Lecture 6 Week 10 (May 12) and Week 11 (May 19)

33459-01 Principles of Knowledge Discovery in Data

Clustering Analysis: Agglomerative, Hierarchical, Density-based and other approaches

Lecture by: Dr. Osmar R. Zaïane

Course Content

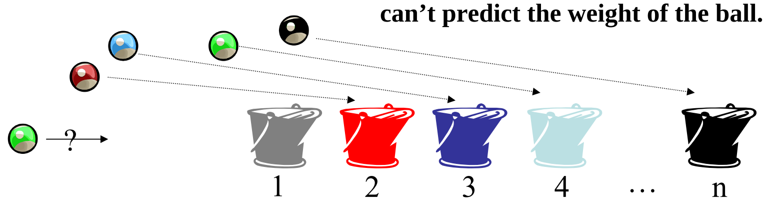
- Introduction to Data Mining
- Association analysis
- Sequential Pattern Analysis
- Classification and prediction
- ~~Contrast Sets~~
- Data Clustering
- Outlier Detection
- Web Mining

What is Classification?

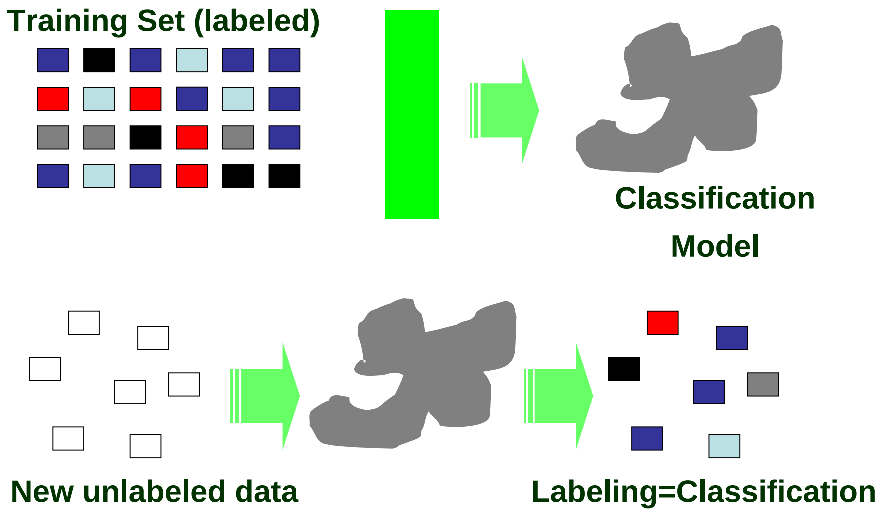
The goal of data classification is to organize and categorize data in distinct classes.

- A model is first created based on the data distribution.
- The model is then used to classify new data.
- Given the model, a class can be predicted for new data.

With classification, I can predict in which bucket to put the ball, but I can't predict the weight of the ball.

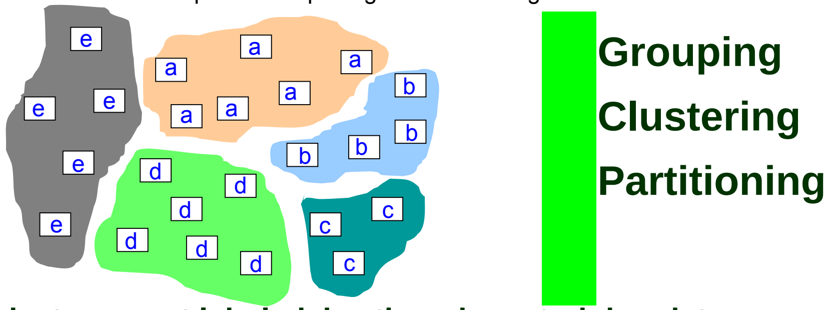


Classification = Learning a Model



What is Clustering?

The process of putting *similar* data together.



–Objects are not labeled, i.e. there is no training data.

–We need a notion of similarity or closeness (what features?)

– Should we know apriori how many clusters exist?

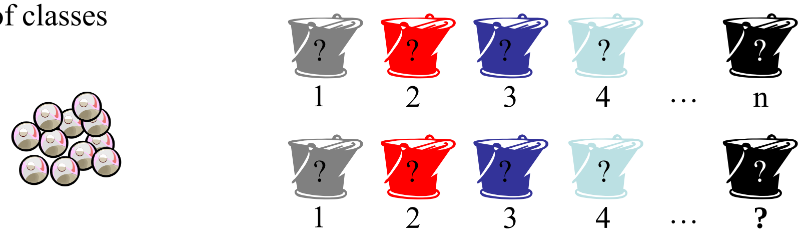
– How do we characterize members of groups?

– How do we label groups?

Supervised and Unsupervised

Supervised Classification = Classification
➔ We know the class labels and the number of classes

Unsupervised Classification = Clustering
➔ We do not know the class labels and may not know the number of classes



Lecture Outline

Part I: What is Clustering in Data Mining *(30 minutes)*

- **Introduction to clustering**
  - **Motivating Examples for clustering**
  - **Taxonomy of Major Clustering Algorithms**
  - **Major Issues in Clustering**
  - **What is Good Clustering?**

Part II: Major Clustering Approaches *(1 hour 20 minutes)*

- **K-means (Partitioning-based clustering algorithm)**
- **Nearest Neighbor clustering algorithm**
- **Hierarchical Clustering**
- **Density-based Clustering**

Part III: Open Problems *(10 minutes)*

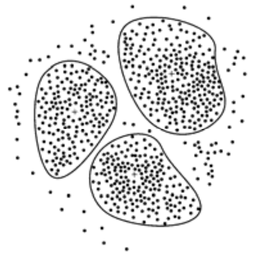
- **Research Issues in Clustering**

What is Clustering in Data Mining?

Clustering is a process of partitioning a set of data (or objects) in a set of meaningful sub-classes, called **clusters**.

– Helps users understand the natural grouping or structure in a data set.

- Cluster: a collection of data objects that are "similar" to one another and thus can be treated collectively as one group.
- A good clustering method produces high quality clusters in which:
  - The **intra-class** (that is, intra-cluster) similarity is high.
  - The **inter-class** similarity is low.
- The **quality** of a clustering result depends on both the similarity measure used and its implementation.
- Clustering = function that maximizes similarity between objects within a cluster and minimizes similarity between objects in different clusters.



Typical Clustering Applications

- As a *stand-alone tool* to
  - get insight into data distribution
  - find the characteristics of each cluster
  - assign the cluster of a new example
- As a *preprocessing step* for other algorithms
  - e.g. numerosity reduction – using cluster centers to represent data in clusters
- It is a building block for many data mining solutions
  - e.g. Recommender systems – group users with similar behaviour or preferences to improve recommendation.

Clustering Example – Fitting Troops

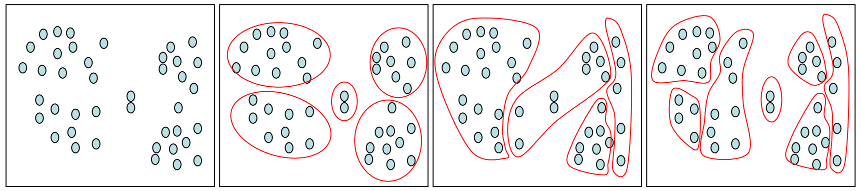
- Fitting the troops – re-design of uniforms for female soldiers in US army
  - Goal: reduce the number of uniform sizes to be kept in inventory while still providing good fit
- Researchers from Cornell University used clustering and designed a new set of sizes
  - Traditional clothing size system: ordered set of graduated sizes where all dimensions increase together
  - The new system: sizes that fit body types
    - E.g. one size for short-legged, small waisted, women with wide and long torsos, average arms, broad shoulders, and skinny necks

Other Examples of Clustering Applications

- Marketing
  - help discover distinct groups of customers, and then use this knowledge to develop targeted marketing programs
- Biology
  - derive plant and animal taxonomies
  - find genes with similar function
- Land use
  - identify areas of similar land use in an earth observation database
- Insurance
  - identify groups of motor insurance policy holders with a high average claim cost
- City-planning
  - identify groups of houses according to their house type, value, and geographical location

Clustering Example - Houses

- Given a dataset it may be clustered on different attributes. The result and its interpretation would be different


Group of houses
Clustered based on geographic distance
Clustered based on value
Clustered based on size and value

Definition of a distance function is highly application dependent

Measures "dissimilarity" between pairs objects *x* and *y*
- Small distance *dist*(*x*, *y*): objects x and *y* are more similar
- Large distance *dist*(*x*, *y*): objects x and y are less similar

Properties of a distance function

$dist(x, y) \geq 0$

$dist(x, y) = 0$ iff $x = y$

$dist(x, y) = dist(y, x)$ (symmetry)

If *dist* is a metric, which is often the case:

$dist(x, z) \leq dist(x, y) + dist(y, z)$ (triangle inequality)

Major Clustering Techniques

- Clustering techniques have been studied extensively in:
  - Statistics, machine learning, and data mining

  with many methods proposed and studied.
- Clustering methods can be classified into 5 approaches:
  - **partitioning algorithms**
  - **hierarchical algorithms**
  - **density-based methods**
  - **grid-based methods**
  - **model-based methods**

← We will cover only these.

Five Categories of Clustering Methods

- **Partitioning algorithms**: Construct various partitions and then evaluate them by some criterion. *(K-Means is the most known)*
- **Hierarchy algorithms**: Create a hierarchical decomposition of the set of data (or objects) using some criterion. There is an agglomerative approach and a divisive approach.
- **Density-based**: based on connectivity and density functions.
- **Grid-based**: based on a multiple-level granularity structure.
- **Model-based**: (Generative approach) A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other. Generative models estimated through maximum likelihood approach. *(EM: Expectation Maximization with a Gaussian Mixture Model, is a typical example)*

Important Issues in Clustering

- Different Types of Attributes
  - Numerical: Generally can be represented in a Euclidean Space. Distance can be used as a measure of dissimilarity.(See classification slides for measures)
  - Categorical: A metric space may not be definable. A similarity measure has to be defined. Jaccard ($\frac{|X \cap Y|}{|X \cup Y|}$); Dice ($\frac{2|X \cap Y|}{|X|+|Y|}$); Overlap ($\frac{|X \cap Y|}{\min(|X|,|Y|)}$); Cosine ($\frac{|X \cap Y|}{\sqrt{|X| \times |Y|}}$) etc.
  - Sequence aware similarity: eg. DNA sequences, web access behaviour. Can use Dynamic Programming.

Important Issues in Clustering (2)

- Noise and outlier Detection
  - Differentiate remote points from internal ones.
  - Noisy points (errors in data) can artificially split or merge clusters.
  - Distinguishing remote noisy points or very small unexpected clusters can be very important for the validity and quality of the results.
  - Noise can bias the results especially in the calculation of cluster characteristics.

Important Issues in Clustering (3)

- High dimensional spaces
  - The more dimensions describe the data the sparser the space is. The sparser the space the smaller the likelihood to discover significant clusters.
  - Large number of dimensions makes it hard to process.
  - What is the meaning of distance or similarity in a high dimensional space?
  - Indexes are not efficient
  - The curse of dimensionality

Important Issues in Clustering (4)

- Parameters
  - Clustering algorithms are typically VERY sensitive to parameters. A small delta makes a huge difference in the final results.
  - Tuning is very difficult.
  - Parameters are not necessarily related to the application domain
  - E.g. **The best number of clusters to discover (k) is not known**
    - **There is no one correct answer to a clustering problem**
    - **domain expertise is required**

Requirements of Clustering in Data Mining

- Scalability (very large number of objects to cluster)
- Dealing with different types of attributes (Numeric & categorical)
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters (minimize parameters)
- Able to deal with noise and outliers
- Insensitive to order of input records
- Handles high dimensionality
- Can be incremental for dynamic change

Not all Clustering Algorithms can deal with all these requirements. They target some of them and ignore the others by making assumptions and imposing approximations.

Lecture Outline

Part I: What is Clustering in Data Mining *(30 minutes)*

- **Introduction to clustering**
  - **Motivating Examples for clustering**
  - **Taxonomy of Major Clustering Algorithms**
  - **Major Issues in Clustering**
  - **What is Good Clustering?**

Part II: Major Clustering Approaches *(1 hour 20 minutes)*

- **K-means (Partitioning-based clustering algorithm)**
- **Nearest Neighbor clustering algorithm**
- **Hierarchical Clustering**
- **Density-based Clustering**

Part III: Open Problems *(10 minutes)*

- **Research Issues in Clustering**

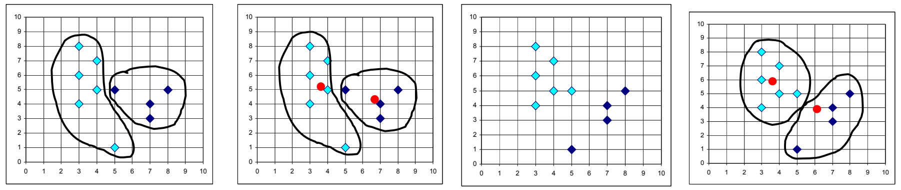
Partitioning Algorithms: Basic Concept

- **Partitioning method:** Construct a partition of a database ***D*** of *N* objects into a set of ***k*** clusters.
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion.
  - Global optimal: exhaustively enumerate all partitions.
  - Heuristic methods: *k-means* and *k-medoids* algorithms.
  - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster.
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw '87): Each cluster is represented by one of the objects in the cluster.

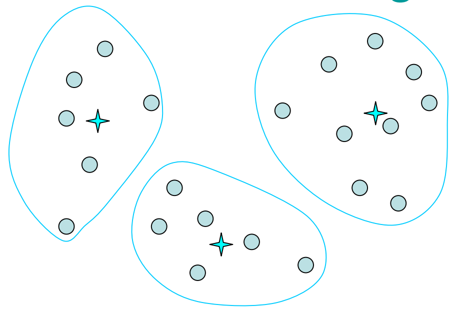
Requires the number of clusters *k* to be pre-specified

K-Means Clustering Method

- Given *k*, the *k-means* algorithm is implemented in 4 steps:
  1. Partition objects into *k* nonempty subsets
  2. Compute seed points as the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
  3. Assign each object to the cluster with the nearest seed point.
  4. Go back to Step 2, stop when no more new assignment.

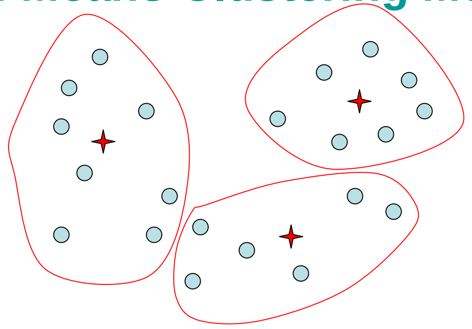


The *K-Means* Clustering Method



1.Select *k* nodes at random (could be a real point or a virtual point)
2.Assign all nodes to *k* clusters based on nearness.

The *K-Means* Clustering Method

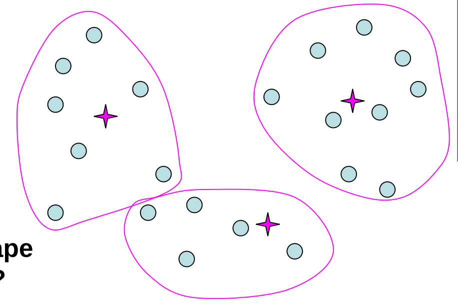


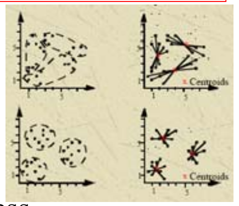
1.Select *k* nodes at random,
2.Assign all nodes to *k* clusters based on nearness.
3. Calculate new means and reassign all nodes.

The *K-Means* Clustering Method

(e.g. minimize sum squared error: dist(*mean, t)* for all *k* clusters)

$$\frac{\sum_{j=1}^{k}\sum_{i=1}^{size(K_j)}\left\|t_{ji}-C_j\right\|^2}{k}$$





- **What is the general shape of clusters?**

1.Select *k* nodes at random,
2.Assign all nodes to *k* clusters based on nearness.
3. Calculate new means and reassign all nodes.
Iterate until the criterion function converges

- **What is the output of k-means?**
- **What are the consequences of outliers/noise?**

Comments on the *K-Means* Method

- Strength of the *k-means*:
  – *Relatively efficient*: *O*(*tkN*), where *N* is # of objects, *k* is # of clusters, and *t* is # of iterations. Normally, *k*, *t* << *N*.
  – Often terminates at a *local optimum*.
- Weakness of the *k-means*:
  – Applicable only when *mean* is defined, then what about categorical data?
  – Need to specify *k, the number* of clusters, in advance.
  – Unable to handle noisy data and *outliers.*
  – Not suitable to discover clusters with *non-convex shapes.*
  – *Not deterministic – depends on initial seeds*
- *Relevant issues:*
  – *Different distance measures can be used*
  – *Data should be normalized first (without normalization, the variable with the largest scale will dominate the measure.)*

Variations of the *K-Means* Method

- A few variants of the *k-means* which differ in:
  – Selection of the initial *k* means.
  – Dissimilarity calculations.
  – Strategies to calculate cluster means.
- Use a representative point rather than a abstract cluster centre: *k-mediods*
- Handling categorical data: *k-modes* (Huang,1998)
- A mixture of categorical and numerical data: *k-prototype* method (Huang, 1997)

The *K-Medoids* Clustering Method

- Find *representative* objects, called medoids, in clusters
  – To achieve this goal, only the definition of distance from any two objects is needed.
- *PAM* (Partitioning Around Medoids, 1987)
  – starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering.
  – *PAM* works effectively for small data sets, but does not scale well for large data sets.
- *CLARA* (Kaufmann & Rousseeuw, 1990) [Multiple samples + PAM]
- *CLARANS* (Ng & Han, 1994): Randomized sampling.
- Focusing + spatial data structure (Ester et al., 1995).

K-Means Algorithm – Pseudo Code

```
Input:
        D={t_1, t_2, …, t_n}    // Set of elements
        k                       // desired number of clusters
Output:
        K   // Set of k clusters
K-means algorithm
        assign initial values for means m_1, m_2, …, m_k      // k seeds
        repeat
                assign each item t_i to the cluster which has the closest mean;
                calculate new mean for each cluster;
        until convergence criteria is met;
```

- ✓ Scalability
- ➡ Dealing with different types of attributes
- ~~Discovery of clusters with arbitrary shape~~
- ~~Minimal requirements for domain knowledge to determine input parameters~~
- ~~Able to deal with noise and outliers~~
- ✓ Insensitive to order of input records
- ✓ Handles high dimensionality
- ✓ Can be incremental for dynamic change

Characteristics of a Cluster

- Consider a cluster K of N points $\{p_1,..,p_N\}$
- *Centroid* – the "middle" of the cluster
  - no need to be an actual data point in the cluster

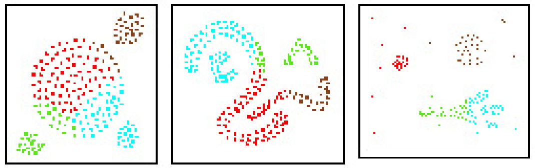
$$C = \frac{\sum_{i=1}^{n} p_i}{N}$$

- *Medoid M* – the centrally located data point (object) in the cluster
- *Radius* – square root of the average mean squared distance from any point in the cluster to the centroid

$$R = \sqrt{\frac{\sum_{i=1}^{N} (p_i - C)^2}{N}}$$

- *Diameter* – square root of the average mean squared distance between all pairs of points in the cluster

$$R = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} (t_i - t_j)^2}{N(N-1)}}$$


Some example results

Nearest Neighbor Clustering Algorithm

- A new instance forms a new cluster or is merged to an existing one depending on how close it is to the existing clusters
  - threshold θ to determine if to merge or create a new cluster

```
Input:
        D={t_1, t_2, …, t_n}    // Set of elements
        A           // Adjacency matrix showing distance between elements
        θ           // threshold
Output:
        K           //Set of k clusters
Nearest-Neighbor algorithm
        K_1 ={t_1};  add K_1 to K;        // t1 initialized the first cluster
        k = 1;
        for i =2 to n do    // for t_2 to t_n add to existing cluster or place in new one
                find the t_m in some cluster K_m in K such that d(t_m,t_i) is the smallest;
                if d(t_m,t_i) < θ then
                        K_m = K_m U {t_i}                    // existing cluster
                else
                        k = k + 1; K_k = {t_i}; add K_k to K     // new cluster
```

- Time complexity: *O(n²), n*-number of items
  - Each item is compared to each item already in the cluster

Nearest Neighbor Clustering - Example

- Given: 5 items with the distance between them
- Task: Cluster them using the Nearest Neighbor algorithm with a threshold θ =2

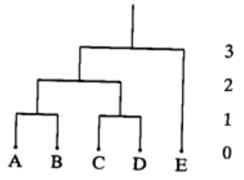
Item	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

- No need to know the number of clusters to discover before hand.
- We need to define the threshold θ.

-A: K1={A}
-B: d(B,A)=1< θ => K1={A,B}
-C: d(C,A)=d(C,B)=2≤ θ =>K1={A,B,C}
-D: d(D,A)=2, d(D,B)=4, d(D,C)=1 =dmin ≤ θ => K1={A,B,C,D}
-E: d(E,A)=3, d(E,B)=3, d(E, C)=5, d(E, D)=3=dmin> θ =>K2={E}

Hierarchical Clustering

- Creates not one set of clusters but <u>several sets</u> of clusters
  - The desired number of clusters *k* is not necessarily an input
- The hierarchy of clusters can be represented as a tree structure called *dendrogram*
- Leaves of the dendrogram consist of 1 item
  - each item is in one cluster
- Root of the dendrogram contains all items
  - all items form one cluster
- Internal nodes represent clusters formed by merging the clusters of the children
- Each level is associated with a distance threshold that was used to merge the clusters
  - If the distance between 2 clusters was smaller than the threshold they were merged
  - This distance increases with the levels.

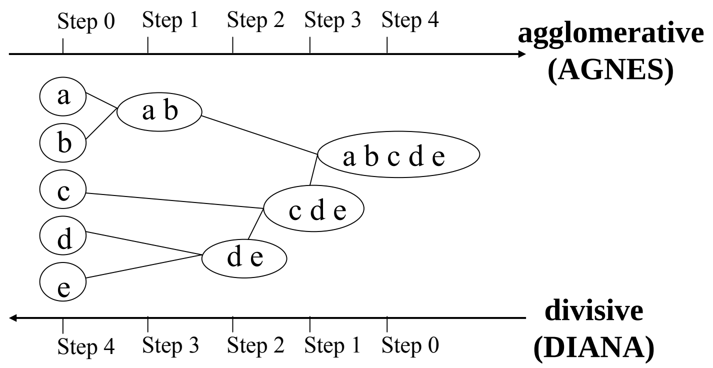


Two Types of Hierarchical Clustering Algorithms

- Agglomerative (bottom-up): merge clusters iteratively.
  - start by placing each object in its own cluster.
  - merge these atomic clusters into larger and larger clusters.
  - until all objects are in a single cluster.
  - Most hierarchical methods belong to this category. They differ only in their definition of *between-cluster similarity*.
- Divisive (top-down): split a cluster iteratively.
  - It does the reverse by starting with all objects in one cluster and subdividing them into smaller pieces.
  - Divisive methods are not generally available, and rarely have been applied.

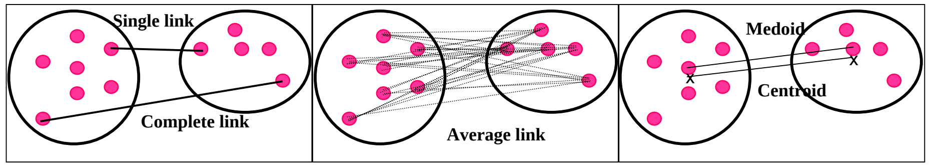
Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition.



Distance Between Clusters

Many interpretations:

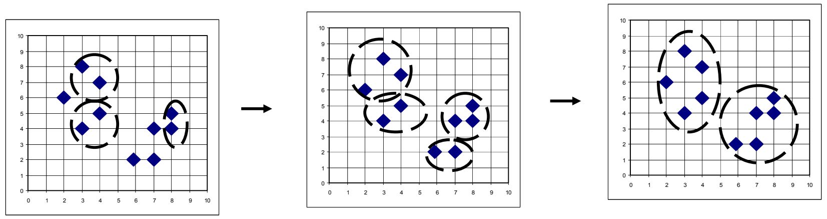


- **Single link – the distance between two clusters is the <u>shortest</u> distance from any member of one cluster to any member of the other cluster.**
- **Complete link – the distance between two clusters is the <u>greatest</u> distance from any member of one cluster to any member of the other cluster.**
- **Average link – the <u>average</u> distance between each element in one cluster and each element in the other.**
- **Centroid – The distance between two clusters is the distance between the the two centroids of the clusters**
- **Medoid – The distance between two clusters is the distance between the medoids of the clusters.**

- **What about categorical data?**

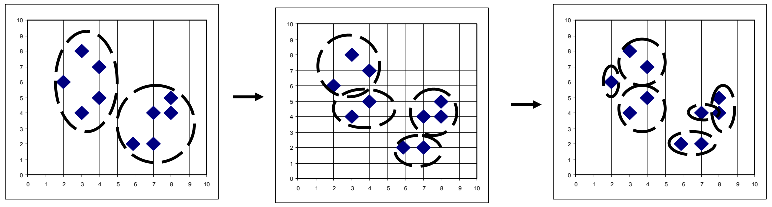
AGNES (Agglomerative Nesting)

- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages, such as S+.
- Use the Single-Link method and the dissimilarity matrix.
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



DIANA (Divisive Analysis)

- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages, such as S+.
- Inverse order of AGNES.
- Eventually each node forms a cluster on its own.

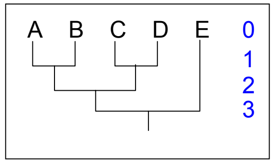


Dendrogram Representation

- **A set of ordered triples *(d,k,K)***
  - ***d* is the distance threshold value**
  - ***k* is the number of clusters**
  - ***K* is the set of clusters**

- **Example**

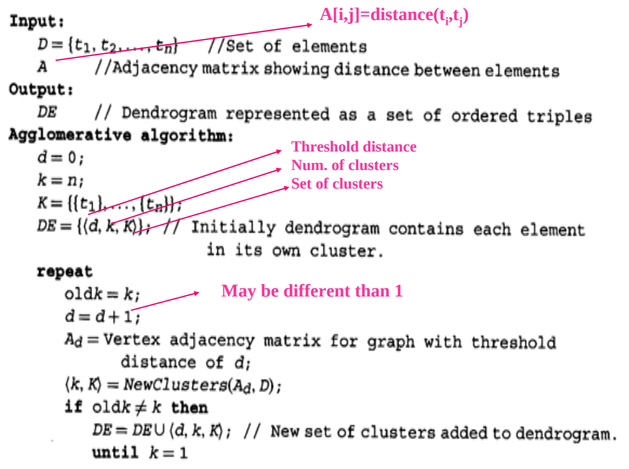
{ (0, 5, {{A},{B},{C}, {D}, {E}}),
(1, 3, {{A,B}, {C,D}, {E}}),
(2, 2, {{A,B,C,D}, {E}}),
(3, 1, {A,B,C,D,E}}) }



- **Thus, the output is not one set of clusters but several. One can determine which of the sets to use.**

Agglomerative Algorithms – Pseudo Code

- **Different algorithms merge clusters at each level differently (procedure `NewClusters`)**



```
Input:
    D = {t_1, t_2, ..., t_n}   //Set of elements
    A      //Adjacency matrix showing distance between elements
Output:
    DE     // Dendrogram represented as a set of ordered triples
Agglomerative algorithm:
    d = 0;
    k = n;
    K = {{t_1}, ..., {t_n}};
    DE = {<d, k, K>};  // Initially dendrogram contains each element
                          in its own cluster.
    repeat
        oldk = k;
        d = d + 1;
        A_d = Vertex adjacency matrix for graph with threshold
                distance of d;
        <k, K> = NewClusters(A_d, D);
        if oldk ≠ k then
            DE = DE ∪ <d, k, K>;  // New set of clusters added to dendrogram.
            until k = 1
```

- **Merge only 2 or more clusters?**
- **If there are several clusters with identical distances, which ones to merge?**
- **How to determine the distance between clusters?**
  - **single link**
  - **complete link**
  - **average link**

NewClusters Procedure

- `NewClusters` typically finds all the clusters that are within distance *d* from each other (according to the distance measure used), merges them and updates the adjacency matrix
- Example:
  - Given: 5 items with the distance between them
  - Task: Cluster them using agglomerative single link clustering

	A	B	C	D	E
A	0	1	2	2	3
B		0	2	4	3
C			0	1	5
D				0	3
E					0

Example – Solution 1

Level 0

	A	B	C	D	E
A	0	1	2	2	3
B		0	2	4	3
C			0	1	5
D				0	3
E					0

- Distance Level 1: merge A&B, C&D; update the adjacency matrix:

Level 1

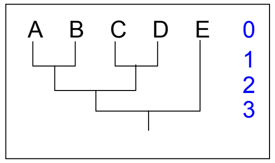
	AB	CD	E
AB	0	2	3
CD		0	3
E			0

- Distance Level 2: merge AB&CD; update the adjacency matrix:

Level 2

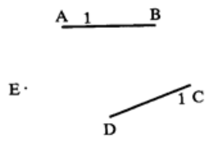
	ABCD	E
ABCD	0	3
E		0

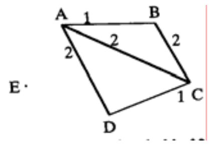
- Distance Level 3: merge ABCD&E; all items are in one cluster; stop
- Dendrogram:



Single Link Algorithm as a Graph Problem

- `NewClusters` can be replaced with a procedure for finding connected components in a graph
  - two components of a graph are connected if there exists a path between any 2 vertices
  - Examples:





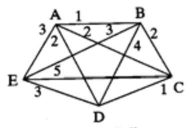
- A and B are connected, C and D are connected

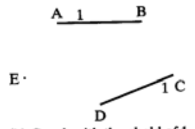
A, B, C and D are connected

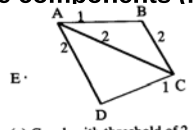
- Show the graph edges with a distance of *d* or below
- Merge 2 clusters if there is at least 1 edge that connects them (i.e. if the minimum distance between any 2 points is <= *d*)
- Increment d

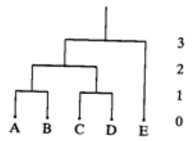
Example – Solution 2

- Procedure `NewClusters`
  - Input: graph defined by a set of vertices + vertex adjacency matrix
  - Output: a set of connected components defined by a number of these components (i.e. number of clusters *k*) and an array with the membership of these components (i.e. *K* - the set of

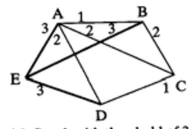

(a) Graph with all distances

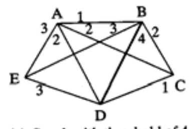

(b) Graph with threshold of 1


(c) Graph with threshold of 2



Single link dendrogram

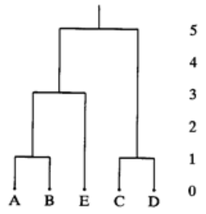

(d) Graph with threshold of 3


(e) Graph with threshold of 4

Single Link vs. Complete Link Algorithm

- Single link suffers from the so called *chain effect*
  - 2 clusters are merged if only 2 of their points are close to each other
  - there may be points in the 2 clusters that are far from each other but this has no effect on the algorithm
  - Thus the clusters may contain points that are not related to each other but simply happen to be near points that are close to each other

- Complete link – the distance between 2 clusters is the <u>largest</u> distance between an element in one cluster and an element in another
  - Generates more compact clusters
  - Dendrogram for the example



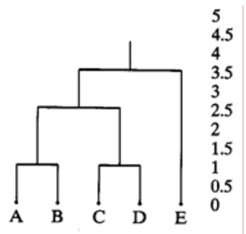
(b) Complete link

Average Link

- Average link - the distance between 2 clusters is the <u>average</u> distance between an element in one cluster and an element in another

```
Input:
   D = {t_1, t_2, ..., t_n}   //Set of elements
   A      //Adjacency matrix showing distance between elements
Output:
   DE     // Dendrogram represented as a set of ordered triples
Average link algorithm:
   d = 0;
   k = n;
   K = {{t_1}, ..., {t_n}};
   DE = ⟨d, k, K⟩; // Initially dendrogram contains each element
                      in its own cluster.
   repeat
      oldk = k;        arbitrary set; can be 1
      d = d + 0.5;
      for each pair of K_i, K_j ∈ K do
         ave = average distance between all t_i ∈ K_i and t_j ∈ K_j;
         if ave ≤ d, then
            K = K − {K_i} − {K_j} ∪ {K_i ∪ K_j};
            k = oldk − 1;
            DE = DE ∪ ⟨d, k, K⟩; // New set of clusters added
                                    to dendrogram.
   until k = 1
```

For our example



(c) Average link

Applicability and Complexity

- Hierarchical clustering algorithms are suitable for domains with natural nesting relationships between clusters
  - Biology- plant and animal taxonomies can be viewed as a hierarchy of clusters
- Space complexity of the algorithm: *$O(n^2)$, n* - number of items
  - The space required to store the adjacency distance matrix
- Space complexity of the dendrogram: *$O(kn)$, k*-number of levels
- Time complexity of the algorithm: *$O(kn^2)$* – 1 iteration for each level of the dendrogram
- Not incremental – assume all data is present

Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points.
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - <u>DBSCAN:</u> Ester, et al. (KDD ’96)
  - <u>DENCLUE</u>: Hinneburg & D. Keim (KDD ’98)
  - <u>CLIQUE</u>: Agrawal, et al. (SIGMOD ’98)
  - <u>OPTICS</u>: Ankerst, et al (SIGMOD ’99).
  - TURN*: Foss and Zaïane (ICDM ’02)

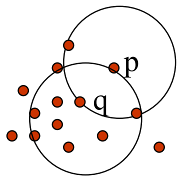
DBSCAN: A Density-Based Clustering

- DBSCAN: Density Based Spatial Clustering of Applications with Noise.
  - Proposed by Ester, Kriegel, Sander, and Xu (KDD'96).
  - Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points.
  - Clusters are dense regions in the data space, separated by regions of lower object density
  - Discovers clusters of arbitrary shape in spatial databases with noise.

Density-Based Clustering: Background

- Intuition for the formalization of the basic idea
  - For any point in a cluster, the local point density around that point has to exceed some threshold
  - The set of points from one cluster is spatially connected
- Two parameters***:***
  - ε : Maximum radius of the neighbourhood
  - ***MinPts***: Minimum number of points in an ε -neighbourhood of that point
- ***$N_\varepsilon(p)$***: ***{q belongs to D | dist(p,q) <= $\mathcal{E}$}***
- Directly density-reachable**:** A point ***p*** is directly density-reachable from a point ***q*** wrt. $\mathcal{E}$, ***MinPts*** if
  - 1) ***p*** belongs to ***N_ε(q)***
  - 2) core point condition:

    | N_ε(q)| >= MinPts

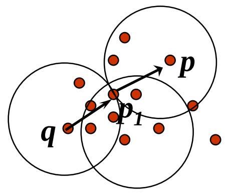


MinPts = 5

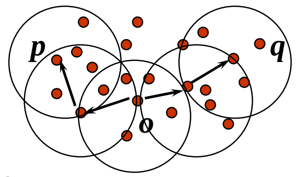
$\mathcal{E}$ = 1 cm

Density-Based Clustering: Background

- Density-reachable:
  - A point *p* is density-reachable from a point *q* wrt. ε, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i



- Density-connected
  - A point *p* is density-connected to a point *q* wrt. ε, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* wrt. ε and *MinPts*.



Clustering with DBSCAN

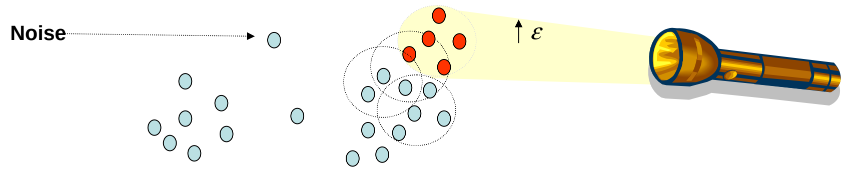
Density-Based Cluster: non-empty subset *S* of database *D* satisfying:

Maximality: if *p* is in *S* and *q* is density-reachable from *p* then *q* is in *S*

Connectivity: each object in *S* is density-connected to all other objects

Density-Based Clustering of a database *D* : $\{S_1, ..., S_n; N\}$ where

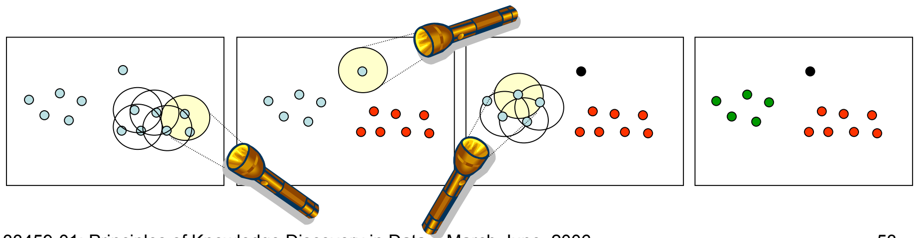
- $S_1, ..., S_n$: all density-based clusters in the database *D*
- $N = D \setminus \{S_1, ..., S_n\}$ is called the ***noise*** (objects not in any cluster)



DBSCAN General Algorithm

```
for each o ∈ D do
    if o is not yet labeled then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

density-reachable objects are collected by performing successive ε-neighborhood queries.



DBSCAN Advantages and Disadvantages

Advantages

- Clusters can have arbitrary shape and size
- Number of clusters is determined automatically
- Can separate clusters from surrounding noise
- Can be supported by spatial index structures

Disadvantages

- Input parameters may be difficult to determine
- Very sensitive to input parameter setting

Runtime complexities

without support (worst case):	$O(n)$	$O(n2)$
tree-based support (e.g. R*-tree) :	$O(\log(n))$	$O(n * \log(n))$
direct access to the neighborhood:	$O(1)$	$O(n)$

Wish list (DBScan)

- ✓ Scalability
- ➡ Dealing with different types of attributes
- ✓ Discovery of clusters with arbitrary shape
- ~~Minimal requirements for domain knowledge to determine input parameters~~
- ✓ Able to deal with noise and outliers
- ✓ Insensitive to order of input records
- ➡ Handles high dimensionality
- ~~Can be incremental for dynamic change~~

TURN*
(Foss and Zaiane, 2002)

- TURN* contains several sub-algorithms.
- TURN-RES computes a clustering of spatial data at a given resolution based on a definition of 'close' neighbours: $d_i - d_j <= 1.0$ for two points i, j and a local density t_i based on a point's distances to its nearest axial neighbours: $t_i = \Sigma_{d=0}^{D} f(d_d)$ for dimensions D.
- This density based approach is fast, identifies clusters of arbitrary shape and isolates noise.

TURN* Component Algorithms

Partitioning
K-nn
Hierarchical
Density-based

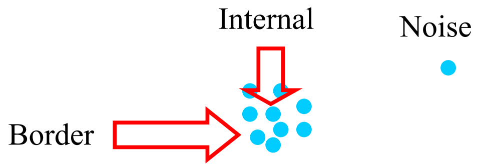
- TURN* wrapper algorithm finds the starting resolution and calls the other algorithms as needed as it scans over the range for which $k \geq 1$.
- TURN-RES generates both a clustering result and certain global statistics, especially a total density – the sum of the point local densities over all points clustered, excluding outliers.
- TurnCut finds the areas of interest in this graph using double differencing on the change values.

TURN*

Partitioning
K-nn
Hierarchical
Density-based

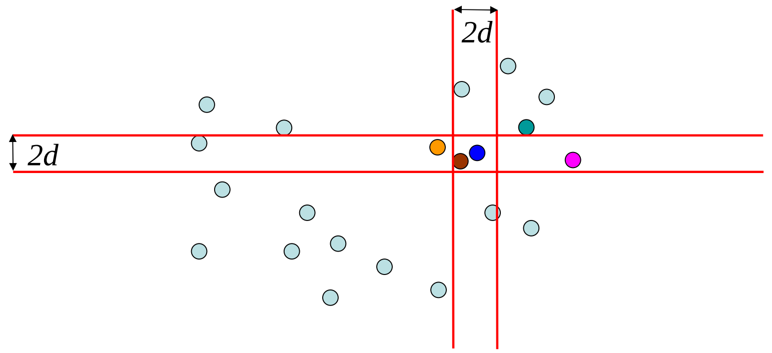
- A clustering result will be found over a certain range of resolutions. Outside of that there is either one cluster *(S_1)* or every point is classified as noise *(S_∞)*.
- TURN* first searches for S_∞ and then scans towards S_1 using TURN-RES to cluster until a clustering optimum is reported by TurnCut assessing the global cluster features collected at each resolution by TURN-RES.

A local density for each point is computed that identifies if a point is 'internal' to a cluster. A border point is not internal but has 'close' neighbours that are and thus gets included in the cluster.



Defining Neighbours

Partitioning
K-nn
Hierarchical
Density-based



A resolution is defined by a distance *d* along each dimensional axis. At this resolution the brown and pink points are nearest neighbours of the blue point along the vertical dimensional axis.

How TURN-RES Clusters

Partitioning
K-nn
Hierarchical
Density-based

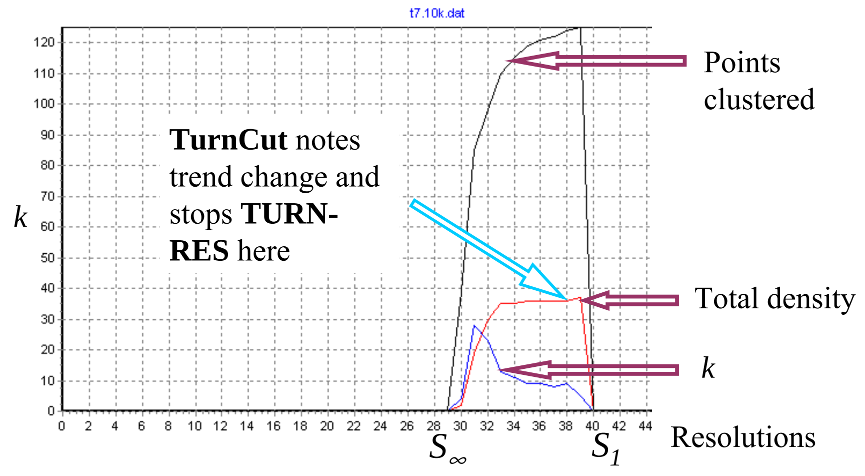
Type	Classified as	Close Neighbour?	Clustered
Interior	Internal	Yes	Yes
Border	External	Yes	Yes
Distant	External	No	No - Noise

All close neighbours to internal points are included into the cluster which pulls in the boundary points without the cluster extending out beyond the boundaries into noisy areas.

TurnCut and Differencing

- Single and double differencing is used in time series analysis to render a series stationary.
- Depends on the series having ‘plateaus’ with steps/transitions in between.
- It is a kind of high-pass filter that can reveal underlying trends.
- Here, we discover areas of stability (‘plateaus’) in the clustering (if any exist).

TurnCut Example



Wish list (Turn*)

- ✓ Scalability
- Dealing with different types of attributes
- ✓ Discovery of clusters with arbitrary shape
- ✓ Minimal requirements for domain knowledge to determine input parameters
- ✓ Able to deal with noise and outliers
- ✓ Insensitive to order of input records
- ~~Handles high dimensionality~~
- ~~Can be incremental for dynamic change~~

K-means, ROCK, CURE and CHAMELEON

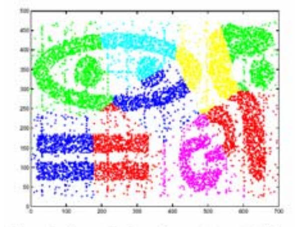


Figure 5: K-means's clustering result on t7.10k.dat with $k = 9$

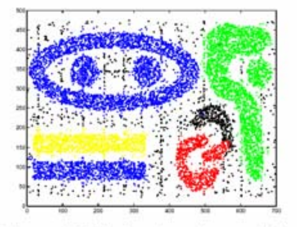


Figure 7: ROCK's clustering result on t7.10k.dat with $\theta = 0.975$ and $k = 1000$

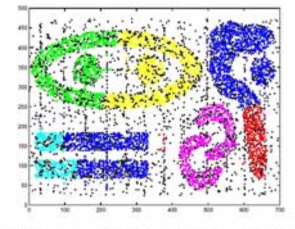


Figure 6: CURE's clustering result on t7.10k.dat with $k = 9$, $\alpha = 0.3$, and number_of_representative_points = 10

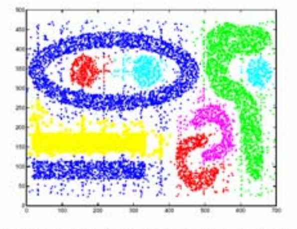


Figure 8: CHAMELEON's clustering result on t7.10k.dat with $nb\_closest\_neighbor = 10$, MinSize = 2.5% and $k = 9$

Complex 2D Spatial Dataset with 9 clusters and noise (from the CHAMELEON Paper)

Best parameters chosen after many runs where needed.

K-means Partitioning based

ROCK, CURE and Chameleon Hierarchical

DBSCAN, WaveCluster and TURN*

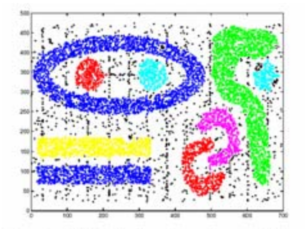


Figure 9: DBSCAN's clustering result on t7.10k.dat with $\epsilon = 5.9$ and MinPts = 4

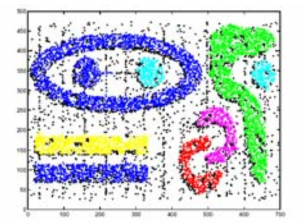


Figure 11: WaveCluster's clustering result on t7.10k.dat with $resolution = 5$ and $\tau = 1.5$

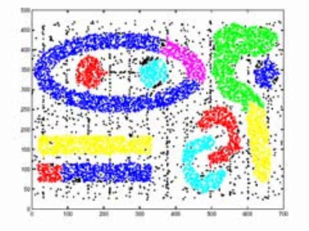


Figure 10: DBSCAN's clustering result on t7.10k.dat with $\epsilon = 5.5$ and MinPts = 4

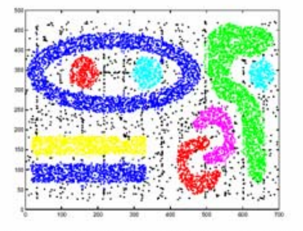


Figure 2: *TURN**'s clustering result on t7.10k.dat before cleaning

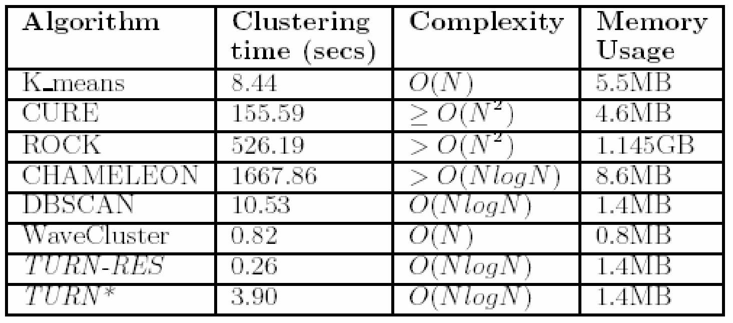
Complex 2D Spatial Dataset with 9 clusters and noise
(from the CHAMELEON Paper)

Best parameters chosen after many runs where needed.

WaveCluster
Grid-based

DBscan and Turn*
Density-based

Tabular Comparison



Algorithm	Clustering time (secs)	Complexity	Memory Usage
K_means	8.44	$O(N)$	5.5MB
CURE	155.59	$\geq O(N^2)$	4.6MB
ROCK	526.19	$> O(N^2)$	1.145GB
CHAMELEON	1667.86	$> O(NlogN)$	8.6MB
DBSCAN	10.53	$O(NlogN)$	1.4MB
WaveCluster	0.82	$O(N)$	0.8MB
TURN-RES	0.26	$O(NlogN)$	1.4MB
*TURN**	3.90	$O(NlogN)$	1.4MB

Table 1: Clustering Speed and Memory Size Results upon a data set with 10,000 data points

Lecture Outline

Part I: What is Clustering in Data Mining *(30 minutes)*

- **Introduction to clustering**
  - **Motivating Examples for clustering**
  - **Taxonomy of Major Clustering Algorithms**
  - **Major Issues in Clustering**
  - **What is Good Clustering?**

Part II: Major Clustering Approaches *(1 hour 20 minutes)*

- **K-means (Partitioning-based clustering algorithm)**
- **Nearest Neighbor clustering algorithm**
- **Hierarchical Clustering**
- **Density-based Clustering**

Part III: Open Problems *(10 minutes)*

- **Research Issues in Clustering**

Interpretation of Clusters and Validation

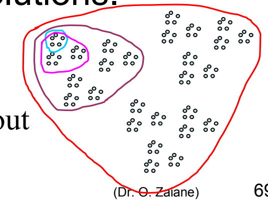
- **Issues with parameters: The best number of clusters is not known (idem with other parameters)**
  - **There is no one correct answer to a clustering problem**
  - **domain expert may be required**
- **Interpreting the semantic meaning of each cluster is difficult**
  - **What are the characteristics that the items have in common?**
  - **Domain expert is needed**
- **Cluster results are dynamic (change over time) if data is dynamic**
  - **Studying the evolution of clusters gives useful insight**
- **Cluster Validation**
  - **How good are the results of clustering algorithms?**
  - **Current validation is weak**

Resolution a Key Issue in Clustering

- All algorithms face it.
- Very few papers discuss it.
- Usually dealt with by setting parameters.
- Trying to find the best parameters misses the point - useful information may be derived from many different settings.
- There may be a 'natural' best resolution or local areas may have different key resolutions.

How many clusters?

As you zoom in and out the view changes.



Fuzzy Clustering

- Usually seen as a weighting of partial cluster membership.
- Can also be seen as a 'flattening' or alternative representation of a dendogram.
- 'Flattening' causes a loss of information regarding the transformation between resolution levels which may be used in Cluster Validation.

Scaling to VLDB

- All algorithms discussed here are for data mining and thus intended to scale to handle VLDBs. However, hierarchical algorithms don't scale well.
- Grid based methods are very effective because they condense the data.
- Methods such as DBSCAN and TURN* also scale well and compete with grid-based without the risks of condensation.

Scaling to High Dimensionality

- As the number of dimensions *D* increases, data gets sparser and any clustering effect is reduced.
- For $D > 16$ strategies for finding the near neighbours, such as indexed trees (e.g. SR-tree), fail and the computational complexity goes to $O(N^2)$.
- The length of each point's description increases requiring more memory or I/O.
- At very high dimensions, clusters may not exist. Forcing the discovery biases the results.
- Discovering sub-spaces where clusters exists and are meaningful is an open problem.

Constraint Based Clustering

- Constraints may dictate the clustering in a real application. The algorithm should consider them (constraints on the cluster or the members in a cluster)
- In spatial data: Bridges (connecting points) or obstacles (such as rivers and highways)
- Adding features to the constraints (bridge length, wall size, etc.).
- Some solutions exist:

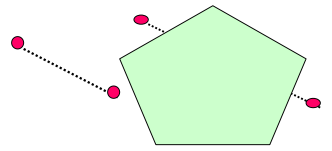
AUTOCLUST+ (Estivill-Castro and Lee, 2000)
Based on graph partitioning

COD-CLARANS (Tung, Hou, and Han, 2001)
Based on CLARANS – partitioning

DBCluC (Zaïane and Lee, 2002)
Based on DBSCAN – density based

Motivating Concepts - Obstacles

- Physical Constraints (Obstacle and Crossing)
  - An Obstacle -Disconnectivity functionality
    - A polygon denoted by *P (V, E)* where *V* is a set of points from an obstacle and *E* is a set of line segments
    - Types: Convex and Concave.
    - ***Visibility***
      - Relation between two data points, if the line segment drawn from one point to the other is not intersected with a polygon P (V, E) representing a given obstacle.



Motivating Concepts - Crossings

- Crossing – Connectivity functionality
  - *Entry point*
    - a point *p* on the perimeter of the polygon crossing, from which when a given point *r* is density-reachable wrt. *Eps*, *r* becomes reachable by any other point *q* density-reachable from any other entry point of the same crossing wrt. *Eps*
  - *Entry edge*
    - an edge of a crossing polygon with a set of entry points starting from one endpoint of the edge to the other separated by an interval value i_e where $i_e \leq$ Eps

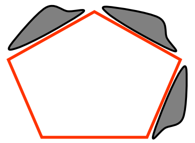
- A *Crossing is a set B of m entry points* $B=\{b_1, b_2, b_3, \ldots, b_m\}$ *generated from all entry edges*

Motivating Concepts - Cluster

- Cluster
  - A set *C* of points $C=\{c_1, c_2, c_3, \ldots, c_c\}$, satisfying the followings, where $c_i, c_j \in C$, $C \subseteq D$, $i \neq j$, and $1 \leq i, j \leq n$
    - *Maximality*: $\forall$ d_i, d_j, if $d_i \in$ C and d_j is density-reachable from d_i with respect to Eps and MinPts, then $d_j \in$ C.
    - *Connectivity*: $\forall$ c_i, $c_j \in C$, c_i is density-connected to c_j with respect to Eps and MinPts.
    - $\forall c_i, c_j \in$ C, c_i *and* c_j *are visible to each other*
    - *Crossing connectivity*: *If* C_m *and* C_k *are connected by a bridge B, then a cluster* C_l *the union of* C_m *and* C_k *is created.* $C_l = C_m \cup C_k$, *where* C_m *and* C_k *are a set of data points.* $l \neq m \neq k$

Modeling Constraints – Obstacles

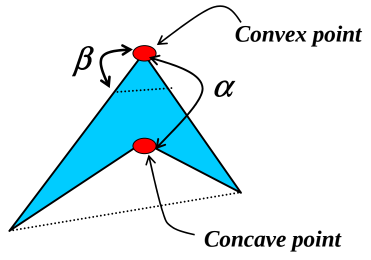
- Obstacle Modeling
  - Objectives
    - Disconnectivity Functionality.
    - Enhance performance of processing obstacles.
  - Polygon Reduction Algorithm
    - Represents an obstacle as a set of Obstruction Lines.
    - Two steps – Convex Point Test and Convex Test
    - Inspiration
      - Maintain a set of *visible spaces** created by an obstacle.



Modeling Constraints

Polygon Reduction Algorithm

- Polygon Reduction Algorithm
  1. Convex Point Test.
     - A pre-stage in order to determine if a polygon is a convex or a concave.

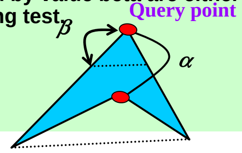


- The definition of a convex or a concave point is observed by a definition in a dictionary.
- If a line that is "close" enough (i.e. $\beta < \alpha$) to a query point is interior to a polygon, then the query point is classified as a convex point.

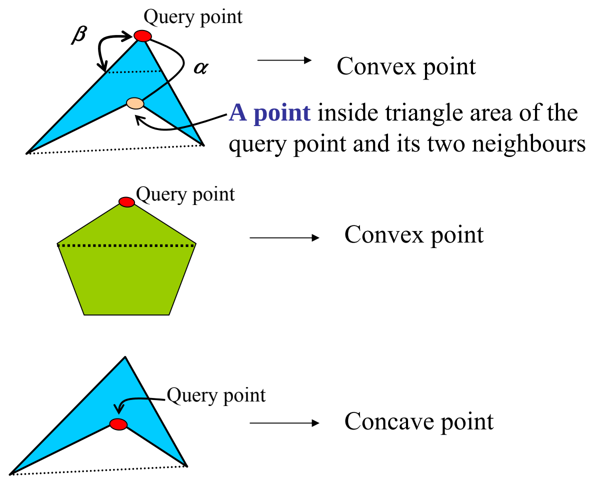
Modeling Constraints

Convex Point Test Algorithm

```
1.  Get_Neighbours(query_point)_and_Store_into_(neigh_1, neigh_2);
2.  IF there is any point insideOfTriangleArea(query_point, neigh_1, neigh_2);
3.  Then
4.      point = Find_Closest_point_from_TriangleArea(query_point,
    neigh_1,neigh_2);
5.      alpha = getDistanceBetween(query_point, point);
6.      beta = setBetaFrom(alpha);
7.      line = drawLine between  two endpoints which are on two adjacent lines of
    query_point and whose distance to query_point is "beta".
8.  Else
9.      line = drawLine between neigh_1 and neigh_2
10. EndIF
11. // since we know all points on "line" computed by value beta are either interior
    exterior to the polygon, we can do the following test.
12. IF a_point_on_line_isExterior()
13. Then
14.     Return query_point_is_CONCAVE;
15.  Return query_point_is_CONVEX;
```



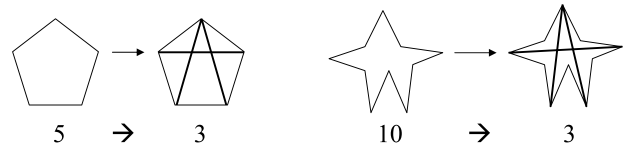
Examples of Convex Point Test



Modeling Constraints
– Polygon Reduction Algorithm

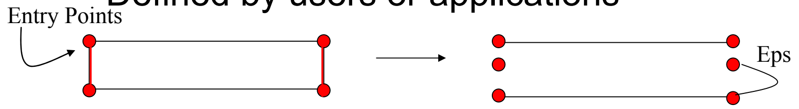
1. Convex Test - Determine if a polygon is a convex or a concave.
   - A polygon is *Concave* if ∃ a concave point in the polygon.
   - A polygon is *Convex* if ∀ points are convex points.

- Convex - ⌈n/2⌉ obstruction lines*.
- Concave – The number of obstruction lines depends on a shape of a given polygon.

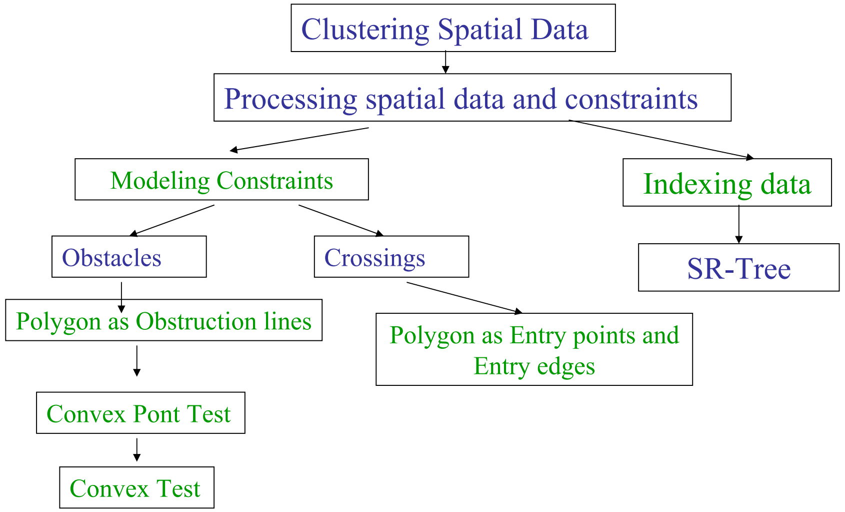


Modeling Constraints – Crossings

- Crossing Modeling
  - Objective
    - Connectivity functionality.
    - Control Flow of data.
  - A polygon with *Entry Points* and *Entry Edge*.
  - Defined by users or applications



DBCluC



DBCluC algorithm

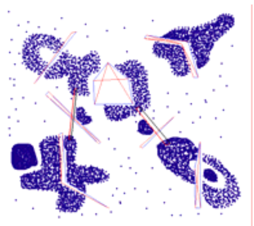
- Start from an arbitrary data point.
- Indexing data points with SR-tree
  - K-NN and Range Query query available.
- Consider crossing constraints *while* clustering.
- Consider obstacles after retrieving neighbours of a given query point.
  - Visibility between a query point and its neighbours is checked for all obstacles.

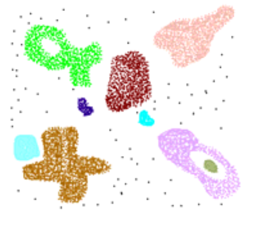
DBCluC - algorithm

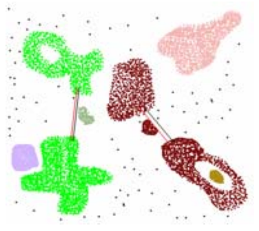
```
DBCluC {
      // While clustering, bridges are taken into account
1.    START CLUSTERING FROM BRIDGE ENTRY POINTS
2.    FOR REMAINING DATA POINTS Point FROM A Database
3.        IF ExpandCluster (Database,Point, ClusterId, Eps, MinPts, Obstacles)
          THEN
4.        ClusterId = nextId(ClusterId)
5.        END IF
6.    END FOR
}
```

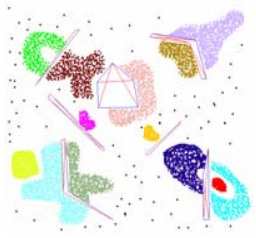
- Complexity
  - Polygon Reduction Algorithm O(n^2), where *n* is the number of points of the largest polygon in terms of edges.
  - DBCluC – O($N*logN*L$), where *N* is the number of data points and *L* is the number of obstruction lines.

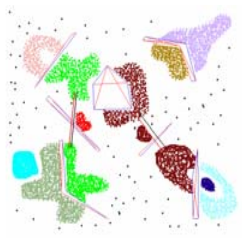
Some Illustrative Results


(a) Before clustering


(b) Clustering ignoring constraints


(a) Clustering with bridges


(a) Clustering with obstacles


(a) Clustering with obstacles and bridges