PA196: Pattern Recognition

10. Clustering

Dr. Vlad Popovici
popovici@recetox.muni.cz

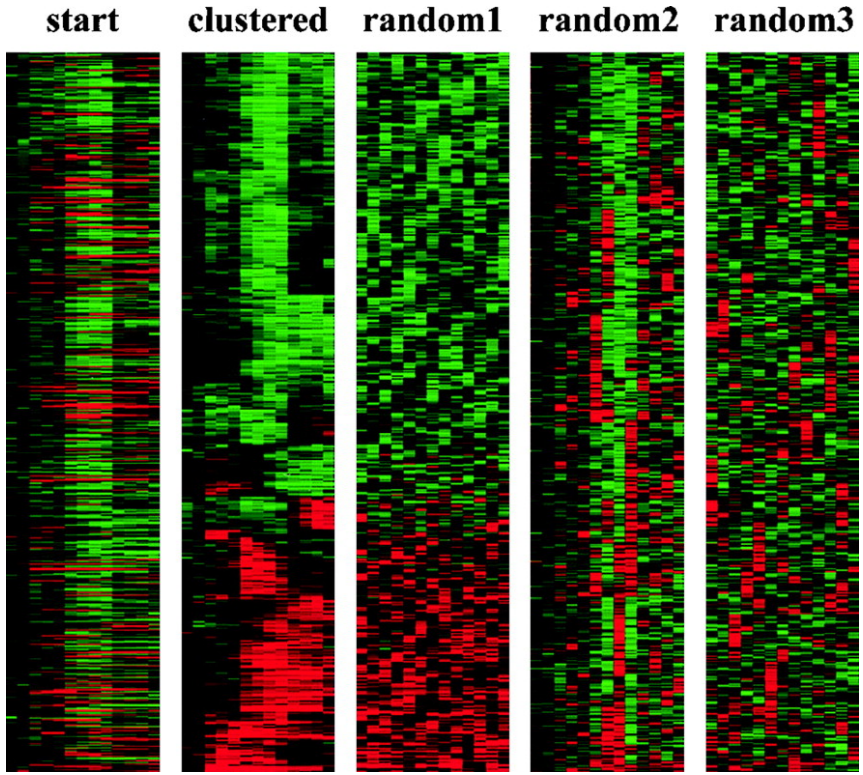
RECETOX
Masaryk University, Brno

Introduction

- clustering: a collection of methods for grouping data based on some measure of similarity
- technique for data exploration
- tries to identify some groupings of data: a partition of the given data set into some subsets that are (more or less) homogeneous in some sense
- different methods may lead to completely different partitions
- can be used as a starting point in a supervised analysis: the cluster labels may guide classifier design

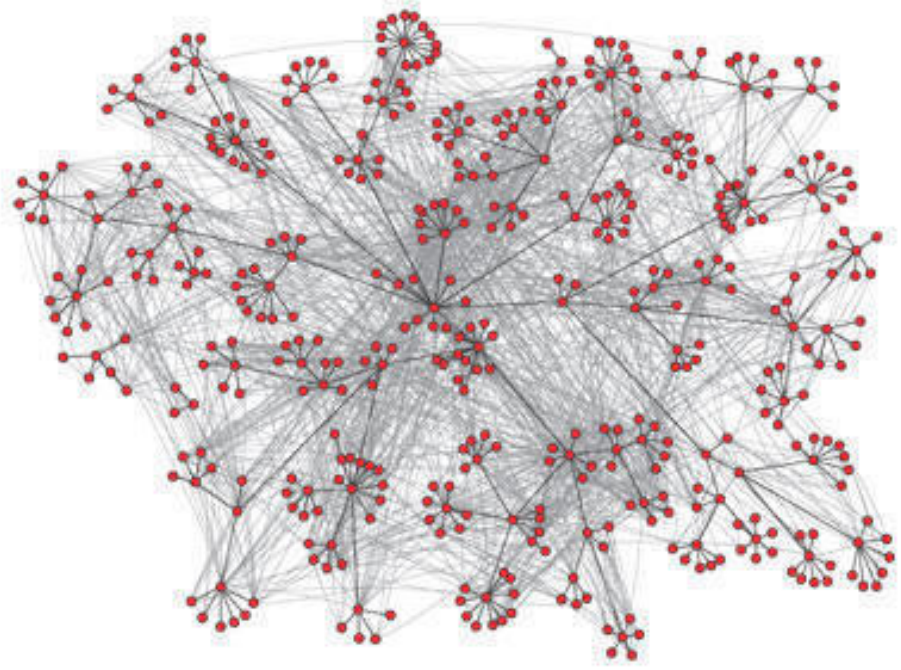
Why clustering?

Example: **Eisen et al, PNAS 1998**: "... statistical algorithms to arrange genes according to similarity in pattern of gene expression."



Why clustering?

Example: **Adamic, Adar, 2005**: the social network - email communication within a corporation



Why clustering?

Example: **Mignotte , PRL 2011**: image segmentation

Why clustering?

Example: **Li et al., IEEE Trans Inf Theory, 2004**

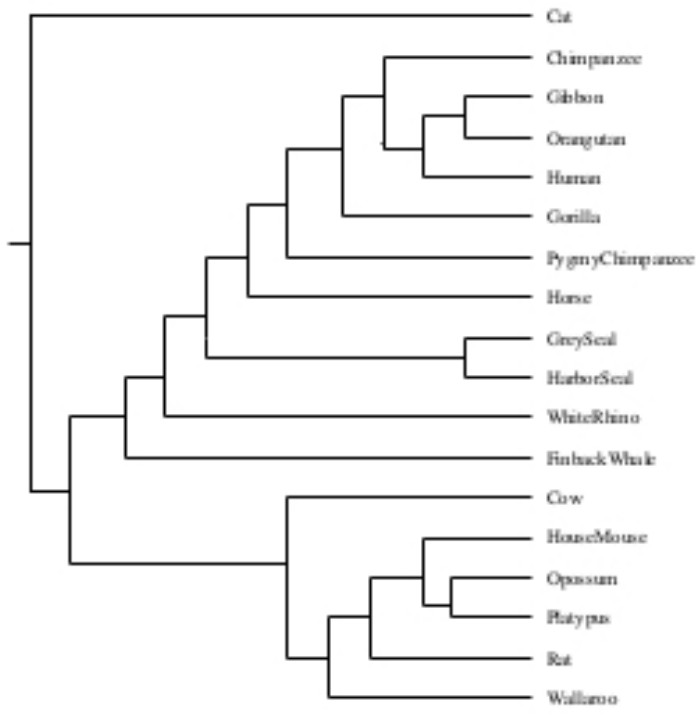


Fig. 1

THE EVOLUTIONARY TREE BUILT FROM COMPLETE MAMMALIAN MTDNA SEQUENCES USING FREQUENCY OF k-MERS.

Two classes of algorithms

- flat clustering the data space is partitioned into a number of subsets (the most "meaningful"); requires the number of partitions to be specified
- hierarchical clustering: build a nested hierarchy of partitions

Outline

Outline

Assume a generative model:

- the observed data $\mathcal{D} = \{\mathbf{x}_i | i = 1, \ldots, n\}$ originates from G groups
- each group has a prior $P(g_k)$, $k = 1, \ldots, G$
- the class-conditional probabilities are of some known parametric form

$$p(\mathbf{x}|g_k, \boldsymbol{\theta}_k)$$

- the labels of the points $\mathbf{x}$ are unknown as are the parameters $\boldsymbol{\theta}_k$

PDF:

$$p(\mathbf{x}|\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_G\}) = \sum_{i=1}^{G} p(\mathbf{x}|g_i, \boldsymbol{\theta}_i) P(g_i)$$

- this is a mixture density
- $p(\cdot)$ are the *mixture components* and $P(\cdot)$ are the *mixing parameters*
- the goal is to estimate $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_G$

Maximum likelihood estimates

Let $\mathbf{T} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_G\}$. The *likelihood* of the observed data is

$$p(\mathcal{D}|\mathbf{T}) = \prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{T})$$

The *maximum likelihood estimate* $\hat{\mathbf{T}}$ is

$$\hat{\mathbf{T}} = \arg\max_{\mathbf{T}} p(\mathcal{D}|\mathbf{T})$$

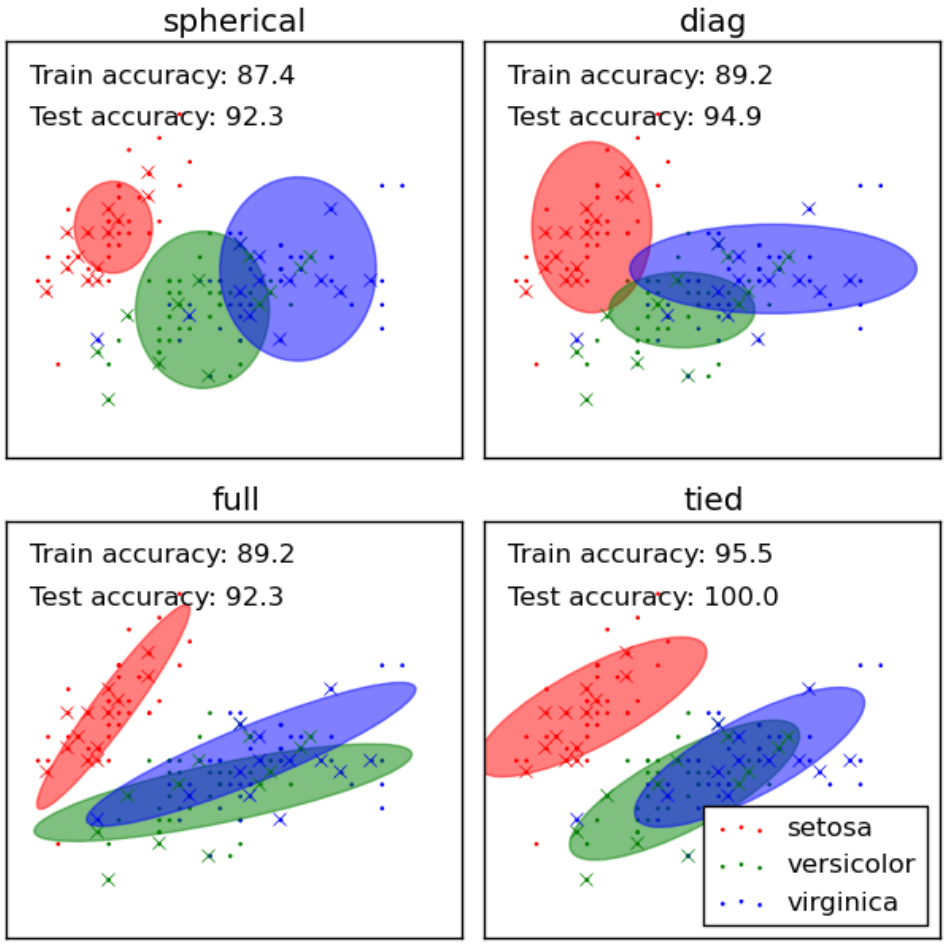
The maximum likelihood estimate is usually obtained through an iterative process: expectation maximization.
Repeat until convergence:

1. **Expectation step** (E-step) compute the expected log-likelihood under current estimates $\hat{\mathbf{T}}^{(t)}$
2. **Maximization step** (M-step) find $\hat{\mathbf{T}}^{(t+1)}$ that maximizes the expectation

Gaussian Mixture Models

- depending on how constraint is the model, different forms can be fitted
- usually, the constraints are on the form of the covariance matrices: spherical, diagonal, full

Example (from scikit-learn):



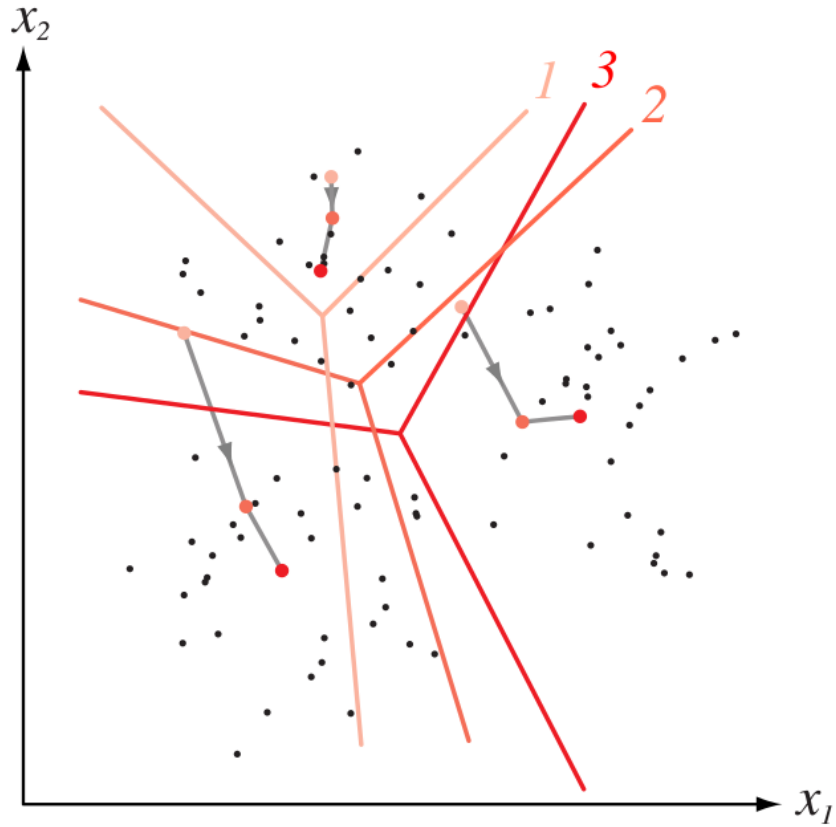
Outline

- given are n data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and the number K of clusters
- in the Gaussian mixture model, assume the covariance matrices to be identity matrices $\rightarrow$ the Mahalanobis distance becomes Euclidean distance
- goal: find the K cluster centres (based on Euclidean distance)

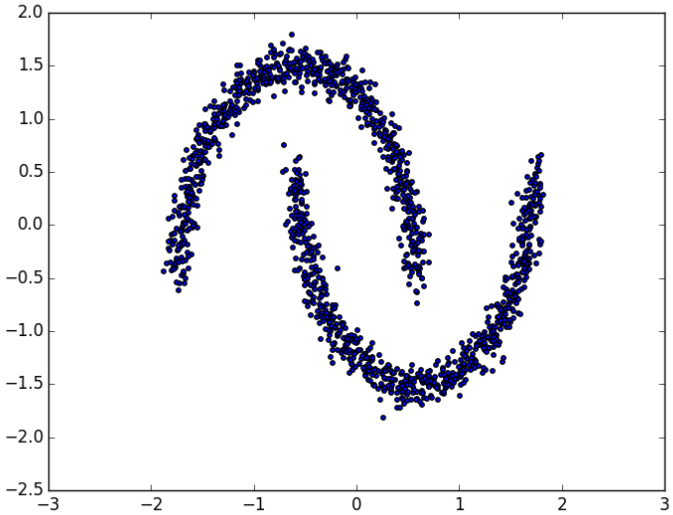
Overview of the algorithm:

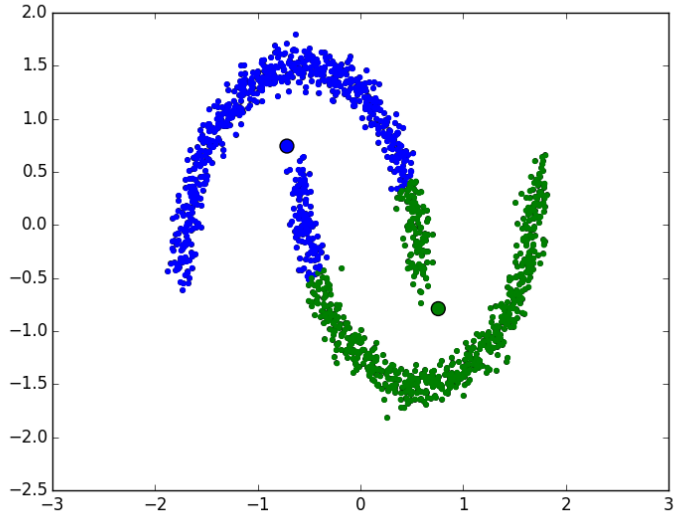
1. choose randomly K cluster centres
2. assign all the points to the cluster defined by the closest centre
3. re-compute the cluster centres
4. repeat 2-3 until no more changes (or some other convergence criterion)

[DHS -Fig 10.3]



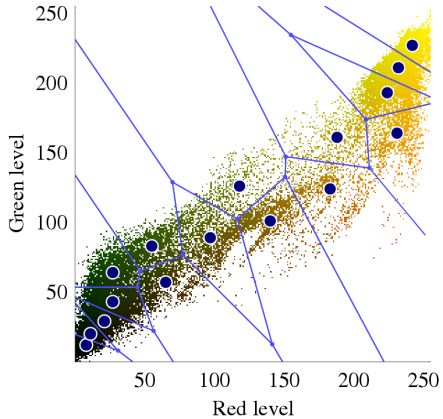
Example:


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3
−2
−1
0
1
2
3


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3
−2
−1
0
1
2
3

Application of K–means clustering: vector quantization

Goal: find a limited number of levels to represent a (possibly continuous) range of values.
Example: color quantization. Reduce the number of colors in an image in an adaptive way. (From Wikipedia:)



"Rosa Gold Glow 2 small noblue color space". Licensed under Public domain via Wikimedia Commons

Outline

Some definitions

- **Graph:** A *(simple, unoriented) graph* G is a pair of sets $G = (V, E)$, where V is a vertex set and E is an edge set.
- An *edge* $(i, j) \in E$ is an unordered pair of vertices $i, j \in V$.
- **Oriented graph**: A *oriented graph* G is a pair of sets $G = (V, E)$, where V is a vertex set and E is an arc set. An *arc* $(i, j) \in E$ is an ordered pair of vertices $i, j \in V$, with i called *tail* and j called *head.*

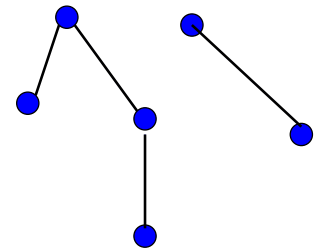


Figure: A simple graph.

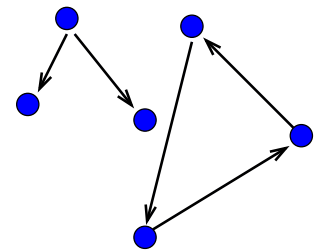


Figure: An oriented graph.

- A *path* of length r from i to j is a sequence of $r + 1$ distinct and adjacent vertices starting with i and ending with j.
- A *connected* graph is a graph where any two vertices may be linked by a path.
- A *connected component* is an induced subgraph maximal that is connected.
- The *degree* d_i of a vertex i is the number of incident edges to i.
- The *in–degree* $d_i^{(i)}$ of a vertex i is the number of arcs ending with i. The *out–degree* $d_i^{(o)}$ of a vertex i is the number of arcs starting with i.

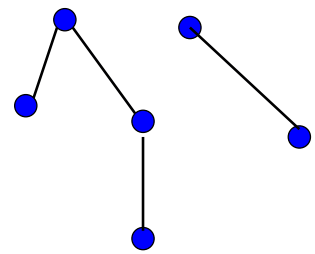


Figure: A graph with 2 connected components.

Graph isomorphism

Two graphs G_1 and G_2 are *isomorphic* if there exists a bijection $\varphi : V_{G_1} \to V_{G_2}$ such that $(i,j) \in E_{G_1}$ iff $(\varphi(i), \varphi(j)) \in E_{G_2}$.

Adjacency and incidence matrices

The *adjacency matrix* $A(G) = [a_{ij}]$ of a graph G is a $|V| \times |V|$ 01–matrix, with $a_{ij} = \mathbb{I}_{(i,j)\in E}$. The *incidence matrix* $B(G) = [b_{ij}]$ of a graph G is a $|V| \times |E|$ matrix with $b_{ik} = -1$ and $b_{jk} = 1$, where $k = (i,j) \in E$.

Properties:

- for a graph G: $BB^t = \text{diag}(d_1, ..., d_{|V|}) - A$
- let G_1 and G_2 be two isomorph graphs; then

$$\det(xI - A(G_1)) = \det(xI - A(G_2)),$$

  so, they have the same spectrum.
- the number of paths of length r from i to j is given by $(A^r)_{ij}$.

Laplacian of a graph

The *Laplacian* of the graph G is the matrix $L(G) = BB^t$, where B is the incidence matrix of G.

Properties:

- $L_{ij} = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$
- L is symmetric (by construction) positive semi–definite
- $L\mathbf{1}_{|V|} = 0$ so the smallest eigenvalue is $\lambda_1 = 0$ and the corresponding eigenvector is $\mathbf{1}_{|V|}$.
- the number of connected components of G equals the number of null eigenvalues.

Warning: there are various other variants of L's definition (e.g. admittance matrix, Kirchhoff matrix).

Generalizations:

- A is replaced by a *weight/similarity matrix* W (still symmetric)
- the *degree* of a vertex i becomes $d_i = \sum_{j=1}^{|V|} w_{ij}$
- the Laplacian becomes (see its properties): $L = D - W$, where $D = \text{diag}(d_1, ..., d_{|V|})$
- the *normalized Laplacian* is defined as $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$.

Overview of spectral clustering

Spectral clustering: partitioning the similarity graph under some constraints:

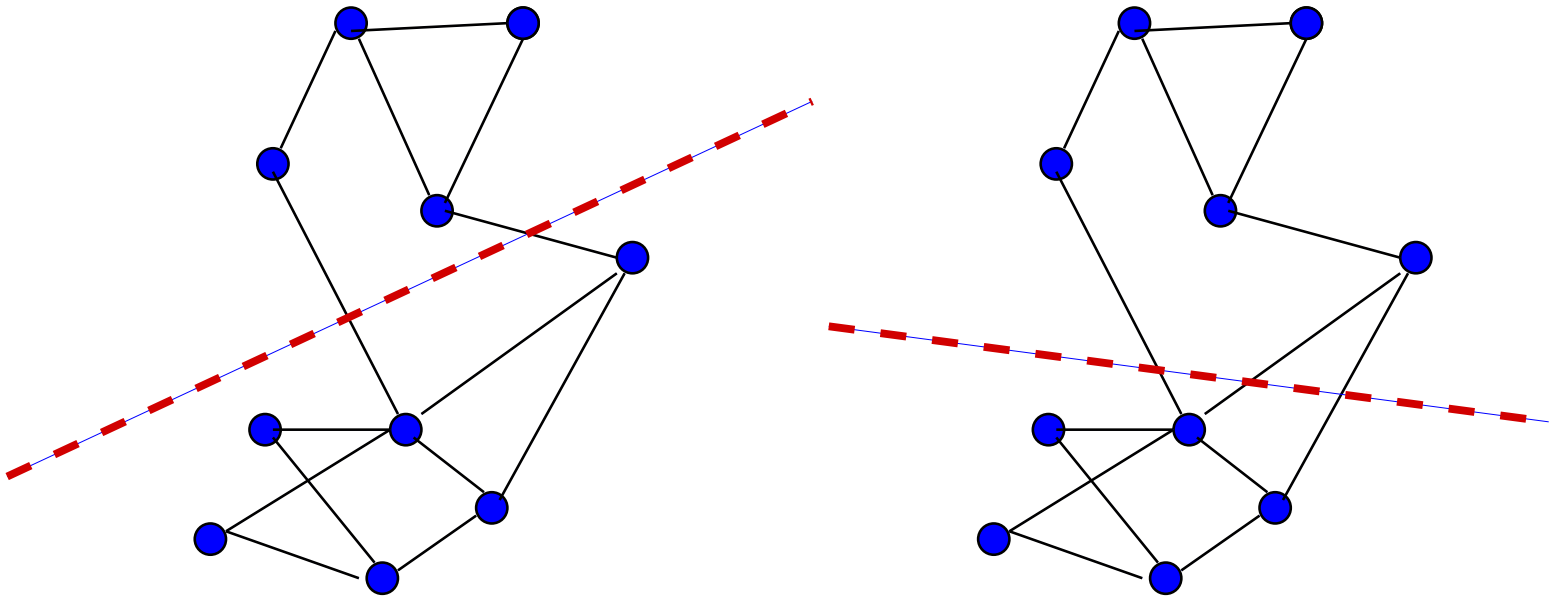


Figure: Where to cut??

- balance the size of the clusters?
- minimize the number of edges removed?
- ...

Let A, B be the two clusters/subgraphs and let $s(A, B) = \sum_{i \in A} \sum_{j \in B} w_{ij}$. Objective functions:

- *ratio cut*: $J(A, B) = \frac{s(A,B)}{|A|} + \frac{s(A,B)}{|B|}$, i.e. minimize similarity between A and B
- *normalized cut*: $J(A, B) = \frac{s(A,B)}{\sum_{i \in A} d_i} + \frac{s(A,B)}{\sum_{i \in B} d_i}$
- *min–max–cut*: $J(A, B) = \frac{s(A,B)}{s(A,A)} + \frac{s(A,B)}{s(B,B)}$, i.e. minimize the similarity between A and B and maximize the similarity within A and B.

All these lead to finding the smallest eigenvectors/eigenvalues of $L = D - W$.

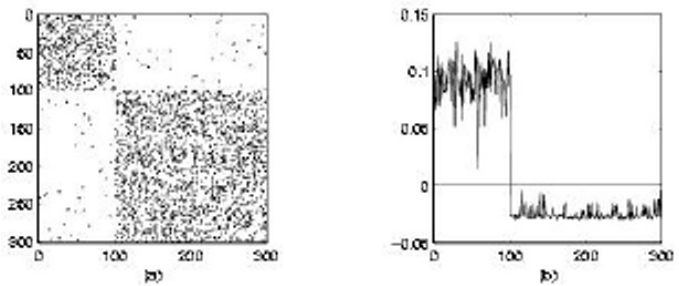
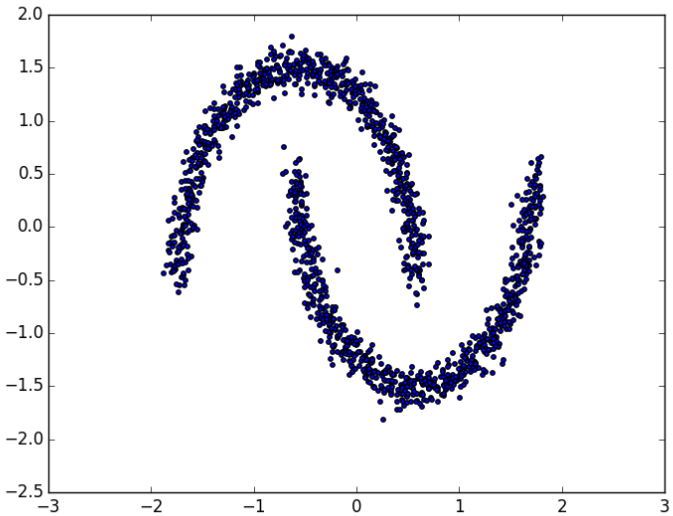


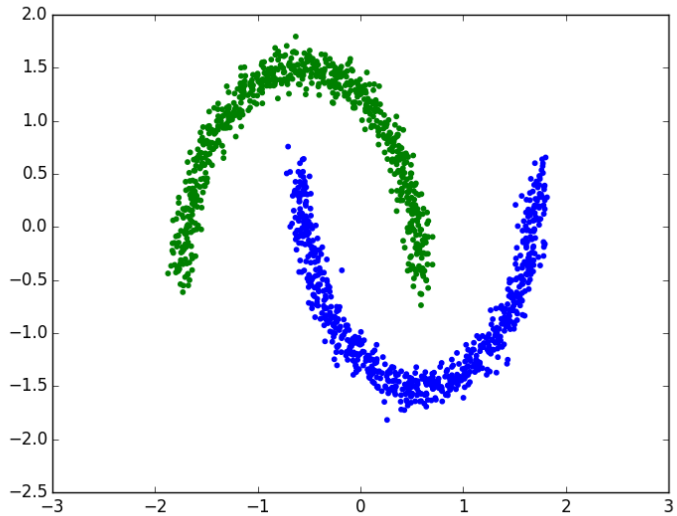
Figure: Adjacency matrix and the 2nd eigenvector

Spectral clustering - main algorithm

- input: a similarity matrix $\mathbf{S}$ and the number of clusters k
- compute the Laplacian L
- compute the eigenvectors $\mathbf{v}_i$ of L end order them in increasing order of the corresponding eigenvalues
- build $\mathbf{V} \in \mathbb{R}^{n \times k}$ with eigenvectors as columns
- the rows of $\mathbf{V}$ are the new data points $\mathbf{z}_i \in \mathbb{R}^k$ to be clustered
- use $k-$means to cluster $\mathbf{z}_i$

Example:

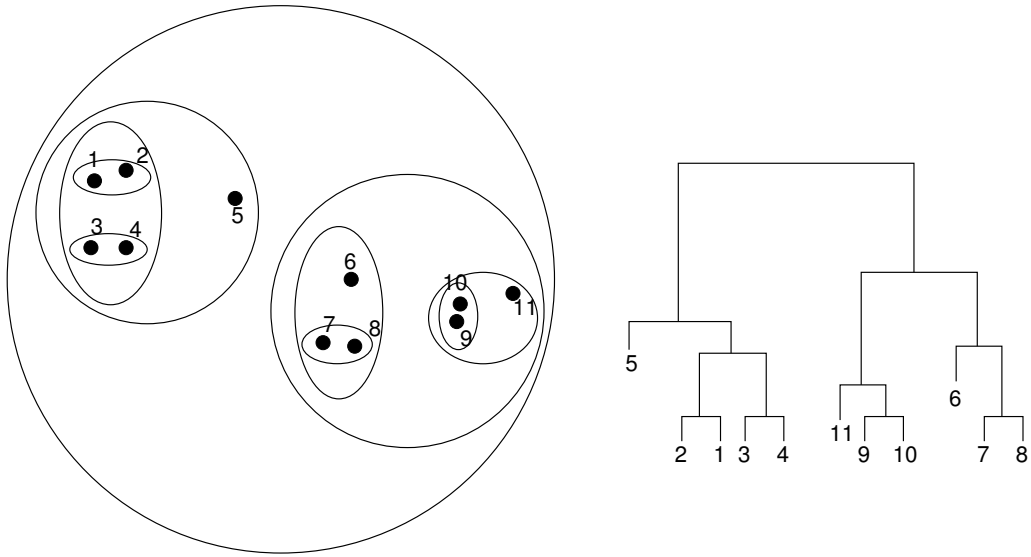

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3
−2
−1
0
1
2
3


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3
−2
−1
0
1
2
3

Outline

Hierarchical clustering

A nested set of partitions and the corresponding *dendrogram*:



- the height of the descendent segments is proportional with the distance/similarity between the partitions
- the ordering is irrelevant (i.e. you can swap left and right sub-trees without altering the meaning)
- it can be seen as a density estimation method

There are two main approaches to construct a hierarchical clustering:

- *agglomerative*: initially, each point is alone in a cluster; the *closest* two clusters/groups are merged to form a new cluster
- *divisive* starts with all points in a single cluster, which is iteratively split

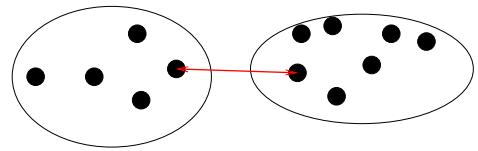
Outline

Linkage algorithms

- bottom-up/agglomerative strategy
- start with each point in its own cluster
- merge the two closest clusters
- continue until there is only one cluster
- Johnson: Hierarchical clustering schemes. Psychometrika, 2:241-254, 1967
- many applications, including phylogenetic trees
- the key is to define a distance over clusters space

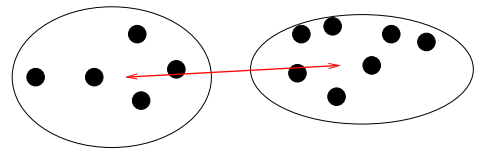
- single linkage:

$$\delta(C_1, C_2) = \min_{\mathbf{x} \in C_1, \mathbf{z} \in C_2} d(\mathbf{x}, bz)$$



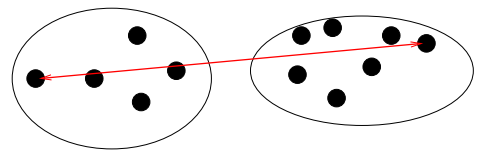
- average linkage:

$$\delta(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{\mathbf{x} \in C_1, \mathbf{z} \in C_2} d(\mathbf{x}, \mathbf{z})$$



- complete linkage:

$$\delta(C_1, C_2) = \max_{\mathbf{x} \in C_1, \mathbf{z} \in C_2} d(\mathbf{x}, \mathbf{z})$$



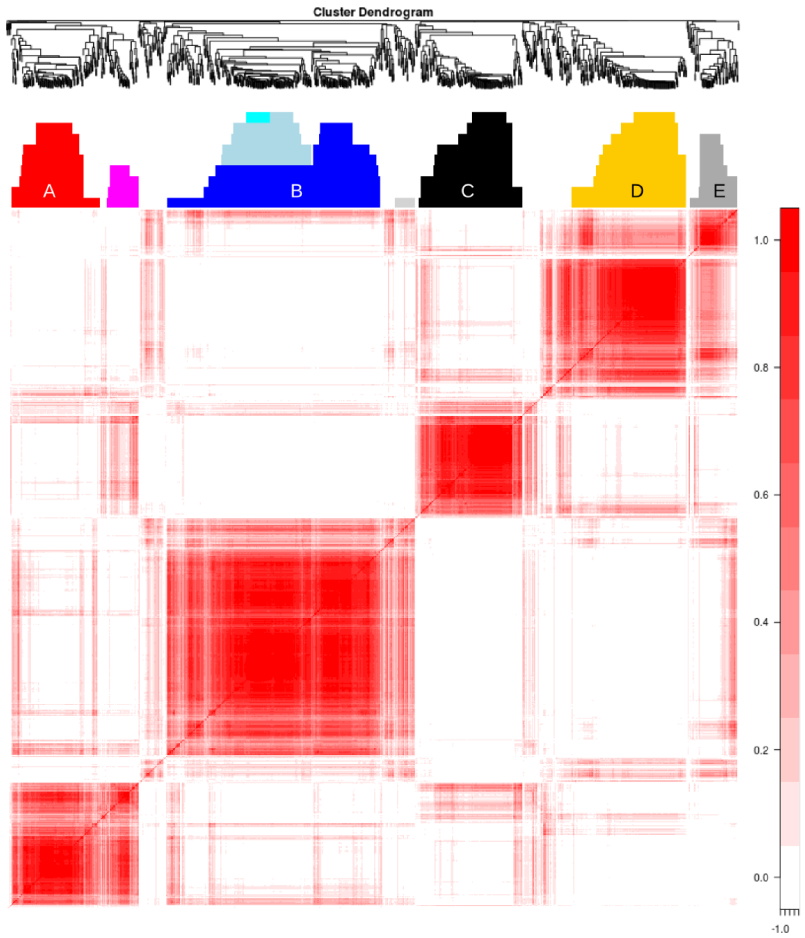
- other variations...

Comments:

- single linkage tends to produce non-balanced trees, with long "chains"
- complete linkage leads to more compact clusters
- single linkage: minimum spanning tree (cut the longest branch in MST and get the first two clusters)
- sensitive to outliers
- does not need a pre-specified number of clusters - but often you have to cut the dendrogram and to decide how many clusters are in data
- the clustering tree is not unique

Example: discovery of molecular subtypes in CRC

See Budinska et al, J Path. 2012



Outline

Outline

Other methods: information-theoretic methods

- idea: construct a clustering the preserves most of the "information" in data
- hence, you need a cost function (distortion function)
- density estimates based on frequencies (counts)
- no notion of similarity
- example: Information Bottleneck

Other methods: iterative refinement

- Karypis, Kumar: multilevel partitioning of graphs, SIAM J Sci Comp 1999
- start with a large graph
- merge nodes in "super-nodes" (coarsen the graph)
- cluster the coarse graph
- uncoarsen again: refine the clustering

Other methods: ensemble methods

- produce a series of clusterings based on perturbed versions of the original data (resampling, different parameters, etc)
- use the ensemble of clusters to decide for the final clustering
- see Strehl, Ghosh: cluster ensembles, JMLR 2002

Other methods

- support vector clustering
- subspace clustering
- co-clustering (bi-clustering): obtain a clustering of rows and columns of the data matrix
- etc. etc.

Outline

Cluster quality

- some cluster methods (e.g. k-means) define an objective function as the goal of the clustering
- there are quality functions that are algorithm independent (many!)
- two quantities are usually accounted for:
  - *within-cluster similarity* (cluster homogeneity): should be high
  - *between-cluster similarity*: should be low
- while mathematically attractive, they usually lead to an NP-hard problem
- the choice of quality function is rather ad-hoc

Cluster stability and optimal k

General idea:

- start with a data set $\mathcal{D} = \{\mathbf{x}_i\}$ and some clustering algorithm $\mathcal{A}$
- for various number of clusters (e.g. $k = 2, 3, \ldots, K$:
  - draw subsamples from $\mathcal{D}$
  - use $\mathcal{A}$ to cluster them into k clusters
- compare the resulting clusters by using a distance between clusterings and compute a *stability index* describing how variable/stable the clustering distances are
- choose k that gives the best stability

Distances between clusterings of the same data: let $f^{(1)}$ and $f^{(2)}$ be two clusterings and define N_{ij} as the number of pairs $(\mathbf{x}, \mathbf{z})$ for which $f^{(1)}(\mathbf{x}, \mathbf{z}) = i$ and $f^{(2)}(\mathbf{x}, \mathbf{z}) = j$, for $i, j \in \{0, 1\}$.
Similarity/distance functions (some of many):

- *Rand index*: $(N_{00} + N_{11})/[n(n-1)]$
- *Jaccard index*: $N_{11}/(N_{11} + N_{01} + N_{10})$
- *Hamming distance*: $(N_{01} + N_{10})/[n(n-1)]$
- information theoretic distance: Entropy($f^{(1)}$) + Entropy($f^{(2)}$) - Mutual information($f^{(1)}, f^{(2)}$)

What if the clusterings are not defined on the same data?

- far from being trivial
- idea "extend" clustering $f^{(1)}$ to $\mathcal{D}_2$ and $f^{(2)}$ to $\mathcal{D}_1$
- some clustering algorithms are easily extended: $k-$means just requires assignment of new data to the defined clusters, etc.
- other clustering algorithms are not so flexible (e.g. spectral clustering)
- use some classifiers for extension... what about classification errors?
- what if the 2 datasets are not exactly "aligned"?

How many clusters?

- most flat clustering algorithms require k
- the hierarchical clustering usually is cut at some height to yield the final number of clusters
- e.g. image segmentation $k =$?
- one way, use cluster quality to choose best k
- or use the stability approach
- or use "gap statistic" - see Tibshirani et al, J Royal Statist Soc 2001

Outline

Cluster validation

- given a clustering, how confident are we that it represents "real" groups of points
- if a new data set is available, would we obtain the same clusters? and as many as before?
- complication: in real applications, data may not always come from the same conditions (e.g. change of measurement device, etc)
- maybe the original data set does not capture all the clusters that could arise in data
- no clear method for validation, but the ideas are the same as for cluster stability and quality

Example (Budinska et al, J Path 2012):