Data Mining and Machine Learning

Bioinspired computational methods

Biological data mining

Clustering of high-dimensional data

Francesco Marcelloni

Department of Information Engineering
University of Pisa
ITALY

Some slides belong to the collection

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign
Simon Fraser University

Clustering high-dimensional data

- Are the traditional distance measures which are frequently used in low-dimensional cluster analysis also effective on high-dimensional data?

Customer Purchase Data

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

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

despite Ada and Cathy look more similar

Clustering high-dimensional data

- 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

Clustering high-dimensional data

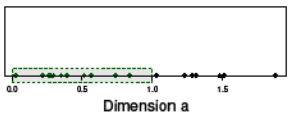
- Clustering high-dimensional data (How high is high-D in clustering?)
  - Many applications: text documents, DNA micro-array data
  - Major challenges:
    - Many irrelevant dimensions may mask clusters
    - Distance measure becomes meaningless—due to equi-distance
    - Clusters may exist only in some subspaces

Clustering high-dimensional data

- Two major kinds of 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

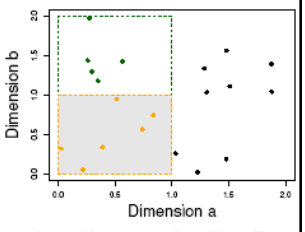
The Curse of Dimensionality

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

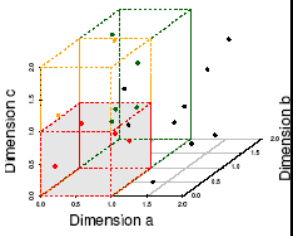


(a) 11 Objects in One Unit Bin

- 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

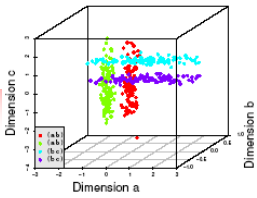


(b) 6 Objects in One Unit Bin



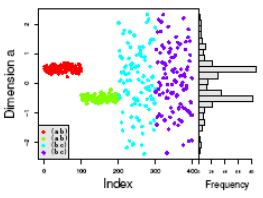
(c) 4 Objects in One Unit Bin

Why Subspace Clustering

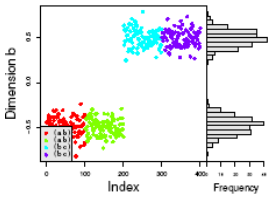


(graphs adapted from Parsons et al. KDD Explorations 2004)

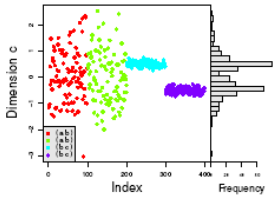
- Clusters may exist only in some subspaces
- Subspace-clustering: find clusters in all the subspaces



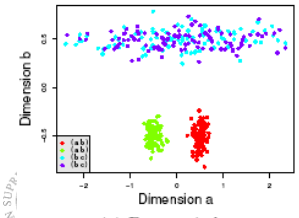
(a) Dimension a



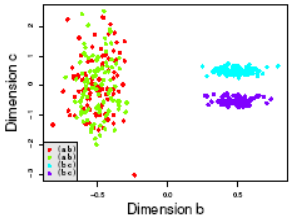
(b) Dimension b



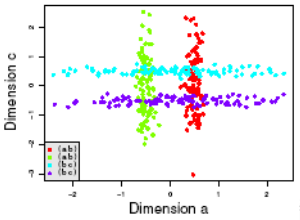
(c) Dimension c



(a) Dims a & b



(b) Dims b & c



(c) Dims a & c

7

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

8

Subspace Clustering Method (I): Subspace Search Methods

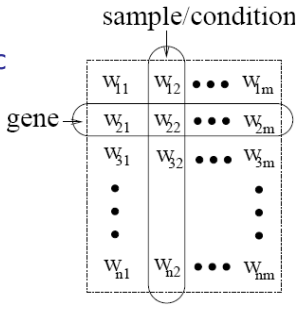
- Search various subspaces to find clusters
- *Bottom-up approaches*
  - Start from low-D subspaces and search higher-D subspaces only when there may be clusters in such subspaces
  - Various pruning techniques to reduce the number of higher-D subspaces to be searched
  - Ex. CLIQUE (Agrawal et al. 1998)
- *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

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



Subspace Clustering Method (III): Bi-Clustering Methods

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

Types of Bi-clusters

- Let A = {a_1, ..., a_n} be a set of genes, B = {b_1, ..., b_n} a set of conditions. Let E=[e_{ij}] be a gene expression data matrix.
- Bi-cluster: 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*, $e_{ij} = c$

	⋯	b_6	⋯	b_{12}	⋯	b_{36}
a_1	⋯	60	⋯	60	⋯	60
⋯	⋯	⋯	⋯	⋯	⋯	⋯
a_{33}	⋯	60	⋯	60	⋯	60
⋯	⋯	⋯	⋯	⋯	⋯	⋯
a_{86}	⋯	60	⋯	60	⋯	60

  - Bi-clusters with constant values on rows:
    - $e_{ij} = c + a_i$
      where a_i is the adjustment for row *i*.
    - Also, it can be constant values on columns

10	10	10	10	10
20	20	20	20	20
50	50	50	50	50
0	0	0	0	0

Types of Bi-clusters

- Bi-clusters with *coherent values* (aka. *pattern-based clusters*). Rows change in a synchronized way with respect to the columns and vice versa
  - $e_{ij} = c + a_i + \beta_j$
  - A I X J is a bicluster with coherent values if and only if for any

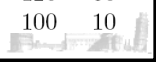
$i_1, i_2 \in I$ and $j_1, j_2 \in J$, then $e_{i_1 j_1} - e_{i_2 j_1} = e_{i_1 j_2} - e_{i_2 j_2}$

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

- Bi-clusters with *coherent* evolutions on rows
  - i.e., only interested in the up- or down- regulated changes across genes or conditions without constraining on the exact values
  - For any

$i_1, i_2 \in I$ and $j_1, j_2 \in J$, then $(e_{i_1 j_1} - e_{i_1 j_2})(e_{i_2 j_1} - e_{i_2 j_2}) \geq 0$

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

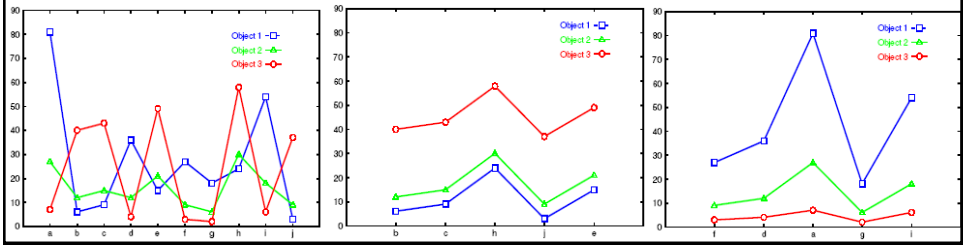


Bi-Clustering Methods

- 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
  - 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 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



Bi-Clustering (I): **δ-Bi-Cluster**

- For a submatrix $I \times J$, 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 $I \times J$ is **δ-bi-cluster** if $H(I \times J) \le \delta$ where $\delta \ge 0$ is a threshold. When $\delta = 0$, $I \times J$ 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 $I \times J$ such that there does not exist another **δ**-bi-cluster $I' \times J'$ *which* contains $I \times J$
- 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

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

- Addition phase:
  - Expand iteratively the **δ**-bi-cluster $I \times J$ 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 $I \times J$ 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 $I \times J$ is a bi-cluster with (perfect) coherent values iff $e_{i1j1} - e_{i2j1} = e_{i1j2} - e_{i2j2}$. For any 2 $\times$ 2 submatrix of $I \times J$, 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 $I \times J$ is a **δ-pCluster** (pattern-based cluster) if the *p*-score of every 2 $\times$ 2 submatrix of $I \times J$ is at most δ, where $\delta \geq 0$ is a threshold specifying a user's tolerance of noise against a perfect bi-cluster

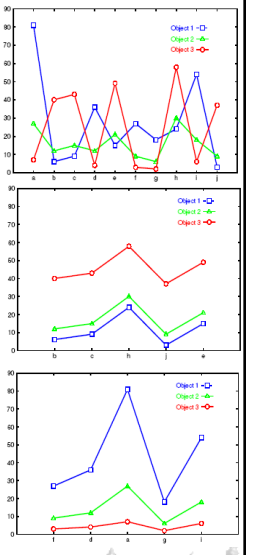
Bi-Clustering (II): δ-**pCluster**

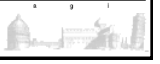
- The *p*-score controls the noise on every element in a bi-cluster, while the mean squared residue captures the average noise
- **Monotonicity**: If $I \times J$ is a δ-pClusters, every $x \times y$ ($x, y \geq 2$) submatrix of $I \times J$ is also a δ-pClusters.
- A δ-pCluster is **maximal** if no 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)
- 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

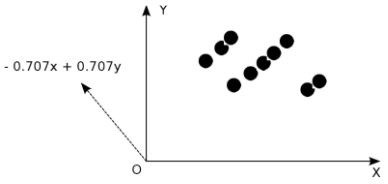
    then I x J is a maximal δ-pCluster.
- Algorithm is very similar to mining frequent closed itemsets





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 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

- 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.

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

Controls how rapidly the affinity matrix falls of with the distance

In NJW, set $W_{ii} = 0$

- Derive a matrix A = *f*(W). NJW defines a matrix, D, as a diagonal matrix such that D_{ii} is the sum of the i-th row of W, i.e.,

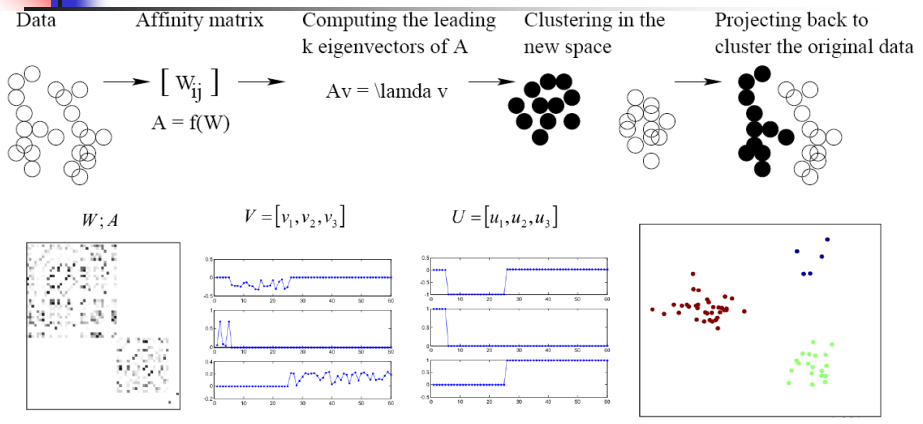
$$D_{ii} = \sum_{j=1}^{n} W_{ij}$$

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

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

- 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
- Assign the original data points to clusters according to how the transformed points are assigned in the clusters obtained

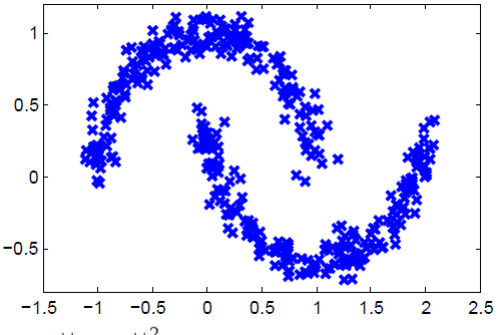
Spectral Clustering: Illustration and Comments



- 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

Spectral Clustering: Illustration and Comments

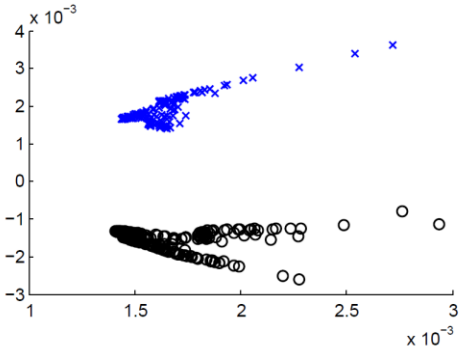
- An example of application: 200 data in each half moon (from Nicola Rebagliati's slide collection



- The similarity is $w(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{||v_i - v_j||_2^2}{0.3^2}}$
- K = 2

Spectral Clustering: Illustration and Comments

- Spectral embedding given by the first two eigenvectors



Spectral Clustering: Illustration and Comments

- Partition obtained by NJW