Unsupervised Learning - Clustering

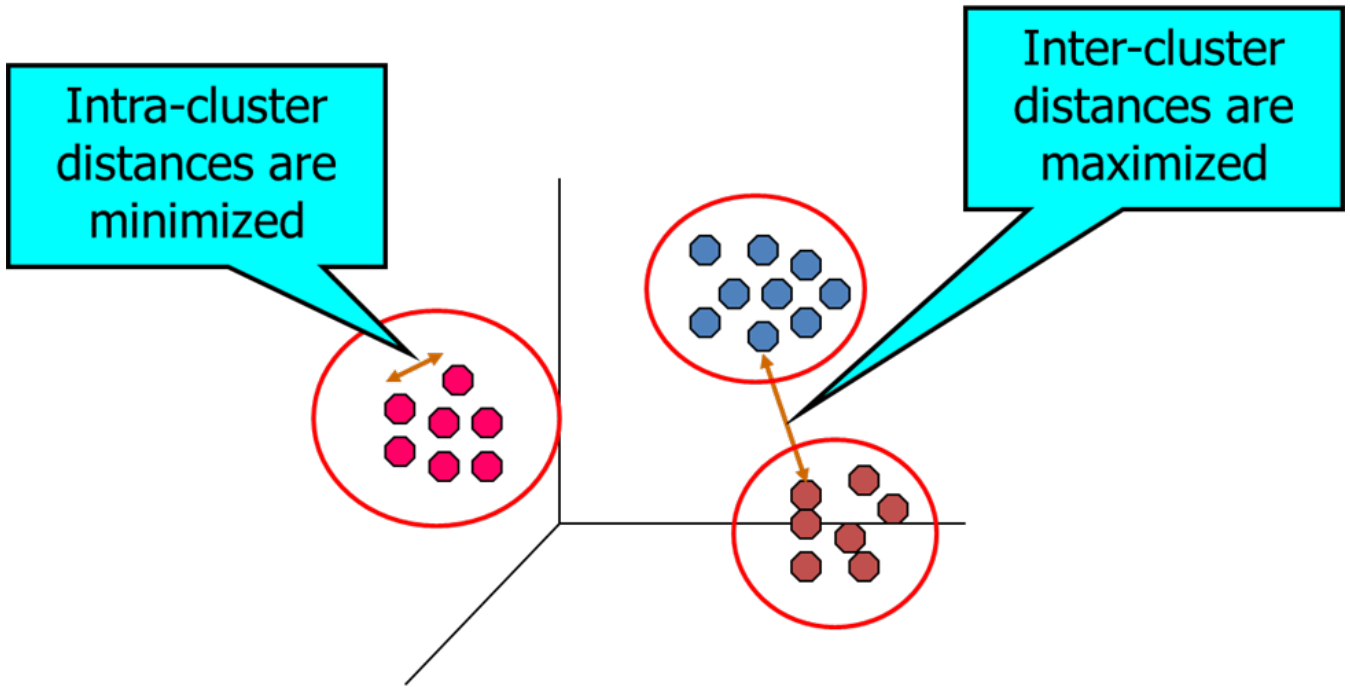
- In Unsupervised Learning, a set of predictors (features) $X_1, X_2, \ldots, X_n$ is available to analyze but not a response variable Y
- We already considered the Dimensionality Reduction task
- Now, we continue with Clustering – a broad class of methods for discovering unknown subgroups in the data
- Clustering is used in many applications, e.g. for
  - Customer segmentation
  - Data analysis
  - Anomaly detection

  to name a few

Clustering

What is cluster analysis?

- Finding subgroups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



- There is no universal definition of what a cluster is: it really depends on the context, and different algorithms will capture different kinds of clusters.

  Some algorithms look for instances centered around a particular point, called a centroid. Others look for continuous regions of densely packed instances: these clusters can take on any shape. Some algorithms are hierarchical, looking for clusters of clusters

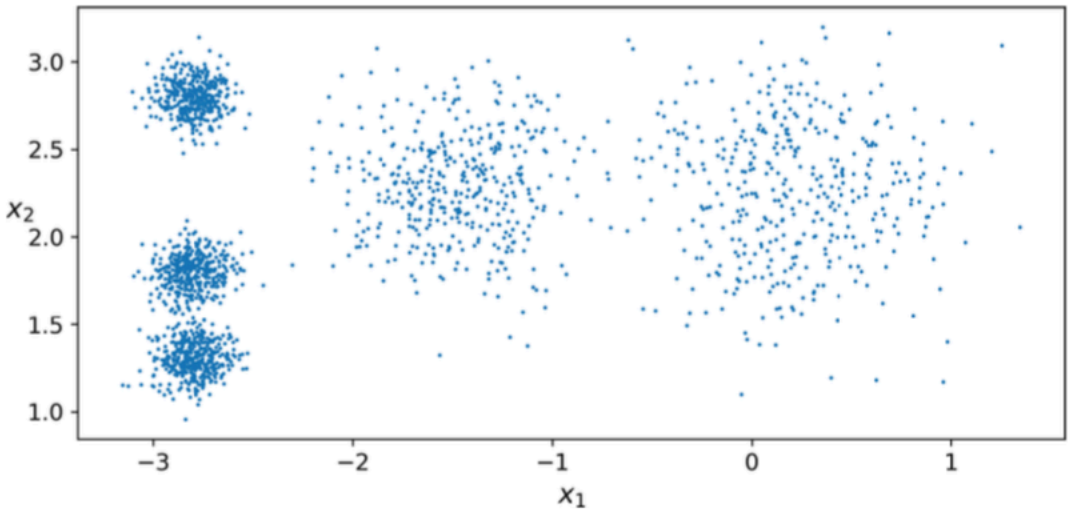
- To make this concrete, we must define what it means for 2 or more observations to be similar or different

- Often this is a domain-specific consideration based on knowledge of the data being studied

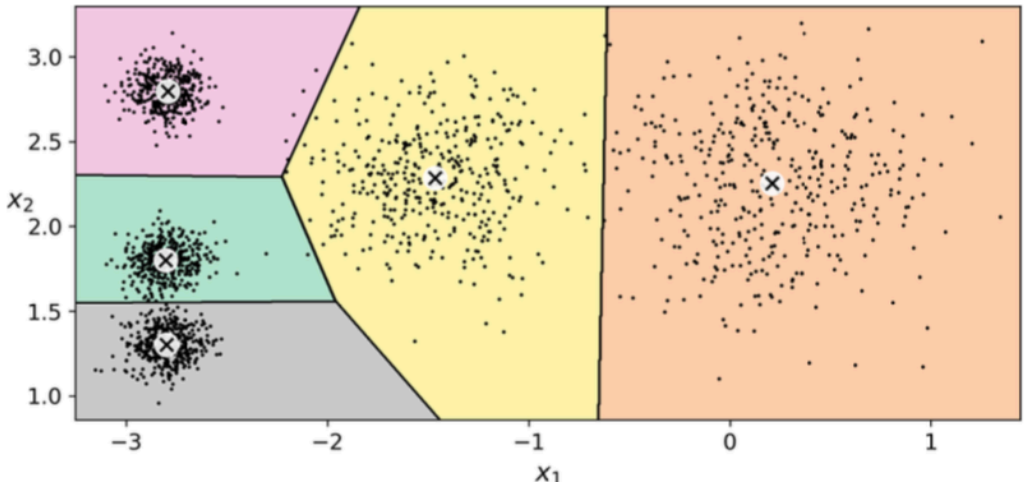
- We will look next at two popular clustering algorithms, K-Means and DBSCAN

K-Means

- Consider the unlabeled dataset below: you can clearly see five blobs of instances. The K-Means algorithm is a simple algorithm capable of clustering this kind of dataset very quickly and efficiently



- If the scikit-learn function KMeans() is run (and you have to specify the # of clusters, 5 in this case) the resulting clusters will look like this:



- Here the x indicates the centroids of the clusters (and the colors indicating clusters, are from a Voronoi tessellation). Most of the instances were clearly assigned to the appropriate cluster, but a few instances were probably mislabeled.
- Indeed, the K-Means algorithm does not behave very well when the blobs have very different diameters because all it cares about when assigning an instance to a cluster is the distance to the centroid.

K-Means Clustering Algorithm

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations (Alternatively, randomly assign K observations to be cluster means and go to **2b**.)

2. Iterate until the cluster assignments stop changing:

    a. For each of the K clusters, compute the cluster's centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster;

    b. Assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance)

http://shabal.in/visuals/kmeans/6.html

Details of K-Means Clustering - I

Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.
2. $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$.

Details of K-Means Clustering - II

- The idea behind K-means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible.
- The within-cluster variation for cluster C_k is a measure $\text{WCV}(C_k)$ of the amount by which the observations within a cluster differ from each other.
- Hence we want to solve the problem

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \text{WCV}(C_k) \right\}. \qquad (2)$$

- In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

Define Within-Cluster Variation

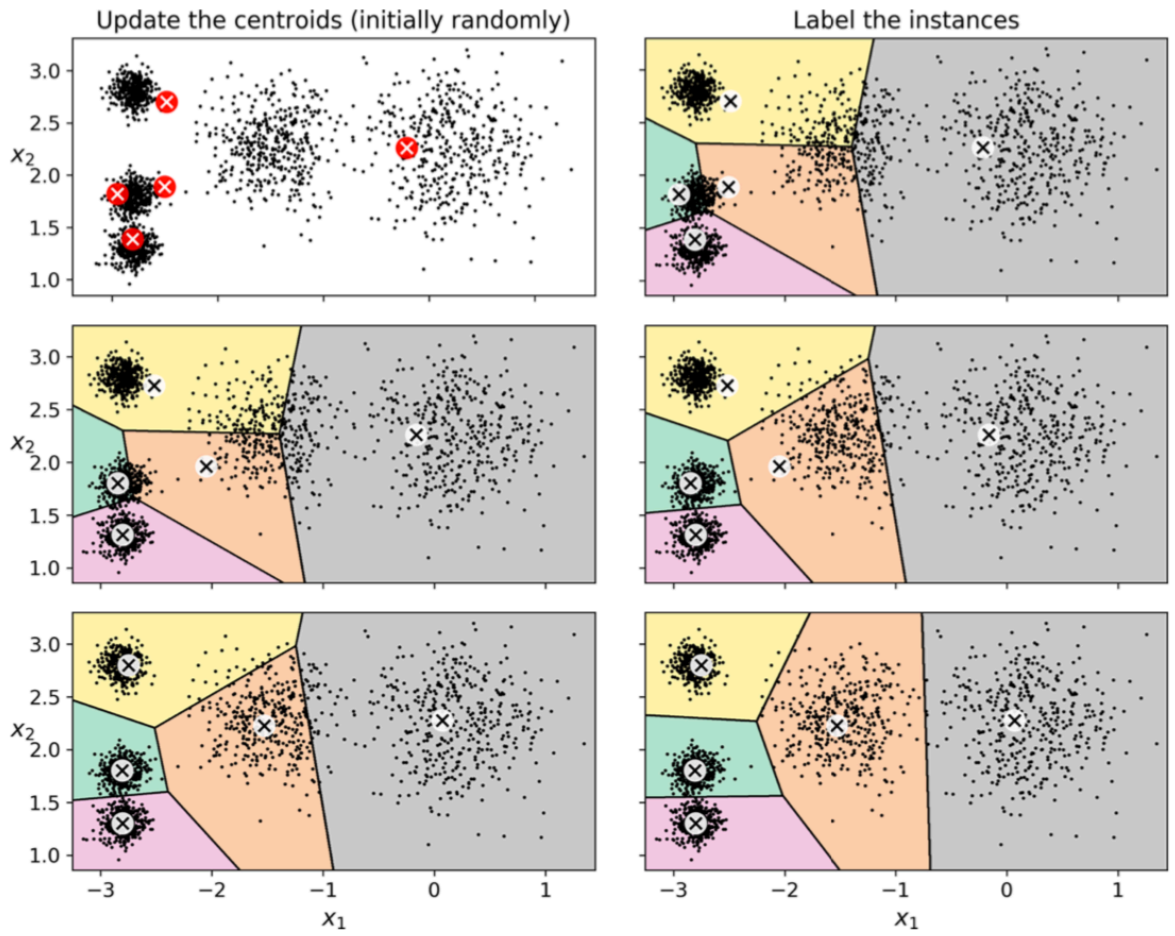
- Typically we use Euclidean distance

$$\mathrm{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2, \qquad (3)$$

where $|C_k|$ denotes the number of observations in the kth cluster.

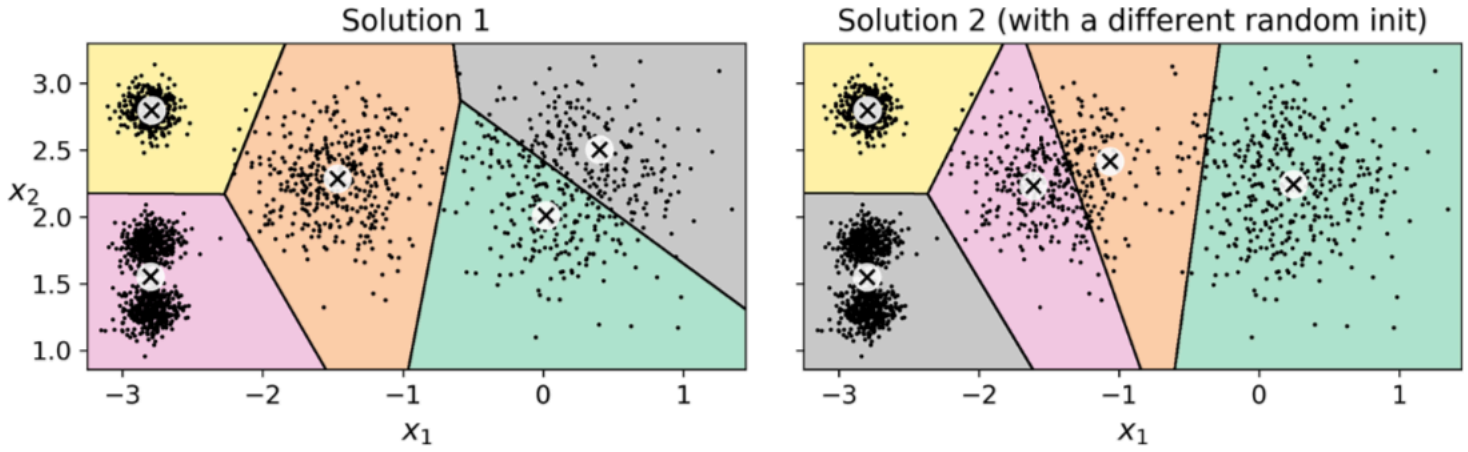
- Combining (2) and (3) gives the optimization problem that defines K-means clustering,

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}. \qquad (4)$$



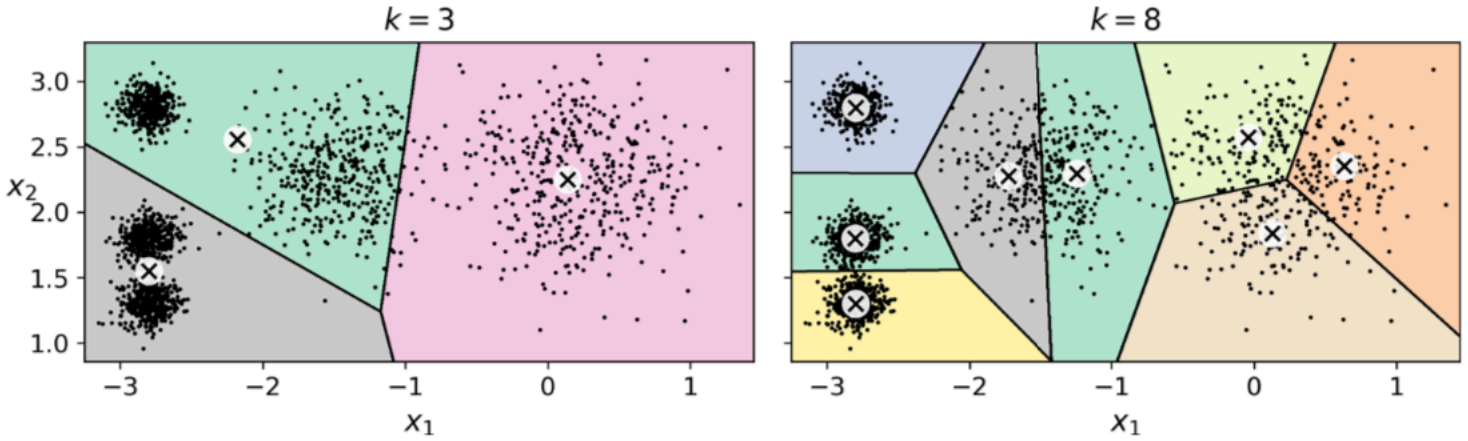
Progression of the K-means for the “5-blob” data.

- The algorithm is guaranteed to converge, but it may not converge to the right solution (i.e., it may converge to a local optimum. Plot below shows 2 suboptimal solutions that the algorithm can converge to if you are not lucky with the random initialization step.



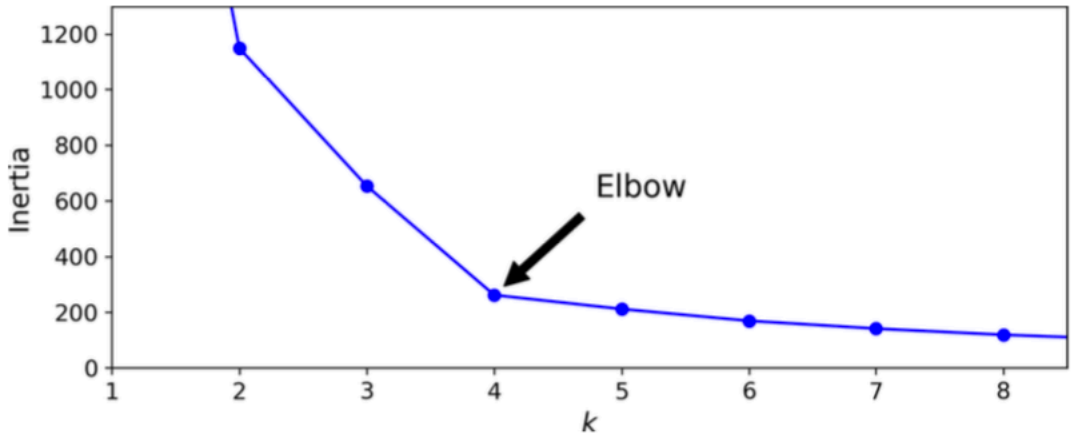
Read in HOML, Chapter 9, p.243-244 about ways of initialization.

- How about the optimal number of clusters?



Bad choices for the # clusters: when k is too small, separate clusters get merged (left), and when k is too large, some clusters get chopped into multiple pieces (right)

- The straightforward method to use is to look for an elbow in the plot of the WCV (inertia) against the # of clusters



- The inertia drops very quickly as we increase *k* up to 4, but then it decreases much more slowly as we keep increasing *k*. If we did not know better, 4 would be a good choice: any lower value would be too small, while any higher value would not help much, and we might just be splitting perfectly good clusters in half for no good reason.

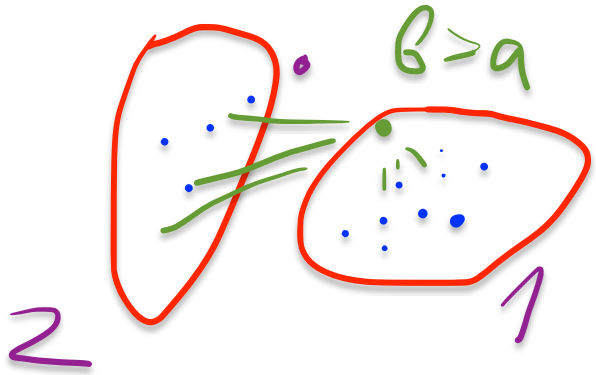
- This technique for choosing the best value for the number of clusters is rather coarse. A more precise approach (but also more computationally expensive) is to use the *silhouette score* = mean(*silhouette coefficient*) over all the instances.

- An instance's silhouette coefficient is equal to

$$(b - a) / max(a, b),$$

where *a* is the mean distance to the other instances in the same cluster (i.e., the mean intra-cluster distance) and b is the mean nearest-cluster distance (i.e., the mean distance to the instances of the next closest cluster, defined as the one that minimizes b, excluding the instance's own cluster).

- The silhouette coefficient can vary b/w -1 and $+1$. A coefficient close to
  - $+1 \implies$ the instance is inside its own cluster
  - $0 \implies$ it is close to a cluster boundary,
  - $-1 \implies$ the instance may have been assigned to the wrong cluster.



DBSCAN

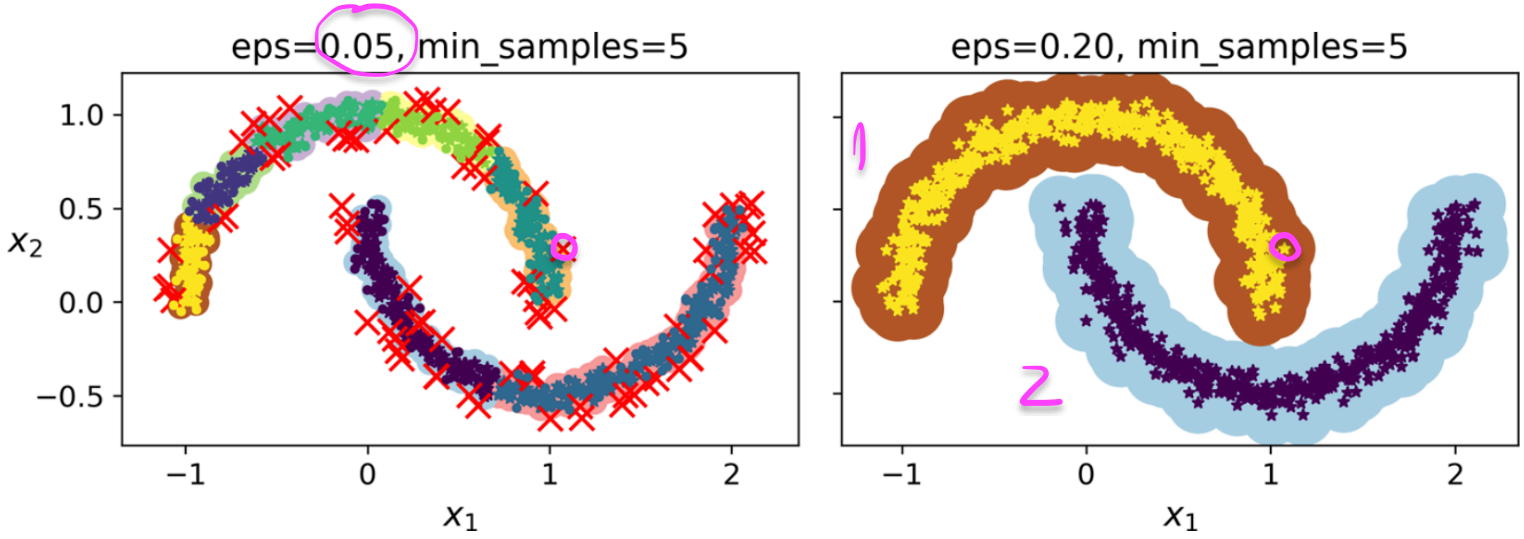
Another popular clustering algorithm that illustrates a very different approach based on local density estimation. This algorithm defines clusters as continuous regions of high density. This approach allows the algorithm to identify clusters of arbitrary shapes.

- For each instance, the algorithm counts how many instances are located within a small distance ε from it. This region is called the instance's ε-neighborhood.

- If an instance has at least min_samples instances in its ε-neighborhood (including itself), then it is considered a *core instance*. In other words, core instances are those that are located in dense regions.

- All instances in the neighborhood of a core instance belong to the same cluster. This neighborhood may include other core instances; therefore, a long sequence of neighboring core instances forms a single cluster.

- Any instance that is not a core instance and does not have one in its neighborhood is considered an anomaly.
- This algorithm works well if all the clusters are dense enough and if they are well separated by low-density regions.
- In the plots below, DBSCAN identified quite a lot of anomalies (red crosses), plus seven (!) different clusters.
- If we widen each instance's neighborhood by increasing eps (ε) from 0.05 to 0.20, we get the clustering on the right, which looks perfect.



Agglomerative Clustering Algorithm

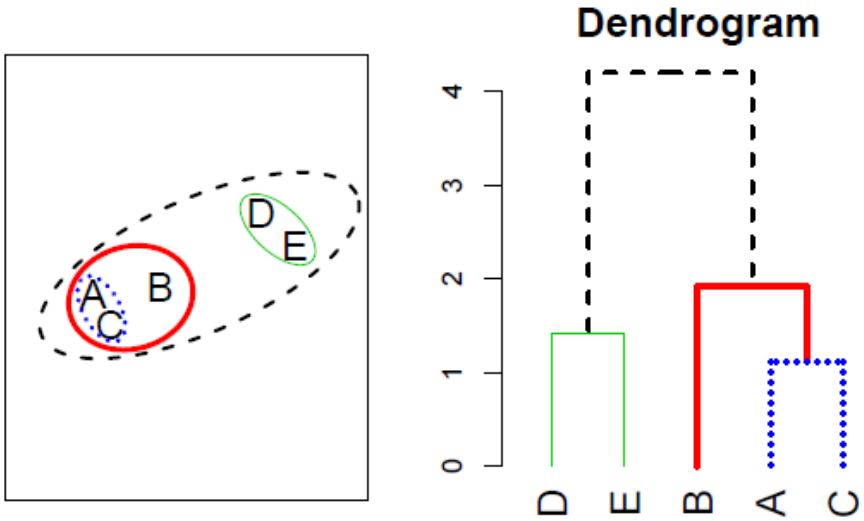
- Hierarchical clustering does not have the disadvantage to set a specific number of clusters in advance

- One example of hierarchical clustering is the agglomerative (bottom-up) clustering. It is the most common type of hierarchical clustering – it is building a dendrogram starting from the “leaves” and combining clusters up to the “trunk”

Hierarchical Clustering Algorithm

The approach in words:

- Start with each point as its own cluster
- Identify the two closest clusters and merge them
- Repeat
- End when all points are in a single cluster



The length of the branches (height) in the dendrogram is the distance b/w clusters

- The basic operation of all such methods is similar:

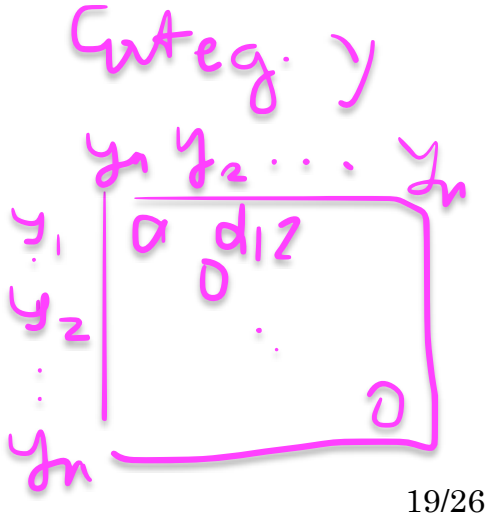
  Start with clusters $C_1, C_2, \ldots, C_n$ each containing a single object

(1) Find the nearest pair of distinct clusters, say C_i and C_j , merge C_i and C_j, delete C_j , and decrease the number of clusters by one.
(2) If the number of clusters equals one, then stop; otherwise return to 1.

- Before the process can begin, an inter observations distance matrix or similarity matrix needs to be calculated
- There are many ways to calculate distances or similarities between pairs of individuals, but the most commonly used one is the Euclidean distance, defined as

$$d_{ij} = \sqrt{\sum_{k=1}^{p} (y_{ik} - y_{jk})^2}$$

for two objects i and j

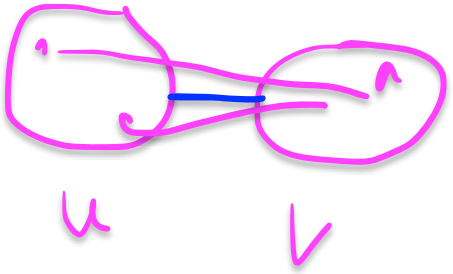


- The Euclidean distances between each pair of observations can be arranged in a matrix $\boldsymbol{D} = \{d_{ij}\}$ that is symmetric because $d_{ij} = d_{ji}$ and has 0s on the main diagonal. Distance calculations from the raw data might happen after standardization (if variables are on very different scales)

- Then a distance between 2 clusters U and V is defined. Typical choices:

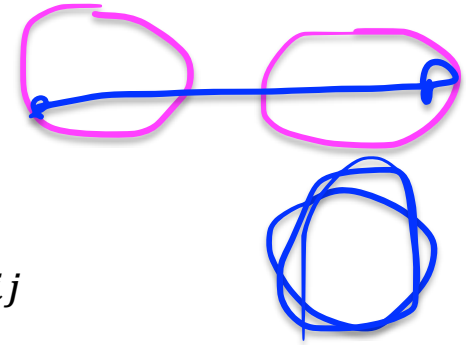
  - Single linkage

$$d_{UV} = \min_{i \in U, j \in V} d_{ij}$$



  - Complete linkage

$$d_{UV} = \max_{i \in U, j \in V} d_{ij}$$

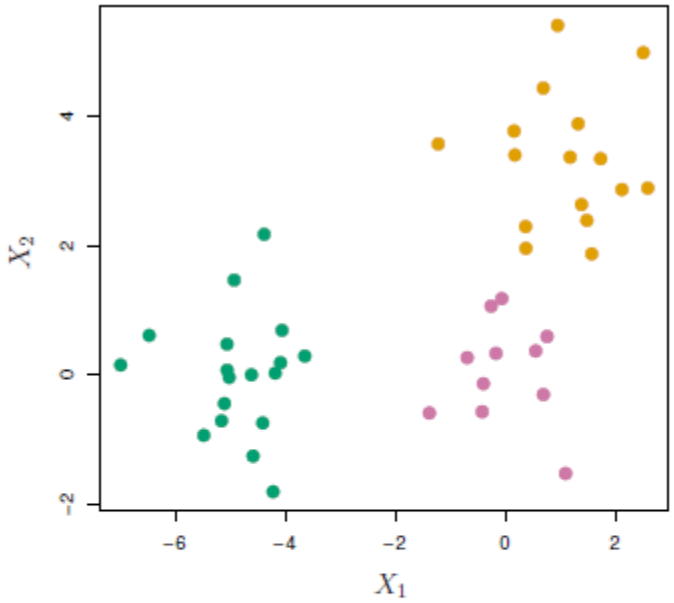


  - Average linkage

$$d_{UV} = \frac{1}{N_U N_V} \sum_{i \in U} \sum_{j \in V} d_{ij}$$

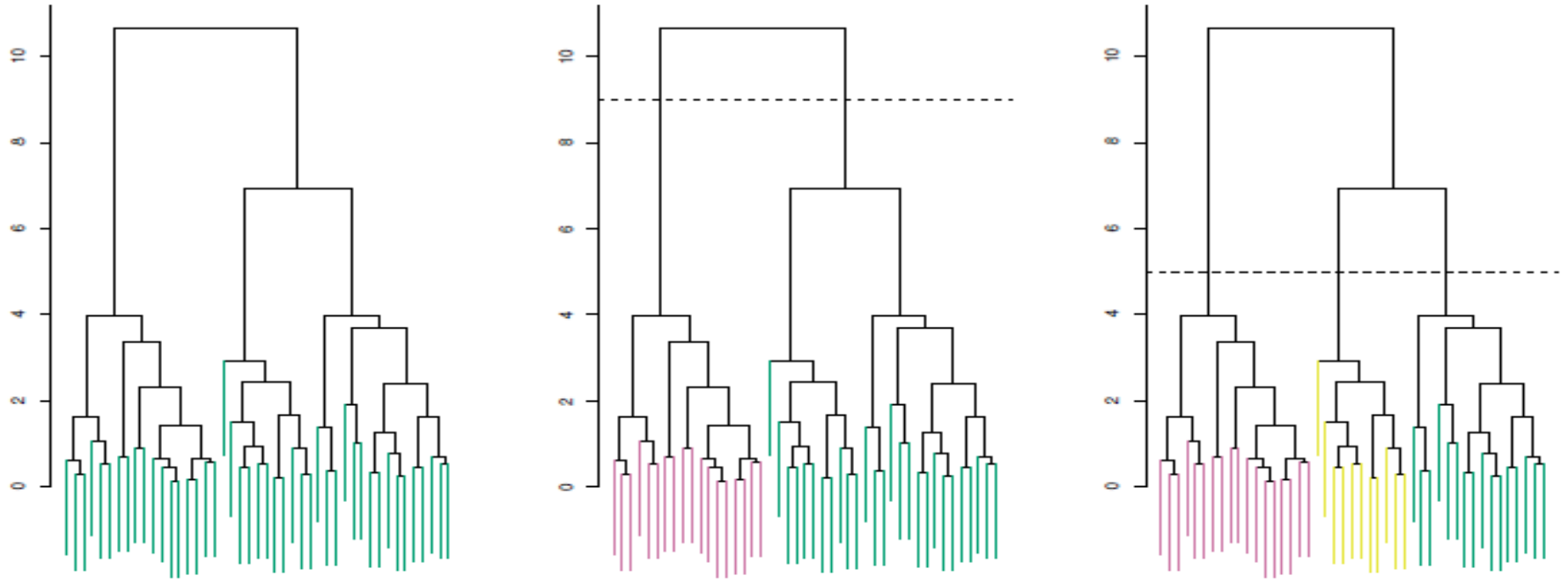
Example

ISLR



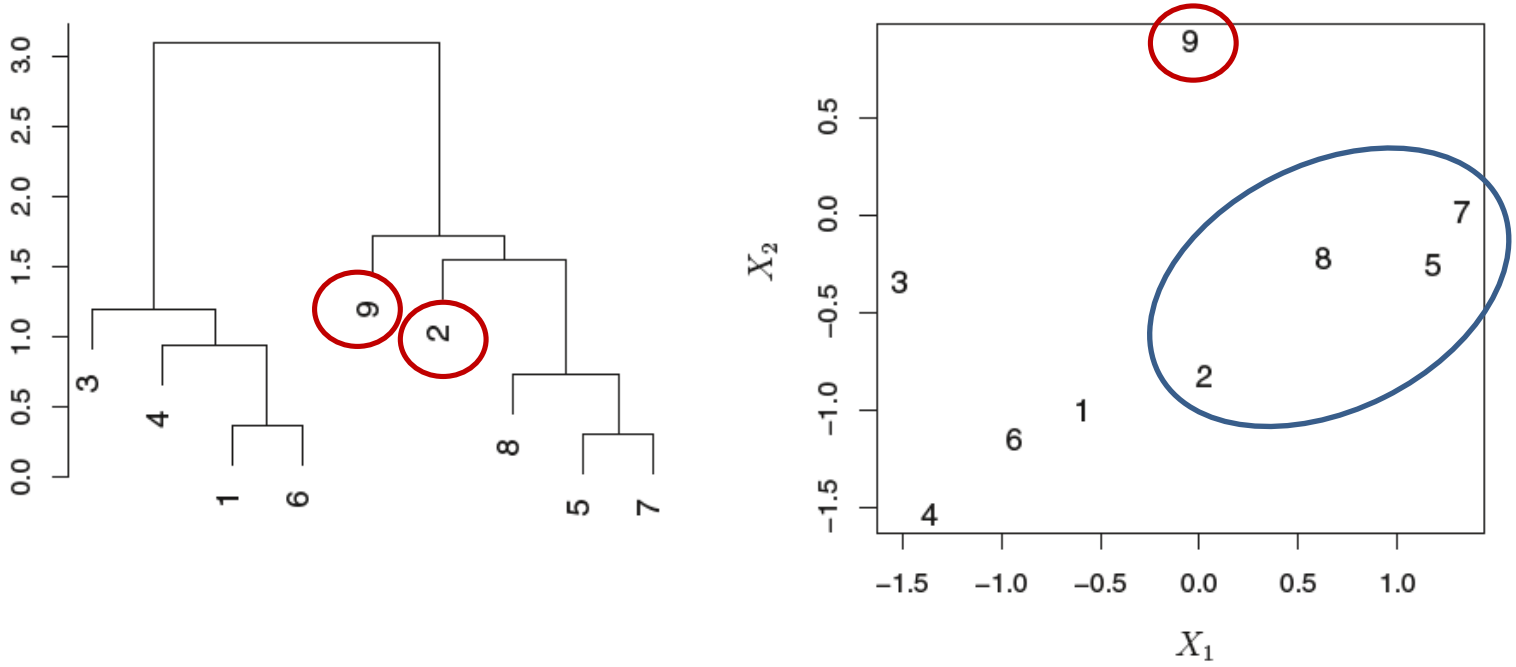
45 observations generated in 2-dimensional space. There are 3 clusters, shown in separate colors.

However, we will treat the class labels as unknown and seek to cluster the observations in order to discover the classes from the data...



- Left: Dendrogram found with complete clustering and Euclidean distance
- Center: The same dendrogram cut at level of 9. This results in 2 clusters
- Right: Now the dendrogram is cut at a height of 5 producing 3 clusters shown in different colors (not from the original clustering)

Note: Observations plotted next to each other on the horizontal line are not necessarily close. Their similarity is based on "height" (vertical line):



On the picture above observations "9" and "2" are plotted next to each other. The distance of "9" to the cluster made of "2", "5", "7" and "8" (and thus to "2" in a sense) is about 1.7 however. This is the height on the dendrogram where these 2 branches/clusters merge. Additional clarification – on the right plot.

Model based clustering

- In the model based approach cluster k has expected proportion p_k of the objects and variable values for this cluster have a p.d.f. $f_k(\mathbf{y})$
- If there are K clusters, the observation vector for a single object has the following mixing distribution

$$f_{Mix}(\mathbf{y}) = \sum_{k=1}^{K} p_k f_k(\mathbf{y})$$

where $p_k \geq 0$ and $\sum_{k=1}^{K} p_k = 1$.

- The clustering problem becomes that of estimating the parameters of the assumed mixture and then using the estimated parameters to calculate the posterior probabilities of cluster membership
- Determining the number of clusters also reduces to a model selection problem for which objective procedures exist

- The most common model uses multivariate normal densities in the mixture, i.e. $f_k(\mathbf{y}) \sim N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1,2,\dots,K$

- Using Maximum Likelihood approach both the number of clusters and the parameters can be determined. Usually there are too many parameters to estimate but in simpler cases useful results can be derived, e.g. if $\boldsymbol{\Sigma}_k = \eta\mathbf{I}, k = 1,2,\dots,K$ the mixture model is approximately the same as K-means clustering

- If we can estimate the parameters for a fixed number of clusters, then we can select the number of clusters applying the Akaike information criterion

$$AIC = 2\ln L_{\max} - 2N\left[K\frac{1}{2}(p+1)(p+2) - 1\right]$$

Here the maximum of the log-likelihood function is "penalized" (meaning decreased) by the number of parameters (and number the observations). The parameter # is $K-1$ for the probabilities (cluster number minus 1), $K \times p$ for the means and $K \times p(p+1)/2$ for the covariance matrix

- For a fixed K, the parameters are determined based on the Expectation-Maximization (EM) algorithm. At convergence, the jth object is assigned to cluster k with the largest condition (posterior) probability

$$p(\text{cluster } k|\mathbf{y}_j) = \hat{p}_j \, f(\mathbf{y}_j|k) / \sum_{i=1}^{K} \hat{p}_i \, f(\mathbf{y}_i|k)$$

- In one implementation, series of mixture models with MVN densities in which some features of the covariance matrix (orientation, size, and shape) vary between clusters is considered

- These new criteria arise from considering the re-parameterization of the covariance matrix $\boldsymbol{\Sigma}_k = \boldsymbol{D}_k \boldsymbol{\Lambda}_k \boldsymbol{D}_k = \boldsymbol{D}_k \lambda_k \mathbf{A}_k \boldsymbol{D}_k$ in terms of its eigenvector (matrix $\boldsymbol{D}_k$) and eigenvalue description.

  Here λ_k is the largest eigenvalue of $\boldsymbol{\Sigma}_k$ and $\mathbf{A}_k = diag(1, \alpha_2, \ldots, \alpha_p)$ contains the ratios of the division of the other eigenvalues by λ_k.

Thus $\boldsymbol{D}_k$ controls the orientation, λ_k - the size (volume) and $\mathbf{A}_k$- the shape