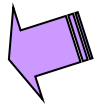
# Cluster Analysis: Advanced Methods

**Excerpt from**
**"Data Mining: Concepts and Techniques", 3rd Ed.**
**Jiawei Han, Micheline Kamber, and Jian Pei**
**Chapter 11**

# Outline of Advanced Clustering Analysis

- Probability Model-Based Clustering
  - Each object may take a probability to belong to a cluster
- Clustering High-Dimensional Data
  - Curse of dimensionality: Difficulty of distance measure in high-D space
- Clustering Graphs and Network Data
  - Similarity measurement and clustering methods for graph and networks
- Clustering with Constraints
  - Cluster analysis under different kinds of constraints, e.g., that raised from background knowledge or spatial distribution of the objects

# Chapter 11. Cluster Analysis: Advanced Methods



- Probability Model-Based Clustering
- Clustering High-Dimensional Data
- Clustering Graphs and Network Data
- Clustering with Constraints
- Summary

# Fuzzy Set and Fuzzy Cluster

- Clustering methods discussed so far
  - Every data object is assigned to exactly one cluster
- Some applications may need for fuzzy or soft cluster assignment
  - Ex. An e-game could belong to both entertainment and software
- Methods: fuzzy clusters and probabilistic model-based clusters
- Fuzzy cluster: A *fuzzy set S:* $F_S : X \rightarrow [0, 1]$ (value between 0 and 1)
- Example: Popularity of cameras is defined as a fuzzy mapping

| Camera | Sales (units) |
|---|---|
| $A$ | 50 |
| $B$ | 1320 |
| $C$ | 860 |
| $D$ | 270 |

$$\text{Pop}(o) = \begin{cases} 1 & \text{if } 1,000 \text{ or more units of } o \text{ are sold} \\ \frac{i}{1000} & \text{if } i \ (i < 1000) \text{ units of } o \text{ are sold} \end{cases}$$

- Then, *A*(0.05), *B*(1), *C*(0.86), *D*(0.27)

# Fuzzy (Soft) Clustering

$$w_{ij} = \frac{|R_i \cap C_j|}{|R_i \cap (C_1 \cup C_2)|} = \frac{|R_i \cap C_j|}{|R_i \cap \{digital\ camera, lens, computer\}|}$$

| Review-id | Keywords |
|---|---|
| $R_1$ | digital camera, lens |
| $R_2$ | digital camera |
| $R_3$ | lens |
| $R_4$ | digital camera, lens, computer |
| $R_5$ | computer, CPU |
| $R_6$ | computer, computer game |

$$M = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

- Example: Let cluster features be
  - $C_1$ :"digital camera" and "lens"
  - $C_2$: "computer"
- *Fuzzy clustering*
  - k fuzzy clusters $C_1, \ldots, C_k$ ,represented as a *partition matrix* $M = [w_{ij}]$
  - P1: for each object $o_i$ and cluster $C_j$, $0 \le w_{ij} \le 1$ (fuzzy set)
  - P2: for each object $o_i$, $\sum_{j=1}^{k} w_{ij} = 1$, equal participation in the clustering
  - P3: for each cluster $C_j$ , $0 < \sum_{i=1}^{n} w_{ij} < n$ ensures there is no empty cluster
- Let $c_1, \ldots, c_k$ as the center of the k clusters
- For an object $o_i$, sum of the squared error (SSE), p is a parameter:
- For a cluster $C_j$, SSE:

$$\mathrm{SSE}(o_i) = \sum_{j=1}^{k} w_{ij}^{p} dist(o_i, c_j)^2$$

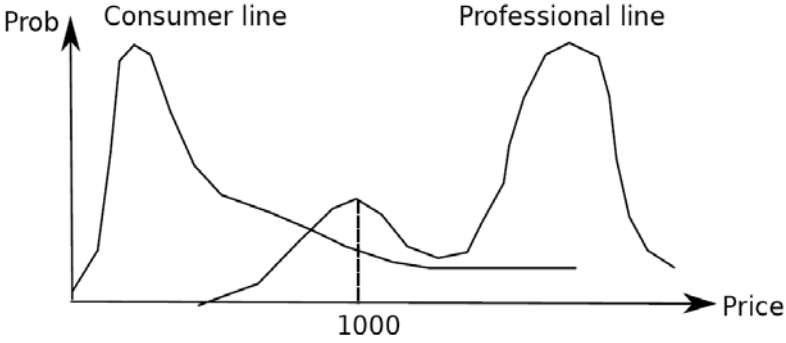
$$\mathrm{SSE}(C_j) = \sum_{i=1}^{n} w_{ij}^{p} dist(o_i, c_j)^2$$

- Measure how well a clustering fits the data:

$$\mathrm{SSE}(\mathcal{C}) = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij}^{p} dist(o_i, c_j)^2$$

# Probabilistic Model-Based Clustering

- Cluster analysis is to find hidden categories.
- A hidden category (i.e., *probabilistic cluster)* is a distribution over the data space, which can be mathematically represented using a probability density function (or distribution function).
- Ex. 2 categories for digital cameras sold
  - consumer line vs. professional line
  - density functions $f_1$, $f_2$ for $C_1$, $C_2$
  - obtained by probabilistic clustering




- A **mixture model** assumes that a set of observed objects is a mixture of instances from multiple probabilistic clusters, and conceptually each observed object is generated independently
- **Our task**: infer a set of *k* probabilistic clusters that is mostly likely to generate *D* using the above data generation process

# Model-Based Clustering

- A set $C$ of $k$ probabilistic clusters $C_1, \ldots, C_k$ with probability density functions $f_1, \ldots, f_k$, respectively, and their probabilities $\omega_1, \ldots, \omega_k$.
- Probability of an object $o$ generated by cluster $C_j$ is $P(o|C_j) = \omega_j f_j(o)$
- Probability of $o$ generated by the set of cluster $\boldsymbol{C}$ is $P(o|\boldsymbol{C}) = \sum_{j=1}^{k} \omega_j f_j(o)$
- Since objects are assumed to be generated independently, for a data set D = {$o_1$, …, $o_n$}, we have,

$$P(D|\boldsymbol{C}) = \prod_{i=1}^{n} P(o_i|\boldsymbol{C}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j f_j(o_i)$$

- Task: Find a set $C$ of $k$ probabilistic clusters s.t. $P(D|\boldsymbol{C})$ is maximized
- However, maximizing $P(D|\boldsymbol{C})$ is often intractable since the probability density function of a cluster can take an arbitrarily complicated form
- To make it computationally feasible (as a compromise), assume the probability density functions being some parameterized distributions

# Univariate Gaussian Mixture Model

- O = {$o_1$, …, $o_n$} (n observed objects), Θ = {$\theta_1$, …, $\theta_k$} (parameters of the k distributions), and $P_j(o_i|\theta_j)$ is the probability that $o_i$ is generated from the j-th distribution using parameter $\theta_j$, we have

$$P(o_i|\Theta) = \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j) \qquad P(\mathbf{O}|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j)$$

- Univariate Gaussian mixture model
  - Assume the probability density function of each cluster follows a Gaussian distribution. Suppose that there are k clusters.
  - The probability density function of each cluster are centered at $\mu_j$ with standard deviation $\sigma_j$, $\theta_j$, = ($\mu_j$, $\sigma_j$), we have

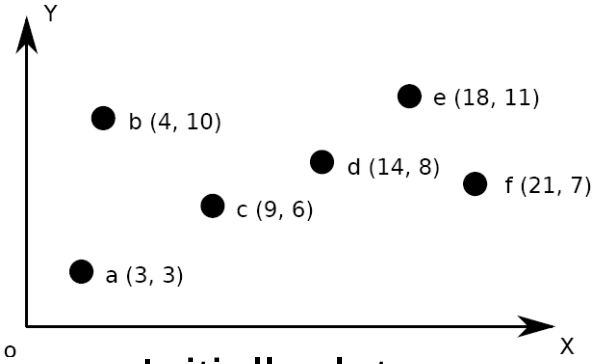
$$P(o_i|\Theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}} \qquad P(o_i|\Theta) = \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

$$P(\mathbf{O}|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

# The EM (Expectation Maximization) Algorithm

- The k-means algorithm has two steps at each iteration:
  - **Expectation Step** (E-step): Given the current cluster centers, each object is assigned to the cluster whose center is closest to the object: An object is *expected to belong to the closest cluster*
  - **Maximization Step** (M-step): Given the cluster assignment, for each cluster, the algorithm *adjusts the center* so that *the sum of distance* from the objects assigned to this cluster and the new center is minimized
- **The (EM) algorithm:** A framework to approach maximum likelihood or maximum a posteriori estimates of parameters in statistical models.
  - **E-step** assigns objects to clusters according to the current fuzzy clustering or parameters of probabilistic clusters
  - **M-step** finds the new clustering or parameters that minimizing the sum of squared error (SSE) or maximizing the expected likelihood in probabilistic model-based clustering

# Fuzzy Clustering Using the EM Algorithm




| Iteration | E-step | M-step |
|---|---|---|
| 1 | $M^T = \begin{bmatrix} 1 & 0 & 0.48 & 0.42 & 0.41 & 0.47 \\ 0 & 1 & 0.52 & 0.58 & 0.59 & 0.53 \end{bmatrix}$ | $c_1 = (8.47, 5.12)$, $c_2 = (10.42, 8.99)$ |
| 2 | $M^T = \begin{bmatrix} 0.73 & 0.49 & 0.91 & 0.26 & 0.33 & 0.42 \\ 0.27 & 0.51 & 0.09 & 0.74 & 0.67 & 0.58 \end{bmatrix}$ | $c_1 = (8.51, 6.11)$, $c_2 = (14.42, 8.69)$ |
| 3 | $M^T = \begin{bmatrix} 0.80 & 0.76 & 0.99 & 0.02 & 0.14 & 0.23 \\ 0.20 & 0.24 & 0.01 & 0.98 & 0.86 & 0.77 \end{bmatrix}$ | $c_1 = (6.40, 6.24)$, $c_2 = (16.55, 8.64)$ |

- Initially, let $c_1$ = a and $c_2$ = b
- 1st E-step: assign o to $c_1$,w. wt = $\frac{\frac{1}{dist(o,c_1)^2}}{\frac{1}{dist(o,c_1)^2} + \frac{1}{dist(o,c_2)^2}} = \frac{dist(o,c_2)^2}{dist(o,c_1)^2 + dist(o,c_2)^2}$
  - $w_{c,c_1} = \frac{41}{45+41} = 0.48$
- 1st M-step: recalculate the centroids according to the partition matrix, minimizing the sum of squared error (SSE)

$$c_j = \frac{\sum_{\text{each point } o} w_{o,c_j}^2 o}{\sum_{\text{each point } o} w_{o,c_j}^2}$$

$$c_1 = \left(\frac{1^2\times3+0^2\times4+0.48^2\times9+0.42^2\times14+0.41^2\times18+0.47^2\times21}{1^2+0^2+0.48^2+0.42^2+0.41^2+0.47^2}, \frac{1^2\times3+0^2\times10+0.48^2\times6+0.42^2\times8+0.41^2\times11+0.47^2\times7}{1^2+0^2+0.48^2+0.42^2+0.41^2+0.47^2}\right) = (8.47, 5.12)$$

- Iteratively calculate this until the cluster centers converge or the change is small enough

# Univariate Gaussian Mixture Model

- O = {$o_1$, …, $o_n$} (n observed objects), Θ = {$\theta_1$, …, $\theta_k$} (parameters of the k distributions), and $P_j(o_i|\theta_j)$ is the probability that $o_i$ is generated from the j-th distribution using parameter $\theta_j$, we have

$$P(o_i|\Theta) = \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j) \qquad P(\mathbf{O}|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j P_j(o_i|\Theta_j)$$

- Univariate Gaussian mixture model
  - Assume the probability density function of each cluster follows a 1-d Gaussian distribution. Suppose that there are k clusters.
  - The probability density function of each cluster are centered at $\mu_j$ with standard deviation $\sigma_j$, $\theta_j$, = ($\mu_j$, $\sigma_j$), we have

$$P(o_i|\Theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}} \qquad P(o_i|\Theta) = \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

$$P(\mathbf{O}|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

# Computing Mixture Models with EM

- Given n objects $O = \{o_1, \ldots, o_n\}$, we want to mine a set of parameters $\Theta = \{\theta_1, \ldots, \theta_k\}$ s.t.,$P(\mathbf{O}|\mathbf{\Theta})$ is maximized, where $\theta_j = (\mu_j, \sigma_j)$ are the mean and standard deviation of the j-th univariate Gaussian distribution
- We initially assign random values to parameters $\theta_j$, then iteratively conduct the E- and M- steps until converge or sufficiently small change
- At the E-step, for each object $o_i$, calculate the probability that $o_i$ belongs to each distribution,

$$P(\Theta_j|o_i, \mathbf{\Theta}) = \frac{P(o_i|\Theta_j)}{\sum_{l=1}^{k} P(o_i|\Theta_l)}$$

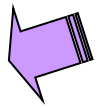
- At the M-step, adjust the parameters $\theta_j = (\mu_j, \sigma_j)$ so that the expected likelihood $P(\mathbf{O}|\mathbf{\Theta})$ is maximized

$$\mu_j = \sum_{i=1}^{n} o_i \frac{P(\Theta_j|o_i, \mathbf{\Theta})}{\sum_{l=1}^{n} P(\Theta_j|o_l, \mathbf{\Theta})} = \frac{\sum_{i=1}^{n} o_i P(\Theta_j|o_i, \mathbf{\Theta})}{\sum_{i=1}^{n} P(\Theta_j|o_i, \mathbf{\Theta})} \quad \sigma_j = \sqrt{\frac{\sum_{i=1}^{n} P(\Theta_j|o_i, \mathbf{\Theta})(o_i - u_j)^2}{\sum_{i=1}^{n} P(\Theta_j|o_i, \mathbf{\Theta})}}$$

# Advantages and Disadvantages of Mixture Models

- Strength
  - Mixture models are more general than partitioning and fuzzy clustering
  - Clusters can be characterized by a small number of parameters
  - The results may satisfy the statistical assumptions of the generative models
- Weakness
  - Converge to local optimal (overcome: run multi-times w. random initialization)
  - Computationally expensive if the number of distributions is large, or the data set contains very few observed data points
  - Need large data sets
  - Hard to estimate the number of clusters

# Chapter 11. Cluster Analysis: Advanced Methods

- Probability Model-Based Clustering
- Clustering High-Dimensional Data 
- Clustering Graphs and Network Data
- Clustering with Constraints
- Summary

# Clustering High-Dimensional Data

- Clustering high-dimensional data (How high is high-D in clustering?)
  - Many applications: text documents, DNA micro-array data
  - Major challenges:
    - Many irrelevant dimensions may mask clusters
    - Distance measure becomes meaningless—due to equi-distance
    - Clusters may exist only in some subspaces
- Methods
  - **Subspace-clustering**: Search for clusters existing in subspaces of the given high dimensional data space
    - CLIQUE, ProClus, and bi-clustering approaches
  - **Dimensionality reduction approaches**: Construct a much lower dimensional space and search for clusters there (may construct new dimensions by combining some dimensions in the original data)
    - Dimensionality reduction methods and spectral clustering

# Traditional Distance Measures May Not Be Effective on High-D Data

- Traditional distance measure could be dominated by noises in many dimensions
- Ex. Which pairs of customers are more similar?

| Customer | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ | $P_7$ | $P_8$ | $P_9$ | $P_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Ada | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bob | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Cathy | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

- By Euclidean distance, we get,

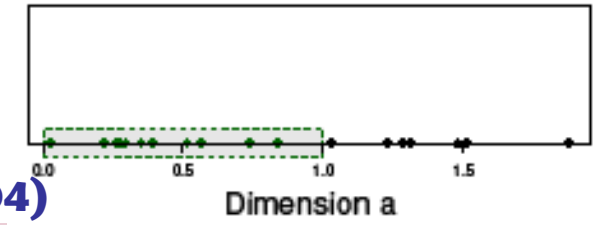
$$dist(\text{Ada}, \text{Bob}) = dist(\text{Bob}, \text{Cathy}) = dist(\text{Ada}, \text{Cathy}) = \sqrt{2}$$

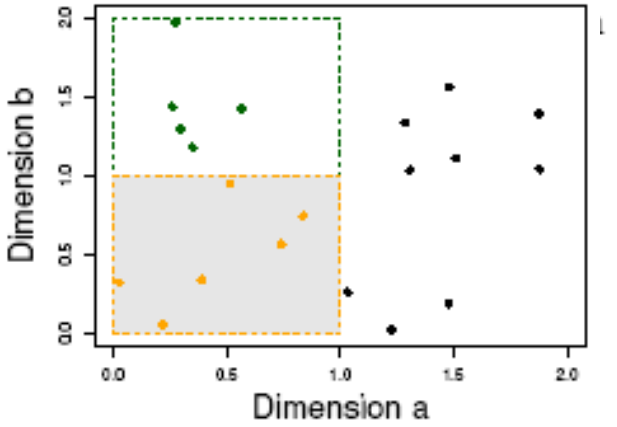
  - despite Ada and Bob being different but similar distance from Cathy
- Clustering should not only consider dimensions but also attributes (features)
  - Feature transformation: effective if most dimensions are relevant (PCA & SVD useful when features are highly correlated/redundant)
  - Feature selection: useful to find a subspace where the data have nice clusters

# The Curse of Dimensionality

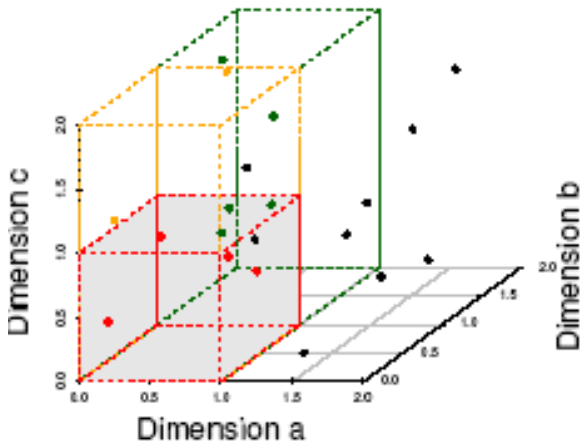
**(graphs adapted from Parsons et al. KDD Explorations 2004)**

- Data in only one dimension is relatively packed
- Adding a dimension "stretch" the points across that dimension, making them further apart
- Adding more dimensions will make the points further apart—high dimensional data is extremely sparse
- Distance measure becomes meaningless—due to equi-distance

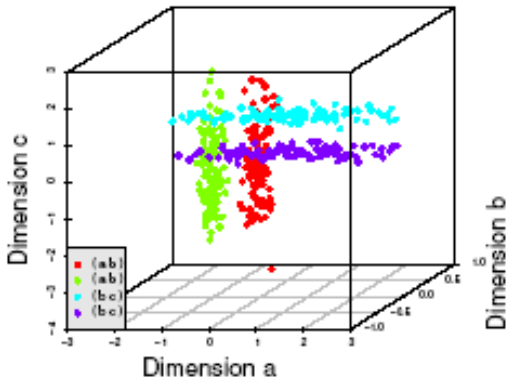



(b) 6 Objects in One Unit Bin



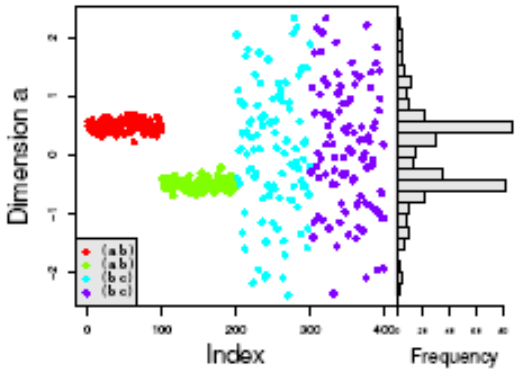

(c) 4 Objects in One Unit Bin



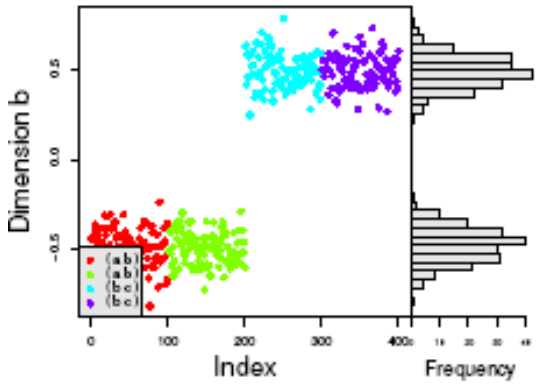

# Why Subspace Clustering?

## (adapted from Parsons et al. SIGKDD Explorations 2004)

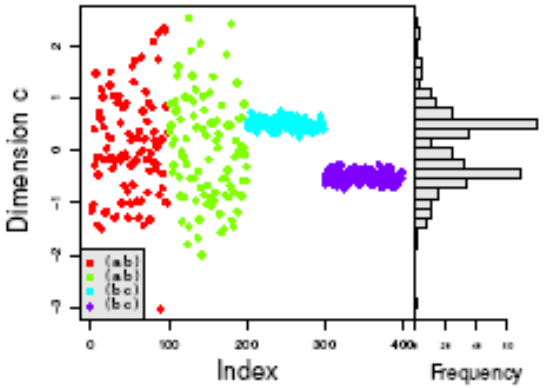
- Clusters may exist only in some subspaces
- Subspace-clustering: find clusters in all the subspaces



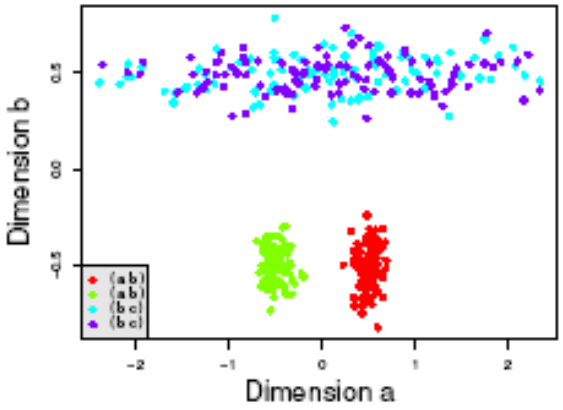

(a) Dimension $a$



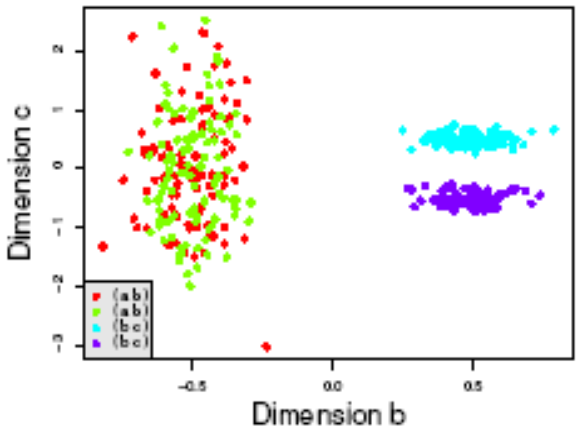

(b) Dimension $b$



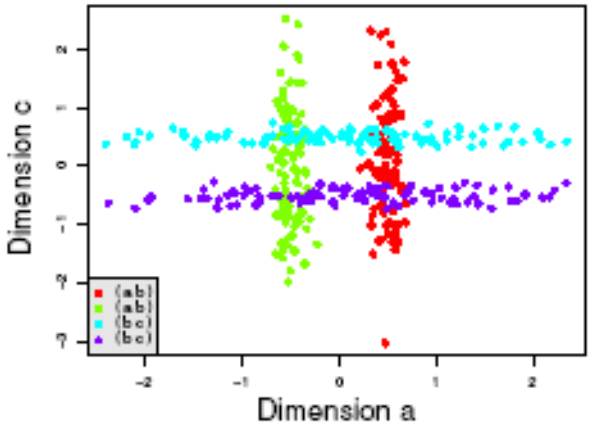

(c) Dimension $c$




(a) Dims $a$ & $b$




(b) Dims $b$ & $c$




(c) Dims $a$ & $c$

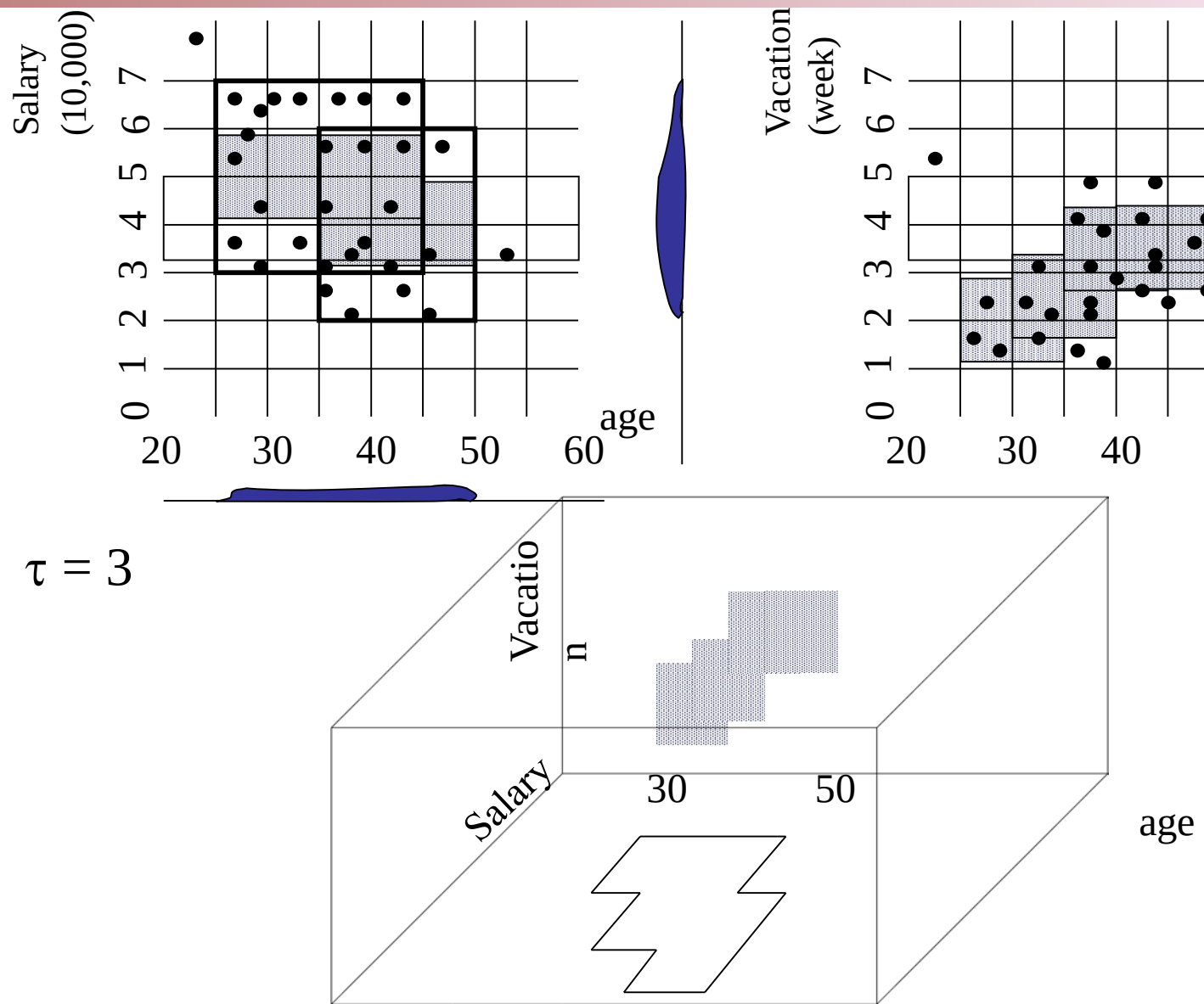
# Subspace Clustering Methods

- Subspace search methods: Search various subspaces to find clusters
  - Bottom-up approaches
  - Top-down approaches
- Correlation-based clustering methods
  - E.g., PCA based approaches
- Bi-clustering methods
  - Optimization-based methods
  - Enumeration methods

# Subspace Clustering Method (I): Subspace Search Methods

- Search various subspaces to find clusters
- *Bottom-up approaches*
  - Start from low-D subspaces and search higher-D subspaces only when there may be clusters in such subspaces
  - Various pruning techniques to reduce the number of higher-D subspaces to be searched
  - Ex. CLIQUE (Agrawal et al. 1998)
- *Top-down approaches*
  - Start from full space and search smaller subspaces recursively
  - Effective only if the *locality assumption* holds: restricts that the subspace of a cluster can be determined by the local neighborhood
  - Ex. PROCLUS (Aggarwal et al. 1999): a *k*-medoid-like method

# CLIQUE: SubSpace Clustering with Aprori Pruning

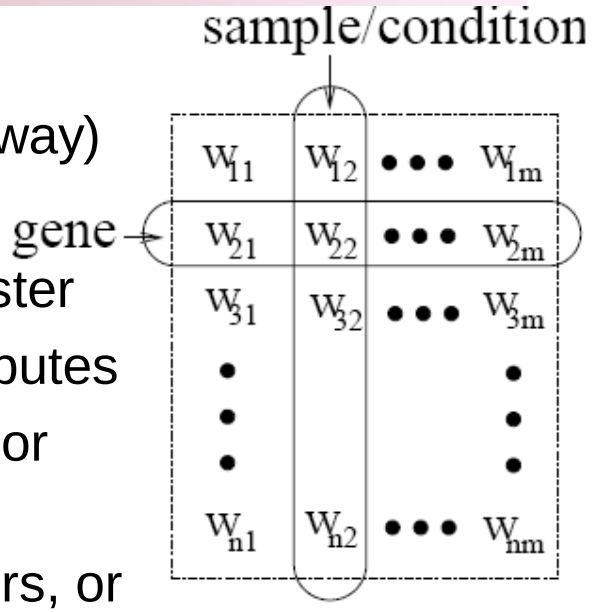


# Subspace Clustering Method (II): Correlation-Based Methods

- **Subspace search method**: similarity based on distance or density
- **Correlation-based method**: based on advanced correlation models
- Ex. PCA-based approach:
  - Apply PCA (for Principal Component Analysis) to derive a set of new, uncorrelated dimensions;
  - then mine clusters in the new space or its subspaces
- Other space transformations:
  - Hough transform
  - Fractal dimensions

# Subspace Clustering Method (III): Bi-Clustering Methods

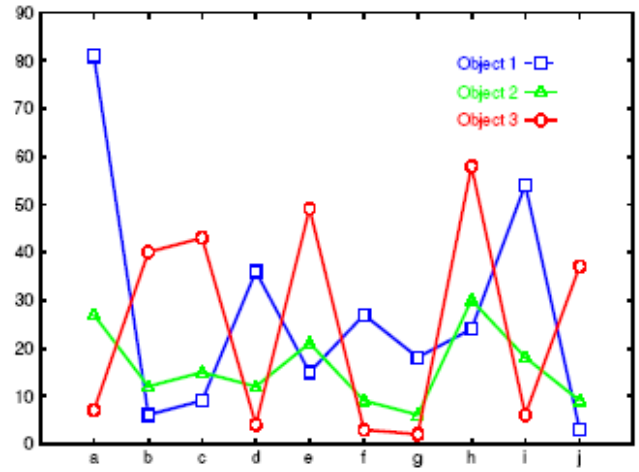
- Bi-clustering: Cluster both objects and attributes simultaneously (treat objs and attrs in symmetric way)
- Four requirements:
  - Only a small set of objects participate in a cluster
  - A cluster only involves a small number of attributes
  - An object may participate in multiple clusters, or does not participate in any cluster at all
  - An attribute may be involved in multiple clusters, or is not involved in any cluster at all
- Ex 1. *Gene expression* or *microarray data:* a gene sample/condition matrix.
  - Each element in the matrix, a real number, records the expression level of a gene under a specific condition
- Ex. 2. Clustering customers and products
  - Another bi-clustering problem

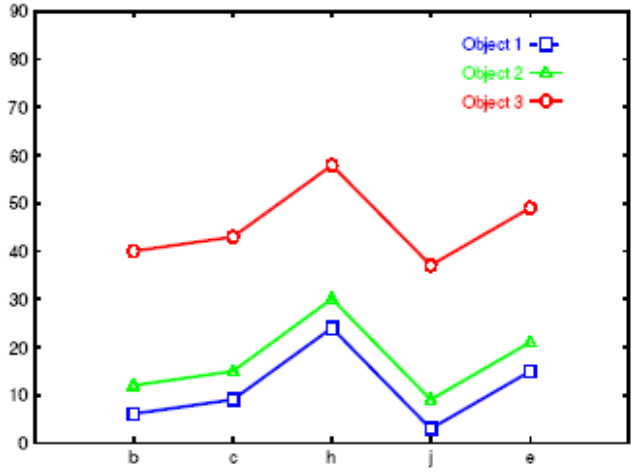


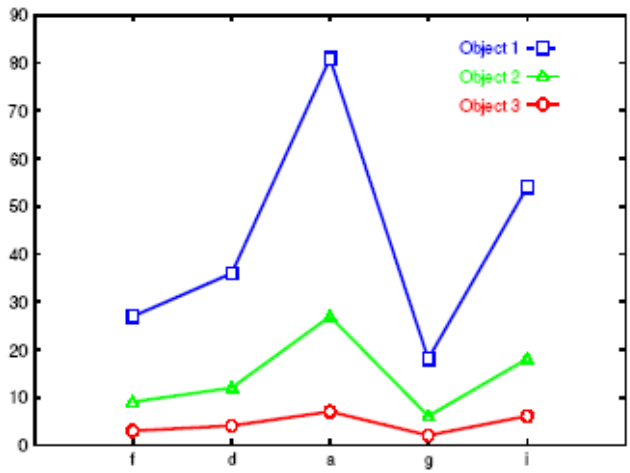

| | products | | | |
|---|---|---|---|---|
| | $w_{11}$ | $w_{12}$ | $\cdots$ | $w_{1m}$ |
| customers | $w_{21}$ | $w_{22}$ | $\cdots$ | $w_{2m}$ |
| | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |
| | $w_{n1}$ | $w_{n2}$ | $\cdots$ | $w_{nm}$ |

# Bi-Clustering for Micro-Array Data Analysis

- Left figure: Micro-array “raw” data shows 3 genes and their values in a multi-D space: Difficult to find their patterns
- Right two: Some subsets of dimensions form nice shift and scaling patterns
- No globally defined similarity/distance measure
- Clusters may not be exclusive
  - An object can appear in multiple clusters



# Types of Bi-clusters

- Let A = {$a_1$, ..., $a_n$} be a set of genes, B = {$b_1$, …, $b_n$} a set of conditions
- A bi-cluster: A submatrix where genes and conditions follow some consistent patterns
- 4 types of bi-clusters (ideal cases)
  - Bi-clusters with constant values:
    - for any *i* in *I* and *j in J*, $e_{ij} = c$

| | | | | |
|---|---|---|---|---|
| 10 | 10 | 10 | 10 | 10 |
| 20 | 20 | 20 | 20 | 20 |
| 50 | 50 | 50 | 50 | 50 |
| 0 | 0 | 0 | 0 | 0 |

  - Bi-clusters with constant values on rows:
    - $e_{ij} = c + \alpha_i$
    - Also, it can be constant values on columns

| | | | | |
|---|---|---|---|---|
| 10 | 50 | 30 | 70 | 20 |
| 20 | 60 | 40 | 80 | 30 |
| 50 | 90 | 70 | 110 | 60 |
| 0 | 40 | 20 | 60 | 10 |

  - Bi-clusters with *coherent values* (aka. *pattern-based clusters*)
    - $e_{ij} = c + \alpha_i + \beta_j$
  - Bi-clusters with *coherent* evolutions on rows

| | | | | |
|---|---|---|---|---|
| 10 | 50 | 30 | 70 | 20 |
| 20 | 100 | 50 | 1000 | 30 |
| 50 | 100 | 90 | 120 | 80 |
| 0 | 80 | 20 | 100 | 10 |

    - $(e_{i1j1} - e_{i1j2})(e_{i2j1} - e_{i2j2}) \geq 0$
    - i.e., only interested in the up- or down- regulated changes across genes or conditions without constraining on the exact values

# Bi-Clustering Methods

- Real-world data is noisy: Try to find approximate bi-clusters
- Methods: Optimization-based methods vs. enumeration methods
- Optimization-based methods
  - Try to find a submatrix at a time that achieves the best significance as a bi-cluster; Iterative Process terminating w/ user-defined rule
  - Due to the cost in computation, greedy search is employed to find local optimal bi-clusters
  - Ex. δ-Cluster Algorithm (Cheng and Church, ISMB'2000)
- Enumeration methods
  - Use a tolerance threshold to specify the degree of noise allowed in the bi-clusters to be mined
  - Then try to enumerate all submatrices as bi-clusters that satisfy the requirements
  - Ex. δ-pCluster Algorithm (H. Wang et al.' SIGMOD'2002, MaPle: Pei et al., ICDM'2003)

# Bi-Clustering (I): δ-Bi-Cluster

- For a submatrix *I x J,* the mean of the *i*-th row: $e_{iJ} = \frac{1}{|J|}\sum_{j\in J} e_{ij}$
  - The mean of the *j*-th column: $e_{Ij} = \frac{1}{|I|}\sum_{i\in I} e_{ij}$
  - The mean of all elements in the submatrix is

$$e_{IJ} = \frac{1}{|I||J|}\sum_{i\in I, j\in J} e_{ij} = \frac{1}{|I|}\sum_{i\in I} e_{iJ} = \frac{1}{|J|}\sum_{j\in J} e_{Ij}$$

  - The quality of the submatrix as a bi-cluster can be measured by the *mean squared residue* value

$$H(I \times J) = \frac{1}{|I||J|}\sum_{i\in I, j\in J} (e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2$$

- A submatrix *I x J* is **δ-bi-cluster** if $H(I \times J) \le \delta$ where $\delta \ge 0$ is a threshold. When $\delta = 0$, *I x J* is a perfect bi-cluster with coherent values. By setting $\delta > 0$, a user can specify the tolerance of average noise per element against a perfect bi-cluster
  - residue($e_{ij}$) = $e_{ij} - e_{iJ} - e_{Ij} + e_{IJ}$

# Bi-Clustering (I): The δ-Cluster Algorithm

- **Maximal δ-bi-cluster** is a **δ**-bi-cluster *I x J* such that there does not exist another **δ**-bi-cluster *I′ x J′ which* contains *I x J*
- Computing is costly: Use heuristic greedy search to obtain local optimal clusters
- Two phase computation: deletion phase and additional phase
- Deletion phase: Start from the whole matrix, iteratively remove rows and columns while the mean squared residue of the matrix is over **δ**
  - At each iteration, for each row/*column*, compute the *mean squared residue:*

$$d(i) = \frac{1}{|J|}\sum_{j\in J}(e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2 \qquad d(j) = \frac{1}{|I|}\sum_{i\in I}(e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2$$

  - Remove the row or column of the largest mean squared residue
- Addition phase:
  - Expand iteratively the **δ**-bi-cluster *I x J* obtained in the deletion phase as long as the **δ**-bi-cluster requirement is maintained
  - Consider all the rows/columns not involved in the current bi-cluster *I x J* by calculating their mean squared residues
  - A row/column of the smallest mean squared residue is added into the current **δ**-bi-cluster
- It finds only one **δ**-bi-cluster, thus needs to run multiple times: replacing the elements in the output bi-cluster by random numbers

# Bi-Clustering (II): δ-pCluster

- Enumerating all bi-clusters (δ-pClusters) [H. Wang, et al., Clustering by pattern similarity in large data sets. SIGMOD'02]
- Since a submatrix *I x J* is a bi-cluster with (perfect) coherent values iff $e_{i1j1} - e_{i2j1} = e_{i1j2} - e_{i2j2}$. For any 2 *x* 2 submatrix of *I x J*, define *p-score*

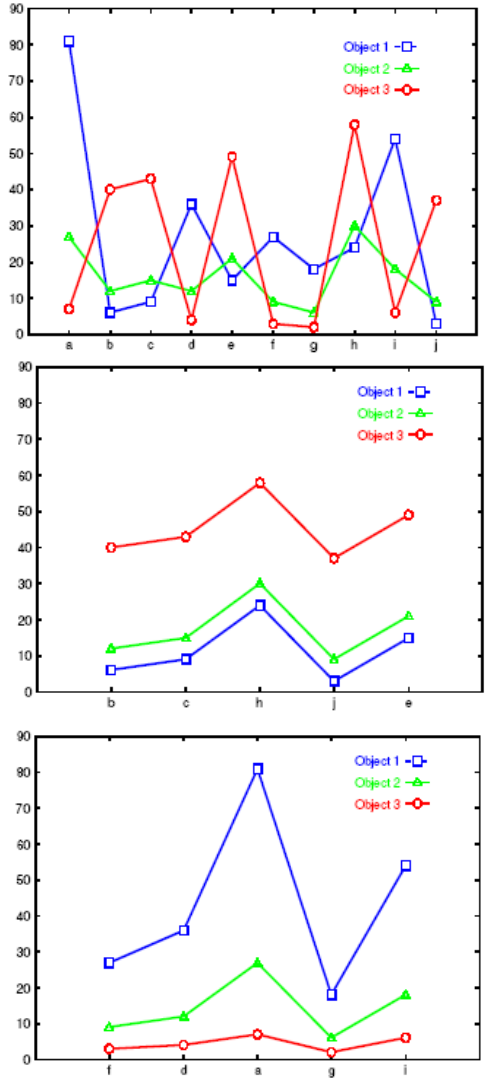
$$p\text{-score}\begin{pmatrix} e_{i_1 j_1} & e_{i_1 j_2} \\ e_{i_2 j_1} & e_{i_2 j_2} \end{pmatrix} = |(e_{i_1 j_1} - e_{i_2 j_1}) - (e_{i_1 j_2} - e_{i_2 j_2})|$$

- A submatrix *I x J* is a **δ-pCluster** (pattern-based cluster) if the *p*-score of every 2 *x* 2 submatrix of *I x J* is at most δ, where δ ≥ 0 is a threshold specifying a user's tolerance of noise against a perfect bi-cluster
- The *p*-score controls the noise on every element in a bi-cluster, while the mean squared residue captures the average noise
- **Monotonicity**: If *I* x *J* is a δ-pClusters, every *x* x *y* (*x,y* ≥ 2) submatrix of *I* x *J* is also a δ-pClusters.
- A δ-pCluster is **maximal** if no more row or column can be added into the cluster and retain δ-pCluster: We only need to compute all maximal δ-pClusters.

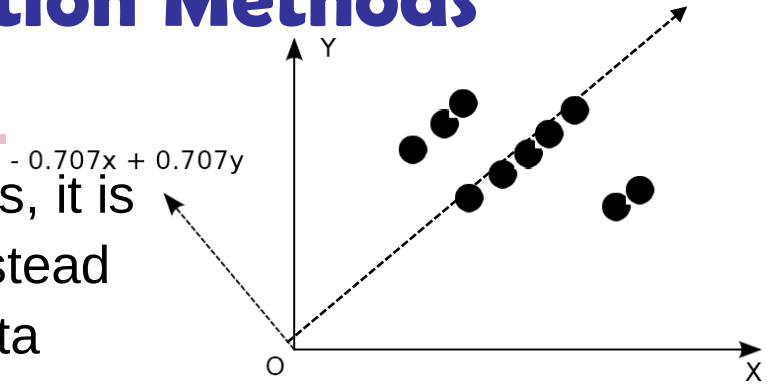
# MaPle: Efficient Enumeration of δ-pClusters

- Pei et al., MaPle: Efficient enumerating all maximal δ-pClusters. ICDM'03
- Framework: Same as pattern-growth in frequent pattern mining (based on the downward closure property)
- Start with small combinations of conditions; for each condition combination J, find the maximal subsets of genes I such that I x J is a δ-pClusters
  - If I x J is not a submatrix of another δ-pClusters

    then I x J is a maximal δ-pCluster.
- Many possible combinations J exist, but MaPle can prune the search space in many ways:
  - When no I exist for a combination J, no further extension of J is necessary
  - I x J is considered only if for every J' (|J'|=|J|-1) subset for J, I x J' is a δ-pCluster
- Algorithm is very similar to mining frequent closed itemsets
- Is complete but can be very time consuming for large matrices



# Dimensionality-Reduction Methods




- Dimensionality reduction: In some situations, it is more effective to construct a new space instead of using some subspaces of the original data
- Ex. To cluster the points in the right figure, any subspace of the original one, X and Y, cannot help, since all the three clusters will be projected into the overlapping areas in X and Y axes.
  - Construct a new dimension as the dashed one, the three clusters become apparent when the points projected into the new dimension
- Dimensionality reduction methods
  - Feature selection and extraction: But may not focus on clustering structure finding
  - Spectral clustering: Combining feature extraction and clustering (i.e., use the *spectrum* of the similarity matrix of the data to perform dimensionality reduction for clustering in fewer dimensions)
    - Normalized Cuts (Shi and Malik, CVPR'97 or PAMI'2000)
    - The Ng-Jordan-Weiss algorithm (NIPS'01)

# Spectral Clustering: The Ng-Jordan-Weiss (NJW) Algorithm

- Given a set of objects $o_1$, …, $o_n$, and the distance between each pair of objects, dist($o_i$, $o_j$), find the desired number k of clusters
- Calculate an affinity matrix W, where σ is a scaling parameter that controls how fast the affinity $W_{ij}$ decreases as dist($o_i$, $o_j$) increases. In NJW, set $W_{ij} = 0$

$$W_{ij} = e^{-\frac{dist(o_i,o_j)}{\sigma^2}}$$

- Derive a matrix A = *f*(W). NJW defines a matrix D to be a diagonal matrix s.t. $D_{ii}$ is the sum of the i-th row of W, i.e.,

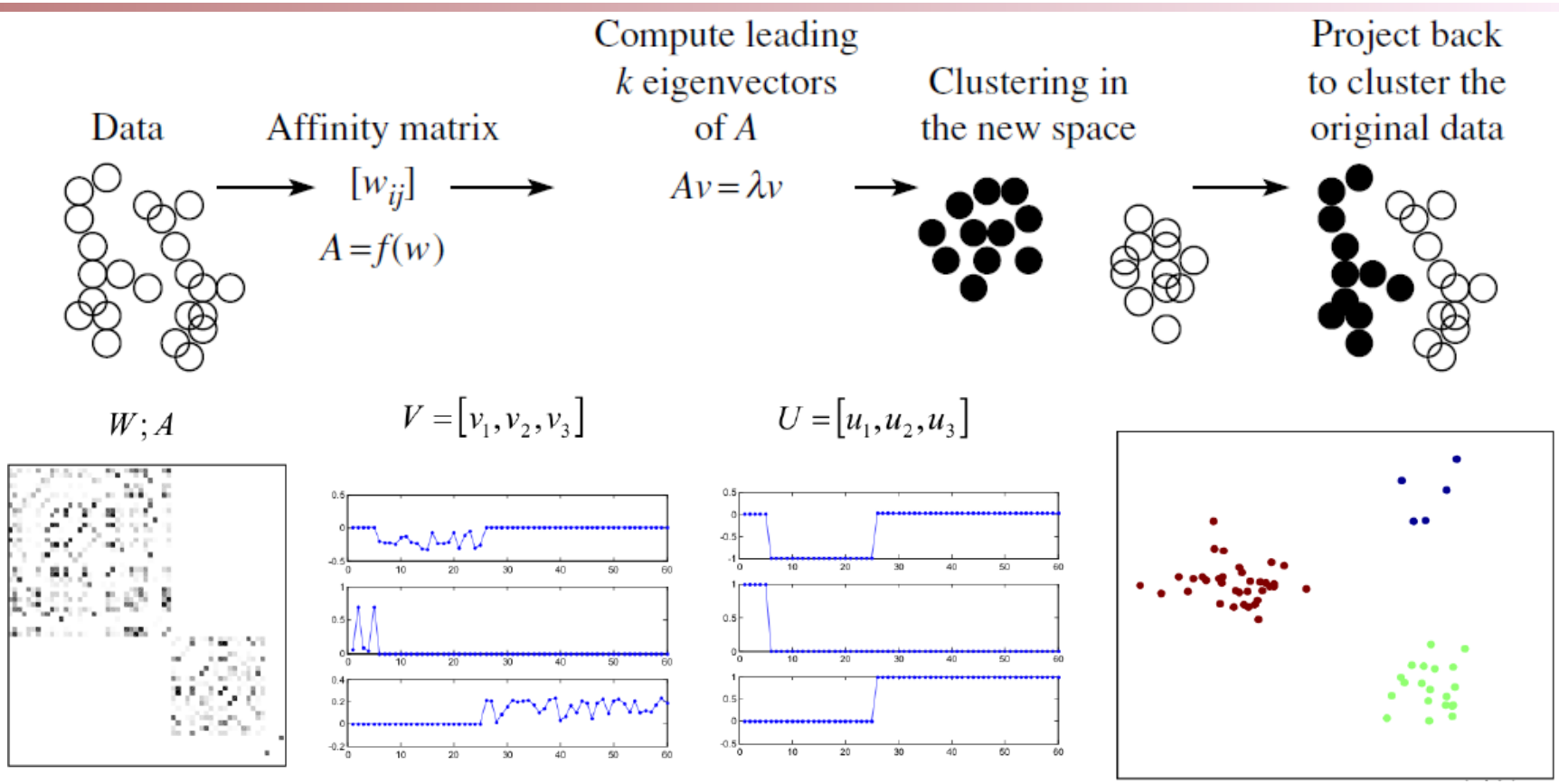
$$D_{ii} = \sum_{j=1}^{n} W_{ij}$$

  Then, A is set to $A = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$

- A spectral clustering method finds the k leading eigenvectors of A
  - A vector v is an eigenvector of matrix A if Av = λv, where λ is the corresponding eigen-value
- Using the k leading eigenvectors, project the original data into the new space defined by the k leading eigenvectors, and run a clustering algorithm, such as *k*-means, to find k clusters
  - If $X = [x_1 x_2 \cdots x_k] \in \mathbb{R}^{n \times k}$, compute $Y_{ij}$ where each row is projection of an original data point in the new space:

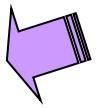
$$Y_{ij} = \frac{X_{ij}}{\sqrt{\sum_{j=1}^{k} X_{ij}^2}}$$

# Spectral Clustering: Illustration and Comments




- Spectral clustering: Effective in tasks like image processing
- Scalability challenge: Computing eigenvectors on a large matrix is costly
- Can be combined with other clustering methods, such as bi-clustering

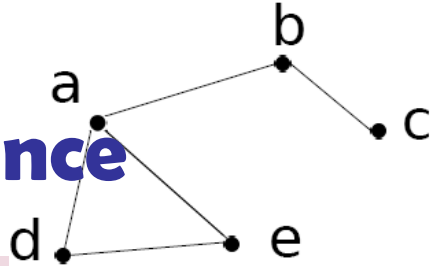
# Chapter 11. Cluster Analysis: Advanced Methods

- Probability Model-Based Clustering
- Clustering High-Dimensional Data
- Clustering Graphs and Network Data 
- Clustering with Constraints
- Summary

# Clustering Graphs and Network Data

- Applications
  - Bi-partite graphs, e.g., customers and products, authors and conferences
  - Web search engines, e.g., click through graphs and Web query graphs
  - Social networks, friendship/coauthor graphs
- Similarity measures
  - Geodesic distances
  - SimRank: Distance based on Structural Similarity
- Graph clustering methods
  - Minimum cuts: FastModularity (Clauset, Newman & Moore, 2004)
  - Density-based clustering: SCAN (Xu et al., KDD'2007)

# Similarity Measure (I): Geodesic Distance




- Geodesic distance (A, B): length (i.e., # of edges) of the shortest path between A and B (if not connected, defined as infinite)
- **Eccentricity** of v, eccen(v): The largest geodesic distance between v and any other vertex $u \in V - \{v\}$.
  - E.g., eccen(a) = eccen(b) = 2; eccen(c) = eccen(d) = eccen(e) = 3
- **Radius** of graph G: The minimum eccentricity of all vertices, i.e., the distance between the "most central point" and the "farthest border"
  - $r = \min_{v \in V} eccen(v)$
  - E.g., radius (g) = 2
- **Diameter** of graph G: The maximum eccentricity of all vertices, i.e., the largest distance between any pair of vertices in G
  - $d = \max_{v \in V} eccen(v)$
  - E.g., diameter (g) = 3
- A **peripheral vertex** is a vertex that achieves the diameter.
  - E.g., Vertices c, d, and e are peripheral vertices

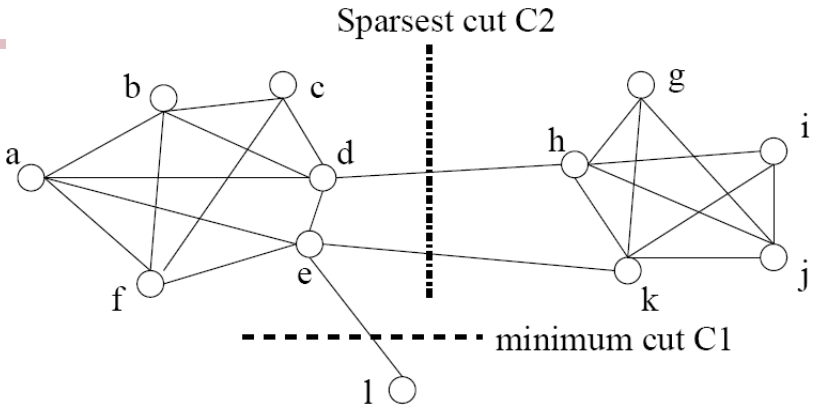
# SimRank: Similarity Based on Structural Context

- SimRank: structural-context similarity, i.e., based on the similarity of its neighbors
- In a directed graph G = (V,E),
  - *individual in-neighborhood* of v: I(v) = {u | (u, v) ∈ E}
  - *individual out-neighborhood* of v: O(v) = {w | (v, w) ∈ E}
- Similarity in SimRank:

$$s(u,v) = \frac{C}{|I(u)||I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s(x,y)$$

- How to compute SimRank?
  - Initialization: $s_0(u,v) = \begin{cases} 0 & \text{if } u \neq v \\ 1 & \text{if } u = v \end{cases}$
  - Then we can compute $s_{i+1}$ from $s_i$ based on the definition

# Graph Clustering: Sparsest Cut




- G = (V,E). The *cut set* of a cut is the set of edges {(u, v) ∈ E | u ∈ S, v ∈ T } and S and T are in two partitions
- *Size* of the cut: # of edges in the cut set
- Min-cut (e.g., $C_1$) is not a good partition
- A better measure: **Sparsity**: $$\Phi = \frac{\text{the size of the cut}}{\min\{|S|, |T|\}}$$
- A cut is ***sparsest*** if its sparsity is not greater than that of any other cut
- Ex. Cut C2 = ({a, b, c, d, e, f, l}, {g, h, i, j, k}) is the sparsest cut
- For k clusters, the **modularity** of a clustering assesses the quality of the clustering: $$Q = \sum_{i=1}^{k} \left(\frac{l_i}{|E|} - \left(\frac{d_i}{2|E|}\right)^2\right)$$

  $l_i$: # edges between vertices in the i-th cluster
  $d_i$: the sum of the degrees of the vertices in the i-th cluster
- The *modularity* of a clustering of a graph is the difference between the fraction of all edges that fall into individual clusters and the fraction that would do so if the graph vertices were randomly connected
- The optimal clustering of graphs maximizes the modularity

# Graph Clustering: Challenges of Finding Good Cuts

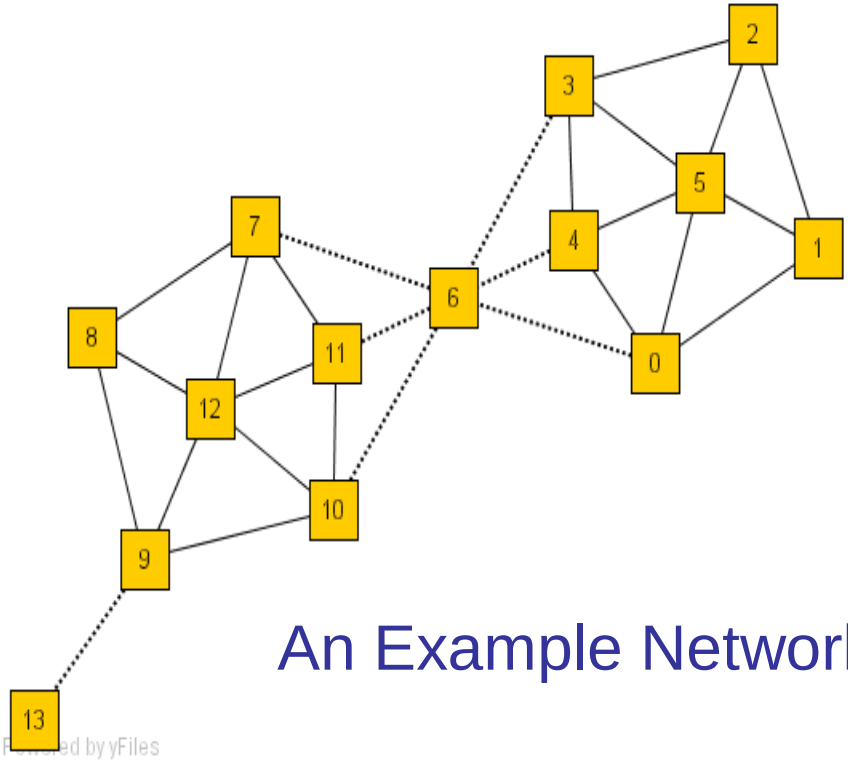
- High computational cost
  - Many graph cut problems are computationally expensive
  - The sparsest cut problem is NP-hard
  - Need to tradeoff between efficiency/scalability and quality
- Sophisticated graphs
  - May involve weights and/or cycles.
- High dimensionality
  - A graph can have many vertices. In a similarity matrix, a vertex is represented as a vector (a row in the matrix) whose dimensionality is the number of vertices in the graph
- Sparsity
  - A large graph is often sparse, meaning each vertex on average connects to only a small number of other vertices
  - A similarity matrix from a large sparse graph can also be sparse

# Two Approaches for Graph Clustering

- Two approaches for clustering graph data
  - Use *generic clustering methods* for high-dimensional data
  - *Designed specifically for clustering graphs*
- Using clustering methods for high-dimensional data
  - Extract a similarity matrix from a graph using a similarity measure
  - A generic clustering method can then be applied on the similarity matrix to discover clusters
  - Ex. Spectral clustering: approximate optimal graph cut solutions
- Methods specific to graphs
  - Search the graph to find well-connected components as clusters
  - Ex. SCAN (Structural Clustering Algorithm for Networks)
    - X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, “SCAN: A Structural Clustering Algorithm for Networks”, KDD'07

# SCAN: Density-Based Clustering of Networks

- How many clusters?
- What size should they be?
- What is the best partitioning?
- Should some points be segregated?

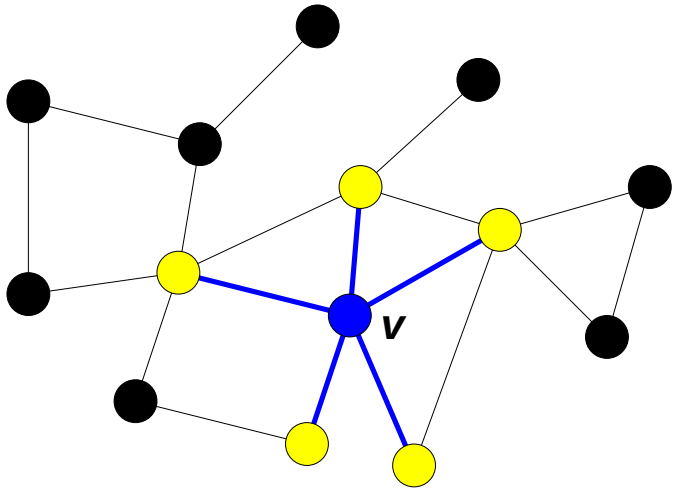



An Example Network

- Application: Given simply information of who associates with whom, could one identify clusters of individuals with common interests or special relationships (families, cliques, terrorist cells)?

# A Social Network Model

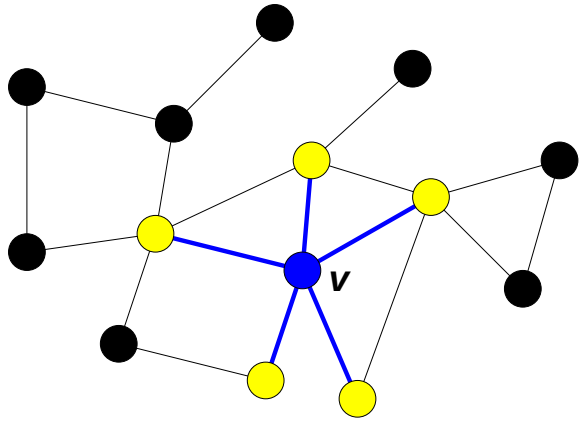
- Cliques, hubs and outliers
  - Individuals in a tight social group, or clique, know many of the same people, regardless of the size of the group
  - Individuals who are hubs know many people in different groups but belong to no single group. Politicians, for example bridge multiple groups
  - Individuals who are outliers reside at the margins of society. Hermits, for example, know few people and belong to no group
- The Neighborhood of a Vertex
  - Define $\Gamma(v)$ as the immediate neighborhood of a vertex (i.e. the set of people that an individual knows )



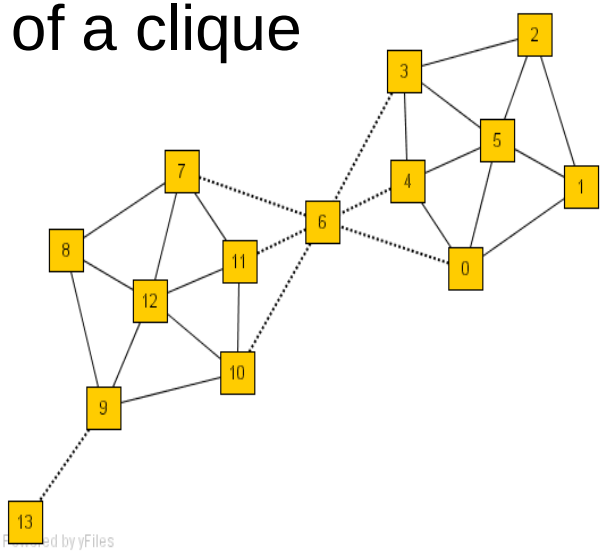
# Structure Similarity

- The desired features tend to be captured by a measure we call Structural Similarity

$$\sigma(v,w) = \frac{|\Gamma(v) \bigcap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$




- Structural similarity is large for members of a clique and small for hubs and outliers



# Structural Connectivity [1]

- $\varepsilon$-Neighborhood: $N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$
- Core: $CORE_{\varepsilon,\mu}(v) \Leftrightarrow \mid N_\varepsilon(v) \mid \geq \mu$
- Direct structure reachable:

$$DirRECH_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge w \in N_\varepsilon(v)$$

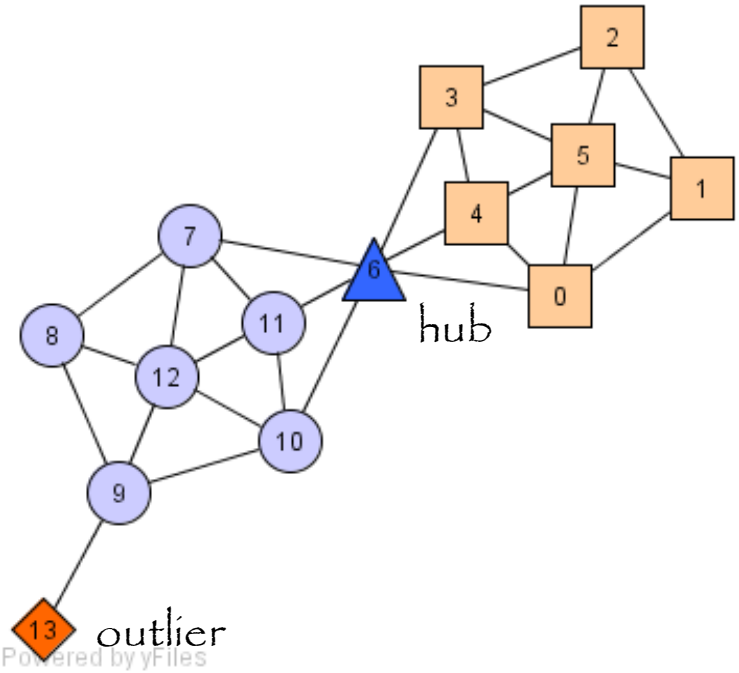
- Structure reachable: transitive closure of direct structure reachability
- Structure connected:

$$CONNECT_{\varepsilon,\mu}(v,w) \Leftrightarrow \exists u \in V : RECH_{\varepsilon,\mu}(u,v) \wedge RECH_{\varepsilon,\mu}(u,w)$$

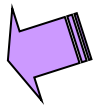
[1] M. Ester, H. P. Kriegel, J. Sander, & X. Xu (KDD'96) “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases

# Structure-Connected Clusters

- Structure-connected cluster C
  - Connectivity: $\forall v, w \in C : CONNECT_{\varepsilon,\mu}(v, w)$
  - Maximality: $\forall v, w \in V : v \in C \wedge REACH_{\varepsilon,\mu}(v, w) \Rightarrow w \in C$
- Hubs:
  - Not belong to any cluster
  - Bridge to many clusters
- Outliers:
  - Not belong to any cluster
  - Connect to less clusters

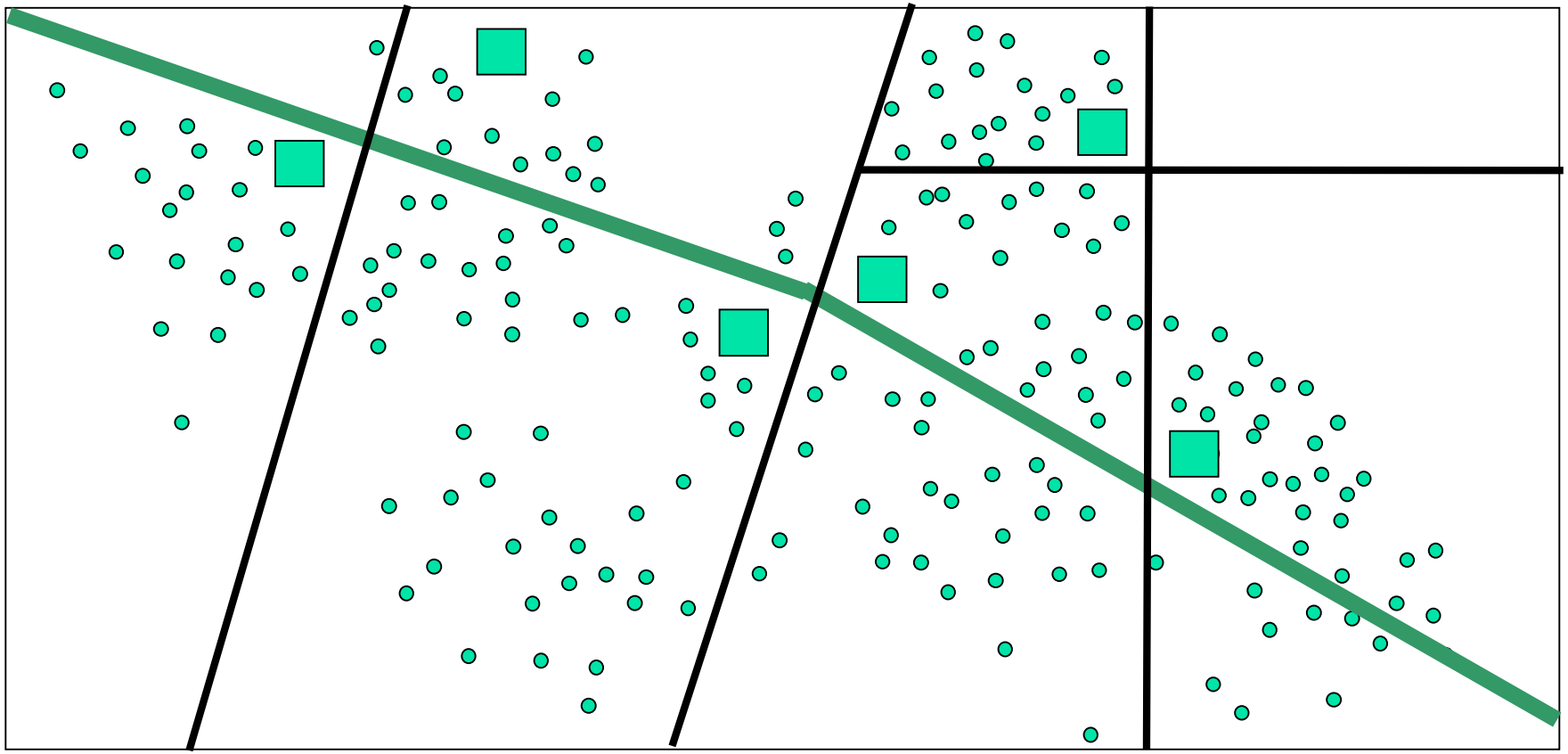


# Chapter 11. Cluster Analysis: Advanced Methods

- Probability Model-Based Clustering
- Clustering High-Dimensional Data
- Clustering Graphs and Network Data
- Clustering with Constraints 
- Summary

# Why Constraint-Based Cluster Analysis?

- Need user feedback: Users know their applications the best
- Less parameters but more user-desired constraints, e.g., an ATM allocation problem: obstacle & desired clusters



# Categorization of Constraints

- Constraints on instances: specifies how a pair or a set of instances should be grouped in the cluster analysis
  - Must-link vs. cannot link constraints
    - must-link(x, y): x and y should be grouped into one cluster
  - Constraints can be defined using variables, e.g.,
    - cannot-link(x, y) if dist(x, y) > d
- Constraints on clusters: specifies a requirement on the clusters
  - E.g., specify the min # of objects in a cluster, the max diameter of a cluster, the shape of a cluster (e.g., a convex), # of clusters (e.g., k)
- Constraints on similarity measurements: specifies a requirement that the similarity calculation must respect
  - E.g., driving on roads, obstacles (e.g., rivers, lakes)
- Issues: Hard vs. soft constraints; conflicting or redundant constraints

# Constraint-Based Clustering Methods (I): Handling Hard Constraints

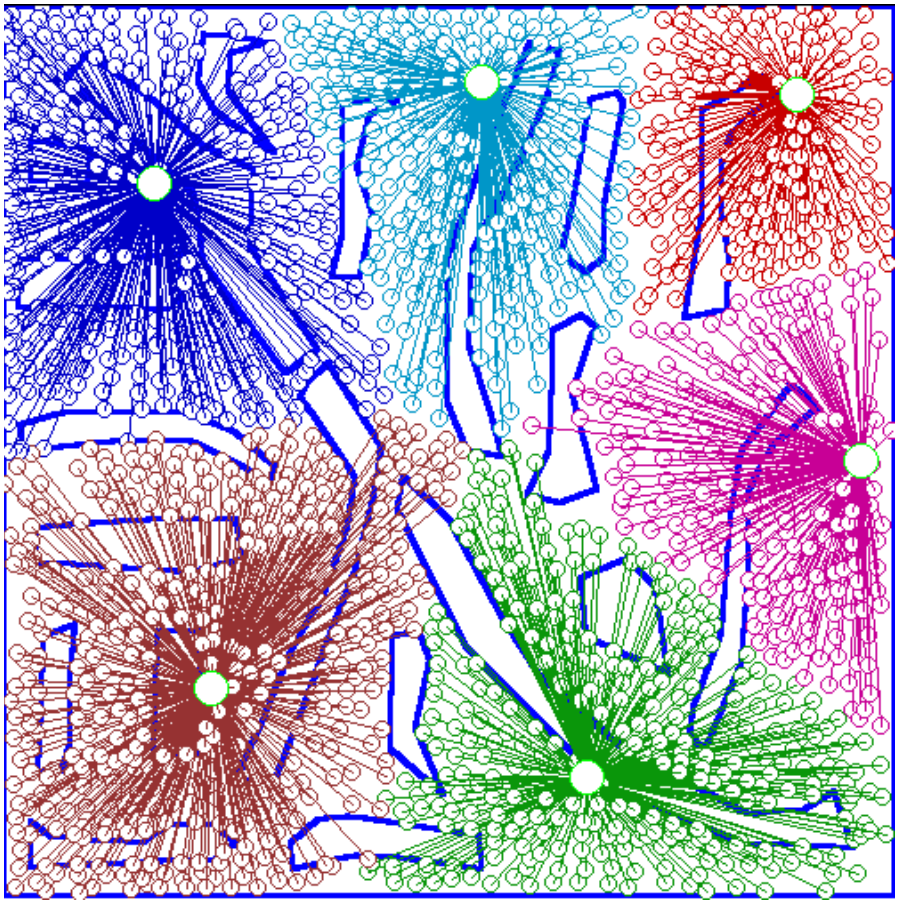
- Handling hard constraints: Strictly respect the constraints in cluster assignments
- Example: The COP-k-means algorithm
  - Generate super-instances for must-link constraints
    - Compute the transitive closure of the must-link constraints
    - To represent such a subset, replace all those objects in the subset by the mean.
    - The super-instance also carries a weight, which is the number of objects it represents
  - Conduct modified k-means clustering to respect cannot-link constraints
    - Modify the center-assignment process in k-means to a *nearest feasible center assignment*
    - An object is assigned to the nearest center so that the assignment respects all cannot-link constraints

# Constraint-Based Clustering Methods (II): Handling Soft Constraints

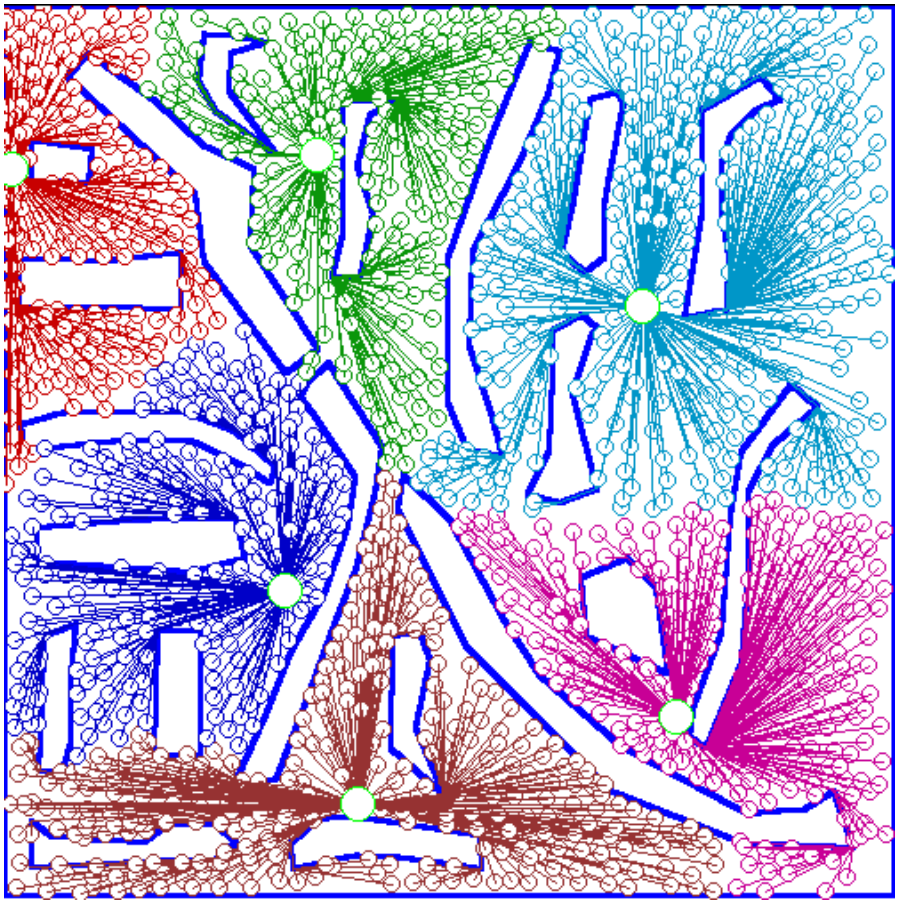
- Treated as an optimization problem: When a clustering violates a soft constraint, a penalty is imposed on the clustering
- Overall objective: Optimizing the clustering quality, and minimizing the constraint violation penalty
- Ex. CVQE (Constrained Vector Quantization Error) algorithm: Conduct k-means clustering while enforcing constraint violation penalties
- Objective function: Sum of distance used in k-means, adjusted by the constraint violation penalties
  - Penalty of a *must-link* violation
    - If objects x and y must-be-linked but they are assigned to two different centers, $c_1$ and $c_2$, dist($c_1$, $c_2$) is added to the objective function as the penalty
  - Penalty of a cannot-link violation
    - If objects x and y cannot-be-linked but they are assigned to a common center c, dist(c, c′), between c and c′ is added to the objective function as the penalty, where c′ is the closest cluster to c that can accommodate x or y

# Complexity of Clustering with Constrains

## An Example: Clustering With Obstacle Objects

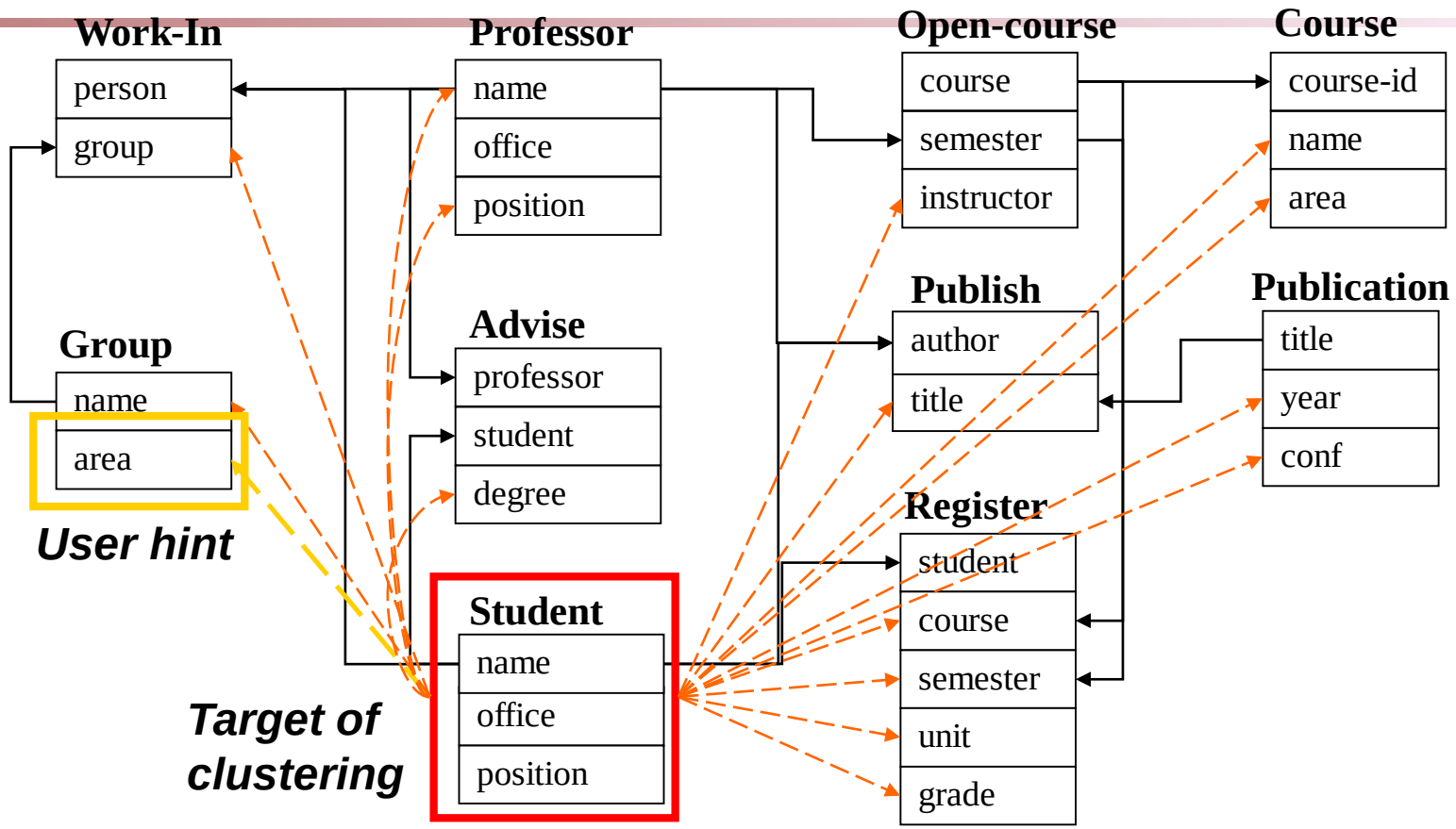


*Not* Taking obstacles into account



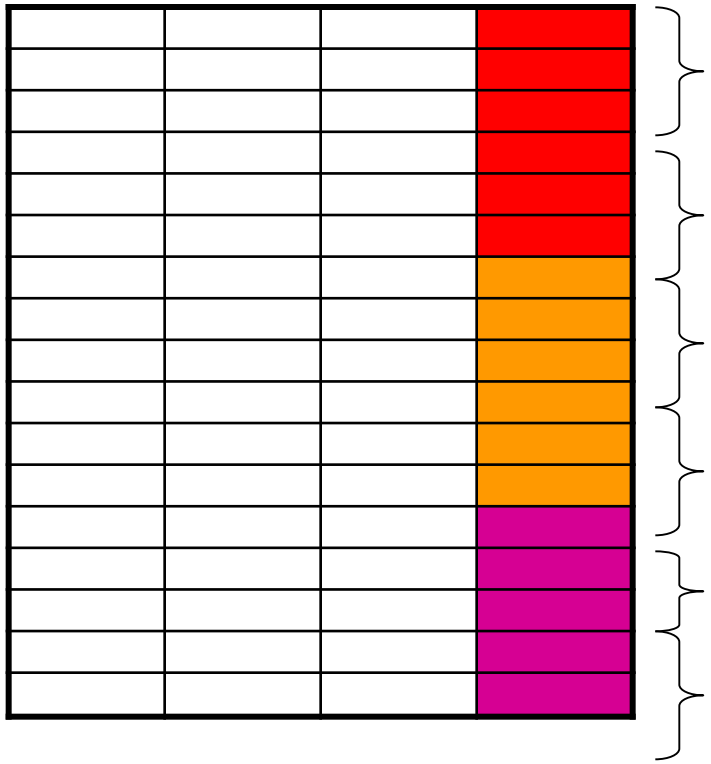
Taking obstacles into account

# User-Guided Clustering: A Special Kind of Constraints




- X. Yin, J. Han, P. S. Yu, “Cross-Relational Clustering with User's Guidance”, KDD'05
- User usually has a goal of clustering, e.g., clustering students by research area
- User specifies his clustering goal to CrossClus

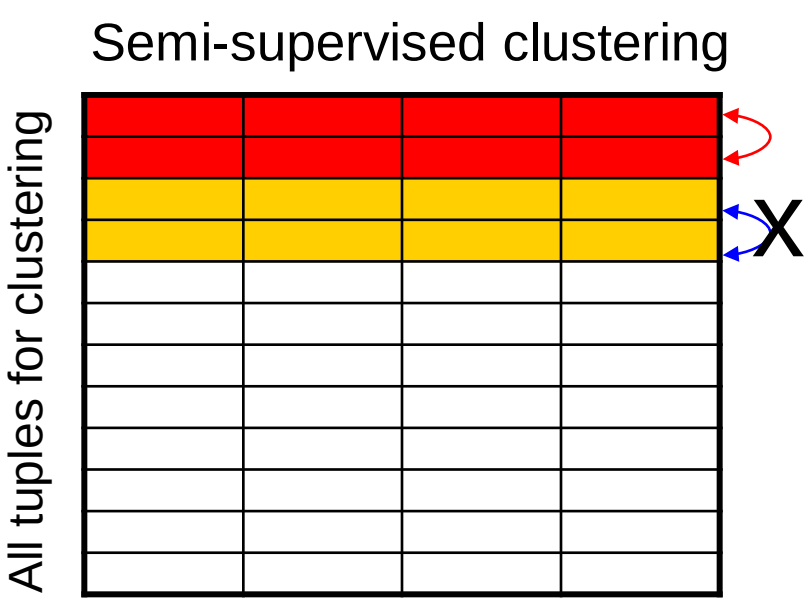
# Comparing with Classification

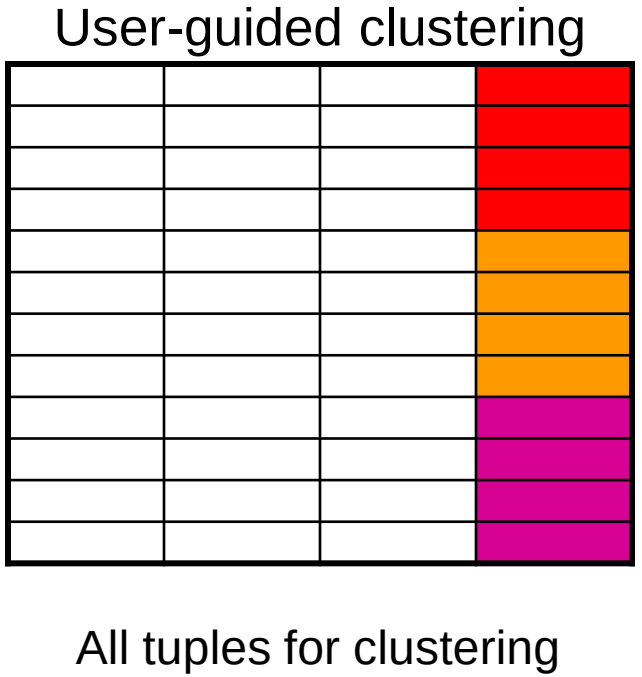



- User-specified *feature* (in the form of *attribute*) is used as a hint, not class labels
  - The attribute may contain too many or too few distinct values, e.g., a user may want to cluster students into 20 clusters instead of 3
  - Additional features need to be included in cluster analysis

# Comparing with Semi-Supervised Clustering

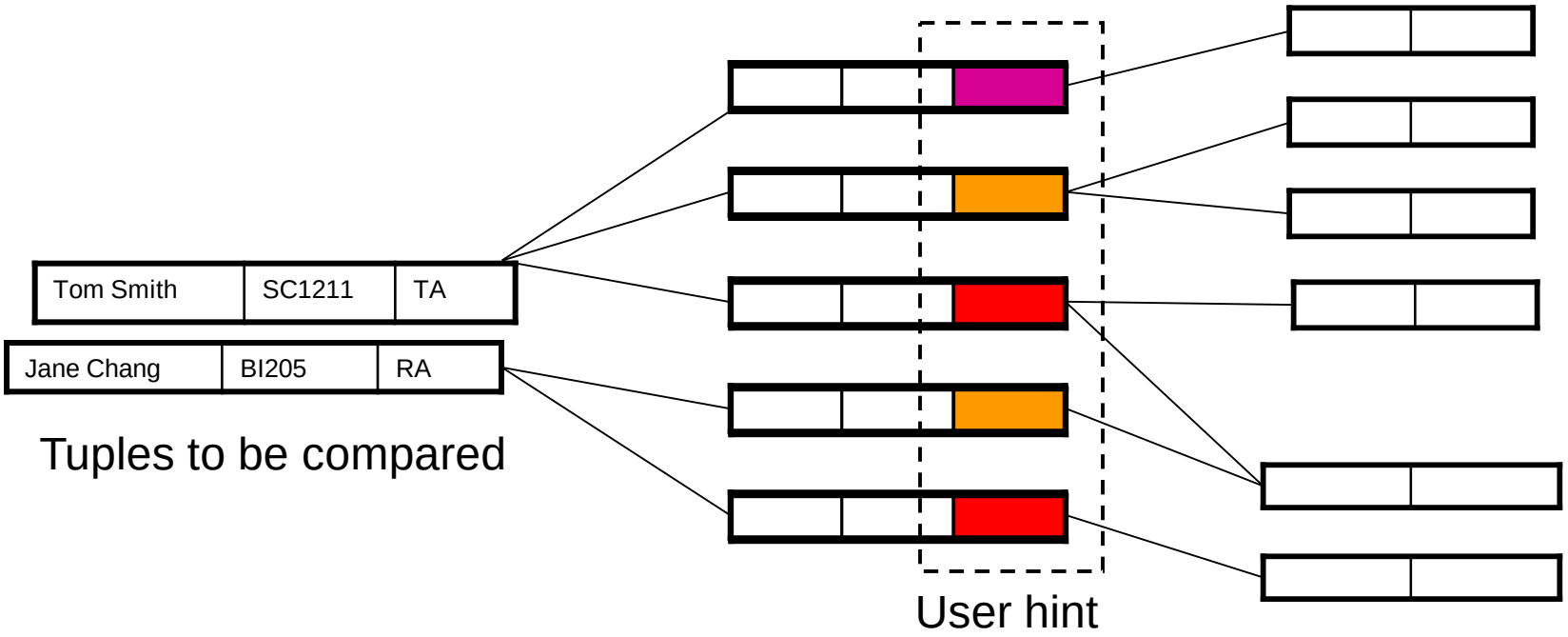
- Semi-supervised clustering: User provides a training set consisting of “similar” (“must-link) and “dissimilar” (“cannot link”) pairs of objects
- User-guided clustering: User specifies an attribute as a hint, and more relevant features are found for clustering



# Why Not Semi-Supervised Clustering?

- Much information (in multiple relations) is needed to judge whether two tuples are similar
- A user may not be able to provide a good training set
- It is much easier for a user to specify an attribute as a hint, such as a student's *research area*



# CrossClus: An Overview

- Measure *similarity between features* by how they group objects into clusters
- Use a heuristic method to search for pertinent features
  - *Start from user-specified feature and gradually expand search range*
- Use *tuple ID propagation* to create feature values
  - Features can be easily created during the expansion of search range, by propagating IDs
- Explore existing clustering algorithms: *k*-means, *k*-medoids, and hierarchical clustering

# Summary

- Probability Model-Based Clustering
  - Fuzzy clustering
  - Probability-model-based clustering
  - The EM algorithm
- Clustering High-Dimensional Data
  - Subspace clustering: bi-clustering methods
  - Dimensionality reduction: Spectral clustering
- Clustering Graphs and Network Data
  - Graph clustering: min-cut vs. sparsest cut
  - High-dimensional clustering methods
  - Graph-specific clustering methods, e.g., SCAN
- Clustering with Constraints
  - Constraints on instance objects, e.g., Must link vs. Cannot Link
  - Constraint-based clustering algorithms

# References (I)


- R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD'98*
- C. C. Aggarwal, C. Procopiuc, J. Wolf, P. S. Yu, and J.-S. Park. Fast algorithms for projected clustering. *SIGMOD'99*
- S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56:5:1–5:37, 2009.
- J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, 1981.
- K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *ICDT'99*
- Y. Cheng and G. Church. Biclustering of expression data. *ISMB'00*
- I. Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. *SDM'05*
- I. Davidson, K. L. Wagstaff, and S. Basu. Measuring constraint-set utility for partitional clustering algorithms. *PKDD'06*
- C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *J. American Stat. Assoc.*, 97:611–631, 2002.
- F. H¨oppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. Wiley, 1999.
- G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. *KDD'02*
- H.-P. Kriegel, P. Kroeger, and A. Zimek. Clustering high dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowledge Discovery from Data (TKDD)*, 3, 2009.
- U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007

# References (II)


- G. J. McLachlan and K. E. Bkasford. *Mixture Models: Inference and Applications to Clustering*. John Wiley & Sons, 1988.
- B. Mirkin. Mathematical classification and clustering. *J. of Global Optimization*, 12:105–108, 1998.
- S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1, 2004.
- A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. NIPS'01
- J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu. Maple: A fast algorithm for maximal pattern-based clustering. *ICDM'03*
- M. Radovanovi´c, A. Nanopoulos, and M. Ivanovi´c. Nearest neighbors in high-dimensional data: the emergence and influence of hubs. *ICML'09*
- S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1:27–64, 2007.
- A. K. H. Tung, J. Hou, and J. Han. Spatial clustering in the presence of obstacles. *ICDE'01*
- A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-based clustering in large databases. *ICDT'01*
- A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: A survey. In *Handbook of Computational Molecular Biology, Chapman & Hall*, 2004.
- K. Wagstaff, C. Cardie, S. Rogers, and S. Schr¨odl. Constrained k-means clustering with background knowledge. *ICML'01*
- H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. *SIGMOD'02*
- X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. SCAN: A structural clustering algorithm for networks. *KDD'07*
- X. Yin, J. Han, and P.S. Yu, "Cross-Relational Clustering with User's Guidance", KDD'05