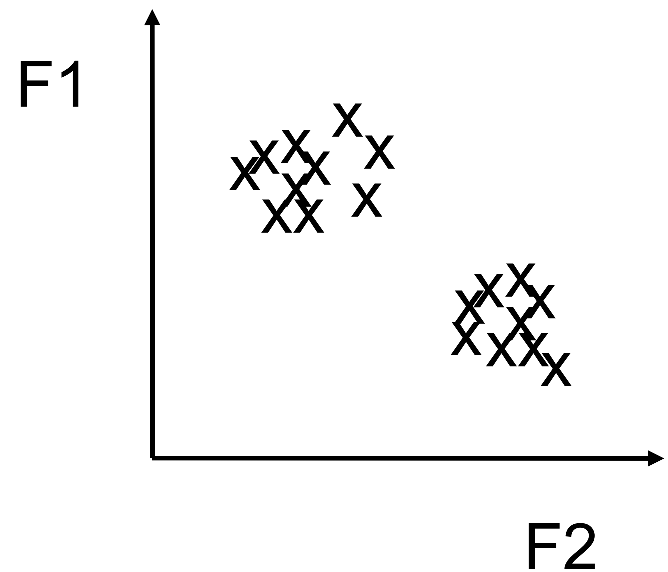
Clustering

Le Ou-Yang

Shenzhen University

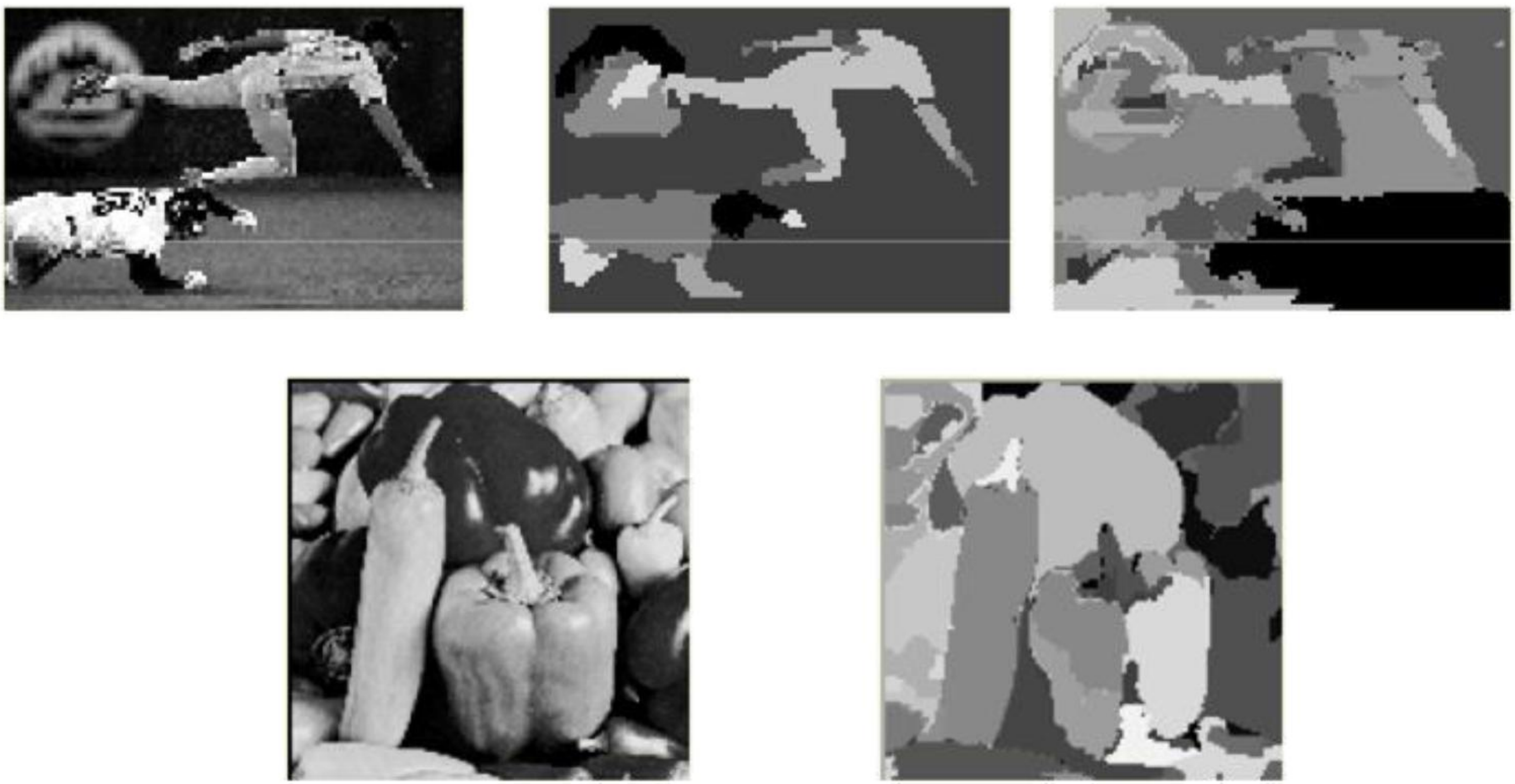
What is clustering?

- Given a set of data points, each described by a set of attributes, find clusters such that:
  - Intra-cluster similarity is maximized
  - Inter-cluster similarity is minimized



- Requires the definition of a similarity measure

Computer vision application: Image segmentation

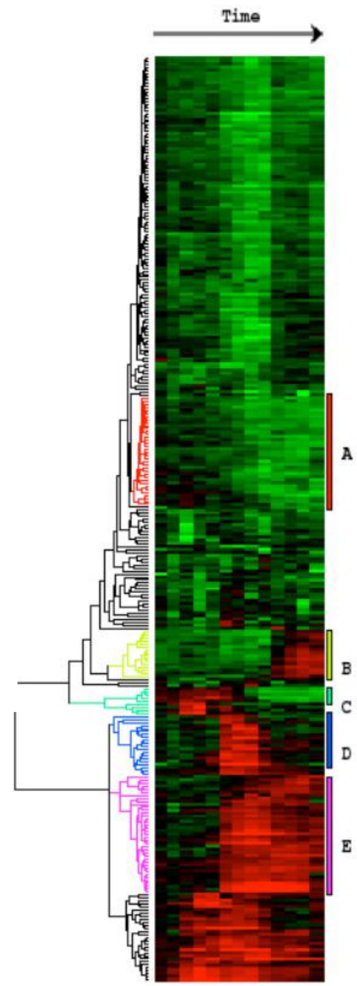


From: Image Segmentation by Nested Cuts, O. Veksler, CVPR2000

Biomedical application

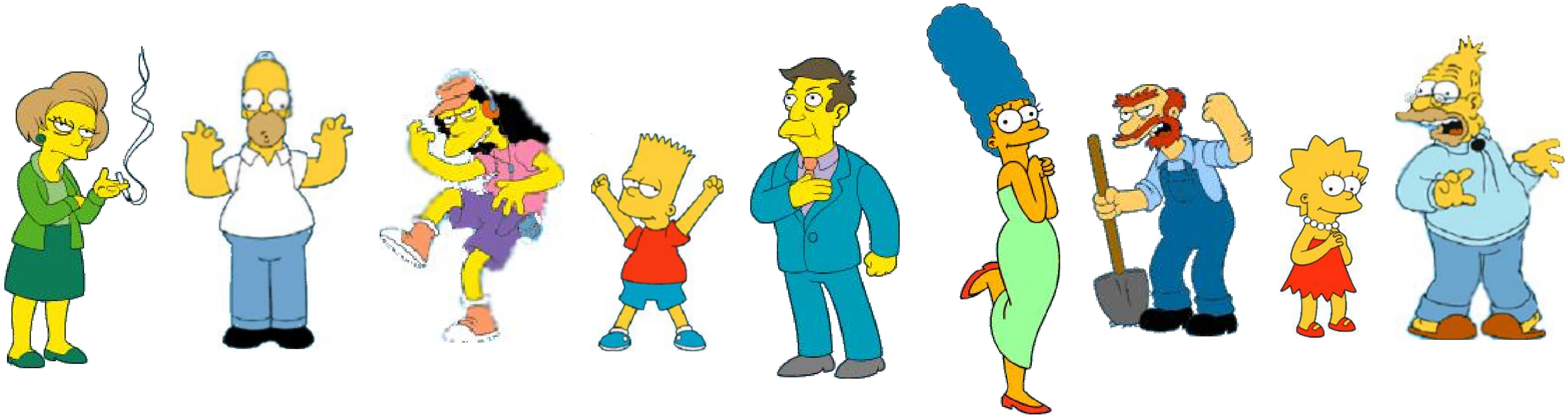
Clustering gene expression data

- Microarrays measure the actvities of all genes in different conditions
- Clustering genes can help determine new functions for unknown genes

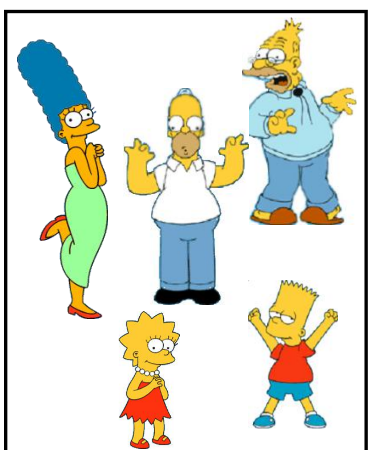


Eisen et al, PNAS 1998

Clustering

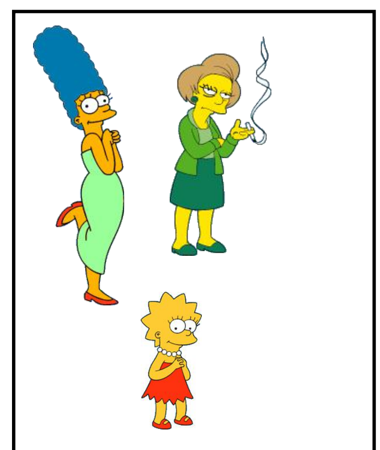


Clustering is subjective

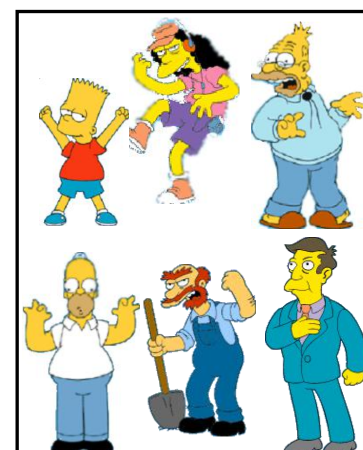


Simpson's Family

School Employees



Females



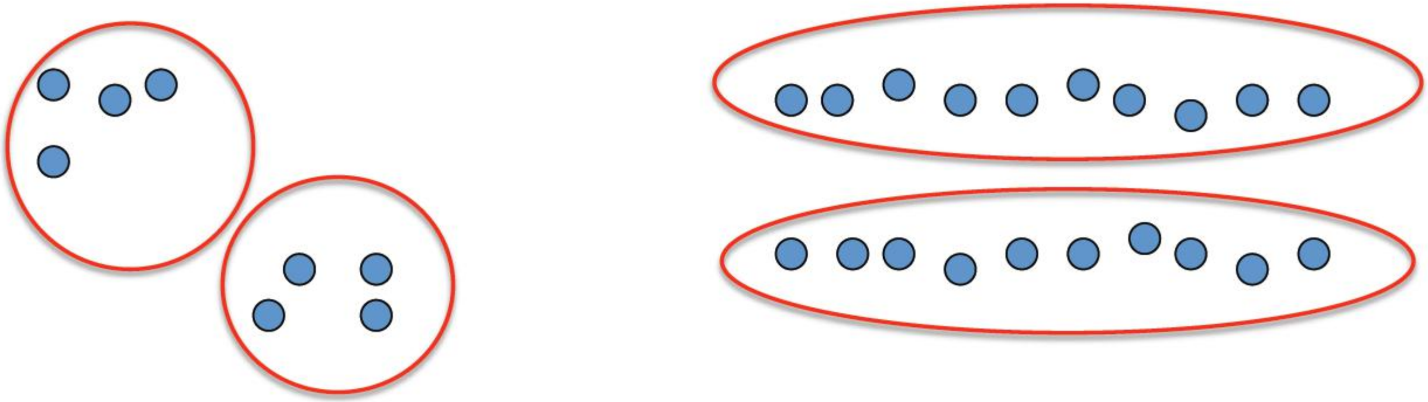
Males

What is Similarity?

Similarity is hard to define, but…
"*We know it when we see it*"

Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



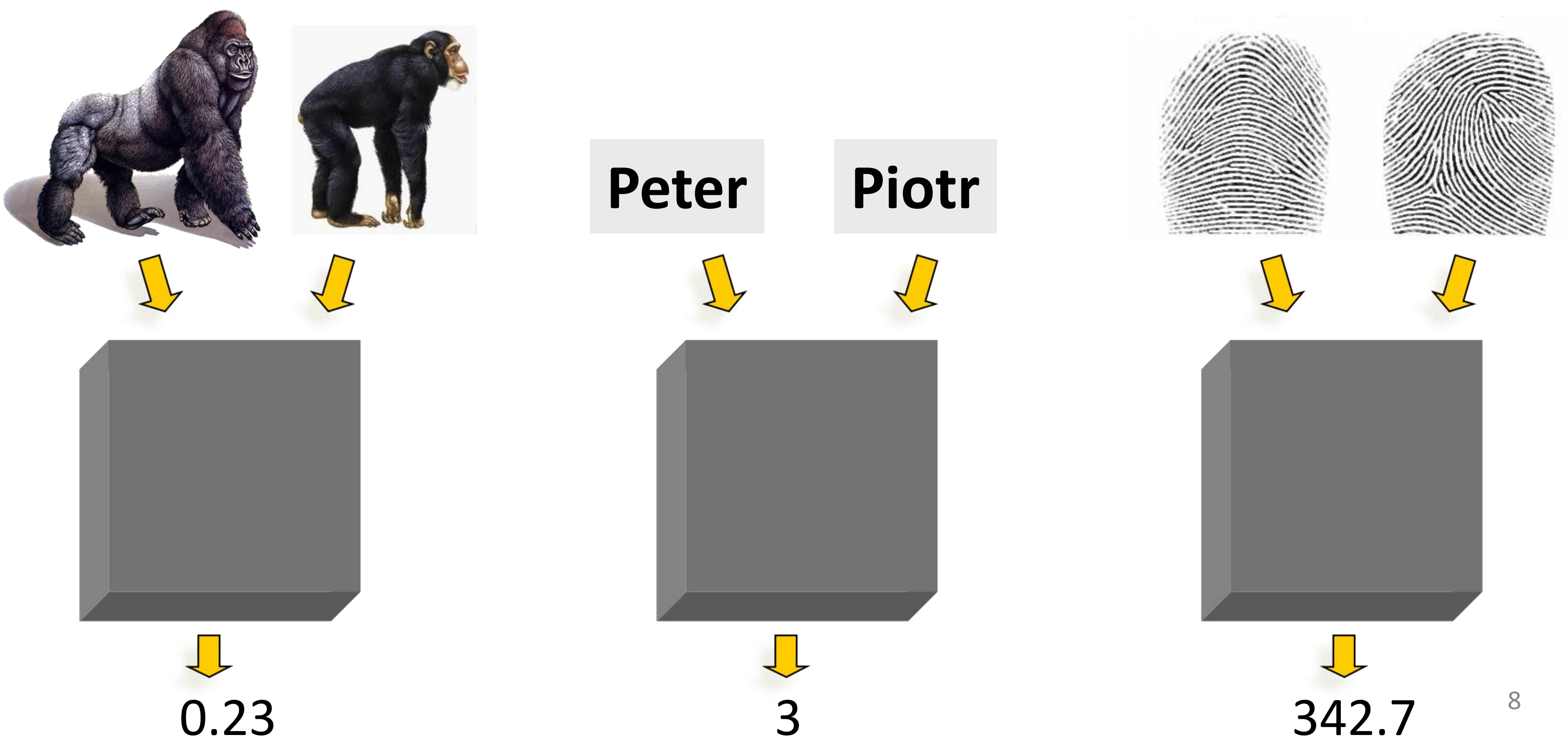
- What could “similar” mean?
  - One option: small Euclidean distance (squared)

$$\text{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$$

  - Clustering results are crucially dependent on the measure of similarity (or distance) between “points” to be clustered

Distance Measures

Definition: 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 $D(O_1,O_2)$



Distance Measures

What properties should a distance measure have?

- $D(A,B) = D(B,A)$ *Symmetry*
- $D(A,A) = 0$ *Constancy of Self-Similarity*
- $D(A,B) = 0$ iif $A = B$ *Positivity (Separation)*
- $D(A,B) \leq D(A,C) + D(B,C)$ *Triangular Inequality*

Desirable Properties of a Clustering Algorithm

- Scalability (in terms of both time and space)
- Ability to deal with different data types
- Minimal requirements for domain knowledge to determine input parameters
- Interpretability and usability

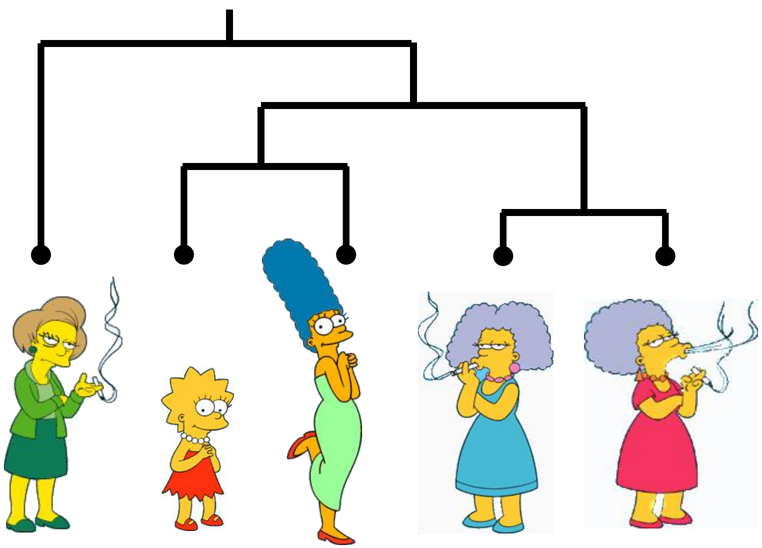
Optional

- Incorporation of user-specified constraints

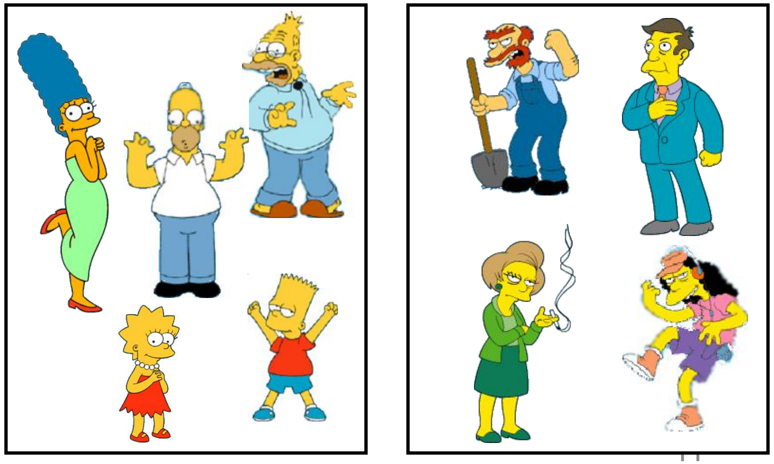
Clustering Methods

- Partitional algorithms
- Hierarchical algorithms
- Density-based algorithms
- Mixture model
- Spectral methods

Hierarchical

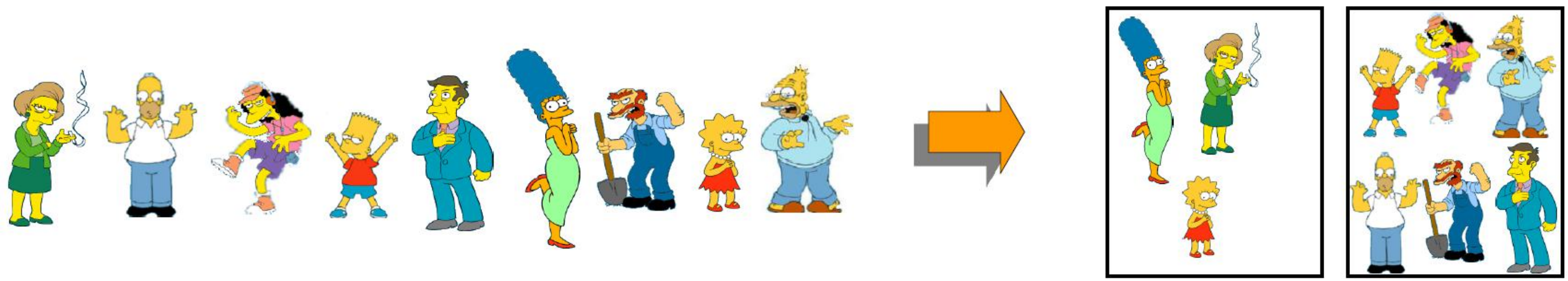


Partitional



Partitional Clustering

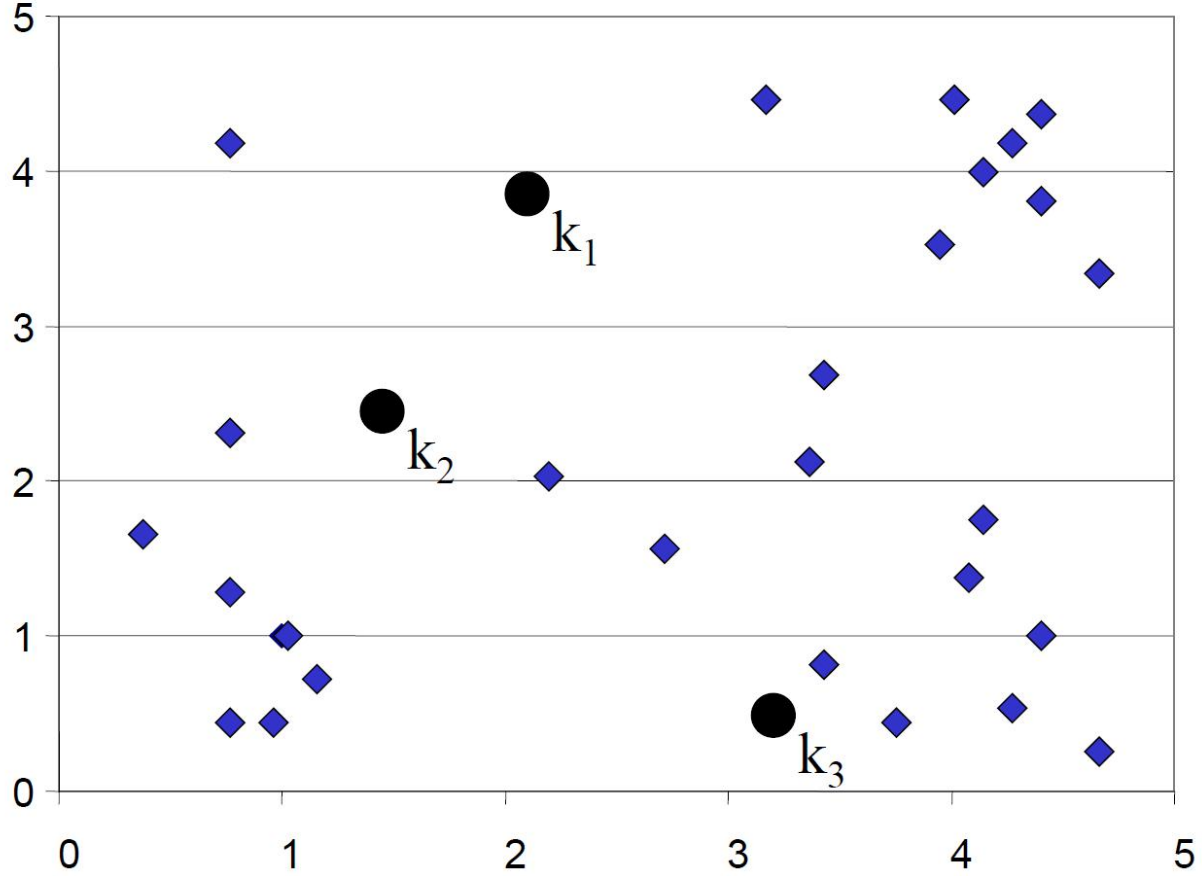
- Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.
- Since the output is only one set of clusters, the user has to specify the desired number of clusters K.



K-means Clustering: Initialization

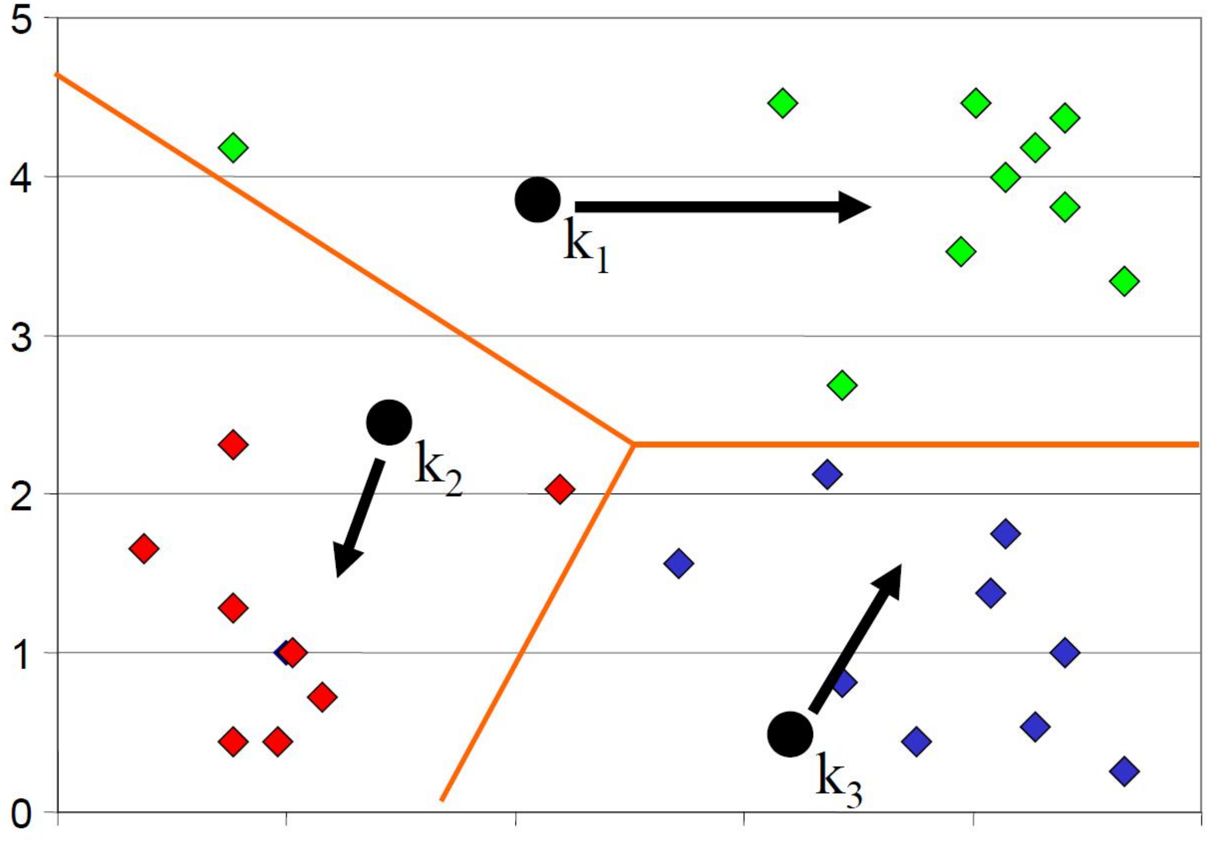
Algorithm: k-means, Distance Metric: Euclidean Distance

Decide K, and initialize K centers (randomly)



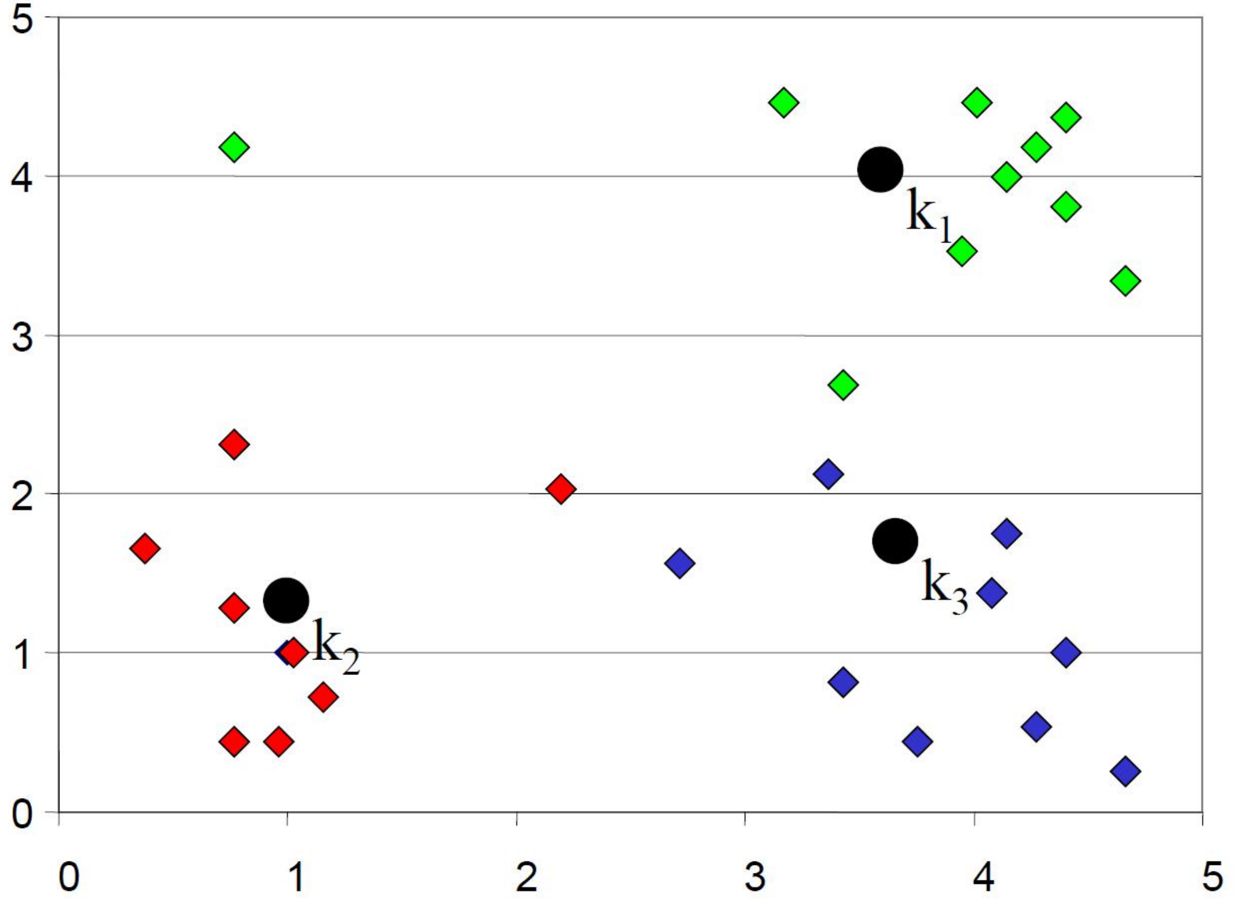
K-means Clustering: Iteration 1

Assign all objects to the nearest center.
Move a center to the mean of its members.



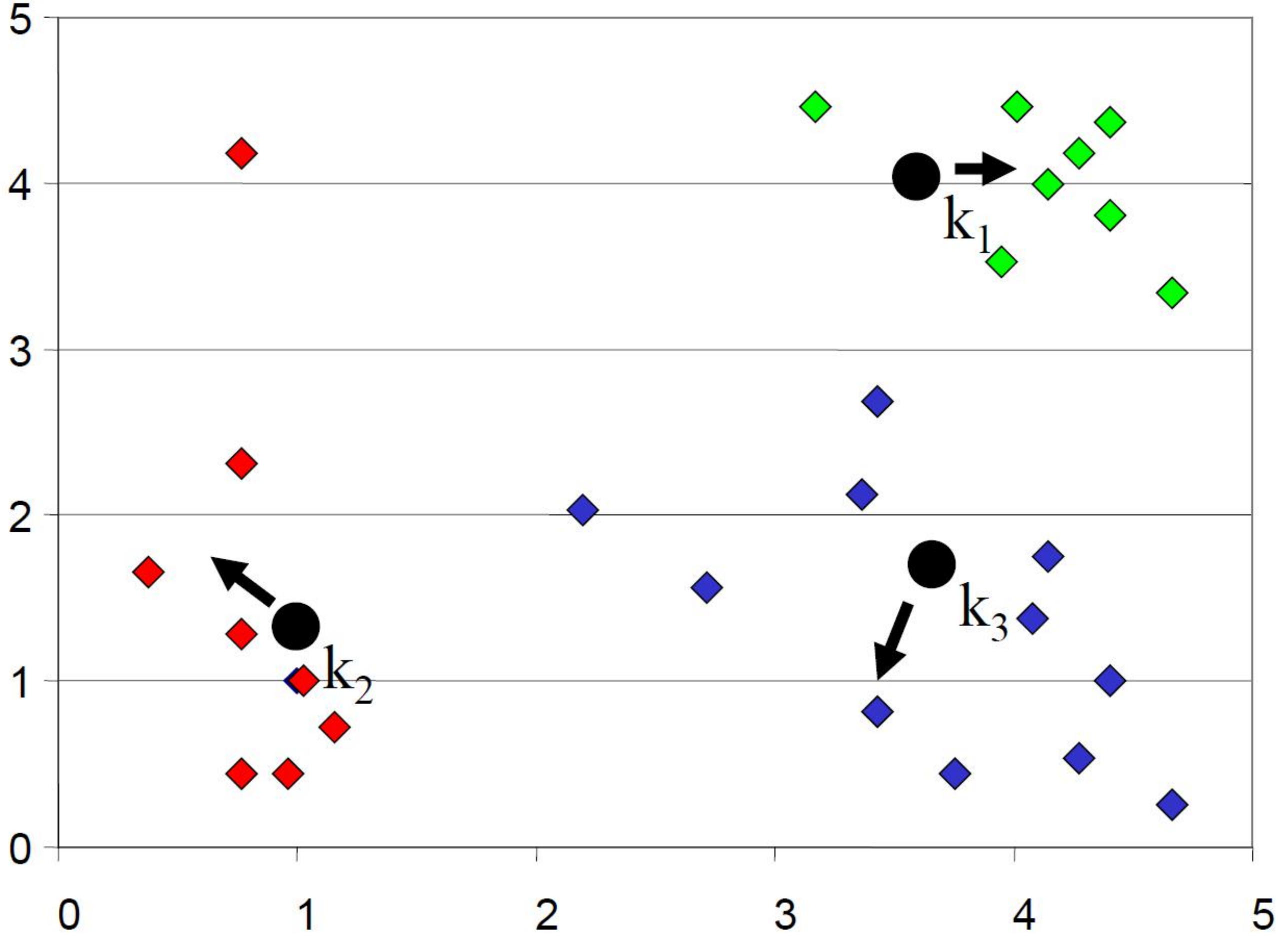
K-means Clustering: Iteration 2

After moving centers, re-assign the objects.



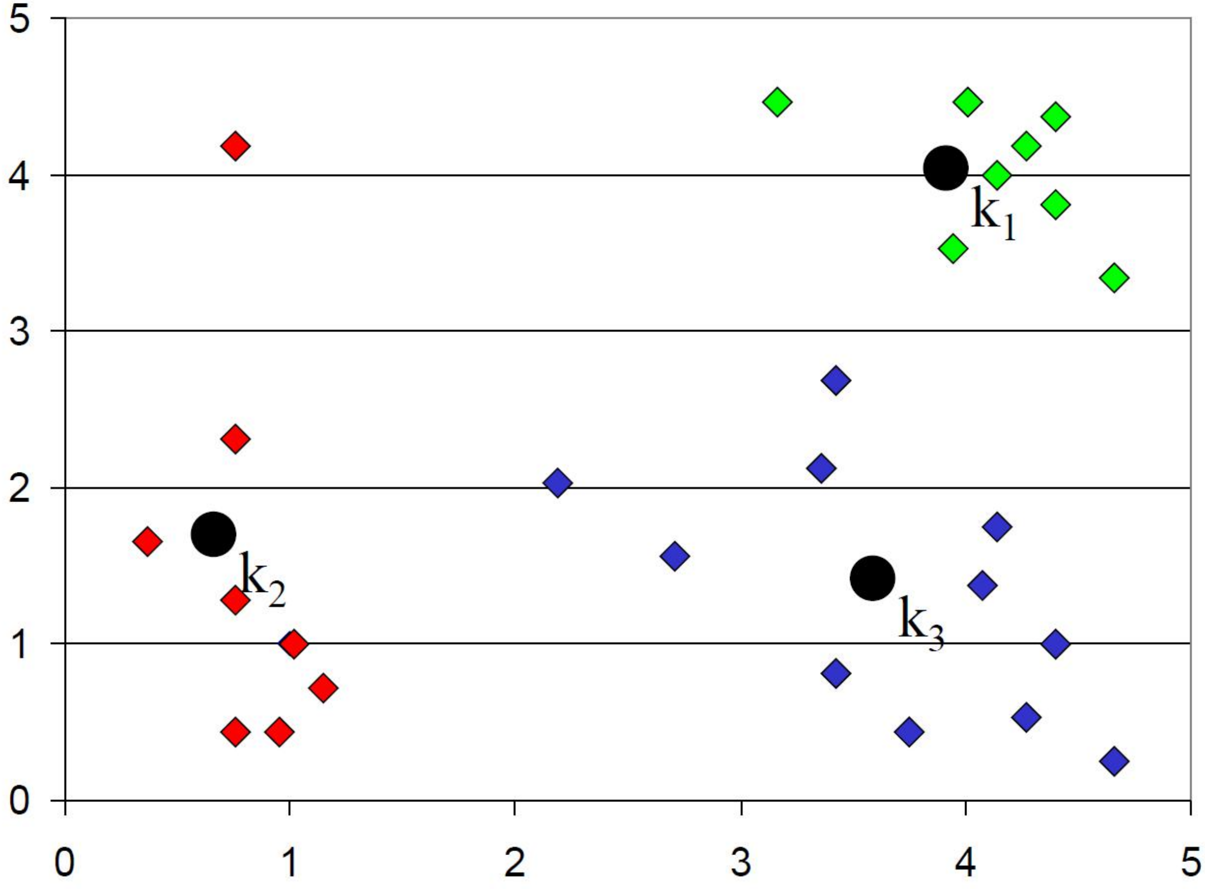
K-means Clustering: Iteration 2

After moving centers, re-assign the objects to nearest centers.
Move a center to the mean of its new members.



K-means Clustering: Finished

Re-assign and move centers, until
no objectds changed membership



Algorithm K-means

1. Decide on a value of K, the number of clusters.
2. Initialize the K cluster centers (randomly, if necessary)
3. Decide the class memberships of the N objects by assigning them to the nearest cluster center.
4. Re-estimate the K cluster centers, by assuming the memberships found above are correct.
5. Repeat 3 and 4 until none of the N objects changed membership in the last iteration.

Algorithm K-means

Objective

$$\min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

1. Fix μ, optimize C:

$$\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 = \min_{c} \sum_{i}^{n} \left|x_i - \mu_{x_i}\right|^2$$

Step 1 of kmeans

2. Fix C, optimize μ:

$$\min_{\mu} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

– Take partial derivative of μ_i and set to zero, we have

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

Step 2 of kmeans

Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

Why K-means Works

- Guaranteed to converge in a finite number of iterations

- Running time per iteration:
    1. Assign data points to closest cluster center

       O(KN) time

    2. Change the cluster center to the average of its assigned points

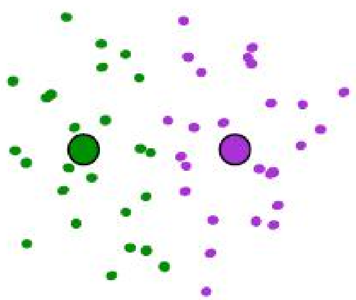
       O(N)

Summary: K-means

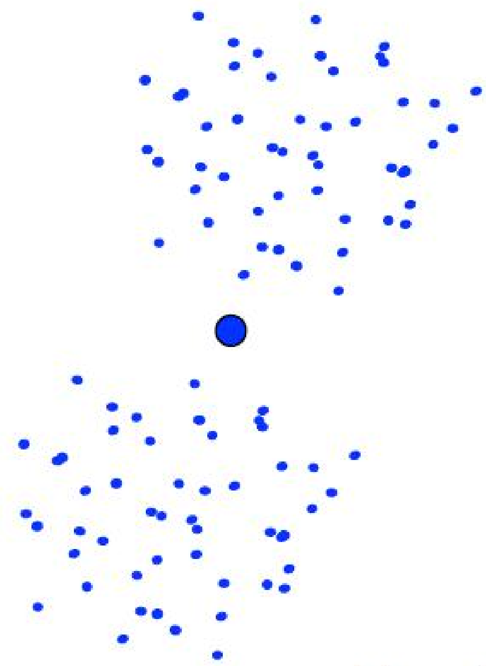
- Strengths
  - *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, $t << n$.
  - Simple, easy to implement.
- Weakness
  - Applicable only when mean is defined, then what about categorical data?
  - Often terminates at a local optimum.
  - Need to specify *K,* the number of clusters, in advance.
  - Unable to handle noisy data and outliers.
  - Not suitable to discover clusters with non-convex shapes.

Summary: K-means

A local optimum:



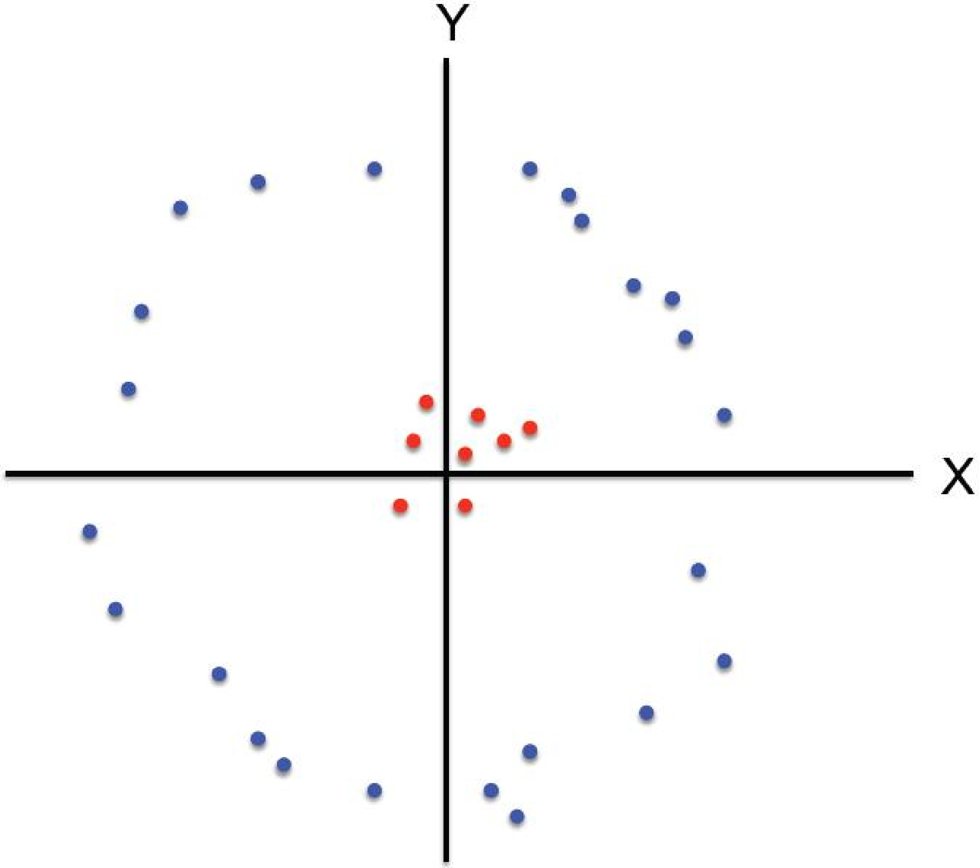
Would be better to have
one cluster here



… and two clusters here

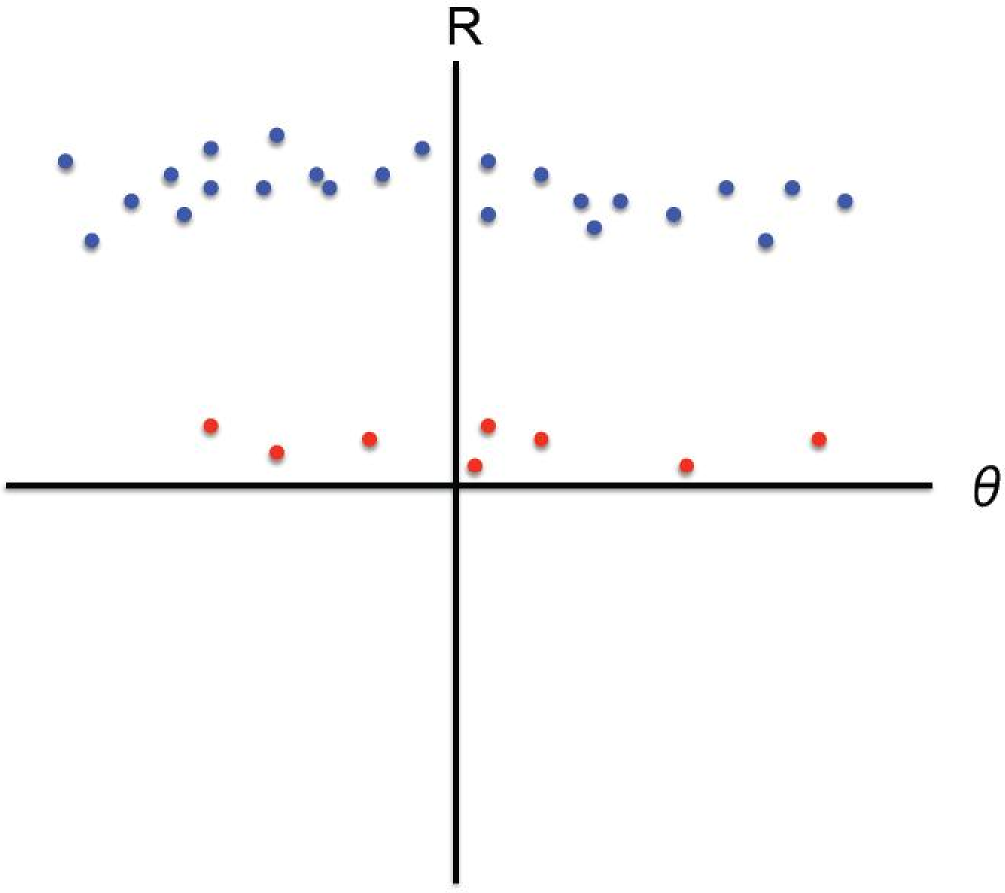
Summary: K-means

K-means not able to properly cluster



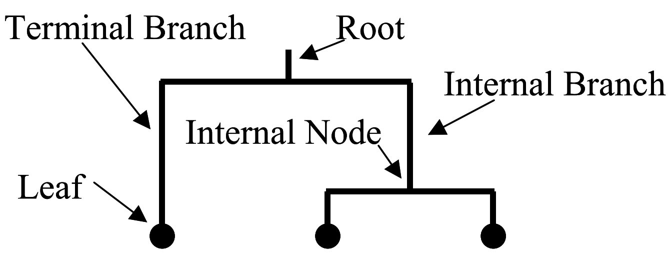
Summary: K-means

Changing the features (distance function) can help

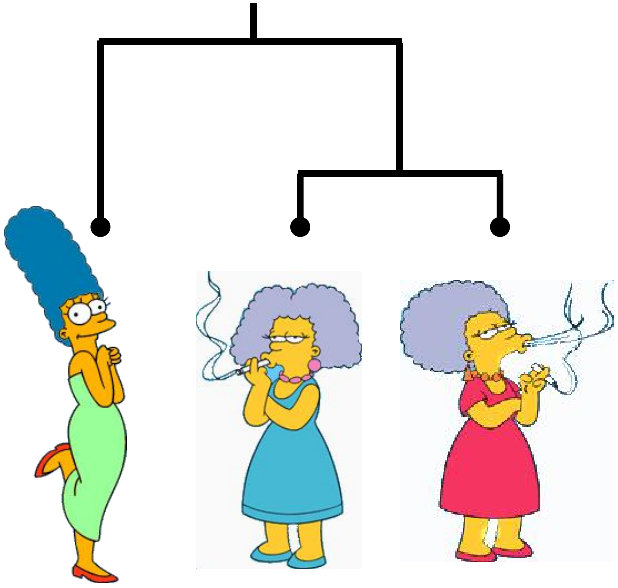


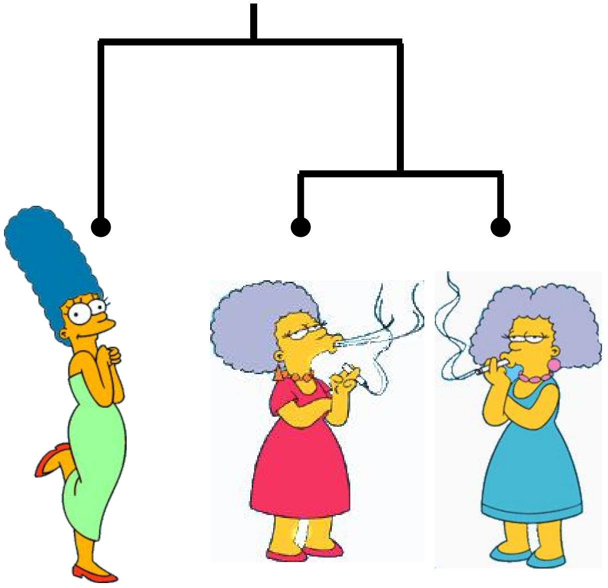
Hierarchical Clustering

Dendrogram: A Useful Tool for Summarizing Similarity Measurements



The similarity between two objects in a dendrogram is represented as the height of the lowest internal node they share.



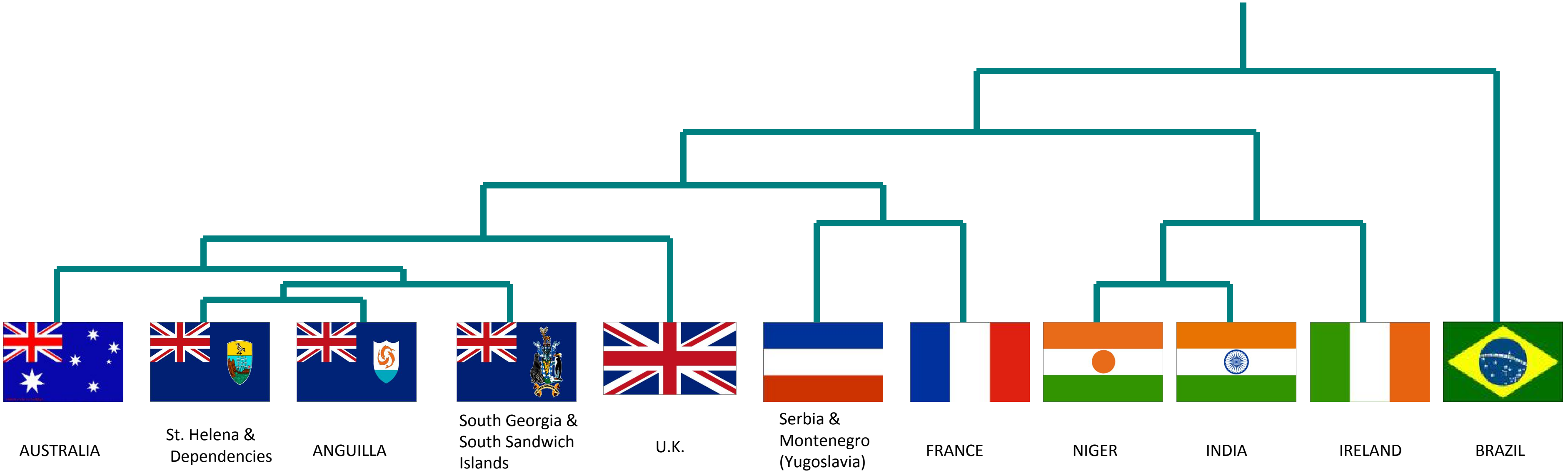


Hierarchical Clustering

Hierarchal clustering can sometimes show patterns that are meaningless or spurious

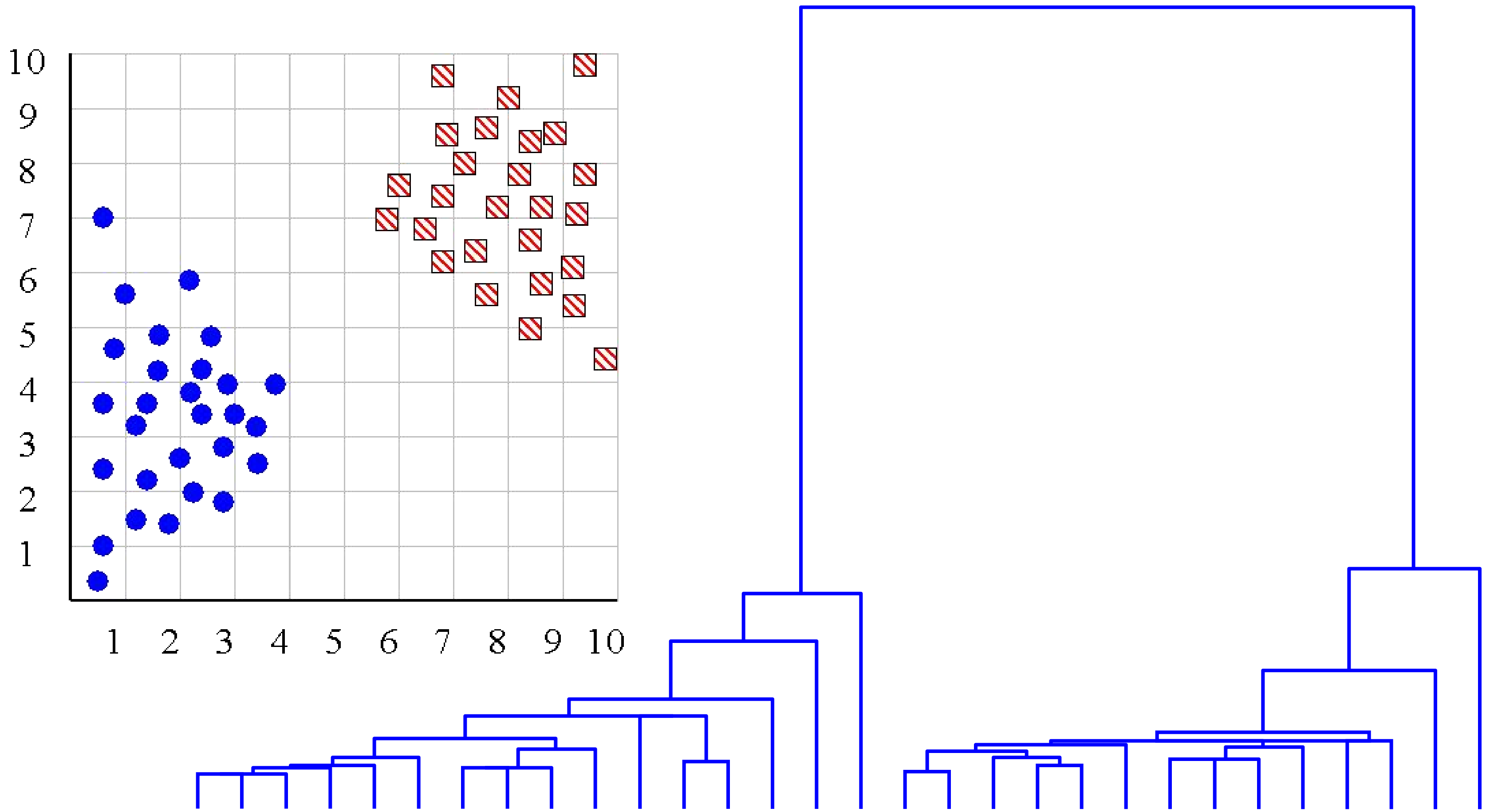
The tight grouping of Australia, Anguilla, St. Helena etc is meaningful; all these countries are former UK colonies

However the tight grouping of Niger and India is completely spurious; there is no connection between the two.



Hierarchical Clustering

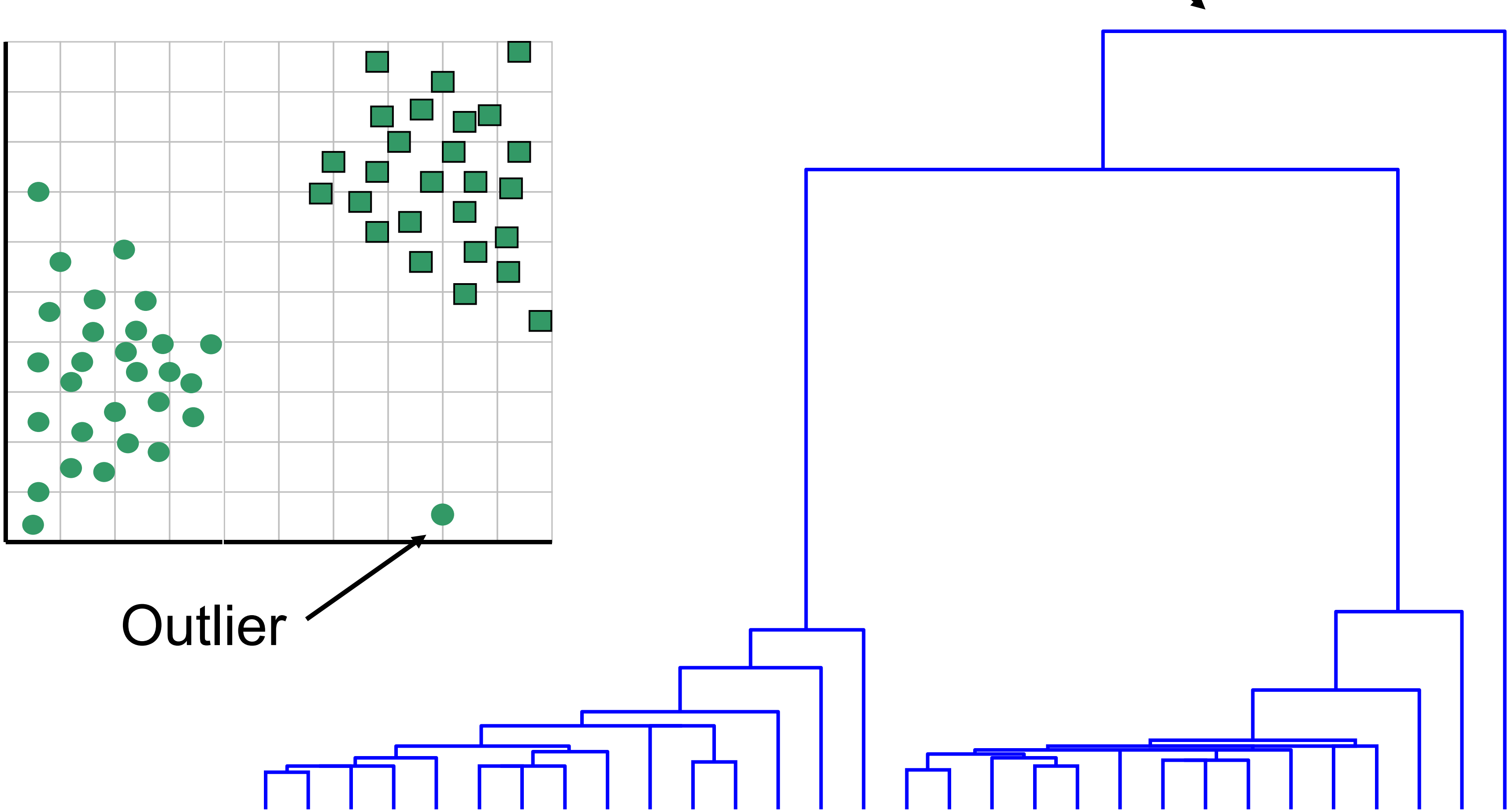
We can look at the dendrogram to determine the “correct” number of clusters.



Hierarchical Clustering

One potential use of a dendrogram: detecting outliers

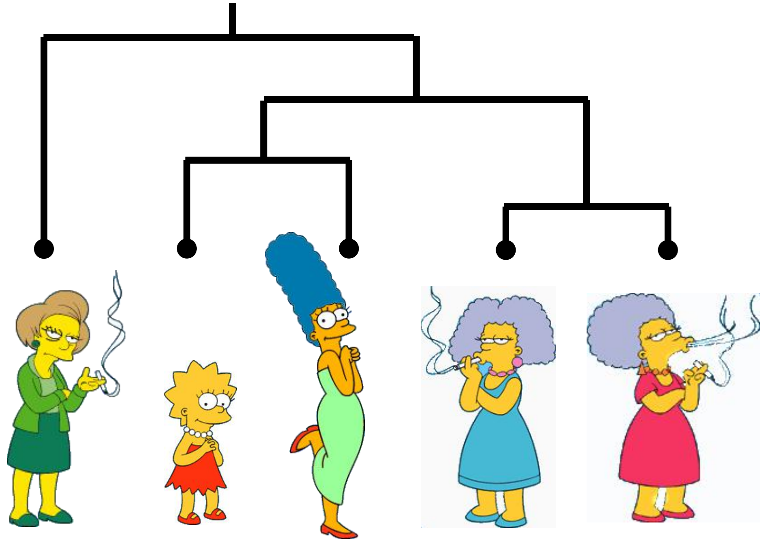
The single isolated branch is suggestive of a data point that is very different to all others



Hierarchical Clustering

The number of dendrograms with n leafs $= (2n-3)!/[(2^{(n-2)})\,(n-2)!]$

Number of Leafs	Number of Possible Dendrograms
2	1
3	3
4	15
5	105
…	…
10	34,459,425



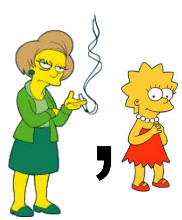
Since we cannot test all possible trees we will have to heuristic search of all possible trees. We could do this..

Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

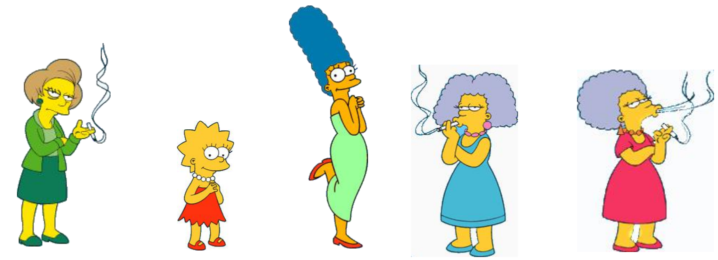
Top-Down (divisive): Starting with all the data in a single cluster, consider every possible way to divide the cluster into two. Choose the best division and recursively operate on both sides.

Hierarchical Clustering

We begin with a distance matrix which contains the distances between every pair of objects in our database.

D() = 8

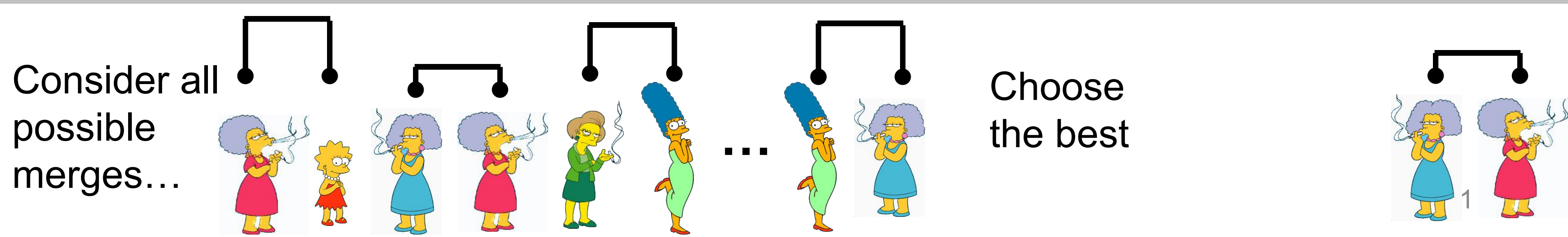
D() = 1



0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

Hierarchical Clustering

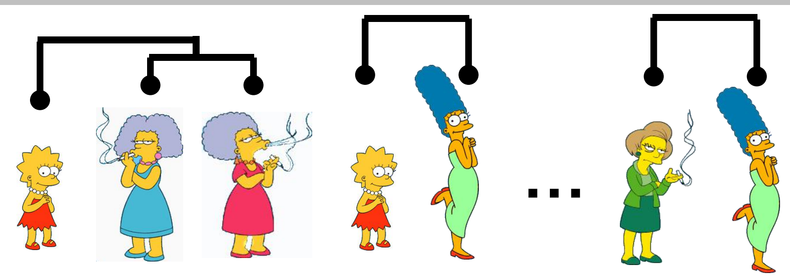
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.



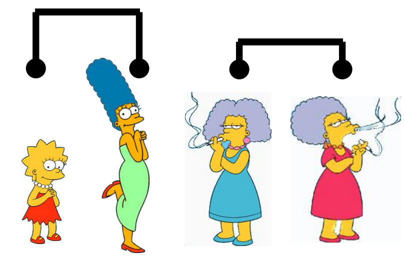
Hierarchical Clustering

Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

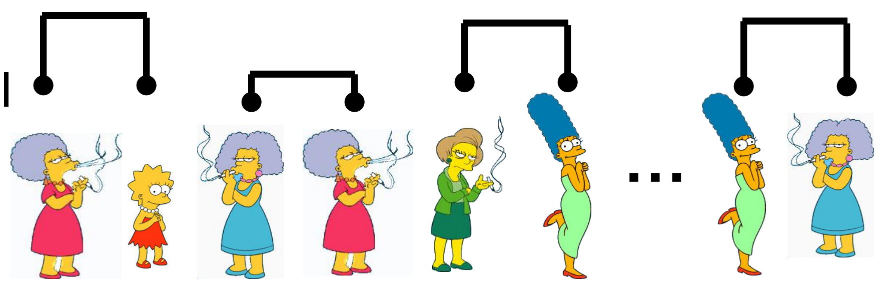
Consider all possible merges…



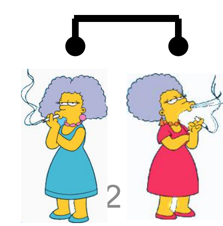
Choose the best



Consider all possible merges…

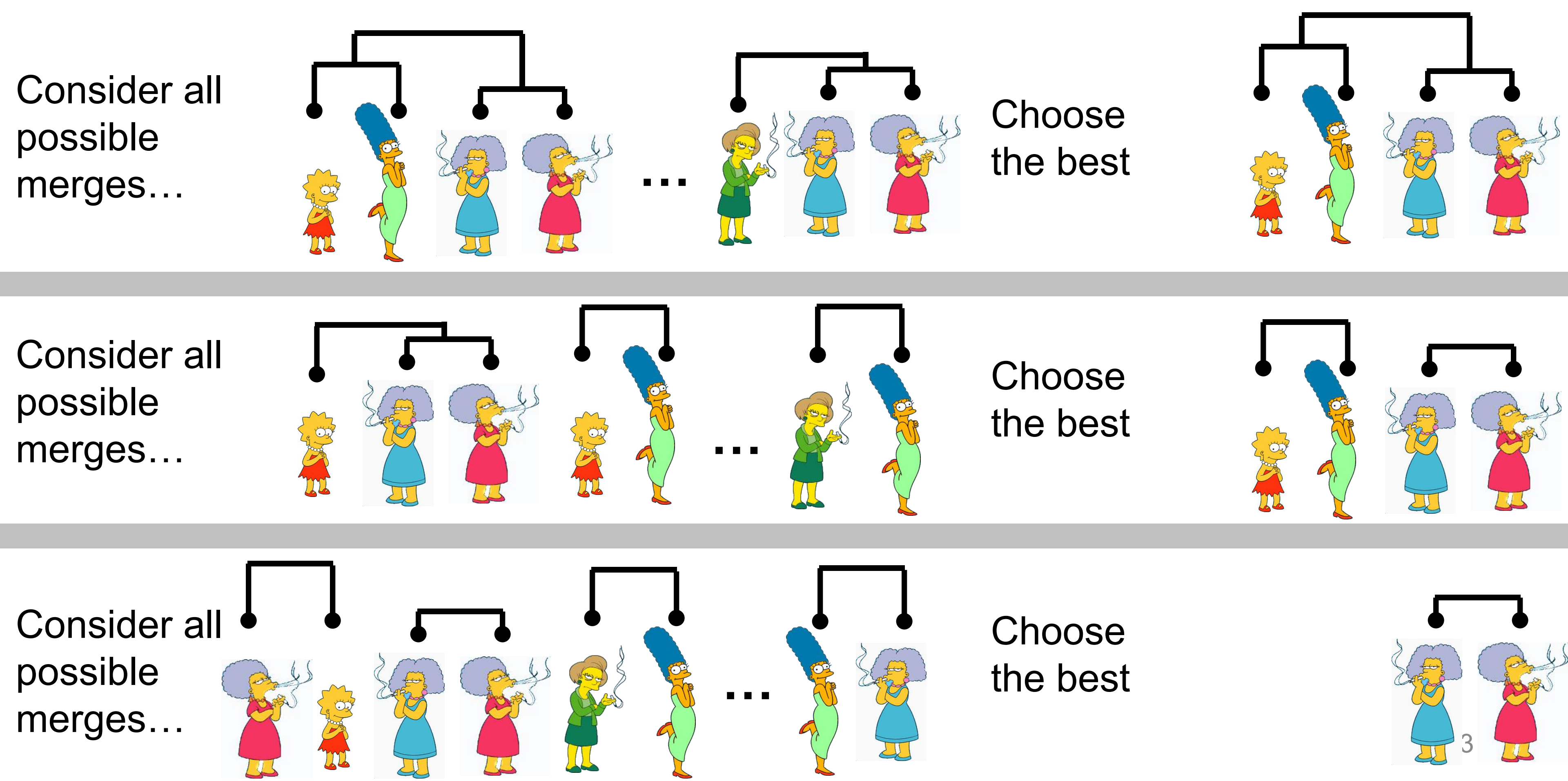


Choose the best



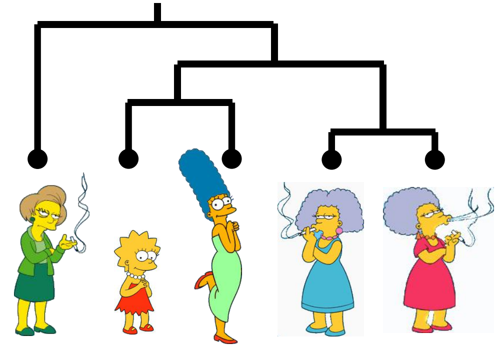
Hierarchical Clustering

Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

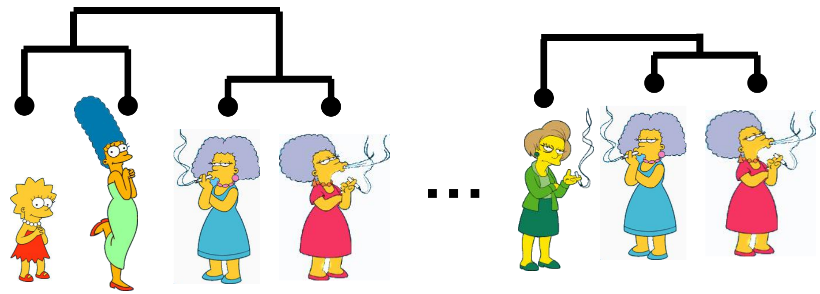


Hierarchical Clustering

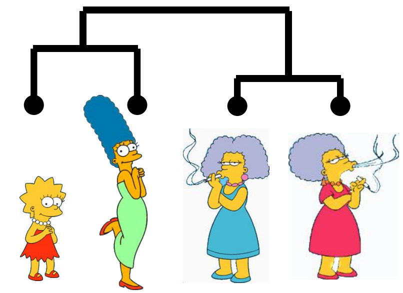
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.



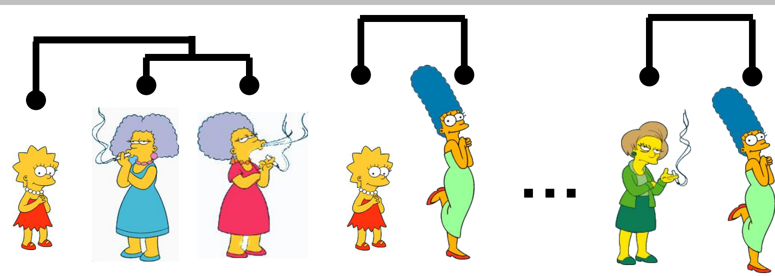
Consider all possible merges…



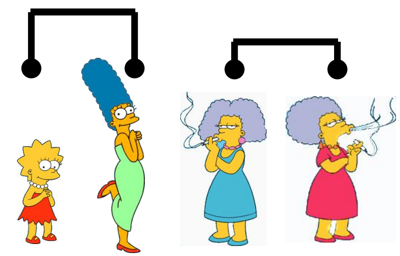
Choose the best



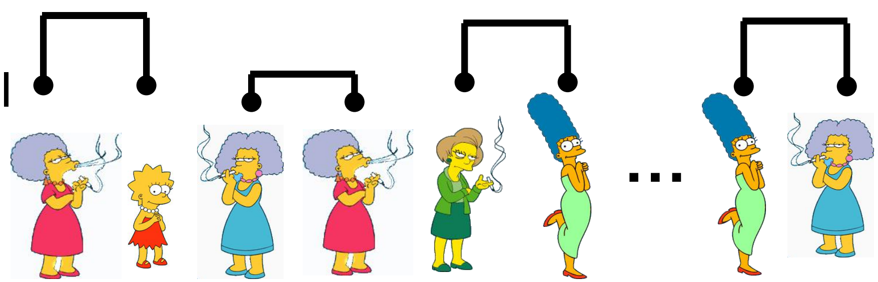
Consider all possible merges…



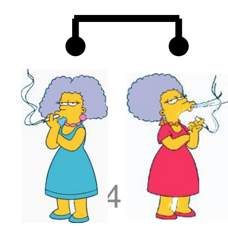
Choose the best



Consider all possible merges…



Choose the best



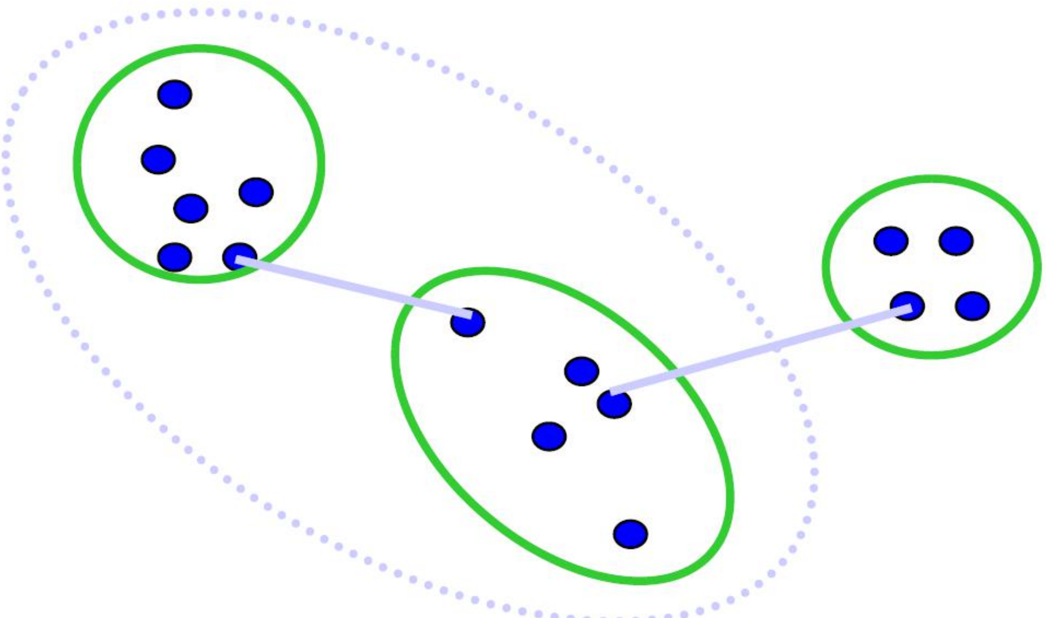
Hierarchical Clustering

We know how to measure the distance between two objects, but defining the distance between an object and a cluster, or defining the distance between two clusters is non obvious.

- **Single linkage (nearest neighbor):** In this method the distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters.
- **Complete linkage (furthest neighbor):** In this method, the distances between clusters are determined by the greatest distance between any two objects in the different clusters (i.e., by the "furthest neighbors").
- **Group average linkage:** In this method, the distance between two clusters is calculated as the average distance between all pairs of objects in the two different clusters.

Computing distance between clusters: Single Link

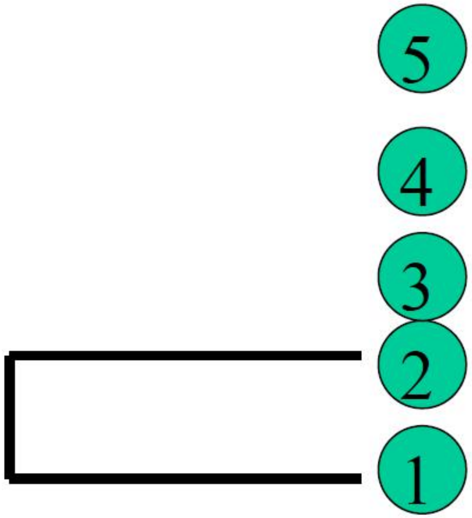
- cluster distance = distance of two closest members in each class



- Potentially long and skinny clusters

Example: Single Link

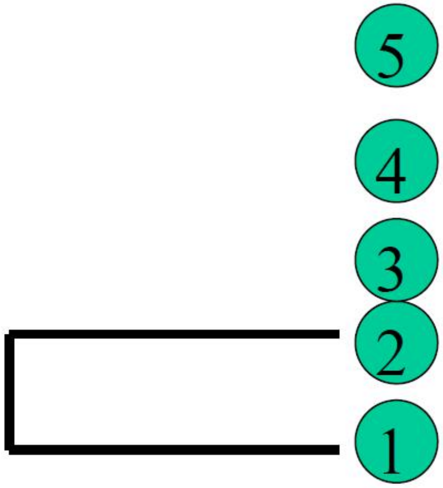
$$\begin{array}{c|c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & & & & \\ 2 & 0 & & & \\ 6 & 3 & 0 & & \\ 10 & 9 & 7 & 0 & \\ 9 & 8 & 5 & 4 & 0 \end{bmatrix} \end{array}$$



Example: Single Link

$$\begin{array}{c|c} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & & & & \\ 2 & 0 & & & \\ 6 & 3 & 0 & & \\ 10 & 9 & 7 & 0 & \\ 9 & 8 & 5 & 4 & 0 \end{bmatrix} \end{array} \Rightarrow \begin{array}{c|c} & \begin{matrix} (1,2) & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} (1,2) \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & & & \\ 3 & 0 & & \\ 9 & 7 & 0 & \\ 8 & 5 & 4 & 0 \end{bmatrix} \end{array}$$

$$d_{(1,2),3} = \min\{d_{1,3}, d_{2,3}\} = \min\{6,3\} = 3$$
$$d_{(1,2),4} = \min\{d_{1,4}, d_{2,4}\} = \min\{10,9\} = 9$$
$$d_{(1,2),5} = \min\{d_{1,5}, d_{2,5}\} = \min\{9,8\} = 8$$

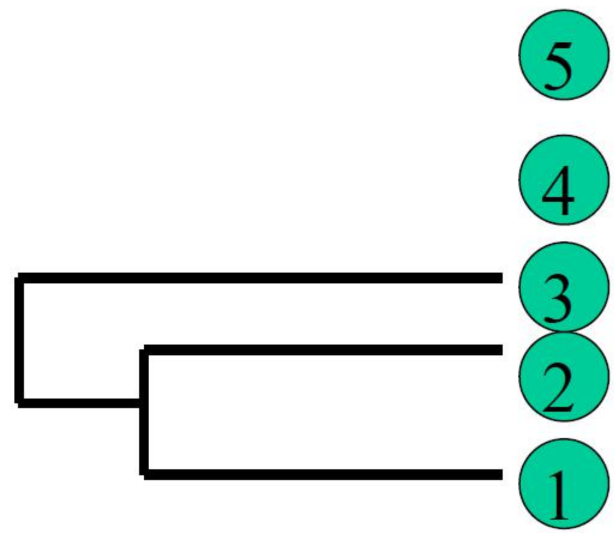


Example: Single Link

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & & & & \\ 2 & 2 & 0 & & & \\ 3 & 6 & 3 & 0 & & \\ 4 & 10 & 9 & 7 & 0 & \\ 5 & 9 & 8 & 5 & 4 & 0 \end{array} \Rightarrow \begin{array}{c|cccc} & (1,2) & 3 & 4 & 5 \\ \hline (1,2) & 0 & & & \\ 3 & 3 & 0 & & \\ 4 & 9 & 7 & 0 & \\ 5 & 8 & 5 & 4 & 0 \end{array} \Rightarrow \begin{array}{c|ccc} & (1,2,3) & 4 & 5 \\ \hline (1,2,3) & 0 & & \\ 4 & 7 & 0 & \\ 5 & 5 & 4 & 0 \end{array}$$

$$d_{(1,2,3),4} = \min\{d_{(1,2),4}, d_{3,4}\} = \min\{9,7\} = 7$$

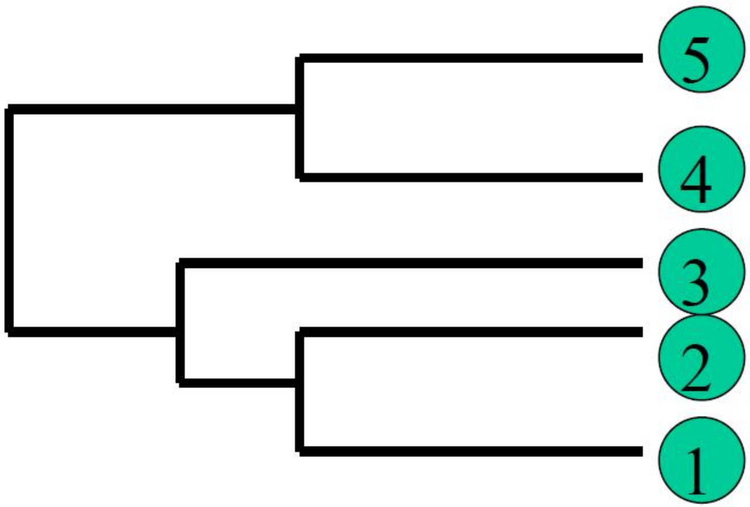
$$d_{(1,2,3),5} = \min\{d_{(1,2),5}, d_{3,5}\} = \min\{8,5\} = 5$$



Example: Single Link

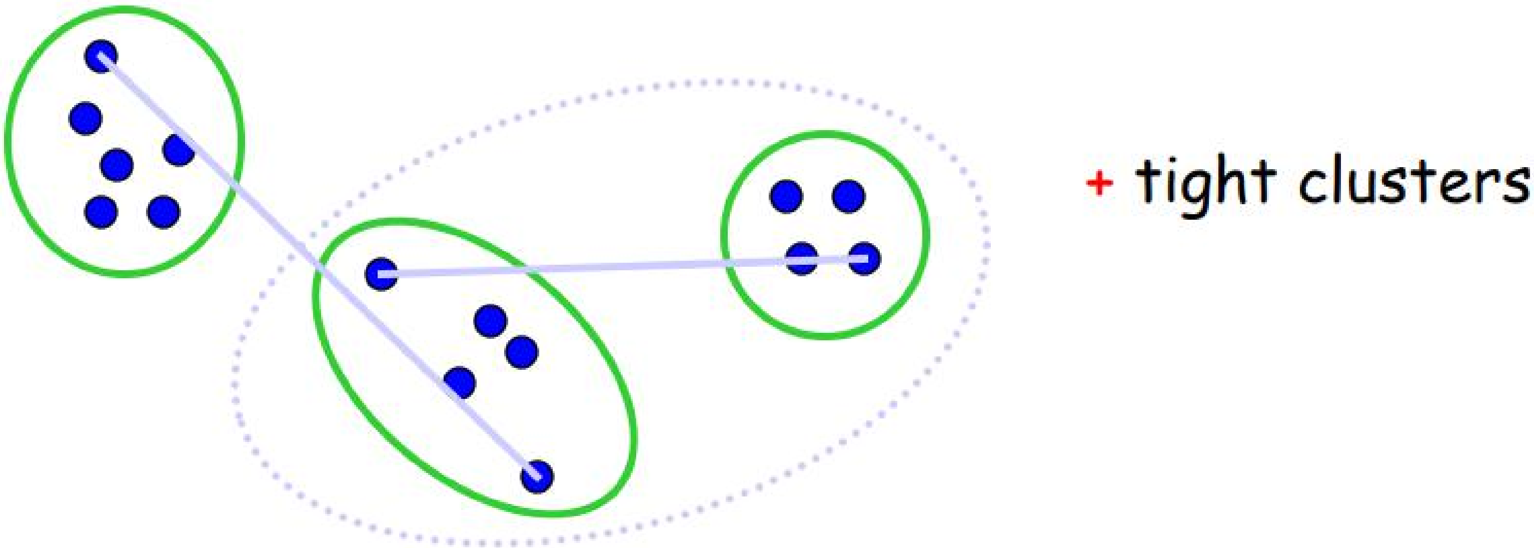
$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & & & & \\
2 & 2 & 0 & & & \\
3 & 6 & 3 & 0 & & \\
4 & 10 & 9 & 7 & 0 & \\
5 & 9 & 8 & 5 & 4 & 0
\end{array}
\Rightarrow
\begin{array}{c|cccc}
 & (1,2) & 3 & 4 & 5 \\
\hline
(1,2) & 0 & & & \\
3 & 3 & 0 & & \\
4 & 9 & 7 & 0 & \\
5 & 8 & 5 & 4 & 0
\end{array}
\Rightarrow
\begin{array}{c|ccc}
 & (1,2,3) & 4 & 5 \\
\hline
(1,2,3) & 0 & & \\
4 & 7 & 0 & \\
5 & 5 & 4 & 0
\end{array}
$$

$$d_{(1,2,3),(4,5)} = \min\{d_{(1,2,3),4}, d_{(1,2,3),5}\} = 5$$



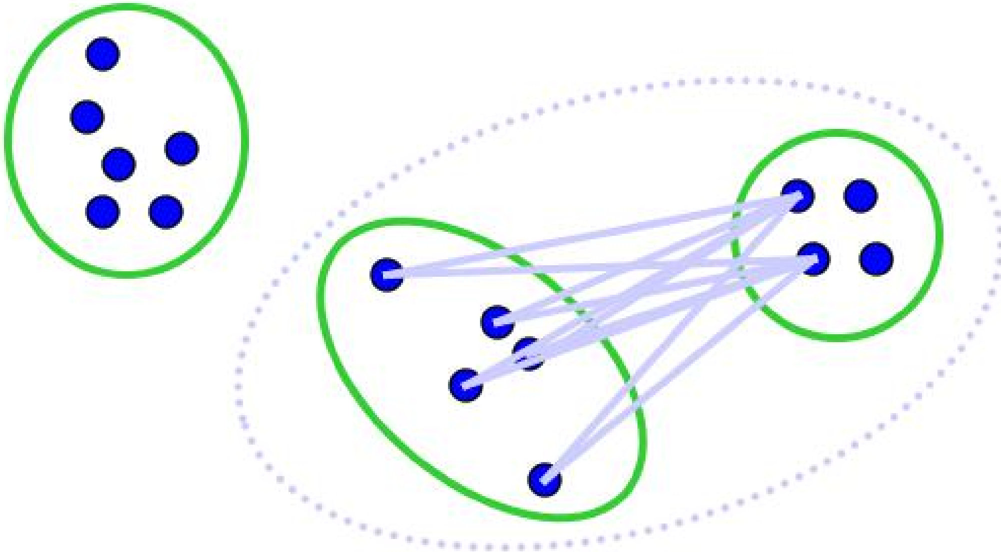
Computing distance between clusters: Complete Link

- cluster distance = distance of two farthest members



Computing distance between clusters: Average Link

- cluster distance = average distance of all pairs

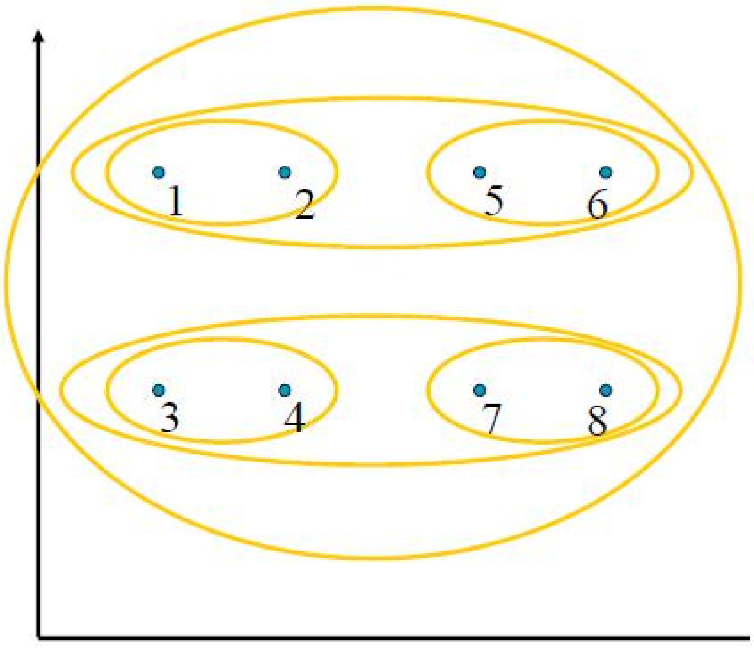


the most widely used measure

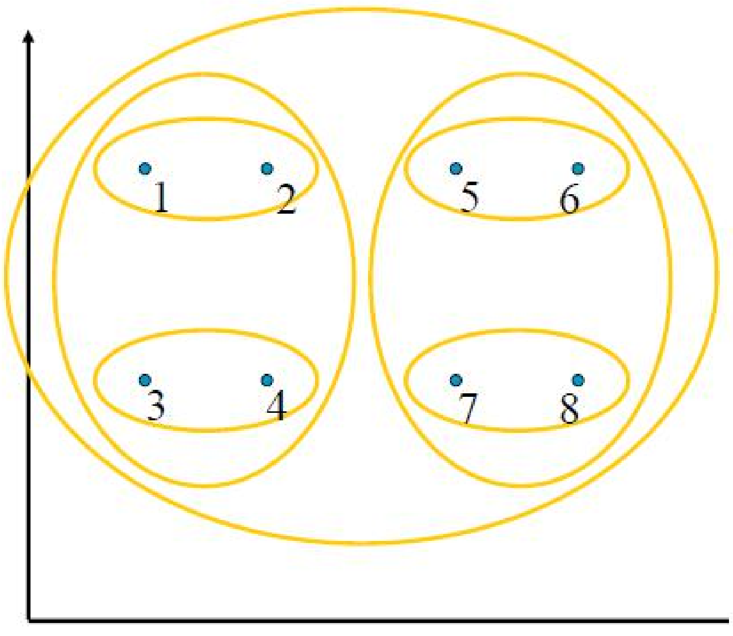
Robust against noise

Hierarchical Clustering

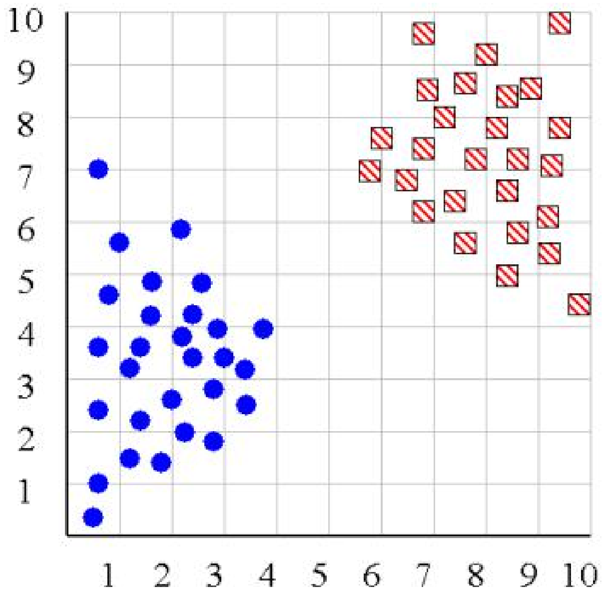
Single Link

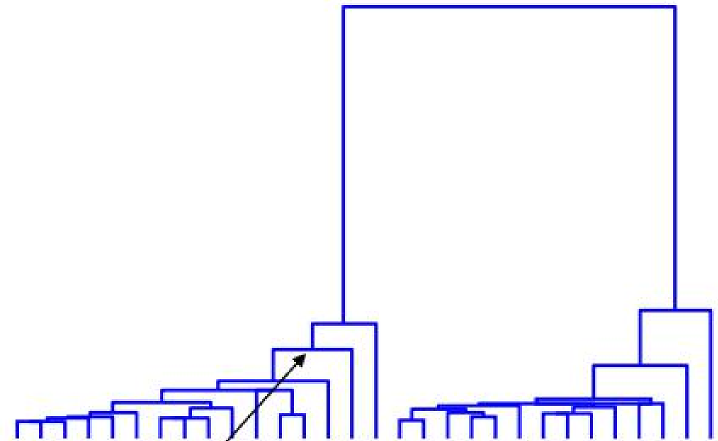


Complete Link

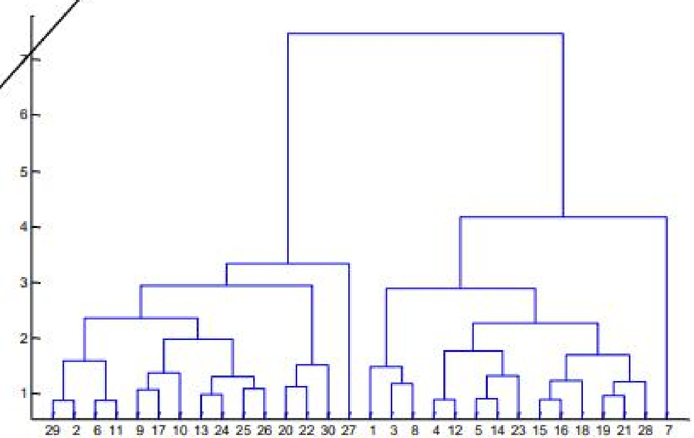


Hierarchical Clustering





Single linkage



Average linkage

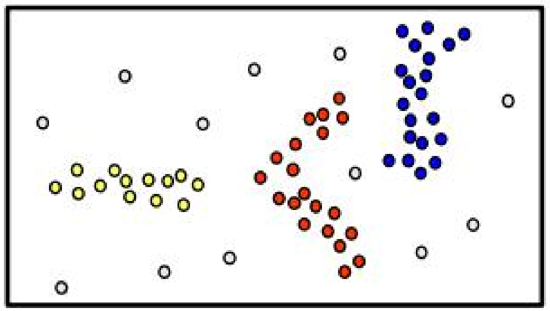
Height represents distance between objects/clusters

Summary: Hierarchical Clustering

- No need to specify the number of clusters in advance
- Hierarchal nature maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least $O(n^2)$, where *n* is the number of total objects
- Like any heuristic search algorithms, local optima are a problem
- Interpretation of results is (very) subjective

Density-based Clustering

- **Basic idea**
  - Clusters are dense regions in the data space, separated by regions of lower object density
  - A cluster is defined as a maximal set of density-connected points
  - Discovers clusters of arbitrary shape
- **Method**
  - DBSCAN

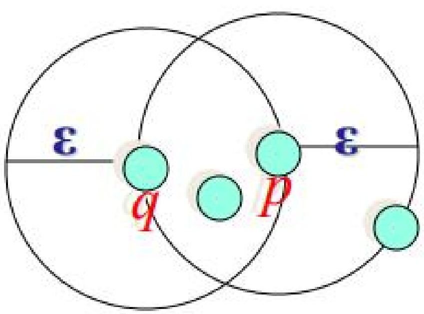


Density Definition

- ε-Neighborhood – Objects within a radius of ε from an object.

$$N_{\varepsilon}(p):\{q \mid d(p,q) \leq \varepsilon\}$$

- "High density" - ε-Neighborhood of an object contains at least *MinPts* of objects.

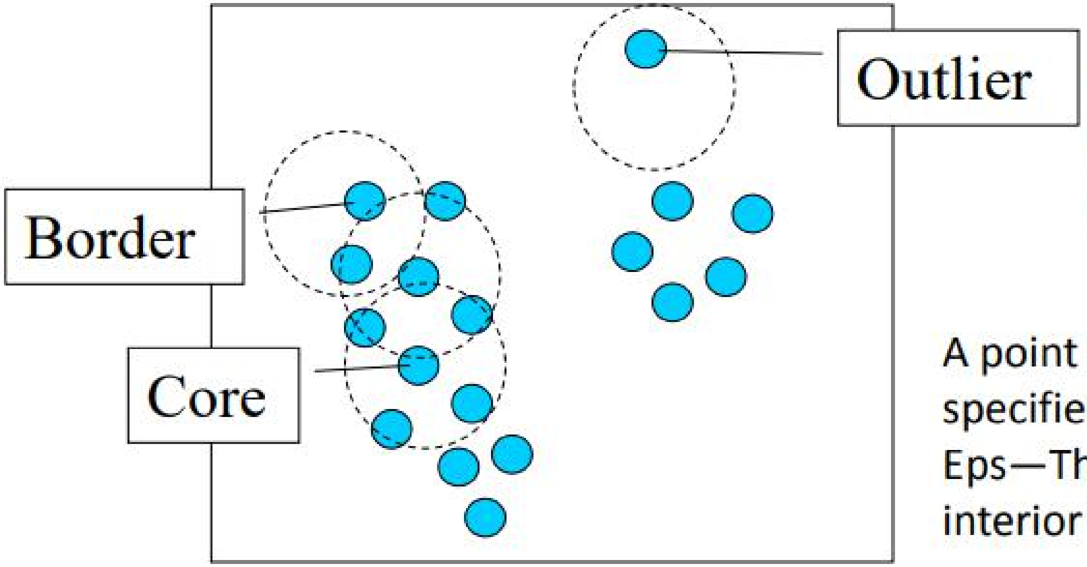


ε-Neighborhood of *p*
ε-Neighborhood of *q*

Density of p is "high" (MinPts = 4)

Density of q is "low" (MinPts = 4)

Core, Border & Outlier



Given ε and *MinPts*, categorize the objects into three exclusive groups.

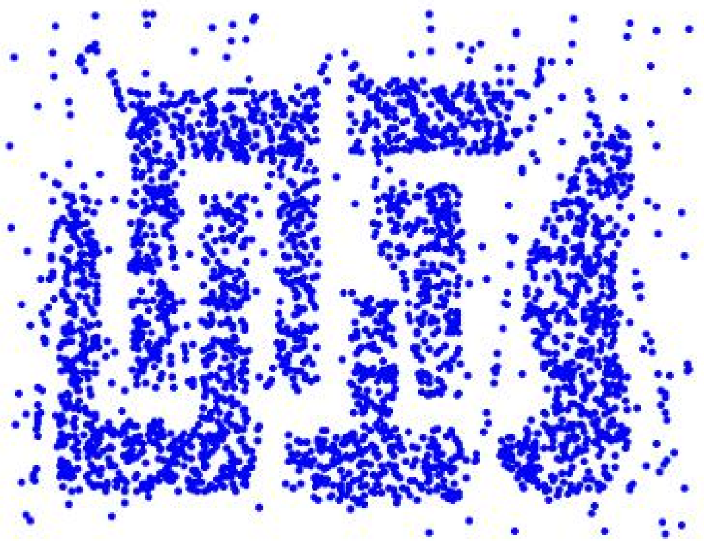
A point is a core point if it has more than a specified number of points (MinPts) within Eps—These are points that are at the interior of a cluster.

ε = 1unit, MinPts = 5

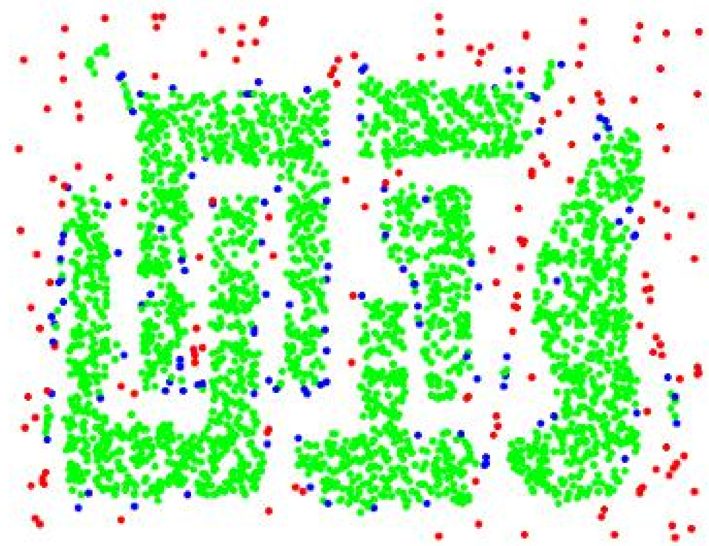
A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

A noise point is any point that is not a core point nor a border point.

Example



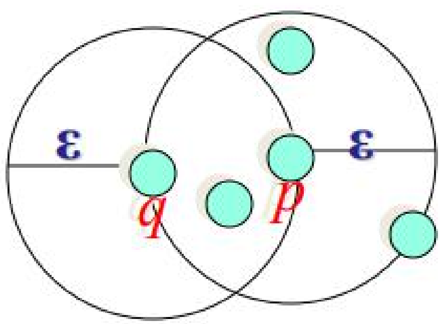
Original Points



Point types: core, border and outliers

Density-reachability

- Directly density-reachable
  - An object *q* is directly density-reachable from object *p* if *p* is a core object and *q* is in p's ε-neighborhood.

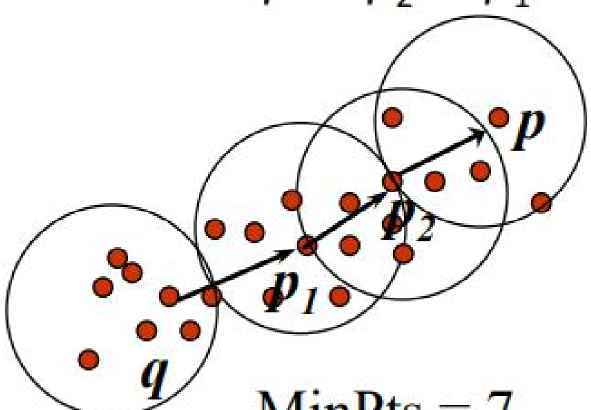


MinPts = 4

- *q* is directly density-reachable from *p*
- *p* is not directly density-reachable from *q*
- Density-reachability is asymmetric

Density-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

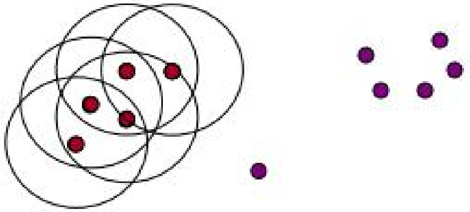


MinPts = 7

- p is (indirectly) density-reachable from q
- q is not density-reachable from p

DBSCAN Algorithm: Example

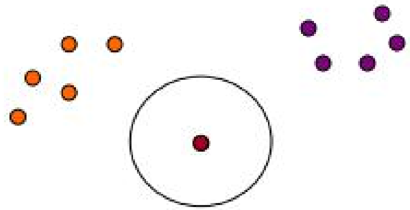
- **Parameter**
  - $\varepsilon = 2$ cm
  - $MinPts = 3$



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

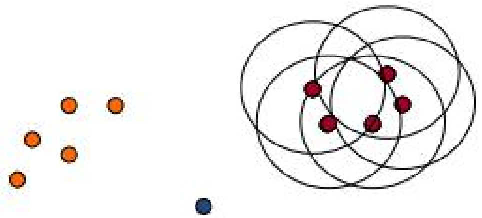
- **Parameter**
  - ε = 2 cm
  - *MinPts* = 3



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

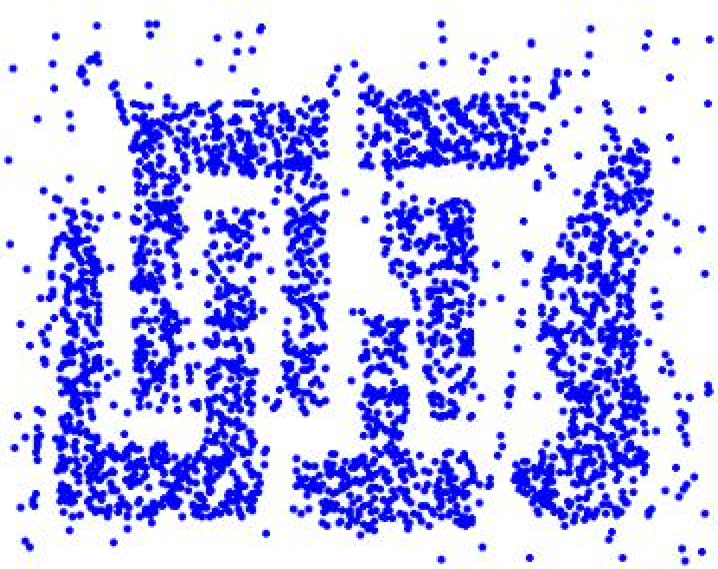
DBSCAN Algorithm: Example

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

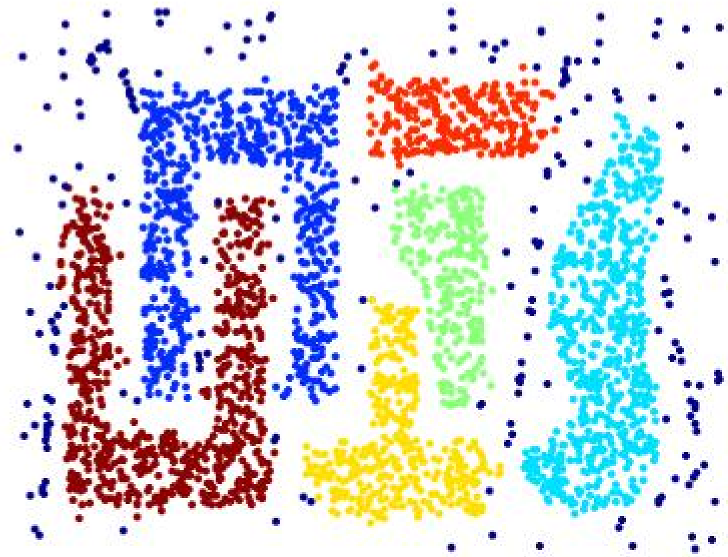


for each $o \in D$ **do**
    if o is not yet classified **then**
        if o is a core-object **then**
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE

When DBSCAN Works Well



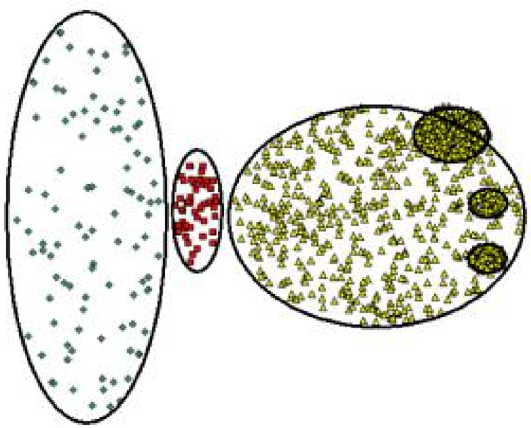
Original Points



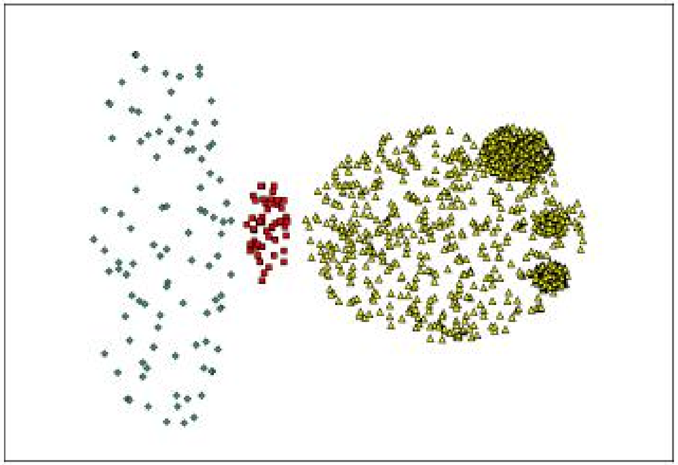
Clusters

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

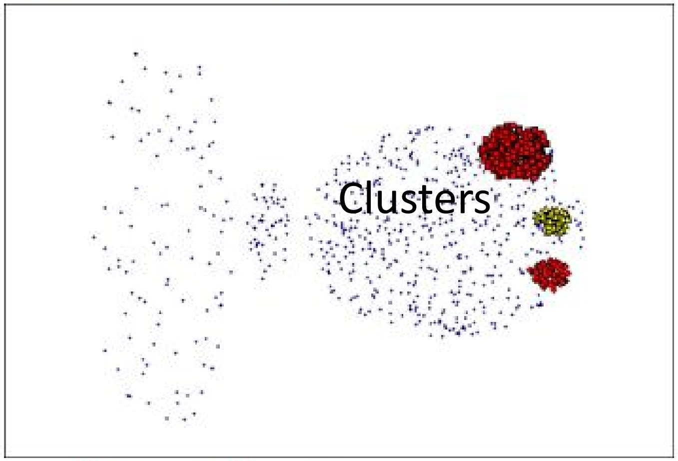
When DBSCAN Does Not Works Well



Original Points



(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

- **Cannot handle varying densities**
- **Sensitive to parameters - hard to determine the correct set of parameters**