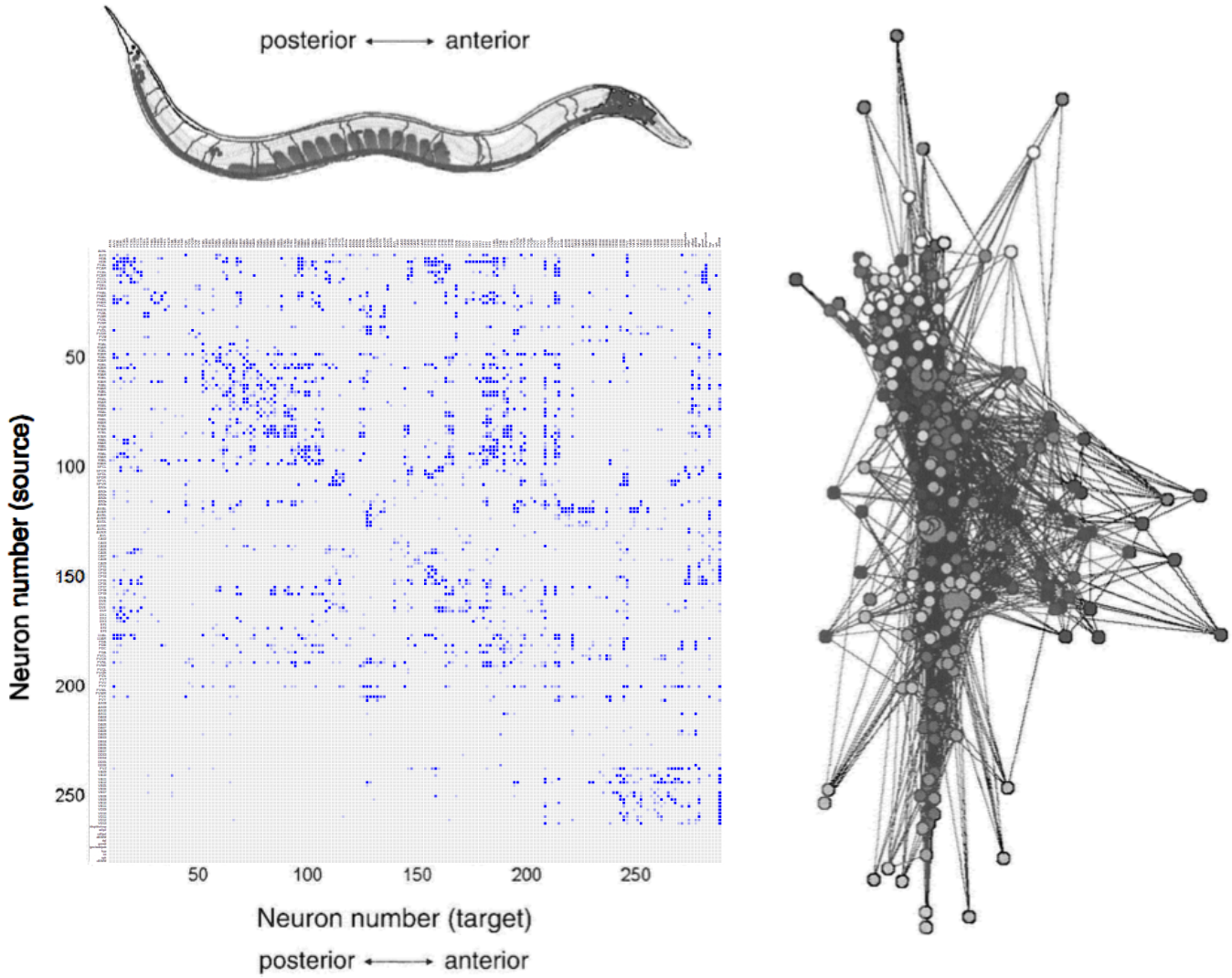
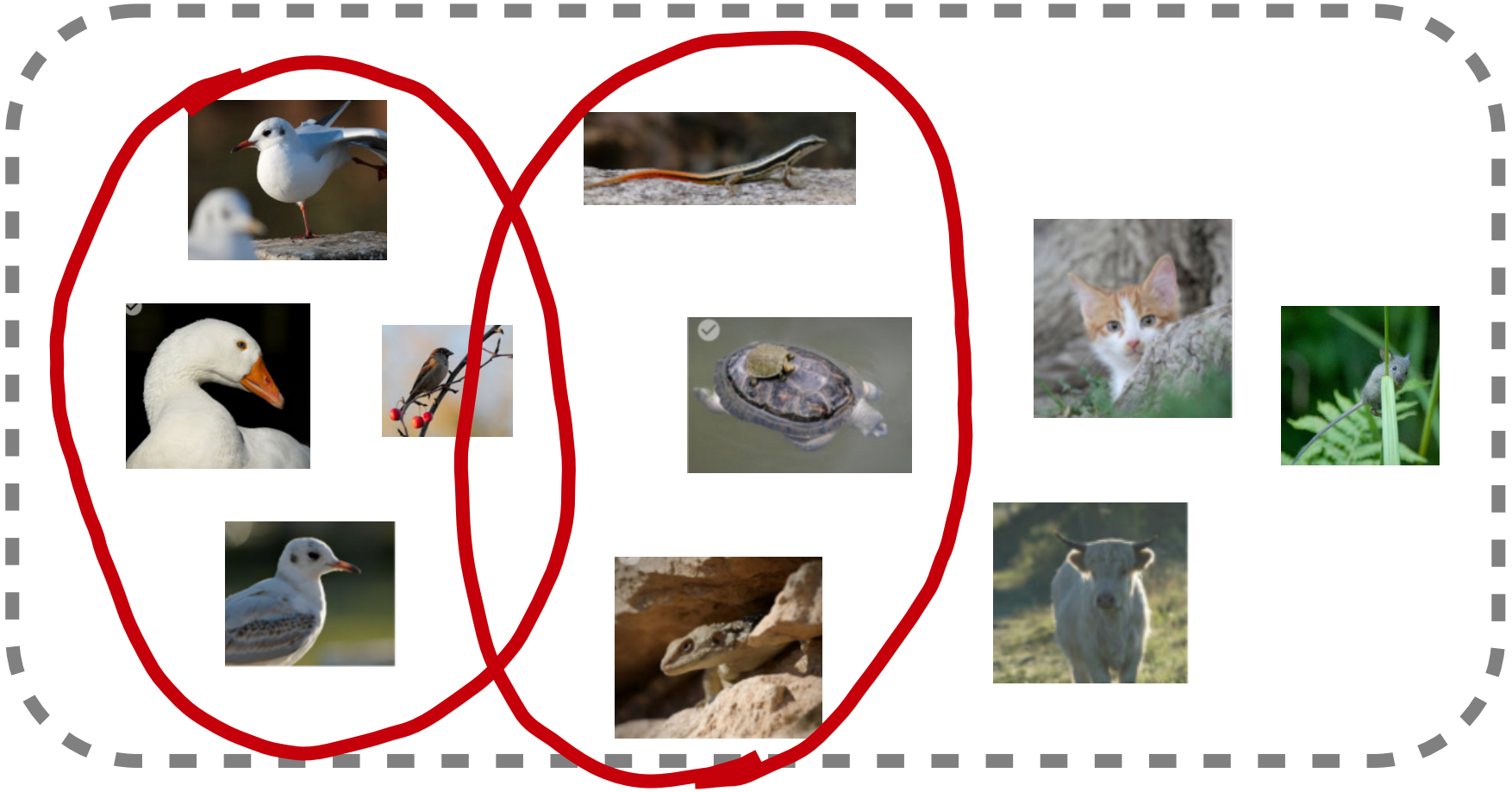
Clustering and Graphs

Data Mining for Business and Governance
2017-10-10
Guest Lecture - Martin Atzmueller

Clustering and Graph Analysis



Clustering

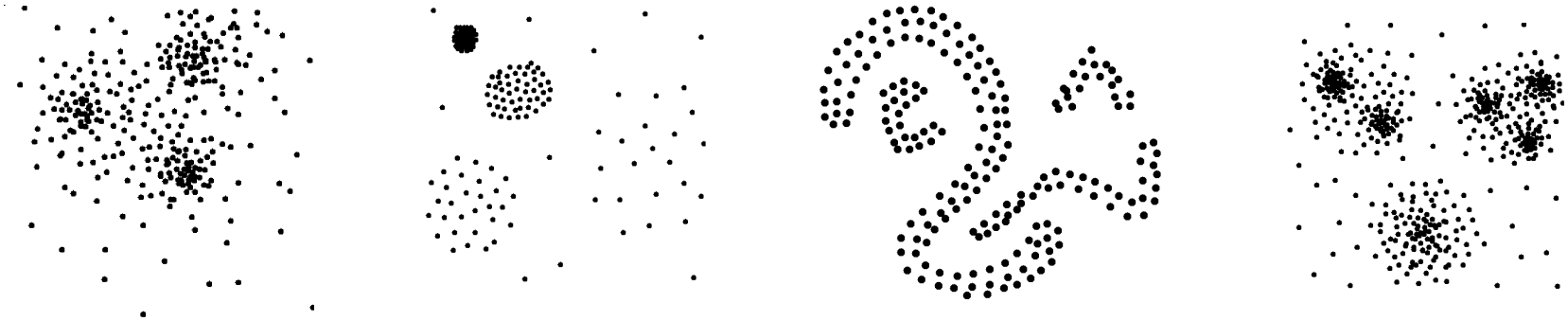


Clustering & Graph Analysis

- Introduction
  - Goals of clustering
  - Distance functions
  - Applications, types of algorithms
- Partitioning Methods
  - k-means, k-medoid, expectation maximization
  - Initialization and parameter selection
  - Density-based clustering
- Hierarchical Methods
- Graph clustering
  - Communities
  - Clique percolation method
  - Divisive hierarchical clustering

Clustering - Goals

- Identify a finite set of categories, classes or groups (clusters) in the data
- Objects in the *same* cluster should be as similar as possible
- Objects in *different* clusters should be as dissimilar as possible



Clusters of different sizes, forms and density
hierarchical clusters

Distance Functions

- Formalizing similarity
  - Sometimes: similarity function
  - Typically: distance function *dist*(o_1,o_2) for pairs of objects o_1 und o_2
  - Small distance ≈ similar objects
  - Large distance ≈ dissimilar objects
- Requirements for distance functions

  (1) $dist(o_1, o_2) = d \in R^{\geq 0}$
  (2) $dist(o_1, o_2) = 0$ iff. $o_1 = o_2$
  (3) $dist(o_1, o_2) = dist(o_2, o_1)$ (Symmetry)
  (4) *additionally, for metrics* (triangular inequality)
  $$dist(o_1, o_3) \leq dist(o_1, o_2) + dist(o_2, o_3).$$

Distance Functions/ Numeric/Categorical Attributes

- Objects $x = (x_1, ..., x_d)$ and $y = (y_1, ..., y_d)$
- *General L_p-metric* (Minkowski-distance) $dist(x,y) = \sqrt[p]{\sum_{i=1}^{d} |(x_i - y_i)|^p}$
- *Euclidean distance* ($p = 2$) $dist(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$
- *Manhattan distance* ($p = 1$) $dist(x,y) = \sum_{i=1}^{d} |x_i - y_i|$
- *Maximum metric* ($p = \infty$) $dist(x,y) = \max\{|x_i - y_i| | 1 \le i \le d\}$
- Categorical attributes (Hamming distance) $dist(x,y) = \sum_{i=1}^{d} \delta(x_i, y_i)$ with $(x_i, y_i) = \begin{cases} 0, & \text{if } x_i = y_i \\ 1, & \text{otherwise} \end{cases}$
- A popular *similarity* function: Correlation coefficient $\in$ [-1,+1]

Example Application
Web Session Clustering

- Entries of a web log

```
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:44:50 +0100] "GET /~lopa/ HTTP/1.0" 200 1364
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:45:11 +0100] "GET /~lopa/x/ HTTP/1.0" 200 712
fixer.sega.co.jp unknown - [04/Mar/1997:01:58:49 +0100] "GET /dbs/porada.html HTTP/1.0" 200 1229
scooter.pa-x.dec.com unknown - [04/Mar/1997:02:08:23 +0100] "GET /dbs/kriegel_e.html HTTP/1.0" 200 1241
```

- Generate sessions:
- Session::= <IP-adress, User-Id, [URL_1, . . ., URL_k]>

⇒ which entries compose a session?

- Distance function for sessions: Jaccard-Coefficient $d(x,y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|}$

Types of Clustering Methods

- Partitioning methods
  - Parameter: *k* – number of clusters, distance function
  - Searches for a "flat“ clustering into *k* clusters with minimal costs
- Hierarchical methods
  - Parameter: Distance function for points and clusters
  - Determines hierarchy of clusters, combines respective most similar clusters
- Density-based methods
  - Parameter: minimal density within a cluster, distance function
  - Extends points with their numbers as long as density is large
- Other clustering methods
  - Fuzzy clustering
  - Graph-theoretical methods
  - ...

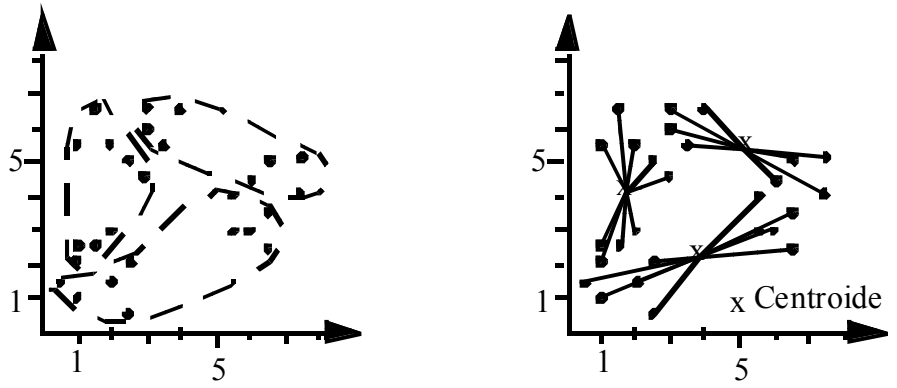
Partitioning

- Goal: Partitioning into *k* clusters with minimal costs
- Locally optimizing algorithm
  - Choose *k* initial cluster representatives
  - Optimize these iteratively
  - Assign each object to ist most similar representative (cluster)
- Types of cluster representatives
  - Cluster mean (*Constructing central points*)
  - Cluster element (*Selecting central points*)
  - Probability distribution of the cluster (*Expectation maximization*)

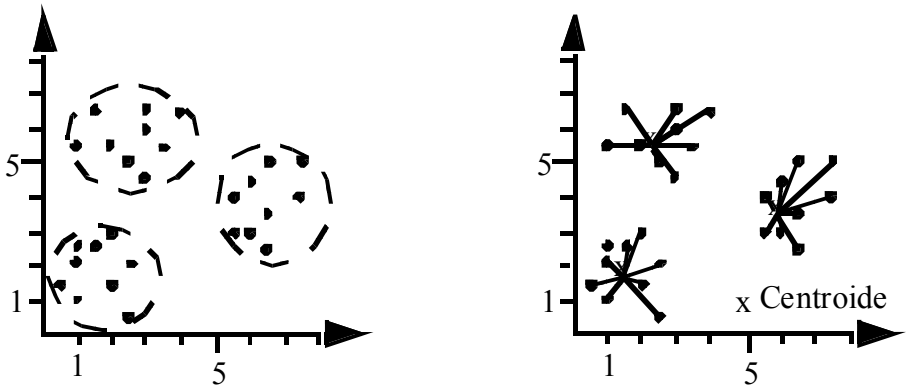
Constructing Central Points

Cluster **Cluster representatives**

Bad clustering



Optimal clustering



Constructing Central Points

- Objects are points $p=(x^p_1, ..., x^p_d)$ in euclidean vector space
- Distance function: euclidean distance
- *Centroid* μ_C: Mean of all points in cluster C
- ***Cost measure*** (compactness) *of a cluster C*

$$TD^2(C) = \sum_{p \in C} dist(p, \mu_C)^2$$

- ***Cost measure*** (compactness) *of a clustering*

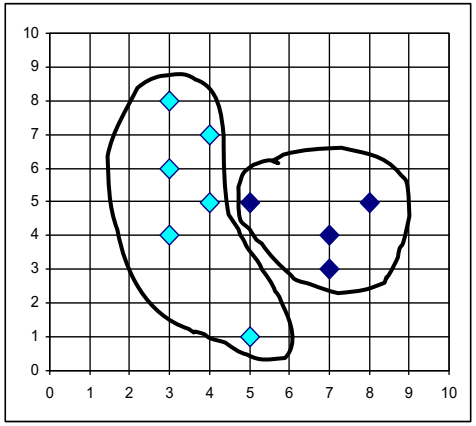
$$TD^2 = \sum_{i=1}^{k} TD^2(C_i)$$

Constructing Central Points Basic Algorithm

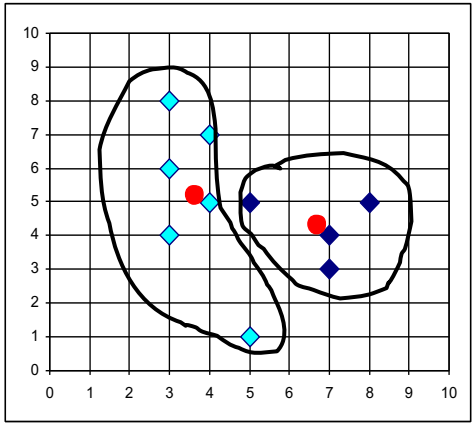
- **ClusteringThroughMinimizingVariance** (Points D, Integer k)

```
Generate an "initial" Partitioning of the set
  of points D into k classes;
Calculate set C'={C1, ..., Ck} denoting the
  centroids for these k classes;
C = {};
repeat until C = C'
    C = C';
    Form k classes by assigning each point to
      the closest centroid in C;
    Calculcate the set C'={C'1, ..., C'k} of
      centroids for the classes (determined in
      the previous step);
return C;
```

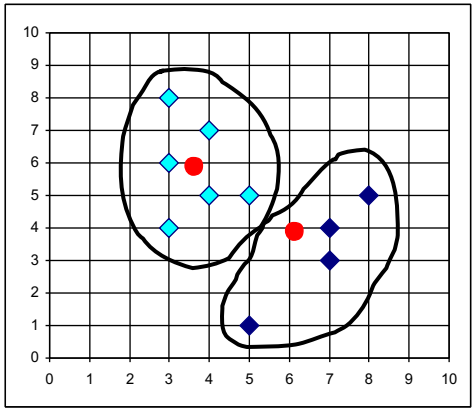
Constructing Central Points



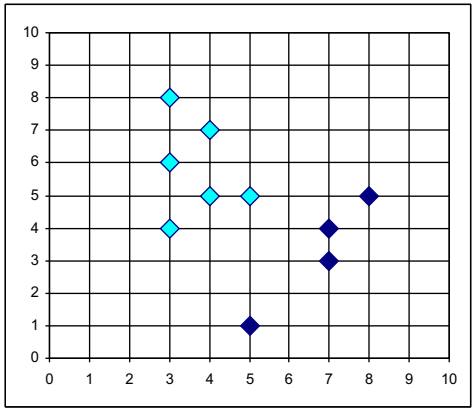
Calculate new centroids



Assign to new centroids



Calculate new centroids



Constructing Central Points - Variants of the Basic Algorithm

- *k-means* [Lloyd 57, MacQueen 67]
  - Basic idea: the centroids are directly updated whenever a point changes its cluster
  - *K*-means – conforms to overall properties of the basic algorithm
  - *However: K*-means is order-dependent
- ISODATA [Ball & Hall, 65]
  - Based on k-means
  - Improvement of the results through
    - Elimination of very small clusters
    - Combination and split operations
  - However: User needs to supply many additional parameters
- K-Means & careful seeding ➔k-means++

Constructing Central Points

- "+"
  - Efficiency: Linear complexity for one iteration
  - Number of iterations is typically small.
  - Simple to implement

- K-means is the most popular partitioning clustering method

- "-"
  - Problem: Noise and outliers
  - All objects contribute to the calculation of the centroid
  - Only convex shaped clusters
  - Often difficult to determine number of clusters (k)
  - Strongly dependent on the initial partitioning (both result quality as well as runtime)

Selecting Representative Points

- Only requires distance function for pairs of objects
- *Medoid*: a central element of the cluster (representative point)
- *Cost measure* (compactness) *of a cluster C*

$$TD(C) = \sum_{p \in C} dist(p, m_C)$$

- *Cost measure* (compactness) *of a clustering*

$$TD = \sum_{i=1}^{k} TD(C_i)$$

- Search space of the clustering algorithm: all partitions with *k* elements

➔ runtime complexity exponential in *k*

Selecting Representative Points

- *K-medoid (PAM)* [Kaufman & Rousseeuw 1990]
  - *"Partitioning around the medoid"*
    - Greedy algorithm: swap only one mediod with a non-medoid in each step
    - In each step: Swap the pair (medoid, non-medoid), with the largest reduction in costs
- *CLARANS* [Ng & Han 1994]
  - Two additional parameters: *maxneighbor* und *numlocal*
    - At most *maxneighbor* of randomly selected pairs are considered (Medoid, Non-medoid)
    - The first swap causing a reduction of the *TD* value is being selected
- The search for the *k* "mediods" is only repeated *numlocal* times

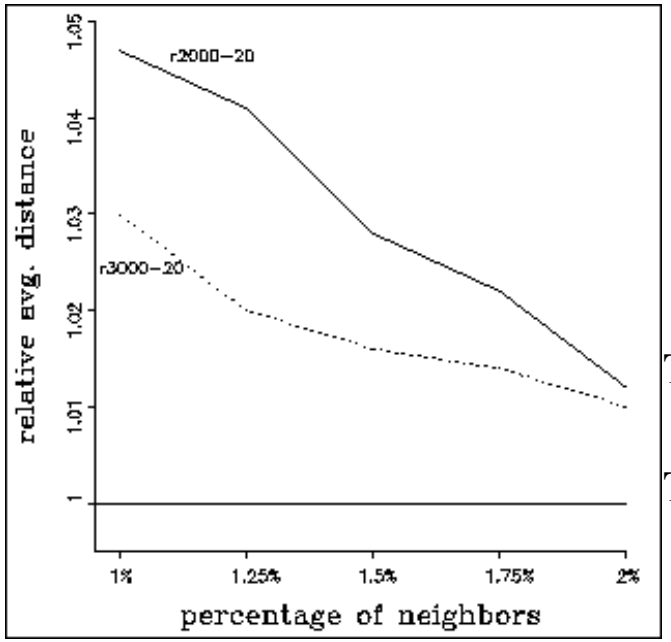
Selecting Representative Points

```
PAM(Object set D, Integer k, Float dist)
  Initialize the k medoids;
  TD_Change := −∞;
  while TD_Change < 0 do
     Calculate the value of TD_{N↔M} for each pair
       (Medoid M, Non-medoid);
     Choose the pair (M, N), for which the value
       TD_Change := TD_{N↔M} − TD is minimal;
     if TD_Change < 0 then
          Replace the Medoid M by the Non-Medoid N;
          Store the current set of medoids as the
            currently best partitioning;
  return medoids;
```

Selecting Representative Points

- *Comparison: PAM vs. CLARANS*

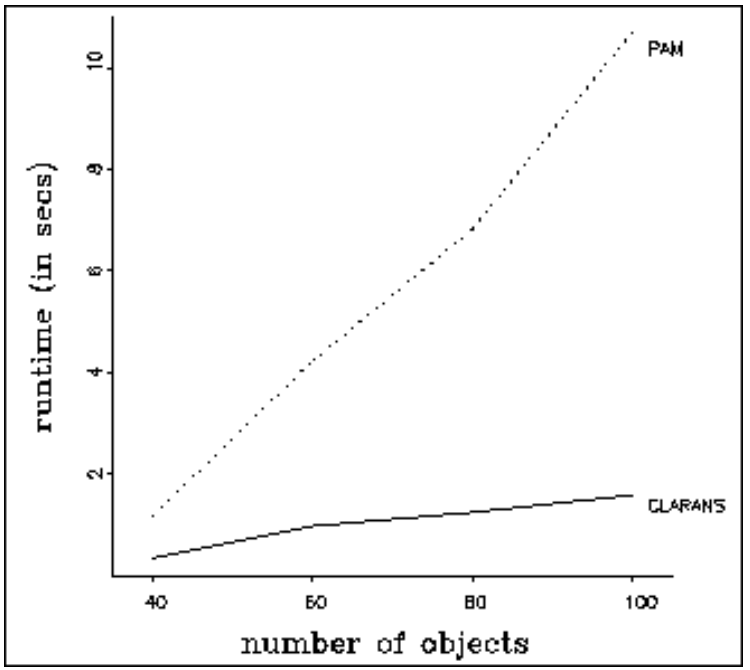
Quality



TD(CLARANS)

TD(PAM)

Runtime



Expectation Maximization

[Dempster, Laird & Rubin 1977]

- Objects are points $p=(x^p_1, ..., x^p_d)$ in euclidean vector space
- A cluster is described by a probablity distribution (typically normal distribution)
- A points belongs to different clusters with different probabilities
- Representation of a cluster *C*
    - Mean μ_C of all points of the cluster
    - *d x d* covariance matrix Σ_C for the points in cluster *C*
- Probability density of a cluster *C*

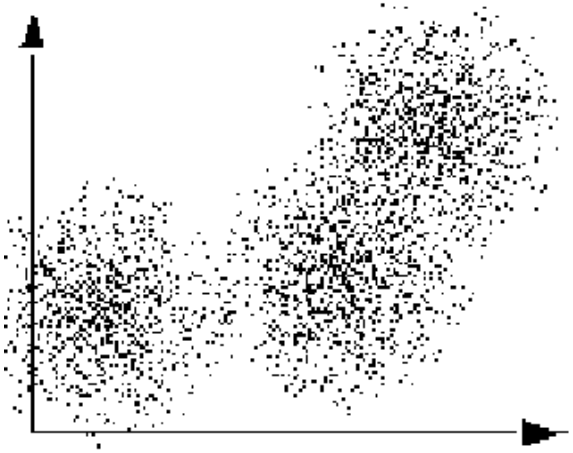
$$P(x \mid C) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma_C \mid}} e^{-\frac{1}{2}\cdot(x-\mu_C)^T \cdot (\Sigma_C)^{-1} \cdot (x-\mu_C)}$$

- Problem: Converges to a (possibly local) optimum
- Number of iterations is typically relatively high

Selecting Initial Clusterings

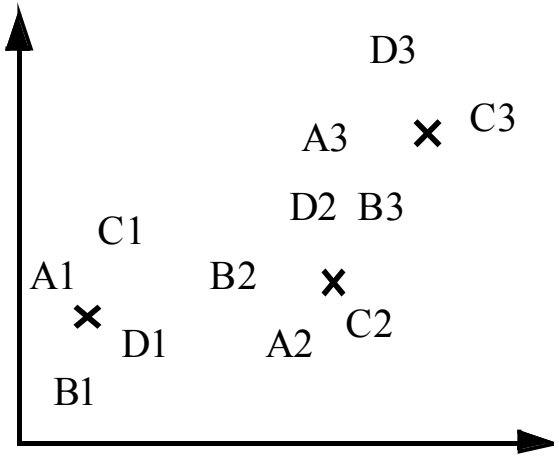
- Basic idea: Clustering of a small sample typically yields good initial clustering
- However: Distribution in samples often different from general population
- Method [Fayyad, Reina & Bradley 1998]
  - Generate *m* independent/different samples
  - Cluster each of the samples
    ➔ *m* different estimators for cluster centers
    - $A = (A_1, A_2, \ldots, A_k)$, $B = (B_1, \ldots, B_k)$, $C = (C_1, \ldots, C_k)$, . . .
  - Now, cluster the set $A \cup B \cup C \cup \ldots$ with m different initializations ...
  - Select the clustering with the best value (cost measure)

Selecting Initial Clusterings



Dataset

***k* = 3 Gaussian clusters**



All/Samples

***m* = 4 samples**

✕ true cluster centres

Which k should we select?

- Method
  - Determine a clustering for $k = 2, ..., n-1$
  - Select the "best“ clustering from the result set
- How to determine the "best“ clustering?
  - Measure needs to be independent of *k*
    - K-means/K-medoid: TD^2 and *TD* decrease monotonically with increasing *k*
    - EM: Cost measure increases *monotonically* with increasing *k*

➔ Silhouette coefficient

Silhouette Coefficient [Kaufman & Rousseeuw 1990]

- A cost measure independent of *k* for k-means/k-medoid
- Let
  - *a*(*o*) denote the distance of an object *o* to the representative of its cluster, and
  - *b*(*o*) the distance to the representative of the "second closest" cluster
- Silhouette *s*(*o*) of *o* $$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$
  - *s*(*o*) = -1 / 0 / +1: bad/ indifferent/ good clustering
- Silhouette coefficient s_C of a clustering
  - Average silhouette of all objects
- Interpretation
  - s_C > 0,7: "strong“ structure,
  - s_C > 0,5: adequate structure, . . .

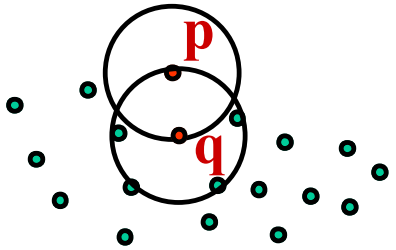
Density-based Clustering

- Basic Idea
  - Cluster as areas in d-dimensional space, where the objects are located "densely close to each other"
  - Separated by areas, in which the objects are not "densely close to each other“
- Requirements of density-based clusters
  - For every object of an cluster: Local (point) density exceeds a given threshold
  - The set of objects forming a cluster is "spatially connected"

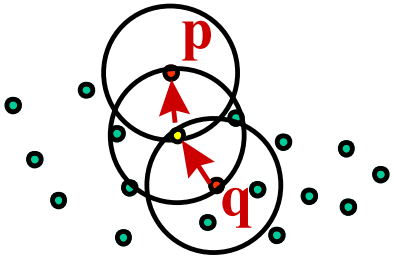
Density-based Clustering

- *DBScan* [Ester, Kriegel, Sander & Xu 1996]

- An object o $\in$ *O* is called *core object* w.r.t. ε and *MinPts,*, if:
- $|N_\varepsilon(o)| \geq MinPts,$ with $N_\varepsilon(o) = \{o' \in O \mid dist(o, o') \leq \varepsilon\}$.

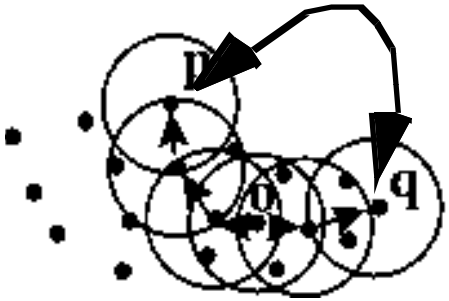


- An object $p \in O$ is *directly reachable* from $q \in O$ w.r.t. ε and *MinPts,* if: $p \in N_\varepsilon(q)$ and *q* is core object in *O.*
- An object *p* is *reachable* from *q*, if there exists a chain of directly reachable objects between *q* and *p*.



Density-based Clustering

- Two objects *p* und *q* are density-connected, if they are both reachable from an object *o.*



- A *cluster C* w.r.t. ε and *MinPts* is a non-empty subset of *O,* for which the following conditions hold:
  - *Maximality*: $\forall p,q \in O$: if $p \in C$ and *q* is reachable from *p*, then also $q \in C$.
  - *Connectedness*: $\forall p,q \in C$: *p* is density-connected to *q*.

Density-based Clustering

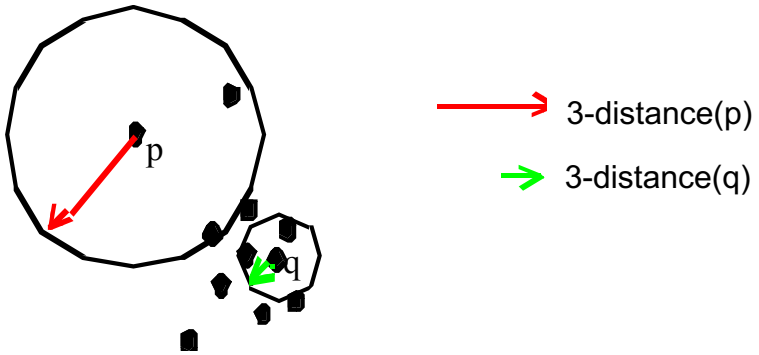
- Definition: Clustering
  - A density-based clustering *CL* of *O* w.r.t. ε and *MinPts* is the set of all density-based clusters w.r.t. ε and *MinPts* in *O*.
  - The set $Noise_{CL}$ is defined as the set of all objects in *O*, which do not belong to one of the density-based clusters C_i
- Basic properties
  - Let *C* denote a density-based cluster with $p \in C$ core object Then: $C = \{o \in O \mid o$ reachable from p w.r.t. ε and *MinPts*$\}$.

Density-based Clustering

- **DBSCAN**(Objects D, Real ε, Integer MinPts)
  // Initially all objects are not-classified,
  // o.ClId = not-classified for all o ∈ D
  ClusterId := nextId(NOISE);
  for i **from** 1 **to** |D| **do**
    object := D.get(i);
    if object.ClId = not-classified **then**
      if ExpandCluster(D, object, ClusterId, ε, MinPts)
      then ClusterId:=nextId(ClusterId);

Density-based Clustering

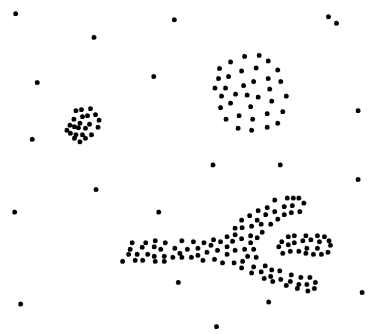
- Cluster: Density larger than the "threshold density" given by ε and *MinPts*
- How to detect the cluster with minimal density in the dataset?
- Heuristic: Consider distances to the *k*-nearest neighbors.

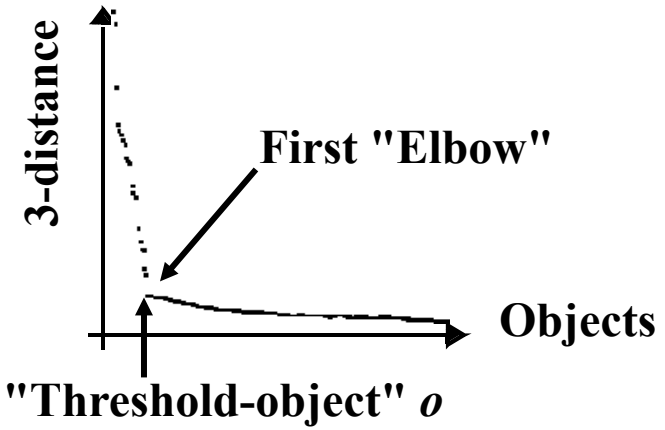


- Funktion *k-distance*: Distance of an object to its *k*-nearest neighbor
- *k-distance-diagram*: *k*-distances of all objects, in descending order

Density-based Clustering

- Example of a *k*-distance-diagram



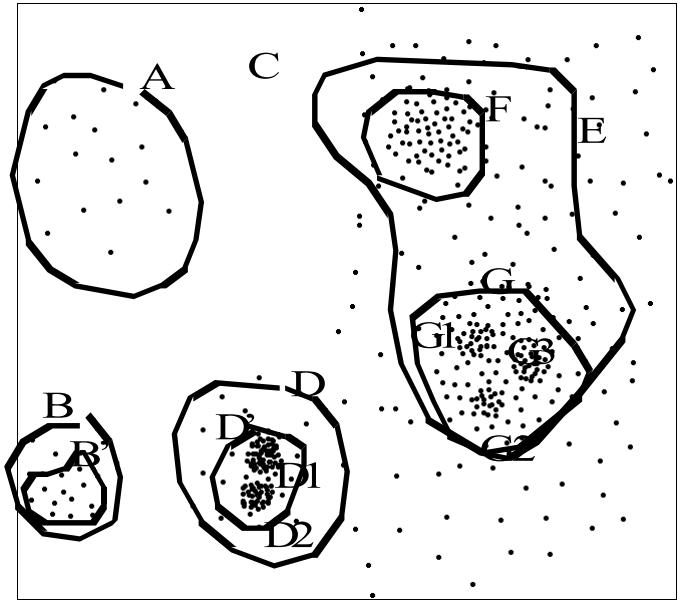


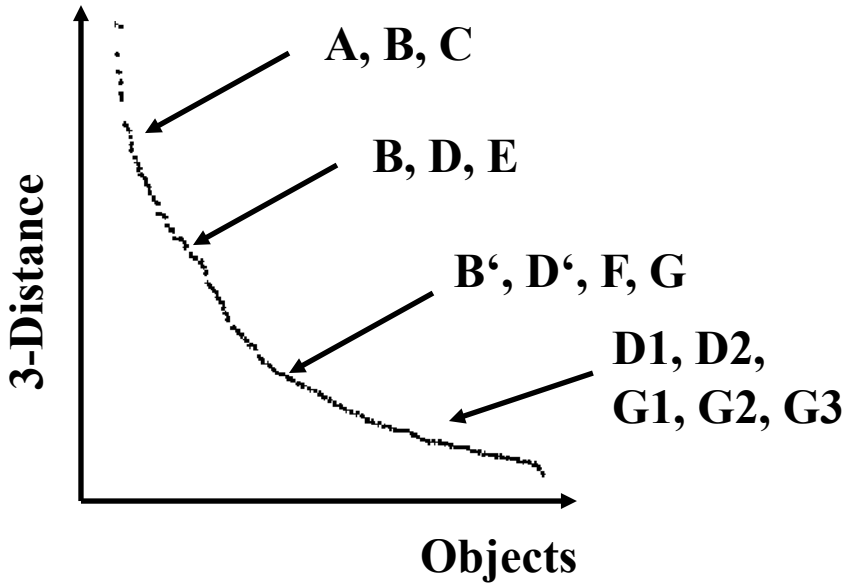
- Heuristic Method

- User provides a value for *k* (Default: *k* = 2**d* - *1*), *MinPts* := *k*+1.
- System calculates and visualizes the *k*-distance-diagram.
- User selects suitable object *o* in the *k*-distance-diagram, ε := *k*-distance(*o*).

Density-based Clustering

- *Problems of parameter estimation*
  - Hierarchical clusters
  - Significantly different density in partitionings of the input space
  - Clusters and noise are not well separated



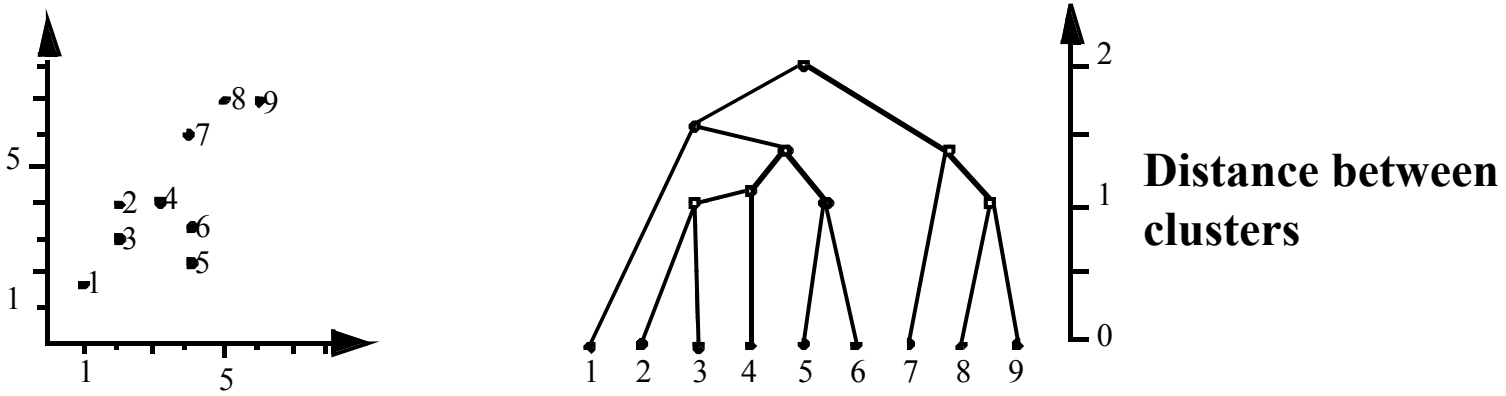


Hierachical Methods

- Goal
  - Construction of a hierarchy of clusters (dendrogram); Clusters with minimal distance are merged
- Dendrogram
  - Tree structure: Nodes denote clusters
    - Root: All objects
    - Leaves: Single objects
    - Inner node: Union of all objects represented in the subtree (starting at that node)

Hierachical Methods

- Example of a dendrogram



- Types of hierarchical methods
  - Bottom-up construction of the dendrogram (*agglomerative*)
  - Top-down construction of the dendrogram (*divisive*)

Single-Link & Variants

Algorithm Single-Link [Jain & Dubes 1988]

- Agglomerative hierarchical clustering:
  1. Generate initial clusters consisting of single objects und calculate the distances between all such pairs
  2. Generate a new cluster by combining the two clusters with minimal distance
  3. Determine the distance between the new cluster and all other clusters
  4. If there is only one cluster left – terminate.
     Otherwise, go to step 2.

Single-Link & Variants

- *Distance functions for clusters*
- Distance function *dist*(*x*,*y*) for pairs of objects
- Let *X*, *Y* denote clusters

- *Single-Link* $dist-sl(X,Y)=\min_{x\in X,y\in Y} dist(x,y)$
- *Complete-Link* $dist-cl(X,Y)=\max_{x\in X,y\in Y} dist(x,y)$
- *Average-Link* $dist-al(X,Y)=\frac{1}{|X|\cdot|Y|}\cdot\sum_{x\in X,y\in Y} dist(x,y)$

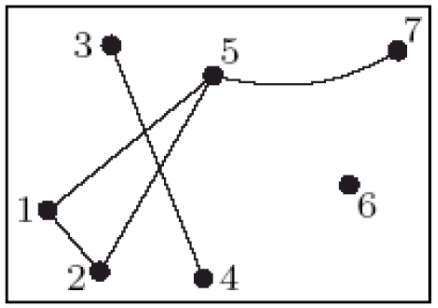
Single-Link & Variants

- *Discussion*
- *"+"*
  - Does not require knowledge about the number of clsuters
  - Result is not only a flat clustering, but a hierarchy of clusters
  - A single clustering can be obtained using the dendrogram (however, this requires domain knowledge) by a "vertical cut“ through the dendrogram
- "-“
  - Decisions during construction cannot be revised
  - Susceptible to noise (single link) – "line“ of objects can connect two clusters
  - Inefficiency: Runtime at least quadratic in the number of objects

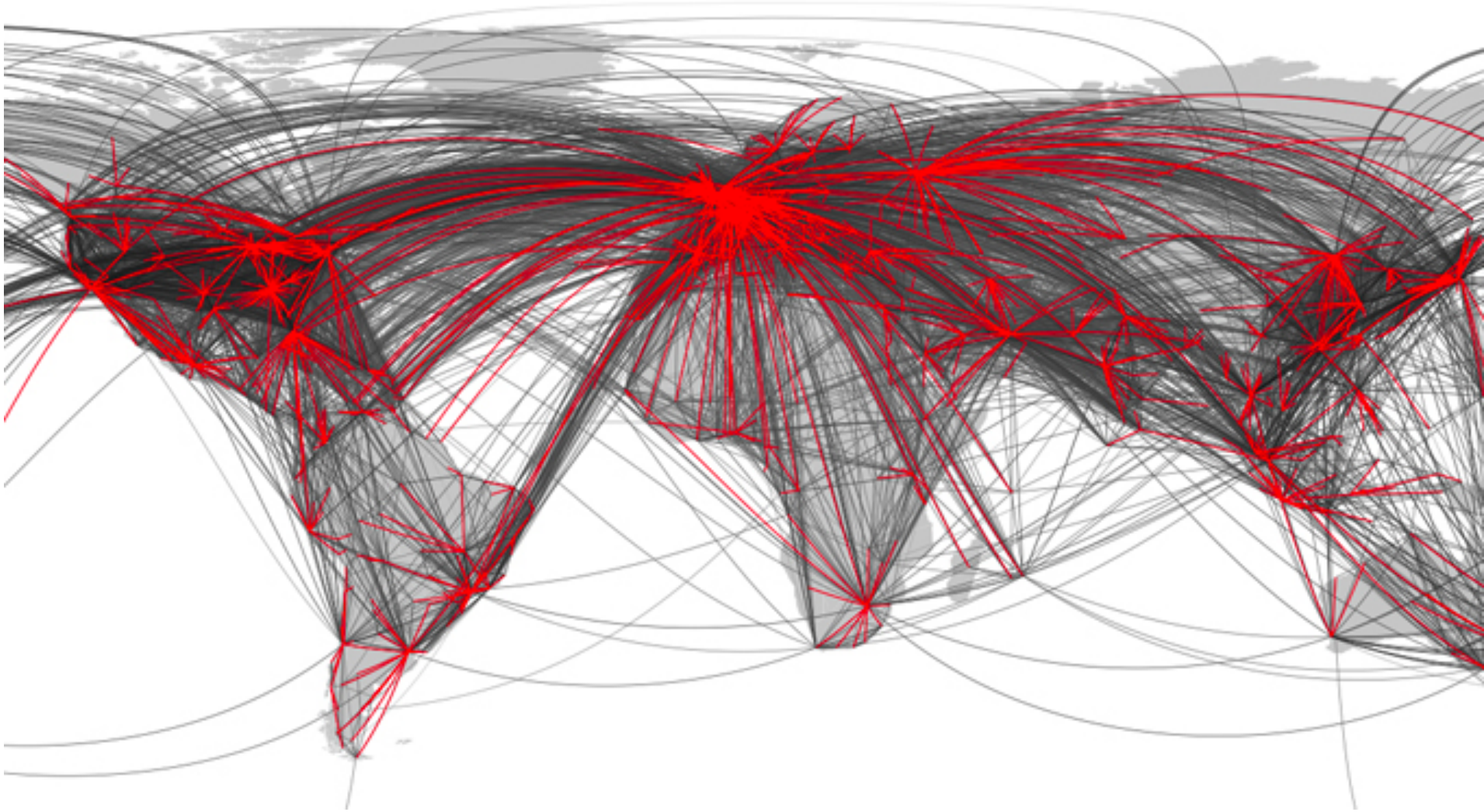
Graphs/Networks

Network:

- A set of atomic entities ➔nodes or vertices („points“)
- A set of links/edges between these nodes
- Edges model pairwise relations
- Edge: directed or undirected
- Network: Nodes + Edges
- For us, a network is an abstract object (list of pairs), independent of its representation
  - Often represented as a **graph**
  - Powerful modeling options for complex (multi-relational) data
  - Analysis: Structural properties
    Here: Clusters/groups/communities



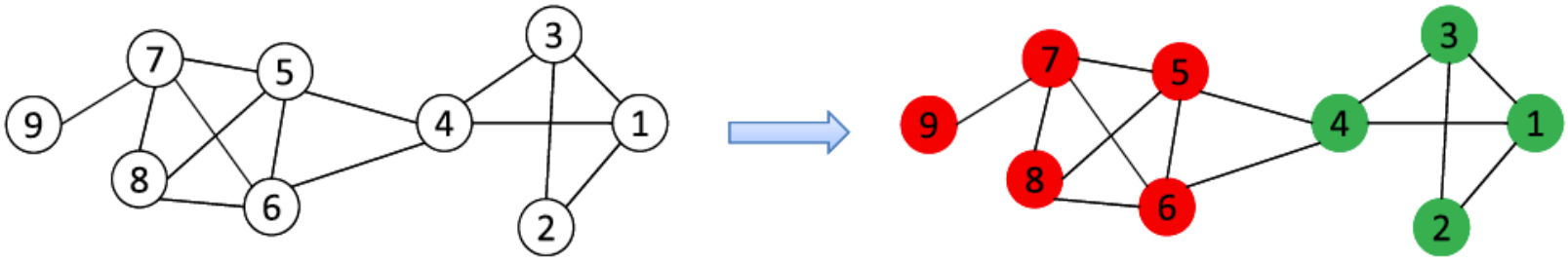
Complex Networks & Graphs



Social Networks

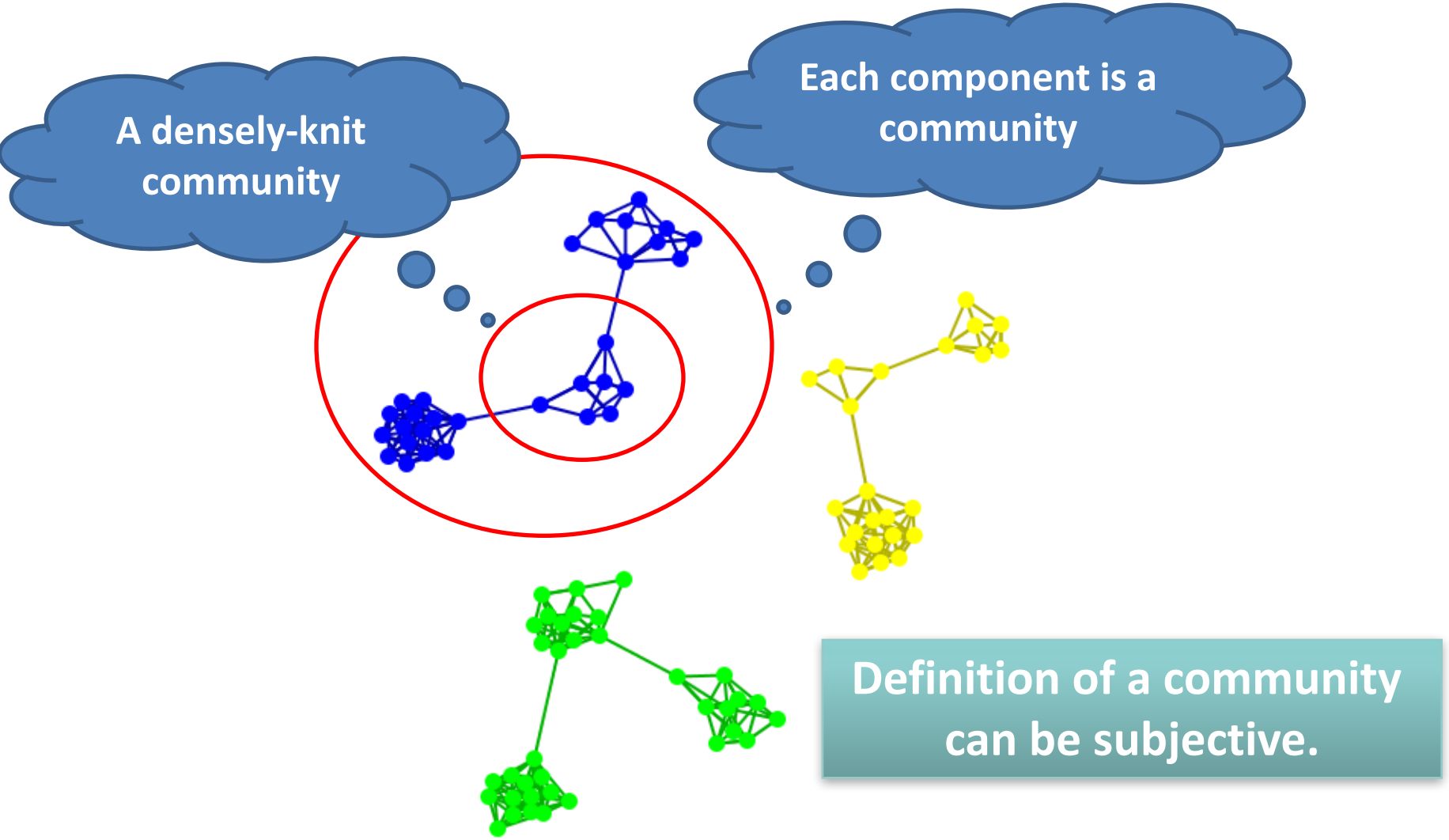
Communities

- A community is a set of nodes between which the interactions are (relatively) frequent
  - A.k.a., *group, cluster, cohesive subgroups, modules*



- Applications:
  - *Recommendation based communities,*
  - *Network Compression*
  - *Visualization of a huge network*

Subjectivity of Community Definition

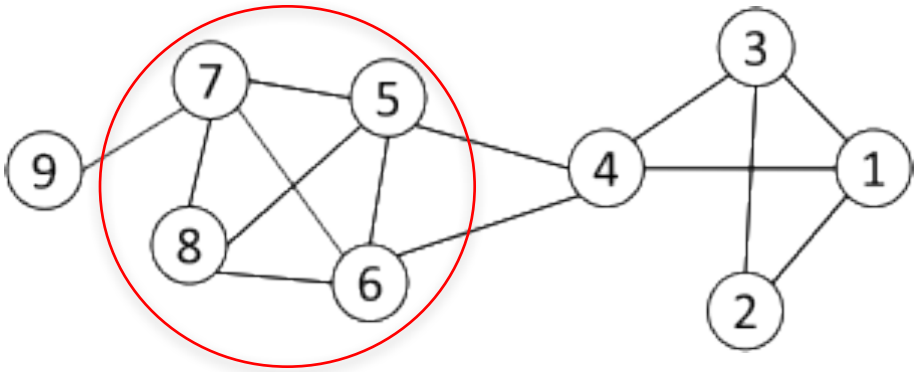


Taxonomy of Community Criteria

- Criteria vary depending on the tasks
- Roughly, community detection methods can be divided into 4 categories (not exclusive):
- Node-Centric Community
  - Each node in a group satisfies certain properties
- Group-Centric Community
  - Consider the connections within a group as a whole. The group has to satisfy certain properties without zooming into node-level
- Network-Centric Community
  - Partition the whole network into several disjoint sets
- Hierarchy-Centric Community
  - Construct a hierarchical structure of communities

Complete Mutuality: Cliques

- Clique: a maximum complete subgraph in which all nodes are adjacent to each other



Nodes 5, 6, 7 and 8 form a clique

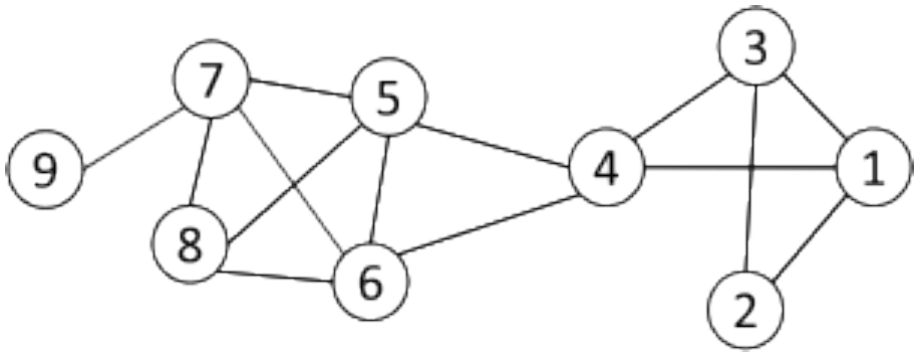
- Difficult to find the maximum clique in a network
- Straightforward implementation to find cliques is very expensive in time complexity

Clique Percolation Method (CPM)

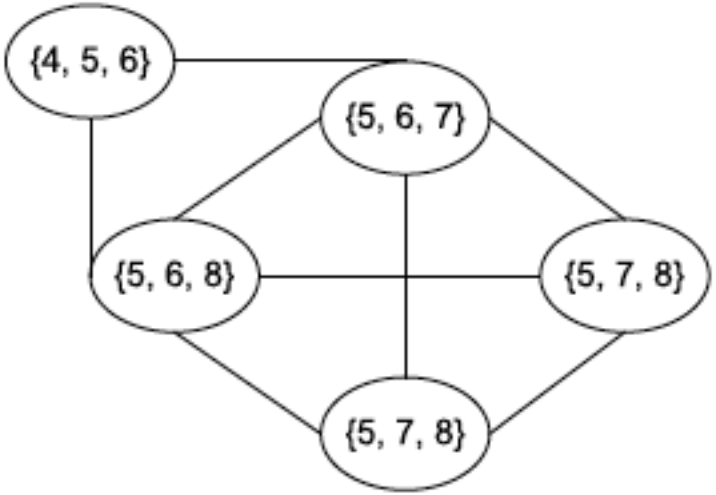
- Clique is a very strict definition, unstable
- Normally use cliques as a core or a seed to find larger communities

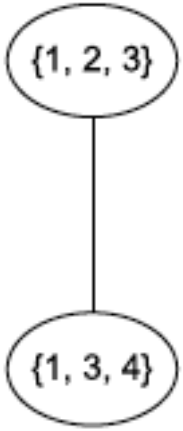
- CPM is such a method to find overlapping communities
  - **Input**
    - A parameter k, and a network
  - **Procedure**
    - Find out all cliques of size k in a given network
    - Construct a clique graph. Two cliques are adjacent if they share k-1 nodes
    - Each connected component in the clique graph forms a community

CPM Example



Cliques of size 3:
{1, 2, 3}, {1, 3, 4}, {4, 5, 6}, {5, 6, 7}, {5, 6, 8}, {5, 7, 8}, {6, 7, 8}

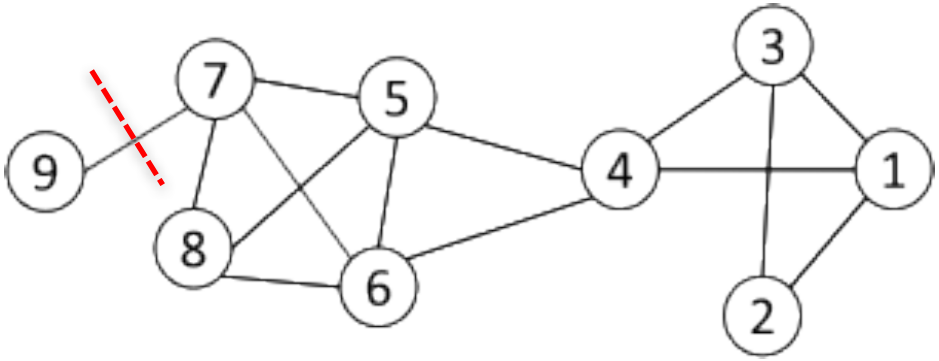




Communities:
{1, 2, 3, 4}
{4, 5, 6, 7, 8}

Cut

- Most interactions are within group whereas interactions between groups are few
- Community detection → Minimum cut problem
- Cut: A partition of vertices of a graph into two disjoint sets
- Minimum cut problem: Find a graph partition such that the number of edges between the two sets is minimized



Hierarchy-Centric Community Detection

- Goal: build a hierarchical structure of communities based on network topology

- Allow the analysis of a network at different resolutions

- Representative approaches:
  - Divisive Hierarchical Clustering
  - Agglomerative Hierarchical clustering

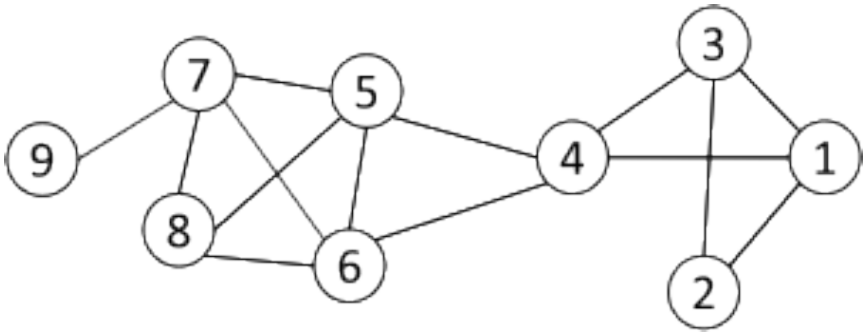
Divisive Hierarchical Clustering

- Divisive clustering
  - Partition nodes into several sets
  - Each set is further divided into smaller ones
  - Network-centric partition can be applied for the partition
- One particular example: recursively remove the "weakest" tie
  - Find the edge with the least strength
  - Remove the edge and update the corresponding strength of each edge
- Recursively apply the above two steps until a network is discomposed into desired number of connected components.
- Each component forms a community

Partitioning: Edge Betweenness

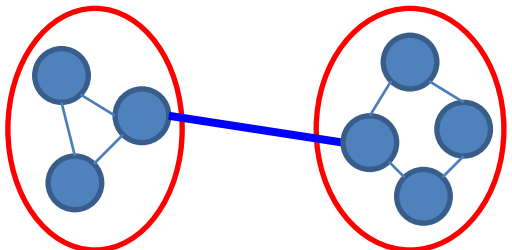
- The strength of a tie can be measured by edge betweenness
- Edge betweenness: the number of shortest paths that pass along with the edge

$$\text{edge-betweenness(e)} = \Sigma_{s<t} \frac{\sigma_{st}(e)}{\sigma_{s,t}}$$

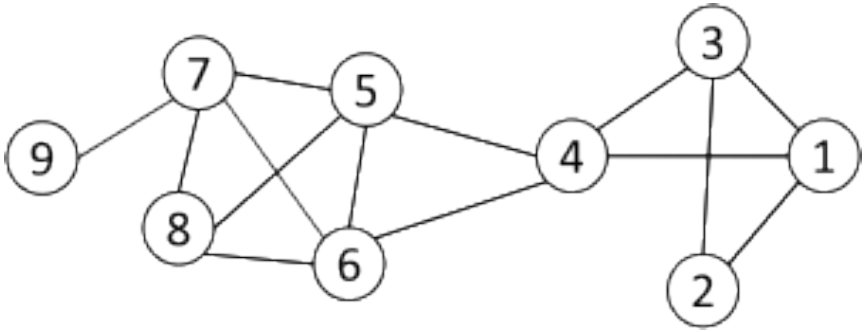


The edge betweenness of e(1, 2) is 4, as all the shortest paths from 2 to {4, 5, 6, 7, 8, 9} have to either pass e(1, 2) or e(2, 3), and e(1,2) is the shortest path between 1 and 2

- The edge with higher betweenness tends to be the bridge between two communities.



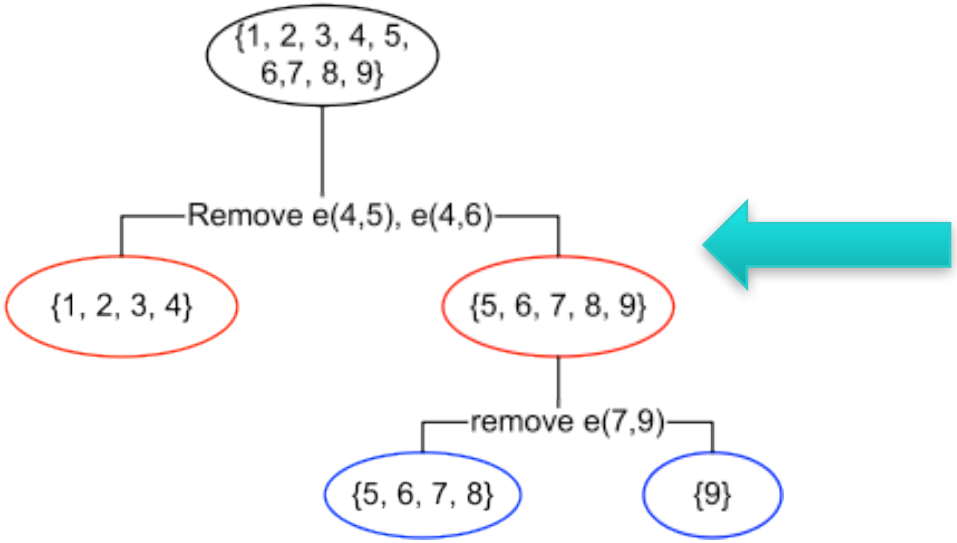
Divisive Clustering based on Edge Betweenness



Initial betweenness values

Table 3.3: Edge Betweenness

	1	2	3	4	5	6	7	8	9
1	0	4	1	9	0	0	0	0	0
2	4	0	4	0	0	0	0	0	0
3	1	4	0	9	0	0	0	0	0
4	9	0	9	0	10	10	0	0	0
5	0	0	0	10	0	1	6	3	0
6	0	0	0	10	1	0	6	3	0
7	0	0	0	0	6	6	0	2	8
8	0	0	0	0	3	3	2	0	0
9	0	0	0	0	0	0	8	0	0



After removing e(4,5), the betweenness of e(4, 6) becomes 20, which is the highest;

After remove e(4,6), the edge e(7,9) has the highest betweenness value 4, and should be removed.

Clustering and Graphs

Data Mining for Business and Governance
10/10/2017
Guest Lecture - Martin Atzmueller