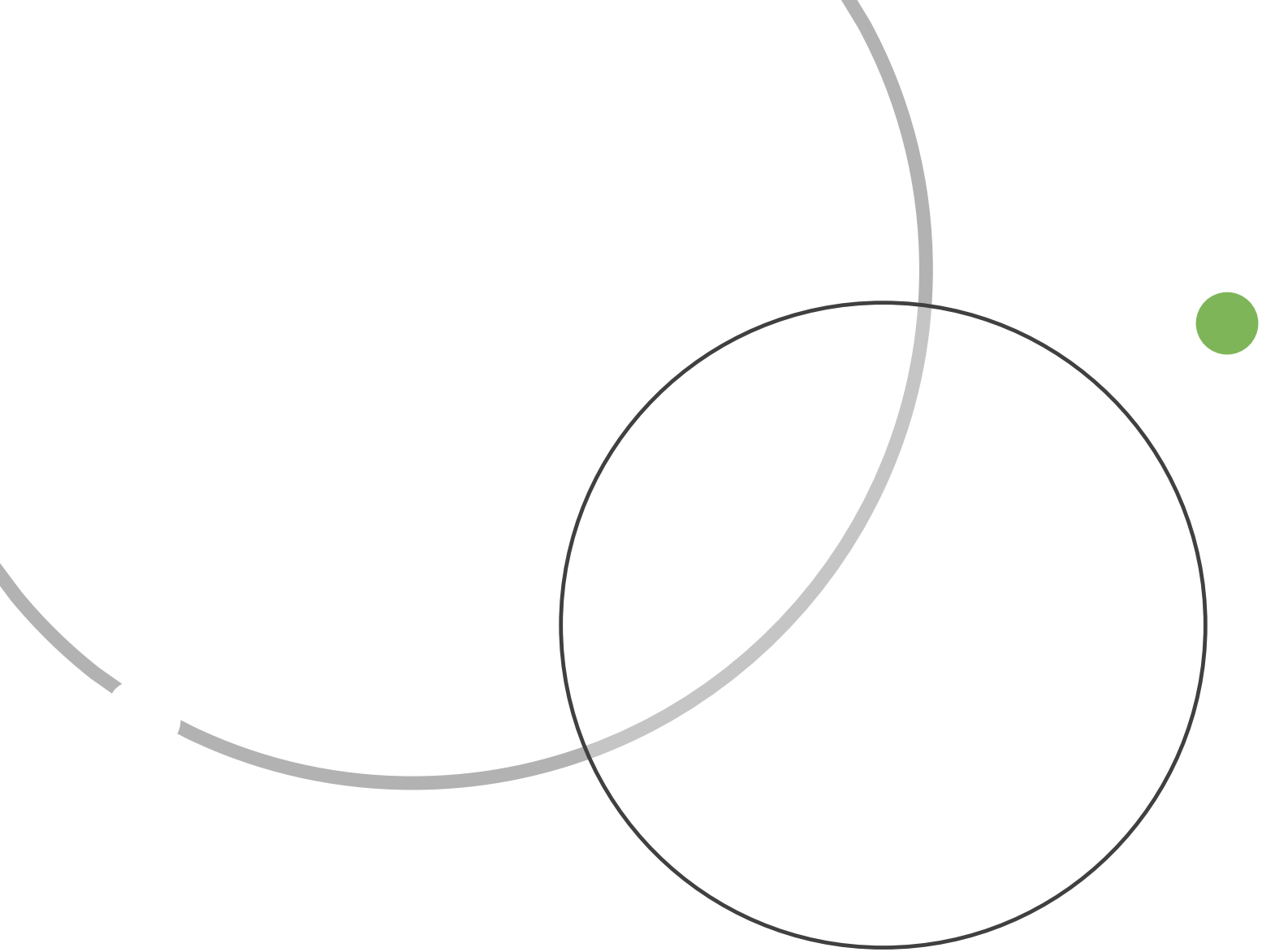


Clustering

Sources: LINK , LINK

Outline

- Definition
- Similarity measures
- Major clustering methodologies

What is Clustering?

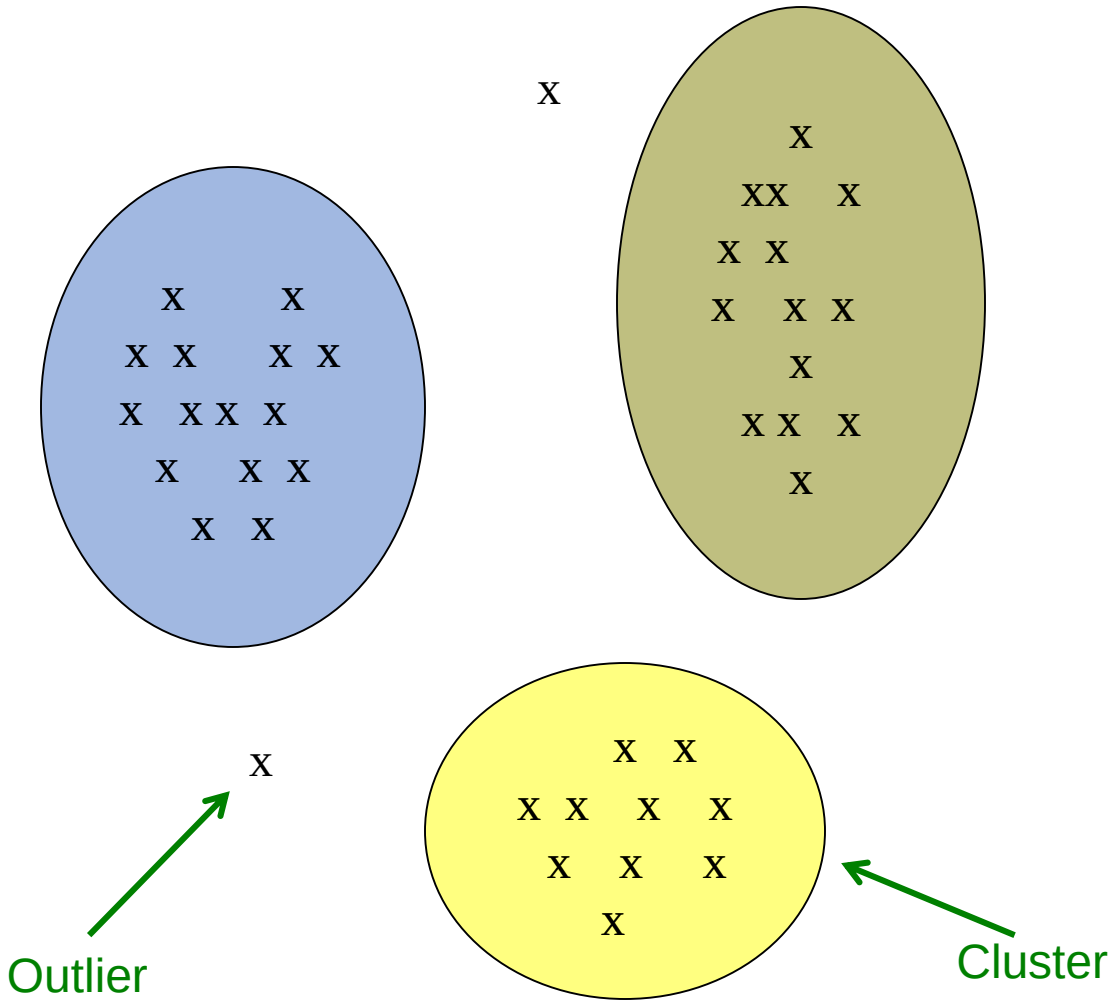
Ako *unsupervised learning*, sometimes called *sorting* by psychologists and *segmentation* by people in marketing

- Task: given a **set of objects (instances)** , with a notion of **distance (similarity)** between them, **group the objects** into some number of ***clusters***, so that
  - Members of a cluster are close/similar to each other
  - Members of different clusters are dissimilar
- **Usually:**
  - objects (vectors in a multi-dimensional space) are high-dimensional
  - Similarity is defined using a distance measure: Euclidean, Cosine, Jaccard, edit distance, …
  - Similarity measures are defined also for discrete (symbolic) features

Where is the «optimization» step?

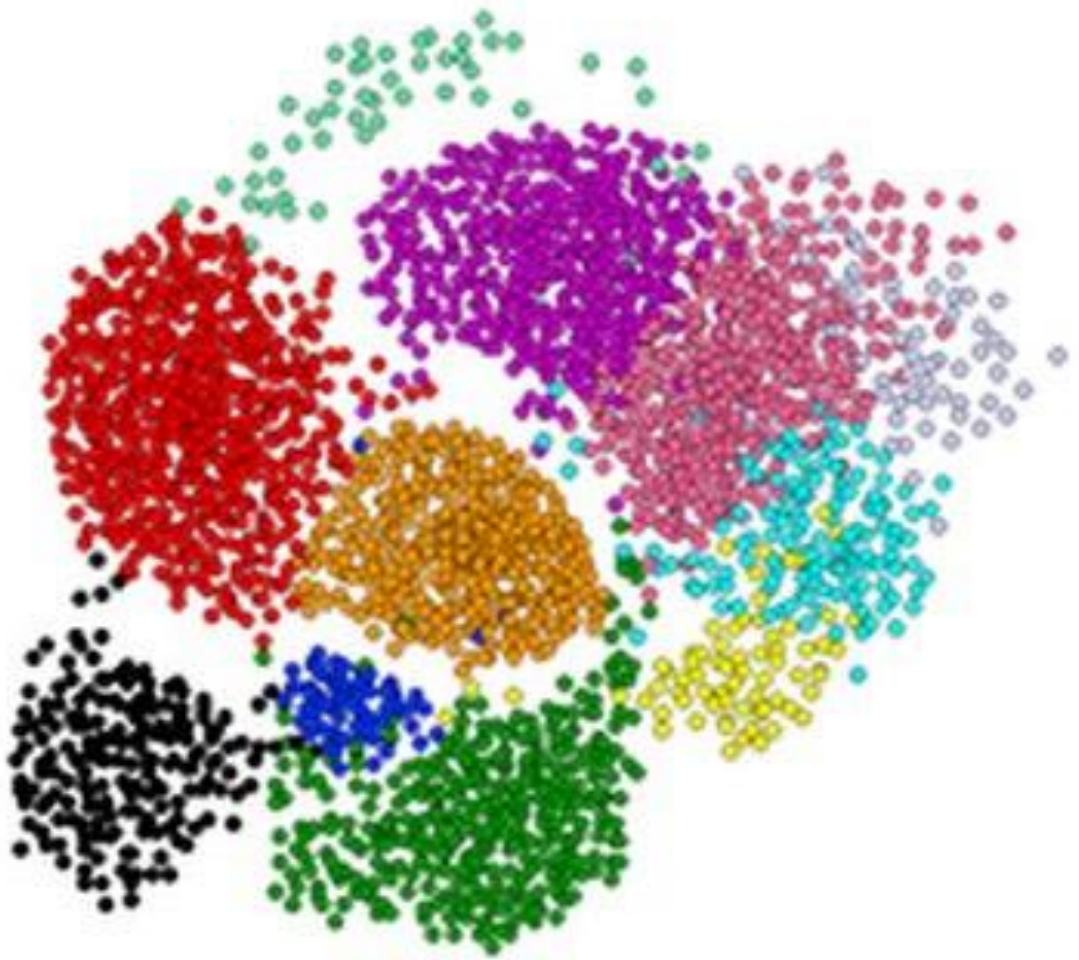
- Let’s reformulate the task:
- Given a **set of points**, with a notion of **distance (similarity)** between points, find the **optimal partition** of points, such that:
  - Similarity between members $\boldsymbol{x}_i^m$, $\boldsymbol{x}_j^m$ of the same cluster C_m is **maximized**
  - Dissimilarity between members $\boldsymbol{x}_i^m$, $\boldsymbol{x}_j^n$ of different clusters C_m, C_n is **maximized**

Example: Clusters & Outliers



Clustering is a NP-hard problem (especially in multi-dimensional spaces)!

Several NP-hardness results for finding optimal clusters exist

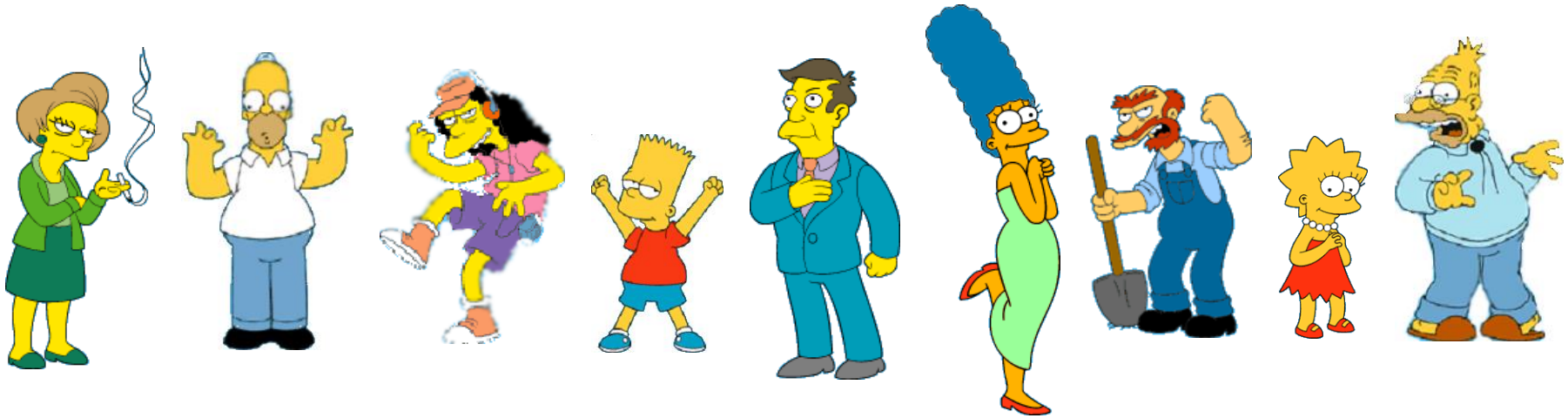


Datasets, http://www.mmds.org

Why is it hard?

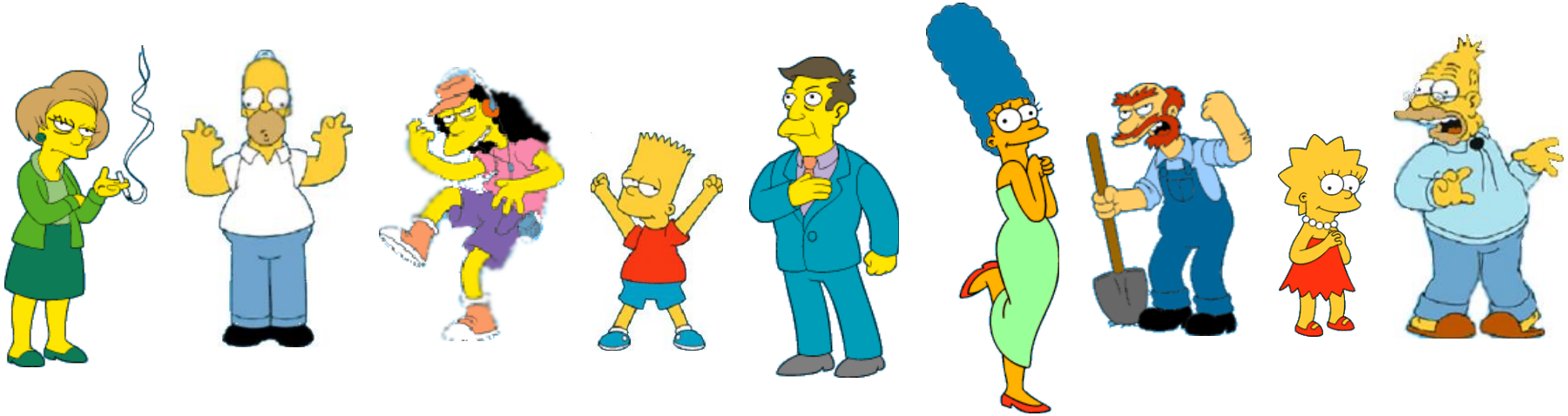
- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy
- But in most cases, **looks are not deceiving**
- Many applications involve not 2, but 100 or 10,000 dimensions
- **High-dimensional spaces "look" different:** Almost all pairs of points are at about the same distance when many dimensions are involved

What is a natural grouping among these objects?

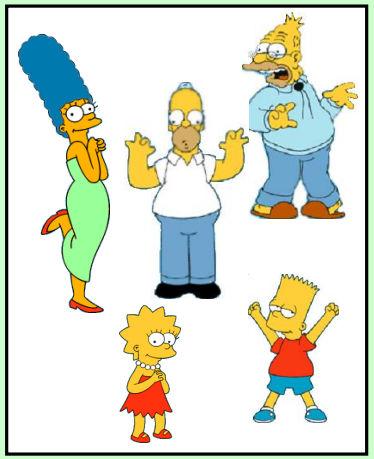


More important question: What are the «dimensions»?

What is a natural grouping among these objects?

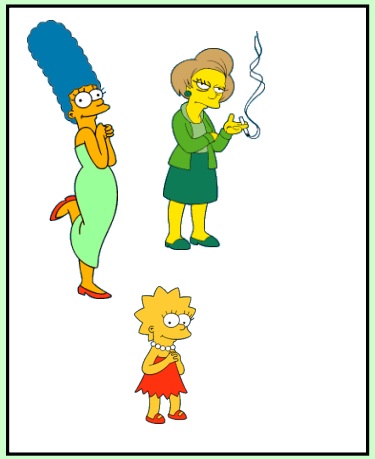


Many dimensions



Simpson's Family

School Employees

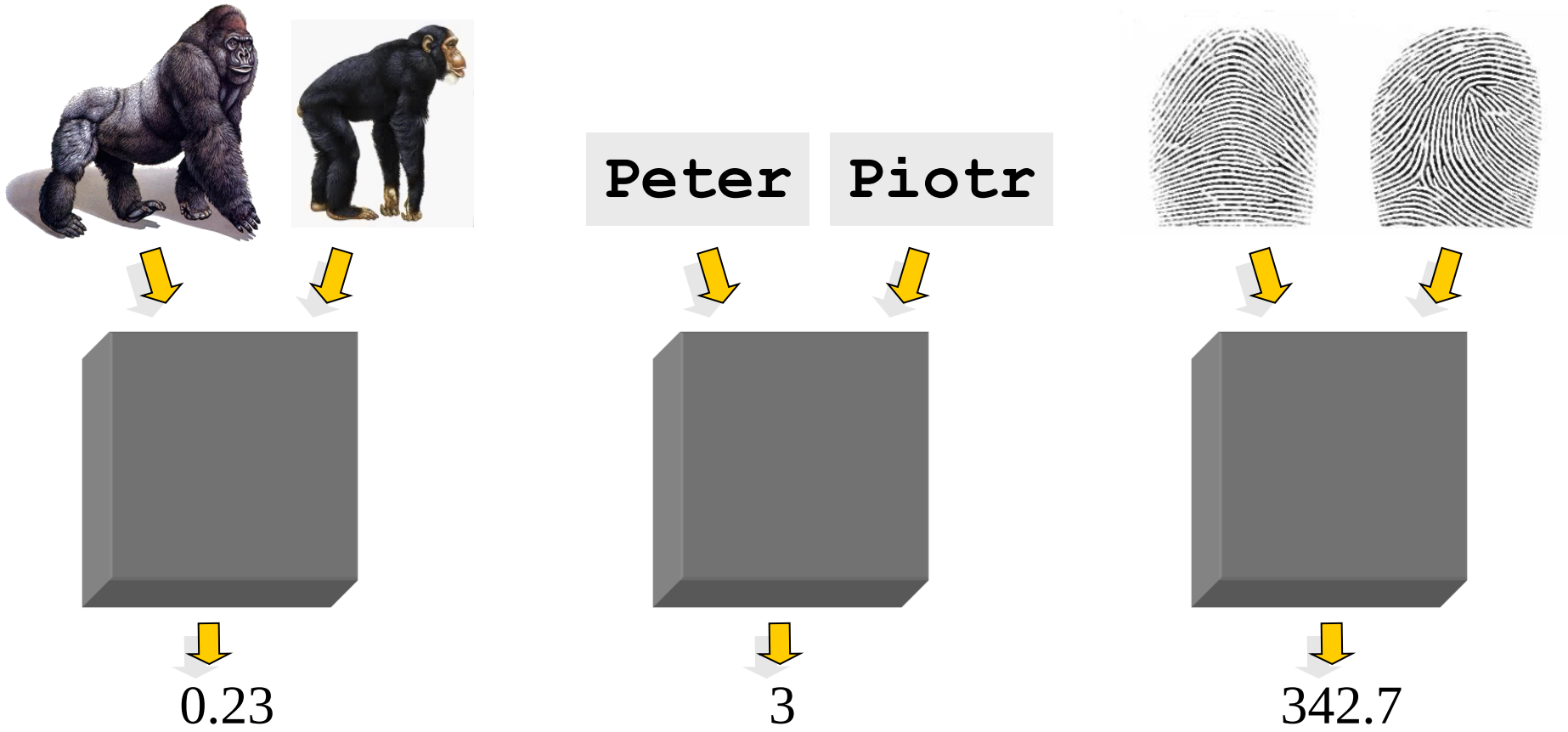


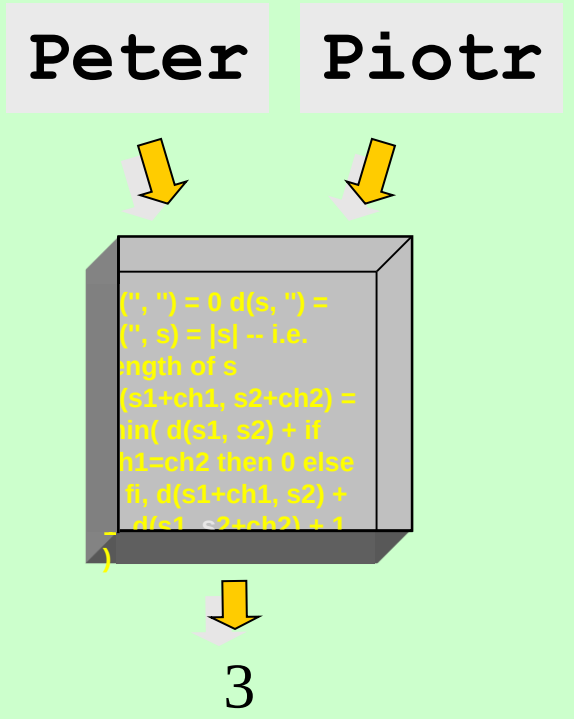
Females

Males

Defining (dis)similarity measures

Definition: Let x_1 and x_2 be two objects from the universe of possible objects. The **distance** (dissimilarity) between x_1 and x_2 is a real number denoted by $d(x_1, x_2)$





(Dis)similarity functions might very simple or very complex.
In either case, **what properties should these functions have?**

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*

Intuitions behind desirable distance measure properties

$d(A,B) = d(B,A)$ *Symmetry*

Otherwise you could claim "Alex looks like Bob, but Bob looks nothing like Alex."

$d(A,A) = 0$ *Constancy of Self-Similarity*

Otherwise you could claim "Alex looks more like Bob, than Bob does."

$d(A,B) = 0$ IIF $A=B$ *Positivity (Separation)*

Otherwise there are objects in your world that are different, but you cannot tell apart.

$d(A,B) \leq d(A,C) + d(B,C)$ *Triangular Inequality*

Otherwise you could claim "Alex is very like Bob, and Alex is very like Carl, but Bob is very unlike Carl."

https://arjun010.github.io/static/papers/voder-infovis18.pdf

What type of measures?

- There is **no single definition** of similarity or dissimilarity between data objects (instances)
- The definition of **similarity or dissimilarity between objects** depends on
  - the **type** of the data considered
  - what **kind** of similarity we are looking for

What types of data?

- **Interval-scaled variables**
- **Binary variables**
- **Nominal, ordinal, and ratio variables**
- **Variables of mixed types**
- **Complex data types**

Interval-scaled variables (1)

- **Def: Continuous measurements** (values) of a roughly linear scale
- For example, weight, height, price..
- The **measurement unit** can affect the cluster analysis (meters or feets?)
- To avoid dependence on the measurement unit, we should **standardize (normalize)** the data

Interval-scaled variables (2)

Note MAD is different from Standard Deviation! In SD you compute the **square** of differences, you average,and then you take the squareroot

To standardize n=|D| measurements of an I-S variables *X*:

- calculate the **mean absolute deviation** of *X* in the dataset D

$$MAD(X,D) = \frac{1}{|D|}\left(|x_1 - \mu_{X,D}| + |x_2 - \mu_{X,D}| + \cdots |x_{|D|} - \mu_{X,D}|\right)$$

where x_i *i*=1..|D| are the *observed values* for feature X in D, and $\mu_{X,D}$ is the mean

- calculate the **standardized measurement (*z-score*)** for each observed value

$$z_{iX} = \frac{x_i - \mu_{X,D}}{MAD(X,D)}$$

Interval-scaled variables: distance measures (1)

- One group of popular distance measures for interval-scaled variables are **Minkowski distances**
- $\mathbf{x}_i$ and $\mathbf{x}_j$ are two instances in D and x_{ik}, x_{jk} are the observed *values* of p interval-scaled variables (features) of $\mathbf{x}_i$ and $\mathbf{x}_j$ (x_{ik} is observed value of k-th feature of instance x_i)

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{ip} - x_{jp}|^q)}$$

where ***q* is a positive integer.**

We assume standardized variables (otherwise they would not stand on the same level, and those with largest values would dominate in computing the similarity)**.**

Interval-scaled variables: distance measures (2)

- If *q = 1*, the distance measure is called **Manhattan (**or **city block) distance**

$$d(\boldsymbol{x}_i, \boldsymbol{x}_k) = \sum_{k=1}^{p} \left| x_{ik} - x_{jk} \right|$$

- If *q = 2*, the distance measure is called **Euclidean distance**

$$d(\boldsymbol{x}_i, \boldsymbol{x}_k) = \sqrt{\sum_{k=1}^{p} \left(x_{ik} - x_{jk} \right)^2}$$

Binary variables

- They assume two values, that we can associate to 0 and 1

Binary variables: distance measures

- A binary variable X has only **two values**: 0 or 1
- Use a contingency table to pairwise compare feature values of 2 instances $\mathbf{x}_i$ and $\mathbf{x}_j$ each with *p* binary features:

		$\mathbf{x}_i$		
		1	0	*sum*
$\mathbf{x}_j$	1	a_{11}	b_{10}	$a_{11} + b_{10}$
	0	c_{01}	d_{00}	$c_{01} + d_{00}$
	sum	$a_{11} + c_{01}$	$b_{10} + d_{00}$	p

a, b, c, d are integers, e.g. **a_{11}**= number of times x_{ik} and x_{jk} with k=1..p **are both 1** (corresponding features are both 1)

Example

```
(0,1,0,0,0,1,0,0,1,1)
(0,0,1,0,0,0,0,0,1,1)
```

$a_{11}= 2$

$d_{00}= 5$

$b_{10}= 2$

$c_{01}= 1$

Binary variables: computing similarity (1)

Based on contingency table

- **Simple matching coefficient** (invariant similarity , if the binary variable is **symmetric**):

$$d(\boldsymbol{x}^i, \boldsymbol{x}^j) = \frac{b_{01} + c_{10}}{a_{11} + d_{00} + b_{01} + c_{10}}$$

How many times corresponding binary features of xi and xj are different

- **Jaccard coefficient** (noninvariant similarity, if the binary variable is **asymmetric**):

$$d(\boldsymbol{x}^i, \boldsymbol{x}^j) = \frac{b_{01} + c_{10}}{a_{11} + b_{01} + c_{10}}$$

SMC considers in denominator both matching 1 (a) and matching 0 (d), while Jaccard only considers **matching 1** (a).

Note a binary variable is **symmetric** if the two values have equal probability, like, e.g., gender. It is «invariant» in that nothing changes if we assign «1» to the value female or to the value male.

When using SMC and when Jaccard

An example (extended from Wikipedia):

- market basket analysis (asymmetry): consider a supermarket with 1000 products and two customers. The basket of the first customer contains salt and pepper and the basket of the second contains salt and sugar.
- Note: «objects» are baskets, binary variables are products (1 present, 0 absent). In this scenario, the similarity between the two baskets as measured by the Jaccard distance would be $=\frac{1+1}{1+1+1}$=2/3, but the distance becomes 0.002 using the SMC (since there are 997 products which are neither in one basket nor in the other, so SMC= $\frac{1+1}{1000}$).
- Here the problem is that, for every product, the probability of being NOT in a basket is >> than that of being in the basket. Therefore, SMC is NOT a good measure.

When using SMC and when Jaccard

- Demographic variables such as gender, would be better compared with the SMC than with the Jaccard index since the impact of gender on similarity should be equal, independently of whether male is defined as a 0 and female as a 1 or the other way around.
- Example: our objects are patients with diseases. We would like to compute the similarity between patients, and gender is one attribute.
- Example follows..

When using SMC and when Jaccard

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4	Test-5
Jack	M	Y	N	P	N	N	N	Y
Mary	F	Y	N	P	N	P	N	N
Jim	M	Y	P	N	N	N	N	N
Jenny	F	N	N	P	N	N	N	Y

- 9 attributes, of which
  - Given the dataset, let's say that Gender and Test-5 are **symmetric** binary attribute, and
  - the remaining attributes are **asymmetric** binary

When using SMC and when Jaccard

- Let the values **Y**(yes) and **P**(positive) be set to 1, and the values **N**(no, negative) be set to 0
- Compute distances between patients based on the asymmetric variables by using Jaccard coefficient

$$d(jack, mary) = \frac{1+0}{2+0+1} = 0.33$$

Note that this **varies** if we invert assignment of 0s and 1s

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{2+1}{1+1+2} = 0.75$$

$$Jaccard(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{b_{01} + c_{10}}{a_{11} + b_{01} + c_{10}}$$

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4	Test-5
Jack	M	1	0	1	0	0	0	Y
Mary	F	1	0	1	0	1	0	N
Jim	M	1	0	0	0	0	0	N
Jenny	F	0	1	1	0	0	0	Y

When using SMC and when Jaccard

- For the symmetric attributes, let the values **Y** and **F** (female) be set to 1. Compute distances between patients based on the symmetric variables by using SMC coefficient

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4	Test-5
Jack	0	Y	N	P	N	N	N	0
Mary	1	Y	N	P	N	P	N	1
Jim	0	Y	P	N	N	N	N	0
Jenny	1	N	N	P	N	N	N	1

$$d(jack, mary) = \frac{2+0}{0+0+2+0} = 1$$

$$d(jack, jim) = \frac{0+0}{0+2+0+0} = 0$$

$$smc(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{b_{01} + c_{10}}{a_{11} + d_{00} + b_{01} + c_{10}}$$

Nominal (categorial) variables

- They can take **more than 2 values**, e.g., red, yellow, blue, green (M total values).
- Method 1: **simple matching**
  - **m:** # of matches (same value of correspondent categorical features)**, p:** total # of categorical features
    - $d(\boldsymbol{x}^i, \boldsymbol{x}^j) = \frac{p-m}{p}$
- Method 2 (binary encoding): **use a large number of binary variables**
  - create a new binary variable for each of the $\mathbf{m_j}$ nominal values of feature Xj (rather than Xj=color, we have a "one-hot" feature vector with m_j binary features: red, yellow, blue… the feature vector can have one value set to 1 and the others to 0 – since feature values are mutually exclusive)

Example of transformation into one-hot vector

- X: (color,shape) color:red, yellow, blue (R,Y,B) shape: large, medium, small (L,M,S)
- Every categorical feature with n values is split into a «one hot vector» with n features. A one hot vector is a binary vector where one feature is 1 and all the others are 0
  - color ➔(color-red,color-yellow,color-blue)
  - Shape➔(shape-large,shape-medium,shape-small)
- **x**:(blue,medium) ➔ **x**:((0,0,1)(0,1,0))
- Now we can compute similarity like for binary valued features

Ordinal variables

- An ordinal variable can be **discrete or continuous**
- **Order of values is important**, e.g., grades of students, timestamps..
- Can be treated like **interval-scaled**
  1. Order the observed feature values of a variable (e.g., 0.1, 40.5, 3.2 ➔ 0.1, 3.2,40.5)
  2. Replace feature values x_{ik} by their rank in |D|=*n* observations
     (e.g.: 0.1, 3.2 , 40.5 ➔ 1,2,3) $r(x_{ik}) \in \{1 \dots n\}$
  3. Map values of each variable Xi onto [0, 1] by replacing *k*-th value in the *i*-the variable by:

$$z_{ik} = \frac{r_{ik} - 1}{n - 1}$$

(e.g. 40.5 becomes (3-1)/(3-1)=1 3,2 becomes (2-1)/(3-1)=0.5 etc.)

In this way we preserve pairwise distances!

4. compute the dissimilarity using **distance methods for interval-scaled variables**

Ratio-scaled variables

- **Def: A positive measurement on a nonlinear scale**, approximately at exponential scale
  - for example, Ae^{Bx} or Ae^{-Bx} with A,B constant values
- **Method:**
  - apply logarithmic transformation $y_{ik} = log(x_{ik})$
  - treat them as continuous ordinal variables and treat them as for interval-scaled

Variables of mixed types (1)

- A dataset may contain features belonging to **all the six types of variables**
- One may use a **weighted formula** to combine their effects:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sum_{k=1}^{p} {\delta_{ij}}^{k} d_{ij}^{k}}{\sum_{k=1}^{p} {\delta_{ij}}^{k}}$$

where ${\delta_{ij}}^{k}$=0 if values x_{ik} or x_{jk} are missing, or if they are both =0, otherwise ${\delta_{ij}}^{k}$=1,

d_{ij}^{k} is the "appropriate" distance measure between correspondent values x_{ik} and x_{jk} in $\mathbf{x_i}$ and $\mathbf{x_j}$ for variable X_k

Variables of mixed types (2)

Contribution of variable X_k to distance $d(\boldsymbol{x_i},\boldsymbol{x_j})$:

- if X_k is binary or nominal:

  d_{ij}^k=0 if x_{ik}=x_{jk} otherwise d_{ij}^k=1

- if X_k is interval-based: use $d_{ij}^k = q\sqrt{\left(|x_{ik} - x_{jk}|\right)^q}$

- if X_k is ordinal or ratio-scaled
  - compute ranks r_{ik} and $z_{ik} = \frac{r_{ik} - 1}{n - 1}$
  - treat z_{ik} as interval-scaled

Complex data types (1)

- All objects considered in data mining that are not relational => **complex types of data**
  - examples of such data are spatial data (coordinates), multimedia data, genetic data, time-series data, text data
  - Often totally different similarity or dissimilarity measures than those listed so far
  - For example, we may need to use
    - string and/or sequence matching,
    - or methods of information retrieval (e.g., based on *semantic similarity*),
    - or vector similarity based on deep compressed representations of the initial objects

Example of complex data types: sequences

Method: EDIT DISTANCE

It is possible to transform any string *Q* into string *C*, using only 3 operations: *Substitution*, *Insertion* and *Deletion*.

Assume that each of these operators has a cost associated with it.

The similarity between two strings can be defined as the **cost of the cheapest transformation from *Q* to *C*.**

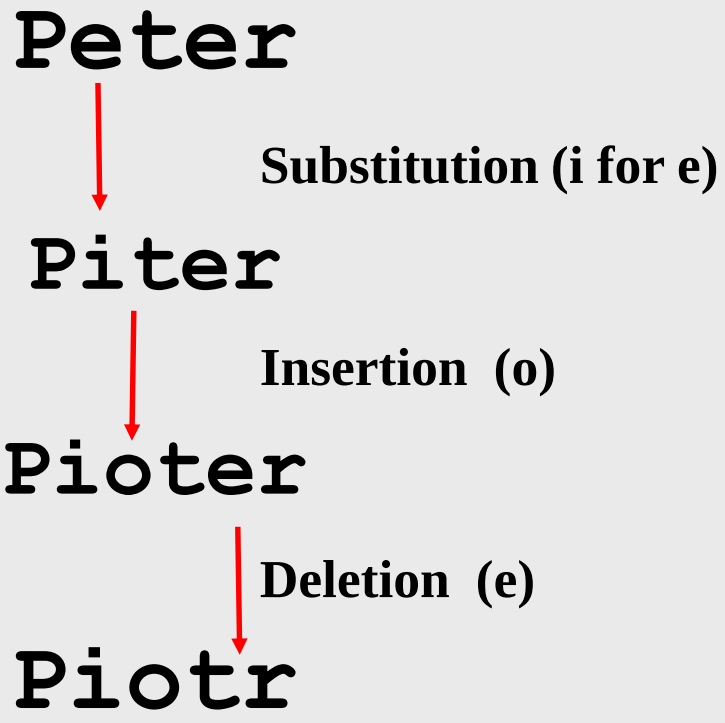
The edit distance can be generalized to sequences other than text (e.g. aminoacid)

How similar are the names "Peter" and "Piotr"?

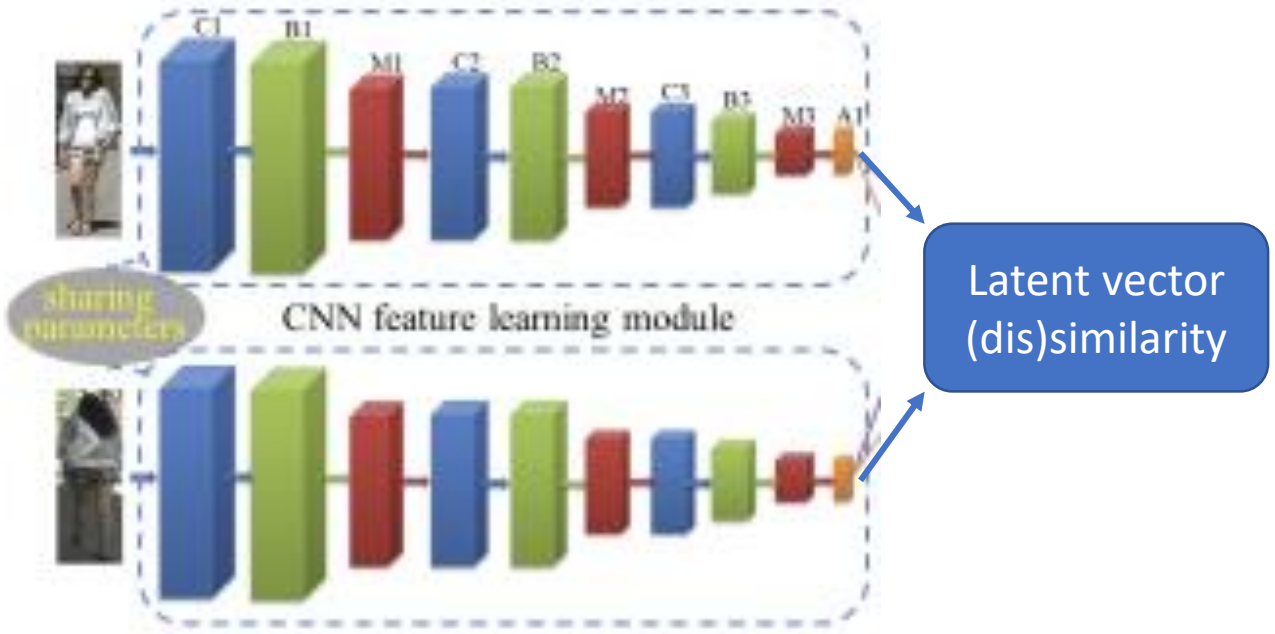
Assume the following cost function

Substitution	1 Unit
Insertion	1 Unit
Deletion	1 Unit

d(`Peter`, `Piotr`) is 3



Similarity among compressed image representations



Other types of complex data

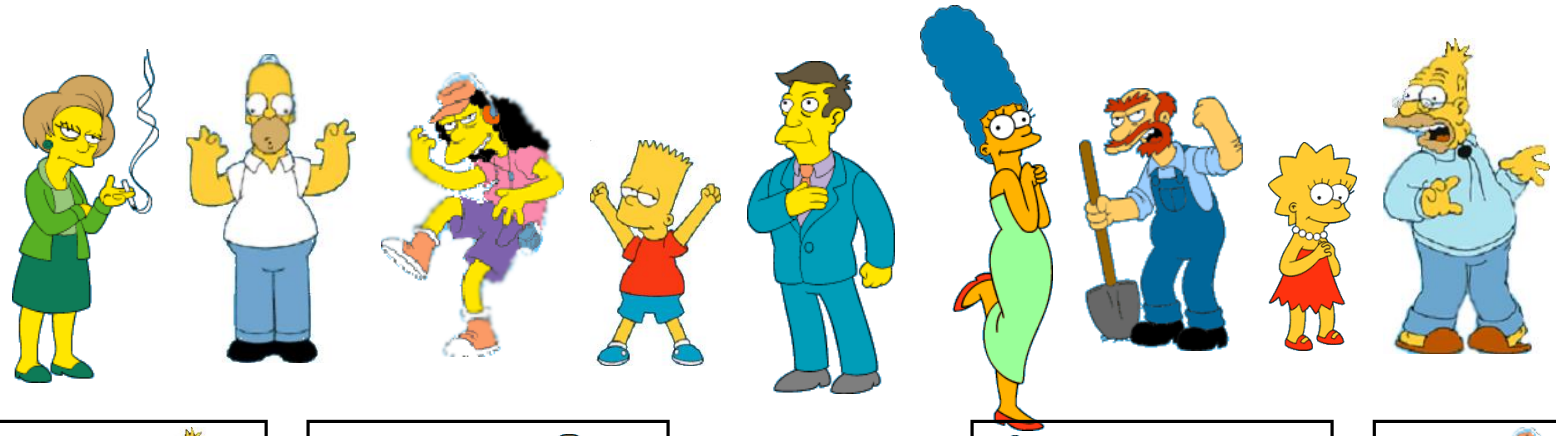
May require defining «ad-hoc» distance measures

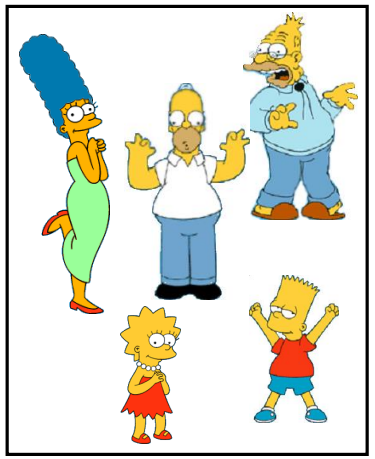
Major Clustering methodologies

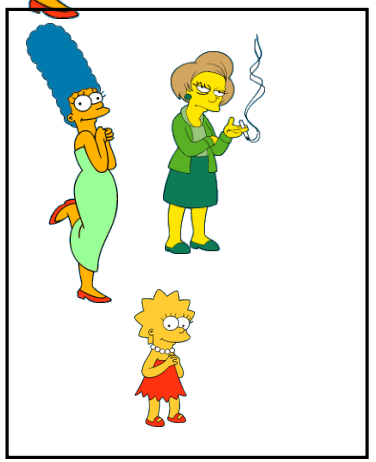
Major Clustering Methodologies (1)

Partitioning Methodology

- Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square distances
- Typical methods: K-means, K-medoids, ……



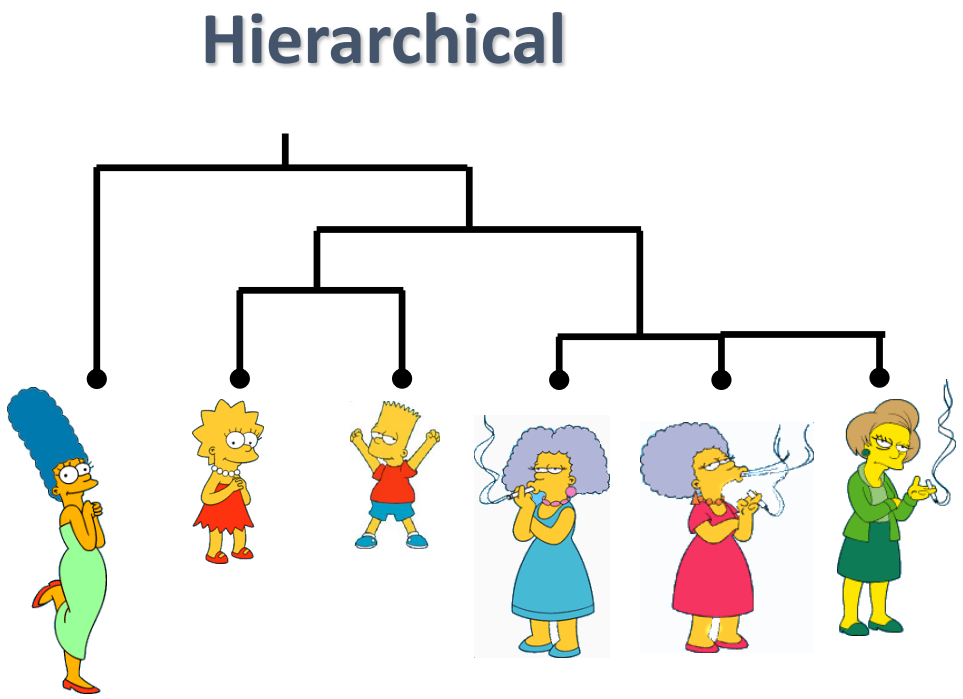




Major Clustering Methodologies (2)

Hierarchical Methodology

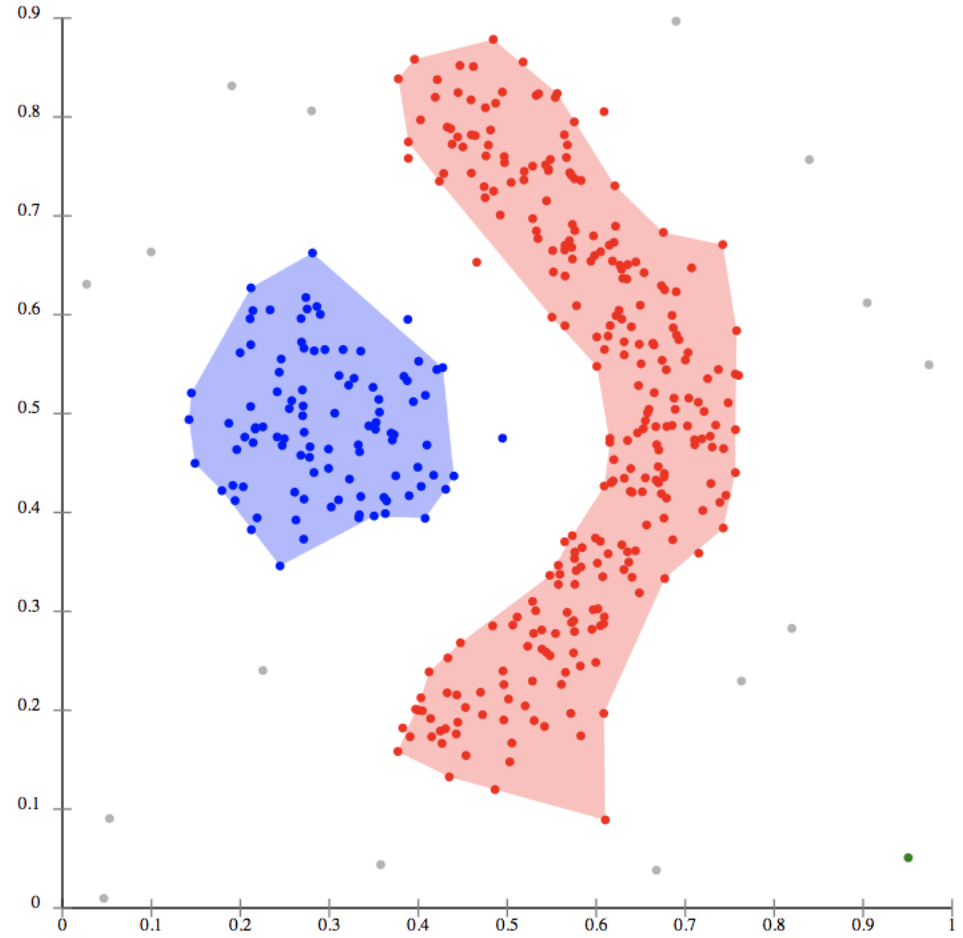
- Create a hierarchical decomposition of the set of data (or objects) using some criterion
- Typical methods: Agglomerative, BIRCH, ROCK, ……



Major Clustering Methodologies (3)

Density-based Methodology

- Based on connectivity and density functions (task is to identify “densest” regions)
- Typical methods: DBSACN, OPTICS, DenClue, ……

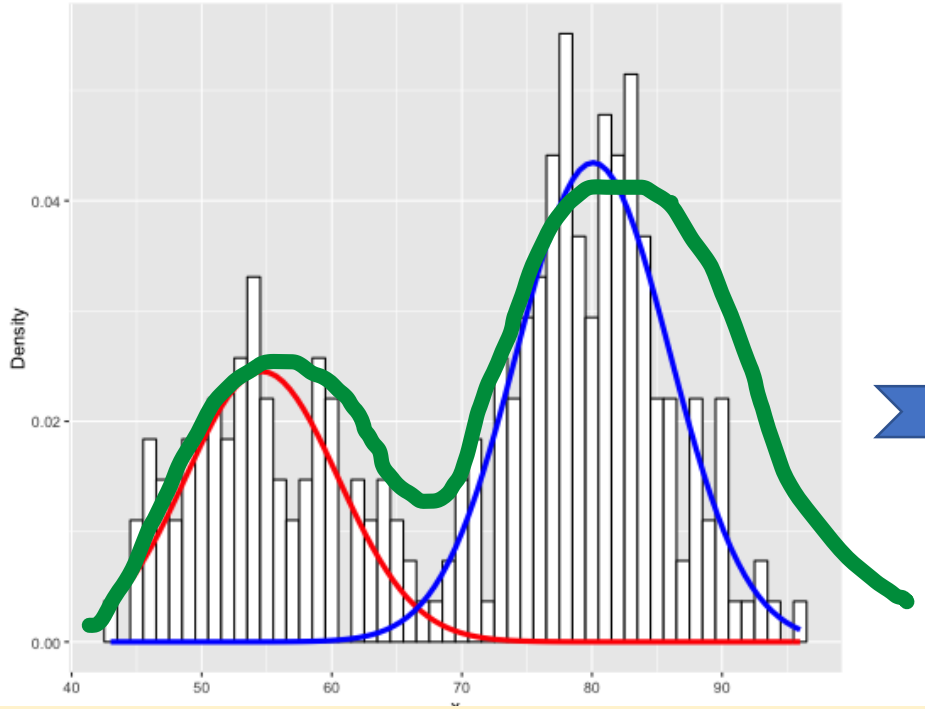


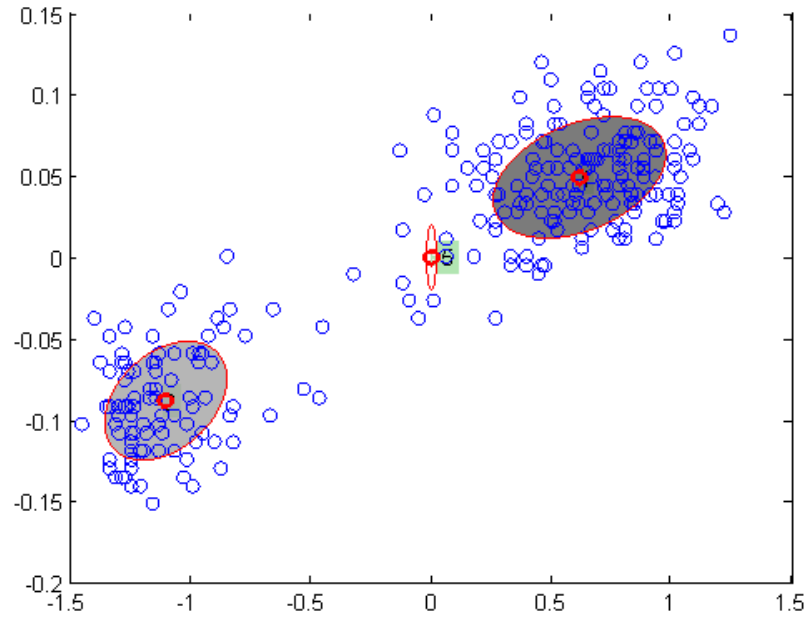
https://commons.wikimedia.org/w/index.php?curid=17085332

Major Clustering Methodologies (4)

Model-based (generative) Methodology

- A **generative** model (see slides on probabilistic ML methods) is hypothesized for each of the clusters and the objective is to find parameters of the model that optimally fit the data
- Typical methods: **Gaussian Mixture Model** (GMM), COBWEB, ……



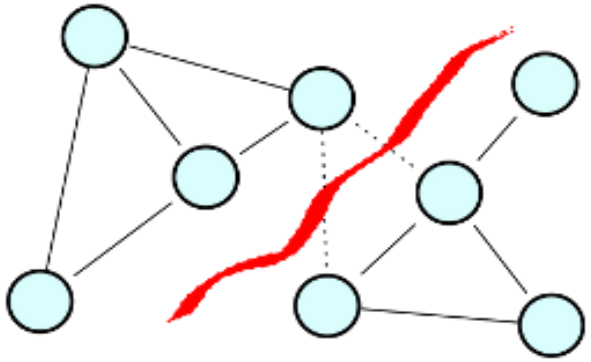


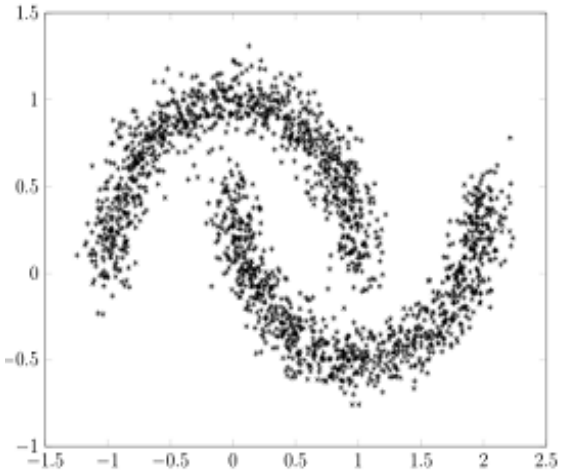
In GMM data are supposed to be generated by a mixture of Gaussian functions.
Task is to estimate model parameters as in MLE. Note that here we do not have examples of the various classes!
i.e., we observe the «mixture» (green distribution) and we must infer the components (blue and red)

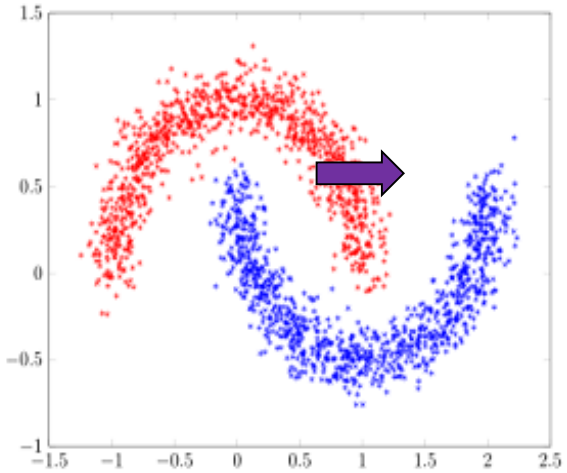
Major Clustering Methodologies (5)

Spectral clustering Methodology

- Convert data set into weighted graph (e.g., edges can be pairwise similarities) , then cut the graph into sub-graphs corresponding to clusters via spectral analysis
- Typical methods: Normalised-Cuts, ……





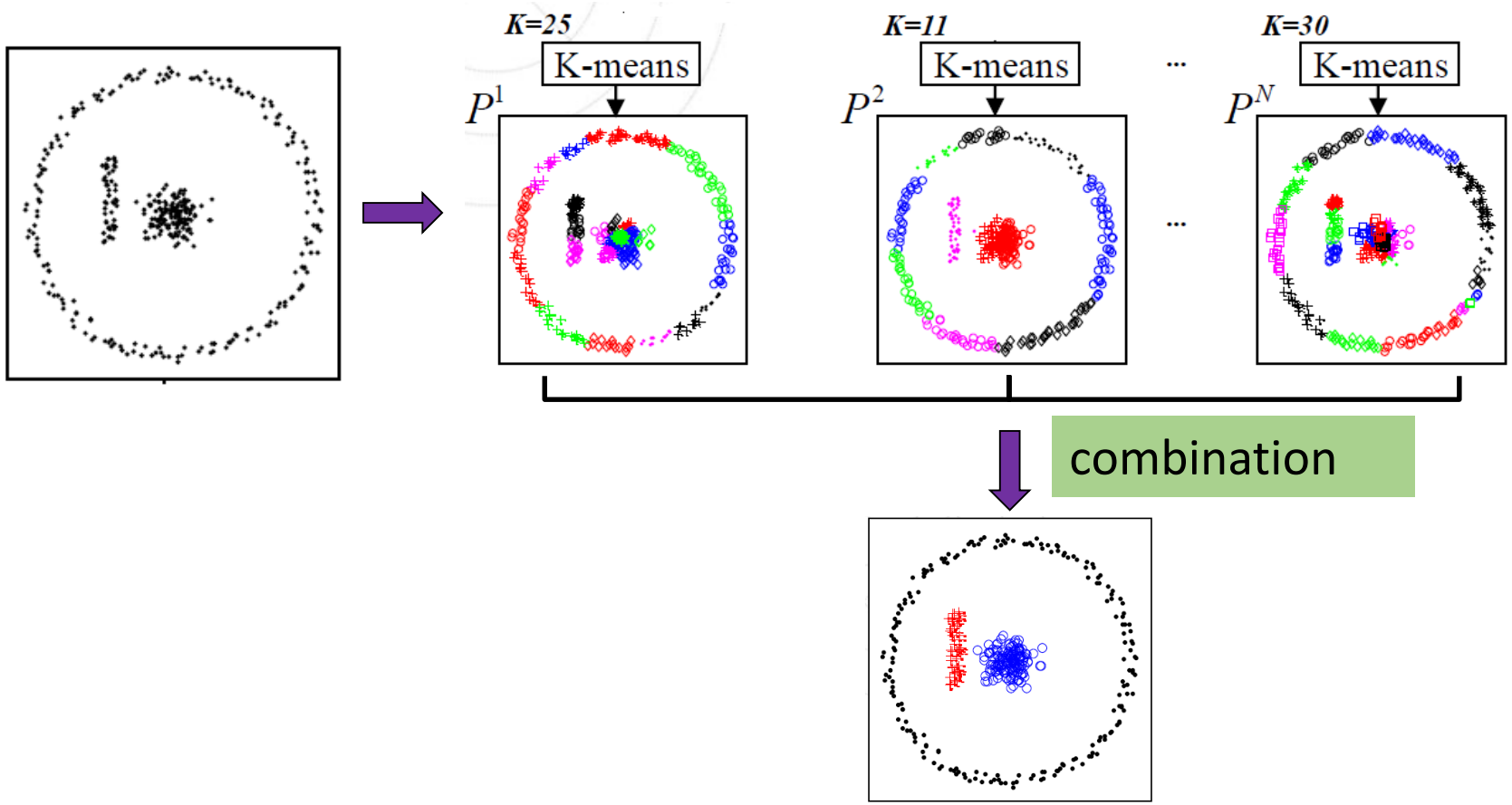


Note that your data can **already be in the form of a graph**, e.g. Social networks

Major Clustering Methodologies (6)

Clustering Ensembles

- Combine multiple clustering results (different partitions) obtained with simple methods
- Typical methods: Evidence-accumulation based, graph-based ……



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 hyperparameters (e.g., number of clusters)
- Able to deal with noise and outliers
- Insensitive to order of input records
- Incorporation of user-specified constraints

Hierarchical Clustering algorithms

- **Agglomerative (bottom-up):**
  - Start with each instance being a single cluster.
  - Eventually all instances belong to the same cluster.
- **Divisive (top-down):**
  - Start with all instances belong to the same cluster.
  - Eventually each instance forms a cluster on its own.
- Does not require to specify the number of clusters ***k*** in advance
- Needs a termination condition

Hierarchical Agglomerative Clustering (HAC)

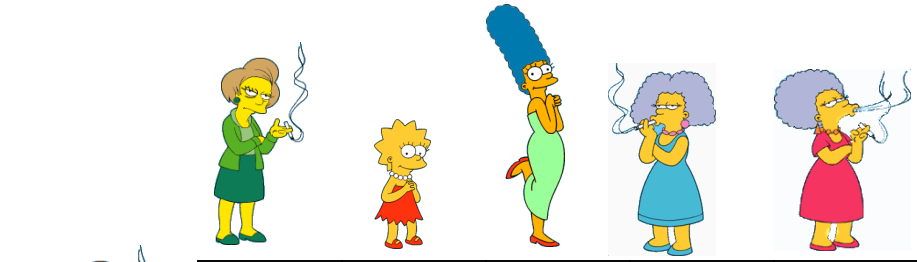
- Assumes a **similarity function** for determining the similarity of two instances (see previous slides).
- Starts with all instances in a separate cluster and then repeatedly joins the two clusters **that are most similar** until there is only one cluster.
- The “history of merging” forms a binary tree or hierarchy.

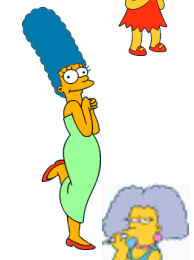
Hierarchical Agglomerative Clustering (HAC) (2)

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

d(,) = 8

d(,) = 1



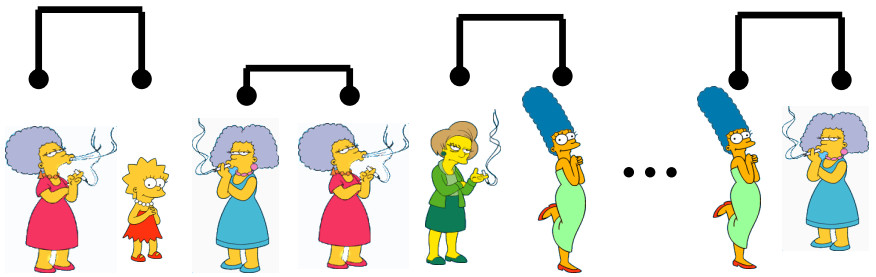


0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

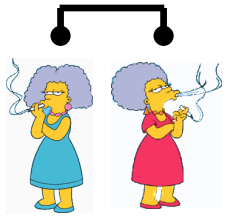
Hierarchical Agglomerative Clustering (HAC) (2)

Bottom-Up (agglomerative):
Starting with each item in its own cluster, find the best pair to merge into a new cluster.

Consider all possible merges…



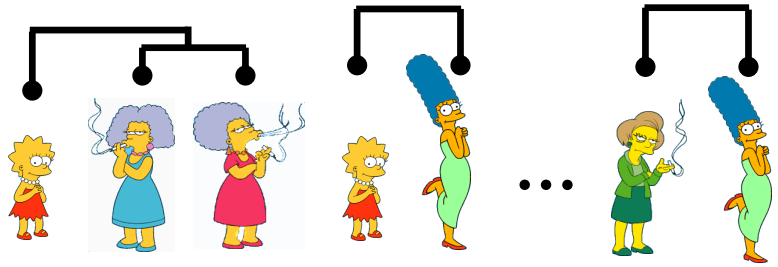
Choose the best



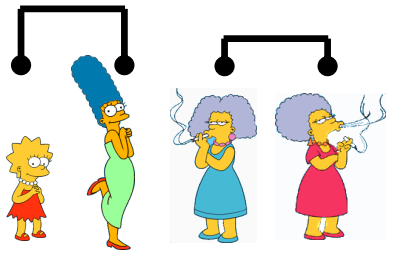
Hierarchical Agglomerative Clustering (HAC) (3)

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.**

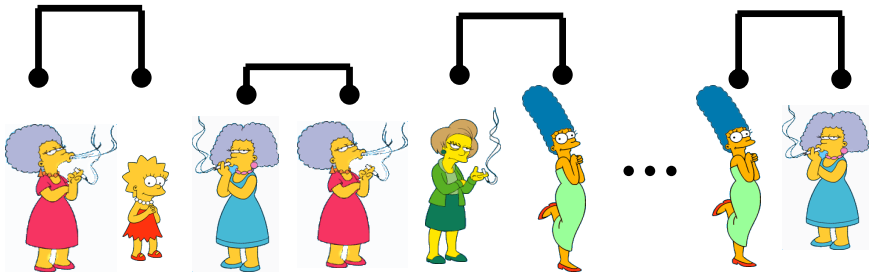
Consider all possible merges…



Choose the best

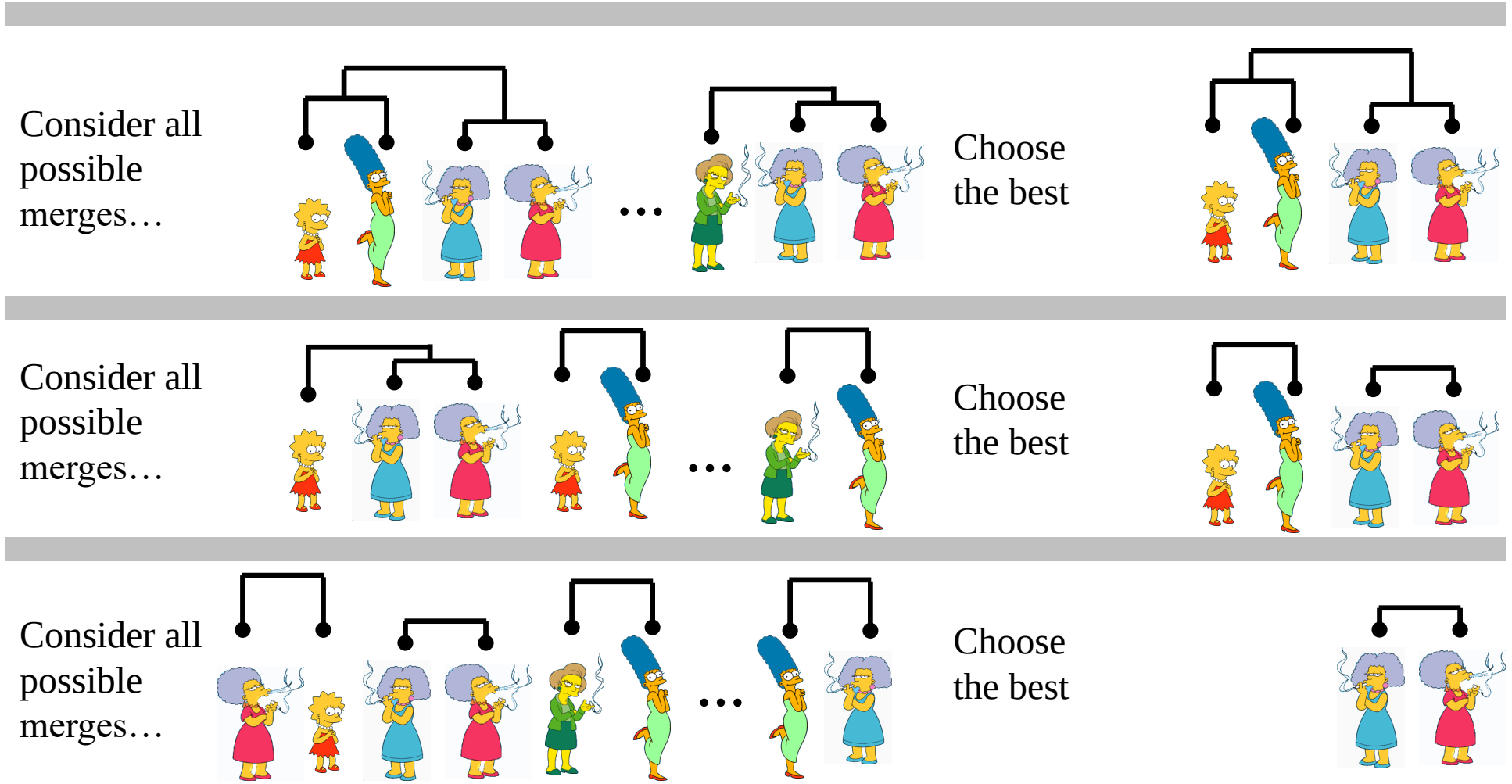


Consider all possible merges…

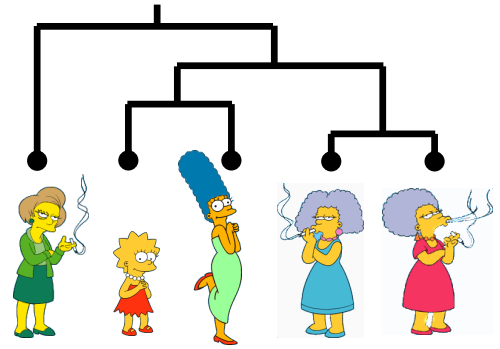


Choose the best

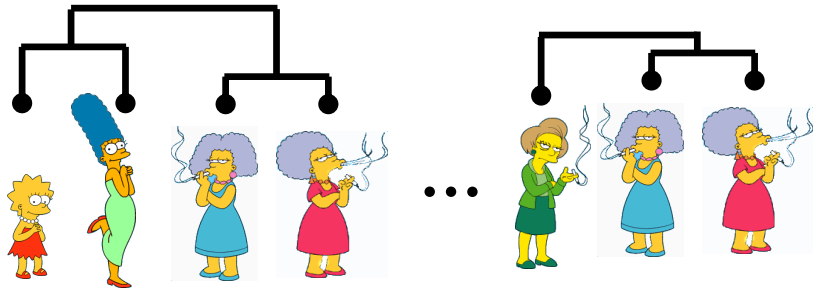
Hierarchical Agglomerative Clustering (HAC) (4)



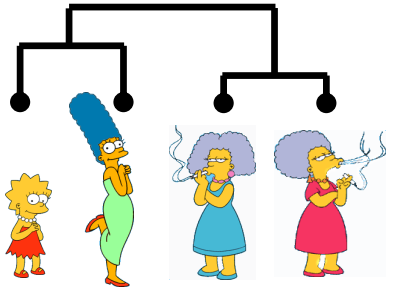
Hierarchical Agglomerative Clustering (HAC) (5)



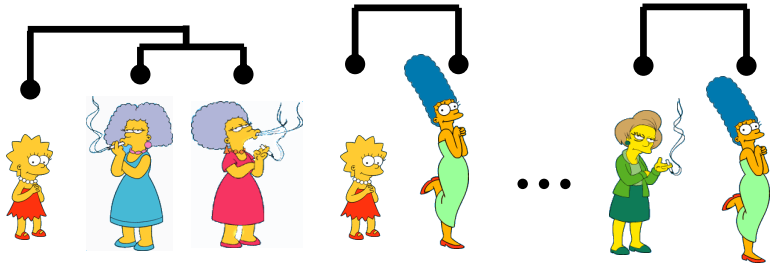
Consider all possible merges…



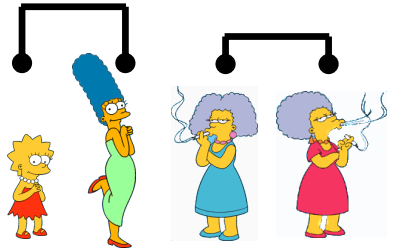
Choose the best



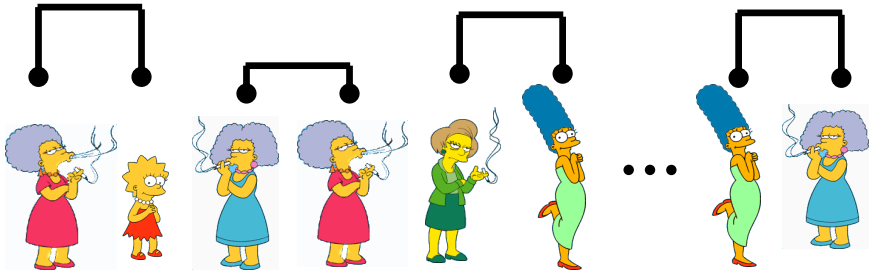
Consider all possible merges…



Choose the best



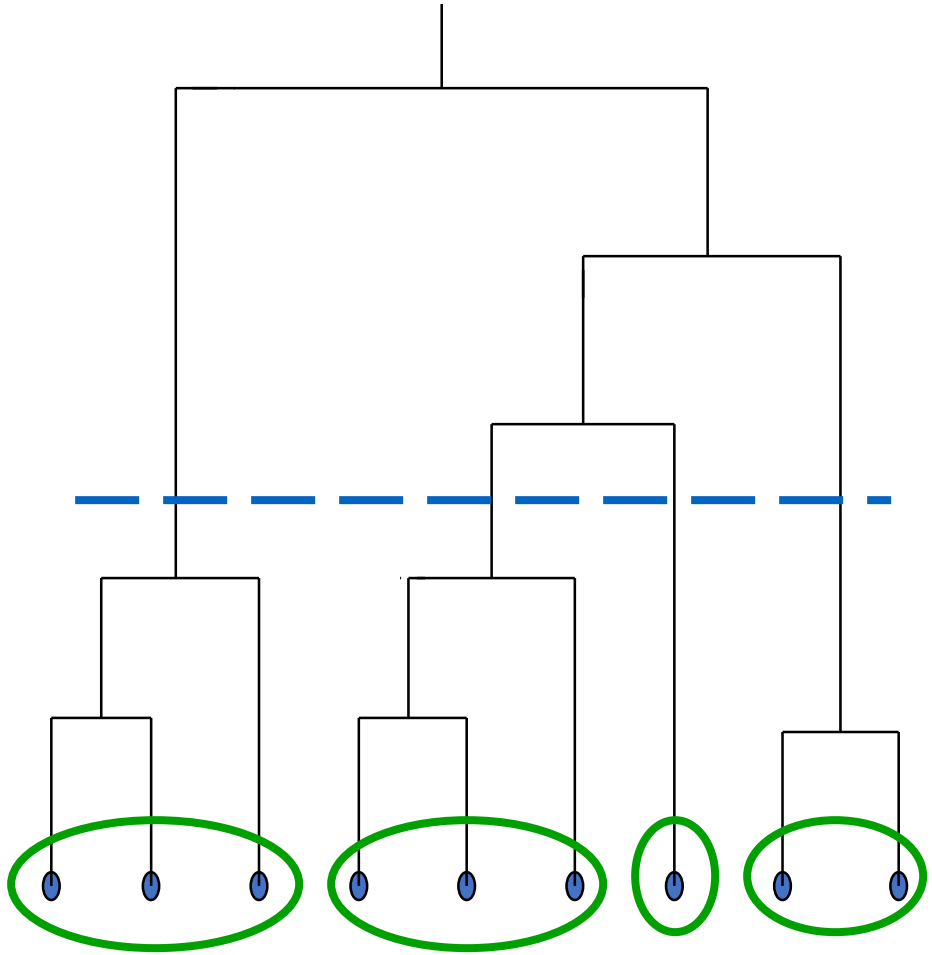
Consider all possible merges…



Choose the best

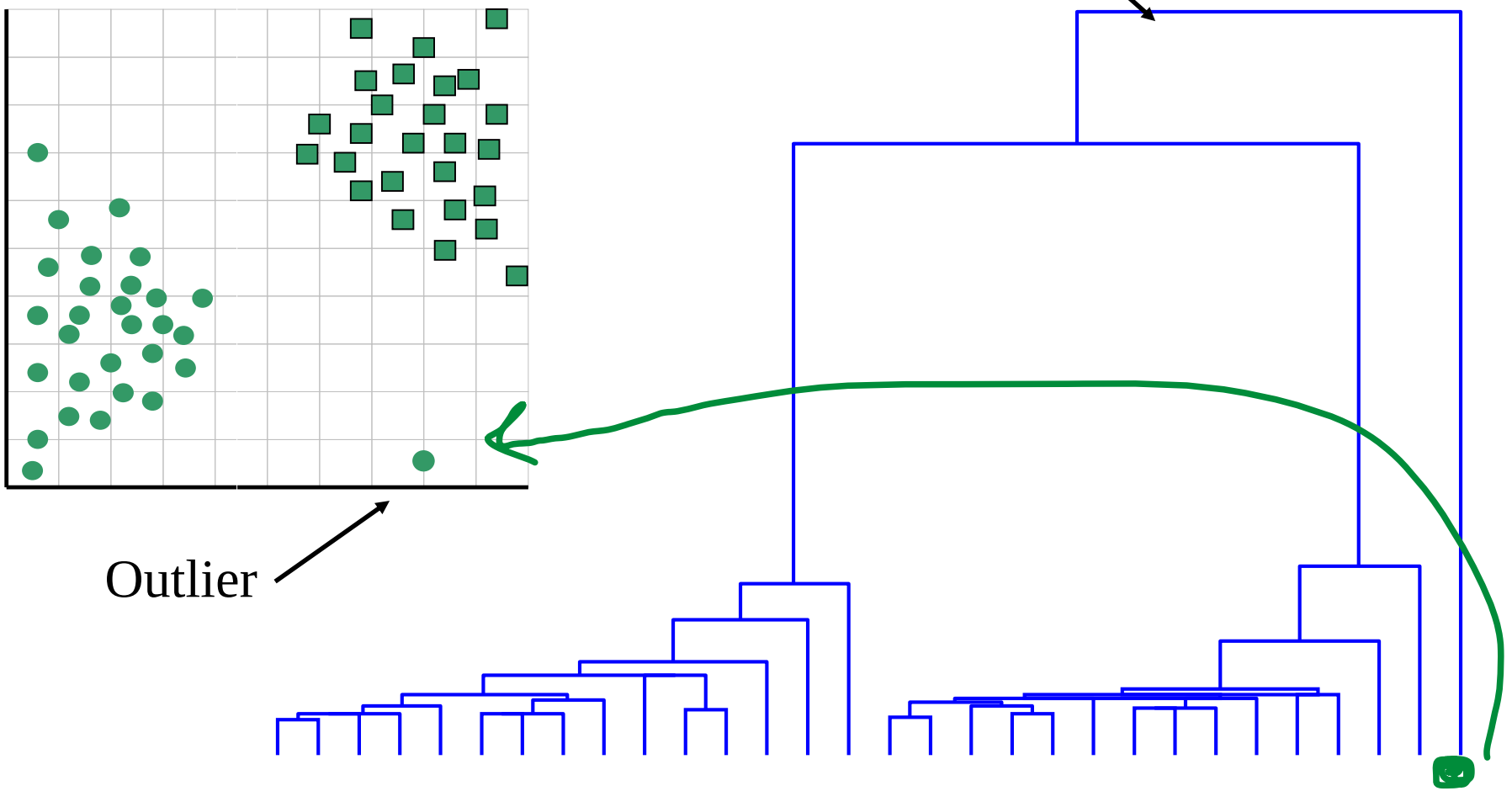
Dendogram: Hierarchical Clustering

- Clustering is obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.



One potential advantage of a dendrogram is to detect outliers

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



HAC: How to select the "closest pair" of **clusters**?

Many variants to defining **closest pair of clusters**

1. **Single-link**

   Distance of the *"closest" points* (single-link)

2. **Complete-link**

   Distance of the "furthest" points

3. **Centroid**

   Distance of the centroids (centers of gravity)

4. **Average-link**

   Average distance between pairs of elements

1. Single Link Agglomerative Clustering

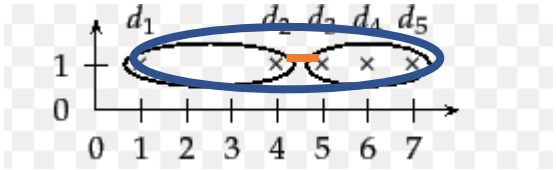
- At each iteration, use maximum similarity (=**minimum distance**) of pairs:

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

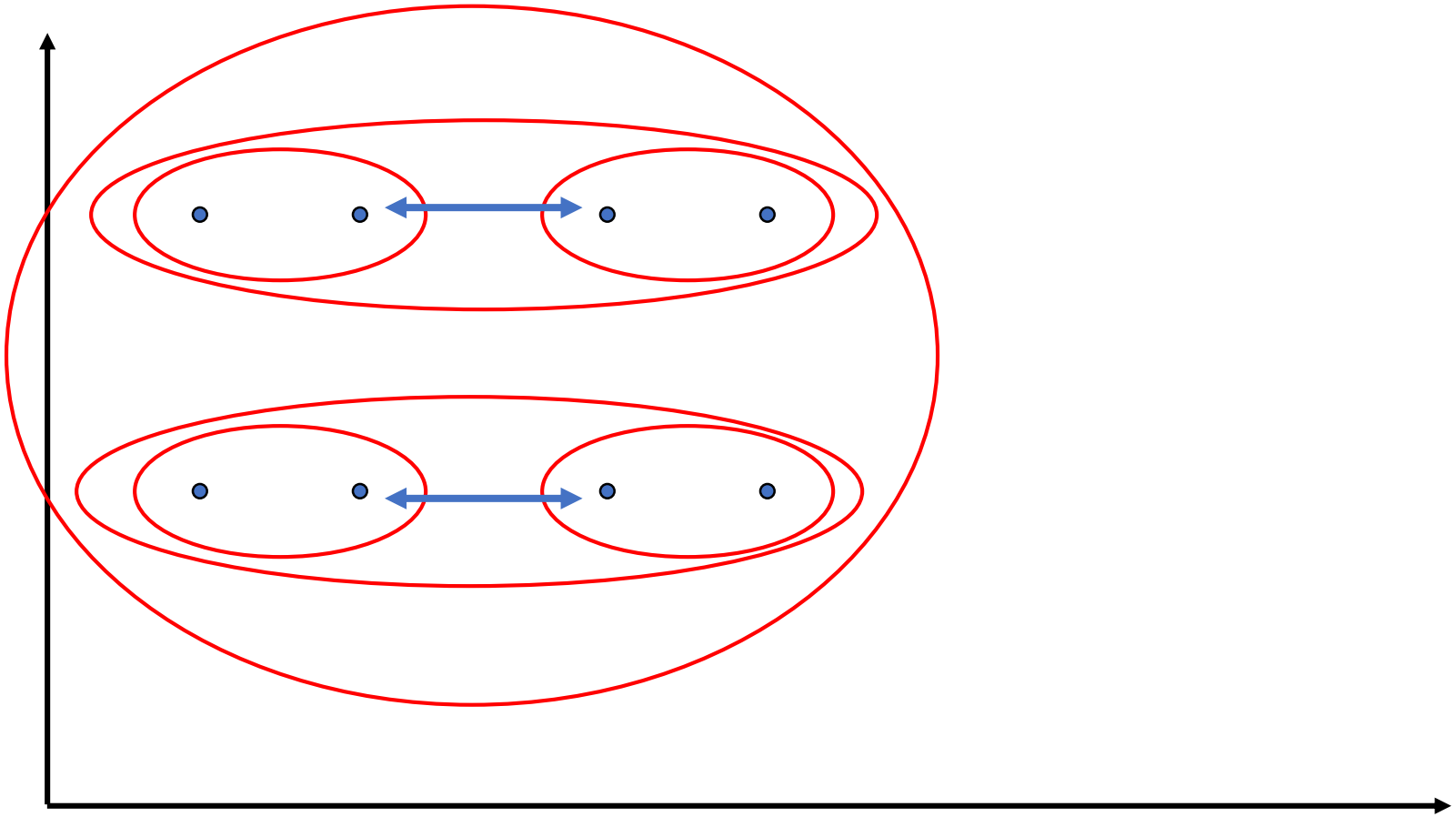
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

  - Can result in “straggly” (long and thin) clusters due to chaining effect.



Single Link Example



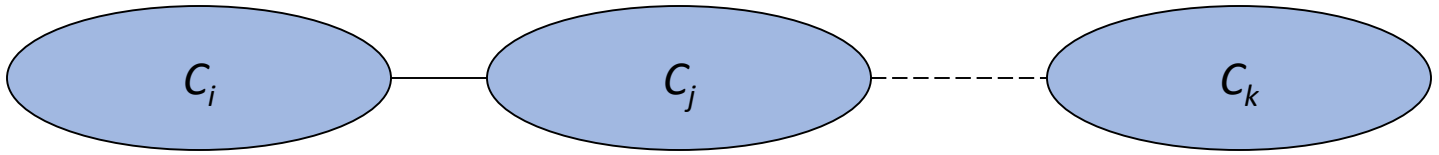
2. Complete Linkage Agglomerative Clustering

- Clusters are agglomerated always based on minimum distance, but the notion of distance between two clusters is the inverse
- The distance between two clusters is computed as the **maximum distance** (=minimum similarity) of pairs belonging to the two clusters:

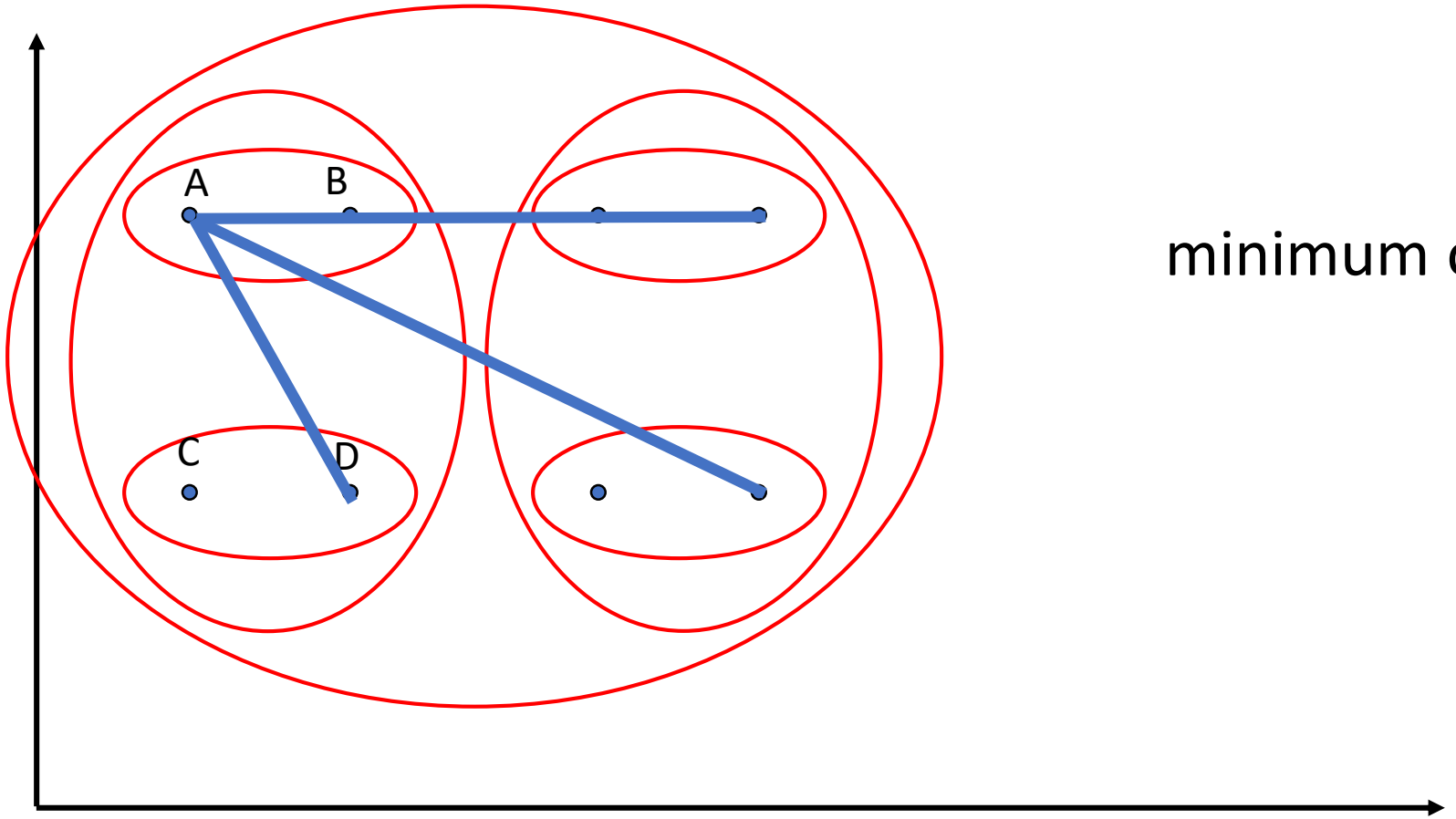
$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- Makes "tighter," spherical clusters that are typically preferable.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$



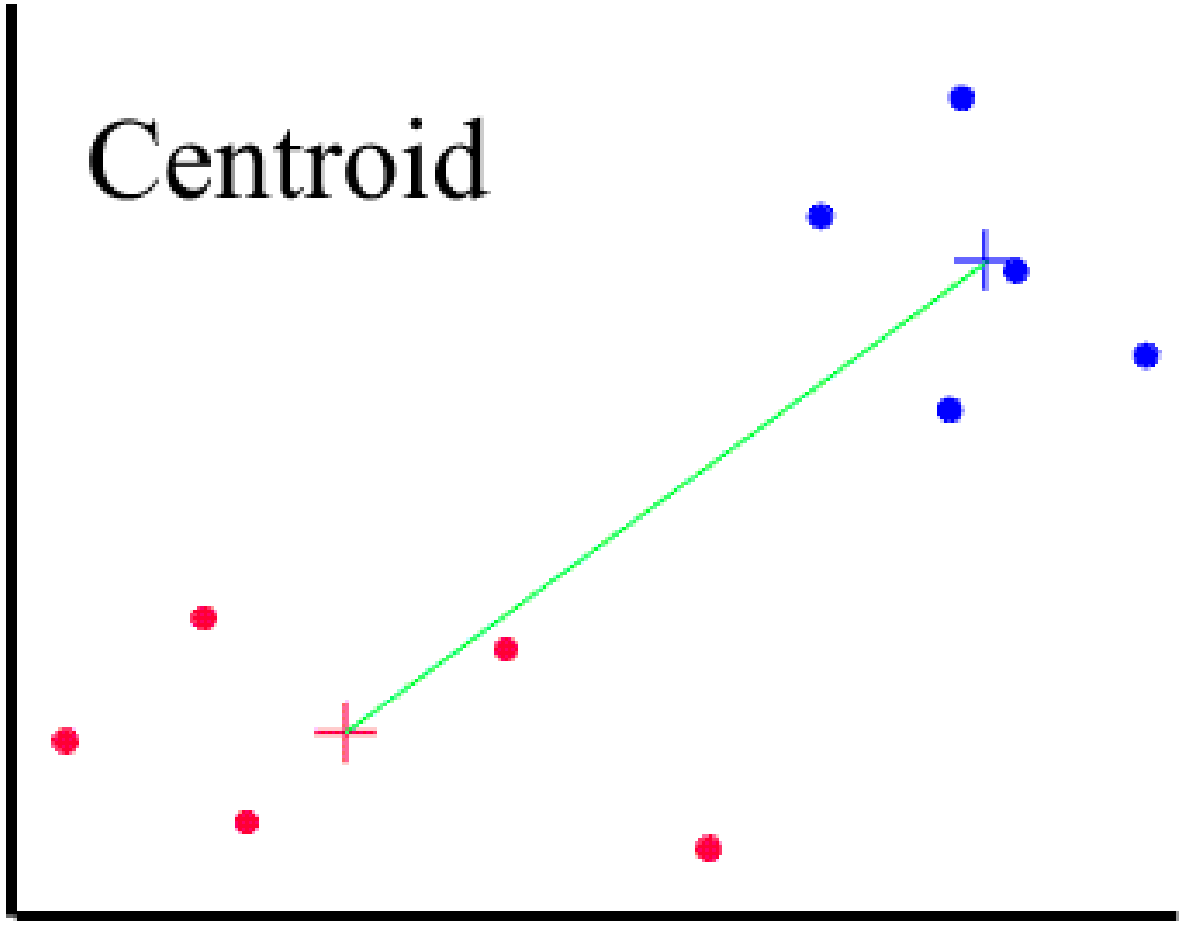
Complete linkage example



minimum of max distances!

3. Centroid-based clustering

- We want a notion of a “representative” point in a cluster
- Representative should be some sort of “typical” or central point in the cluster, e.g.,
  - **A point that is the “average” of all elements in the cluster (not necessarily an “existing” point**
    - Centroid or center of gravity



3. Centroid-based Similarity (2)

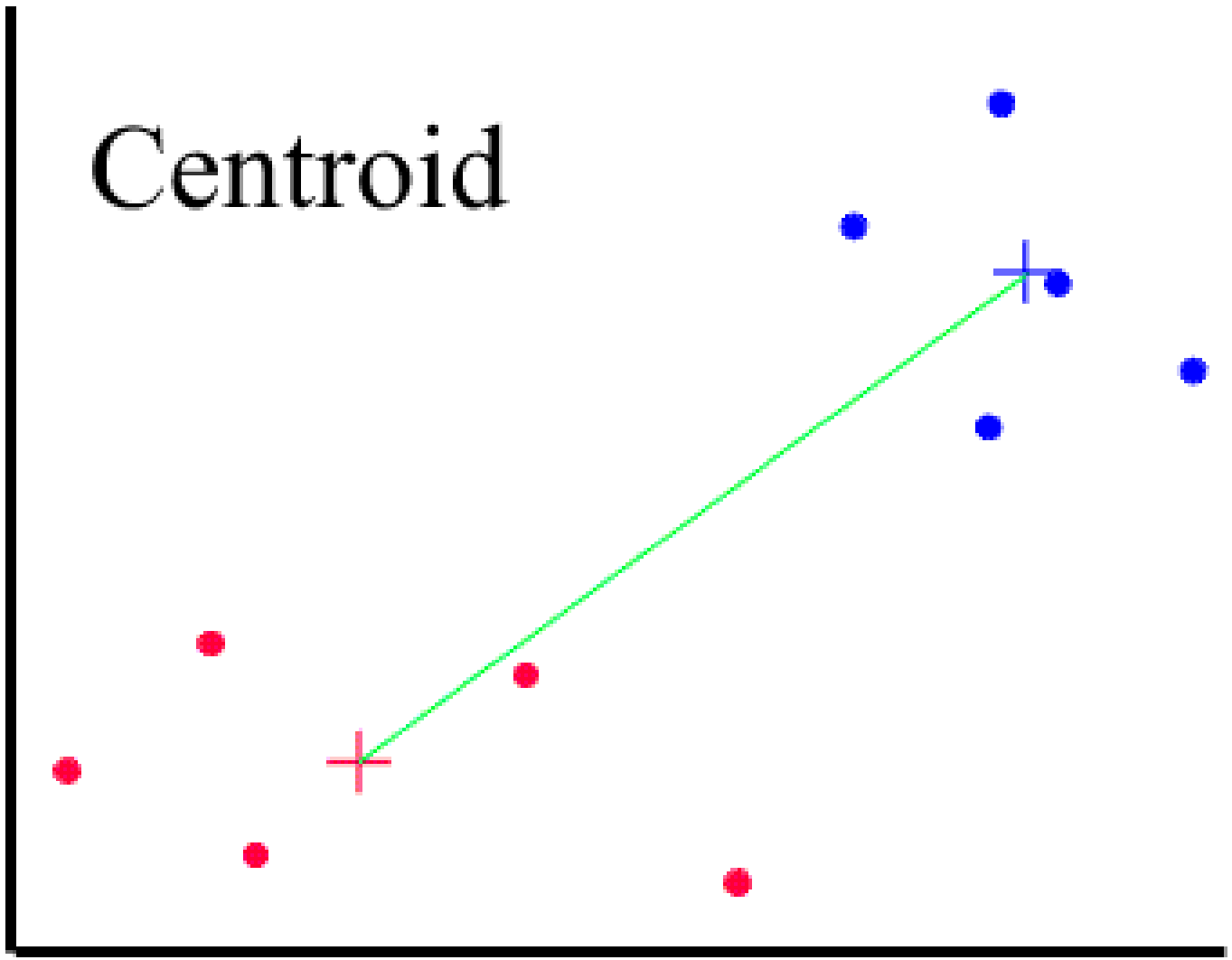
- Compute average of vectors in each cluster:

$$\vec{s}(c_j) = \frac{\sum_{\vec{x} \in c_j} \vec{x}}{|c_j|}$$

- Compute similarity of clusters by:

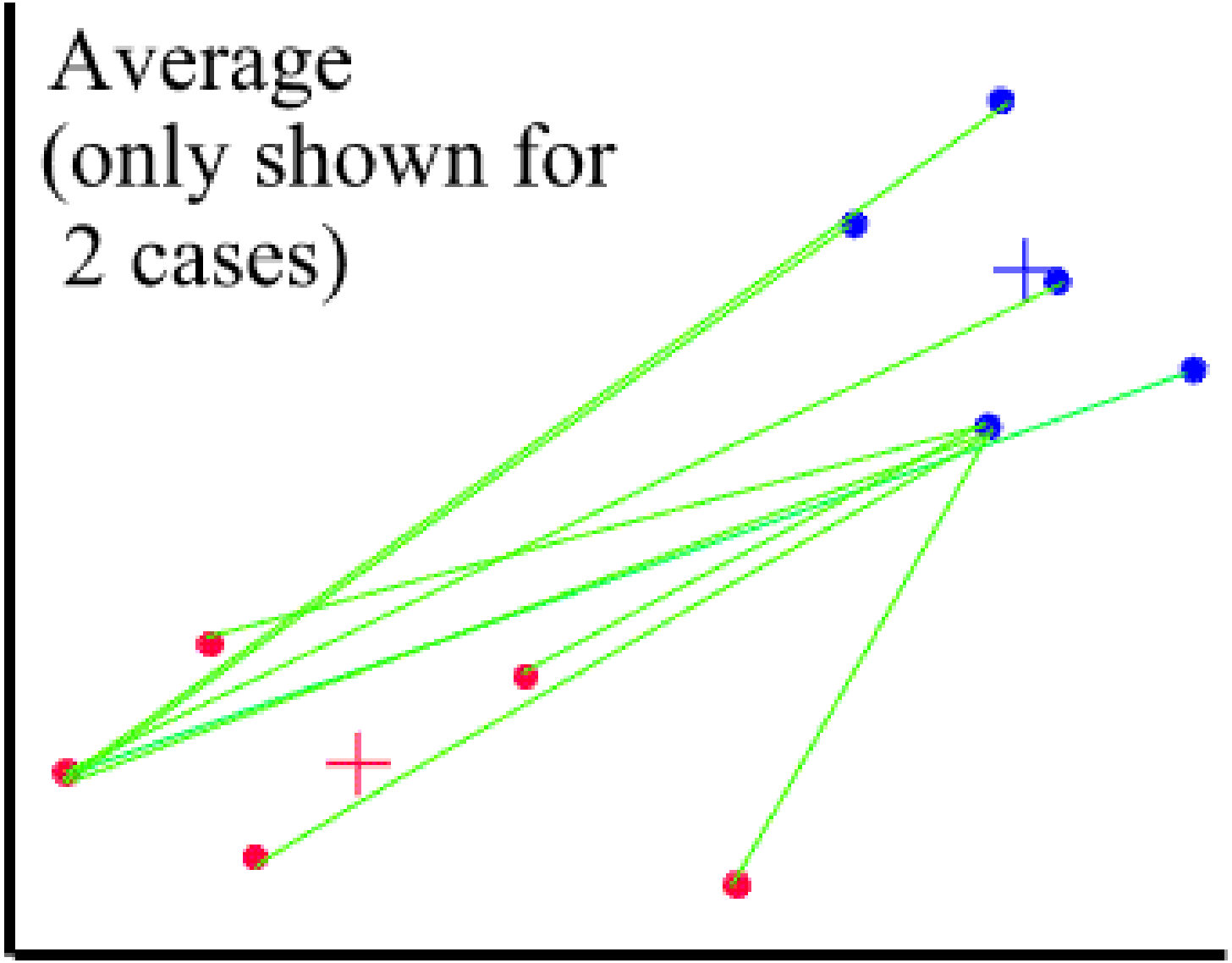
$$sim(c_i, c_j) = sim(s(c_i), s(c_j))$$

- For non-vector representation of instances, **can't** always compute a centroid (e.g, matrixes)!


Centroid

4. Average similarity

- Based on computing pairwise
- intra-cluster distances and then
- averaging (average of all
- similarities)

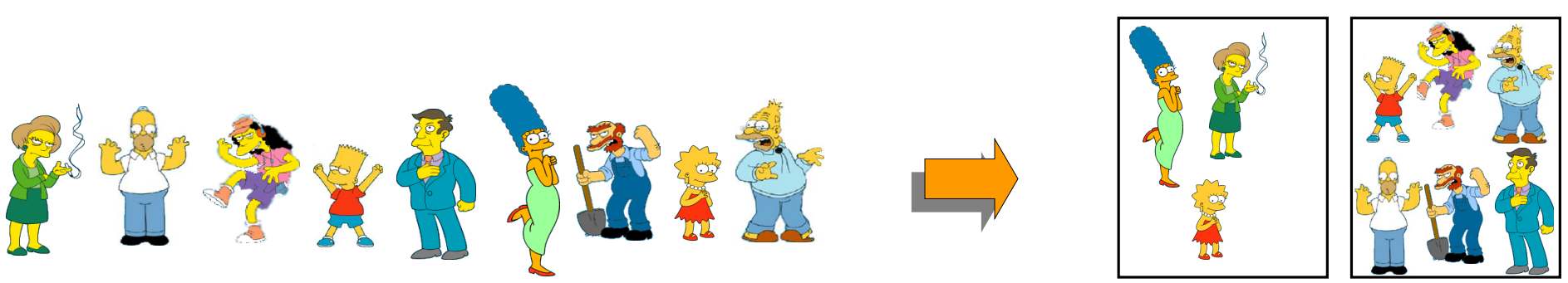


Summary of Hierarchal Clustering Methods

- No need to specify the number of clusters in advance.
- Hierarchal nature maps nicely onto human intuition for some domains
- 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.

Partitional Clustering

- Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.
- Since only one set of clusters is output, the user normally has to input the desired number of clusters K (methods are available to “guess” K).



- **Algorithm** *K-means*

1. Decide on a value for *K*.

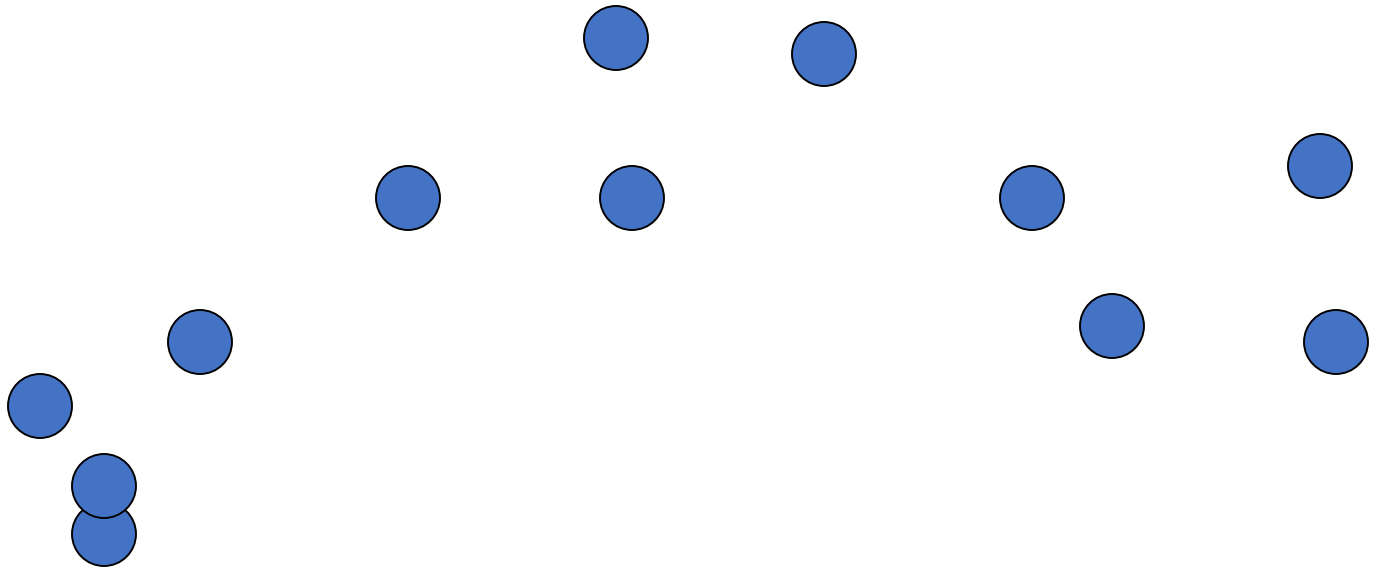
2. Initialize the *K* cluster centers by selecting K objects in D (randomly, if necessary).

3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.

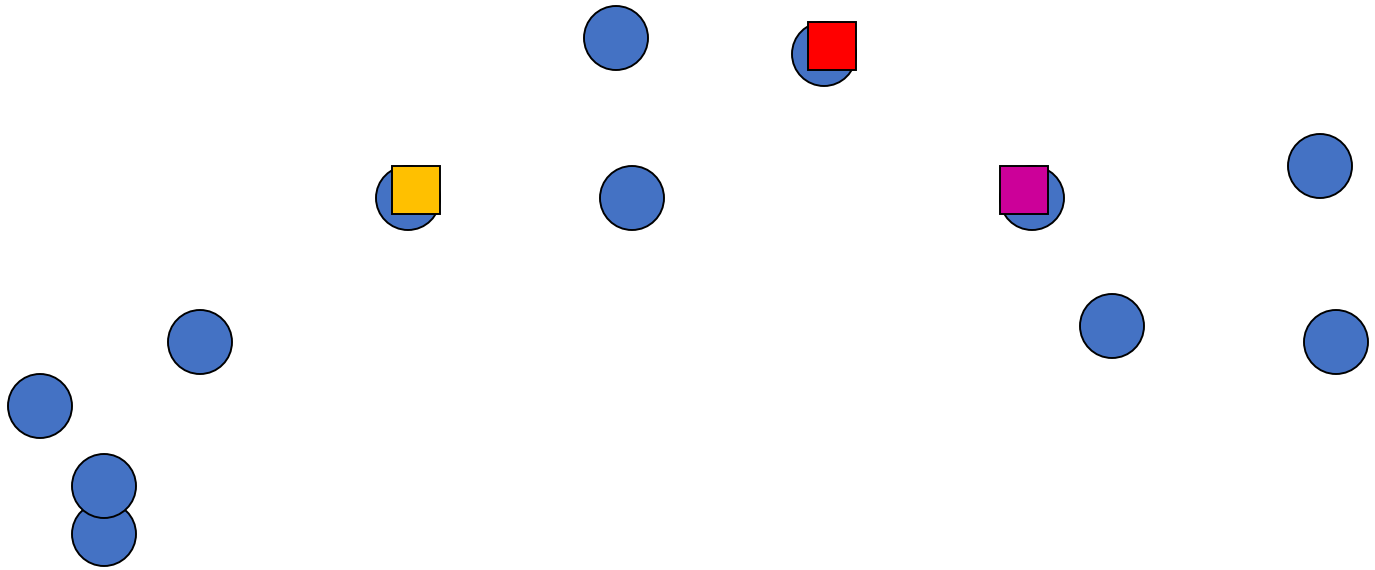
4. Re-estimate *K* cluster centeroids, based on current assignments of objects .

5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise go to 3.

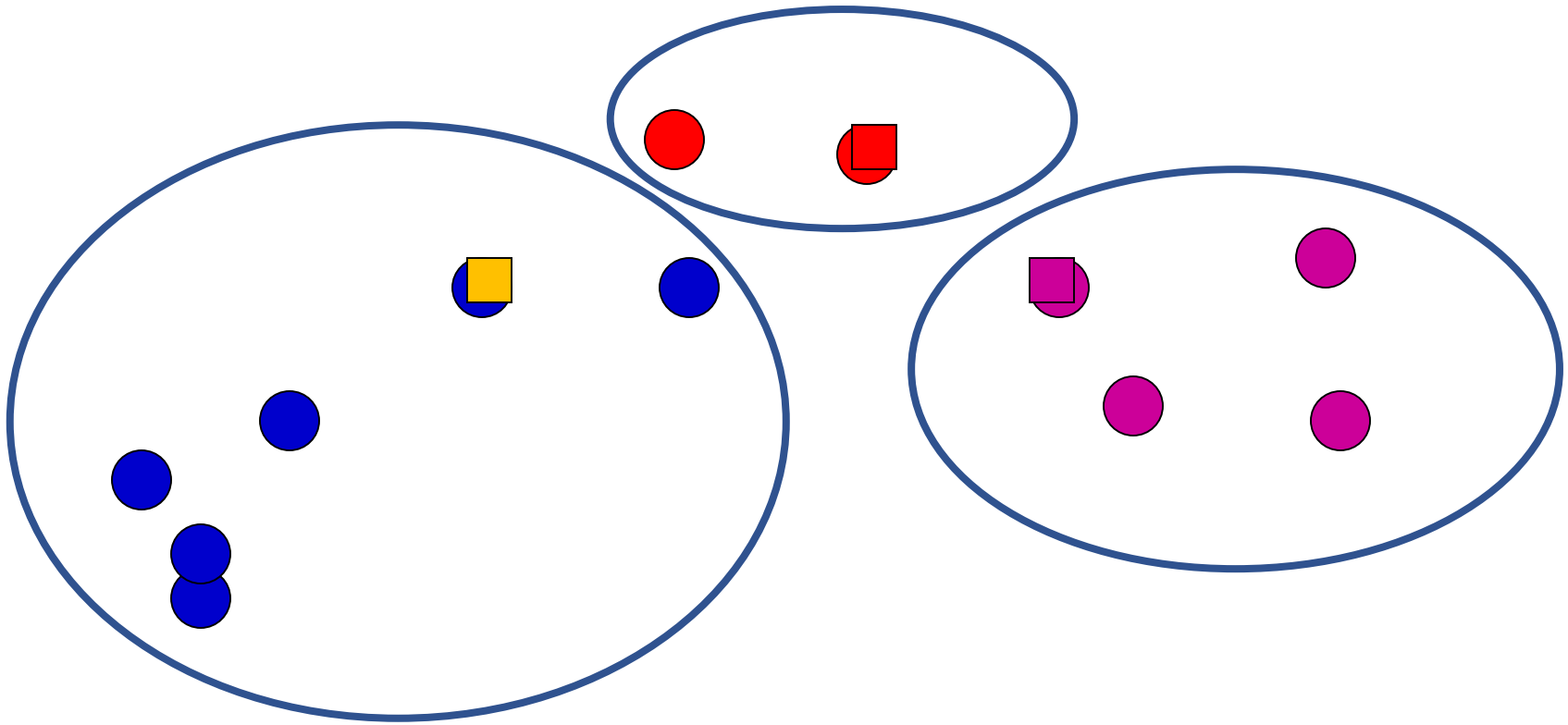
K-means: an example



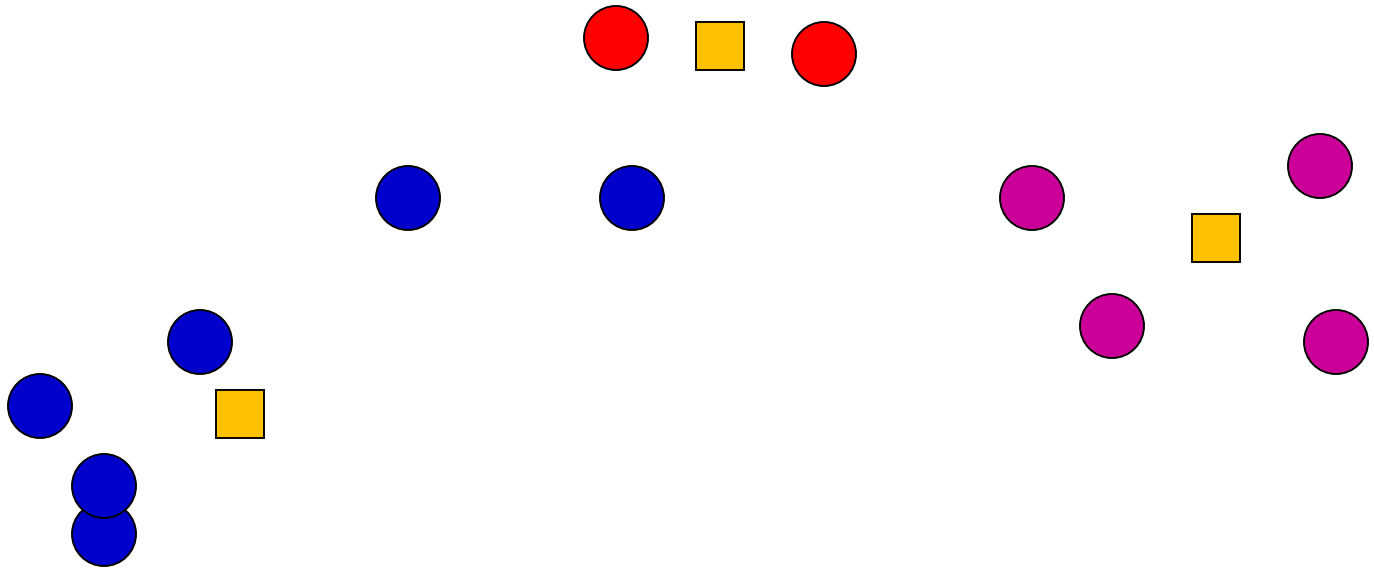
K-means: Initialize centers randomly



K-means: assign points to nearest center

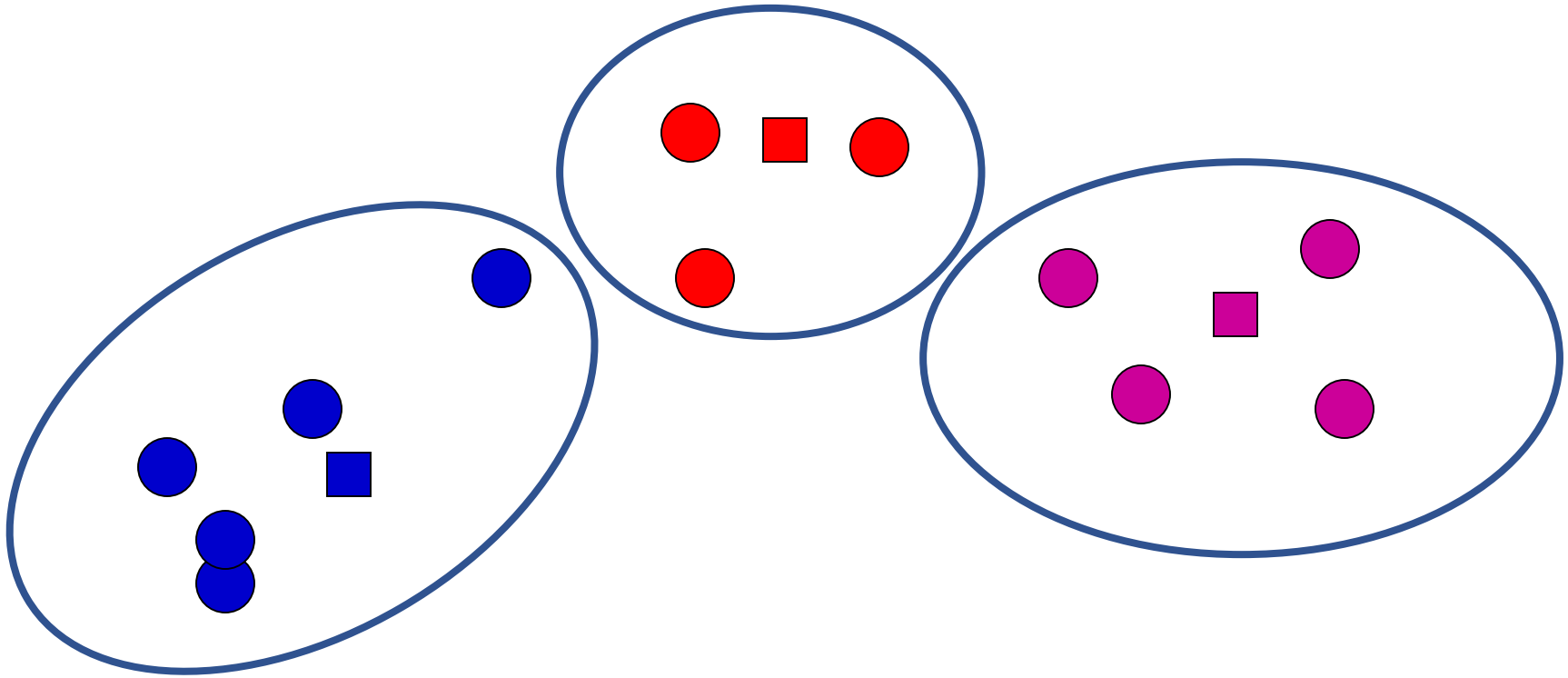


K-means: readjust centers

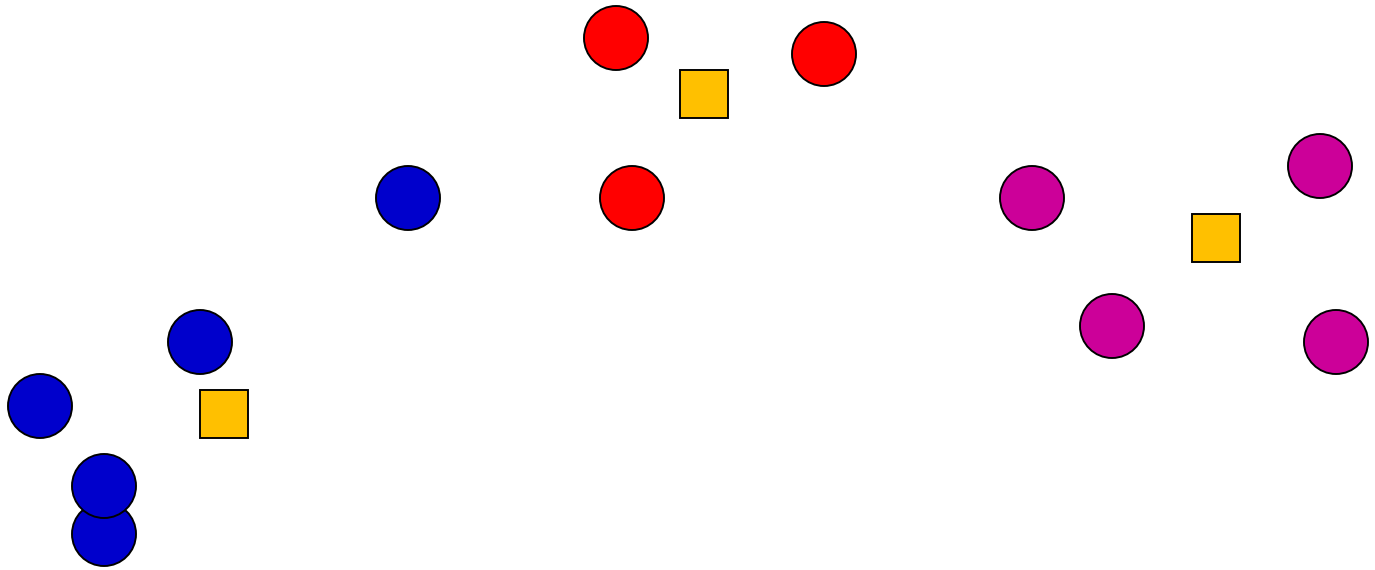


Note: new centers are centroids! Not necessarily real objects

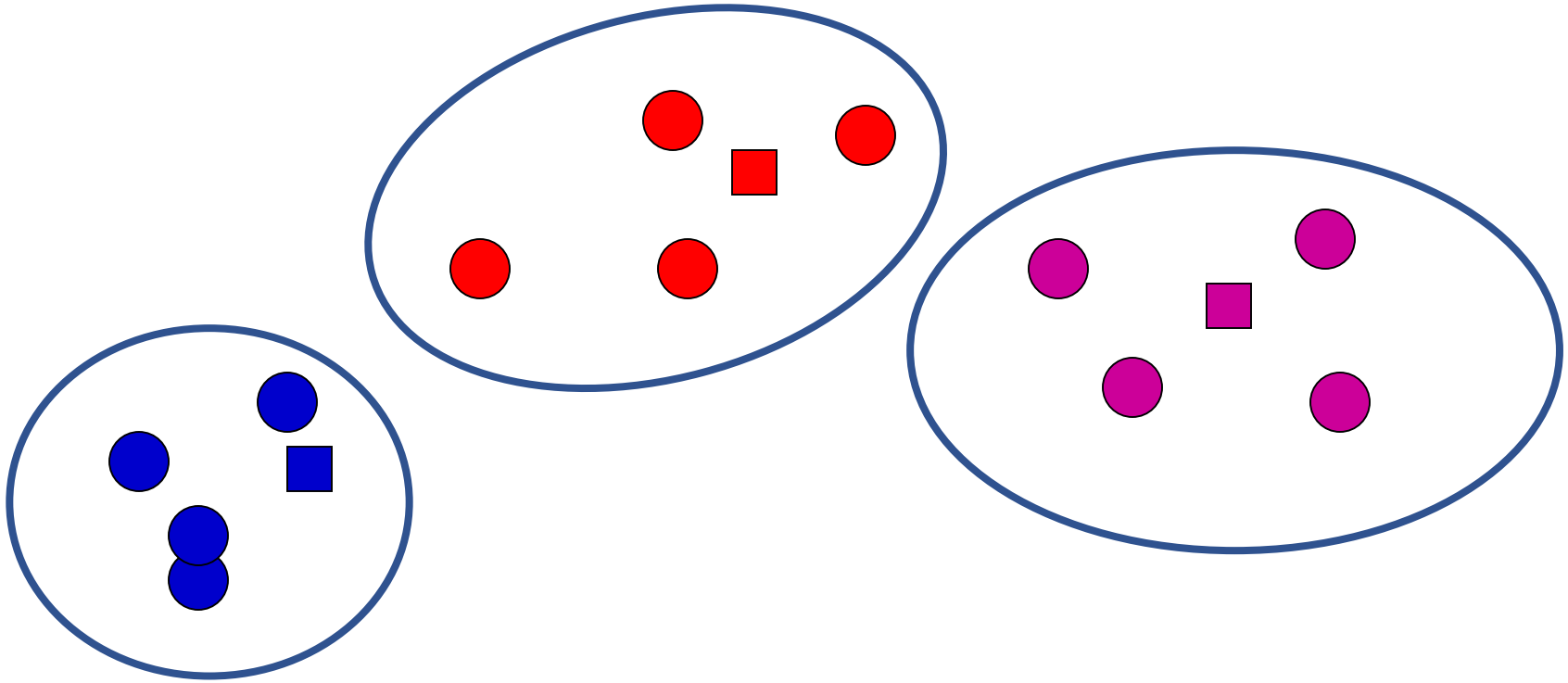
K-means: assign points to nearest center



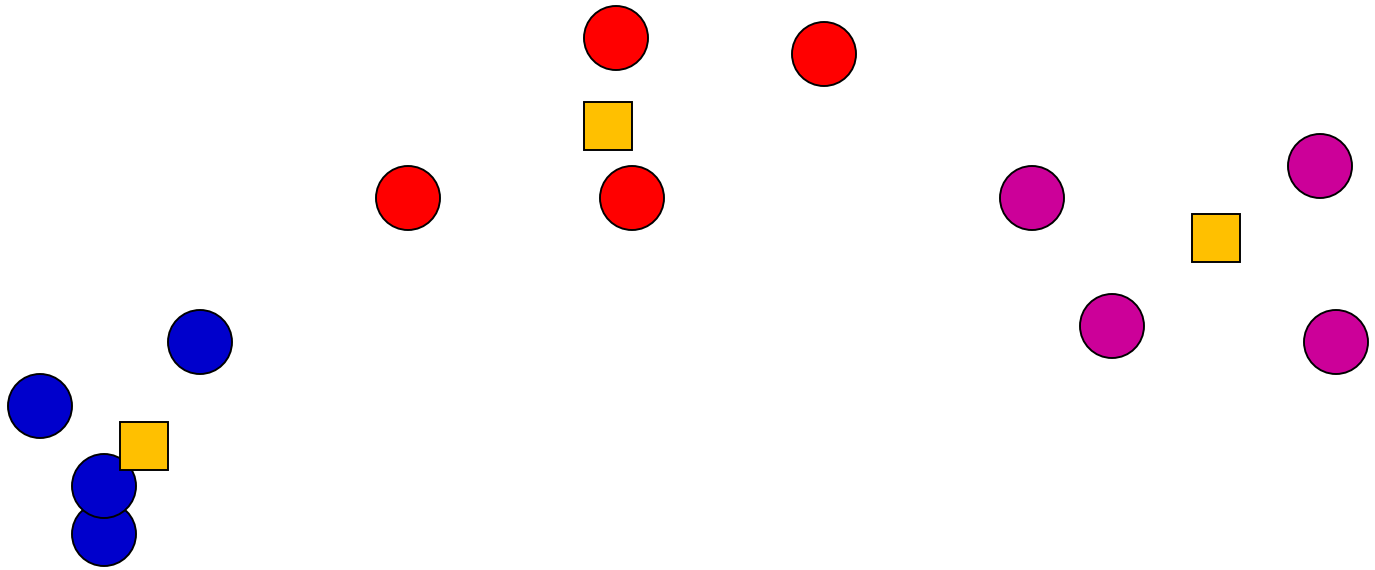
K-means: readjust centers



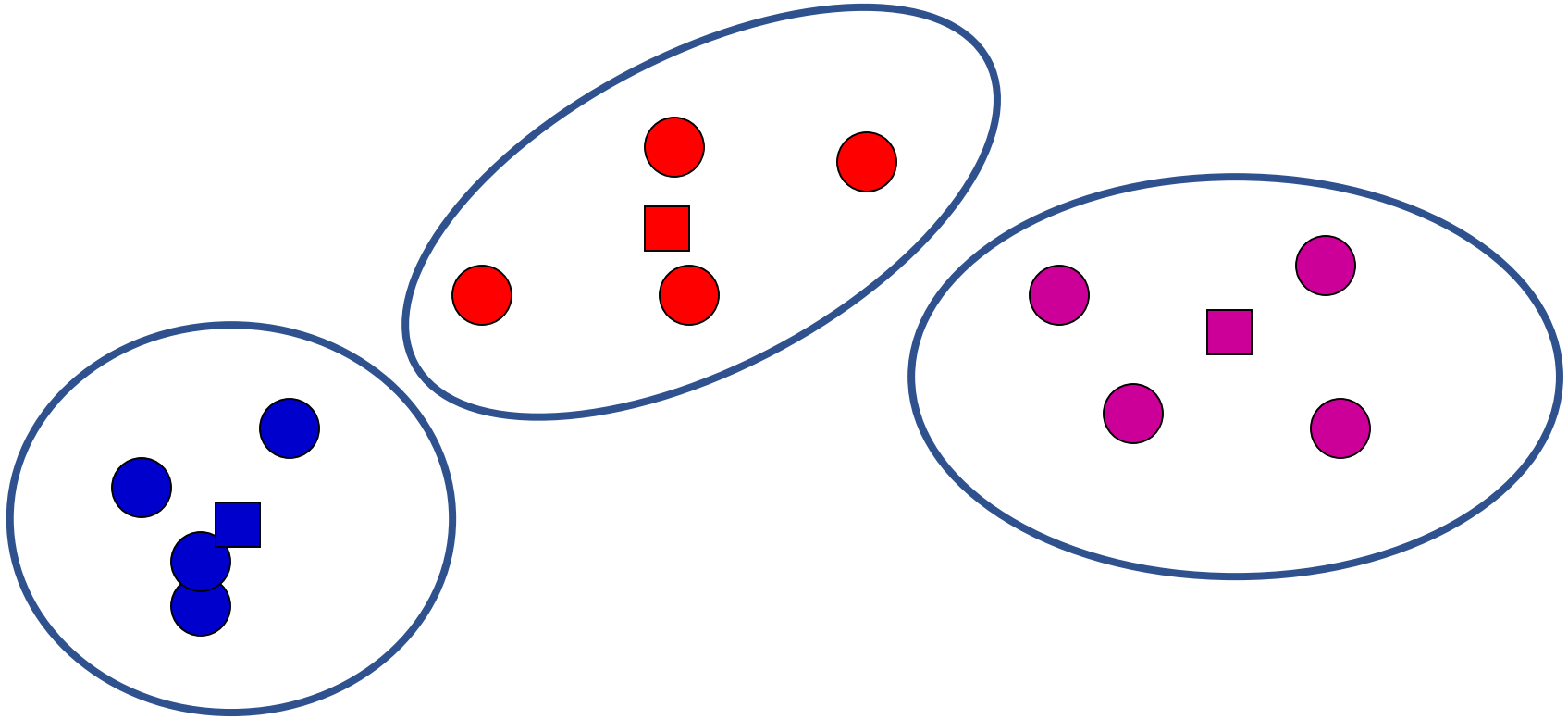
K-means: assign points to nearest center



K-means: readjust centers



K-means: assign points to nearest center

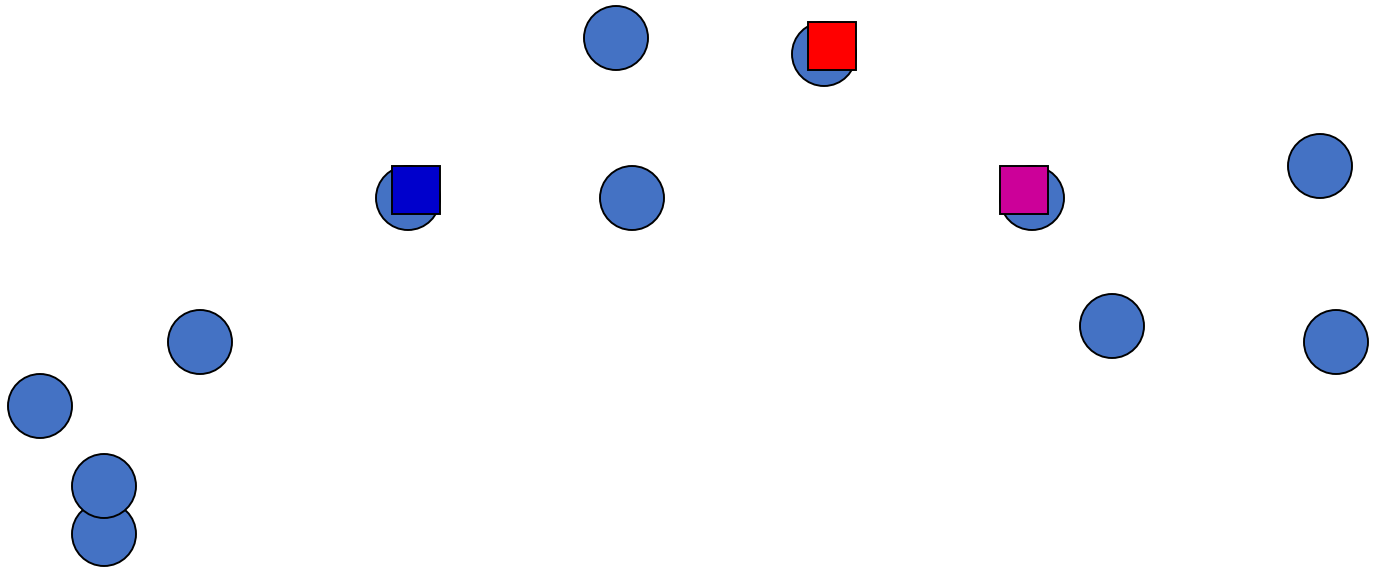


No changes: Done

K-means

Iterate:

- **Assign/cluster each example to closest center**
- Recalculate centers as the mean of the points in a cluster



How do we do this?

K-means

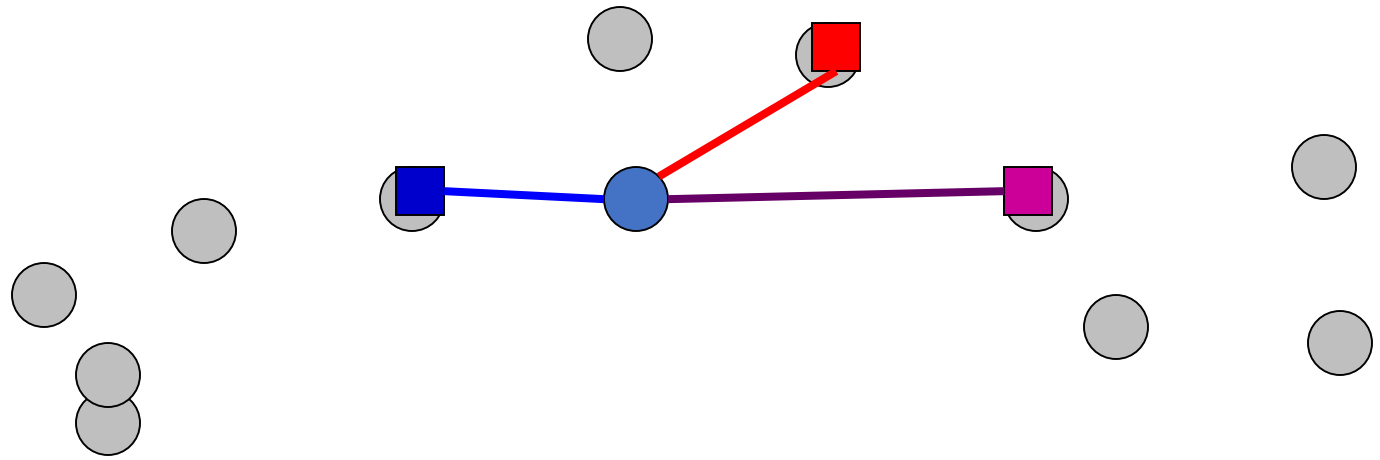
Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:

  - get distance to each cluster center
  - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



K-means

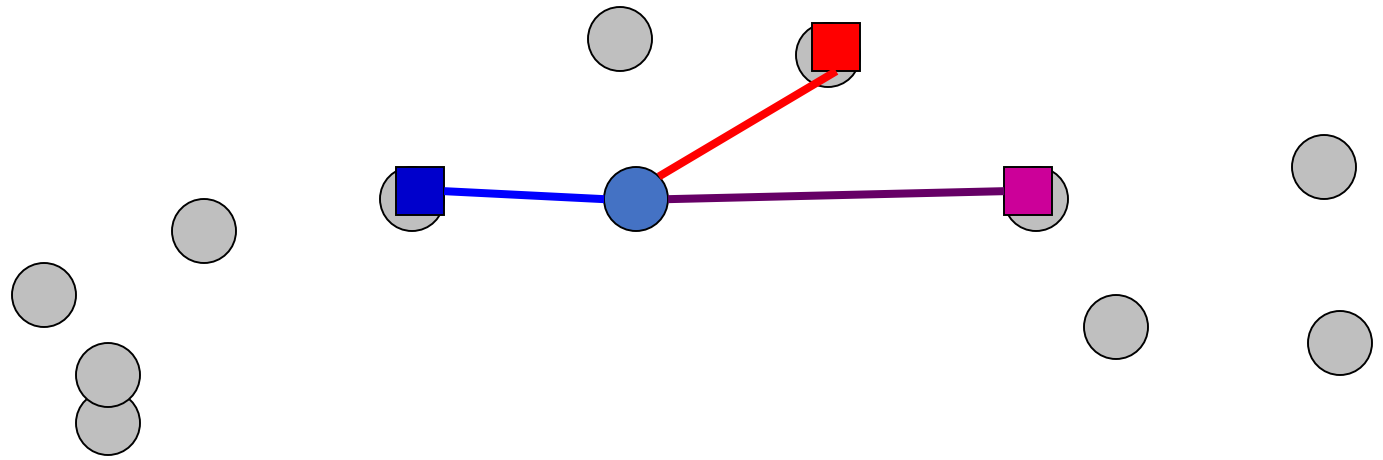
Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:

  - get **distance** to each cluster center
  - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



Comments on the *K-Means* Method

Strength

- *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, $t << n$.
- Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Can be used in ensambles

Weakness

- Applicable only when a *mean* is defined, then what about categorical data?
- Need to specify *k,* the *number* of clusters, in advance (true for most clustering algorithms)
- Unable to handle noisy data and *outliers (all points are assigned to a cluster)*

Additional readings

- Here a collection of most cited surveys link