Un-Supervised Learning

Clustering Techniques

Hierarchical & Kmeans

Credits : Some of the slides are taken from:

Data Analytics (CS40003) by Dr. Debasis Samanta

CS 578: Statistical Machine Learning by Yexiang Xue

Introduction to data mining, by Tan, Steinbach, and Kumar

University at buffalo NY state University

CSE 185 Introduction to Computer Vision
University of California MERCED

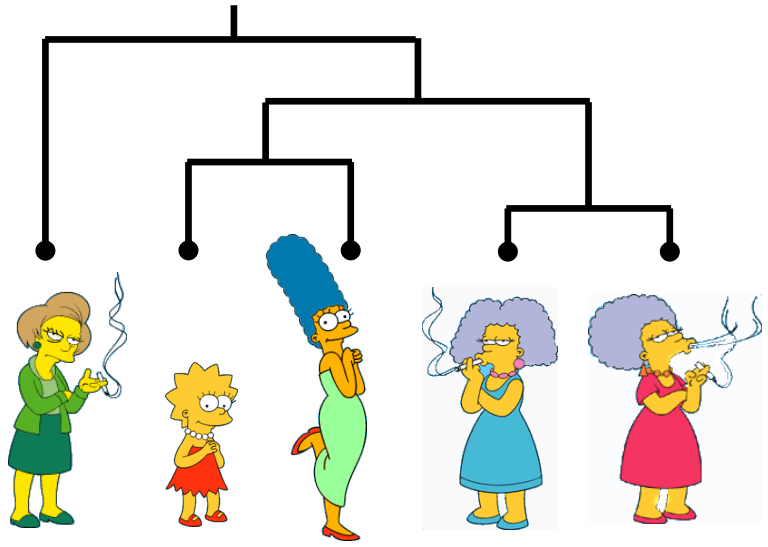
Usman RoshanQ Assaf Gottlieb

Slide based on one by Eamonn Keogh

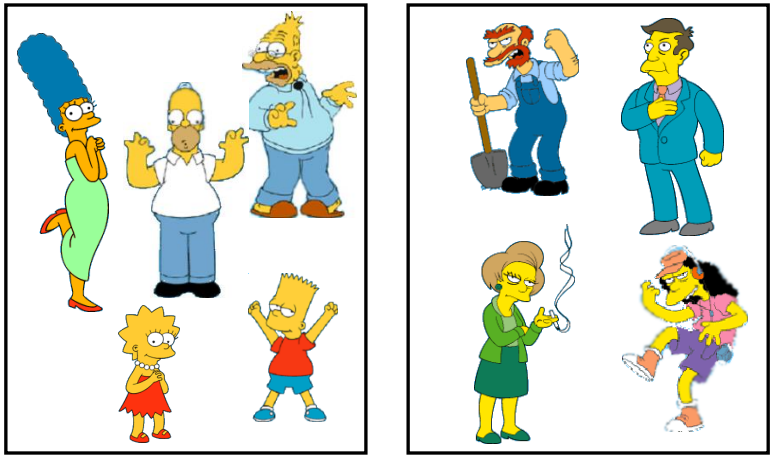
Two Types of Clustering

- **Partitional algorithms:** Construct various partitions and then evaluate them by some criterion
- **Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion

Hierarchical

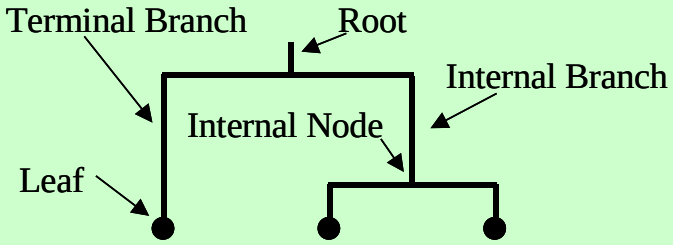


Partitional

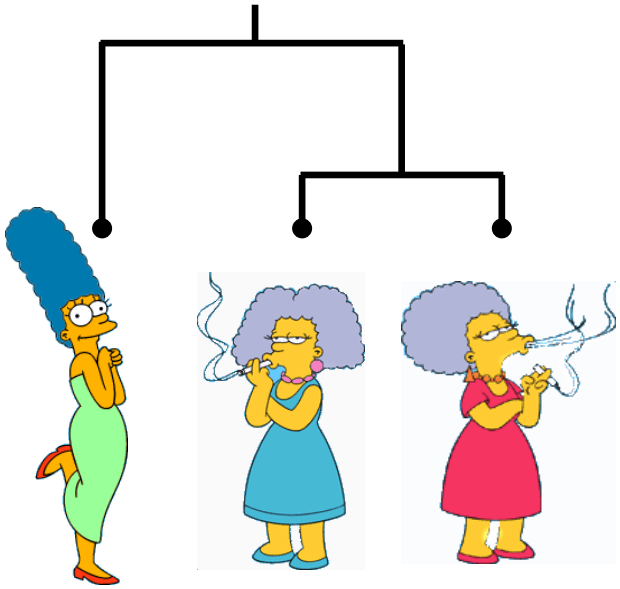


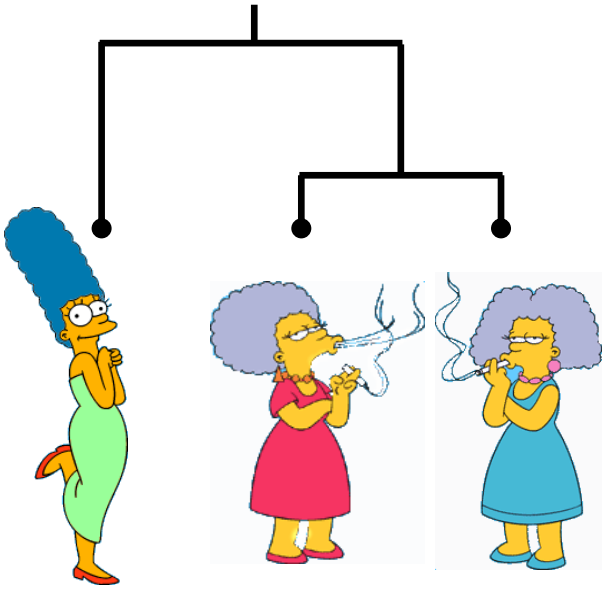
Slide based on one by Eamonn Keogh

Dendogram: A Useful Tool for Summarizing Similarity Measurements



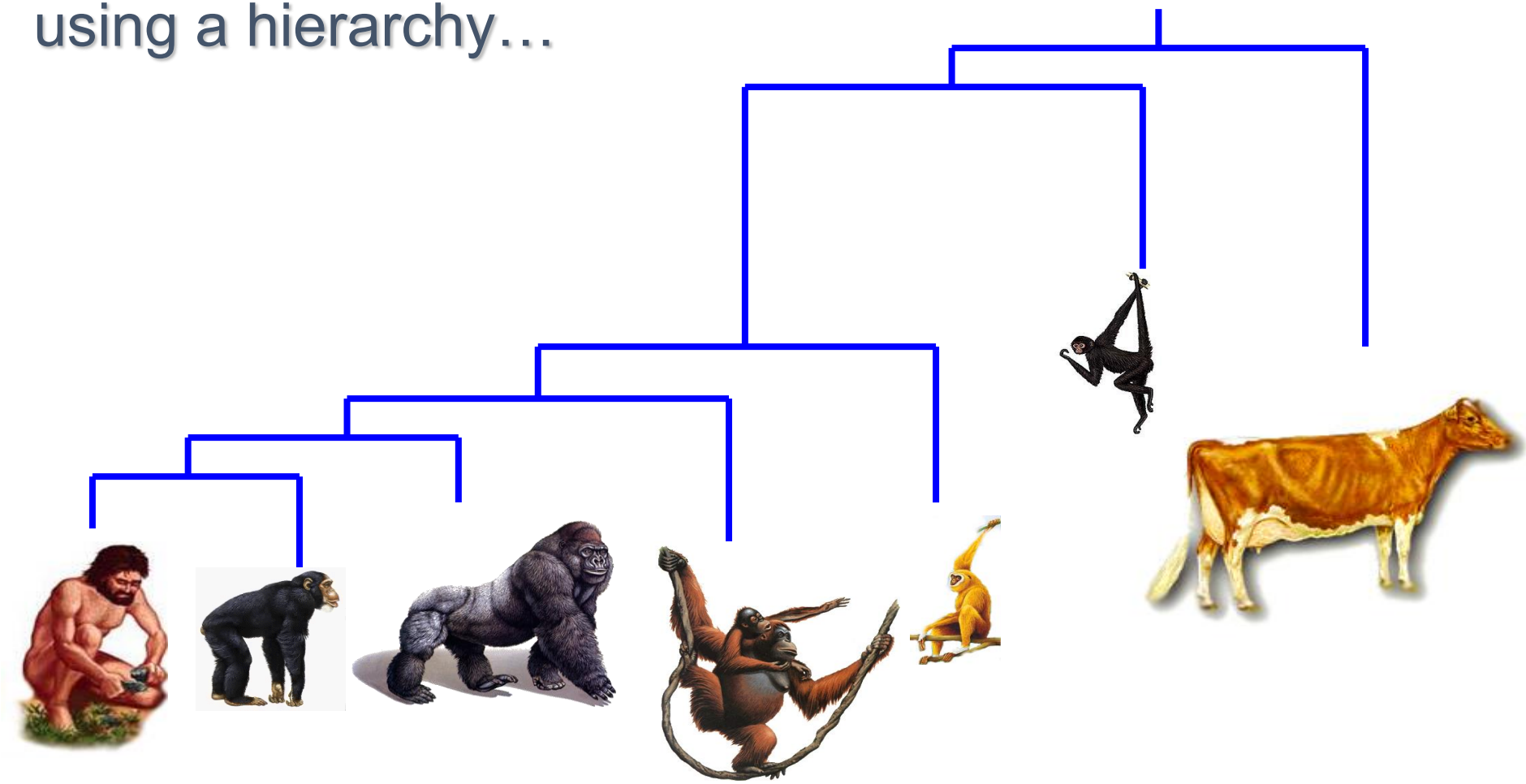
The similarity between two objects in a dendrogram is represented as the height of the lowest internal node they share.





Slide based on one by Eamonn Keogh

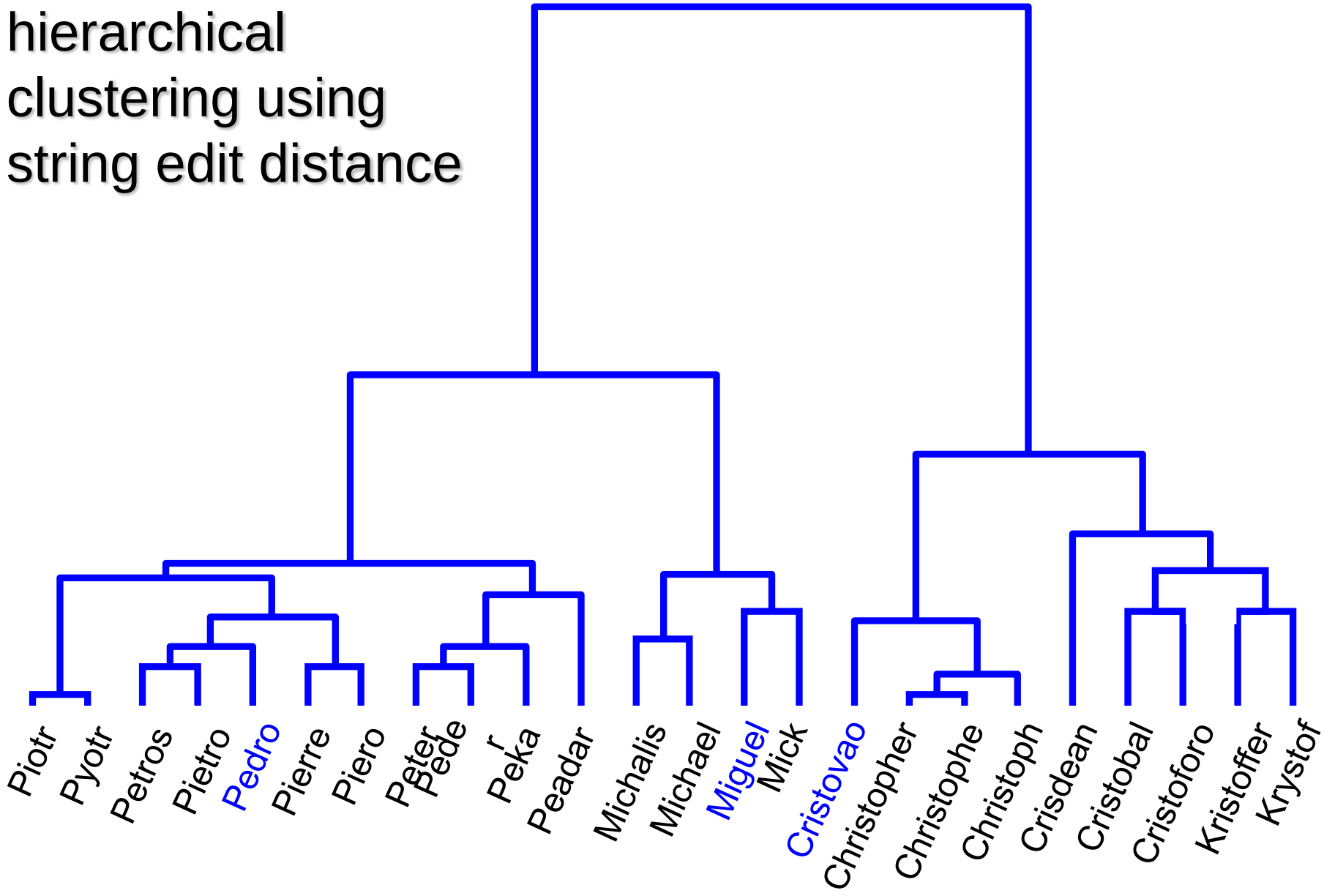
There is only one dataset that can be perfectly clustered using a hierarchy…



(Bovine:0.69395, (Spider Monkey 0.390, (Gibbon:0.36079,(Orang:0.33636,(Gorilla:0.17147,(Chimp:0.19268, Human:0.11927):0.08386):0.06124):0.15057):0.54939);

A demonstration of hierarchical clustering using string edit distance

Slide based on one by Eamonn Keogh

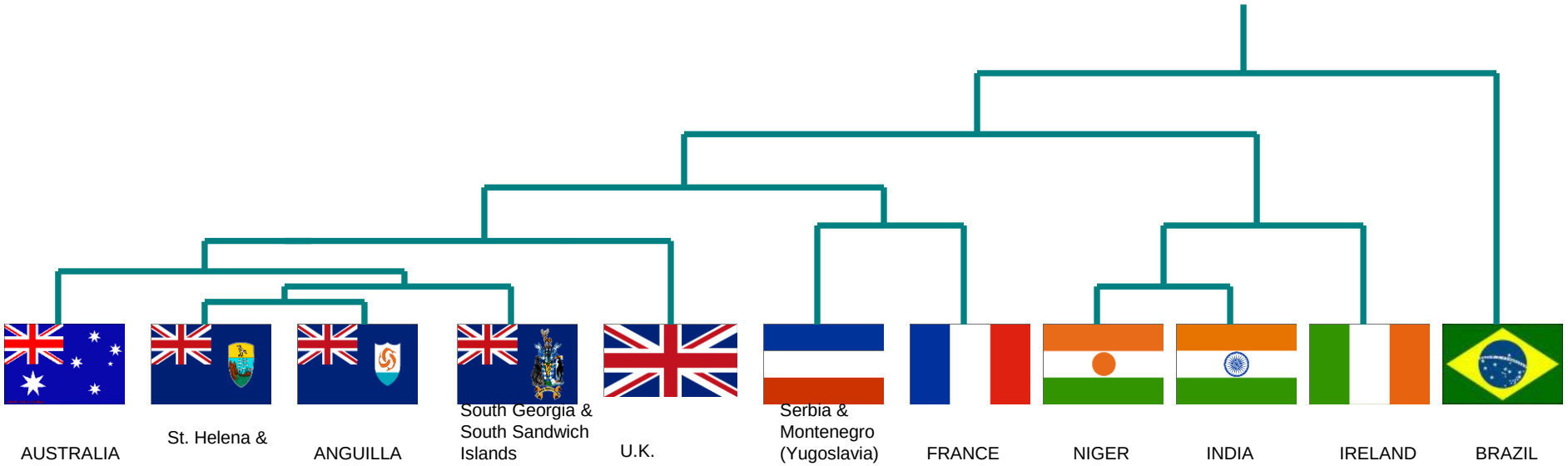


Slide based on one by Eamonn Keogh

Hierarchal clustering can sometimes show patterns that are meaningless or spurious

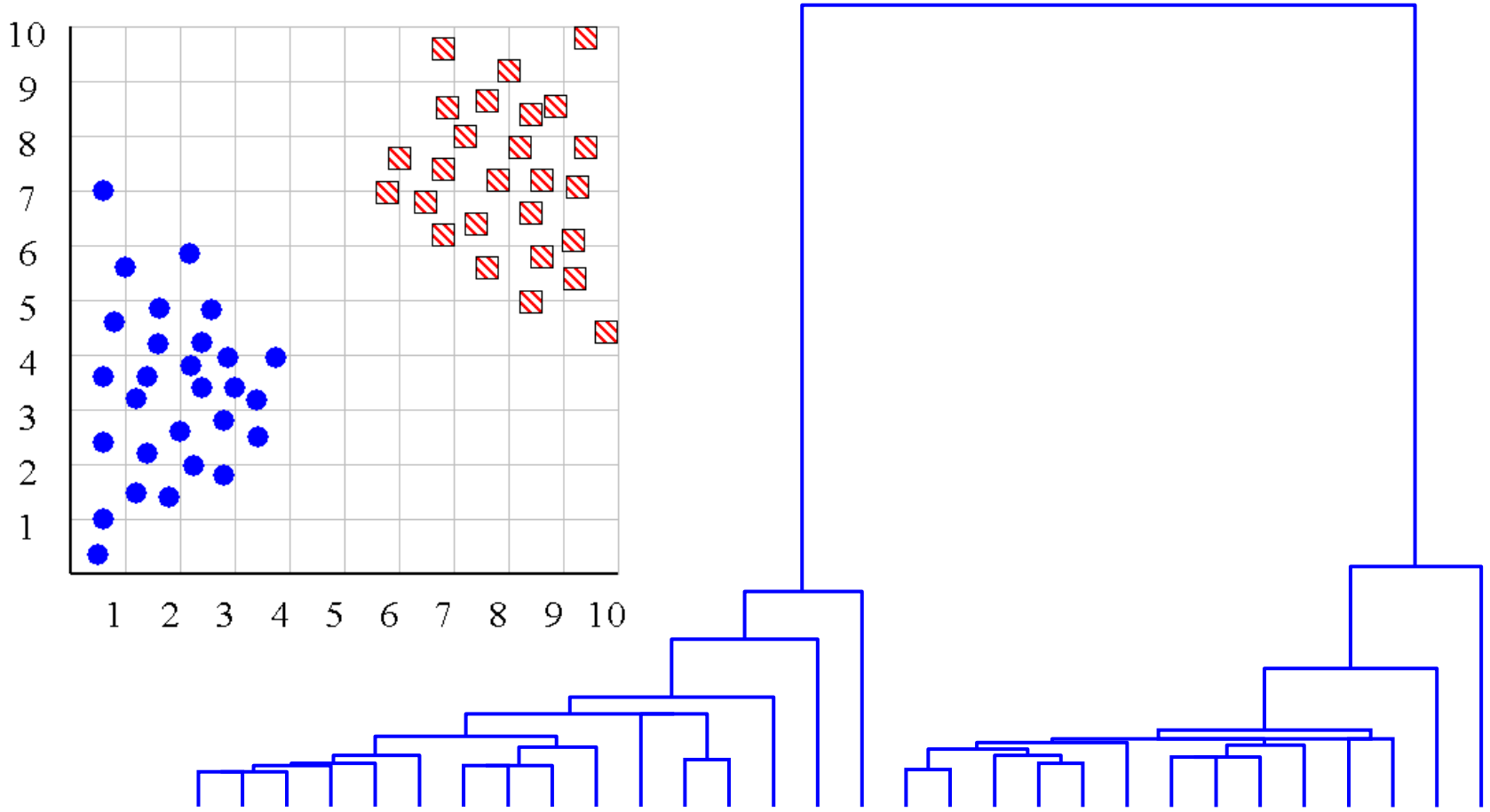
The tight grouping of Australia, Anguilla, St. Helena etc is meaningful; all these countries are former UK colonies

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



Slide based on one by Eamonn Keogh

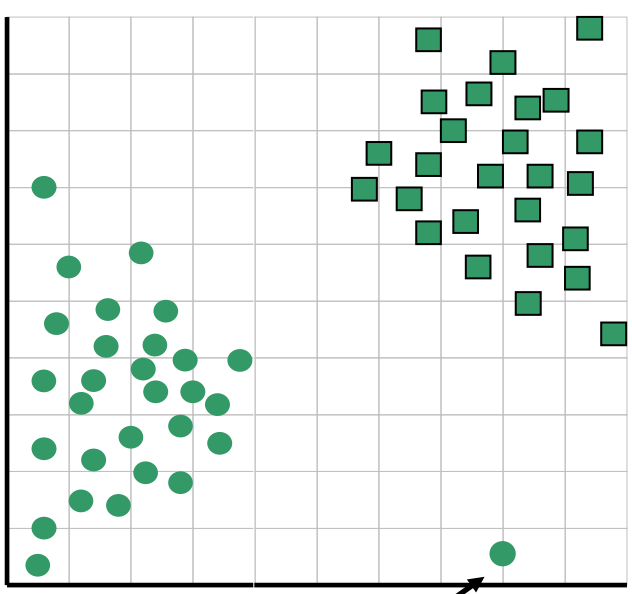
We can look at the dendrogram to determine the “correct” number of clusters.



Slide based on one by Eamonn Keogh

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



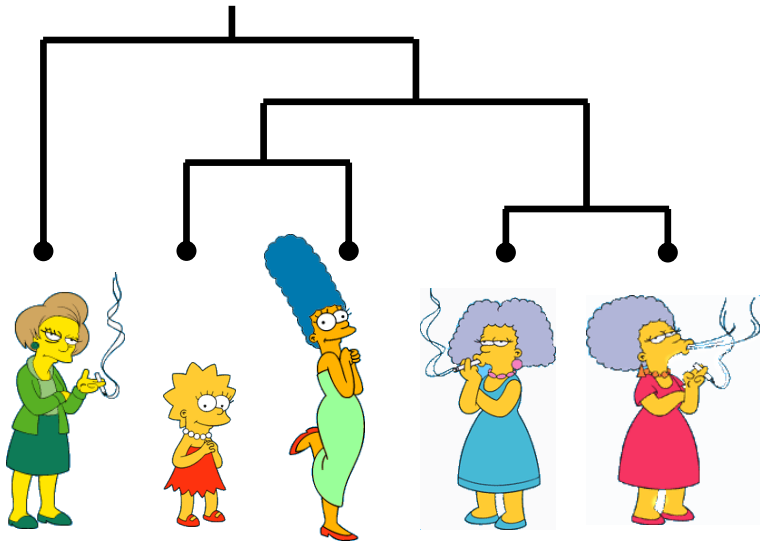
Outlier

Slide based on one by Eamonn Keogh

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



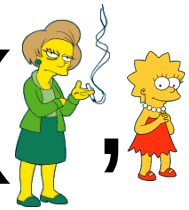
Since we cannot test all possible trees, we will have to heuristic search of all possible trees. We could do this..

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.

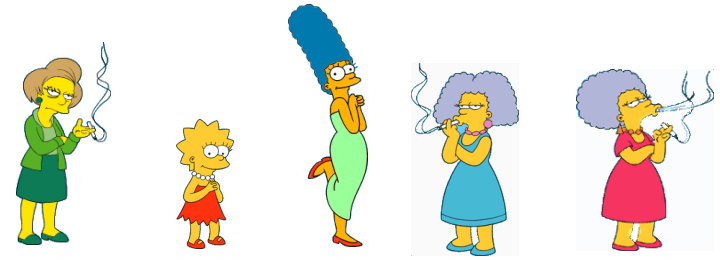
Top-Down (divisive): Starting with all the data in a single cluster, consider every possible way to divide the cluster into two. Choose the best division and recursively operate on both sides.

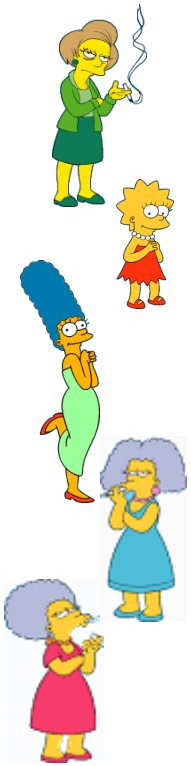
Slide based on one by Eamonn Keogh

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

D() = 8

D() = 1



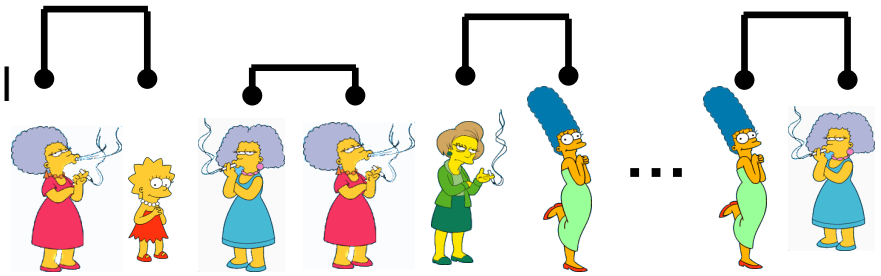


0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

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.

This slide and next 4 based on slides by Eamonn Keogh

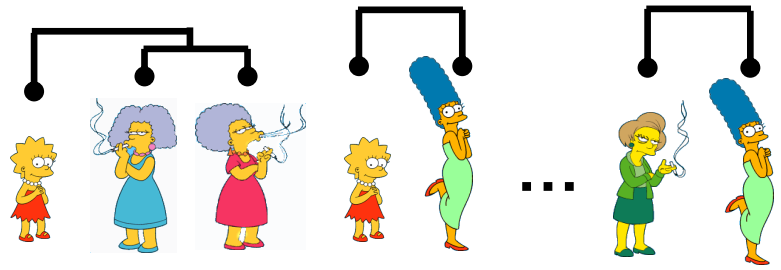
Consider all possible merges…



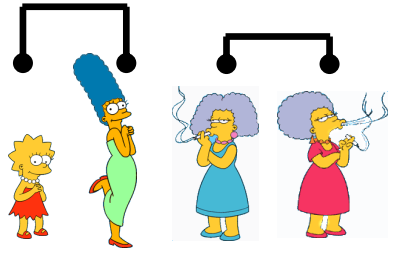
Choose the best

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.

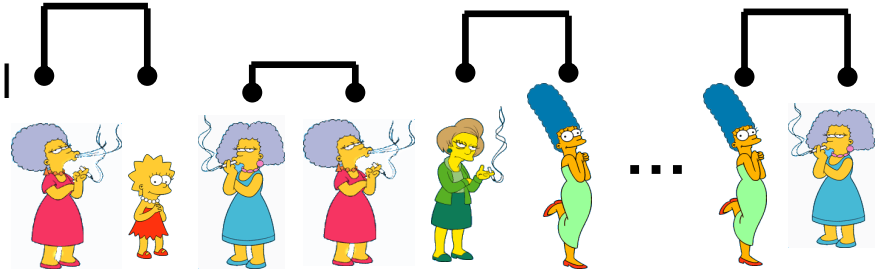
Consider all possible merges…



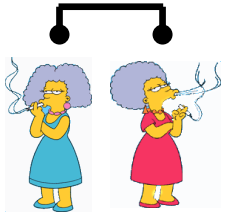
Choose the best



Consider all possible merges…

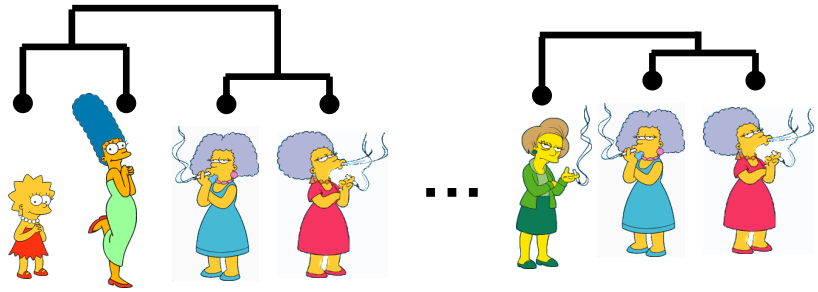


Choose the best

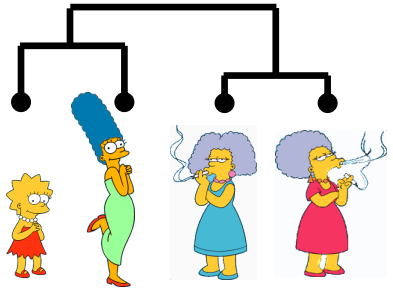


Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

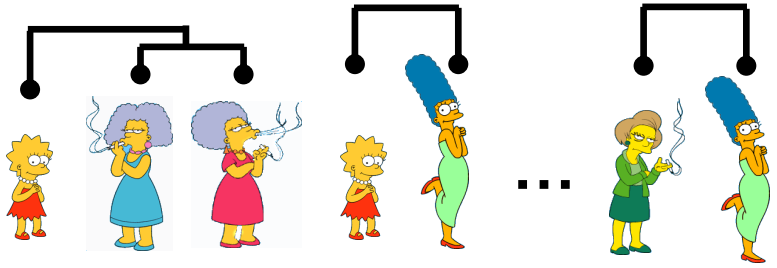
Consider all possible merges…



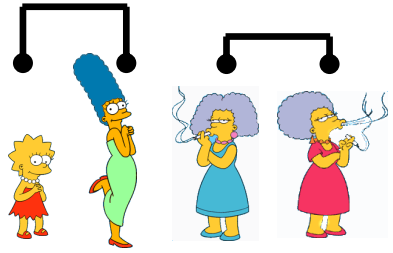
Choose the best



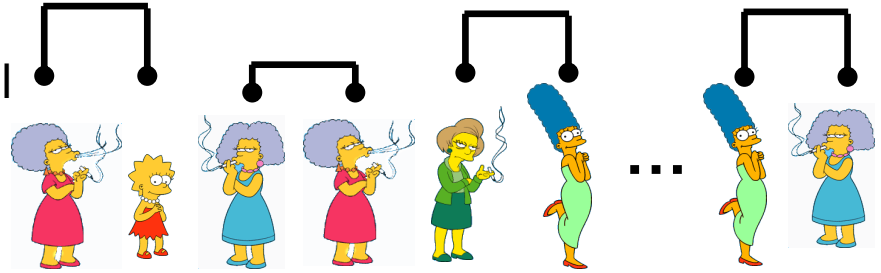
Consider all possible merges…



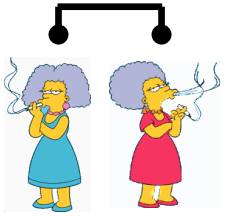
Choose the best



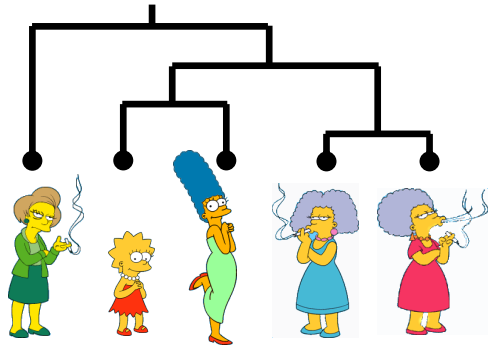
Consider all possible merges…



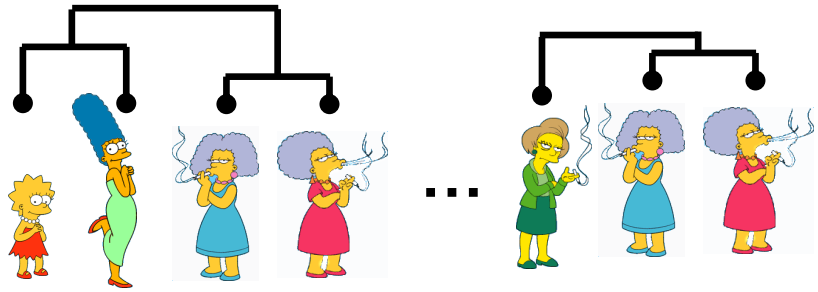
Choose the best



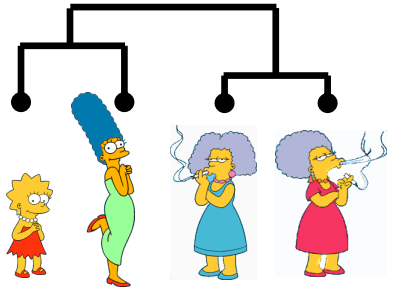
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.



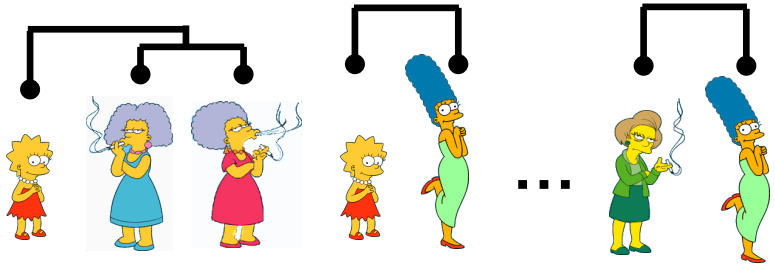
Consider all possible merges…



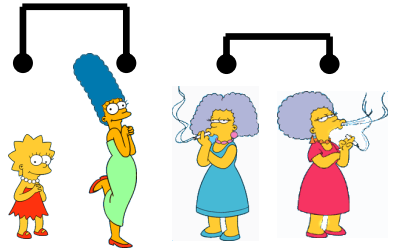
Choose the best



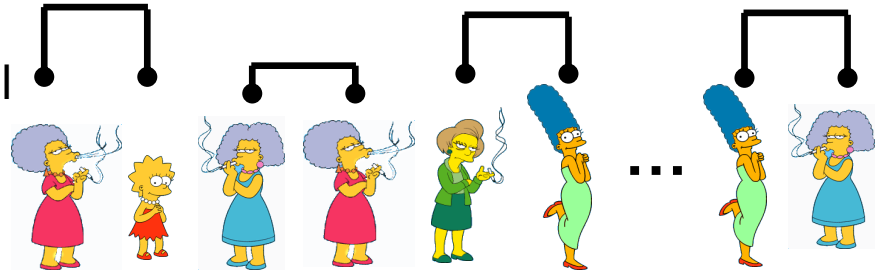
Consider all possible merges…



Choose the best



Consider all possible merges…

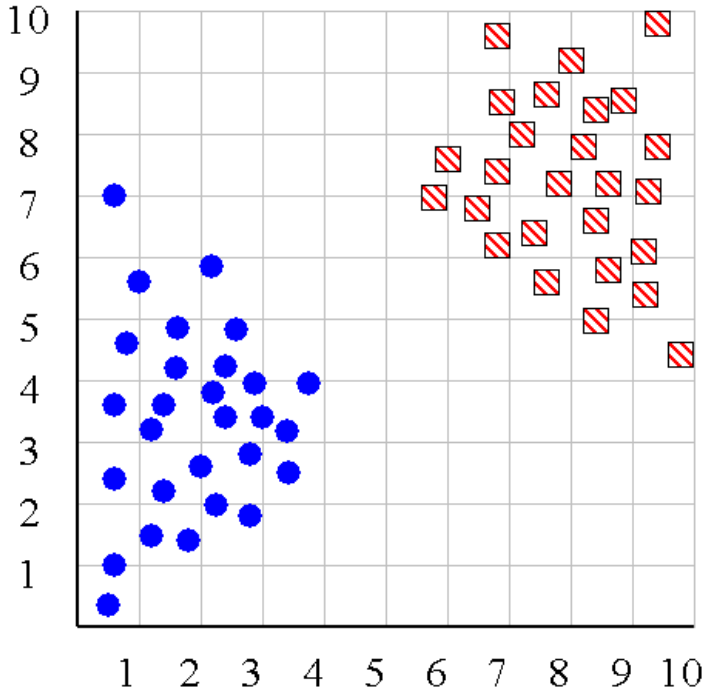


Choose the best

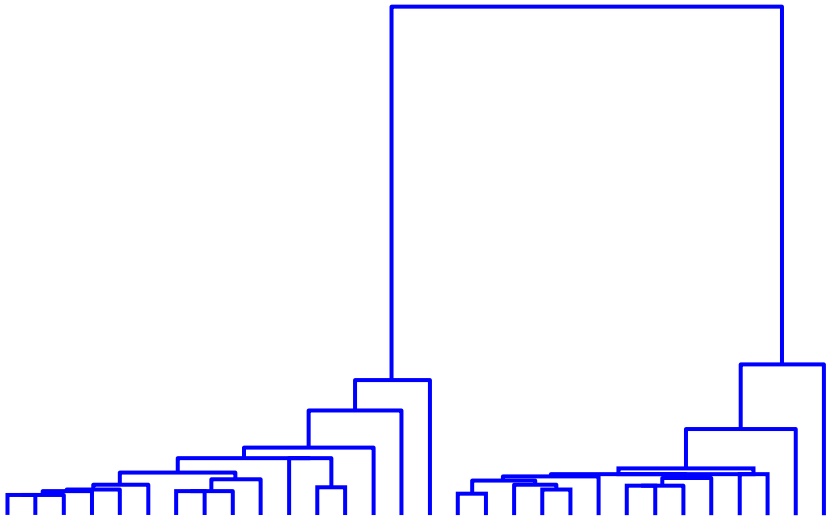
Slide based on one by Eamonn Keogh

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 nonobvious.

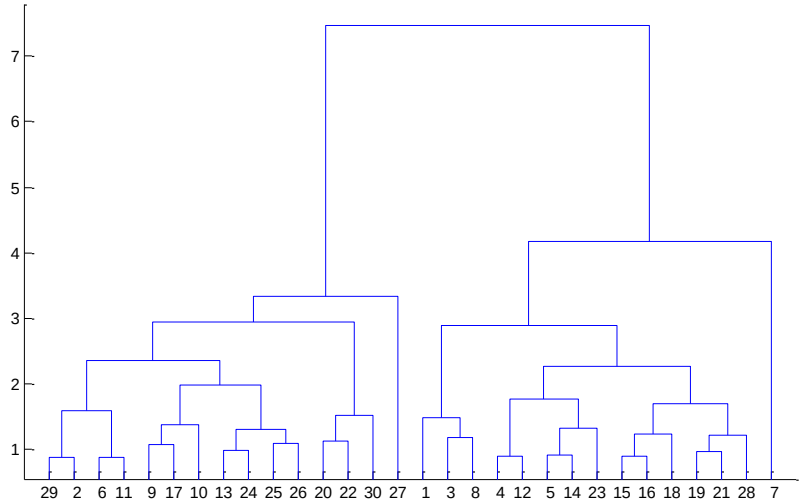
- **Single linkage (nearest neighbor):** In this method the distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters.
- **Complete linkage (furthest neighbor):** In this method, the distances between clusters are determined by 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.



Slide based on one by Eamonn Keogh



Single linkage



Average linkage

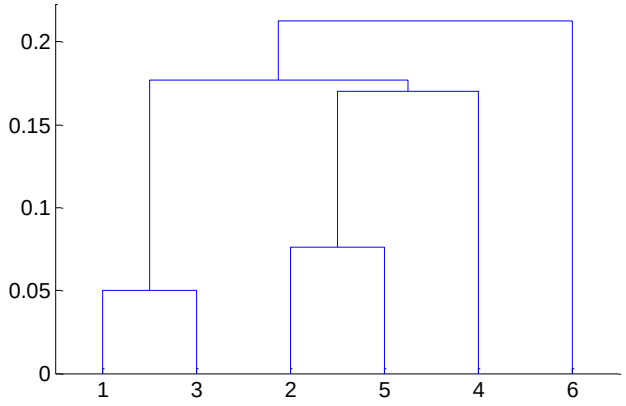
Slide based on one by Eamonn Keogh

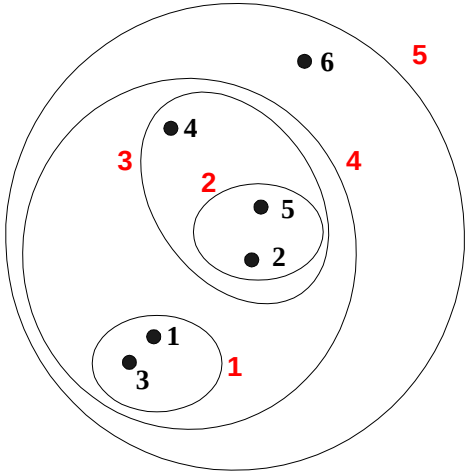
Hierarchal Clustering Methods Summary

- 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

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level

- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

Hierarchical Clustering

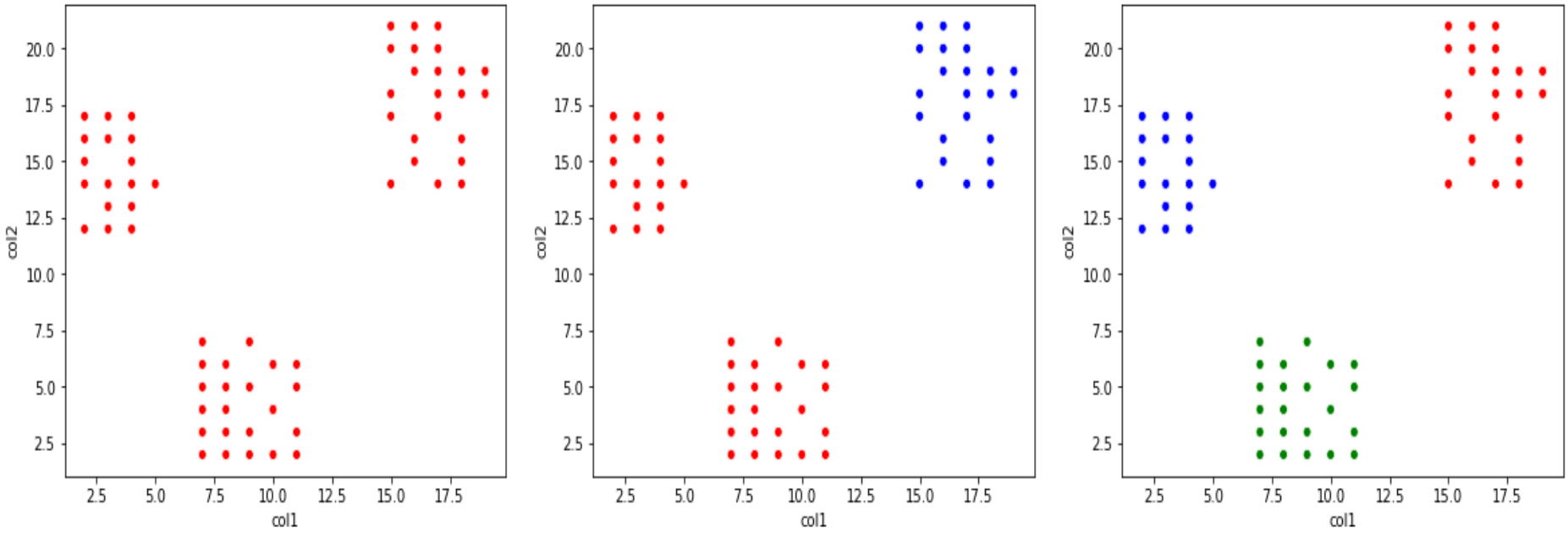
- Two main types of hierarchical clustering
  - Agglomerative (bottom up):
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive (top down):
    - Start with one, all-inclusive cluster
    - At 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

Divisive Clustering:

The divisive clustering algorithm is a top-down clustering approach, initially, all the points in the dataset belong to one cluster and split is performed recursively as one moves down the hierarchy.

- Steps of Divisive Clustering:
  1. Initially, all points in the dataset belong to one single cluster.
  2. Partition the cluster into two least similar cluster
  3. Proceed recursively to form new clusters until the desired number of clusters is obtained.

Example:



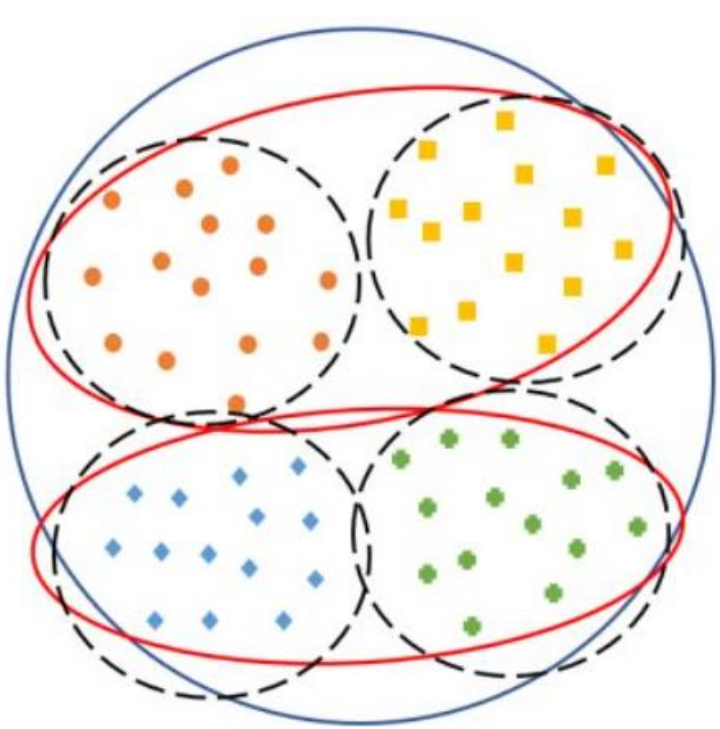
1st Image: All the data points belong to one cluster,
2nd Image: 1 cluster is separated from the previous single cluster,
3rd Image: Further 1 cluster is separated from the previous set of clusters.

The cluster with the largest SSE value is separated into 2 clusters, hence forming a new cluster. In the above image, it is observed red cluster has larger SSE so it is separated into 2 clusters forming 3 total clusters.

how to split the chosen cluster into 2 clusters. One way is to use Ward's criterion to chase for the largest reduction in the difference in the SSE criterion as a result of the split.

Other options : The minimum variance criterion, Lance–Williams algorithms

- The divisive algorithm may be implemented by using the k-means algorithm with k = 2 to perform the splits at each iteration
- We have already studied this approach
- Another approach-DIANA (DIvisive ANAlysis)



A divisive clustering proceeds by a series of successive splits. At step 0 all objects are together in a single cluster. At each step a cluster is divided, until at step `n - 1` all data objects are apart (forming `n` clusters, each with a single object).

Each step divides a cluster, let us call it `R` into two clusters `A` and `B`. Initially, `A` equals `R` and `B` is empty. In a first stage, we have to move one object from `A` to `B`. For each object `i` of `A`, we compute the average dissimilarity to all other objects of `A`:

$$d(i, A\backslash\{i\}) = \frac{1}{|A| - 1} \sum_{j \neq i} d(i, j)$$

The object `i'` for which equation above attains its maximal value will be moved, so we put

$$A_{new} = A_{old}\backslash\{i'\} \qquad B_{new} = B_{old} \cup \{i'\}$$

In the next stages, we look for other points to move from `A` to `B`. As long as `A` still contains more than one object, we compute

$$d(i, A\backslash\{i\}) - d(i, B) = \frac{1}{|A| - 1} \sum_{j \in A, j \neq i} d(i, j) - \frac{1}{|B|} \sum_{h \in B} d(i, h)$$

for each object `i` of `A` and we consider the object `i''` that maximizes this quantity. When the maximal value of the equation above is strictly positive, we move `i''` from `A` to `B` and then look in the new `A` for another object that might be moved. On the other hand, when the maximal value of the difference is negative or 0 we stop the process and the division of `R` into `A` and `B` is completed.

At each step of the divisive algorithm we also have to decide which cluster to split. For this purpose we compute the diameter

$$diam(Q) = \max_{j \in Q, h \in Q} d(j, h)$$

for each cluster `Q` that is available after the previous step, and choose the cluster for which diameter is largest.

DIANA (Divisive ANAlysis)

- DIANA is a divisive hierarchical clustering technique.

outline of the algorithm.

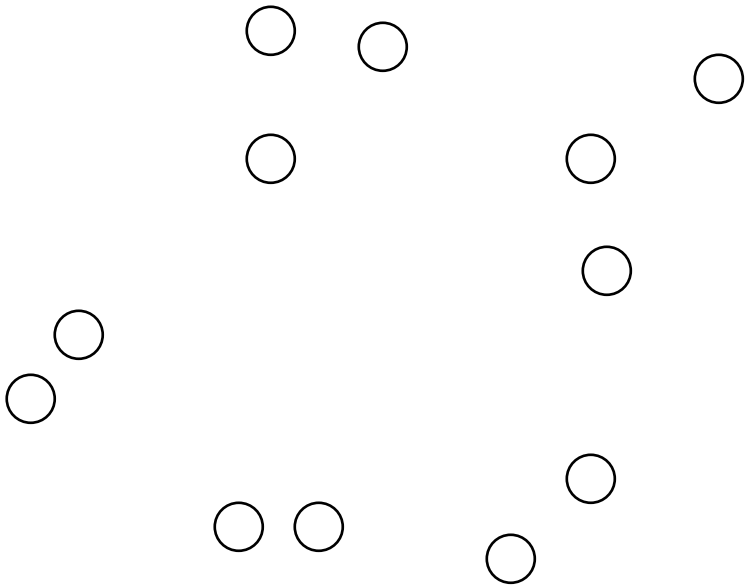
- Step 1. Suppose that cluster Cl is going to be split into clusters Ci and Cj
- Step 2. Let Ci = Cl and Cj = Ø.
- Step 3. For each object x ∈ Ci :
- (a) For the first iteration, compute the average distance of x to all other objects.
- (b) For the remaining iterations, compute Dx = average {d(x, y) : y ∈ Ci} – average{d(x, y) : y ∈ Cj}.

Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Starting Situation

- Start with clusters of individual points and a proximity matrix

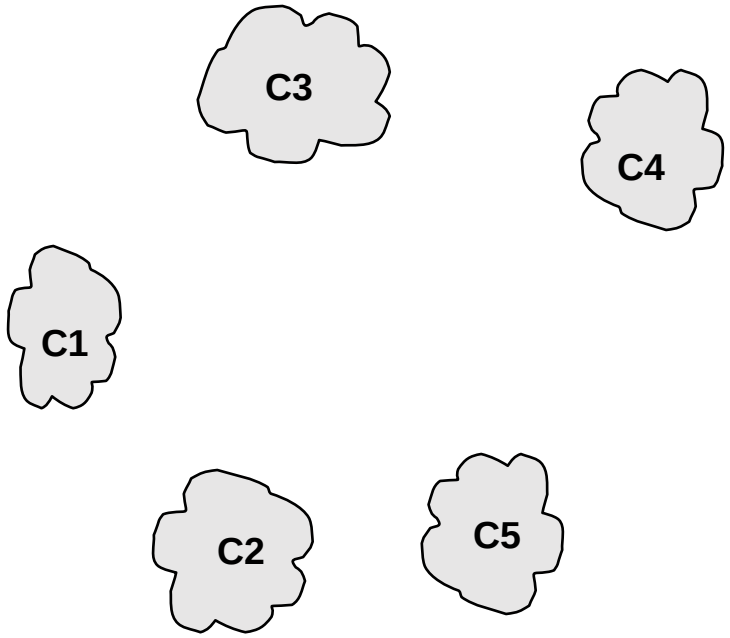


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

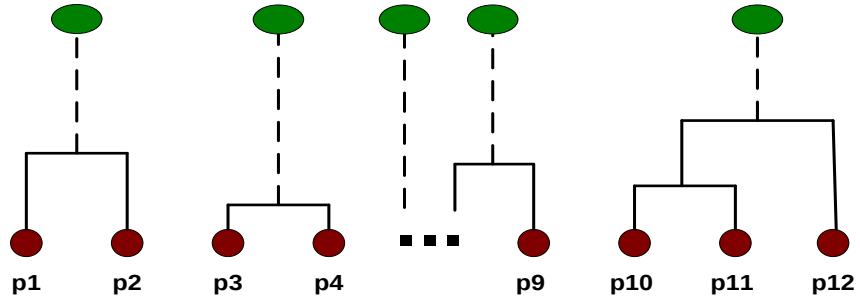
Intermediate Situation

- After some merging steps, we have some clusters



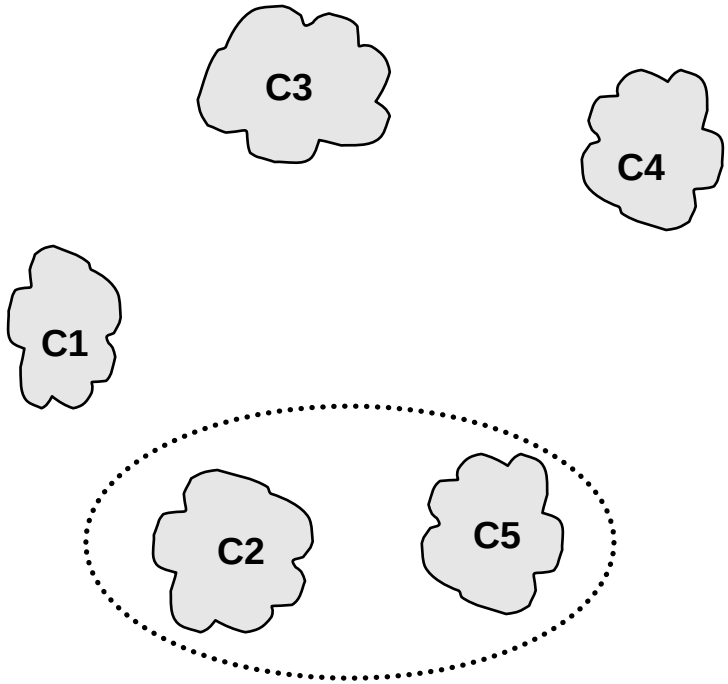
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



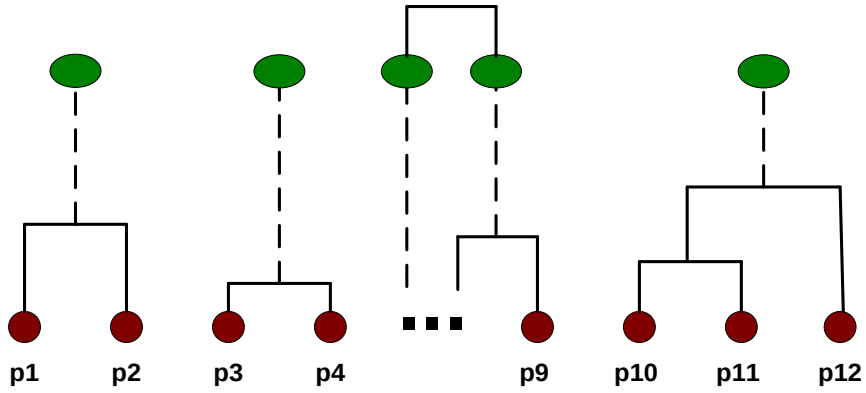
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



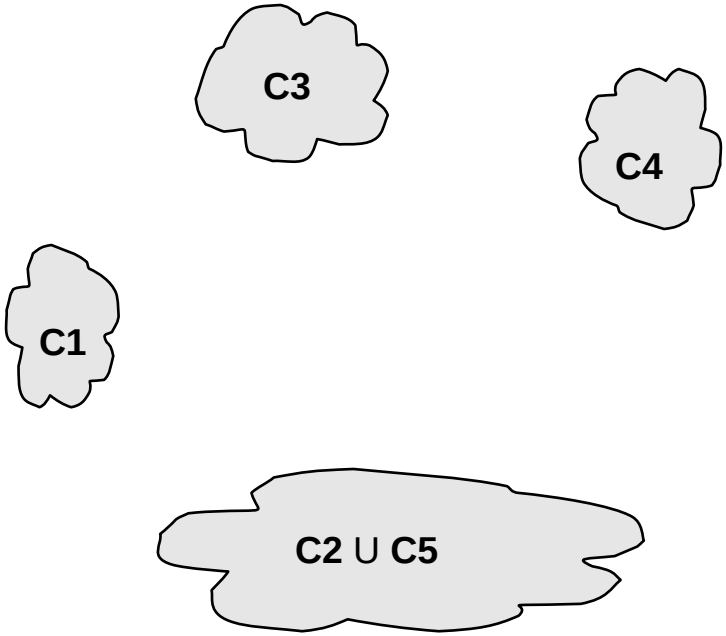
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



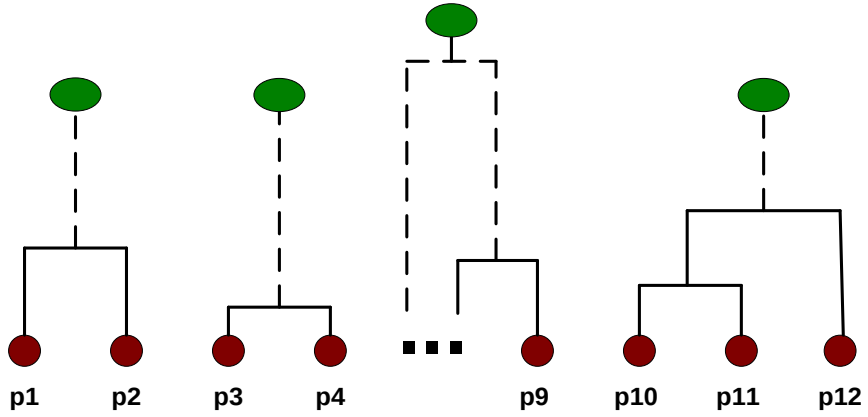
After Merging

- The question is “How do we update the proximity matrix?”

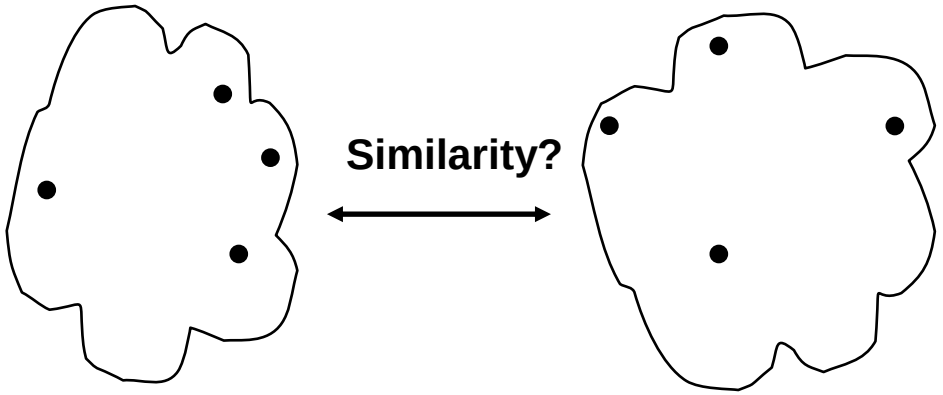


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity



- MIN
- MAX
- Group Average
- Distance Between Centroids
- Ward's method (not discussed)

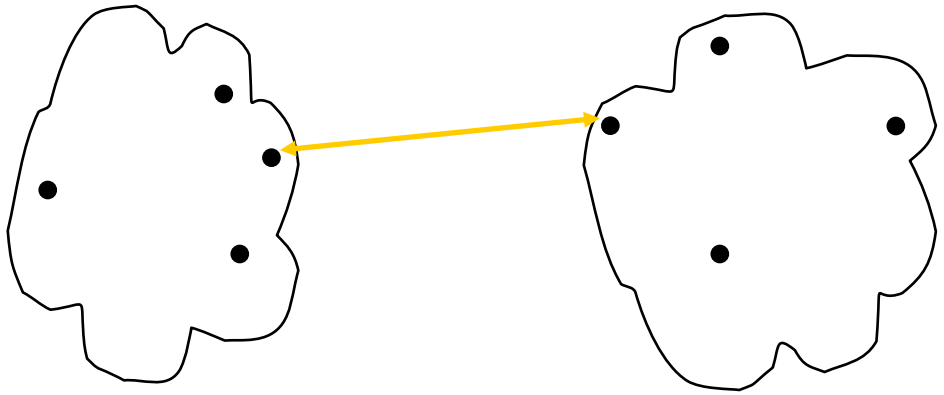
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

.

.

Proximity Matrix

How to Define Inter-Cluster Similarity

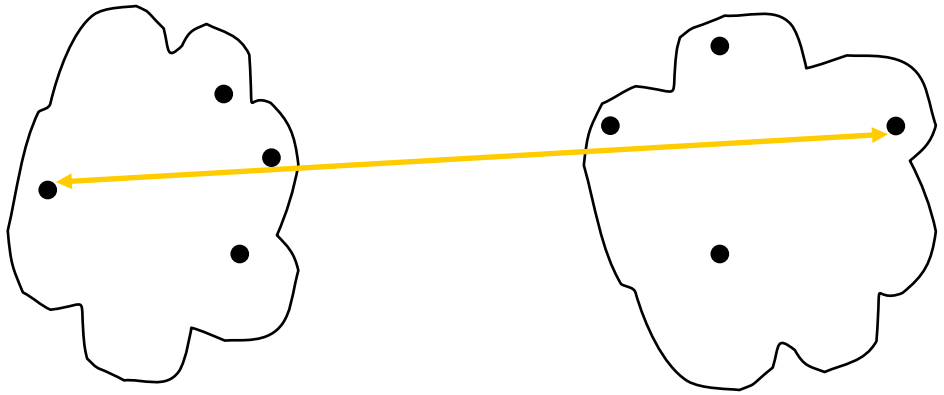


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

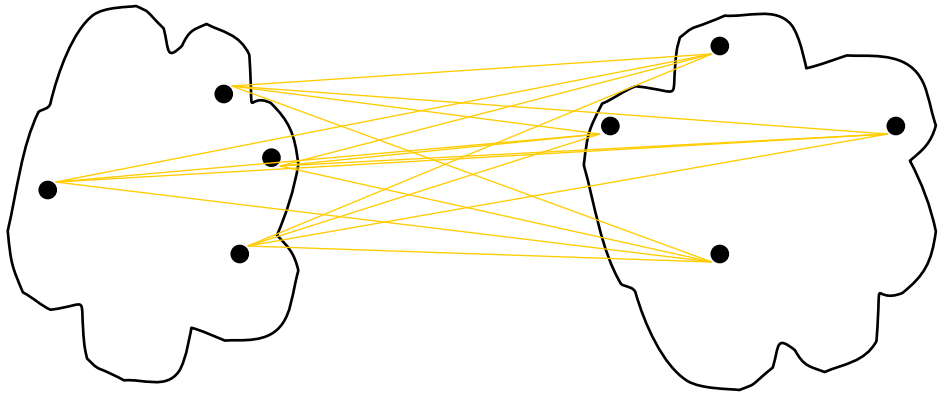


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

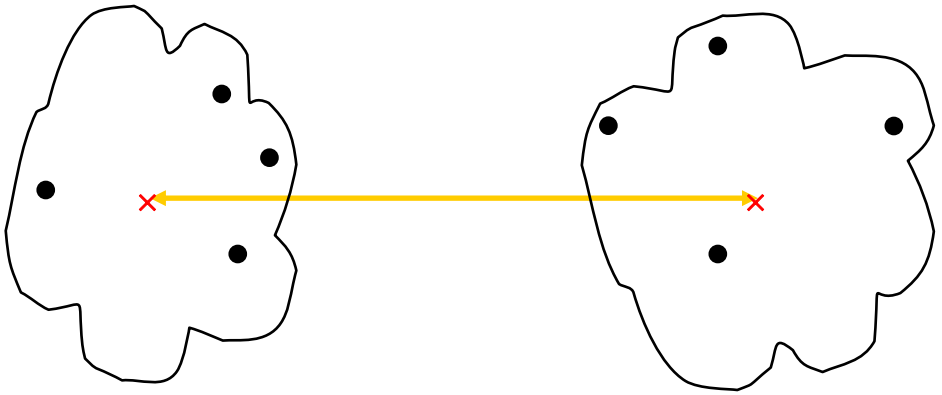


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity



- MIN
- MAX
- Group Average
- Distance Between Centroids

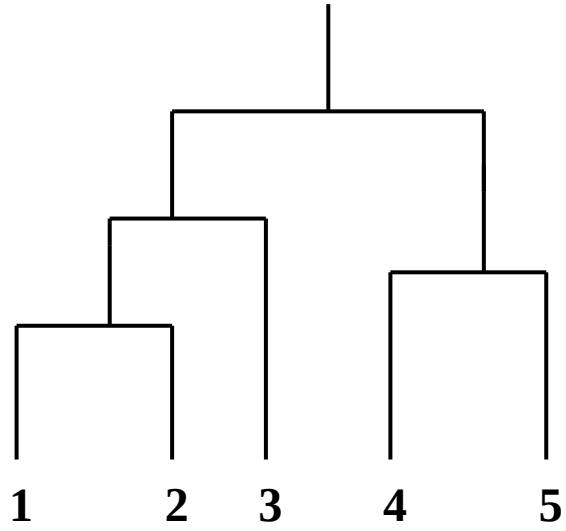
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

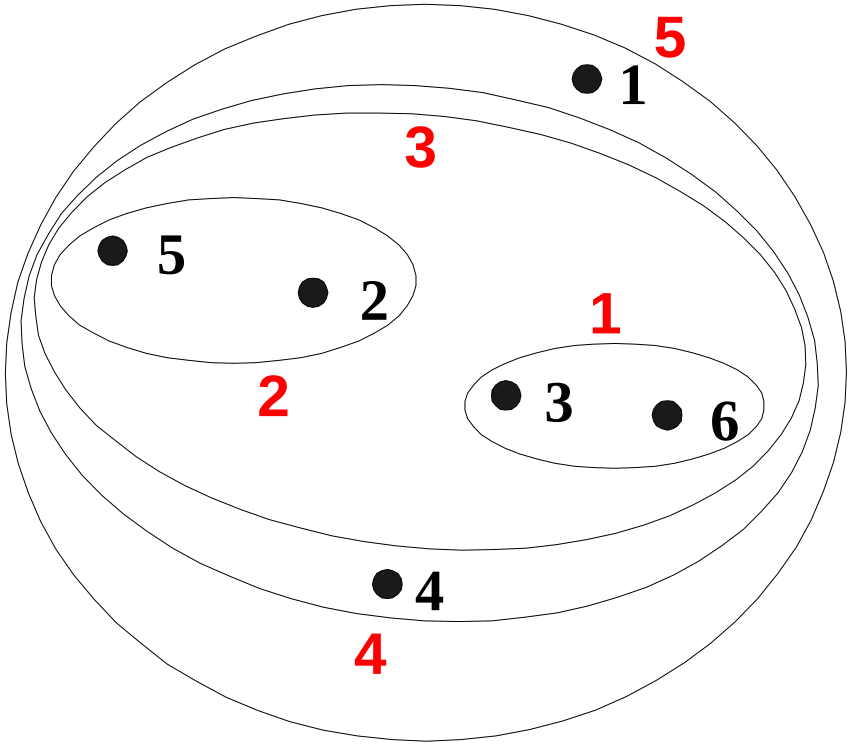
Cluster Similarity: MIN or Single Link

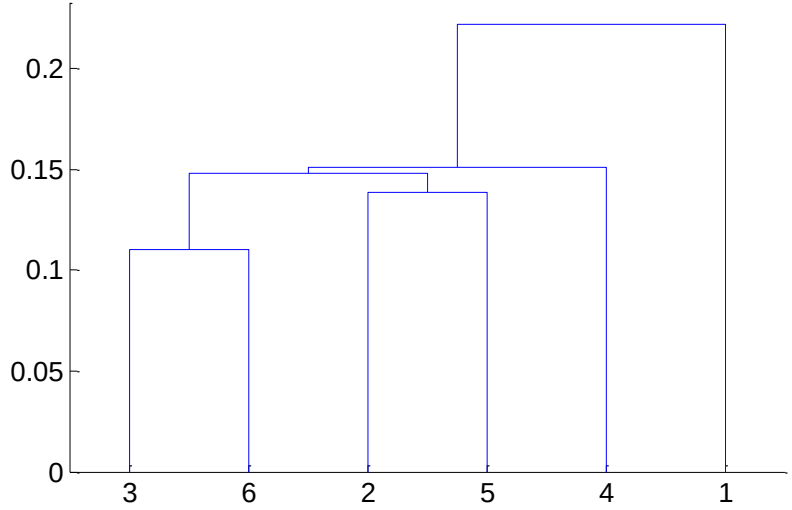
- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph.

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MIN

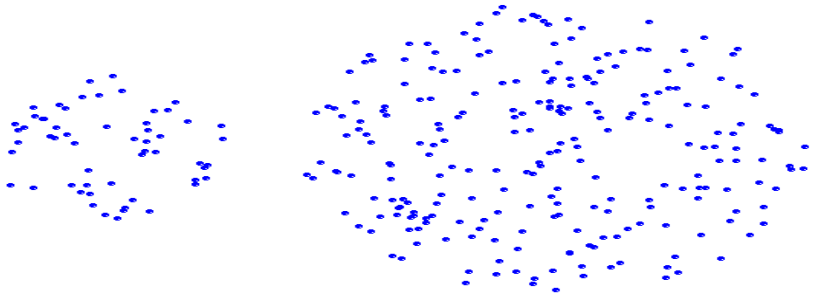




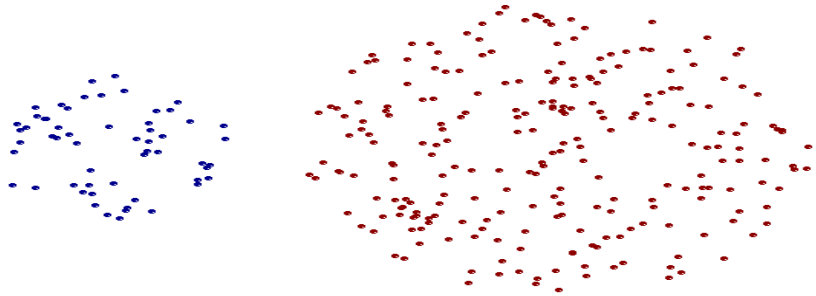
Nested Clusters

Dendrogram

Strength of MIN



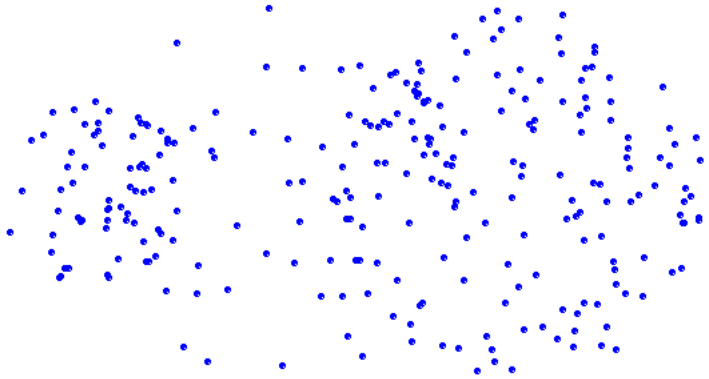
Original Points



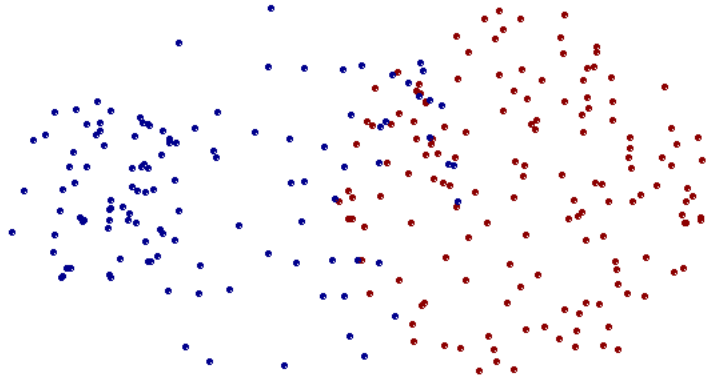
Two Clusters

- **Can handle non-elliptical shapes**

Limitations of MIN



Original Points



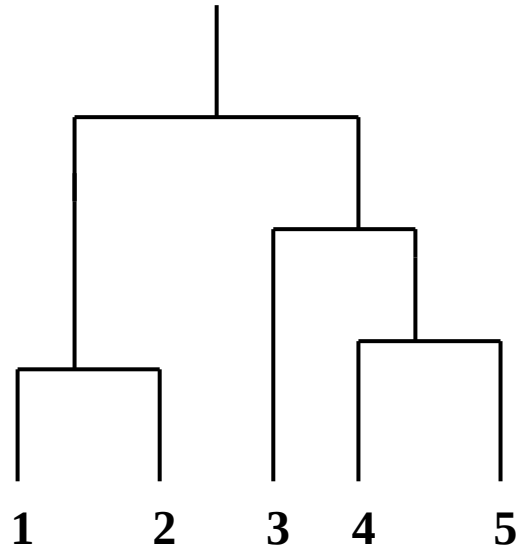
Two Clusters

- **Sensitive to noise and outliers**

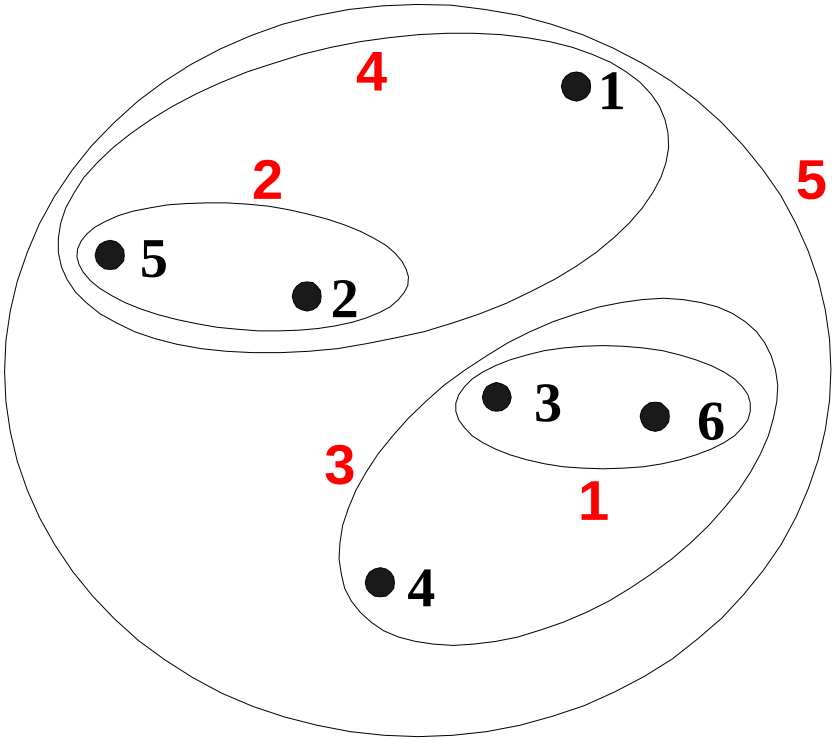
Cluster Similarity: MAX or Complete Linkage

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

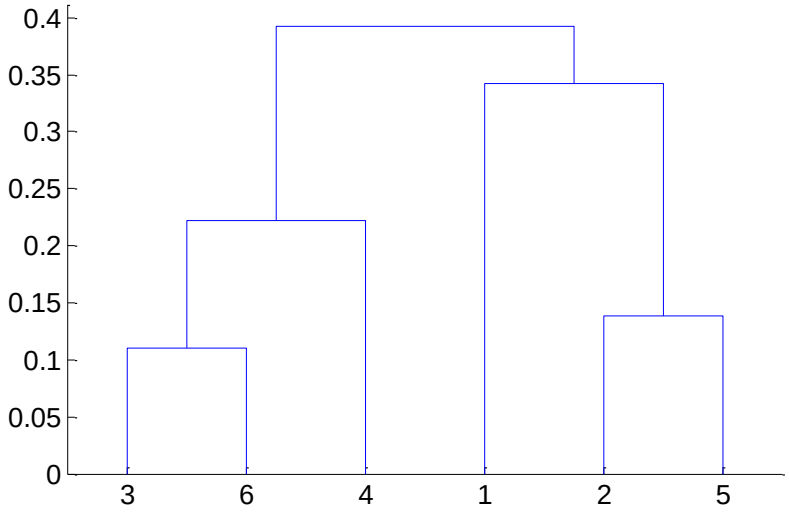
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MAX

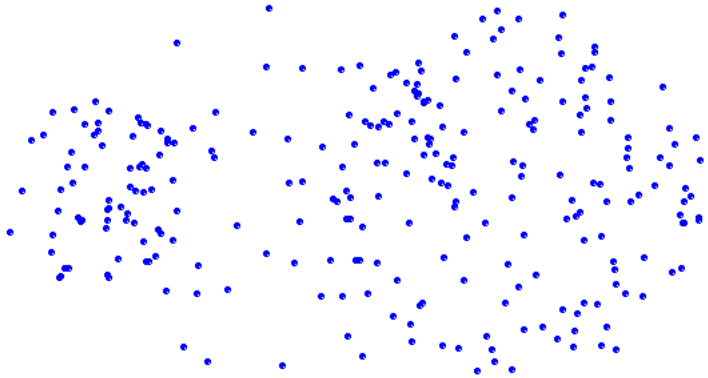


Nested Clusters

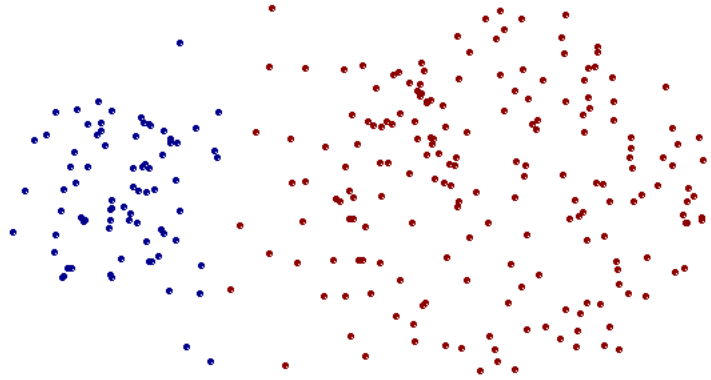


Dendrogram

Strength of MAX



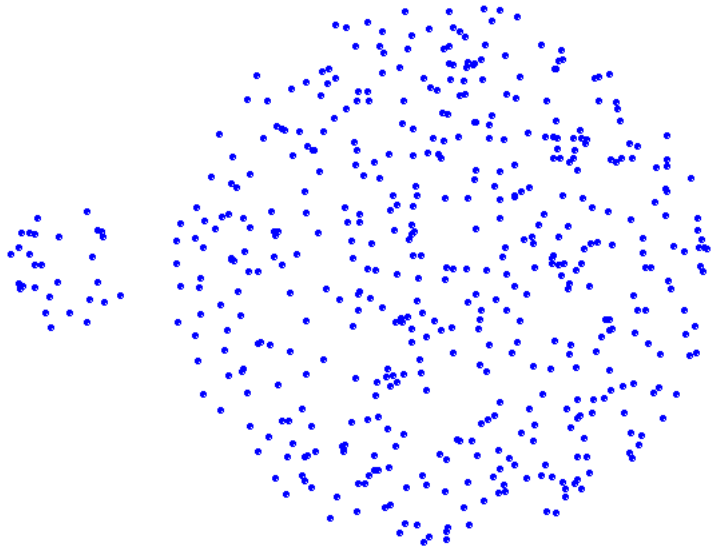
Original Points



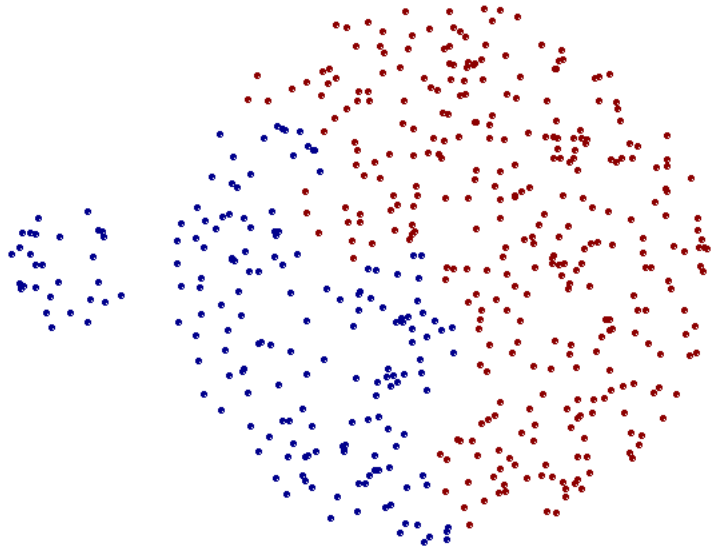
Two Clusters

- **Less susceptible to noise and outliers**

Limitations of MAX



Original Points



Two Clusters

- **Tends to break large clusters**
- **Biased towards globular clusters**

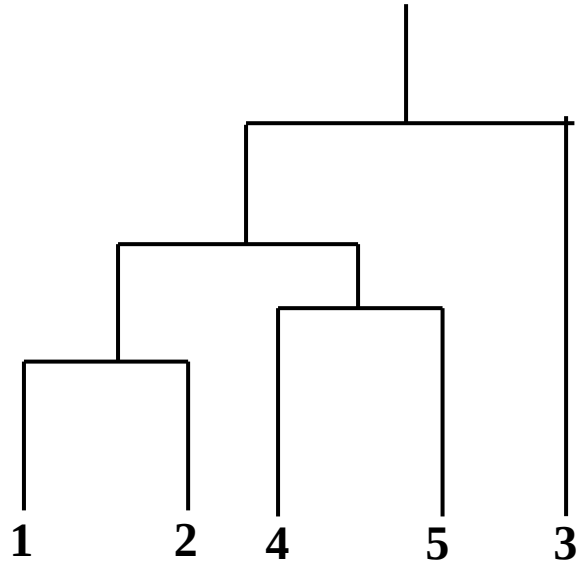
Cluster Similarity: Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

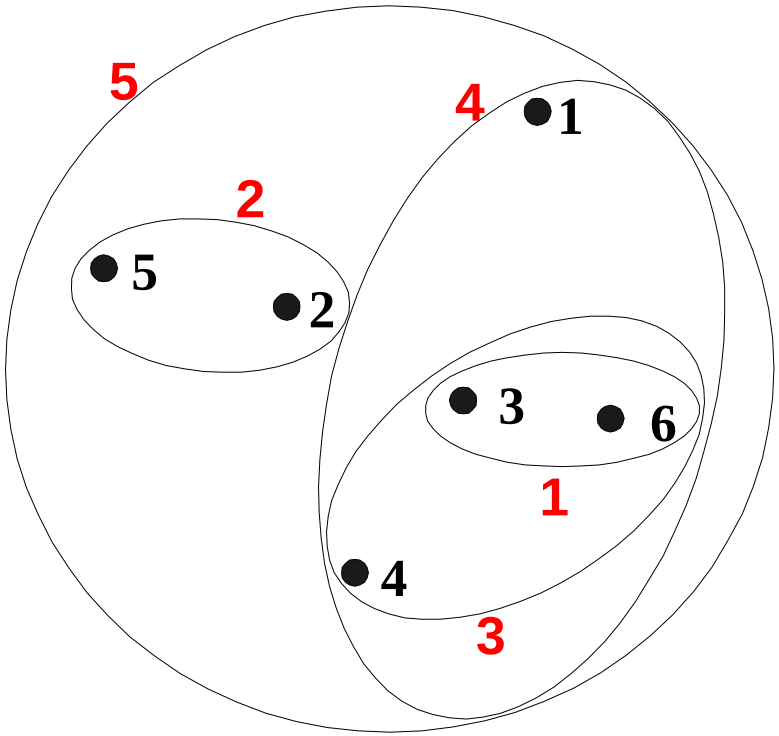
$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters

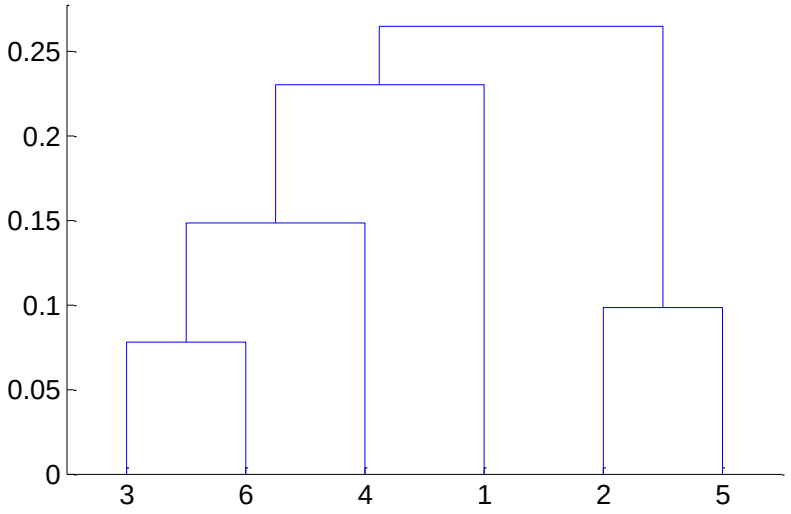
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: Group Average



Nested Clusters



Dendrogram

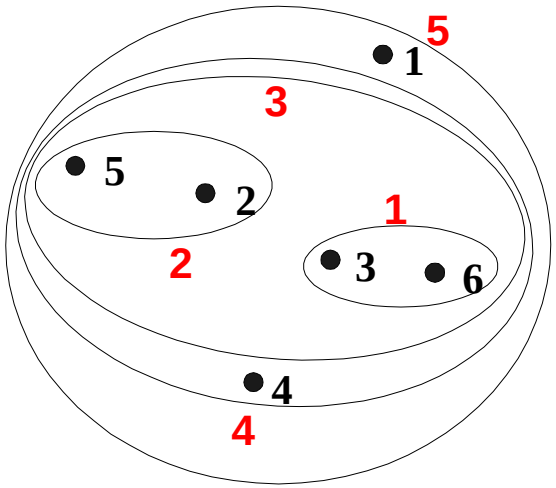
Hierarchical Clustering: Group Average

- Compromise between Single and Complete Link

- Strengths
  - Less susceptible to noise and outliers

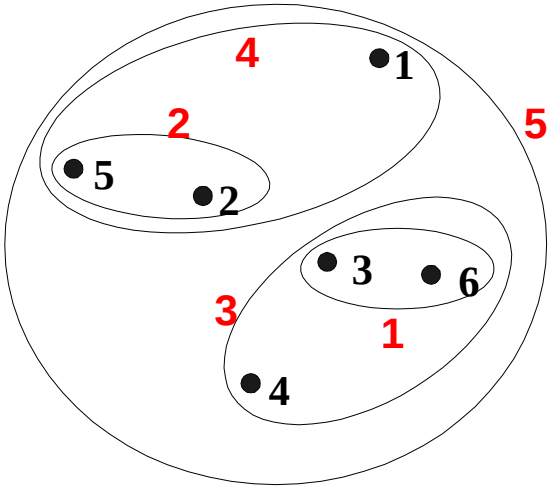
- Limitations
  - Biased towards globular clusters

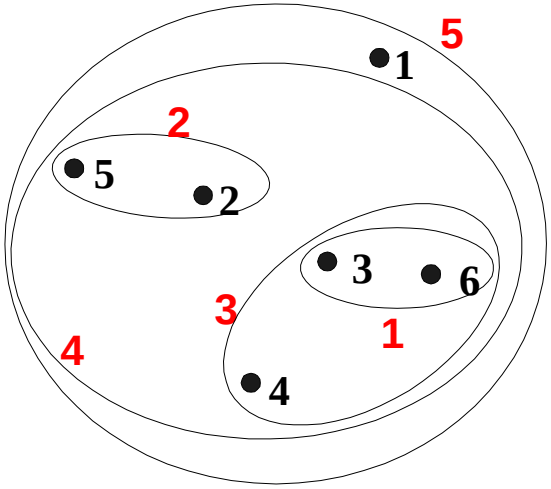
Hierarchical Clustering: Comparison



MIN

MAX





Group Average

Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters (divisive)
- Dendrogram correspond to a given hierarchical clustering is not unique, since for each merge one needs to specify which subtree should go on the left and which on the right
- They impose structure on the data, instead of revealing structure in these data.
- How many clusters? (some suggestions later)

k-Means Algorithm

- k-Means clustering algorithm proposed by J. Hartigan and M. A. Wong [1979].

- Given a set of *n* distinct objects, the k-Means clustering algorithm partitions the objects into *k* number of clusters such that intracluster similarity is high but the intercluster similarity is low.

- In this algorithm, user must specify *k*, the number of clusters and consider the objects are defined with numeric attributes and thus using any one of the distance metric to demarcate the clusters.

Clustering

- Suppose we want to cluster *n* vectors in R^d into two groups. Define C_1 and C_2 as the two groups.
- Our objective is to find C_1 and C_2 that minimize

$$\sum_{i=1}^{2} \sum_{x_j \in C_i} \| x_j - m_i \|^2$$

where m_i is the mean of class C_i

Clustering

- NP hard even for 2-means

12. ^ Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. (2009). "NP-hardness of Euclidean sum-of-squares clustering". *Machine Learning* **75**: 245–249. doi:10.1007/s10994-009-5103-0.
13. ^ Dasgupta, S. and Freund, Y. (July 2009). "Random Projection Trees for Vector Quantization". *Information Theory, IEEE Transactions on* **55**: 3229–3242. arXiv:0805.1390. doi:10.1109/TIT.2009.2021326.

- NP hard even on plane

14. ^ Mahajan, M.; Nimbhorkar, P.; Varadarajan, K. (2009). "The Planar k-Means Problem is NP-Hard". *Lecture Notes in Computer Science* **5431**: 274–285. doi:10.1007/978-3-642-00202-1_24.

- K-means heuristic
  - Popular and hard to beat
  - Introduced in 1950s and 1960s

K-means algorithm for two clusters

Input: $x_i \in R^d, i = 1 \ldots n$

Algorithm:

1. Initialize: assign x_i to C_1 or C_2 with equal probability and compute means:

$$m_1 = \frac{1}{|C_1|} \sum_{x_i \in C_1} x_i \qquad m_2 = \frac{1}{|C_2|} \sum_{x_i \in C_2} x_i$$

2. Recompute clusters: assign x_i to C_1 if $||x_i - m_1|| < ||x_i - m_2||$, otherwise assign to C_2
3. Recompute means m_1 and m_2
4. Compute objective

$$\sum_{i=1}^{2} \sum_{x_j \in C_i} \| x_j - m_i \|^2$$

5. Compute objective of new clustering. If difference is smaller than α then stop, otherwise go to step 2.

k-Means Algorithm

The algorithm can be stated as follows.

- First it selects *k* number of objects at random from the set of n objects. These *k* objects are treated as the centroids or center of gravities of *k* clusters.

- For each of the remaining objects, it is assigned to one of the closest centroid. Thus, it forms a collection of objects assigned to each centroid and is called a cluster.

- Next, the centroid of each cluster is then updated (by calculating the mean values of attributes of each object).

- The assignment and update procedure is until it reaches some stopping criteria (such as, number of iteration, centroids remain unchanged or no assignment, etc.)

k-Means Algorithm

Algorithm 16.1: k-Means clustering

Input: D is a dataset containing *n* objects, *k* is the number of cluster

Output: A set of *k* clusters

Steps:

1. Randomly choose *k* objects from D as the initial cluster centroids.

2. **For** each of the objects in D **do**
   - Compute distance between the current objects and *k* cluster centroids
   - Assign the current object to that cluster to which it is closest.

3. Compute the "cluster centers" of each cluster. These become the new cluster centroids.

4. Repeat step 2-3 until the convergence criterion is satisfied

5. Stop

k-Means Algorithm

Note:

1) Objects are defined in terms of set of attributes. $A = \{A_1, A_2, \ldots\ldots, A_m\}$ where each A_i is continuous data type.

2) Distance computation: Any distance such as L_1, L_2, L_3 or cosine similarity.

3) Minimum distance is the measure of closeness between an object and centroid.

4) Mean Calculation: It is the mean value of each attribute values of all objects.

5) Convergence criteria: Any one of the following are termination condition of the algorithm.
   - Number of maximum iteration permissible.
   - No change of centroid values in any cluster.
   - Zero (or no significant) movement(s) of object from one cluster to another.
   - Cluster quality reaches to a certain level of acceptance.

Illustration of k-Means clustering algorithms

Table 16.1: 16 objects with two attributes A_1 and A_2.

A_1	A_2
6.8	12.6
0.8	9.8
1.2	11.6
2.8	9.6
3.8	9.9
4.4	6.5
4.8	1.1
6.0	19.9
6.2	18.5
7.6	17.4
7.8	12.2
6.6	7.7
8.2	4.5
8.4	6.9
9.0	3.4
9.6	11.1

Fig 16.1: Plotting data of Table 16.1

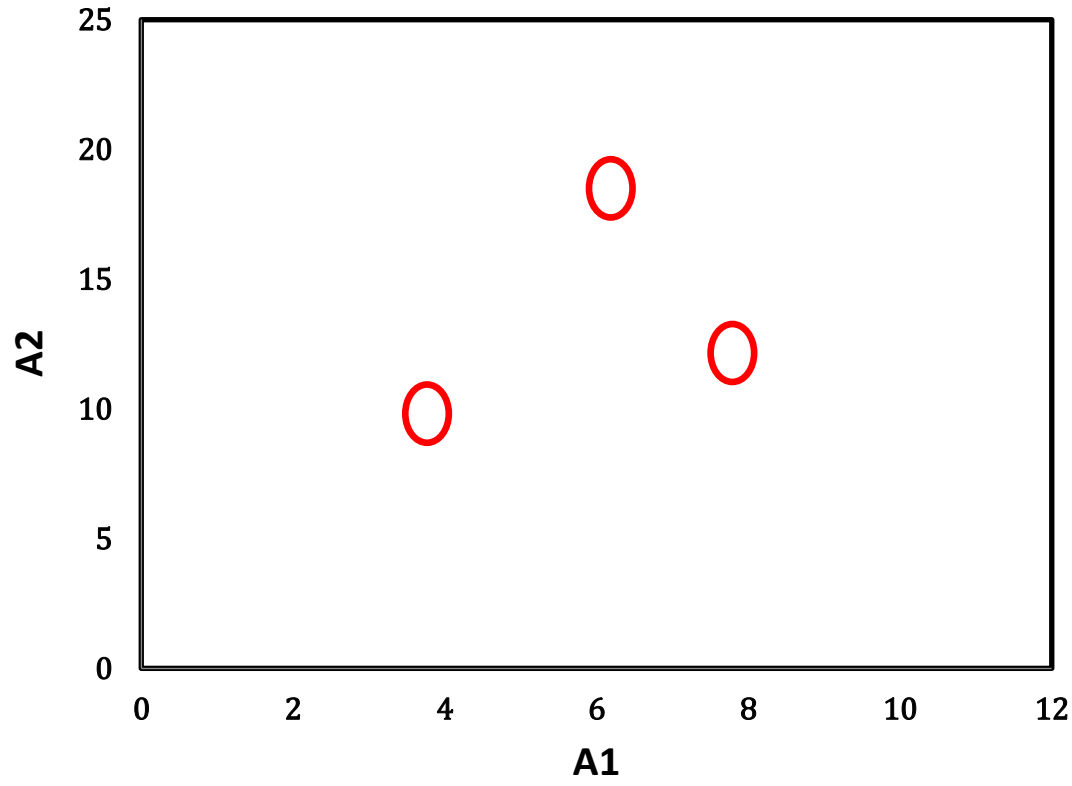


Illustration of k-Means clustering algorithms

- Suppose, k=3. Three objects are chosen at random shown as circled (see Fig 16.1). These three centroids are shown below.

Initial Centroids chosen randomly

Centroid	Objects	
	A1	A2
c_1	3.8	9.9
c_2	7.8	12.2
c_3	6.2	18.5

- Let us consider the Euclidean distance measure (L_2 Norm) as the distance measurement in our illustration.
- Let d_1, d_2 and d_3 denote the distance from an object to c_1, c_2 and c_3 respectively. The distance calculations are shown in Table 16.2.
- Assignment of each object to the respective centroid is shown in the right-most column and the clustering so obtained is shown in Fig 16.2.

Illustration of k-Means clustering algorithms

Table 16.2: Distance calculation

A_1	A_2	d_1	d_2	d_3	cluster
6.8	12.6	4.0	1.1	5.9	2
0.8	9.8	3.0	7.4	10.2	1
1.2	11.6	3.1	6.6	8.5	1
2.8	9.6	1.0	5.6	9.5	1
3.8	9.9	0.0	4.6	8.9	1
4.4	6.5	3.5	6.6	12.1	1
4.8	1.1	8.9	11.5	17.5	1
6.0	19.9	10.2	7.9	1.4	3
6.2	18.5	8.9	6.5	0.0	3
7.6	17.4	8.4	5.2	1.8	3
7.8	12.2	4.6	0.0	6.5	2
6.6	7.7	3.6	4.7	10.8	1
8.2	4.5	7.0	7.7	14.1	1
8.4	6.9	5.5	5.3	11.8	2
9.0	3.4	8.3	8.9	15.4	1
9.6	11.1	5.9	2.1	8.1	2

Fig 16.2: Initial cluster with respect to Table 16.2

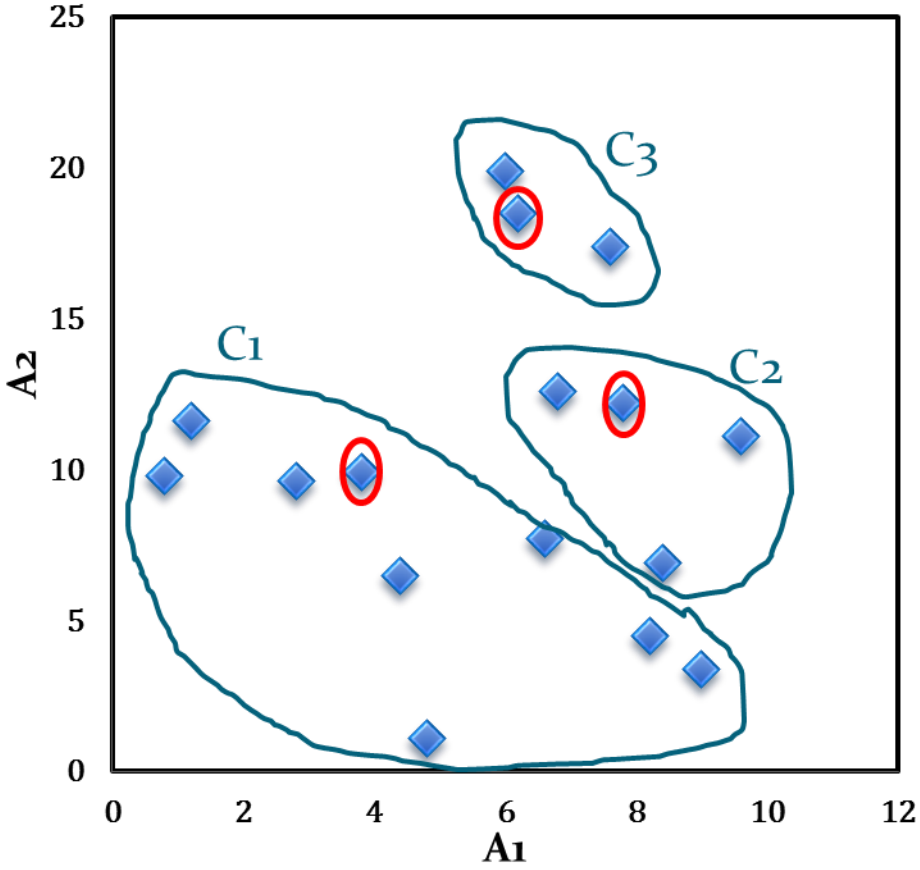


Illustration of k-Means clustering algorithms

The calculation new centroids of the three cluster using the mean of attribute values of A_1 and A_2 is shown in the Table below. The cluster with new centroids are shown in Fig 16.3.

Calculation of new centroids

New Centroid	Objects	
	A1	A2
c_1	4.6	7.1
c_2	8.2	10.7
c_3	6.6	18.6

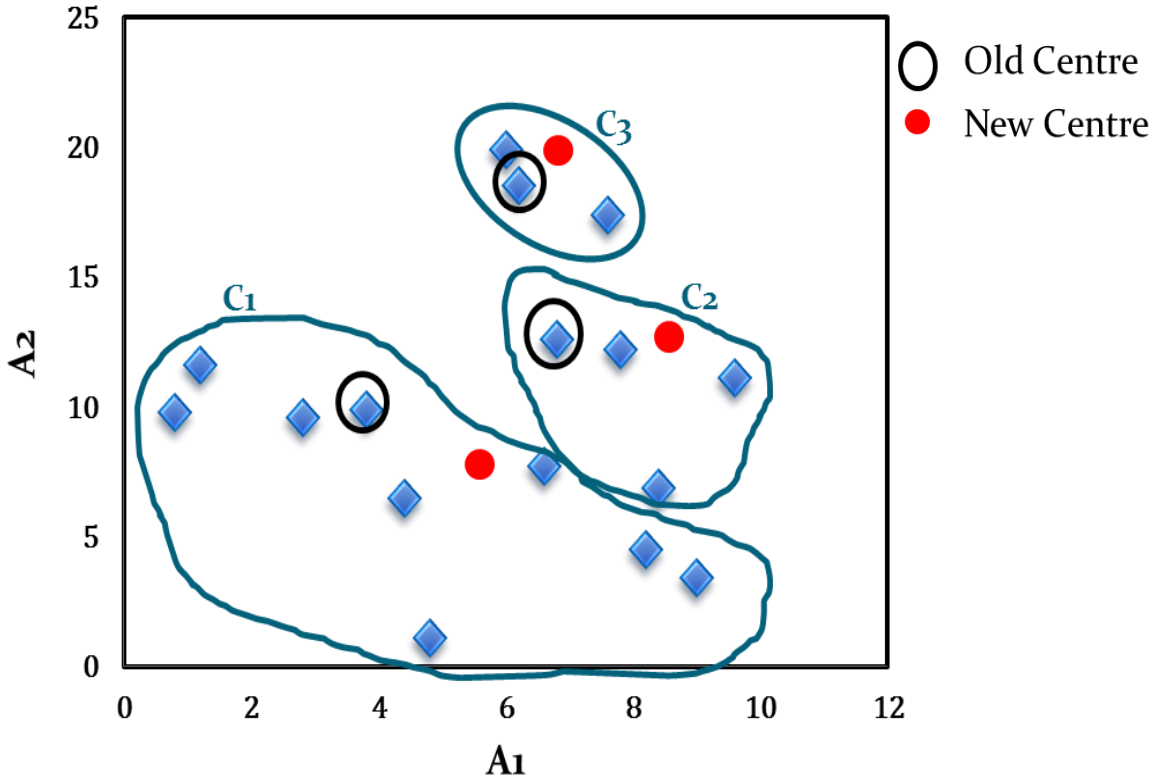


Fig 16.3: Initial cluster with new centroids

Illustration of k-Means clustering algorithms

We next reassign the 16 objects to three clusters by determining which centroid is closest to each one. This gives the revised set of clusters shown in Fig 16.4.

Note that point p moves from cluster C_2 to cluster C_1.

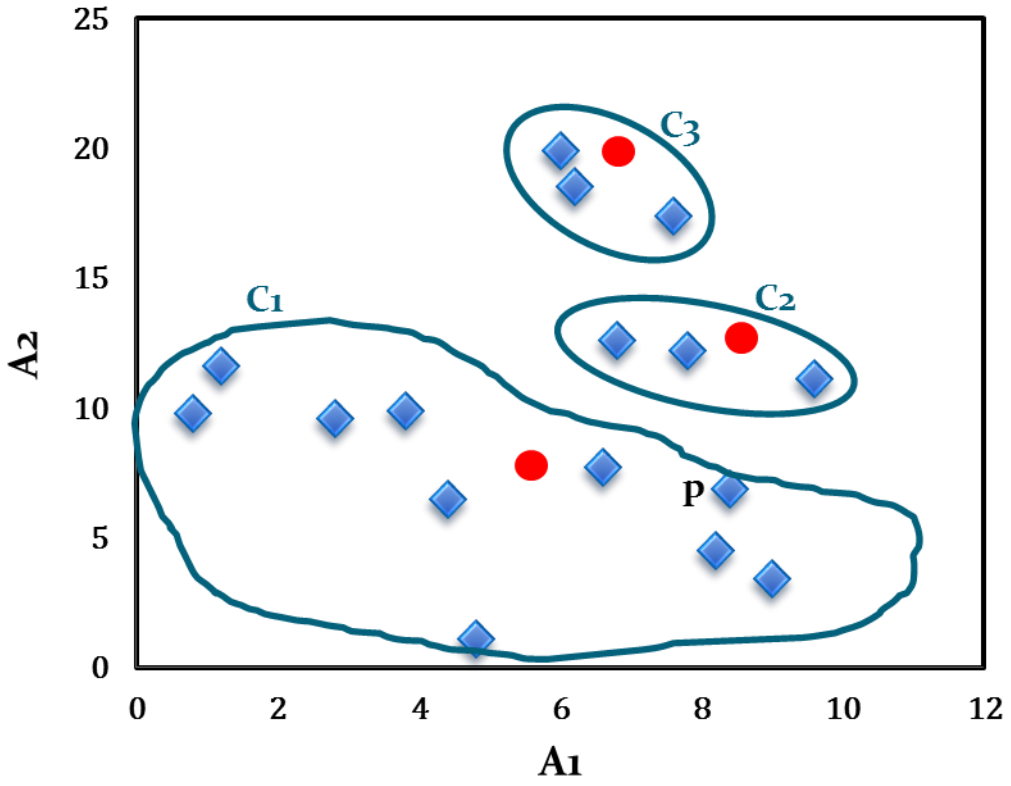


Fig 16.4: Cluster after first iteration

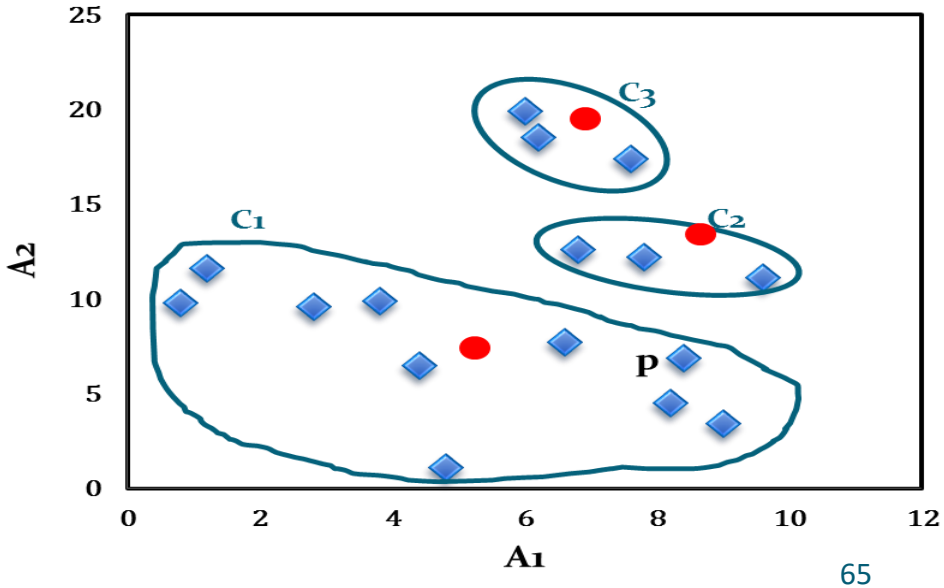
Illustration of k-Means clustering algorithms

- The newly obtained centroids after second iteration are given in the table below. Note that the centroid c_3 remains unchanged, where c_2 and c_1 changed a little.
- With respect to newly obtained cluster centres, 16 points are reassigned again. These are the same clusters as before. Hence, their centroids also remain unchanged.
- Considering this as the termination criteria, the k-means algorithm stops here. Hence, the final cluster in Fig 16.5 is same as Fig 16.4.

Cluster centres after second iteration

Centroid	Revised Centroids	
	A1	A2
c_1	5.0	7.1
c_2	8.1	12.0
c_3	6.6	18.6

Fig 16.5: Cluster after Second iteration



Comments on k-Means algorithm

Let us analyse the k-Means algorithm and discuss the pros and cons of the algorithm.

We shall refer to the following notations in our discussion.

- Notations:
  - x : an object under clustering
  - n : number of objects under clustering
  - $\boldsymbol{C}_i$: the *i-th* cluster
  - c_i : the centroid of cluster $\boldsymbol{C}_i$
  - n_i : number of objects in the cluster $\boldsymbol{C}_i$
  - c : denotes the centroid of all objects
  - k : number of clusters

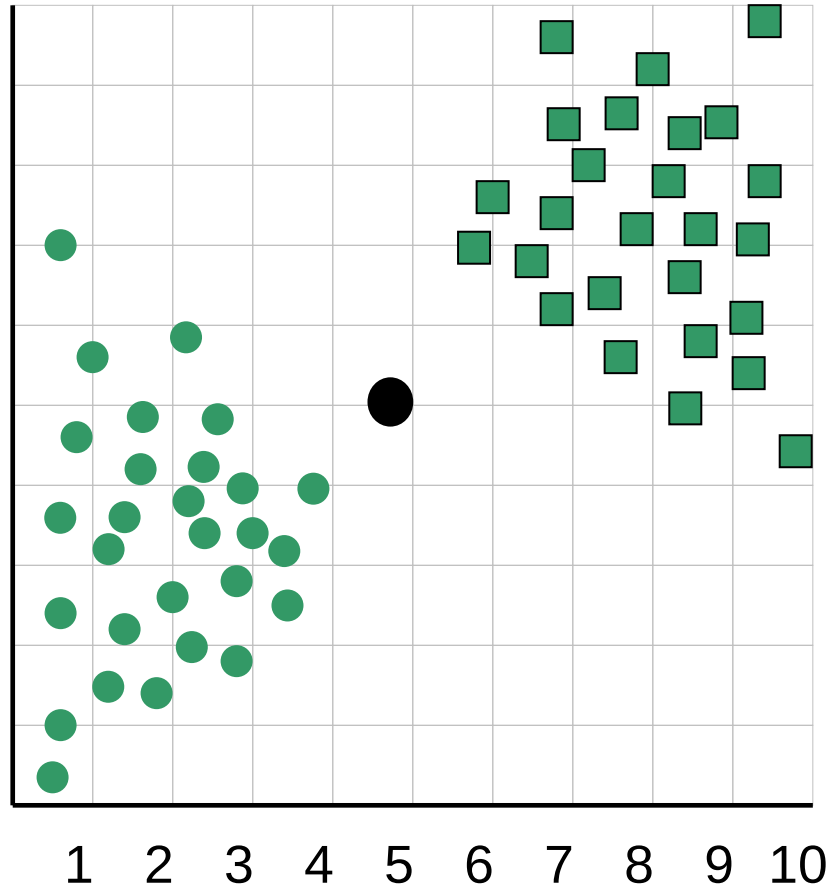
Comments on k-Means algorithm

1. **Value of k:**
   - The k-means algorithm produces only one set of clusters, for which, user must specify the desired number, k of clusters.
   - In fact, k should be the best guess on the number of clusters present in the given data. Choosing the best value of k for a given dataset is, therefore, an issue.
   - We may not have an idea about the possible number of clusters for high dimensional data, and for data that are not scatter-plotted.
   - Further, possible number of clusters is hidden or ambiguous in image, audio, video and multimedia clustering applications etc.
   - There is no principled way to know what the value of k ought to be. We may try with successive value of k starting with 2.
   - The process is stopped when two consecutive k values produce more-or-less identical results (with respect to some cluster quality estimation).
   - Normally $k \ll n$ and there is heuristic to follow $k \approx \sqrt{n}$.

Um, what about k?

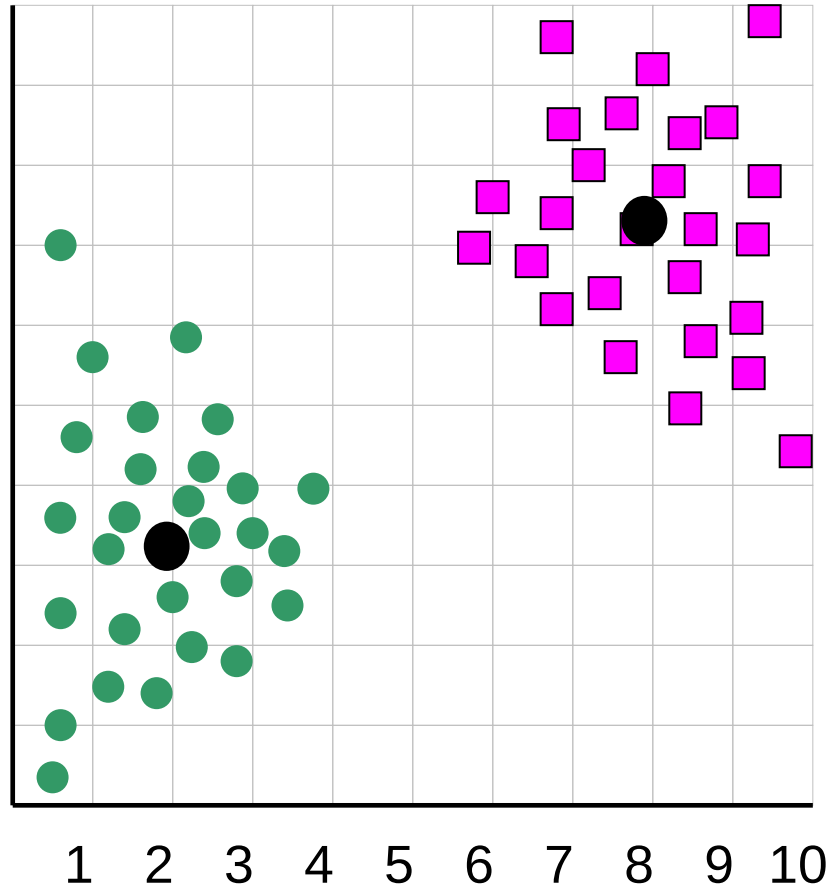
- Idea 1: Use our new trick of cross validation to select k
  - What should we optimize? SSE? Trace?
  - Problem?
- Idea 2: Let our domain expert look at the clustering and decide if they like it
  - How should we show this to them?
  - Problem?
- Idea 3: The “knee” solution

When k = 1, the objective function is 873.0



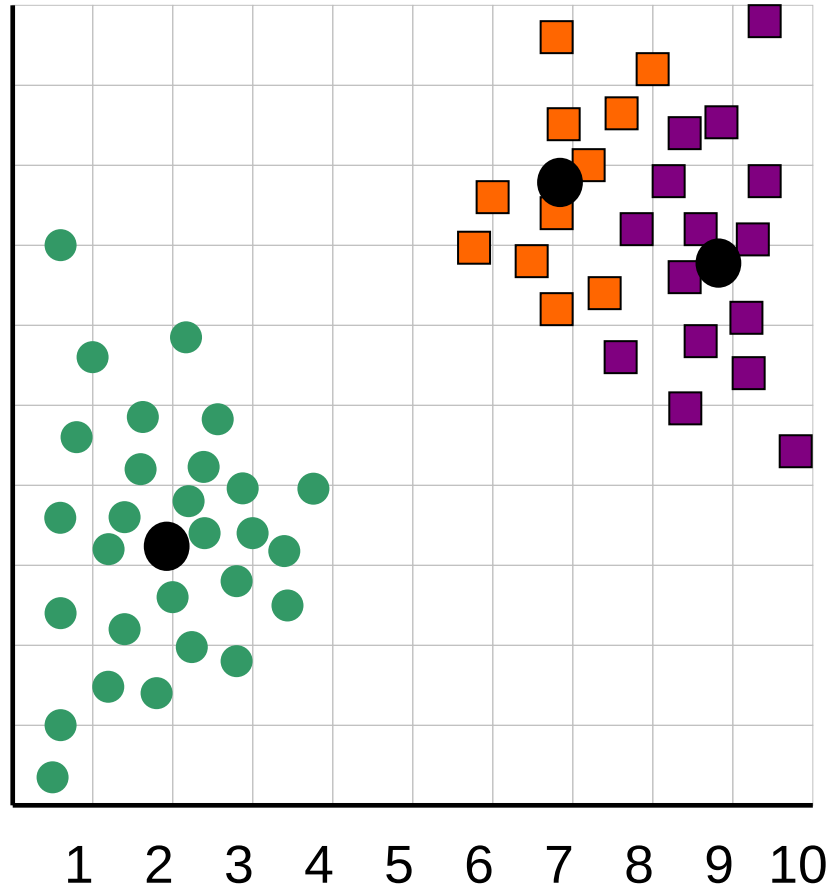
Slide based on one by Eamonn Keogh

When k = 2, the objective function is 173.1



Slide based on one by Eamonn Keogh

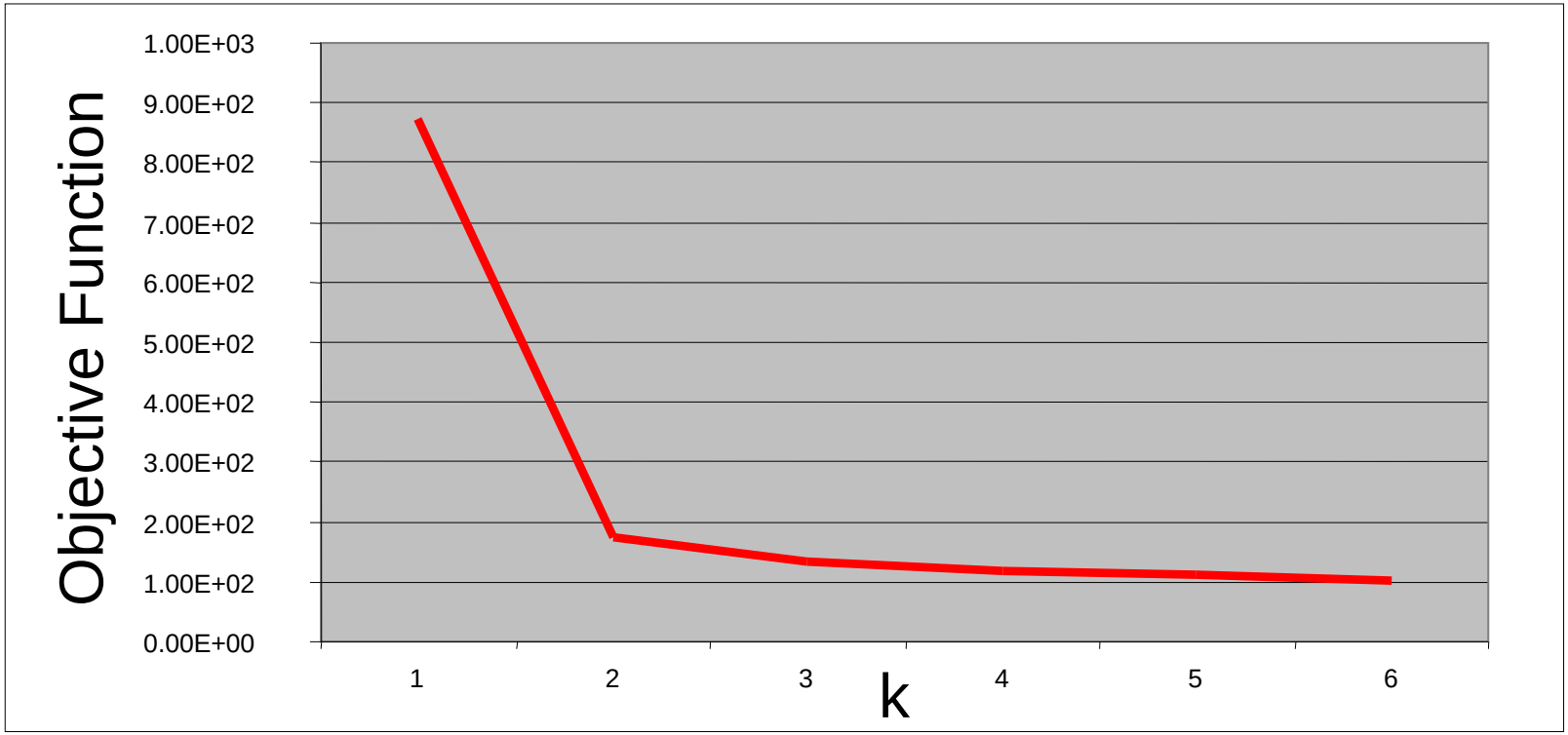
When k = 3, the objective function is 133.6



Slide based on one by Eamonn Keogh

We can plot the objective function values for k equals 1 to 6…

The abrupt change at k = 2, is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as “knee finding” or “elbow finding”.



Slide based on one by Eamonn Keogh

K-Means Clustering

- Inputs: data set and k (number of clusters)
- Output: each point assigned to one of k clusters

- K-Means Algorithm:
  - Initialize the k-means
    - assign from randomly selected points
    - randomly or equally distributed in space
  - Assign each point to nearest mean
  - Update means from assigned points
  - Repeat until convergence

K-Means Clustering: Convergence

- Squared-Error Criterion

- Converged when SE criterion stops changing
- Increasing K reduces SE - can’t determine K by finding minimum SE
- Instead, plot SE as function of K

K-Means Clustering

- Efficient
  - K << N, so assigning points is $O(K*N) < O(N^2)$
  - updating means can be done during assignment
  - usually # of iterations << N
  - Overall $O(N*K*\text{iterations})$ closer to $O(N)$ than $O(N^2)$
- Gets stuck in local minima
  - Sensitive to initialization
- Number of clusters must be pre-specified
- Requires vector space date to calculate means

Soft K-Means Clustering

- Instance of EM (Expectation Maximization)
- Like K-Means, except each point is assigned to each cluster with a probability
- Cluster means updated using weighted average
- Generalizes to Standard_Deviation/Covariance
- Works well if cluster models are known

Soft K-Means Clustering (EM)

- Initialize model parameters:
  - means
  - std_devs
  - ...
- Assign points probabilistically to each cluster
- Update cluster parameters from weighted points
- Repeat until convergence to local minimum

Comments on k-Means algorithm

Example 16.1: k versus cluster quality

- Usually, there is some objective function to be met as a goal of clustering. One such objective function is sum-square-error denoted by SSE and defined as

$$SSE = \sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} (x - c_i)^2$$

- Here, $x - c_i$ denotes the error, if x is in cluster $\boldsymbol{C}_i$ with cluster centroid c_i.

- Usually, this error is measured as distance norms like L_1, L_2, L_3 or Cosine similarity, etc.

Comments on k-Means algorithm

Example 16.1: k versus cluster quality

- With reference to an arbitrary experiment, suppose the following results are obtained.

k	SSE
1	62.8
2	12.3
3	9.4
4	9.3
5	9.2
6	9.1
7	9.05
8	9.0

- With respect to this observation, we can choose the value of $k \approx 3$, as with this smallest value of k it gives reasonably good result.
- Note: If $k = n$, then SSE=0; However, the cluster is useless! This is another example of overfitting.

Comments on k-Means algorithm

2. Choosing initial centroids:

- Another requirement in the k-Means algorithm to choose initial cluster centroid for each *k* would be clusters.
- It is observed that the k-Means algorithm terminate whatever be the initial choice of the cluster centroids.
- It is also observed that initial choice influences the ultimate cluster quality. In other words, the result may be trapped into local optima, if initial centroids are chosen properly.
- One technique that is usually followed to avoid the above problem is to choose initial centroids in multiple runs, each with a different set of randomly chosen initial centroids, and then select the best cluster (with respect to some quality measurement criterion, e.g. SSE).
- However, this strategy suffers from the combinational explosion problem due to the number of all possible solutions.

Comments on k-Means algorithm

2. Choosing initial centroids:

- A detail calculation reveals that there are $c(n, k)$ possible combinations to examine the search of global optima.

$$c(n,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^{k-i}\binom{k}{i}(i)^n$$

- For example, there are $o(10^{10})$ different ways to cluster 20 items into 4 clusters!

- Thus, the strategy having its own limitation is practical only if
  1) The sample is negatively small (~100-1000), and
  2) *k* is relatively small compared to *n* (i.e.. $k \ll n$).

Comments on k-Means algorithm

3. Distance Measurement:

- To assign a point to the closest centroid, we need a proximity measure that should quantify the notion of "closest" for the objects under clustering.
- Usually Euclidean distance (L_2 norm) is the best measure when object points are defined in n-dimensional Euclidean space.
- Other measure namely cosine similarity is more appropriate when objects are of document type.
- Further, there may be other type of proximity measures that appropriate in the context of applications.
- For example, Manhattan distance (L_1 norm), Jaccard measure, etc.

Comments on k-Means algorithm

3. Distance Measurement:

Thus, in the context of different measures, the sum-of-squared error (i.e., objective function/convergence criteria) of a clustering can be stated as under.

Data in Euclidean space (L_2 norm):

$$SSE = \sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} (c_i - x)^2$$

Data in Euclidean space (L_1 norm):

The Manhattan distance (L_1 norm) is used as a proximity measure, where the objective is to minimize the sum-of-absolute error denoted as SAE and defined as

$$SAE = \sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} |c_i - x|$$

Comments on k-Means algorithm

Distance with document objects

Suppose a set of *n* document objects is defined as *d* document term matrix (DTM) (a typical look is shown in the below form).

Document	Term			
	t_1	t_2	... ...	t_n
D_1	f_{11}	f_{12}		f_{1n}
D_2	f_{21}	f_{22}		f_{2n}
⋮				
D_n	f_{n1}	f_{n2}		f_{nn}

Here, the objective function, which is called Total cohesion denoted as TC and defined as

$$TC = \sum_{i=1}^{k} \sum_{x \in C_i} \cos(x, c_i)$$

where $\cos(x, c_i) = \frac{x \cdot c_i}{\|x\| \|c_i\|}$

$$x \cdot c_i = \sum_j x_j \, c_{ij} \quad \text{and} \quad \|x\| = \sqrt{\sum_j^p {x_j}^2}$$

$$\hat{x} = \sum_{j=1}^{p} \hat{x}_j \qquad \hat{c}_i = \sum_{j=1}^{p} \hat{c}_{ij} \qquad \|\|c_{ij}\|\| = \sqrt{\sum_j^p {c_{ij}}^2}$$

Comments on k-Means algorithm

Note: The criteria of objective function with different proximity measures

1. SSE (using L_2 norm) : To minimize the SSE.
2. SAE (using L_1 norm) : To minimize the SAE.
3. TC(using cosine similarity) : To maximize the TC.

Comments on k-Means algorithm

4. Type of objects under clustering:

- The k-Means algorithm can be applied only when the mean of the cluster is defined (hence it named k-Means). The cluster mean (also called centroid) of a cluster C_i is defied as

$$c_i = \frac{1}{n_i} \sum_{x \in C_i} x$$

- In other words, the mean calculation assumed that each object is defined with numerical attribute(s). Thus, we cannot apply the k-Means to objects which are defined with categorical attributes.
- More precisely, the k-means algorithm require some definition of cluster mean exists, but not necessarily it does have as defined in the above equation.
- In fact, the k-Means is a very general clustering algorithm and can be used with a wide variety of data types, such as documents, time series, etc.

?

How to find the mean of objects with composite attributes?

Comments on k-Means algorithm

Note:

1) When SSE (L_2 norm) is used as objective function and the objective is to minimize, then the cluster centroid (i.e. mean) is the mean value of the objects in the cluster.

2) When the objective function is defined as SAE (L_1 norm), minimizing the objective function implies the cluster centroid as the median of the cluster.

The above two interpretations can be readily verified as given in the next slide.

Comments on k-Means algorithm

Case 1: SSE

We know,

$$SSE = \sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} (c_i - x)^2$$

To minimize SSE means, $\frac{\partial (SSE)}{\partial c_i} = 0$

Thus,

$$\frac{\partial}{\partial c_i} \left(\sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} (c_i - x)^2 \right) = 0$$

Or,

$$\sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} \frac{\partial}{\partial c_i} (c_i - x)^2 = 0$$

Comments on k-Means algorithm

Or,

$$\sum_{x \in C_i} 2(c_i - x) = 0$$

Or,

$$n_i \cdot c_i = \sum_{x \in C_i} x$$

Or,

$$c_i = \frac{1}{n_i} \sum_{x \in C_i} x$$

- Thus, **the best centroid for minimizing SSE of a cluster is the mean of the objects in the cluster**.

Comments on k-Means algorithm

Case 2: SAE

We know,

$$SAE = \sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} |c_i - x|$$

To minimize SAE means, $\frac{\partial (SAE)}{\partial c_i} = 0$

Thus,

$$\frac{\partial}{\partial c_i} \left(\sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} |c_i - x| \right) = 0$$

Or,

$$\sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} \frac{\partial}{\partial c_i} |c_i - x| = 0$$

Comments on k-Means algorithm

Or,

$$\sum_{x \in \boldsymbol{C}_i} \left\{ (x - c_i)\Big|_{if\ x > \boldsymbol{c}_i} + (c_i - x)\Big|_{if\ \boldsymbol{c}_i > x} \right\} = 0$$

Solving the above equation, we get

$$c_i = median\,\{x | x \in \boldsymbol{C}_i\}$$

- Thus, the best centroid for minimizing SAE of a cluster is the median of the objects in the cluster.

?

Interpret the best centroid for maximizing TC (with Cosine similarity measure) of a cluster.

The above mentioned discussion is quite sufficient for the validation of k-Means algorithm.

Comments on k-Means algorithm

5. Complexity analysis of k-Means algorithm

Let us analyse the time and space complexities of k-Means algorithm.

Time complexity:

The time complexity of the k-Means algorithm can be expressed as

$$T(n) = O(n \times m \times k \times l)$$

where n = number of objects

m = number of attributes in the object definition

k = number of clusters

l = number of iterations.

Thus, time requirement is a linear order of number of objects and the algorithm runs in a modest time if $k \ll n$ and $l \ll n$ (the iteration can be moderately controlled to check the value of l).

Comments on k-Means algorithm

5. Complexity analysis of k-Means algorithm

Space complexity: The storage complexity can be expressed as follows.

It requires $n \times m$ space to store the objects and $n \times k$ space to store the proximity measure from n objects to the centroids of k clusters.

Thus the total storage complexity is

$$S(n) = O(n \times (m + k))$$

That is, space requirement is in the linear order of n if $k \ll n$.

Comments on k-Means algorithm

6. Final comments:

Advantages:

- k-Means is simple and can be used for a wide variety of object types.
- It is also efficient both from storage requirement and execution time point of views. By saving distance information from one iteration to the next, the actual number of distance calculations, that must be made can be reduced (specially, as it reaches towards the termination).

? How similarity metric can be utilized to run k-Means faster? What is the updation in each iteration?

Limitations:

- The k-Means is not suitable for all types of data. For example, k-Means does not work on categorical data because mean cannot be defined.
- k-means finds a local optima and may actually minimize the global optimum.

Comments on k-Means algorithm

6. Final comments:

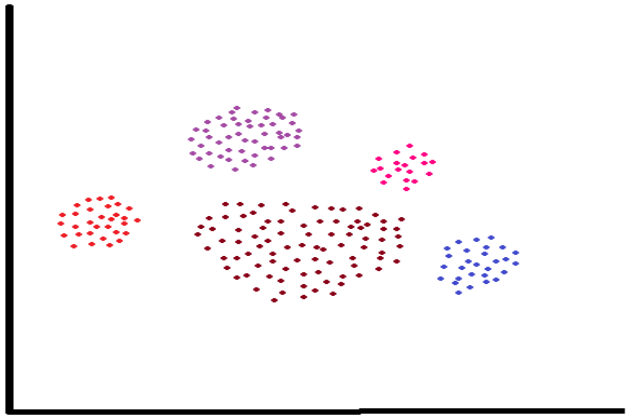
Limitations :

- k-means has trouble clustering data that contains outliers. When the SSE is used as objective function, outliers can unduly influence the cluster that are produced. More precisely, in the presence of outliers, the cluster centroids, in fact, not truly as representative as they would be otherwise. It also influence the SSE measure as well.

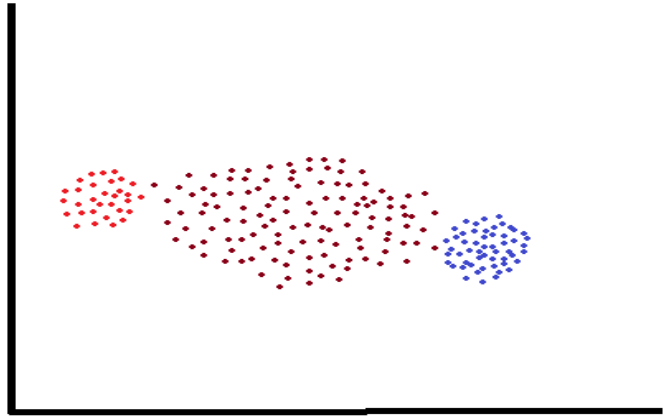
- k-Means algorithm cannot handle non-globular clusters, clusters of different sizes and densities (see Fig 16.6 in the next slide).

- k-Means algorithm not really beyond the scalability issue (and not so practical for large databases).

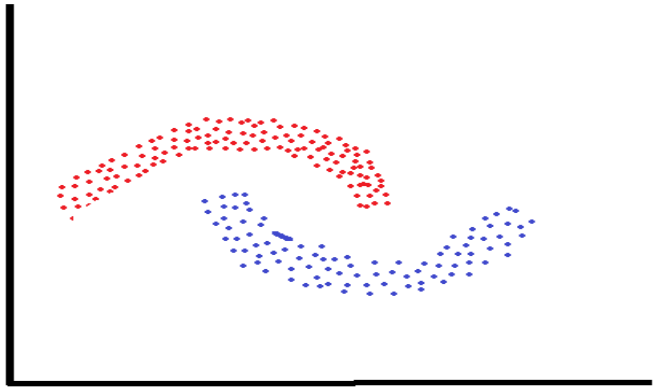
Comments on k-Means algorithm



Cluster with different sizes



Cluster with different densities



Non-convex shaped clusters

Fig 16.6: Some failure instance of k-Means algorithm

Different variants of k-means algorithm

There are a quite few variants of the k-Means algorithm. These can differ in the procedure of selecting the initial *k* means, the calculation of proximity and strategy for calculating cluster means. Another variants of k-means to cluster categorical data.

Few variant of k-Means algorithm includes

- Bisecting k-Means (addressing the issue of initial choice of cluster means).
  1. M. Steinbach, G. Karypis and V. Kumar "A comparison of document clustering techniques", *Proceedings of KDD workshop on Text mining*, 2000.
- Mean of clusters (Proposing various strategies to define means and variants of means).
  - B. zhan "Generalised k-Harmonic means – Dynamic weighting of data in unsupervised learning", *Technical report, HP Labs*, 2000.
  - A. D. Chaturvedi, P. E. Green, J. D. Carroll, "k-Modes clustering", *Journal of classification*, Vol. 18, PP. 35-36, 2001.
  - D. Pelleg, A. Moore, "x-Means: Extending k-Means with efficient estimation of the number of clusters", *17th International conference on Machine Learning*, 2000.

Different variants of k-means algorithm

- N. B. Karayiannis, M. M. Randolph, “Non-Euclidean c-Means clustering algorithm”, *Intelligent data analysis journal*, Vol 7(5), PP 405-425, 2003.
- V. J. Olivera, W. Pedrycy, “Advances in Fuzzy clustering and its applications”, Edited book. John Wiley [2007]. (Fuzzy c-Means algorithm).
- A. K. Jain and R. C. Bubes, “Algorithms for clustering Data”, Prentice Hall, 1988.

  Online book at http://www.cse.msu.edu/~jain/clustering_Jain_Dubes.pdf
- A. K. Jain, M. N. Munty and P. J. Flynn, “Data clustering: A Review”, *ACM computing surveys*, 31(3), 264-323 [1999]. Also available online.