Applied Statistical Methods

Unsupervised Learning: Clustering

Xuemao Zhang
East Stroudsburg University

April 21, 2023

Outline

- Clustering
- K-Means Clustering
- Hierarchical Clustering
- Python: K-Means
- Python: Hierarchical Clustering

Clustering

- Clustering refers to a very broad set of techniques for finding subgroups, or clusters, in a data set.
  - Group or segment the data set (a collection of objects) into subsets so that those within each subset are more closely related to others than those objects assigned to other subsets.
  - Each group (subset) is called a cluster.
- Clustering looks for homogeneous subgroups among the observations.
- We must define what it means for two or more observations to be similar or different.
  - What is a meaningful cluster?
  - How do we validate clustering results?

Clustering

- What are meaningful clusters?

How many clusters?

Six Clusters

Two Clusters

Four Clusters

Proximity and Dissimilarity Matrices

Clustering results are crucially dependent on the measure of dissimilarity (or distance) between the "points" to be clustered.

- Proximity Matrix: $n \times n$ with the ij-th element d_{ij} measuring the proximity (alikeness or affinity) between the ith and the jth objects (or observations). D is typically symmetric.
- One can use a dissimilarity matrix instead.
- Dissimilarity between points i and i': $\mathbf{x}_i$ and $\mathbf{x}_{i'}$:

$$d(\mathbf{x}_i, \mathbf{x}_{i'}) = \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j})$$

- A weighted version:

$$d(\mathbf{x}_i, \mathbf{x}_{i'}) = \sum_{j=1}^{p} w_j d_j(x_{ij}, x_{i'j}), \quad \sum_{j=1}^{p} w_j = 1.$$

Proximity and Dissimilarity Matrices

Types of Distances:

- Squared distance: $d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$
- Absolute difference: $d_j(x_{ij}, x_{i'j}) = |x_{ij} - x_{i'j}|$
- Correlation:

$$\rho(\mathbf{x}_i, \mathbf{x}_{i'}) = \frac{\sum_j (x_{ij} - \bar{x}_i)(x_{i'j} - \bar{x}_{i'})}{\sqrt{\sum_j (x_{ij} - \bar{x}_i)^2 \sum_j (x_{i'j} - \bar{x}_{i'})^2}}, \quad \bar{x}_i = \sum_j x_{ij}/p.$$

- If inputs are standardized, $\sum_j (x_{ij} - x_{i'j})^2 \propto 2(1 - \rho(\mathbf{x}_i, \mathbf{x}_{i'}))$: clustering based on correlation (similarity) is equivalent to that based on squared distance (dissimilarity).

Clustering

Two clustering methods:

- In K-means clustering, we seek to partition the observations into a pre-specified number of clusters.
- In hierarchical clustering, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a dendrogram, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n.

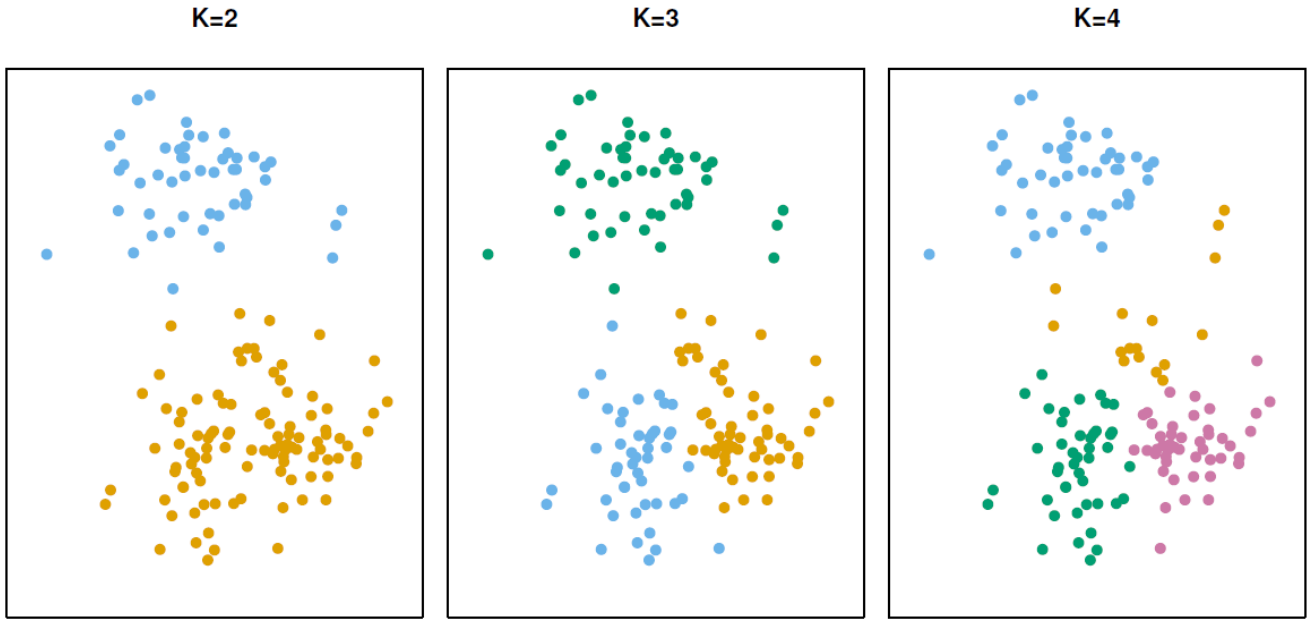
K-means clustering

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

- 1. $C_1 \cup \cdots \cup C_K = \{1, 2, \ldots, n\}$. 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 $i \in C_k$.

K-means clustering



- A simulated data set with 150 observations in 2-dimensional space. Panels show the results of applying K-means clustering with different values of K, the number of clusters. The color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm. Note that there is no ordering of the clusters, so the cluster coloring is arbitrary. These cluster labels were not used in clustering; instead, they are the outputs of the clustering procedure.

Clustering - Combinatorial Algorithm

- The idea behind K-means clustering is that a good clustering is one for which the within-cluster variation is as small as possible.
- Goal: Find the clustering C such that WCV(C) (within cluster variation/dissimilarity) is minimized:

$$WCV(C) = \frac{1}{2}\sum_{k=1}^{K}\left(\sum\sum_{i,i' \in C_k} d(\mathbf{x}_i, \mathbf{x}_{i'})\right)$$

- Total Dissimilarity:

$$T = \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n} d_{ij} = \frac{1}{2}\sum_{k=1}^{K}\sum_{i \in C_k}\left(\sum_{j \in C_k} d_{ij} + \sum_{j \notin C_k} d_{ij}\right)$$

Clustering - Combinatorial Algorithm

- And within cluster dissimilarity is

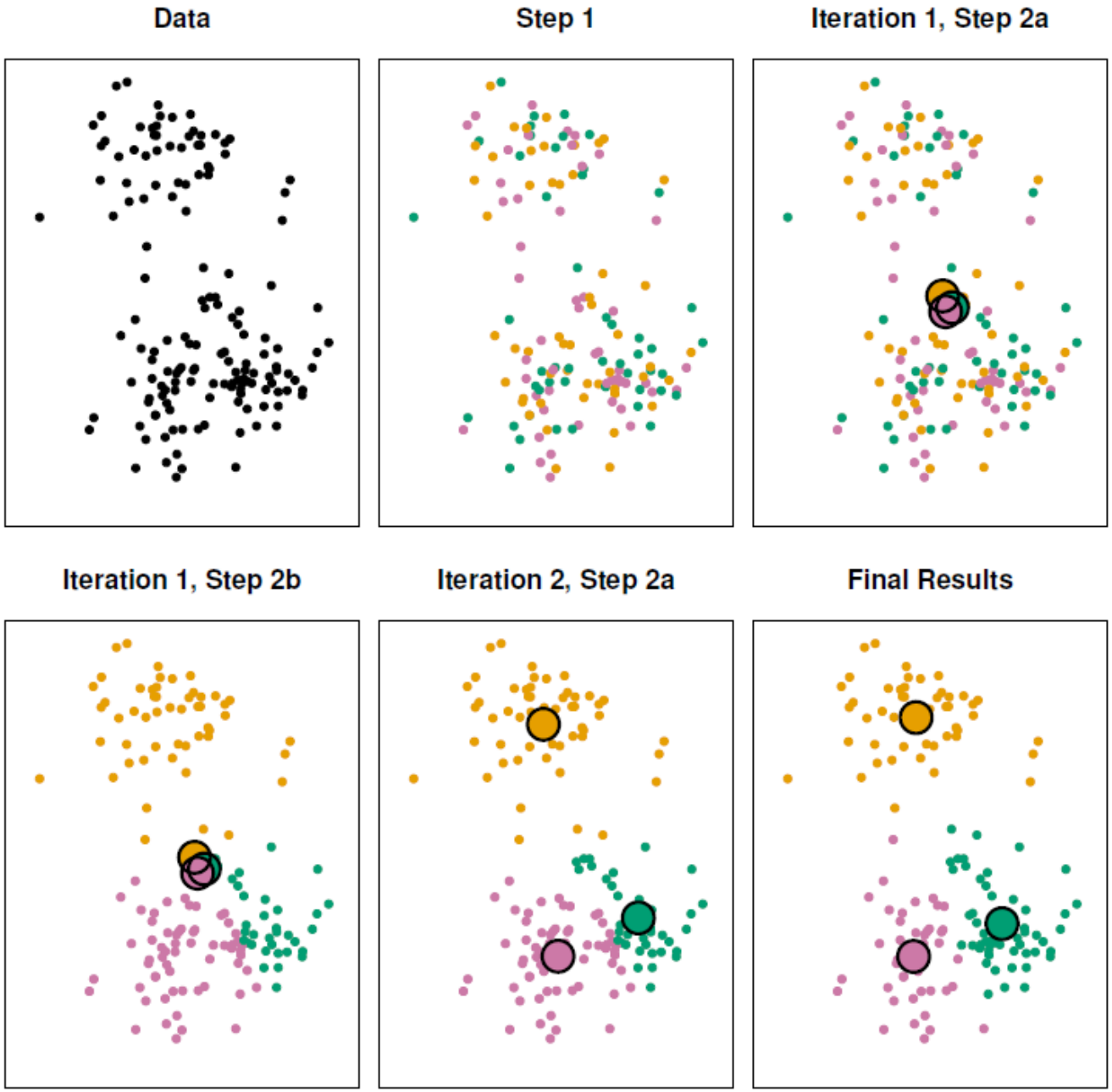
$$BCV(C) = \frac{1}{2}\sum_{k=1}^{K}\left(\sum\sum_{i\in C_k, i'\notin C_k} d(\mathbf{x}_i, \mathbf{x}_{i'})\right)$$

- Therefore, $WCV(C) = T - BCV(C)$.
- Minimizing $WCV(C)$ is equivalent to maximizing $BCV(C)$.
- One needs to minimize $WCV(C)$ over all possible assignments of n points to K clusters. It is not feasible when the number of data points is large (Jain and Dubes, 1988).

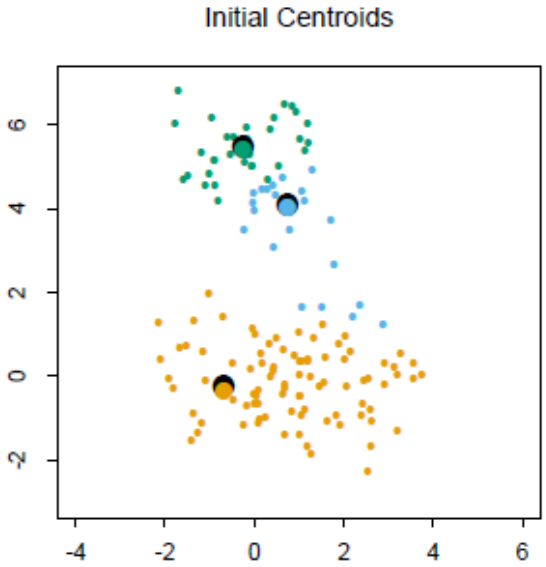
K-Means Clustering Algorithm

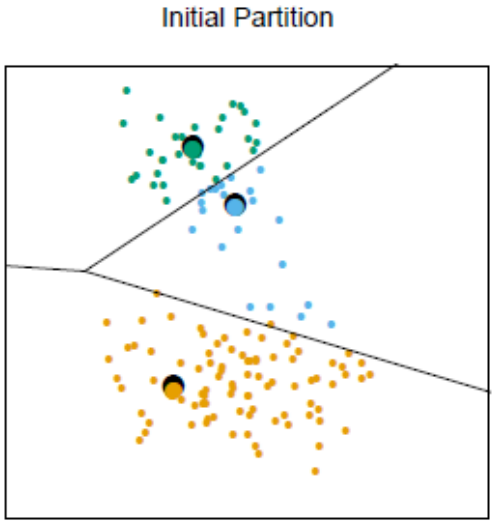
- 1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
- 2. Iterate until the cluster assignments stop changing:
  - 2.1 **Find cluster means** (cluster assignments is fixed): For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster: $\bar{\mathbf{x}}_k = \frac{1}{n_k} \sum_{i \in C_k} \mathbf{x}_i$, where n_k is the number of points in cluster C_k.
  - 2.2 **Find cluster assignments**(cluster means are fixed): Assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance).
- However it is not guaranteed to give the global minimum which implies that K-means clustering is depending on the initial cluster assignments.

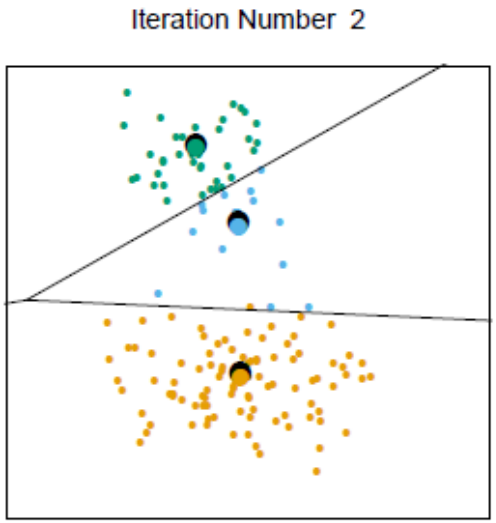
K-Means Clustering Algorithm

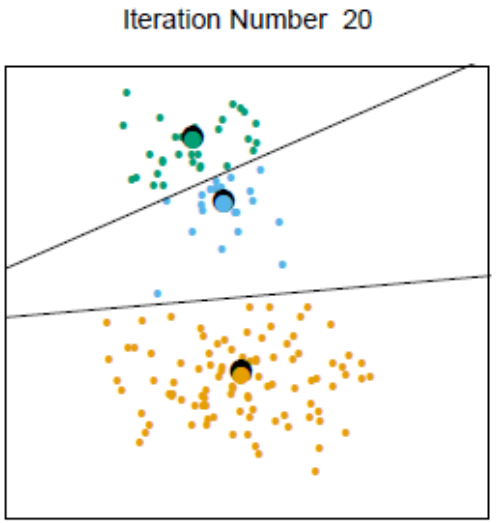


K-Means Clustering Algorithm









K-means Properties

- Steps 1 and 2 decrease $WCV(C)$.
- Local solution - not necessarily global solution.
- Depends on starting values (initialization).
- K needs to be set before.
- Best for compact, spherical clusters.
- Does not work well when cluster sizes are very different.

K-means - Initializations

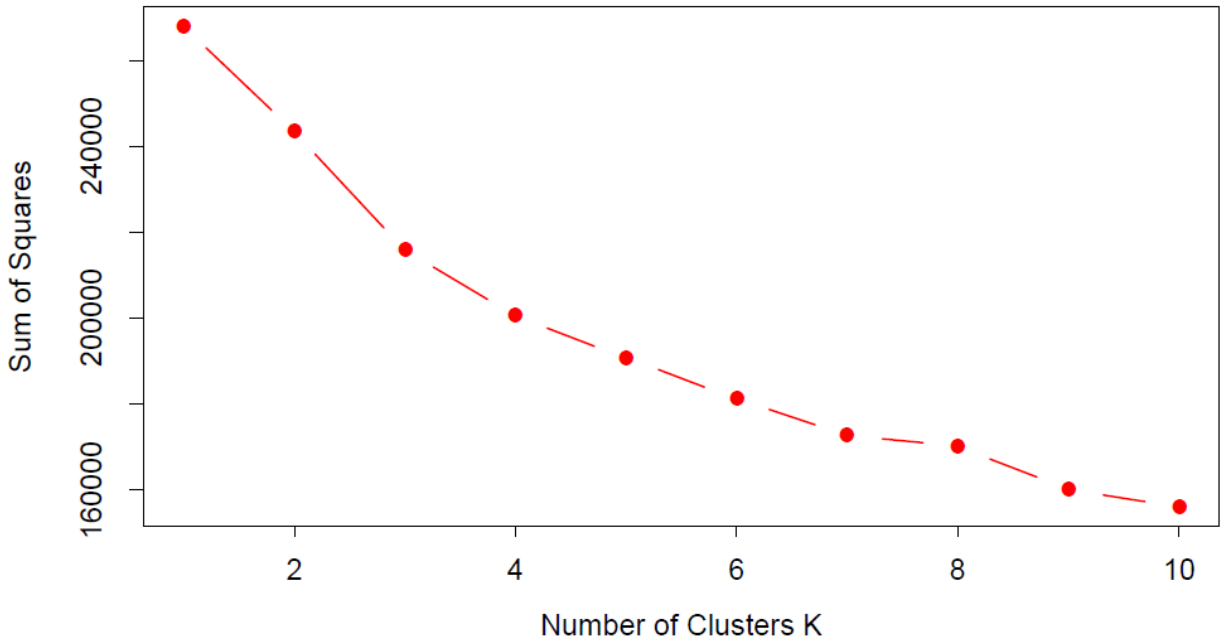
K-means - Initializations

Details of Previous Figure:

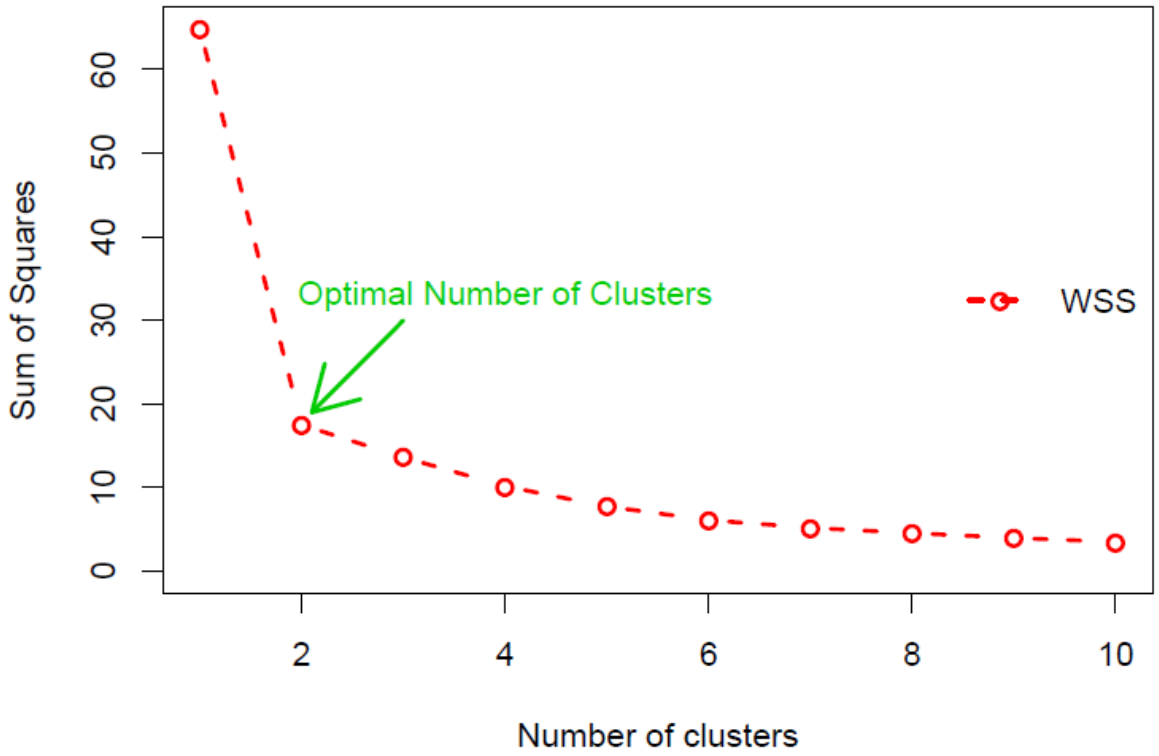
- K-means clustering performed six times on the data from previous figure with $K = 3$, each time with a different random assignment of the observations in Step 1 of the K-means algorithm.
- Above each plot is the value of the objective $WCV(C)$.
- Three different local optima were obtained, one of which resulted in a smaller value of the objective and provides better separation between the clusters.
- Those labeled in red all achieved the same best solution, with an objective value of 235.8.

How to Choose K?

- Can we choose K that minimizes $WCV(C)$?
- Can we choose K by a Validation set? Cross-Validation?



How to Choose K?

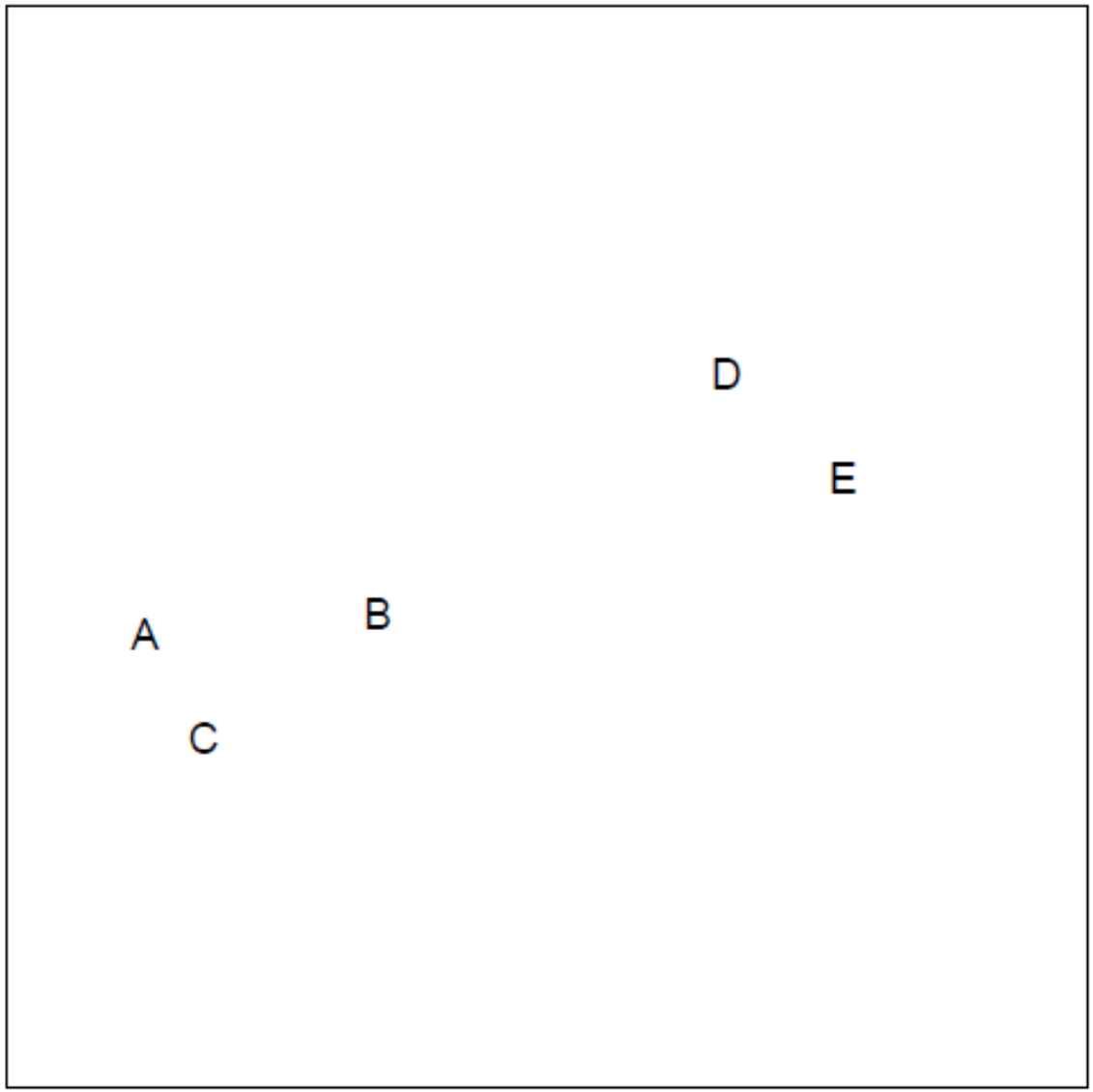


Hierarchical Clustering

- K-means clustering requires us to pre-specify the number of clusters K.
- Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of K.
- We will describe **bottom-up** or **agglomerative** clustering. This is the most common type of hierarchical clustering, and refers to the fact that a dendrogram is built starting from the leaves and combining clusters up to the trunk.

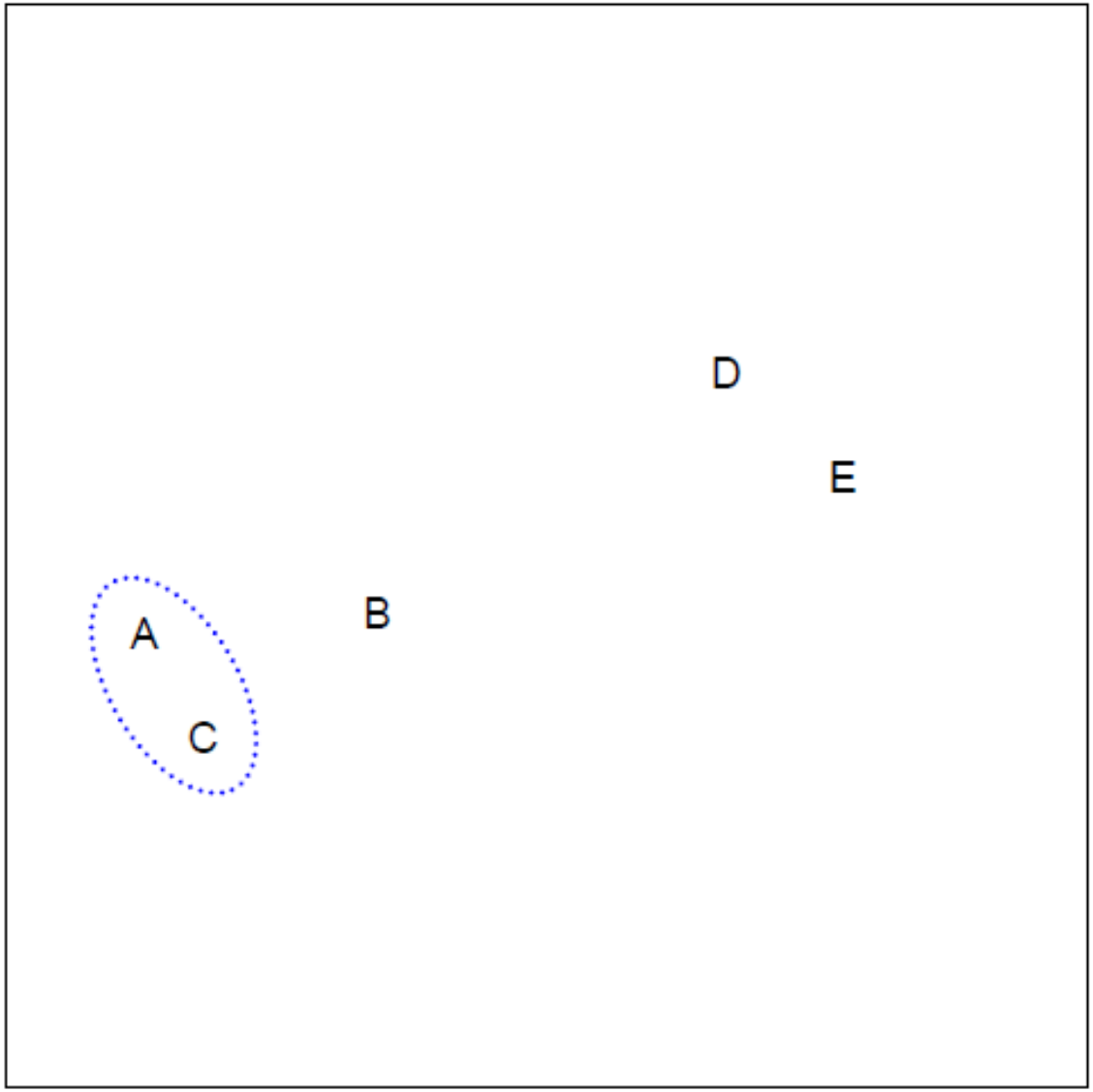
Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion . . .



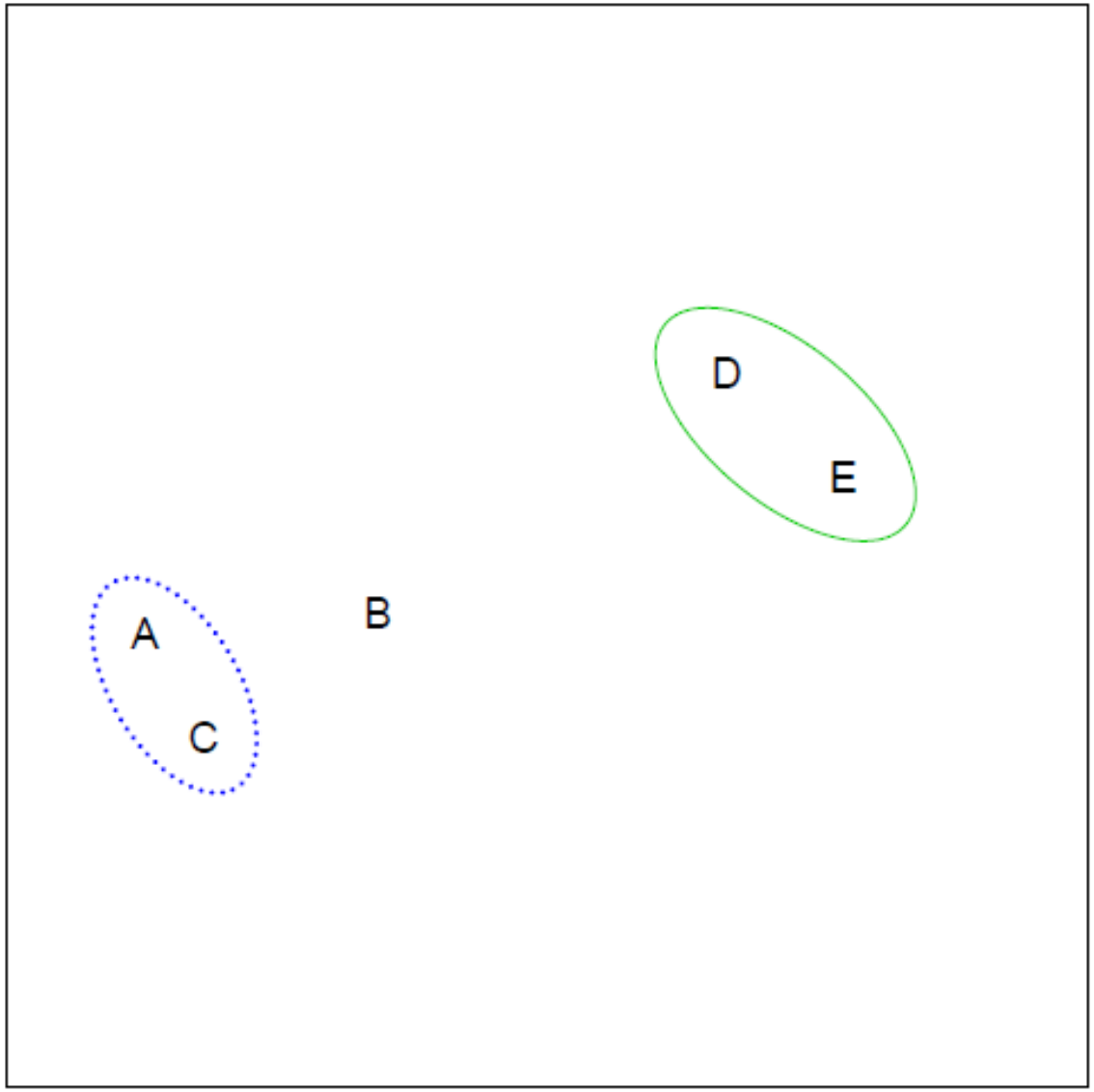
Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion . . .



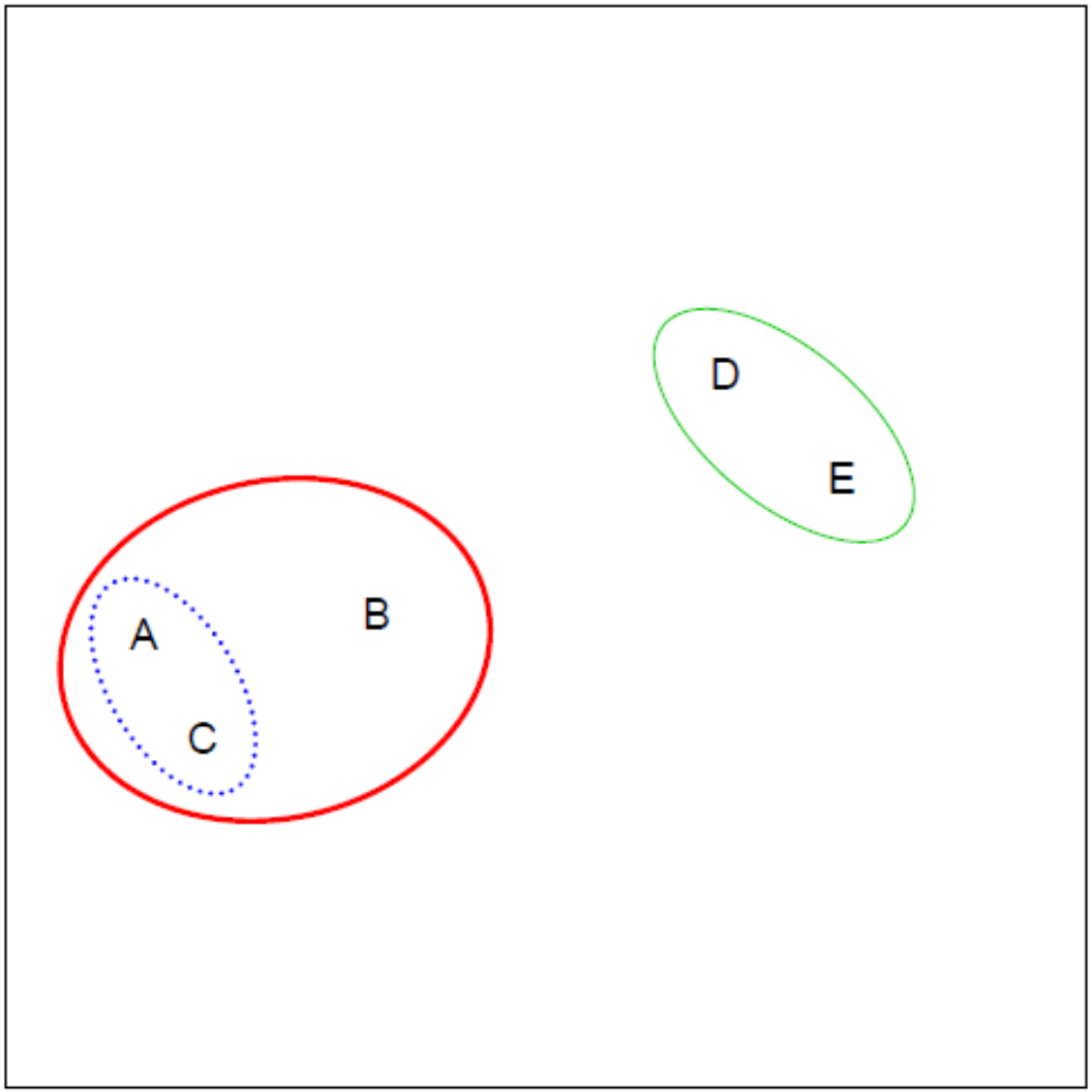
Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion . . .



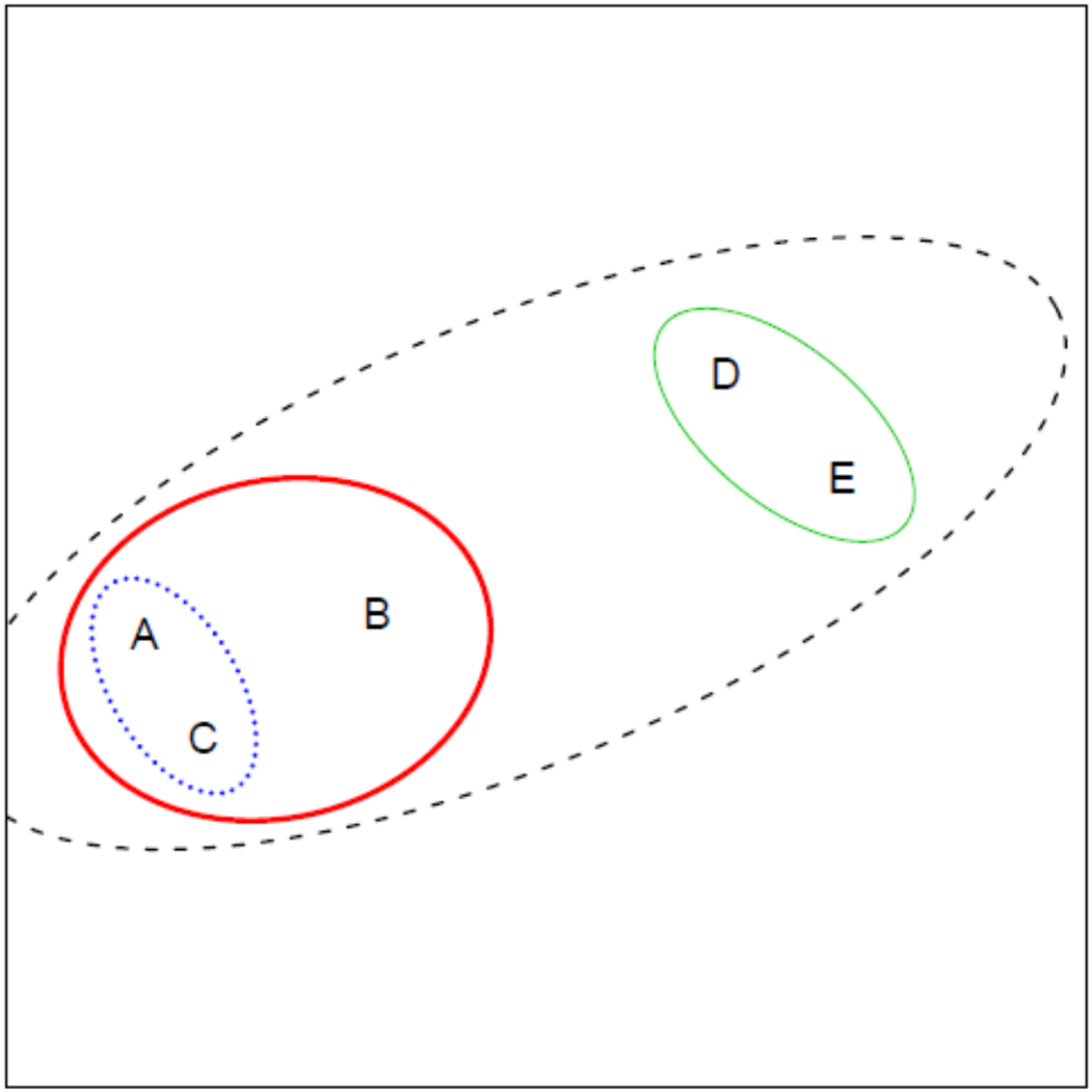
Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion . . .



Hierarchical Clustering: the idea

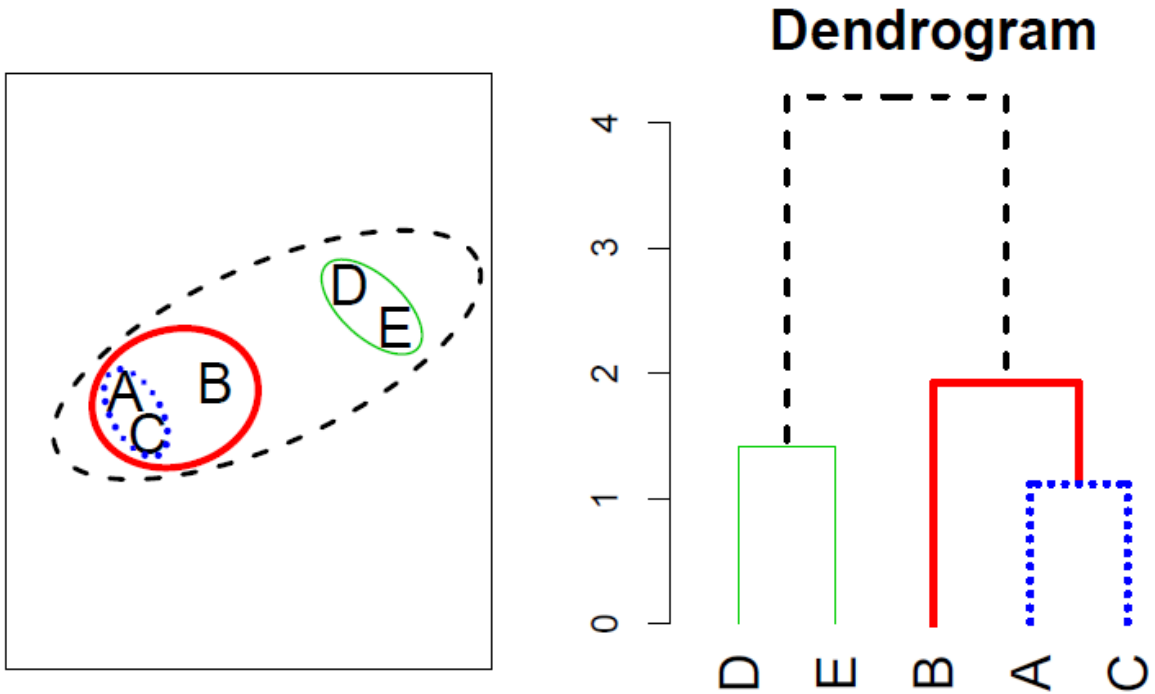
Builds a hierarchy in a "bottom-up" fashion . . .



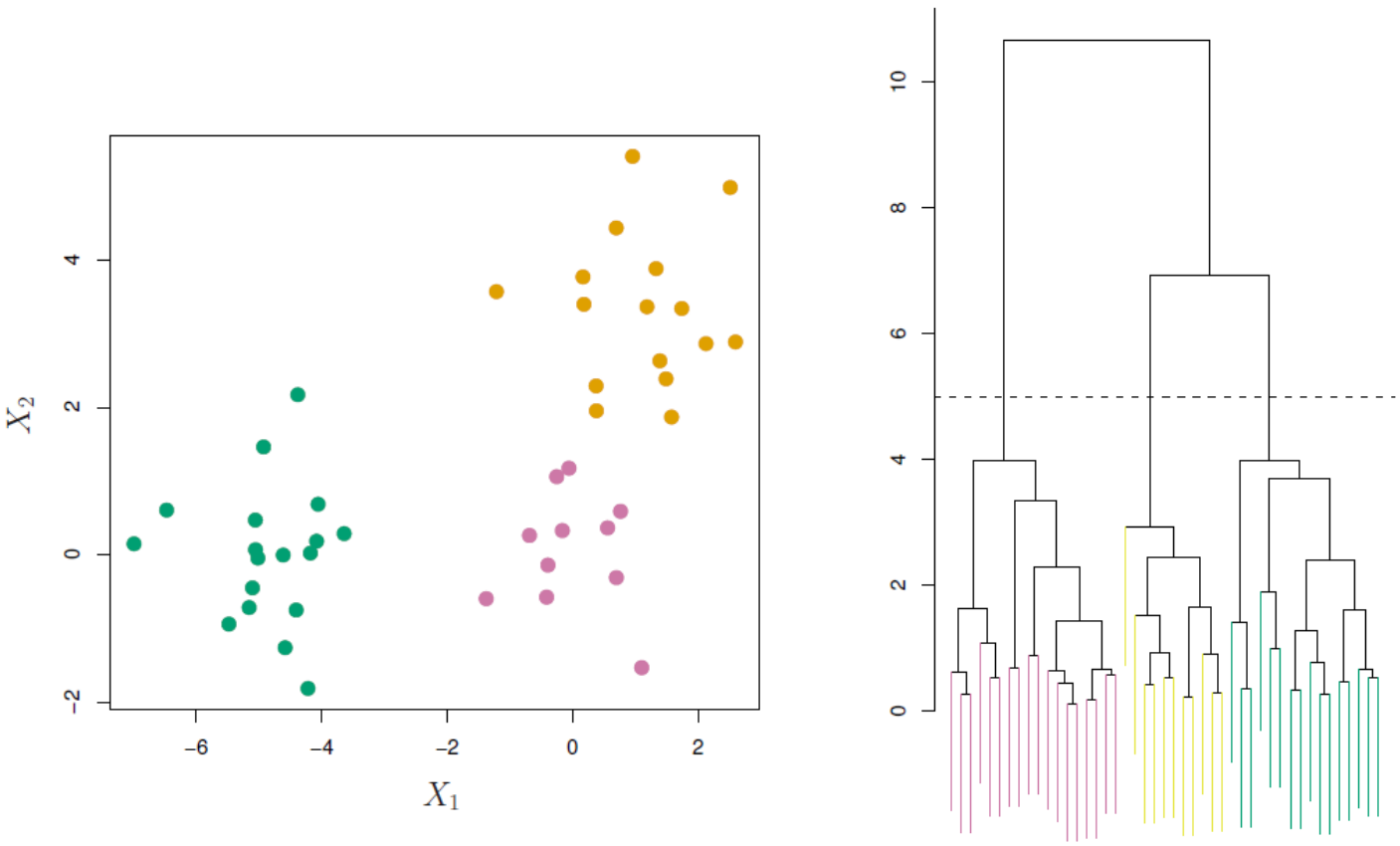
Hierarchical Clustering

The approach in words:

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



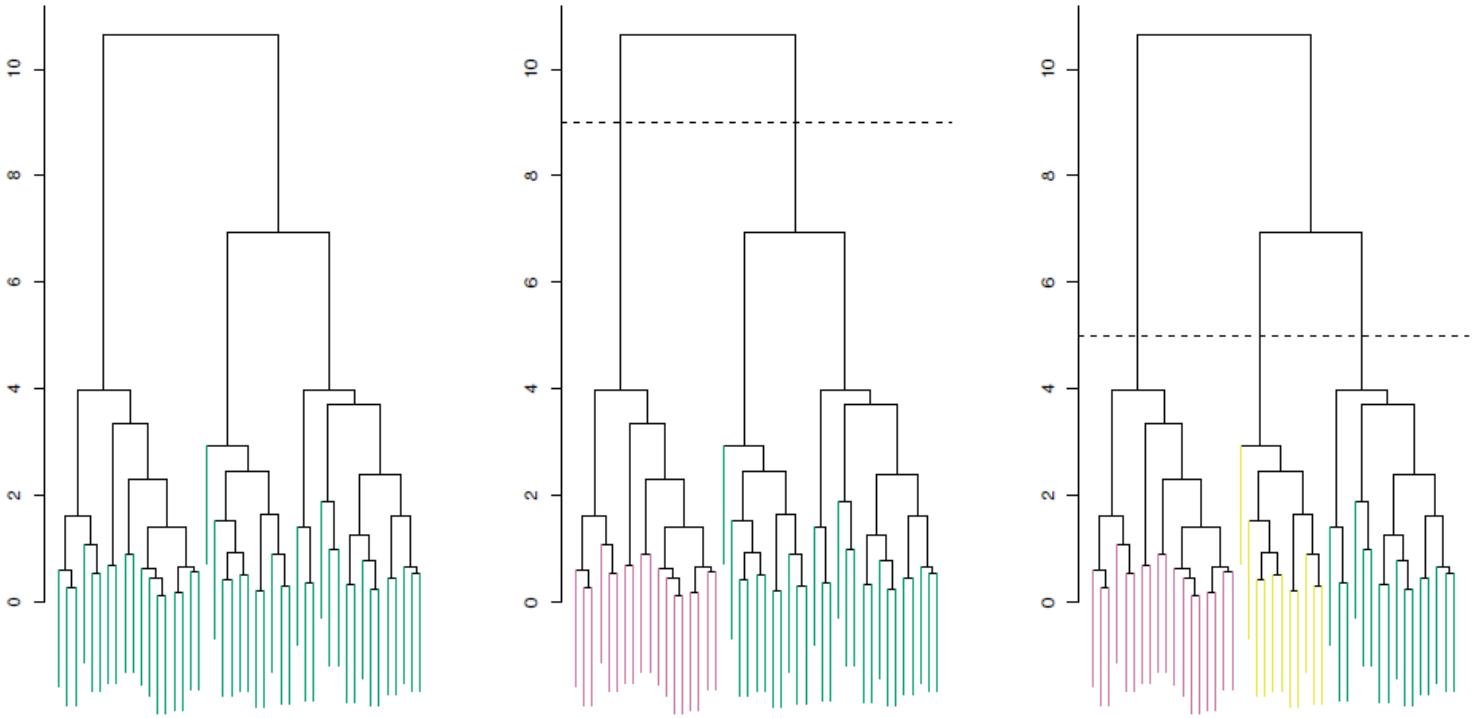
Interpreting a Dendogram



Interpreting a Dendogram

- Bottom of the tree - leaf for each observation.
- As we move up the tree, some leaves begin to fuse into branches: these are observations that are similar to each other.
- The lower in the tree fusions occur, the more similar the groups of observations are to each other.
- Observations that fuse near the top of the tree, can be quite different.
- Height of fusions indicate how similar objects are.
- Horizontal axis does not indicate how similar objects are - just the vertical.

Interpreting a Dendogram



Interpreting a Dendogram

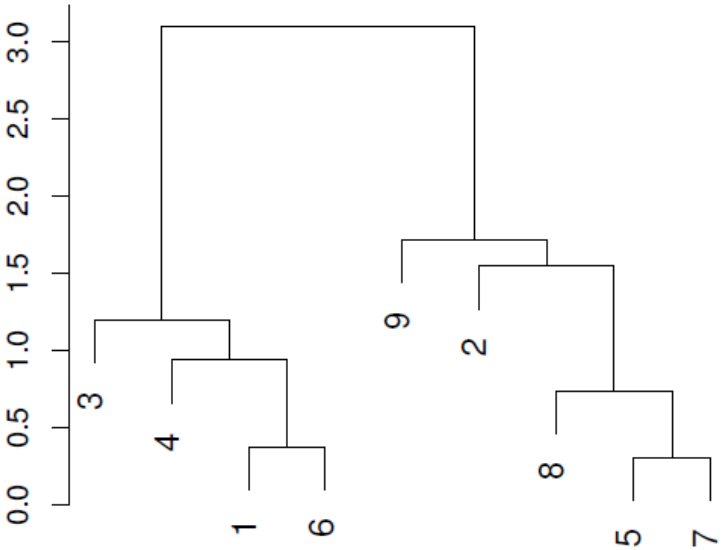
Details of previous figure

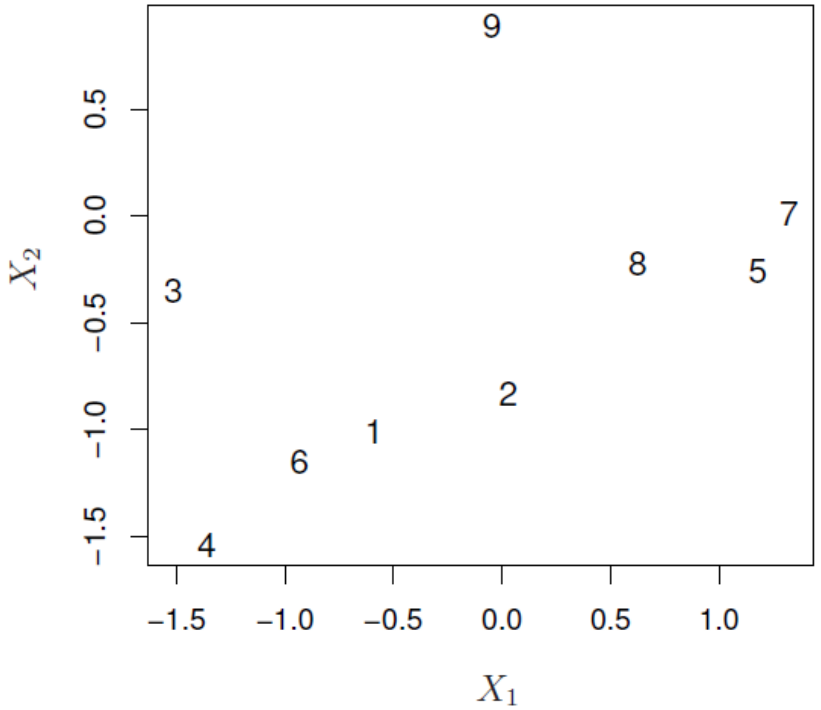
- Left: Dendrogram obtained from hierarchically clustering, with complete linkage and Euclidean distance.
- Center: The dendrogram from the left-hand panel, cut at a height of 9 (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.
- Right: The dendrogram from the left-hand panel, now cut at a height of 5. This cut results in three distinct clusters, shown in different colors. Note that the colors were not used in clustering, but are simply used for display purposes in this figure.

Hierarchical Clustering Algorithm

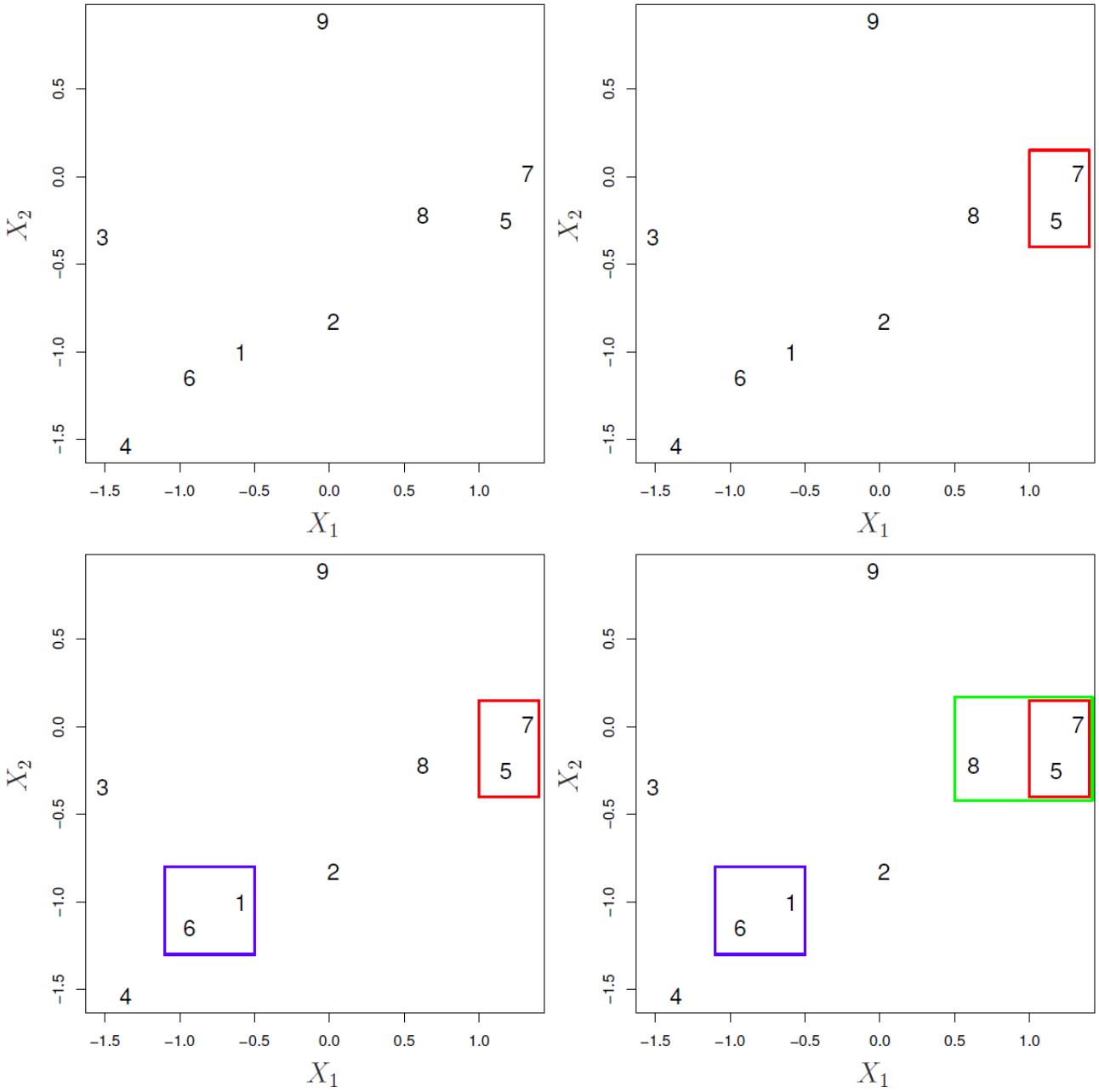
- Begin with every observation representing a singleton cluster.
- At each step, merge two "closest" clusters into one cluster and reduce the number of clusters by one.
- Need a measure of dissimilarity between two clusters - called **linkages**.
- Dissimilarity between cluster G and cluster H: $d(G, H)$, function of the set of pairwise dissimilarities d_{ij}, point i is in G and point j is in H.

Hierarchical Clustering Algorithm





Hierarchical Clustering Algorithm



Linkages

- Linkages - Measure of dissimilarity between two sets of objects that determine how two set of objects are merged.

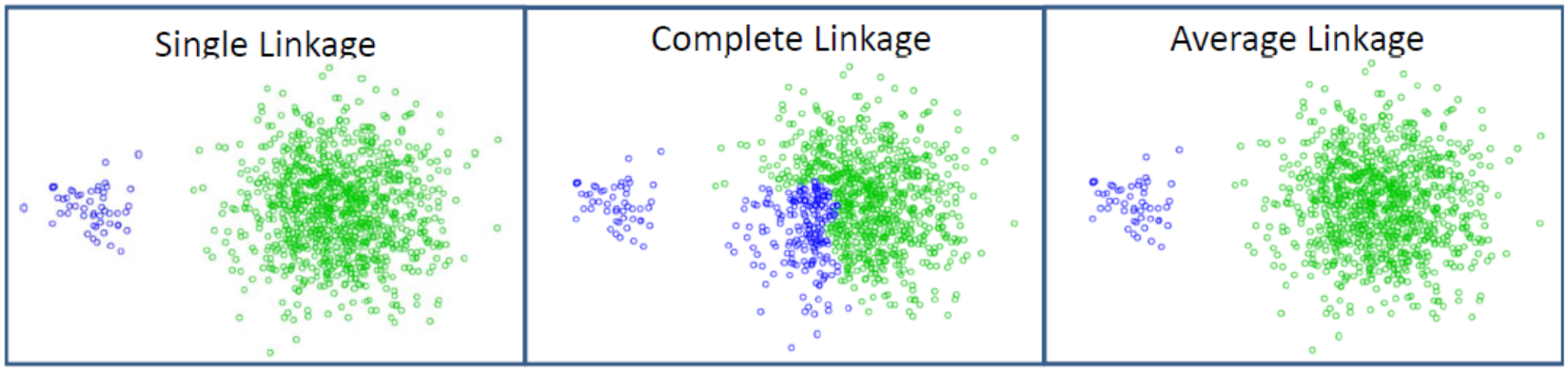
Major Types:

- Single linkage.
- Complete linkage.
- Average Linkage.
- Centroid Linkage.
- Ward's Linkage.

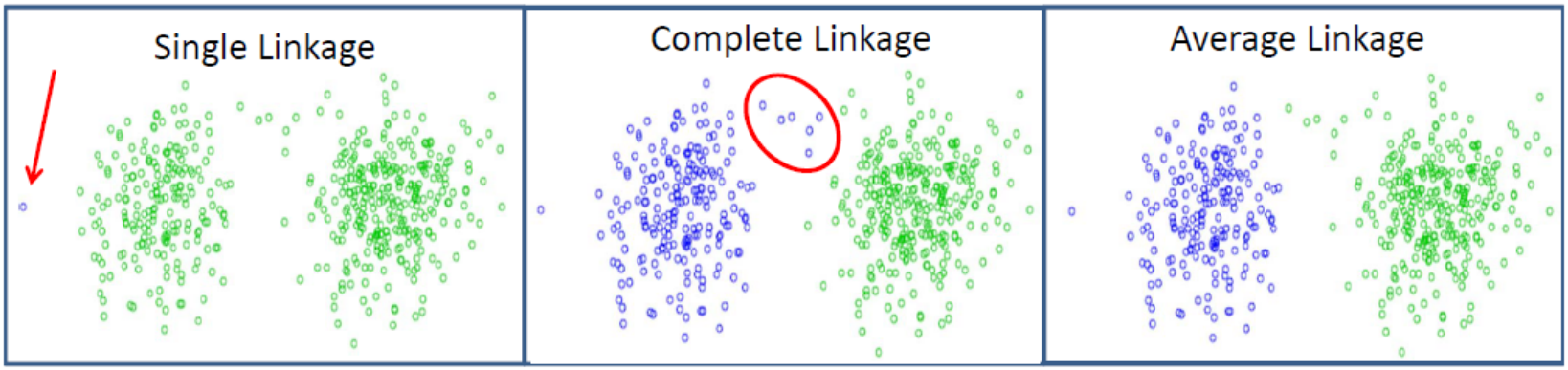
Linkages

Linkage	Description
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the **largest** of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the **smallest** of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the **average** of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable inversions.
Ward	This method does not directly define a measure of distance between two points or clusters. It is an ANOVA based approach. One-way univariate ANOVAs are done for each variable with groups defined by the clusters at that stage of the process. At each stage, two clusters merge that provide the smallest increase in the combined error sum of squares.

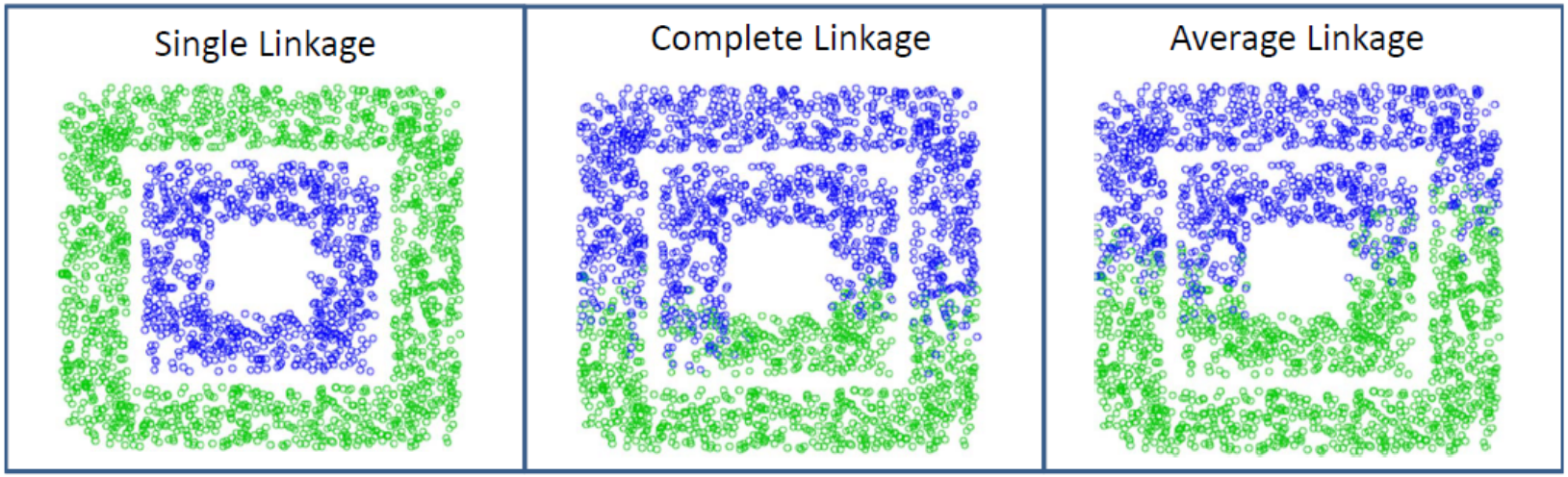
Linkage Examples



Linkage Examples



Linkage Examples



Hierarchical Clustering - Summary

Strengths:

- Simple/intuitive.
- Visualization.
- Family of possible clusterings (nested).
- Extremely popular!!

Weaknesses:

- Local Solution.
- Unstable Solution.
- Depends heavily on type of linkage.
- No optimization criterion - purely algorithmic.

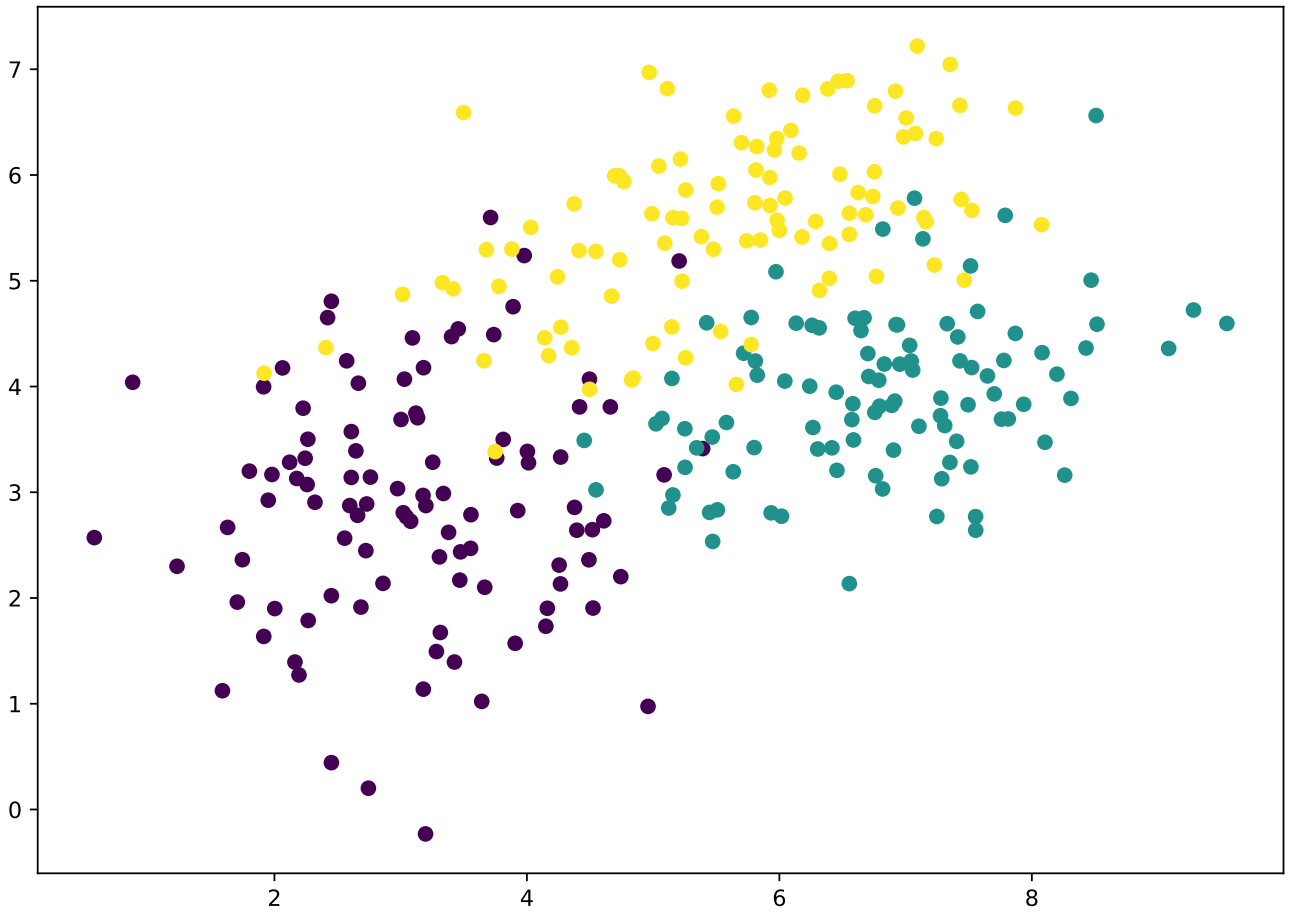
Python: K-Means

- The sklearn function `Kmeans()` performs K-means clustering
- Let's conduct a simulation study

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(123)
n = 300 # 100 is int(n/3)
mu1 = [3,3]
mu2 = [7,4]
mu3 = [5.5,5.5]
Sig = [[1, 0.5], [0.5, 1]]
x1=np.full((100, 2), mu1[0]) +np.random.normal(size = (100, 2))
xx=np.random.normal(size = (100, 2))  # 2 columns of standard normal
x2=np.vstack([mu2] * 100) +np.dot(xx, np.linalg.cholesky(Sig))
#Cholesky matrix transforms uncorrelated variables into
#variables whose variances and covariances are given by Sig
xx=np.random.normal(size = (100, 2))
x3=np.vstack([mu3] * 100) +np.dot(xx, np.linalg.cholesky(Sig))
X=np.vstack((x1,x2,x3))
y=np.hstack(([0]*100,[1]*100,[2]*100))
```

Python: K-Means

```
import matplotlib.pyplot as plt
plt.scatter(X[:,0], X[:,1],c=y)
plt.show()
```



Python: K-Means

- We now perform K-means clustering with K = 3:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 3, random_state = 123).fit(X)
```

```
## C:\Users\xzhang2\AppData\Local\Programs\Python\PYTHON~2\lib\site-package
##   warnings.warn(
```

- The centers of the 3 clusters

```
print(kmeans.cluster_centers_)
```

```
## [[3.14017319 2.99164475]
##  [6.77052936 3.86791152]
##  [5.90133697 5.8023215 ]]
```

- The cluster assignments of the 300 observations are contained in `kmeans.labels_`:

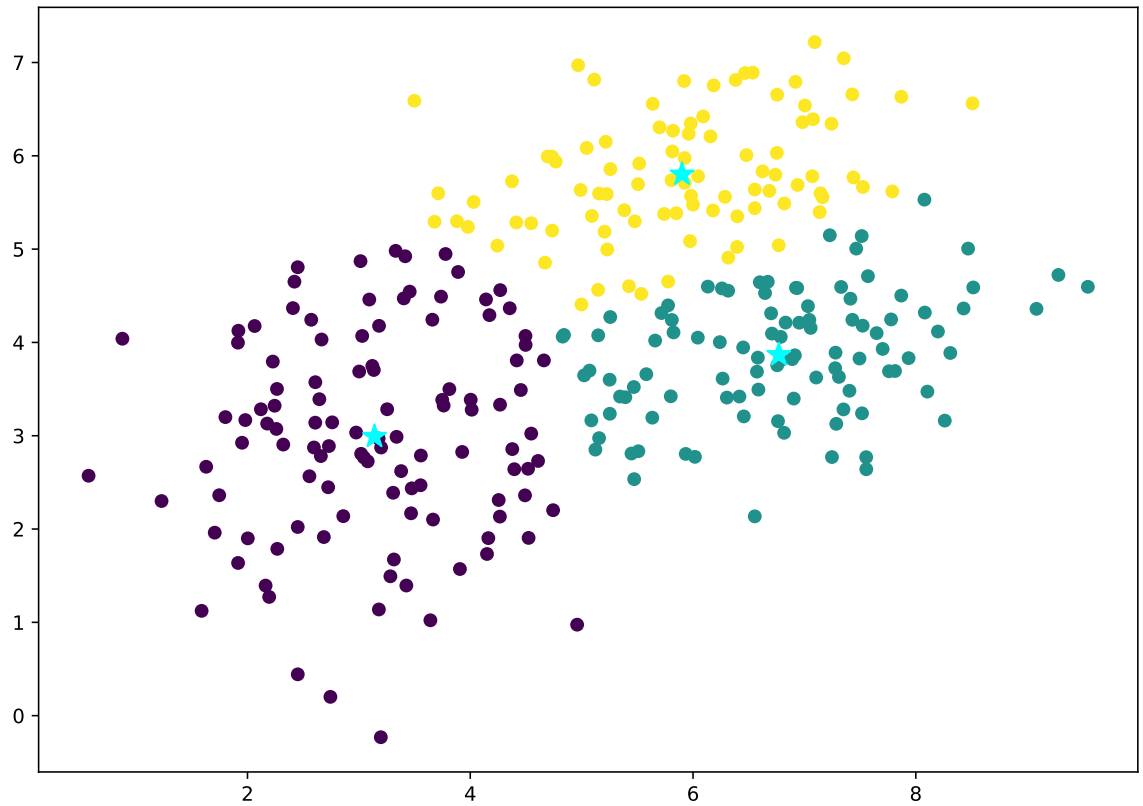
```
print(kmeans.labels_)
```

```
## [0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0
##  0 0 0 0 0 0 0 1 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 2 1 1
##  1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
```

Python: K-Means

- Plot of the clustering

```
plt.scatter(X[:,0], X[:,1], c = kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:, 0],kmeans.cluster_centers_[:, 1],
marker = '*', s =150, color = 'cyan', label = 'Centers')
plt.show()
```



Python: K-Means

- Here the observations can be easily plotted because they are two-dimensional.
- We know the true clustering information, so we can check the prediction accuracy.

```
from sklearn.metrics import accuracy_score, confusion_matrix
print(confusion_matrix(y, kmeans.labels_).T)
```

```
## [[95  2 13]
##  [ 2 90  8]
##  [ 3  8 79]]
```

```
print(accuracy_score(y, kmeans.labels_))
```

```
## 0.88
```

- In this example, we knew that there really were three clusters because we generated the data. However, for real data, in general we do not know the true number of clusters.

Python: K-Means

- To run the `Kmeans()` function in python with multiple initial cluster assignments, we use the `n_init` argument (default: 10). If a value of `n_init` greater than one is used, then K-means clustering will be performed using multiple random assignments, and the `Kmeans()` function will report only the best results. Here we compare using `n_init = 1`:

```
km_out_single_run = KMeans(n_clusters = 4, n_init = 1, random_state = 123).fit(X)
km_out_single_run.inertia_
```

```
## 386.8845850067187
```

- to `n_init = 50`:

```
km_out_single_run = KMeans(n_clusters = 4, n_init = 50, random_state = 123).fit(X)
km_out_single_run.inertia_
```

```
## 386.51676454531844
```

- Note that `.inertia_` is the total within-cluster sum of squares, which we seek to minimize by performing K-means clustering.
- It is generally recommended to always run K-means clustering with a large value of `n_init`, such as 20 or 50 to avoid getting stuck in an undesirable local optimum.
- When performing K-means clustering, in addition to using multiple initial cluster assignments, it is also important to set a random seed using the random_state parameter. This way, the initial cluster assignments can be replicated, and the K-means output will be fully *reproducible*.

Python: Hierarchical Clustering

- The `linkage()` function from `scipy` implements several clustering functions in python. In the following example we use the data from the previous section to plot the hierarchical clustering dendrogram using complete, single, average, centroid and Ward's linkage clustering, with Euclidean distance as the dissimilarity measure.
  - https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html
  - We begin by clustering observations using complete linkage:

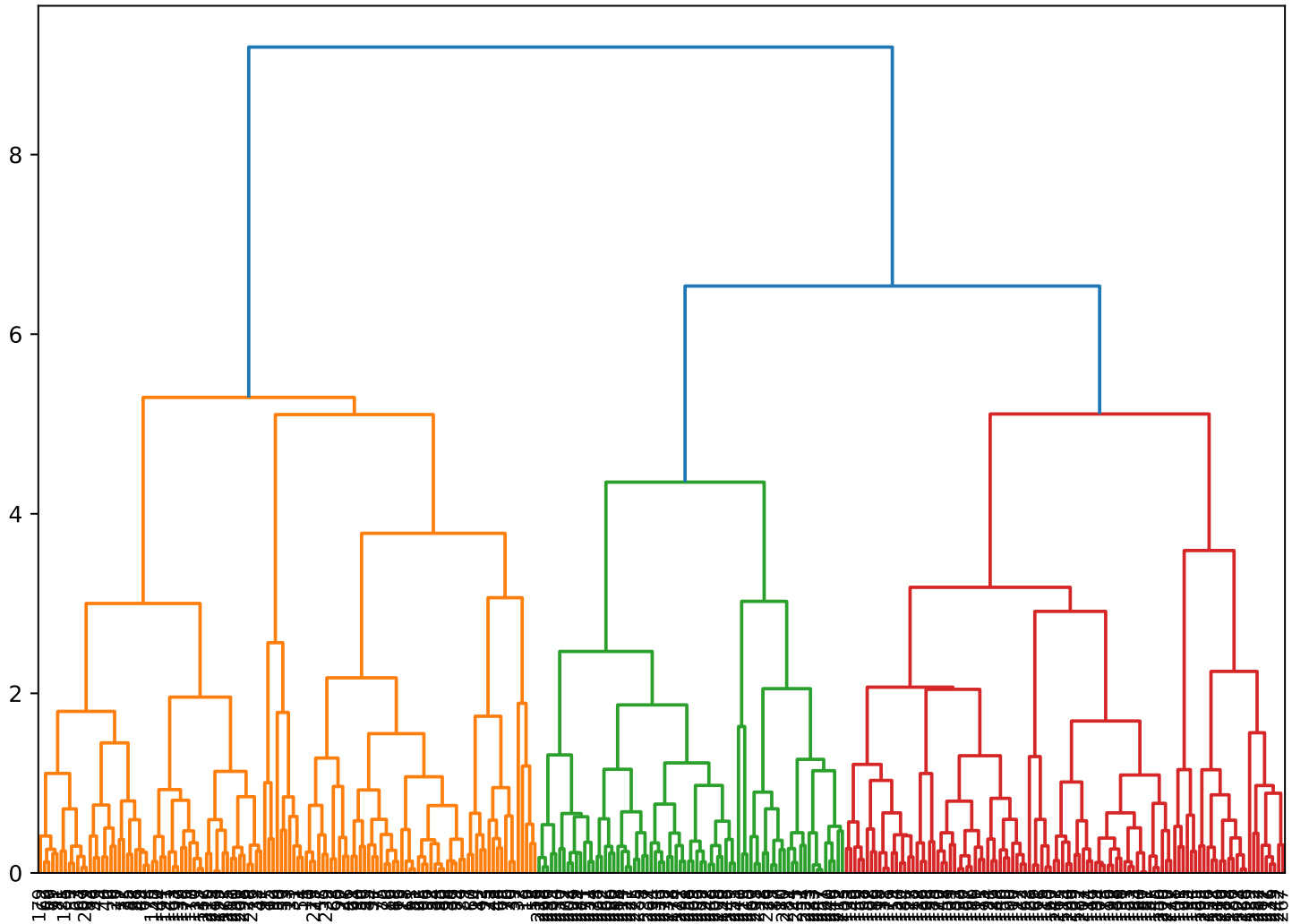
```
from scipy.cluster.hierarchy import dendrogram, linkage
hc_complete = linkage(X, "complete")
```

Python: Hierarchical Clustering

- We plot the dendrograms obtained using the usual `dendrogram()` function. The numbers at the bottom of the plot identify each observation:
  - https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html

```
from scipy.cluster.hierarchy import dendrogram
dendrogram(
    hc_complete,
    leaf_rotation=90.,  # rotates the x axis labels
    leaf_font_size=8.,  # font size for the x axis labels
)
plt.show()
```

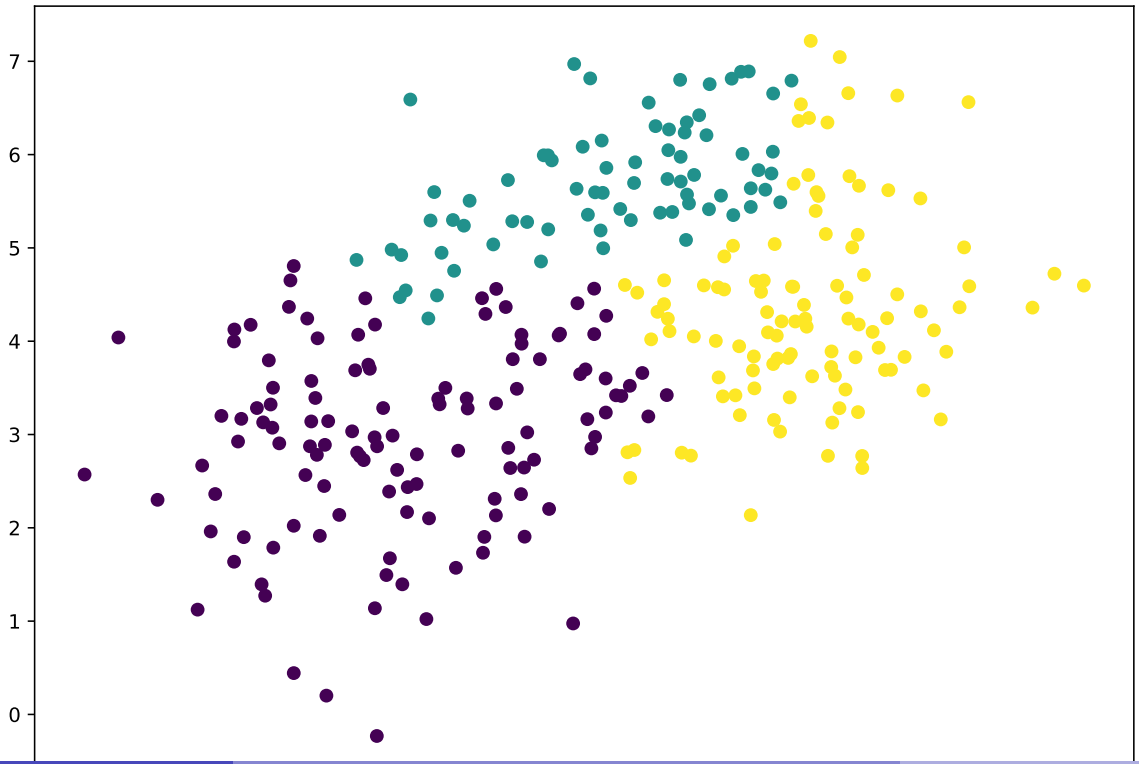
Python: Hierarchical Clustering



Python: Hierarchical Clustering

- To determine the cluster labels for each observation associated with a given cut of the dendrogram, we can use the `cut_tree()` function in `scipy`.

```
from scipy.cluster.hierarchy import cut_tree
clustering1=cut_tree(hc_complete, n_clusters = 3)
plt.scatter(X[:,0], X[:,1], c = clustering1)
plt.show()
```



Python: Hierarchical Clustering

```
print(confusion_matrix(y, clustering1).T)
```

```
## [[93 14 13]
##  [ 7  2 65]
##  [ 0 84 22]]
```

```
print(accuracy_score(y, clustering1))
```

```
## 0.39
```

Python: Hierarchical Clustering

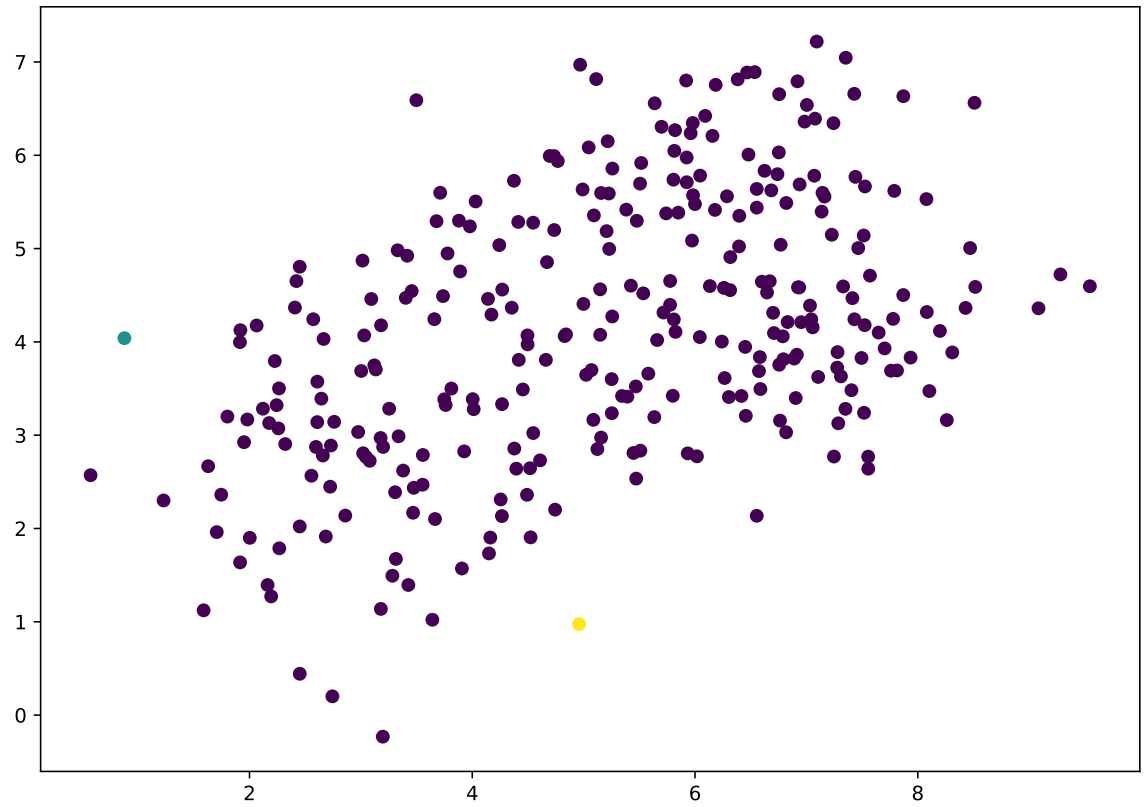
- single linkage

```
hc_single = linkage(X, "single")
```

```
clustering2=cut_tree(hc_single, n_clusters = 3)
plt.scatter(X[:,0], X[:,1], c = clustering2)
plt.show()
```

Python: Hierarchical Clustering

- single linkage



Python: Hierarchical Clustering

- single linkage

```
print(confusion_matrix(y, clustering2).T)
```

```
## [[ 98 100 100]
##  [  1   0   0]
##  [  1   0   0]]
```

```
print(accuracy_score(y, clustering2))
```

```
## 0.32666666666666666
```

Python: Hierarchical Clustering

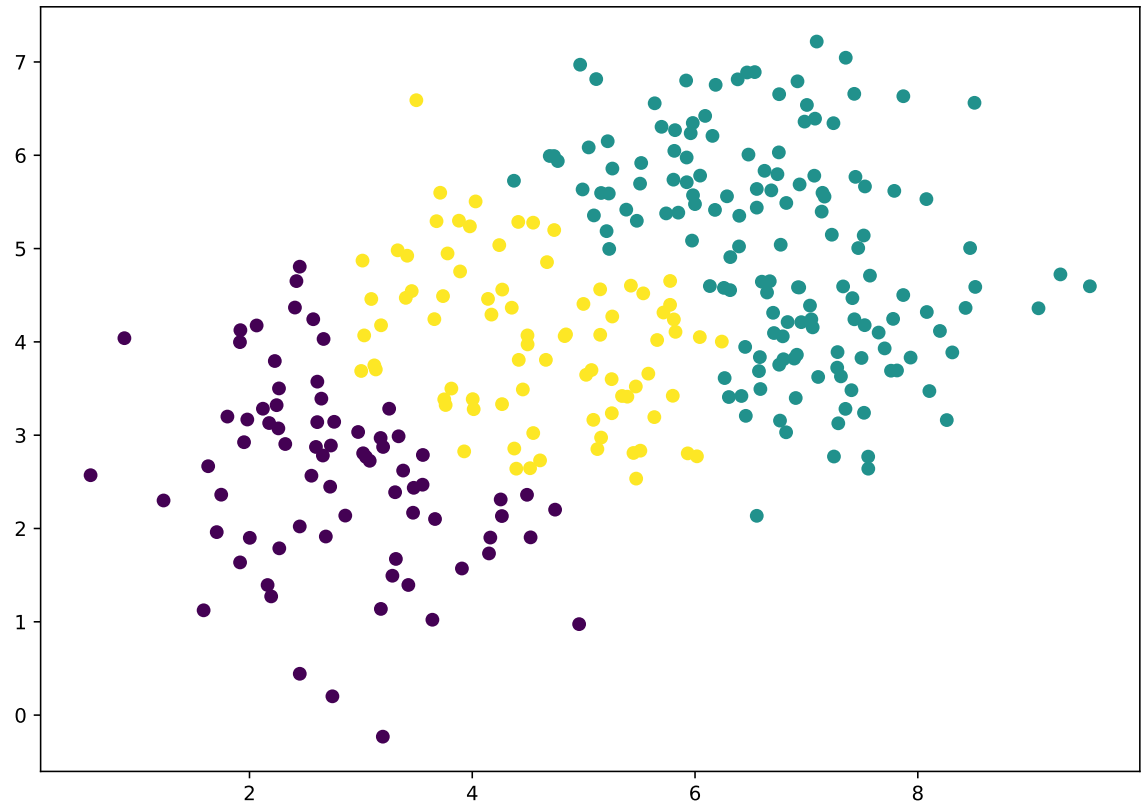
- average linkage

```
hc_average = linkage(X, "average")
```

```
clustering3=cut_tree(hc_average, n_clusters = 3)
plt.scatter(X[:,0], X[:,1], c = clustering3)
plt.show()
```

Python: Hierarchical Clustering

- average linkage



Python: Hierarchical Clustering

- average linkage

```
print(confusion_matrix(y, clustering3).T)
```

```
## [[72  0  2]
##  [ 1 74 70]
##  [27 26 28]]
```

```
print(accuracy_score(y, clustering3))
```

```
## 0.58
```

Python: Hierarchical Clustering

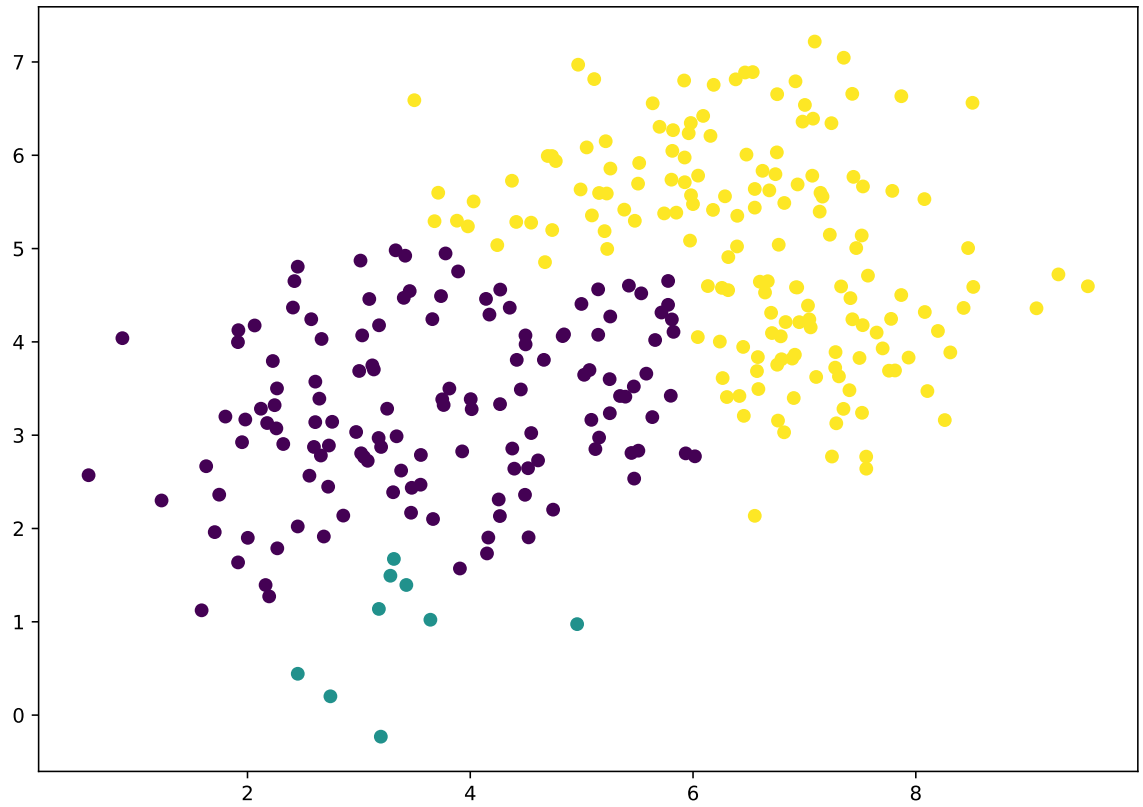
- Centroid linkage

```
hc_centroid = linkage(X, "centroid")
```

```
clustering4=cut_tree(hc_centroid, n_clusters = 3)
plt.scatter(X[:,0], X[:,1], c = clustering4)
plt.show()
```

Python: Hierarchical Clustering

- Centroid linkage



Python: Hierarchical Clustering

- Centroid linkage

```
print(confusion_matrix(y, clustering4).T)
```

```
## [[88 24 21]
##  [ 9  0  0]
##  [ 3 76 79]]
```

```
print(accuracy_score(y, clustering4))
```

```
## 0.5566666666666666
```

Python: Hierarchical Clustering

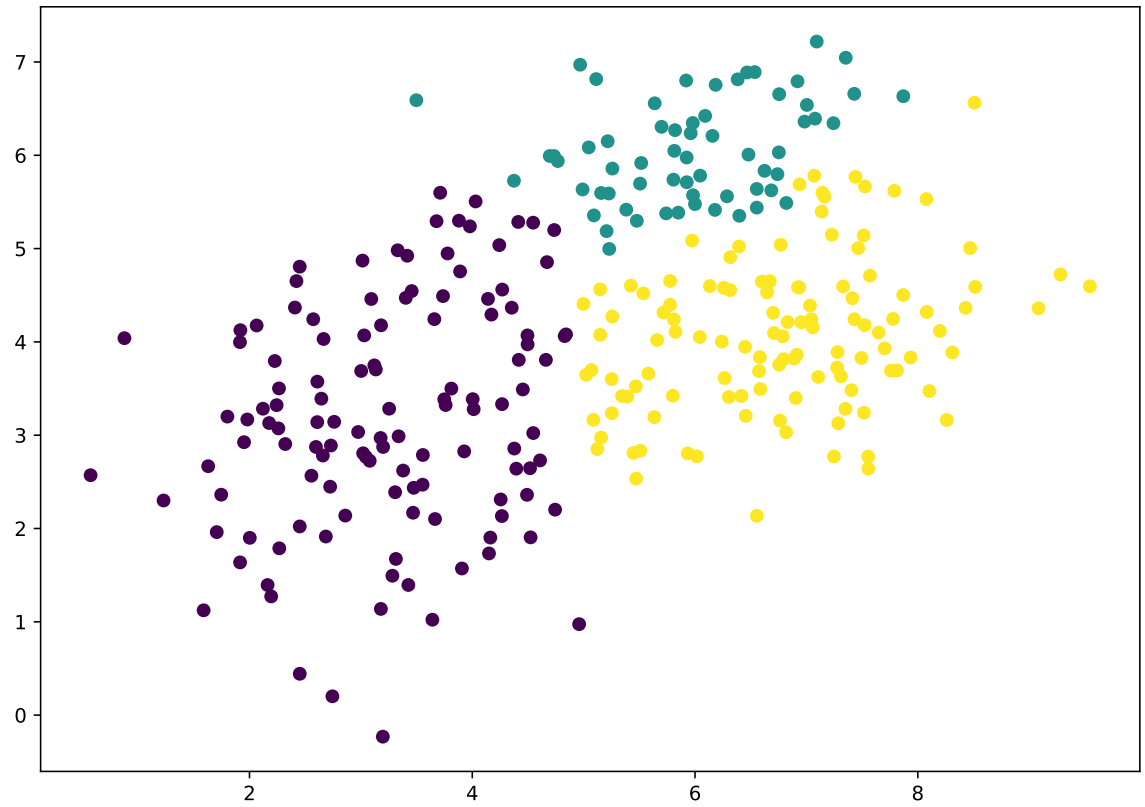
- Ward's linkage

```
hc_ward = linkage(X, "ward")
```

```
clustering5=cut_tree(hc_ward, n_clusters = 3)
plt.scatter(X[:,0], X[:,1], c = clustering5)
plt.show()
```

Python: Hierarchical Clustering

- Ward's linkage



Python: Hierarchical Clustering

- Ward's linkage

```
print(confusion_matrix(y, clustering5).T)
```

```
## [[97  2 23]
##  [ 1  1 60]
##  [ 2 97 17]]
```

```
print(accuracy_score(y, clustering5))
```

```
## 0.3833333333333336
```

Python: Hierarchical Clustering

- We can apply different distances for the above linkages specified by the `metric` argument in the function `linkage()`.
- The default method is Euclidean distance: `metric='euclidean'`
  - For more measures, see https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html

License