Clustering

Machine Learning

Hamid R Rabiee – Zahra Dehghanian

Spring 2025

Sharif University
of Technology

Unsupervised learning

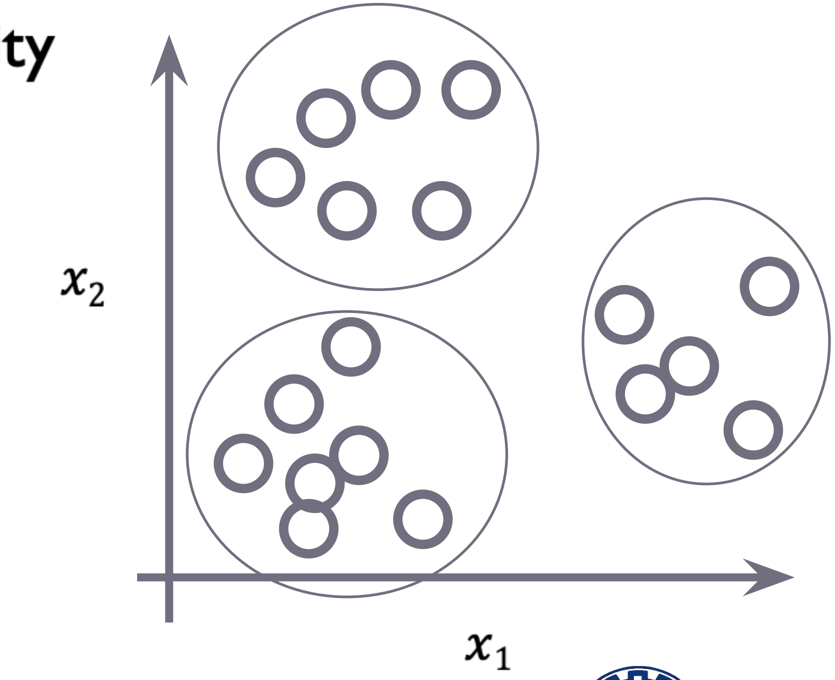
- **Clustering**: partitioning of data into groups of similar data points.

- **Density estimation**
  - Parametric & non-parametric density estimation

- **Dimensionality reduction**: data representation using a smaller number of dimensions while preserving (perhaps approximately) some properties of the data.

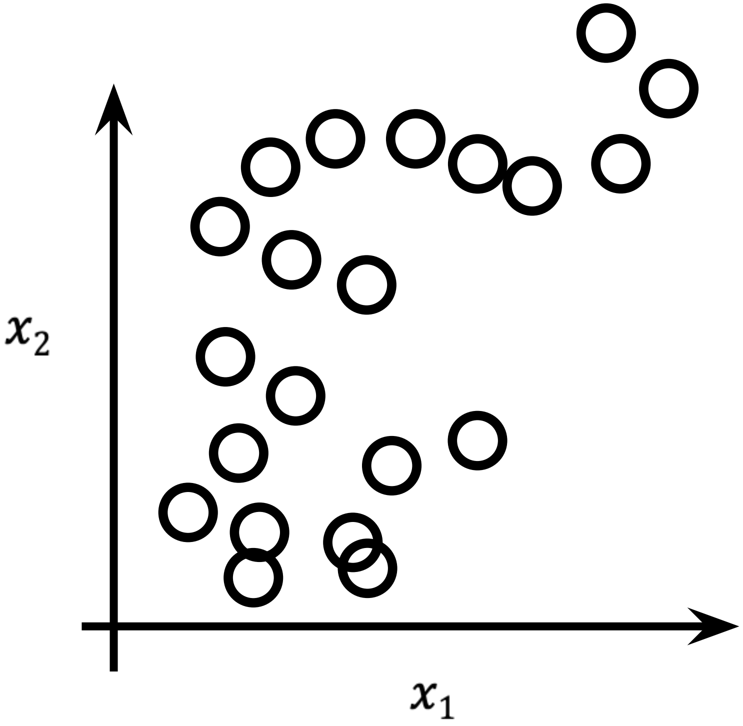
Clustering: Definition

- We have a set of unlabeled data points $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ and we intend to **find groups of similar objects** (based on the observed features)
  - high intra-cluster similarity
  - low inter-cluster similarity



Clustering: Another Definition

- Density-based definition:
  - Clusters are regions of high density that are separated from one another by regions of low density



Difficulties

- Clustering is not as well-defined as classification
- Clustering is subjective
    - Natural grouping may be ambiguous

Clustering Purpose

- **Preprocessing stage** to index, compress, or reduce the data

- Representing high-dimensional data in a low-dimensional space (e.g., for visualization purposes).

- Knowledge discovery from data: As a tool to **understand the hidden structure** in data or to **group** them
  - To gain insight into the structure of the data (prior to classifier design)
  - Provides information about the internal structure of the data

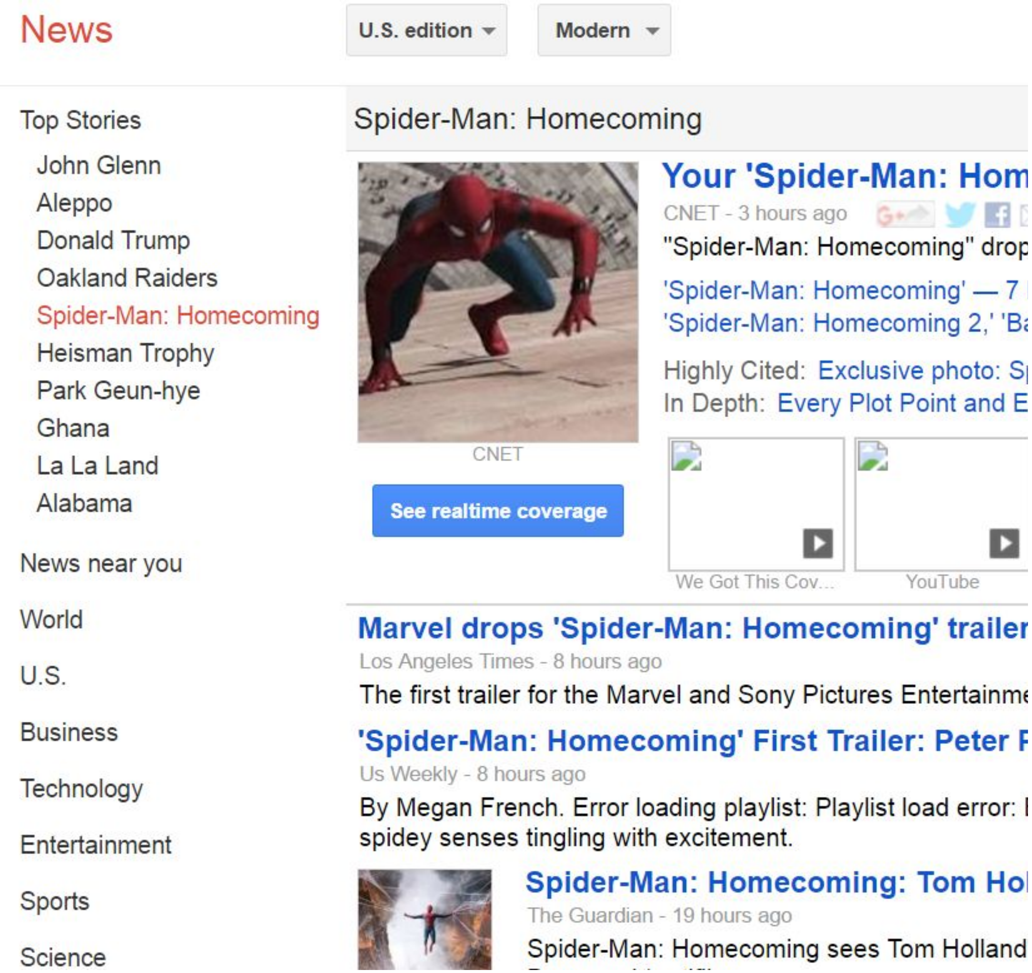
- To group or partition the data when no label is available

Clustering Applications

- Information retrieval (search and browsing)
  - Cluster text docs or images based on their content
  - Cluster groups of users based on their access patterns on webpages

Clustering of docs

- Google news



Spider-Man: Homecoming

Your 'Spider-Man: Hom

CNET - 3 hours ago

"Spider-Man: Homecoming" drop

'Spider-Man: Homecoming' — 7

'Spider-Man: Homecoming 2,' 'Ba

Highly Cited: Exclusive photo: Sp

In Depth: Every Plot Point and E

Marvel drops 'Spider-Man: Homecoming' trailer

Los Angeles Times - 8 hours ago

The first trailer for the Marvel and Sony Pictures Entertainme

'Spider-Man: Homecoming' First Trailer: Peter P

Us Weekly - 8 hours ago

By Megan French. Error loading playlist: Playlist load error: I spidey senses tingling with excitement.

Spider-Man: Homecoming: Tom Hol

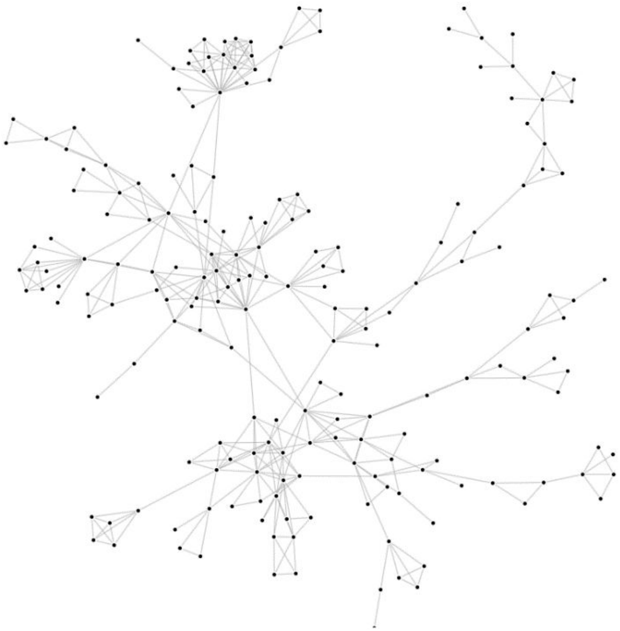
The Guardian - 19 hours ago

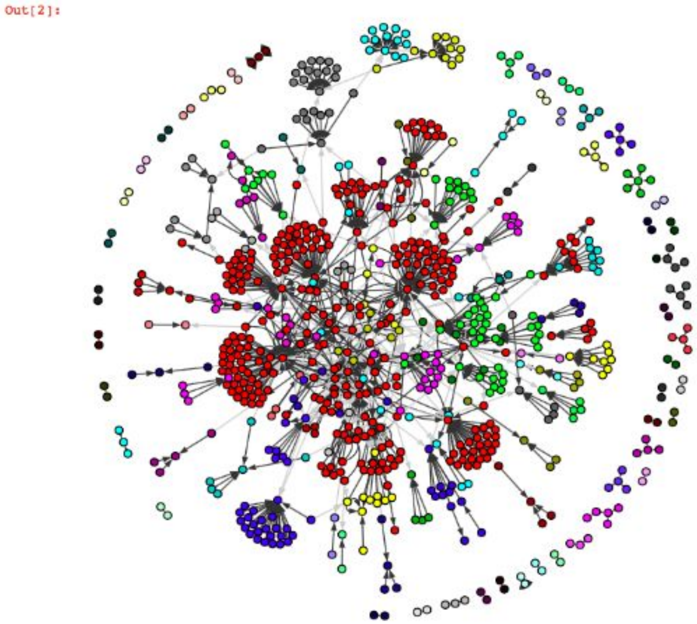
Spider-Man: Homecoming sees Tom Holland

Clustering Applications

- Information retrieval (search and browsing)
  - Cluster text docs or images based on their content
  - Cluster groups of users based on their access patterns on webpages
- **Cluster users of social networks** by interest (community detection).

Social Network: Community Detection





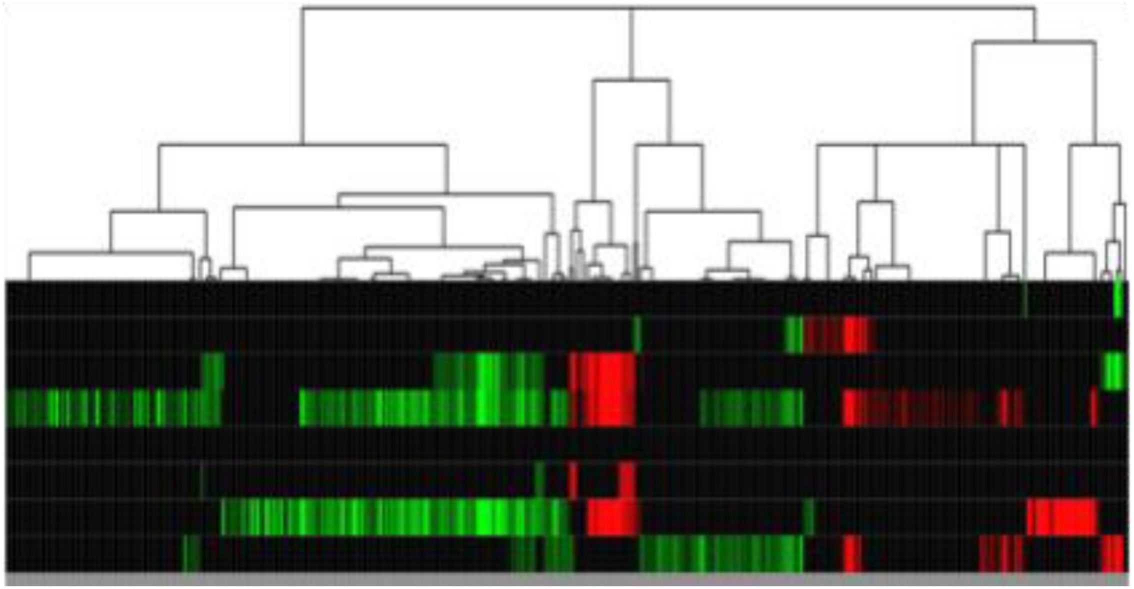
Out[2]:

Clustering Applications

- Information retrieval (search and browsing)
  - Cluster text docs or images based on their content
  - Cluster groups of users based on their access patterns on webpages
- Cluster users of social networks by interest (community detection).
- **Bioinformatics**
  - cluster similar proteins together (similarity w.r.t. chemical structure and/or functionality etc)
  - or cluster similar genes according to microarray data

Gene clustering

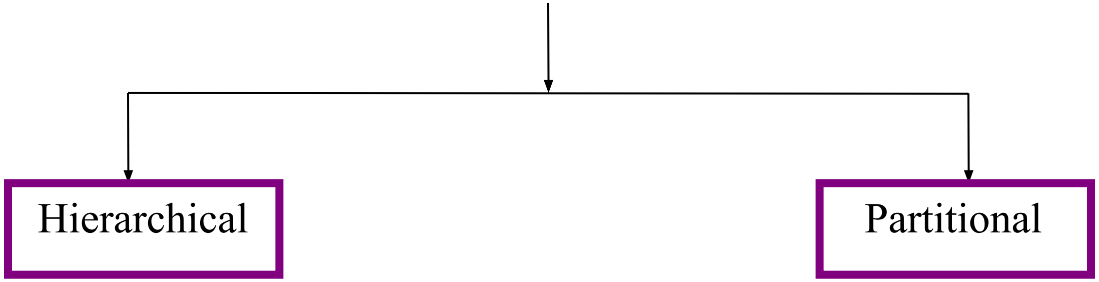
- Microarrays measures the expression of all genes
- Clustering genes can help to determine new functions for unknown genes by grouping genes



Clustering Applications

- Information retrieval (search and browsing)
  - Cluster text docs or images based on their content
  - Cluster groups of users based on their access patterns on webpages
- Cluster users of social networks by interest (community detection).
- Bioinformatics
  - Cluster similar proteins together (similarity wrt chemical structure and/or functionality etc) or similar genes according to microarray data
- **Market segmentation**
  - Clustering customers based on the their purchase history and their characteristics
- Image segmentation
- Many more applications

Categorization of Clustering Algorithms



Partitional algorithms: Construct various partitions and then evaluate them by some criterion
the desired number of clusters K must be specified.

Hierarchical algorithms: Create a hierarchical decomposition of the set of objects using some criterion

Clustering methods we will discuss

- Objective based clustering
  - K-means
  - EM-style algorithm for clustering for mixture of Gaussians (in the next lecture)

- Hierarchical clustering

Partitional Clustering

- $\mathcal{X} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$
- $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\}$
  - $\forall j, \mathcal{C}_j \neq \emptyset$
  - $\bigcup_{j=1}^{K} \mathcal{C}_j = \mathcal{X}$
  - $\forall i, j,\ \mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ (disjoint partitioning for hard clustering)

Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.

Hard clustering: Each data can belong to one cluster only

- Since the output is only one set of clusters the user has to specify the desired number of clusters K.

Partitioning Algorithms: Basic Concept

- Construct a partition of a set of N objects into a set of K clusters
  - The number of clusters K is given in advance
  - Each object belongs to **exactly one** cluster in hard clustering methods

- K-means is the most popular partitioning algorithm

Objective Based Clustering

- **Input:** A set of N points, also a distance/dissimilarity measure
- **Output:** a partition of the data.

- **k-median:** find centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K$ to minimize

$$\sum_{i=1}^{N} \min_{j \in 1,\ldots,K} d(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j)$$

- **k-means:** find centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K$ to minimize

$$\sum_{i=1}^{N} \min_{j \in 1,\ldots,K} d^2(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j)$$

Distance Measure

- Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $d(O_1, O_2)$

- Specifying the distance $d(\boldsymbol{x}, \boldsymbol{x}')$ between pairs $(\boldsymbol{x}, \boldsymbol{x}')$.
  - E.g., for texts: # keywords in common, edit distance
  - Example: Euclidean distance in the space of features

K-means Clustering

- **Input**: a set $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ of data points (in a d-dim feature space) and an integer K

- **Output**: a set of K representatives $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K \in \mathbb{R}^d$ as the cluster representatives
  - data points are assigned to the clusters according to their distances to $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$
    - Each data is assigned to the cluster whose representative is nearest to it

- **Objective**: choose $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$ to minimize:

$$\sum_{i=1}^{N} \min_{j \in 1,\dots,K} d^2(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j)$$

Euclidean k-means Clustering

- **Input**: a set $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ of data points (in a d-dim feature space) and an integer K

- **Output**: a set of K representatives $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K \in \mathbb{R}^d$ as the cluster representatives
  - data points are assigned to the clusters according to their distances to $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$
    - Each data is assigned to the cluster whose representative is nearest to it

- **Objective**: choose $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$ to minimize:

$$\sum_{i=1}^{N} \min_{j \in 1,\dots,K} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c}_j \right\|^2$$

each point assigned to its closest cluster representative

Euclidean k-means Clustering: Computational Complexity

- • To find the optimal partition, we need to exhaustively enumerate all partitions
  - In how many ways can we assign k labels to N observations?
- NP hard: even for $k = 2$ or $d = 2$
- For k=1: $\min_{\boldsymbol{c}} \sum_{i=1}^{N} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c} \right\|^2$
  - $\boldsymbol{c} = \boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}^{(i)}$
- For $d = 1$, dynamic programming in time $O(N^2 K)$.

Common Heuristic in Practice: The Lloyd's method

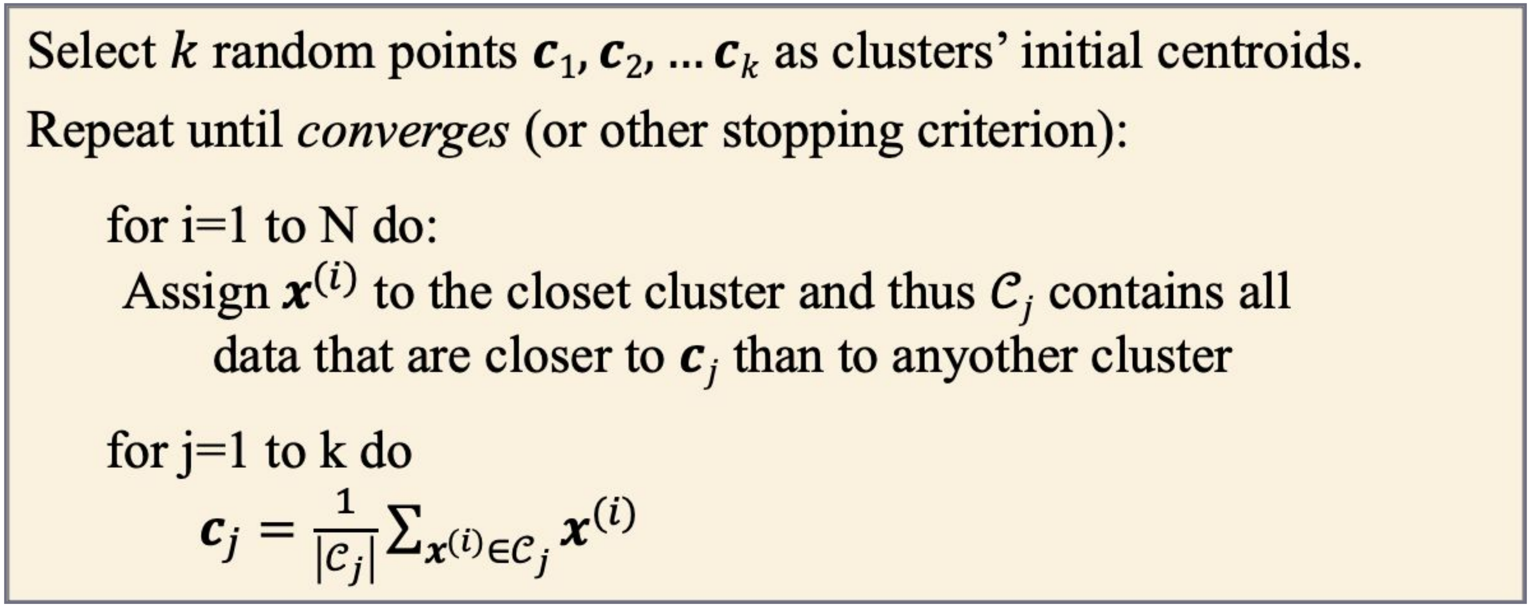
- Input: A set $\mathcal{X}$ of N datapoints $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ in $\mathbb{R}^d$

- **Initialize** centers $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K \in \mathbb{R}^d$ in any way.
- **Repeat** until there is no further change in the cost.
  - For each j: $\mathcal{C}_j \leftarrow \{\boldsymbol{x} \in \mathcal{X} | \text{where } \boldsymbol{c}_j \text{ is the closest center to } \boldsymbol{x}\}$
  - For each j: $\boldsymbol{c}_j \leftarrow$mean of members of $\mathcal{C}_j$

Holding centers $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$ fixed
Find optimal assignments $\mathcal{C}_1, \dots, \mathcal{C}_K$ of data points to clusters

Holding cluster assignments $\mathcal{C}_1, \dots, \mathcal{C}_K$ fixed
Find optimal centers $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$

K-means Algorithm (The Lloyd's method)



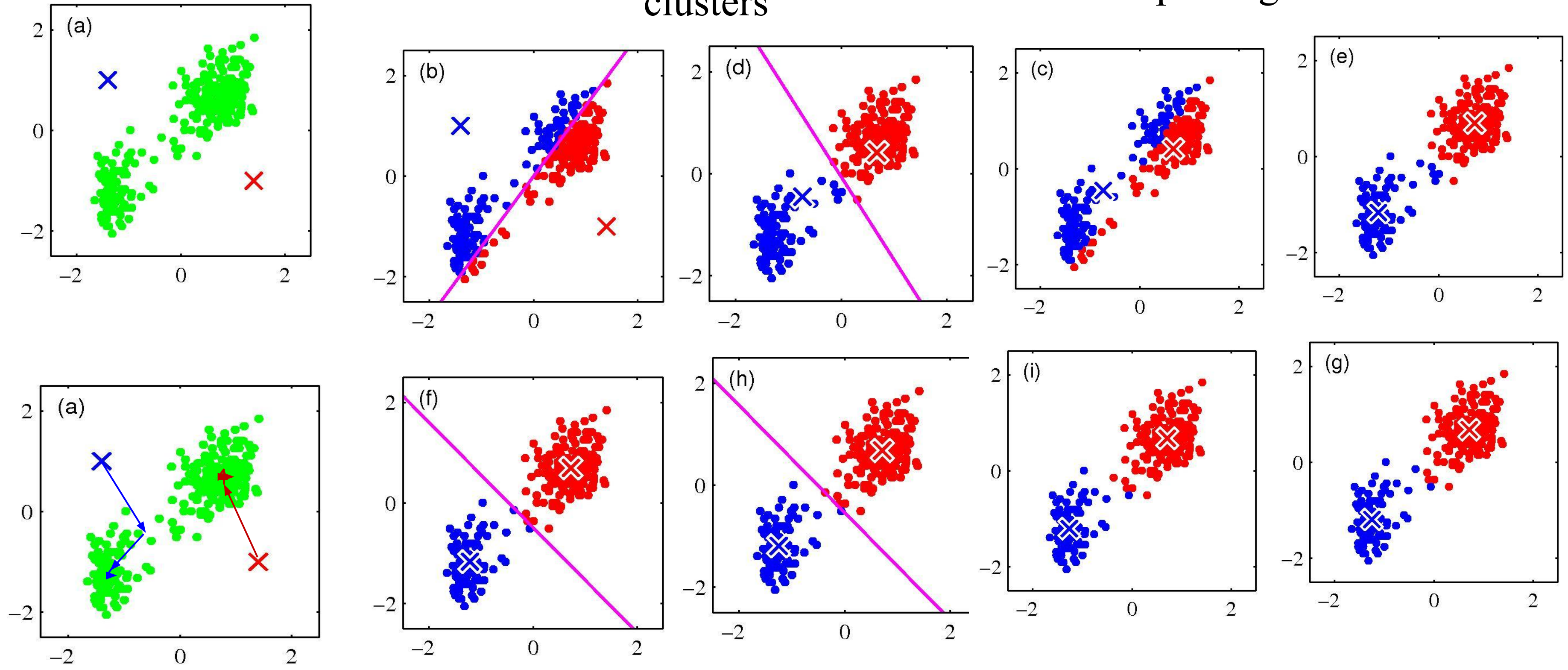
```
Select $k$ random points $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots \boldsymbol{c}_k$ as clusters' initial centroids.
Repeat until *converges* (or other stopping criterion):
    for i=1 to N do:
        Assign $\boldsymbol{x}^{(i)}$ to the closet cluster and thus $\mathcal{C}_j$ contains all
            data that are closer to $\boldsymbol{c}_j$ than to anyother cluster
    for j=1 to k do
        $\boldsymbol{c}_j = \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \boldsymbol{x}^{(i)}$
```

Assign data based on current centers

Re-estimate centers based on current assignment

Assigning data to clusters

Updating means



Intra-cluster similarity view

- k-means optimizes intra-cluster similarity:

$$J(\mathcal{C}) = \sum_{j=1}^{K} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c}_j \right\|^2$$

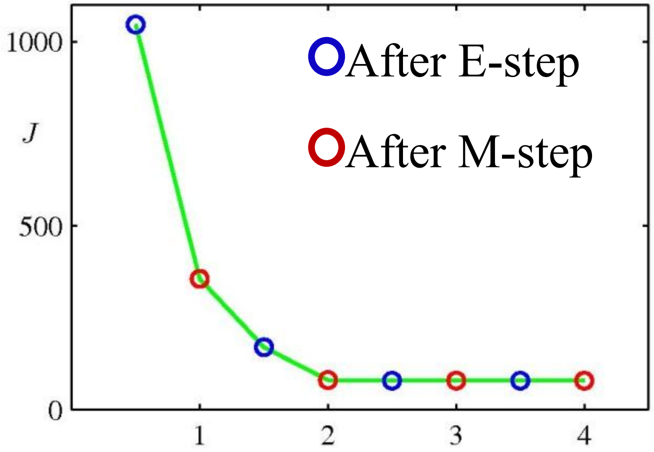
$$\boldsymbol{c}_j = \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \boldsymbol{x}^{(i)}$$

$$\sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c}_j \right\|^2 = \frac{1}{2|\mathcal{C}_j|} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \sum_{\boldsymbol{x}^{(i')} \in \mathcal{C}_j} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i')} \right\|^2$$

the average distance to members of the same cluster

K-means: Convergence

- It always converges.
- Why should the *K*-means algorithm ever reach a state in which clustering doesn't change.
  - Reassignment stage monotonically decreases *J* since each vector is assigned to the closest centroid.
  - Centroid update stage also for each cluster minimizes the sum of squared distances of the assigned points to the cluster from its center.



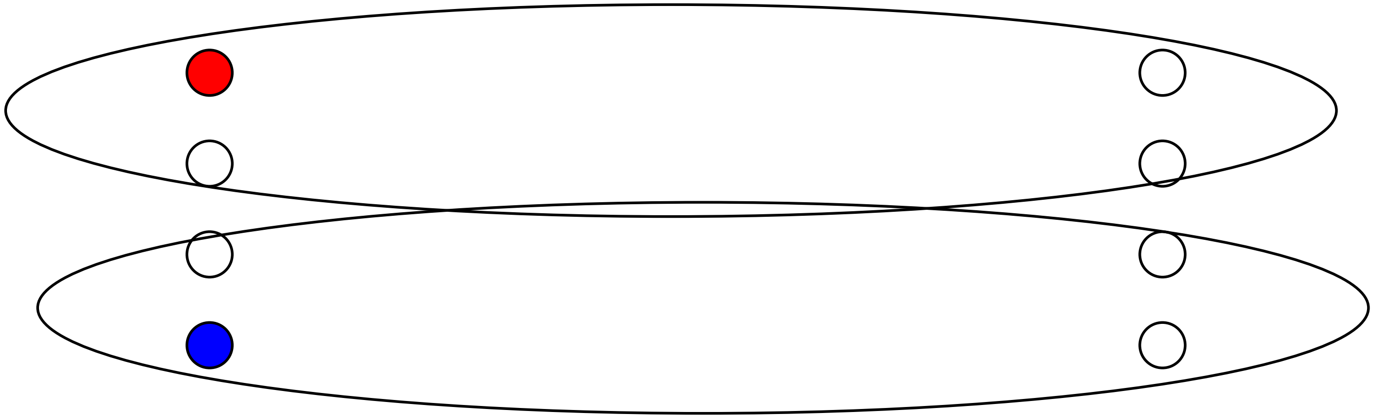
[Bishop]

Local optimum

- It always converges
- but it may converge at a local optimum that is different from the global optimum
  - may be arbitrarily worse in terms of the objective score.

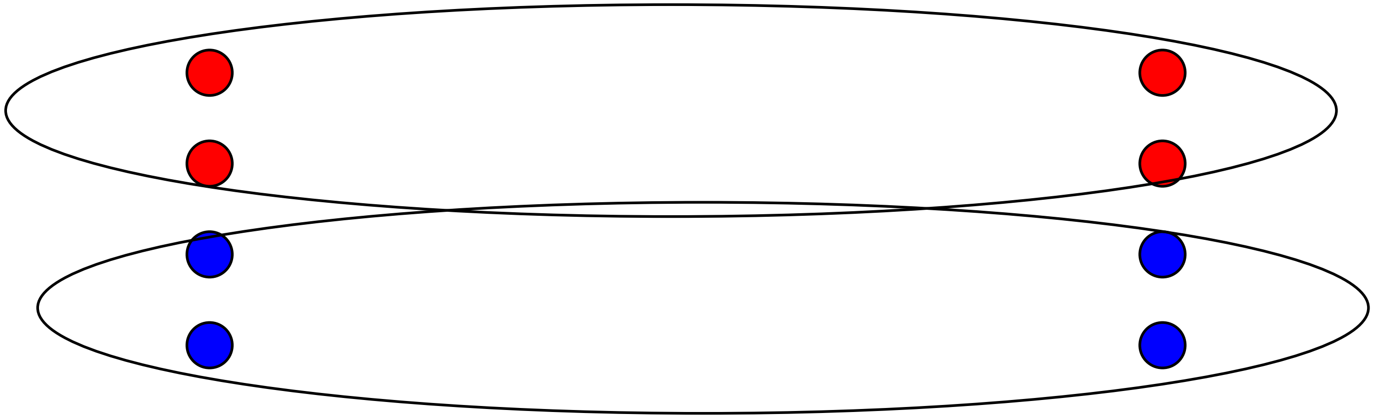
Local optimum

- It always converges
- but it may converge at a local optimum that is different from the global optimum
  - may be arbitrarily worse in terms of the objective score.



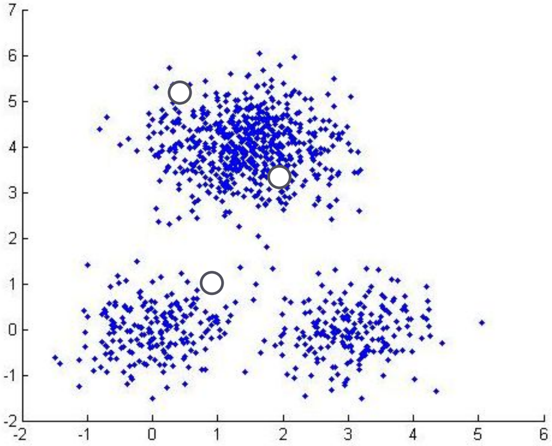
Local optimum

- It always converges
- but it may converge at a local optimum that is different from the global optimum
  - may be arbitrarily worse in terms of the objective score.

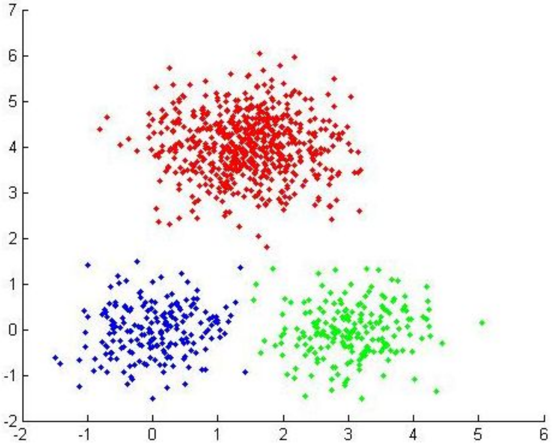


Local optimum: every point is assigned to its nearest center and every center is the mean value of its points.

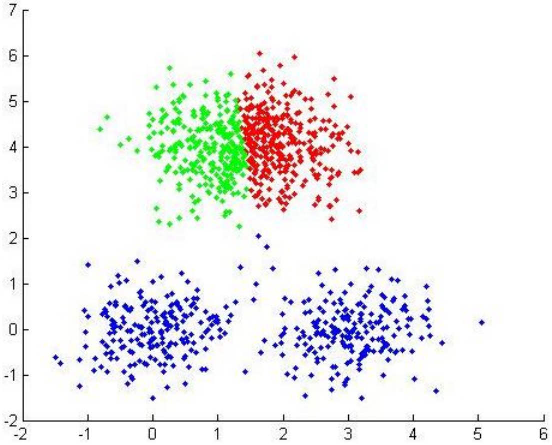
K-means: Local Minimum Problem



Original Data



Optimal Clustering



The obtained Clustering

The Lloyd’s method: Initialization

- Initialization is crucial (how fast it converges, quality of clustering)
  - Random centers from the data points
    - Multiple runs and select the best ones
  - Initialize with the results of another method
  - Select good initial centers using a heuristic
    - Furthest traversal
    - K-means ++ (works well and has provable gaurantees)

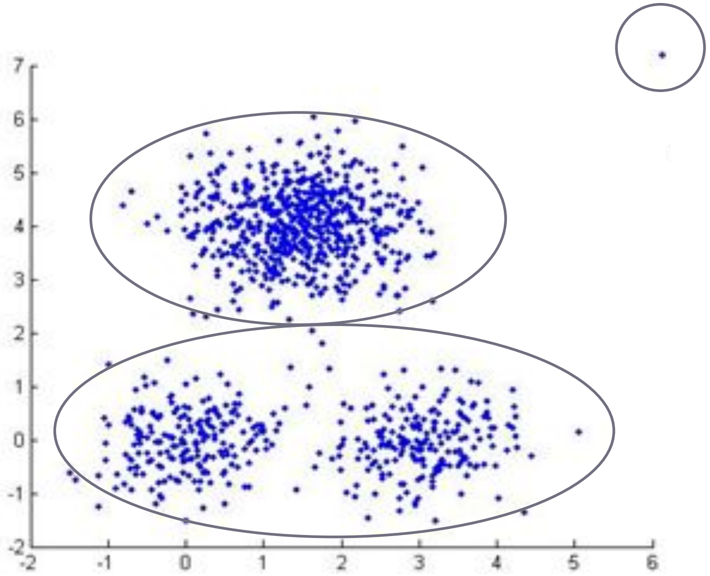
Another Initialization Idea: Furthest Point Heuristic

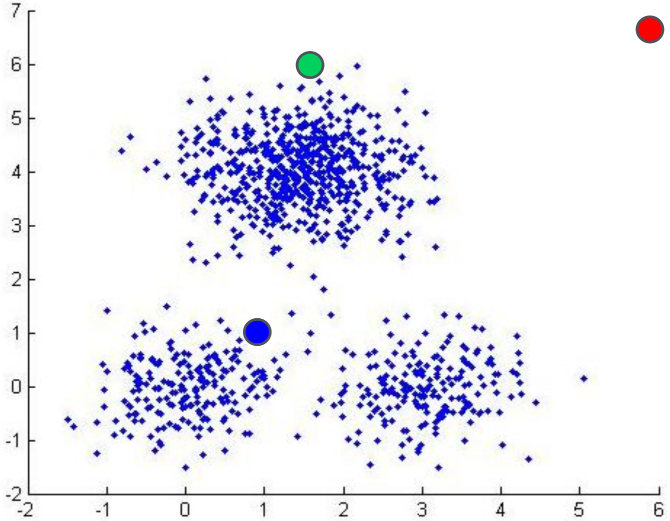
-

- Choose $\boldsymbol{c}_1$ arbitrarily (or at random).
- For $j = 2, \dots, K$
  - Select $\boldsymbol{c}_j$ among datapoints $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ that is farthest from previously chosen $\boldsymbol{c}_1, \dots, \boldsymbol{c}_{j-1}$

Another Initialization Idea: Furthest Point Heuristic

- It is sensitive to outliers





K-means++ Initialization: D2 sampling
[D. Arthur and S. Vassilvitskii, 2007]

- Combine random initialization and furthest point initialization ideas

- Let the probability of selection of the point be proportional to the distance between this point and its nearest center.
  - probability of selecting of $\boldsymbol{x}$ is proportional to $D^2(\boldsymbol{x}) = \min_{k<j} \|\boldsymbol{x} - \boldsymbol{c}_k\|^2$.

- Choose $\boldsymbol{c}_1$ arbitrarily (or at random).

- For $j = 2, \dots, K$
  - Select $\boldsymbol{c}_j$ among data points $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ according to the distribution:

$$\Pr(\boldsymbol{c}_j = \boldsymbol{x}^{(i)}) \propto \min_{k<j} \left\|\boldsymbol{x}^{(i)} - \boldsymbol{c}_k\right\|^2$$

- **Theorem:** K-means++ always attains an $O(\log k)$ approximation to optimal k-means solution in expectation.

K-means Clustering: Cost Function

- Minimizes the within-cluster dispersion to the cluster centers:

$$J(\mathcal{C}) = \sum_{j=1}^{k} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}_j \right\|^2$$

$$\boldsymbol{\mu}_j = \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \boldsymbol{x}^{(i)}$$

K-median: $J(\mathcal{C}) = \sum_{j=1}^{k} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c}_j \right\|_1$

K-means Algorithm

1. Choose k centroids $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k\}$ at random

2. Initial partition data into k clusters by assigning them to the closest centroid

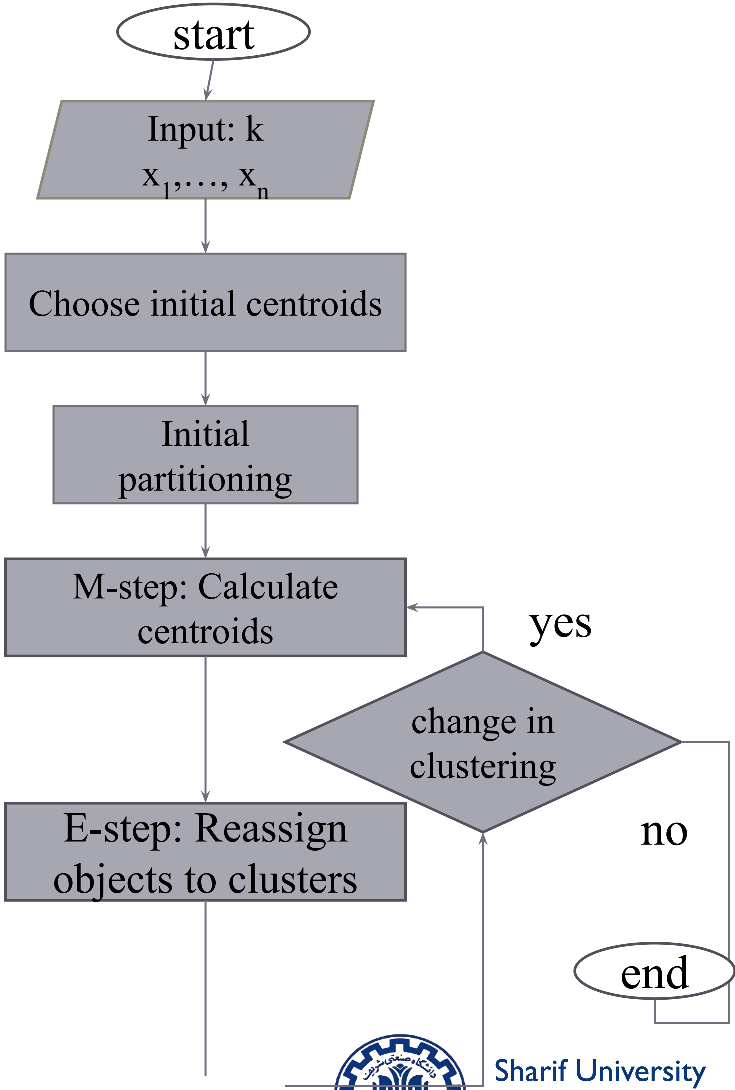
3. M-step: Calculate the centroid (mean) of each of the k clusters.

$$\boldsymbol{\mu}_j = \frac{1}{|\mathcal{C}_j|} \sum\nolimits_{\boldsymbol{x} \in \mathcal{C}_j} \boldsymbol{x}$$

4. E-step: Reassign data to the closest centroids.

$$\mathcal{C}_j = \{i \mid \forall k, \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_j\| < \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k\|\}$$

5. Repeat 3 and 4 until no reallocations occur



K-means: Termination Conditions

- Several possibilities, e.g.,
  - A fixed number of iterations is reached
  - Data partitioning is unchanged
  - Centroid positions don’t change
    - Does this mean that the docs in a cluster are unchanged?

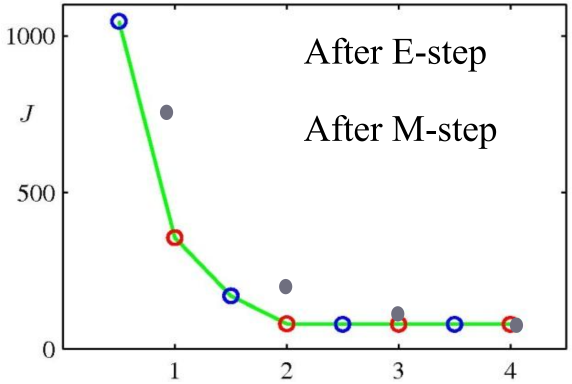
How Many Clusters?

- Number of clusters *k* is given in advance in the k-means algorithm
  - However, finding the "right" number of clusters is a part of the problem

- Tradeoff between having better focus within each cluster and having too many clusters

How Many Clusters?

- Heuristic:
  - Find large gap between $k-1$-means cost and k-means cost.
  - “knee finding” or “elbow finding”.



- Hold-out validation/cross-validation on auxiliary task (e.g., supervised learning task).

- Optimization problem: penalize having lots of clusters
  - some criteria can be used to automatically estimate k
    - Penalize the number of bits you need to describe the extra parameter

$$J'(\mathcal{C}) = J(\mathcal{C}) + |\mathcal{C}| \times \log N$$

- Hierarchical clustering

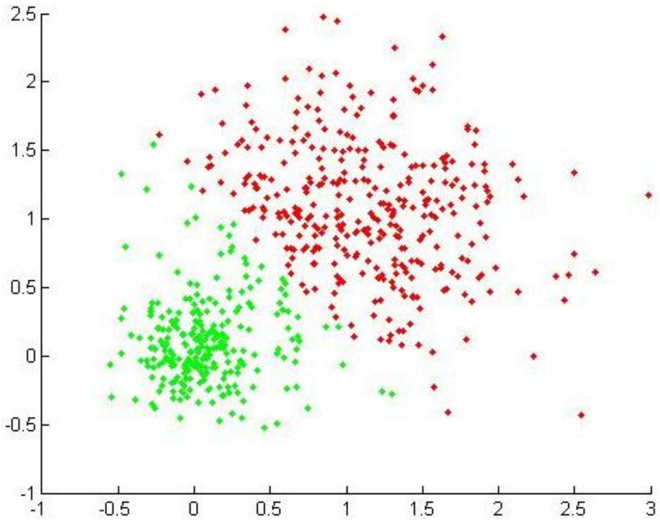
K-means: Advantages and disadvantages

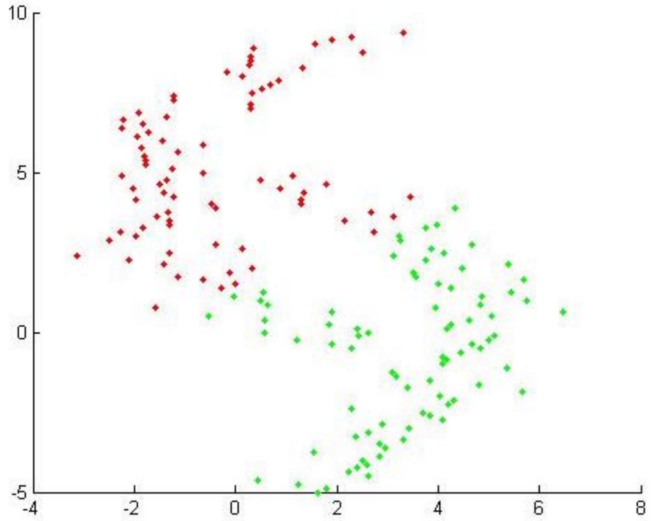
- Strength
  - It is a simple method and easy to implement.
  - Relatively efficient: $O(tKNd)$, where t is the number of iterations.
    - *K*-means typically converges quickly
      - Usually $t \ll n$.
    - Exponential # of rounds in the worst case [Andrea Vattani 2009].

- Weakness
  - Need to specify *K*, the *number* of clusters, in advance
  - Often terminates at a *local optimum*.
    - Initialization is important.
  - Not suitable to discover clusters with arbitrary shapes
  - Works for numerical data. What about categorical data?
  - Noise and outliers can be considerable trouble to *K*-means

k-means Algorithm: Limitation

- In general, k-means is unable to find clusters of arbitrary shapes, sizes, and densities
  - Except to very distant clusters





K-means

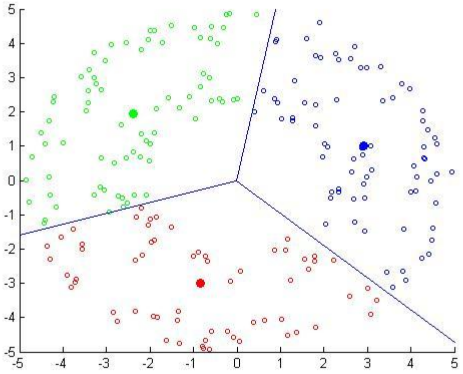
- K-means was proposed near 60 years ago
  - thousands of clustering algorithms have been published since then
  - However, K-means is still widely used.

- This speaks to the difficulty in designing a general purpose clustering algorithm and the ill-posed problem of clustering.

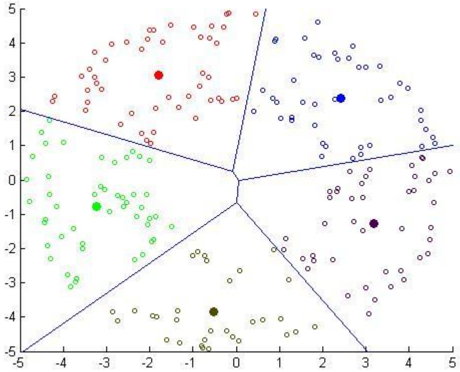
A.K. Jain, Data Clustering: 50 years beyond k-means,2010.

K-means: Vector Quantization

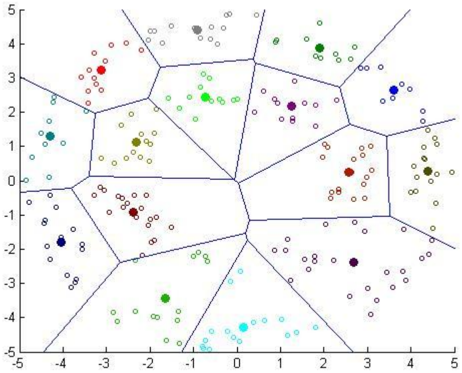
- Data Compression
  - Vector quantization: construct a codebook using k-means
    - cluster means as prototypes representing examples assigned to clusters.



$k = 3$

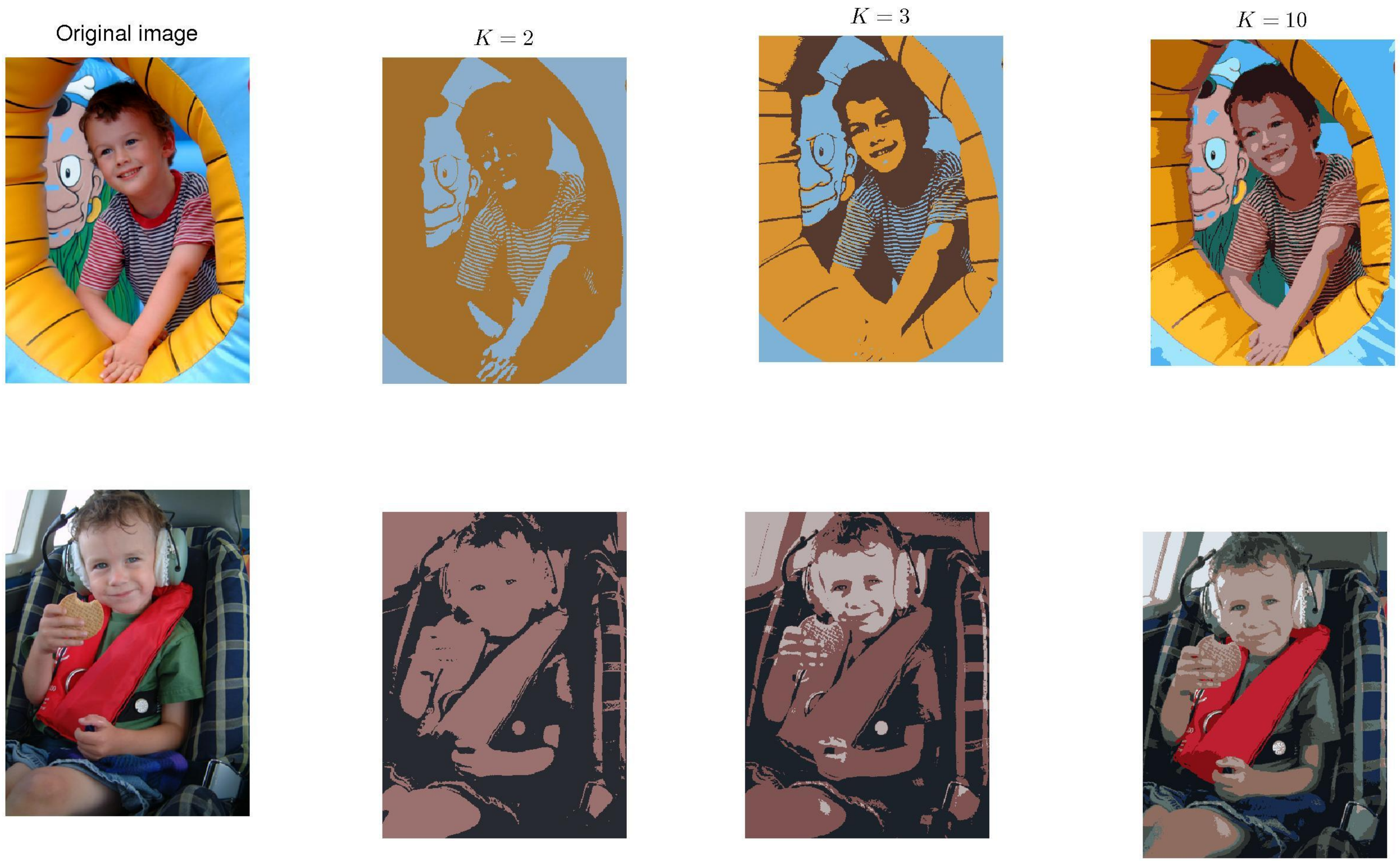


$k = 5$



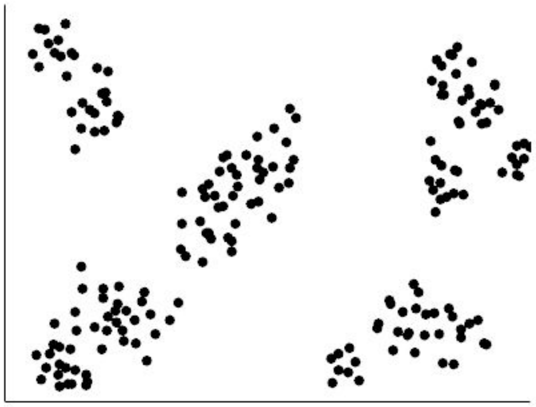
$k = 15$

K-means: Image Segmentation

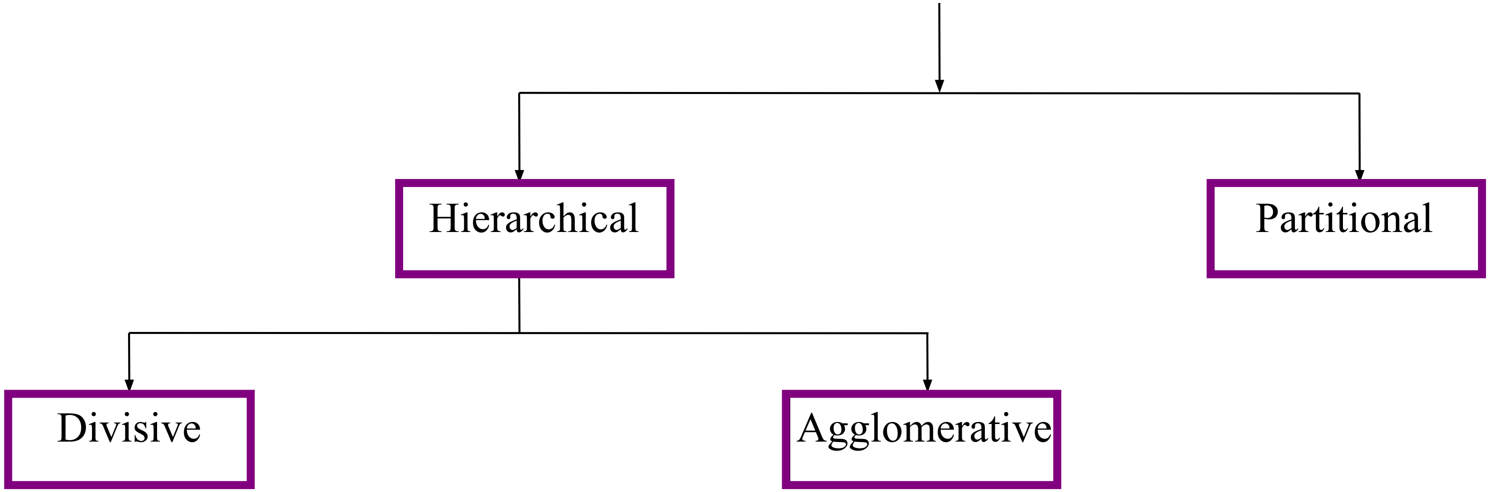


Hierarchical Clustering

- Notion of a cluster can be ambiguous?
- How many clusters?
- Hierarchical Clustering: Clusters contain sub-clusters and sub-clusters themselves can have sub-sub-clusters, and so on
  - Several levels of details in clustering
- A hierarchy might be more natural.
  - Different levels of granularity



Categorization of Clustering Algorithms



Hierarchical Clustering

- Agglomerative (bottom up):
  - Starts with each data in a separate cluster
  - Repeatedly joins the closest pair of clusters, until there is only one cluster (or other stopping criteria).

- Divisive (top down):
  - Starts with the whole data as a cluster
  - Repeatedly divide data in one of the clusters until there is only one data in each cluster (or other stopping criteria).

Hierarchical Agglomerative Clustering (HAC)

- Algorithm
  1. Maintain a set of clusters
  2. Initially, each instance forms a cluster
  3. While there are more than one cluster

     Pick the two closest one

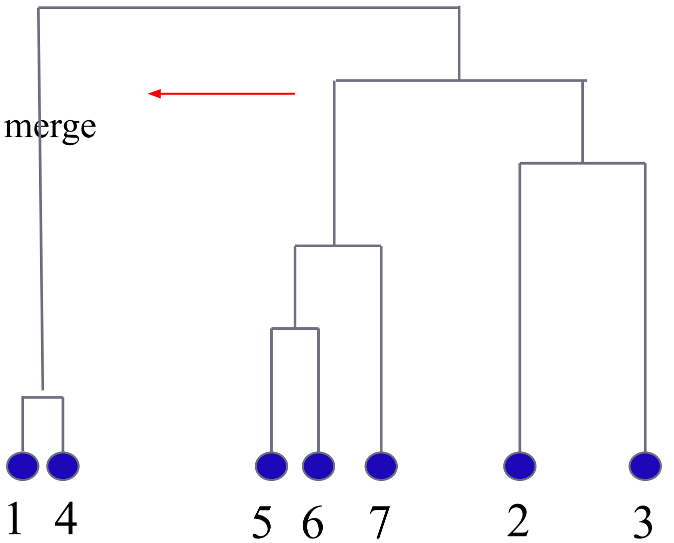
     Merge them into a new cluster

Hierarchical Agglomerative Clustering (HAC)

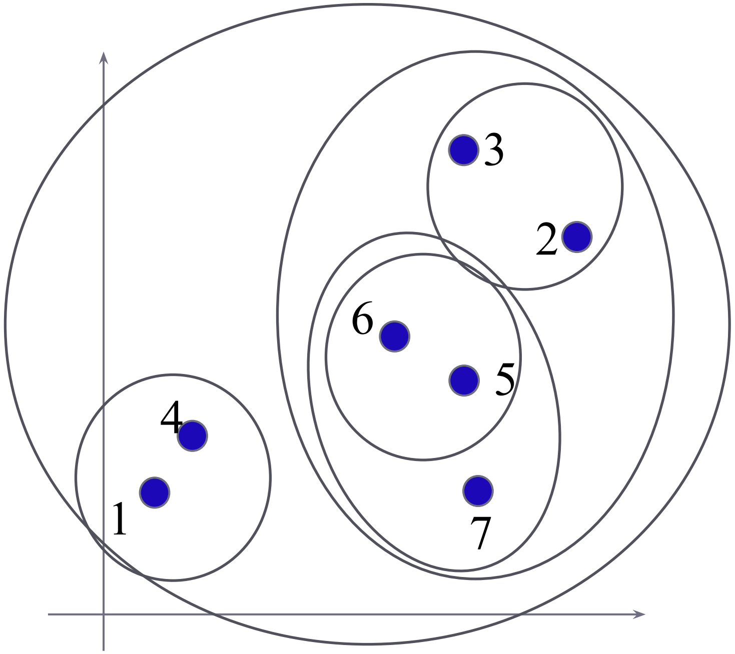
- Algorithm
  1. Maintain a set of clusters
  2. Initially, each instance forms a cluster
  3. While there are more than one cluster

     Pick the two closest one

     Merge them into a new cluster



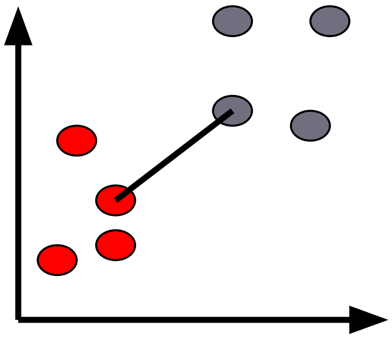
Height represents the distance at which the merge occurs



Distances between Cluster Pairs

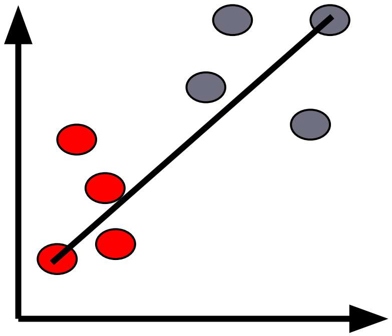
- Many variants to defining distances between pair of clusters
  - **Single-link**
    - Minimum distance between different pairs of data
  - **Complete-link**
    - Maximum distance between different pairs of data
  - **Centroid (Ward's)**
    - Distance between centroids (centers of gravity)
  - **Average-link**
    - Average distance between pairs of elements

Distances between Cluster Pairs



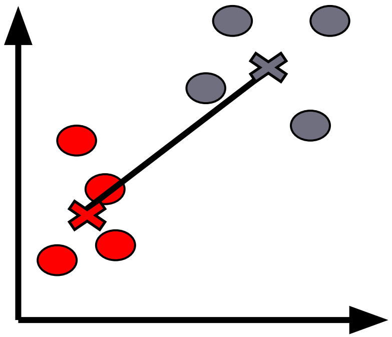
Single-link

$$dist_{SL}(\mathcal{C}_i,\mathcal{C}_j)=\min_{\boldsymbol{x}\in\mathcal{C}_i,\,\boldsymbol{x}'\in\mathcal{C}_j} dist(\boldsymbol{x},\boldsymbol{x}')$$



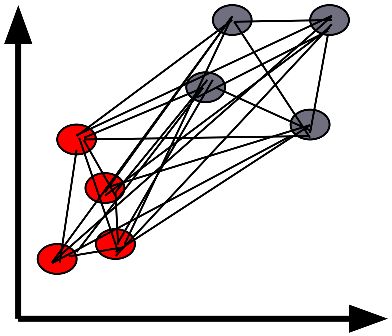
Complete-link

$$dist_{CL}(\mathcal{C}_i,\mathcal{C}_j)=\max_{\boldsymbol{x}\in\mathcal{C}_i,\,\boldsymbol{x}'\in\mathcal{C}_j} dist(\boldsymbol{x},\boldsymbol{x}')$$



Ward's

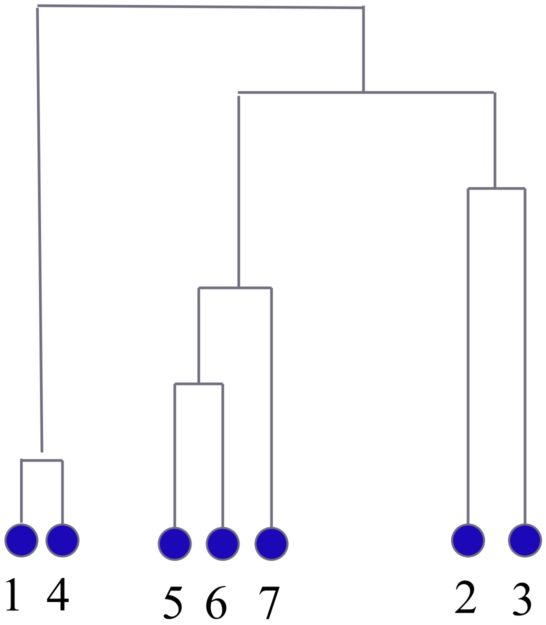
$$dist_{Ward}(\mathcal{C}_i,\mathcal{C}_j)=\frac{|\mathcal{C}_i||\mathcal{C}_j|}{|\mathcal{C}_i|+|\mathcal{C}_j|}dist(\boldsymbol{c}_i,\boldsymbol{c}_j)$$

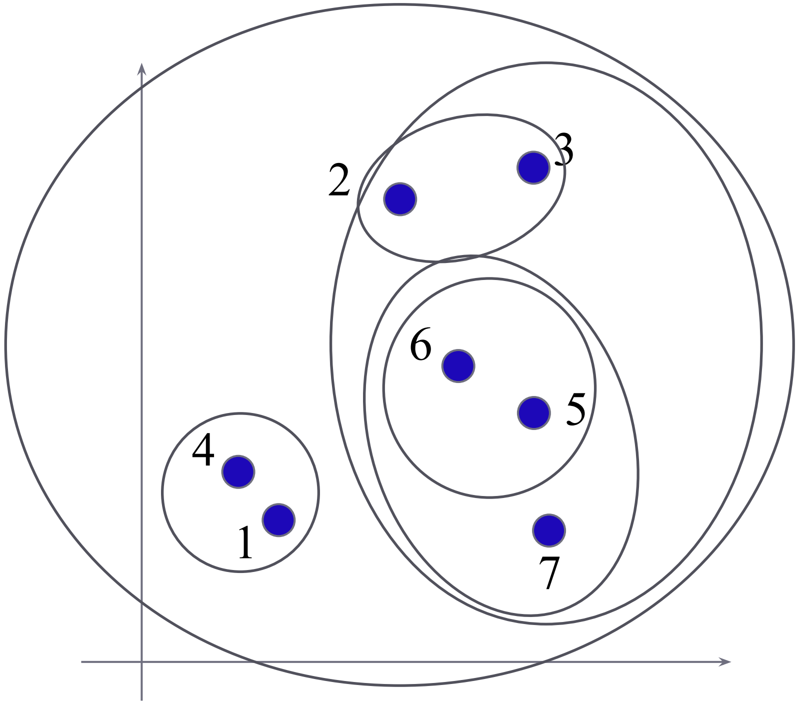


Average-link

$$dist_{AL}(\mathcal{C}_i,\mathcal{C}_j)=\frac{1}{|\mathcal{C}_i\cup\mathcal{C}_j|}\sum_{\boldsymbol{x}\in\mathcal{C}_i\cup\mathcal{C}_j}\sum_{\boldsymbol{x}'\in\mathcal{C}_i\cup\mathcal{C}_j} dist(\boldsymbol{x},\boldsymbol{x}')$$

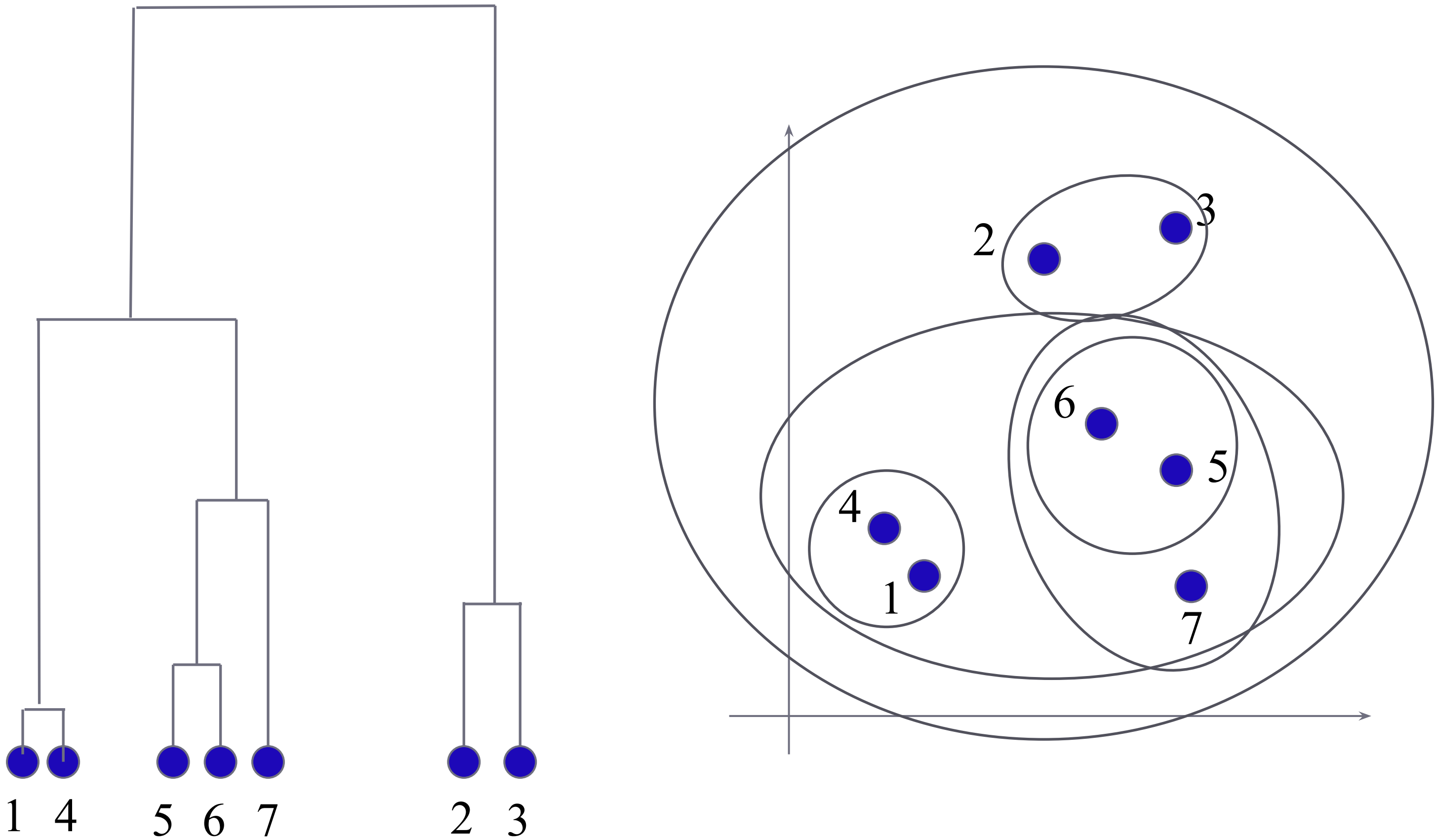
Single-Link





keep max bridge length as small as possible.

Complete Link



keep max diameter as small as possible.

Centroid Linkage

- The distance between two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ is then defined as the distance between their centeroids:

$$dist_{Centroid}\left(\mathcal{C}_i, \mathcal{C}_j\right) = dist(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$$

$$\boldsymbol{\mu}_j = \frac{1}{\left|\mathcal{C}_j\right|} \sum_{\boldsymbol{x} \in \mathcal{C}_j} \boldsymbol{x}$$

- Compute similarity of clusters in constant time:

Average Linkage

- $$dist_{Average}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\boldsymbol{x} \in C_i} \sum_{\boldsymbol{x'} \in C_j} dist(\boldsymbol{x}, \boldsymbol{x}')$$

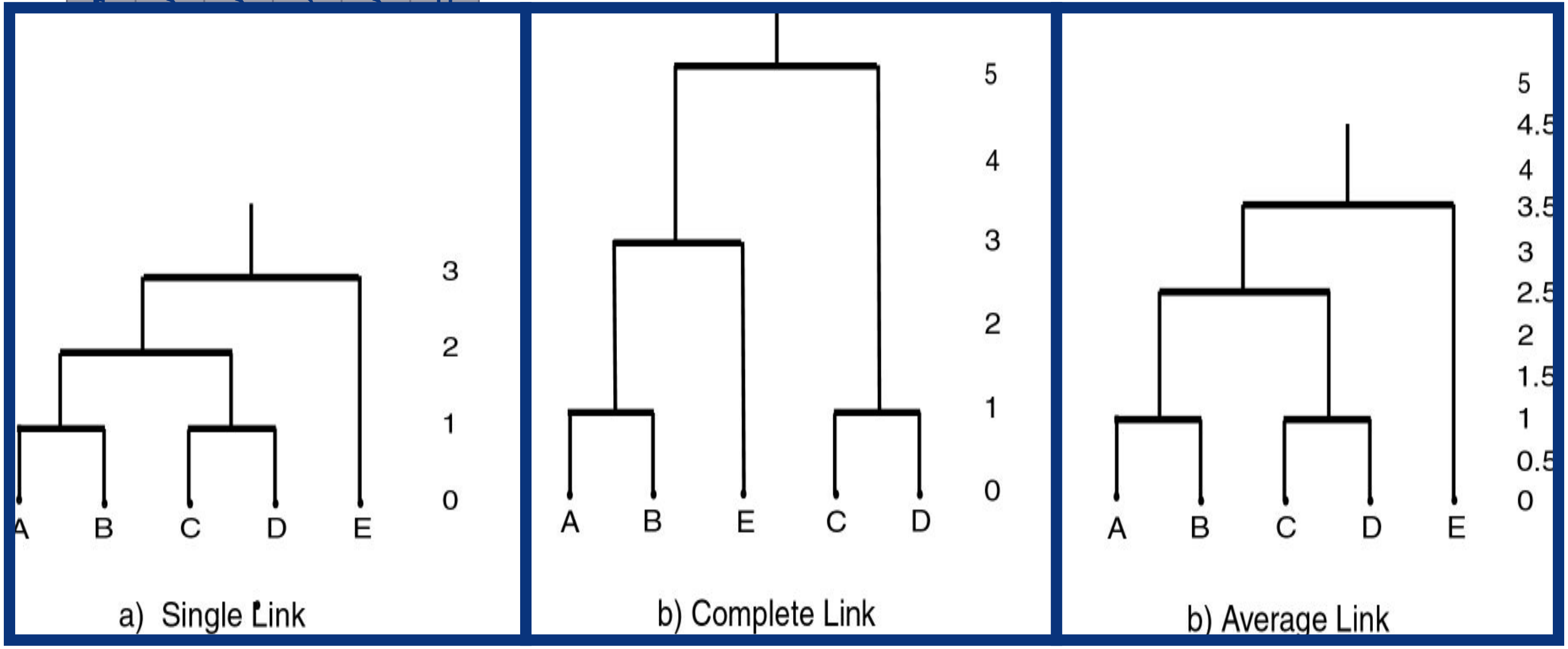
- Similarity of two clusters = average similarity of all pairs within merged cluster.

$$sim(C_i, C_j) = \frac{1}{|C_i \cup C_j|(|C_i \cup C_j| - 1)} \sum_{x \in (C_i \cup C_j)} \sum_{y \in (C_i \cup C_j), y \neq x} sim(x, y)$$

- Compromise between single and complete link.
- Two options:
  - Averaged across all ordered pairs in the merged cluster
  - Averaged over all pairs *between* the two original clusters
- No clear difference in efficancy

EXAMPL E

	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



a) Single Link

b) Complete Link

b) Average Link

Ward's method

- The distances between centers of the two clusters (weighted to consider sizes of clusters too):

$$dist_{Ward}(\mathcal{C}_i, \mathcal{C}_j) = \frac{|\mathcal{C}_i||\mathcal{C}_j|}{|\mathcal{C}_i| + |\mathcal{C}_j|} dist(\boldsymbol{c}_i, \boldsymbol{c}_j)$$

- Merge the two clusters such that the increase in k-means cost is as small as possible.
- Works well in practice.

Distances between Clusters: Summary

- Which distance is the best?
  - Complete linkage prefers compact clusters.
  - Single linkage can produce long stretched clusters.

- The choice depends on what you need.
  - expert opinion is helpful

Similarity Measure

- Similarity measure $s: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is an upper-bounded function
  - shows how close to each other each pair of instances
- Dissimilarity measure $d: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a lower-bounded function
  - that for each pair of instances shows how they far from each other

- Examples of similarity measure:
  - Dot product: $s(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$
  - Cosine: $s(\boldsymbol{x}, \boldsymbol{x}') = \frac{\boldsymbol{x}^T \boldsymbol{x}'}{\|\boldsymbol{x}\|\|\boldsymbol{x}'\|}$
  - Tanimoto: $s(\boldsymbol{x}, \boldsymbol{x}') = \frac{\boldsymbol{x}^T \boldsymbol{x}'}{\|\boldsymbol{x}\|^2 + \|\boldsymbol{x}'\|^2 - \boldsymbol{x}^T \boldsymbol{x}'}$

Distance Metric

- • The distance function $d: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **metric** if
  - $d(\boldsymbol{x}, \boldsymbol{x}') \geq 0$ (non-negativity)
  - $d(\boldsymbol{x}, \boldsymbol{x}') = d(\boldsymbol{x}', \boldsymbol{x})$ (symmetry)
  - $d(\boldsymbol{x}, \boldsymbol{x}') = 0$ iff $\boldsymbol{x} = \boldsymbol{x}'$ (isolation)
  - $d(\boldsymbol{x}, \boldsymbol{x}') \leq d(\boldsymbol{x}, \boldsymbol{x}'') + d(\boldsymbol{x}'', \boldsymbol{x}')$ (triangular inequality) [Why do we need it?]

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

Feature (Attribute) Types

- Real-value
  - e.g., weight
- Binary
  - e.g., gender (M/F), has-diabetes(T/F)
- Nominal (categorical)
  - e.g., Color (Red, Green, Blue, Yellow, …)
- Ordinal/Ranked
  - e.g., quality (bad, average, good, excellent)

Distance Metrics for Real-Valued Data

- L_p norms ($\mathrm{p} \in \mathbb{N}$) or **Minkowski** distance:

$$L_p(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_p = \left(\sum_{\mathrm{i}=1}^{\mathrm{d}} |\mathrm{x_i} - \mathrm{x_i'}|^p\right)^{1/p}$$

  - L_1 ($\mathrm{p} = 1$) is the **Manhattan** (or city block) distance:
    - $L_1(\mathbf{x}, \mathbf{x}') = \sum_{\mathrm{i}=1}^{\mathrm{d}} |\mathrm{x_i} - \mathrm{x_i'}|$
  - L_2 ($\mathrm{p} = 2$) is the **Euclidean** distance:
    - $L_2(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{d}} (\mathrm{x_i} - \mathrm{x_i'})^2}$
  - L_∞ ($\mathrm{p} = +\infty$) distance:
    - $L_\infty(\mathbf{x}, \mathbf{x}') = \max_{\mathrm{i}=1,\ldots\mathrm{d}} |\mathrm{x_i} - \mathrm{x_i'}|$

Distance Metrics for Real-Valued Data

- Weighted Euclidean distance:
  - Positive weights associated with variables based on data semantics.

$$\sqrt{\sum_{i=1}^{d} w_i \left(x_i - x_i'\right)^2}$$

- Mahalanobis distance ($\boldsymbol{B}$ is a symmetric positive semi-definite matrix):

$$d_{\boldsymbol{B}}(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{B} (\boldsymbol{x} - \boldsymbol{x}')}$$

  - Weighted Euclidean corresponds to $d_{\boldsymbol{B}}(\boldsymbol{x}, \boldsymbol{x}')$ where $\boldsymbol{B}$ is a diagonal matrix with diagonal elements $w_1, w_2, \dots, w_d$
  - Mahalanobis distance is equivalent to the Euclidean distance in the transformed space $\boldsymbol{A}^T \boldsymbol{x}$ where $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{B}$

Distance Metrics for Binary Data

- Jaccard (Tanimoto) similarity between binary vectors X and Y:

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

  - Jaccard distance between binary vectors X and Y:
    - $1 - Jaccard(X,Y)$

- Hamming distance between binary vectors:
  - Number of corresponding elements that are different
  - Equal to L_1 metric.

Data Matrix vs. Distance Matrix

- **Data** (or pattern) Matrix: $N \times d$ (features of data):

$$\boldsymbol{X} = \begin{bmatrix} x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}$$

- **Distance** Matrix: $N \times N$ (distances of each pattern pair):

$$\boldsymbol{D} = \begin{bmatrix} d(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & \cdots & d(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(N)}) \\ \vdots & \ddots & \vdots \\ d(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(1)}) & \cdots & d(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(N)}) \end{bmatrix}$$

Single-link, complete-link, and average link only needs the distance matrix.

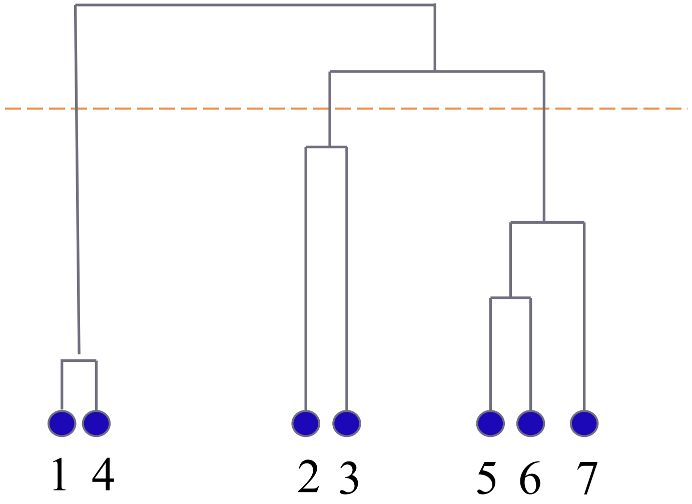
Computational Complexity

- In the first iteration, all HAC methods compute similarity of all pairs of N individual instances which is $O(N^2)$ similarity computation.
- In each $N-2$ merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- If done naively $O(N^3)$ but if done more cleverly $O(N^2 \log N)$

Dendrogram: Hierarchical Clustering

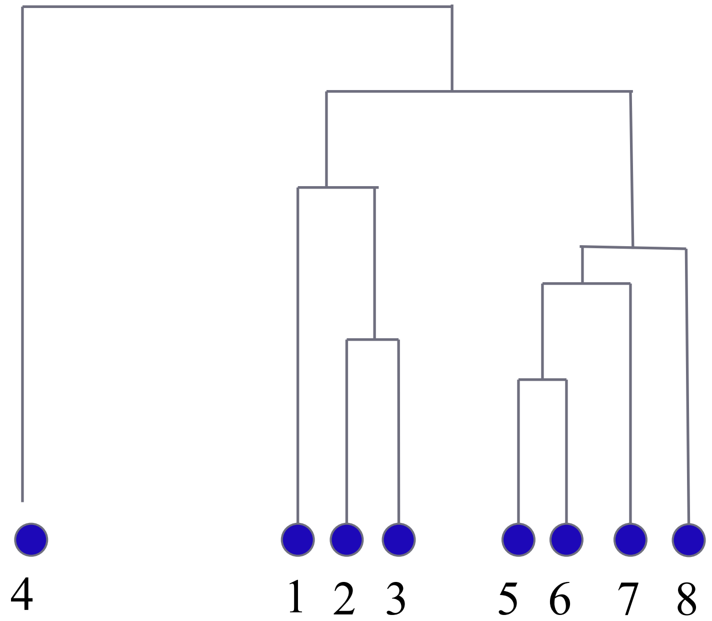
- Clustering obtained by cutting the dendrogram at a desired level
  - Cut at a pre-specified level of similarity
  - where the gap between two successive combination similarities is largest
  - select the cutting point that produces *K* clusters

Where to "cut" the dendrogram is user-determined.



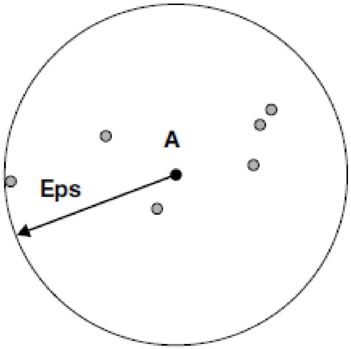
Outliers

- We can detect outliers (that are very different to all others) by finding the isolated branches

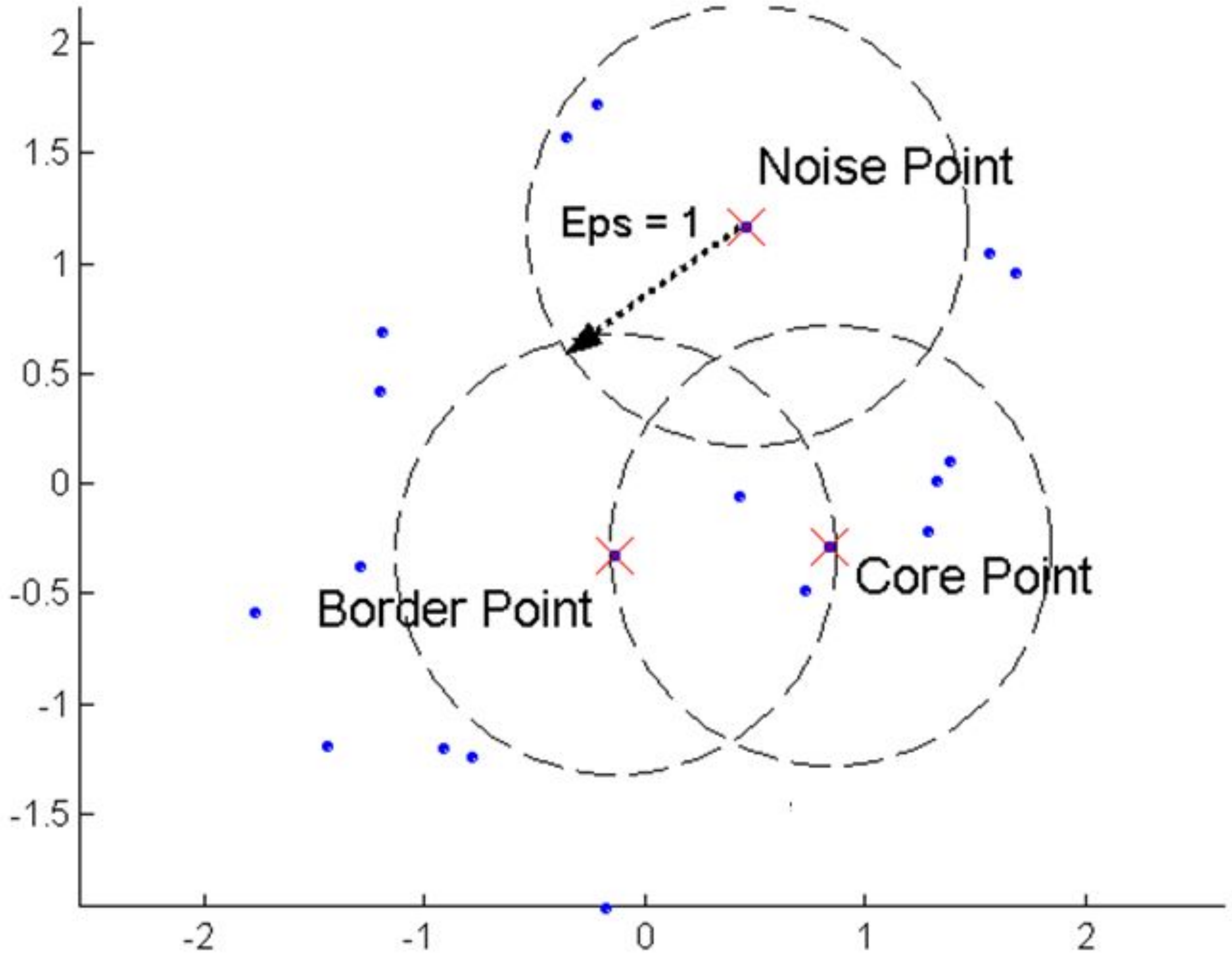


DBSCAN

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
  - A point is a core point if it has more than a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
  - 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.

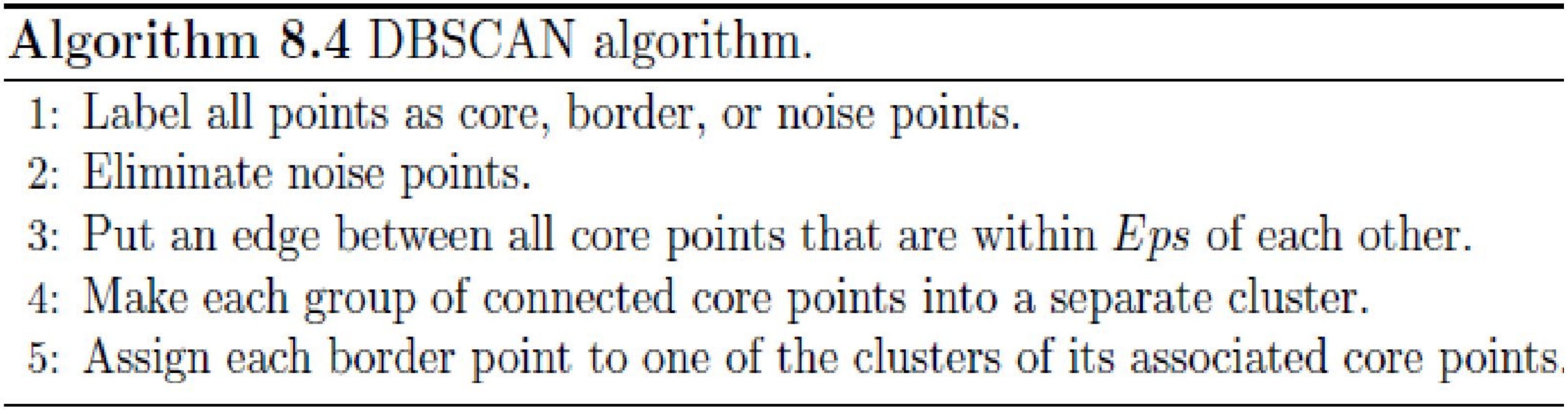


DBSCAN: Core, Border, and Noise Points



Sharif University of Technology

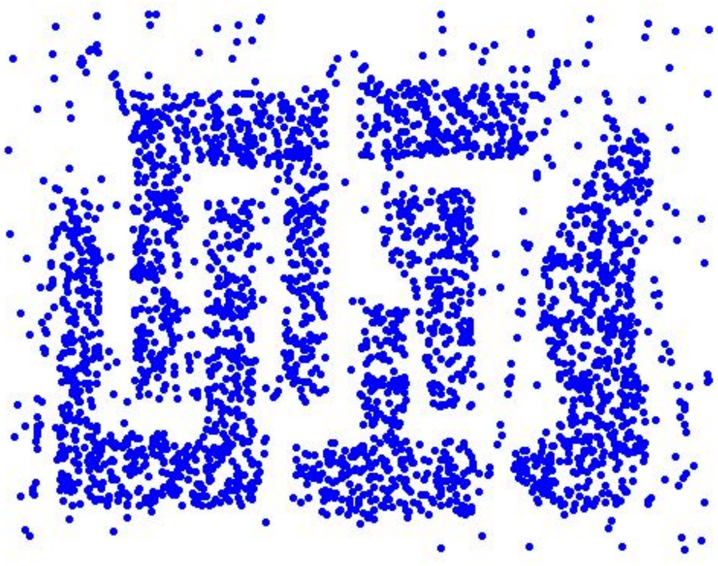
DBSCAN

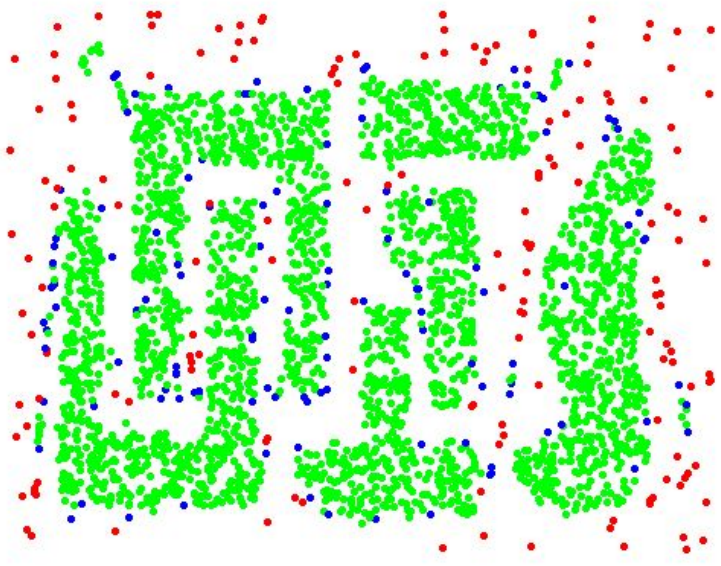


Algorithm 8.4 DBSCAN algorithm.

```
Label all points as core, border, or noise points.
Eliminate noise points.
Put an edge between all core points that are within $Eps$ of each other.
Make each group of connected core points into a separate cluster.
Assign each border point to one of the clusters of its associated core points.
```

DBSCAN: Core, Border and Noise Points

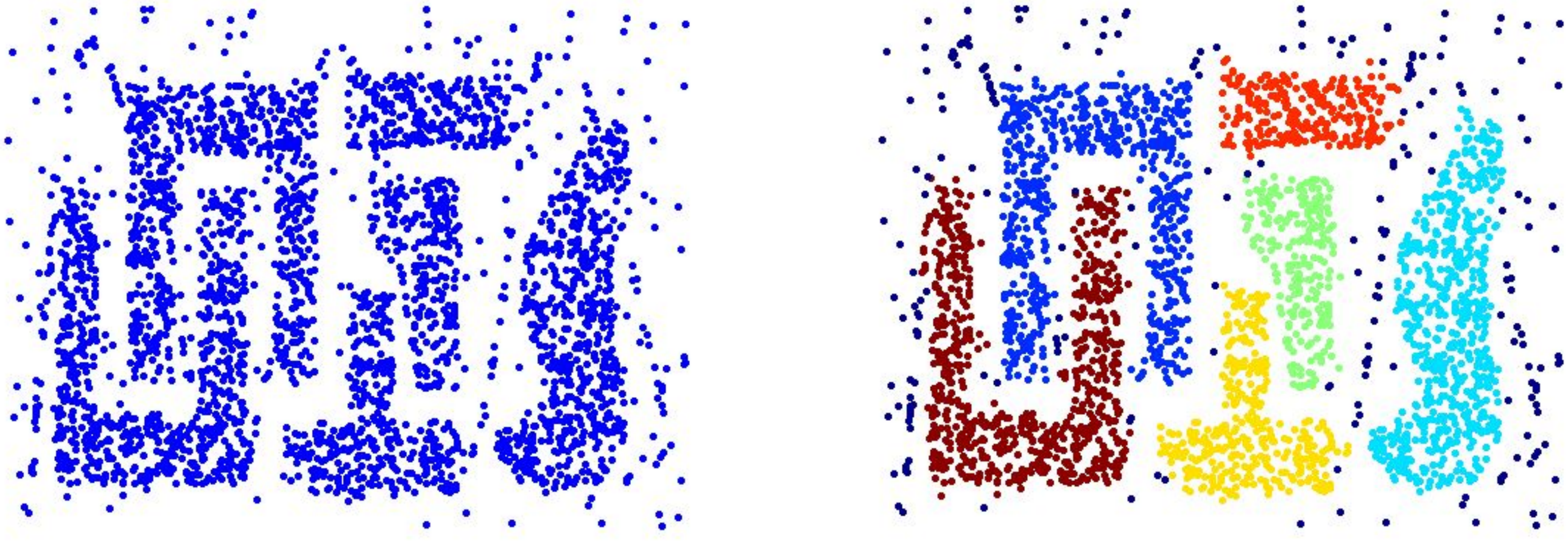




core border

noise

DBSCAN



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

DBSCAN: Determining EPS and MinPts

how to determine the parameters *Eps* and *MinPts*

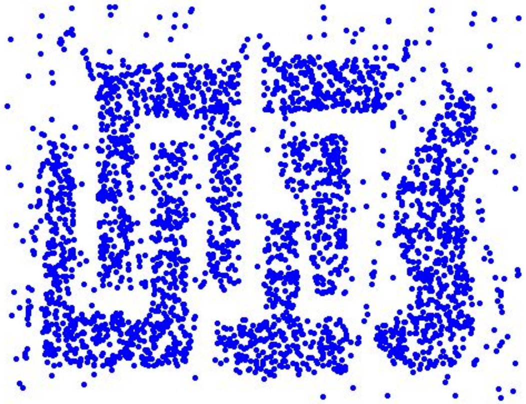
MinPts:

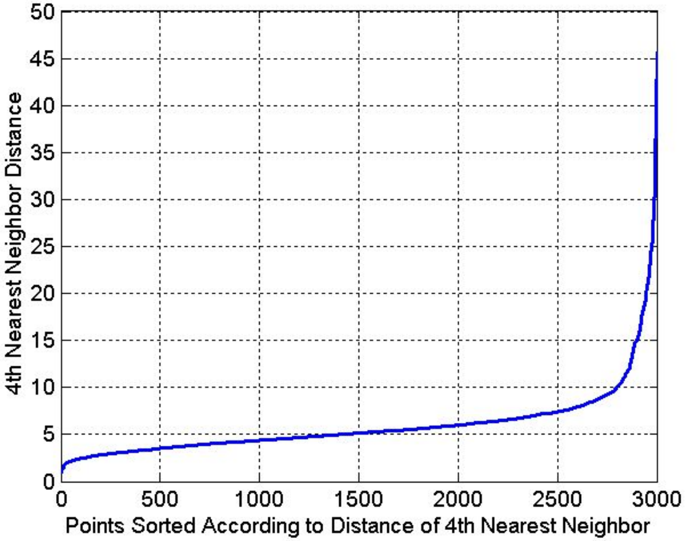
- MinPts=K **too small**, **noise or outliers** will be incorrectly labeled as clusters
- k is too large, **small clusters** are likely to be labeled as noise (k = 4)

Eps:

- look at the behavior of the distance from a point to its kth nearest neighbor(k-dist)
- Points belong to some cluster, the value of k-dist small if k is not larger than the cluster size
- points not in a cluster, such as noise points, the *k*-dist relatively large
- **compute the *k*-dist** for all the data points for some *k*
- **sort them** in increasing order, and then plot the sorted values
- **a sharp change** at the value of *k*-dist

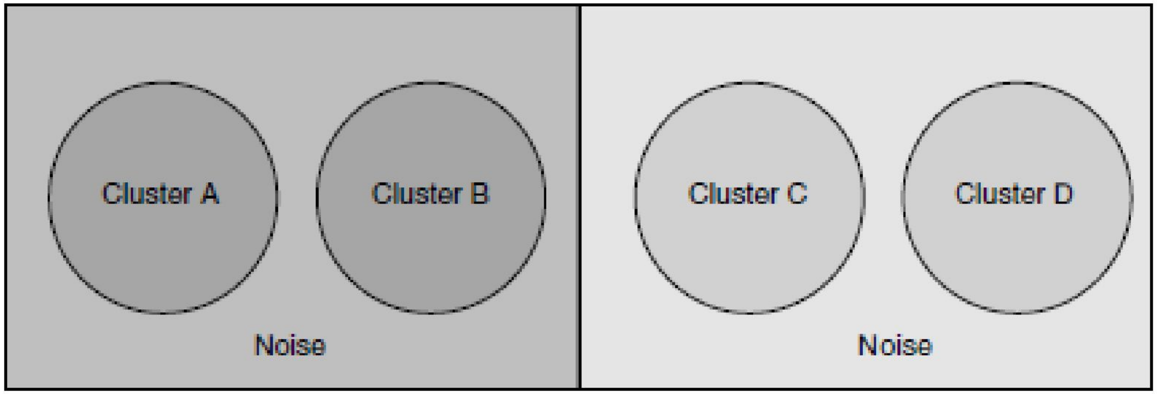
DBSCAN: Determining EPS and MinPts





Clusters of Varying Density

DBSCAN can have trouble with density if the **density of clusters** varies widely



- *Eps* threshold is low enough that DBSCAN finds *C* and *D* as clusters, then *A* and *B* and the points surrounding them will become a single cluster

- *Eps threshold high enough that DBSCAN finds A and B as separate clusters, and the points surrounding them are marked as noise, then C and D and the points surrounding them will also be marked as noise*

Cluster Validity

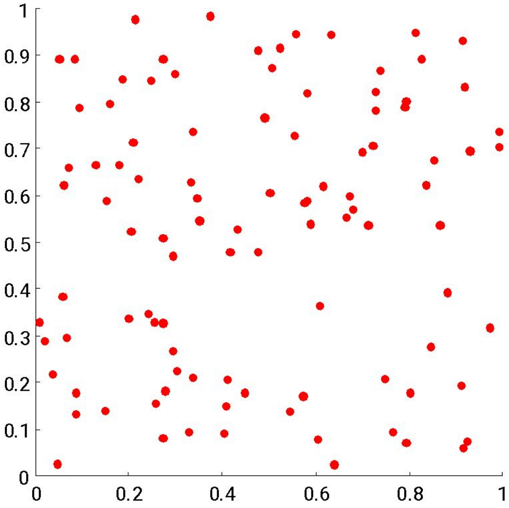
- For supervised classification we have a variety of measures to evaluate how good our model is
  - Accuracy, precision, recall

- For cluster analysis, the analogous question is how to evaluate the "goodness" of the resulting clusters?

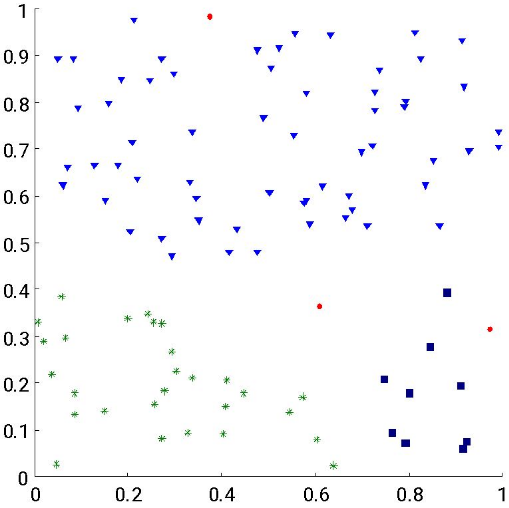
- Then why do we want to evaluate them?
  - To avoid finding patterns in noise
  - To compare clustering algorithms
  - To compare two sets of clusters
  - To compare two clusters

Clusters found in Random Data

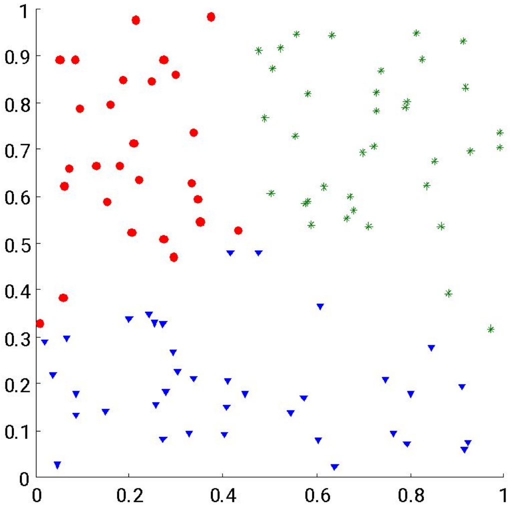
Random Points



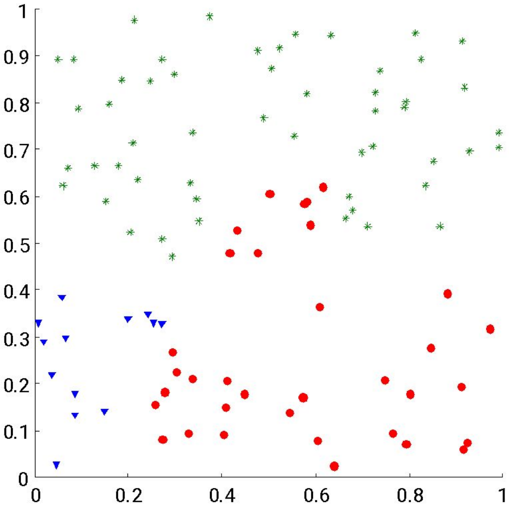
DBSCAN



K-means



Complete Link



Clustering Validity

- We need to determine whether the found clusters are real or compare different clustering methods.

- What is a good clustering?
  - clustering quality measurement

- Main approaches:
  - **Internal index**: evaluate how well the clustering fit the data without reference to an external information.
  - **External index**: evaluate how well is the clustering result with respect to known categories.
    - Assumption: Ground truth labels are available

Internal Index: Stability

- Evaluate cluster stability to minor perturbation of data.
  - For example, evaluate a clustering result by comparing it with the obtained result after subsampling of data (e.g., subsampling 80% of data).

- To find stability, we need a measure of similarity between two k-clusterings.
  - It is based on comparing two k-clusterings
    - Similar to external indices that compare the clustering result with the ground truth.

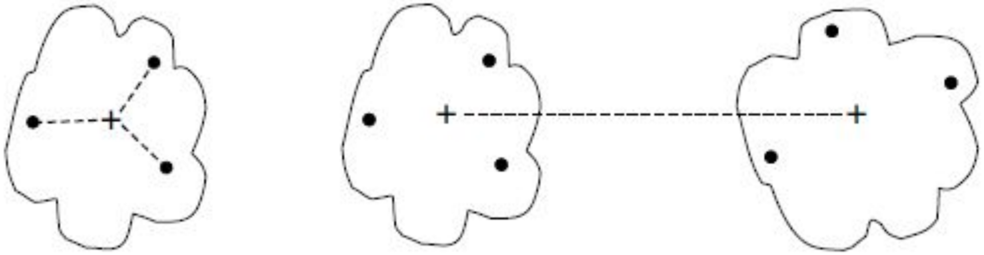
Internal Index: Coherence

- Internal criterion is usually based on coherence:
  - Compactness of the data in the clusters
    - high intra-cluster similarity (closeness of cluster elements)
  - Separability of distinct clusters
    - low inter-cluster similarity

- Some internal indices: Davies-Bouldin (DB), Silhouette , DUNN, Bayesian information criterion (BIC), Calinski-Harabasz (CH)

Unsupervised Measures: Cohesion and Separation

Center-Based View



(a) Cohesion.

(b) Separation.

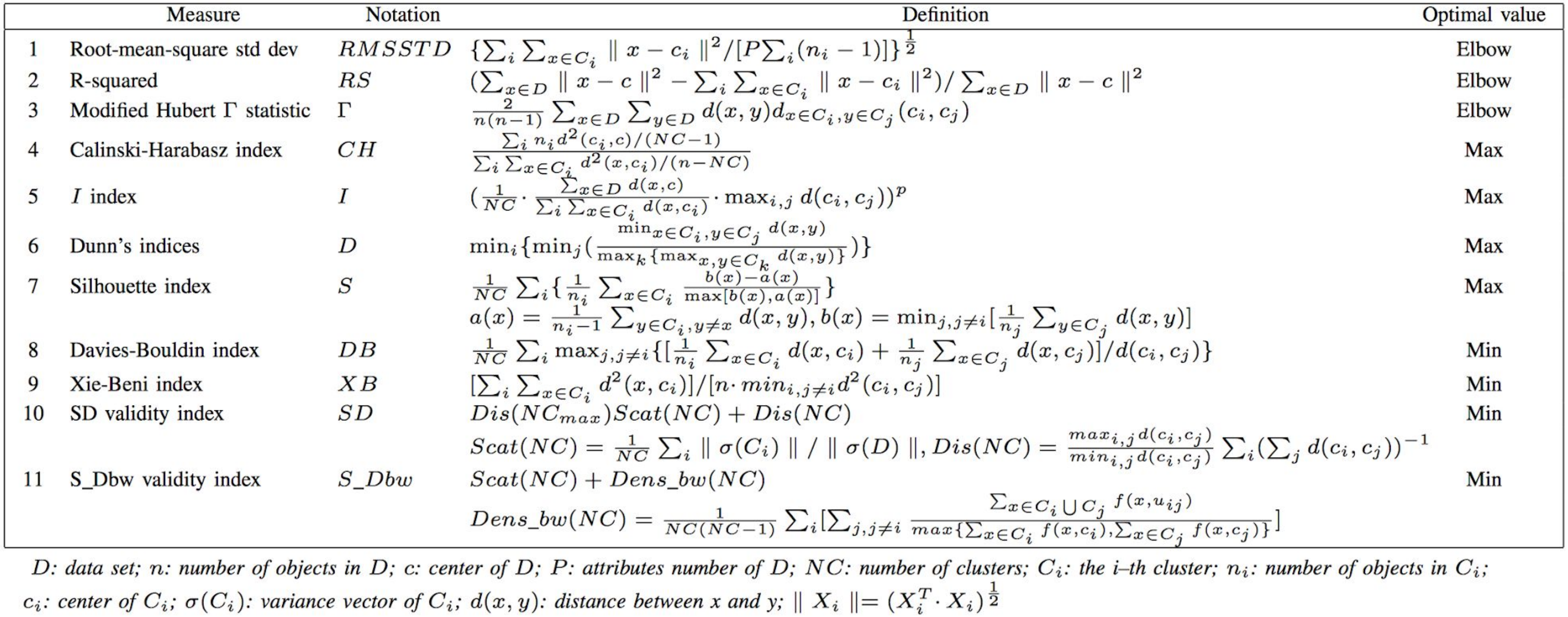
Figure 8.28. Prototype-based view of cluster cohesion and separation.

$$cohesion(C_i) = \sum_{\mathbf{x} \in C_i} proximity(\mathbf{x}, \mathbf{c}_i)$$

$$\begin{aligned} separation(C_i, C_j) &= proximity(\mathbf{c}_i, \mathbf{c}_j) \\ separation(C_i) &= proximity(\mathbf{c}_i, \mathbf{c}) \end{aligned}$$

title

Internal Clustering Validation Measures



	Measure	Notation	Definition	Optimal value
1	Root-mean-square std dev	$RMSSTD$	$\{\sum_i \sum_{x \in C_i} \lVert x - c_i \rVert^2 / [P \sum_i (n_i - 1)]\}^{\frac{1}{2}}$	Elbow
2	R-squared	RS	$(\sum_{x \in D} \lVert x - c \rVert^2 - \sum_i \sum_{x \in C_i} \lVert x - c_i \rVert^2) / \sum_{x \in D} \lVert x - c \rVert^2$	Elbow
3	Modified Hubert Γ statistic	Γ	$\frac{2}{n(n-1)} \sum_{x \in D} \sum_{y \in D} d(x, y) d_{x \in C_i, y \in C_j}(c_i, c_j)$	Elbow
4	Calinski-Harabasz index	CH	$\frac{\sum_i n_i d^2(c_i, c)/(NC-1)}{\sum_i \sum_{x \in C_i} d^2(x, c_i)/(n-NC)}$	Max
5	I index	I	$(\frac{1}{NC} \cdot \frac{\sum_{x \in D} d(x,c)}{\sum_i \sum_{x \in C_i} d(x, c_i)} \cdot \max_{i,j} d(c_i, c_j))^p$	Max
6	Dunn's indices	D	$\min_i \{\min_j (\frac{\min_{x \in C_i, y \in C_j} d(x,y)}{\max_k \{\max_{x,y \in C_k} d(x,y)\}})\}$	Max
7	Silhouette index	S	$\frac{1}{NC} \sum_i \{\frac{1}{n_i} \sum_{x \in C_i} \frac{b(x) - a(x)}{\max[b(x), a(x)]}\}$ $a(x) = \frac{1}{n_i - 1} \sum_{y \in C_i, y \neq x} d(x, y), b(x) = \min_{j, j \neq i} [\frac{1}{n_j} \sum_{y \in C_j} d(x, y)]$	Max
8	Davies-Bouldin index	DB	$\frac{1}{NC} \sum_i \max_{j, j \neq i} \{[\frac{1}{n_i} \sum_{x \in C_i} d(x, c_i) + \frac{1}{n_j} \sum_{x \in C_j} d(x, c_j)] / d(c_i, c_j)\}$	Min
9	Xie-Beni index	XB	$[\sum_i \sum_{x \in C_i} d^2(x, c_i)] / [n \cdot min_{i, j \neq i} d^2(c_i, c_j)]$	Min
10	SD validity index	SD	$Dis(NC_{max}) Scat(NC) + Dis(NC)$ $Scat(NC) = \frac{1}{NC} \sum_i \lVert \sigma(C_i) \rVert / \lVert \sigma(D) \rVert, Dis(NC) = \frac{max_{i,j} d(c_i, c_j)}{min_{i,j} d(c_i, c_j)} \sum_i (\sum_j d(c_i, c_j))^{-1}$	Min
11	S_Dbw validity index	S_Dbw	$Scat(NC) + Dens\_bw(NC)$ $Dens\_bw(NC) = \frac{1}{NC(NC-1)} \sum_i [\sum_{j, j \neq i} \frac{\sum_{x \in C_i \bigcup C_j} f(x, u_{ij})}{max\{\sum_{x \in C_i} f(x, c_i), \sum_{x \in C_j} f(x, c_j)\}}]$	Min

D: data set; n: number of objects in D; c: center of D; P: attributes number of D; NC: number of clusters; C_i: the i–th cluster; n_i: number of objects in C_i; c_i: center of C_i; $\sigma(C_i)$: variance vector of C_i; $d(x, y)$: distance between x and y; $\lVert X_i \rVert = (X_i^T \cdot X_i)^{\frac{1}{2}}$

External Index: Rand Index and Clustering F-measure

- $RI = \frac{TP+TN}{TP+TN+FP+FN}$

- $P = \frac{TP}{TP+FP}, R = \frac{TP}{TP+FN}$

- $F_{\beta} = \frac{(\beta^2+1)PR}{\beta^2 P+R}$
  - *F measure* in addition supports differential weighting of P and R.

- $Jaccard = \frac{TP}{TP+FP+FN}$

TP: # pairs that cluster together in both C and $\hat{C}$
TN: # pairs that are in separate clusters in both C and $\hat{C}$
FN: # pairs that cluster together in C but not in $\hat{C}$
FP: # pairs that cluster together in $\hat{C}$ but not in C

	Same	Different
Same	TP	FN
Different	FP	TN

Major Dilemma [Jain, 2010]

- What is a cluster?
  - What **features** should be used?
  - Should the data be **normalized**?
  - How do we define the **pair-wise similarity**?
  - Which **clustering method** should be used?
  - How **many clusters** are present in the data?
  - Does the data contain any **outliers**?
  - Does the data have any clustering **tendency**?
  - Are the discovered clusters and partition **valid**?

K-means vs. Hierarchical

- Time cost:
  - K-means is usually fast while hierarchical methods do not scale well
- Human intuition
  - Hierarchical structure provides more natural output compatible with human intuition in some domains
- Local minimum problem
  - It is very common for k-means
  - Hierarchical methods like any heuristic search algorithms also suffer from local optima problem.
    - Since they can never undo what was done previously and greedily merge clusters
- Choosing of the number of clusters
  - There is no need to specify the number of clusters in advance for hierarchical methods

External Index

- Comparing clustering result with externally known clustering, e.g., to externally given class labels.

External validation

- For this we need an external source that contains related, but usually not identical information.
- For example, assume we are clustering web pages based on the car pictures they contain.
- We have independently grouped these pages based on the text description they contain.
- Can we use the text based grouping to determine how well our clustering works?

- Suppose we have generated k clusters C1,...,Ck. How do we assess the significance of their relation to m known (potentially overlapping) categories G1,...,Gm?
- Let's start by comparing a single cluster C with a single category Gj. The p-value for such a match is based on the hyper-geometric distribution.
- Board.
- This is the probability that a randomly chosen |Ci| elements out of n would have l elements in common with Gj.

External Criteria: Purity

- Target Clusters: $C = \{C_1,K ,C_c\}$

- Found Clusters: $C' = \{C'_1,K ,C'_k\}$

$|C_i| = n_i$

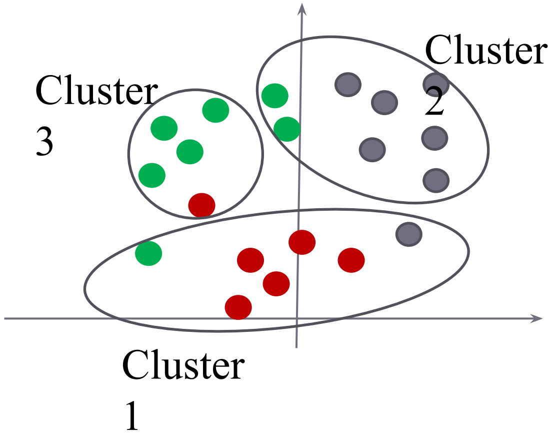
$|\overset{\boxtimes}{C}_i| = n'_i$

$n_{ij} = |C_i \boxtimes \overset{\boxtimes}{C}_i|$

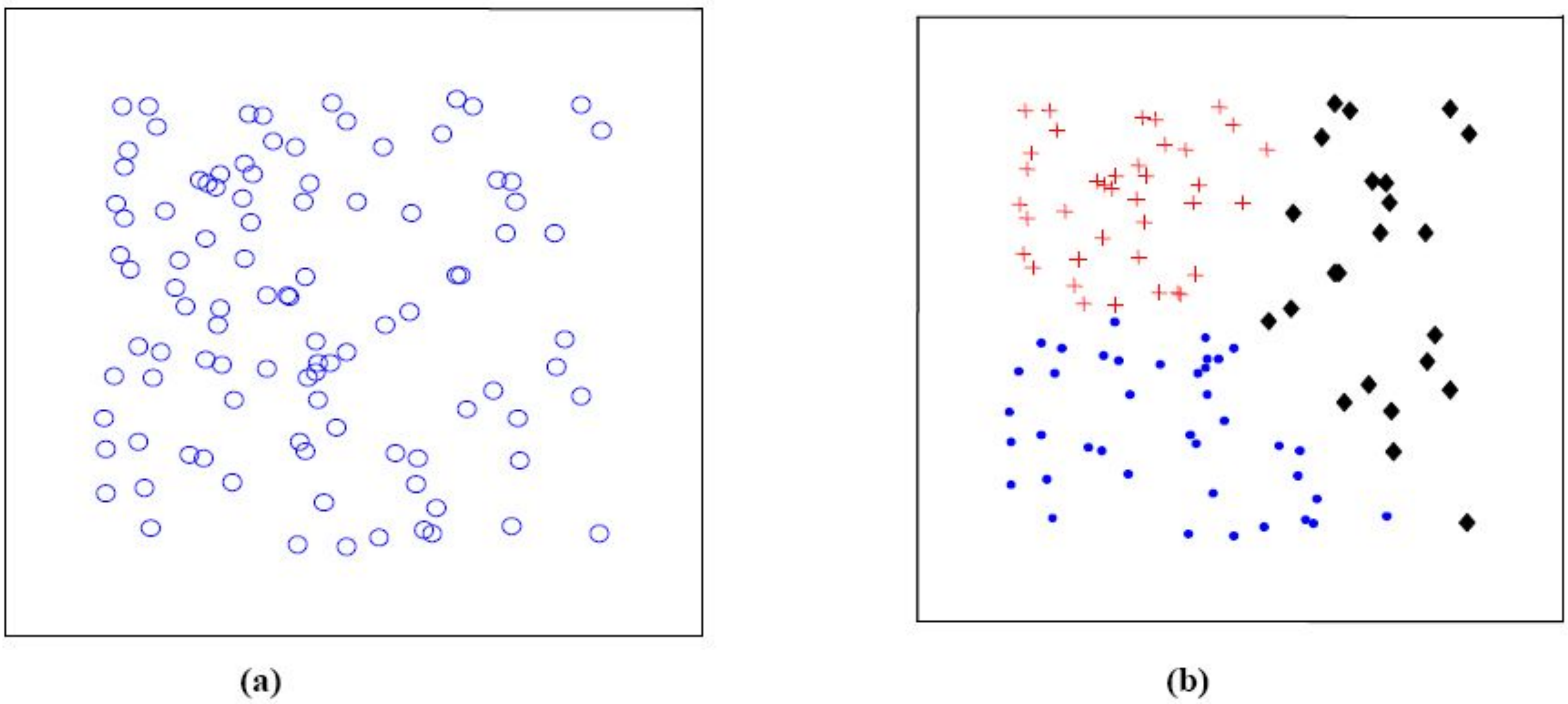
$$Purity(\mathcal{C},\hat{\mathcal{C}}) = \frac{1}{N}\sum_{i=1}^{k}\max_{j=1,\ldots,c}|\mathcal{C}_j \cap \hat{\mathcal{C}}_i|$$

- Purity prefers more clusters

$$\frac{5+6+4}{20} = 0.75$$



Cluster Validation: Clustering Tendency



(a)

(b)

Figure 8 Cluster validity. (a) A dataset with no “natural” clustering; (b) K-means partition with *K* = 3.

Within and Between Cluster Criteria

Let's consider total point scatter for a set of *N* data points:

$$T = \frac{1}{2}\sum_{i=1}^{N}\sum_{l=1}^{N} d^2(\mathbf{x}_i, \mathbf{x}_l)$$

squared distance between two points

T can be re-written as:

$$T = \frac{1}{2}\sum_{j=1}^{k}\sum_{\mathbf{x}_i \in C_j}\Big(\sum_{\mathbf{x}_l \in C_j} d^2(\mathbf{x}_i, \mathbf{x}_l) + \sum_{\mathbf{x}_l \notin C_j} d^2(\mathbf{x}_i, \mathbf{x}_l)\Big)$$

$$= W(C) + B(C)$$

Within cluster scatter →

$$W(C) = \frac{1}{2}\sum_{j=1}^{k}\sum_{\mathbf{x}_i \in C_j}\sum_{\mathbf{x}_l \in C_j} d^2(\mathbf{x}_i, \mathbf{x}_l)$$

Between cluster scatter →

$$B(C) = \frac{1}{2}\sum_{j=1}^{k}\sum_{\mathbf{x}_i \in C_j}\sum_{\mathbf{x}_l \notin C_j} d^2(\mathbf{x}_i, \mathbf{x}_l)$$

If *d* is square Euclidean distance, then

$$W(C) = \sum_{j=1}^{k} |C_j| \sum_{\mathbf{x}_i \in C_j} \left\| \boldsymbol{\mu}\!\!\!\mathbf{x}_i - {}_j \right\|^2$$

$$B(C) = k\sum_{j=1}^{K} |C_j| \left\| \boldsymbol{\mu}_j - \boldsymbol{\mu} \right\|^2$$

Total or grand mean

Minimizing *W*(*C*) is equivalent to maximizing *B*(*C*)

K-means issues, variations, etc.

- Recomputing the centroid after every assignment
  - Instead of computing it after all points are re-assigned
  - It can improve speed of convergence of *K*-means

- Assumes clusters are spherical in vector space
  - Sensitive to coordinate changes, weighting etc.

- Disjoint and exhaustive
  - Doesn't have a notion of "outliers" by default
  - But can add outlier filtering

K-medoids Algorithm

- It must choose a set of k points $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_k\}$ from dataset $\mathcal{X}$ and form clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k\}$
  - similar to k-means except to the location of the cluster representatives that must be selected only on the data points locations
  - Also known as PAM (Partitioning Around Medoids)

- Steps of a k-medoids algorithm:
  - Select randomly k medoids from the original data points $\mathcal{X}$
  - repeat until there is no change
    - Assign each of the $N - k$ remaining points in $\mathcal{X}$ to their closest medoid
    - For each medoid m and (non-medoid) data point o associated to m
      - Swap m and o if it improves the total clustering cost

Within and Between Cluster Criteria

Let's consider total point scatter for a set of *N* data points:

$$T = \frac{1}{2}\sum_{i=1}^{N}\sum_{l=1}^{N} d^2(\mathbf{x}_i, \mathbf{x}_l)$$

squared distance between two points

T can be re-written as:

$$T = \frac{1}{2}\sum_{j=1}^{k}\sum_{\mathbf{x}_i \in C_j}\Big(\sum_{\mathbf{x}_l \in C_j} d^2(\mathbf{x}_i, \mathbf{x}_l) + \sum_{\mathbf{x}_l \notin C_j} d^2(\mathbf{x}_i, \mathbf{x}_l)\Big)$$

$$= W(C) + B(C)$$

Within cluster scatter →

$$W(C) = \frac{1}{2}\sum_{j=1}^{k}\sum_{\mathbf{x}_i \in C_j}\sum_{\mathbf{x}_l \in C_j} d^2(\mathbf{x}_i, \mathbf{x}_l)$$

Between cluster scatter →

$$B(C) = \frac{1}{2}\sum_{j=1}^{k}\sum_{\mathbf{x}_i \in C_j}\sum_{\mathbf{x}_l \notin C_j} d^2(\mathbf{x}_i, \mathbf{x}_l)$$

If *d* is square Euclidean distance, then

$$W(C) = \sum_{j=1}^{k}|C_j|\sum_{\mathbf{x}_i \in C_j}\left\|\mathbf{x}_i - \quad_j\right\|^2$$

$$B(C) = k\sum_{j=1}^{K}|C_j|\left\|\boldsymbol{\mu}_j - \boldsymbol{\mu}\right\|^2$$

Total or grand mean

Minimizing *W*(*C*) is equivalent to maximizing *B*(*C*)

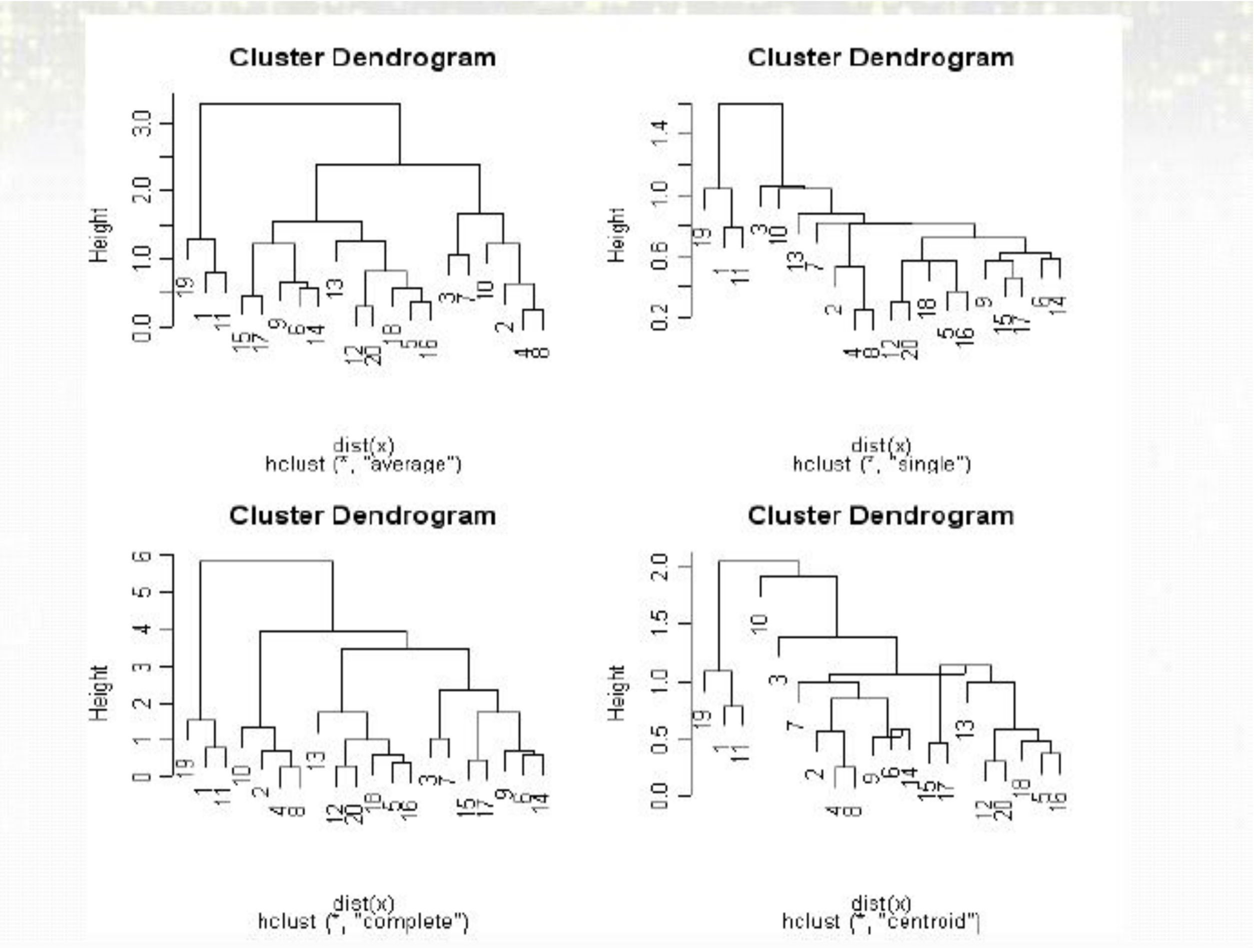
K-means issues, variations, etc.

- Recomputing the centroid after every assignment
  - Instead of computing it after all points are re-assigned
  - It can improve speed of convergence of *K*-means

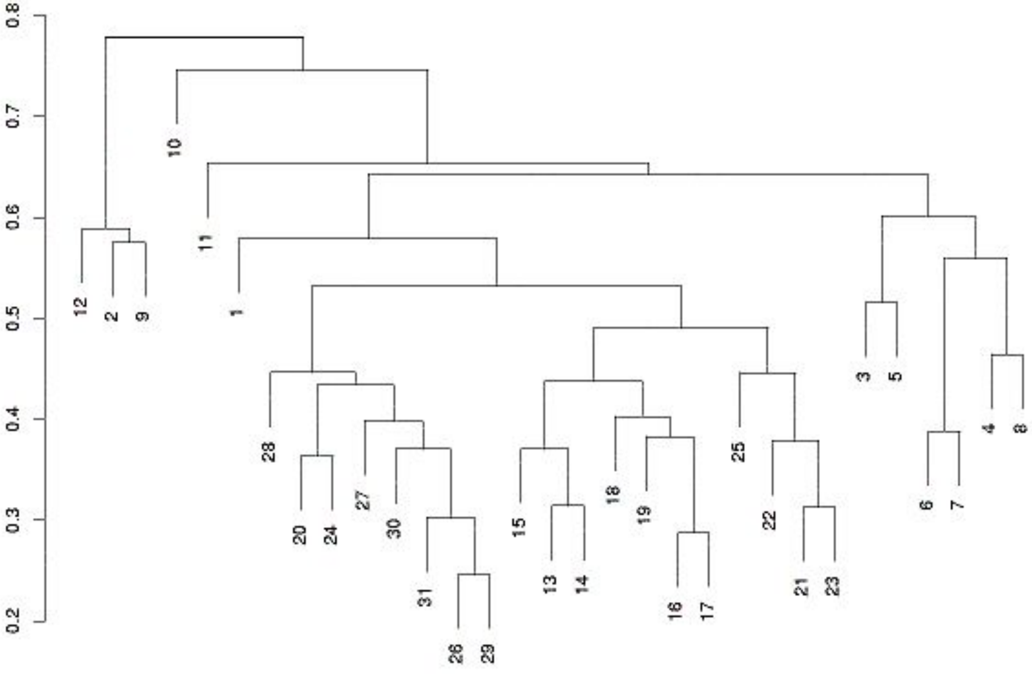
- Assumes clusters are spherical in vector space
  - Sensitive to coordinate changes, weighting etc.

- Disjoint and exhaustive
  - Doesn't have a notion of "outliers" by default
  - But can add outlier filtering

Hierarchical clustering



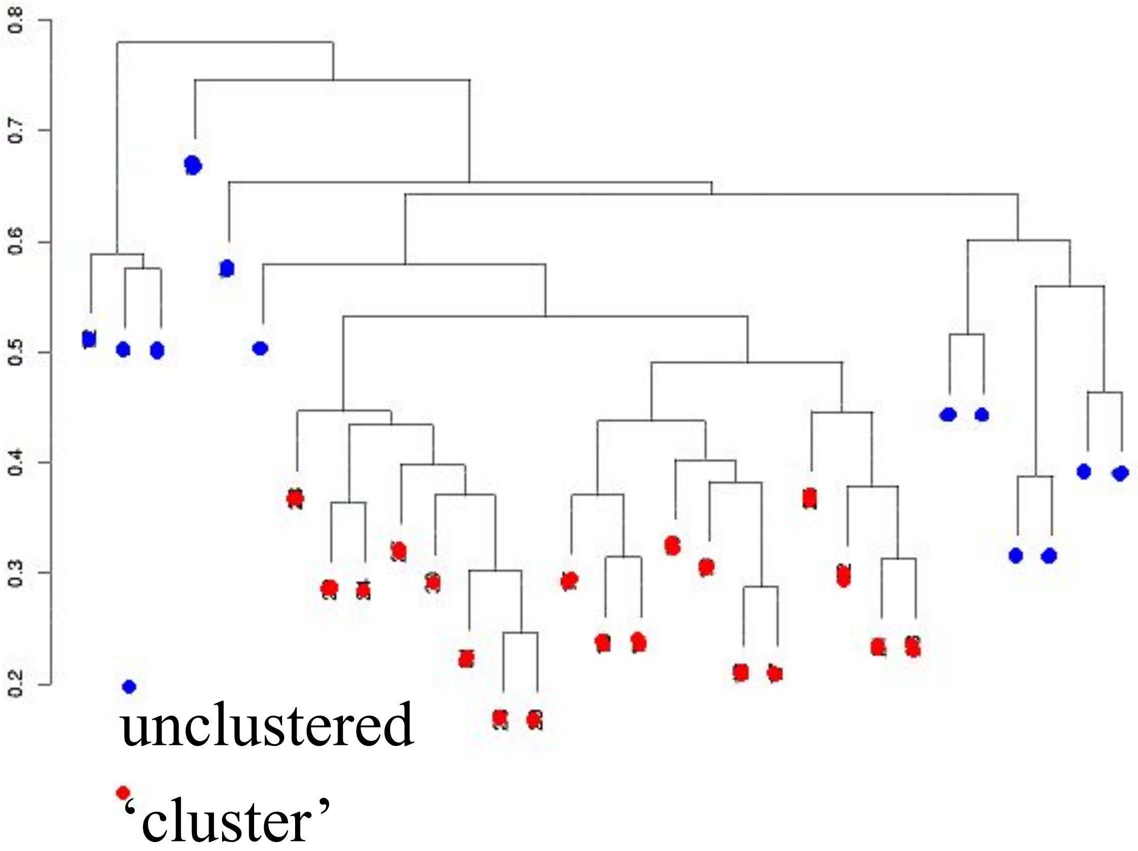
Average linkage hierarchical clustering, melanoma only



How many clusters are present?

Average linkage hierarchical clustering, melanoma only

1-ρ = .54


● unclustered
● ‘cluster’