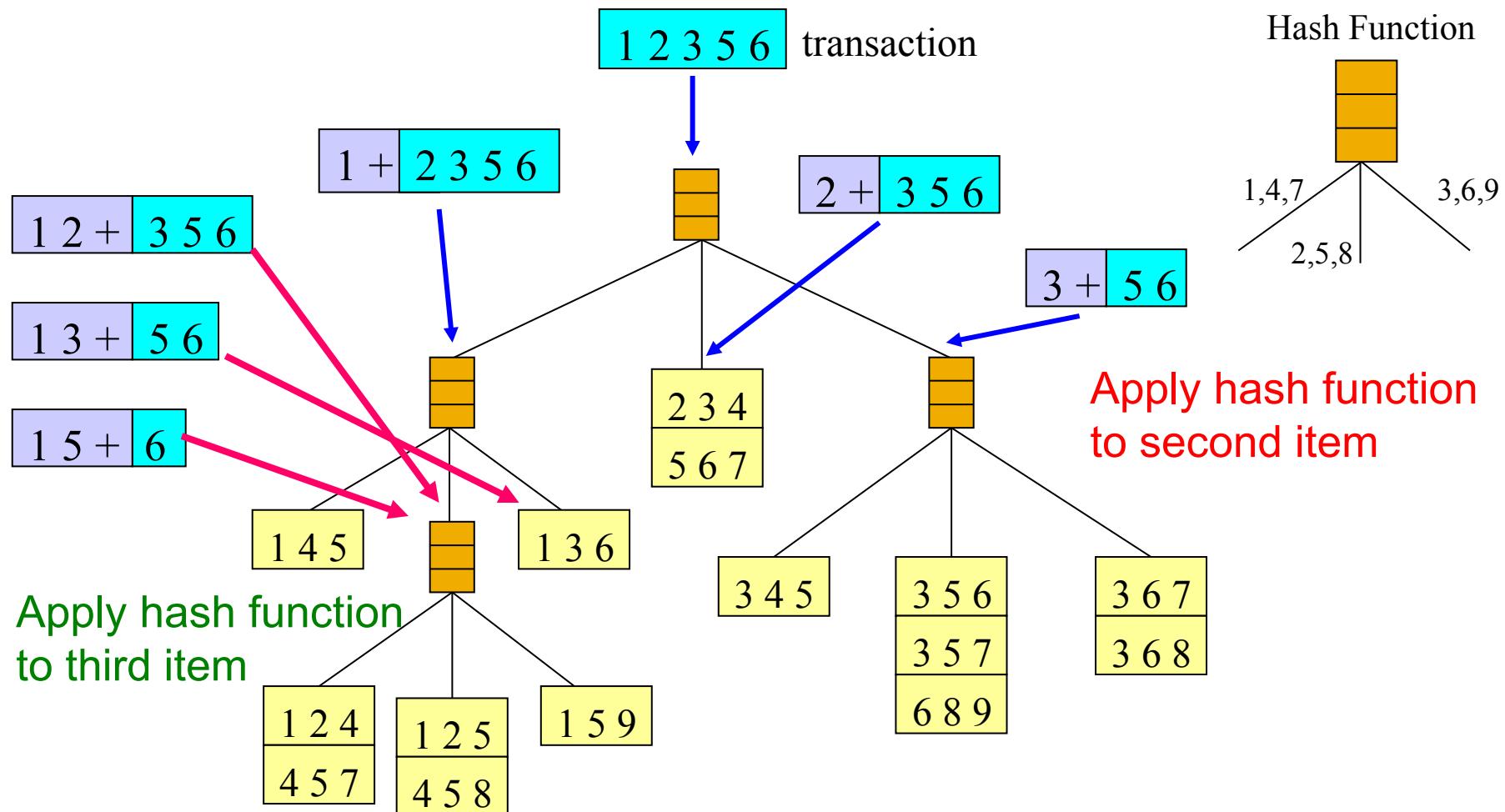


# Support Counting Using a Hash Tree

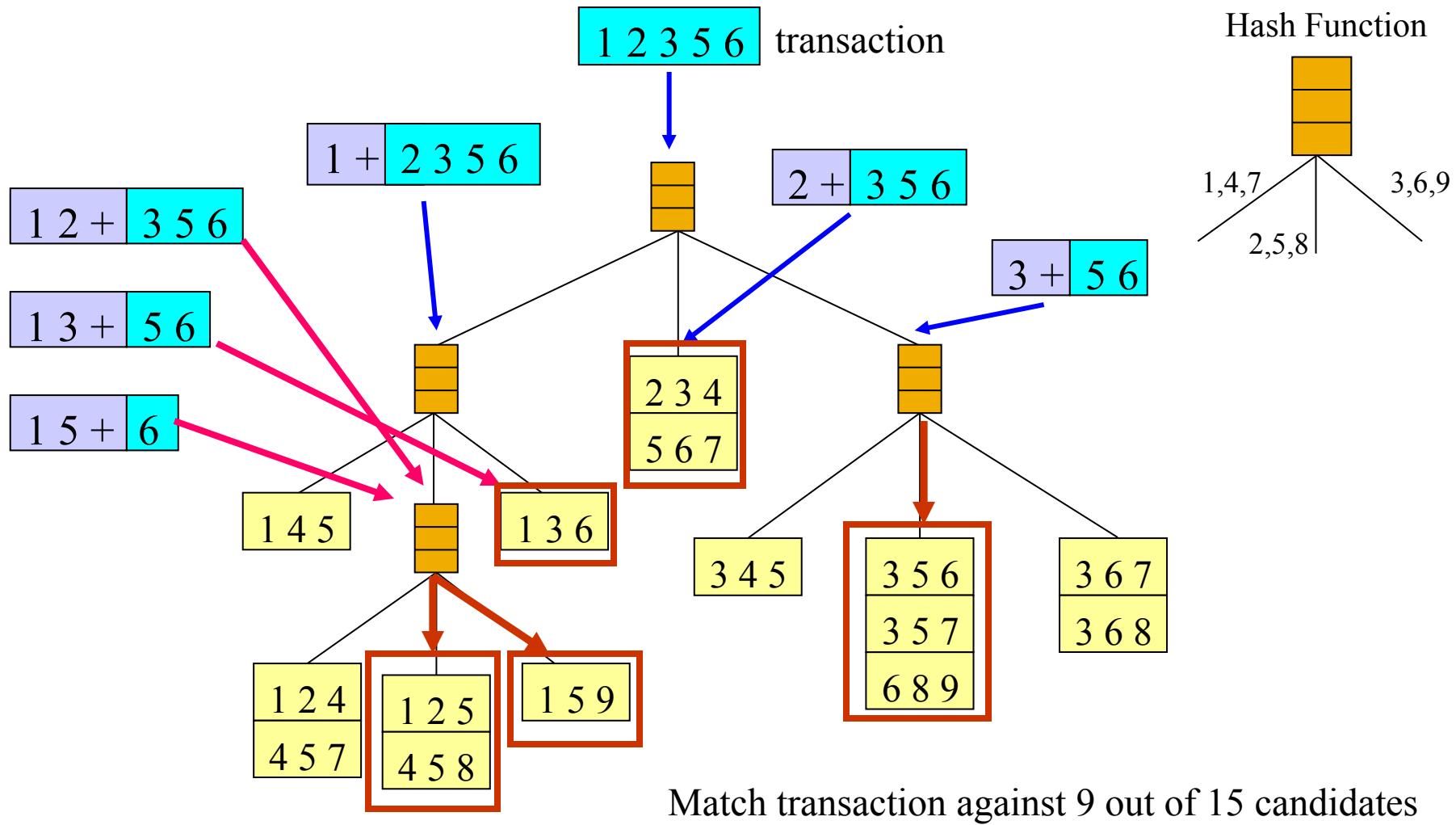
Order items in a transaction in the same order of items in itemsets



# Support Counting Using a Hash Tree



# Support Counting Using a Hash Tree

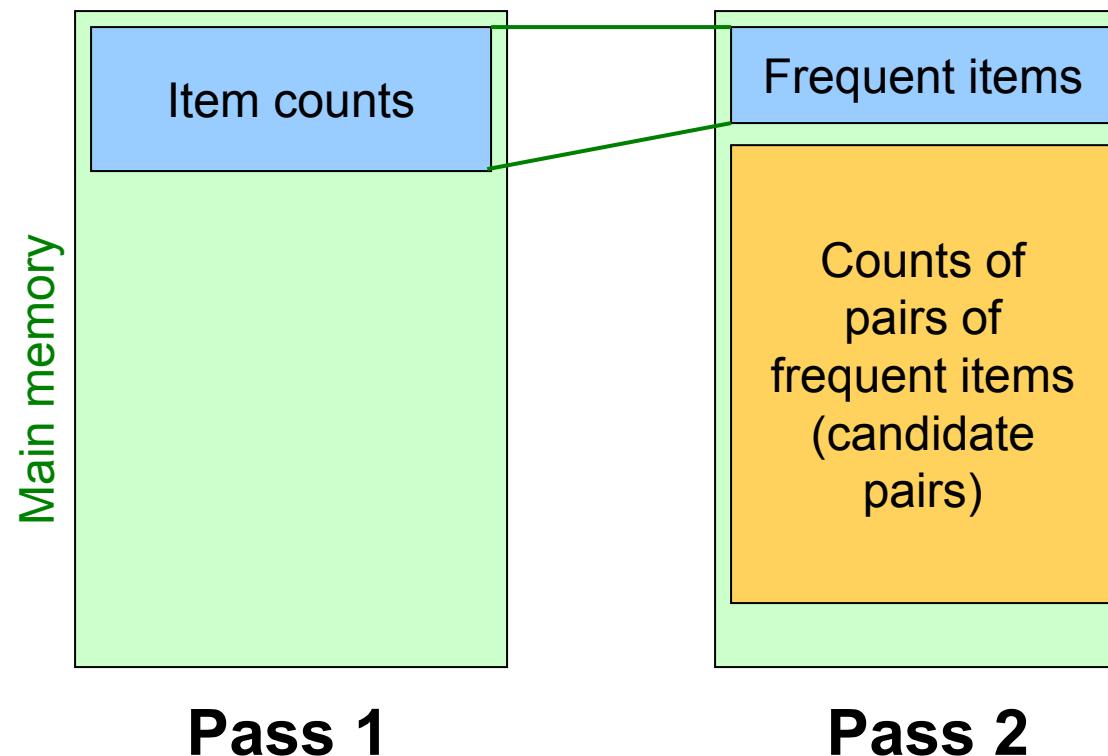


# **Implementation issues of A-Priori Algorithm**

# Implementation of A-Priori Algorithm

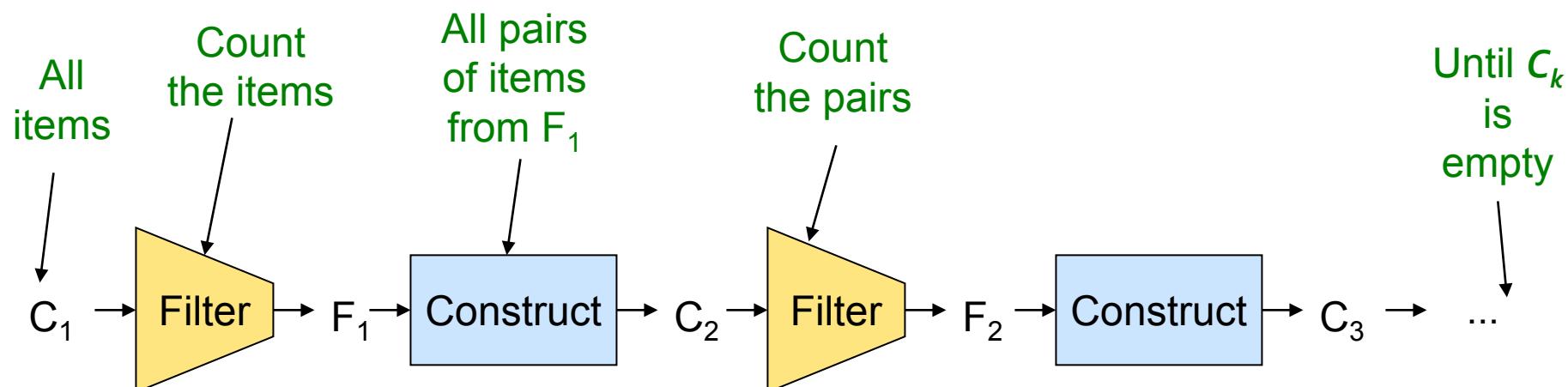
- **Pass 1:** Read baskets and count in main memory the occurrences of each **individual item**
  - Requires only memory proportional to #items
- **Items that appear  $\geq s$  times are the frequent items**
- **Pass 2:** Read baskets again and count in main memory only those pairs where both elements are frequent (from Pass 1)
  - Requires memory proportional to square of **frequent** items only (for counts)
  - Plus a list of the frequent items (so you know what must be counted)

# Main-Memory: Picture of A-Priori



# Frequent Triples, Etc.

- Other passes: For each  $k$ , we construct two sets of  **$k$ -itemsets** (sets of size  $k$ ):
  - $C_k$  = **candidate  $k$ -itemsets**
  - $F_k$  = the set of truly frequent  $k$ -itemsets
  - For each  $k$ , need room in main memory to count each candidate, and need to make a pass of the data to count the occurrences for itemsets in  $C_k$



# PCY (Park-Chen-Yu) Algorithm

# PCY (Park-Chen-Yu) Algorithm

## ■ Observation:

In pass 1 of A-Priori, most memory is idle

- We store only individual item counts
- **Can we use the idle memory to reduce memory required in pass 2?**

## ■ Pass 1 of PCY:

In addition to item counts, maintain a hash table with as many buckets as fit in memory

- Keep a **count** for each bucket into which **pairs** of items are hashed
  - **For each bucket just keep the count, not the actual pairs that hash to the bucket!**

# PCY Algorithm – First Pass

New  
in  
PCY {

```
FOR (each basket) :  
    FOR (each item in the basket) :  
        add 1 to item's count;  
    FOR (each pair of items) :  
        hash the pair to a bucket;  
        add 1 to the count for that bucket;
```

## ■ Few things to note:

- Pairs of items need to be generated from the input file; they are not present in the file
- We are not just interested in the presence of a pair, but we need to see whether it is present at least  $s$  (support) times

# Observations about Buckets

- **Observation:** If a bucket contains a frequent pair, then the bucket is surely frequent
- However, even without any frequent pair, a bucket can still be frequent ☹
  - So, we cannot use the hash to eliminate any member (pair) of a “frequent” bucket
- **But, for a bucket with total count less than  $s$ , none of its pairs can be frequent ☺**
  - Pairs that hash to this bucket can be eliminated as candidates (even if the pair consists of 2 frequent items)
- **Pass 2:**  
Only count pairs that hash to frequent buckets

# Example: PCY Algorithm First Pass

- **Items** = {milk, coke, beer, pepsi, juice}
- **Baskets:**

$$B_1 = \{m, c, b\}$$

$$B_3 = \{m, b\}$$

$$B_5 = \{m, p, b\}$$

$$B_7 = \{c, b, j\}$$

$$B_2 = \{m, p, j\}$$

$$B_4 = \{c, j\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_8 = \{b, c\}$$

- By assigning  $milk = 1, coke = 2, beer = 3, pepsi = 4, juice = 5$  then **Baskets:**

$$B_1 = \{1, 2, 3\}$$

$$B_3 = \{1, 3\}$$

$$B_5 = \{1, 3, 4\}$$

$$B_7 = \{2, 3, 5\}$$

$$B_2 = \{1, 4, 5\}$$

$$B_4 = \{2, 5\}$$

$$B_6 = \{1, 2, 3, 5\}$$

$$B_8 = \{2, 3\}$$

# Example: PCY Algorithm First Pass

- Define a hash function (Hashing pair  $(i, j)$  to bucket  $K$ ):

$$h(i, j) = (i + j)\%5 = K$$

Example:

$$h(1,3) = (1 + 3)\%5 = 4 \rightarrow \text{Bucket } 4$$

# Example: PCY Algorithm First Pass

## ■ First Pass:

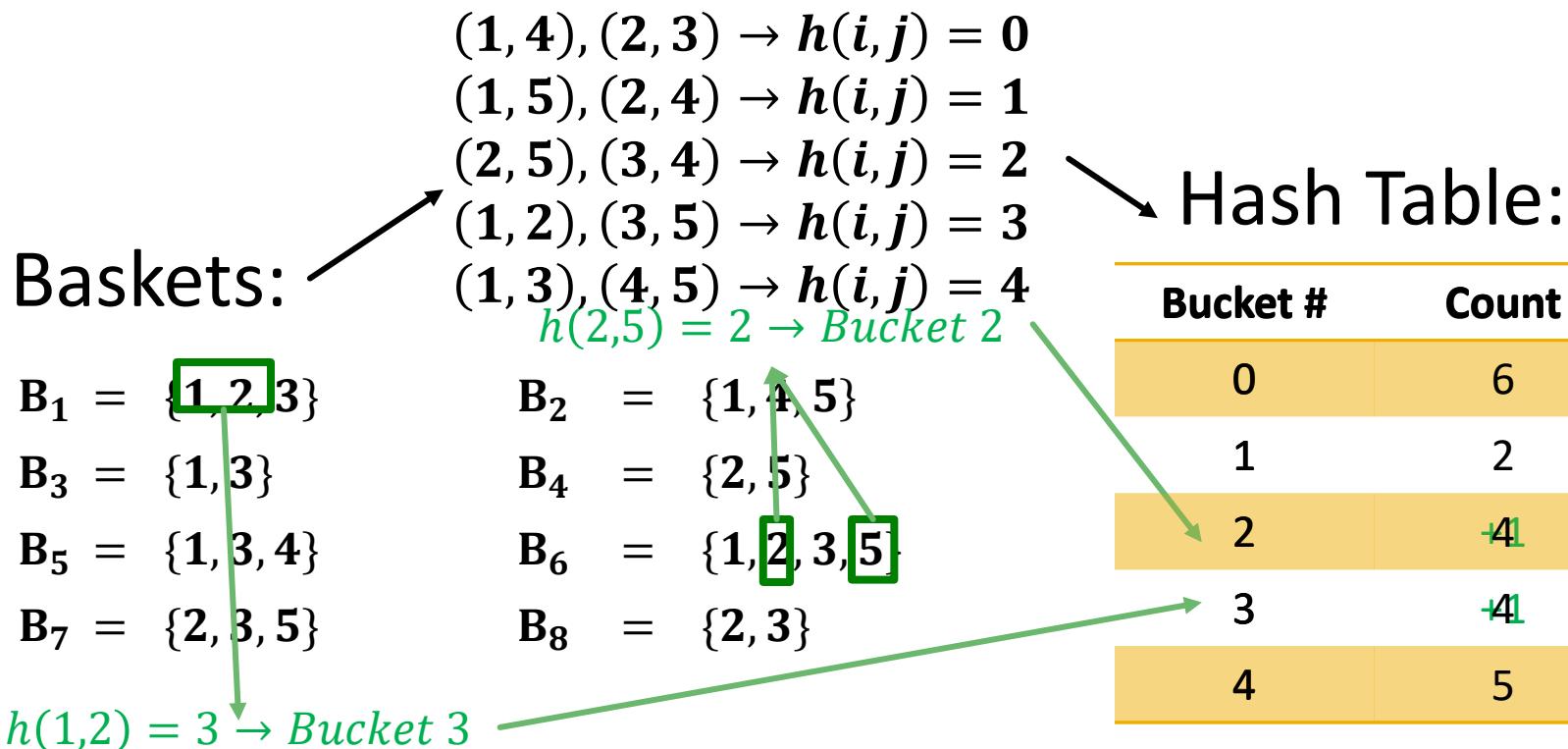
```
FOR (each basket) :  
    FOR (each item in the basket) :  
        add 1 to item's count;  
  
    FOR (each pair of items) :  
        hash the pair to a bucket;  
        add 1 to the count for that bucket;
```

## Baskets:

		Item #	Count
$B_1 = \{1, 2, 3\}$	$B_2 = \{1, 4, 5\}$	1	+5
$B_3 = \{1, 3\}$	$B_4 = \{2, 5\}$	2	5
$B_5 = \{1, 3, 4\}$	$B_6 = \{1, 2, 3, 5\}$	3	5
$B_7 = \{2, 3, 5\}$	$B_8 = \{2, 3\}$	4	2
		5	4

# Example: PCY Algorithm First Pass

■ First Pass: FOR (each basket) :  
    FOR (each item in the basket) :  
        add 1 to item's count;  
    FOR (each pair of items) :  
        hash the pair to a bucket;  
        add 1 to the count for that bucket;



# Example: PCY Algorithm First Pass

- For support threshold  $s = 3$

Item Count

Item #	Count
1	5
2	5
3	5
4	2
5	4

Itemsets in each bucket

$(1, 4), (2, 3) \rightarrow 0$   
 $(1, 5), (2, 4) \rightarrow 1$   
 $(2, 5), (3, 4) \rightarrow 2$   
 $(1, 2), (3, 5) \rightarrow 3$   
 $(1, 3), (4, 5) \rightarrow 4$

Hash Table

Bucket #	Count
0	6
1	2
2	4
3	4
4	5

- For  $s = 3$ ,  $L_1 = \{1, 2, 3, 5\}$ , and Bitmap  $\{1, 0, 1, 1, 1\}$

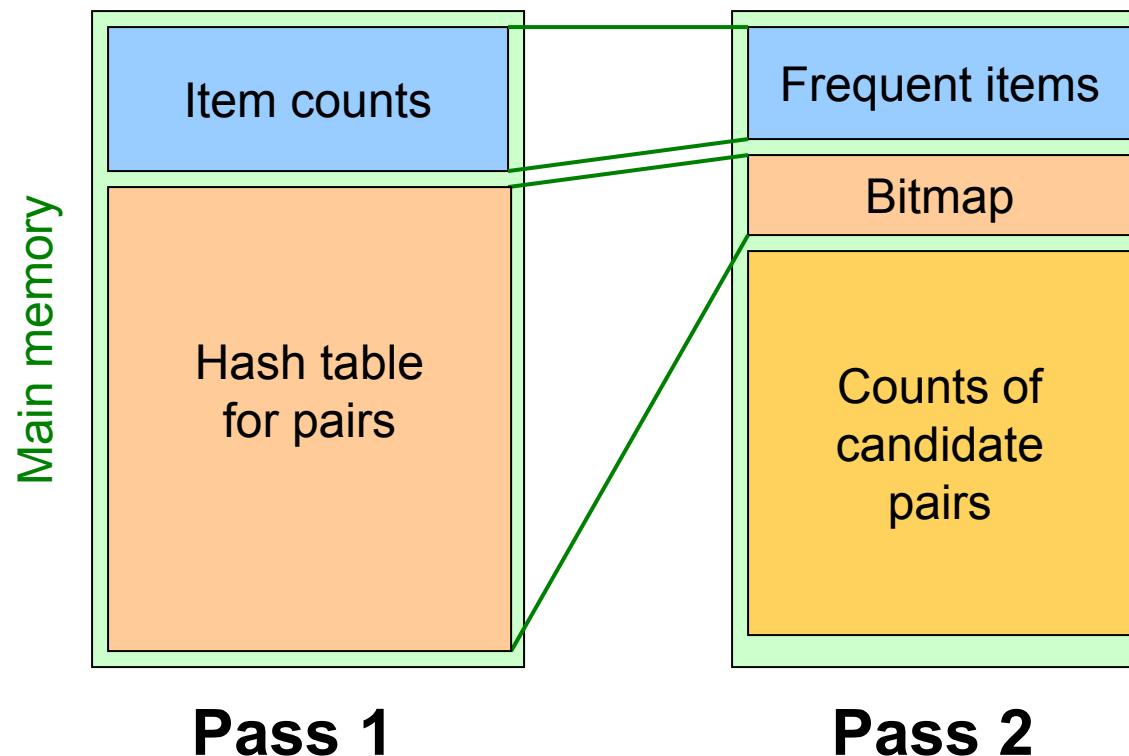
# PCY Algorithm – Between Passes

- Optimization: Replace the buckets by a bit-vector:
  - 1 means the bucket count exceeded the support  $s$  (call it a **frequent bucket**); 0 means it did not
  - 4-byte integer counts are replaced by bits, so the bit-vector requires **1/32 of memory**
- Also, decide which items are frequent and list them for the second pass

# PCY Algorithm – Pass 2

- Count all pairs  $\{i, j\}$  that meet the conditions for being a **candidate pair**:
  1. Both  $i$  and  $j$  are frequent items
  2. The pair  $\{i, j\}$  hashes to a bucket whose bit in the bit vector is **1** (i.e., a **frequent bucket**)
- **Both conditions are necessary for the pair to have a chance of being frequent**
- Implementation details: On second pass, a table of **(item, item, count)** triples is essential (we cannot use triangular matrix approach, **why?**)
  - Thus, hash table must eliminate approx. 2/3 of the candidate pairs for PCY to beat A-Priori

# Main-Memory: Picture of PCY



# Example: PCY Algorithm 2nd Pass

For  $\{i, j\}$  to be a **candidate pair**:

1. Both  $i$  and  $j$  are frequent items
2. The pair  $\{i, j\}$  hashes to a bucket whose bit in the bit vector is **1** (i.e., a **frequent bucket**)

Item Count

Item #	Count
1	5
2	5
3	5
4	2
5	4

Hash Table

Bucket #	Count
0	6
1	2
2	4
3	4
4	5

$(\cancel{1}, \cancel{4}), (\cancel{2}, 3) \rightarrow h(i, j) = 0$   
 $(1, \cancel{5}), (\cancel{2}, \cancel{4}) \rightarrow h(i, j) = 1$   
 $(\cancel{2}, 5), (\cancel{3}, \cancel{4}) \rightarrow h(i, j) = 2$   
 $(1, 2), (\cancel{3}, 5) \rightarrow h(i, j) = 3$   
 $(1, 3), (\cancel{4}, 5) \rightarrow h(i, j) = 4$

Frequent Items:  $\{1, 2, 3, 5\}$ , frequent buckets:  $0, 2, 3, 4$

# Example: PCY Algorithm 2nd Pass

For  $\{i, j\}$  to be a **candidate pair**:

1. Both  $i$  and  $j$  are frequent items
2. The pair  $\{i, j\}$  hashes to a bucket whose bit in the bit vector is **1** (i.e., a **frequent bucket**)

Item Count

Item #	Count
1	5
2	5
3	5
4	2
5	4

Hash Table

Bucket #	Count
0	6
1	2
2	4
3	4
4	5

$$\begin{aligned} & \cancel{(1, 4)}, \cancel{(2, 3)} \rightarrow h(i, j) = 0 \\ & \cancel{(1, 5)}, \cancel{(2, 4)} \rightarrow h(i, j) = 1 \\ & (2, 5), \cancel{(3, 4)} \rightarrow h(i, j) = 2 \\ & (1, 2), (3, 5) \rightarrow h(i, j) = 3 \\ & (1, 3), \cancel{(4, 5)} \rightarrow h(i, j) = 4 \end{aligned}$$

Frequent Items:  $\{1, 2, 3, 5\}$ , frequent buckets: 0,2,3,4

# Example: PCY Algorithm 2nd Pass

For  $\{i, j\}$  to be a **candidate pair**:

1. Both  $i$  and  $j$  are frequent items
2. The pair  $\{i, j\}$  hashes to a bucket whose bit in the bit vector is **1** (i.e., a **frequent bucket**)

Candidate Pairs & Counts

~~(1,4), (2, 3)  $\rightarrow h(i, j) = 0$~~   
~~(1, 5), (2, 4)  $\rightarrow h(i, j) = 1$~~   
~~(2, 5), (3, 4)  $\rightarrow h(i, j) = 2$~~   
~~(1, 2), (3, 5)  $\rightarrow h(i, j) = 3$~~   
~~(1, 3), (4, 5)  $\rightarrow h(i, j) = 4$~~



Pair	Count
(2,3)	4
(2,5)	3
(1,2)	2
(3,5)	2
(1,3)	4

→ Frequent Itemsets are: {1}, {2}, {3}, {5}, {1, 3}, {2, 3}, {2, 5}

**Interestingness of association  
rules, other types of patterns**

# Interesting Association Rules

- Not all high-confidence rules are interesting

- The rule  $X \rightarrow \text{milk}$   $\text{conf}(X \rightarrow \text{milk}) = \frac{\text{support}(X \cup \text{milk})}{\text{support}(X)}$
- The rule  $X \rightarrow \text{milk}$  may have high confidence for many itemsets  $X$ , because milk is just purchased very often (independent of  $X$ ) and the confidence will be high

- Interest of an association rule  $I \rightarrow j$ : difference between its confidence and the fraction of baskets that contain  $j$

$$\text{Interest}(I \rightarrow j) = \text{conf}(I \rightarrow j) - \Pr[j]$$

- Interesting rules are those with high positive or negative interest values (ABS usually above 0.5)

# Example: Interest

$$B_1 = \{m, c, b\}$$

$$B_3 = \{m, \textcolor{red}{c}, b\}$$

$$B_5 = \{m, p, \textcolor{red}{c}, b\}$$

$$B_7 = \{c, b, j\}$$

$$B_2 = \{m, p, j\}$$

$$B_4 = \{c, j\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_8 = \{b, c\}$$

$$\text{Interest}(I \rightarrow j) = \text{conf}(I \rightarrow j) - \Pr[j]$$

- Association rule:  $\{m, b\} \rightarrow c$

- Confidence =  $4/4 = 1$
- Interest =  $|4/4 - 7/8| = 1/8$ 
  - Item **c** appears in  $7/8$  of the baskets
  - Rule is not very interesting!

- Another rule:  $\{j\} \rightarrow c$

- Interest =  $|3/4 - 7/8| = 1/8$

# More about Judging if a Rule/Pattern Is Interesting?

- Interestingness measures: Objective vs. subjective
  - Objective interestingness measures
    - Support, confidence, interest, correlation(lift), ...
  - Subjective interestingness measures:
    - Let a user specify
      - Query-based: Relevant to a user's particular request
    - Judge against one's knowledge-base
      - unexpected, freshness, timeliness

# Limitation of the Support-Confidence

- Another Example: Suppose one school may have the following statistics on # of students who may play basketball and/or eat cereal:

	play-basketball	not play-basketball	sum (row)
eat-cereal	400	350	750
not eat-cereal	200	50	250
sum(col.)	600	400	1000

2-way contingency table

- Association rule mining may generate the following:
  - $\text{play-basketball} \Rightarrow \text{eat-cereal}$  [support: 40%, conf: 66.7%]
- But the overall % of students eating cereal is **75% > 66.7%** ( eat-cereal appears in most of data). a more telling rule:
  - $\neg \text{play-basketball} \Rightarrow \text{eat-cereal}$  [support: 35%, conf: **87.5%**]

# Interestingness Measure: Lift

- Measure of dependent/correlated events: **lift**

$$lift(B, C) = \frac{conf(B \rightarrow C)}{support(C)} = \frac{support(B \cup C)}{support(B) \times support(C)}$$

Note that here **support(.)** is defined as relative support

- Lift( $B, C$ ) may tell how  $B$  and  $C$  are correlated
  - Lift( $B, C$ ) = 1:  $B$  and  $C$  are independent
  - > 1: positively correlated
  - < 1: negatively correlated
- For our example,

	B	$\neg B$	$\Sigma_{\text{row}}$
C	400	350	750
$\neg C$	200	50	250
$\Sigma_{\text{col.}}$	600	400	1000

$$lift(B, C) = \frac{400/1000}{600/1000 \times 750/1000} = 0.89$$

$$lift(B, \neg C) = \frac{200/1000}{600/1000 \times 250/1000} = 1.33$$

- $B$  and  $C$  are negatively correlated since  $lift(B, C) < 1$ ;
- $B$  and  $\neg C$  are positively correlated since  $lift(B, \neg C) > 1$

# Compact Representation of Frequent Itemsets

- Some frequent itemsets are redundant because their supersets are also frequent

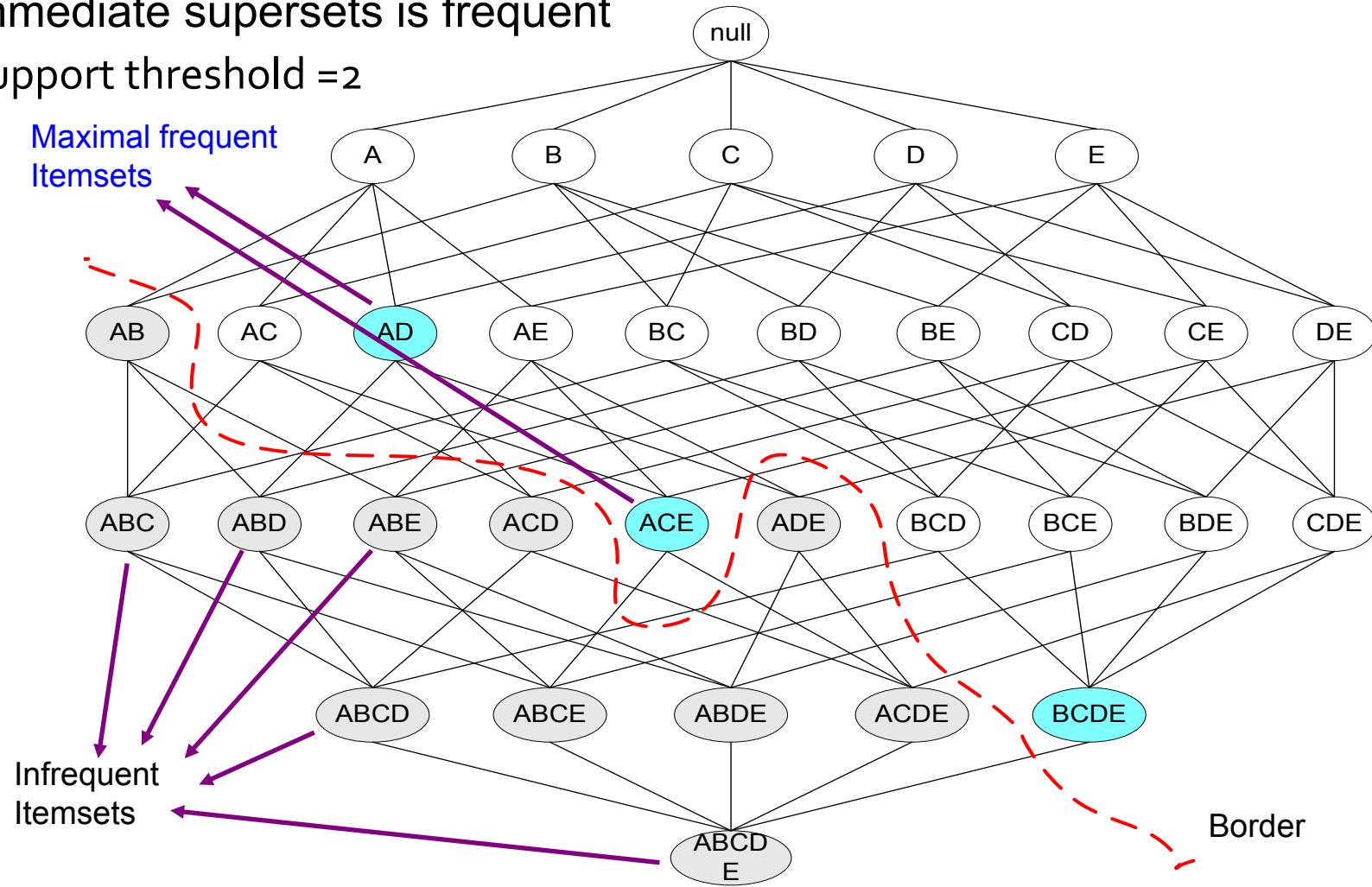
Consider the following data set. Assume support threshold =5

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0		
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0		
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0		
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0		
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0		
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		

- Number of frequent itemsets =  $3 \times \sum_{k=1}^{10} \binom{10}{k}$
- Need a compact representation

# Maximal Frequent Itemset

An itemset is **maximal frequent** if it is frequent and none of its immediate supersets is frequent  
support threshold = 2



# What are the Maximal Frequent Itemsets in this Data?

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	

Minimum support threshold = 5

(A<sub>1</sub>-A<sub>10</sub>)

(B<sub>1</sub>-B<sub>10</sub>)

(C<sub>1</sub>-C<sub>10</sub>)

# Closed Itemset

- An itemset X is closed if none of its immediate supersets has the same support as the itemset X.
- X is NOT closed if at least one of its immediate supersets has support count as X, i.e., contained by the same set of baskets as X .

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

- ❖ {C}:4 is closed: none of its immediate supersets (AC:3, BC:3, DC:2; EC:2)has the same support.
  - ❖ {A}:3 is NOT closed: its immediate superset AC has the same support.
  - ❖ {AB}:2 is NOT closed
  - ❖ {ABC}:2 is closed
- Closed Frequent itemset: Closed & frequent

# What are the Closed Itemsets in this Data?

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	

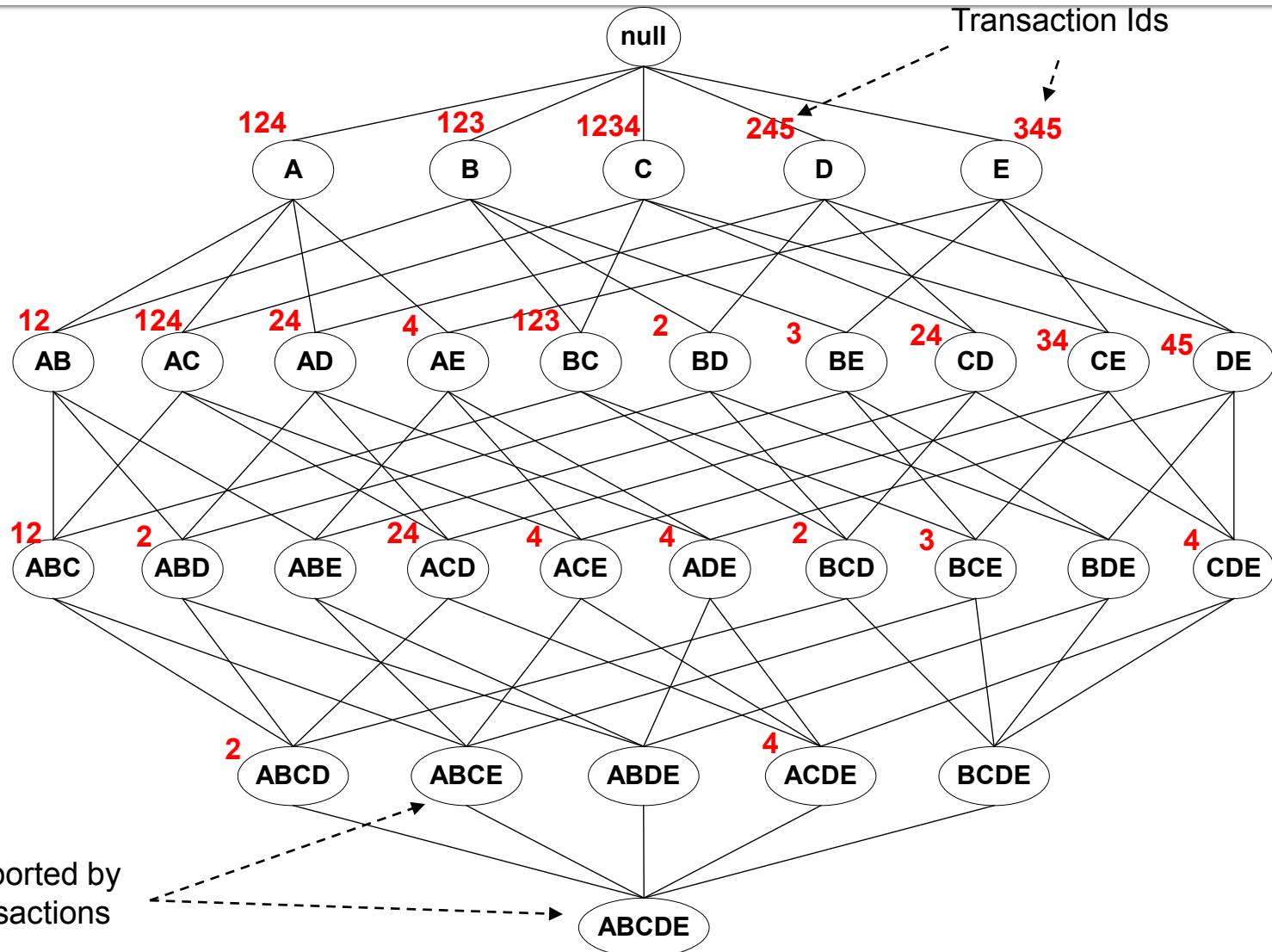
(A<sub>1</sub>-A<sub>10</sub>)

(B<sub>1</sub>-B<sub>10</sub>)

(C<sub>1</sub>-C<sub>10</sub>)

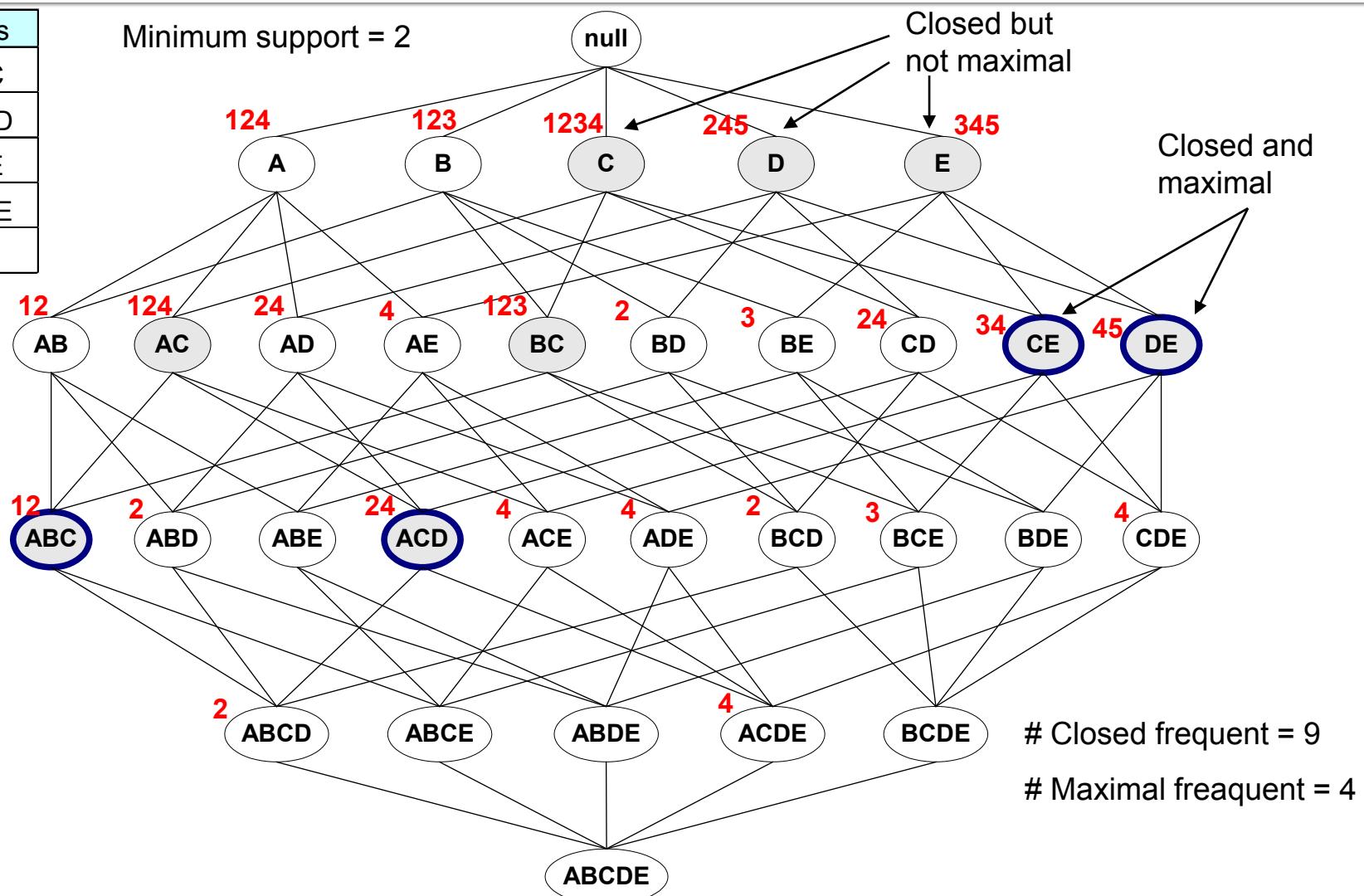
# Lattice representation of itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

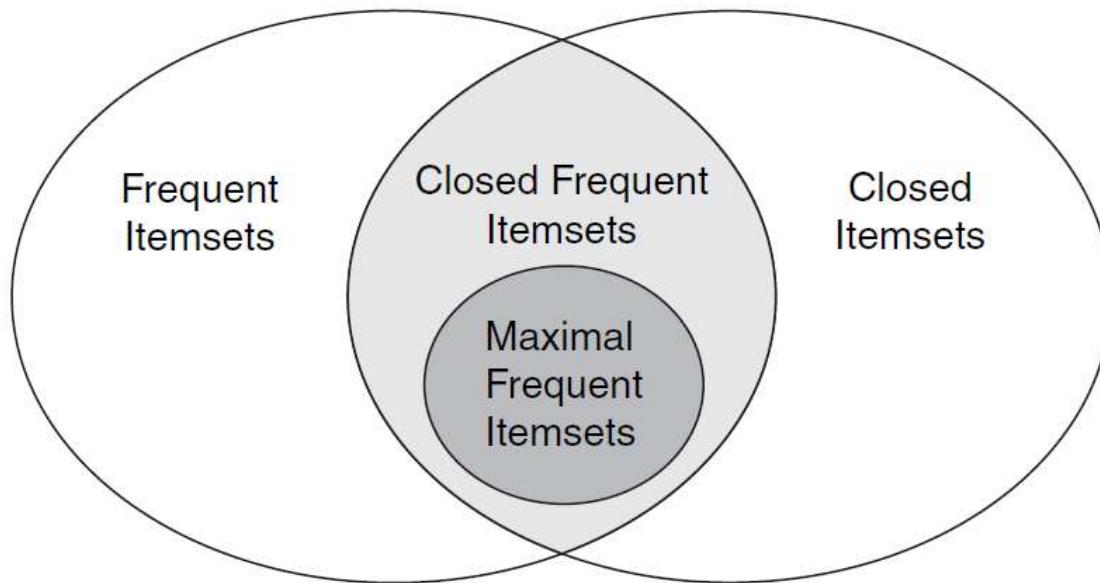


# Maximal Frequent vs Closed Frequent Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



# Maximal vs Closed Itemsets



**Figure 5.18.** Relationships among frequent, closed, closed frequent, and maximal frequent itemsets.

# Summary

- **Frequent itemsets and Association rules**
  - Understand the concepts
- **Algorithms for finding frequent itemsets**
  - Understand the A-Priori algorithm
- **Efficient Implementation of Algorithms**
  - Understand how to implement Apriori and PCY algorithms
- **Interestingness of association rules**
  - Understand the basic concept
- **Closed itemset and maximal itemset.**
  - Understand the concepts and their differences

# Frequent Itemset Mining & Association Rules (Part 2)

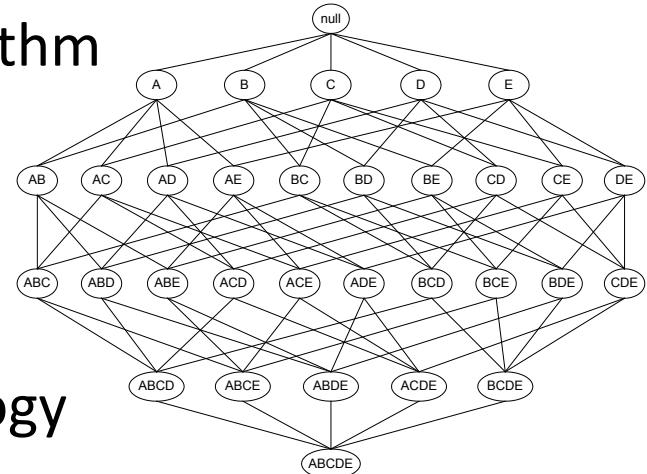
Some slides adapted from UIUC data mining course, and the data mining book by Kumar etc.

# Outline

- A different frequent itemset/pattern mining algorithm
- Sequential pattern mining

# Why Mining Frequent Patterns by Pattern Growth?

- Apriori: A *breadth-first search* mining algorithm
  - First find the complete set of frequent k-itemsets
  - Then derive frequent (k+1)-itemset candidates
  - Scan DB again to find true frequent (k+1)-itemsets
- Motivation for a different mining methodology
  - Can we develop a *depth-first search* mining algorithm?
  - For a frequent itemset  $\rho$ , can subsequent search be confined to only those transactions that contain  $\rho$ ?
- Such thinking leads to a frequent pattern growth approach:
  - FP-Growth (Han et al “Mining Frequent Patterns without Candidate Generation,” SIGMOD 2000)



# Example: Frequent Pattern Growth Algorithm

- Mining FP w/o candidate generation.
- Steps:
  - {
    - 1. Find the frequency of 1-itemset.
    - 2. Construct Ordered-Item set.
    - 3. Construct FP-tree (i.e., inserting ordered-item set).
    - 4. Recursively mine FP-tree and grow frequent patterns obtained so far
      - a. Construct Conditional database.
      - b. Construct conditional FP-tree and generate Frequent Patterns.  
Repeat the process on each newly created Conditional FP-tree for mining (longer) frequent patterns. (...until the resulting FP-tree is empty, or it contains only one path.)

# Example: Frequent Pattern Growth Algorithm

## ■ Transaction DB

Transaction ID	Items	Transaction ID	Items
T1	{f, a, c, d, g, i, m, p}	T5	{a, f, c, e, l, p, m, n}
T2	{a, b, c, f, l, m, o}	T6	{c, j, m, b, n}
T3	{b, f, h, j, o}	T7	{d, e, f, h}
T4	{b, c, k, s, p}	T8	{a, g, i, k, s, f}

## 1-Itemset Frequency

Item	Frequency
a	4
b	4
c	5
f	6
m	4
p	3
d, e, g, h, i, j, k, l, n, o, s	2

- Step 1:  
Find the frequency of 1-itemset.

# Example: Frequent Pattern Growth Algorithm

## ■ Transaction DB

1-Itemset Frequency

Transaction ID	Items	Transaction ID	Items
T1	{f, a, c, d, g, i, m, p}	T5	{a, f, c, e, l, p, m, n}
T2	{a, b, c, f, l, m, o}	T6	{c, j, m, b, n}
T3	{b, f, h, j, o}	T7	{d, e, f, h}
T4	{b, c, k, s, p}	T8	{a, g, i, k, s, f}

Item	Frequency
a	4
b	4
c	5
f	6
m	4
p	3
<i>d, e, g, h, i, j, k, l, n, o, s</i>	<i>2</i>

## ■ Step 2: Construct Ordered-Item set.

- (Let the minimum support threshold  $s = 3$ )
- Frequent Items:  $(a, b, c, f, m, p)$

# Example: Frequent Pattern Growth Algorithm

## ■ Transaction DB

Transaction ID	Items	Transaction ID	Items
T1	{f, a, c, d, g, i, m, p}	T5	{a, f, c, e, l, p, m, n}
T2	{a, b, c, f, l, m, o}	T6	{c, j, m, b, n}
T3	{b, f, h, j, o}	T7	{d, e, f, h}
T4	{b, c, k, s, p}	T8	{a, g, i, k, s, f}

Item	Frequency
a	4
b	4
c	5
f	6
m	4
p	3



Frequent Item set:

$$L = \{f: 6, c: 5, a: 4, b: 4, m: 4, p: 3\}$$

Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	

Header Table references the occurrences of the frequent items in the FP-tree

# Example: Frequent Pattern Growth Algorithm

## Step 2: Construct Ordered-Item Set

- For (each transaction):
  - For (each item in  $L$ ):
    - If (item) in (transaction):
      - Insert (item) to (Ordered-Item Set)

$$L = \{f: 6, c: 5, a: 4, b: 4, m: 4, p: 3\}$$

Transaction ID	Items	Ordered-item Set	Transaction ID	Items	Ordered-item Set
T1	$\{f, a, c, d, g, i, m, p\}$	$\{f, c, a, m, p\}$	T5	$\{a, f, c, e, l, p, m, n\}$	
T2	$\{a, b, c, f, l, m, o\}$		T6	$\{c, j, m, b, n\}$	
T3	$\{b, f, h, j, o\}$		T7	$\{d, e, f, h\}$	
T4	$\{b, c, k, s, p\}$		T8	$\{a, g, i, k, s, f\}$	

# Example: Frequent Pattern Growth Algorithm

## ■ Step 2: Construct Ordered-Item Set

- For (each transaction):
  - For (each item in  $L$ ):
    - If (item) in (transaction):
      - Insert (item) to (Ordered-Item Set)

$$L = \{f: 6, c: 5, a: 4, b: 4, m: 4, p: 3\}$$

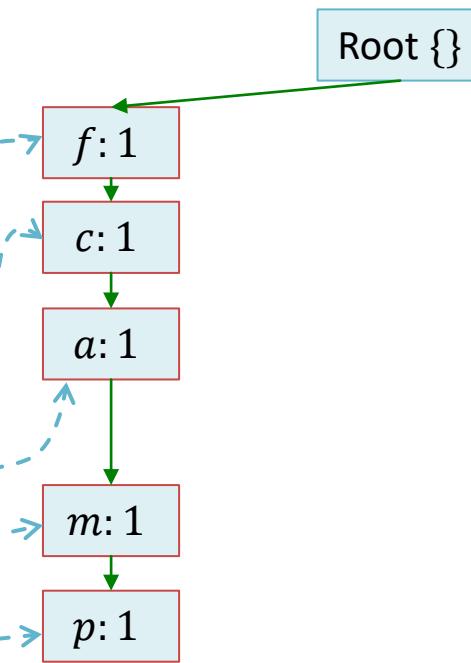
Transaction ID	Items	Ordered-Item Set	Transaction ID	Items	Ordered-Item Set
T1	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}	T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T2	{a, b, c, f, l, m, o}	{f, c, a, b, m}	T6	{c, j, m, b, n}	{c, b, m}
T3	{b, f, h, j, o}	{f, b}	T7	{d, e, f, h}	{f}
T4	{b, c, k, s, p}	{c, b, p}	T8	{a, g, i, k, s, f}	{f, a}

# Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T1	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
T2	{a, b, c, f, l, m, o}	{f, c, a, b, m}
T3	{b, f, h, j, o}	{f, b}
T4	{b, c, k, s, p}	{c, b, p}

Insert  $\{f, c, a, m, p\}$

Header Table		
Item	Frequency	header
f	6	-
c	5	-
a	4	-
b	4	-
m	4	-
p	3	-



## Step 3: Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

# Example: Frequent Pattern Growth Algorithm

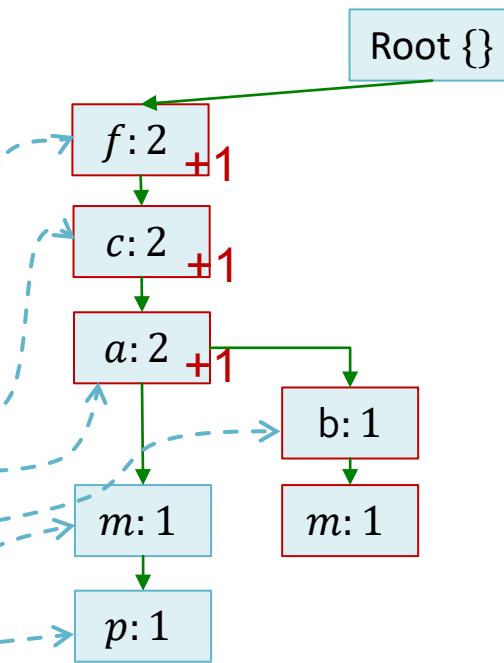
Transaction ID	Items	Ordered-Item Set
T1	$\{f, a, c, d, g, i, m, p\}$	$\{f, c, a, m, p\}$
T2	$\{a, b, c, f, l, m, o\}$	$\{f, c, a, b, m\}$
T3	$\{b, f, h, j, o\}$	$\{f, b\}$
T4	$\{b, c, k, s, p\}$	$\{c, b, p\}$

## Step 3: Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

Insert  $\{f, c, a, b, m\}$

Header Table		
Item	Frequency	header
f	6	-
c	5	-
a	4	-
b	4	-
m	4	-
p	3	-



# Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T1	$\{f, a, c, d, g, i, m, p\}$	$\{f, c, a, m, p\}$
T2	$\{a, b, c, f, l, m, o\}$	$\{f, c, a, b, m\}$
T3	$\{b, f, h, j, o\}$	$\{f, b\}$
T4	$\{b, c, k, s, p\}$	$\{c, b, p\}$

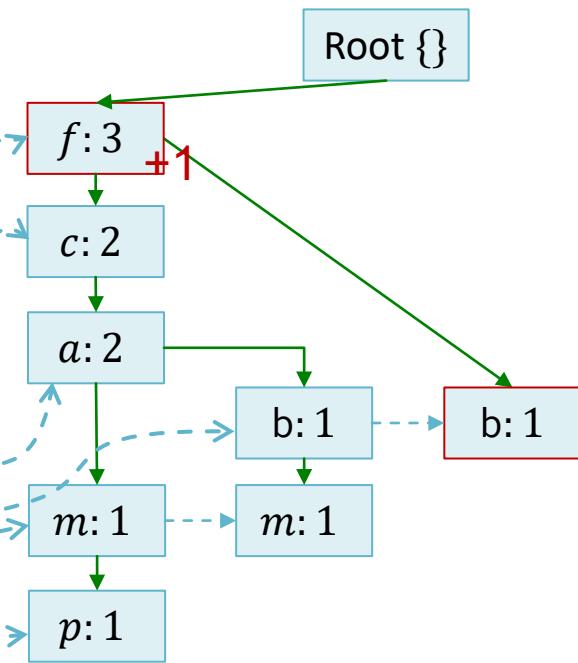
## Step 3: Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

Insert  $\{f, b\}$

Header Table

Item	Frequency	header
f	6	-
c	5	-
a	4	-
b	4	-
m	4	-
p	3	-



# Example: Frequent Pattern Growth Algorithm

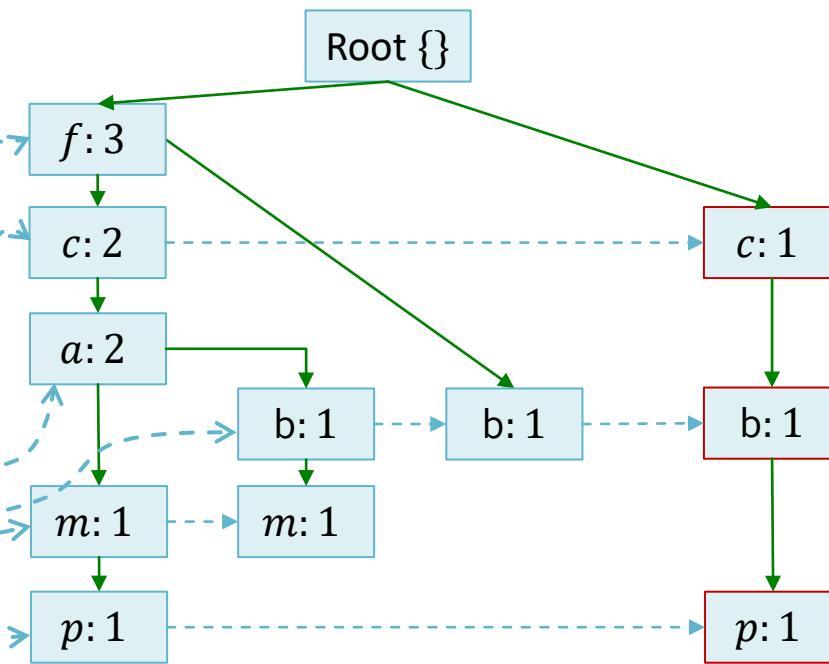
Transaction ID	Items	Ordered-Item Set
T1	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
T2	{a, b, c, f, l, m, o}	{f, c, a, b, m}
T3	{b, f, h, j, o}	{f, b}
T4	{b, c, k, s, p}	{c, b, p}

Insert {c, b, p}

Header Table		
Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	

## Step 3: Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.



# Example: Frequent Pattern Growth Algorithm

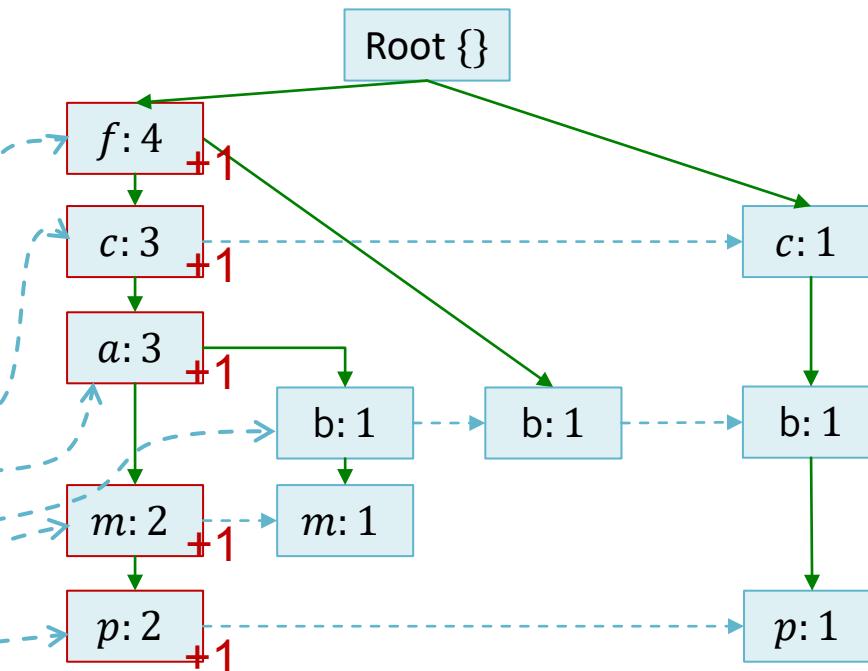
Transaction ID	Items	Ordered-Item Set
T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T6	{c, j, m, b, n}	{c, b, m}
T7	{d, e, f, h}	{f}
T8	{a, g, i, k, s, f}	{f, a}

Insert  $\{f, c, a, m, p\}$

Header Table		
Item	Frequency	header
f	6	-
c	5	-
a	4	-
b	4	-
m	4	-
p	3	-

## Step 3: Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.



# Example: Frequent Pattern Growth Algorithm

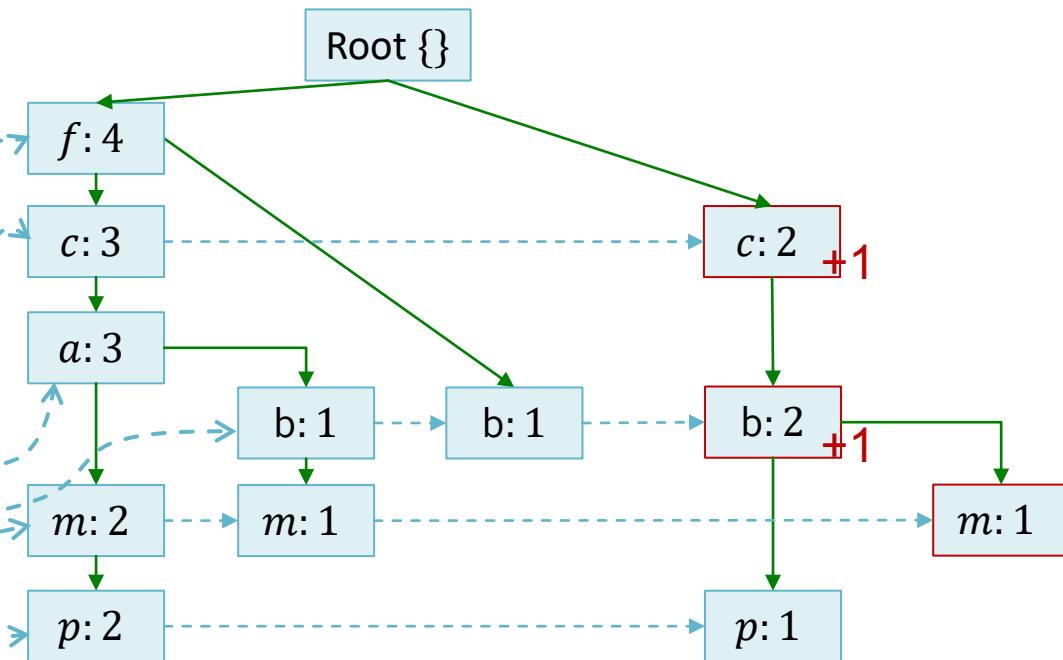
Transaction ID	Items	Ordered-Item Set
T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T6	{c, j, m, b, n}	{c, b, m}
T7	{d, e, f, h}	{f}
T8	{a, g, i, k, s, f}	{f, a}

Insert {c, b, m}

Header Table		
Item	Frequency	header
f	6	-
c	5	-
a	4	-
b	4	-
m	4	-
p	3	-

## Step 3: Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.



# Example: Frequent Pattern Growth Algorithm

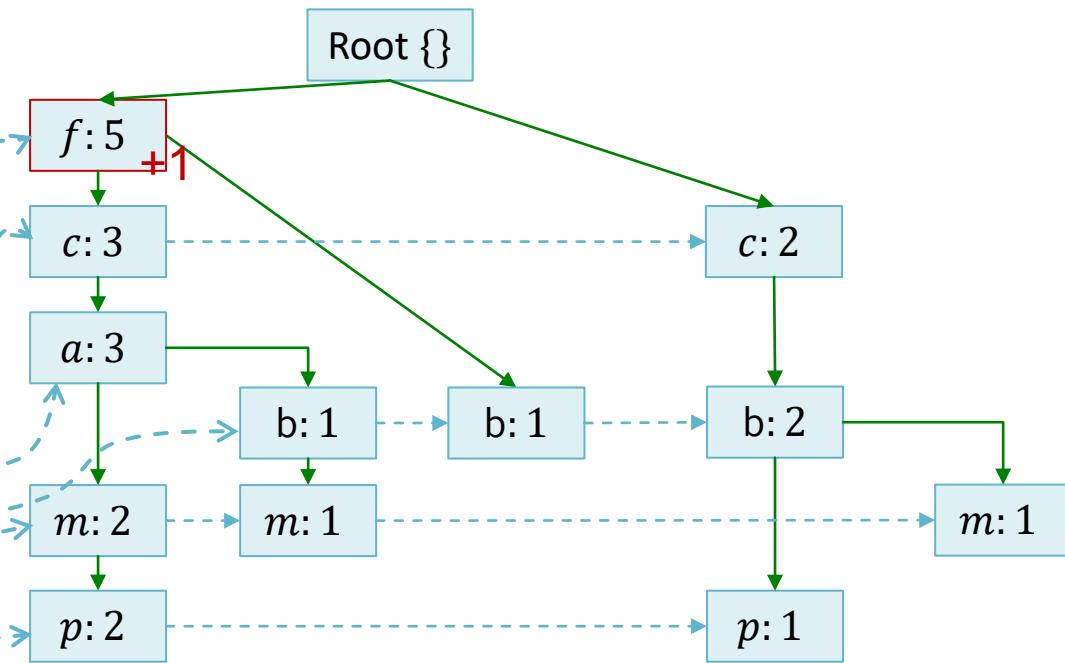
Transaction ID	Items	Ordered-Item Set
T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T6	{c, j, m, b, n}	{c, b, m}
T7	{d, e, f, h}	{f}
T8	{a, g, i, k, s, f}	{f, a}

Insert {f}

Header Table		
Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	

## Step 3: Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

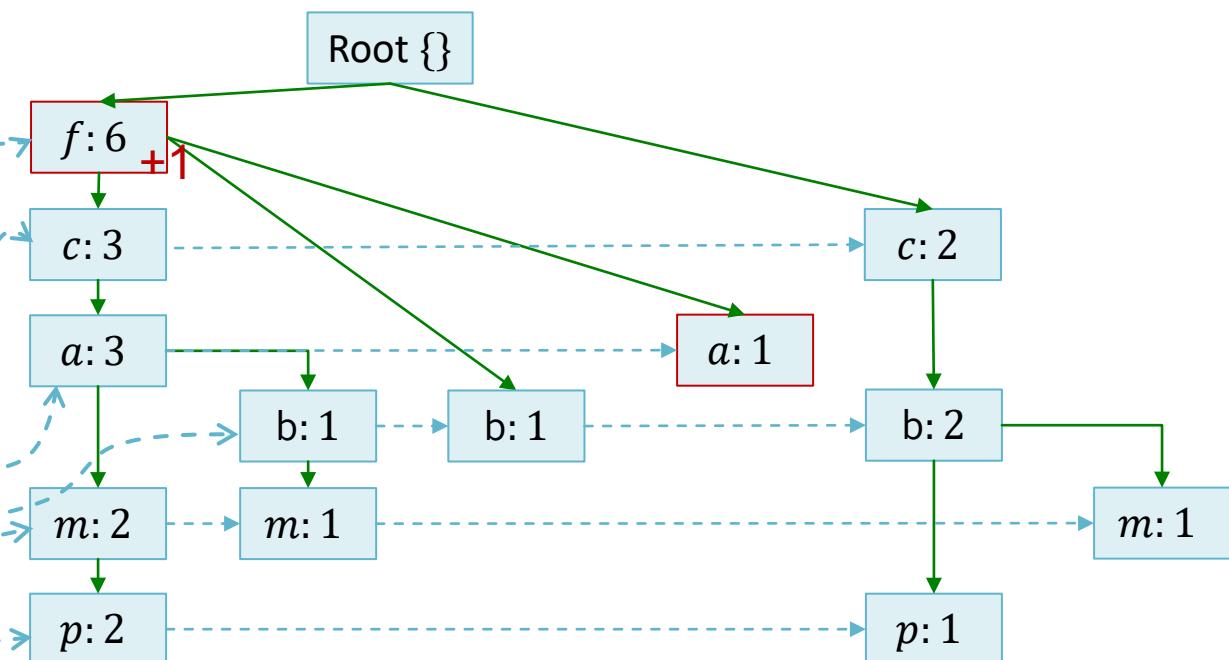


# Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T5	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}
T6	{c, j, m, b, n}	{c, b, m}
T7	{d, e, f, h}	{f}
T8	{a, g, i, k, s, f}	{f, a}

Insert {f, a}

Header Table		
Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	

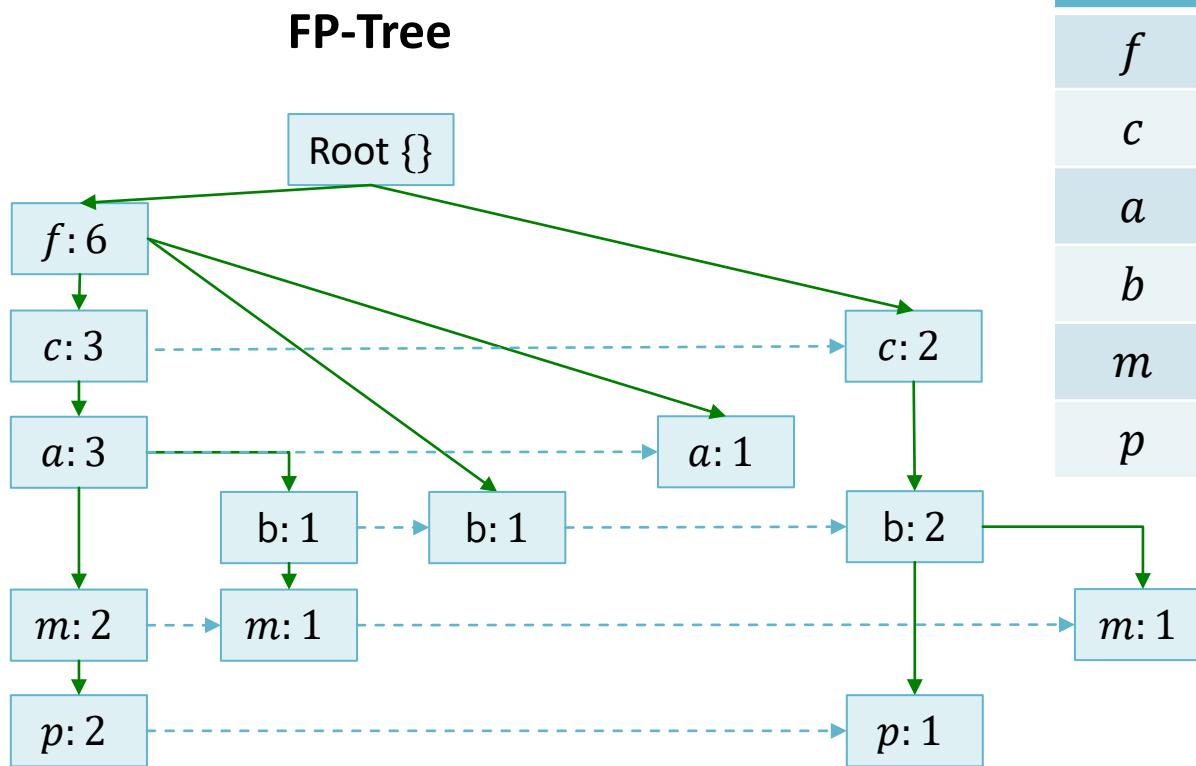


## Step 3: Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

# Example: Frequent Pattern Growth Algorithm

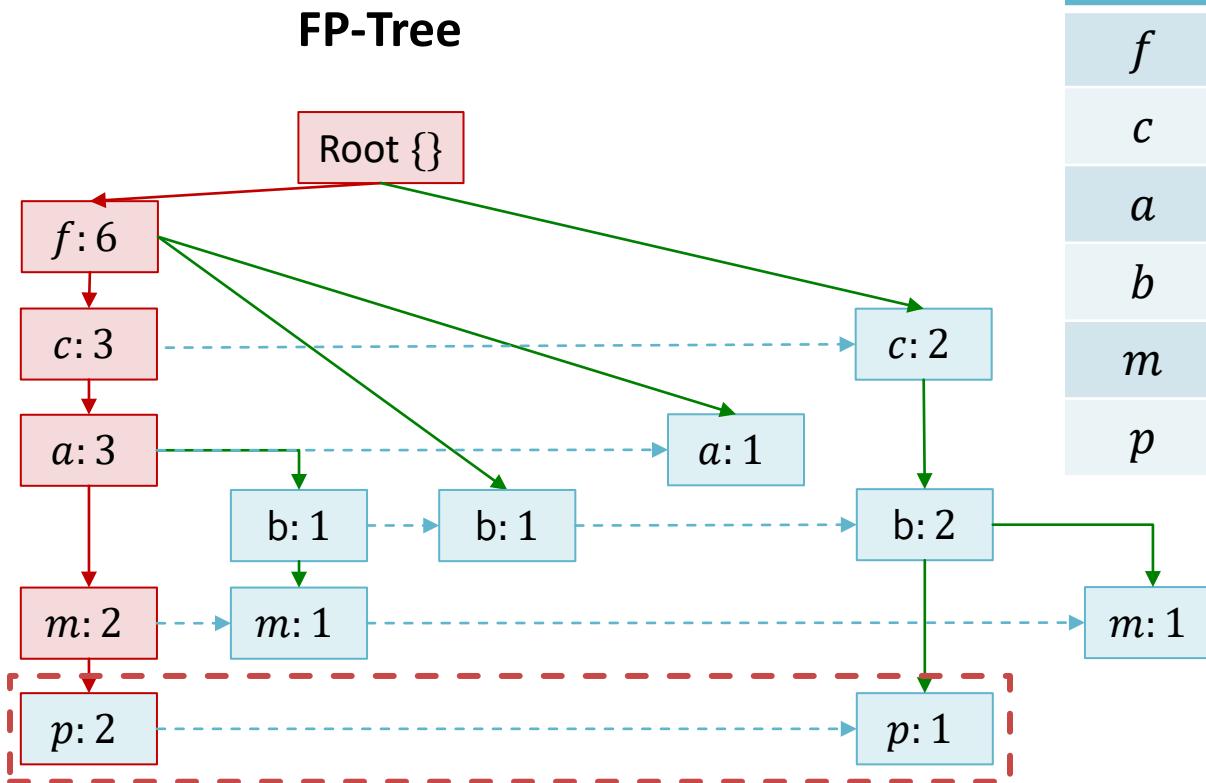
- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).



Items	Conditional Data Base
f	
c	
a	
b	
m	
p	

# Example: Frequent Pattern Growth Algorithm

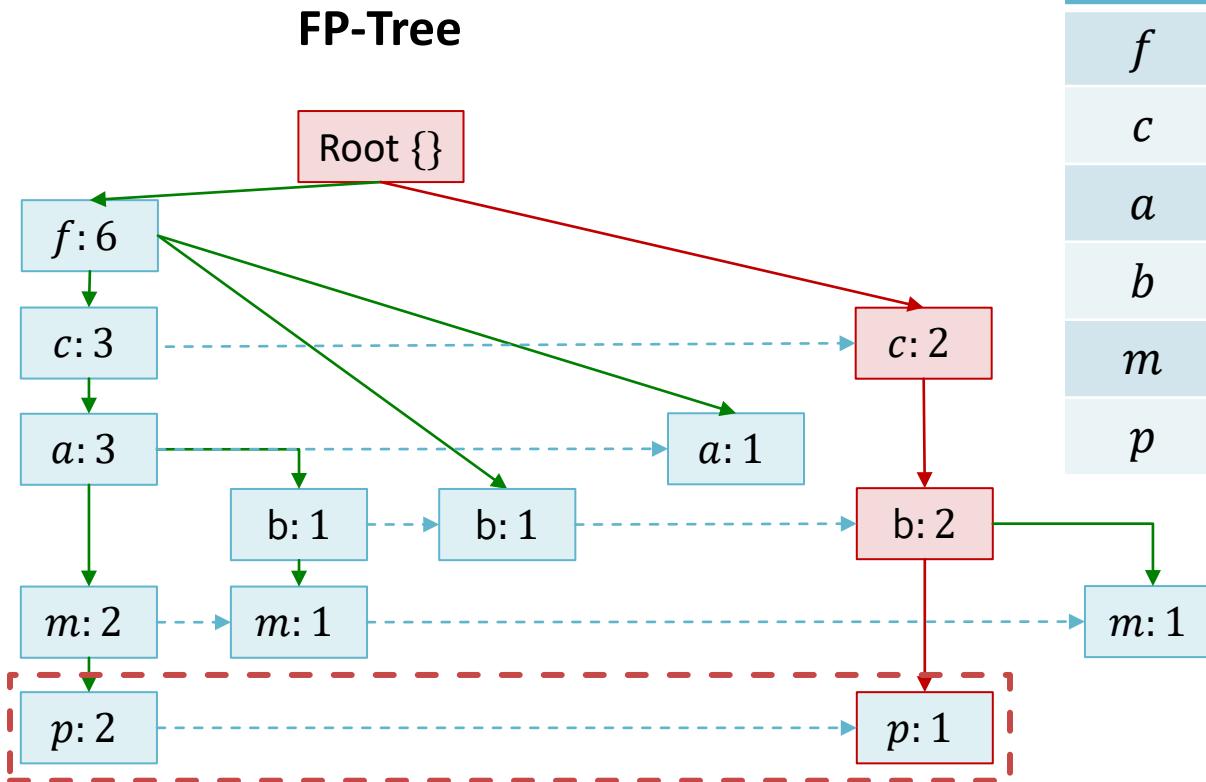
- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).



Items	Conditional Data Base
$f$	
$c$	
$a$	
$b$	
$m$	
$p$	$\{f, c, a, m: 2\}$

# Example: Frequent Pattern Growth Algorithm

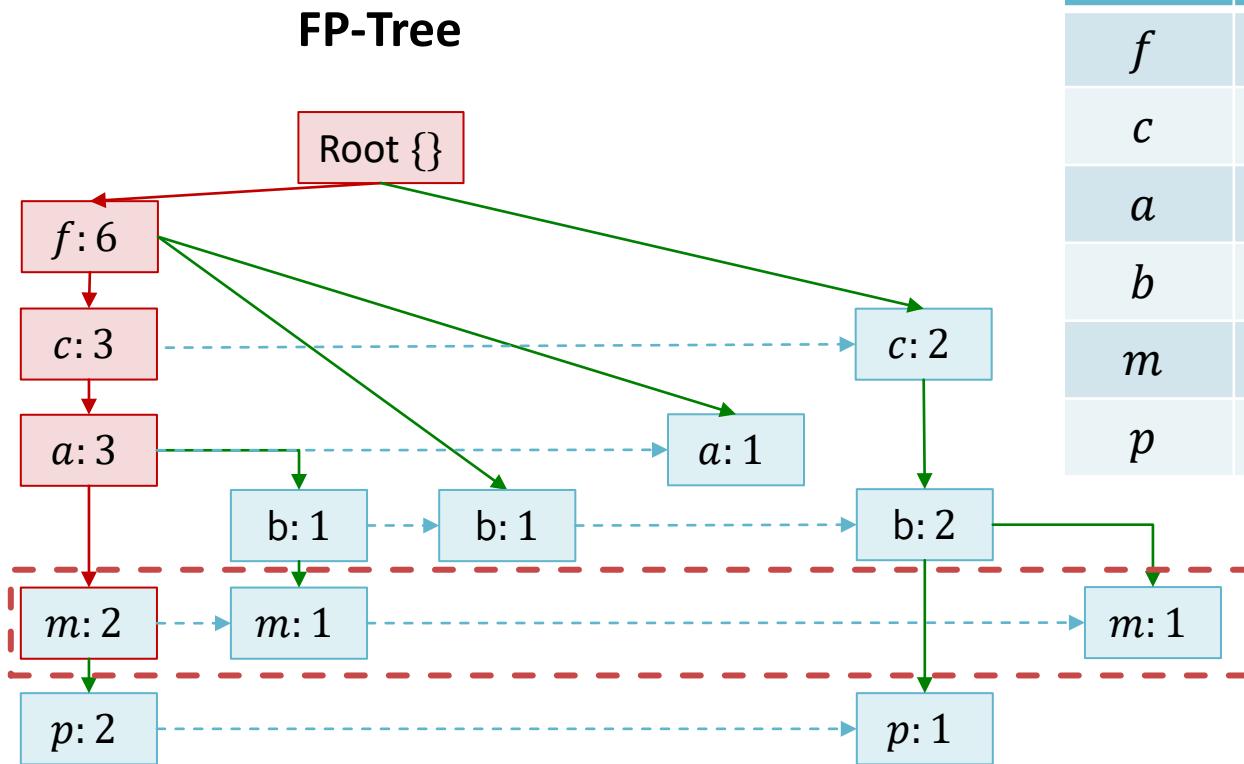
- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).



Items	Conditional Data Base
f	
c	
a	
b	
m	
p	{f, c, a, m: 2}, {c, b: 1}

# Example: Frequent Pattern Growth Algorithm

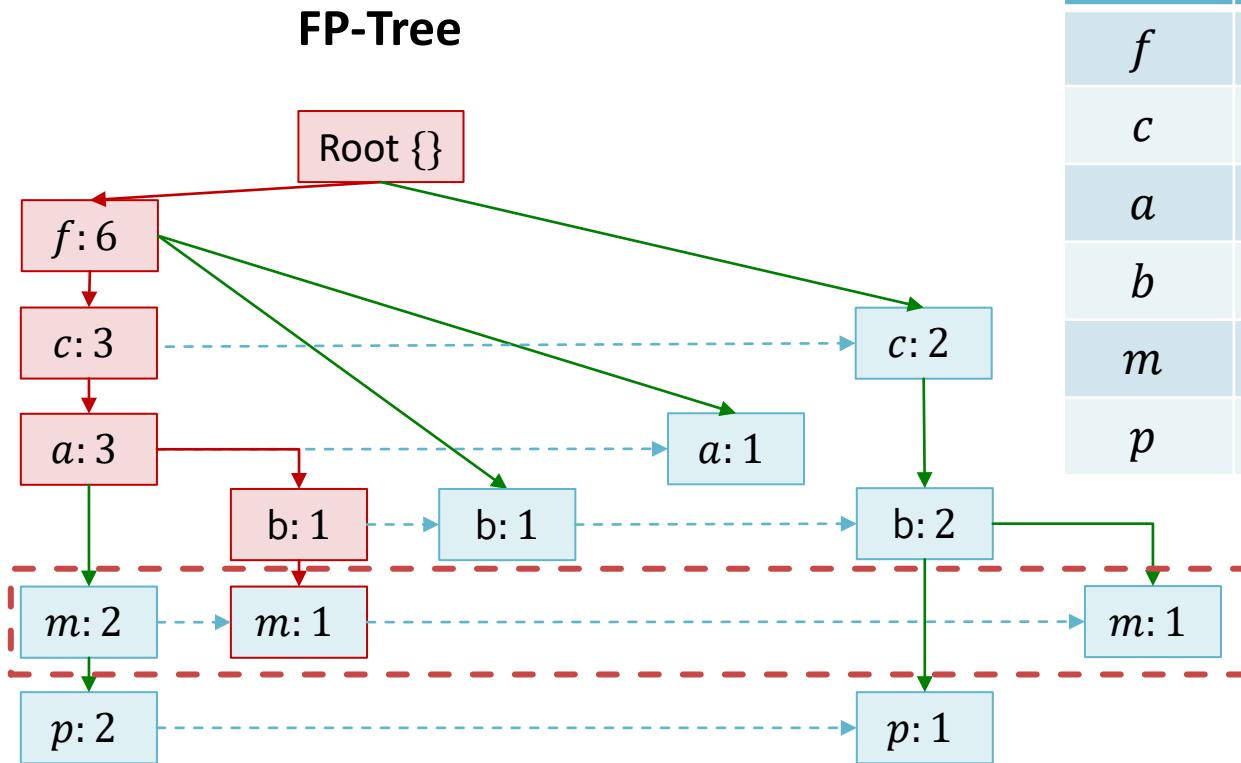
- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).



Items	Conditional Data Base
$f$	
$c$	
$a$	
$b$	
$m$	$\{f, c, a: 2\}$
$p$	$\{f, c, a, m: 2\}, \{c, b: 1\}$

# Example: Frequent Pattern Growth Algorithm

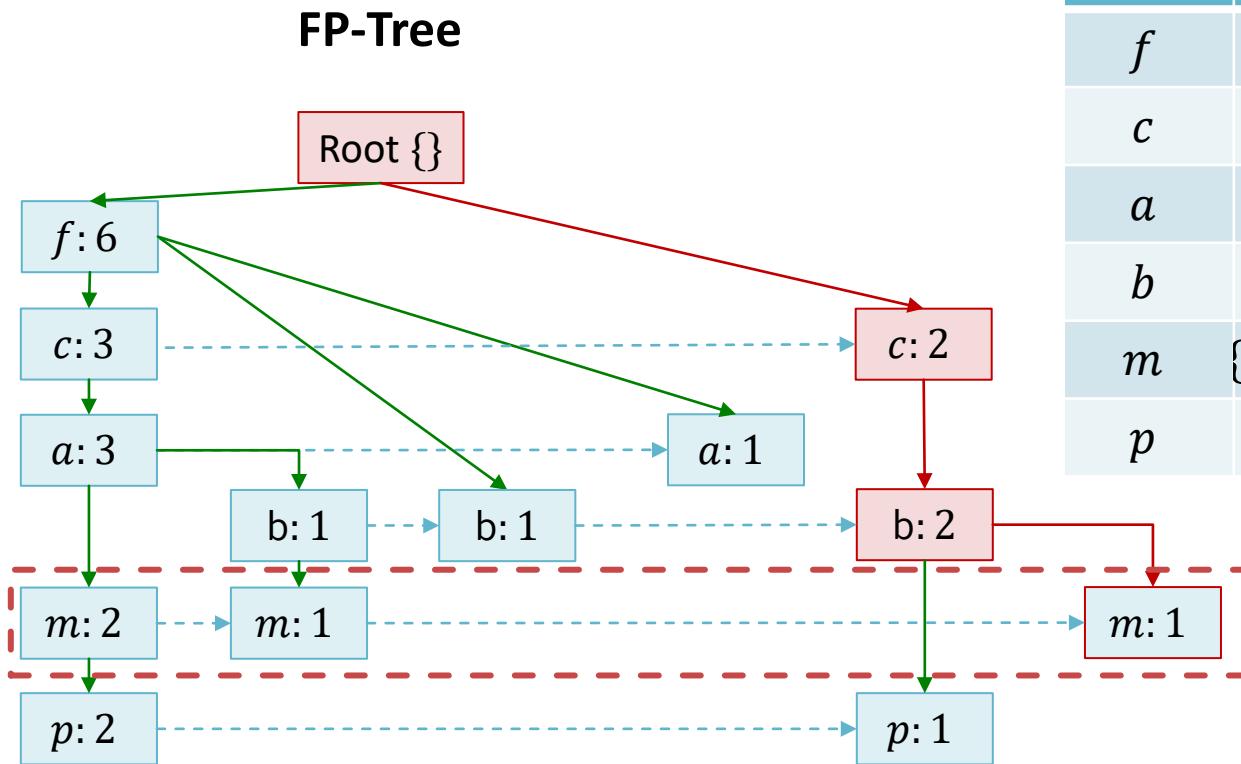
- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).



Items	Conditional Data Base
$f$	
$c$	
$a$	
$b$	
$m$	$\{f, c, a: 2\}, \{f, c, a, b: 1\}$
$p$	$\{f, c, a, m: 2\}, \{c, b: 1\}$

# Example: Frequent Pattern Growth Algorithm

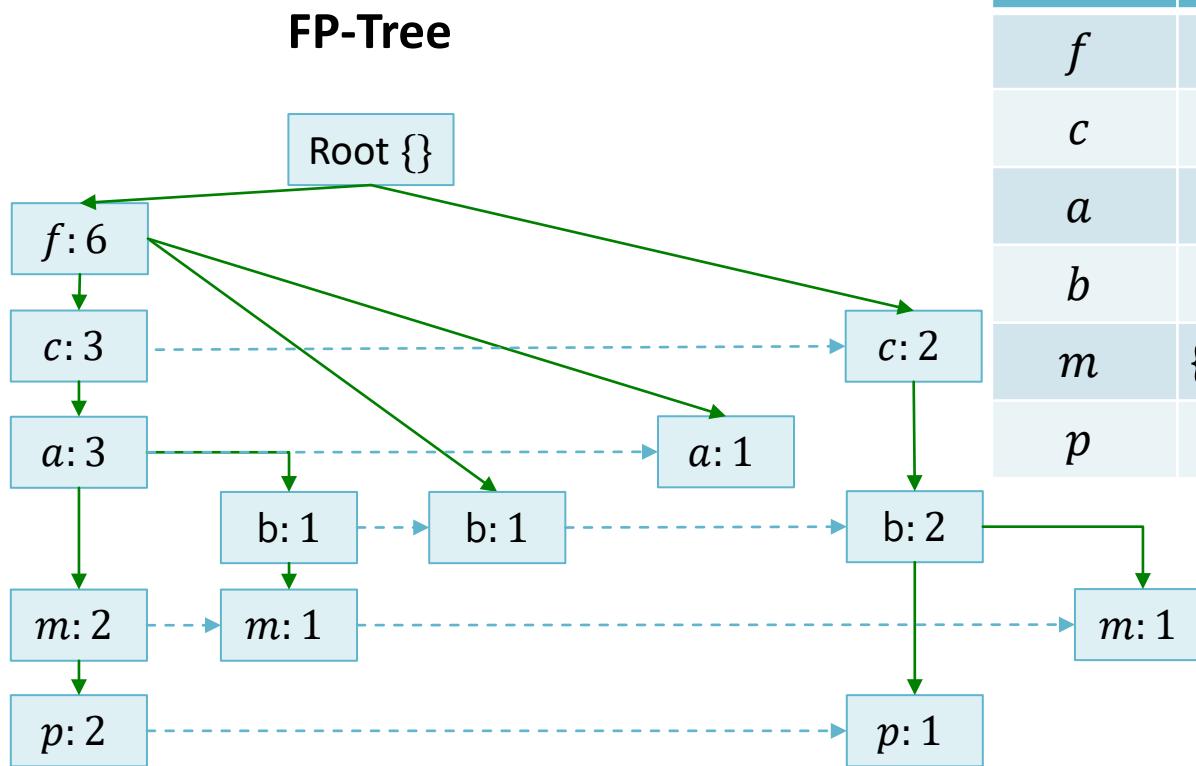
- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).



Items	Conditional Data Base
$f$	
$c$	
$a$	
$b$	
$m$	$\{f, c, a: 2\}, \{f, c, a, b: 1\}, \{c, b: 1\}$
$p$	$\{f, c, a, m: 2\}, \{c, b: 1\}$

# Example: Frequent Pattern Growth Algorithm

- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).



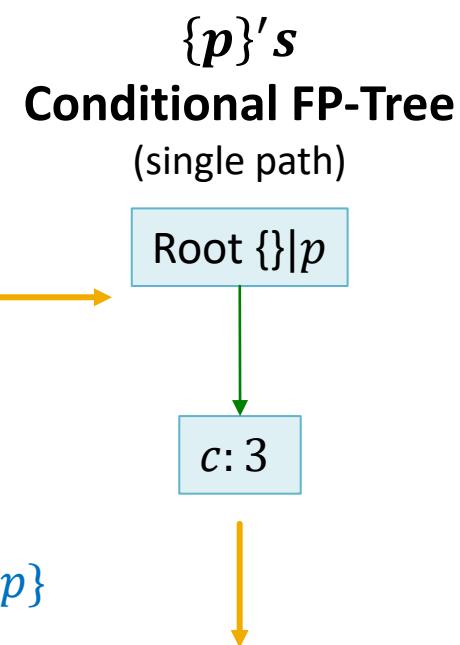
Items	Conditional Data Base
$f$	{}
$c$	$\{f: 3\}$
$a$	$\{f, c: 3\}, \{f: 1\}$
$b$	$\{f, c, a: 1\}, \{f: 1\}, \{c: 2\}$
$m$	$\{f, c, a: 2\}, \{f, c, a, b: 1\}, \{c, b: 1\}$
$p$	$\{f, c, a, m: 2\}, \{c, b: 1\}$

# Example: Frequent Pattern Growth Algorithm

- Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns
  - Find frequent items from each  $\{X\}$ 's conditional DB. Each frequent item  $i$  and  $X$  will form a new frequent pattern  $\{i, X\}$
  - Respective conditional FP-tree is constructed.

Items	Conditional Data Base
$f$	$\{\}$
$c$	$\{f: 3\}$
$a$	$\{f, c: 3\}, \{f: 1\}$
$b$	$\{f, c, a: 1\}, \{f: 1\}, \{c: 2\}$
$m$	$\{f, c, a: 2\}, \{f, c, a, b: 1\}, \{c, b: 1\}$
$p$	$\{f, c, a, m: 2\}, \{c, b: 1\}$

Item	#
$f$	2
$c$	3
$a$	2
$b$	1
$m$	2



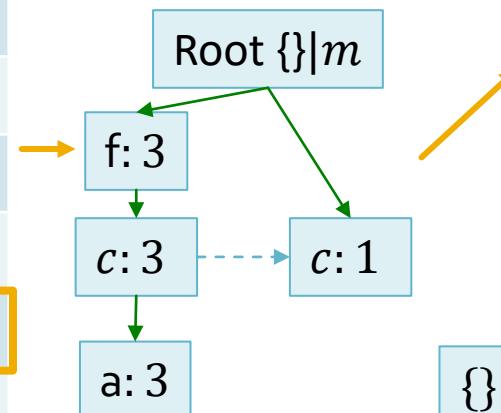
Frequent patterns contain  $p$ :  
 $\{\{p\}, \{c, p\}\}$

# Example: Frequent Pattern Growth Algorithm

- Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns
  - Find frequent items from each  $\{X\}$ 's conditional DB. Each frequent item  $i$  and  $X$  will form a new frequent pattern  $\{i, X\}$
  - Respective conditional FP-tree is constructed.

Items	Conditional Data Base
$f$	$\emptyset$
$c$	$\{f: 3\}$
$a$	$\{f, c: 3\}, \{f: 1\}$
$b$	$\{f, c, a: 1\}, \{f: 1\}, \{c: 2\}$
$m$	$\{f, c, a: 2\}, \{f, c, a, b: 1\}, \{c, b: 1\}$
$p$	$\{f, c, a, m: 2\}, \{c, b: 1\}$

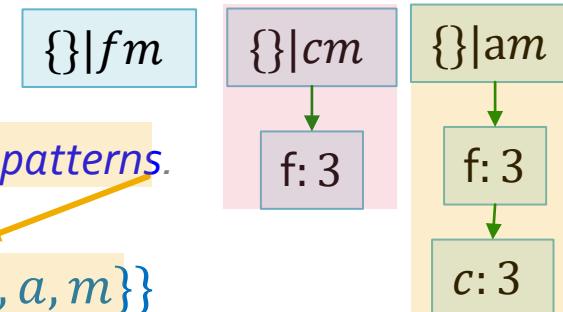
Conditional FP-Tree  
(multiple paths)



Conditional Database 2

Items	Cond DB
$mf$	$\emptyset$
$mc$	$\{f: 3\}$
$ma$	$\{f, c: 3\}$

Conditional FP-Tree 2



If the conditional FP-tree contains a single path, simply enumerate all patterns.

Frequent patterns contain  $m$  but not  $p$ :

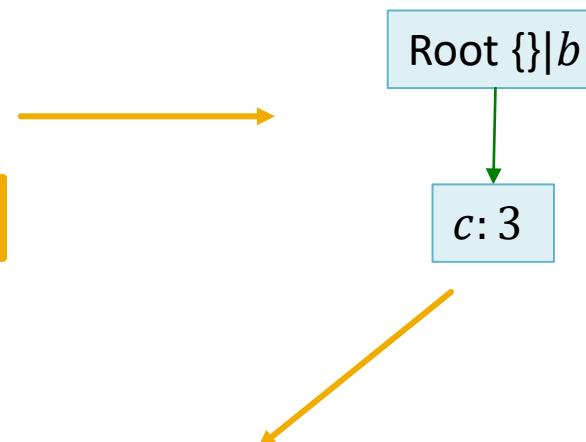
$\{\{m\}, \{f, m\}, \{c, m\}, \{f, c, m\}, \{a, m\}, \{f, c, a, m\}, \{c, a, m\}, \{f, a, m\}\}$

# Example: Frequent Pattern Growth Algorithm

- Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns
  - Find frequent items from each  $\{X\}$ 's conditional DB. Each frequent item  $i$  and  $X$  will form a new frequent pattern  $\{i, X\}$
  - Respective conditional FP-tree is constructed.

Items	Conditional Data Base
$f$	$\{\}$
$c$	$\{f: 3\}$
$a$	$\{f, c: 3\}, \{f: 1\}$
$b$	$\{f, c, a: 1\}, \{f: 1\}, \{c: 2\}$
$m$	$\{f, c, a: 2\}, \{f, c, a, b: 1\}, \{c, b: 1\}$
$p$	$\{f, c, a, m: 2\}, \{c, b: 1\}$

Conditional FP-Tree  
(single path)

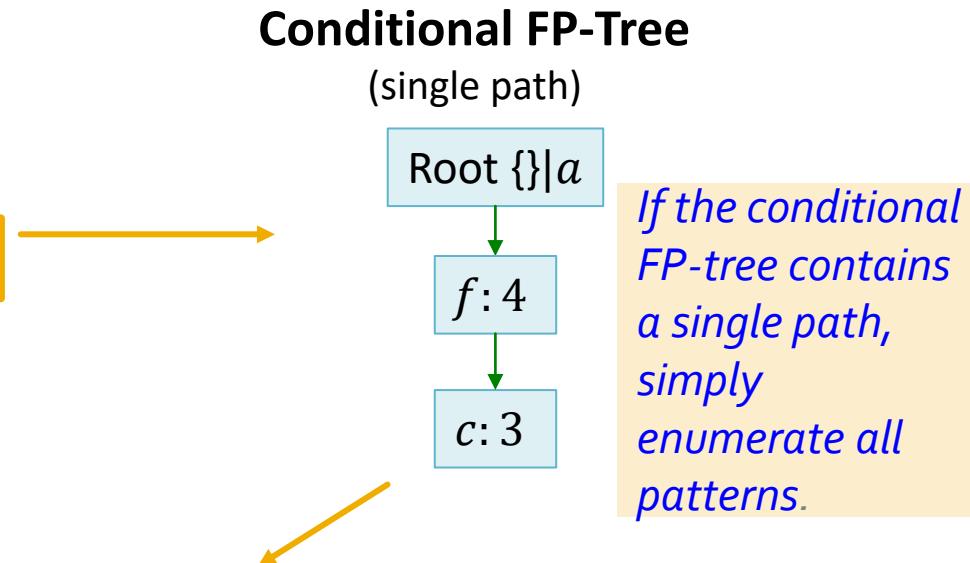


Frequent patterns contain  $b$  but not  $m$  or  $p$ :  
 $\{\{b\}, \{c, b\}\}$

# Example: Frequent Pattern Growth Algorithm

- Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns
  - Find frequent items from each  $\{X\}$ 's conditional DB. Each frequent item  $i$  and  $X$  will form a new frequent pattern  $\{i, X\}$
  - Respective conditional FP-tree is constructed.

Items	Conditional Data Base
$f$	$\{\}$
$c$	$\{f: 3\}$
$a$	$\{f, c: 3\}, \{f: 1\}$
$b$	$\{f, c, a: 1\}, \{f: 1\}, \{c: 2\}$
$m$	$\{f, c, a: 2\}, \{f, c, a, b: 1\}, \{c, b: 1\}$
$p$	$\{f, c, a, m: 2\}, \{c, b: 1\}$



Frequent patterns contain  $a$  but not  $m$  or  $p$  or  $b$ :  
 $\{\{a\}, \{f, a\}, \{c, a\}, \{f, c, a\}\}$

# Example: Frequent Pattern Growth Algorithm

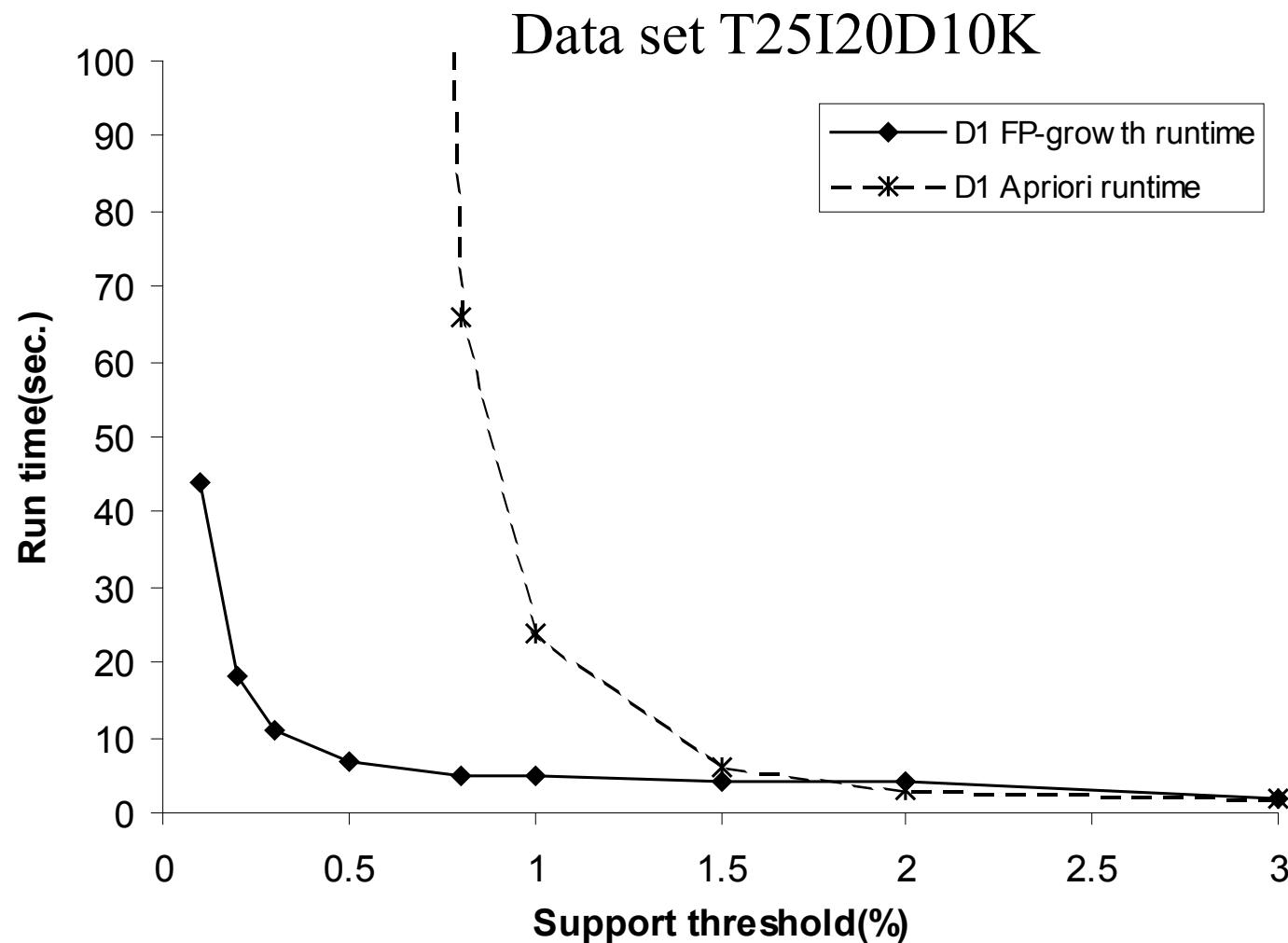
## Final results

Frequent Pattern Subsets	Frequent Patterns
Patterns contain $p$	$\{\{p\}, \{c, p\}\}$
Patterns contain $m$ but not $p$	$\{\{m\}, \{f, m\}, \{c, m\}, \{f, c, m\}, \{a, m\}, \{f, c, a, m\}, \{c, a, m\}, \{f, a, m\}\}$
Patterns contain $b$ but not $p$ or $m$	$\{\{b\}, \{c, b\}\}$
Patterns contain $a$ but not $p$ or $m$ or $b$	$\{\{a\}, \{f, a\}, \{c, a\}, \{f, c, a\}\}$
Patterns contain $c$ but not $p, m, b, a$	$\{\{c\}, \{f, c\}\}$
Patterns contain $f$ but not $p, m, b, a, c$	$\{\{f\}\}$

# Summary of Mining Frequent Patterns Using FP-tree

- General idea (divide-and-conquer)
  - Recursively grow frequent patterns using FP-tree
- Frequent patterns can be partitioned into subsets according to L-order
  - L-order=  $f: 6, c: 5, a: 4, b: 4, m: 4, p: 3$
  - Patterns containing p
  - Patterns having m but no p
  - Patterns having b but no m or p
  - ...
  - Patterns having c but no a nor b, m, p
  - Pattern f

# FP-growth vs. Apriori: Scalability With the Support Threshold



# Why Is Frequent Pattern Growth Fast?

- Performance study shows
  - FP-growth can be an order of magnitude faster than Apriori
- Reasons
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operations are counting and FP-tree building
- Challenges
  - When FP-tree cannot fit in memory

# Outline

- A different frequent itemset/pattern mining algorithm
- Sequential pattern mining

# Examples of Sequence

- Sequence of different transactions by a customer at an online store:  
< {Digital Camera,iPad} {memory card} {headphone,iPad cover} >
- Sequence of initiating events causing the nuclear accident at 3-mile Island:  
([http://stellar-one.com/nuclear/staff\\_reports/summary\\_SOE\\_the\\_initiating\\_event.htm](http://stellar-one.com/nuclear/staff_reports/summary_SOE_the_initiating_event.htm))  
< {clogged resin} {outlet valve closure} {loss of feedwater}  
  {condenser polisher outlet valve shut} {booster pumps trip}  
  {main waterpump trips} {main turbine trips} {reactor pressure increases}>
- Sequence of books checked out at a library:  
<{Fellowship of the Ring} {The Two Towers} {Return of the King}>

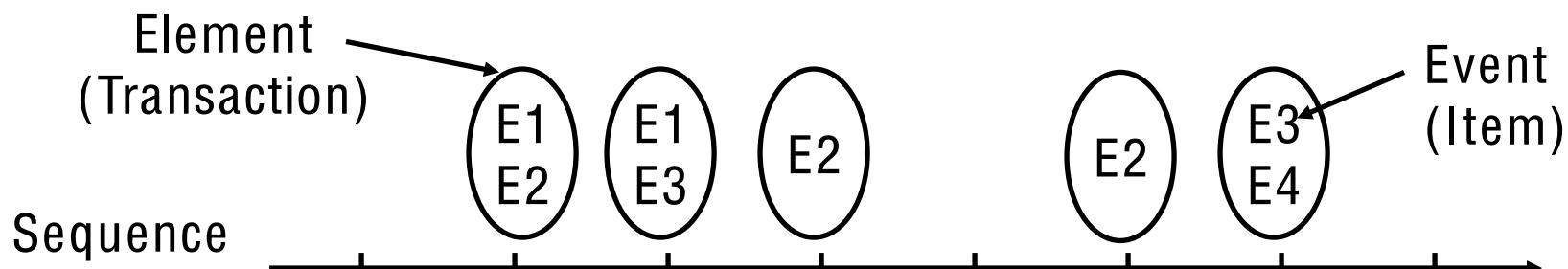
# Sequential Pattern Discovery: Examples

- In point-of-sale transaction sequences,
  - Computer Bookstore:  
(Intro\_To\_Visual\_C) (C++\_Primer) -->  
(Perl\_for\_dummies,Tcl\_Tk)
  - Athletic Apparel Store:  
(Shoes) (Racket, Racketball) --> (Sports\_Jacket)
- Detecting Erroneous Sentences using Automatically Mined Sequential Patterns [1]
  - <the, more, the, JJ (base form of adjective)> in erroneous sentences

[1] Guihua Sun, Xiaohua Liu, Gao Cong, Ming Zhou, Zhongyang Xiong, John Lee, Chin-Yew Lin:  
*Detecting Erroneous Sentences using Automatically Mined Sequential Patterns*. ACL 2007

# Examples of Sequence Data

Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	An element of the DNA sequence	Bases A,T,G,C



# Sequence Data vs. Market-basket Data

Sequence Database:

Customer	Date	Items bought
A	10	2, 3, 5
A	20	1,6
A	23	1
B	11	4, 5, 6
B	17	2
B	21	1,2,7,8
B	28	1, 6
C	14	1,7,8

Market- basket Data

Events
2, 3, 5
1,6
1
4,5,6
2
1,2,7,8
1,6
1,7,8

# Formal Definition of a Sequence

- A sequence is an ordered list of **elements**

$$s = \langle e_1, e_2, e_3, \dots \rangle$$

- Each element contains a collection of **events (items)**

$$e_i = \{i_1, i_2, \dots, i_k\}$$

- **Length** of a sequence,  $|s|$ , is given by the number of elements in the sequence
- A **k-sequence** is a sequence that contains **k events (items)**
  - $\langle \{a,b\} \{a\} \rangle$  has a length of 2 and it is a 3-sequence

# Formal Definition of a Subsequence

- A sequence  $t: < a_1 \ a_2 \ \dots \ a_n >$  is **contained** in another sequence  $s: < b_1 \ b_2 \ \dots \ b_m >$  ( $m \geq n$ ) if there exist integers  $i_1 < i_2 < \dots < i_n$  such that  $a_1 \subseteq b_{i1}, a_2 \subseteq b_{i2}, \dots, a_n \subseteq b_{in}$
- Illustrative Example:

s:             $b_1$              $b_2$              $b_3$              $b_4$              $b_5$   
t:             $a_1$              $a_2$                                      $a_3$

t is a subsequence of s if  $a_1 \subseteq b_2, a_2 \subseteq b_3, a_3 \subseteq b_5$ .

Data sequence	Subsequence	Contain?
$< \{2,4\} \{3,5,6\} \{8\} >$	$< \{2\} \{8\} >$	Yes
$< \{1,2\} \{3,4\} >$	$< \{1\} \{2\} >$	No
$< \{2,4\} \{2,4\} \{2,5\} >$	$< \{2\} \{4\} >$	Yes
$< \{2,4\} \{2,5\} \{4,5\} >$	$< \{2\} \{4\} \{5\} >$	No
$< \{2,4\} \{2,5\} \{4,5\} >$	$< \{2\} \{5\} \{5\} >$	Yes
$< \{2,4\} \{2,5\} \{4,5\} >$	$< \{2, 4, 5\} >$	No

# Sequential Pattern Mining: Definition

- The **support** of a subsequence  $w$  is defined as the fraction (or number) of data sequences that contain  $w$ . *We use fraction in the rest slides*
- A *sequential pattern* is a **frequent subsequence** (i.e., a subsequence whose support is  $\geq \text{minsup}$ )
- Given:
  - a database of sequences
  - a user-specified minimum support threshold,  $\text{minsup}$
- Task:
  - Find all subsequences with support  $\geq \text{minsup}$

# Sequential Pattern Mining: Example

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

$minsup = 3$

Examples of Frequent Subsequences:

- |                                   |       |
|-----------------------------------|-------|
| $\langle \{1,2\} \rangle$         | $s=3$ |
| $\langle \{2,3\} \rangle$         | $s=3$ |
| $\langle \{2,4\} \rangle$         | $s=4$ |
| $\langle \{3\} \{5\} \rangle$     | $s=4$ |
| $\langle \{1\} \{2\} \rangle$     | $s=4$ |
| $\langle \{2\} \{2\} \rangle$     | $s=3$ |
| $\langle \{1\} \{2,3\} \rangle$   | $s=3$ |
| $\langle \{2\} \{2,3\} \rangle$   | $s=3$ |
| $\langle \{1,2\} \{2,3\} \rangle$ | $s=3$ |

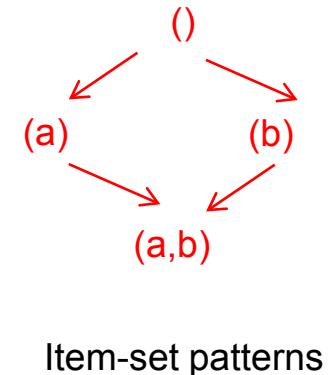
# Extracting Sequential Patterns

- Given  $n$  events:  $i_1, i_2, i_3, \dots, i_n$
- Candidate 1 subsequences:  
 $\langle\{i_1\}\rangle, \langle\{i_2\}\rangle, \langle\{i_3\}\rangle, \dots, \langle\{i_n\}\rangle$
- Candidate 2 subsequences:  
 $\langle\{i_1, i_2\}\rangle, \langle\{i_1, i_3\}\rangle, \dots,$   
 $\langle\{i_1\} \{i_1\}\rangle, \langle\{i_1\} \{i_2\}\rangle, \dots, \langle\{i_n\} \{i_n\}\rangle$
- Candidate 3 subsequences:  
 $\langle\{i_1, i_2, i_3\}\rangle, \langle\{i_1, i_2, i_4\}\rangle, \dots,$   
 $\langle\{i_1, i_2\} \{i_1\}\rangle, \langle\{i_1, i_2\} \{i_2\}\rangle, \dots,$   
 $\langle\{i_1\} \{i_1, i_2\}\rangle, \langle\{i_1\} \{i_1, i_3\}\rangle, \dots,$   
 $\langle\{i_1\} \{i_1\} \{i_1\}\rangle, \langle\{i_1\} \{i_1\} \{i_2\}\rangle, \dots$

# Extracting Sequential Patterns: Simple example

- Given 2 events: a, b
- Candidate 1 subsequences:

$\langle \{a\} \rangle, \langle \{b\} \rangle.$



- Candidate 2 subsequences:  
 $\langle \{a\} \{a\} \rangle, \langle \{a\} \{b\} \rangle, \langle \{b\} \{a\} \rangle, \langle \{b\} \{b\} \rangle, \langle \{a, b\} \rangle.$
- Candidate 3 subsequences:  
 $\langle \{a\} \{a\} \{a\} \rangle, \langle \{a\} \{a\} \{b\} \rangle, \langle \{a\} \{b\} \{a\} \rangle, \langle \{a\} \{b\} \{b\} \rangle,$   
 $\langle \{b\} \{b\} \{b\} \rangle, \langle \{b\} \{b\} \{a\} \rangle, \langle \{b\} \{a\} \{b\} \rangle, \langle \{b\} \{a\} \{a\} \rangle$   
 $\langle \{a, b\} \{a\} \rangle, \langle \{a, b\} \{b\} \rangle, \langle \{a\} \{a, b\} \rangle, \langle \{b\} \{a, b\} \rangle$

# Generalized Sequential Pattern (GSP)

- **Step 1:**
  - Make the first pass over the sequence database D to yield all the 1-element frequent sequences
- **Step 2:** Repeat until no new frequent sequences are found
  - **Candidate Generation:**
    - Merge pairs of frequent subsequences found in the  $(k-1)th$  pass to generate candidate sequences that contain k items
  - **Candidate Pruning (Apriori):**
    - Prune candidate  $k$ -sequences that contain infrequent  $(k-1)$ -subsequences
  - **Support Counting:**
    - Make a new pass over the sequence database D to find the support for these candidate sequences
  - **Candidate Elimination:**
    - Eliminate candidate  $k$ -sequences whose actual support is less than  $minsup$

# Candidate Generation

- Base case ( $k=2$ ):
  - Merging two frequent 1-sequences  $\langle\{i_1\}\rangle$  and  $\langle\{i_2\}\rangle$  will produce the following candidate 2-sequences:  $\langle\{i_1\} \{i_1\}\rangle$ ,  $\langle\{i_1\} \{i_2\}\rangle$ ,  $\langle\{i_2\} \{i_2\}\rangle$ ,  $\langle\{i_2\} \{i_1\}\rangle$  and  $\langle\{i_1, i_2\}\rangle$ . (Note:  $\langle\{i_1\}\rangle$  can be merged with itself to produce:  $\langle\{i_1\} \{i_1\}\rangle$ )
- General case ( $k>2$ ):
  - A frequent  $(k-1)$ -sequence  $w_1$  is merged with another frequent  $(k-1)$ -sequence  $w_2$  to produce a candidate  $k$ -sequence if **the subsequence obtained by removing an event from the first element in  $w_1$  is the same as the subsequence obtained by removing an event from the last element in  $w_2$**

# Candidate Generation

- Base case ( $k=2$ ):
  - Merging two frequent 1-sequences  $\langle\{i_1\}\rangle$  and  $\langle\{i_2\}\rangle$  will produce the following candidate 2-sequences:  $\langle\{i_1\} \{i_1\}\rangle$ ,  $\langle\{i_1\} \{i_2\}\rangle$ ,  $\langle\{i_2\} \{i_2\}\rangle$ ,  $\langle\{i_2\} \{i_1\}\rangle$  and  $\langle\{i_1 i_2\}\rangle$ . (Note:  $\langle\{i_1\}\rangle$  can be merged with itself to produce:  $\langle\{i_1\} \{i_1\}\rangle$ )
- General case ( $k>2$ ):
  - A frequent  $(k-1)$ -sequence  $w_1$  is merged with another frequent  $(k-1)$ -sequence  $w_2$  to produce a candidate  $k$ -sequence if **the subsequence obtained by removing an event from the first element in  $w_1$  is the same as the subsequence obtained by removing an event from the last element in  $w_2$** 
    - The resulting candidate after merging is given by extending the sequence  $w_1$  as follows-
      - If the last element of  $w_2$  has only one event, append it to  $w_1$
      - Otherwise add the event from the last element of  $w_2$  (which is absent in the last element of  $w_1$ ) to the last element of  $w_1$

# GSP: Apriori-Based Sequential Pattern Mining

- Initial candidates: All 8-singleton sequences
  - $\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle, \langle e \rangle, \langle f \rangle, \langle g \rangle, \langle h \rangle$
- Scan DB once, count support for each candidate
- Base case ( $k=2$ ): Generate candidate 2 -subsequences

*minsup = 2*

Cand.	sup
$\langle a \rangle$	3
$\langle b \rangle$	5
$\langle c \rangle$	4
$\langle d \rangle$	3
$\langle e \rangle$	3
$\langle f \rangle$	2
$\langle g \rangle$	1
$\langle h \rangle$	1

	$\langle a \rangle$	$\langle b \rangle$	$\langle c \rangle$	$\langle d \rangle$	$\langle e \rangle$	$\langle f \rangle$
$\langle a \rangle$	$\langle \{a\}\{a\} \rangle$	$\langle ab \rangle$	$\langle ac \rangle$	$\langle ad \rangle$	$\langle ae \rangle$	$\langle af \rangle$
$\langle b \rangle$	$\langle ba \rangle$	$\langle bb \rangle$	$\langle bc \rangle$	$\langle bd \rangle$	$\langle be \rangle$	$\langle bf \rangle$
$\langle c \rangle$	$\langle ca \rangle$	$\langle cb \rangle$	$\langle cc \rangle$	$\langle cd \rangle$	$\langle ce \rangle$	$\langle cf \rangle$
$\langle d \rangle$	$\langle da \rangle$	$\langle db \rangle$	$\langle dc \rangle$	$\langle dd \rangle$	$\langle de \rangle$	$\langle df \rangle$
$\langle e \rangle$	$\langle ea \rangle$	$\langle eb \rangle$	$\langle ec \rangle$	$\langle ed \rangle$	$\langle ee \rangle$	$\langle ef \rangle$
$\langle f \rangle$	$\langle fa \rangle$	$\langle fb \rangle$	$\langle fc \rangle$	$\langle fd \rangle$	$\langle fe \rangle$	$\langle ff \rangle$

Due to the space limit, we do not show {} in the above table

	$\langle a \rangle$	$\langle b \rangle$	$\langle c \rangle$	$\langle d \rangle$	$\langle e \rangle$	$\langle f \rangle$
$\langle a \rangle$		$\langle (ab) \rangle$	$\langle (ac) \rangle$	$\langle (ad) \rangle$	$\langle (ae) \rangle$	$\langle (af) \rangle$
$\langle b \rangle$			$\langle (bc) \rangle$	$\langle (bd) \rangle$	$\langle (be) \rangle$	$\langle (bf) \rangle$
$\langle c \rangle$				$\langle (cd) \rangle$	$\langle (ce) \rangle$	$\langle (cf) \rangle$
$\langle d \rangle$					$\langle (de) \rangle$	$\langle (df) \rangle$
$\langle e \rangle$						$\langle (ef) \rangle$
$\langle f \rangle$						

SID	Sequence
10	$\langle (bd)cb(ac) \rangle$
20	$\langle (bf)(ce)b(fg) \rangle$
30	$\langle (ah)(bf)abf \rangle$
40	$\langle (be)(ce)d \rangle$
50	$\langle a(bd)bcb(ade) \rangle$

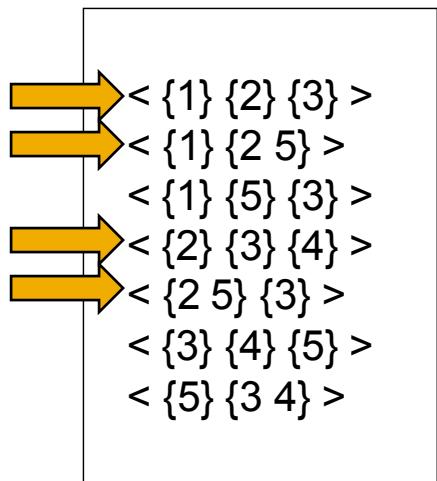
- Without Apriori pruning: (8 singletons)  $8*8+8*7/2 = 92$  candidates
- With pruning, candidates:  $36 + 15 = 51$

# Candidate Generation Examples

- Merging  $w_1 = \langle \{1\} \{2\} \{3\} \{4\} \{6\} \rangle$  and  $w_2 = \langle \{2\} \{3\} \{4\} \{6\} \{5\} \rangle$  produces the **candidate sequence**  $\langle \{1\} \{2\} \{3\} \{4\} \{6\} \{5\} \rangle$  because the last element of  $w_2$  has only one event
- Merging  $w_1 = \langle \{1\} \{2\} \{3\} \{4\} \rangle$  and  $w_2 = \langle \{2\} \{3\} \{4\} \{5\} \rangle$  produces the **candidate sequence**  $\langle \{1\} \{2\} \{3\} \{4\} \{5\} \rangle$  because the last element in  $w_2$  has more than one event
- Merging  $w_1 = \langle \{1\} \{2\} \{3\} \rangle$  and  $w_2 = \langle \{2\} \{3\} \{4\} \rangle$  produces the **candidate sequence**  $\langle \{1\} \{2\} \{3\} \{4\} \rangle$  because the last element in  $w_2$  has more than one event
- We do **not** merge the sequences  $w_1 = \langle \{1\} \{2\} \{6\} \{4\} \rangle$  and  $w_2 = \langle \{1\} \{2\} \{4\} \{5\} \rangle$  to produce the candidate  $\langle \{1\} \{2\} \{6\} \{4\} \{5\} \rangle$  because if the latter is a viable candidate, then it can be obtained by merging  $w_1$  with  $\langle \{2\} \{6\} \{4\} \{5\} \rangle$

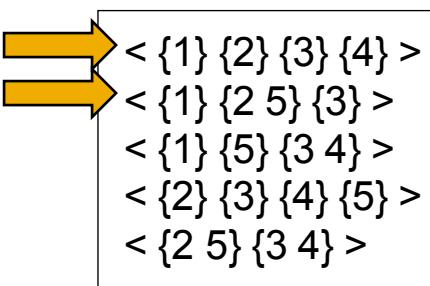
# GSP Example

Frequent  
3-sequences



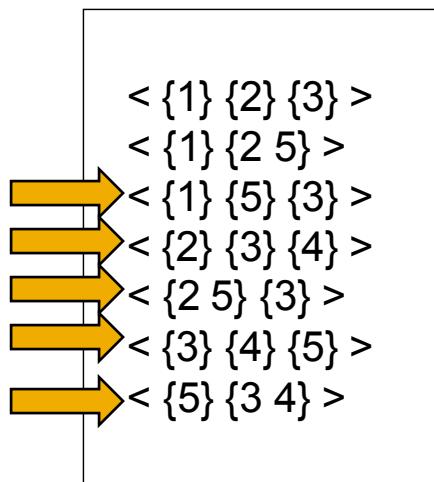
A large red arrow points from the box above to the text below.

Candidate  
Generation

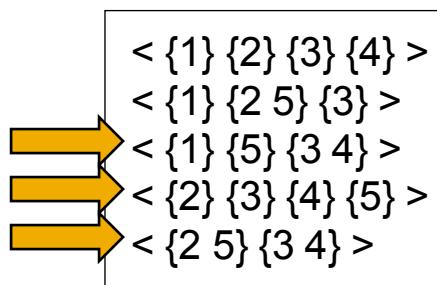


# GSP Example

Frequent  
3-sequences



Candidate  
Generation



Candidate  
Pruning

< {1} {2 5} {3} >

# Constraints on sequence patterns

- We can impose different constraints for two items in a sequence pattern. For example
  - a max-gap for two items
  - A min-gap for two items (e.g., contiguous)
- Need to check these constraints when deciding if a pattern is contained by a data sequence
- One solution: Mine sequential patterns without constraints
  - Postprocess the discovered patterns

# Summary

- FP-tree algorithms
  - Understand the algorithm
- Sequential pattern mining
  - Understand the algorithm

# Classification (Part 1)

Some slides adapted from Stanford data mining course,  
“Introduction to Data Mining “ by Kumar etc, and UIC data  
mining course

# Supervised Learning

## ■ Data is labeled:

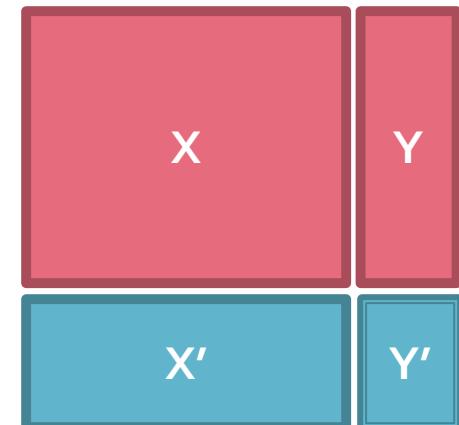
- Have many pairs  $\{(x, y)\}$ 
  - $x$  ... vector of binary, categorical, real valued features
  - $y$  ... class ( $\{+1, -1\}$ , or a real number)

## ■ Where $y$ can be:

- **Real number:** Regression
- **Categorical:** Classification
- Complex object:
  - Ranking of items, etc.

## ■ Task of prediction:

estimate a function  $f(x)$  so that  $y = f(x)$



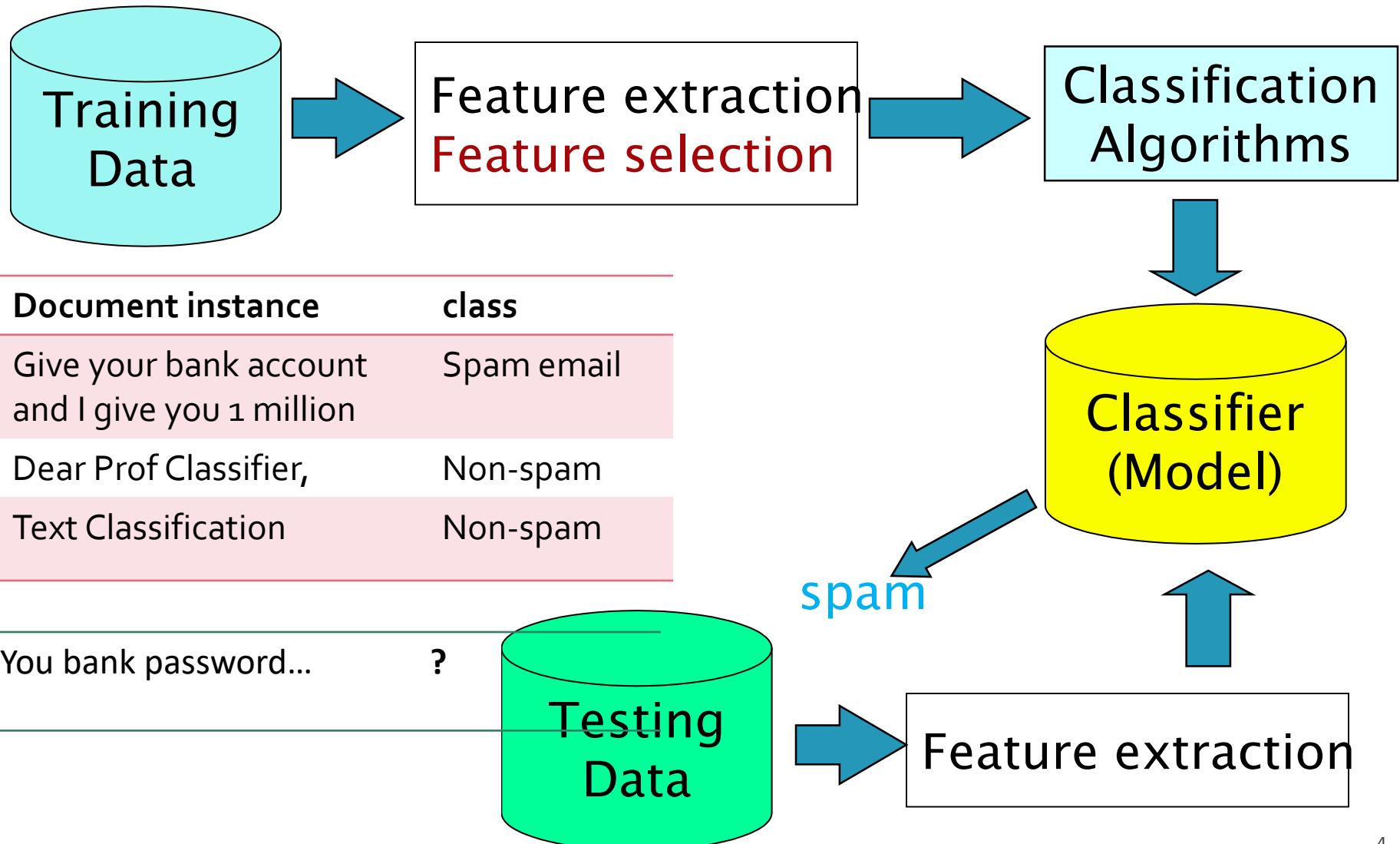
Training and test set

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

# Examples of Classification Task

Task	Attribute set, $x$	Class label, $y$
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from x-rays or MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

# Classification Framework



# Classification Methods

- Decision Tree based Methods
- Rule-based Methods
- Memory-based or Instance-based Methods
- Naïve Bayes Classifiers
- Support Vector Machines
- Neural Networks , Deep Neural Nets  
(<https://www.deeplearningbook.org/> for your reference)
- Ensemble Classification

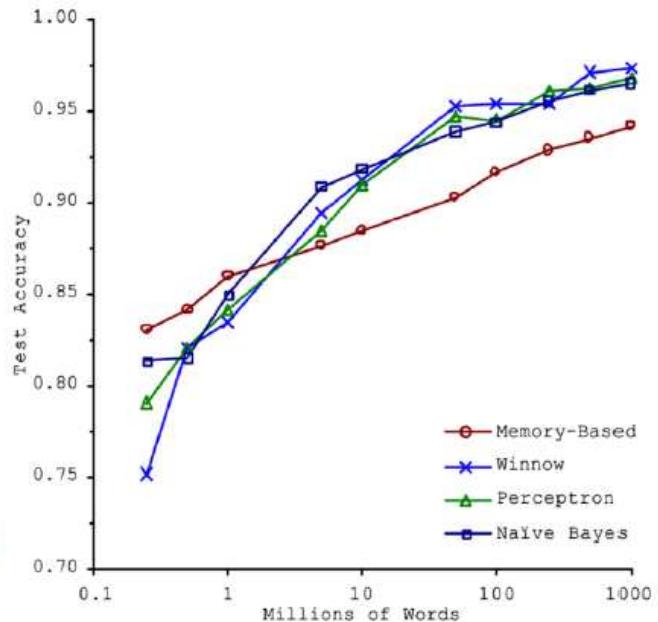
# Why large scale ML?

## ■ Brawn or Brains?

- In 2001, Microsoft researchers ran a test to evaluate 4 of different approaches to ML-based language translation

## ■ Findings:

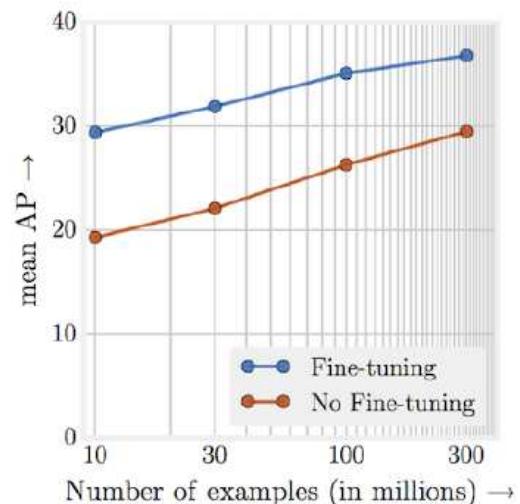
- **Size of the dataset** used to train the model **mattered more** than the model itself
- As the dataset grew large, performance difference between the models became small



Banko, M. and Brill, E. (2001), "[Scaling to Very Very Large Corpora for Natural Language Disambiguation](#)"

# Why large scale ML?

- **The Unreasonable Effectiveness of Data**
  - In 2017, Google revisited the same type of experiment with the latest Deep Learning models in computer vision
- **Findings:**
  - Performance increases logarithmically based on volume of training data
  - Complexity of modern ML models (i.e., deep neural nets) allows for even further performance gains
- Large datasets + large ML models => amazing results!!



"Revisiting Unreasonable Effectiveness of Data in Deep Learning Era": <https://arxiv.org/abs/1707.02968>

# Why Worry About Non-Deep Models?

## A few reasons why this is important:

- Classical tasks in NLP and Vision are getting commoditized (you take pretrained model and fine tune it), but there are many other unique ML tasks.
- Deep models are often hard to scale and require lots and lots of data. Traditional models allow you to encode prior knowledge better and give you more control.
- Personally, if I am working on a well understood problem I'd use deep learning, but if I am the first person to work on a new problem/classifier I'd use techniques we'll discuss here.

# Outline

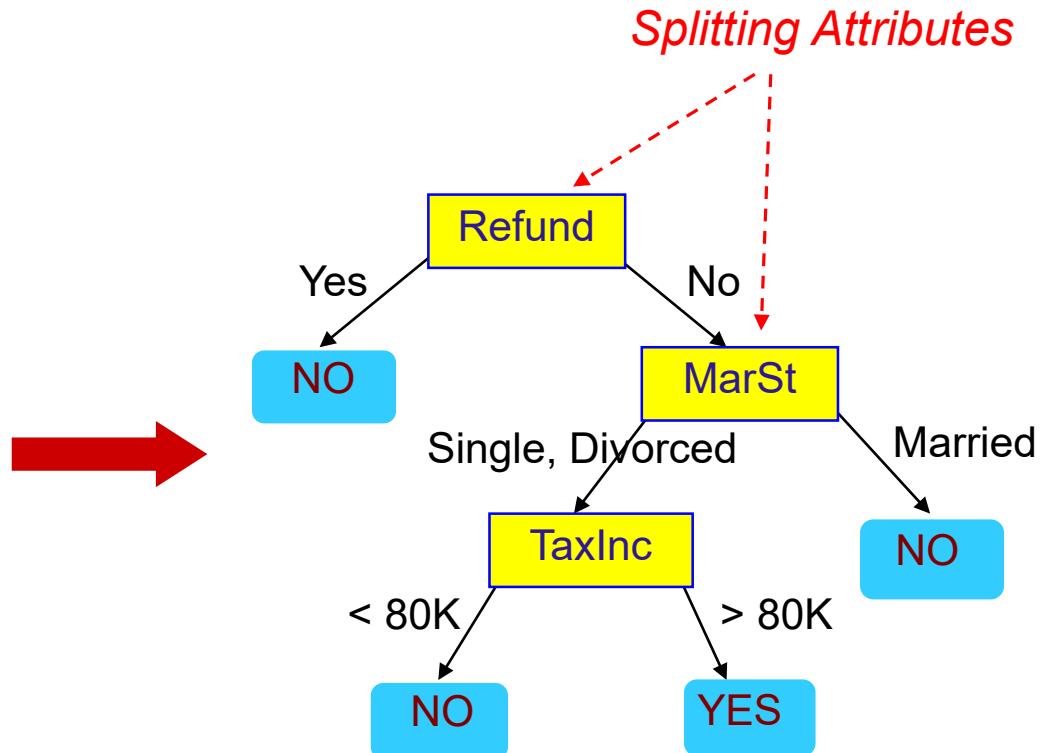
- Classification Techniques
  - Decision Tree
  - Classification based on association rules (CBA)
  - Ensemble classifier
  - Overfitting
  - Classification evaluation

# Example of a Decision Tree

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

Training Data

categorical  
categorical  
continuous  
class

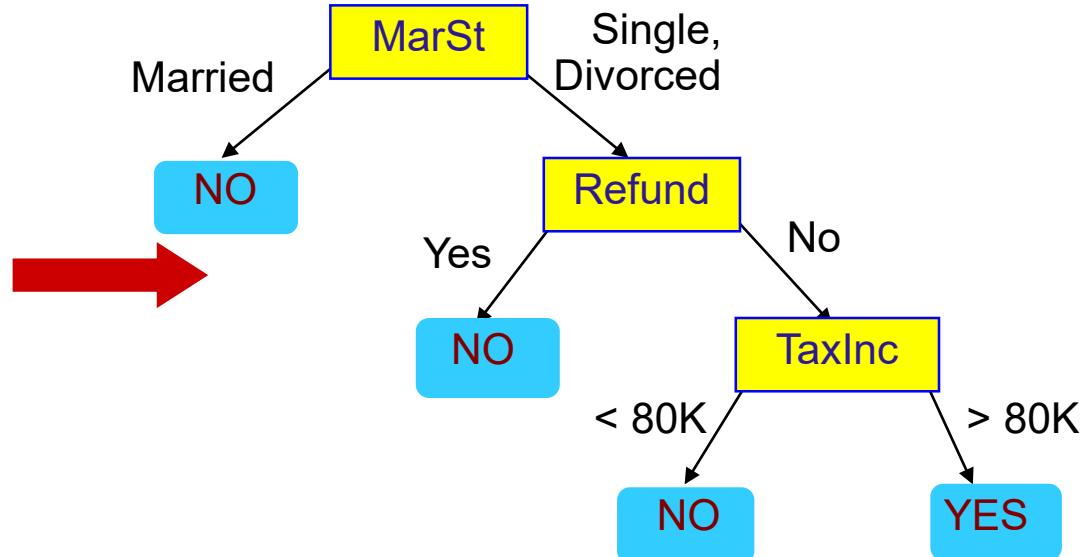


Model: Decision Tree

# Another Example of Decision Tree

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

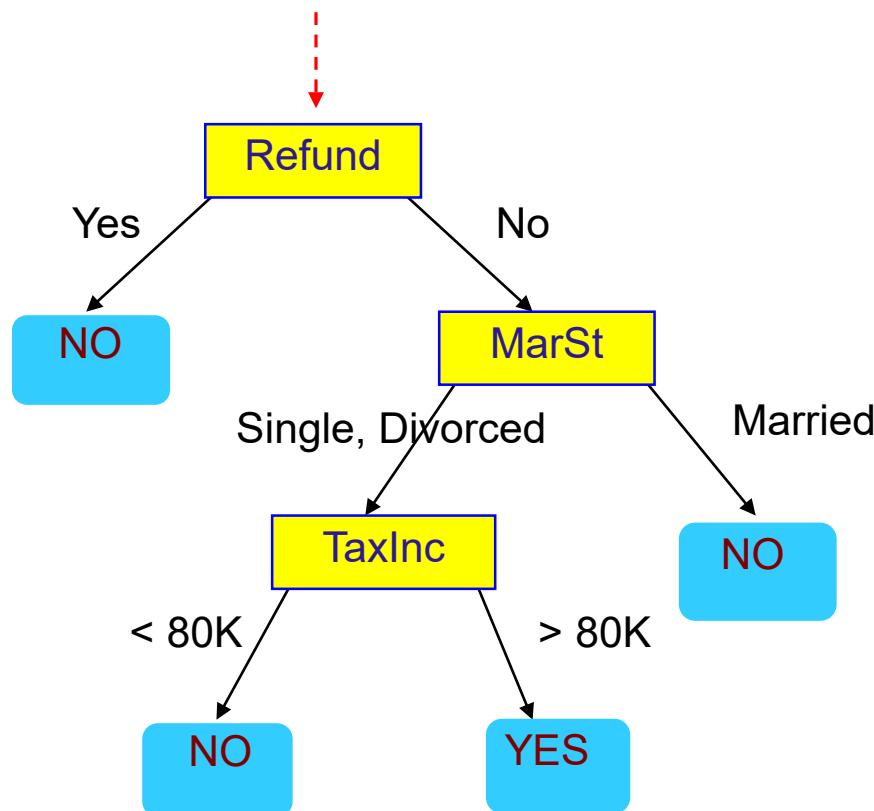
Training Data



There could be more than one tree that fits the same data!

# Apply Model to Test Data

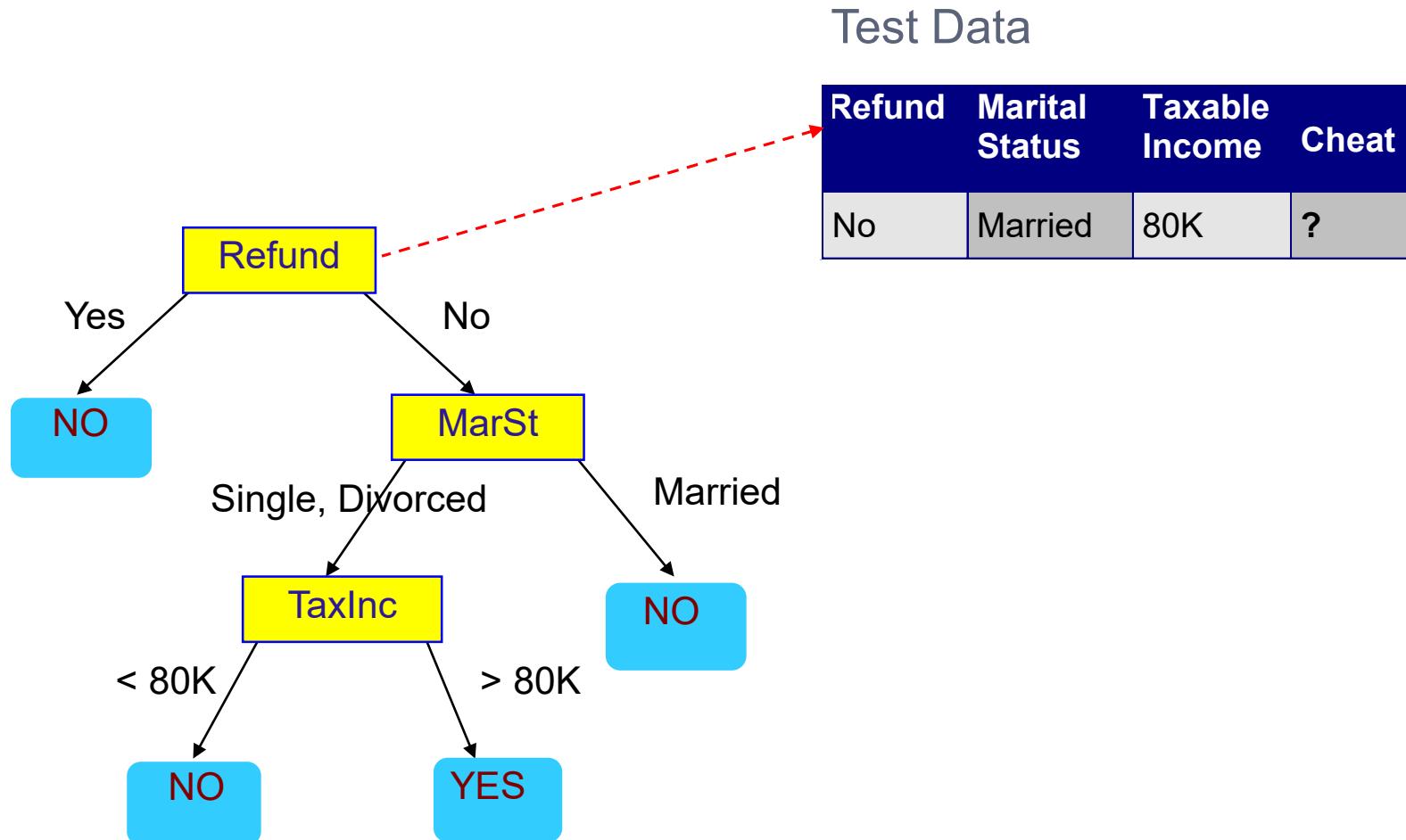
Start from the root of tree.



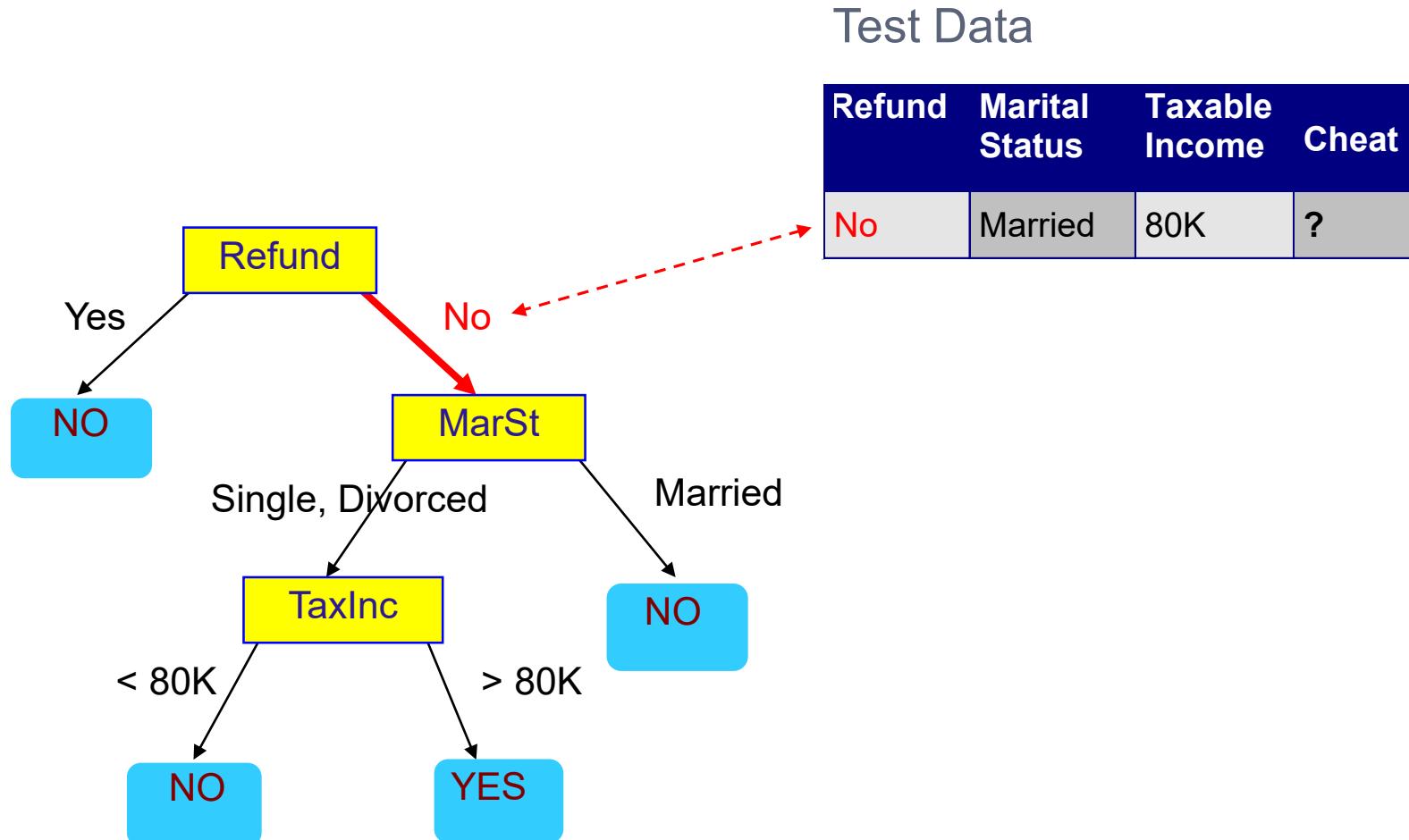
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

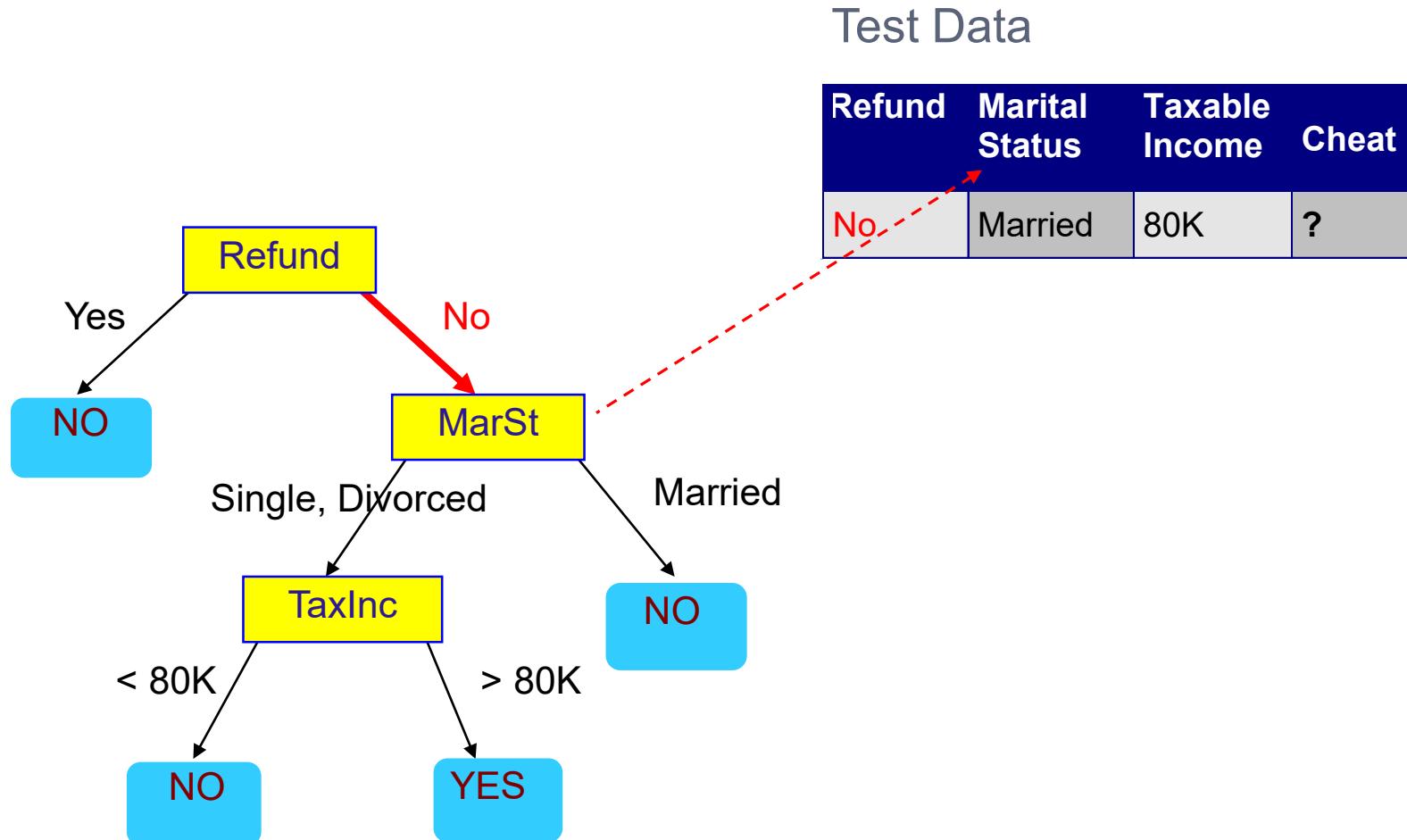
# Apply Model to Test Data



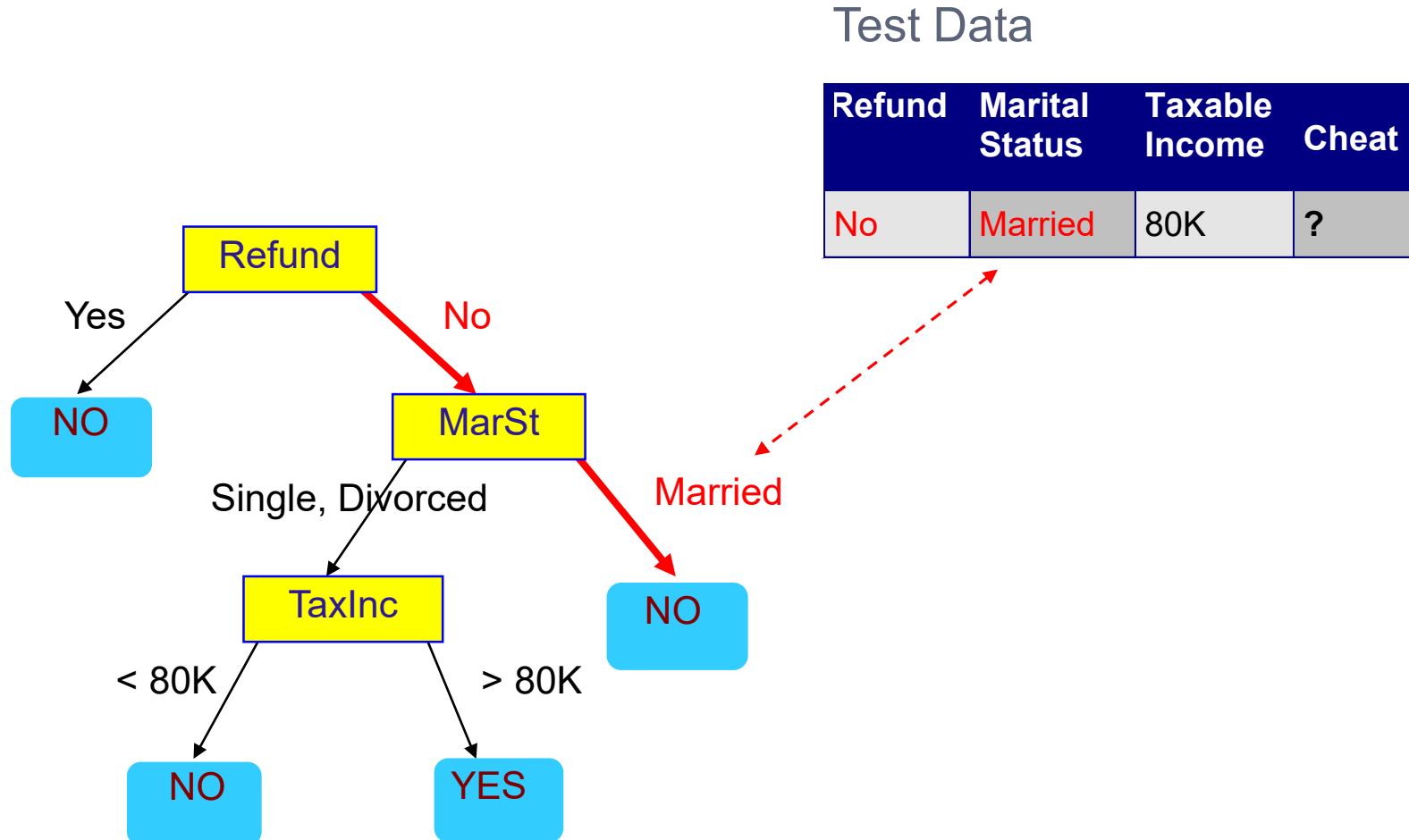
# Apply Model to Test Data



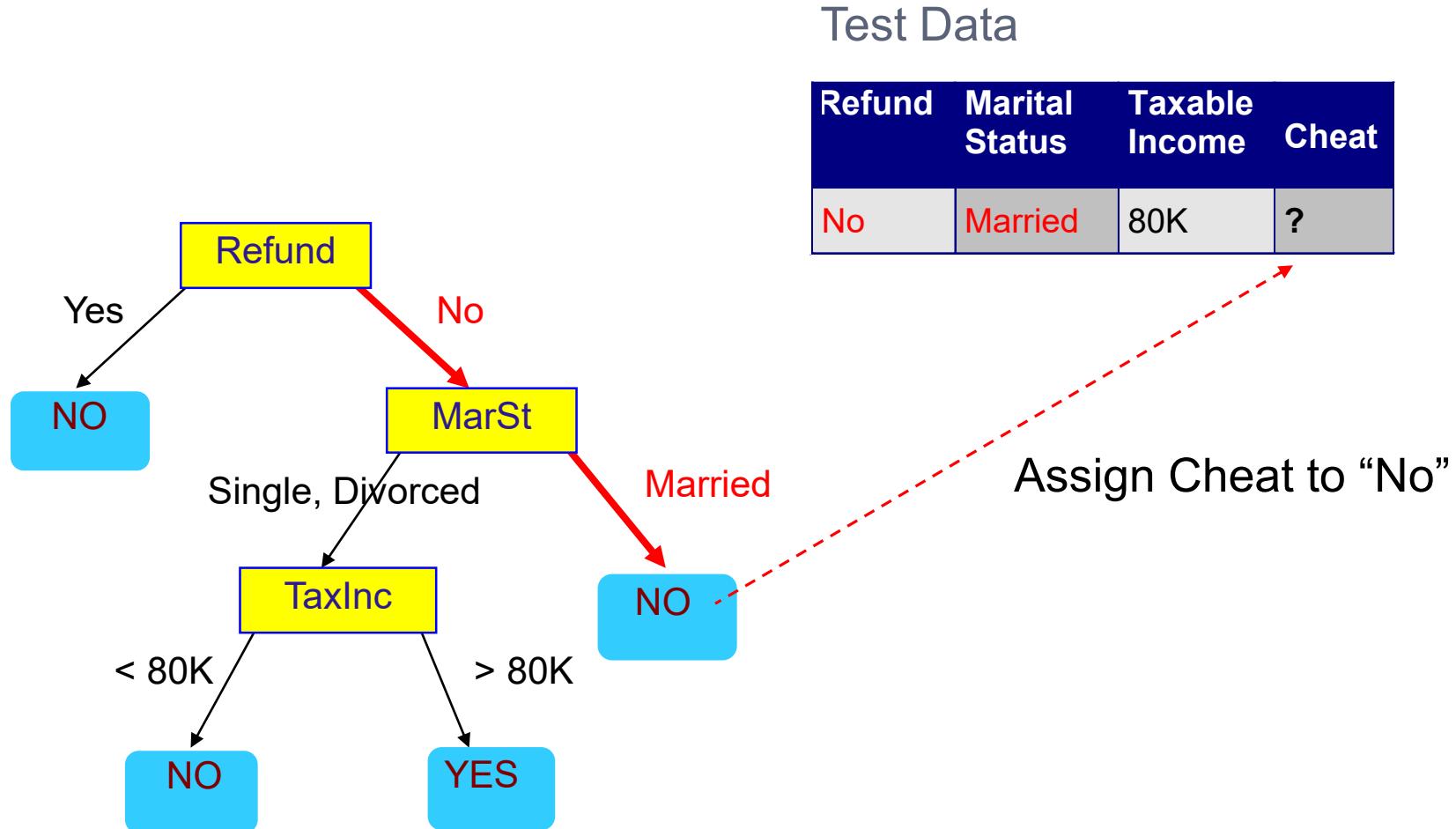
# Apply Model to Test Data



# Apply Model to Test Data



# Apply Model to Test Data



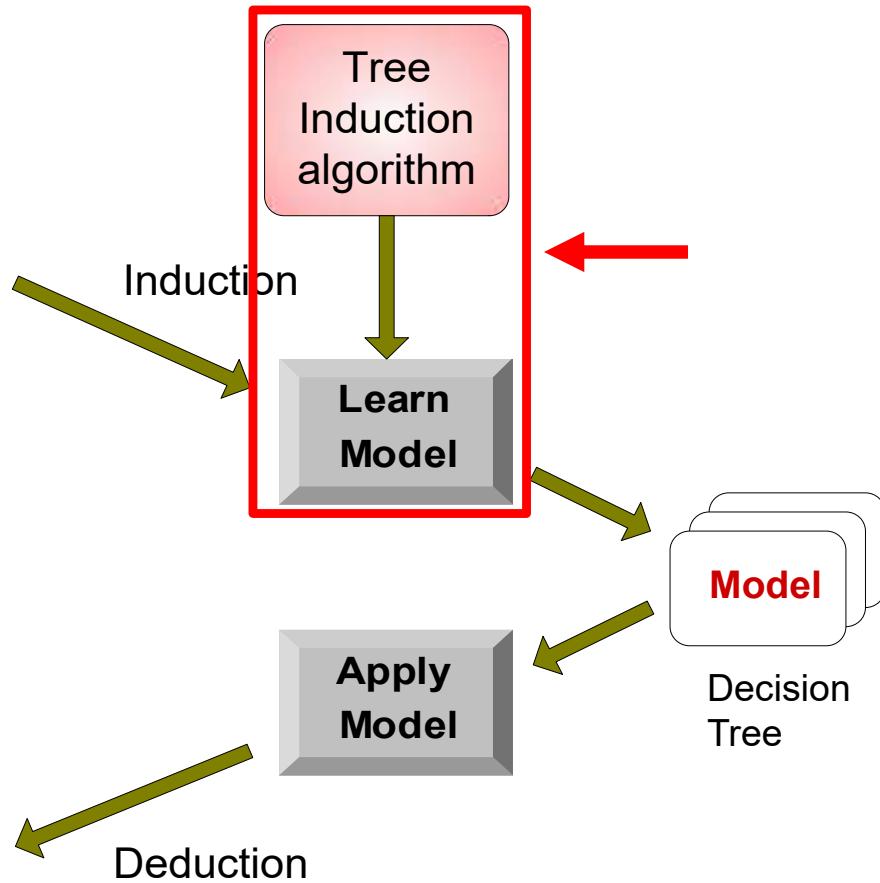
# Decision Tree Classification Task

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



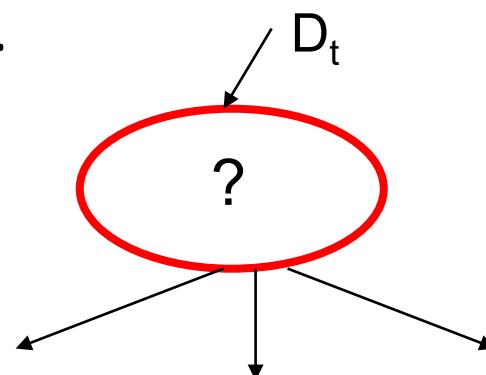
# Decision Tree Induction

- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT

# General Structure of Hunt's Algorithm

- Let  $D_t$  be the set of training records that reach a node  $t$
- General Procedure:
  - If  $D_t$  contains records that belong to the same class  $y_t$ , then  $t$  is a **leaf** node labeled as  $y_t$
  - If  $D_t$  contains records that belong to more than one class, use an **attribute test** to **split** the data into smaller subsets. Recursively apply the above procedure to each subset.

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



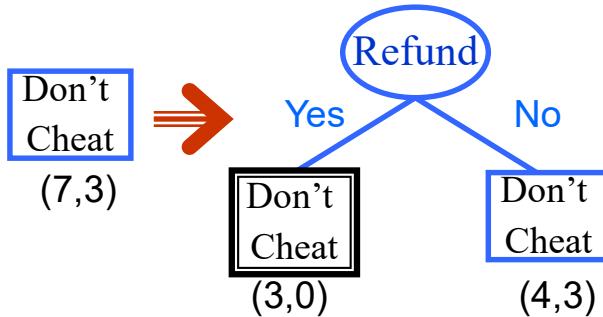
# Hunt's Algorithm

Don't  
Cheat

(7,3)

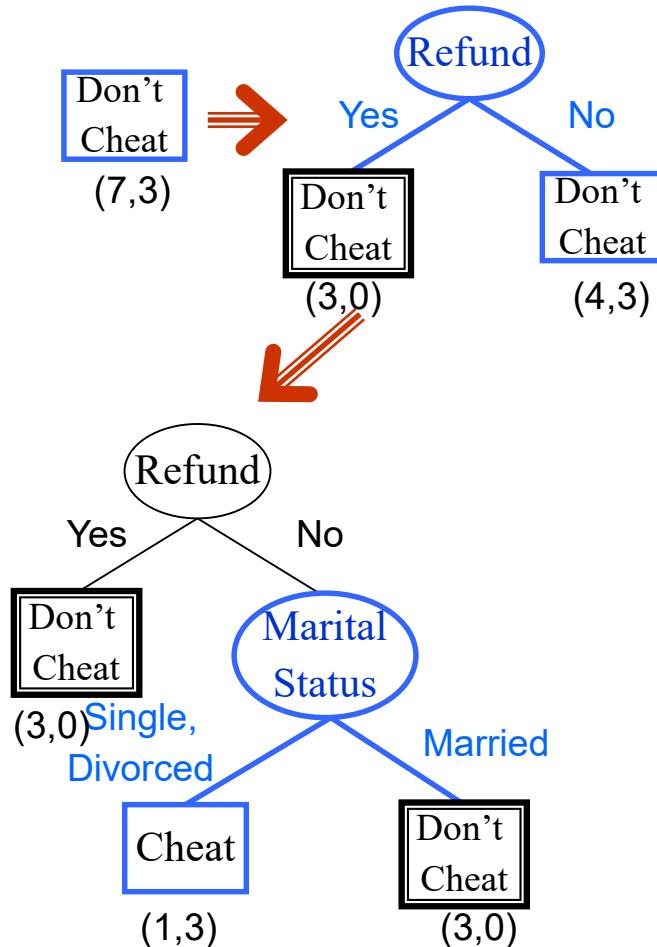
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Hunt's Algorithm



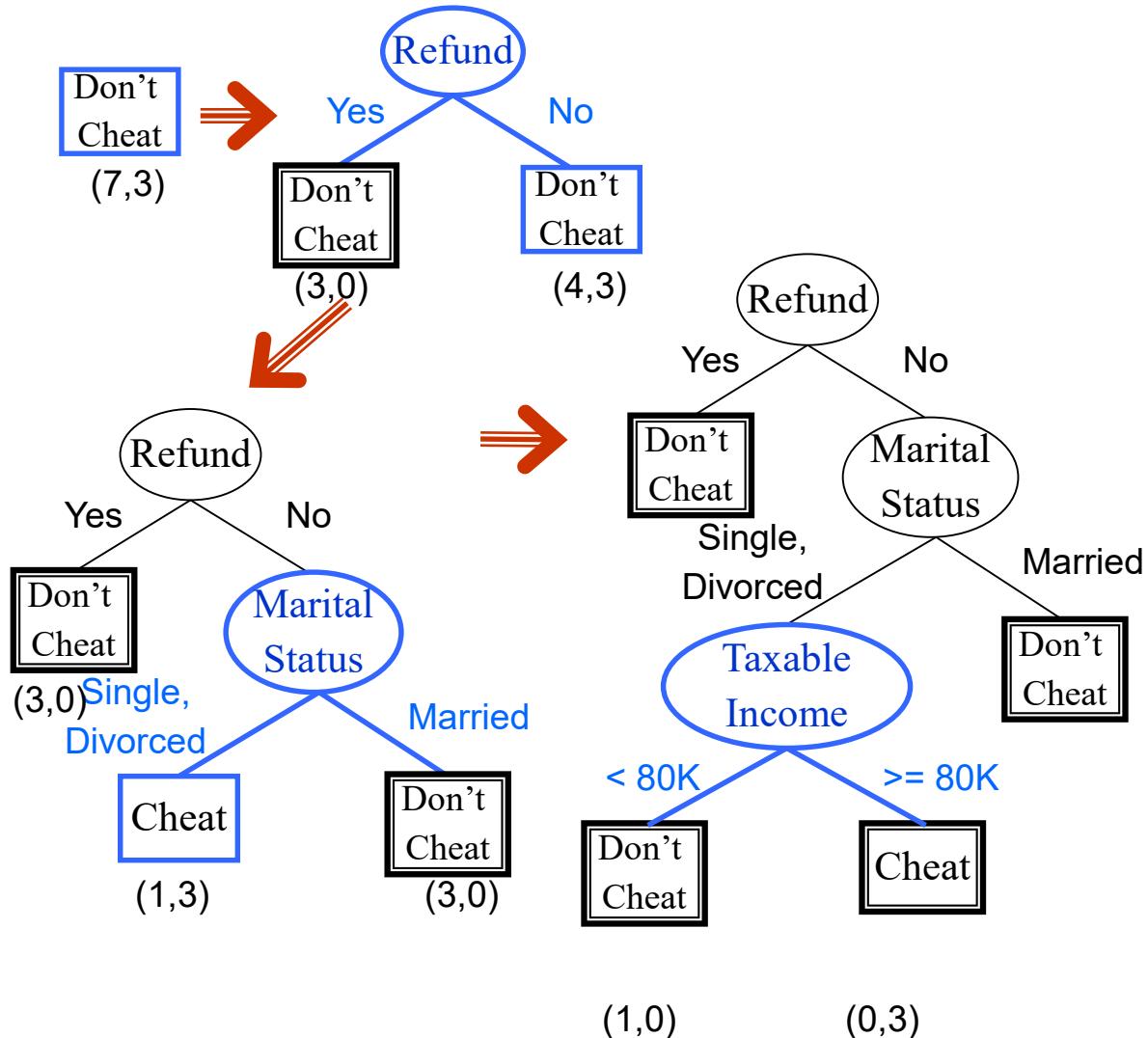
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Hunt's Algorithm



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

# Hunt's Algorithm



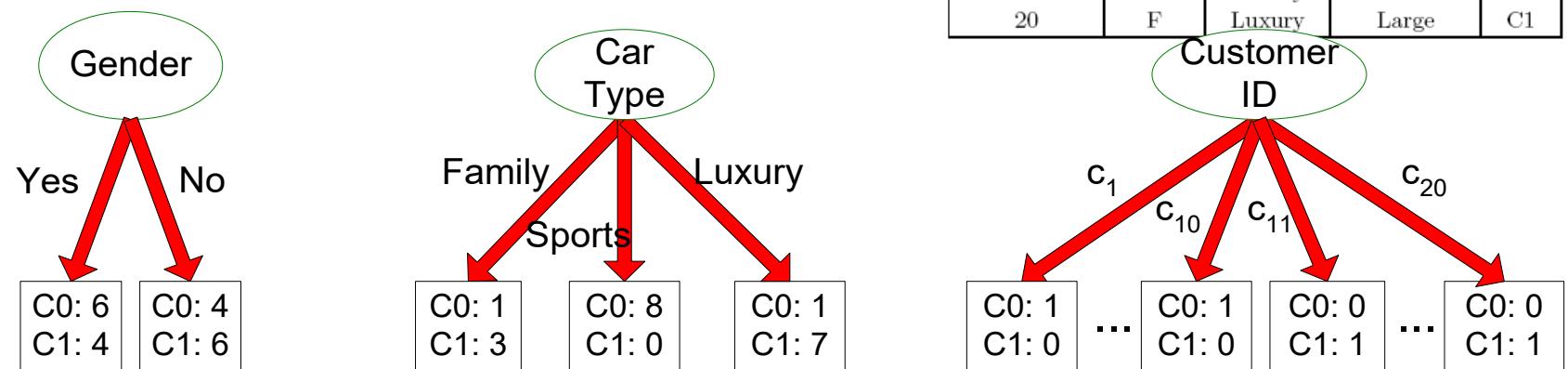
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Design Issues of Decision Tree Induction

- The idea is to use greedy strategy.
  - Split the records based on **an attribute test** that optimizes certain criterion.
- Issues
  - Determine how to **split** the records
    - Which attribute to split and how to determine the best split?
  - Determine when to **stop splitting**

# How to determine the Best Split

Before Splitting: 10 records of class 0,  
10 records of class 1



Which test condition is the best?

# How to determine the Best Split

- Greedy approach:
  - Nodes with **homogeneous** ( purer) class distribution are preferred
- Need a measure of node impurity:

C0: 5
C1: 5

Non-homogeneous,  
High degree of impurity

C0: 9
C1: 1

Homogeneous,  
Low degree of impurity

# Finding the Best Split

1. Compute impurity measure ( $P$ ) before splitting
2. Compute impurity measure ( $M$ ) after splitting
  - Compute impurity measure of each child node
  - $M$  is the weighted impurity of child nodes
3. Choose the attribute and the split that produces the highest gain. *Note an attribute may have multiple ways to split*

$$\text{Gain} = P - M$$

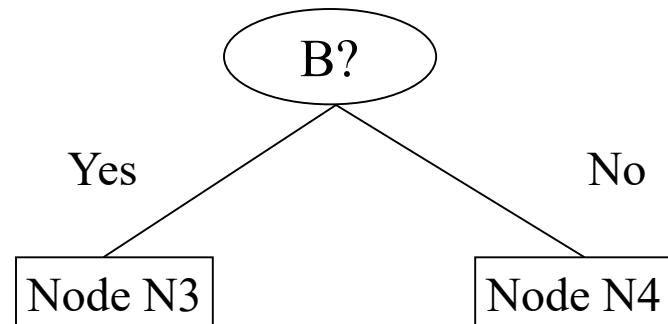
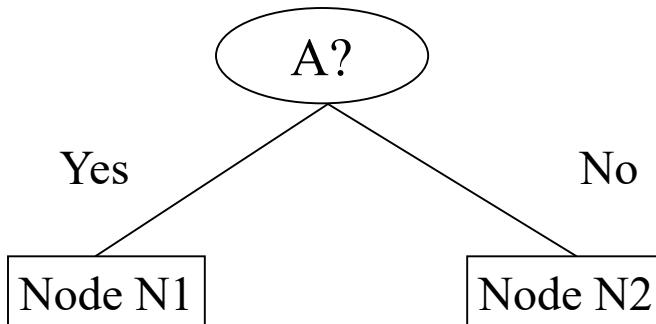
or equivalently, lowest impurity measure after splitting ( $M$ )

# Finding the Best Split

Before Splitting:

C0	<b>N00</b>
C1	<b>N01</b>

→ P



C0	<b>N10</b>
C1	<b>N11</b>

C0	<b>N20</b>
C1	<b>N21</b>

C0	<b>N30</b>
C1	<b>N31</b>

C0	<b>N40</b>
C1	<b>N41</b>

M11

M12

M21

M22

M1

M2

$$\text{Gain} = P - M1 \quad \text{vs} \quad P - M2$$

# Measures of Node Impurity

- **Gini Index**

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where  $p_i(t)$  is the probability of class  $i$  at node  $t$ , and  $c$  is the total number of classes

- **Entropy**

$$Entropy = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

- **Misclassification error**

$$Classification\ error = 1 - \max[p_i(t)]$$

# Computing GINI Measure

- Gini Index for a given node  $t$

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where  $p_i(t)$  is the probability of class  $i$  at node  $t$ , and  $c$  is the total number of classes

- Maximum of  $1 - 1/c$  when records are equally distributed among all classes, implying the least beneficial situation for classification
- Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification

C1	<b>0</b>
C2	<b>6</b>
<b>Gini=0.000</b>	

C1	<b>1</b>
C2	<b>5</b>
<b>Gini=0.278</b>	

C1	<b>2</b>
C2	<b>4</b>
<b>Gini=0.444</b>	

C1	<b>3</b>
C2	<b>3</b>
<b>Gini=0.500</b>	

# Computing Gini Index of a Single Node

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

For 2-class problem

C1	<b>0</b>
C2	<b>6</b>

$$\begin{aligned} P_{(C1)}(t) &= 0/6 = 0 & P_{(C2)}(t) &= 6/6 = 1 \\ \text{Gini} &= 1 - P_{(C1)}(t)^2 - P_{(C2)}(t)^2 = 1 - 0 - 1 = 0 \end{aligned}$$

C1	<b>1</b>
C2	<b>5</b>

$$\begin{aligned} P_{(C1)}(t) &= 1/6 & P_{(C2)}(t) &= 5/6 \\ \text{Gini} &= 1 - (1/6)^2 - (5/6)^2 = 0.278 \end{aligned}$$

C1	<b>2</b>
C2	<b>4</b>

$$\begin{aligned} P_{(C1)}(t) &= 2/6 & P_{(C2)}(t) &= 4/6 \\ \text{Gini} &= 1 - (2/6)^2 - (4/6)^2 = 0.444 \end{aligned}$$

# Computing Gini Index for a Collection of Nodes

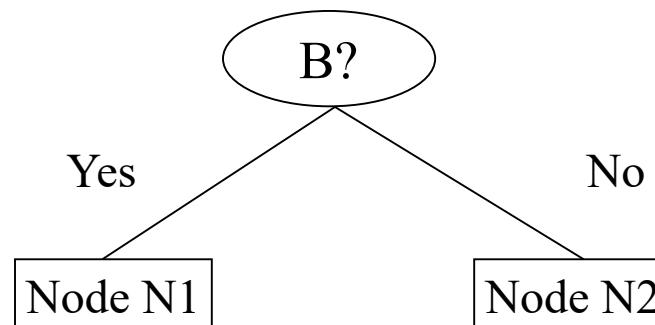
- In general, when a node  $p$  is split into  $k$  partitions (children)

$$Gini_{split} = \sum_{i=1}^k \frac{n_i}{n} Gini(i)$$

where,  $n_i$  = number of records at child  $i$ ,  
 $n$  = number of records at parent node  $p$ .

# Binary Attributes: Computing GINI Index

- **Binary Attributes:** Splits into two partitions (child nodes)
- Effect of Weighing partitions:
  - Larger and purer partitions are sought



$Gini(N_1)$

$$\begin{aligned} &= 1 - (5/6)^2 - (1/6)^2 \\ &= 0.278 \end{aligned}$$

$Gini(N_2)$

$$\begin{aligned} &= 1 - (2/6)^2 - (4/6)^2 \\ &= 0.444 \end{aligned}$$

	<b>N1</b>	<b>N2</b>
C1	<b>5</b>	<b>2</b>
C2	<b>1</b>	<b>4</b>
<b>Gini=0.361</b>		

	<b>Parent</b>
C1	<b>7</b>
C2	<b>5</b>
<b>Gini = 0.486</b>	

Weighted Gini of N1 N2

$$\begin{aligned} &= 6/12 * 0.278 + \\ &\quad 6/12 * 0.444 \\ &= 0.361 \end{aligned}$$

$$\text{Gain} = 0.486 - 0.361 = 0.125$$

# Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

CarType			
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	<b>0.163</b>		

Two-way split

(find best partition of values)

CarType		
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	<b>0.468</b>	

CarType		
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	<b>0.167</b>	

Which of these is the best?

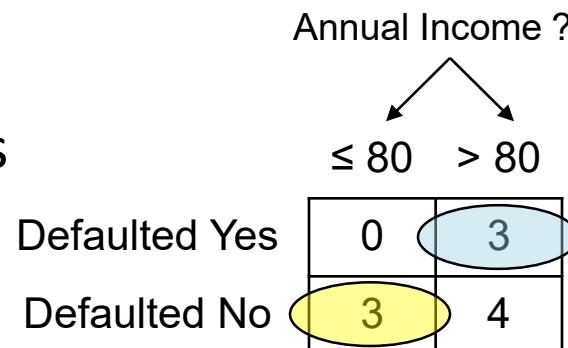
# Continuous attributes: Computing Gini Index

- Two ways of handling continuous attributes
  - Discretization to form an (ordinal) categorical attribute, then use the method in last slide
    - Discretization will be introduced in later week
  - Binary Decision:  $(A < v)$  or  $(A \geq v)$ 
    - consider all possible splits and finds the best cut
    - can be more computationally intensive

# Continuous Attributes: Computing Gini Index

- Use **Binary Decisions** based on one value
- Choices for the splitting value
  - Each distinct value is a possible splitting value
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions,  $A \leq v$  and  $A > v$
- Simple method to choose best  $v$ 
  - For each  $v$ , scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient!  
Repetition of work.

ID	Home Owner	Marital Status	Annual Income	Defaulted
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



# Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No	
Annual Income											
→	60	70	75	85	90	95	100	120	125	220	
→	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	

# Design Issues of Decision Tree Induction

- The idea is to use greedy strategy.
  - Split the records based on **an attribute test** that optimizes certain criterion.
- Issues
  - Determine how to **split** the records
    - Which attribute to split and how to determine the best split?
  - Determine when to **stop splitting**

# Stopping Criteria for Tree Induction

- Many different heuristic options
- Three methods:
  - Stop expanding a node when all the records in the leaf belong to the same class, or mostly belong to one class
  - When number of examples in the leaf is too small. For example, < 100
  - Stop expanding a node when all the records have similar attribute values

# Measure of Impurity: Entropy (for reference & self-study)

- Entropy at a given node  $t$

$$\text{Entropy} = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

Where  $p_i(t)$  is the probability of class  $i$  at node  $t$ , and  $c$  is the total number of classes

- ◆ Maximum of  $\log_2 c$  when records are equally distributed among all classes, implying the least beneficial situation for classification
  - ◆ Minimum of 0 when all records belong to one class, implying most beneficial situation for classification
- 
- Entropy based computations are quite similar to the GINI index computations

# Computing Entropy of a Single Node (for reference & self-study)

$$\text{Entropy} = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

C1	<b>0</b>
C2	<b>6</b>

$$P_{(C1)}(t) = 0/6 = 0 \quad P_{(C2)}(t) = 6/6 = 1$$
$$\text{Entropy} = -0 \log 0 - 1 \log 1 = -0 - 0 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P_{(C1)}(t) = 1/6 \quad P_{(C2)}(t) = 5/6$$
$$\text{Entropy} = -(1/6) \log_2 (1/6) - (5/6) \log_2 (1/6) = 0.65$$

C1	<b>2</b>
C2	<b>4</b>

$$P_{(C1)}(t) = 2/6 \quad P_{(C2)}(t) = 4/6$$
$$\text{Entropy} = -(2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$$

# Computing Information Gain After Splitting(for reference & self-study)

- Information Gain:

$$Gain_{split} = Entropy(p) - \sum_{i=1}^k \frac{n_i}{n} Entropy(i)$$

Parent Node,  $p$  is split into  $k$  partitions (children)

$n_i$  is number of records in child node  $i$

- Choose the split that achieves most reduction (maximizes GAIN)
- Information gain is the mutual information between the class variable and the splitting variable

# Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets

# Outline

- Decision tree
- Classification based on association rules (CBA)
- Ensemble Classifiers
- Overfitting
- Classification evaluation

# Rule-Based Classifier

- Classify records by using a collection of “if...then...” rules
- Rule:  $(Condition) \rightarrow y$ 
  - where
    - $Condition$  is a conjunction of tests on attributes
    - $y$  is the class label
  - Examples of classification rules:
    - $(\text{Blood Type}=\text{Warm}) \wedge (\text{Lay Eggs}=\text{Yes}) \rightarrow \text{Birds}$
    - $(\text{Taxable Income} < 50K) \wedge (\text{Refund}=\text{Yes}) \rightarrow \text{Evade}=\text{No}$

# Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

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

# Application of Rule-Based Classifier

- A rule  $r$  **covers** an instance  $x$  if the attributes of the instance satisfy the condition of the rule

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
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

# Building Classification Rules

## ■ Direct Method:

- Extract rules directly from data
- Examples: RIPPER, CN2, Holte's 1R, CBA

## ■ Indirect Method:

- Extract rules from other classification models (e.g. decision trees, etc).

# Decision tree vs. rules

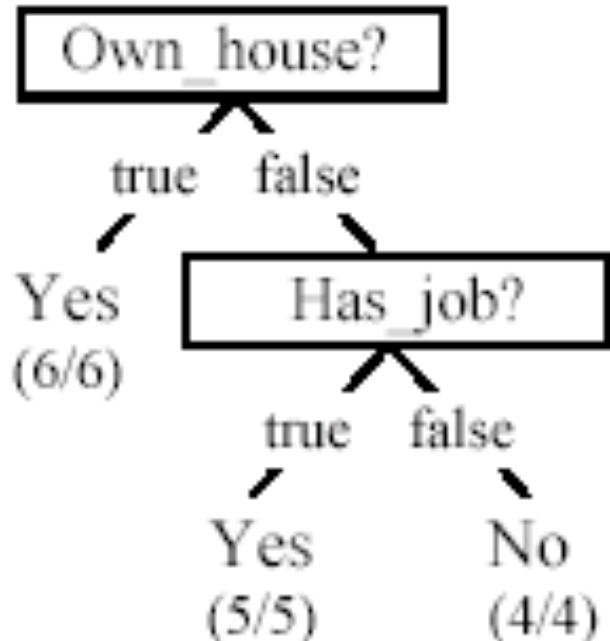
- The decision tree below generates the following 3 rules.

Own\_house = true → Class =Yes [sup=6/15, conf=6/6]

Own\_house = false, Has\_job = true → Class=Yes [sup=5/15, conf=5/5]

Own\_house = false, Has\_job = false → Class=No [sup=4/15, conf=4/4]

- But there are many other rules that are not found by the decision tree



# CBA: Class Association Rules

- **Association rules:** have no fixed target, but we can fix a target. target attribute: **Class attribute**
- **Class association rules (CAR):** has a target class attribute. E.g.,

Own\_house = true → Class =Yes [sup=6/15, conf=6/6]

- CARs can obviously be used for classification.

Note that we use ratio for support in this part

# Class Association Rules

Age = young, Has_job = true → Class=Yes	[sup=2/15, conf=2/2]
Age = young, Has_job = false → Class=No	[sup=3/15, conf=3/3]
Credit_Rating = fair → Class=No	[sup=4/15, conf=4/4]
Credit_Rating = good → Class=Yes	[sup=5/15, conf=5/6]

and many more, if we use minsup = 2/15 = 13.3% and minconf = 80%.

- CAR mining finds all of them.
- In many cases, rules not in the decision tree may perform classification better.

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	excellent	No
3	young	true	false	good	Yes
4	young	true	true	good	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

# Considerations in CAR mining

- Set different minimum class supports
  - Deal with imbalanced class distribution, e.g., some class is rare, 98% negative and 2% positive.
  - We can set the  $\text{minsup}(\text{positive}) = 0.2\%$  and  $\text{minsup}(\text{negative}) = 2\%$ .
- Rule pruning may be performed.
  - Pruning off less useful rules.
  - E.g., A rule  $a, b, c \rightarrow \text{class1}$  is pruned if its confidence is 60%, but a subset rule  $a, b \rightarrow \text{class1}$  has a confidence 70%.
    - $a, b \rightarrow \text{class1}$  must have a higher support than  $a, b, c \rightarrow \text{class1}$

# Example of pruning rules

## ■ Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

$r_1: <(\{(A, e), (B, p)\}, 3), ((C, y), 2)>$

Confidence = 2/3 = 67%

Error rate of = 33%

$r_1^-: <(\{A, e\}, 4), ((C, y), 3)>$

Confidence = 3/4 = 75%

Error rate of = 25%

$r_1^-$  is a subset of  $r_1$  and  
 $r_1^-$  has a higher confidence (lower Error rate) than  $r_1$

⇒

Prune  $r_1$

# Classification and Association Rule Mining

- There are many ways to build classifiers using CARs. Several existing systems available.
- Strongest rules: After CARs are mined, do nothing.
  - For each test case, we simply choose the most confident rule that covers the test case to classify it.
  - Or, using a combination of rules.
- Selecting a subset of Rules to build CBA
  - To be explained .
  - Reference: Bing Liu, Wynne Hsu, Yiming Ma:Integrating Classification and Association Rule Mining. KDD 1998: 80-86

# CBA Algorithm

A CBA classifier  $L$  is of the form:

$$L = \langle r_1, r_2, \dots, r_k, \text{default-class} \rangle$$

CBA algorithm consists of two parts:

1. Rule Generator (CBA-RG):

- Generate CARs for each class.
- A simple extension of Apriori algorithm. We have a tutorial question on it.

2. Classifier builder (CBA-CB): select a subset of rules

# CBA: Rules are sorted first

**Definition:** Given two rules,  $r_i$  and  $r_j$ ,  $r_i \succ r_j$  (also called  $r_i$  precedes  $r_j$  or  $r_i$  has a higher **precedence** than  $r_j$ ) if

- the confidence of  $r_i$  is greater than that of  $r_j$ , or
- their confidences are the same, but the support of  $r_i$  is greater than that of  $r_j$

# Example: CBA-CB (M1)

## ■ Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

## ■ CARs after pruning:

- $r_1: A = e \rightarrow y \quad sup = 30\% \quad conf = 75\%$
- $r_2: A = g \rightarrow n \quad sup = 30\% \quad conf = 60\%$
- $r_3: B = p \rightarrow y \quad sup = 20\% \quad conf = 66\%$
- $r_4: B = q \rightarrow y \quad sup = 30\% \quad conf = 60\%$
- $r_5: B = w \rightarrow n \quad sup = 20\% \quad conf = 100\%$
- $r_6: A = g, B = q \rightarrow y \quad sup = 20\% \quad conf = 66\%$

# Example: CBA-CB (M1)

## Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

## CARs after pruning:

$r_1: A = e \rightarrow y$	$sup = 30\%$	$conf = 75\%$
$r_2: A = g \rightarrow n$	$sup = 30\%$	$conf = 60\%$
$r_3: B = p \rightarrow y$	$sup = 20\%$	$conf = 66\%$
$r_4: B = q \rightarrow y$	$sup = 30\%$	$conf = 60\%$
$r_5: B = w \rightarrow n$	$sup = 20\%$	$conf = 100\%$
$r_6: A = g, B = q \rightarrow y$	$sup = 20\%$	$conf = 66\%$

## Sort CARs ( $>$ Precedence):

# of highest confidence rule

$$r_5 > r_1 > r_3 > r_6 > r_2 > r_4$$

# CBA-CB: M1 Algorithm

## CBA (R,D) Algorithm: R CARs, D training data

1.  $R = \text{sort}(R)$ ; //soring is done based on precedence definition
2.  $\text{RuleList} = \emptyset$
3. **for** each rule  $r$  in  $R$  in sequence **do**
4.     **If**  $D$  is not  $\emptyset$  AND  $r$  classifies at least one tuple in  $D$  correctly  
    **then**
  5.         delete all the training tuples covered by  $r$  from  $D$ ;
  6.         Add  $r$  at the end of  $\text{RuleList}$ ;
  7.         Add the majority class in the uncovered training data as  
           the default class at the end of  $\text{RuleList}$
  8.         compute the total number of errors of  $\text{RuleList}$ ;
9.     Find the first rule  $p$  in  $\text{RuleList}$  with the lowest total number of  
        errors and drop all the rules after  $p$  in  $\text{RuleList}$ ; add default class;

# Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ( $\succ$ Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8:

$$r_5: B = w \rightarrow n$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3

# Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ( $\succ$  Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8:

$$r_5: B = w \rightarrow n$$

$$r_1: A = e \rightarrow y$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3

# Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ( $\succ$ Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8 :

$$r_5: B = w \rightarrow n$$

$$r_1: A = e \rightarrow y$$

$$r_3: B = p \rightarrow y$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-

# Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ( $\succ$ Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8:

$$r_5: B = w \rightarrow n$$

$$r_1: A = e \rightarrow y$$

$$r_6: A = g, B = q \rightarrow y$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2

# Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ( $\succ$  Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8 :

$$r_5: B = w \rightarrow n$$

$$r_1: A = e \rightarrow y$$

$$r_6: A = g, B = q \rightarrow y$$

$$r_2: A = g \rightarrow n$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2
2	-	-	-	-	-

# Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ( $\succ$ Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8:

$$r_5: B = w \rightarrow n \quad r_6: A = g, B = q \rightarrow y$$

$$r_1: A = e \rightarrow y \quad r_4: B = q \rightarrow y$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2
2	-	-	-	-	-
4	-	-	-	-	-

# Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- C:

$r_5: B = w \rightarrow n$        $r_6: A = g, B = q \rightarrow y$

$r_1: A = e \rightarrow y$        ~~$r_4: B = q \rightarrow y$~~

- Algorithm Line 9:

Find the first rule  $p$  in *RuleList* with the lowest total number of errors ....

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2
2	-	-	-	-	-
4	-	-	-	-	-

# Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Algorithm Line 9: default class

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2

$$\Rightarrow C: \begin{cases} r_5: & B = w \rightarrow n \\ r_1: & A = e \rightarrow y \\ r_6: & A = g, B = q \rightarrow y \end{cases} \quad \text{Default Class} \quad \text{n}$$

# Additional information on CBA algorithm

- This algorithm is CBA-CB:M1 algorithm in the original paper, which is inefficient
- CBA in the original paper has a more efficient algorithm (quite sophisticated) that scans the data at most two times.

# Decision tree vs. CARs

- Association mining require discrete attributes.  
Decision tree learning uses both discrete and continuous attributes.
  - CAR mining requires continuous attributes discretized. There are several such algorithms (to be covered in this course).
- However, decision tree is
  - not constrained by minsup or minconf, and may find rules with very low support or confidence.

# Summary

- Classification Techniques
  - Decision Tree
    - Understand the algorithm
  - Rule-Based Classifier
    - Understand the algorithm
- Next:
  - Ensemble classifiers
  - Overfitting
  - Classification evaluation

# Classification (Part 2)

Some slides adapted from Stanford data mining course,  
“Introduction to Data Mining “ by Kumar etc, and UIC data  
mining course

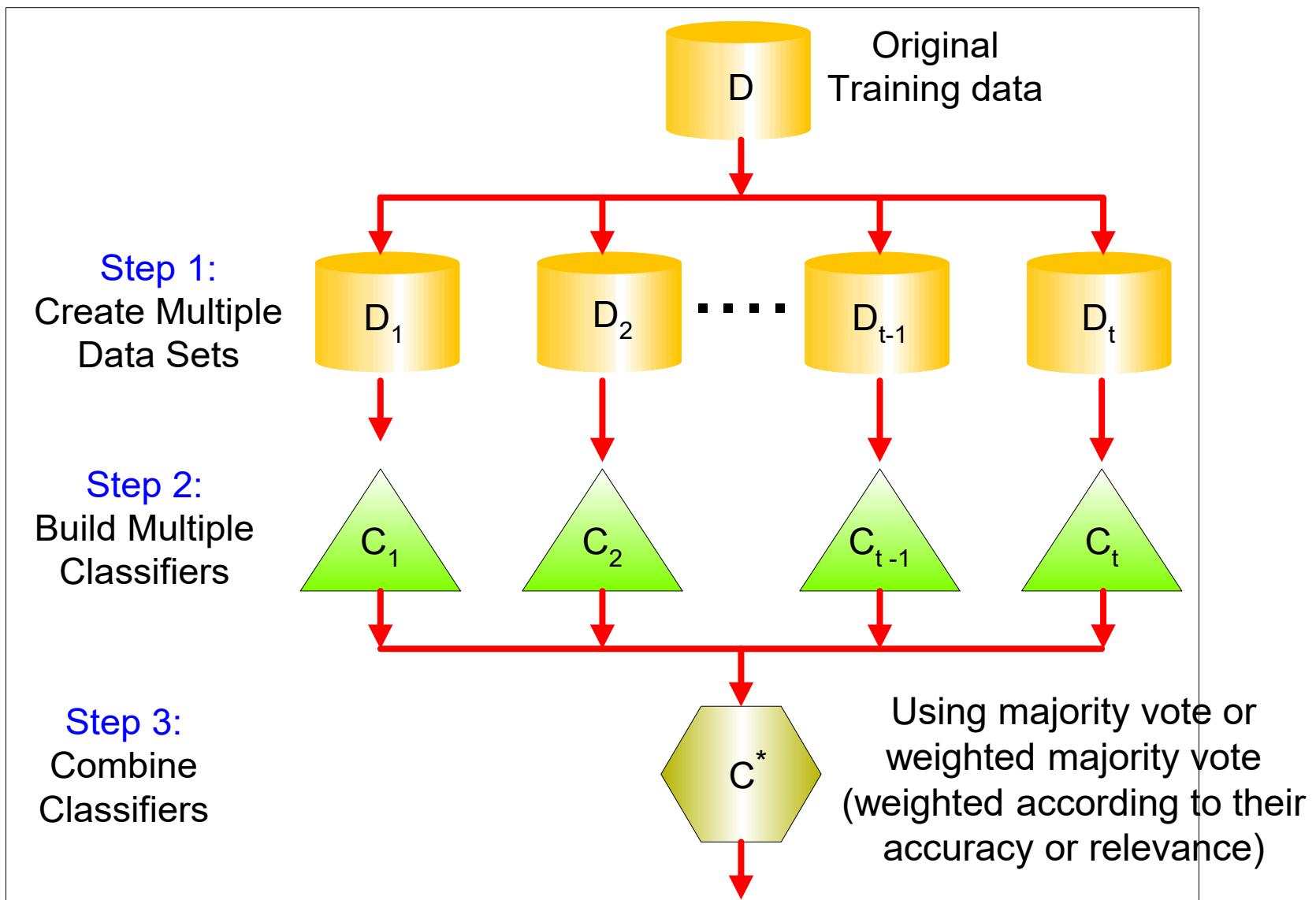
# Outline

- Classification Techniques
  - Decision Tree
  - Classification based on association rules
  - Ensemble classifier
  - Overfitting
  - Classification evaluation

# Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers, such as majority of voting
- Assumption:
  - Individual classifiers (voters) could be lousy (stupid), but the aggregate (electorate) can usually classify (decide) correctly.

# General Idea



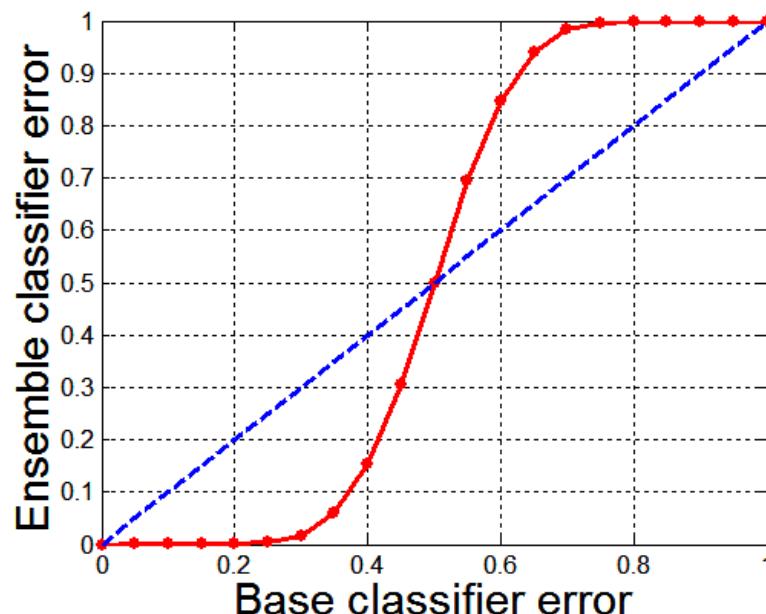
## Example: Why Do Ensemble Methods Work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate,  $\epsilon = 0.35$
  - Majority vote of classifiers used for classification
  - If all classifiers are identical:
    - Error rate of ensemble =  $\epsilon (0.35)$
  - If all classifiers are independent (errors are uncorrelated):
    - Error rate of ensemble = probability of having more than half of base classifiers being wrong

$$e_{\text{ensemble}} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1 - \epsilon)^{25-i} = 0.06$$

# Necessary Conditions for Ensemble Methods

- Ensemble Methods work better than a single base classifier if:
  - All base classifiers are independent of each other
  - All base classifiers perform better than random guessing (error rate < 0.5 for binary classification)



Classification error for an ensemble of 25 base classifiers, assuming their errors are uncorrelated.

# Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
  - Bagging
  - Boosting

# Bagging (Bootstrap AGGregatING)

- Bootstrap sampling: sampling 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

- Build classifier on each bootstrap sample

# Bagging Algorithm

---

## Algorithm 4.5 Bagging algorithm.

---

- 1: Let  $k$  be the number of bootstrap samples.
  - 2: **for**  $i = 1$  to  $k$  **do**
  - 3:     Create a bootstrap sample of size  $N$ ,  $D_i$ .
  - 4:     Train a base classifier  $C_i$  on the bootstrap sample  $D_i$ .
  - 5: **end for**
  - 6:  $C^*(x) = \operatorname{argmax}_y \sum_i \delta(C_i(x) = y).$   
     $\{\delta(\cdot) = 1 \text{ if its argument is true and } 0 \text{ otherwise.}\}$
-

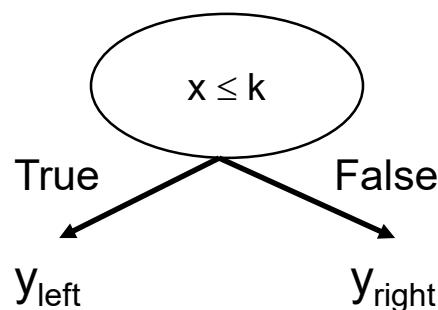
# Bagging Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump (decision tree of size 1)
  - Decision rule:  $x \leq k$  versus  $x > k$
  - Split point  $k$  is chosen based on entropy



# Bagging Example

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 Example

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	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

$$x \leq 0.7 \rightarrow y = 1$$
$$x > 0.7 \rightarrow y = 1$$

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 Example

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

$$x \leq 0.05 \rightarrow y = 1$$
$$x > 0.05 \rightarrow y = 1$$

# Bagging Example

- Summary of Trained Decision Stumps:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

# Bagging Example

- Use majority vote (sign of sum of predictions) to determine class of ensemble classifier

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

Predicted Class

# Improvement of Bagging: Random Forest

- Train a **Bagged Decision Tree**
- But use a modified tree learning algorithm that selects (at each candidate split) **a random subset of features**
  - If we have  $d$  features, consider  $\sqrt{d}$  random features
- **This is called: Feature bagging**
  - **Benefit:** Breaks correlation between trees
    - If one feature is very strong predictor, then every tree will select it, causing trees to be correlated.
- **Random Forests achieve state-of-the-art results in many classification problems!**

# Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, sampling weights may change at the end of boosting round
- Successful algorithms
  - Example 1: adaBoost (some slides for your information or self-study)
  - Example 2: Gradient Boosted Decision Trees

# Boosting

- Records that are **wrongly** classified will have their weights **increased**
- Records that are **correctly** classified will have their weights **decreased**

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

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

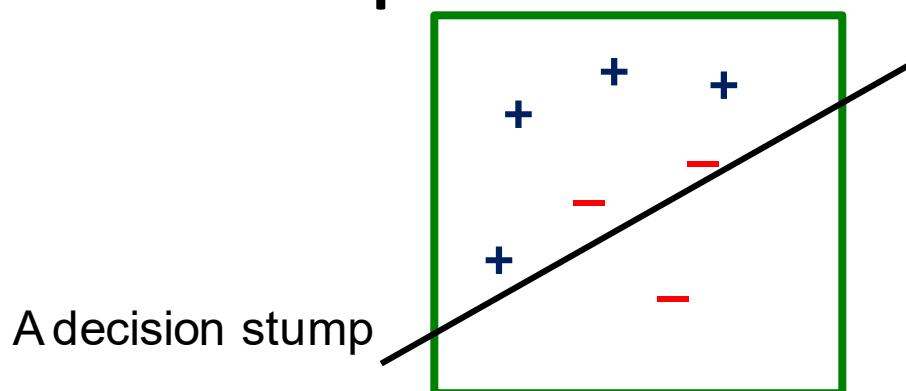
# Build Decision Trees with AdaBoost

Suppose we have training data  $\{(x_i, y_i)\}_{i=1}^N, y_i \in \{1, -1\}$

- Initialize equal weights for all observations
- At each iteration t:
  1. Train a stump  $G_t$  using data weighted by  $w_i$
  2. Compute the misclassification error adjusted by  $w_i$
  3. Compute the weight of the current tree
  4. Reweight each observation based on prediction accuracy

# AdaBoost: Weak learner

- **1. build Decision “stumps”:**
  - 1-level decision tree
  - A decision boundary based on one feature
    - E.g.: If someone is not a smoker, then predict them to live past 80 years old
  - Building blocks of AdaBoost algorithm
  - **Decision stump is a weak learner**



**Boosting theory:**  
if weak learners have  
>50% accuracy then  
we can learn a perfect  
classifier.

# Update Step

2. Calculate the weighted misclassification error

$$err_t = \frac{\sum_{i=1}^N w_i I(y_i \neq G_t(x_i))}{\sum_{i=1}^N w_i}$$

3. Use the error score to weight the current tree in the final classifier:

$$\alpha_t = \log \left( \frac{1 - err_t}{err_t} \right)$$

A classifier with 50% accuracy is given a weight of zero;

4. Use misclassification error and tree weight to reweight the training data:

$$w_i \leftarrow w_i \exp[\alpha_t I(y_i \neq G_t(x_i))]$$

Harder to classify training instances get higher weight

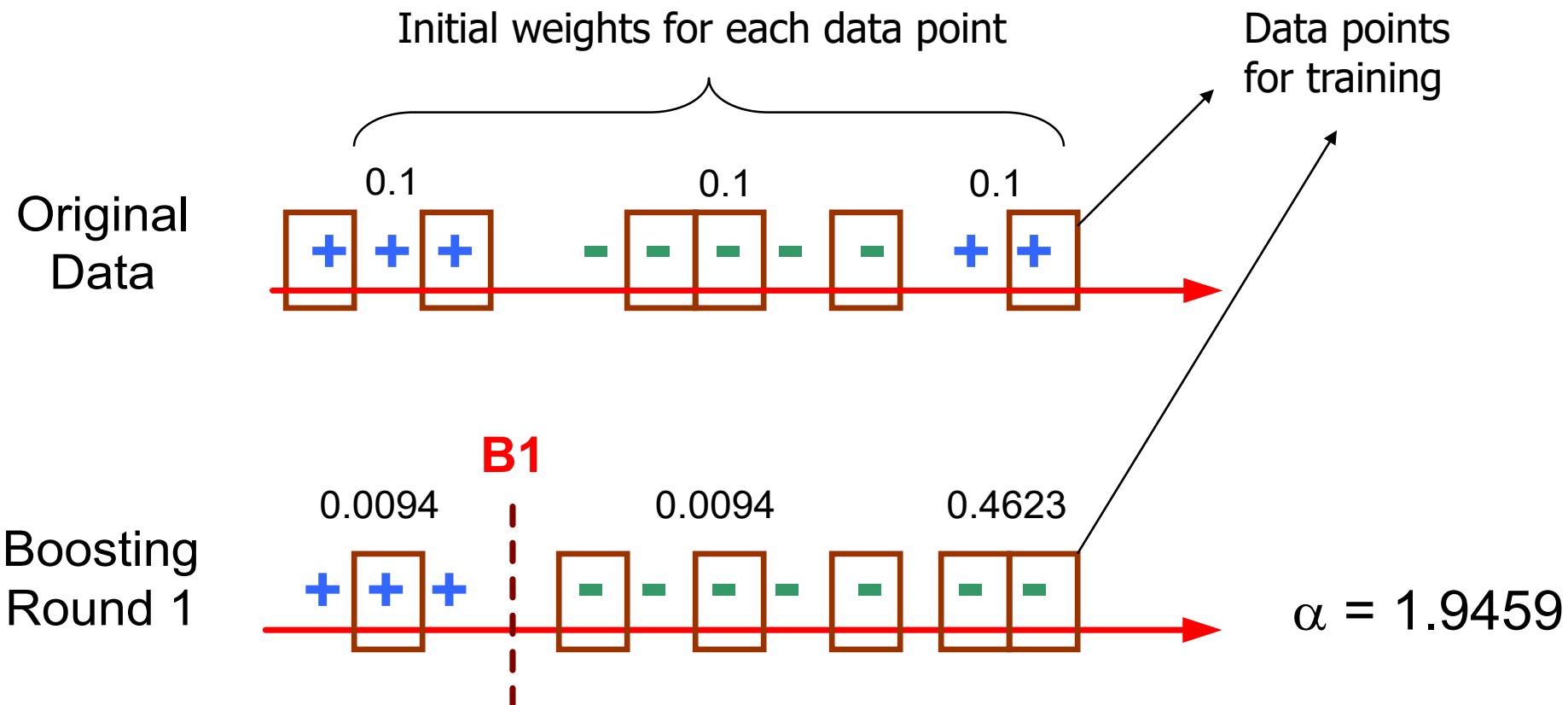
# Final Prediction

- Final prediction is a weighted sum of the predictions from each stump:

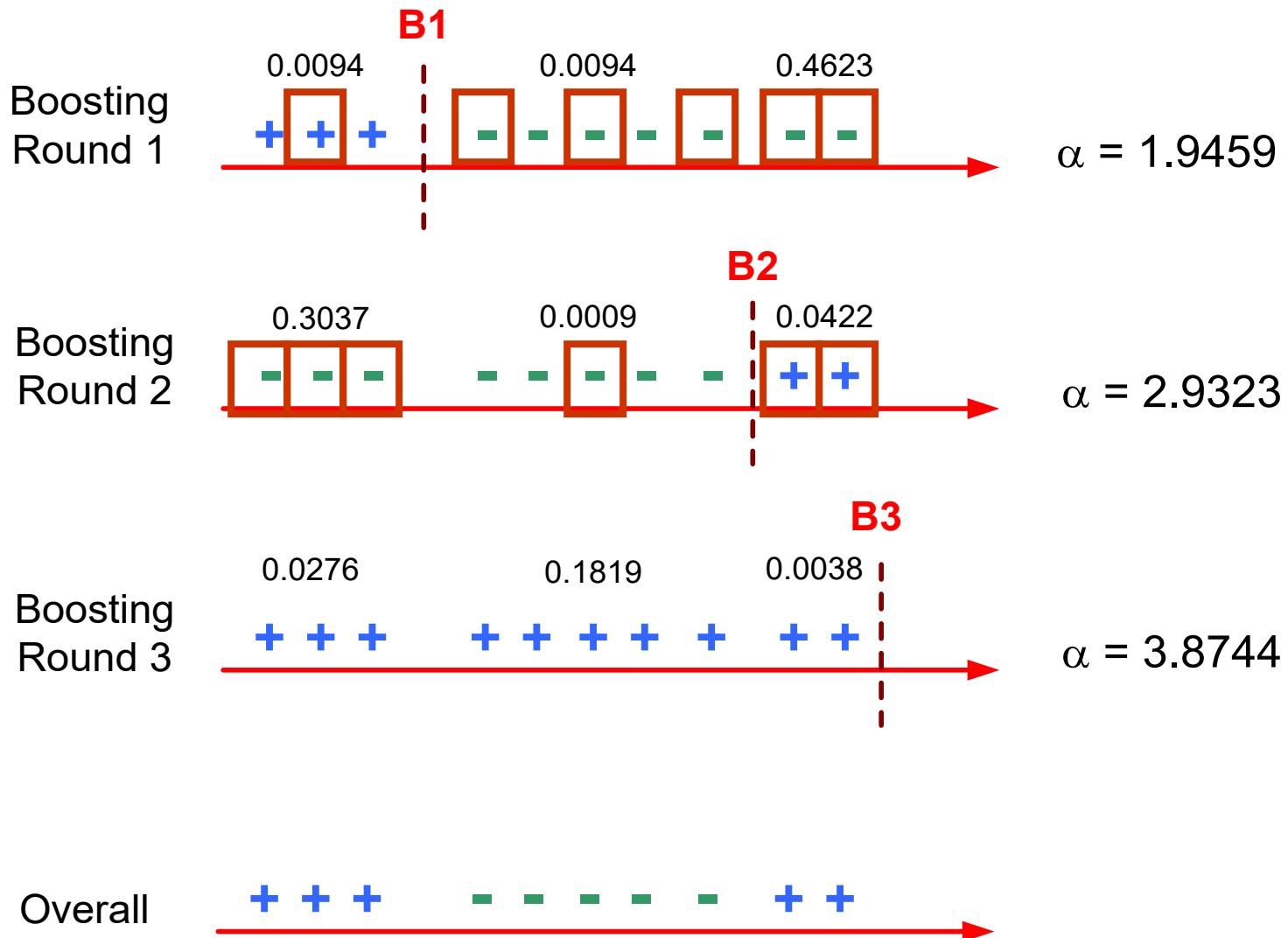
$$G(x) = \text{sign} \left[ \sum_{t=1}^T \alpha_t G_t(x) \right]$$

- More accurate trees are weighted higher in the final model

# Illustrating AdaBoost



# Illustrating AdaBoost

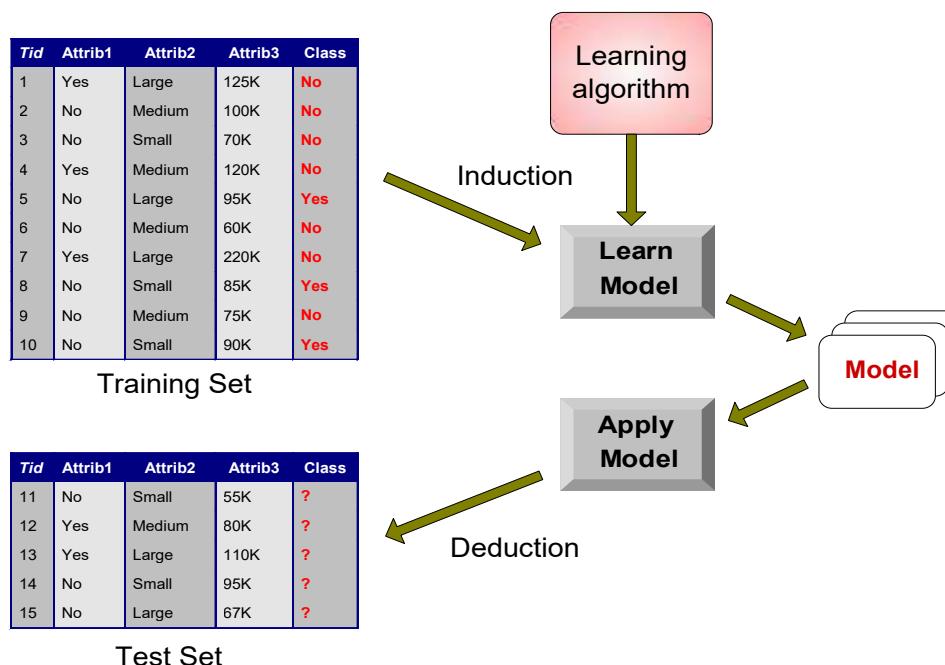


# Outline

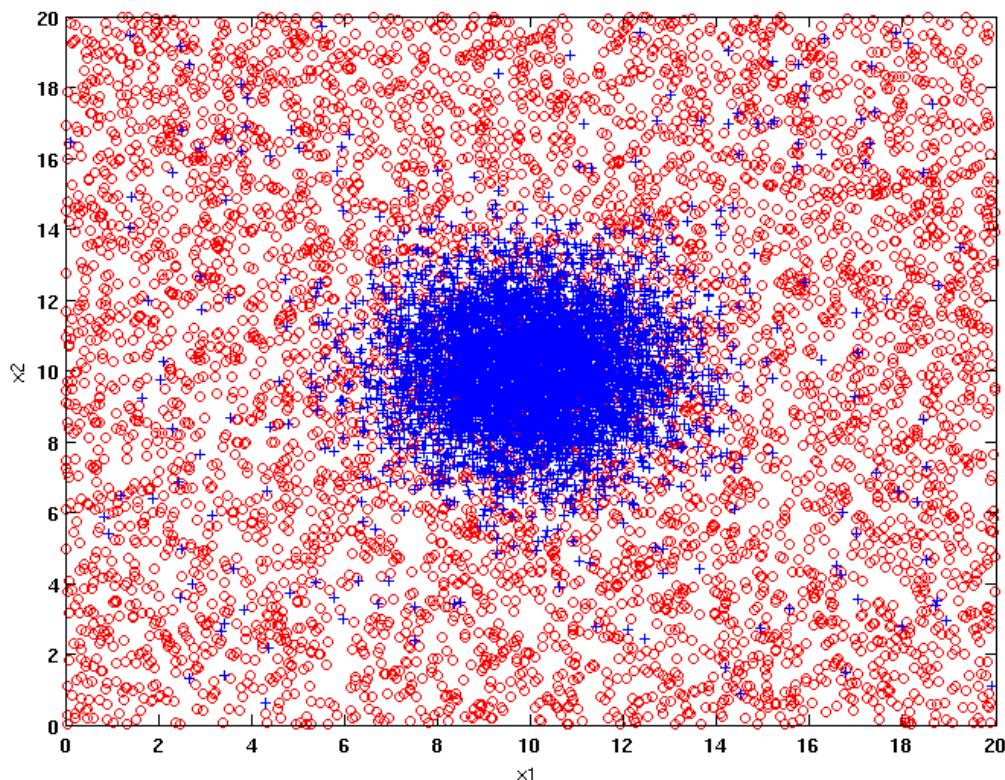
- Classification Techniques
  - Decision Tree
  - Classification based on association rules
  - Ensemble classifier
  - Overfitting
  - Classification evaluation

# Classification Errors

- **Training errors:** Errors committed on the training set
- **Test errors:** Errors committed on the test set
- **Generalization errors:** Expected error of a model over random selection of records from same distribution



# Example Data Set



**Two class problem:**

**+ : 5400 instances**

- 5000 instances generated from a Gaussian centered at  $(10,10)$

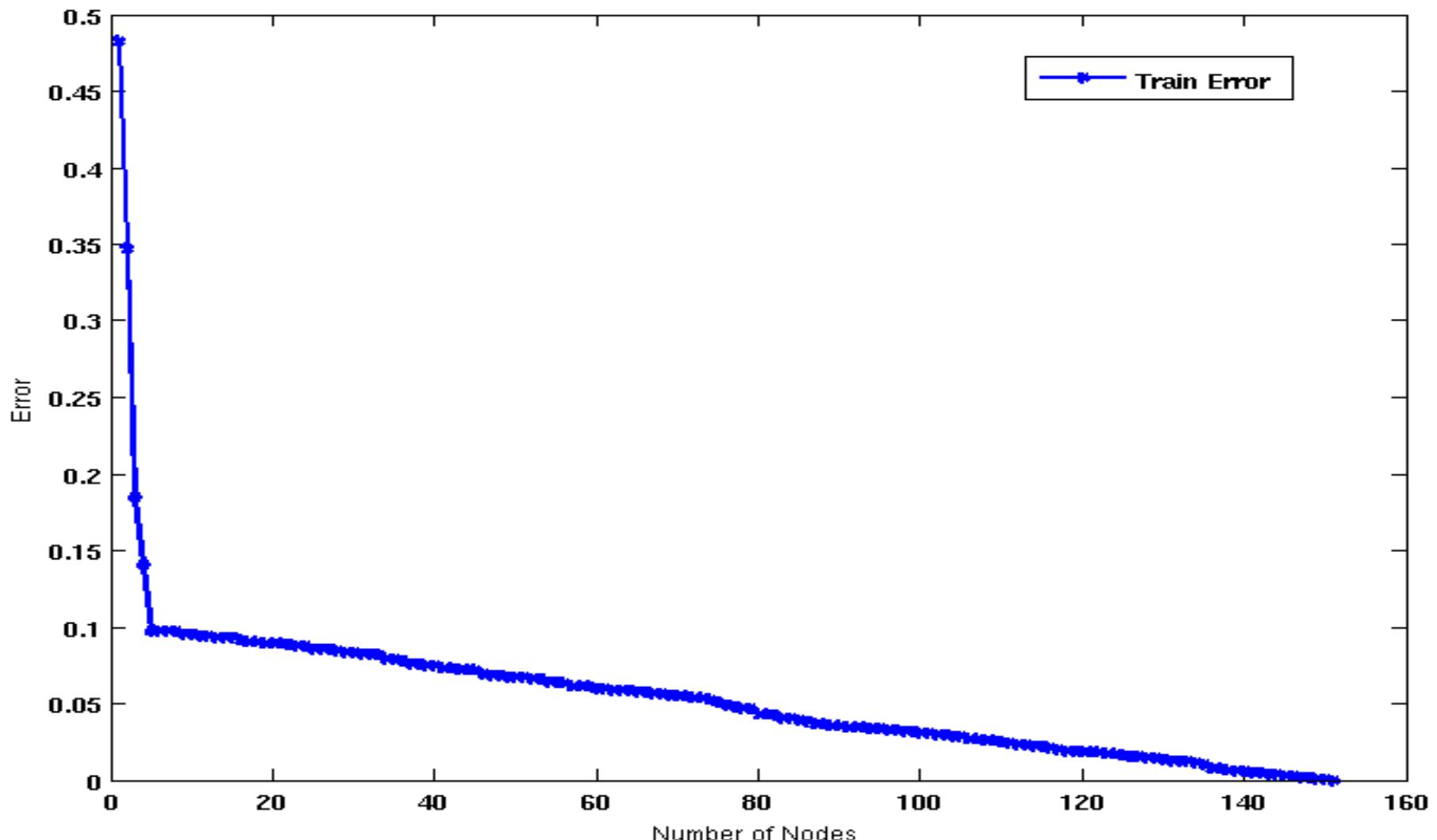
- 400 noisy instances added

**o : 5400 instances**

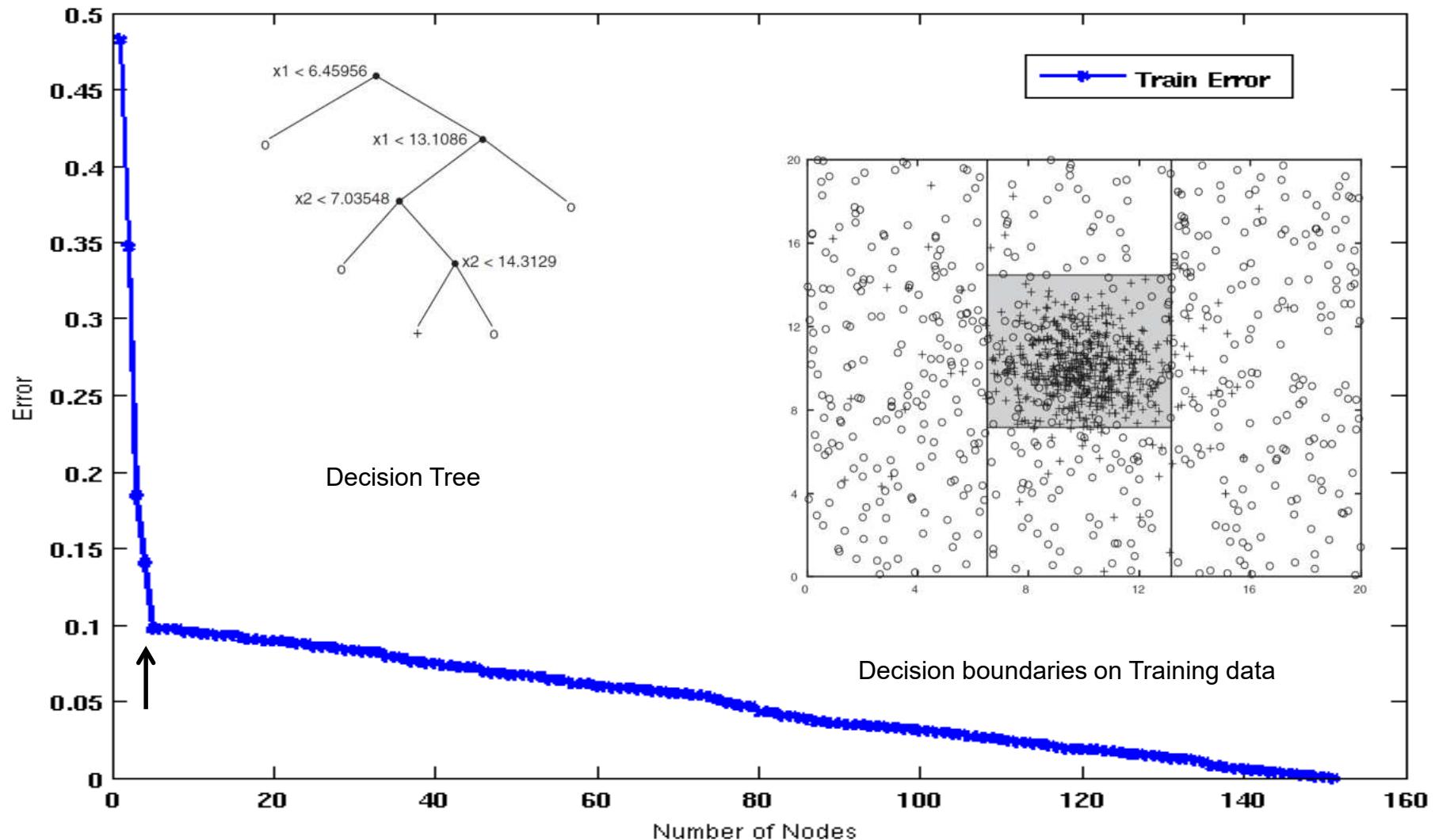
- Generated from a uniform distribution

**10 % of the data used for training and 90% of the data used for testing**

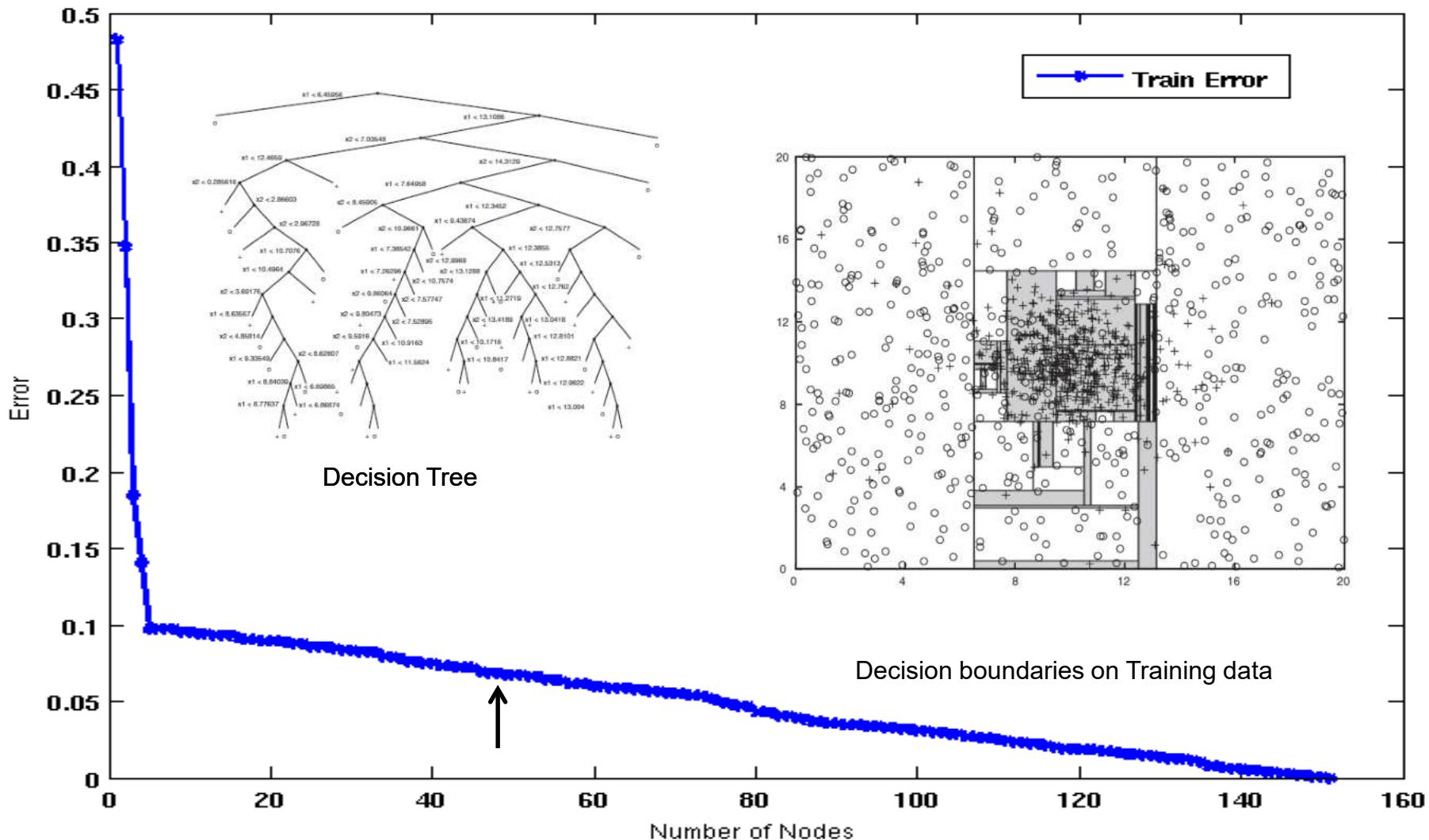
# Increasing number of nodes in Decision Trees



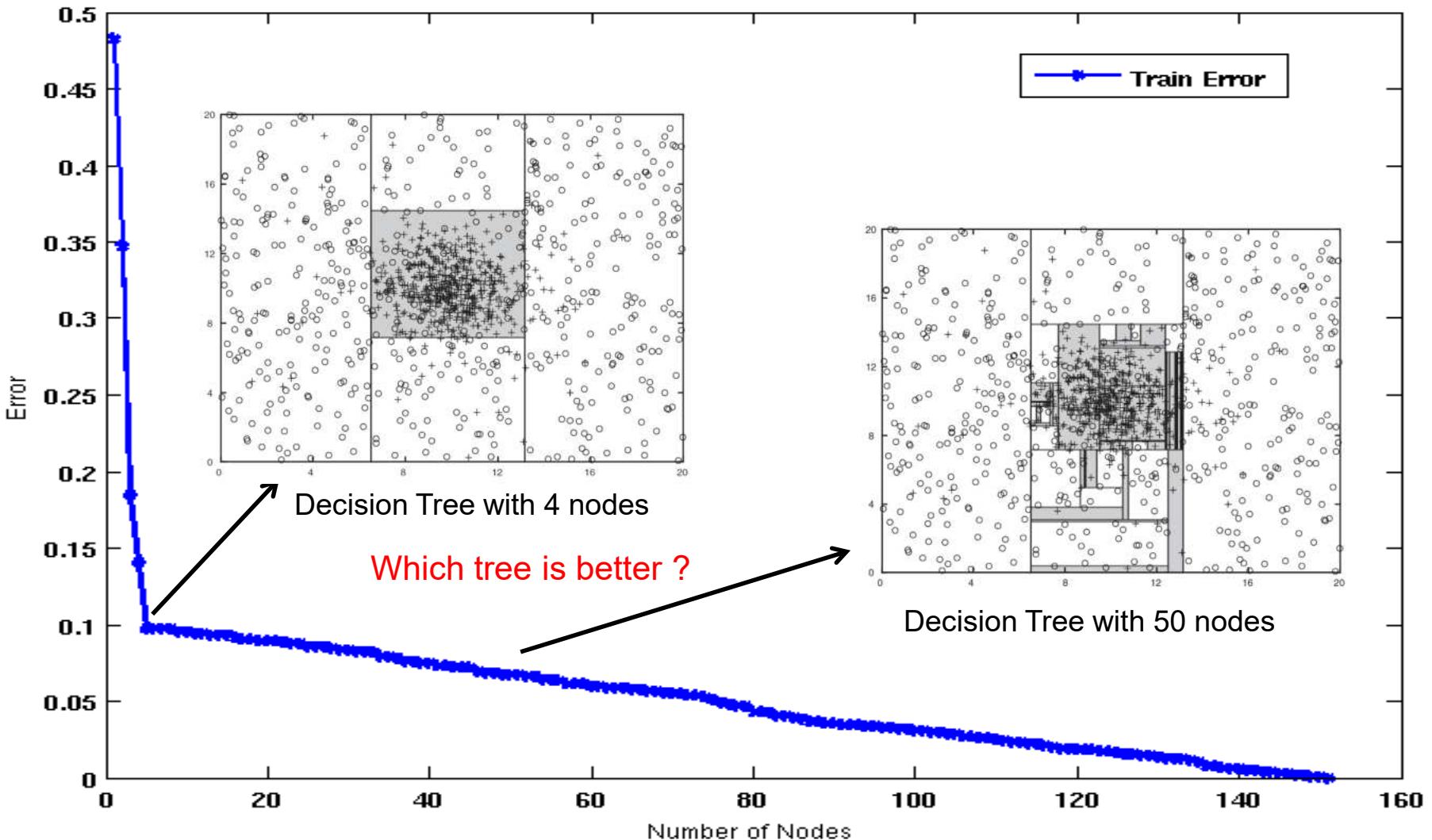
# Decision Tree with 4 nodes



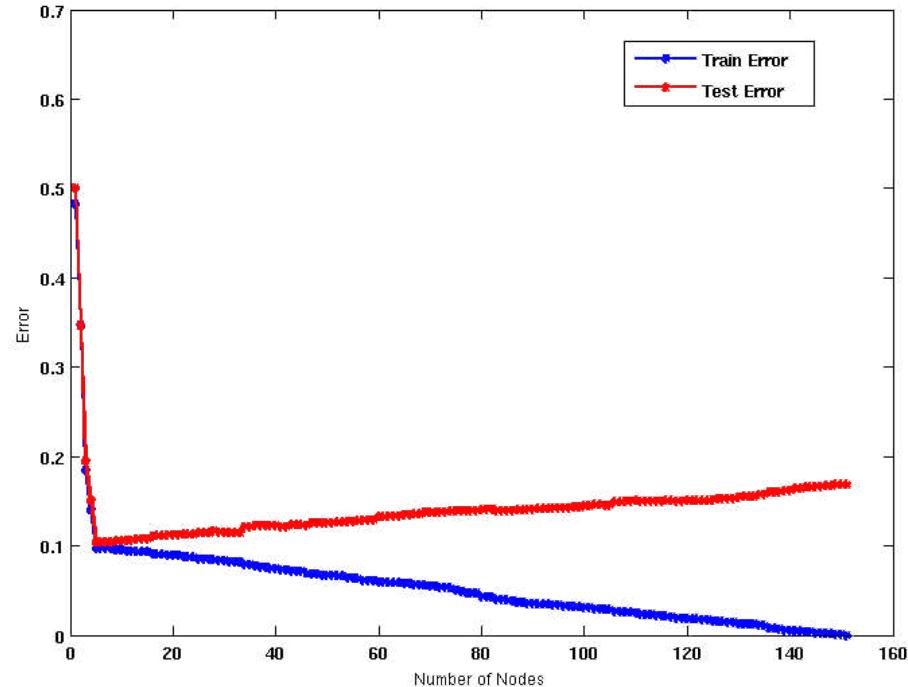
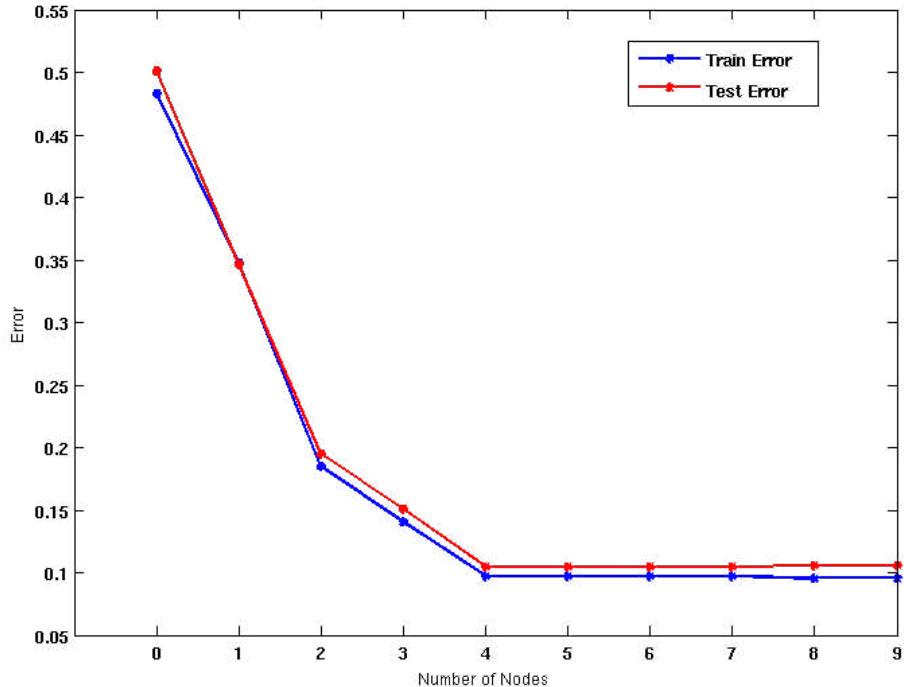
# Decision Tree with 50 nodes



# Which tree is better?



# Model Underfitting and Overfitting

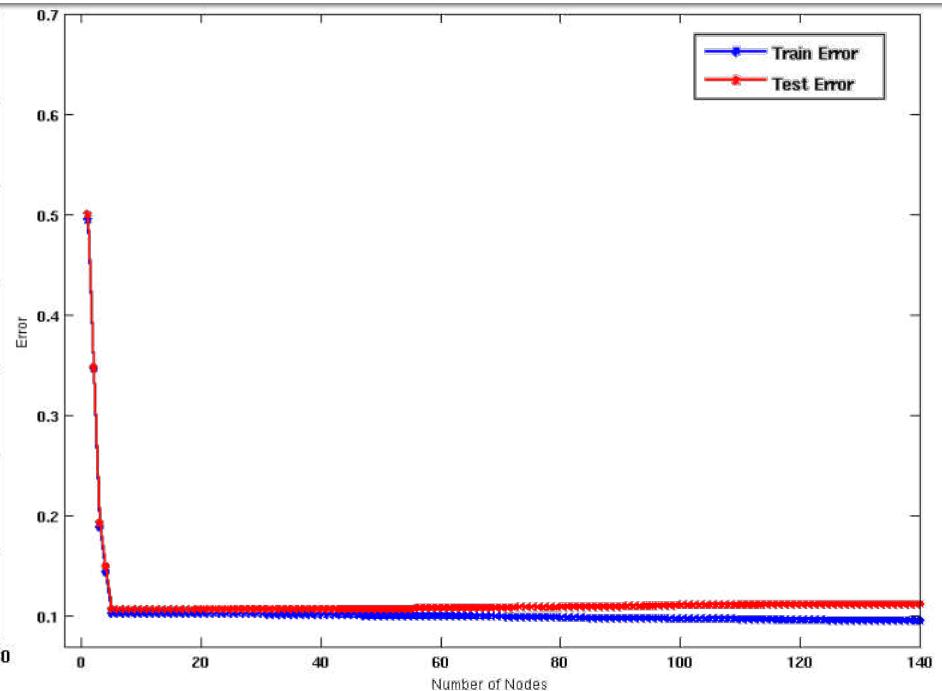
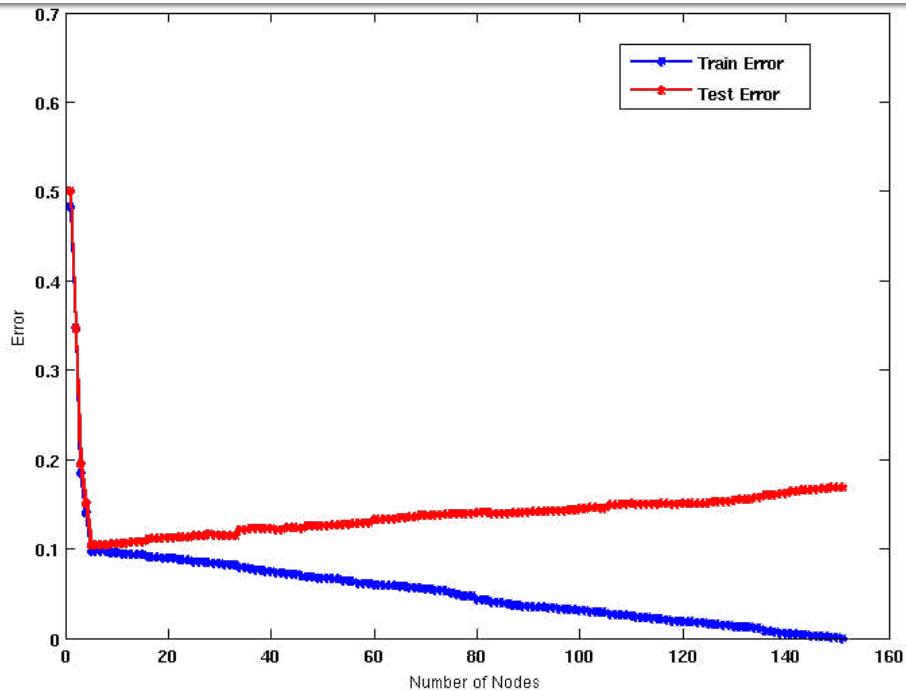


- As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing

**Underfitting:** when model is too simple, both training and test errors are large

**Overfitting:** when model is too complex, training error is small but test error is large

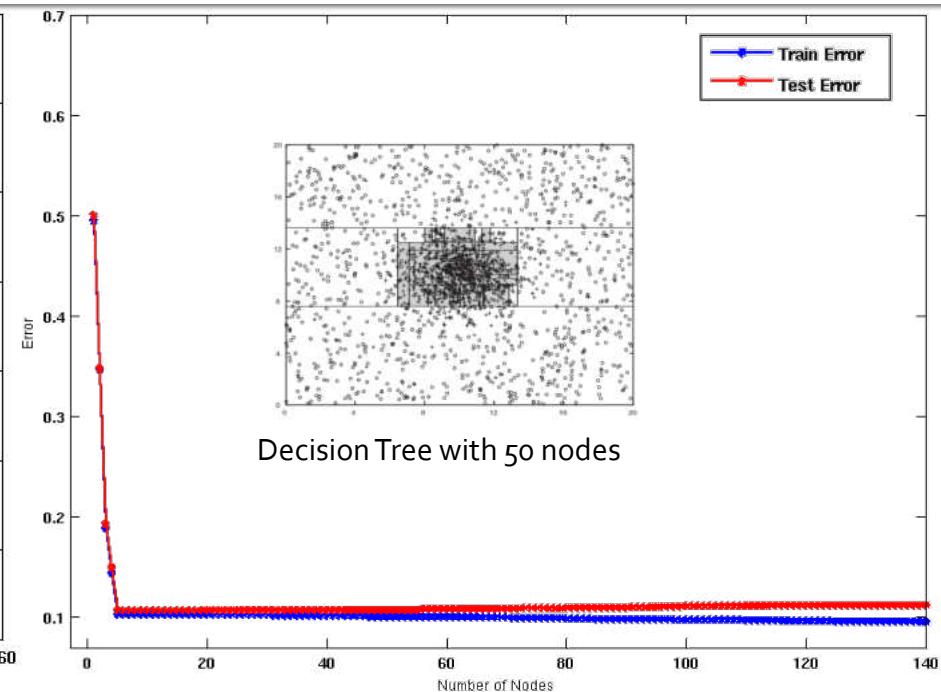
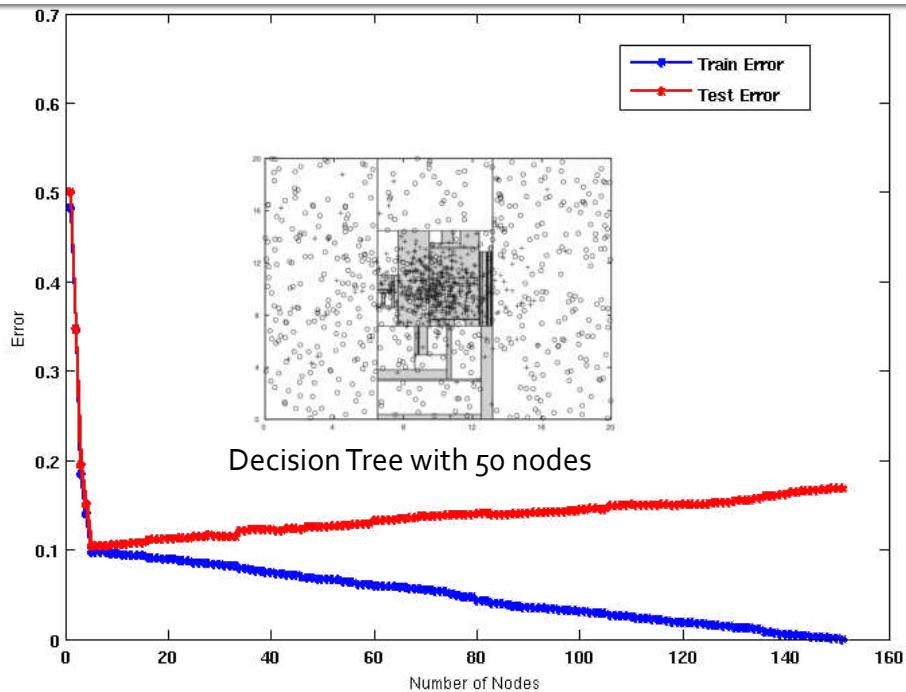
# Model Overfitting – Impact of Training Data Size



Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

# Model Overfitting – Impact of Training Data Size



Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

# Reasons for Model Overfitting

- Not enough training data
- High model complexity
  - Multiple Comparison Procedure

# Notes on Overfitting

Use decision trees as an example:

- Overfitting results in decision trees that are more complex than necessary
- Training error does not provide a good estimate of how well the tree will perform on previously unseen records
- Need ways for estimating generalization errors

# Model Selection

- Performed during model building
- Purpose is to ensure that model is not overly complex (to avoid overfitting)
- Need to estimate generalization error
  - Using Validation Set
  - Incorporating Model Complexity (e.g., number of leaf nodes in decision trees) in model training
    - Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

# Using Validation Set

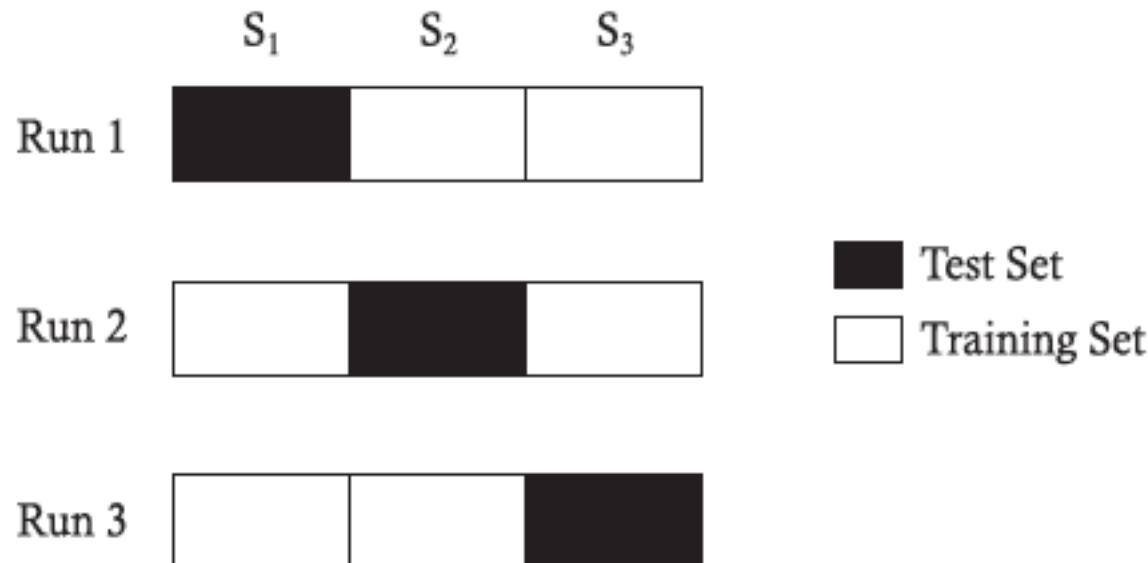
- Divide training data into two parts:
  - Training set:
    - use for model building
  - Validation set:
    - use for estimating generalization error
    - Note: validation set is not the same as test set
- Drawback:
  - Less data available for training

# Classification Model Evaluation

- Purpose:
  - To estimate performance of classifier on previously unseen data (test set)
- Holdout
  - Reserve  $k\%$  for training and  $(100-k)\%$  for testing
  - Random subsampling: repeated holdout
- Cross validation
  - Partition data into  $k$  disjoint subsets
  - $k$ -fold: train on  $k-1$  partitions, test on the remaining one
  - Leave-one-out:  $k=n$

# Cross-validation Example

## ■ 3-fold cross-validation



# Summary

- Classification Techniques
  - Decision Tree
  - Classification based on association rules
  - Ensemble classifier
    - Understand the high-level idea
  - Overfitting
    - Understand the high-level idea
  - Classification evaluation
    - Understand the high-level idea



---

# Introduction to Recommendation Systems

**Jin Yao CHIN**

Nanyang Technological University  
Singapore

# Outline

- ▷ **WHERE** can we find recommendation systems?
- ▷ **WHY** do we need recommendation systems?
- ▷ **WHAT** is recommendation?
- ▷ **HOW** do we make recommendations?
  
- ▷ Different **types** of recommendation systems
- ▷ **Challenges** encountered by recommendation systems



1.

*WHERE* can we find  
recommendation systems?

# Recommendation Systems – WHERE?



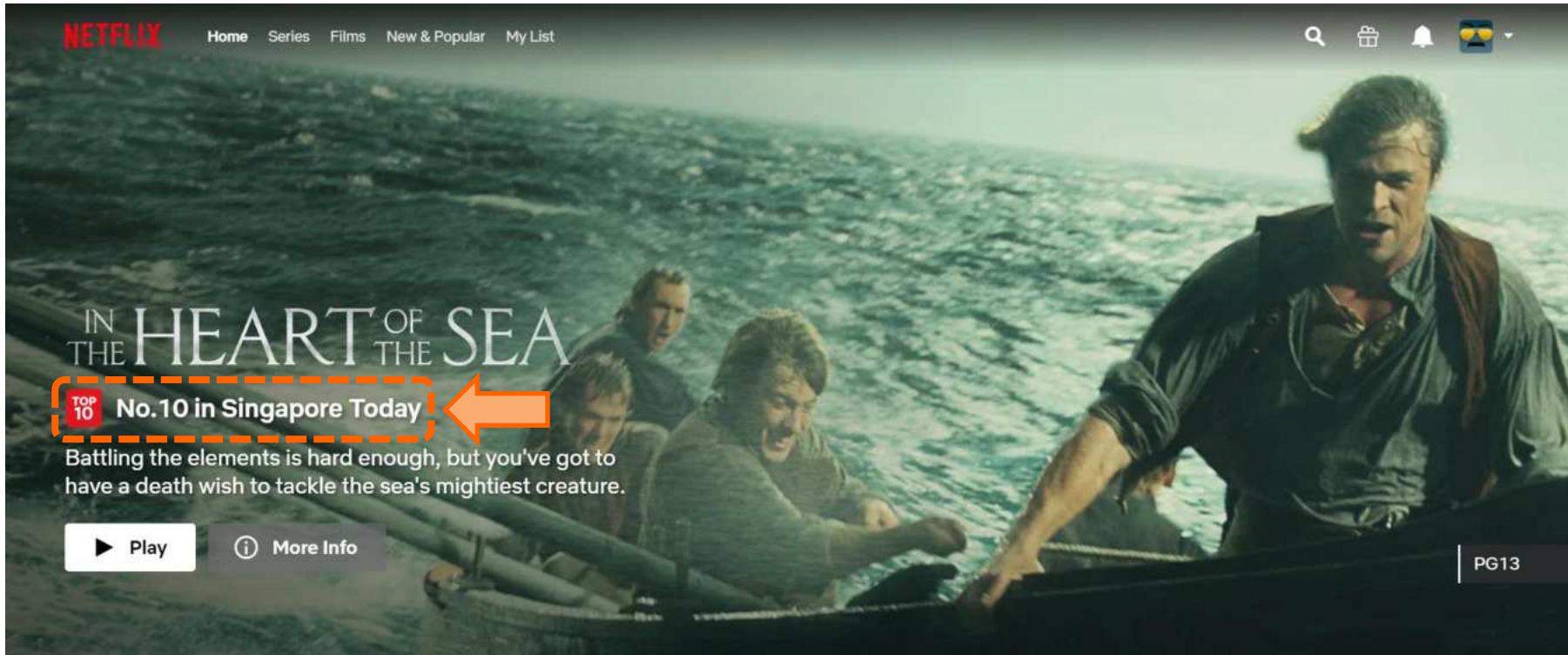
FOURSQUARE

淘宝网  
Taobao.com



Google Maps

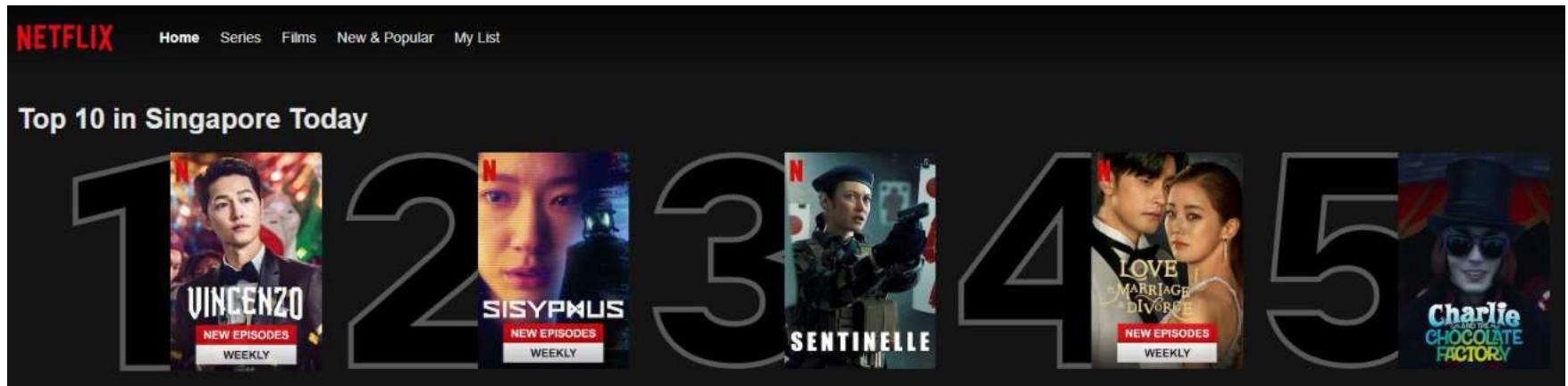
# Recommendation Systems – WHERE?



## ▷ E.g., Netflix

- Netflix might try to recommend a show which is very popular based on your current *location* (E.g., *No. 10 in Singapore Today*)

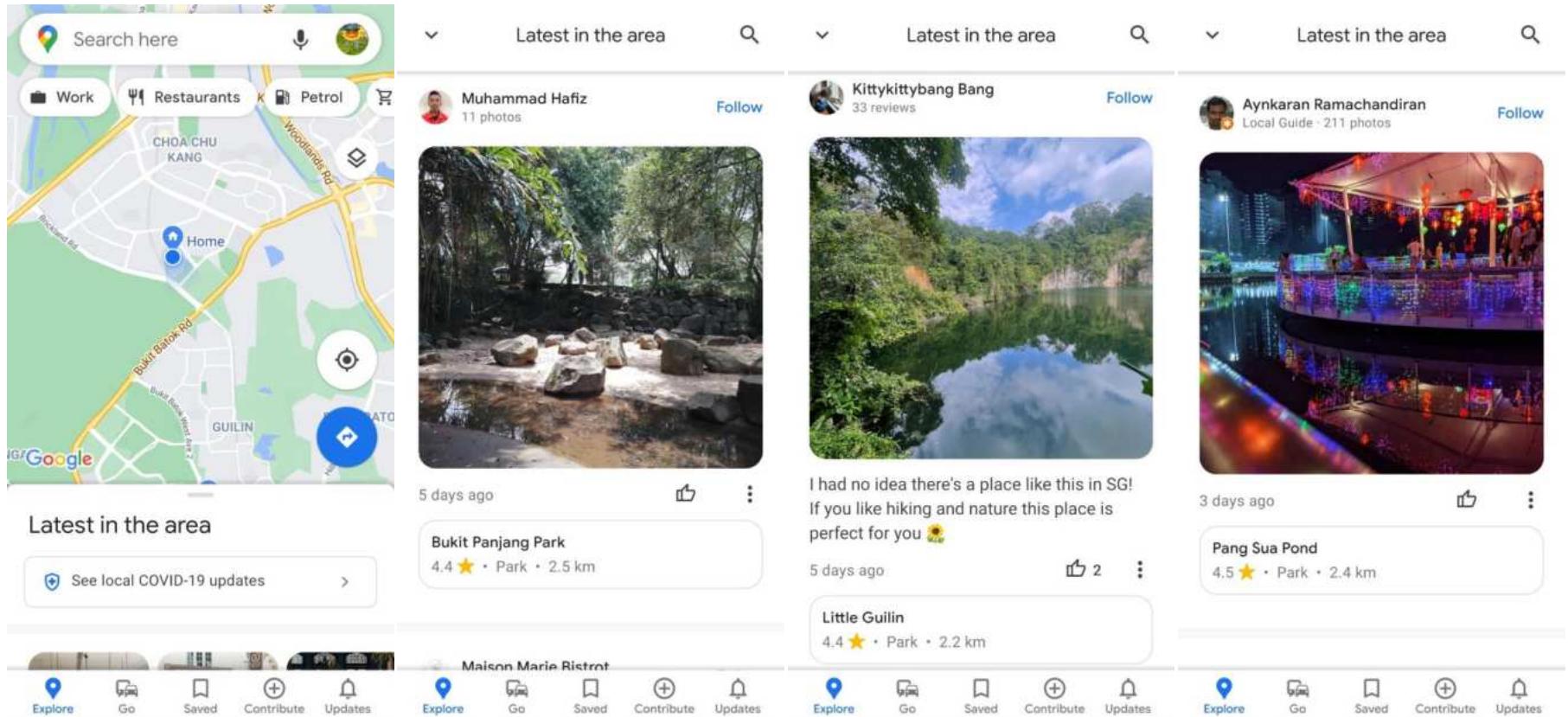
# Recommendation Systems – WHERE?



## ▷ E.g., Netflix

- Netflix might try to recommend the Top 10 shows based on your current *location*
- However, note that these are *not* personalized recommendations
  - Based on other users located in *Singapore*
  - Everyone in *Singapore* gets the same recommendations

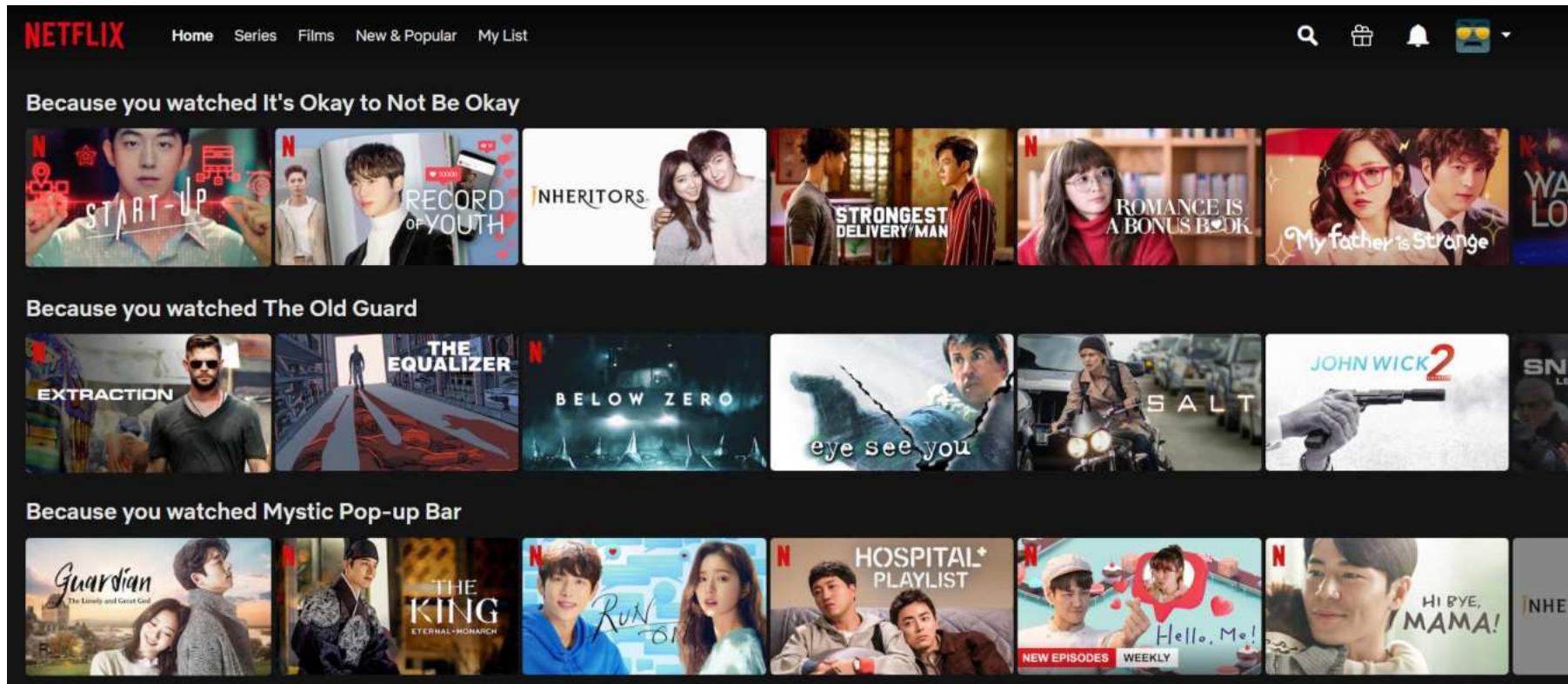
# Recommendation Systems – WHERE?



## ▷ E.g., Google Maps

- Given your current *location*, Google Maps might recommend popular locations *near you*

# Recommendation Systems – WHERE?



▷ E.g., Netflix

- Shows are often recommended based on your viewing *history* (i.e., shows that you have watched previously)

# Recommendation Systems – WHERE?



The screenshot shows the Amazon Singapore website. At the top, there's a navigation bar with links for 'Hello Select your address', a search bar, and categories like 'All', 'Best Sellers', 'Today's Deals', 'Home', 'Prime', 'Customer Service', 'Home Improvement', 'Electronics', 'New Releases', 'Books', 'Gift Ideas', and 'Computers'. Below this is a secondary navigation bar for 'Books' with categories such as 'All Books', 'Arts & Photography', 'Business & Investing', 'Children's Books', 'Fiction', 'History', 'Mystery & Suspense', 'School Books', and 'Travel & Holiday'. The main content area shows the product page for 'Artificial Intelligence: A Modern Approach' (Fourth Edition) by Stuart Russell and Peter Norvig. The book cover features a grid of various AI-related images, including chess pieces, a man in a suit, a woman, a landscape, and a plane. The title 'Artificial Intelligence' is prominently displayed at the bottom of the cover. The product details include the title, authors, a 5-star rating with 1 rating, a link to see all formats and editions, and a price of S\$255.41 for Hardcover. It also notes that the item is an international product from outside Singapore.

Artificial Intelligence: A Modern Approach Hardcover – 10 November 2020

by Stuart Russell (Author), Peter Norvig (Author)

5 star rating 1 rating

See all formats and editions

Hardcover S\$255.41

1 New from S\$255.41

International product from outside Singapore Learn More.

The most comprehensive, up-to-date introduction to the theory and practice of artificial intelligence. The long-anticipated revision of Artificial Intelligence: A Modern Approach explores the full breadth and depth of the field of artificial intelligence (AI). The 4th Edition brings readers up to date on the latest technologies, presents concepts in a more unified manner, and offers new or expanded coverage of machine learning, deep learning, transfer learning, multiagent systems, robotics, natural language processing, causality, probabilistic programming, privacy, fairness, and safe AI.

## ▷ E.g., Amazon

- Let's assume that we are viewing some book in Amazon (e.g., *Artificial Intelligence: A Modern Approach*)

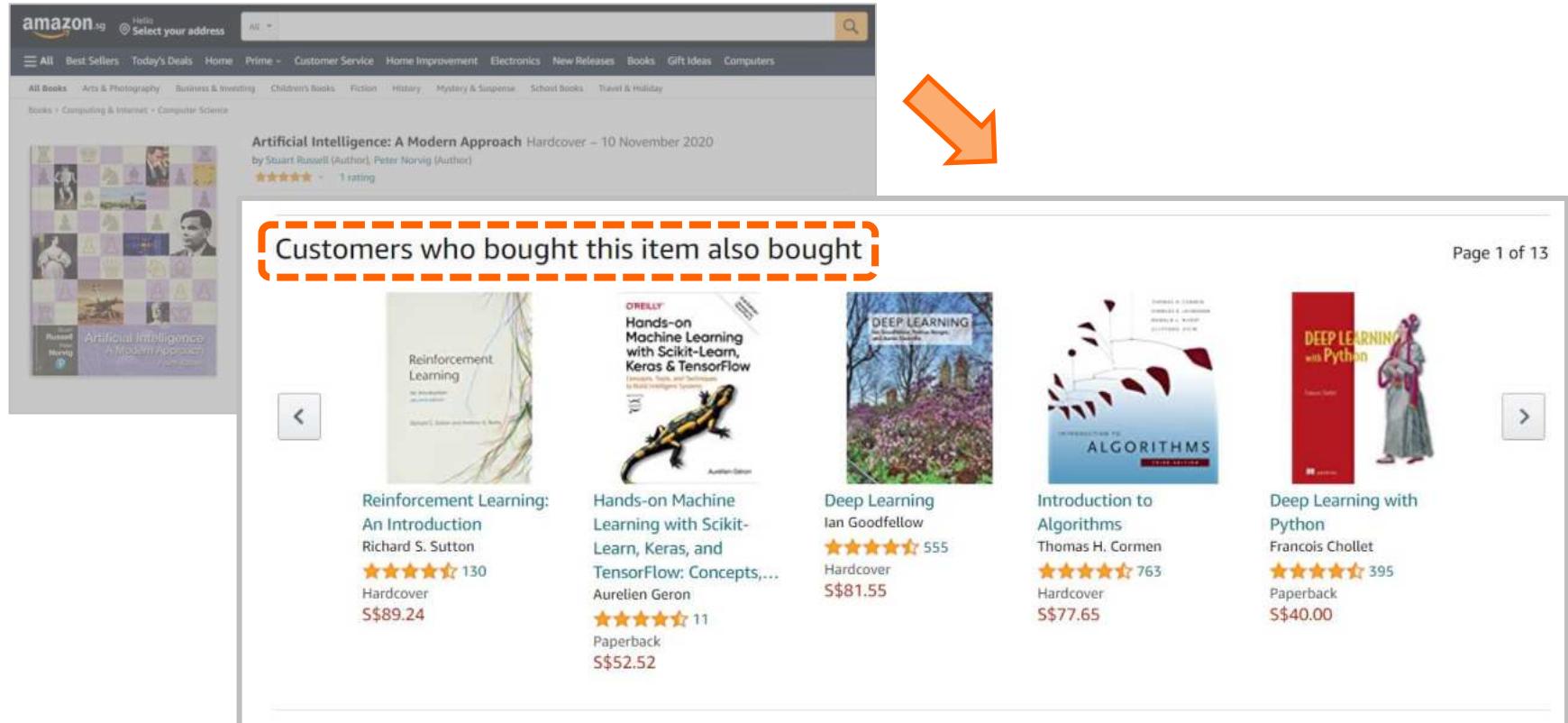
# Recommendation Systems – WHERE?

The screenshot shows a product page for "Artificial Intelligence: A Modern Approach" on Amazon.sg. At the top, there's a navigation bar with categories like All, Best Sellers, Today's Deals, Home, Prime, Customer Service, Home Improvement, Electronics, New Releases, Books, Gift Ideas, and Computers. Below the navigation, a breadcrumb trail shows Books > Computing & Internet > Computer Science. The main product image is a collage of various AI-related images. To the right of the product image, an orange arrow points down to a section titled "Frequently bought together". This section contains three recommended items: "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques... by Aurelien Geron" (Paperback, S\$52.52), "Deep Learning" by Ian Goodfellow (Hardcover, S\$81.55), and another book whose cover is partially visible. A total price of S\$389.48 is displayed, along with a button to "Add all three to Cart".

## ▷ E.g., Amazon

- On the same page, Amazon would try to recommend other items (i.e., books) that are *frequently bought together*

# Recommendation Systems – WHERE?



## ▷ E.g., Amazon

- On the same page, Amazon would try to recommend other items (i.e., books) that were *bought* by ‘similar’ customers

# Recommendation Systems – WHERE?

- ▷ Recommendation systems can be found everywhere!
  - Movies, Books, Music, Points of Interests, ...
- ▷ (Literally) an indispensable part of our daily lives



2.

*WHY* do we need  
recommendation systems?

# Recommendation Systems – WHY?

1. *Large catalogue*
  - w/ up to millions of items
2. *Beneficial to both the end users & the businesses*



amazon



yelp\*

NETFLIX



YouTube

FOURSQUARE



淘宝网  
Taobao.com



Google Maps

# Large Catalogue 😞

- ▷ “Amazon has an inventory of about **12 million items** across all its categories and services” [1]
- ▷ Let’s say... I would like to get a new chair (*but I am not entirely sure what kind of a chair would be good*)
  - There could be thousands of chairs being sold
  - Different types of chairs: Office, Dining, ...
  - Different colours, sizes, prices, ...
  - It could be very challenging and tedious to find something ideal!
  - Recommender systems can help to narrow down the options effectively 😊



# Large (and expanding) Catalogue 😞

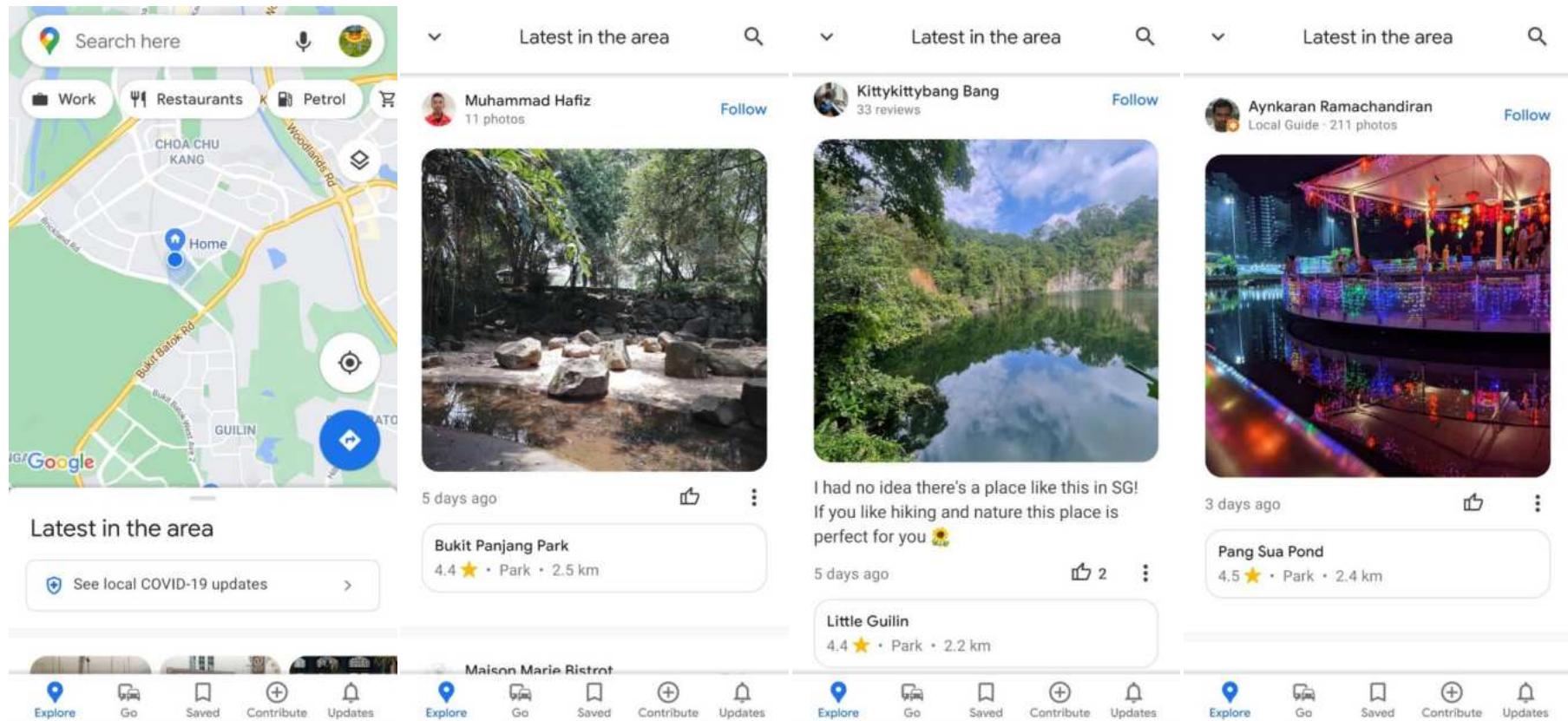
- ▷ Platforms such as YouTube, TikTok, etc. are based on User Generated Content (**UGC**)
- ▷ In contrast to platforms such as Netflix, the catalogue is *expanding rapidly!*



[1] <https://sg.oberlo.com/blog/youtube-statistics> (Retrieved on 08/03/2021)

[2] <https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/> (Retrieved on 08/03/2021)

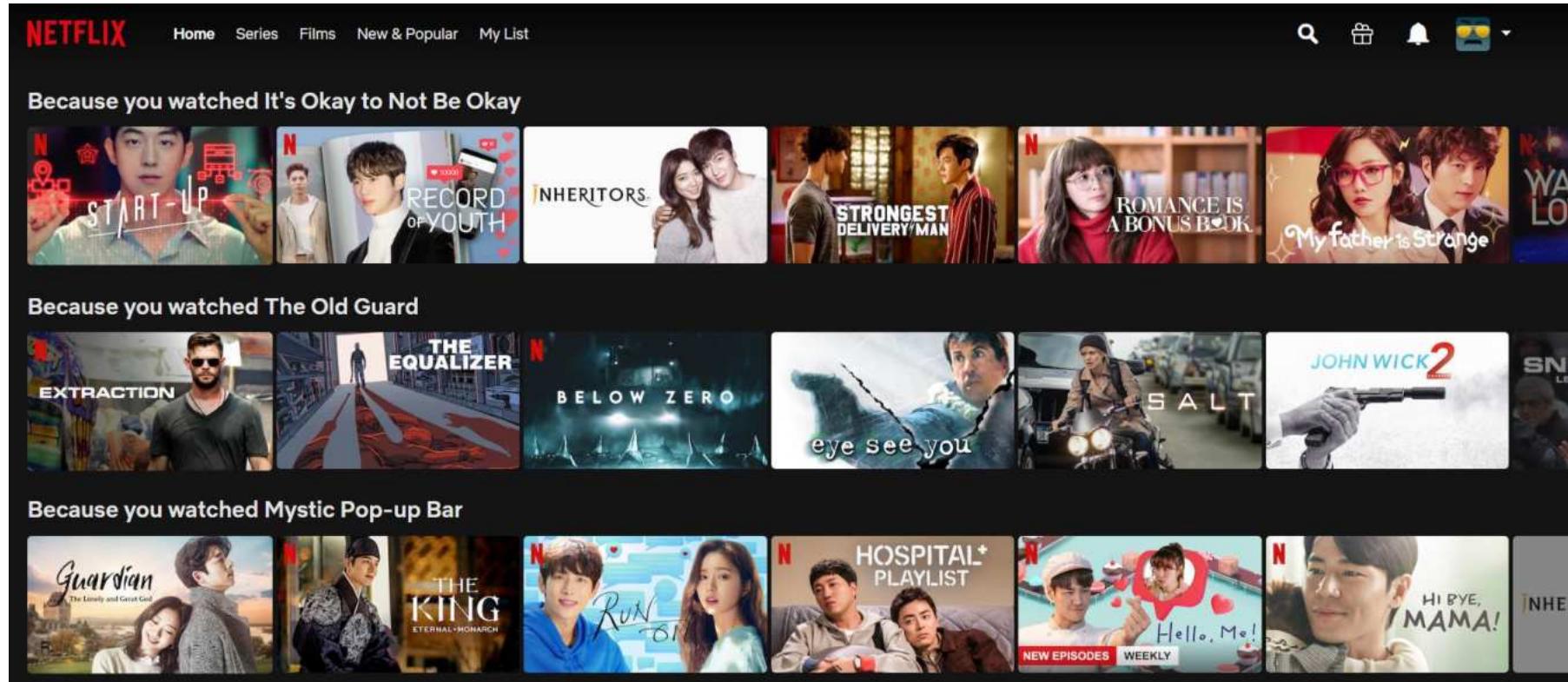
# Beneficial to End Users 😊



## ▷ E.g., Google Maps

- The ‘explore’ feature assists users in finding *nearby points of interests*
- Users might not be aware of these locations before this!

# Beneficial to End Users ☺



## ▷ E.g., Netflix

- By leveraging your **viewing history**, Netflix could recommend other shows which you are most likely to watch...
- Similarly, you might not be aware of such shows prior to this!

# Beneficial to End Users ☺

- ▷ A good recommender system would be akin to having a friend who knows you even better than yourself
  - Assists users in finding *items of interest*
  - Introduces new items to users (*Novelty*)
  - Presents unexpected items to users (*Serendipity*)



# Beneficial to Businesses 😊

- ▷ Helps item providers (e.g., sellers, content creators, etc.) deliver their products to its intended audience
- ▷ Improves customer satisfaction
  - Less time spent on browsing / searching
  - Caters to the preferences of different users
- ▷ The underlying recommendation system could be the **key difference** between...
  - 2 platforms having the same catalogue
  - 2 platforms with the same type of products



# 3. *WHAT* is recommendation?

# Recommendation Systems – WHAT?

## ▷ Where?

- Everywhere!
- Part and parcel of our daily lives

## ▷ Why?

- Large (and perhaps expanding) catalogues
- Good for both end users & businesses

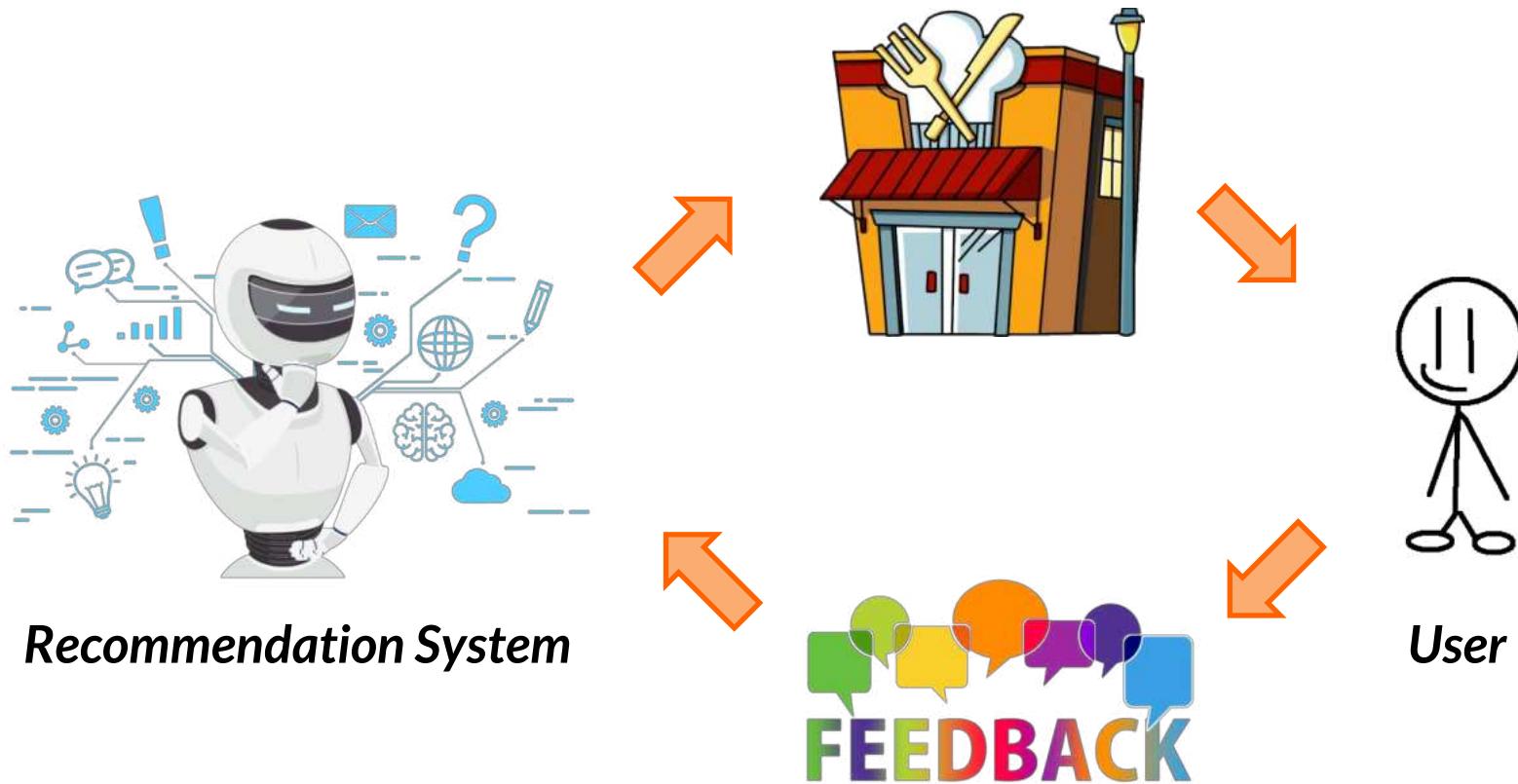
## ▷ What? → *Formalizing the recommendation problem*

- What are the inputs? Outputs?
- Are we making good (or bad) recommendations?



# A (Virtuous) Cycle

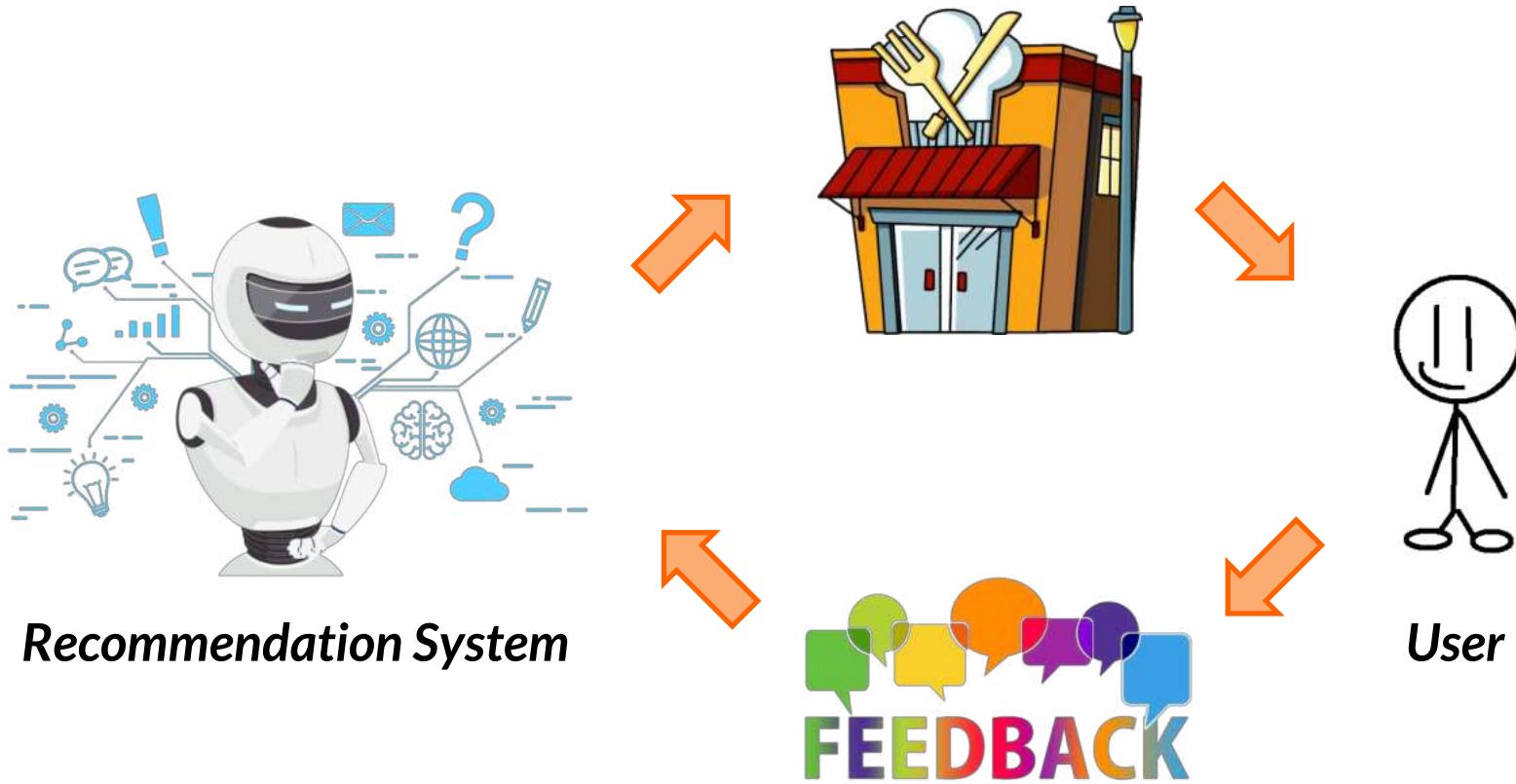
- ▷ Recommendation systems are driven by user interactions



- More interactions → Better awareness of user preferences
- Better awareness of user preferences → Better recommendations
- Better recommendations → More interactions

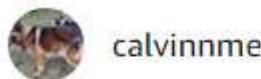
# Recommendations & Feedback

- ▷ Recommendation systems
  - **Outputs:** Items of interest (e.g., Movies, Books, etc.)
  - **Inputs:** User feedback (*Explicit & Implicit*)



# Explicit Feedback

- ▷ For example, users can provide a *rating* for an item



★★★★★

17 December 2020 - **Published on Amazon.com**

Adam Zhang

★★★★★

2 November 2020 - **Published on Amazon.com**

**Verified Purchase**

★★★★★

by S\*\*\* ✓ **Verified Purchase**

★★★★★

by Bhaskar V. ✓ **Verified Purchase**



★★★★★

26 May 2020 - **Published on Amazon.com**

**Verified Purchase**



★★★★★

17 August 2020 - **Published on Amazon.com**

**Verified Purchase**

★★★★★

vie\*\*\* (SG 🇸🇬 ) Feb 26, 2021

★★★★★

sui\*\*\* (SG 🇸🇬 ) Feb 25, 2021

# Explicit Feedback

- ▷ For example, users can provide a rating for an item
  - Furthermore, some platforms allow users to provide a *review*



**Solid MI**

By [Zimmer](#) on September 2, 2018

Format: Blu-ray

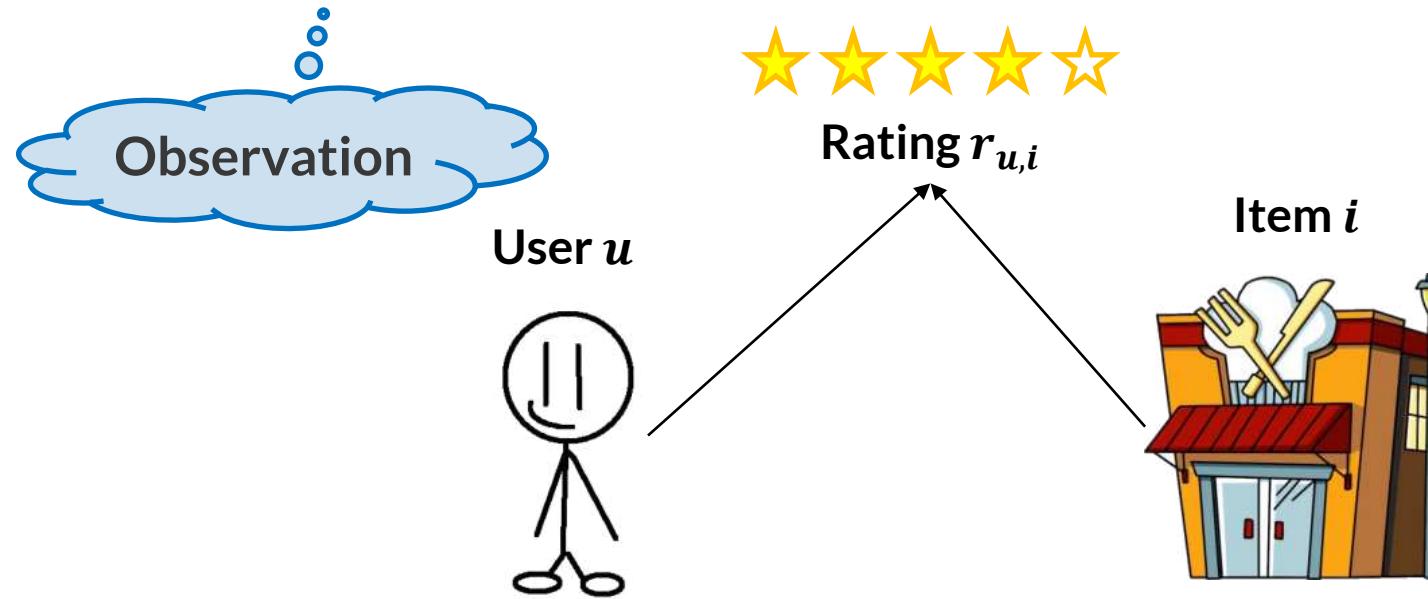
No doubt one of if not the best movie released this year and, just my IMO, in the top 3 Mission films. However im not sure it is quite deserving of the high RT rating it received. It does drag a bit in the second act when Solomon Lane is introduced again. The film needed a truly great scene stealing villain IMO to compete with the great action and Cruise's stunts, and Lane just isnt that interesting. Much has been said of Cavill and his amazing moustache and he's decent but a bit wooden. Great physical presence though. Cruise is solid as usual. Really really enjoyed the first act and the action scenes toward the end were great. The score by Lorne Balfe might just be the best MI score yet. Should have cut the running time a bit tho

# Explicit Feedback – Rating Prediction

For each user  $u$ , we would like to estimate the rating  $\hat{r}_{u,i}$  for any new item  $i$

▷ (Input) Matrix:  $R \in \mathbb{R}^{N \times M}$

- $N$  users,  $M$  items
- $r_{u,i} = \{1, \dots, 5\}$  if user  $u$  has interacted with item  $i$ , 0 otherwise



▷ Recommend new items that the user would rate highly

# Explicit Feedback – Rating Prediction

- ▷ Given the *observed entries*, recover the *missing entries*
  - $\{r_{ui}\}$ : The set of *observed entries*



	5		4
		3	1
	4	2	
			3
			5



# Explicit Feedback – Rating Prediction

- ▷ Given the *observed entries*, recover the *missing entries*
  - $\{r_{ui}\}$ : The set of *observed entries*
  - $\{\hat{r}_{ui}\}$ : The set of *missing entries (to be recovered)*



	5	?	4	?
	?	3	?	1
	4	2	?	?
	?	?	3	5



# Rating Prediction – Evaluation

- ▷ Are we making good or bad recommendations?
  - Compare *predicted ratings* against *actual ratings*
- ▷ Evaluation (for rating prediction) are usually done using *pointwise metrics*
  - Mean Absolute Error (**MAE**)
  - Root Mean Squared Error (**RMSE**)



# A Simple Example

1	5	?	4	3
2	2	3	?	1
3	4	2	2	4
4	?	4	3	5

**Missing Entries**  
(in red)



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

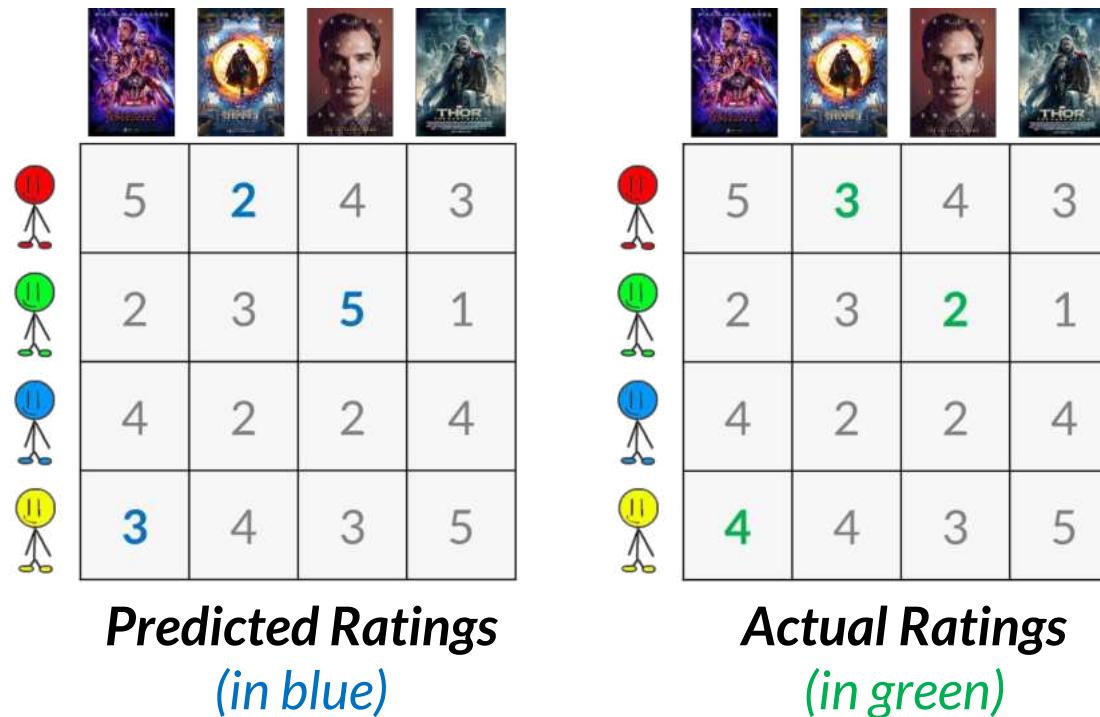
**Predicted Ratings**  
(in blue)



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

**Actual Ratings**  
(in green)

# A Simple Example – MAE & RMSE



$$\triangleright MAE = (|2 - 3| + |5 - 2| + |3 - 4|)/3 \approx 1.66$$

$$\triangleright RMSE = \sqrt{((2 - 3)^2 + (5 - 2)^2 + (3 - 4)^2)/3} \approx 1.91$$

- As compared to MAE, RMSE penalizes *large prediction errors* more (*Errors are squared*)

# Implicit Feedback

- ▷ Explicit Feedback (e.g. ratings, reviews, ...)
  - Users need to provide them consciously
  - Might not always be readily available



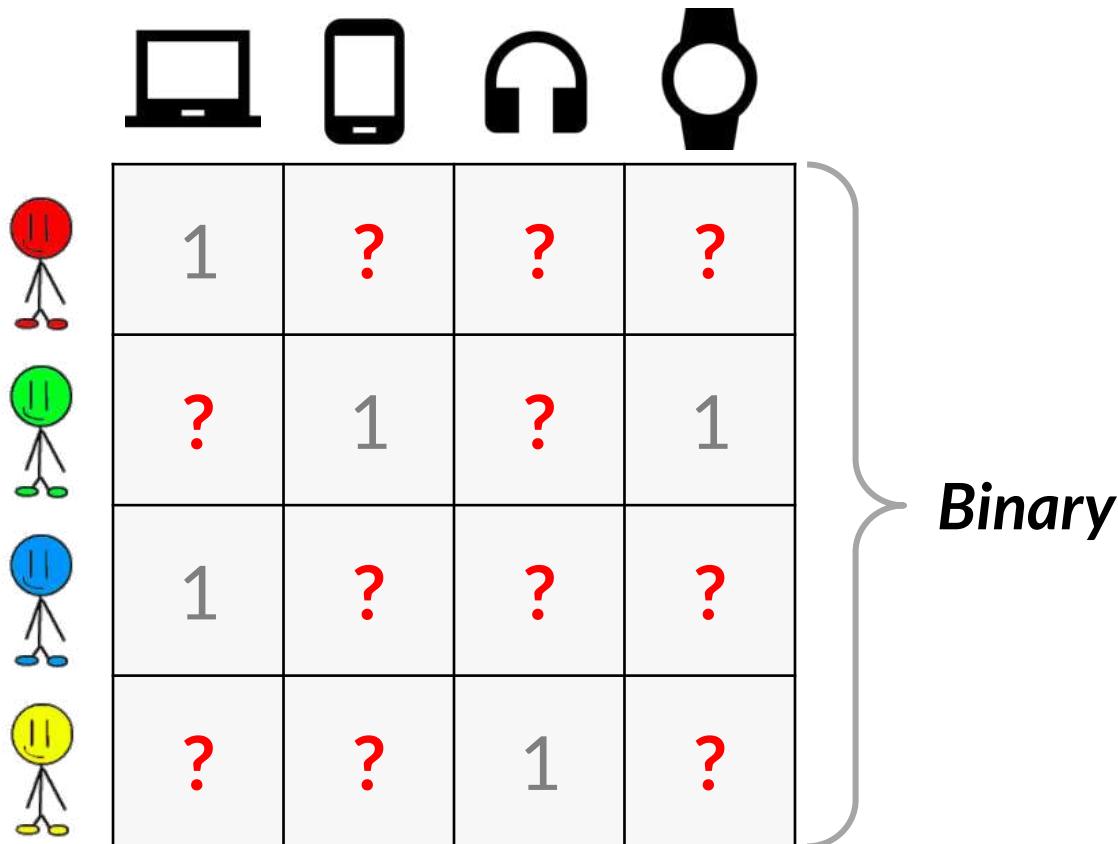
- ▷ Alternative: Implicit Feedback
  - Collected in the background
  - Examples:
    - User has *clicked* on a link
    - User has *watched* a movie
    - User has *viewed* a product
    - User has *purchased* a product



# Implicit Feedback – Top-K Recommendation

▷ Given the *observed entries*, predict the likelihoods for the *missing entries*

- $\{r_{ui}\}$ : The set of *observed entries*
- $\{\hat{r}_{ui}\}$ : The set of *missing entries (to be predicted)*



# A Simple Example

	Laptop	Smartphone	Headphones	Watch
User 1	1	?	?	?
User 2	?	1	?	1
User 3	1	?	?	?
User 4	?	?	1	?

*Missing Observations  
(in red)*

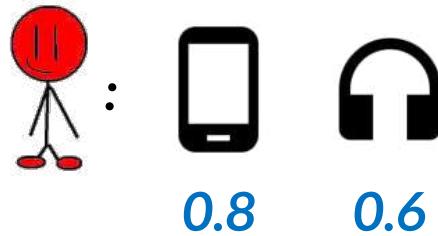


	Laptop	Smartphone	Headphones	Watch
User 1	1	0.8	0.6	0.1
User 2	0.2	1	0.7	1
User 3	1	0.5	0.3	0.4
User 4	0.8	0.6	1	0.9

*Predicted Likelihoods  
(in blue)*



*Top-2 Recommendations for*



# A Simple Example

*Top-2 Recommendations for*



:

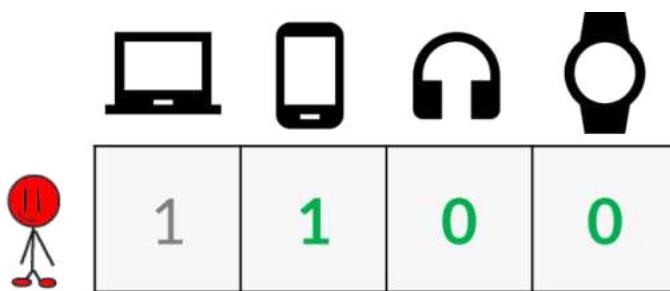


0.8



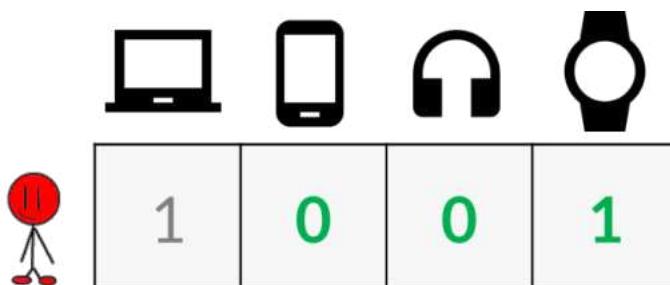
0.6

*Scenario 1:*



- User purchased a phone
- Recommendation is **good!** 😊

*Scenario 2:*



- User purchased a watch
- Recommendation is **bad!** 😞

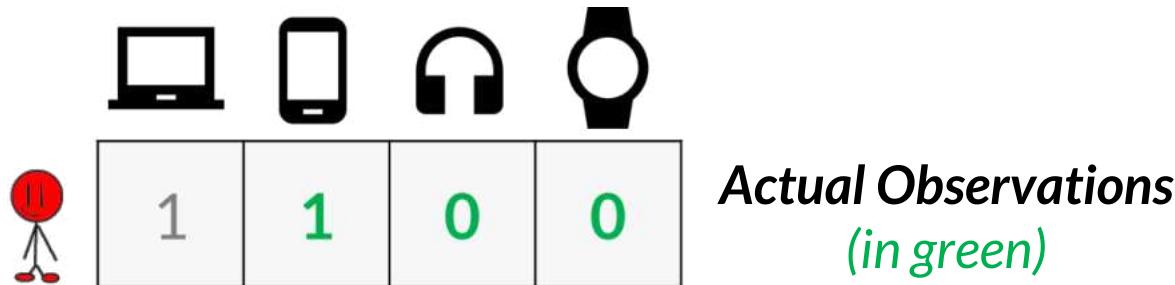
*Actual Observations  
(in green)*

# Top-K Recommendation – Evaluation

- ▷ Are we making good or bad recommendations?
  - Compare *actual observations* against the list of top-K recommended items
- ▷ Evaluation are usually done using *classification metrics* and/or *ranking-based metrics*
  - **Classification:** Precision, Recall, ...
  - **Ranking-based:**
    - Normalized Discounted Cumulative Gain ( $nDCG$ )
    - Mean Reciprocal Rank ( $MRR$ )
    - ...



# Classification vs Ranking-based



*Top-2 Recommendations for*



:



or



0.8      0.6

Option 1

0.8      0.6

Option 2

- For **classification-based metrics**, Options 1 & 2 are equally good...
  - The phone (*actual purchase*) is in both top-2 lists
  - ***Order does not matter***
- For **ranking-based metrics**, Option 1 is better than Option 2!
  - The phone (*actual purchase*) is ranked above the headphones for Option 1 ☺

# Implicit Feedback – Caveat

- ▷ E.g., User did not *watch* a movie
  - Possibility 1: User dislikes that movie
  - Possibility 2: User is *not aware* of such a movie
- ▷ E.g., User has *watched* a movie
  - Does the user like or dislike the movie?
- ▷ E.g., User *purchased* a product
  - Does the user like the product?
  - What if it was bought as a gift for someone else?
- ▷ For *implicit feedback*, observations (or the lack thereof) do not necessarily reflect preferences



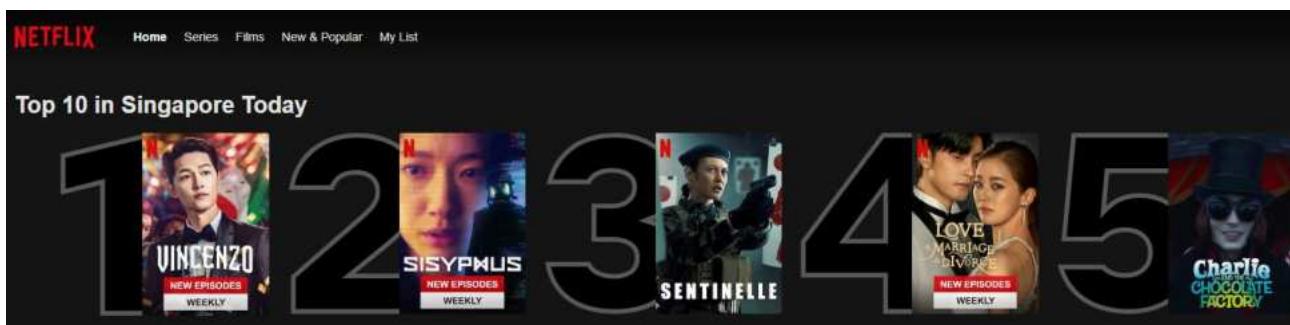
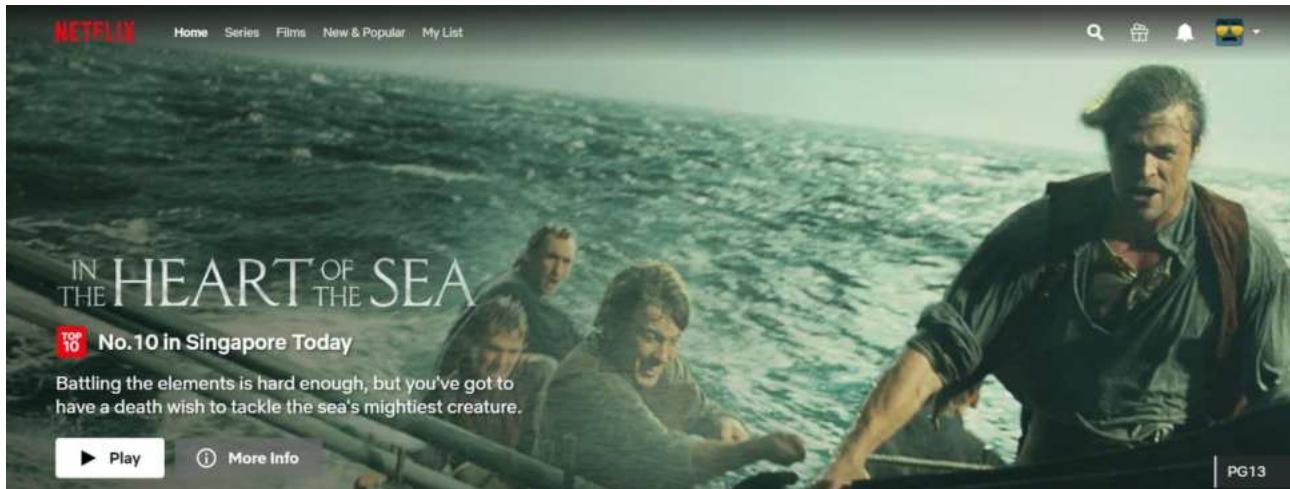
4.

*HOW* do we make  
recommendations?

# Making Recommendations

## ▷ Non-personalized

- Random
- Most Popular (e.g., Top 10 movies)
- Most Recent (e.g., Latest uploads on YouTube)



# Making Recommendations

## ▷ Personalized

- Adapts to the preferences of each user
- In most cases, provides much better recommendations than non-personalized methods

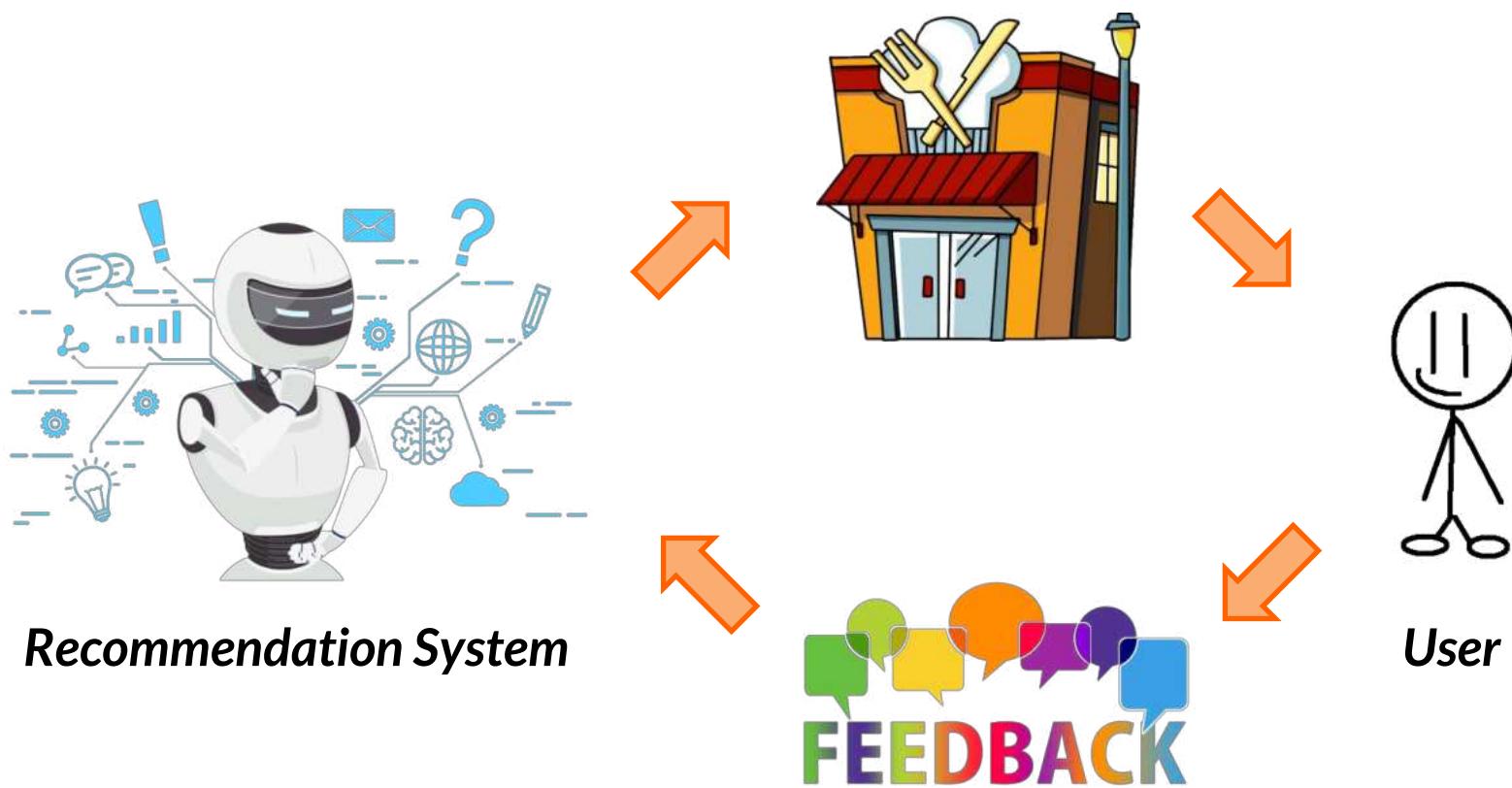
The screenshot shows the Netflix homepage with three main personalized recommendation sections:

- Because you watched It's Okay to Not Be Okay**: Includes thumbnails for "START-UP", "RECORD OF YOUTH", "INHERITORS", "STRONGEST DELIVERYMAN", "ROMANCE IS A BONUS BOOK", and "My father is Strange".
- Because you watched The Old Guard**: Includes thumbnails for "EXTRACTION", "THE EQUALIZER", "BELOW ZERO", "eye see you", "SALT", and "JOHN WICK 2".
- Because you watched Mystic Pop-up Bar**: Includes thumbnails for "Guardian: The Lonely and Great God", "THE KING ETERNAL MONARCH", "RUN 2", "HOSPITAL PLAYLIST", "Hello, Me!", and "HI BYE, MAMA!".

The top navigation bar includes links for Home, Series, Films, New & Popular, My List, and icons for search, gift, notifications, and profile.

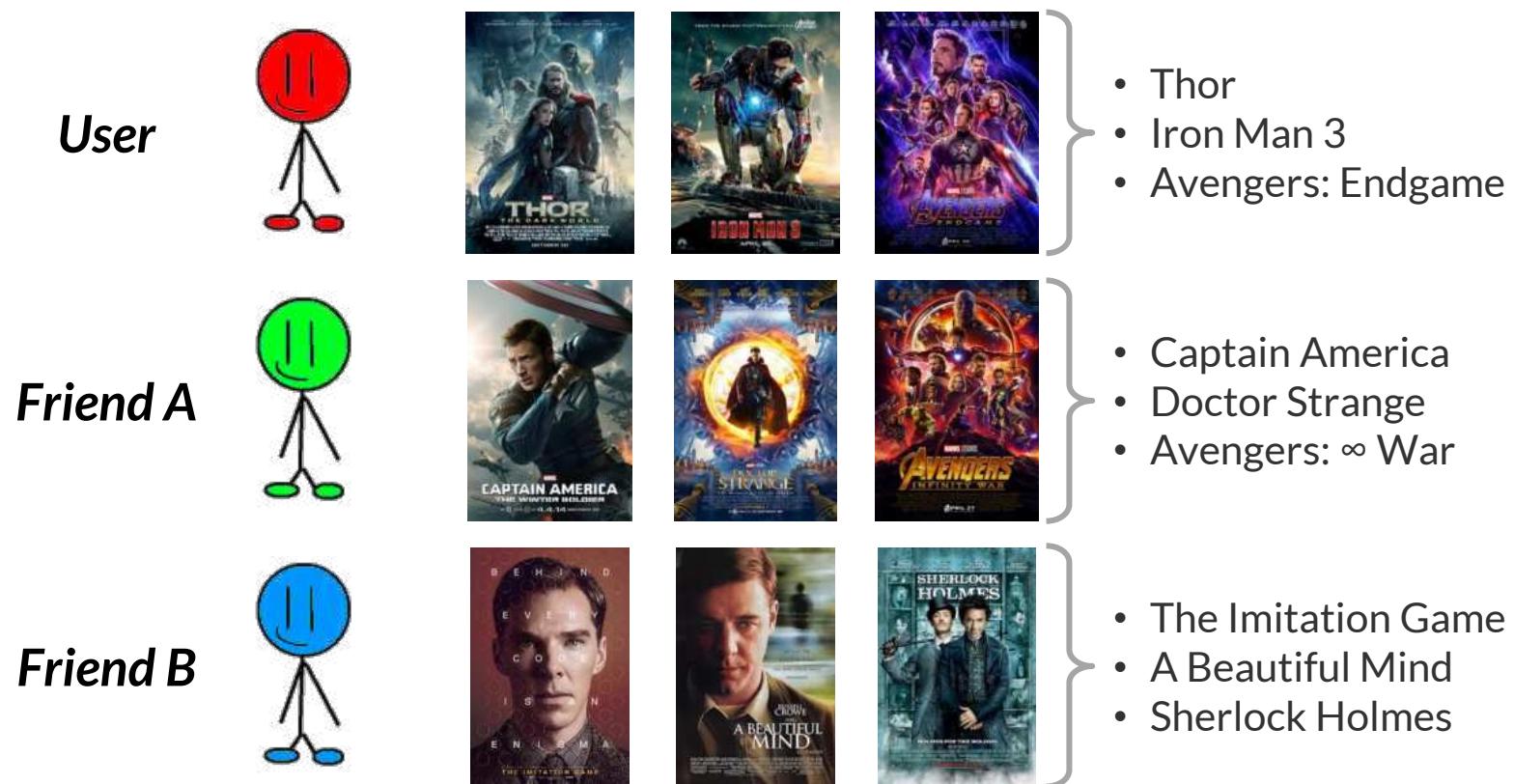
# Personalized Recommendations

- ▷ Collaborative Filtering (Concept)
- ▷ Memory-based Approach (Methodology)
- ▷ Model-based Approach (Methodology)



# What is Collaborative Filtering?

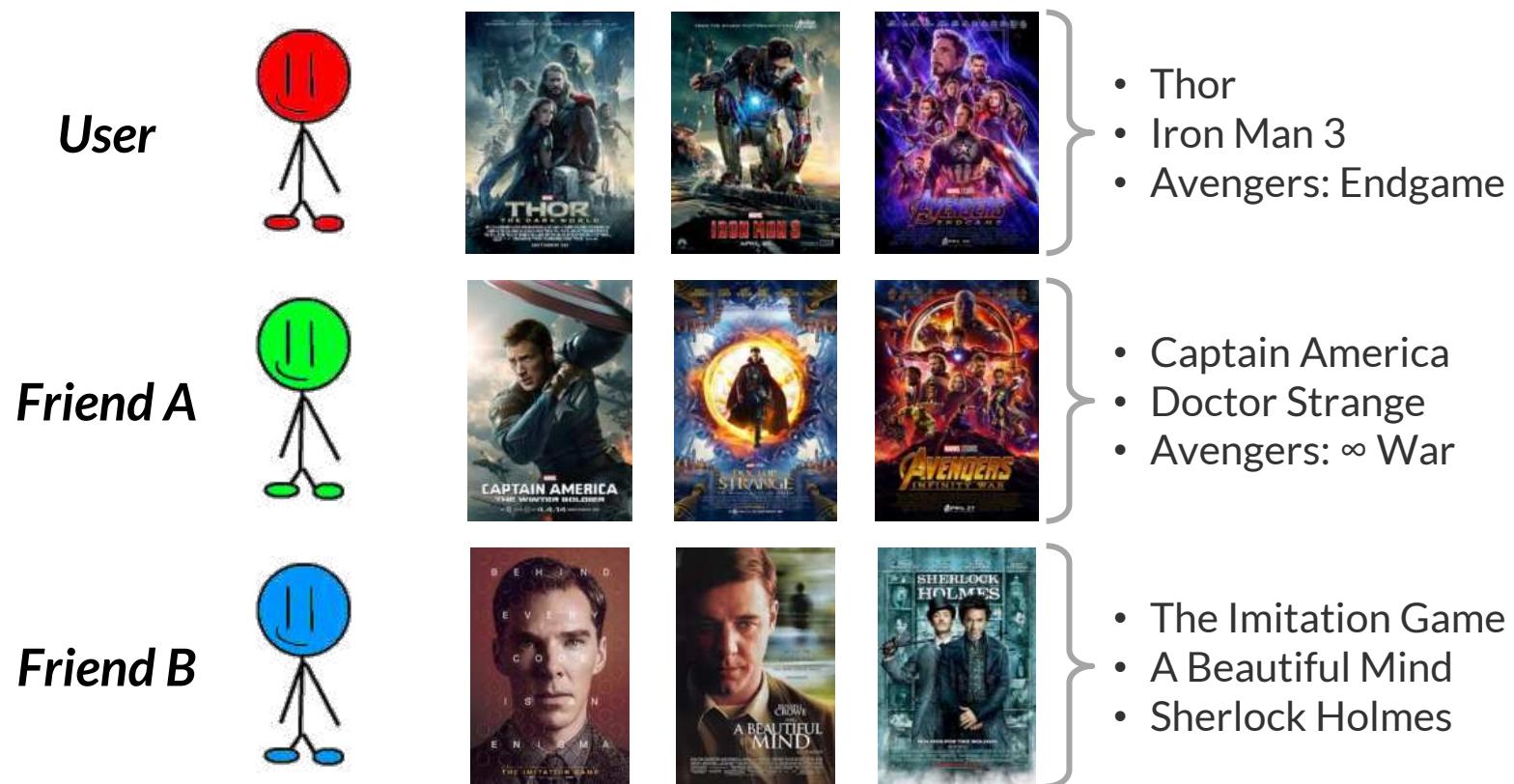
- ▷ Key Idea: People often get the best recommendations from someone with *similar tastes* as themselves!



- Friend A has similar preferences as the User
- Intuitively, Friend A's recommendations should be better than Friend B

# What is Collaborative Filtering?

- ▷ Key Idea: People often get the best recommendations from someone with *similar tastes* as themselves!



- The **filtering** (recommending a subset of items) is performed with the help of other similar users (i.e., in a **collaborative** manner)

# What is Collaborative Filtering?

- ▷ Key Idea: People often get the best recommendations from someone with *similar tastes* as themselves!
- ▷ Relies (mostly) on the historical interactions
  - Of the *target user*
  - As well as those of other similar users!
- ▷ Not something new
  - Has been proposed a long time ago
  - However, it's still a very powerful tool
  - Most (if not all) recommendation systems are still based on this concept!

# Memory-based Approaches

## ▷ Memory-based Approach (*Methodology*)

- Based on similarities between users (or items)
- Leverages upon the *historical interactions* of the most similar users (or items) to perform recommendations



# Memory-based Approaches

- ▷ Memory-based Approach (*Methodology*)
  - Based on similarities between users (or items)
  - Leverages upon the *historical interactions* of the most similar users (or items) to perform recommendations
- ▷ Similarity Measures
  - Jaccard Similarity
  - Cosine Similarity
  - Pearson Correlation
  - ...
- ▷ Examples
  - User-based k-Nearest Neighbour (**UserKNN**)
  - Item-based k-Nearest Neighbour (**ItemKNN**)

# Memory-based Approaches

## ▷ Benefits

- Intuitive & Explainable
- *"We are recommending Movie X to you because other similar users (who have also watched Movies Y and Z) liked Movie X as well"*

## ▷ Drawbacks

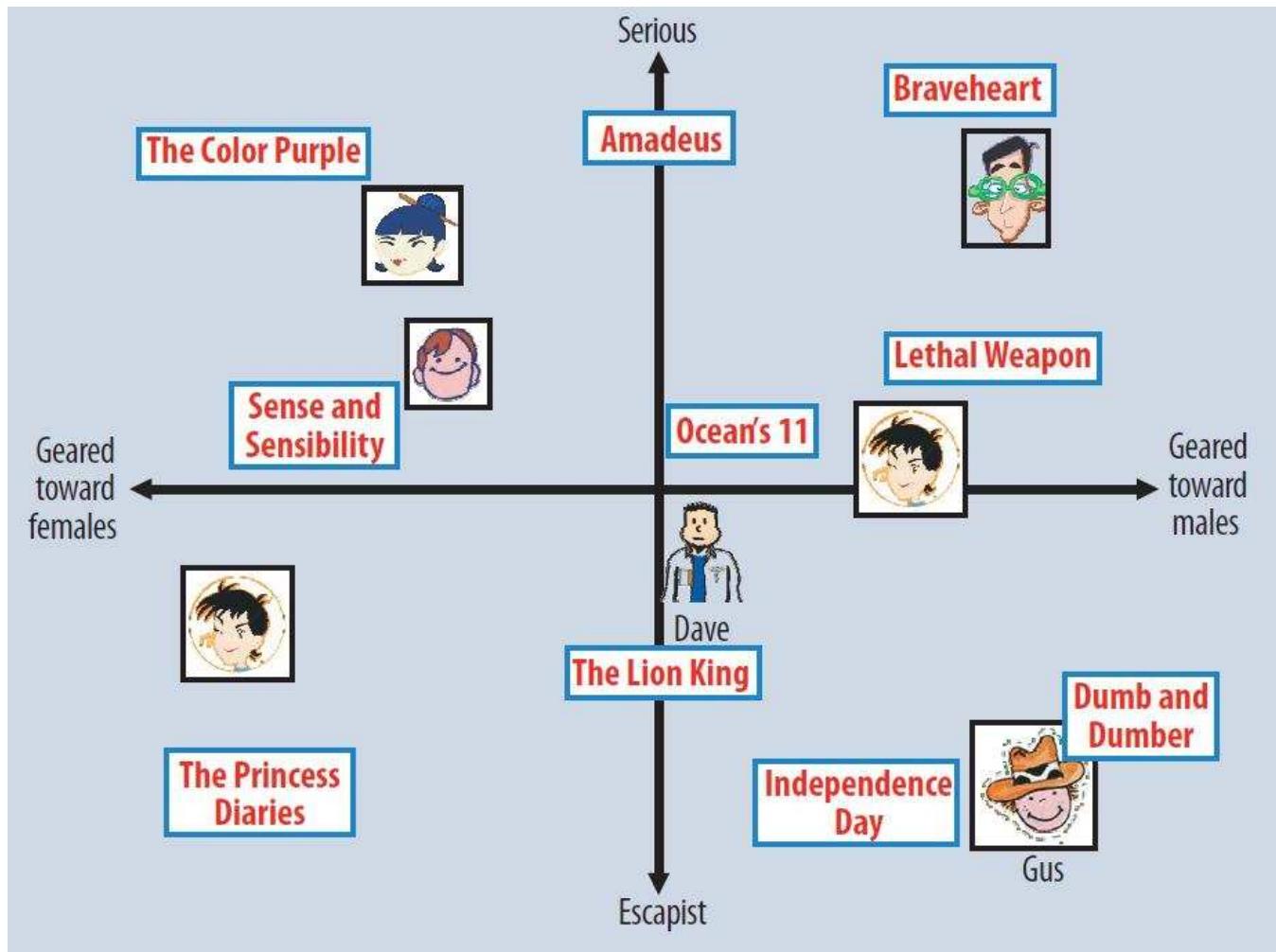
- Cannot handle sparse data well
  - E.g., If most users only interact with small number of items, it would be challenging (or maybe impossible) to find similar users
- Scalability
  - Less computational efficient when the number of users and items grow

# Model-based Approaches

- ▷ Model-based Approach (Methodology)
  - Models are built using different data mining (and more recently, machine / deep learning) techniques
- ▷ Examples
  - Bayesian Networks
  - Clustering-based Models
  - *Latent Factor Models (LFMs)* 
    - Matrix Factorization (MF)
    - ...
  - ...



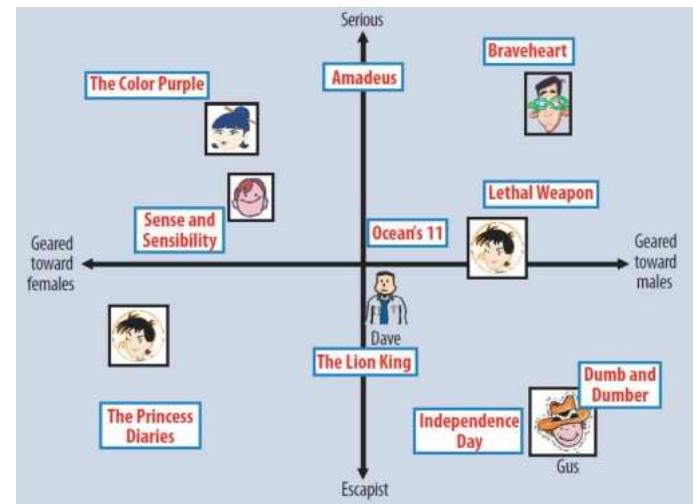
# Latent Factor Models



Users and items embedded in a 2-dimensional latent space (From [1])

# Latent Factor Models

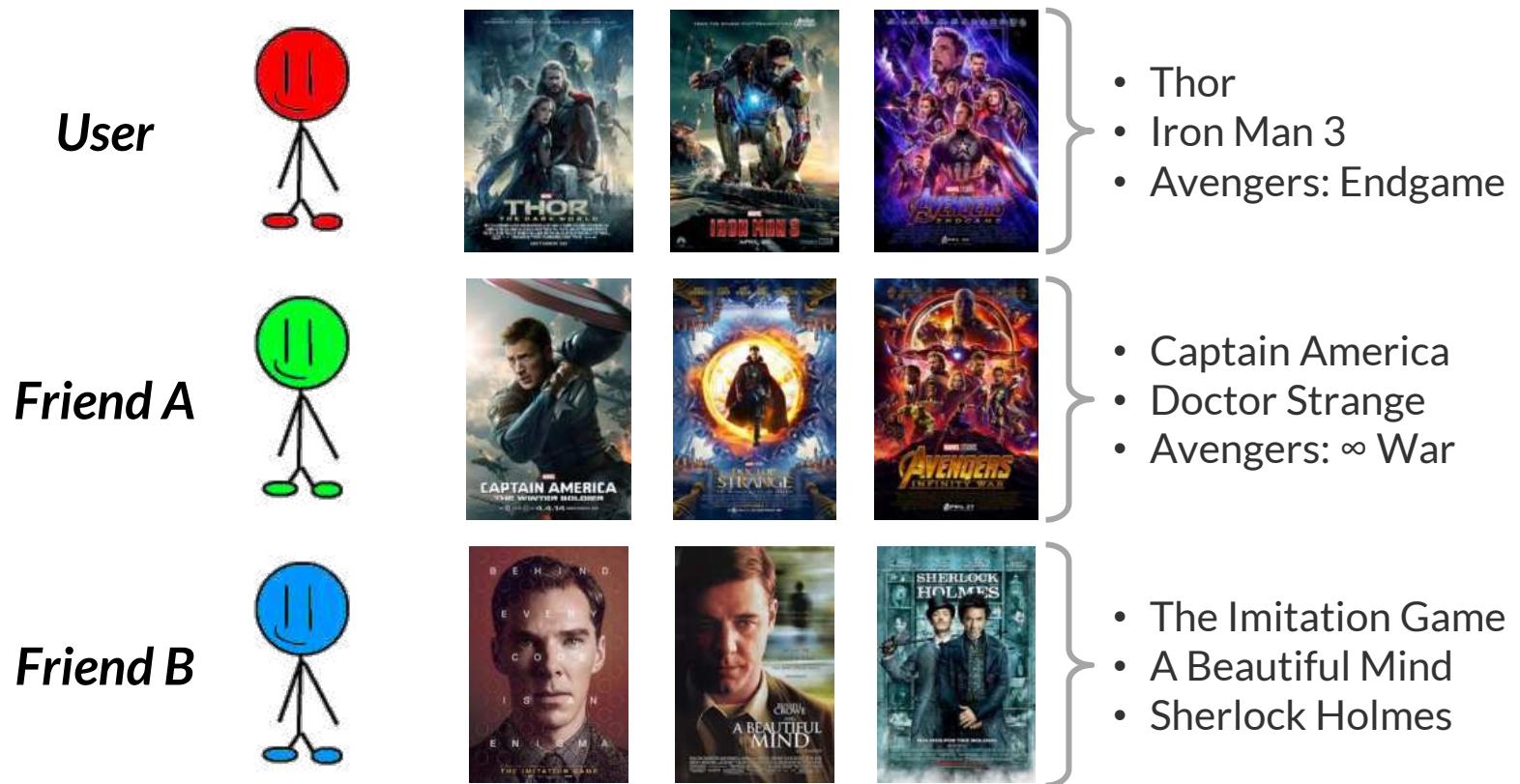
- ▷ Shared Latent Space
  - Each user is represented by a vector (of latent factors)
  - Each item is represented by a vector (of latent factors)
- ▷ Latent Factors
  - Learned automatically from historical interactions
  - These could be *movie genres, item attributes, ...*
  - But they are latent, so we do not really know their true meaning



\* **Latent:** hidden or concealed

# Latent Factor Models

- ▷ Similar users are embedded close to each other
- ▷ Similar items are embedded close to each other



- $Distance(\text{User}, \text{Friend A}) < Distance(\text{User}, \text{Friend B})$

# Latent Factor Models

- ▷ How to compute the user-item rating (or likelihood) using latent factors?
  - **Simple method:** Dot product

$$a \cdot b = \sum_{i=1}^n a_i b_i$$

$a$  = 1st vector

$b$  = 2nd vector

$n$  = dimension of the vector space

$a_i$  = component of vector a

$b_i$  = component of vector b

- **Alternatives:** Learning the user-item interaction function using *neural networks* (e.g. [1]), ...

# Model-based Approaches

## ▷ Benefits

- Works well even with sparse data
- Efficient and scalable
- *# of latent factors <<< # of users & # of items*
  - *User-item matrix is compressed into a low-dimensional latent space*
  - Dot product between two vectors is very fast

## ▷ Drawbacks

- Interpretability
  - Black box
  - What is the true meaning of the latent factors? ☹

# Hybrid Models

- ▷ E.g., Combination of memory-based & model-based
  - Overcomes the limitations of individual approaches
  - Could end up with increased complexity
- ▷ E.g., Training multiple models and combining their predictions



# 5.

## *Different types of recommendation systems*

# Different Types?

- ▷ Due to the different item types
  - E.g., Movies, News, Points of Interests (**POIs**), ...



- ▷ Due to the available information
  - **Content:** Text, Images, User demographics, Item attributes, ...
  - **Context:** Time, Location, ...



# POI Recommendation Systems

- ▷ In contrast to movies, books, etc., POIs are ‘physical items’
- ▷ When users interact with POIs, they are limited by *physical constraints*
  - Distance
    - Recommendations cannot be too far away
  - Time
    - Travel time required to reach recommended location
- ▷ As a result, bad recommendations could be somewhat more ‘costly’



# POI Recommendation Systems

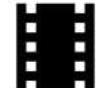
## ▷ Context-sensitive

- Distance
  - Current location (e.g., Home vs Workplace)
- Time
  - Weekday vs Weekend
  - Time of the day (Morning/Afternoon/Night)

## ▷ E.g., Workplace + Friday Evening → Restaurant / Bar



## ▷ E.g., Home + Saturday Evening → Cinema



## ▷ E.g., Home + Sunday Morning → Park



# POI Recommendation Systems

## ▷ Locality

- People's activities in urban environments are usually concentrated in certain geographical regions
- E.g., near our home or workplace

## ▷ Challenges for POI recommendation

- Very sparse data [1]
  - Density for **Netflix (Movies)**: 1.2%
  - Density for **Yelp & Foursquare (POIs)**: 0.1%
- Incorporating the rich context information adequately



# Group Recommendation Systems

- ▷ Recommending items to a group of users
  - A group of co-workers
  - A group of friends
  - A couple
  - A family
  - ...
- ▷ Users in a group tend to have...
  - Different preferences
  - Different levels of influence (Leaders vs Followers)
  - ...
- ▷ To be covered later... ☺



# 6.

*Challenges encountered by  
recommendation systems*

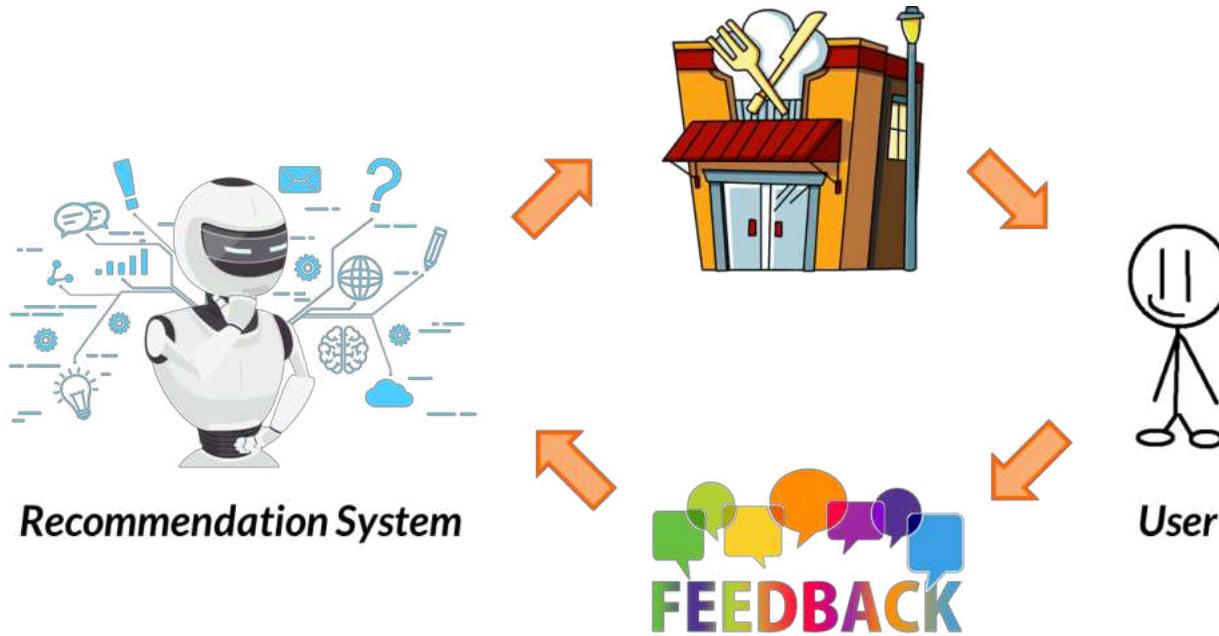
# Challenges

- ▷ Cold-start Problem
- ▷ Popularity Bias
- ▷ Shilling Attacks
- ▷ ...



# Cold-start Problem

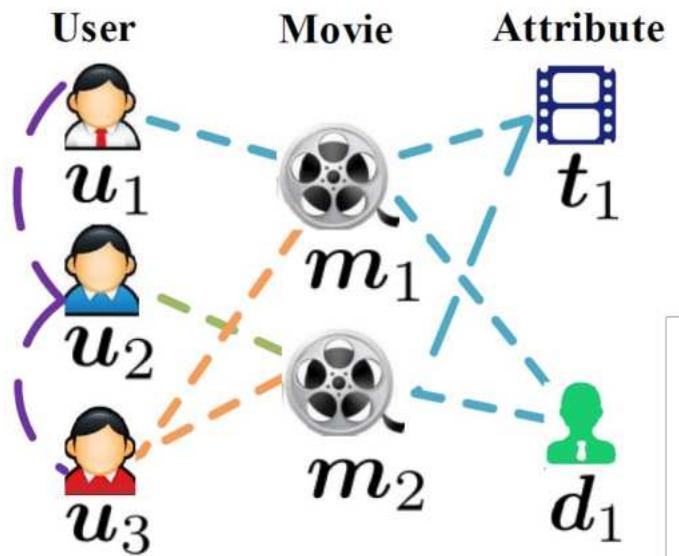
- ▷ Recommendation systems are driven by user interactions



- How about *new users or items* w/ no historical interactions?
- How about users or items w/ little historical interactions?
  - Difficult to *calculate similarity* and/or *derive a good representation*

# Cold-start Problem

- ▷ In some cases, we could utilize the content information
  - E.g., **User demographics** (Age, Gender, Occupation, ...)
  - E.g., **Item attributes** (Genre, Year, Actors, ...)



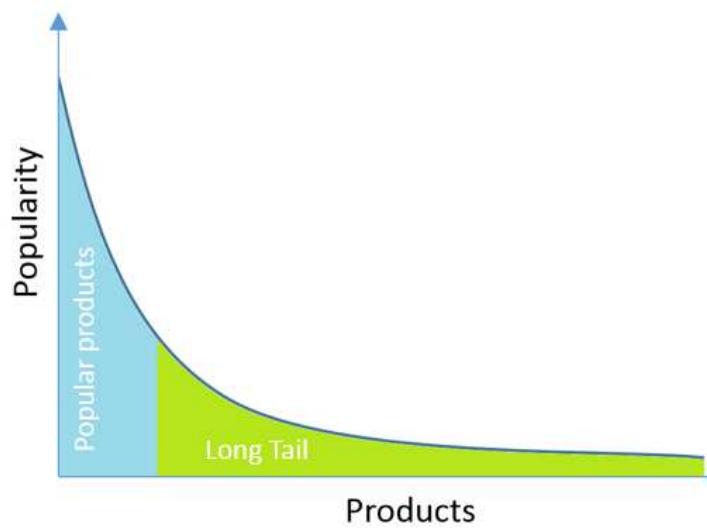
- ▷ Possible solutions
  - Content-based Filtering
  - Hybrid Model (Collaborative Filtering + Content)

# Popularity Bias

- ▷ Most recommendation systems suffer from *popularity bias*
  - Popular items tend to get recommended over and over again
  - Less popular (or niche) products are less likely to be discovered



Items w/ more Ratings

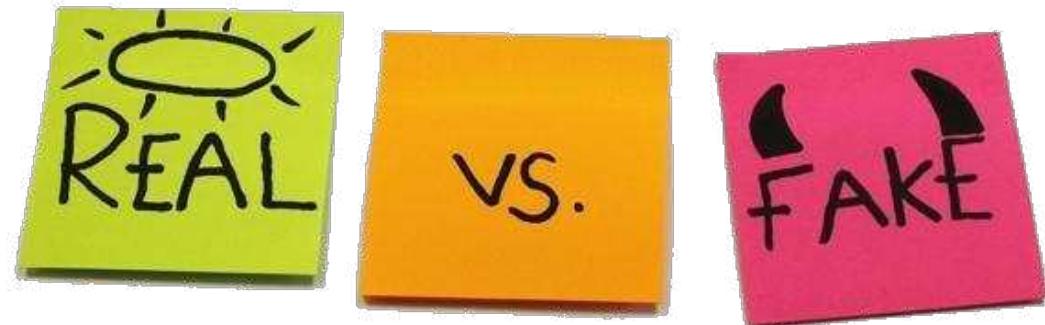


Items w/ less Ratings

# Shilling Attacks

## ▷ Collaborative Filtering

- “Users who shared similar preferences in the past will likely agree in the future as well”
- Based on “*wisdom of the crowd*”
  - i.e., what others think of an item or a group of items to compute recommendations
- Shortcoming: Unable to distinguish *genuine user profiles* from *fake ones*



# Shilling Attacks

- ▷ Characterised by...
  - Several *fake user profiles*
  - Often by an *adversarial party*
  - Harvest recommendation outcomes towards a *malicious desire*
- ▷ “Malicious Desire”
  - Personal gain, market penetration, causing mischief, etc.



# Robustness Against Shilling Attacks

- ▷ Existing work tends to cover *3 main directions*
  1. Attack Designs
  2. Detection Algorithms
  3. Defence Strategies



Thanks!  
Any questions?

Email:  
[S160005@e.ntu.edu.sg](mailto:S160005@e.ntu.edu.sg)

# Computation on Collaborative Filtering

slides adapted from Stanford data mining  
course

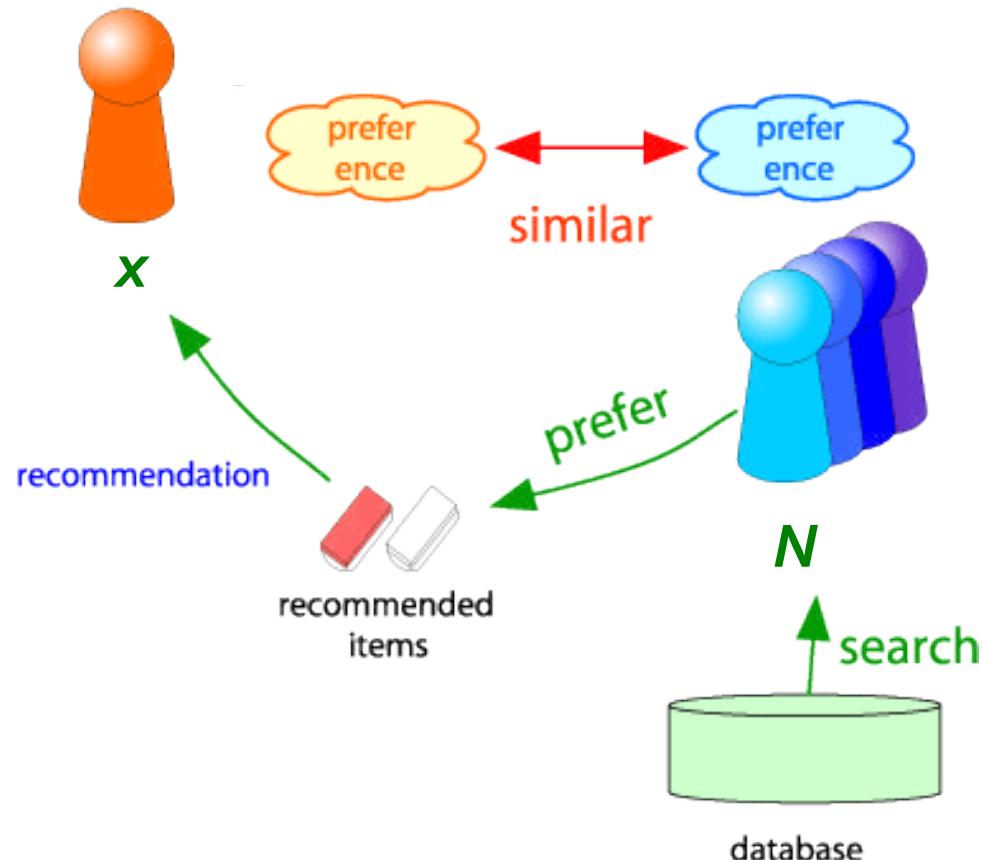
# Outline

---

- User based Collaborative Filtering (CF) and item based Collaborative Filtering (CF)
- Latent factor approach—Matrix Factorization

# User based Collaborative Filtering

- Consider user  $x$
- Find set  $N$  of other users whose ratings are “similar” to  $x$ 's ratings
- Estimate  $x$ 's ratings based on ratings of users in  $N$



# Finding “Similar” Users

$$\begin{aligned}r_x &= [* , \_, \_, *, **] \\r_y &= [* , \_, **, **, \_]\end{aligned}$$

- Let  $r_x$  be the vector of user  $x$ 's ratings
- Jaccard similarity measure**
  - Problem:** Ignores the value of the rating
- Cosine similarity measure**

$$\begin{aligned}r_x, r_y \text{ as sets:} \\r_x &= \{1, 4, 5\} \\r_y &= \{1, 3, 4\}\end{aligned}$$

- $\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{\|r_x\| \cdot \|r_y\|}$   
**Problem:** Treats missing ratings as “negative”
- Pearson correlation coefficient**

- $S_{xy}$  = items rated by both users  $x$  and  $y$

$$\text{sim}(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)(r_{ys} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \bar{r}_y)^2}}$$

$\bar{r}_x, \bar{r}_y \dots$  avg.  
rating of  $x, y$

$$\begin{aligned}r_x, r_y \text{ as points:} \\r_x &= \{1, 0, 0, 1, 3\} \\r_y &= \{1, 0, 2, 2, 0\}\end{aligned}$$

# Similarity Metric

**Cosine sim:**  
$$sim(x, y) = \frac{\sum_i r_{xi} \cdot r_{yi}}{\sqrt{\sum_i r_{xi}^2} \cdot \sqrt{\sum_i r_{yi}^2}}$$

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- **Intuitively we want:**  $sim(A, B) > sim(A, C)$
- **Jaccard similarity:**  $1/5 < 2/4$
- **Cosine similarity:**  $0.38 > 0.32$

# Rating Predictions

## From similarity metric to recommendations:

- Let  $r_x$  be the vector of user  $x$ 's ratings
- Let  $N$  be the set of  $k$  users most similar to  $x$  who have rated item  $i$
- Prediction for item  $i$  of user  $x$ :**

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$

Shorthand:  
 $s_{xy} = sim(x, y)$

- Many other options/tricks possible...**

# Item-based Collaborative Filtering

- So far: User-based collaborative filtering
- Another view: Item-based
  - For item  $i$ , find other similar items
  - Estimate rating for item  $i$  based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

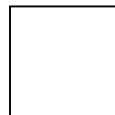
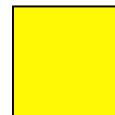
$s_{ij}$ ... similarity of items  $i$  and  $j$

$r_{xj}$ ... rating of user  $x$  on item  $j$

$N(i;x)$ ... set items rated by  $x$  similar to  $i$

# Item-based CF ( $|N|=2$ )

	users											
	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

 - unknown rating     - rating between 1 to 5

# Item-based CF ( $|N|=2$ )

	users											
	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

 - estimate rating of movie 1 by user 5

# Item-based CF ( $|N|=2$ )

	users												
	1	2	3	4	5	6	7	8	9	10	11	12	$\text{sim}(1,m)$
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.96
3	2	4		1	2		3		4	3	5		0.66
4		2	4		5			4			2		-0.84
5			4	3	4	2					2	5	-0.89
6	1		3		3			2			4		0.77

## Neighbor selection:

Identify movies similar to movie 1, rated by user 5

Here we use Pearson correlation as similarity:

1) Subtract mean rating  $m_i$  from each movie  $i$

$$m_1 = (1+3+5+5+4)/5 = 3.6$$

row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

2) Compute similarities between rows

# Item-based CF ( $|N|=2$ )

	users												
	1	2	3	4	5	6	7	8	9	10	11	12	$\text{sim}(1,m)$
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.96
3	2	4		1	2		3		4	3	5		0.66
4		2	4		5			4			2		-0.84
5			4	3	4	2					2	5	-0.89
6	1		3		3			2			4		0.77

Compute similarity weights:

$$s_{1,3}=0.66, s_{1,6}=0.77$$

# Item-based CF ( $|N|=2$ )

	users											
	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		2.6	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

$$r_{1,5} = (0.66 \cdot 2 + 0.77 \cdot 3) / (0.66 + 0.77) = 2.54$$

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

# Item-based vs. User-based

- In practice, it has been observed that item-based often works better than user-based
- Why? Items are simpler, users have multiple tastes

# Pros/Cons of Collaborative Filtering

- + Works for any kind of item
  - No feature selection needed
- - Cold Start:
  - Need enough users in the system to find a match
- - Sparsity:
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- - First rater:
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items
- - Popularity bias:
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items

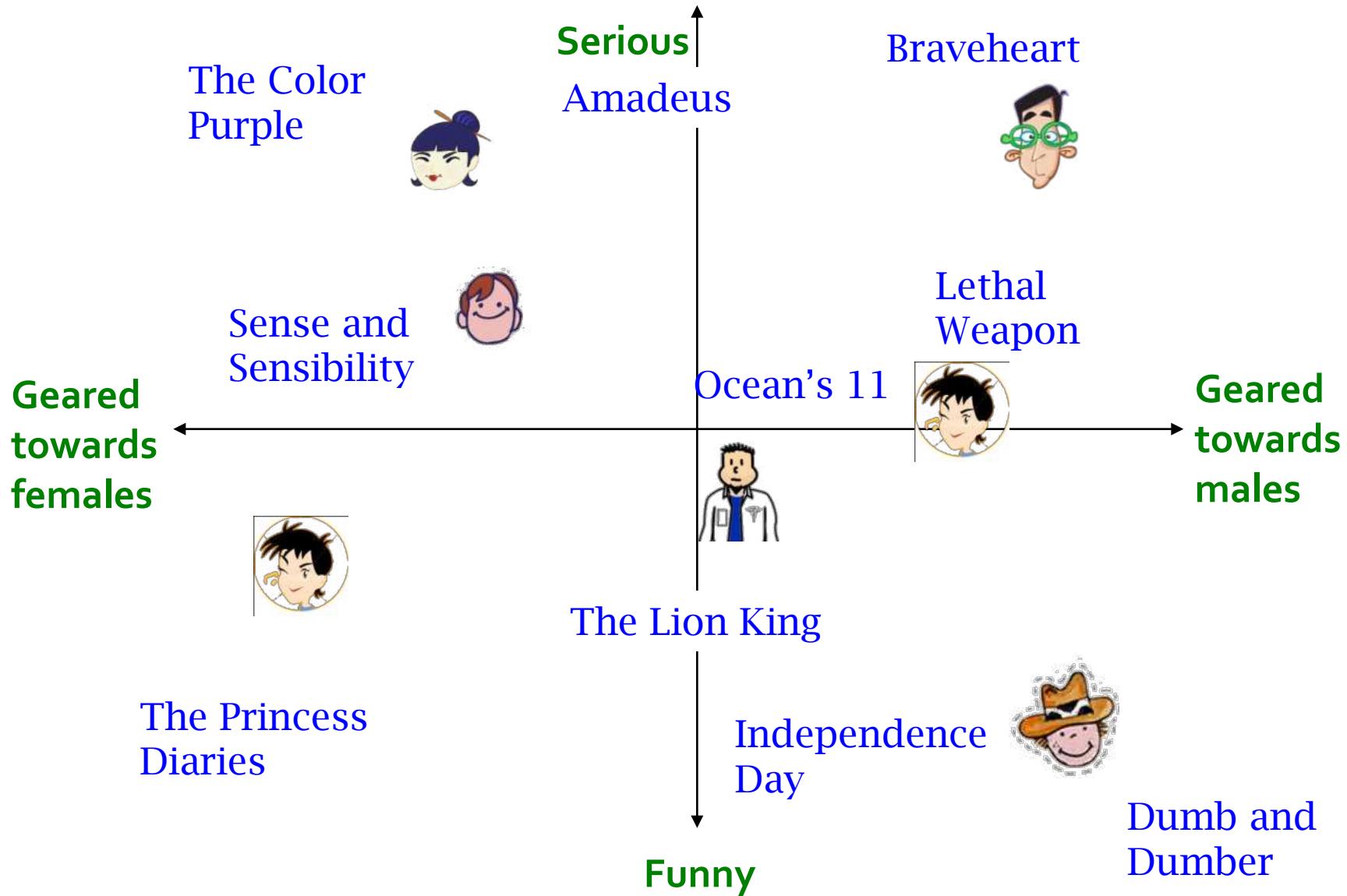
# Collaborative Filtering: Complexity

- Expensive step is finding  $k$  most similar customers:  $\mathbf{O}(|X|)$
- **Too expensive to do at runtime**
  - Could pre-compute
- Naïve pre-computation takes time  $\mathbf{O}(k \cdot |X|)$ 
  - $X$  ... set of customers
- **We already know how to do this!**
  - Near-neighbor search in high dimensions (**LSH**)
  - Clustering
  - Dimensionality reduction

# Outline

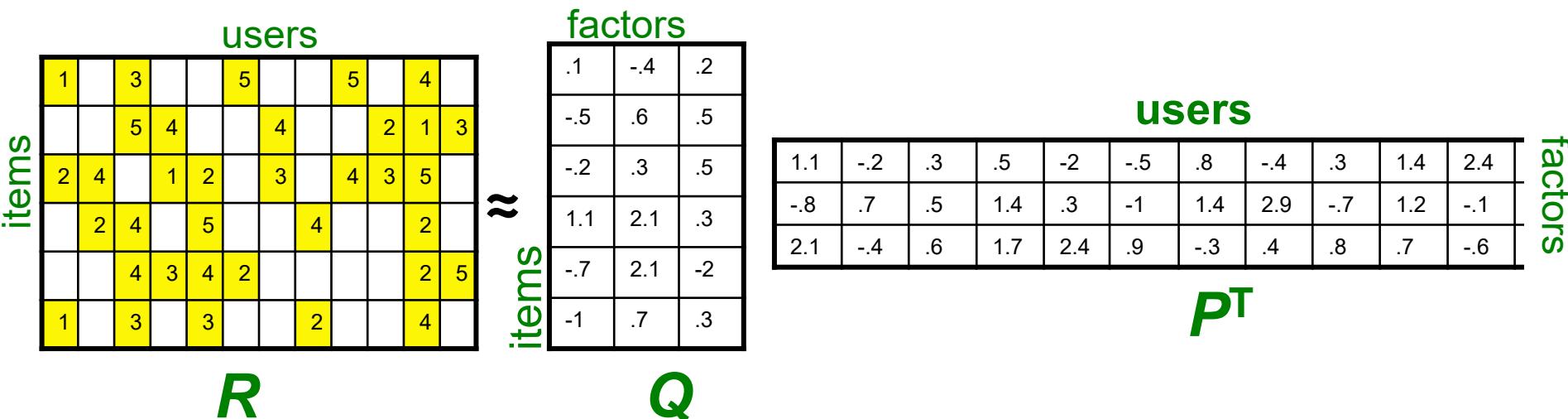
- User CF and item CF
- Latent factor approach—Matrix Factorization

# Latent Factor Models



# Latent Factor Models

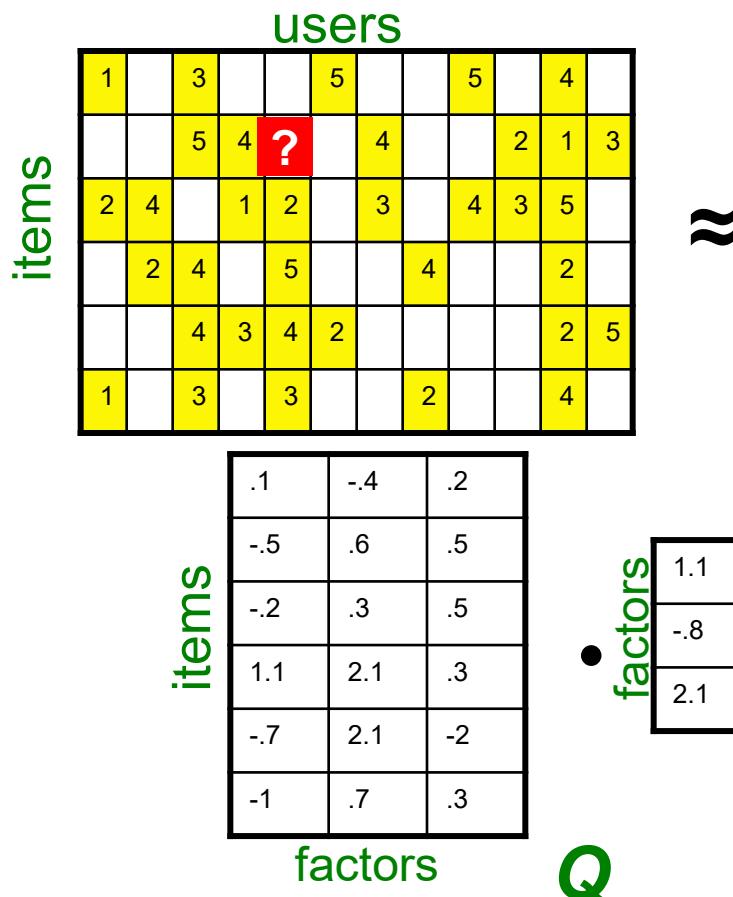
- Matrix Factorization:  $\mathbf{R} \approx \mathbf{Q} \cdot \mathbf{P}^T$



- For now let's assume we can approximate the rating matrix  $R$  as a product of "thin"  $Q \cdot P^T$ 
  - $R$  has missing entries but let's ignore that for now!
    - Basically, we will want the reconstruction error to be small on known ratings and we don't care about the values on the missing ones

# Ratings as Products of Factors

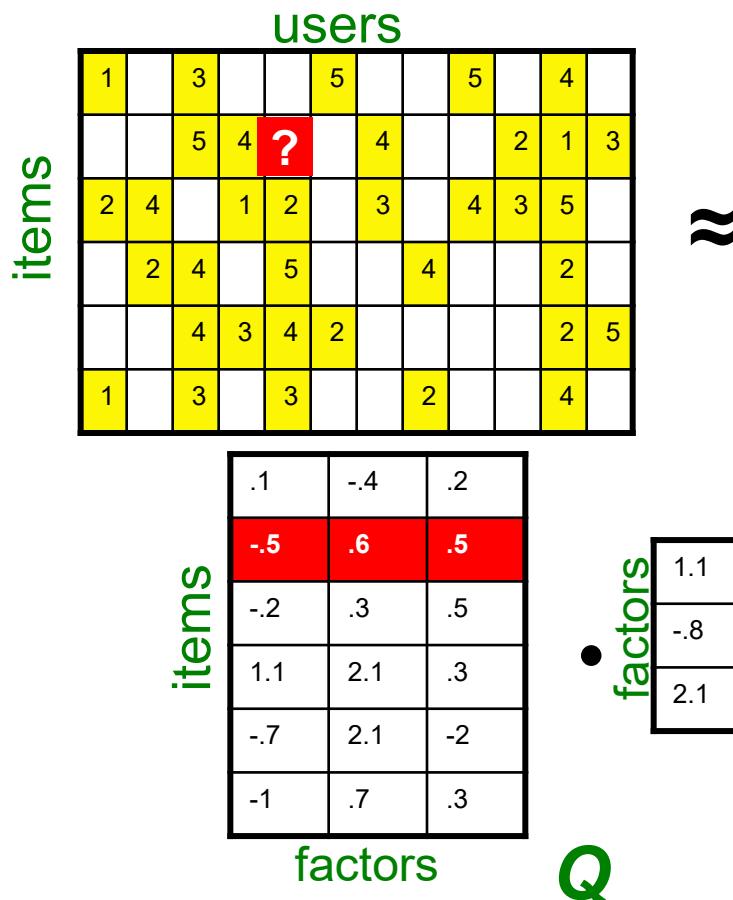
- How to estimate the missing rating of user  $x$  for item  $i$ ?



$q_i$  = row  $i$  of  $Q$   
 $p_x$  = column  $x$  of  $P^T$

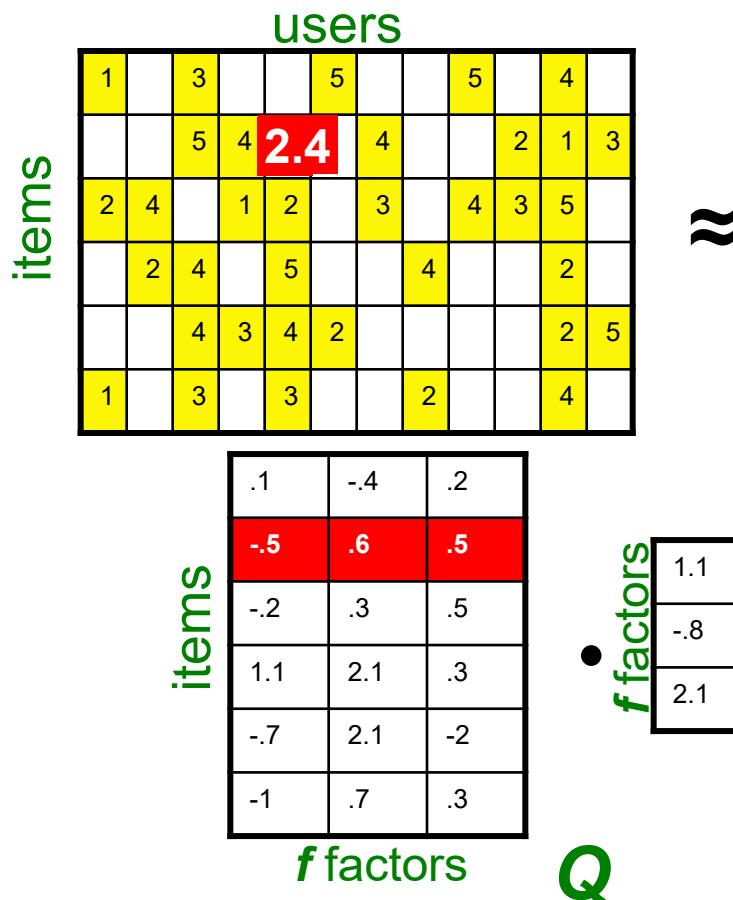
# Ratings as Products of Factors

- How to estimate the missing rating of user  $x$  for item  $i$ ?



# Ratings as Products of Factors

- How to estimate the missing rating of user  $x$  for item  $i$ ?



$$\begin{aligned}\hat{r}_{xi} &= q_i \cdot p_x \\ &= \sum_f q_{if} \cdot p_{xf} \\ q_i &= \text{row } i \text{ of } Q \\ p_x &= \text{column } x \text{ of } P^T\end{aligned}$$

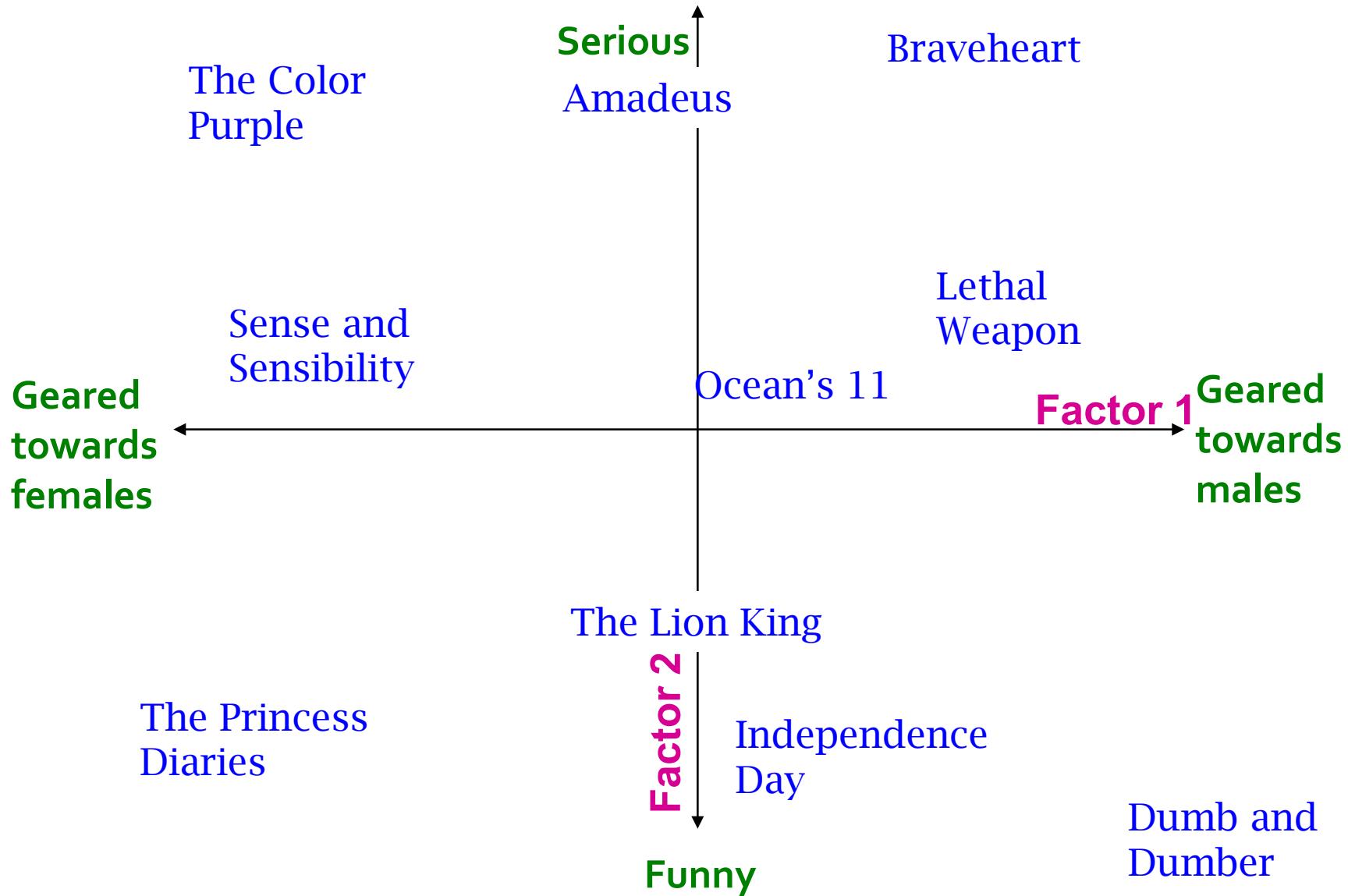
users

.1	-.4	.2										
<b>-.5</b>	<b>.6</b>	<b>.5</b>										
-.2	.3	.5										
1.1	2.1	.3										
-.7	2.1	-2										
-1	.7	.3										

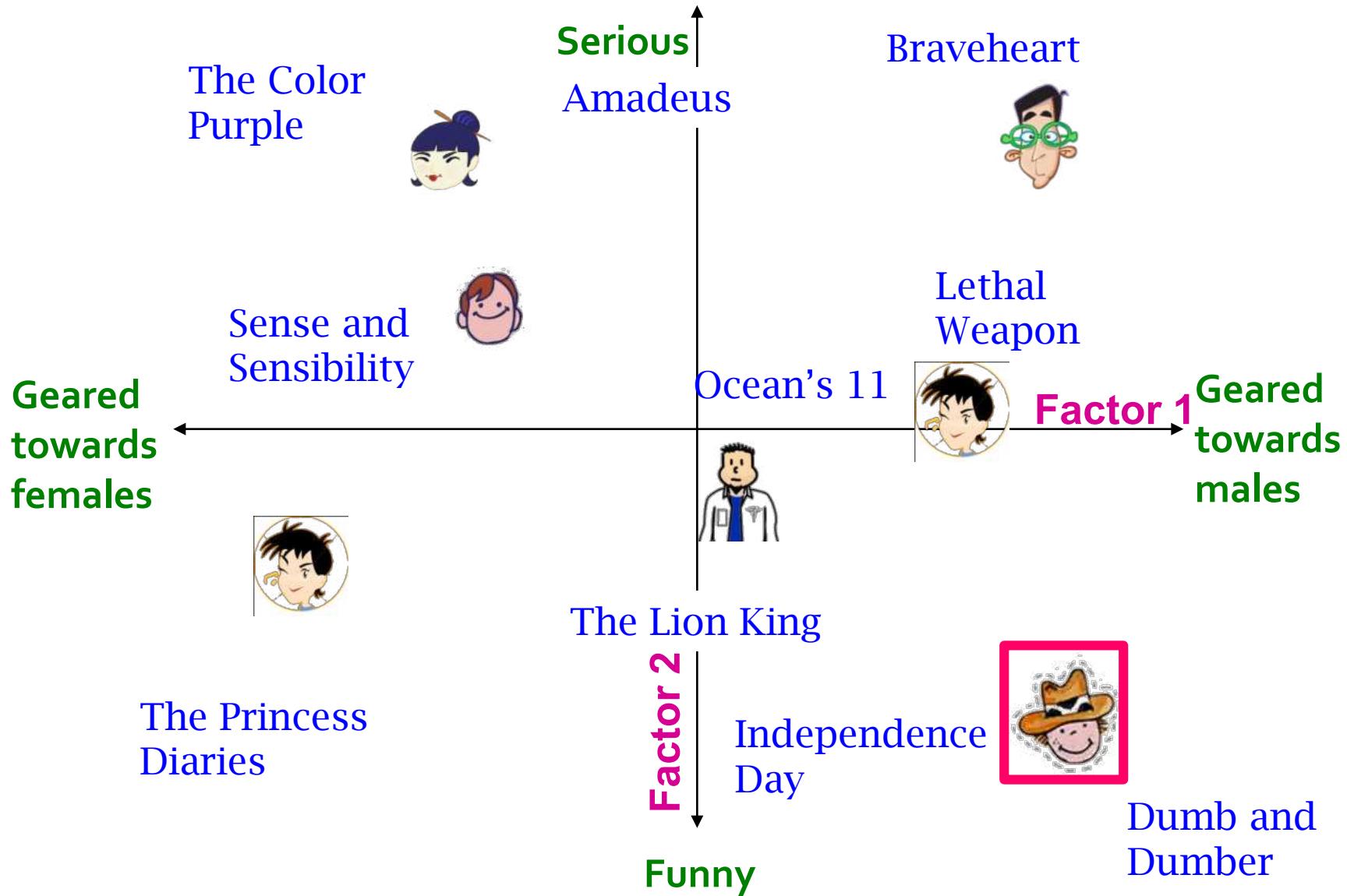
$P^T$

$f$  factors

# Latent Factor Models



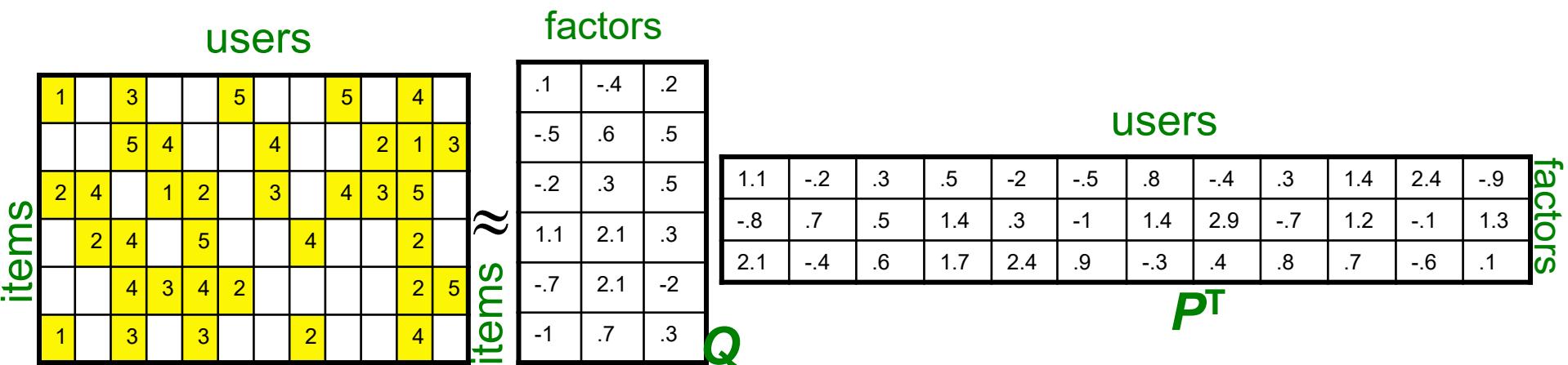
# Latent Factor Models



# Latent Factor Models

- Our goal is to find  $P$  and  $Q$  such that:

$$\min_{P,Q} \sum_{(i,x) \in R} (r_{xi} - q_i \cdot p_x)^2$$



Stochastic Gradient Descent

# Back to Our Problem

- Want to minimize Sum of Squared Errors (SSE) for unseen test data
- Idea: Minimize SSE on training data
  - Want large  $k$  (# of factors) to capture all the signals
  - But, SSE on test data begins to rise for  $k > 2$
- This is a classical example of **overfitting**:
  - With too much freedom (too many free parameters) the model starts fitting noise
    - That is it fits too well the training data and thus **not generalizing** well to unseen test data

1	3	4			
3	5		5		
4	5		5		
3					
3					
2		?	?	?	?
	2	1		?	?
	3		?		
1					

# Dealing with Missing Entries

- To solve overfitting we introduce regularization:

- Allow rich model where there are sufficient data
- Shrink aggressively where data are scarce

1	3	4		5
3	5			5
4	5			5
3				
3				
2		?	?	?
	2	1		?
3			?	
1				

$$\min_{P,Q} \underbrace{\sum_{\text{training}} (r_{xi} - q_i p_x)^2}_{\text{"error"}} + \left[ \underbrace{\lambda_1 \sum_x \|p_x\|^2}_{\text{"length"}^2} + \underbrace{\lambda_2 \sum_i \|q_i\|^2}_{\text{"length"}^2} \right]$$

$\lambda_1, \lambda_2 \dots$  user set regularization parameters

**Note:** We do not care about the “raw” value of the objective function, but we care in P,Q that achieve the minimum of the objective

# Summary

- User CF and item CF
  - Understand how to compute
- Latent factor approach—Matrix Factorization
  - Understand the high-level idea