References

Recap

- **k-Means** as a simple-yet-popular clustering method that produces a flat clustering of data points

- **Elbow method** to determine a suitable number of clusters (or tune other parameters of a clustering method)

- **External** (e.g., purity) and **internal** (e.g., Dunn Index) **measures** to determine quality of clustering

- **Hierarchical clustering** determines a **sequence of clusterings** that can be visualized in a **dendrogram**
4.4 Density-based Clustering

- k-Means as a representative-based clustering method can only find *convey clusters* and *must assign every data point* to a cluster.

- Density-based clustering methods determine clusters as *regions* having consistently *high density* and label *isolated data points* as *noise*.

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

Source: Zaki and Meira [3]
Epsilon Neighborhood of a data point \( x \)

\[
N_\varepsilon(x) = \{ y \mid d(x, y) \leq \varepsilon \}
\]

contains all points having distance less than or equal to \( \varepsilon \)

Data point \( x \) is called a core point if its epsilon neighborhood contains at least \( \text{minpts} \) data points (including \( x \))

Data point \( x \) is called a border point, if it is not a core, but belongs to the epsilon neighborhood of a core

All other data points are considered noise
### Core, Border, and Noise

- Data point $x$ is a **core**
- Data point $y$ is a **border point**
- Data point $z$ is **noise**

Source: Zaki and Meira [3]

$minpts = 6$
Reachability

- Data point $x$ is **directly reachable** from data point $y$, if $y$ is a **core** and $x$ belongs to the epsilon neighborhood of $y$, i.e.

  $$x \in N_{\varepsilon}(y)$$

- Data point $x$ is **(density) reachable** from data point $y$, if there is a chain of data points $x_0, \ldots, x_l$, so that

  $$x_0 = x \land x_l = y$$

  $$\forall 1 \leq i \leq l : x_i \text{ is directly reachable from } x_{i-1}$$

- Reachability is **not symmetric**, since the data point $y$ could be a core, but the data point $x$ is not
Connectedness and Density-Based Clusters

- Two data points \(x\) and \(y\) are called **connected**, if there is a core \(z\), so that both \(x\) and \(y\) are reachable from \(z\).

- **Density-based cluster** is a **maximal subset of connected data points**, i.e., there are no data points that could be added.
DBSCAN

- **Intuition:**
  - Compute **epsilon neighborhoods** for all data points
  - Determine all **cores**
  - Determine **noise**
  - **Grow a new density-based cluster from each data point** that does not yet belong to an already-determined cluster

- Note that DBSCAN is not deterministic, since the assignment of data point to clusters depends on the **order in which data points are considered**
dbScan(D, \( \epsilon \), minpts) { \\
  // Cores 
  Cores = \emptyset; \\
  \\
  for(x \in D) { 
    // Compute epsilon neighborhoods 
    \( N_\epsilon(x) = \text{computeNeighborhood}(x, \epsilon) \); 
    
    // Initialize cluster id 
    id(x) = \emptyset; 
    
    // Check whether data point is a core 
    if (\( N_\epsilon(x) \geq \text{minpts} \)) Cores = Cores \cup \{x\}; 
  } 
  
  // Grow density-based cluster from each core 
  k = 0; 
  for(x \in Cores) { 
    if(id(x) == \emptyset) { 
      k++; 
      id(x) = k; 
      densityConnected(x,k); 
    } 
  } 
  
  // Determine clustering, border points, and noise 
  \( \mathcal{C} = \emptyset; \)
  for(i = 1 \ldots k) \( \mathcal{C} = \mathcal{C} \cup \{x \in D : id(x) = k\} \); 
  Noise = \{x \in D : id(x) = \emptyset\}; 
  \text{Border} = D \setminus \{\text{Cores} \cup \text{Noise}\}; 

DBSCAN

Source: Zaki and Meira [3]
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import DBSCAN
from matplotlib.backends.backend_pdf import PdfPages
import matplotlib.pyplot as plt

# load data
cars = pd.read_csv("/path/to/auto-mpg.data", sep="\s+", header=None)

# extract power and weight as data matrix X
X = cars.iloc[:,[3,4]].values

# extract origin as target value y
y = cars.iloc[:, 7].values

# normalize data
min_max_scaler = MinMaxScaler()
min_max_scaler.fit(X) # determine min and max
X_normalized = min_max_scaler.transform(X)
Clustering Cars based on Power and Weight

```python
# apply k-Means
db = DBSCAN(eps=0.05, min_samples=5, metric='euclidean')
db.fit_predict(X_normalized)

# plot cars
# U.S. : o / Europe: x / Japan : +
m = ['o' if o==1 else 'x' if o==2 else '+' for o in y]
# Noise : black / Cluster 1 : red / Cluster 2 : blue /
# Cluster 3 : green / Cluster 4 : yellow
c = ['black' if l==-1 else 'red' if l==0 else 'blue' if l==1 else 'green' if l==2 else 'yellow' for l in db.labels_]
for i in range(0, len(X)):
    plt.scatter(X[i, 0], X[i, 1], color=c[i], marker=m[i])
plt.xlabel('Power [hp]')
plt.ylabel('Weight [lbs]')
plt.show()
```

Clustering Cars based on Power and Weight
Summary

- **DBSCAN** as a density-based clustering method can find **non-convex clusters** and is able to label **data points as noise**

- **DBSCAN** comes with two hyper parameters $\varepsilon$ and $\text{minpts}$ that need to be **carefully tuned** based on the data
References

