Technische Hochschule Rosenheim

Modul - Unsupervised and Reinforcement Learning (URL)

Bachelor Programme AAI

02 - Neighborhood Approaches and DBSCAN

Prof. Dr. Marcel Tilly

Faculty of Computer Science, Cloud Computing

Vers. 24.10.2022

Agenda

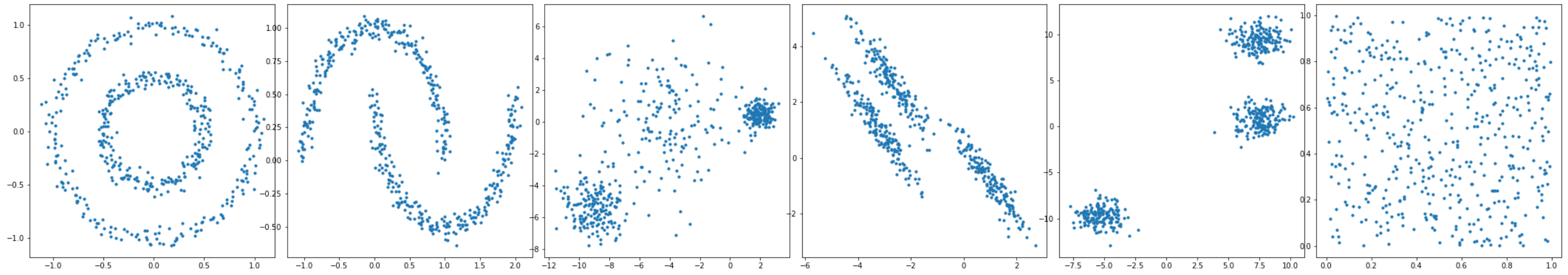
- We will look into another approach on clustering: DBSCAN
- We will define *Neighborhood* and *Density*
- Try to motivate the DBSCAN algorithm
- Compare k-means, hierarchical clustering, and DBSCAN

Clustering so far

- **k-means**: Clusters around centroids
- **Hierarchical Clustering**: *bottom-up* and *top_down*
    - No assumption of a particular number of clusters (i.e. k-means)
    - May correspond to meaningful taxonomies
    - Once a decision is made to combine two clusters, it can't be undone
    - Too slow for large data sets, O(n2 log(n))

Datasets

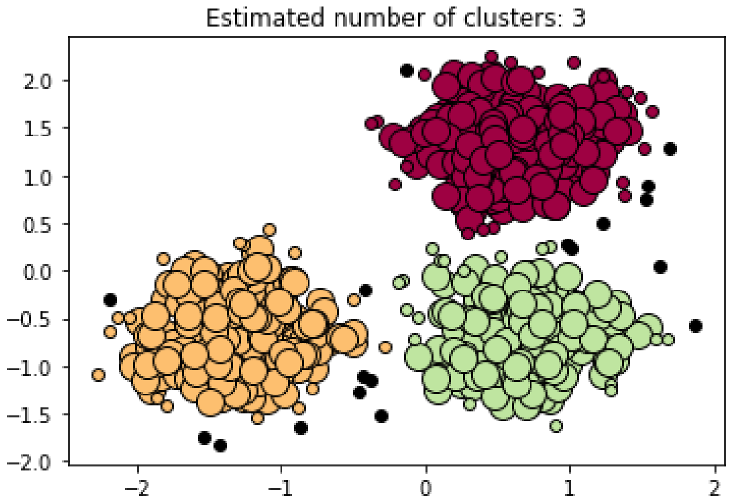
- Does the form of the data matters?
- Where are the limits of k-means and HClustering?
- Which algorithm can work on which data?



- What can we change?

Density-Based Clustering

- Basic Idea:
  - Clusters are dense regions in the data space, separated by regions of lower object density
  - A cluster is defined as a maximal set of density-connected points
  - Detect arbitrarily shaped clusters
- Method:
  - DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
  - Paper: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf



Density-Based Clustering

Basic Concept

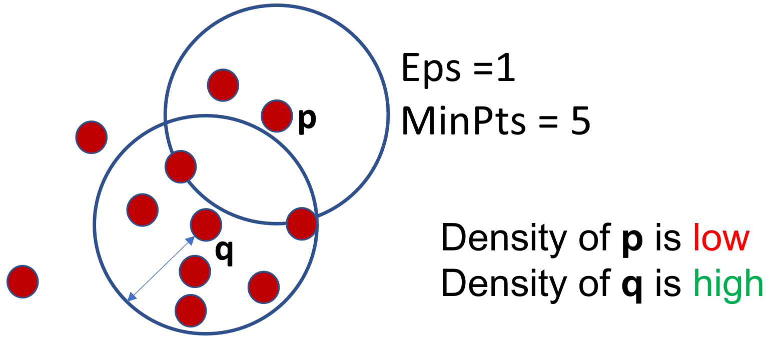
- Intuition for the formalization of the basic idea
  - For any point in a cluster, the local point density around that point has to exceed some threshold
  - The set of points from one cluster is spatially connected
- Local point density at a point q defined by two parameters
  - **ε–radius (eps)** for the neighborhood of point q:

$$N_\varepsilon(q) := \{p \in D | dist(p,q) \leq \varepsilon\}$$

  - **MinPts** – minimum number of points in the given neighbourhood $N_\varepsilon(q)$
- q is called a **core object** (or **core point**) if $|N_\varepsilon(q)| \geq MinPts$

High Density v.s. Low Density

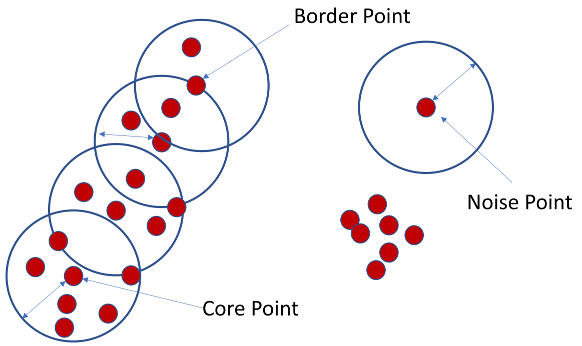
- Two parameters
  - Eps (ε): Maximum radius of the neighborhood
  - MinPts : Minimum number of points in the Eps-neighborhood of a point
- **High density**: ε-Neighborhood of an object contains at least MinPts of objects
- **Low density** : ε-Neighborhood of an object *does not* contain at MinPts of objects



Try this: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Types of Points in DBSCAN Clustering

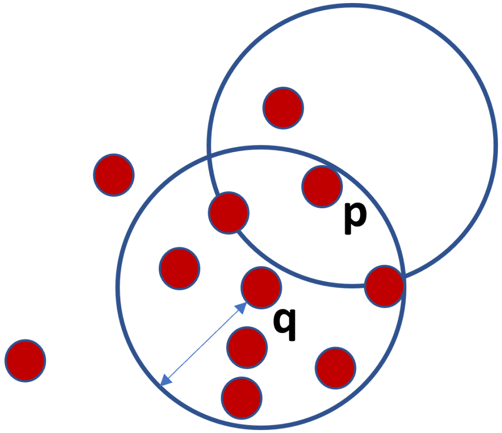
- **Core point**: A core point is one in which at least have minPts number of points (including the point itself) in its surrounding region within the radius eps: $if |N (p)|≥ MinPts§
- **Border point**: has fewer than MinPts within its ϵ-neighborhood (N), but it lies in the neighborhood of another core point.
- **Noise point**: A noise point (**outlier**) is neither a core point and nor is it reachable from any core points.



Density-based related points

Direct density reachability

An object p is *directly density-reachable* from object q if (1) q is a core object and (2) p is in q's ε-neighborhood.



Eps =1

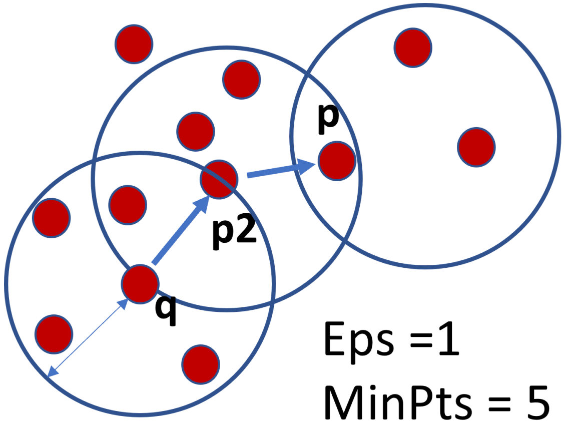
MinPts = 5

Density-based related points

Density reachability

A point p is *density-reachable* from a point q if there is a chain of points p1, ..., pn, p1 = q, pn = p such that p_{i+1} is directly density-reachable from p_i.

$$p_1 = q \rightarrow p_2 \rightarrow \ldots \rightarrow p_n = q$$



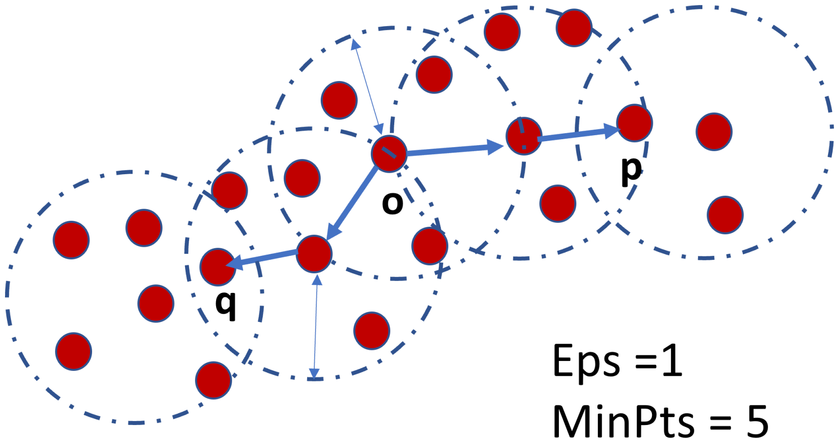
Eps =1

MinPts = 5

Density-based related points

Density connectivity

A point p is *density-connected* to a point q if there is a point o such that both p and q are density-reachable from o



Eps =1

MinPts = 5

DBSCAN Clustering Algorithm

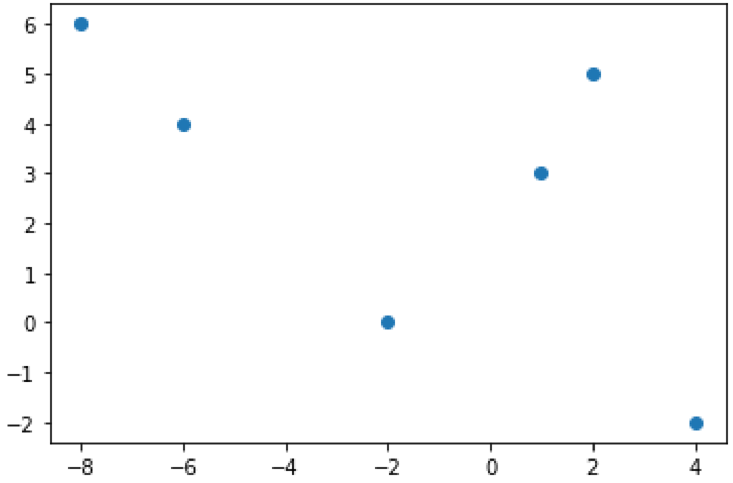
We start with the data points and values of eps and minPts as input:

1. Out of *n* unvisited sample data points, we'll first move through each point in a loop and mark each one as visited.
2. From each point, we'll look at the distance to every other point in the dataset.
3. All points that fall within the neighborhood radius hyperparameter (*eps*) should be considered as neighbors.
4. The number of neighbors should be at least as many as the minimum points required (*MinPts*).
5. If the minimum point threshold is reached, the points should be grouped together as a cluster, or else marked as noise.
6. This process should be repeated until all data points are categorized in clusters or as noise.

Step1: DBSCAN

Given six sample data points, view each point as its own cluster X = np.array([[1,3] , [-8,6], [-6,4] , [4,-2], [2,5], [-2,0]]):

```
import numpy as np
import matplotlib.pyplot as plt


X = np.array([[1,3] , [-8,6], [-6,4]
, [4,-2], [2,5], [-2,0]])

plt.scatter(X[:,0],X[:,1])
plt.show()
```



Step2: Calculate Pairwise Euclidean Distance

```
np.linalg.norm(X[:, None, :] - X[None, :, :], axis=-1)
```

This results into:

```
      [[1,3]      , [-8,6]     , [-6,4]     , [4,-2]     , [2,5]      , [-2,0]
[1,3] [ 0.        ,  9.48683298,  7.07106781,  5.83095189,  2.23606798, 4.24264069],
[-8,6][ 9.48683298,  0.        ,  2.82842712, 14.4222051 , 10.04987562, 8.48528137],
[-6,4][ 7.07106781,  2.82842712,  0.        , 11.66190379,  8.06225775, 5.65685425],
[4,-2][ 5.83095189, 14.4222051 , 11.66190379,  0.        ,  7.28010989, 6.32455532],
[2,5] [2.23606798, 10.04987562,  8.06225775,  7.28010989,  0.        , 6.40312424],
[-2,0][4.24264069,  8.48528137,  5.65685425,  6.32455532,  6.40312424, 0.        ]])
```

- From each point, expand a neighborhood size outward and form clusters (use *eps=5*).
- This means that any two points will be neighbors if the distance between them is less than five units.
- For example, point (1,3) has points (2,5) and (-2,0) as neighbors.

Step3: Categories

- Depending on the number of points in the neighborhood of a given point, the point can be classified into the following three categories:
  - **Core Point**: If the point under observation has data points greater than the minimum number of points in its neighborhood that make up a cluster, then that point is called a core point of the cluster. All core points within the neighborhood of other core points are part of the same cluster. However, all the core points that are not in same neighborhood are part of another cluster.
  - **Boundary Point**: If the point under observation does not have sufficient neighbors (data points) of its own, but it has at least one core point (in its neighborhood), then that point represents the boundary point of the cluster. Boundary points belong to the same cluster of their nearest core point.
  - **Noise Point**: A data point is treated as a noise point if it does not have the required minimum number of data points in its neighborhood and is not associated with a core point. This point is treated as pure noise and is excluded from clustering.

Step4

Points that have neighbors are then evaluated to see whether they pass the minimum points threshold.

- In this example:
  - If we had passed through a minimum points threshold of two (*MinPts=3*), then points (1,3), (2,5), and (-2,0) could formally be grouped together as a cluster.
  - If we had a minimum points threshold of four, then these three data points would be considered superfluous noise.

```
      [[1,3]      , [-8,6]     , [-6,4]     , [4,-2]     , [2,5]      , [-2,0]
[1,3] [ 0.        ,  9.48683298,  7.07106781,  5.83095189,  2.23606798, 4.24264069],
[-8,6][ 9.48683298,  0.        ,  2.82842712, 14.4222051 , 10.04987562, 8.48528137],
[-6,4][ 7.07106781,  2.82842712,  0.        , 11.66190379,  8.06225775, 5.65685425],
[4,-2][ 5.83095189, 14.4222051 , 11.66190379,  0.        ,  7.28010989, 6.32455532],
[2,5] [2.23606798, 10.04987562,  8.06225775,  7.28010989,  0.        , 6.40312424],
[-2,0][4.24264069,  8.48528137,  5.65685425,  6.32455532,  6.40312424, 0.         ]])
```

Step5

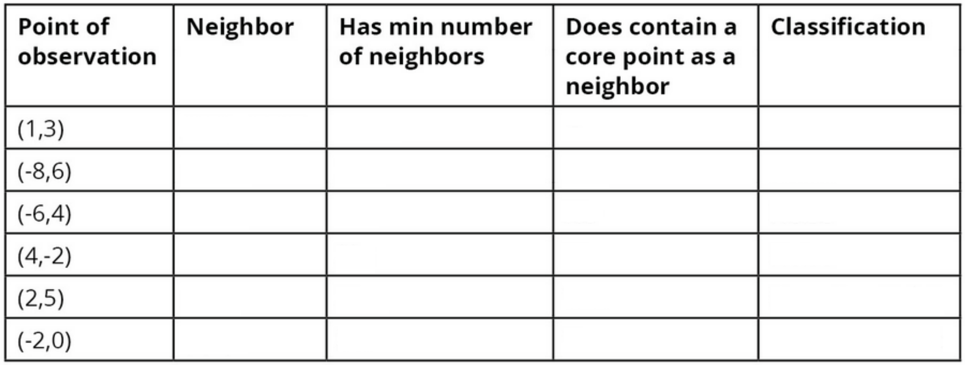
> Points that have fewer neighbors than the minimum number of neighboring points required and whose neighborhood does not contain a core point are marked as noise and remain unclustered.

- Thus, points (-6,4), (4,-2), and (-8,6) fall under this category.
- However, points such as (2,5) and (2,0), though don't satisfy the criteria of the minimum number of points in neighborhood, do contain a core point as their neighbor, and are therefore marked as boundary points.

```
      [[1,3]      , [-8,6]     , [-6,4]     , [4,-2]     , [2,5]      , [-2,0]
[1,3] [ 0.        ,  9.48683298,  7.07106781,  5.83095189,  2.23606798, 4.24264069],
[-8,6][ 9.48683298,  0.        ,  2.82842712, 14.4222051 , 10.04987562, 8.48528137],
[-6,4][ 7.07106781,  2.82842712,  0.        , 11.66190379,  8.06225775, 5.65685425],
[4,-2][ 5.83095189, 14.4222051 , 11.66190379,  0.        ,  7.28010989, 6.32455532],
[2,5] [2.23606798, 10.04987562,  8.06225775,  7.28010989,  0.        , 6.40312424],
[-2,0][4.24264069,  8.48528137,  5.65685425,  6.32455532,  6.40312424, 0.        ]])
```

Task 1

Can you fill the table?



Point of observation	Neighbor	Has min number of neighbors	Does contain a core point as a neighbor	Classification
(1,3)				
(-8,6)				
(-6,4)				
(4,-2)				
(2,5)				
(-2,0)				

Step6:

Taken the distances:

```
       [[1,3]      , [-8,6]     , [-6,4]     , [4,-2]     , [2,5]      , [-2,0]
 [1,3] [ 0.         ,  9.48683298,  7.07106781,  5.83095189,  2.23606798, 4.24264069],
 [-8,6][ 9.48683298,  0.         ,  2.82842712, 14.4222051 , 10.04987562, 8.48528137],
 [-6,4][ 7.07106781,  2.82842712,  0.         , 11.66190379,  8.06225775, 5.65685425],
 [4,-2][ 5.83095189, 14.4222051 , 11.66190379,  0.         ,  7.28010989, 6.32455532],
 [2,5] [2.23606798, 10.04987562,  8.06225775,  7.28010989,  0.         , 6.40312424],
 [-2,0][4.24264069,  8.48528137,  5.65685425,  6.32455532,  6.40312424, 0.         ]])
```

It results into:

Point of observation	Neighbor	Has min number of neighbors	Does contain a core point as a neighbor	Classification
(1,3)	(2,5) , (-2,0)	Yes	-	Core Point
(-8,6)	(-6,4)	No	No	Noise
(-6,4)	(-8,6)	No	No	Noise
(4,-2)		No	No	Noise
(2,5)	(1,3)	No	Yes	Boundary Point
(-2,0)	(1,3)	No	Yes	Boundary Point

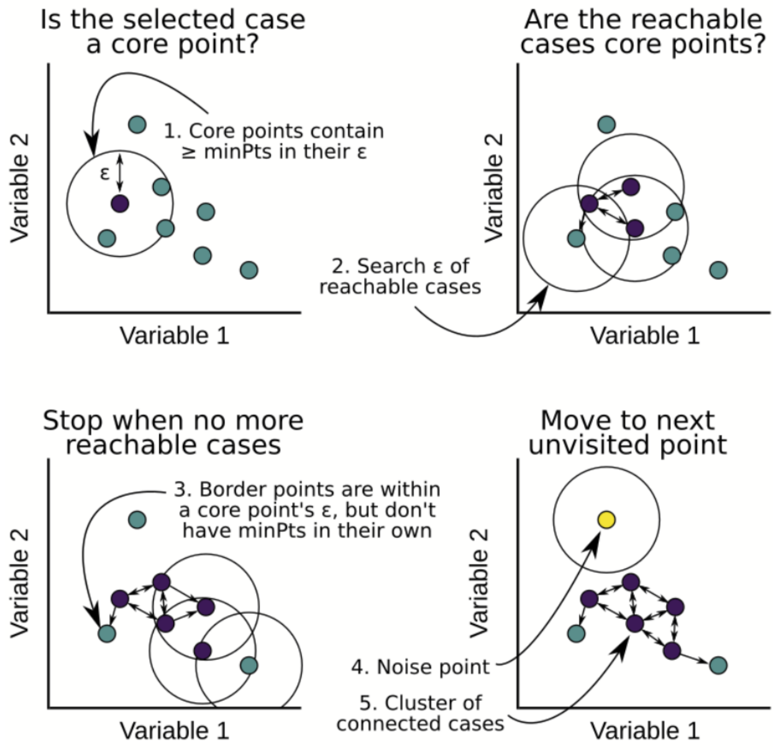
Step7

- Repeat this process on any remaining unvisited data points.
- At the end of this process, you will have sorted your entire dataset into either clusters or unrelated noise.
- DBSCAN performance is highly dependent on the threshold hyperparameters you choose: eps and MinPts!
    - This means that you may have to run DBSCAN a couple of times with different hyperparameter options to get an understanding of how they influence overall performance.

> Note that DBSCAN does not require the centroids that we saw in both k-means and centroid-focused implementation of hierarchical clustering. This feature allows DBSCAN to work better for complex datasets, since most data is not shaped like clean blobs.

DBSCAN is also more effective against outliers and noise than k-means or hierarchical clustering.

The DBSCAN Algorithm


Is the selected case a core point?
Are the reachable cases core points?
1. Core points contain ≥ minPts in their ε
ε
2. Search ε of reachable cases
Stop when no more reachable cases
Move to next unvisited point
3. Border points are within a core point's ε, but don't have minPts in their own
4. Noise point
5. Cluster of connected cases
Variable 1
Variable 2

The DBSCAN Algorithm

- You can find a Juptyer Notebook with DBSCAN on GtHub or hosted

```
DBSCAN(D, eps, MinPts):
    C = 0
    for each unvisited point P in dataset D
        NeighborPts = findNeighbors(P, eps)
        if sizeof(NeighborPts) < MinPts
            mark P as NOISE
        else
            C = next cluster
            expandCluster(P, NeighborPts, C, eps, MinPts)

findNeighbors(P, eps) return all points within P's eps-neighborhood (including P)

expandCluster(P, NeighborPts, C, eps, MinPts):
    add P to cluster C
    for each point P' in NeighborPts
        if P' is not yet member of any cluster
            add P' to cluster C
        elif P' is not visited
            mark P' as visited
            NeighborPts' = regionQuery(P', eps)
            if sizeof(NeighborPts') >= MinPts
                NeighborPts = NeighborPts joined with NeighborPts‘
```

DBSCAN: findNeighbors

For a given point p find all points in dataset x so that $dist(p, x[i]) < eps$

```python
def findNeighbors(x, p, eps):
    neighbors = []

    # For each point in the dataset...
    for potential_neighbor in range(0, x.shape[0]):

        # If a nearby point falls below the neighborhood radius
        #  threshold, add to neighbors list
        if np.linalg.norm(x[p] - x[potential_neighbor]) < eps:
            neighbors.append(potential_neighbor)

    return neighbors
```

DBSCAN: expandCluster

Technische Hochschule Rosenheim

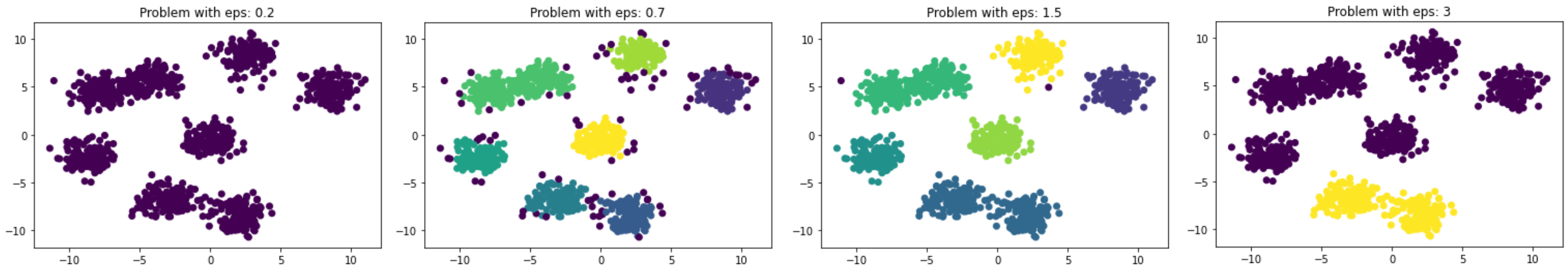
Expand cluster for all reachable points:

```python
def expandCluster(x, labels, p, neighbors, C, eps, min_pts):
    # Assign the cluster label to original point
    labels[p] = C

    # Look at each neighbor of p (by index, not the points themselves) and
    #evaluate
    i = 0
    while i < len(neighbors):
        # Get the next point from the queue.
        potential_neighbor_ix = neighbors[i]
        # If potential_neighbor_ix is noise from previous runs, we can assign it to current cluster
        if labels[potential_neighbor_ix] == -1:
            labels[potential_neighbor_ix] = C

        # Otherwise, if potential_neighbor_ix is unvisited, we can add it to current cluster
        elif labels[potential_neighbor_ix] == 0:
            labels[potential_neighbor_ix] = C

            # Further find neighbors of potential neighbor
            potential_neighbors_cluster = findNeighbors(x, potential_neighbor_ix, eps)

            if len(potential_neighbors_cluster) >= min_pts:
                neighbors = neighbors + potential_neighbors_cluster

        # Evaluate next neighbor
        i += 1
```

Hyperparameter: EPS

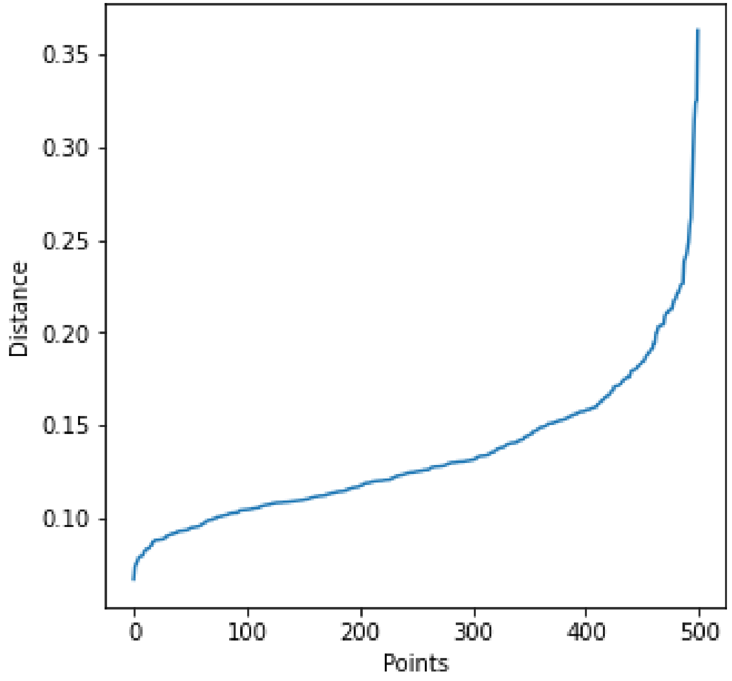
Cluster changes with varying neighborhood radius sizes

```python
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

X, y = make_blobs(n_samples=1000, centers=8, n_features=2, random_state=800)

eps = [0.2,0.7,1.5,3]

for ep in eps:
    db = DBSCAN(eps=ep, min_samples=10, metric='euclidean')
    plt.scatter(X[:,0], X[:,1], c=db.fit_predict(X))
    plt.title('Problem with eps: ' + str(ep))
    plt.show()
```



Finding the Optimal value of Eps

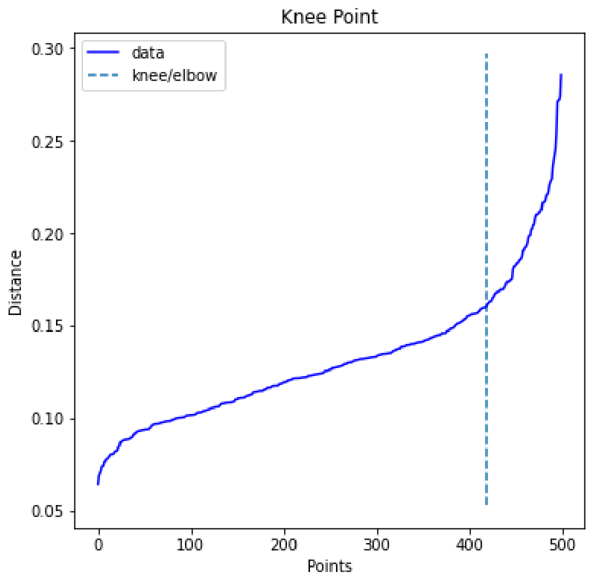
- Rather than experimenting with different values of epsilon, we can use the elbow point detection method to arrive at a suitable value of epsilon.
- **Approach**
  - Average the distance between each point and its k-NearestNeighbors (e.g. k = MinPts) and plot the average k-distances in ascending order on a k-distance graph.
  - The optimal value for epsilon is the point with maximum curvature or bend, i.e. at the greatest slope.



Identifying Elbow Point with Kneed Package

- To determine the location of maximum curvature visually can be difficult.
- Hence, a Python package called *kneed* can be used to detect the knee or elbow point.
    - This can be installed with ‘pip install kneed’

```
from kneed import KneeLocator

i = np.arange(len(distances))

knee = KneeLocator(i, distances, S=1,
  curve='convex',
  direction='increasing',
  interp_method='polynomial')

knee.plot_knee()
print(distances[knee.knee])
```



Task2

Technische
Hochschule
Rosenheim

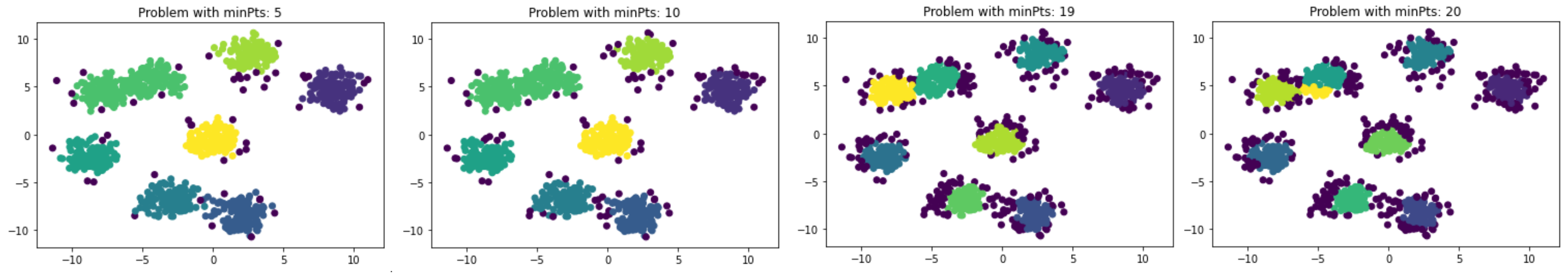
Hyperparameter: MinPts

Vary the minPts parameter between 5 and 20 and check the influence on clustering!

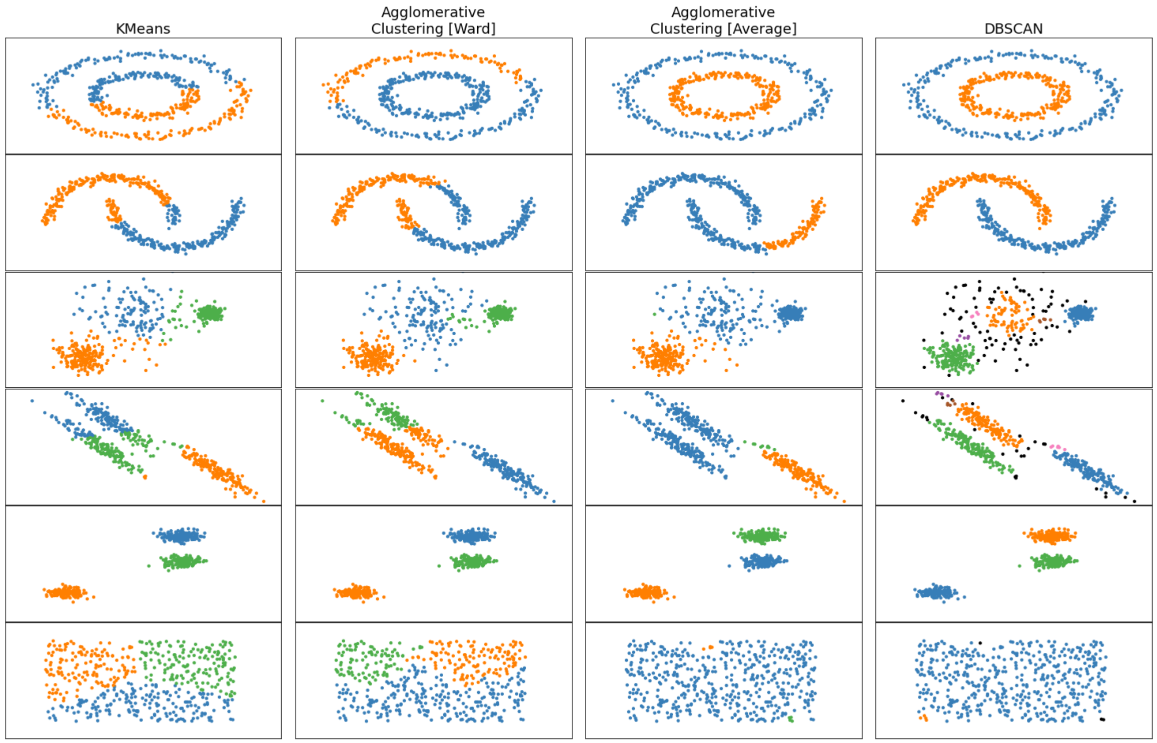
Hyperparameter: MinPts

Vary the minPts parameter between 5 and 20 and check the influence on clustering!

```
X, y = make_blobs(n_samples=1000, centers=8, n_features=2, random_state=800)
minPts = [5, 10, 19, 20]

for p in minPts:
    db = DBSCAN(eps=0.7, min_samples=p, metric='euclidean')
    plt.scatter(X[:,0], X[:,1], c=db.fit_predict(X))
    plt.title('Problem with minPts: ' + str(p))
    plt.show()
```



Clustering Comparison



- Code: https://github.com/matworx/aai-url-notebooks or [hosted]()

k-mean vs. HClustering vs. DBSCAN

- kmeans and hierarchical clustering are good when you have some idea regarding the number of clusters in your data.
- DBSCAN, instead, takes a more bottom-up approach by working with your hyperparameters and finding the clusters it views as important. It is robust to noise and can detect arbitrarily-shaped clusters.
- Compared to k-means and hierarchical clustering, DBSCAN can be seen as being potentially more efficient, since it only has to look at each data point once.
- K-means and hierarchical clustering requires multiple iterations of finding new centroids and evaluating where their nearest neighbors are, once a point has been assigned to a cluster in DBSCAN, it does not change cluster membership.
- No need to pass a number of clusters to DBSCAN and hierarchical clustering both share, in comparison with k-means.

Summary

- Today we discussed the basic idea of density-based clustering
- The two important parameters and the definitions of neighborhood and density in DBSCAN
- Core, border and outlier points
- DBSCAN algorithm
- DBSCAN's pros and cons

Exercise

- On GitLab: https://inf-git.fh-rosenheim.de/aai-url/02_exercise
- Work on Python, again!
- Work on approaches
- Try DBSCAN