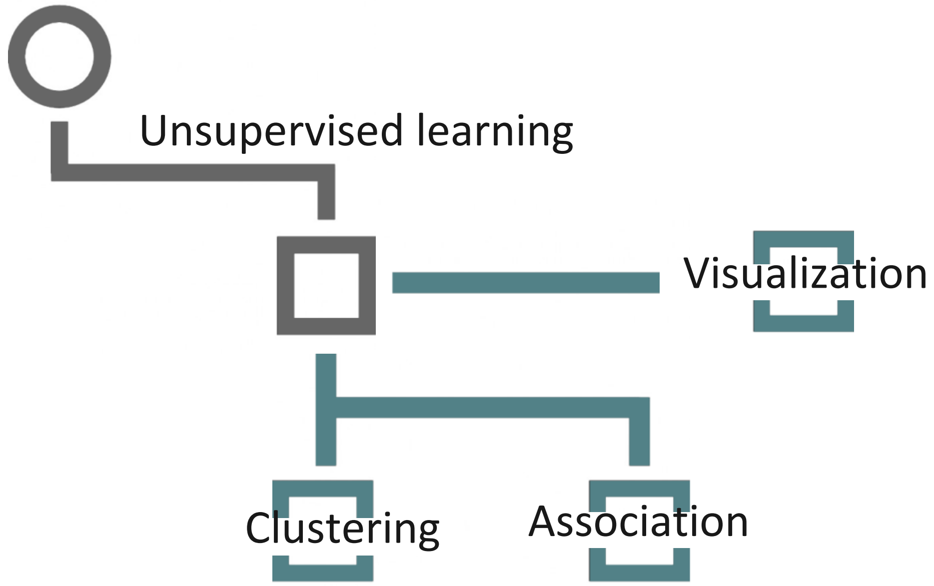
POLIMI GRADUATE SCHOOL OF MANAGEMENT

CLUSTERING LAB

ANDREA MOR

UNSUPERVISED LEARNING



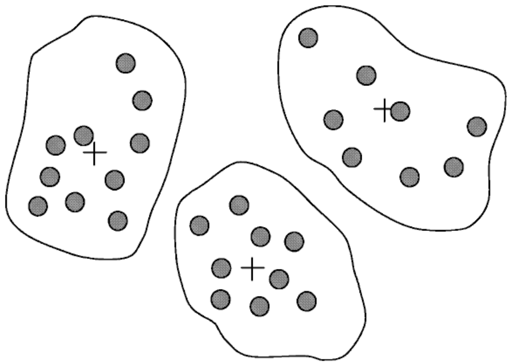
Unsupervised learning:

Given a set D of data points $D=\{X_i\}$ ($i \in M$), discover hidden patterns in the data set to get useful insights.

CLUSTERING

Divide a dataset of examples into homogenous groups.

Example: Divide the customers of a company based on their purchasing behaviour.



ASSOCIATION RULES

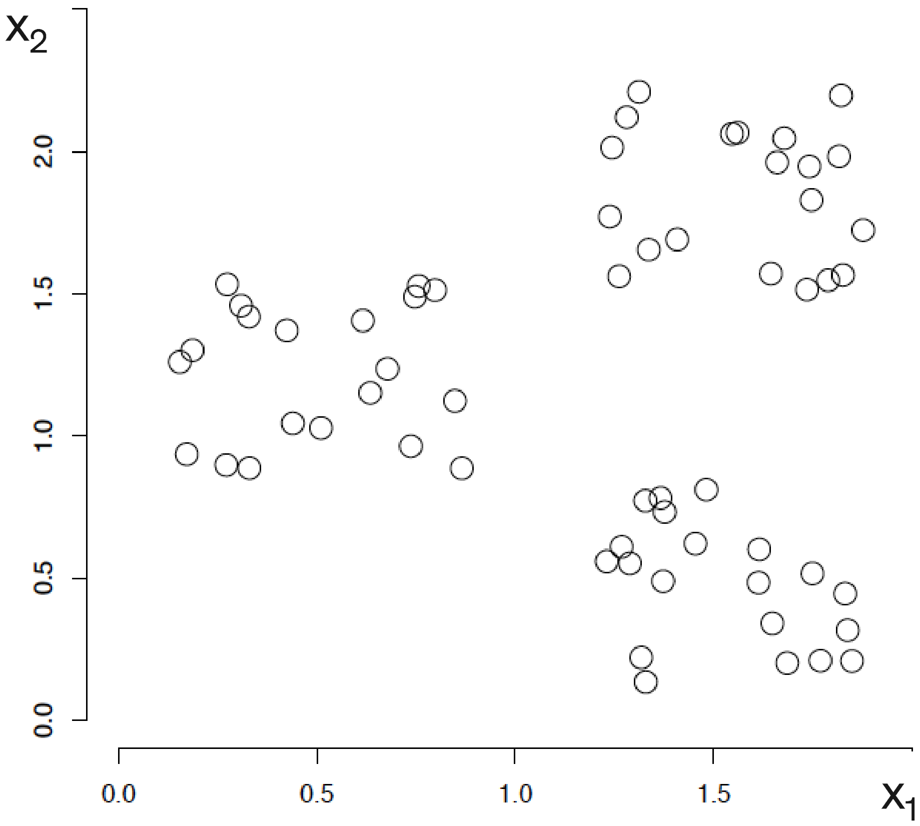
Identify recurrences among single or group of events within a dataset of transactions.

Example: Identify associations between items in shopping baskets.

Product A $\Rightarrow$ Product B and C

Pr 0.85

CLUSTERING METHODS



How would you design an algorithm for grouping these examples?

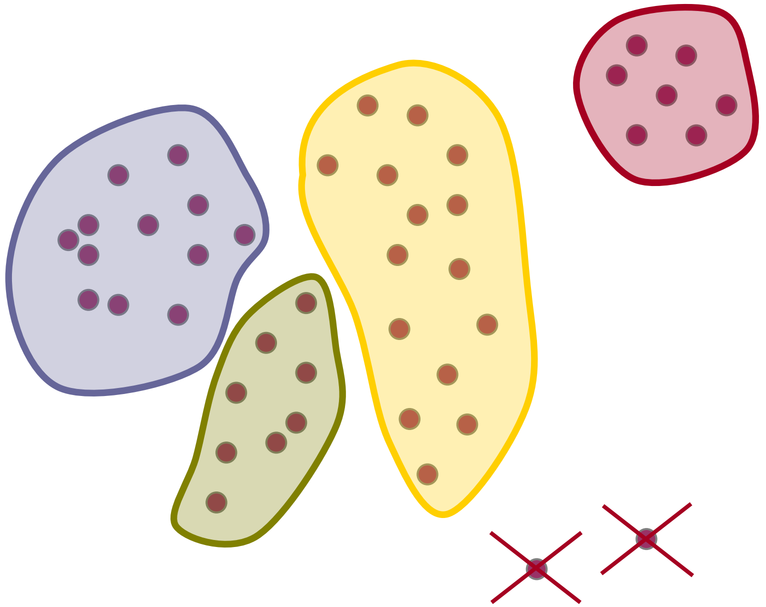
CLUSTERING METHODS

Objective $\Rightarrow$ homogeneous groups of examples (CLUSTERS):

- Examples within a cluster should be similar
- Examples from different clusters should be dissimilar

Can be used for:

- Getting interpretable segments
- Preliminary grouping of examples
- Detecting outliers
- Selecting landmarks



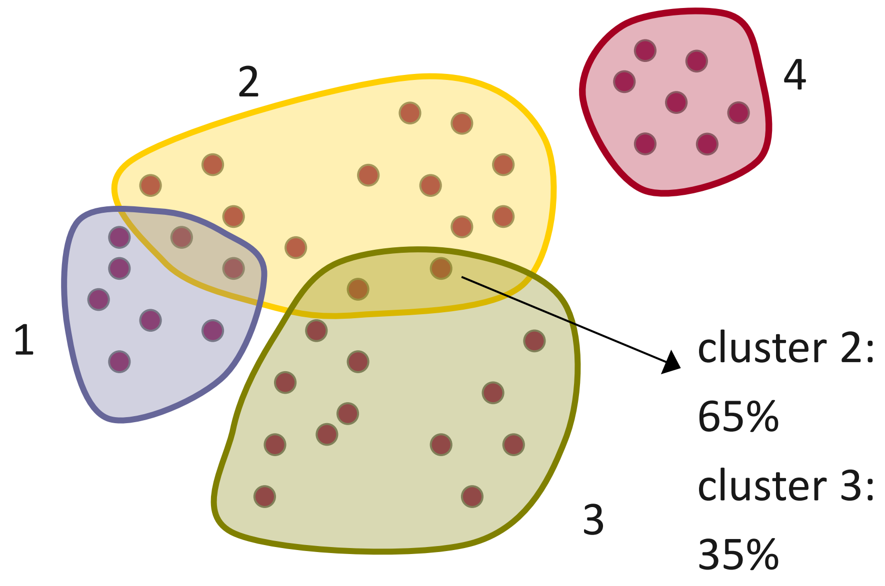
Based on the logic used for assigning the points we can have:

EXCLUSIVE METHODS

FUZZY METHODS

COMPLETE METHODS

PARTIAL METHODS



CLUSTERING APPLICATIONS

Several application domains…

- Marketing
  Partition consumers into market segments (understand the relationships between different groups of consumers)

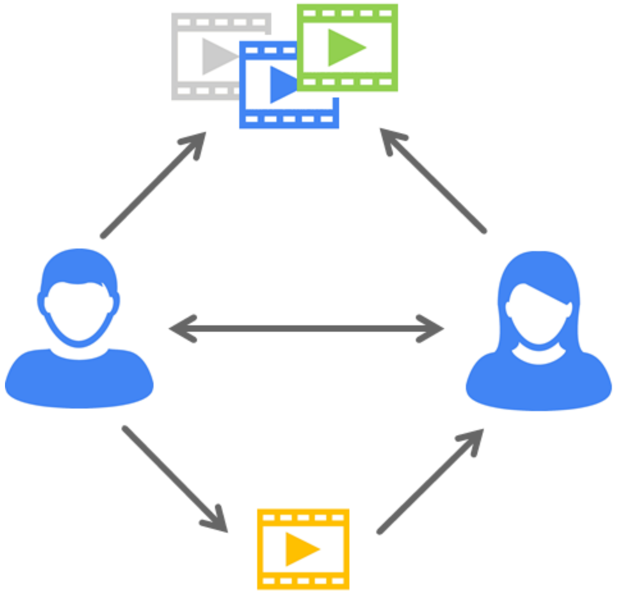
- Social network analysis
  Detect communities (hubs) within large groups of people (disrupt dark networks)

- Genomics
  Build groups of genes with co-related expression patterns

- Social science
  Identify areas (hot spots) where there are greater incidences of similar types of crime (manage law enforcement resources)

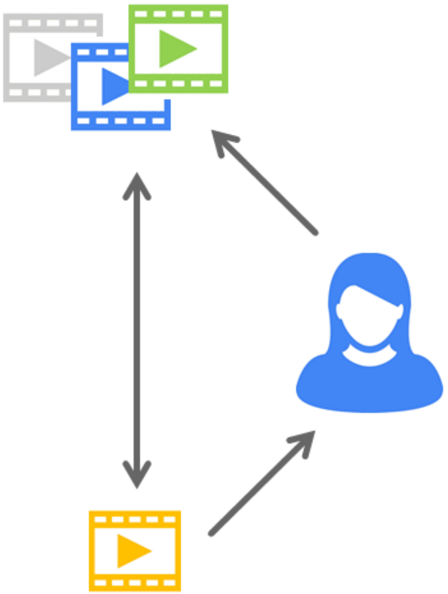
CLUSTERING AND RECOMMENDER SYSTEMS

Collaborative filtering



Computes the similarity among users.
If A has the same opinion of B on a given set of contents, B is more likely to have B's opinion on a different content.

Item-based filtering



Computes the similarity among items.
The item recommended to the user is similar to the ones the user found interesting in the past.

CLUSTERING METHODS

Based on the logic used for building clusters we can have:

PARTITION METHODS

The data set is divided into a pre-fixed number of clusters

HIERARCHICAL METHODS

Perform several partitions based on a tree structure

General requirements:

- FLEXIBILITY $\Rightarrow$ numeric and categorical attributes
- ROBUSTNESS $\Rightarrow$ stability of the clusters (noise)
- EFFICIENCY $\Rightarrow$ small computing time

A huge number of possible partitions...
... clustering is *NP*- hard for $K \geq 3$

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n$$

# of possible combinations

$\Downarrow$

Most methods are heuristic in nature!

HOW MANY DIFFERENT CLUSTERING MODELS?

Suppose we have m=5 objects (A, B, C, D, E) and we want to cluster them into k=2 groups...

1 -> (A) , (B,C,D,E)
2 -> (B) , (A,C,D,E)
3 -> (C) , (A,B,D,E)
4 -> (D) , (A,B,C,E)
5 -> (E) , (A,B,C,D)

6 -> (A,B) , (C,D,E)
7 -> (A,C) , (B,D,E)
8 -> (A,D) , (B,C,E)
9 -> (A,E) , (B,C,D)
10 -> (B,C) , (A,D,E)

11 -> (B,D) , (A,C,E)
12 -> (B,E) , (A,C,D)
13 -> (C,D) , (A,B,E)
14 -> (C,E) , (A,B,D)
15 -> (D,E) , (A,B,C)

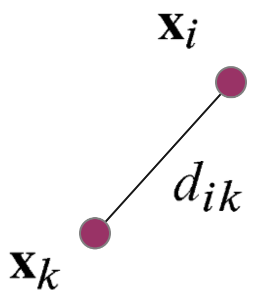
Suppose we have m=20 objects and we want to cluster them into k=3 groups...

580,606,446
possible clustering models!

AFFINITY MEASURES

DISTANCE MATRIX

$$D = [d_{ik}] = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1,m-1} & d_{1m} \\ & 0 & \cdots & d_{2,m-1} & d_{2m} \\ & & \cdots & \vdots & \vdots \\ & & & 0 & d_{m-1,m} \\ & & & & 0 \end{bmatrix}$$



$$d_{ik} = \mathrm{dist}(\mathbf{x}_i, \mathbf{x}_k) = \mathrm{dist}(\mathbf{x}_k, \mathbf{x}_i), \quad i, k \in \mathcal{M}$$

SIMILARITY MEASURE

$$s_{ik} = \frac{1}{1 + d_{ik}} \qquad s_{ik} = \frac{d_{max} - d_{ik}}{d_{max}}$$

AFFINITY MEASURES: NUMERIC ATTRIBUTES

EUCLIDEAN DISTANCE:

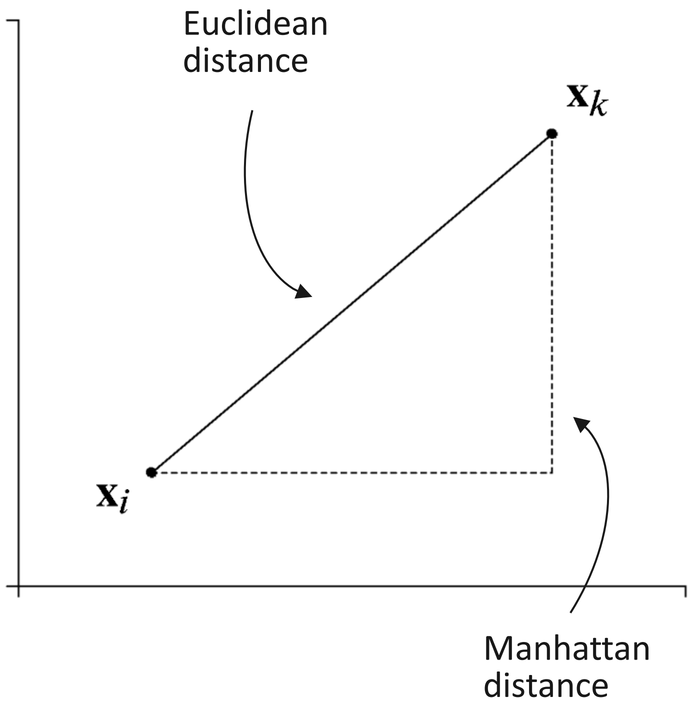
$$\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_k) = \sqrt{\sum_{j=1}^{n} (x_{ij} - x_{kj})^2}$$

MANHATTAN DISTANCE:

$$\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_k) = \sum_{j=1}^{n} |x_{ij} - x_{kj}|$$

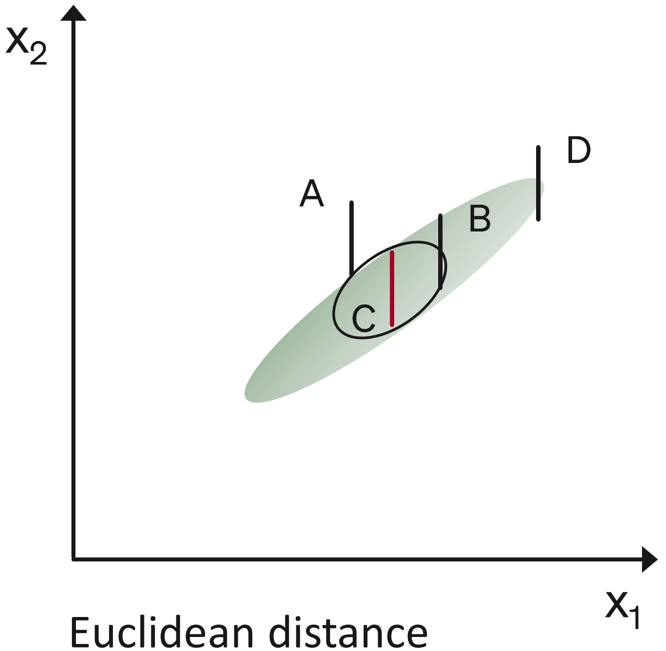
MINKOWSKI DISTANCE:

$$\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_k) = \sqrt[q]{\sum_{j=1}^{n} |x_{ij} - x_{kj}|^q}$$



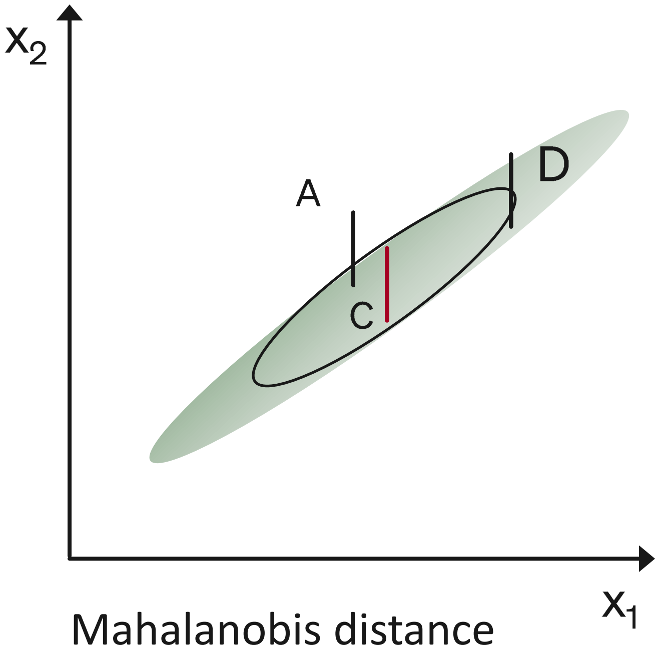
AFFINITY MEASURES: NUMERIC ATTRIBUTES

MAHALANOBIS DISTANCE $\text{dist}(\mathbf{x}_i, \mathbf{x}_k) = \sqrt{(\mathbf{x}_i - \mathbf{x}_k)\mathbf{V}^{-1}(\mathbf{x}_i - \mathbf{x}_k)'}$



Euclidean distance

→ The set of points equidistant from a given location is a sphere



Mahalanobis distance

→ Stretches the sphere to account for correlation among variables

AFFINITY MEASURES: BINARY ATTRIBUTES

Contingency table

		point $\mathbf{x}_k$		
		0	1	totale
point $\mathbf{x}_i$	0	p	q	$p+q$
	1	u	v	$u+v$
	totale	$p+u$	$q+v$	n

COEFFICIENT OF SIMILARITY
(symmetric attributes)

$$\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_k) = \frac{q+u}{p+q+u+v}$$

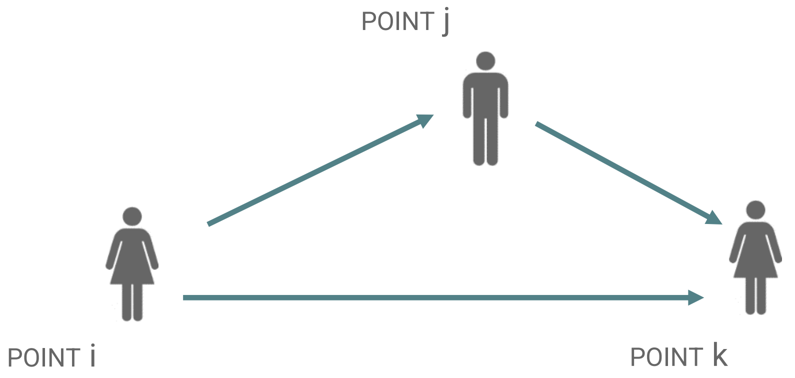
JACCARD DISTANCE
(asymmetric attributes)

$$\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_k) = \frac{q+u}{q+u+v}$$

AFFINITY MEASURES

Several measures can be used but, in any case, the following properties must be satisfied: (i.e., what makes a function a distance?)

1) Dist (POINT i, POINT i) = 0
2) Dist (POINT i, POINT k) > 0 if POINT i != POINT k
3) Dist (POINT i, POINT k) = Dist (POINT k, POINT i)
4) Dist (POINT i, POINT k) ≤
   Dist (POINT i, POINT j) + Dist (POINT j, POINT k)

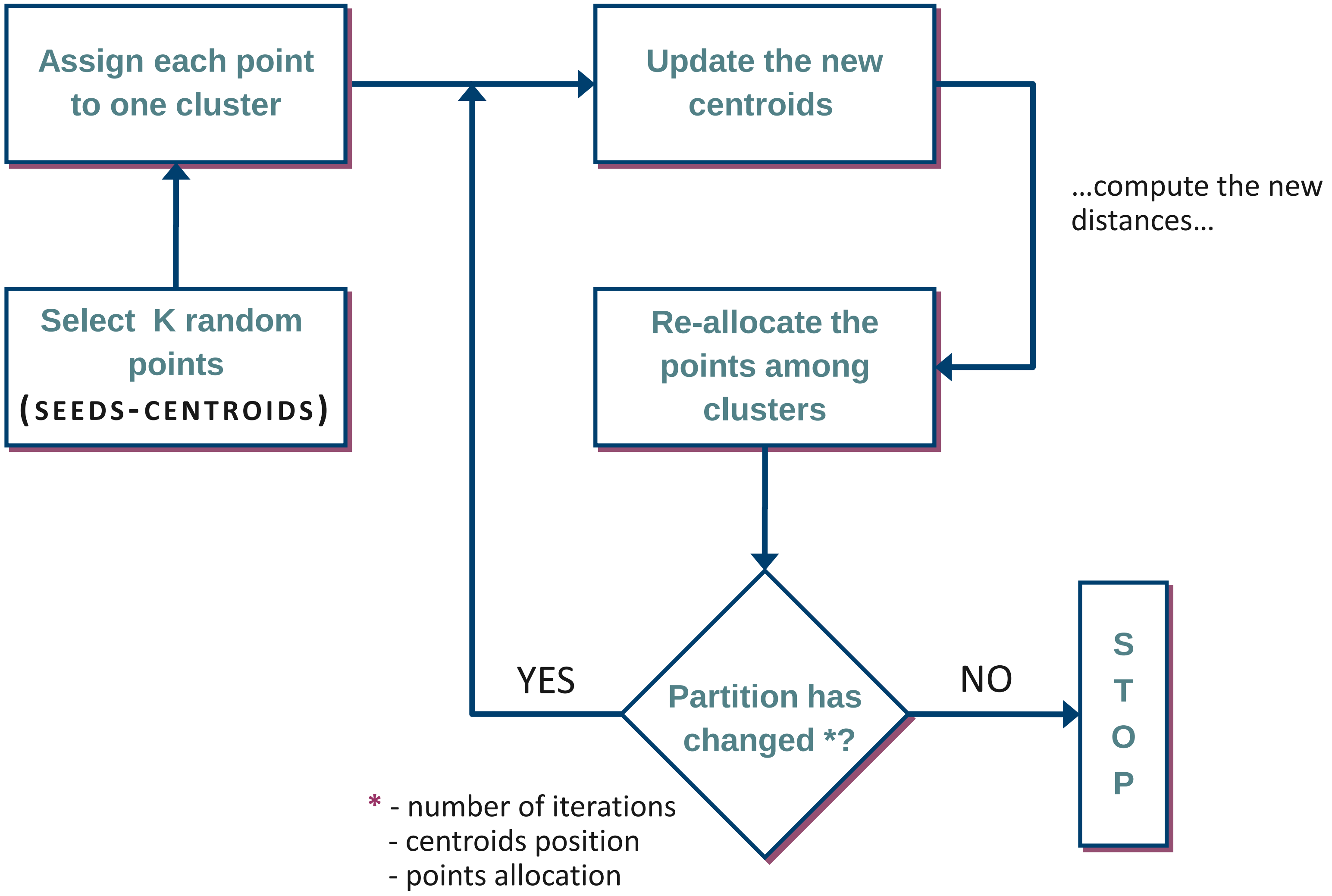


PARTITION METHODS: General framework

- Initialization $\Rightarrow$ Points are divided into K non-empty groups (usually exhaustive and mutually exclusive)
- Iteration $\Rightarrow$ Points are re-assigned with the aim of improving the quality of the partition
- Stop $\Rightarrow$ No points are further re-assigned (other stopping criteria)

Partition methods are greedy!

K-MEANS ALGORITHM

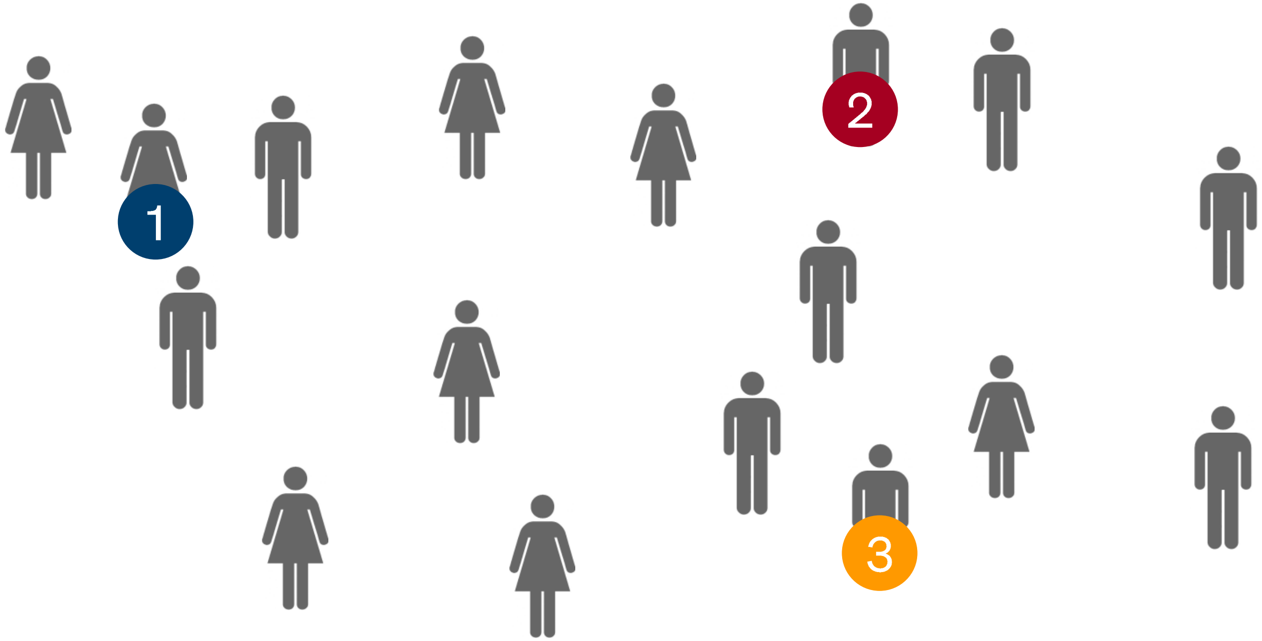


K-MEANS ALGORITHM

Let us suppose the following individuals are our employees:

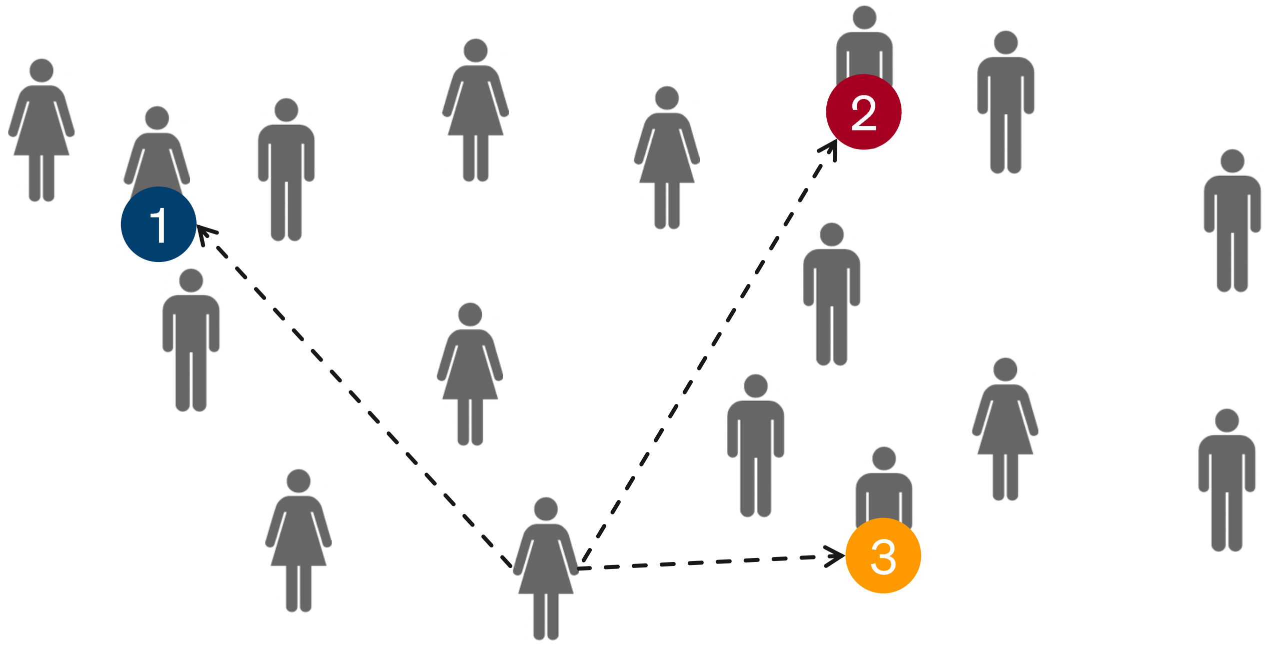
K-MEANS ALGORITHM

Let’s randomly locate 3 initial cluster centers (seeds):



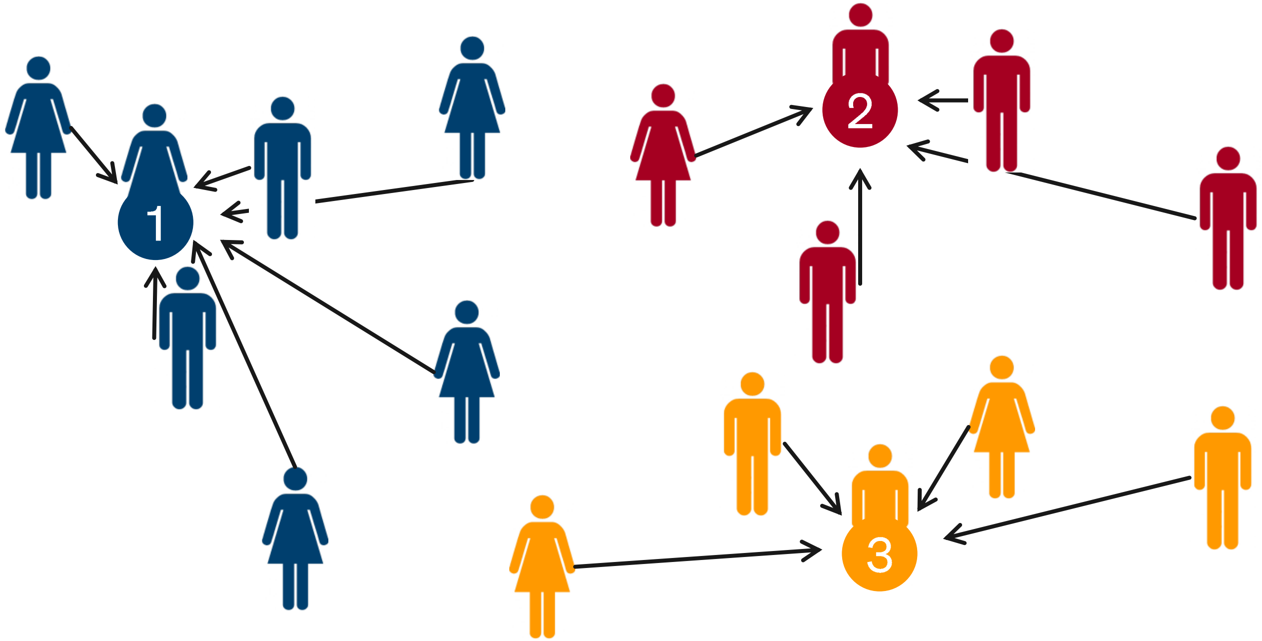
K-MEANS ALGORITHM

Find the distance of each individual from each center:



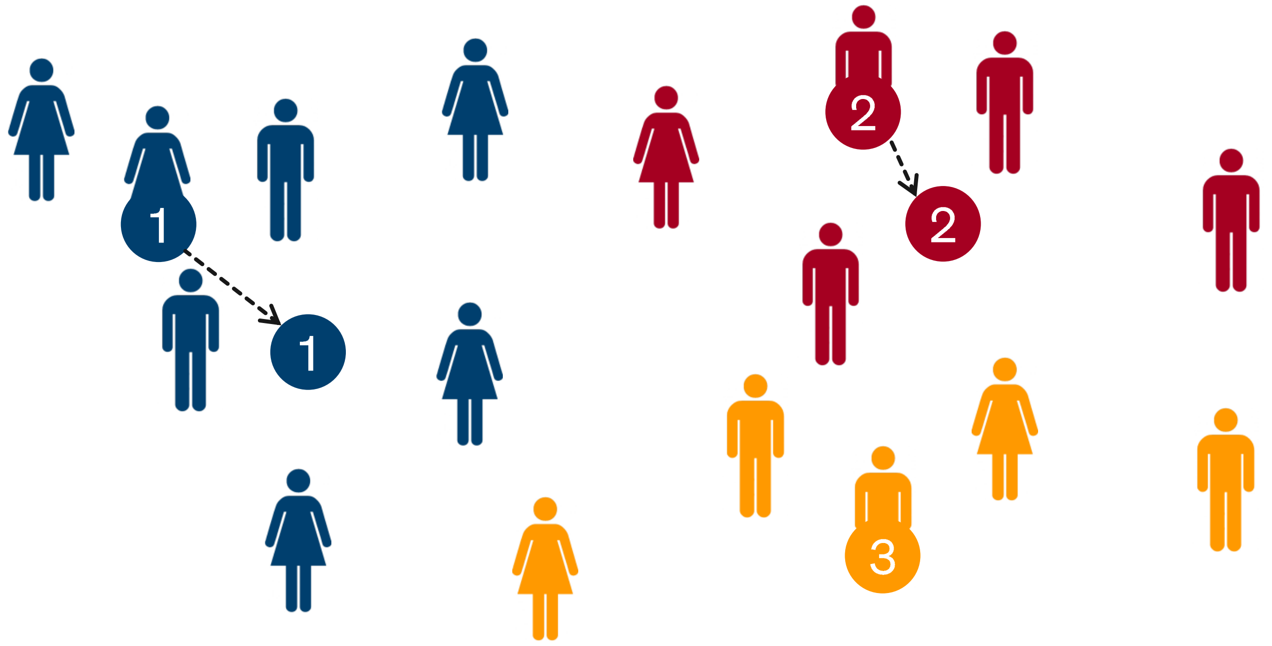
K-MEANS ALGORITHM

Assign individuals to the nearest center:



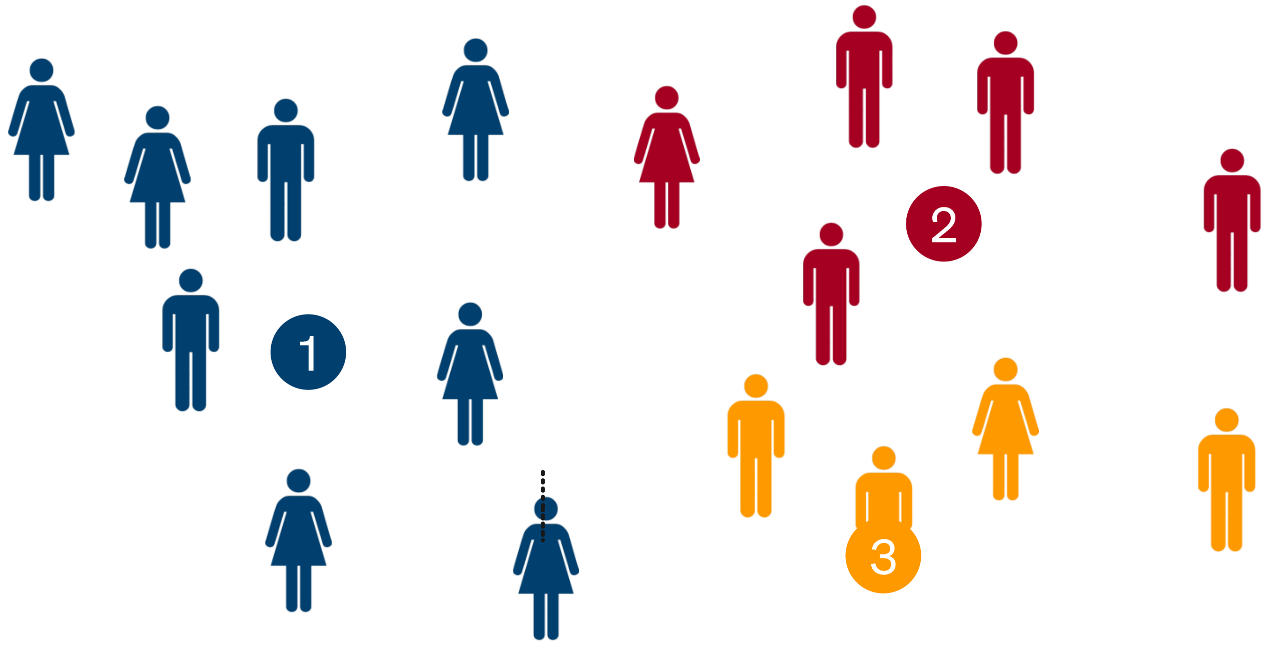
K-MEANS ALGORITHM

Find the new centroids:



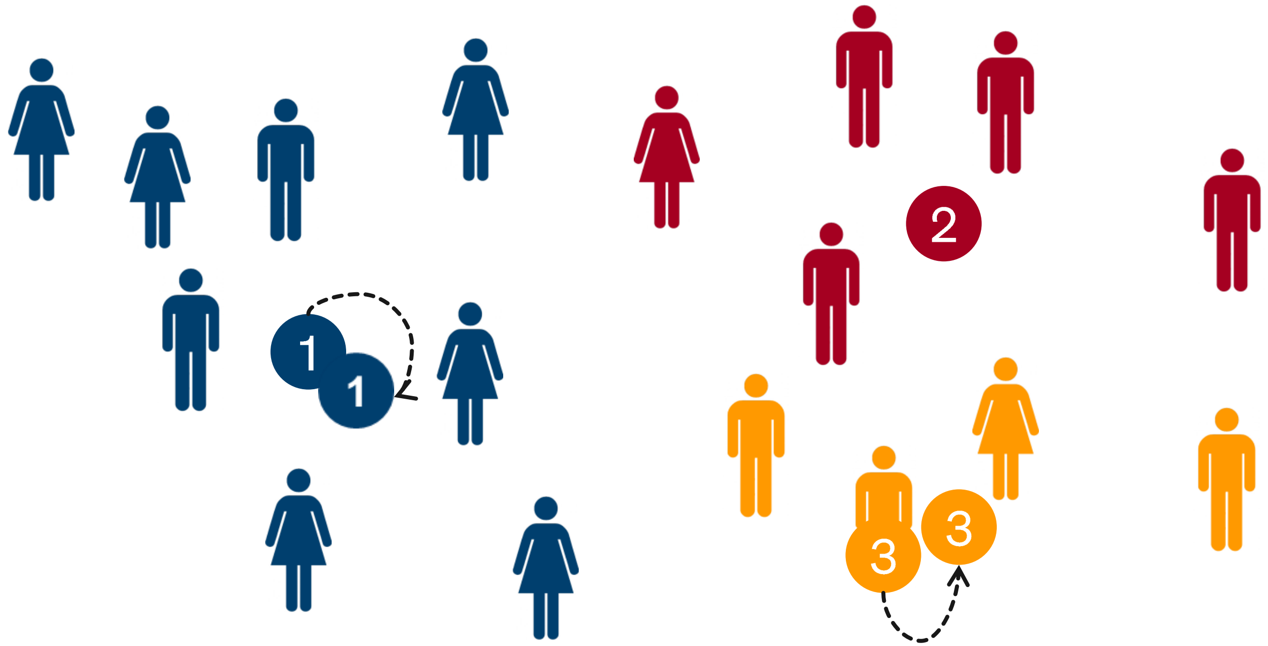
K-MEANS ALGORITHM

Re-allocate individuals to new centroids if required:



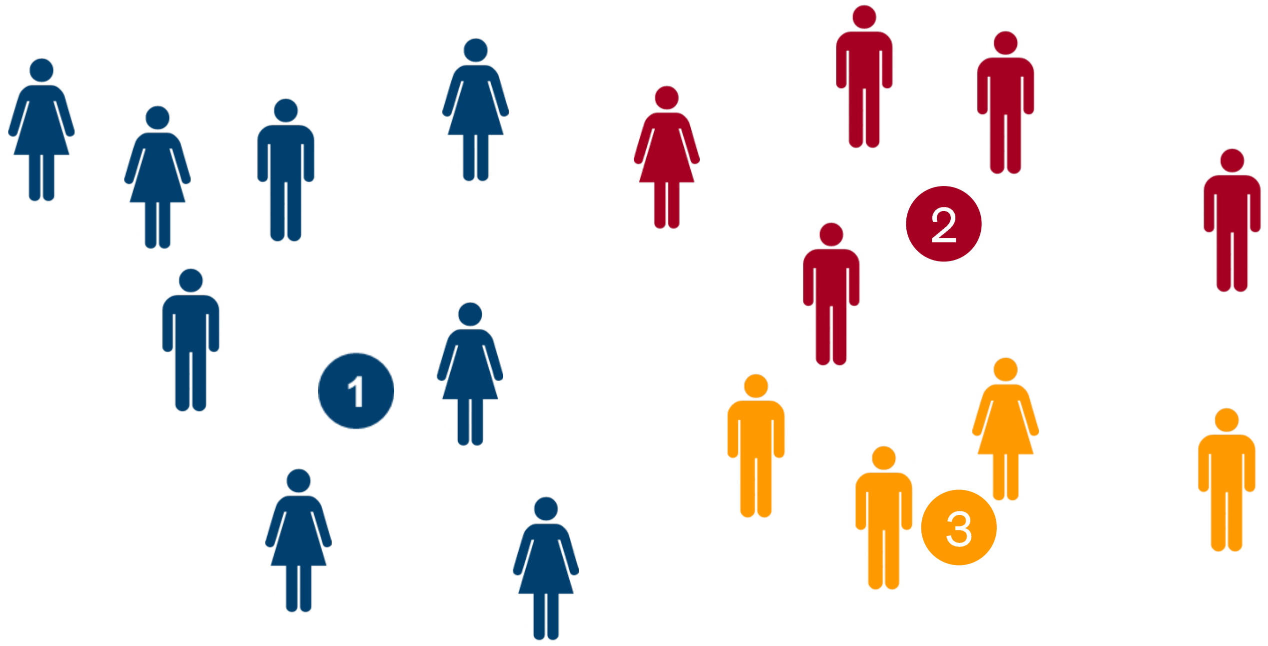
K-MEANS ALGORITHM

Find new centroids and re-allocate individuals:

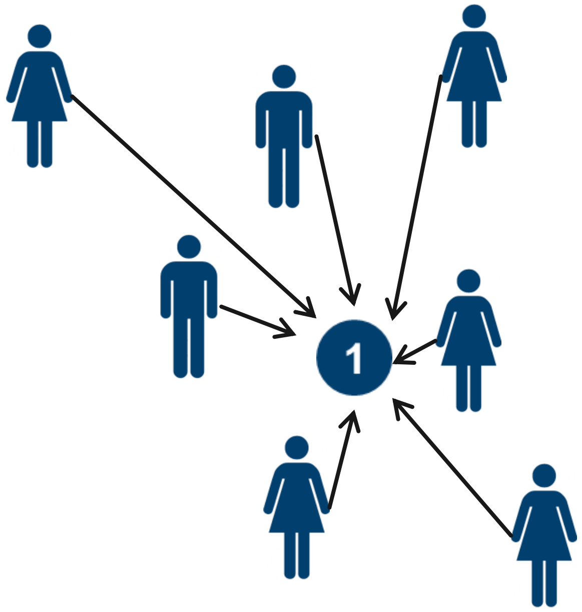


K-MEANS ALGORITHM

Final clusters of individuals:



CHOOSING AMONG DIFFERENT MODELS



The "error" is the distance of each point to the center of its own cluster:

SUM OF SQUARED ERRORS*

$$SSE = \sum_{f=1}^{K} \sum_{\mathbf{x}_i \in C_f} \|\mathbf{x}_i - \mathbf{m}_f\|^2$$

Given two clustering models we can choose the one with the smallest SSE.

*also known as total within-cluster variance

HOW MANY CLUSTERS?

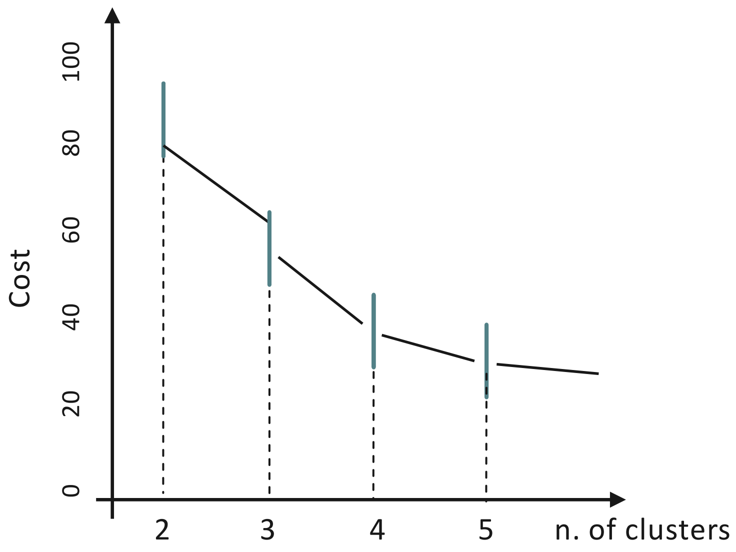
Alternative methods are available…(not exhaustive list)

- RULE OF THUMB $K \approx \left(\frac{m}{2}\right)^{1/2}$

- ELBOW METHOD

  The cost* the clustering model is set a
  a
  function of the number of clusters

  *Within-cluster variance (SSE) (sum of squared distances of each data point to its respective centroid/medoid).



- SILHOUETTE-BASED METHOD

  Choose the number of clusters giving rise to the largest average silhouette

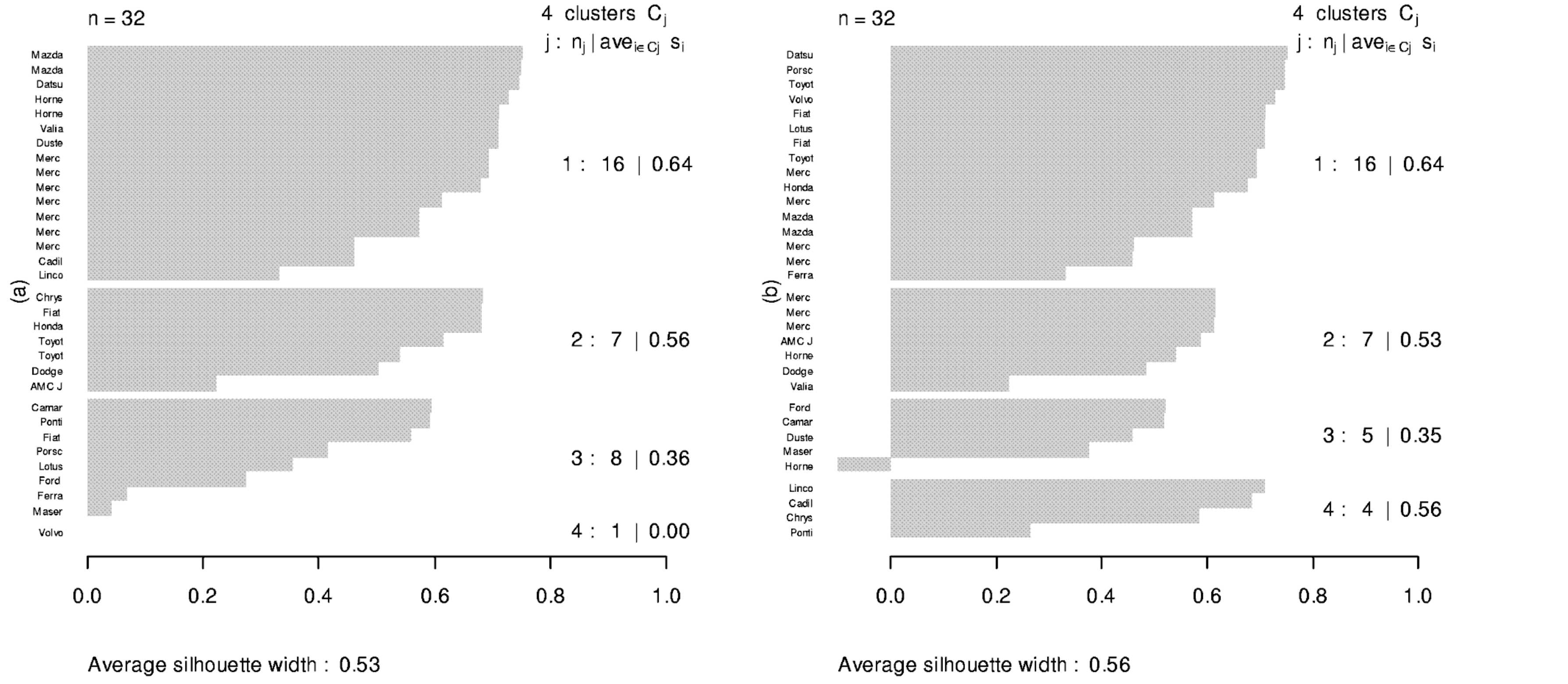
SILHOUETTE COEFFICIENT

- Compute the average distance u_i of $\mathbf{x}_i$ from all the other points in the same cluster
- Compute the average distance of $\mathbf{x}_i$ from all the other points comprised in a different cluster
  (let v_i be the minimum of these distances)
- The silhouette coefficient is given by

$$\text{silh}\left(\mathbf{x}_i\right) = \frac{v_i - u_i}{\max\left(u_i, v_i\right)}$$

- within [-1,1]
- the closer to 1 the better
- average silhouette

CLUSTERING EVALUATION



What is a good clustering?
Both external and internal validity measures are available

Internal measures: Used to measure the goodness of a clustering structure independently of external information

COHESION of each cluster
How close are elements in a cluster

$$\text{coes}(C_h) = \sum_{\substack{\mathbf{x}_i \in C_h \\ \mathbf{x}_k \in C_h}} \text{dist}(\mathbf{x}_i, \mathbf{x}_k)$$

SEPARATION of a pair of clusters
How distinct/well-separated a cluster is from the other

$$\text{sep}(C_h, C_f) = \sum_{\substack{\mathbf{x}_i \in C_h \\ \mathbf{x}_k \in C_f}} \text{dist}(\mathbf{x}_i, \mathbf{x}_k)$$

OVERALL COHESION

$$\text{coes}(\mathcal{C}) = \sum_{C_h \in \mathcal{C}} \text{coes}(C_h)$$

OVERALL SEPARATION

$$\text{sep}(\mathcal{C}) = \sum_{\substack{C_h \in \mathcal{C} \\ C_f \in \mathcal{C}}} \text{sep}(C_h, C_f)$$

The lower the cohesion and the higher the separation are, the better the clustering is.

CLUSTERING EVALUATION

External measures: Used to measure the extent to which cluster labels match externally supplied class labels (ground truth)

PURITY INDEX

Ratio between the dominant class in a given cluster and the size of the cluster

RAND INDEX

Percentage of points “correctly” assigned among clusters:

$$RI = \frac{\mathrm{n_A} + \mathrm{n_B}}{\mathrm{n_A} + \mathrm{n_B} + \mathrm{n_C} + \mathrm{n_D}}$$

	Same cluster in clustering	Different clusters in clustering
Same class in ground truth	n_A	n_C
Different classes in ground truth	n_D	n_B

COMMENTS ON THE K-MEANS ALGORITHM

STRENGTH AND WEAKNESS

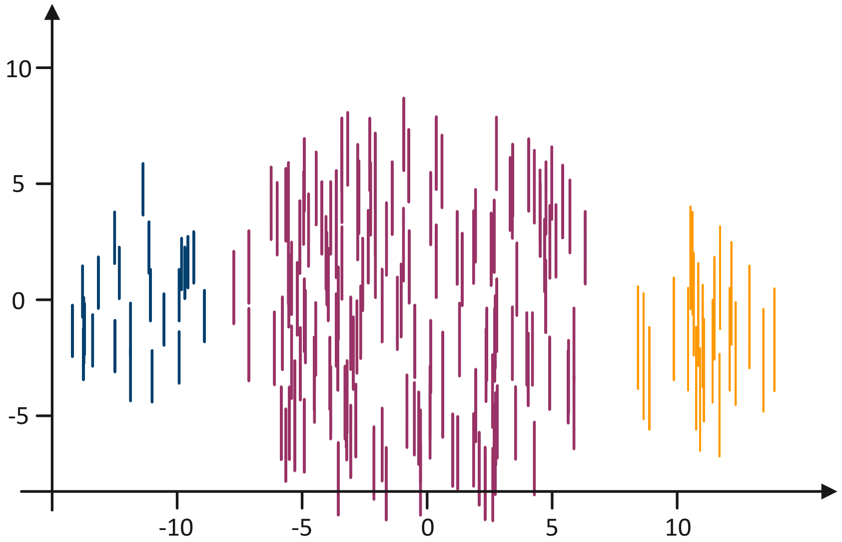
- Relatively efficient: O(m*K*t) [m points, K clusters, t iterations]
- It converges for common similarity measures (most of the convergence happens in the first few iterations)
- Need to specify the number of clusters in advance
- Often terminates at a local optimum (greedy) which are usually close to the global best
- Results may vary based on random seed selection:
  - clusters may be different from one run to another
  - very hard to repeat the clustering results

  [Try out multiple starting points]
- Applicable only when mean is defined…what about categorical data?
- Unable to handle noisy data and outliers

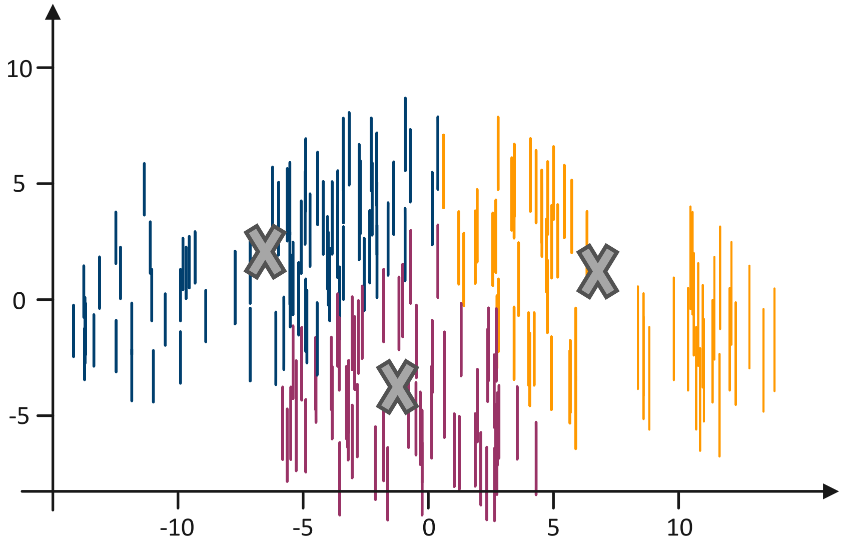
COMMENTS ON THE K-MEANS ALGORITHM

STRENGTH AND WEAKNESS

- It may have problems when (natural) clusters:
  - are of different sizes



original data set

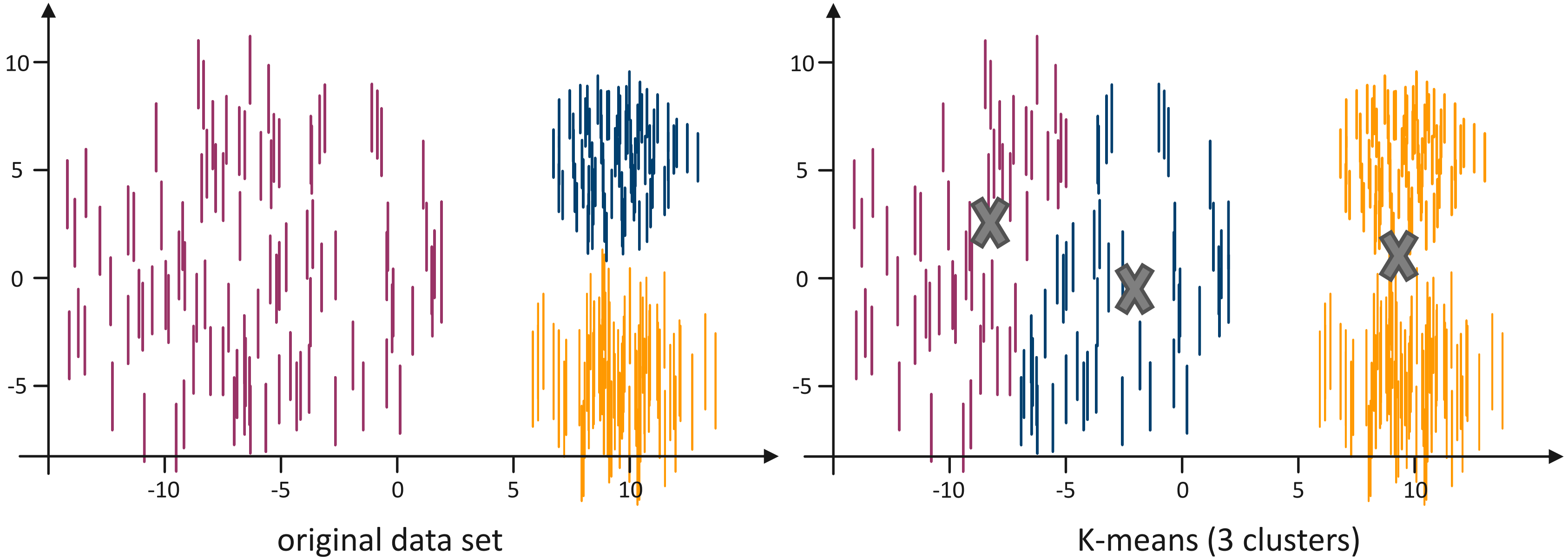


K-means (3 clusters)

COMMENTS ON THE K-MEANS ALGORITHM

STRENGTH AND WEAKNESS

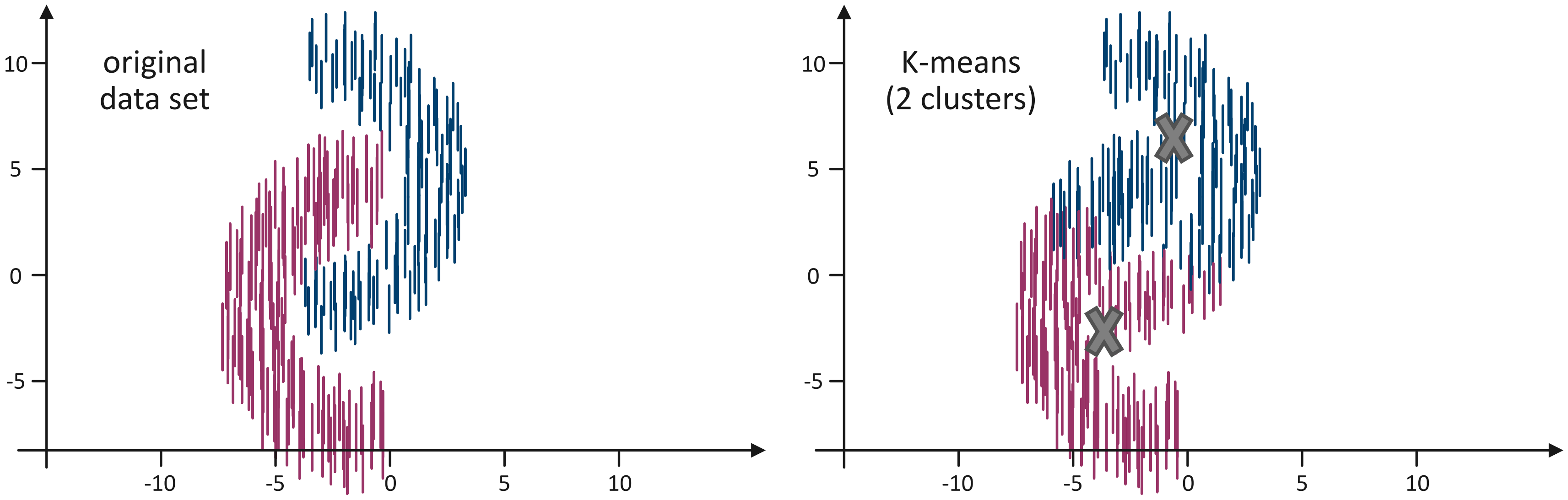
- It may have problems when (natural) clusters:
  - are of different sizes
  - have different densities



COMMENTS ON THE K-MEANS ALGORITHM

STRENGTH AND WEAKNESS

- It may have problems when (natural) clusters:
  - are of different sizes
  - have different densities
  - have non-convex shapes



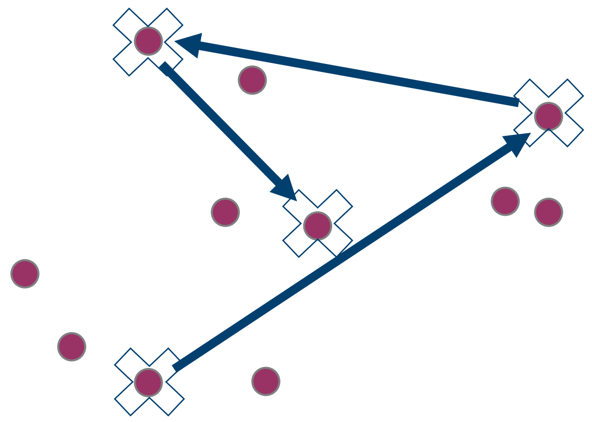
One solution is to use many clusters ⇒ find parts of clusters ⇒ need to put them together

To find a good solution to the K-center problem (initial selection of seeds/medoids).

FFT ALGORITHM

Let T be the set of selected centers (seeds/medoids):

1) Pick any data point from the dataset and add it to the list T.
2) While not the list T is completed:
   - Among the not-yet-selected points find a point that has the maximum distance from the selected points;
   - Add this point to T.



K-MEANS++ (PROBABILISTIC FFT)

1) Pick any data point from the dataset and add it to the list T.
2) While not the list T is completed:
   - For each not-yet-selected points compute a probability (to get selected) proportional to its squared-distance from the selected points;
   - Select a point based on its probability and add it to T.

HIERARCHICAL METHODS

HIERARCHICAL METHODS

- Are based on a tree structure (dendogram)
- Use the distances among points to derive clusters merging or splitting
- Do not require the number K of clusters in input

AGGLOMERATIVE ALGORITHMS (bottom-up techniques)

- Initially each point represents a single cluster
- Iteratively the two clusters with the minimum distance are merged together
- All points are comprised in a single cluster $\Rightarrow$ Stop

DIVISIVE ALGORITHMS (top-down techniques)

- Initially all points are comprised in a single cluster
- Iteratively split a cluster to obtain two clusters with the maximum distance
- Each point represents a single cluster $\Rightarrow$ Stop

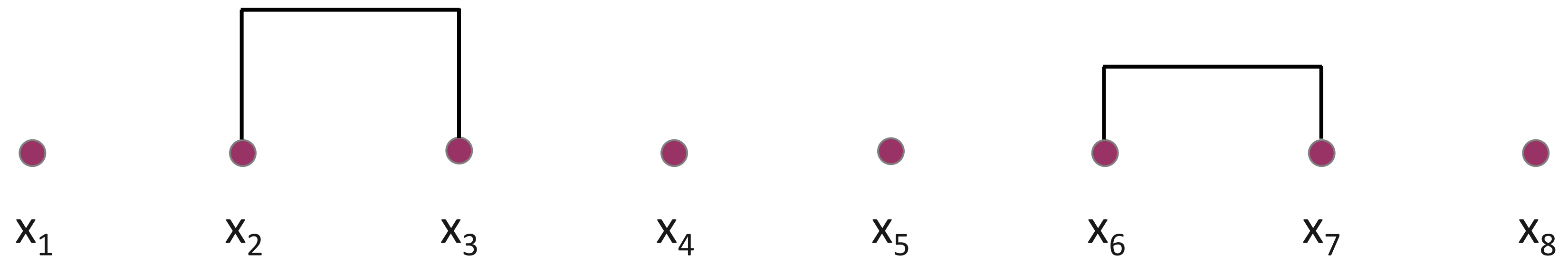
AGGLOMERATIVE HIERARCHICAL METHODS

Iteration 1: 8 clusters (init)

AGGLOMERATIVE HIERARCHICAL METHODS

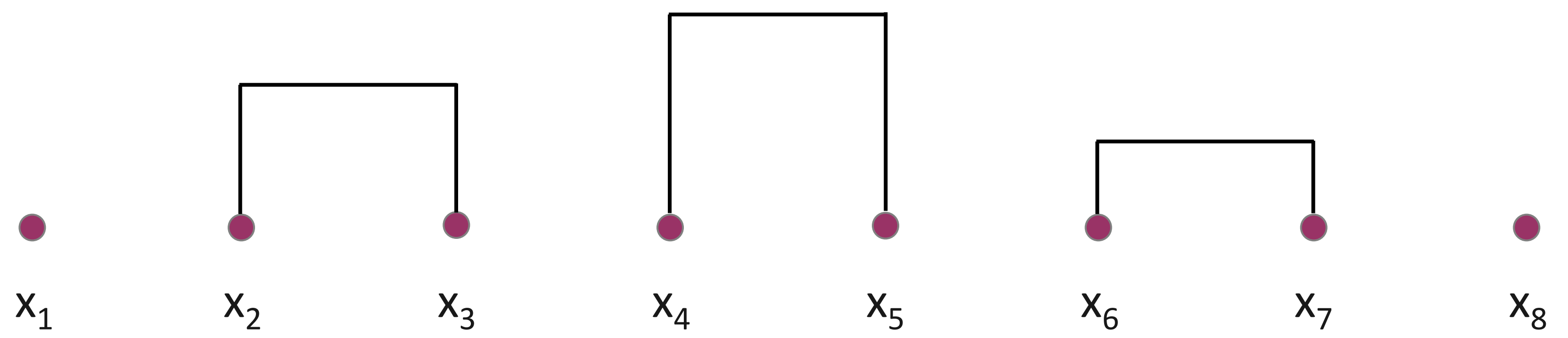
Iteration 2: 7 clusters

Iteration 3: 6 clusters



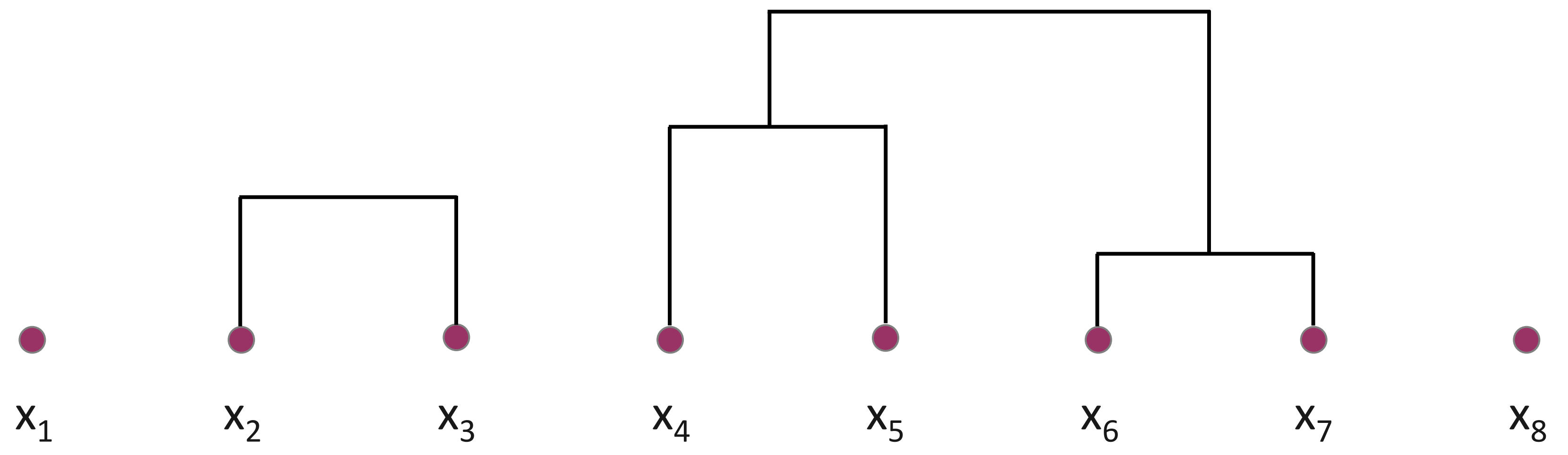
AGGLOMERATIVE HIERARCHICAL METHODS

Iteration 4: 5 clusters



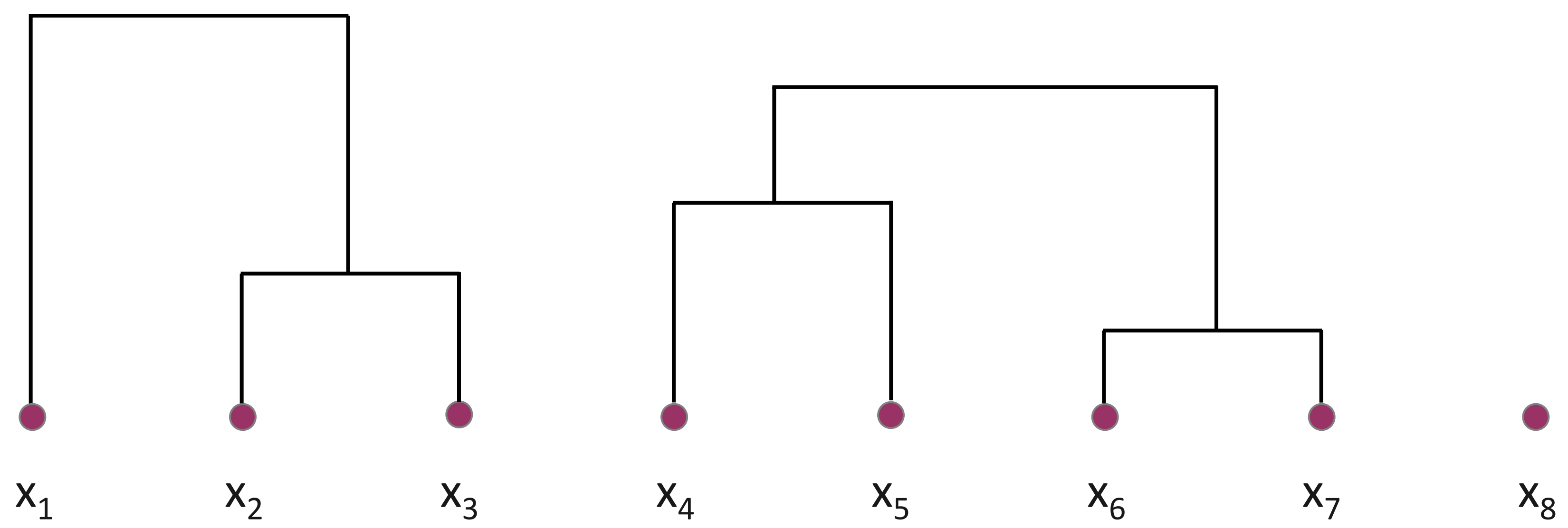
AGGLOMERATIVE HIERARCHICAL METHODS

Iteration 5: 4 clusters



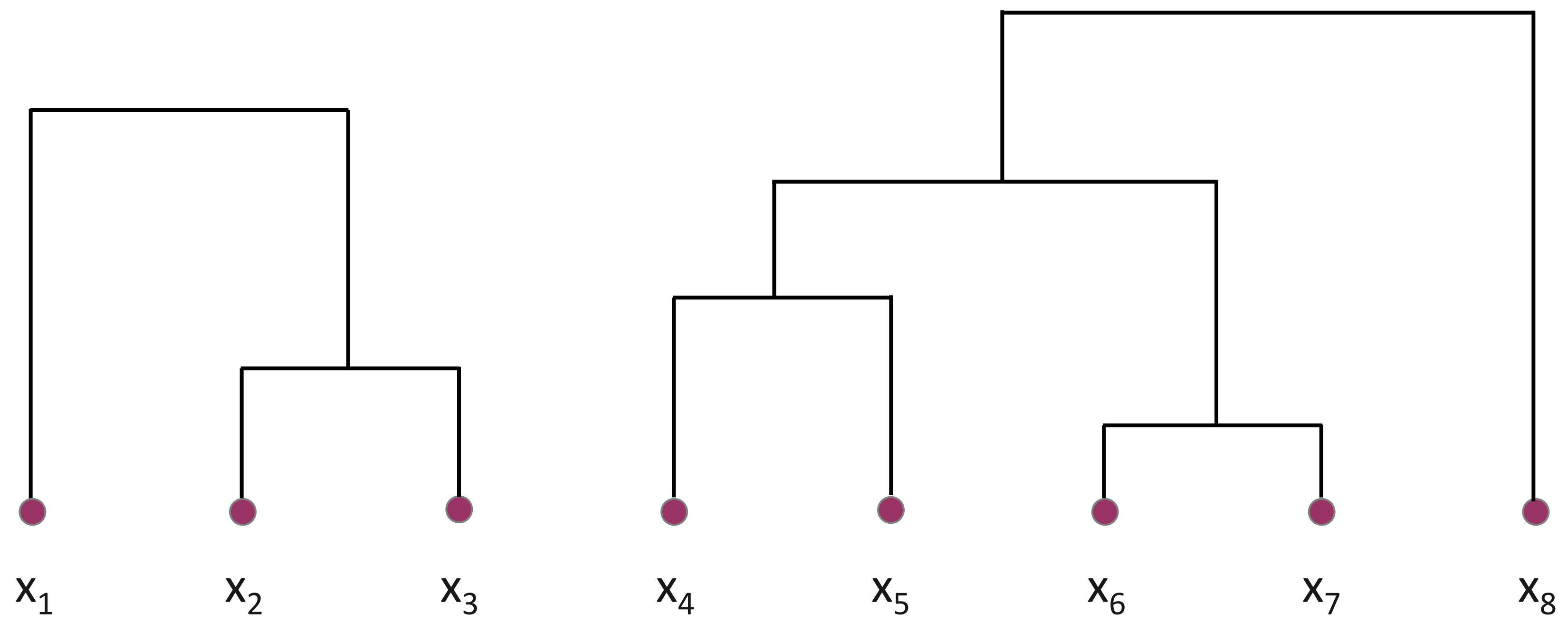
AGGLOMERATIVE HIERARCHICAL METHODS

Iteration 6: 3 clusters

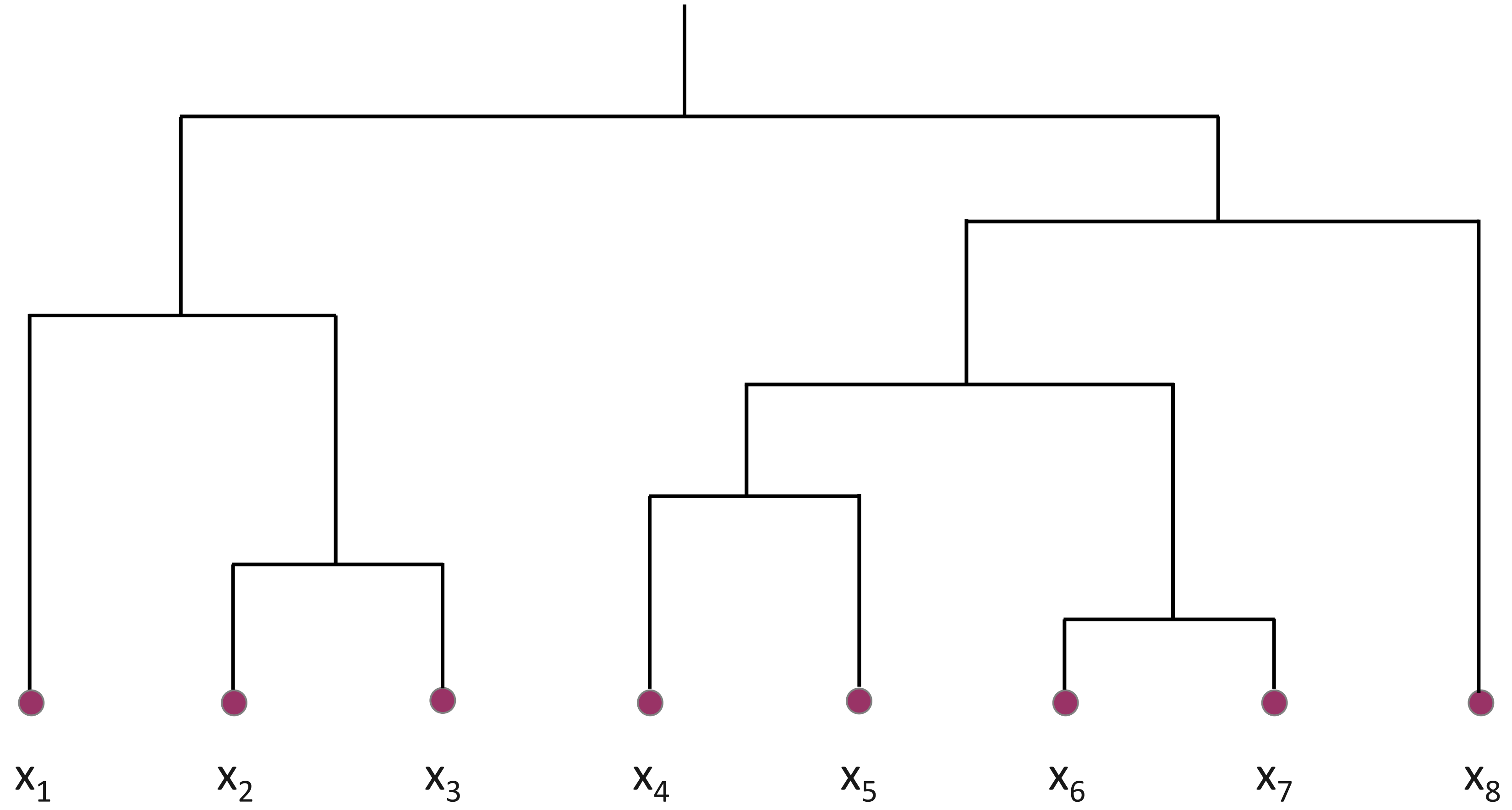


AGGLOMERATIVE HIERARCHICAL METHODS

Iteration 7: 2 clusters



Iteration 8: 1 cluster



The criterion for choosing the pair of clusters to merge at each step is based on the optimization of an objective function…such as their "proximity".

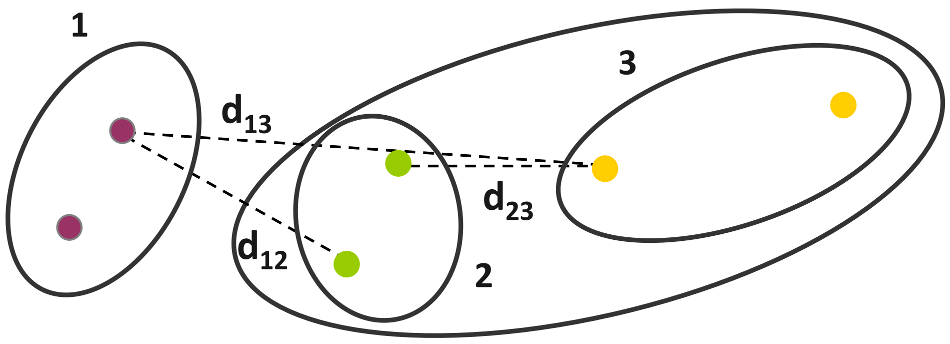
Several metrics to evaluate the proximity (distance) between a pair of clusters:

MINIMUM DISTANCE (SINGLE LINKAGE)

The distance of two clusters is based on the two most similar (closest) points in the different clusters

$$\mathrm{dist}(C_h, C_f) = \min_{\substack{\mathbf{x}_i \in C_h \\ \mathbf{x}_k \in C_f}} \mathrm{dist}(\mathbf{x}_i, \mathbf{x}_k)$$

- sensitive to noise and outliers
- biased towards elliptical clusters

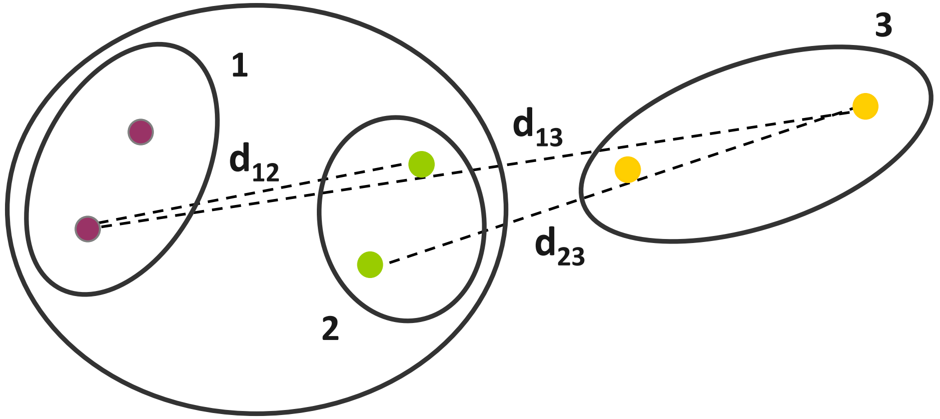


MAXIMUM DISTANCE (COMPLETE LINKAGE)

The distance of two clusters is based on the two least similar (most distant) points in the different clusters

$$\mathrm{dist}(C_h, C_f) = \max_{\substack{\mathbf{x}_i \in C_h \\ \mathbf{x}_k \in C_f}} \mathrm{dist}(\mathbf{x}_i, \mathbf{x}_k)$$

- less sensitive to noise and outliers
- tends to break large clusters
- biased towards globular clusters

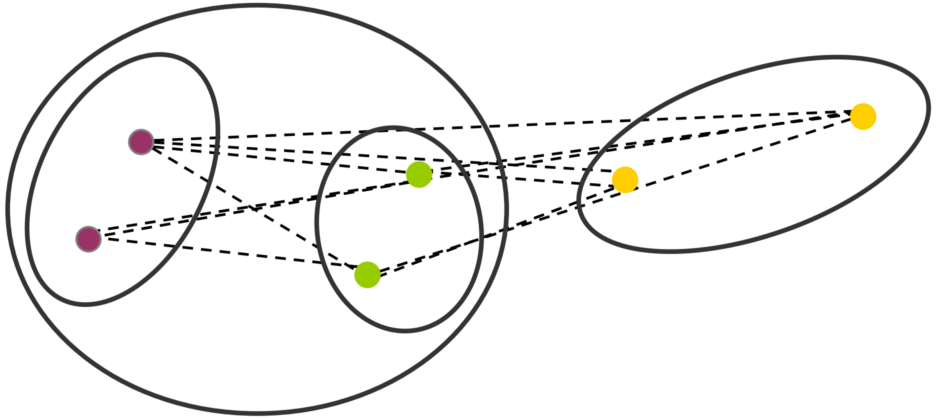


AVERAGE DISTANCE

The distance of two clusters is the average of the sum of the pairwise distances of the points in the two clusters

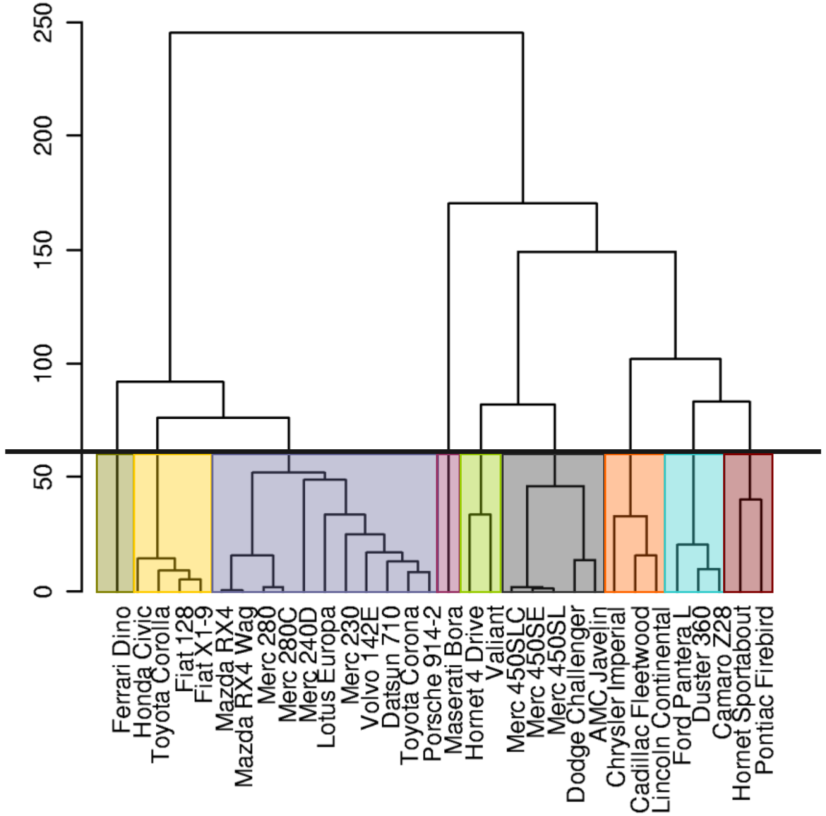
$$\mathrm{dist}(C_h, C_f) = \frac{\sum_{\mathbf{x}_i \in C_h} \sum_{\mathbf{x}_k \in C_f} \mathrm{dist}(\mathbf{x}_i, \mathbf{x}_k)}{\mathrm{card}\{C_h\}\,\mathrm{card}\{C_f\}}$$

- less sensitive to noise and outliers
- biased towards globular clusters

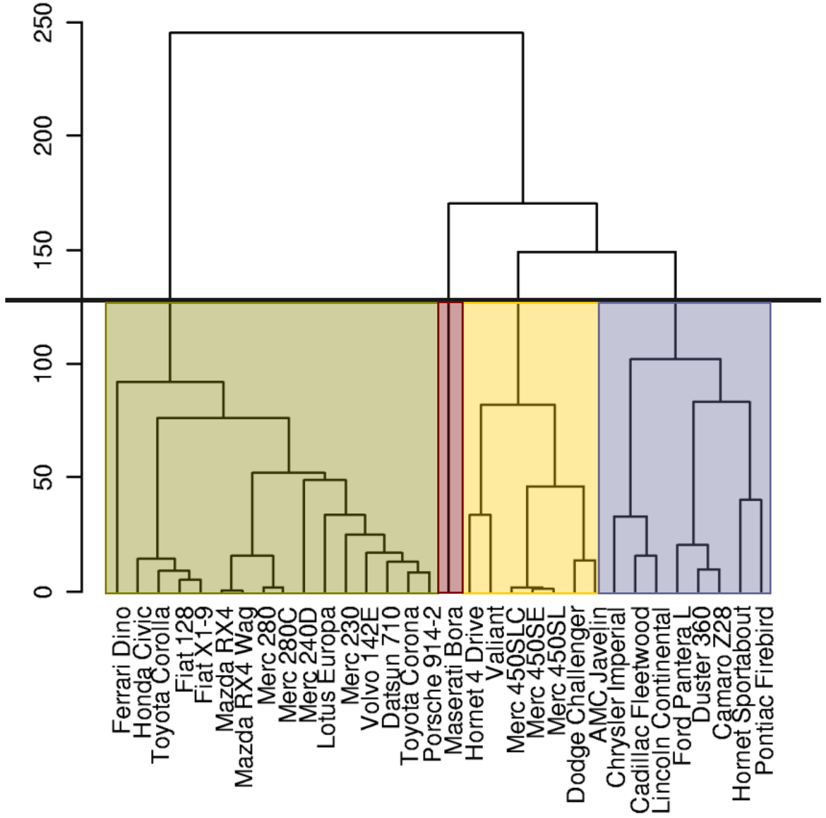


AGGLOMERATIVE HIERARCHICAL METHODS

Any desired number of clusters can be obtained by “cutting” the dendogram at the proper level.

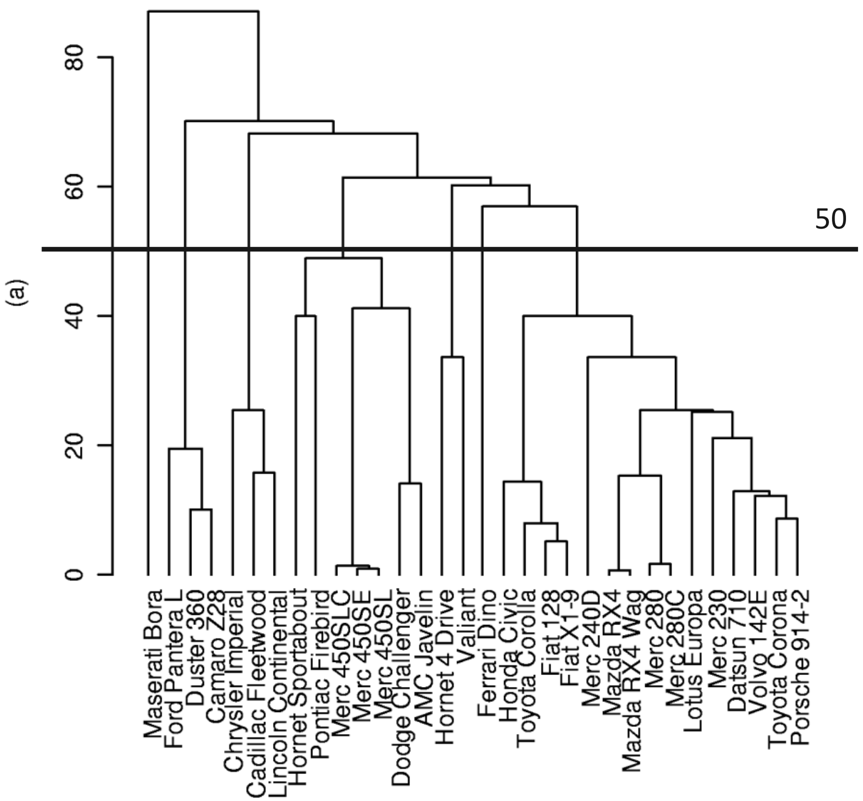


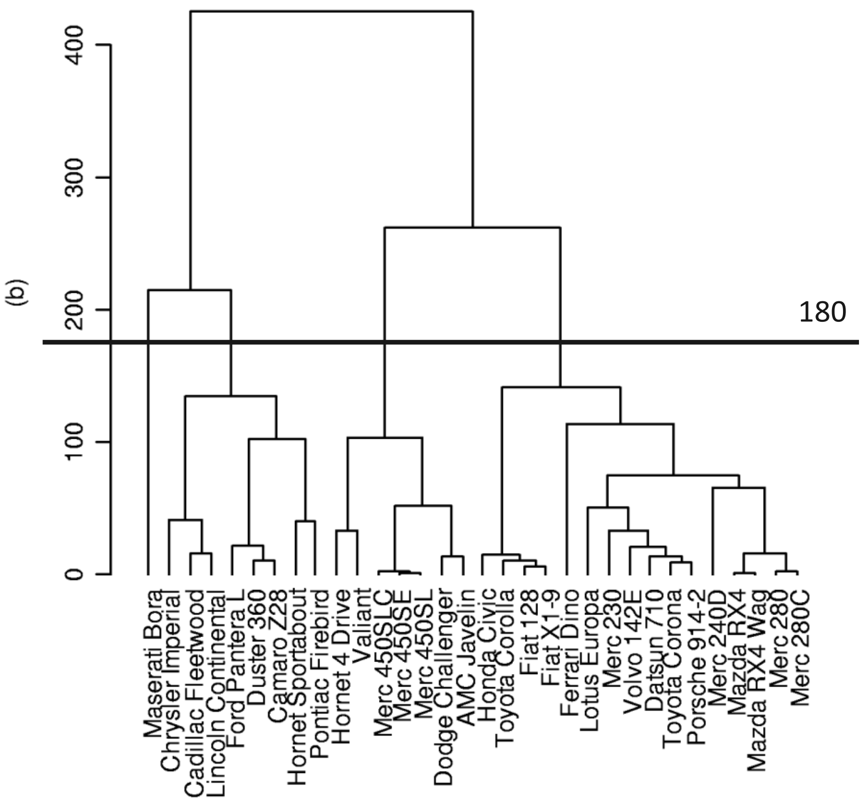
Average Euclidean distance
9 clusters cut



Average Euclidean distance
4 clusters cut

AGGLOMERATIVE HIERARCHICAL METHODS





Minimum Euclidean distance

Cut interpretation

For each point x_i in a cluster there is another point x_j in the same cluster for which $d_{ij} \le 50$

Maximum Euclidean distance

Cut interpretation

For each point x_i in a cluster every other points x_j in the same cluster satisfies $d_{ij} \le 180$

COMMENTS ON HIERARCHICAL METHODS

STRENGTH AND WEAKNESS

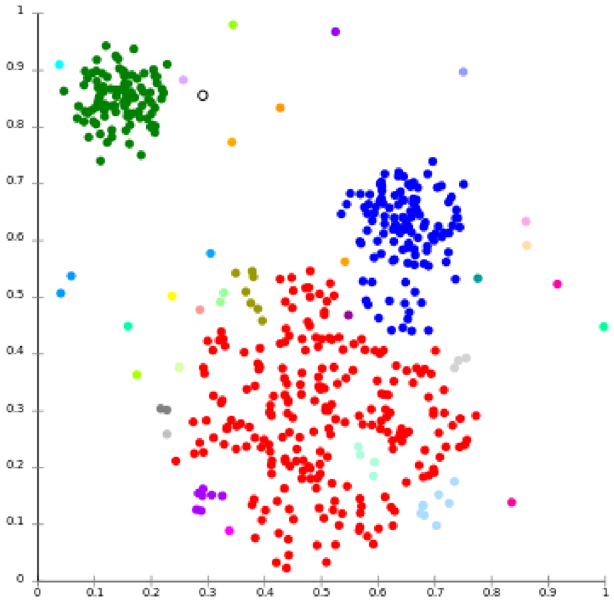
- They do not produce a unique partitioning of the data set but a hierarchy
  (User needs to choose appropriate clusters)

- The complexity is $O(m^3)$ (too slow for large data sets)
  (more efficient algorithms [$O(m^2)$] have been proposed)

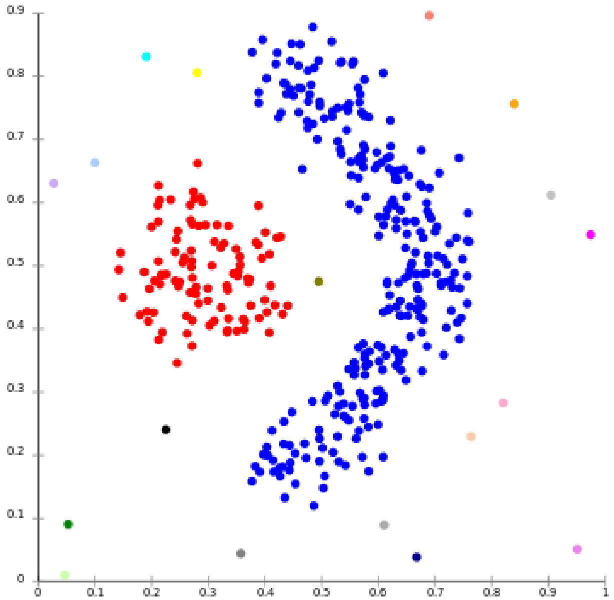
- They are not very robust towards outliers
  Outliers may show up as additional clusters or cause other clusters to merge ("chaining phenomenon" in particular with single-linkage clustering)

COMMENTS ON HIERARCHICAL METHODS

On the choice of the number of clusters in the presence of noise and outliers…



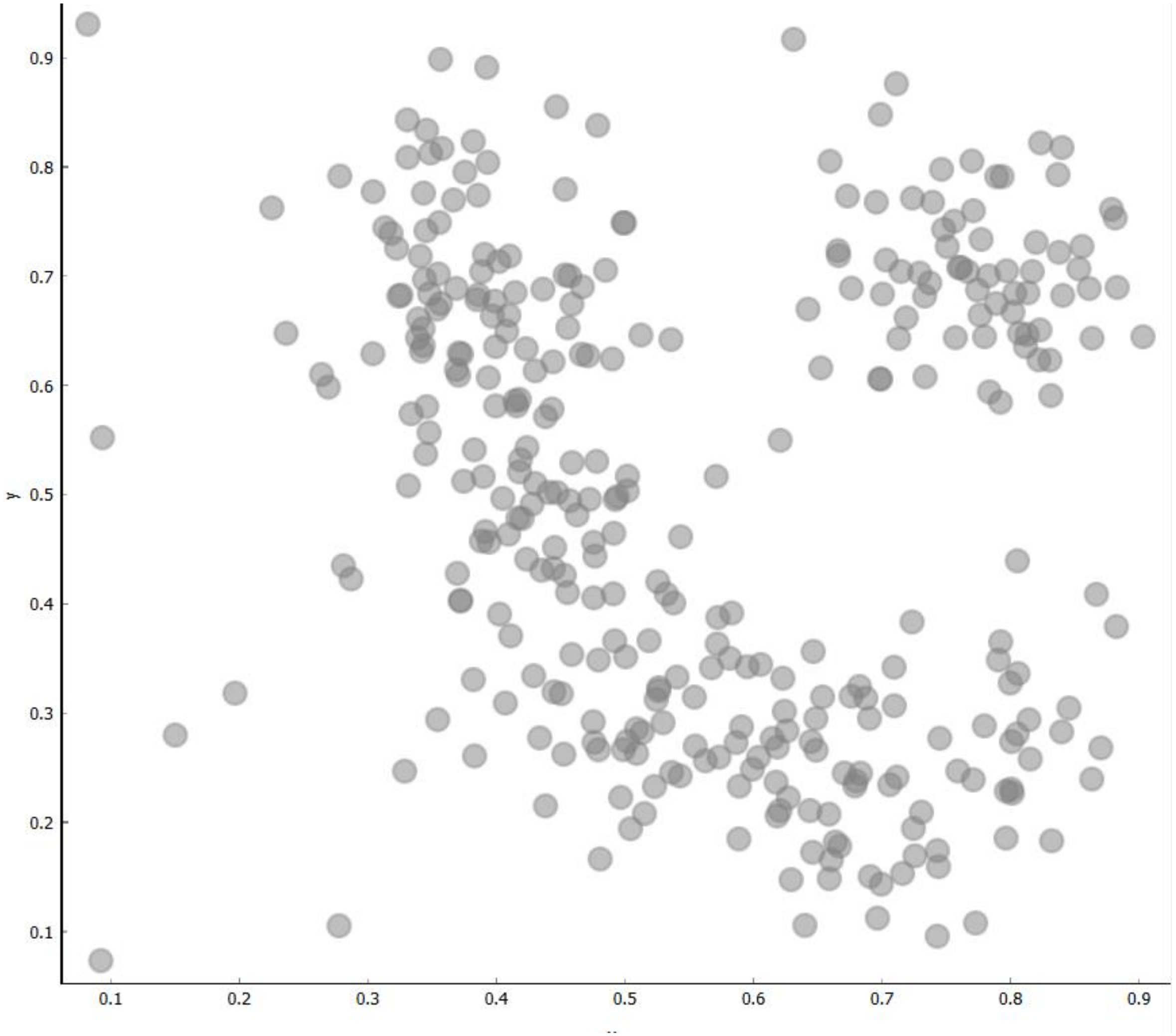
Single-linkage on Gaussian data.
At 35 clusters, the biggest cluster starts fragmenting into smaller parts.



Single-linkage on density-based clusters.
20 clusters extracted, most of which contain single elements (these methods do not have a notion of "noise“).

ANOTHER WAY OF CLUSTERING

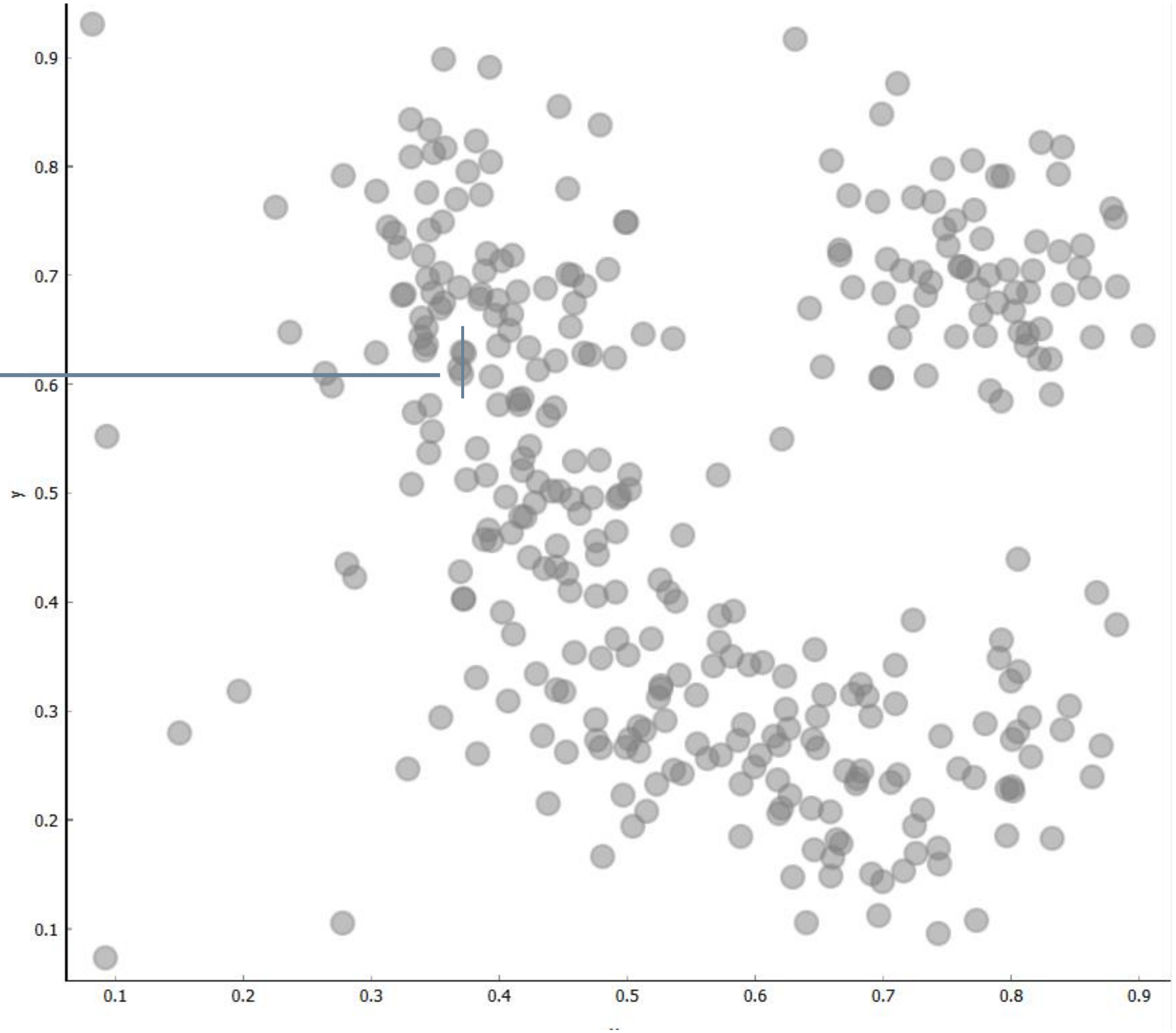
How do we (human) group observations?



ANOTHER WAY OF CLUSTERING

How do we (human) group observations?

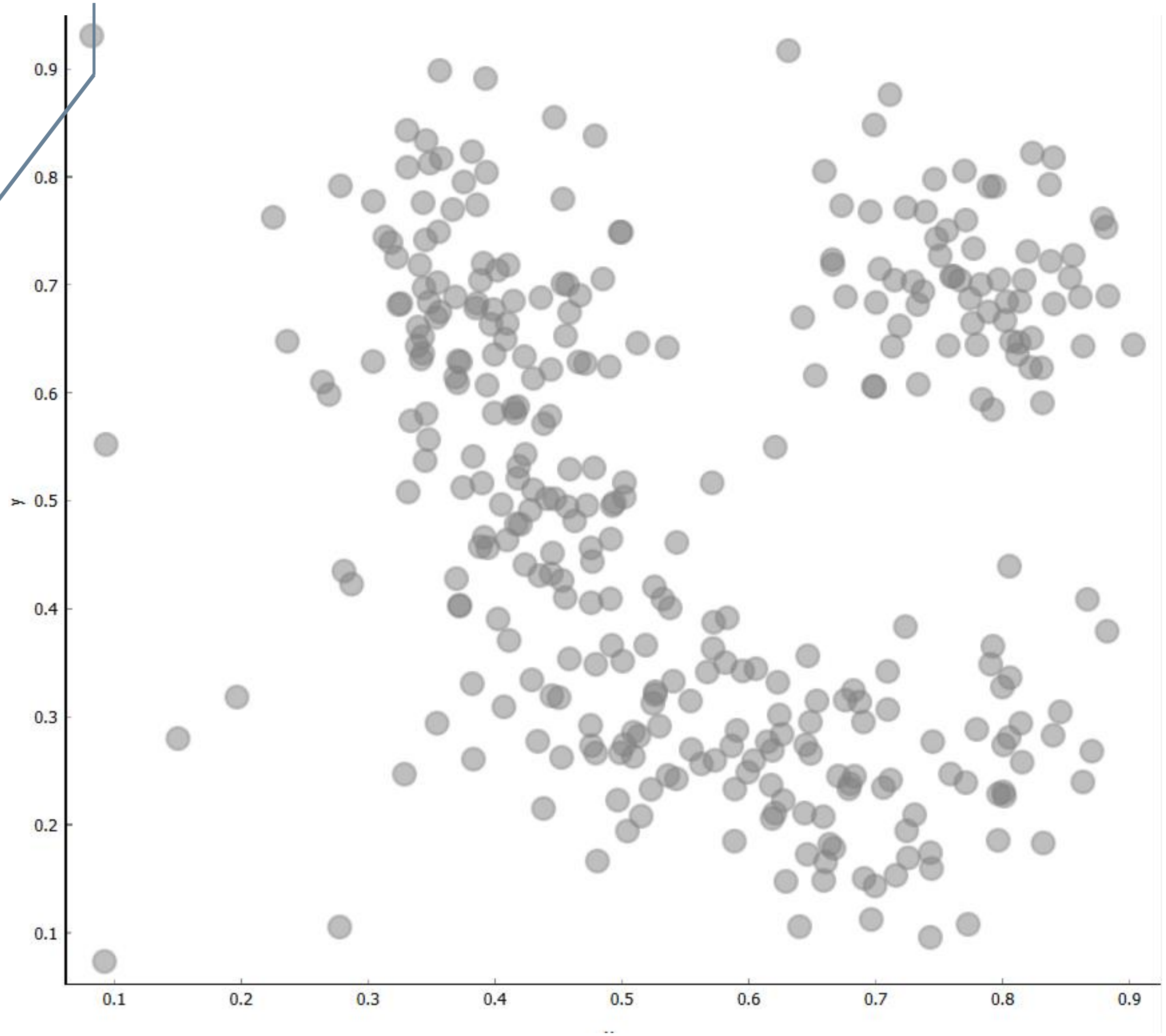
Part of a “cloud” of points



ANOTHER WAY OF CLUSTERING

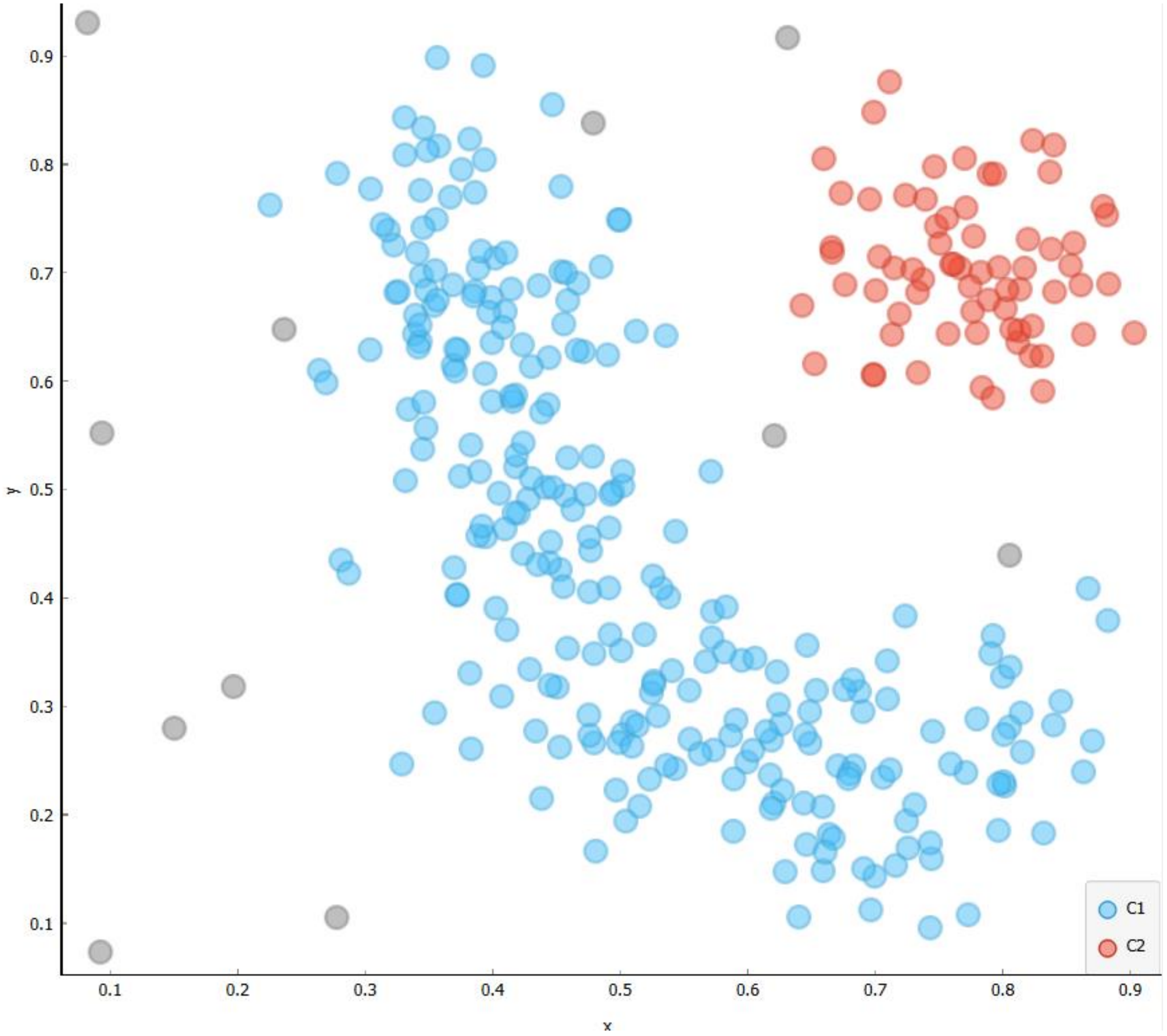
How do we (human) group observations?

Far away from anything else



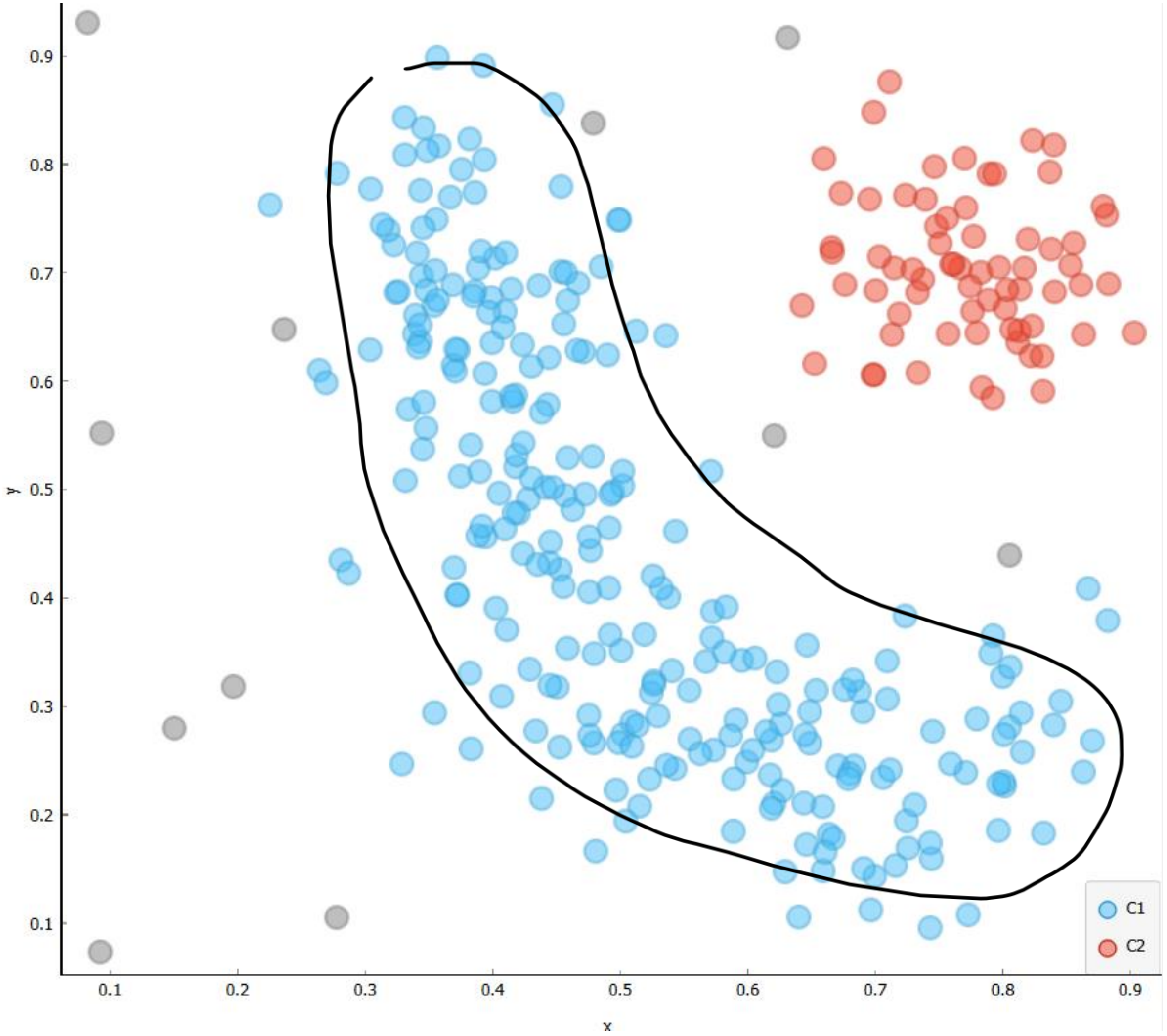
ANOTHER WAY OF CLUSTERING

How do we (human) group observations?



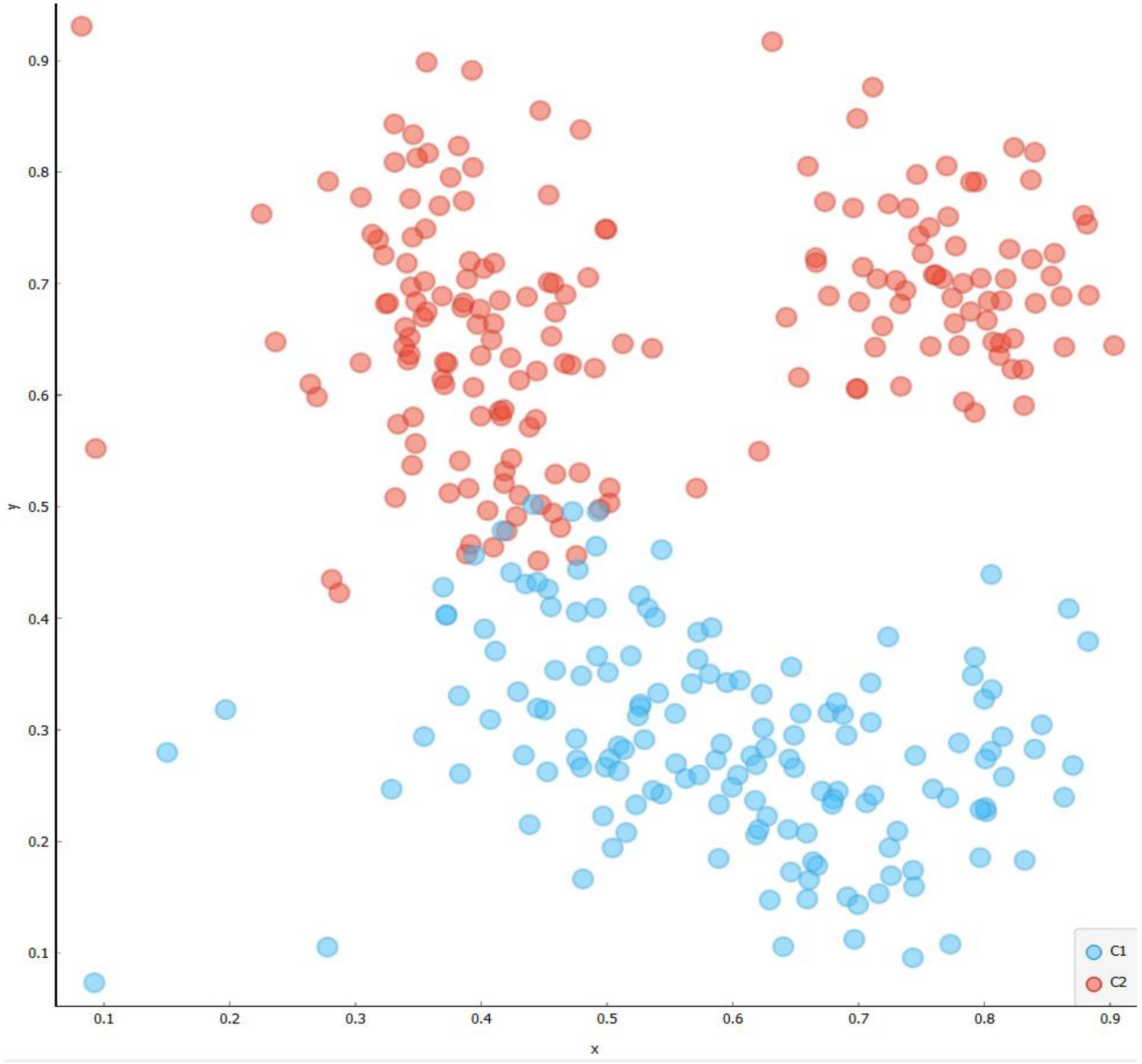
ANOTHER WAY OF CLUSTERING

Non convex shape!



ANOTHER WAY OF CLUSTERING

K-means gets us this



DBSCAN

1) Count the number of neighbors of each observation
(given a distance threshold)

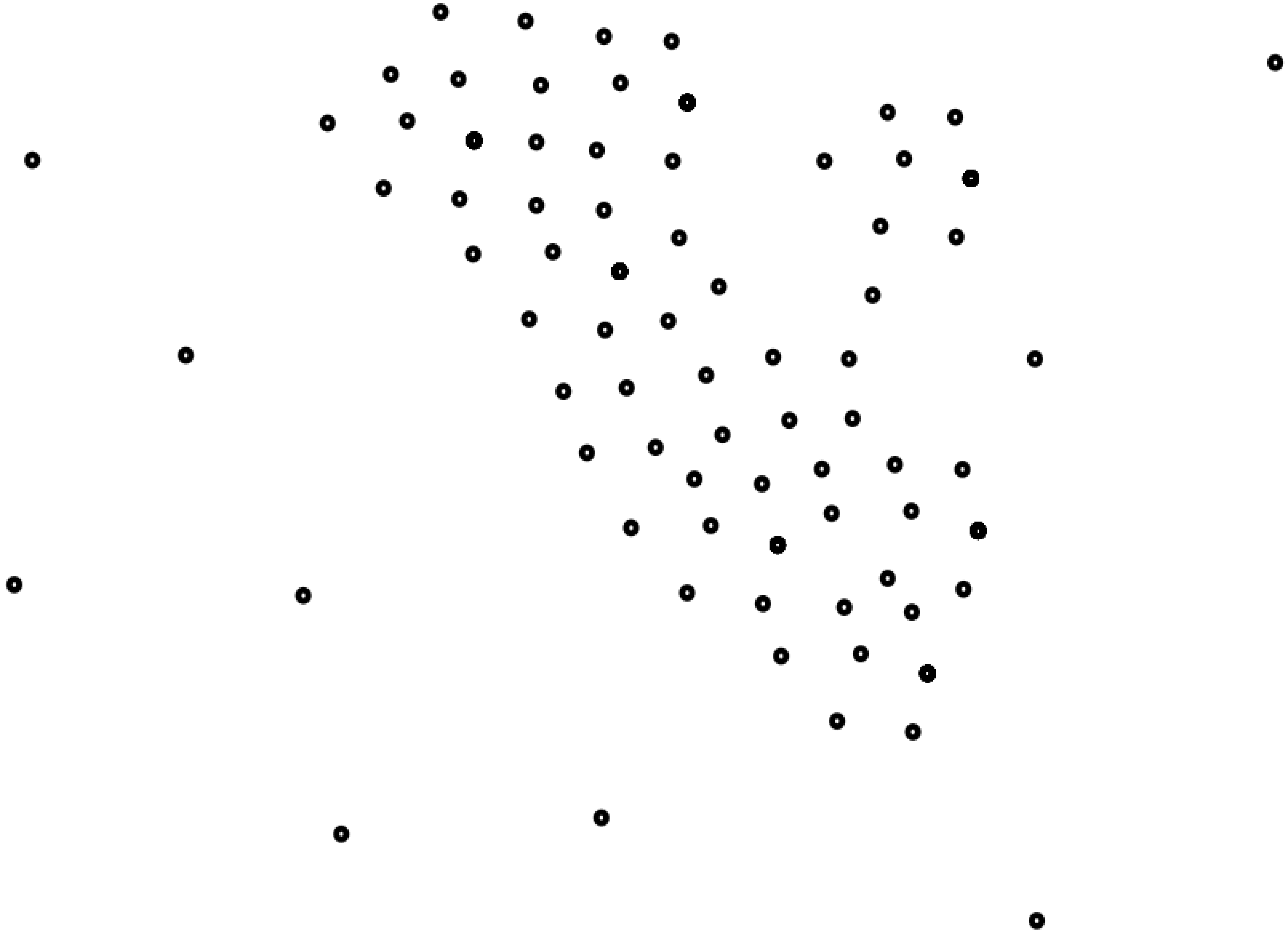
2) Identify the number of core observations
(observations with #neighbors ≥ threshold)

3) Randomly pick a core observation and assign it to cluster a new cluster C
4) Recursively add core obs. (only!) in the neighborhood of core obs. in C to C
5) Add unclustered non-core points in the neighborhood of core obs. in C

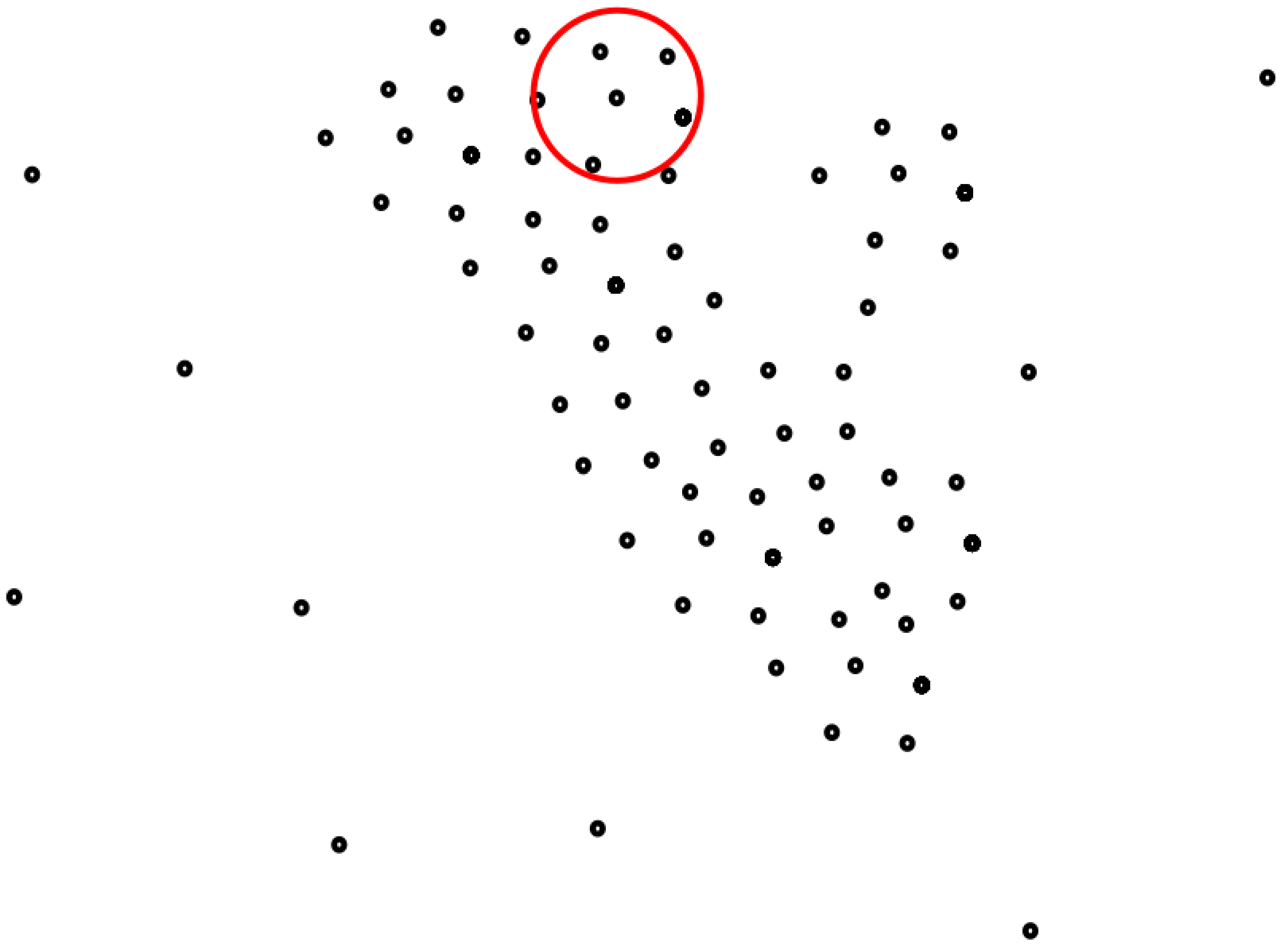
6) Repeat 3) to 5) until no core observations remain

7) Any non-core observation remaining is not clustered

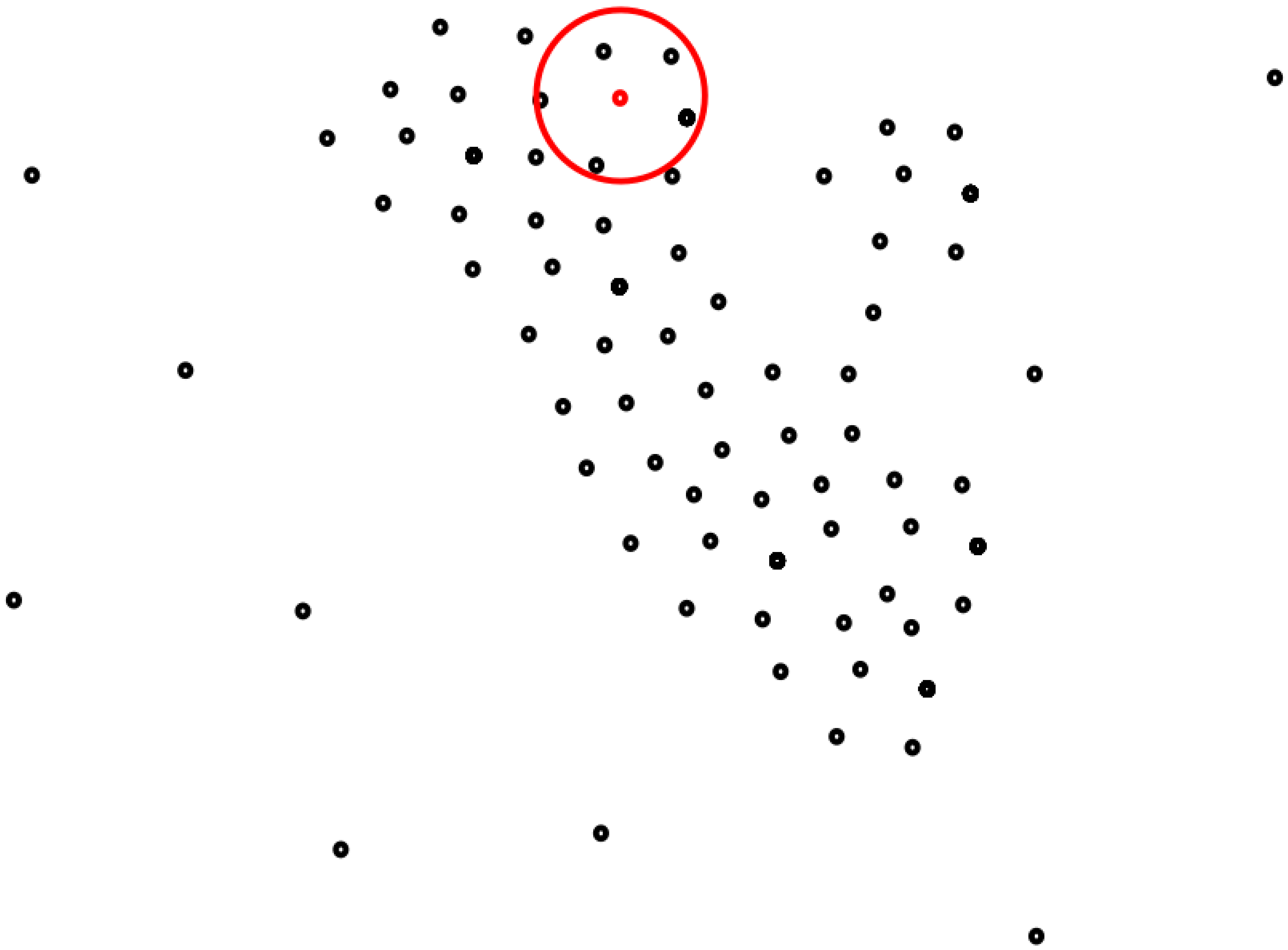
DBSCAN



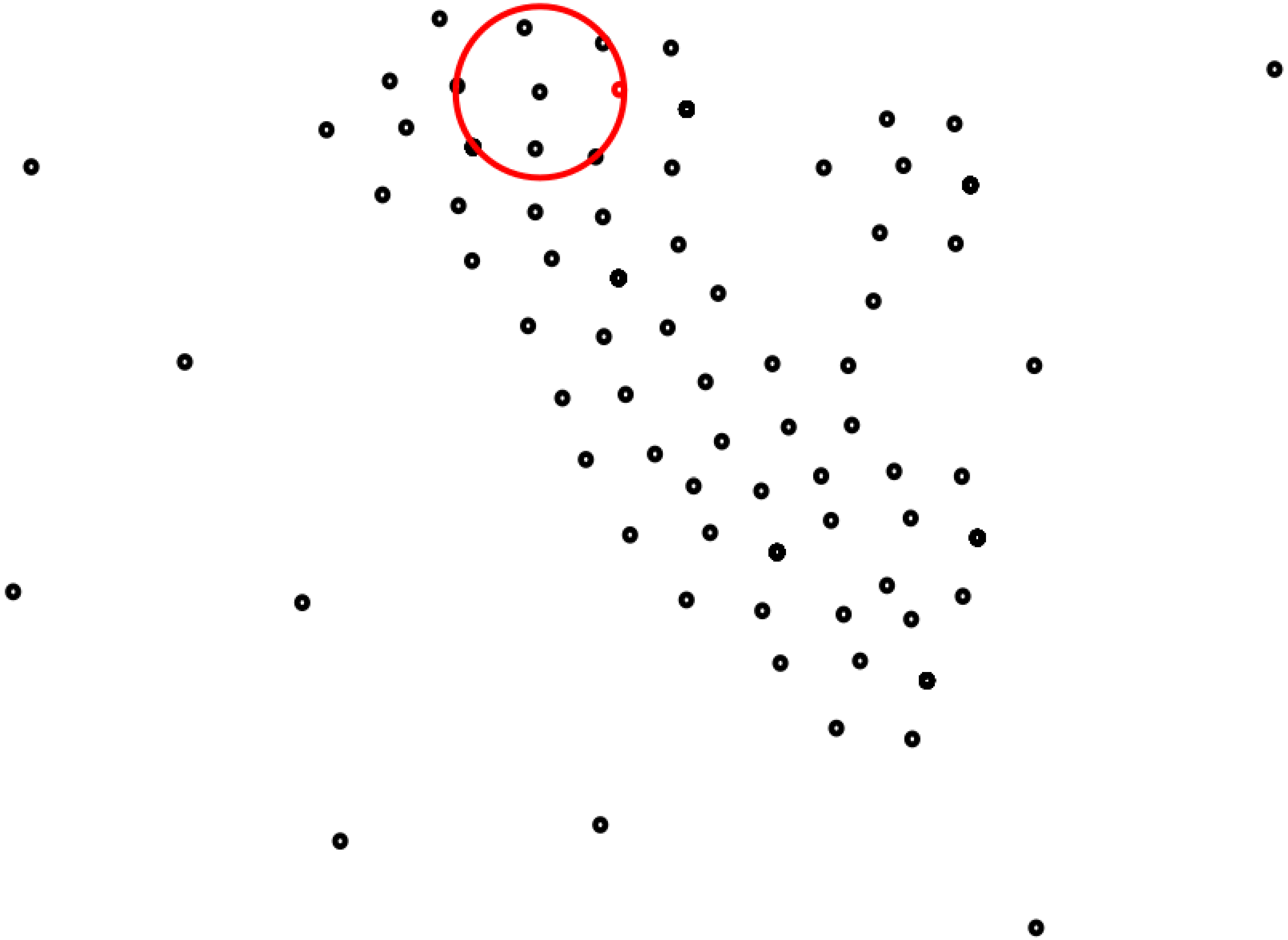
DBSCAN



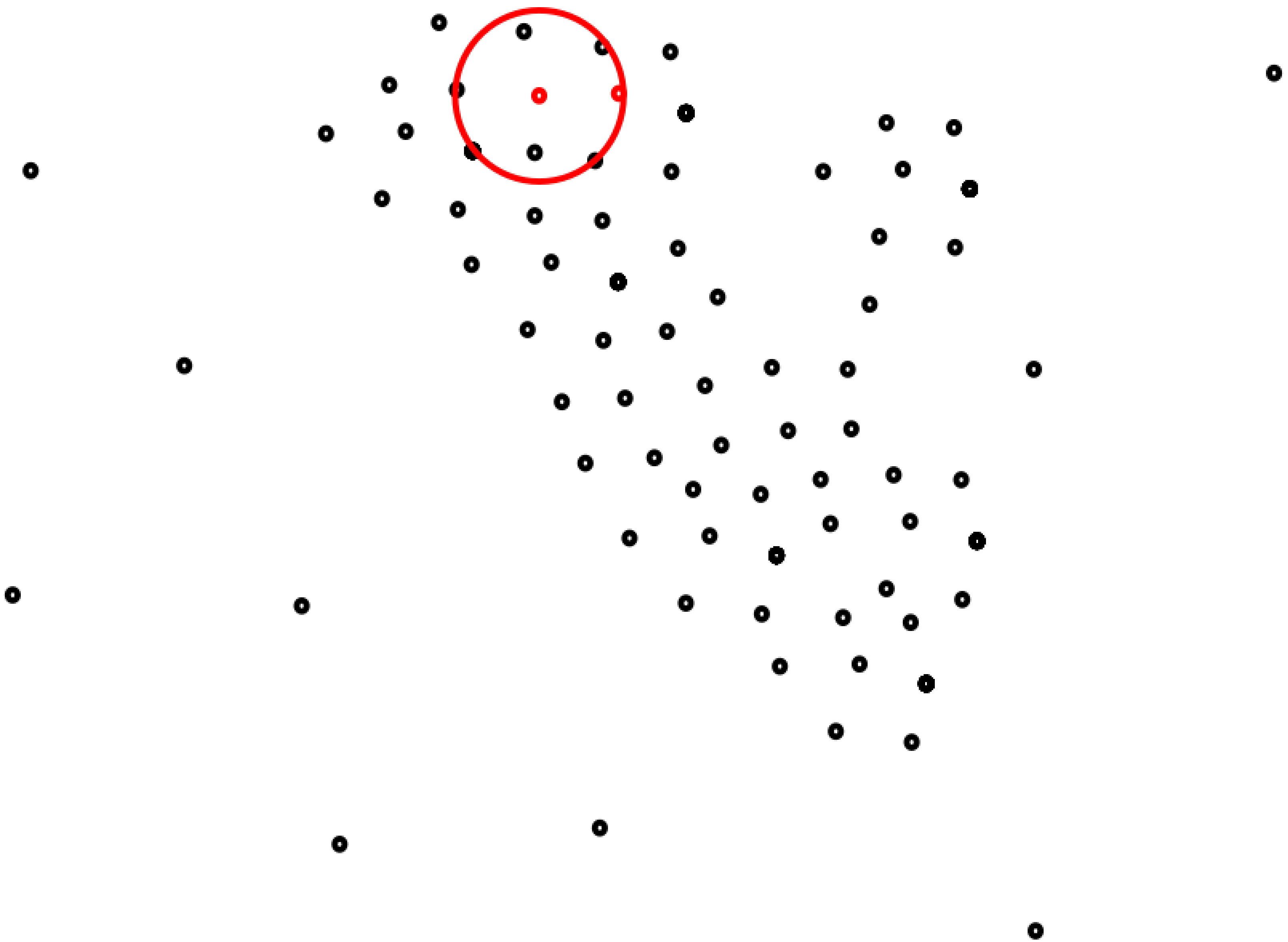
DBSCAN



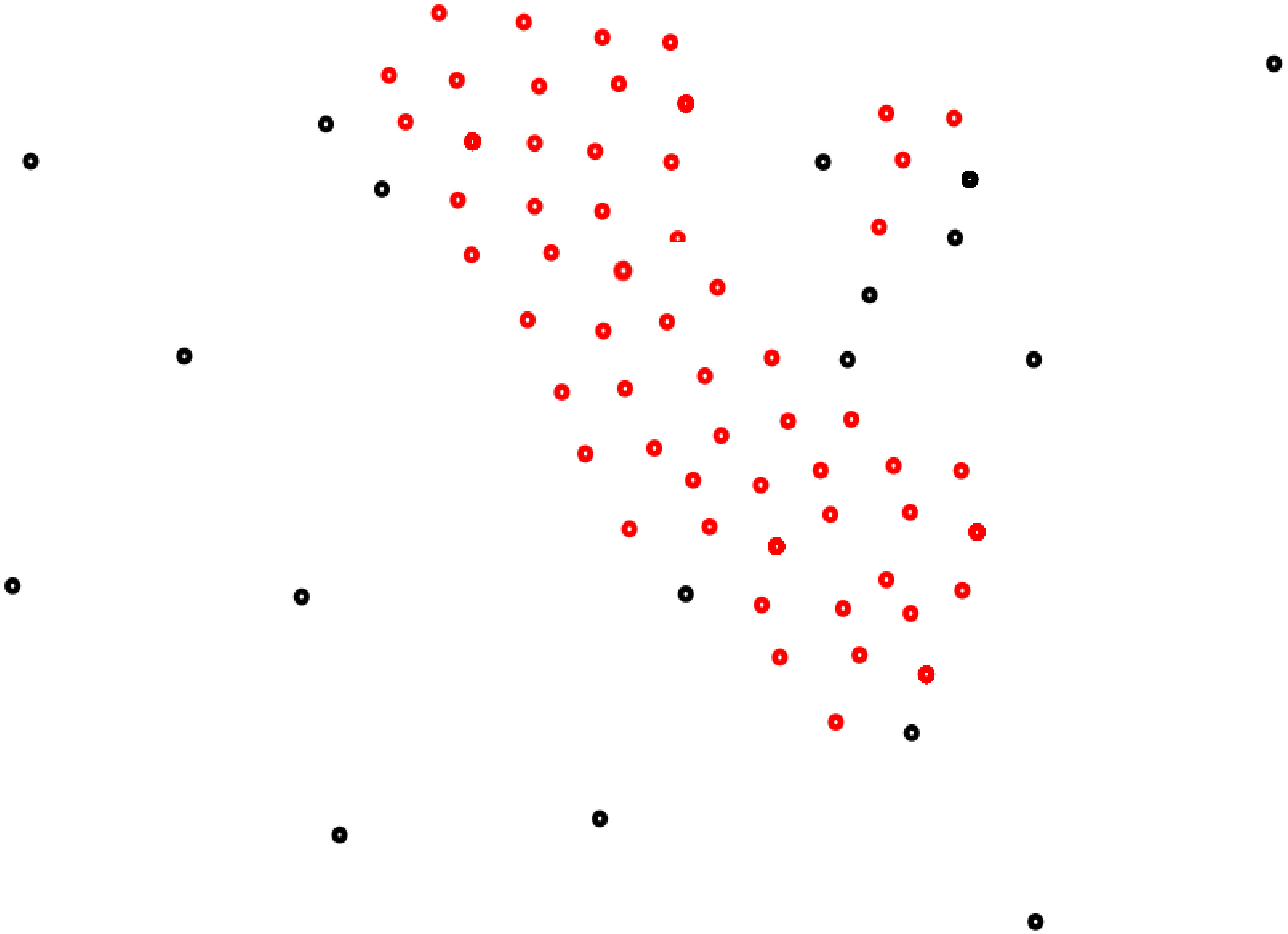
DBSCAN



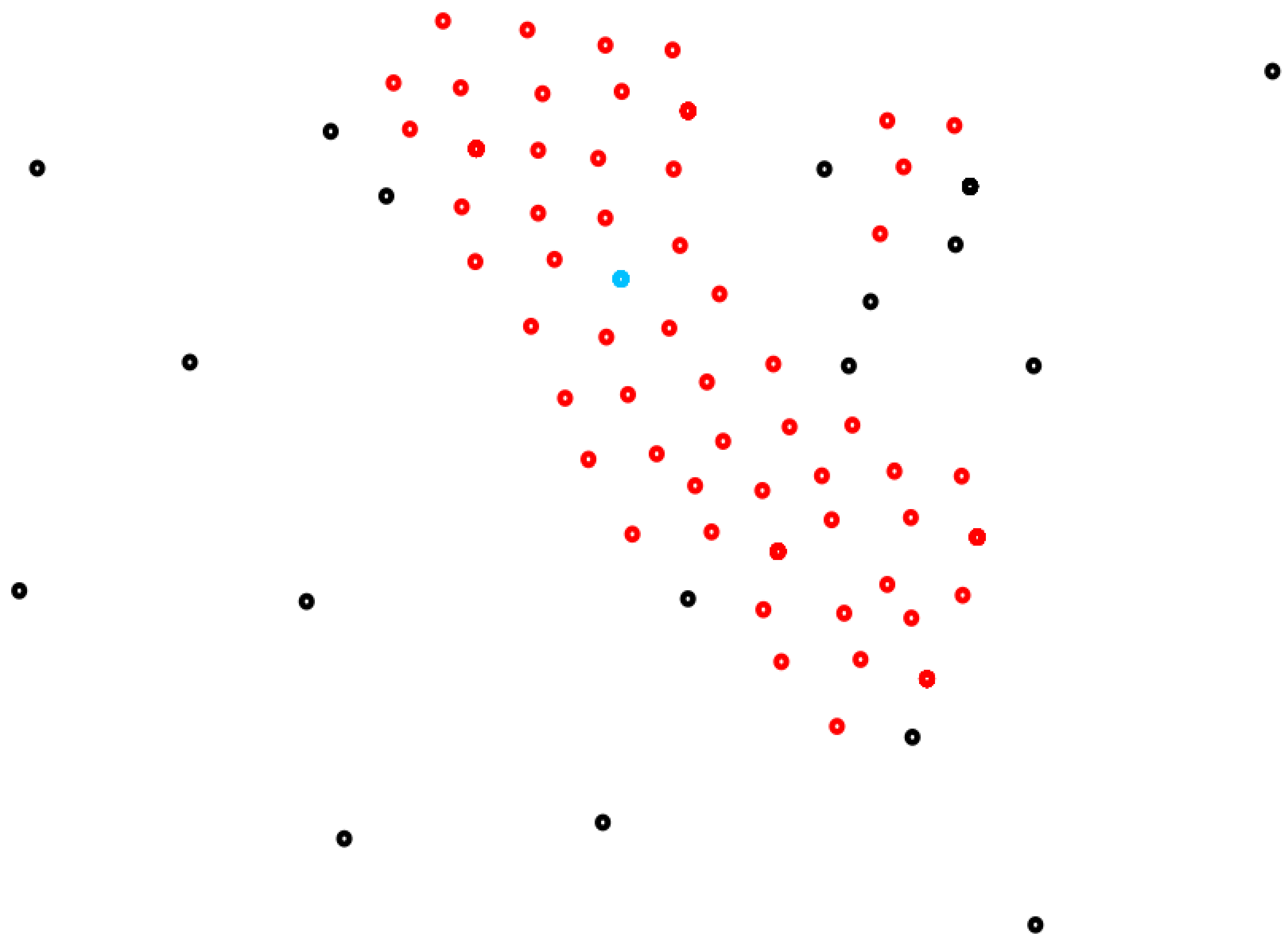
DBSCAN



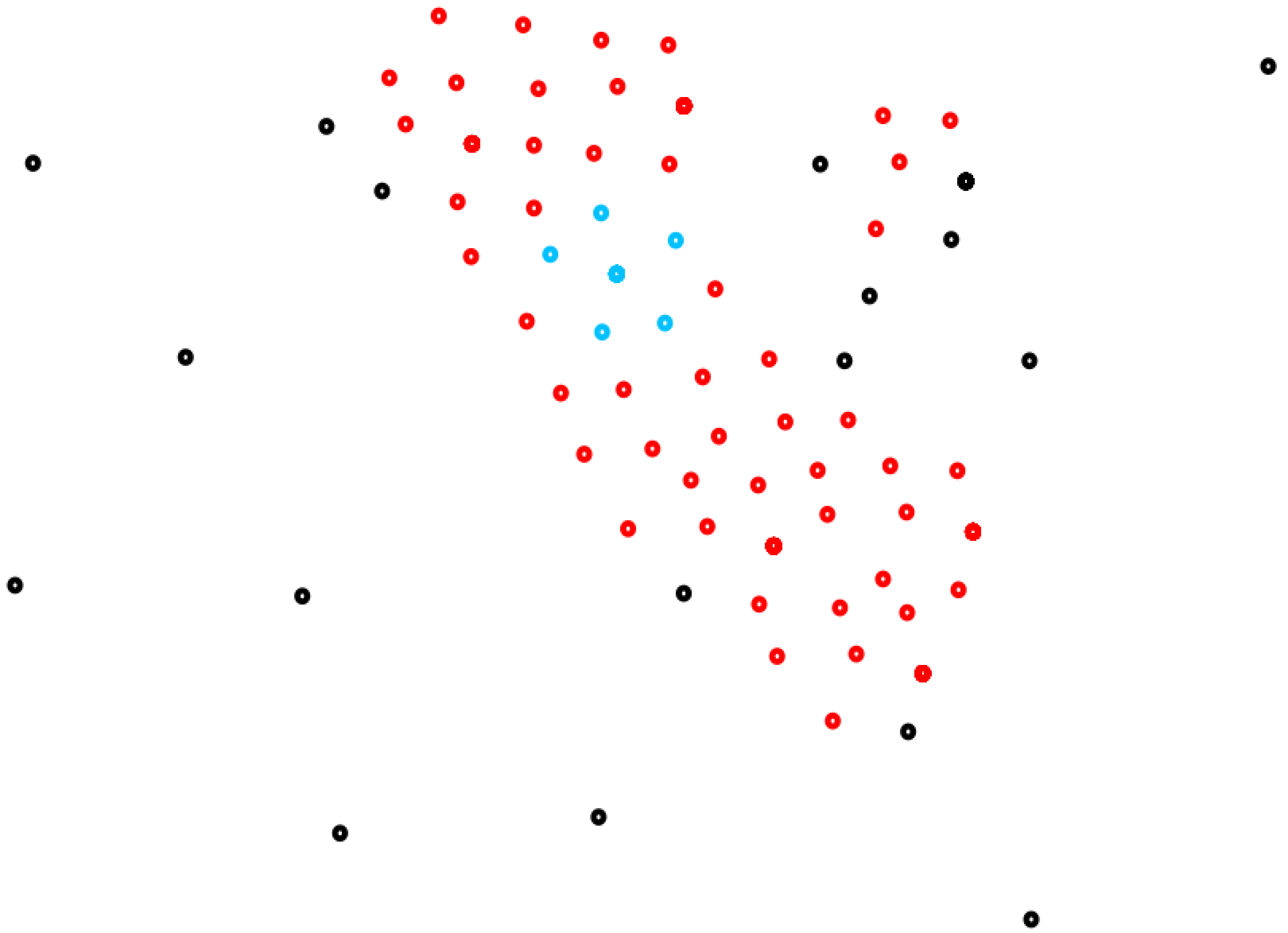
DBSCAN



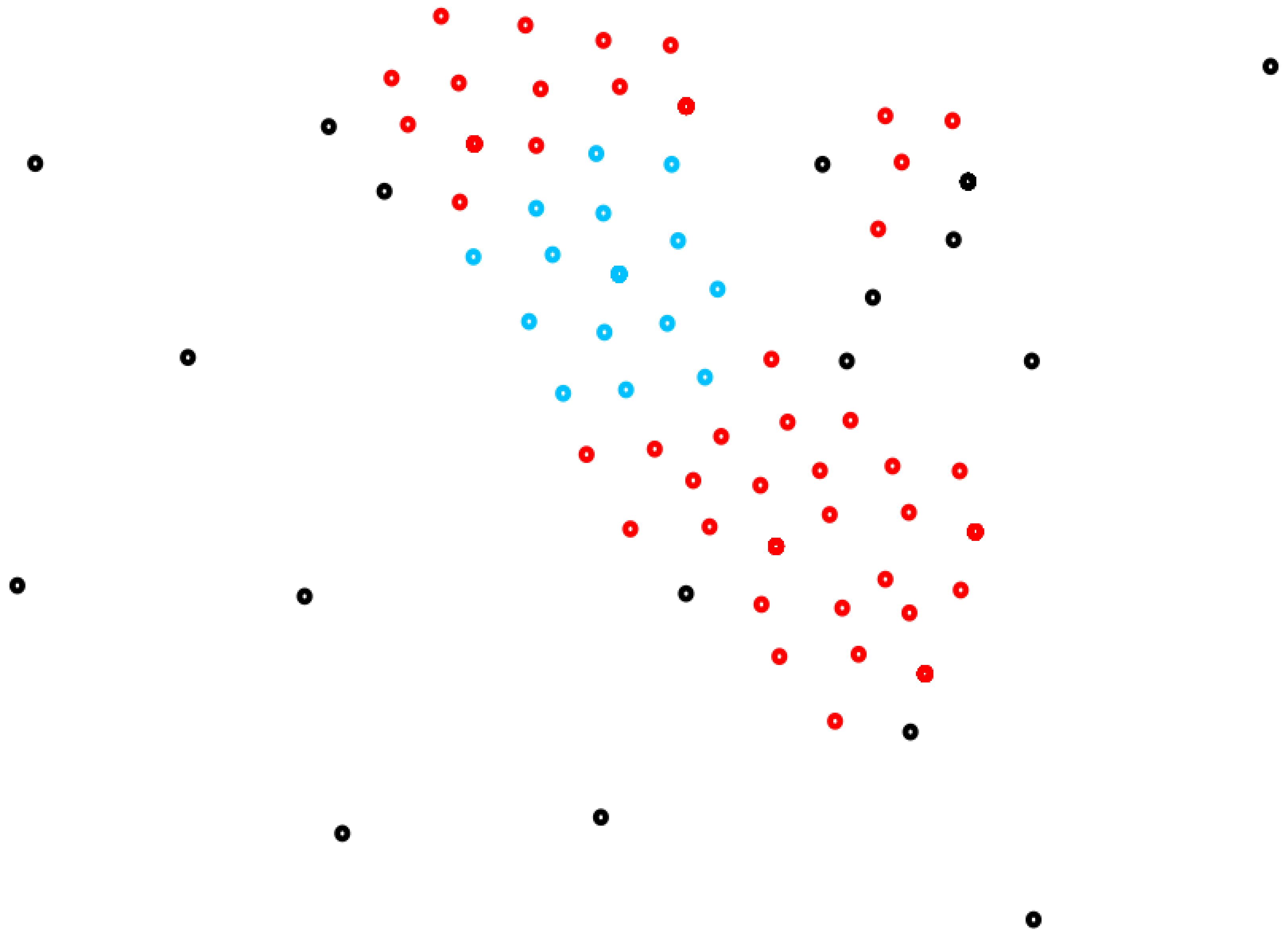
DBSCAN



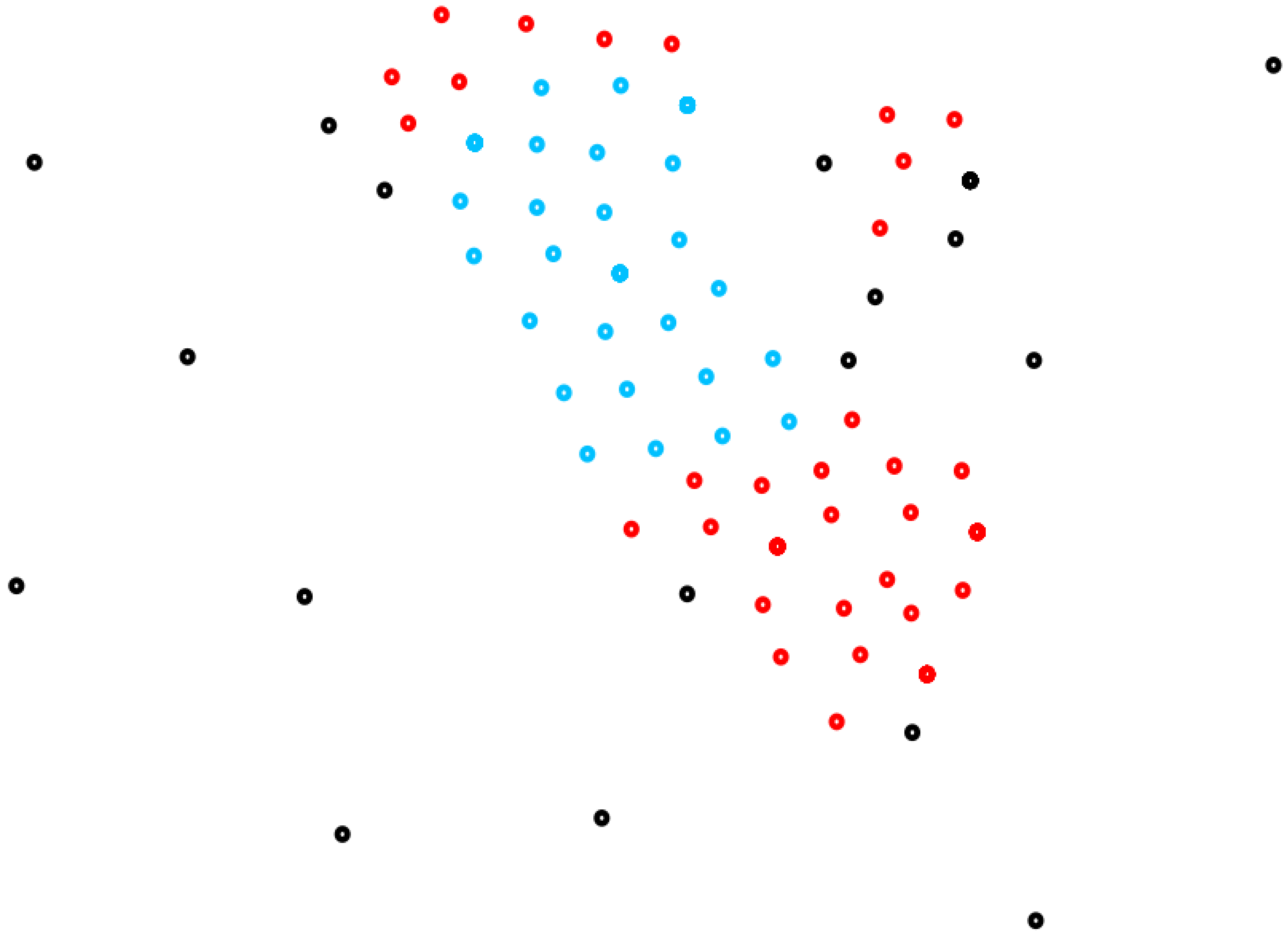
DBSCAN



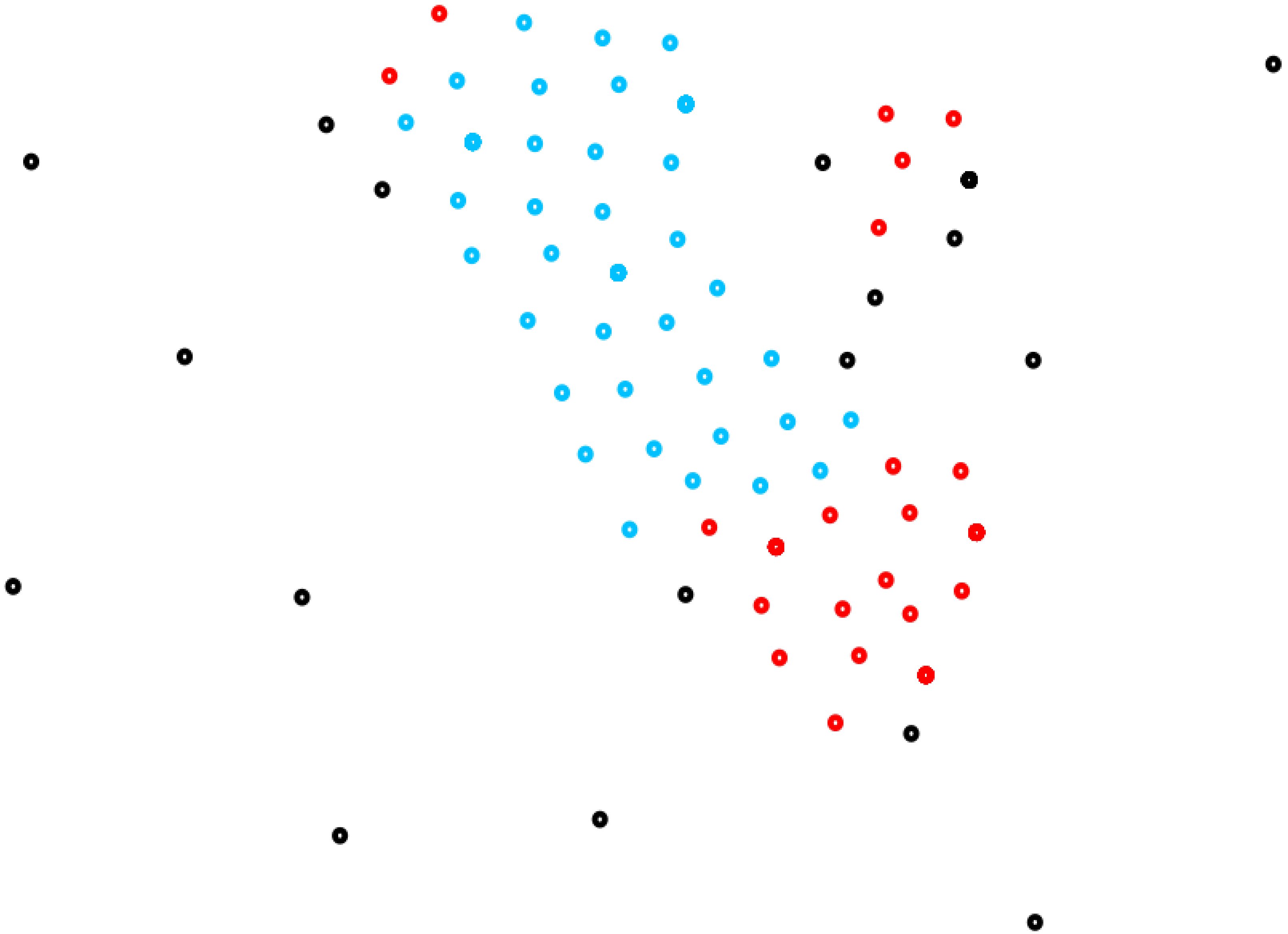
DBSCAN



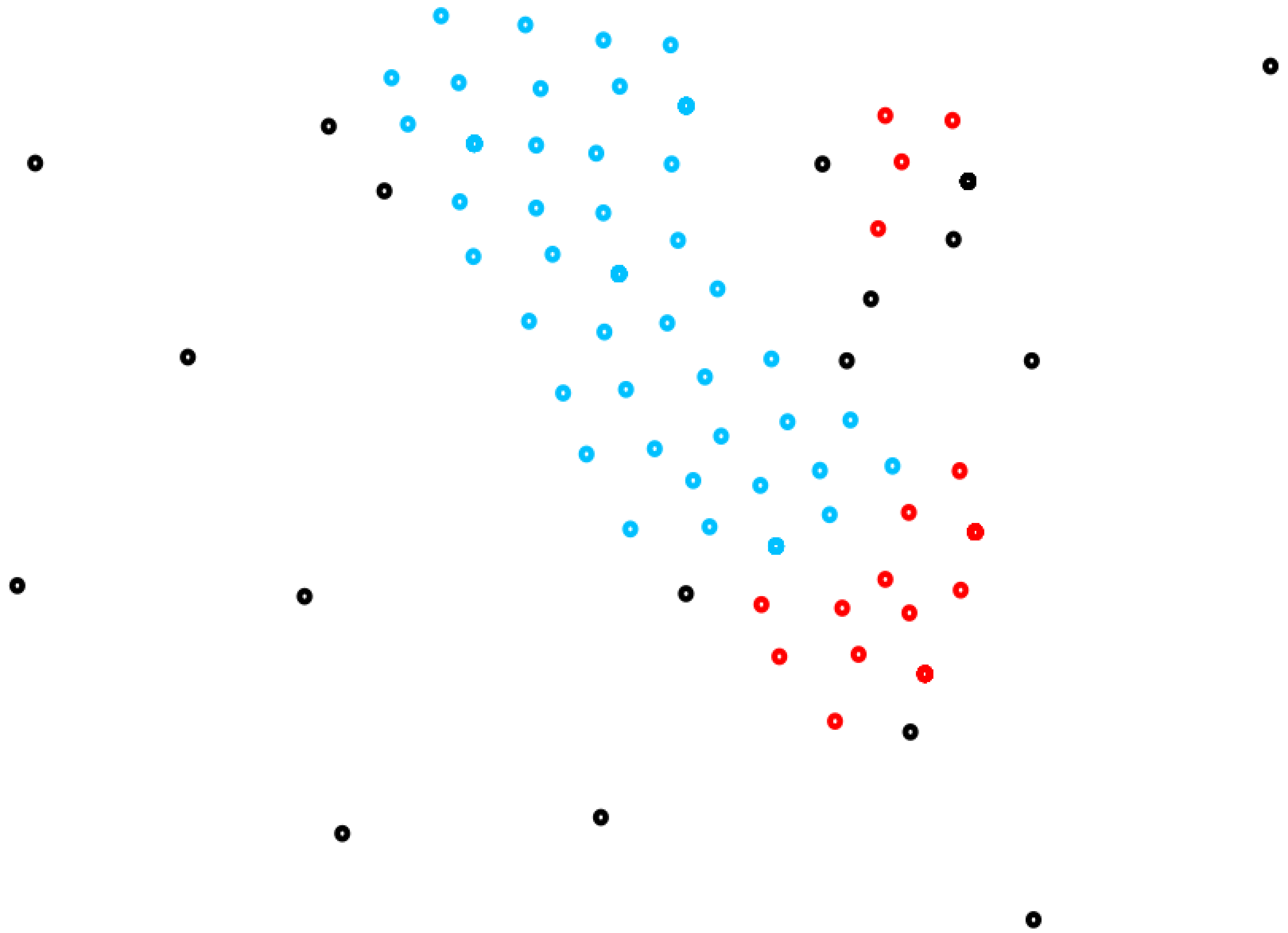
DBSCAN



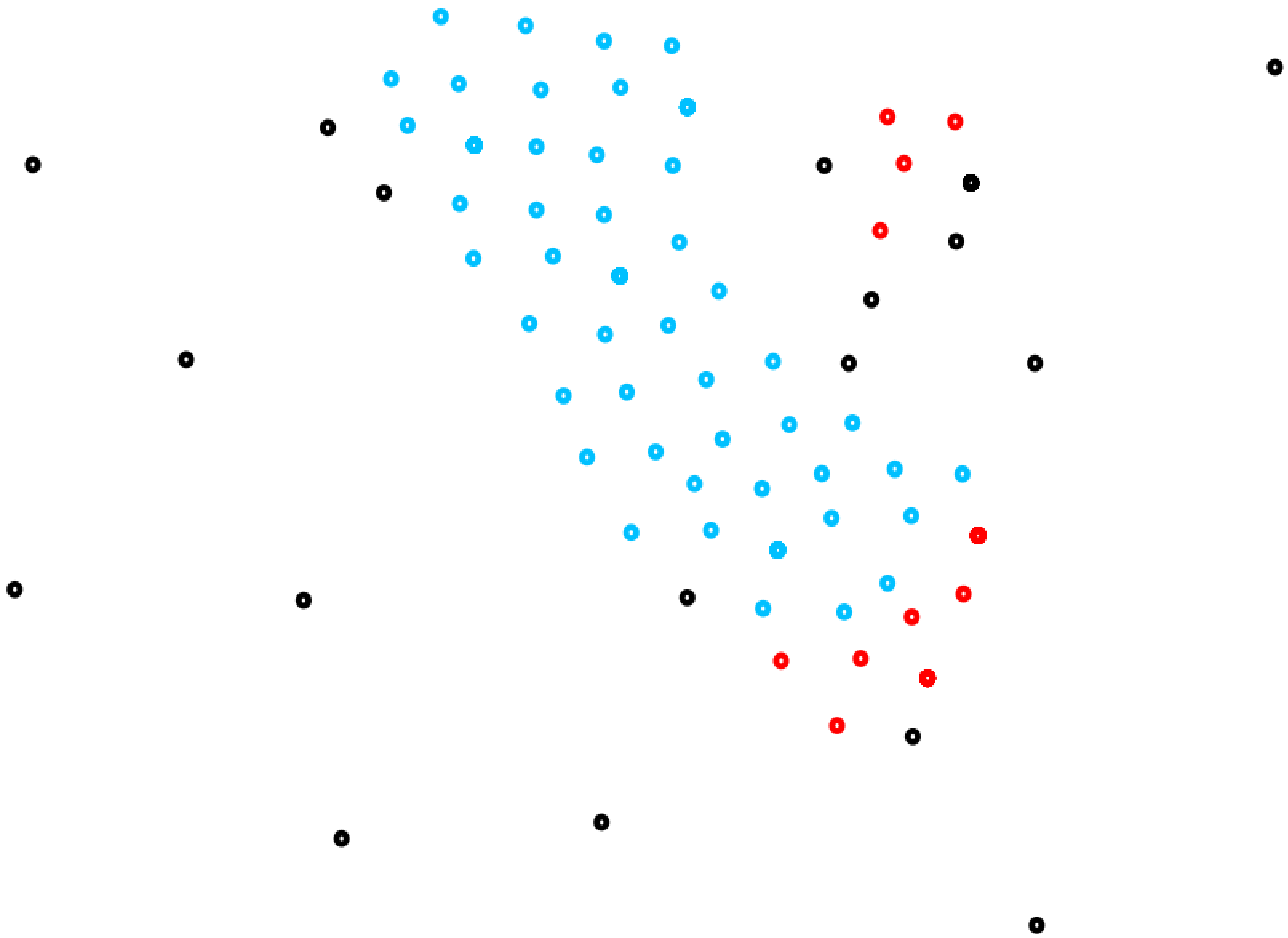
DBSCAN



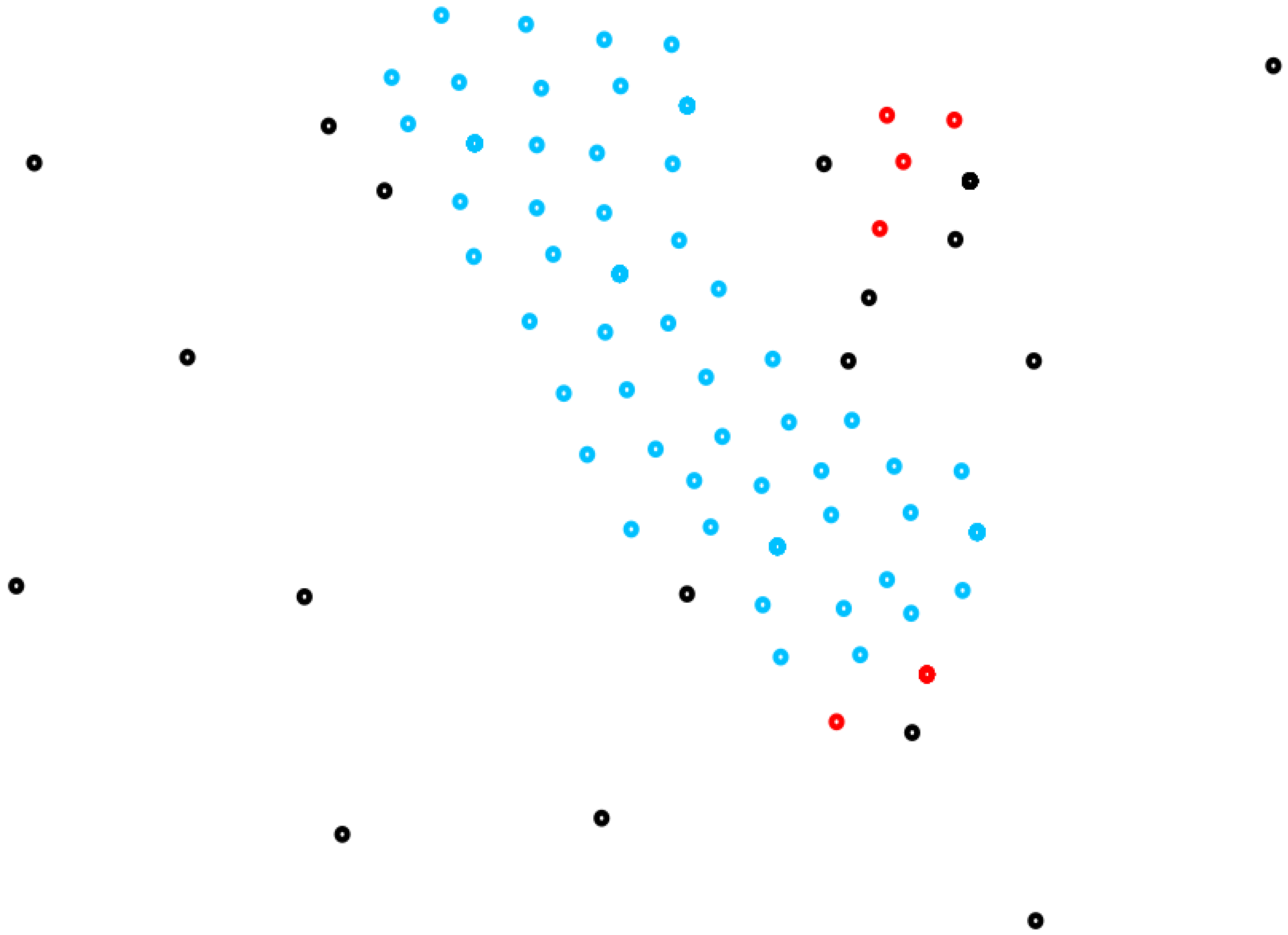
DBSCAN



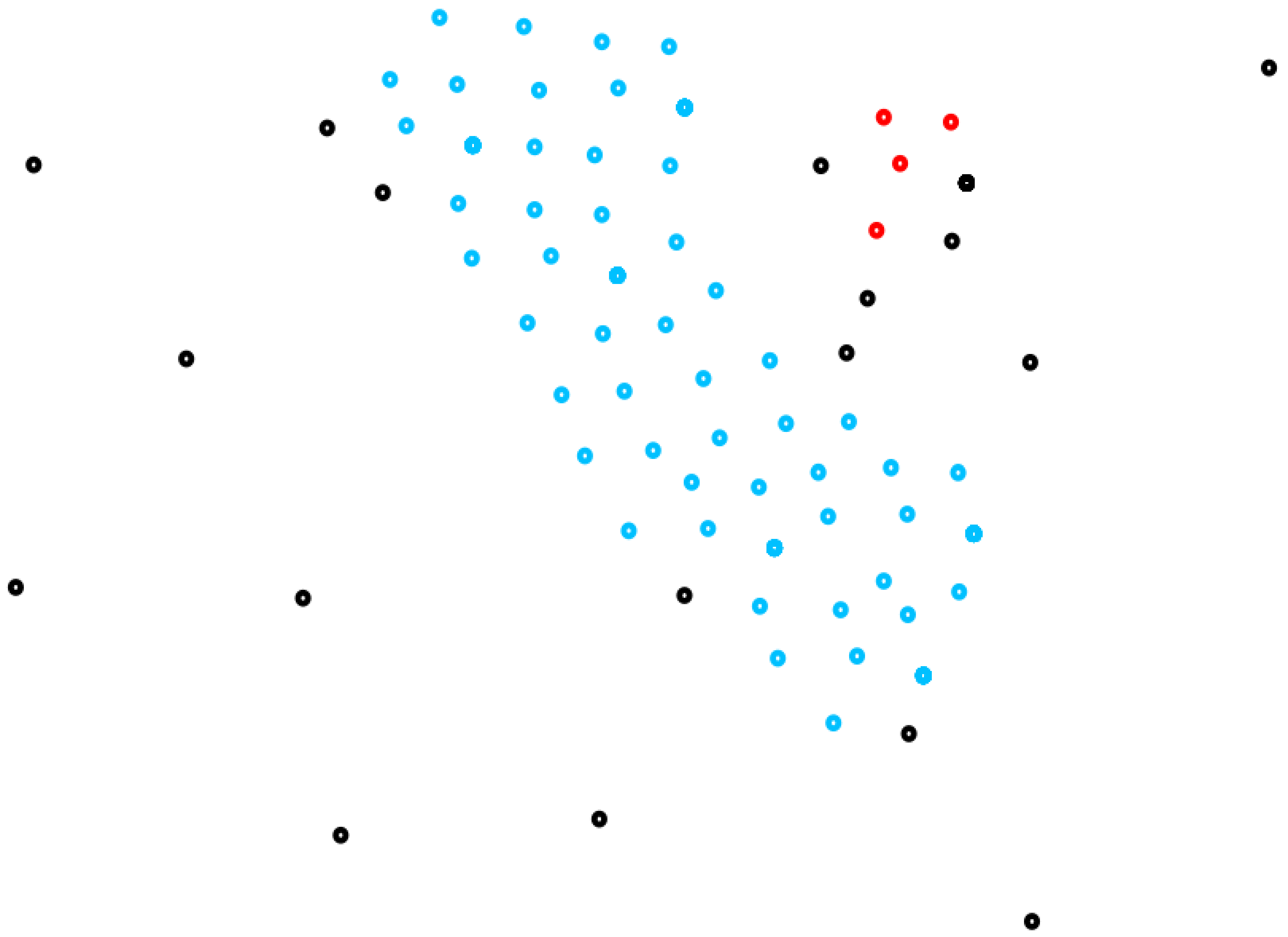
DBSCAN



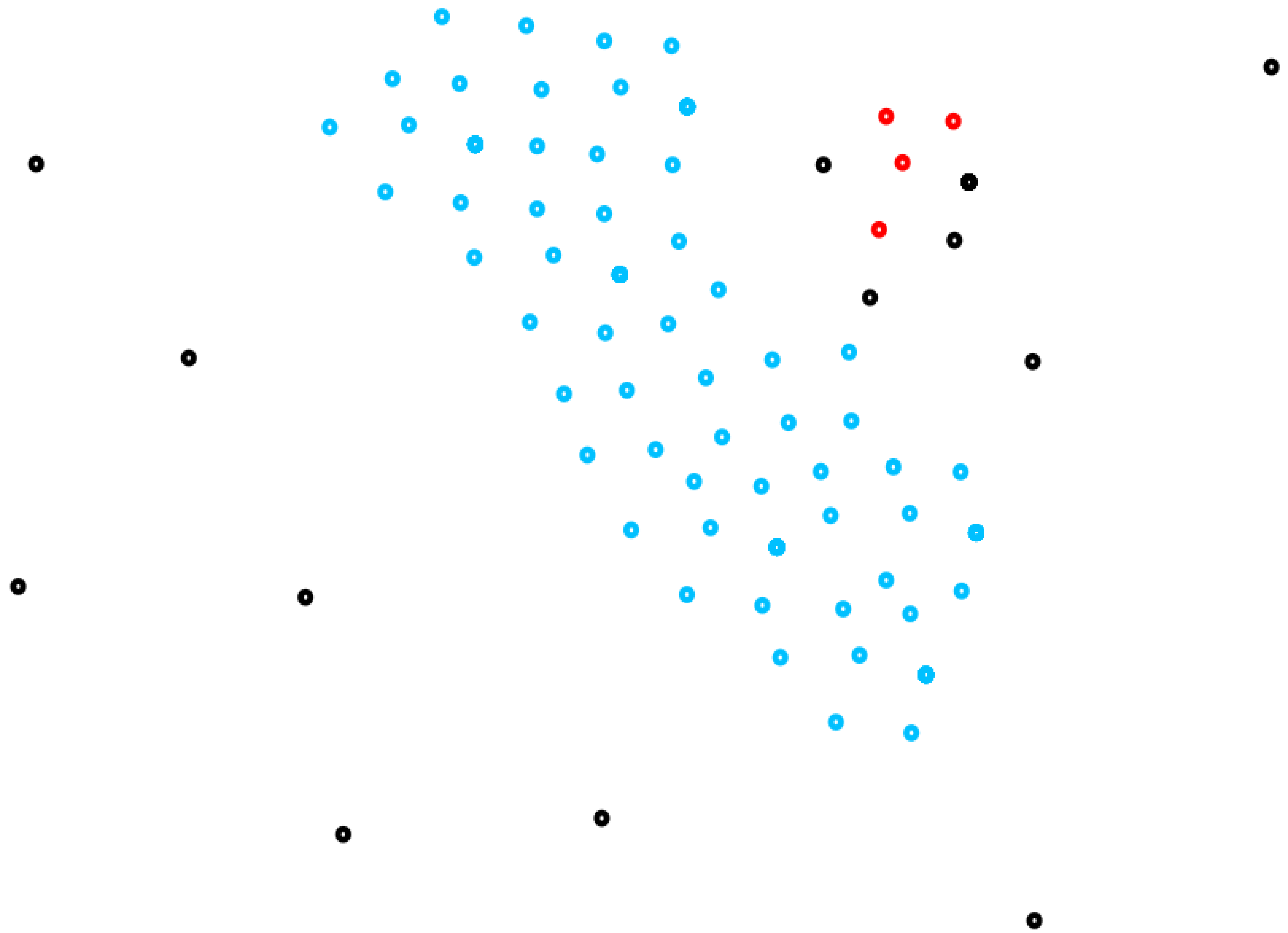
DBSCAN



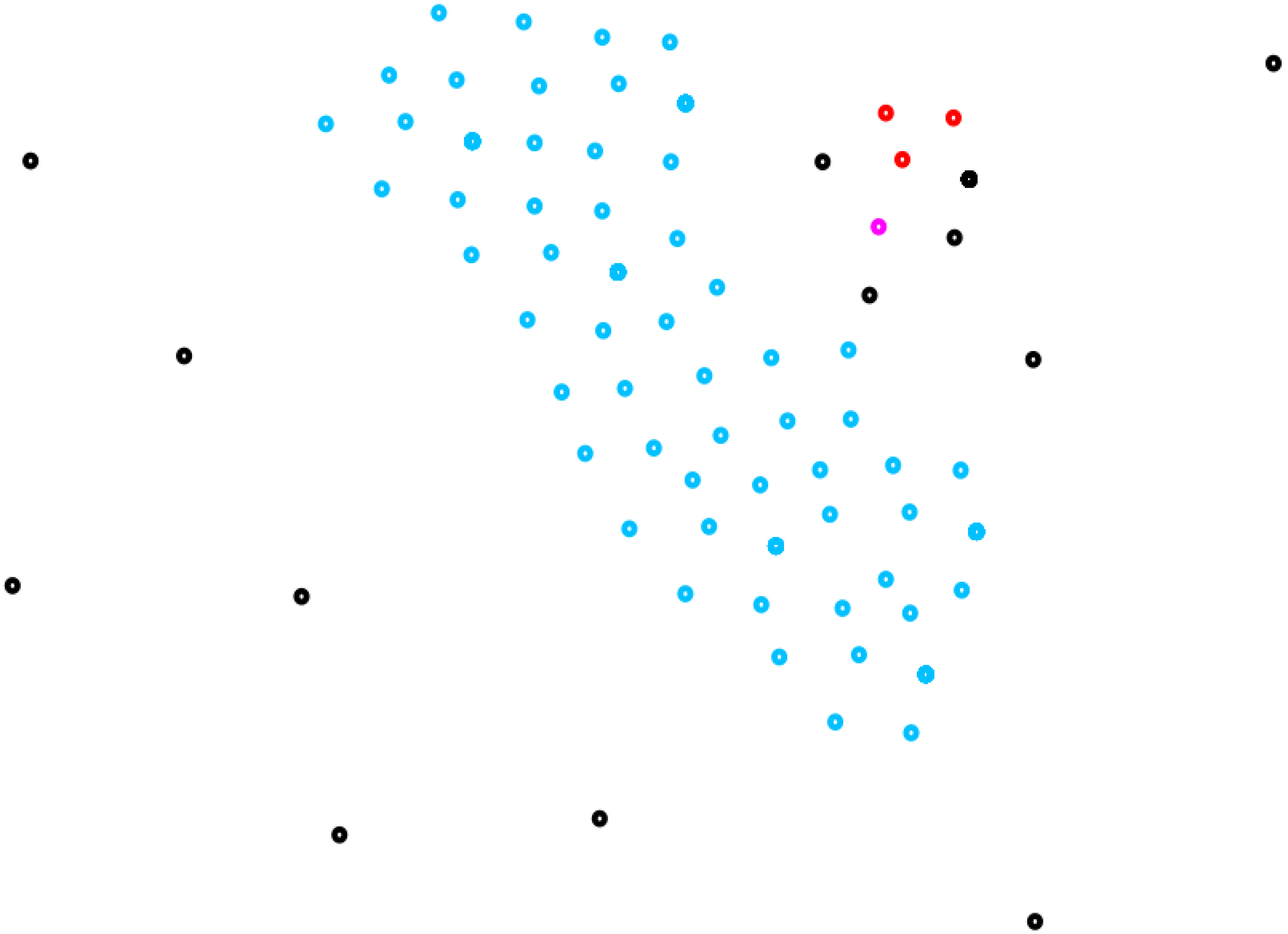
DBSCAN



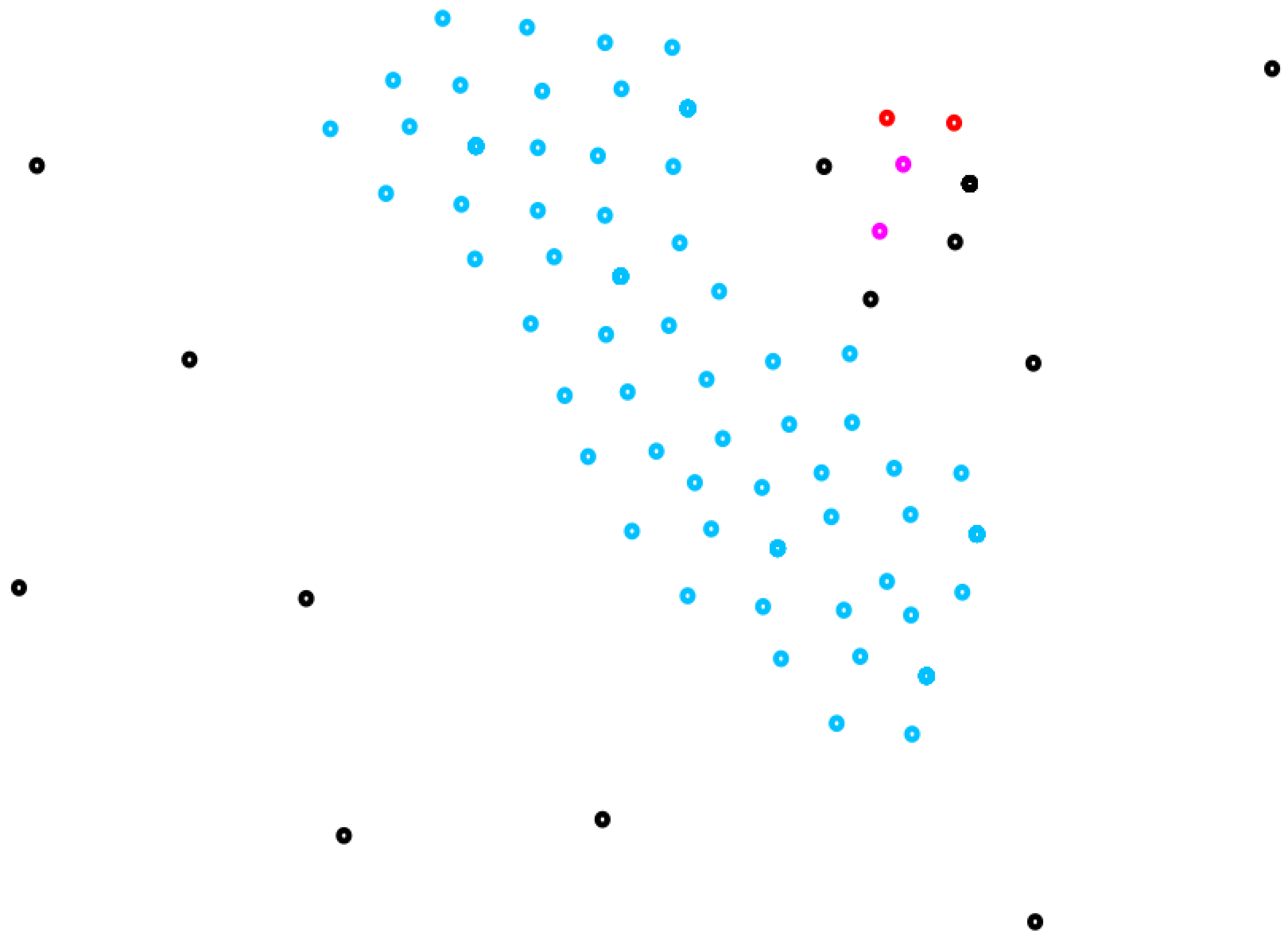
DBSCAN



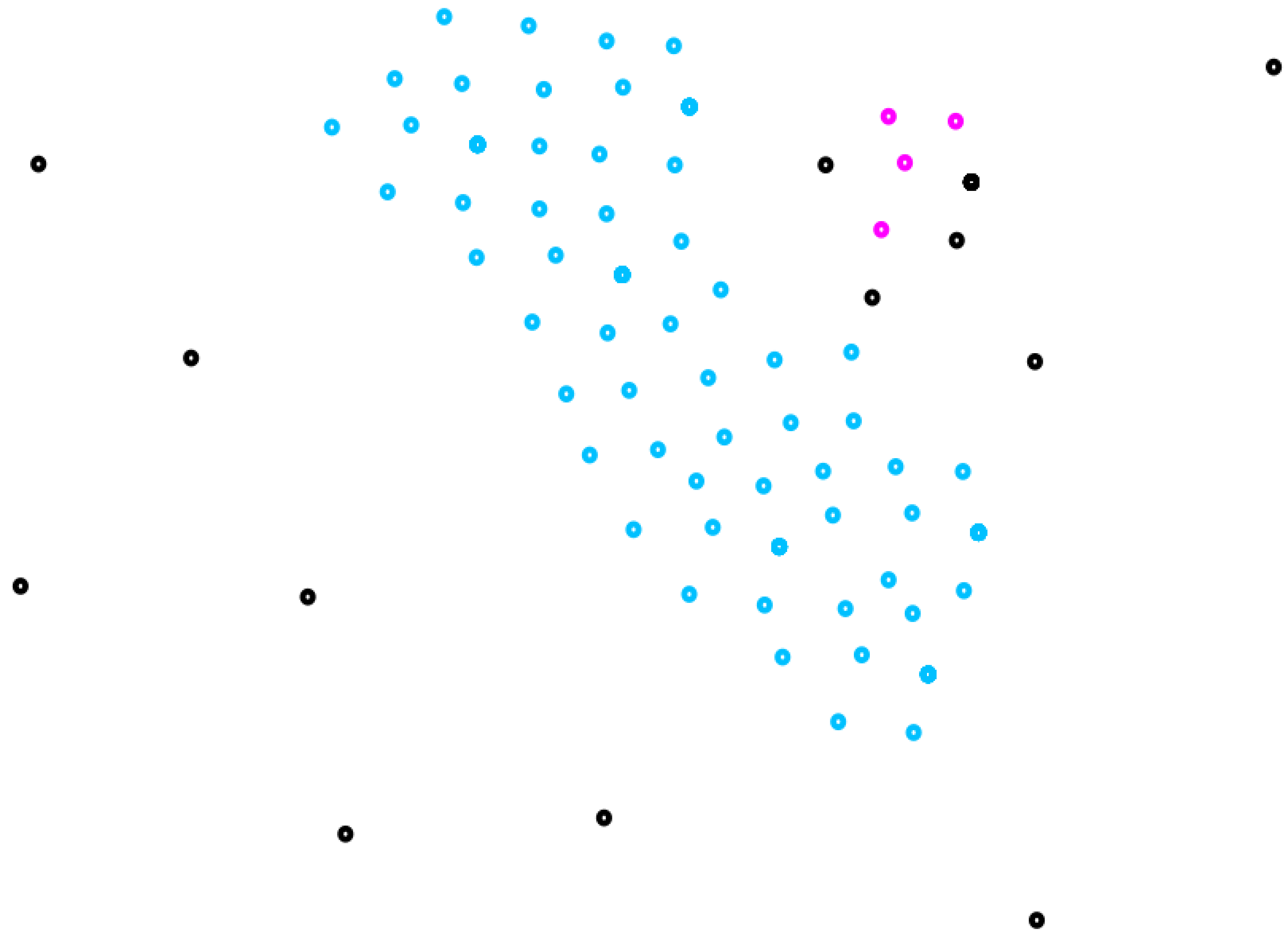
DBSCAN



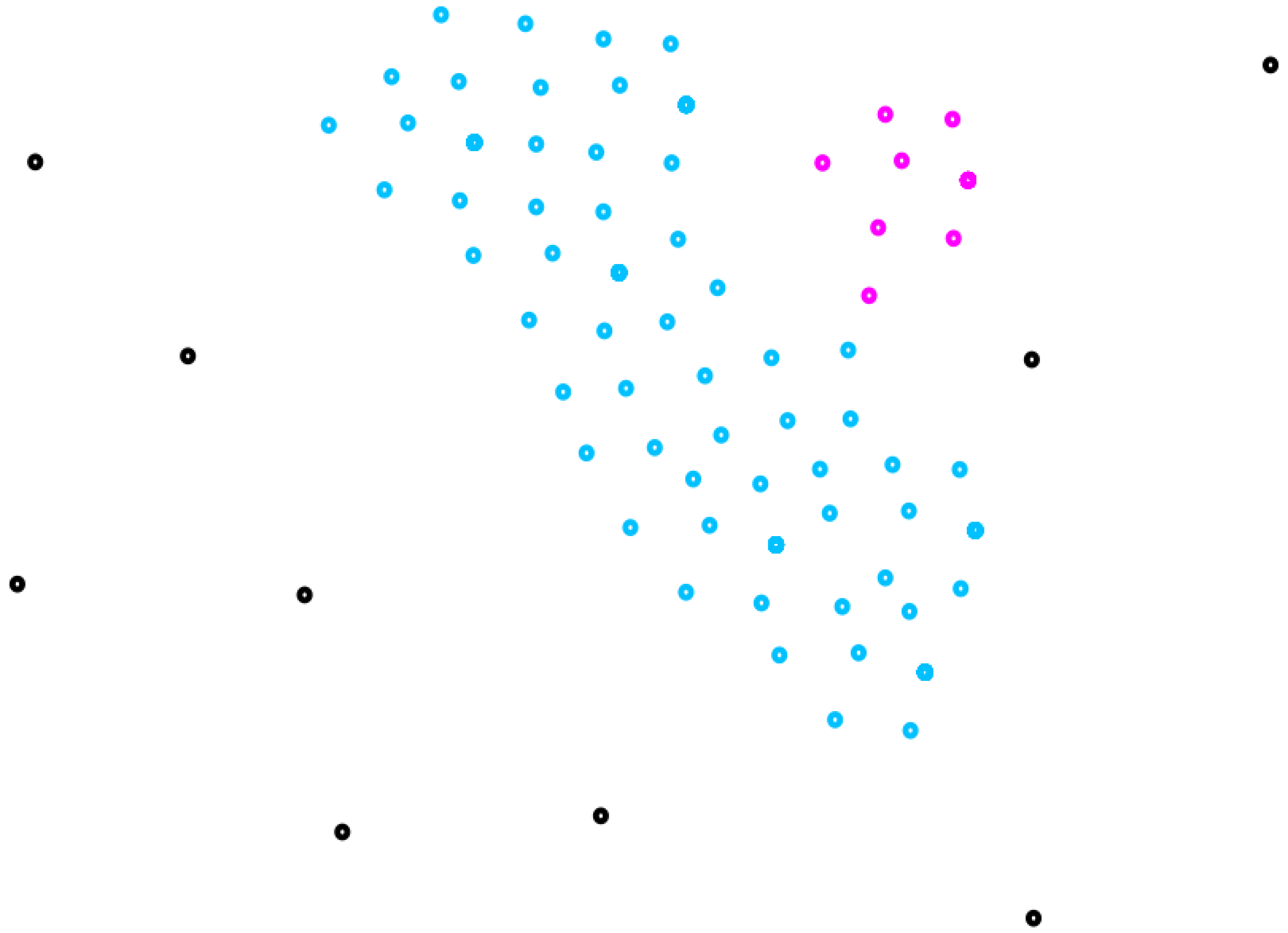
DBSCAN



DBSCAN



DBSCAN



THANK YOU