Chapter 13

Big Data Mining - Clustering

Acknowledgements

- Georg Gerber. BIO3120.
- Kamal Nigam. Internet Search Technologies, Lecture 8: Clustering.
- Pandu Nayak and Prabhakar Raghavan. CS276: Information Retrieval and Web Search, Lecture 12: Clustering.
- DBSCAN – Density-Based Spatial Clustering of Applications with Noise. (referring to : M.Ester, H.P.Kriegel, J.Sander and Xu. A density-based algorithm for discovering clusters in large spatial databases.)
- Iris Zhang. Density-Based Clustering Algorithms. HKU CS.
- University at Buffalo. Why Density-Based Clustering?

Chapter Outline

- What is clustering?
- Applications
- Issues for clustering
- Algorithms
- Big data clustering

What is clustering?

What is clustering?

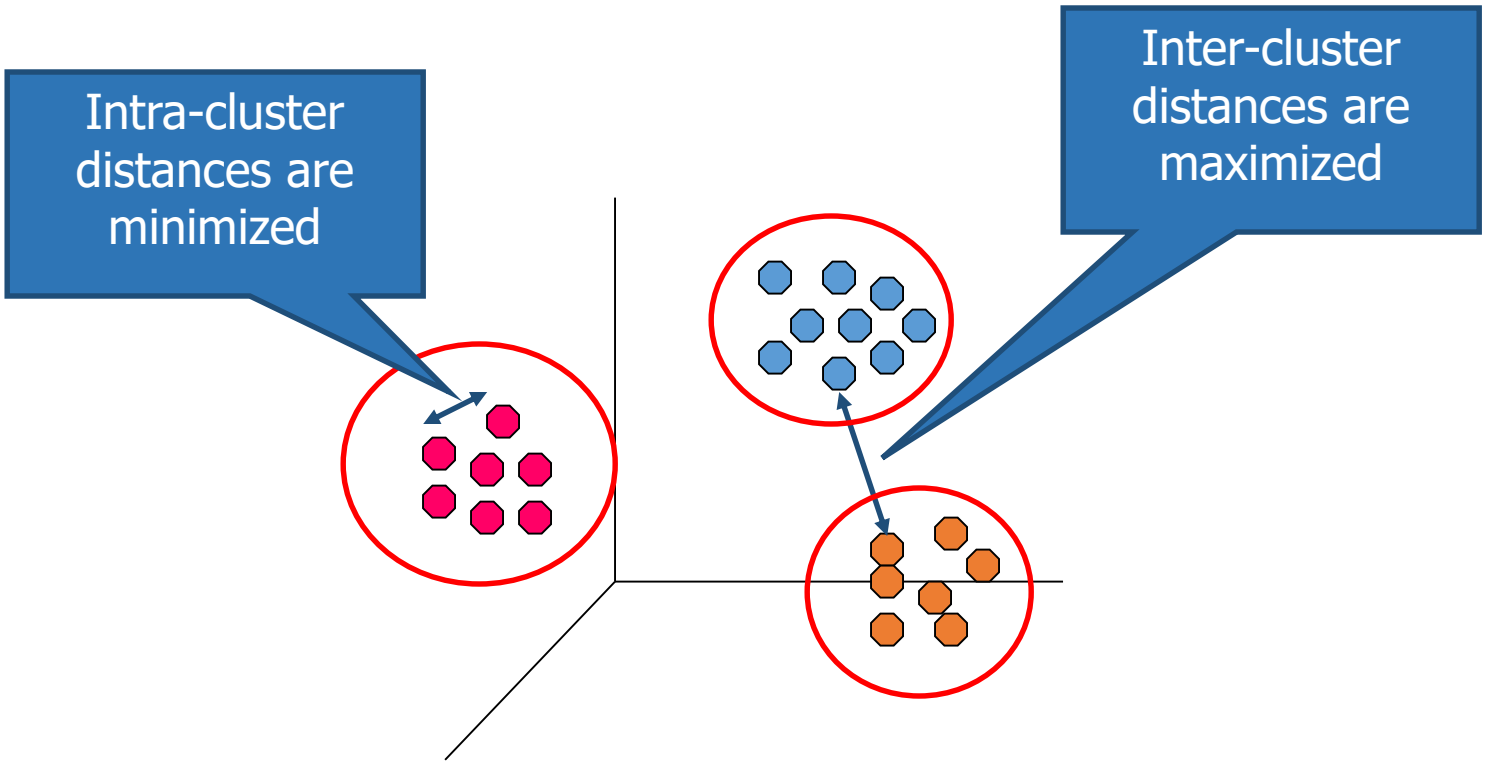
- **Clustering**: the process of grouping a set of objects in such a way that
  - Distance between objects within partition (I.e. same **cluster**) is minimized
  - Distance between objects from different clusters is maximized

- The commonest form of ***unsupervised learning***
  - Unsupervised learning = learning from raw data, as opposed to supervised data where a classification of examples is given
  - A common and important task that finds many applications in IR and other places

Why do we cluster?

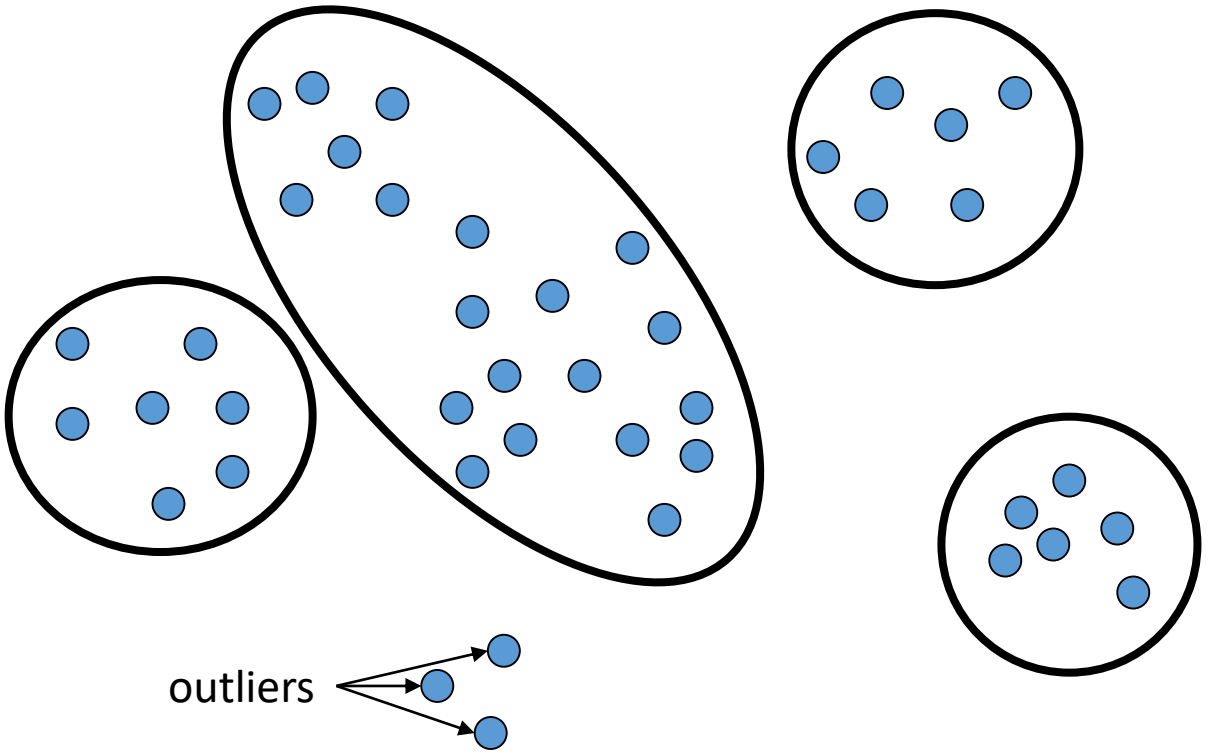
- Clustering results are used:
  - As a stand-alone tool to get insight into data distribution
    - Visualization of clusters may unveil important information
  - As a preprocessing step for other algorithms
    - Efficient indexing or compression often relies on clustering

Clustering


Intra-cluster distances are minimized
Inter-cluster distances are maximized

Outliers

- **Outliers** are **objects that do not belong to any cluster** or form clusters of very small cardinality
- In some applications we are interested in discovering outliers, not clusters (outlier analysis)



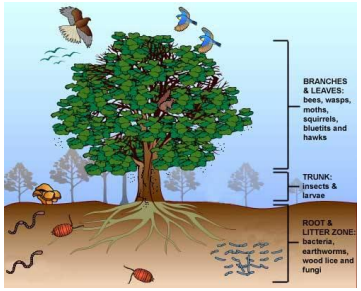
Hard vs. soft clustering

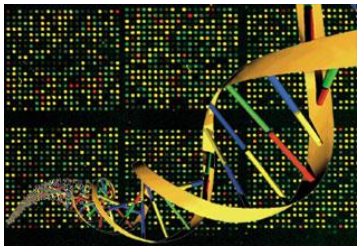
- **Hard clustering**: Each document belongs to exactly one cluster
  - More common and easier to do
- **Soft clustering**: A document can belong to more than one cluster.
  - Makes more sense for applications like creating browsable hierarchies
  - You may want to put a pair of sneakers in two clusters: (i) sports apparel and (ii) shoes
  - You can only do that with a soft clustering approach.

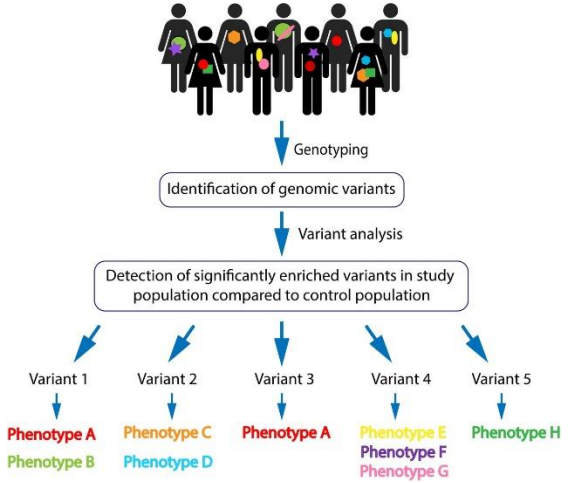
Applications

Biology and Bioinformatics

- Plant and animal ecology (动植物生态)
  - Describe and make spatial and temporal comparisons of communities of organisms in heterogeneous environments
- Transcriptomics (转录组学)
  - Build groups of genes with related expression patterns (also known as coexpressed genes) as in HCS clustering algorithm
- High-throughput genotyping platforms (基因型分型)
  - Automatically assign genotypes
- Human genetic clustering (基因序列)
  - The similarity of genetic data is used in clustering to infer population structures

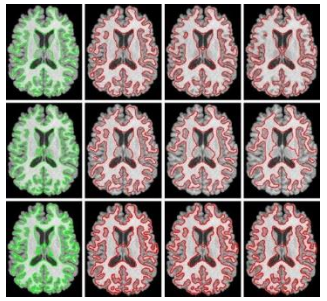






Medicine

- Medical imaging (医疗影像)
  - Differentiate between different types of tissue in a three-dimensional image for many different purposes
- Analysis of antimicrobial activity (抗菌活性)
  - Analyze patterns of antibiotic resistance
  - Classify antimicrobial compounds according to their mechanism of action
  - Classify antibiotics according to their antibacterial activity
- IMRT segmentation (调强放射治疗)
  - Divide a fluence map into distinct regions for conversion into deliverable fields in MLC-based Radiation Therapy



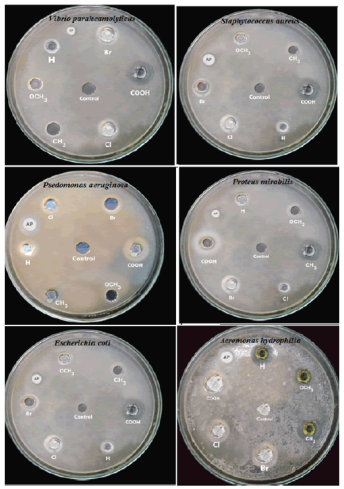
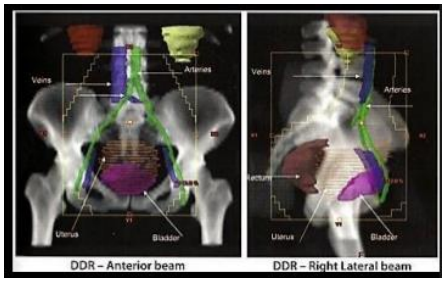


Figure 1: Antimicrobial activity of substituted 2-benzylidene-1,3-indandiones.

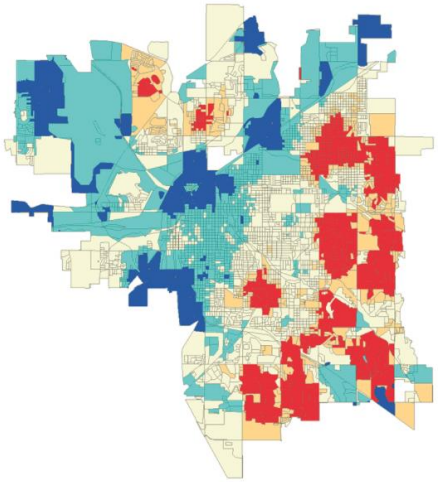


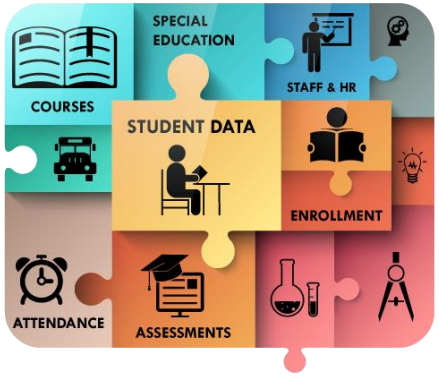
Business and Marketing

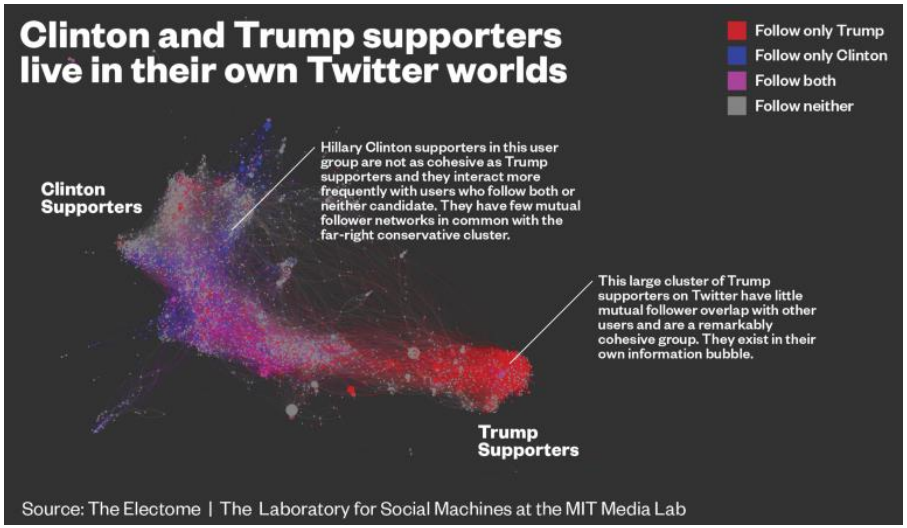
- Market research
  - Work with multivariate data from surveys and test panels
  - Partition the general population of consumers into market segments and to better understand the relationships between different groups of consumers/potential customers
  - For use in market segmentation, Product positioning, New product development and Selecting test markets
- Grouping of shopping items
  - Group all the shopping items available on the web into a set of unique products
  - For example, all the items on eBay can be grouped into unique products. (eBay doesn't have the concept of a SKU)

Social Science

- Crime analysis
  - Identify areas where there are greater incidences of particular types of crime
  - Manage law enforcement resources more effectively
- Educational data mining
  - Identify groups of schools or students with similar properties
- Typologies
  - Discern typologies of opinions, habits, and demographics that may be useful in politics and marketing







Issues for clustering

Some Questions

- What does "similar" mean?
- What is a good partition of the objects?
  - I.e., how is the quality of a solution measured?
- How many clusters?
  - Fixed a priori?
  - Completely data driven?
- Can we label or name the clusters?
- What algorithm and approach do we take?
  - Top-down: k-means
  - Bottom-up: hierarchical agglomerative clustering
- How do we make it efficient and scalable?

Issues for clustering

Data Distance and Similarity

Cluster Distance and Similarity

Clustering Evaluation

Data Structures

- *Data* matrix

attributes/dimensions

tuples/objects

$$\begin{bmatrix} x_{11} & \cdots & x_{1\ell} & \cdots & x_{1d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{i\ell} & \cdots & x_{id} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{n\ell} & \cdots & x_{nd} \end{bmatrix}$$

- *Distance* matrix

objects

objects

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Distance functions

- The distance **d(*x*, *y*)** between two objects ***x*** and ***y*** is a metric if
  - **d(*i*, *j*)≥0** (non-negativity)
  - **d(*i*, *i*)=0** (isolation)
  - **d(*i*, *j*)= d(*j*, *i*)** (symmetry)
  - **d(*i*, *j*) ≤ d(*i*, *h*)+d(*h*, *j*)** (triangular inequality) [**Why do we need it?**]

- The definitions of distance functions are usually different for real, boolean, categorical, and ordinal variables.

- Weights may be associated with different variables based on applications and data semantics.

Data Types

- Real-value/Interval-scaled variables
  - e.g., salary, height
- Binary variables
  - e.g., gender (M/F), has_cancer(T/F)
- Nominal (categorical) variables
  - e.g., religion (Christian, Muslim, Buddhist, Hindu, *etc*.)
- Ordinal/Ranked variables
  - e.g., military rank (soldier, sergeant, lutenant, captain, *etc*.)
- Variables of mixed types
  - multiple attributes with various types

Interval-valued variables

- Standardize data
  - Calculate the mean absolute deviation:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|)$$

where $m_f = \frac{1}{n}(x_{1f} + x_{2f} + ... + x_{nf})$.

  - Calculate the standardized measurement (*z-score*)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

- Using mean absolute deviation is more robust than using standard deviation

Similarity Between Objects

- Distances are normally used to measure the similarity or dissimilarity between two data objects
- Some popular ones include: *Minkowski distance*:

$$d(i,j)=\sqrt[q]{(|x_{i1}-x_{j1}|^q+|x_{i2}-x_{j2}|^q+...+|x_{ip}-x_{jp}|^q)}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two p-dimensional data objects, and q is a positive integer

- If $q = 1$, d is Manhattan distance

$$d(i,j)=|x_{i1}-x_{j1}|+|x_{i2}-x_{j2}|+...+|x_{ip}-x_{jp}|$$

Similarity Between Objects

- *If q = 2, d* is Euclidean distance:

$$d(i,j)=\sqrt{(|x_{i_1}-x_{j_1}|^2+|x_{i_2}-x_{j_2}|^2+...+|x_{i_p}-x_{j_p}|^2)}$$

  - Properties
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$

- Also one can use weighted distance, parametric Pearson product moment correlation, or other disimilarity measures.

Binary Variables

- **Jaccard similarity** between binary vectors ***X*** and ***Y***

$$JSim(X,Y) = \frac{X \cap Y}{X \cup Y}$$

- **Jaccard distance** between binary vectors ***X*** and ***Y***

$$JDist(X,Y) = 1 - JSim(X,Y)$$

- Example:
  - JSim = 1/6
  - JDist = 5/6

	Q1	Q2	Q3	Q4	Q5	Q6
X	1	0	0	1	1	1
Y	0	1	1	0	1	0

Nominal Variables

- A generalization of the binary variable in that it can take more than 2 states, e.g., red, yellow, blue, green
- Method 1: Simple matching
  - *m*: # of matches, *p*: total # of variables

$$d(i, j) = \frac{p - m}{p}$$

- Method 2: use a large number of binary variables
  - creating a new binary variable for each of the *M* nominal states

Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
  - replacing x_{if} by their rank $r_{if} \in \{1, \ldots, M_f\}$
  - map the range of each variable onto [0, 1] by replacing *i*-th object in the *f*-th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

  - compute the dissimilarity using methods for interval-scaled variables

Variables of Mixed Types

- A database may contain all the six types of variables
  - symmetric binary, asymmetric binary, nominal, ordinal, interval and ratio.
- One may use a weighted formula to combine their effects.

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

  - f is binary or nominal:
    $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$, or $d_{ij}^{(f)} = 1$ o.w.
  - f is interval-based: use the normalized distance
  - f is ordinal or ratio-scaled
    - compute ranks r_{if} and
    - and treat z_{if} as interval-scaled

Issues for clustering

Data Distance and Similarity

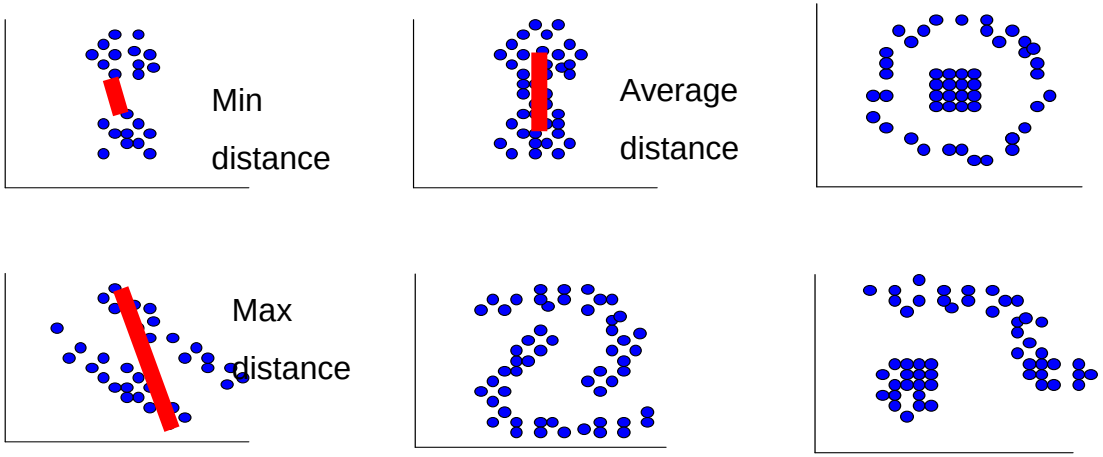
Cluster Distance and Similarity

Clustering Evaluation

Cluster Distances

- **Single-link clustering**
  - also called the connectedness or minimum method
  - equal to the shortest distance from any member of one cluster to any member of the other cluster
  - If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster
- **Complete-link clustering**
  - also called the diameter or maximum method
  - equal to the longest distance from any member of one cluster to any member of the other cluster
- **Average-link clustering**
  - equal to the average distance from any member of one cluster to any member of the other cluster.

Cluster Distances

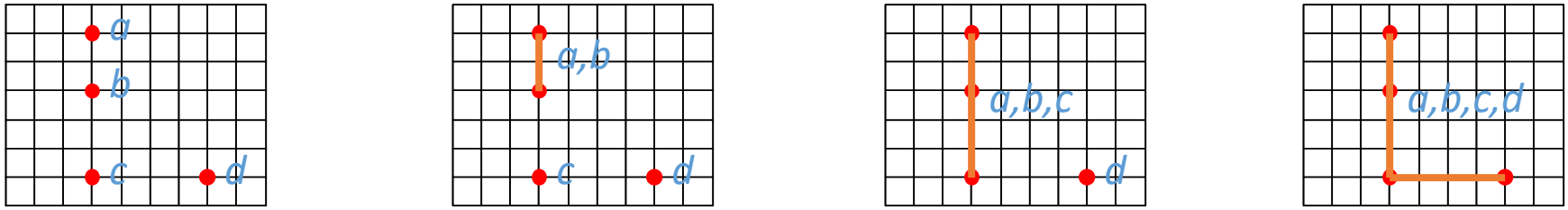


- ❑Single-Link Method / Nearest Neighbor
- ❑Complete-Link / Furthest Neighbor
- ❑Their Centroids.
- ❑Average of all cross-cluster pairs.

Cluster Distances

- Single-Link Method

Euclidean Distance



(1) (2) (3)

	b	c	d
a	2	5	6
b		3	5
c			4

	b	c	d
a	2	5	6
b		3	5
c			4

	c	d
a,b	3	5
c		4

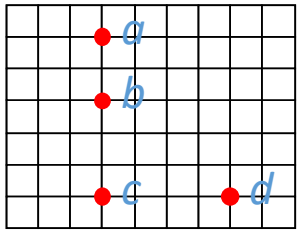
	d
a,b,c	4

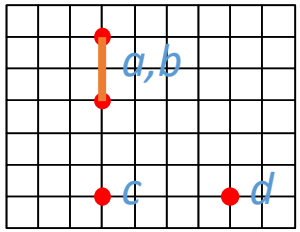
Distance Matrix

Cluster Distances

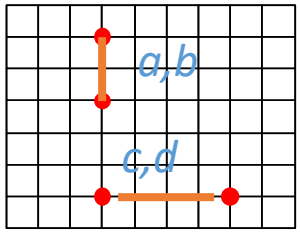
- Complete-Link Method

Euclidean Distance

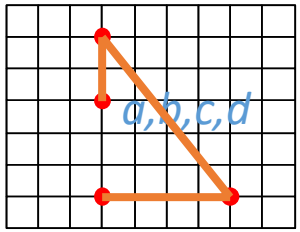




(1)



(2)



(3)

	b	*c*	*d*
a	2	5	6
b		3	5
c			4

	b	*c*	*d*
a	2	5	6
b		3	5
c			4

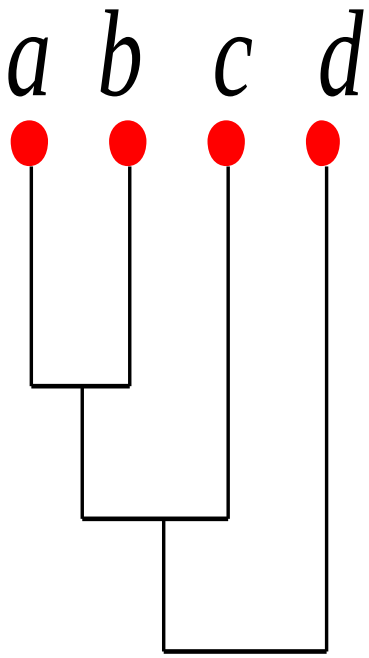
	c	*d*
a,b	5	6
c		4

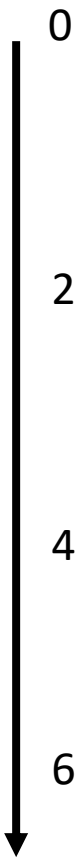
	c,d
a,b	6

Distance Matrix

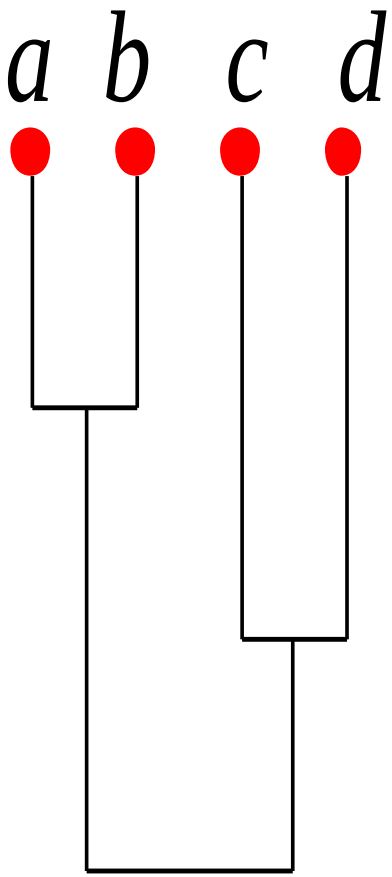
Compare Dendrograms

Single-Link





Complete-Link



Other Distances

- Centroid distance between clusters C_i and C_j is the distance between the centroid r_i of C_i and the centroid r_j of C_j

$$D_{centroids}(C_i, C_j) = d(r_i, r_j)$$

- Ward's distance between clusters C_i and C_j is the *difference* between the *total within cluster sum of squares for the two clusters separately*, and the *within cluster sum of squares resulting from merging the two clusters* in cluster C_{ij}

$$D_w(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

r_i: centroid of C_i
r_j: centroid of C_j
r_{ij}: centroid of C_{ij}

Issues for clustering

Data Distance and Similarity
Cluster Distance and Similarity
Clustering Evaluation

What Is A Good Clustering?

- **Internal criterion**: A good clustering will produce high quality clusters in which:
  - the **intra-cluster** similarity is high
  - the **inter-cluster** similarity is low
- The measured quality of a clustering depends on both the document representation and the similarity measure used
- Criteria
  - Purity
  - Rand index
  - F-measure

Purity

- Quality measured by its ability to discover some or all of the hidden patterns or latent classes in gold standard data
- Assesses a clustering with respect to ground truth … requires *labeled data*
- Assume documents with *C* gold standard classes, while our clustering algorithms produce *K* clusters, $\omega_1, \omega_2, \ldots, \omega_K$ with n_i members.

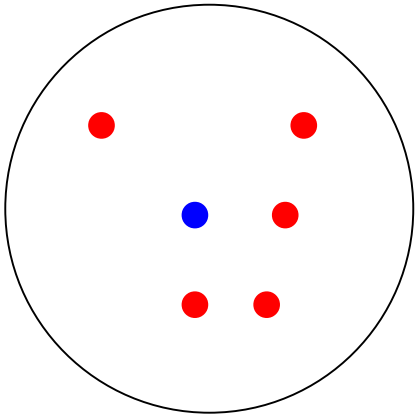
Purity

- Simple measure: purity, the ratio between the dominant class in the cluster π_i and the size of cluster ω_i

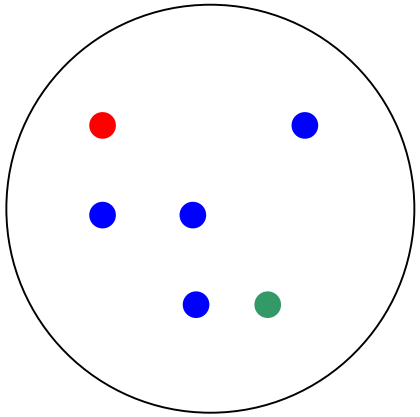
$$Purity(\omega_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

- Biased because having *n* clusters maximizes purity
- Others are entropy of classes in clusters (or mutual information between classes and clusters)

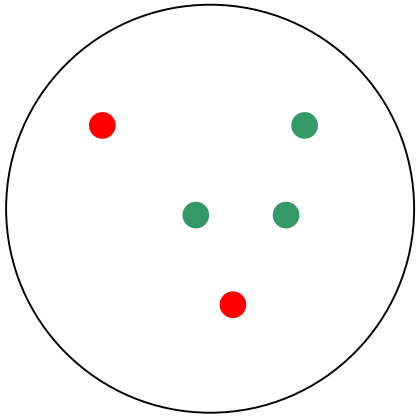
Purity example



Cluster I



Cluster II



Cluster III

Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Rand Index

- Rand Index measures between pair decisions. Here RI = 0.68

Number of points	Same Cluster in clustering	Different Clusters in clustering
Same class in ground truth	20	24
Different classes in ground truth	20	72

Rand Index and Cluster F-measure

$$RI = \frac{A + D}{A + B + C + D}$$

Compare with standard Precision and Recall:

$$P = \frac{A}{A + B} \qquad R = \frac{A}{A + C}$$

People also define and use a cluster F-measure, which is probably a better measure.

Methods

Partition Based Clustering

Density Based Clustering

Hierarchical Clustering

Partitioning Algorithms

- **Partitioning method**: Construct a partition of *n* documents into a set of *K* clusters
- **Given**: a set of documents and the number *K*
- **Find**: a partition of *K* clusters that optimizes the chosen partitioning criterion
  - Globally optimal
    - Intractable for many objective functions
    - Ergo, exhaustively enumerate all partitions
- Different metrics:
  - *K*-means
  - *K*-medoids
  - *K*-median
  - *K*-center

K-Means

- Assumes documents are real-valued vectors.
- Clusters based on *centroids* (aka the *center of gravity* or mean) of points in a cluster, *c*:

$$\vec{\mu}(c) = \frac{1}{|c|} \sum_{\vec{x} \in c} \vec{x}$$

- Reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities)

K-Means Algorithm

Select *K* random docs $\{s_1, s_2, \ldots s_K\}$ as seeds.

Until clustering *converges* (or other stopping criterion):

For each data point d_i:

Assign x_i to the cluster c_j such that $dist(x_i, s_j)$ is minimal.

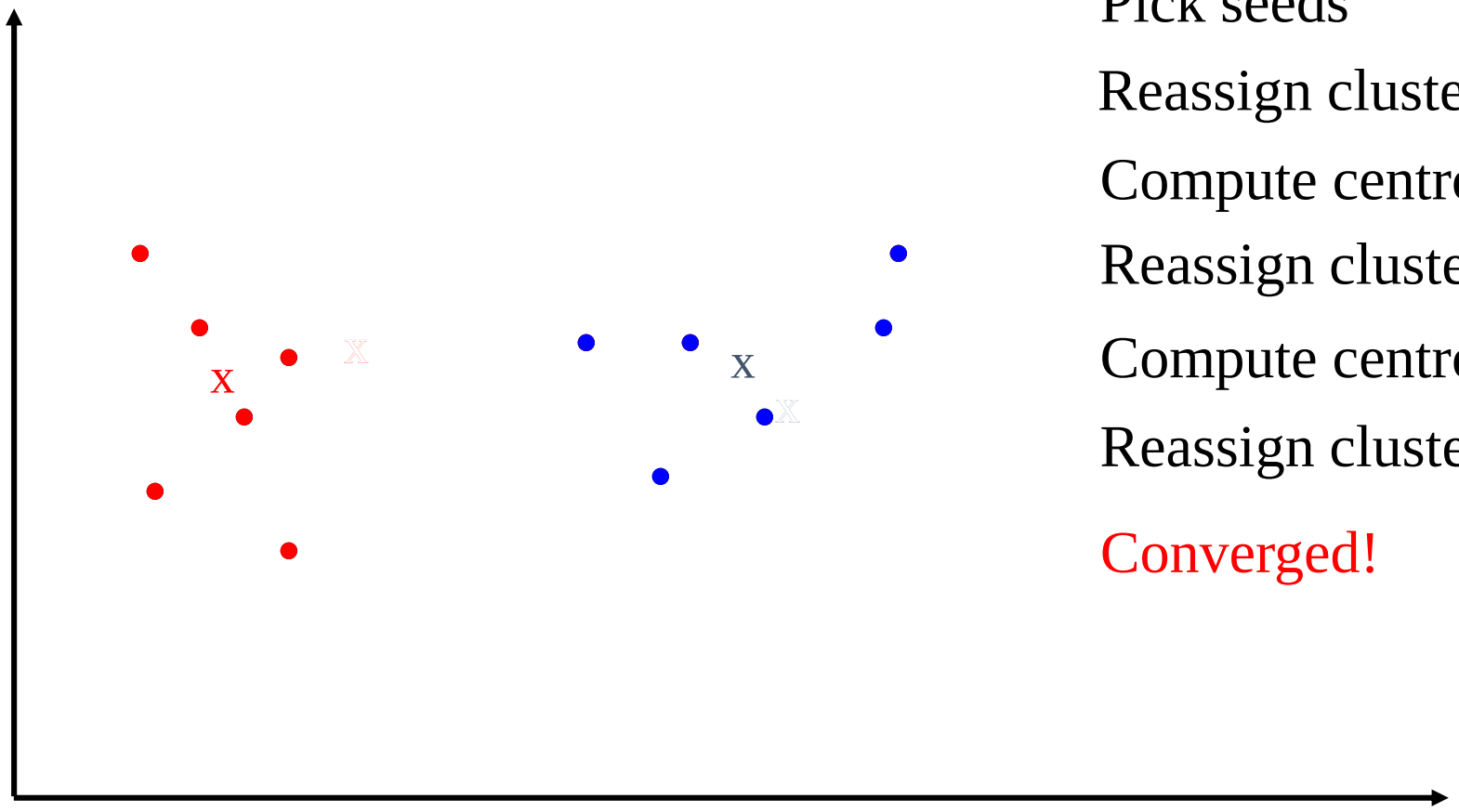
(*Next, update the seeds to the centroid of each cluster*)

For each cluster c_j

Compute the new centers of the clusters

$$Centroid(S) = \sum_{i=1}^{n} x_i / n, x_1, \ldots, x_n \in c_j$$

K-Means Example (K=2)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

Convergence

- Why should the *K*-means algorithm ever reach a *fixed point*?
  - A state in which clusters don't change.
- *K*-means is a special case of a general procedure known as the *Expectation Maximization* (EM) *algorithm*.
  - EM is known to converge.
  - Number of iterations could be large.
    - But in practice usually isn't
- Termination conditions, e.g.,
  - A fixed number of iterations.
  - Clusters unchanged. (Converged)
  - Centroid positions don't change. (Converged)

Seed Choice

- Results can vary based on random seed selection.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
  - Select good seeds using a heuristic
    - e.g., pick the most distant (from each other) points as cluster centers (kmeans++ algorithm)
  - Try out multiple starting points
  - Initialize with the results of another method.

Example showing sensitivity to seeds

In the above, if you start with B and E as centroids you converge to {A,B,C} and {D,E,F}
If you start with D and F you converge to {A,B,D,E} {C,F}

How Many Clusters?

- Number of clusters *K* is given
  - Partition *n* data points into predetermined number of clusters
- Finding the "right" number of clusters is part of the problem
  - Partition into an "appropriate" number of subsets
  - E.g., for query results - ideal value of *K* not known up front - though UI may impose limits
- Can usually take an algorithm for one flavor and convert to the other

Non-specified *K*

- Solve an optimization problem: penalize having lots of clusters
  - application dependent, e.g., compressed summary of search results list.
- Tradeoff between having more clusters (better focus within each cluster) and having too many clusters

Non-specified *K*

- Given a clustering, define the Benefit for a data point to be the cosine similarity to its centroid
- Define the Total Benefit to be the sum of the individual doc Benefits.
- For each cluster, we have a Cost *C*.
- Thus for a clustering with *K* clusters, the Total Cost is *KC*.
- Define the **Value** of a clustering to be

  Value = Total Benefit - Total Cost
- Find the clustering of ***highest value***, over all choices of *K*.
  - Total benefit increases with increasing *K*. But can stop when it doesn't increase by "much". The Cost term enforces this.

Methods

Partition Based Clustering

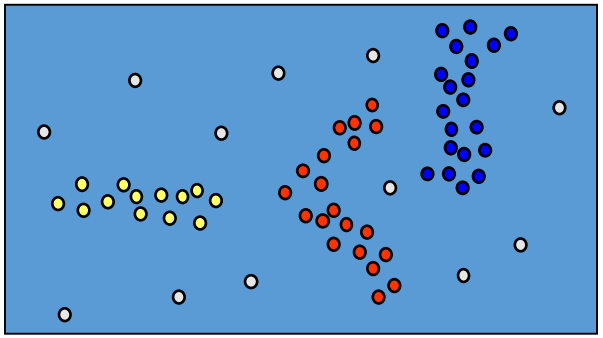
Density Based Clustering

Hierarchical Clustering

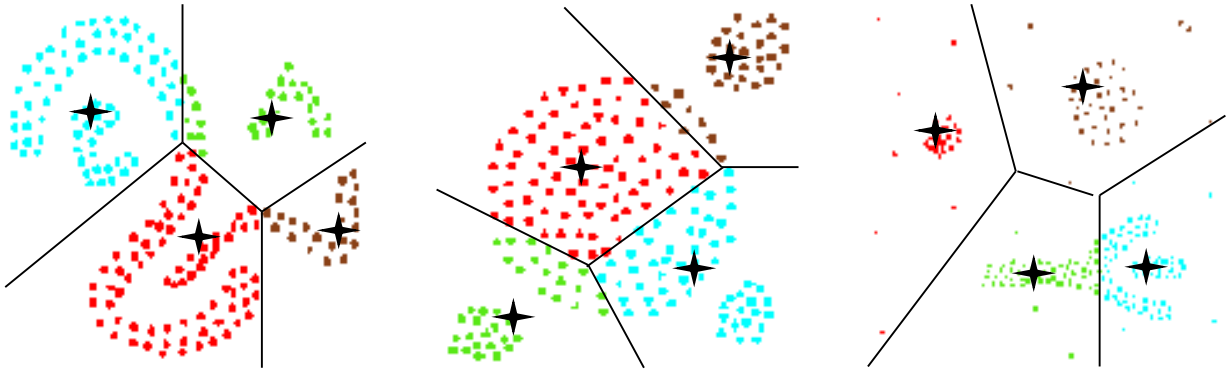
Density-Based Clustering

- *Basic Idea*:

  Clusters are dense regions in the data space, separated by regions of lower object density



- Why Density-Based Clustering?



Results of a *k*-medoid algorithm for *k*=4

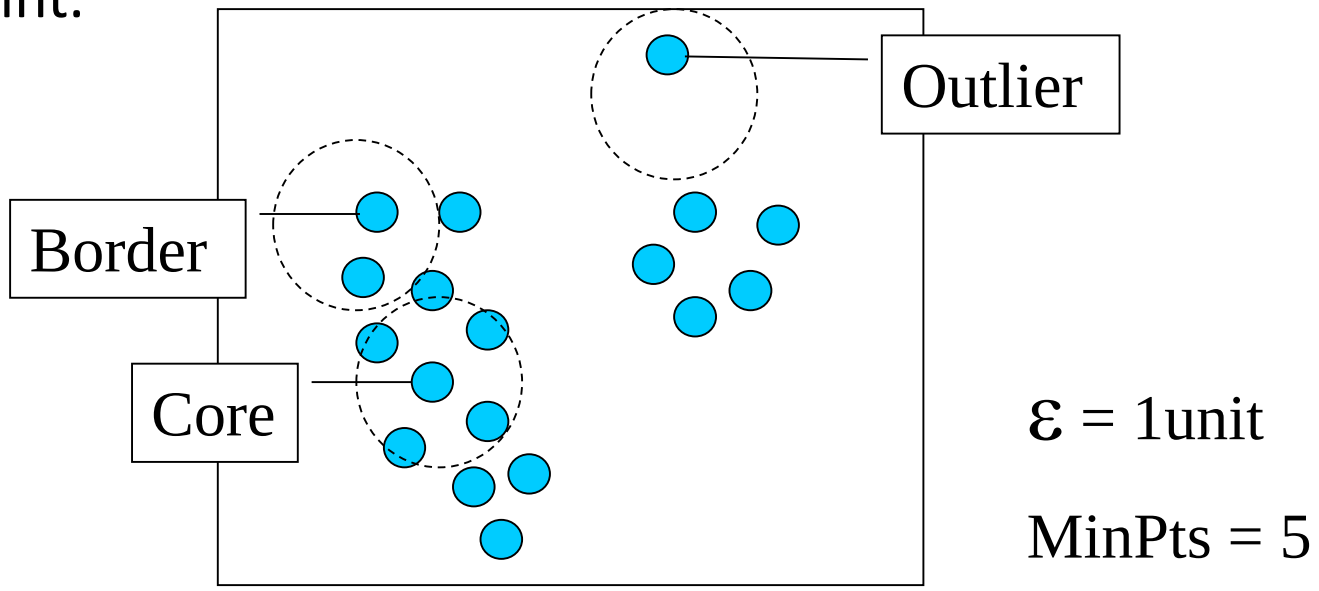
Different density-based approaches exist (see Textbook & Papers)
Here we discuss the ideas underlying the DBSCAN algorithm

Density-Based Clustering

- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters
- Several interesting studies:
  - **DBSCAN**: Ester, et al. (KDD'96)
  - **DENCLUE**: Hinneburg & D. Keim (KDD'98/2006)
  - **OPTICS**: Ankerst, et al (SIGMOD'99).
  - **CLIQUE**: Agrawal, et al. (SIGMOD'98)

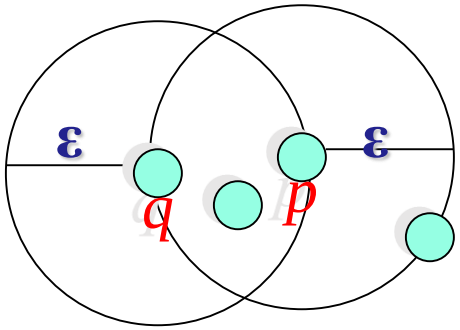
DBSCAN: Basic Concepts

- **Density** = number of points within a specified radius *r* (**Eps**)
- A point is a **core point** if it has more than a specified number of points (**MinPts**) within Eps
- A **border point** has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A **noise point** is any point that is not a core point or a border point.



Concepts: ε-Neighborhood

- **ε-Neighborhood** - Objects within a radius of ε from an object. (epsilon-neighborhood)
- **Core objects** - ε-Neighborhood of an object contains at least MinPts of objects



ε-Neighborhood of p

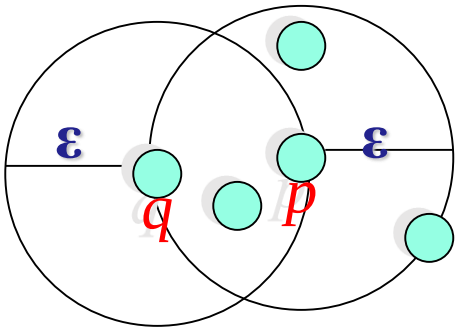
ε-Neighborhood of q

p is a core object (MinPts = 4)

q is not a core object

Concepts: Reachability

- **Directly density-reachable**
  - An object *q* is directly density-reachable from object *p* if *q* is within the ε-Neighborhood of *p* and *p* is a core object.

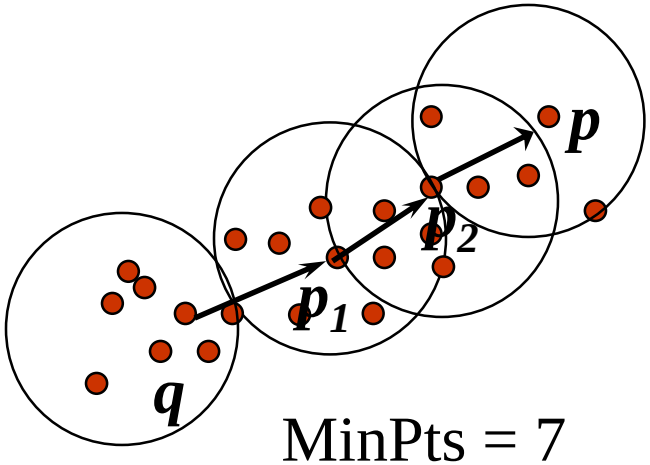


MinPts = 4

- *q* is directly density-reachable from *p*
- *p* is not directly density- reachable from *q*?

Concepts: Reachability

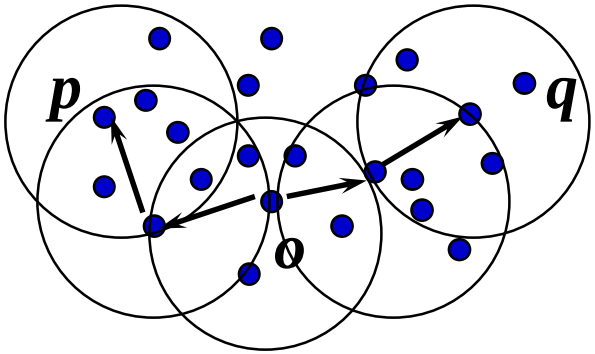
- Density-Reachable (directly and indirectly):
  - A point p is directly density-reachable from p_2;
  - p_2 is directly density-reachable from p_1;
  - p_1 is directly density-reachable from q;
  - $p \leftarrow p_2 \leftarrow p_1 \leftarrow q$ form a chain.



- p is (indirectly) density-reachable from q
- q is not density- reachable from p?
- Transitive closure of direct density-Reachability, **asymmetric**

Concepts: Connectivity

- **Density-connectivity**
  - Object *p* is density-connected to object *q* w.r.t *ε* and *MinPts* if there is an object *o* such that both *p* and *q* are density-reachable from *o* w.r.t *ε* and *MinPts*

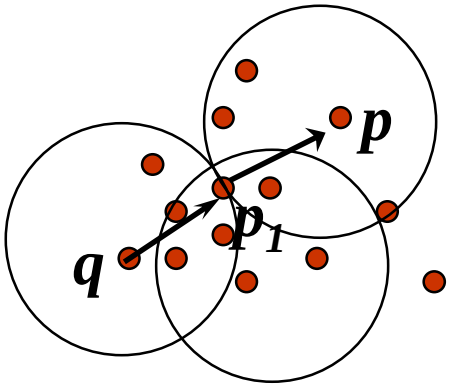


- *p* and *q* are density-connected to each other by *o*
- Density-connectivity is **symmetric**

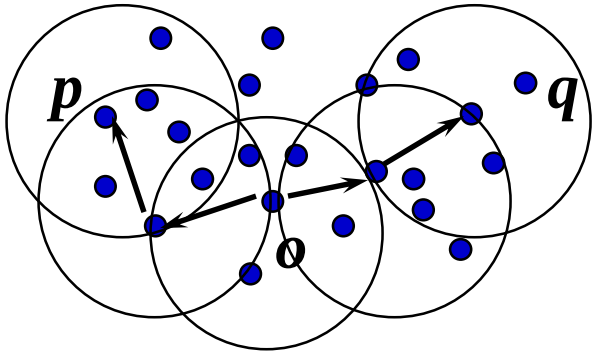
Concepts: Cluster & Noise

- **Cluster**: a cluster ***C*** in a set of objects ***D*** w.r.t ε and *MinPts* is a non empty subset of ***D*** satisfying
  - Maximality: For all *p*, *q* if $p \in \boldsymbol{C}$ and if *q* is density-reachable from *p* w.r.t ε and *MinPts*, then also $q \in \boldsymbol{C}$.
  - Connectivity: for all *p*, $q \in \boldsymbol{C}$, *p* is density-connected to *q* w.r.t ε and *MinPts* in ***D.***
  - ***Note:*** cluster contains *core objects* as well as *border objects*
- **Noise:** objects which are not directly density-reachable from at least one core object.

Density-reachable



Density-connected

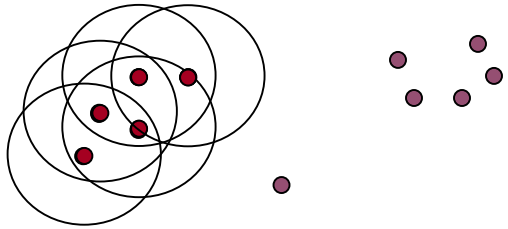


DBSCAN: The Algorithm

1. Arbitrary select a point *p*
2. Retrieve all points density-reachable from *p* wrt *Eps* and *MinPts*.
3. If *p* is a core point, a cluster is formed.
4. If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database.
5. Continue the process until all of the points have been processed.

DBSCAN Algorithm: Example

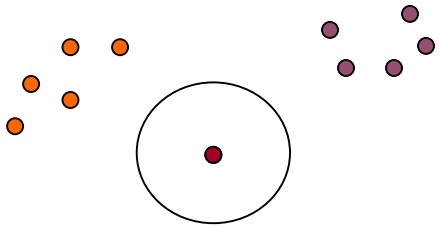
- Parameter
  - ε = 2 cm
  - *MinPts* = 3



for each $o \in D$ **do**
  if o is not yet classified **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

DBSCAN Algorithm: Example

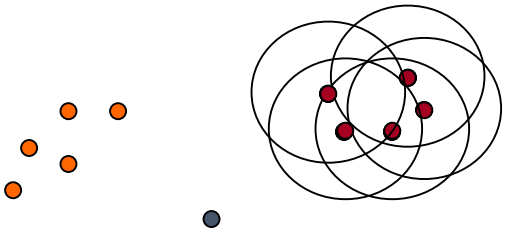
- Parameter
  - ε = 2 cm
  - *MinPts* = 3



for each $o \in D$ **do**
  if o is not yet classified **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

DBSCAN Algorithm: Example

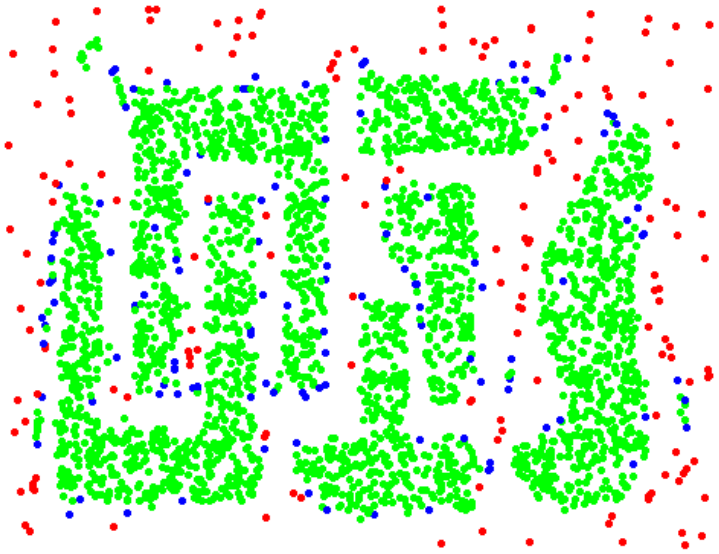
- Parameter
  - ε = 2 cm
  - *MinPts* = 3



for each $o \in D$ **do**
  if o is not yet classified **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

When DBSCAN Works Well

Original Points

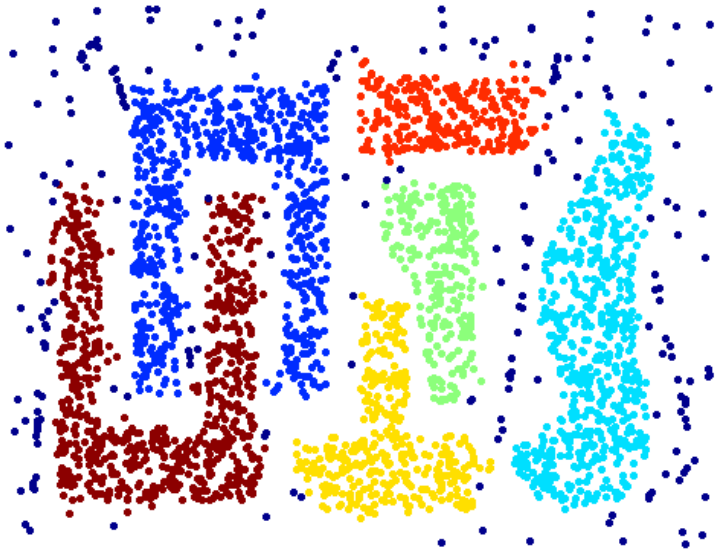


Point types: core, border and noise

Eps = 10, MinPts = 4

When DBSCAN Works Well

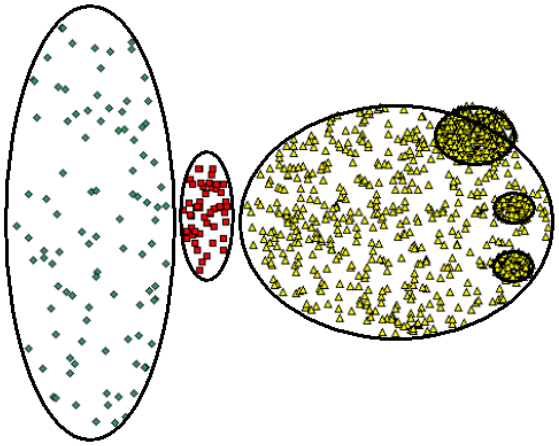
Original Points



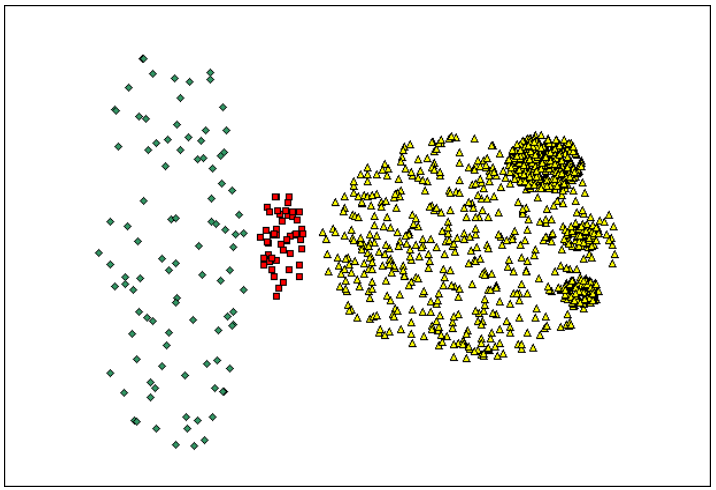
Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

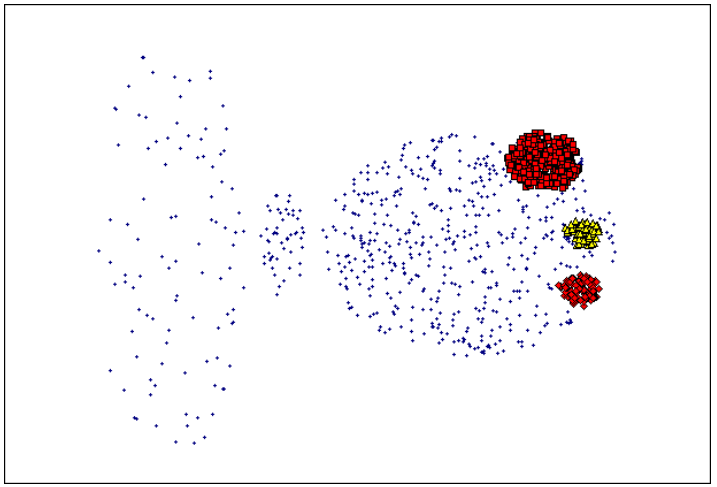
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

- Varying densities
- High-dimensional data

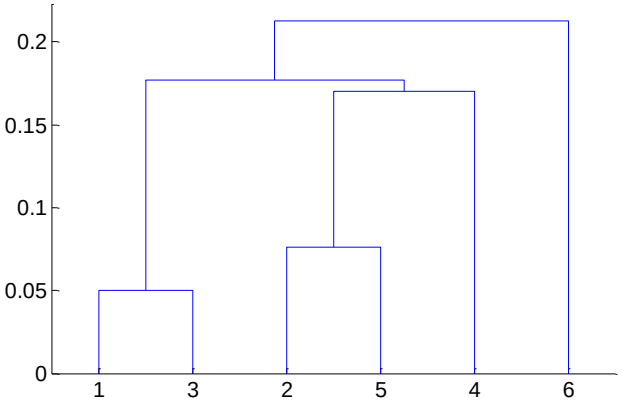
Traits

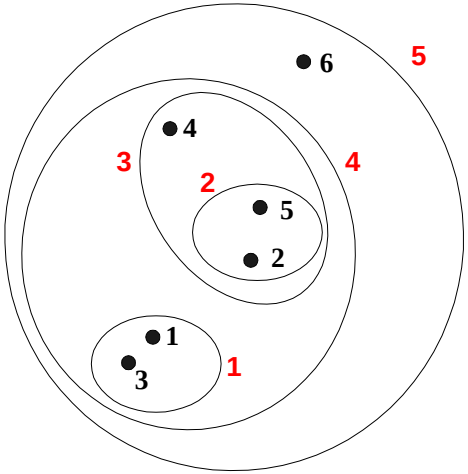
- Pros
  - Clusters can have arbitrary shape and size
  - Not very sensitive to noise
  - Number of clusters is determined automatically
  - Can be supported by spatial index structures
  - Supports outlier detection
  - beside *k*-means the 2nd most used clustering algorithm
- Cons
  - Does not work well in high-dimensional datasets
  - In some situations very sensitive to input parameter setting

Partition Based Clustering
Density Based Clustering
Hierarchical Clustering

Hierarchical Clustering

- Produces a set of ***nested clusters*** organized as a hierarchical tree
- Basic idea:
  1. Starts with each data point in a separate cluster
  2. Then repeatedly joins the closest pair of clusters, until there is only one cluster
- Can be visualized as a **dendrogram**
  - A tree-like diagram that records the sequences of merges or splits





Traits

- Pros
  - No assumptions on the number of clusters
    - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
  - Hierarchical clusterings may correspond to meaningful taxonomies
    - Example in biological sciences (e.g., phylogeny reconstruction, etc), web (e.g., product catalogs), *etc*.
- Cons
  - Cluster distance matrix is used for deciding which clusters to merge/split
  - At least quadratic in the number of data points
  - Not usable for large datasets

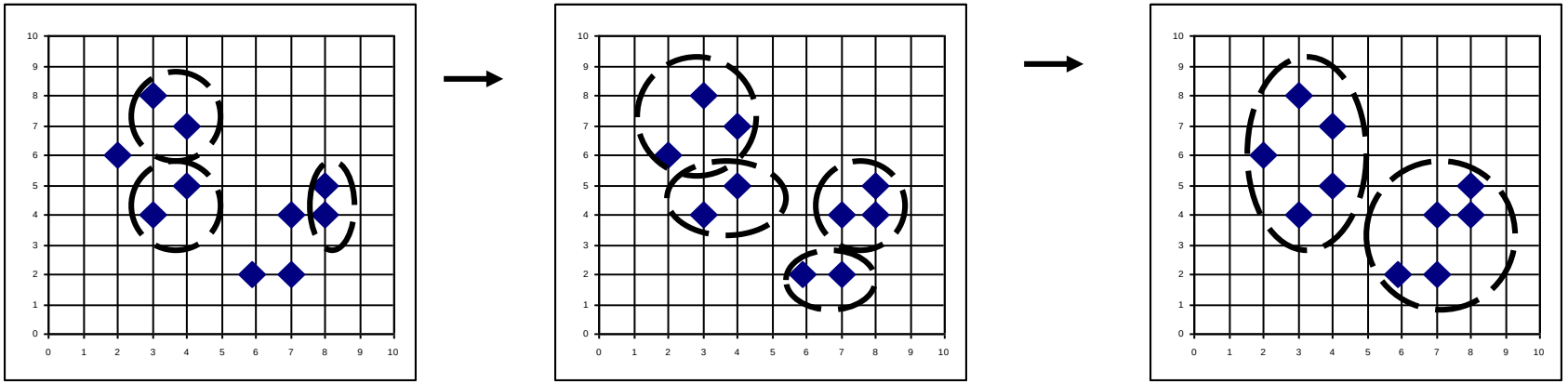
Hierarchical Clustering

- Two main types of hierarchical clustering
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or *k* clusters) left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are *k* clusters)

- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

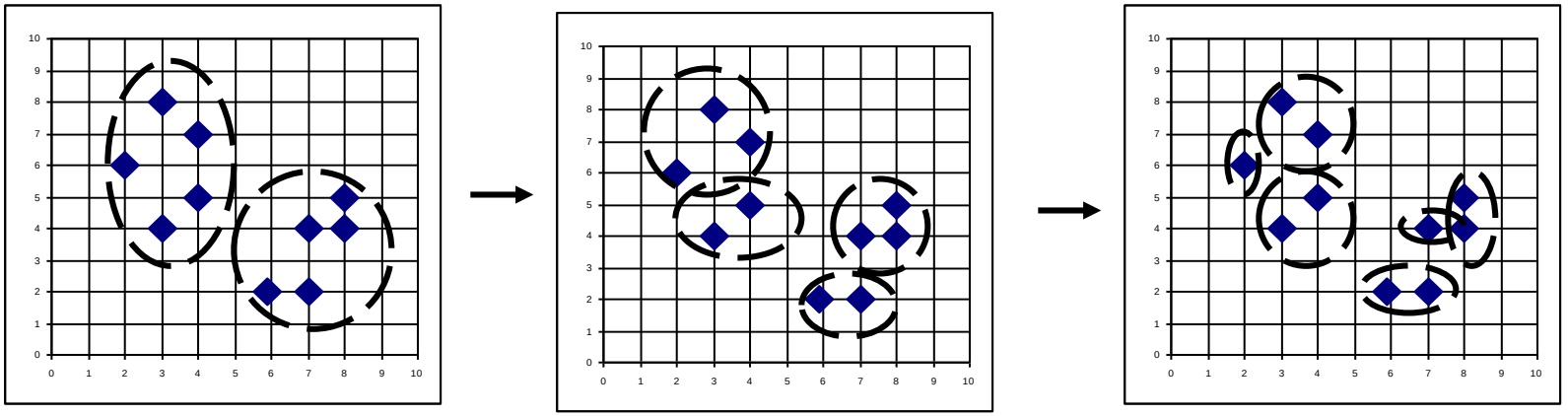
AGNES (Agglomerative Nesting)

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



Big data clustering

Large Clustering Problems

- Problems
  - Many examples
  - Many clusters
  - Many dimensions
- Example Domains
  - Text
  - Images
  - Protein structure

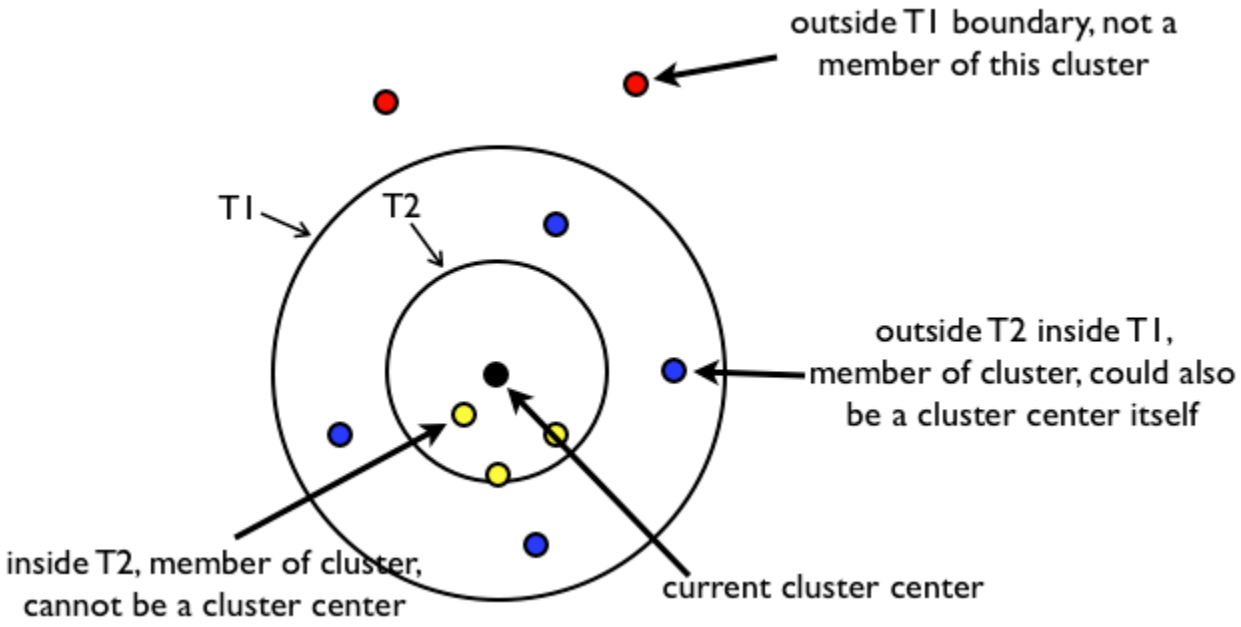
Canopies Approach
CLARA
MapReduce Approach

The Canopies Approach

- Two distance metrics: cheap & expensive
- First Pass
  - very inexpensive distance metric
  - create overlapping canopies
- Second Pass
  - expensive, accurate distance metric
  - canopies determine which distances calculated
- Implemented in Apache Mahout

Two Thresholds

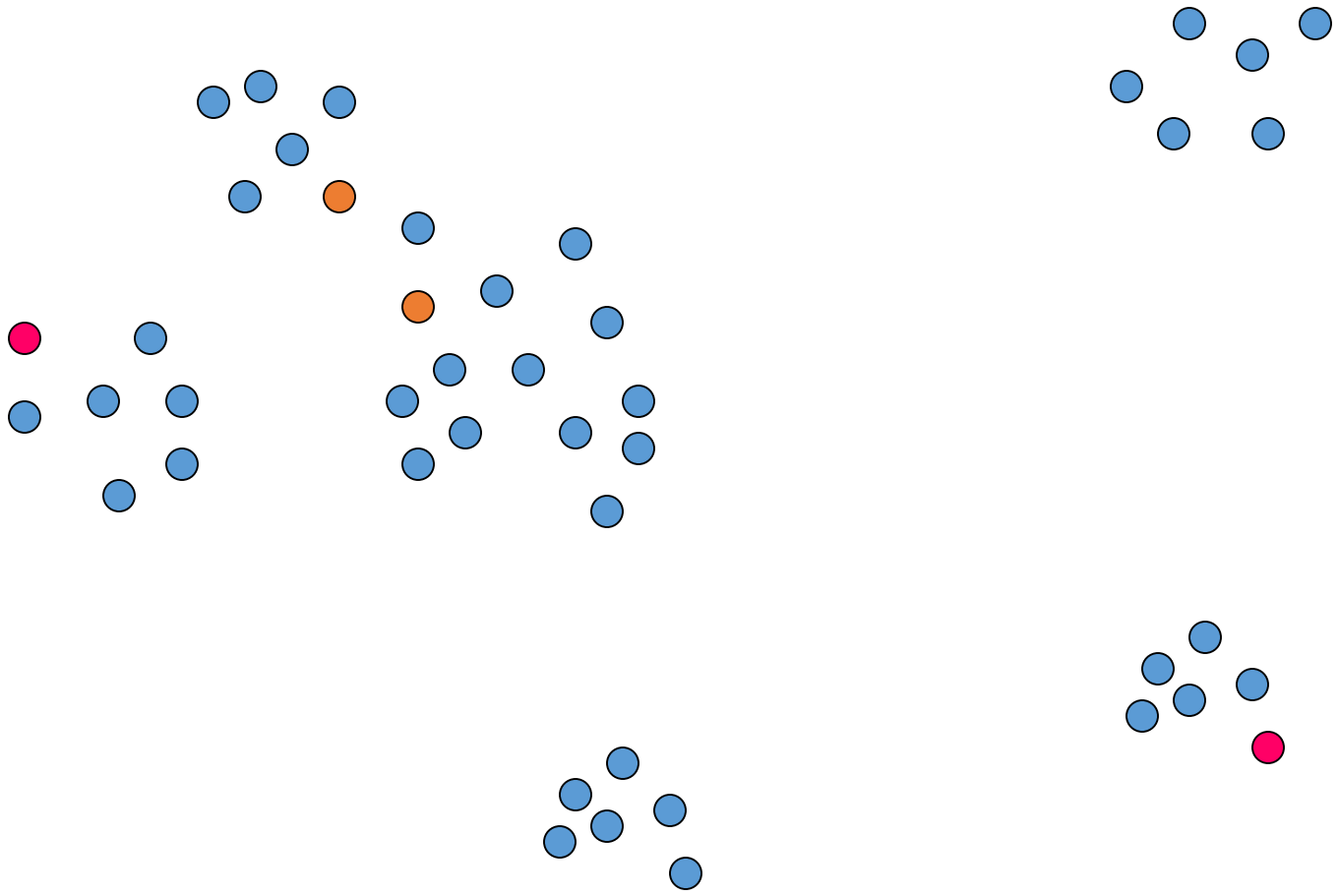
- Put all points in D
- Loop:
  - Pick a point *X* from D
  - Put points within **T2** of X in canopy
  - Remove points within **T1** of X from D



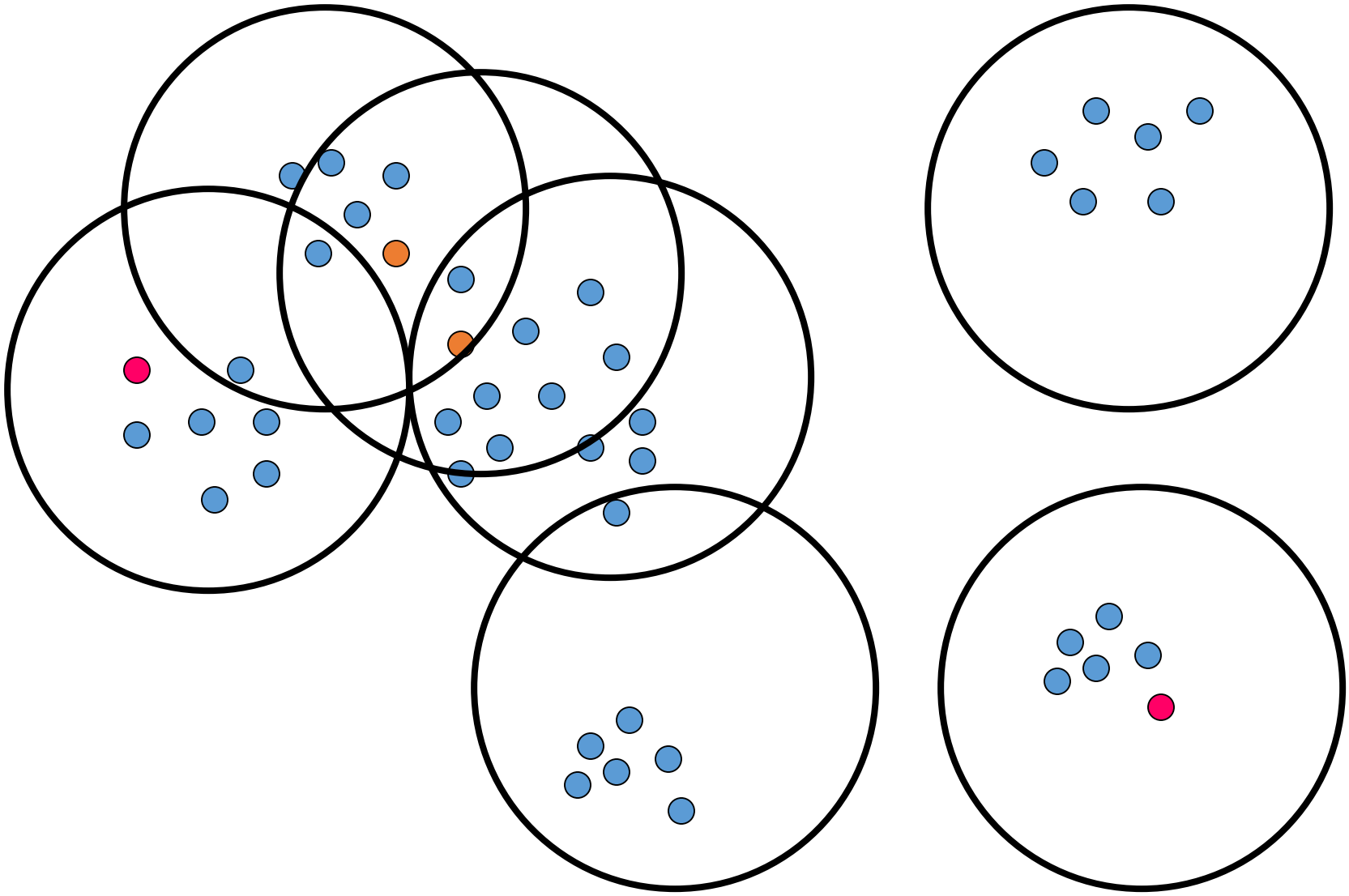
Creating Canopies

1. while **D** is not empty
2. select element *d* from **D** to initialize canopy *c*
3. remove *d* from **D**
4. Loop through remaining elements (d_i) in **D**
5. if distance between d_i and *c* < T1 : add element to the canopy *c*
6. if distance between d_i and *c* < T2 : remove element from **D**
7. end
8. add canopy *c* to the list of canopies **C**
9. end

Illustrating Canopies



Overlapping Canopies



Computational Savings

- inexpensive metric << expensive metric
  - Canopy creation nearly for free
- Number of canopies: c (large)
- Number of canopies per point: f (small but > 1)

- fn/c points per canopy (if evenly spread)
- $O(c(fn/c)^2)$ distance calculations initially

- Complexity reduction: $O(f^2/c)$

Canopies Approach

CLARA

MapReduce Approach

CLARA

- **Cl**ustering **Lar**ge **A**pplications
  - Draws multiple samples of the data set, applies PAM on each sample, and gives the best clustering as the output
- Strength:
  - Deals with larger data sets than PAM
- Weakness:
  - Efficiency depends on the sample size
- A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

Canopies Approach
CLARA
MapReduce Approach

Map

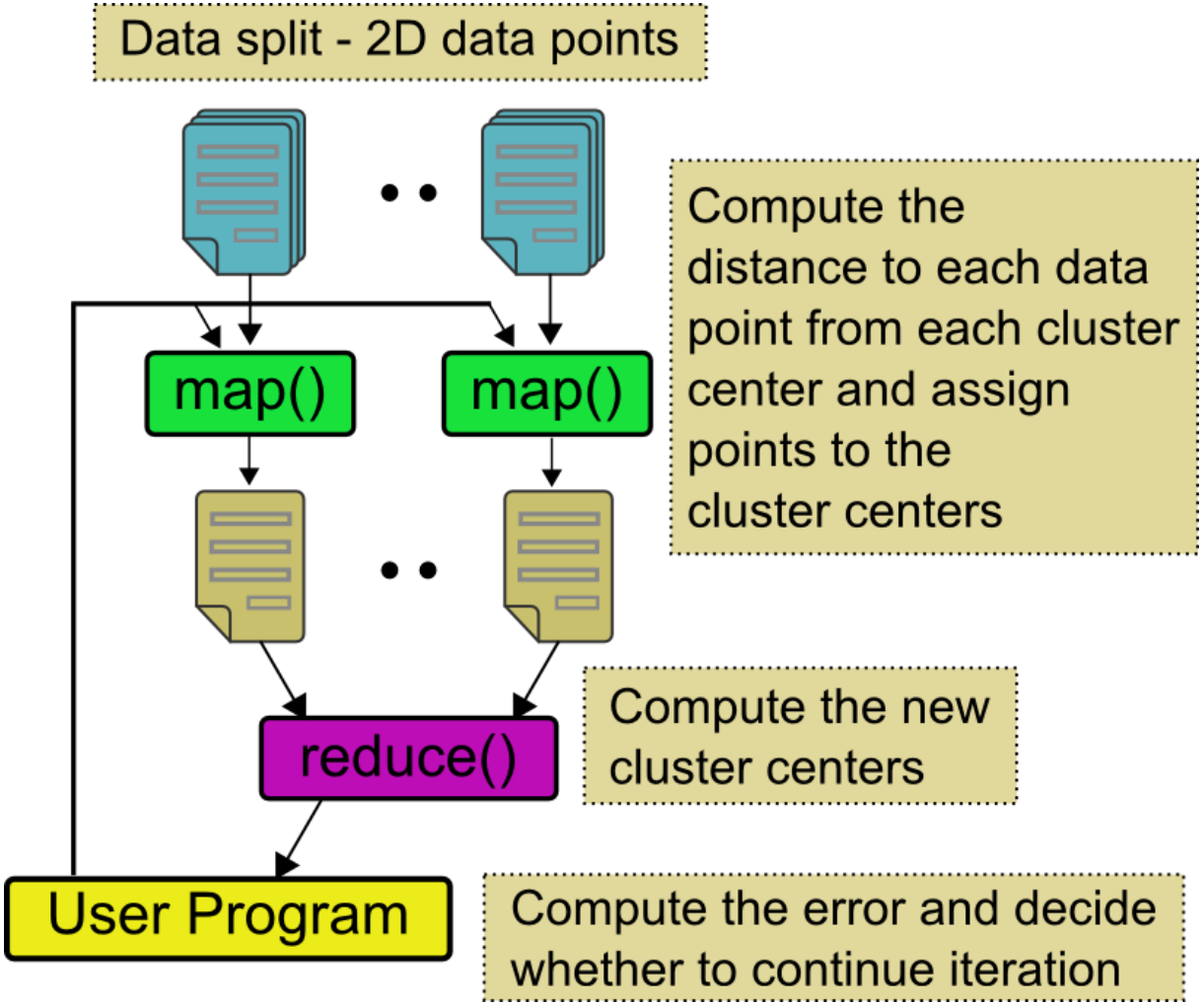
- Steps
  1. Each map task read in a shared file containing all the cluster centroids. The cluster centroids file can be typically stored in a Hadoop distributed cache.
  2. Each map task read in a small slice of input data points. For each data point, it assign it to the cluster whose centroid is closest to the current data.
  3. Emit <*centroid*, *data point*>

- These data items do not change over the iterations, and it is loaded once for the entire set of iterations.
- The variable data is the current cluster centers calculated during the previous iteration and hence used as the input value for the map function.

Reduce

- A reduce function computes the average of the partial cluster centers and produce the new cluster centers for the next step.
- Termination check:
  - Multiple MapReduce jobs are chained together to simulate the multiple iterations of the algorithm.
  - Main program, once it gets these new cluster centers, calculates the difference between the new cluster centers and the previous cluster centers
  - And determines if it needs to execute another cycle of MapReduce computation.

Overview


Data split - 2D data points
map()
map()
Compute the distance to each data point from each cluster center and assign points to the cluster centers
reduce()
Compute the new cluster centers
User Program
Compute the error and decide whether to continue iteration

End of Chapter 13