Clustering

Samir Kanaan-Izquierdo

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Course 2019/20

Contents of this unit

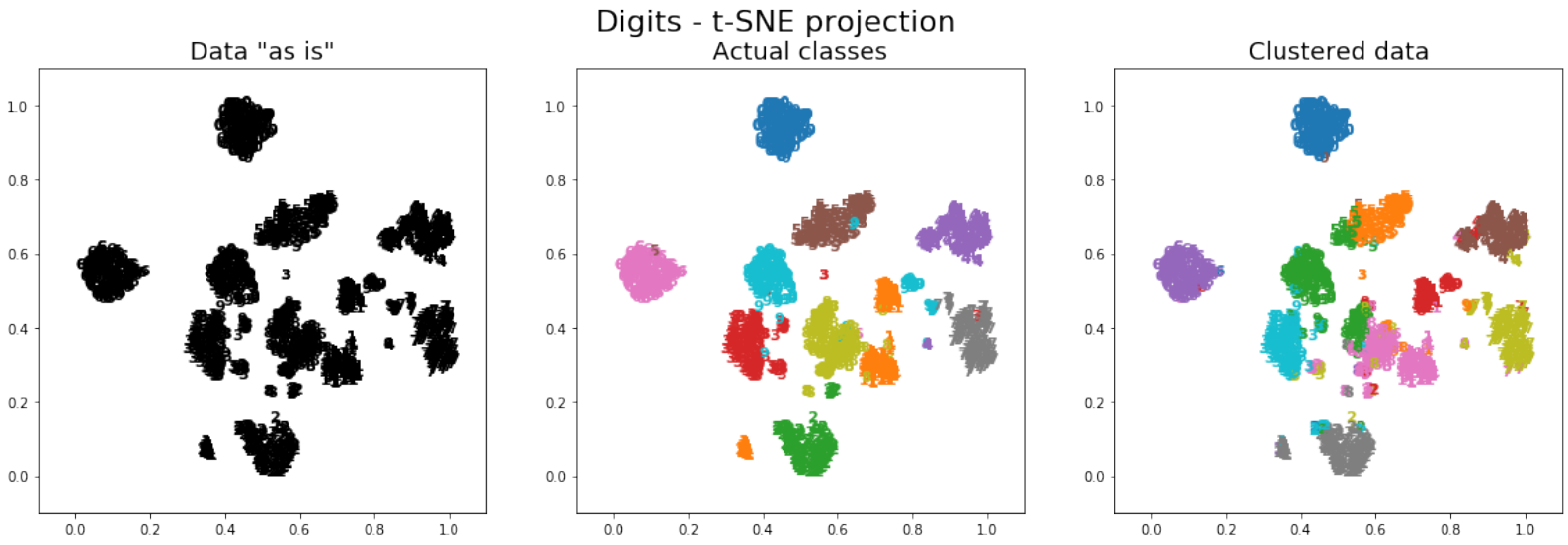
Outline

Clustering

1. Starting point: unlabeled data of which maybe we know nothing
2. Goal: find some structure in the data, give it some meaning
3. Strategy: create groups or **clusters** of similar points

Example: the digits dataset.

Clustering the data



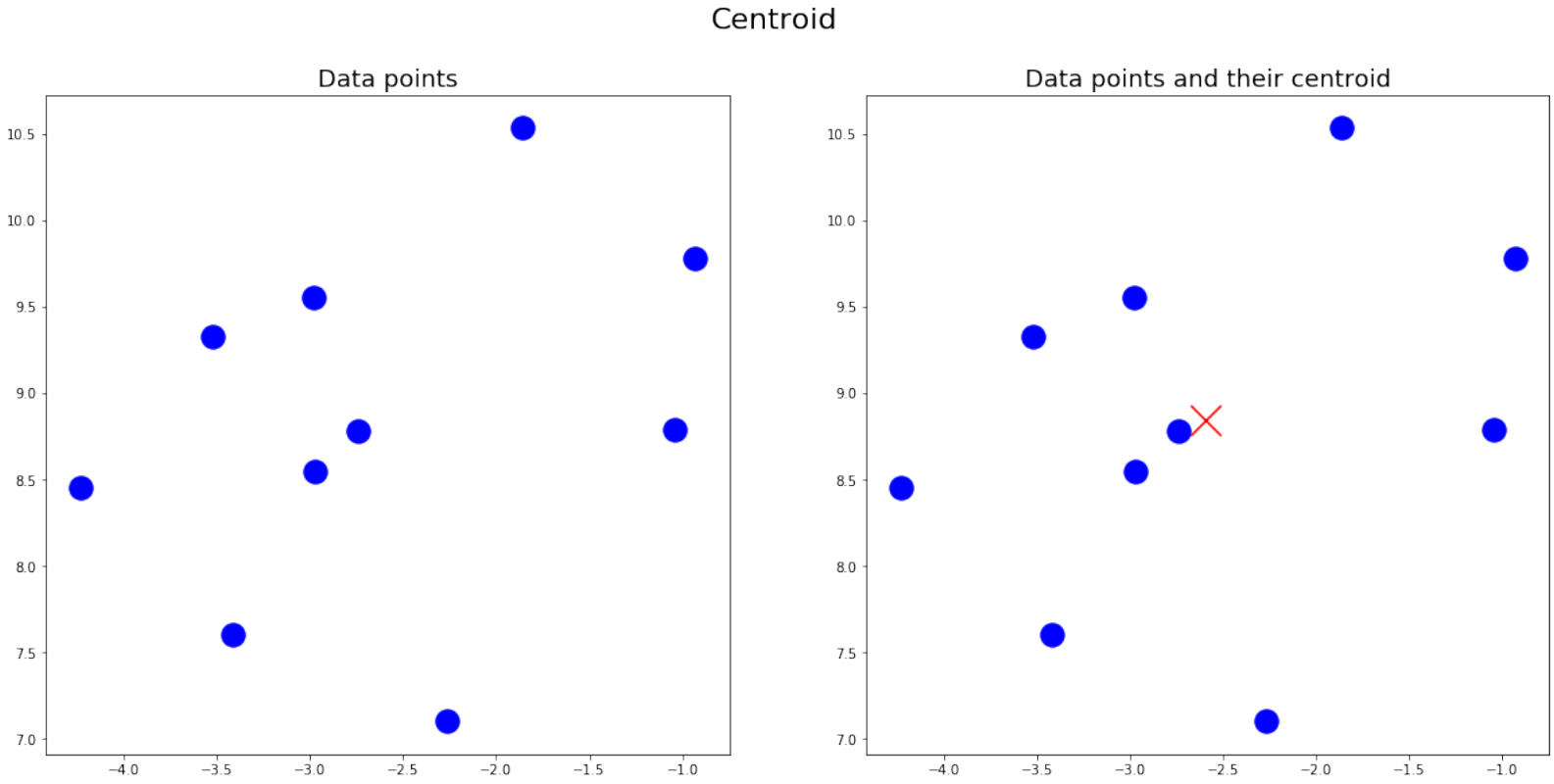
Applications

- Discover useful/unknown structures in the data
- Establish classes based on evidence (attributes), not on human experts
- Classify new observations (based on previous clustering)
- Group similar observations even if there are no actual classes (customers, situations, movies...)
- Areas: finance, medicine, sales, population, document filters, risk assessment...

Unsupervised data is cheap! Labelling data is expensive!

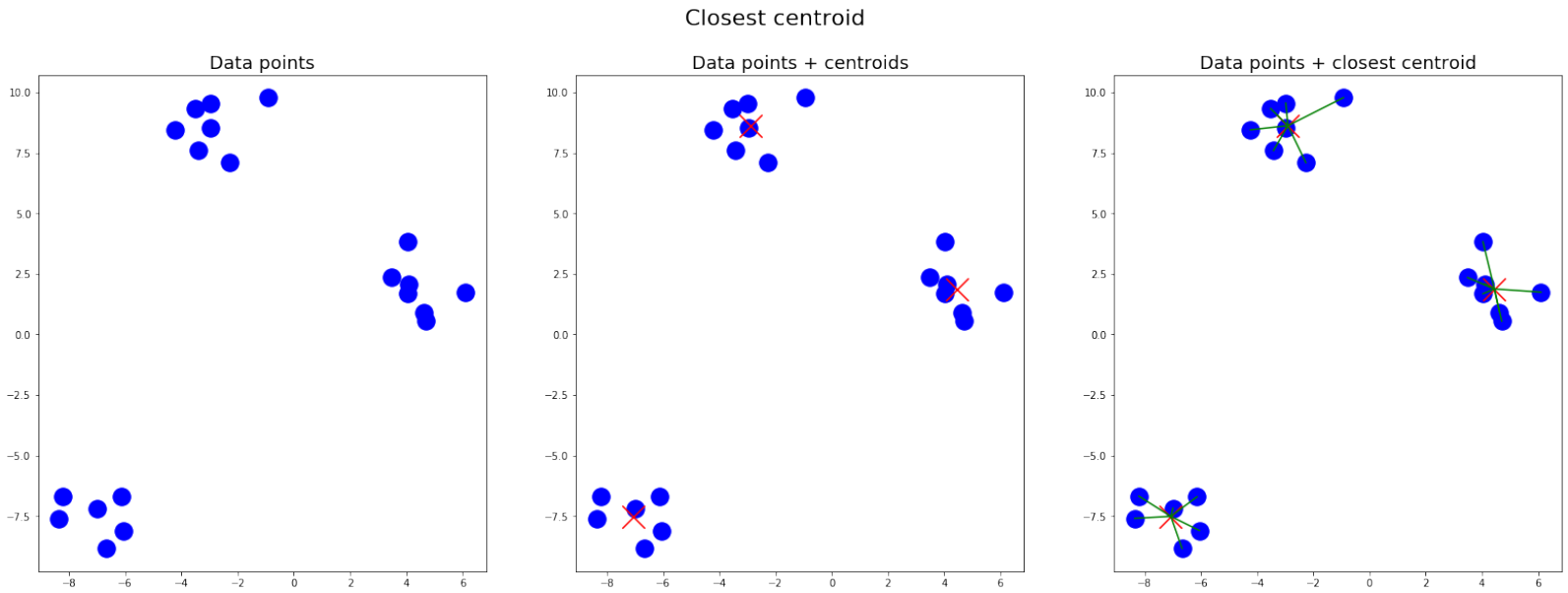
Outline

K-Means concepts: centroid



It is called **centroid** because it is not one of the data points, only their *computed* (geometrical) center.

K-Means concepts: closest centroid



The data points are assigned to the closest centroid; therefore the centroids arrange the creation of point clusters.

K-Means algorithm

Input: n points, k number of clusters

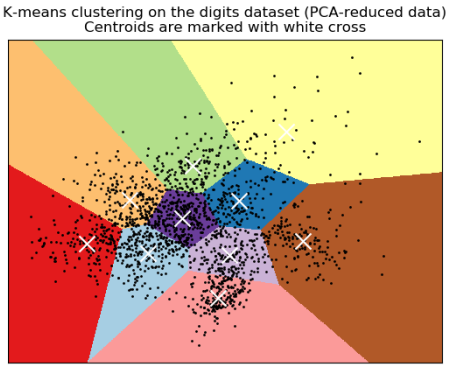
1. Randomly choose k data points as initial centroids
2. Assign the data points to their closest centroid (cluster)
3. Compute the center of each cluster (new k centroids)
4. Go back to step 2 until the assignment does not change or a number of iterations is reached

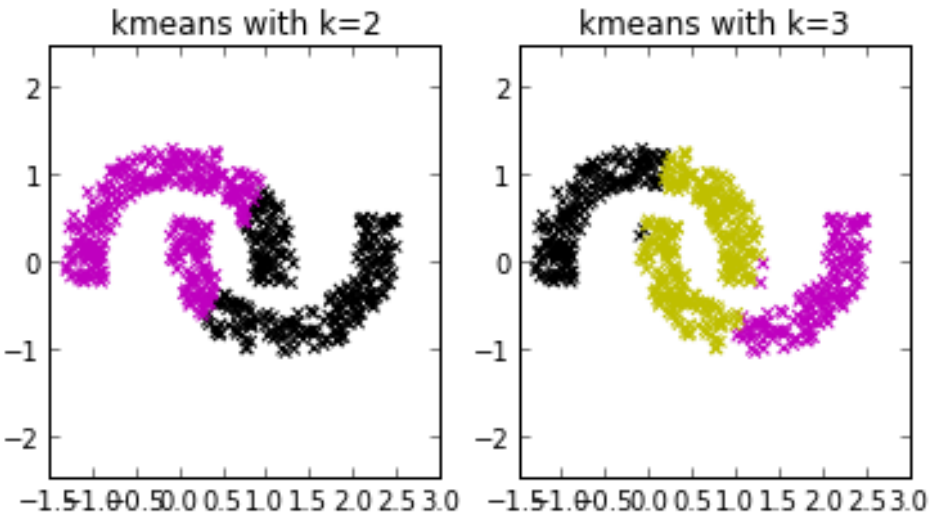
Ouput: k centroids (points), vector of cluster assignment of each input point (n integers $\in [0, k-1]$)

K-Means features

K-Means is (probably) the most used clustering method.

- Fast
- Distance-based (Voronoi partition)
- Fails in non-convex clusters
- Random solutions
- Requires the user to guess the k parameter





K-Means usage and improvements


```
from sklearn.cluster import KMeans
mykm = KMeans(n_clusters=2).fit(X)
mykm.labels_
mykm.cluster_centers_
```

- random_state=None: reproducible results
- init='k-means++': improved selection of initial centroids
- n_init=10: repeat K-Means to get more robust results
- max_iter=300: control the number of iterations

K-Means: exercise

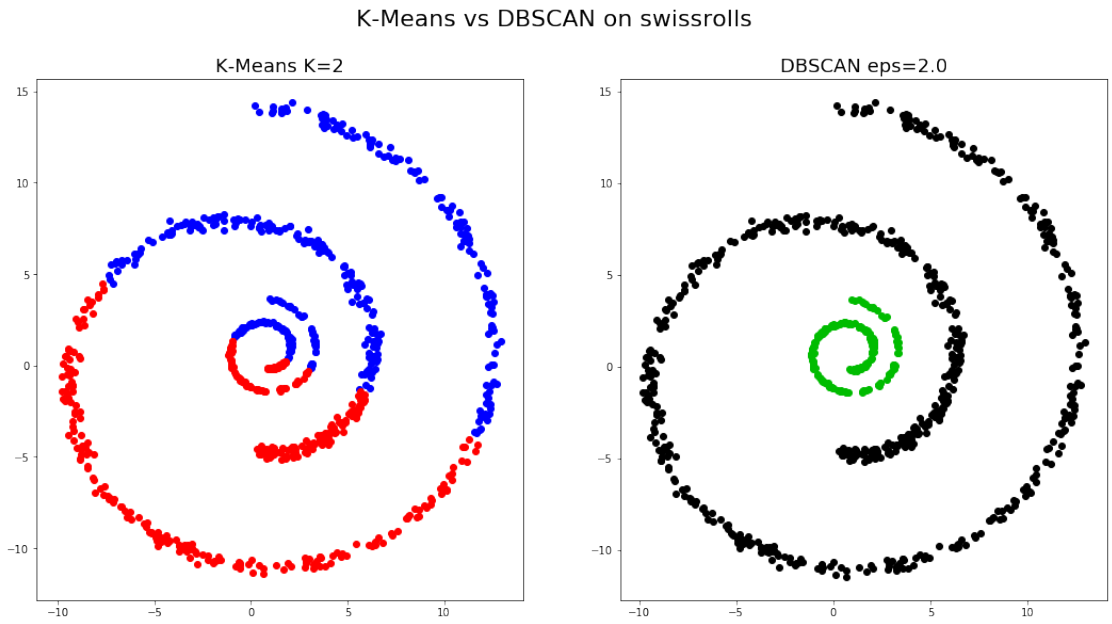
Load the *Iris* dataset from sklearn. Do the following activities:

- Decide if normalization is advisable
- Apply K-Means with K=2
- Plot the data on 2D using a dimensionality reduction method, with the cluster assignment as point color.
- Repeat with K-Means and K=3
- Compare the clusters obtained

Outline

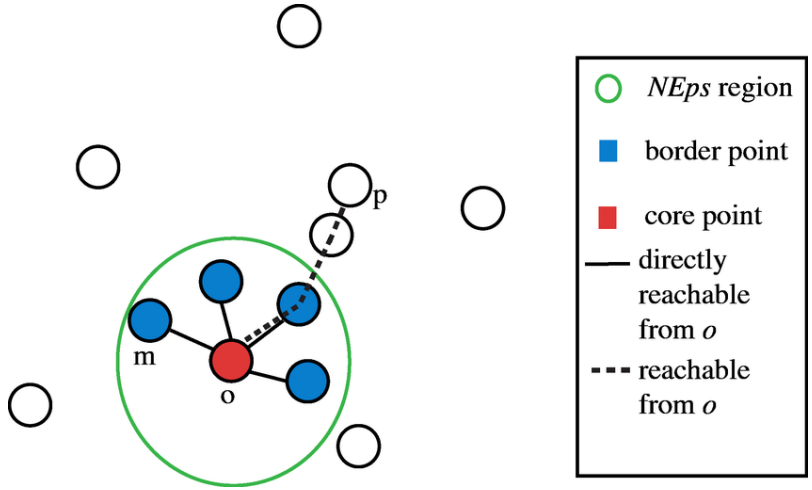
DBSCAN: features

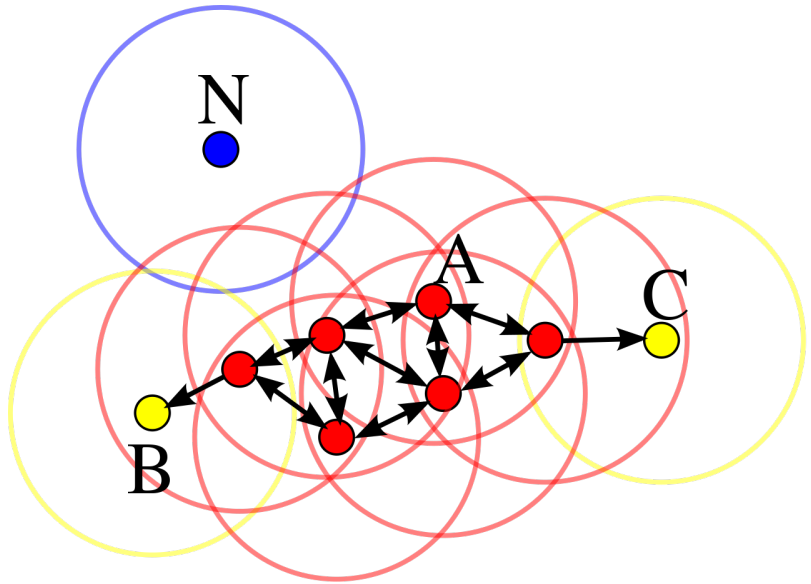
Goal: be able to find non-convex clusters.



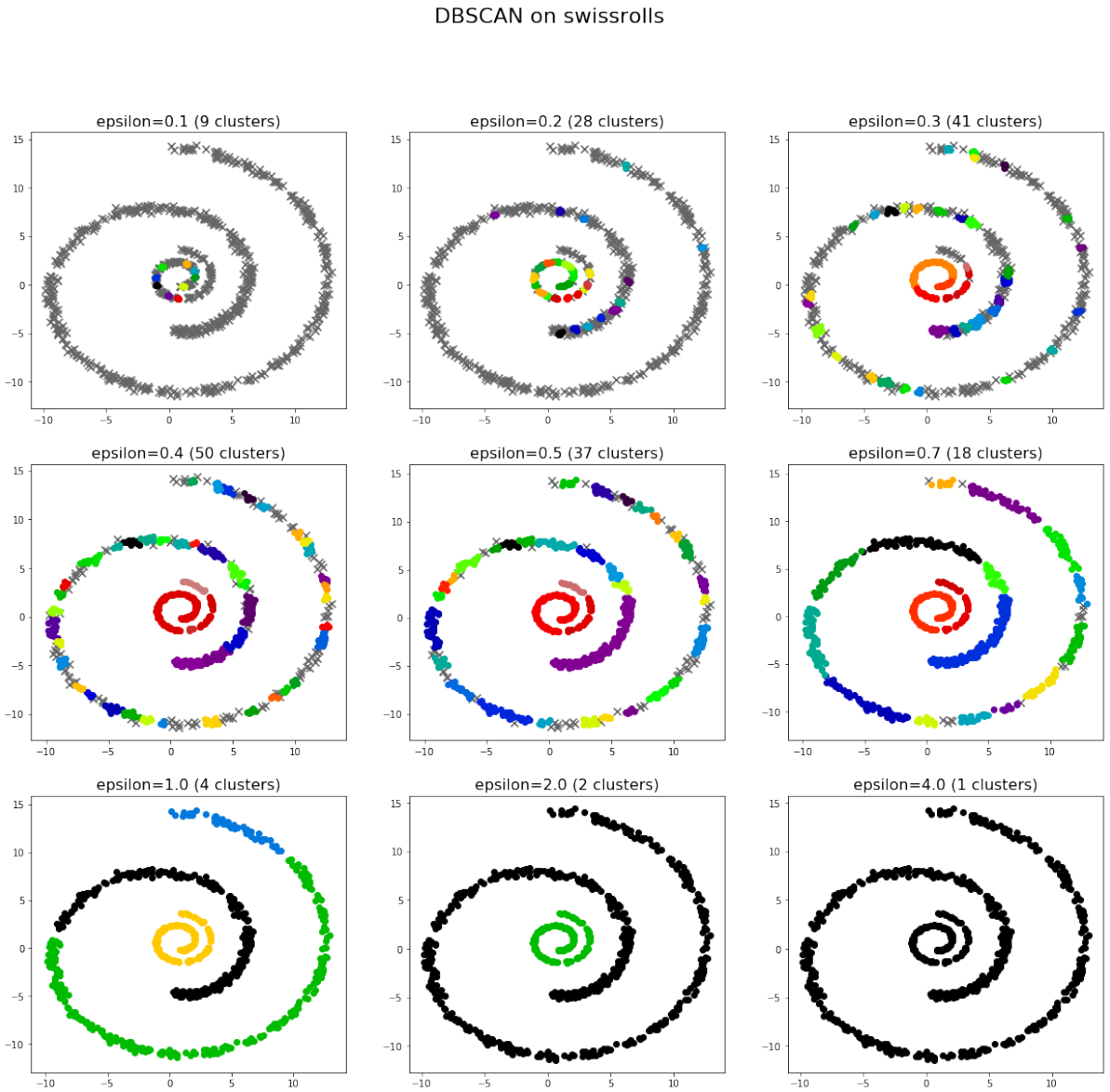
- Message-passing or jumps from point to point
- Parameter: ϵ instead of K
- Detects outliers

DBSCAN: how it works (ϵ)





Effect of the ϵ parameter



DBSCAN in Python


```
from sklearn.cluster import DBSCAN
mydbs = DBSCAN(eps=0.5, min_samples=5).fit(X)
mydbs.labels_
```

DBSCAN: exercise

Load the *Iris* dataset from sklearn. Do the following activities:

- Decide if normalization is advisable
- Apply DBSCAN with different values of ϵ
- Plot the data on 2D using a dimensionality reduction method, with the cluster assignment as point color
- Analyze the clusters obtained. Choose the best ϵ for this dataset

Outline

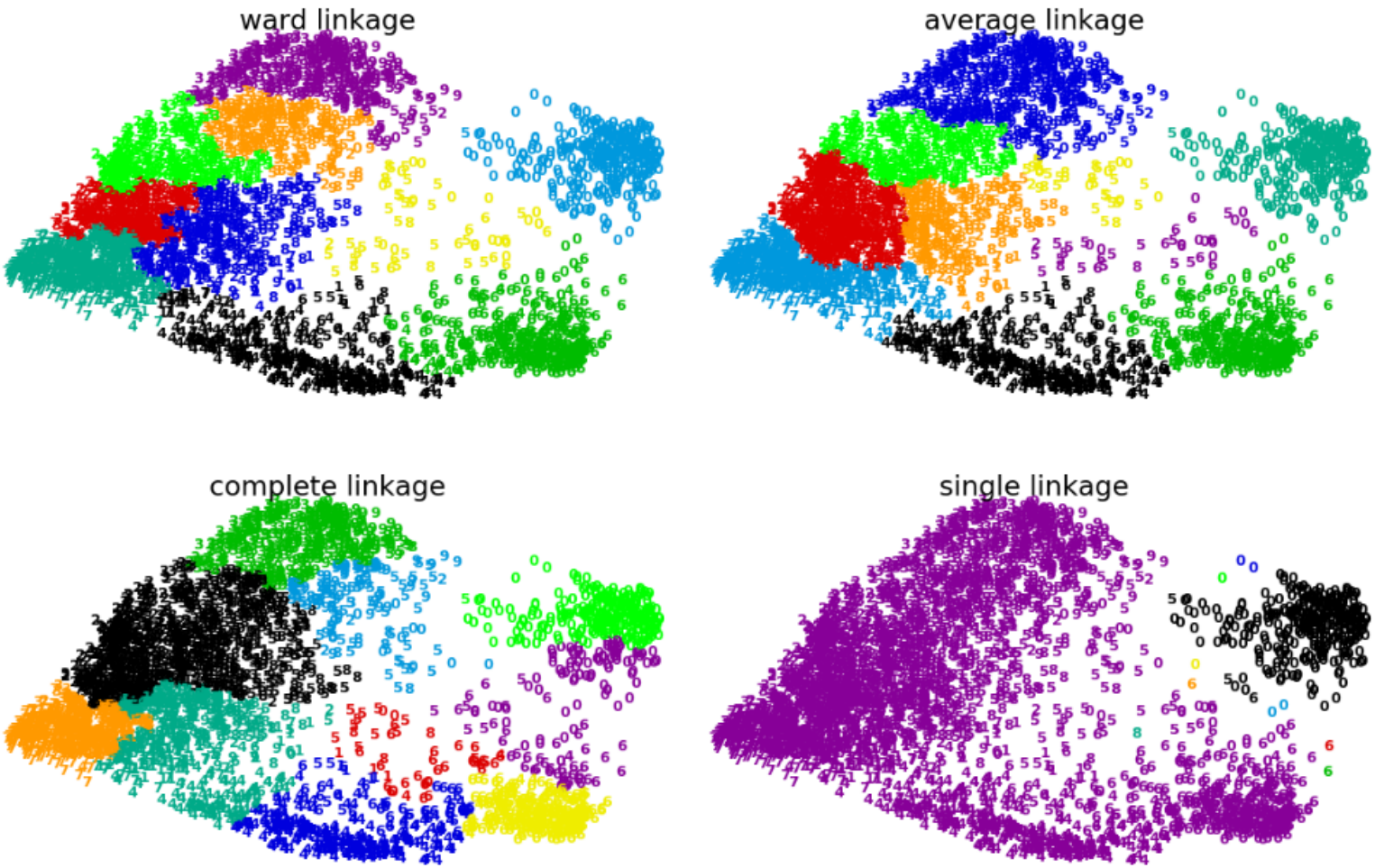
Introduction

Goal: gradually merge the data points into clusters, and the clusters into bigger ones, until all points are merged together (**agglomerative** clustering).

Features:

- Linking strategies
- Provides all granularities of clustering
- Can generate *dendrograms* (tree diagrams) of the clustering process

Effect of the linking strategy



General Explanation of Linkage in Hierarchical Clustering

In hierarchical clustering—particularly agglomerative clustering—we merge clusters step by step. The rule for deciding which clusters to merge is defined by the **linkage** method. The linkage determines how we measure the "distance" between two clusters.

Ward Linkage

- Ward linkage merges the two clusters that produce the **smallest increase** in the **within-cluster variance**.
- Minimizes the increase in the **sum of squared distances** (variance) when forming a new cluster.
- Tends to create **compact, spherical clusters**.

Average Linkage

- Uses the **average distance** between members of the two clusters.

$$\text{Distance}(C_k, C_\ell) = \frac{1}{|C_k| \times |C_\ell|} \sum_{\mathbf{x} \in C_k} \sum_{\mathbf{y} \in C_\ell} d(\mathbf{x}, \mathbf{y})$$

- Balances extremes:
  - Less chaining than single linkage.
  - Less sensitivity to outliers than complete linkage.

Complete Linkage

- Also called **maximum linkage**.
- Uses the **maximum distance** between any point in one cluster and any point in the other:

$$\text{Distance}(C_k, C_\ell) = \max_{\mathbf{x} \in C_k, \mathbf{y} \in C_\ell} d(\mathbf{x}, \mathbf{y})$$

- Produces **compact clusters**, but can be sensitive to outliers, since one outlier can increase the distance significantly.

Single Linkage

- Also called **minimum linkage**.
- Uses the **minimum distance** between any point in one cluster and any point in the other:

$$\text{Distance}(C_k, C_\ell) = \min_{\mathbf{x} \in C_k, \mathbf{y} \in C_\ell} d(\mathbf{x}, \mathbf{y})$$

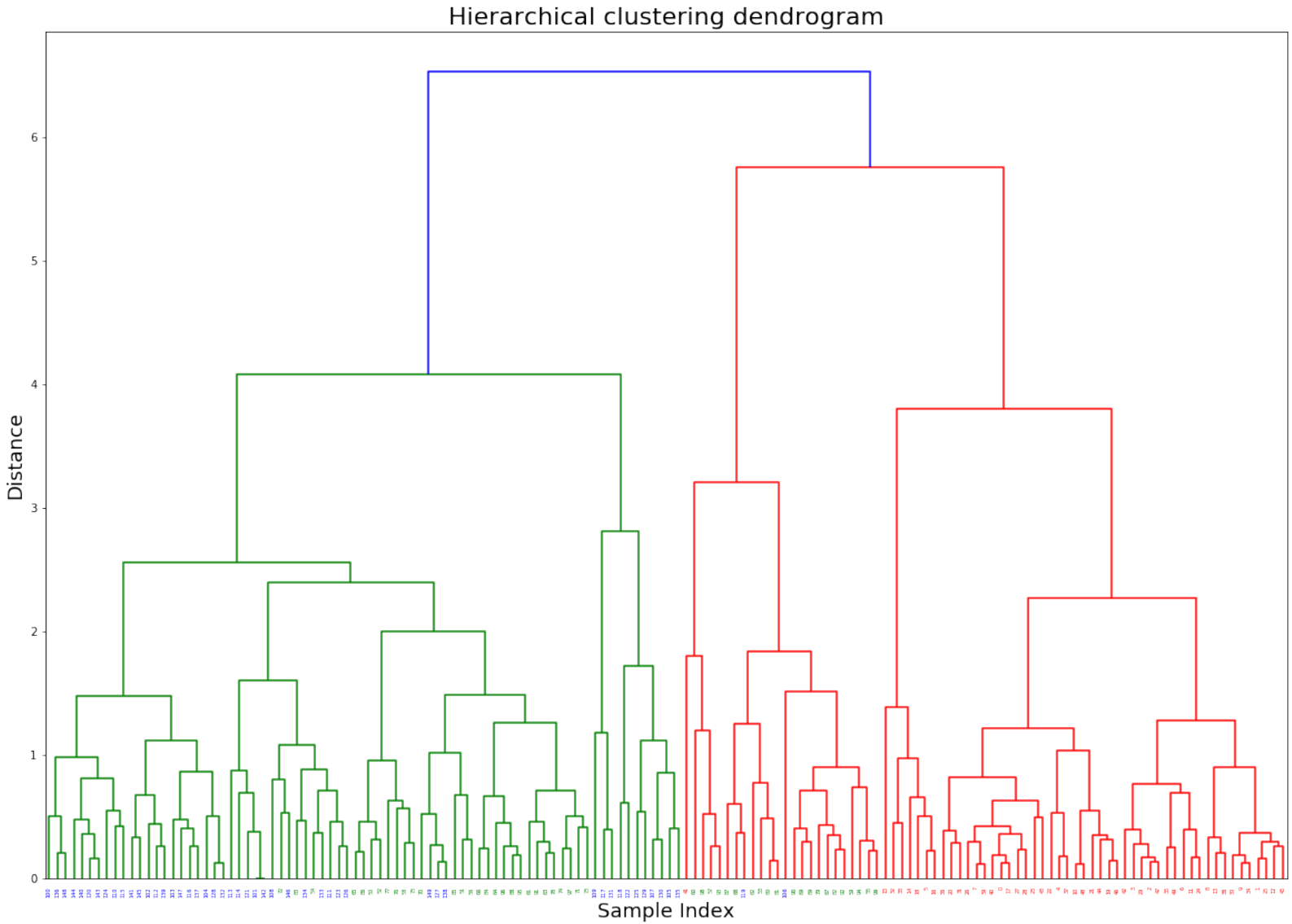
- Can cause the **chaining effect**, potentially resulting in long "snake-like" clusters.

Which Linkage Strategy Should You Use?

- **Ward Linkage**:
  - Ideal if you expect roughly spherical clusters (like k-means).
  - Minimizes the increase in within-cluster variance.
- **Average Linkage**:
  - Balanced approach.
  - Less susceptible to outliers than complete linkage.
  - Less chaining than single linkage.
- **Complete Linkage**:
  - Ensures each cluster has relatively close points.
  - Sensitive to outliers (one distant point can inflate the distance).
- **Single Linkage**:
  - Useful if data naturally forms chains or if bridging points are relevant.
  - Can produce elongated, chain-like clusters.

Tip: Experiment with different linkage methods and use domain knowledge to see which clusters make sense for your data.

Dendrogram: tree diagram



Hierarchical Clustering in Python


```
from scipy.cluster.hierarchy import dendrogram, linkage
mylink = linkage(values, method = 'complete', metric = 'euclidean')

fig, ax = plt.subplots(figsize=(20, 14))
dendrogram(mylink, orientation = 'top', color_threshold=6)
plt.show()
```

Outline

Clustering evaluation: issues

Two problems:

- Cluster numbers are arbitrary and may not match class numbers
- The *solution* (real classes) may not even exist

Approaches:

- If real classes are known: compare the group coincidences (external evaluation)
- Otherwise: measure the coherence of the groups (internal evaluation)

External evaluation: Rand Index

$$\text{RI} = \frac{a+b}{a+b+c+d}$$

Where, given our clustering Y and the reference classes X:

a. Number of pairs in the **same** cluster on both X and Y
b. Number of pairs in **different** clusters on both X and Y
c. Number of pairs in the **same** cluster on X but on **different** clusters on Y
d. Number of pairs in **different** clusters on X but in the **same** cluster on Y

(Adjusted) Rand Index in Python

The **Adjusted Rand Index** is designed to compensate the random chance of getting two points in the same cluster

Ranges from +1 (perfect) to -1 (worst)

```
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score

mykm = KMeans(n_clusters=7)
mykm.fit(data)

adjusted_rand_score(classes, mykm.labels_)

0.1618769997592638
```

Internal evaluation: silhouette

- $a(i)$: mean distance of point i to the points in the same cluster
- $b(i)$: mean distance of point i to the points in its nearest cluster (i.e. cluster with lowest mean distance to i)

$$\text{s(i)} = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

Clustering evaluation: average silhouette of all points.

Ranges from +1 (perfect) to -1 (worst)

Silhouette score in Python

The Silhouette score function receives the clustering assignment and the original data points to compute the distances

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

mykm = KMeans(n_clusters=7)
mykm.fit(data)

silhouette_score(classes, mykm.labels_)

0.20482219186567105
```

Summary

- Clustering
- K-Means: K centroids
- DBSCAN: ϵ-connected points
- Hierarchical: gradually merge points/clusters. Dendrogram
- Clustering evaluation: external (Rand Index), internal (Silhouette score)