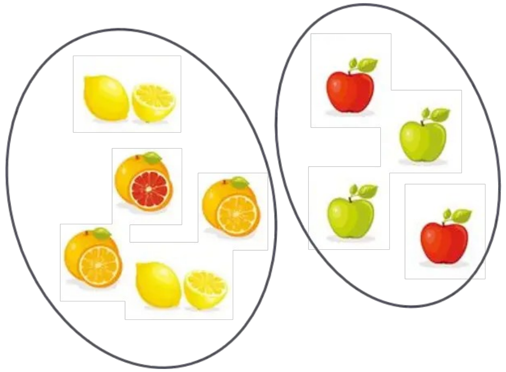
Unsupervised Learning

Unsupervised learning

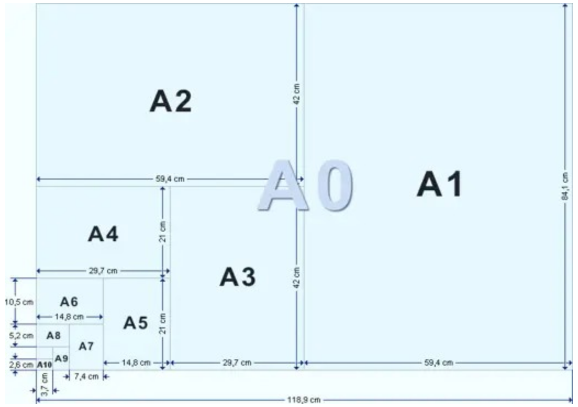
- Labeled data (data whose classification is known) are sometimes not available
- Sometimes not even the classes and their characteristic features are known
- Clusters are data groups in which the points have small distances/high similarity, where the different clusters have a large distance/low similarity

Unsupervised learning

- Raw data processing
- No desired response vectors
- Finding similarities
- Distinguish groups
- Cluster analysis
- Dimensionality reduction



Cluster analysis

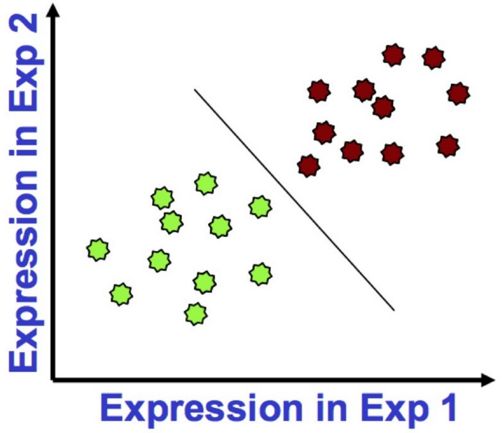


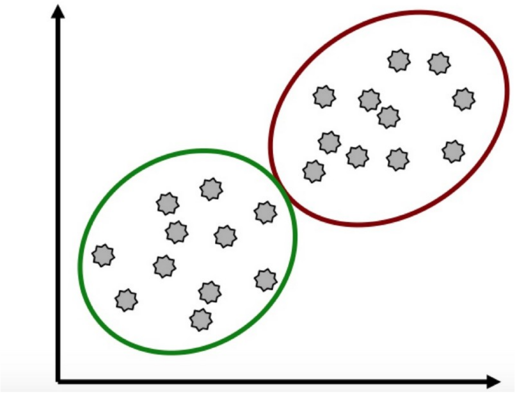
Dimensionality reduction

Clustering

Clustering vs. classification

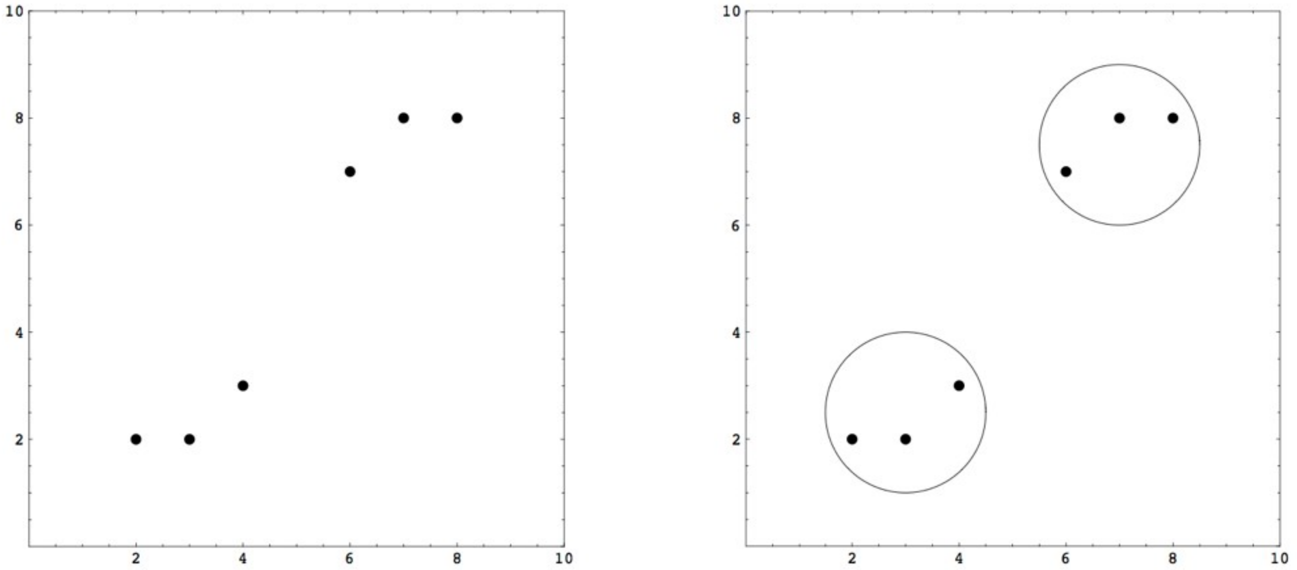
- Objects are characterized by one or more features
- Classification
  - Points have labels
  - Want a “rule” that will accurately assign labels to new points
  - Supervised learning
- Clustering
  - No labels
  - Group points into clusters based on how “near” they are to one another
  - Identify structure in the data
  - Unsupervised learning





What is a cluster?

- A group of data in which the elements are very similar/close to each other, while the data in different clusters are significantly different/distant from each other.



Reasons for the usage

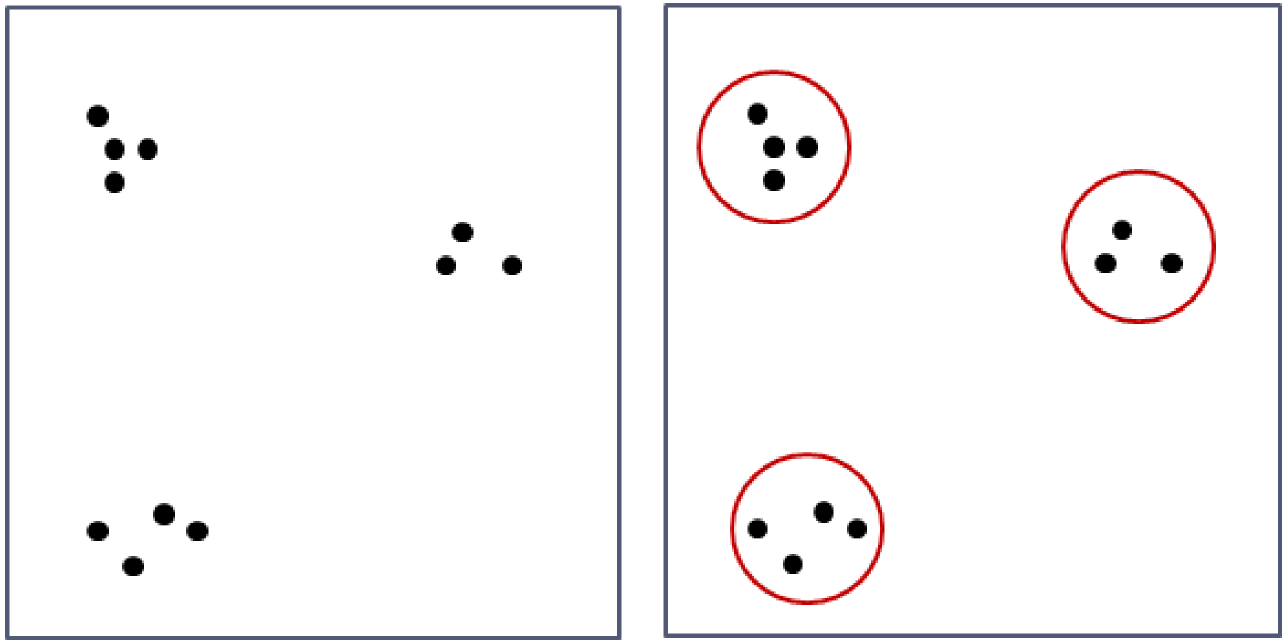
- Labeled (classified) data are not always available
- Sometimes even classes or characteristic properties are not known
- Reducing the number of data

Few application

- Signal and image decomposition
- Data analysis
- Market analysis (identification of target customer groups)
- Content management (grouping similar documents)
- Fuzzy rule extraction
- …

A simple example

- We want to build the rule base from a dataset
- The data set must first be clustered
- Simple example:



Basic requirements

- We have *n* data samples in *p*-dimensional space
- So, all the data are:

The goal is to define a subset system where:

Agglomerative clustering

- We start with the most rigorous clustering, i.e. *n* single-element clusters
- Then we iteratively merge the two closest clusters
- We continue until we have only one cluster with all the data elements

- We do not create a partition, but a sequence of partitions
- We do not need to specify the number of clusters in advance
- Most of these methods are slow and only work on small data sets

SAHN Clustering

Sequential Agglomerative Hierarchical Non-overlapping Clustering

1. Given: and *d* distance metrics
2. Initially:
3. For we look for the pair, for which is minimal.
4. Output:

SAHN distance metrics

- Minimum

- Maximum

- Average

Prototype based clustering

- Not all individual data are kept in clusters
- Rather, in each C_i cluster, we designate a point v_i in the data space, usually called the cluster center
- Each data point belongs to that cluster whose center point is the closest to the data
    - So: if

Partition matrix

- Which cluster each data point belongs to
-

- All data are elements of a cluster:

- There is no empty cluster:

k-means clustering

- Iterative process for clustering
    - Minimize the following function

- Calculating the centroids

(1)

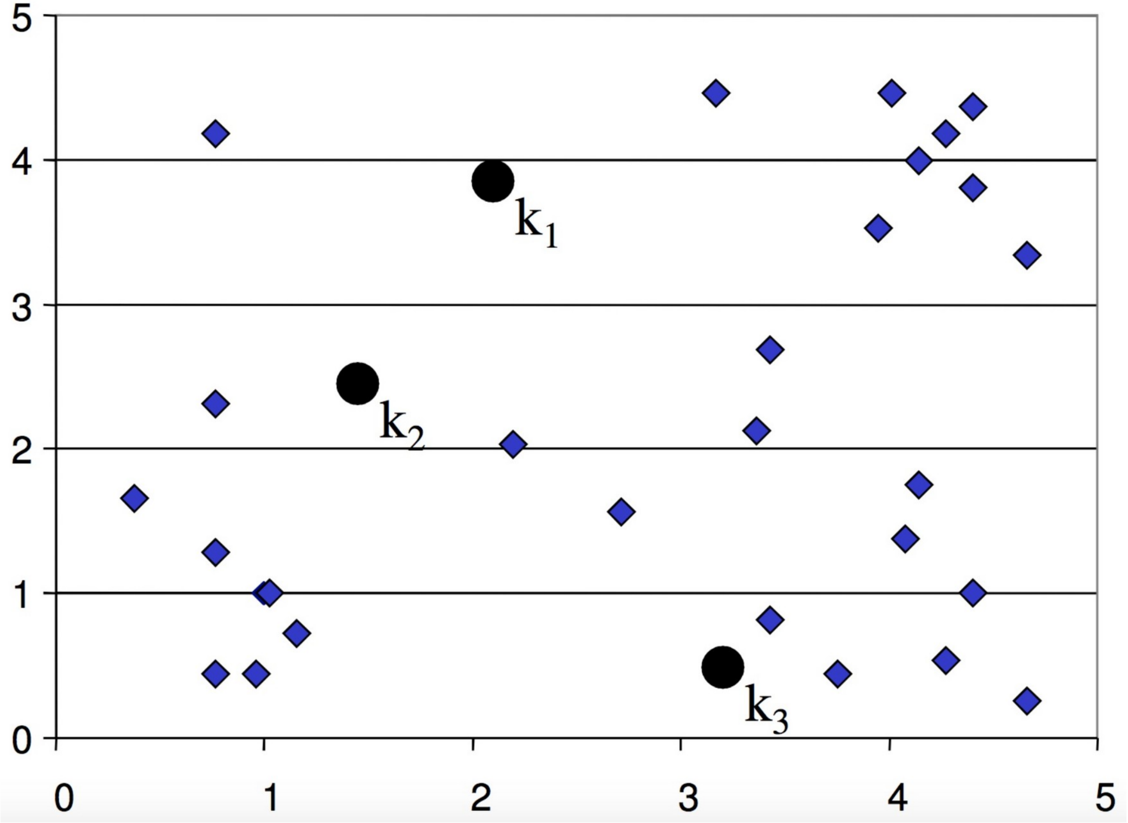
k-means clustering

1. Input: a data set , ‖.‖ is a norm defined on , K is the predefined number of clusters, t_max is the maximum number of iterations, is a distance measure, ε is the tolerance
2. Initialization:
3. For
    1. Determine the partition matrix
    2. Determine the cluster centroids (from equation (1))
    3. If ,end
4. Result:

   The partition matrix U and the cluster centers V

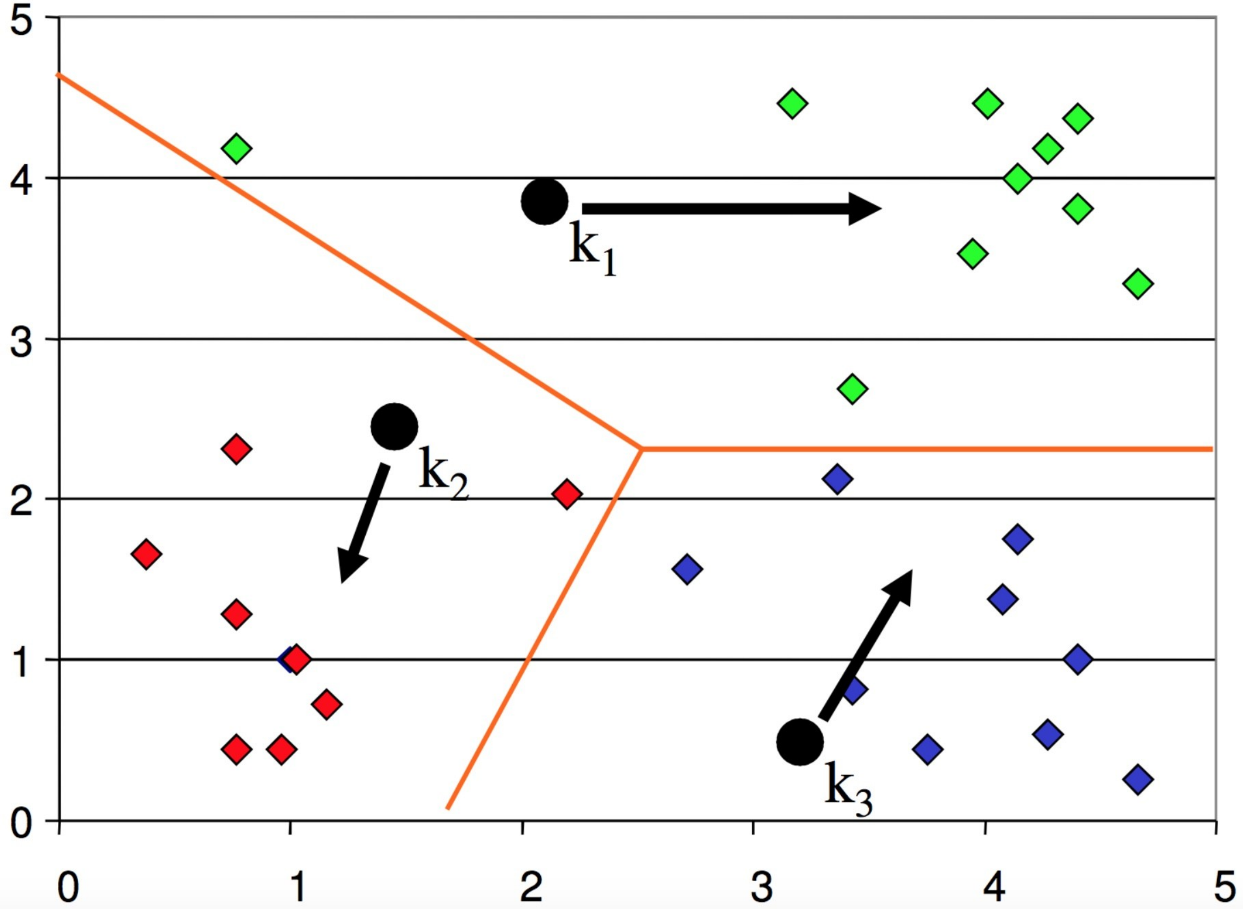
k-means clustering

- Initialization:
  - decide K
  - initialize K cluster centers randomly



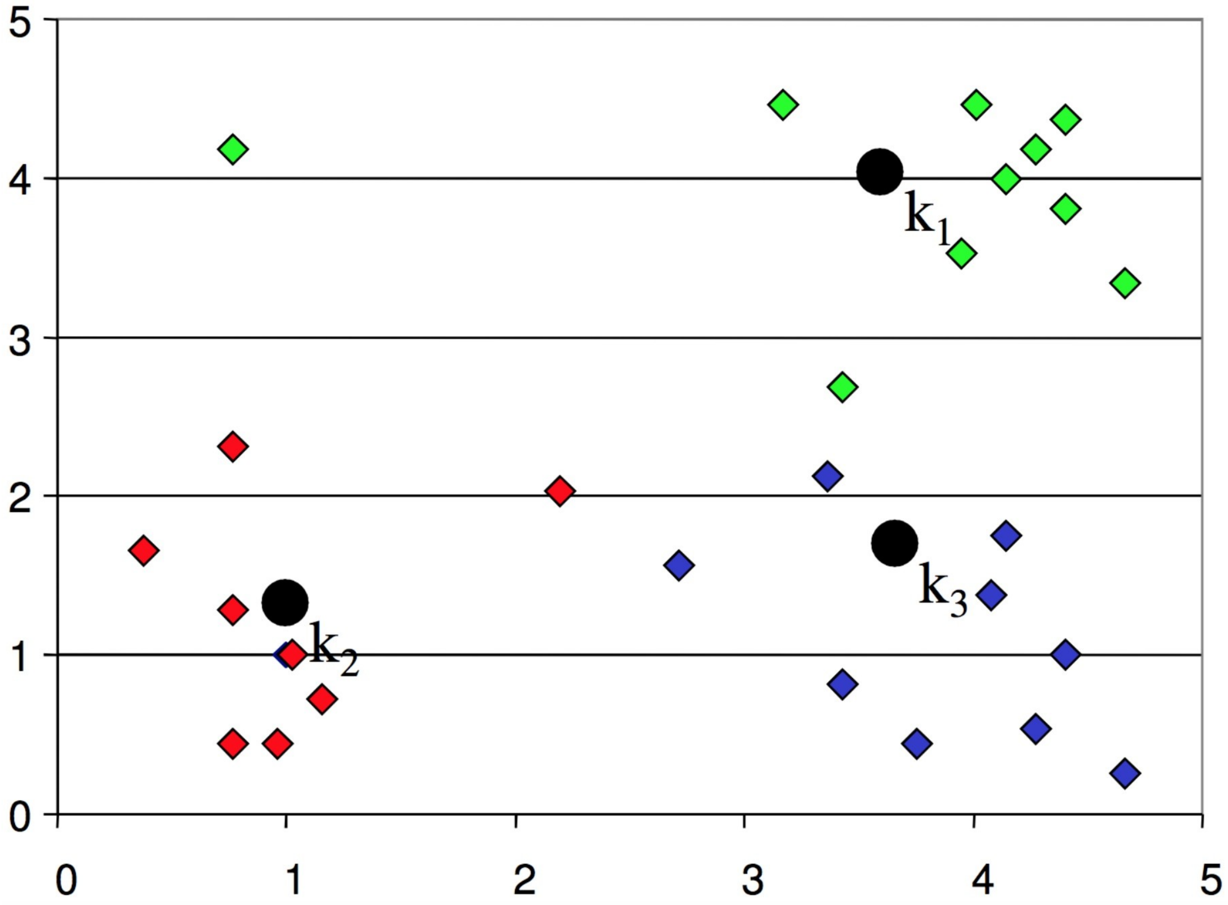
k-means clustering

- Iteration:
  - assign all points to the nearest cluster center
  - move the centers to the mean of their members



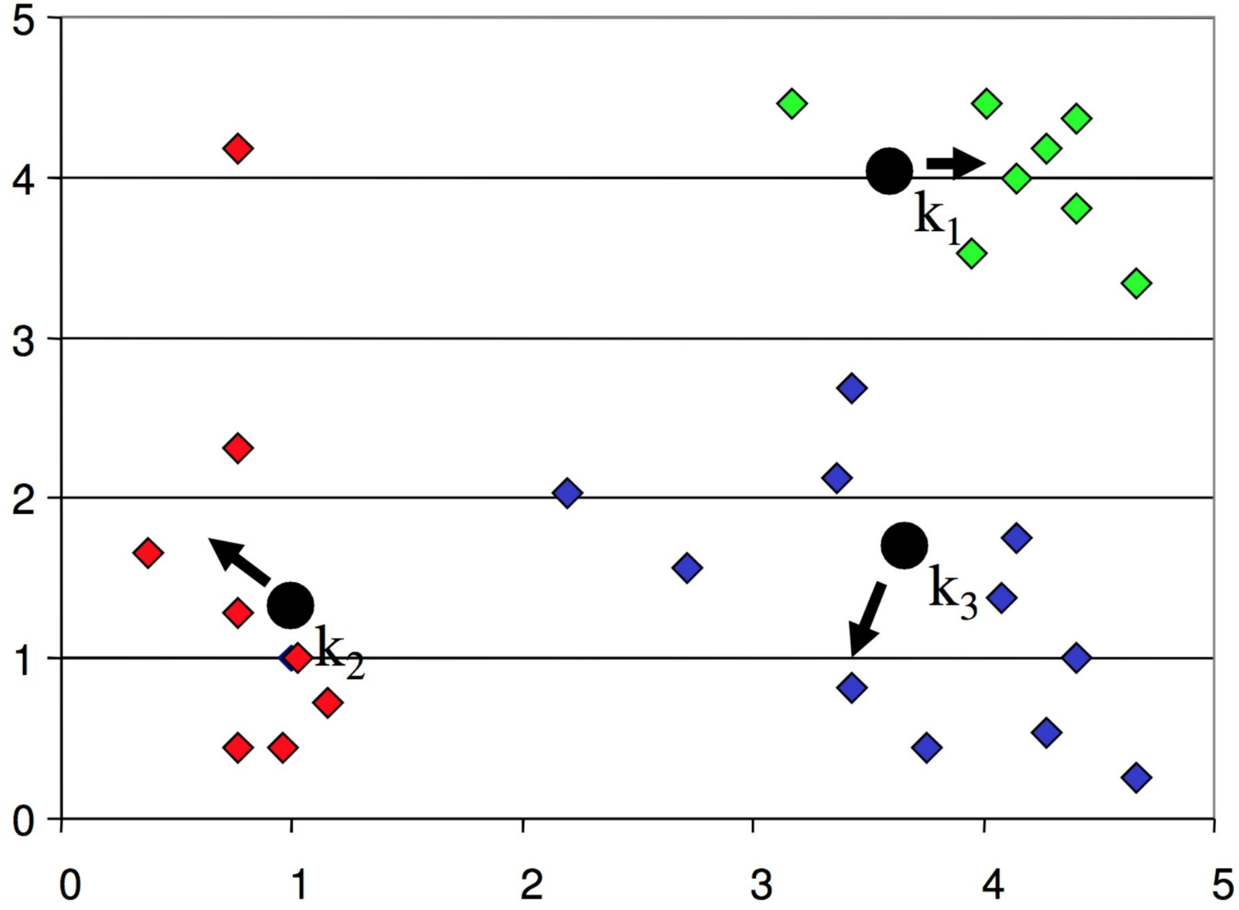
k-means clustering

- Iteration:
    - after moving centers, re-assign the points to the nearest center



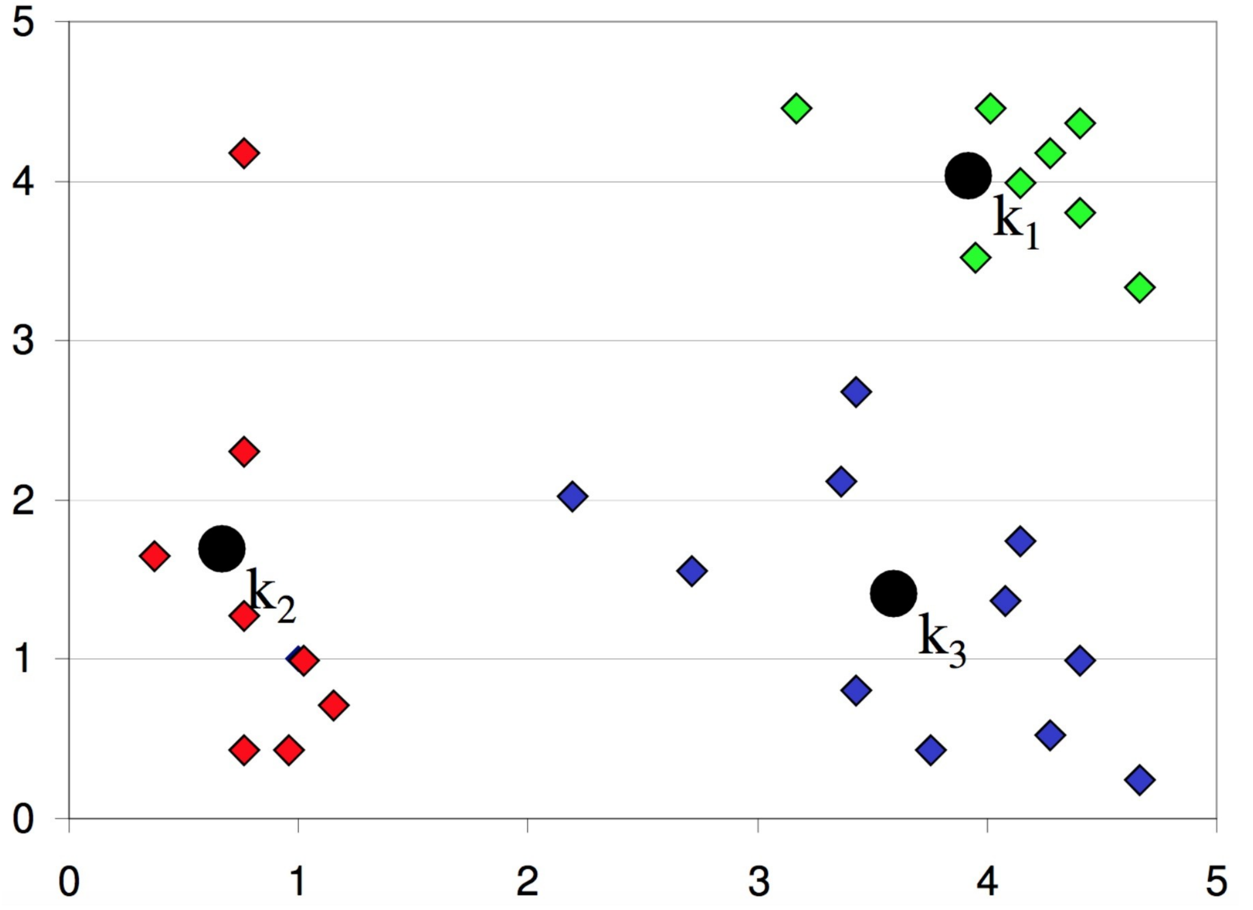
k-means clustering

- Iteration:
  - move the centers to the mean of their members



k-means clustering

- Finished:
  - re-assign and move centers, until no points changed membership



k-means clustering

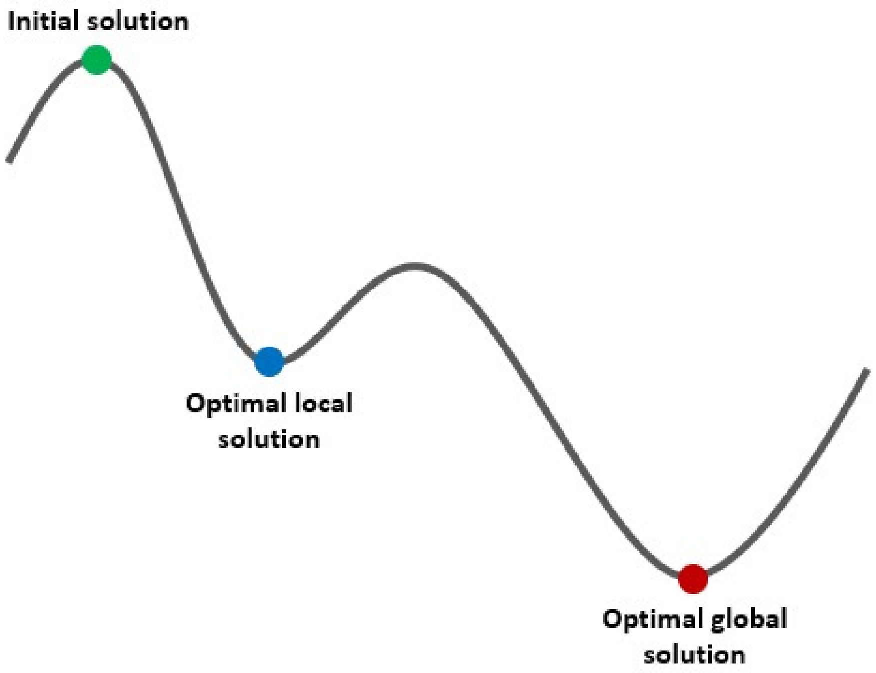
- 1. Decide the value of *K*, the number of clusters
- 2. Initialize the *K* cluster centers randomly
- 3. Decide the class membership of the *N* points by assigning them to the nearest cluster center
- 4. Re-estimate the *K* cluster centers, by assuming the memberships found above are correct
- 5. Repeat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

k-means clustering

- **Strength:**
  - Simple, easy to implement
  - Intuitive objective function: optimizes intra-cluster similarity
  - Relatively efficient: $O(t \cdot k \cdot n)$, where n: number of points (data), k: number of clusters, t: number of iterations, usually: $k,t<<n$
- **Weakness:**
  - Applicable only when *mean* is defined
  - Often terminates at a local optimum. Initialization is important
  - Need to specify K, the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes

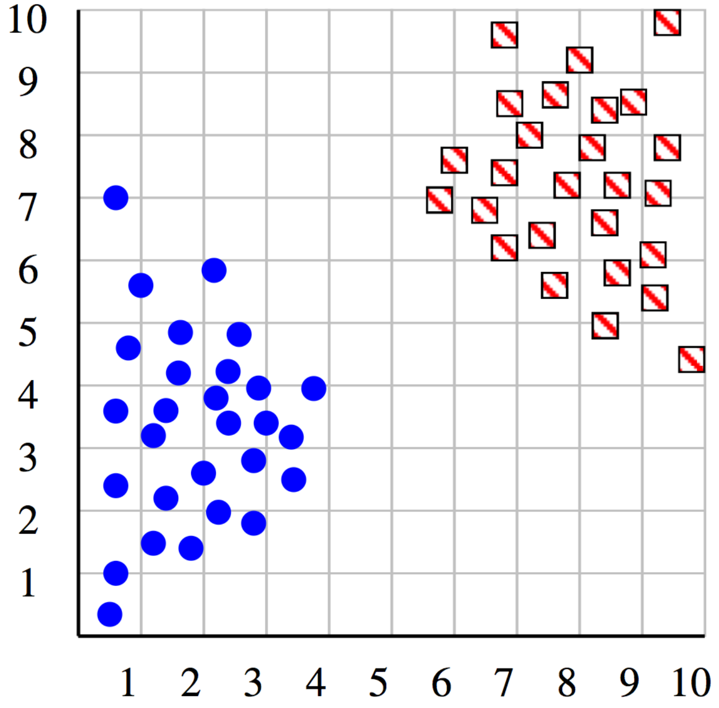
k-means clustering

Initialization of centroids can influence the final clustering result



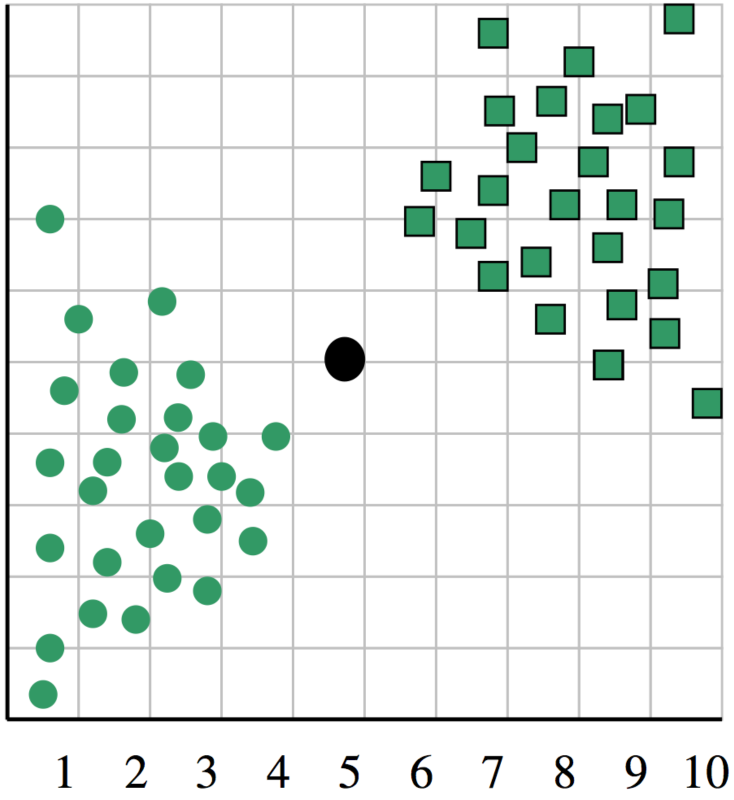
How to define the right number of clusters?

In general, this is a unsolved problem. However there are many approximate methods. In the next few slides we will see an example.



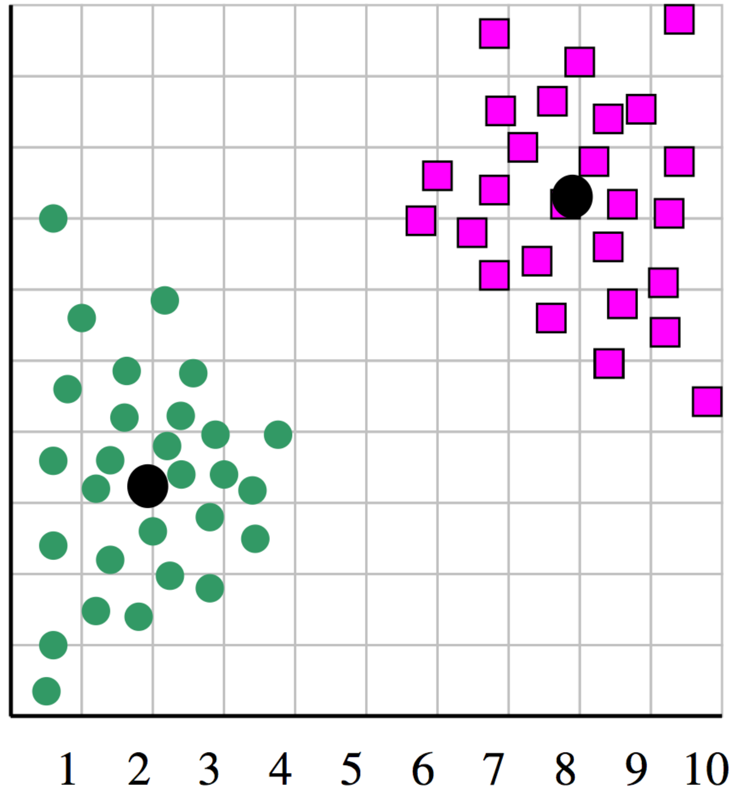
How to define the right number of clusters?

When k = 1, the objective function is 873.0



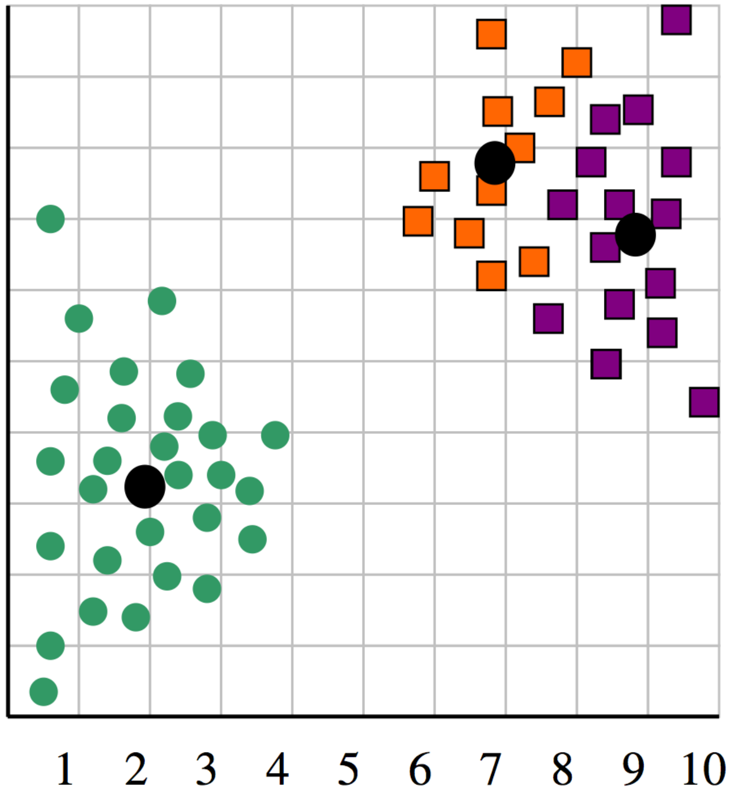
How to define the right number of clusters?

When k = 2, the objective function is 173.1



How to define the right number of clusters?

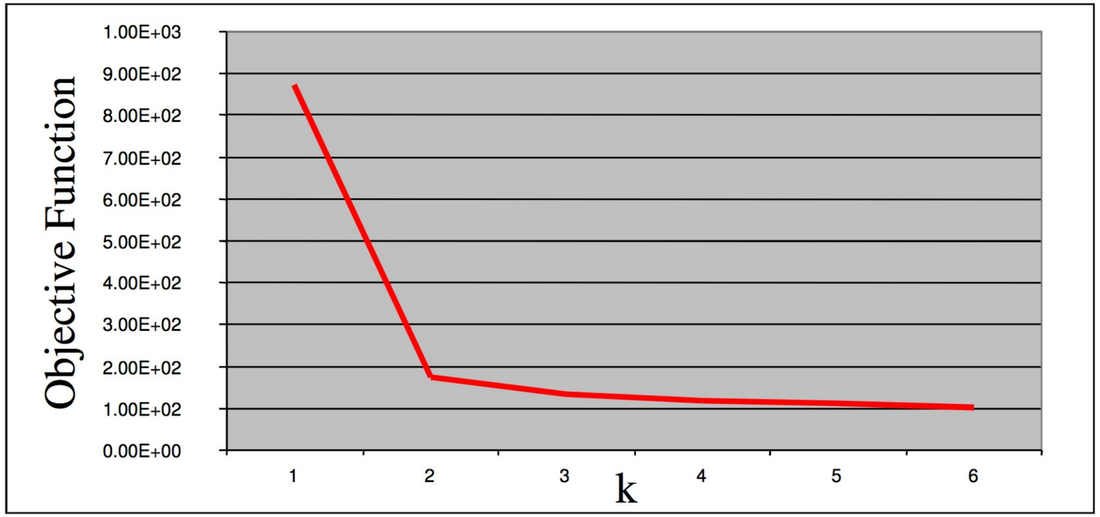
When k = 3, the objective function is 133.6



How to define the right number of clusters?

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

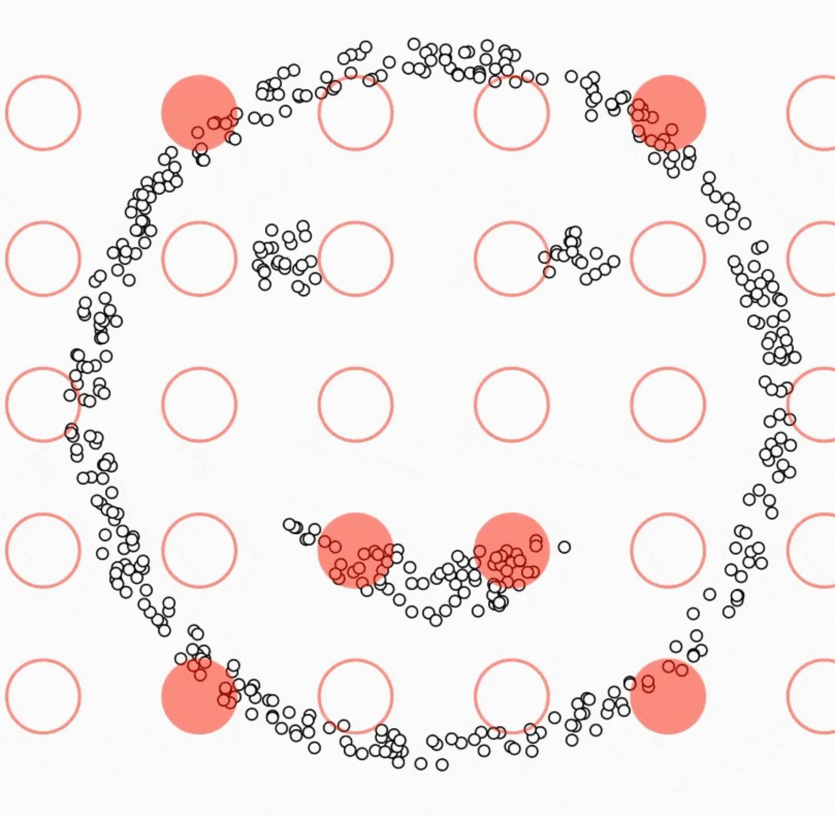
The abrupt change at k = 2, is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as “knee finding” or “elbow finding”.



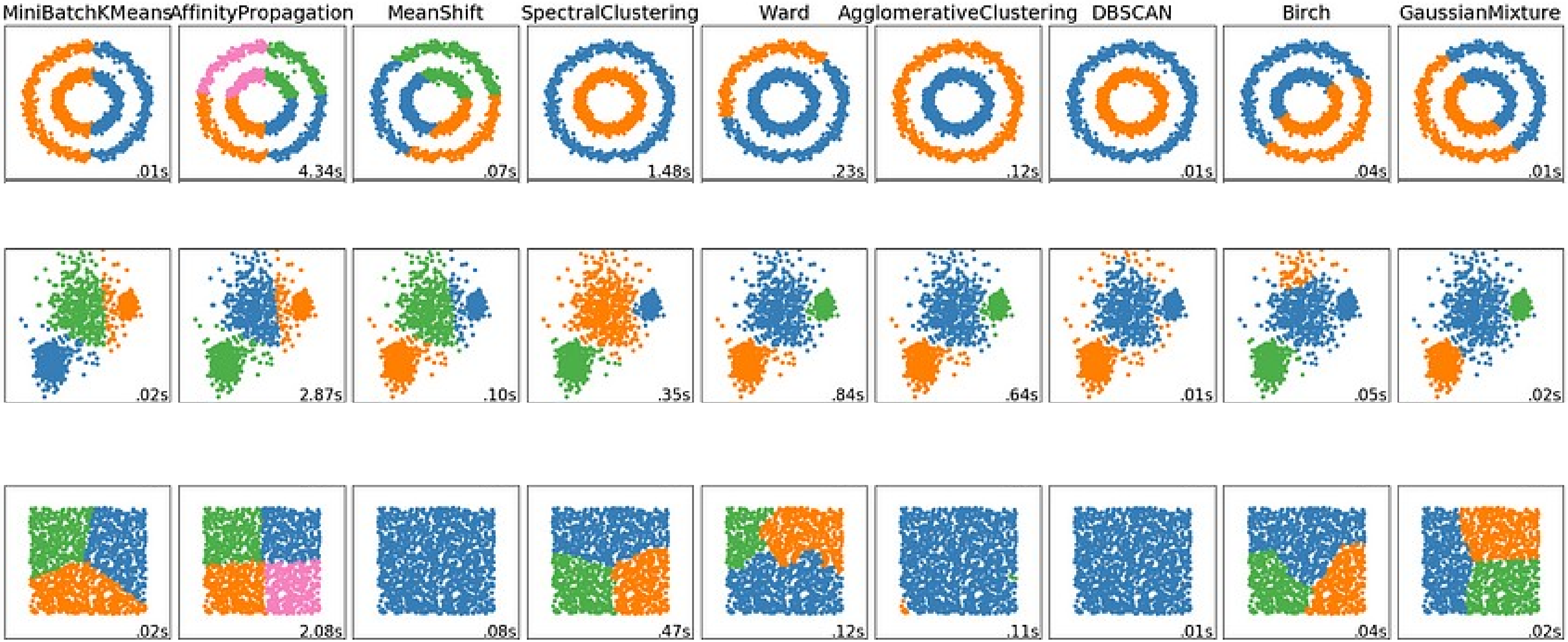
Note that the results are not always as clear cut as in this toy example

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

1. Set the parameters:
   - Epsilon (ε): The maximum distance between two data points for them to be considered neighbors.
   - MinPts: The minimum number of data points required to form a dense region.
2. Start with an arbitrary unvisited data point.
   - If the point has not been visited, mark it as visited.
   - Retrieve all its neighboring points within distance ε.
3. Check the density of the current point:
   - If the number of neighboring points is greater than or equal to MinPts, the point is considered a core point.
   - If the number of neighboring points is less than MinPts, the point is considered a border point.
4. Expand the cluster:
   - If the point is a core point, create a new cluster and add the point to the cluster.
   - Retrieve all the reachable neighboring points (directly or transitively) from the core point within distance ε.
   - Add these points to the cluster and mark them as visited.
5. Repeat steps 2 to 4 for all unvisited data points in the dataset.
   - If a border point is encountered, it is not added to any cluster.
6. The algorithm continues until all data points have been visited.
7. Classification of noise: Any unvisited data point that does not belong to any cluster is considered noise.
8. The final result is a set of clusters, where each cluster consists of the core points and their reachable neighboring points.



Comparison of clustering algorithms


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
Birch
GaussianMixture
.01s
4.34s
.07s
1.48s
.23s
.12s
.01s
.04s
.01s
.02s
2.87s
.10s
.35s
.84s
.64s
.01s
.05s
.02s
.02s
2.08s
.08s
.47s
.12s
.11s
.01s
.04s
.02s