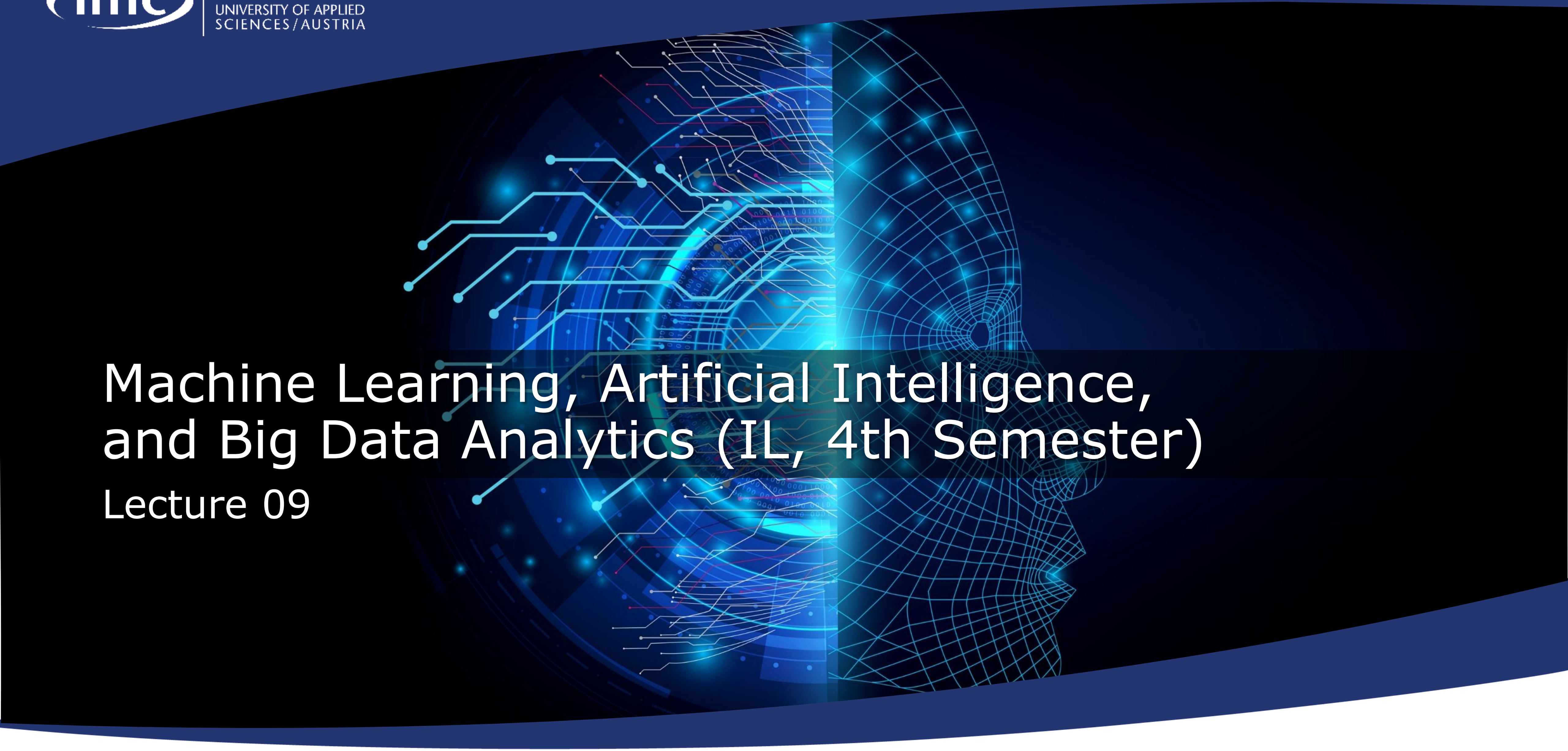
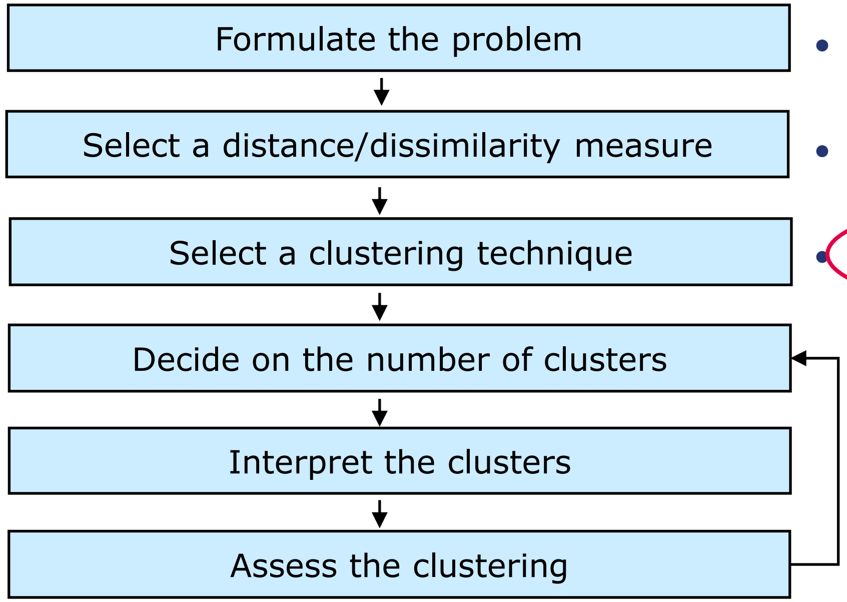
FH KREMS
UNIVERSITY OF APPLIED SCIENCES / AUSTRIA



Machine Learning, Artificial Intelligence, and Big Data Analytics (IL, 4th Semester)

Lecture 09

Summary of previous lecture



Formulate the problem

Select a distance/dissimilarity measure

Select a clustering technique

Decide on the number of clusters

Interpret the clusters

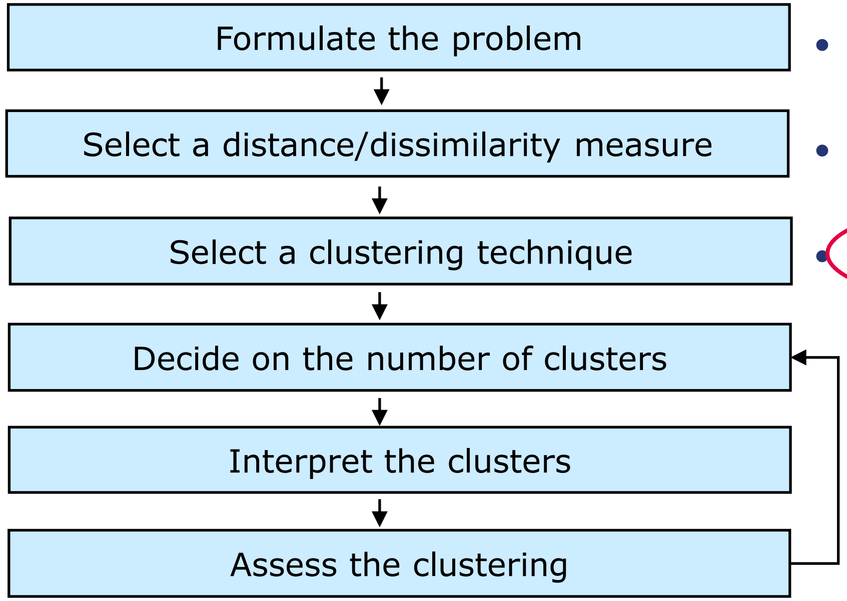
Assess the clustering

- Given a set of <…>, I want to find <…>, based on <…>
- Euclidean, Manhattan, Cosine, Jaccard, …
- K-means, PAM, HAC, HDC, DBScan, …

partitional hierarchical

- Using Silhouette, Elbow, Dendograms, …

Summary of previous lecture



Formulate the problem

Select a distance/dissimilarity measure

Select a clustering technique

Decide on the number of clusters

Interpret the clusters

Assess the clustering

- Given a set of <…>, I want to find <…>, based on <…>
- Euclidean, Manhattan, Cosine, Jaccard, …
- K-means, PAM, HAC, HDC, **DBScan**, …

partitional hierarchical Density-based (focus of today)

- Using Silhouette, Elbow, Dendograms, …

Agenda

- Introduction to Density-Based clustering
- DBScan
- HDBscan and OPTICS
- Final considerations and wrap-up on clustering techniques

Density-based clustering

- Density based clustering are based on **connectivity** and **density** functions!
  - 👍 Discovers clusters of arbitrary shape
  - 👍 Handle noise
  - 👍 No initial assumption on the n. of clusters
  - 👎 Requires to tune density parameters

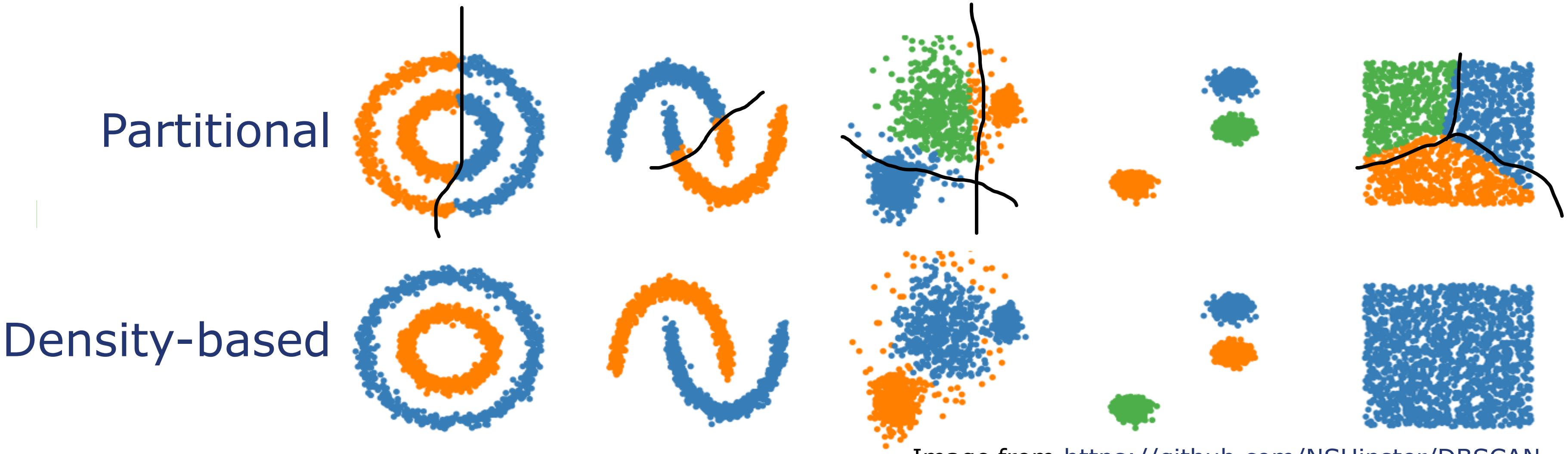
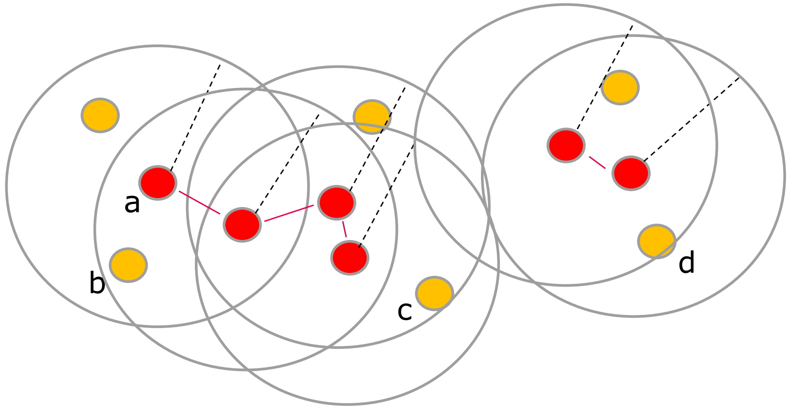


Image from https://github.com/NSHipster/DBSCAN

Density-based clustering

Main concepts

- Core objects: Objects with at least *m* other objects within a radius (neighborhood).
- Direct Density Reachable: An object *i* is DDR to a core object *j* if it lies in *j* neighborhood.
- Density reachable: A point *i* is DR to *j* if there is a chain of DDR objects between *i* and *j*.
- Density-Based Cluster: Connected objects w.r.t a maximum reachability



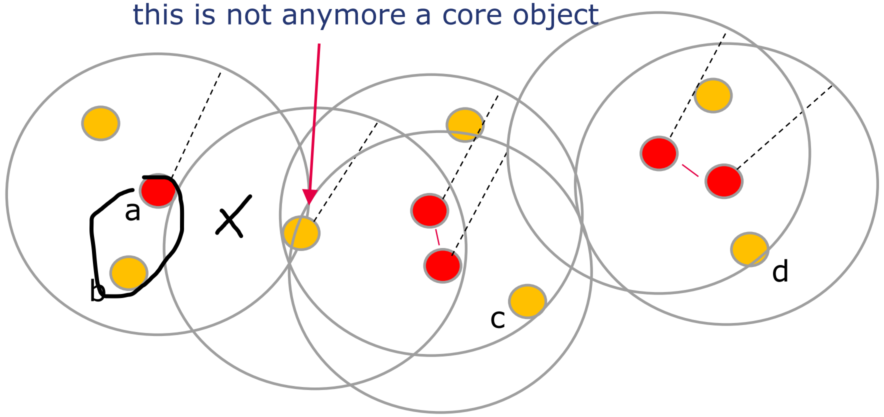
- Example with m = 3
    - *a* and *b* are directly density reachable
    - *a* and *c* are density reachable chain
    - *a* and *d* are not density reachable

Density-based clustering

Main concepts

- Core objects: Objects with at least *m* other objects within a radius (neighborhood).
- Direct Density Reachable: An object *i* is DDR to a core object *j* if it lies in *j* neighborhood.
- Density reachable: A point *i* is DR to *j* if there is a chain of DDR objects between *i* and *j*.
- Density-Based Cluster: Connected objects w.r.t a maximum reachability

The chain of DDR is broken because
this is not anymore a core object



- Example with m = 3
  - *a* and *b* are directly density reachable
  - *a* and *c* are **not** density reachable
  - *a* and *d* are not density reachable

DBScan

Density-BaSed Clustering for Application with Noise

- Two parameters
  - **Eps**: Maximum radius of the neighborhood.
  - **MinPts**: Minimum number of points in an Eps-neighborhood (including the core point).

- A point is a **core** point if it has more than *MinPts* points within *Eps*
- A point is a **border** point if it has fewer than *MinPts* points within *Eps*, but is in the neighborhood of a core point
- A **noise** point is any point that is not a core point or a border point

DBScan

The algorithm

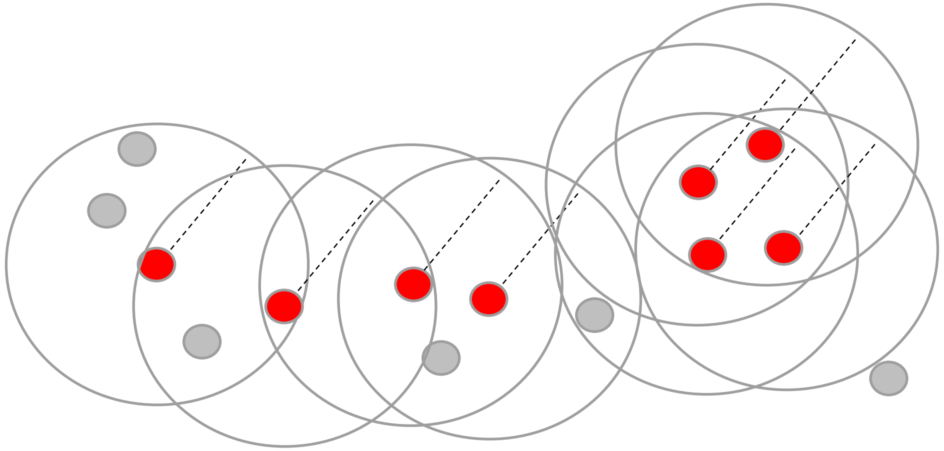
```
Identify all core points
cluster_label ← 0

for all core points in data do
    if core point has no cluster_label then
        cluster_label ← cluster_label + 1
        assign cluster_label to core point
    endif
    for all points in eps-neighborhood(core point)) do
        if point has no cluster_label then
            assign cluster_label to point
        endif
    endfor
endfor
```

Example

Eps = 1 cm, MinPTS = 3

```
Identify all core points
cluster_label ← 0
```

🟡 border points
🔴 core points
⚪ noise points

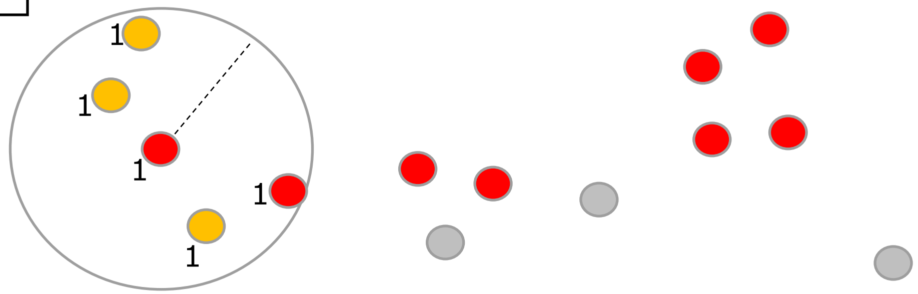


Example

Eps = 1 cm, MinPTS = 3

🟡 border points
🔴 core points
⚪ noise points

```
if core point has no cluster_label then
        cluster_label ← cluster_label + 1
        assign cluster_label to core point
for all points in eps-neighborhood(core point)) do
        if point has no cluster_label then
            assign cluster_label to point
        endif
    endfor
```

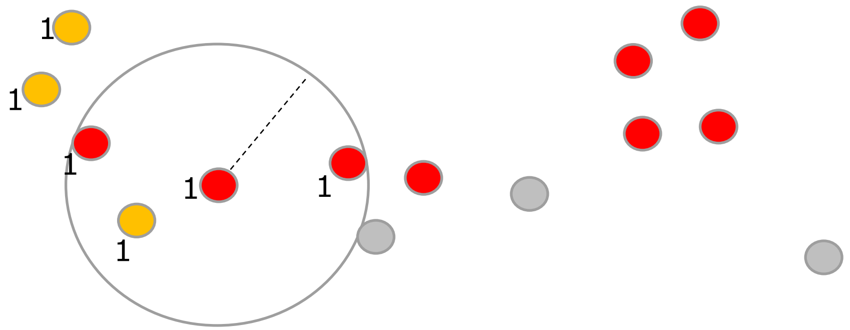


Example

Eps = 1 cm, MinPTS = 3

● border points
● core points
○ noise points

```
if core point has no cluster_label then
    cluster_label ← cluster_label + 1
    assign cluster_label to core point
for all points in eps-neighborhood(core point)) do
    if point has no cluster_label then
      assign cluster_label to point
    endif
  endfor
```

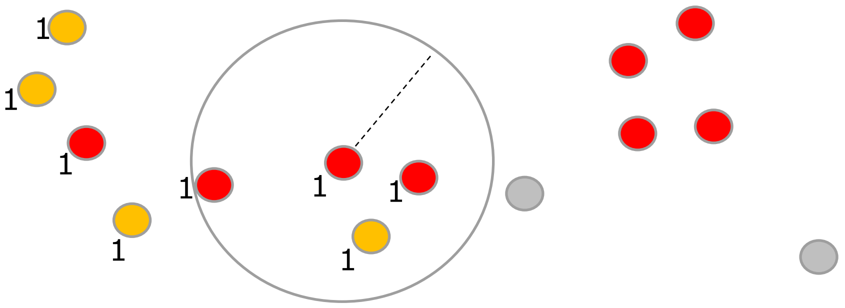


Example

Eps = 1 cm, MinPTS = 3

🟡 border points
🔴 core points
⚪ noise points

```
if core point has no cluster_label then
        cluster_label ← cluster_label + 1
        assign cluster_label to core point
for all points in eps-neighborhood(core point)) do
        if point has no cluster_label then
            assign cluster_label to point
        endif
    endfor
```

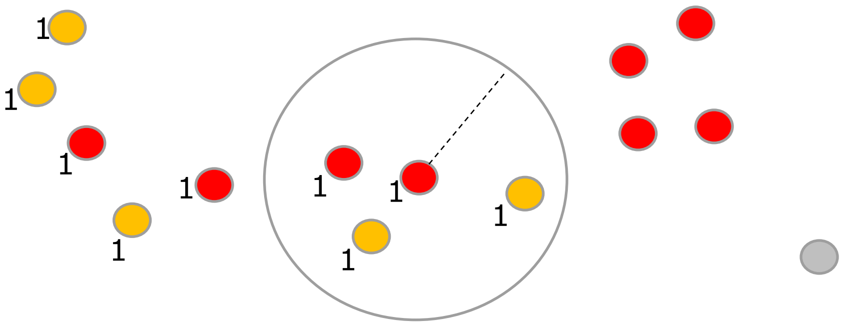


Example

Eps = 1 cm, MinPTS = 3

🟡 border points
🔴 core points
⚪ noise points

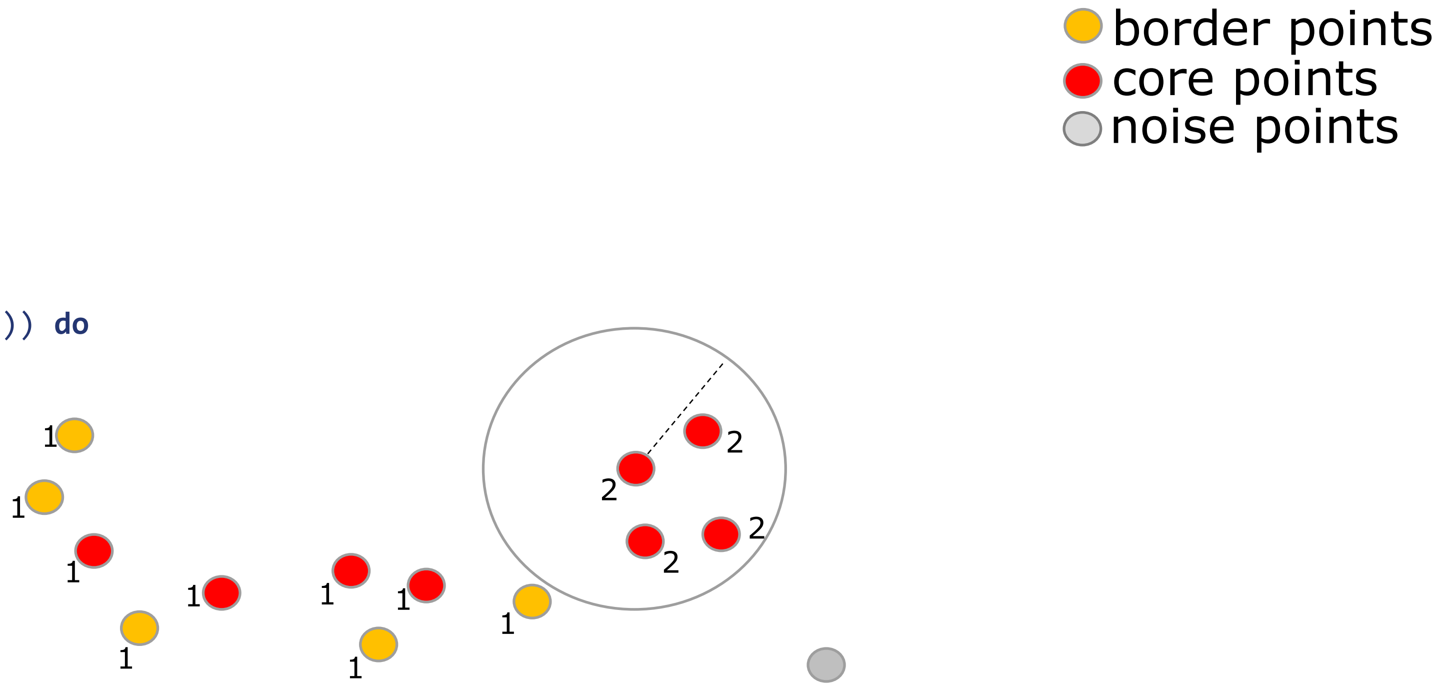
```
if core point has no cluster_label then
      cluster_label ← cluster_label + 1
      assign cluster_label to core point
for all points in eps-neighborhood(core point)) do
      if point has no cluster_label then
         assign cluster_label to point
      endif
   endfor
```



Example

Eps = 1 cm, MinPTS = 3

```
if core point has no cluster_label then
        cluster_label ← cluster_label + 1
        assign cluster_label to core point
for all points in eps-neighborhood(core point)) do
        if point has no cluster_label then
            assign cluster_label to point
        endif
    endfor
```



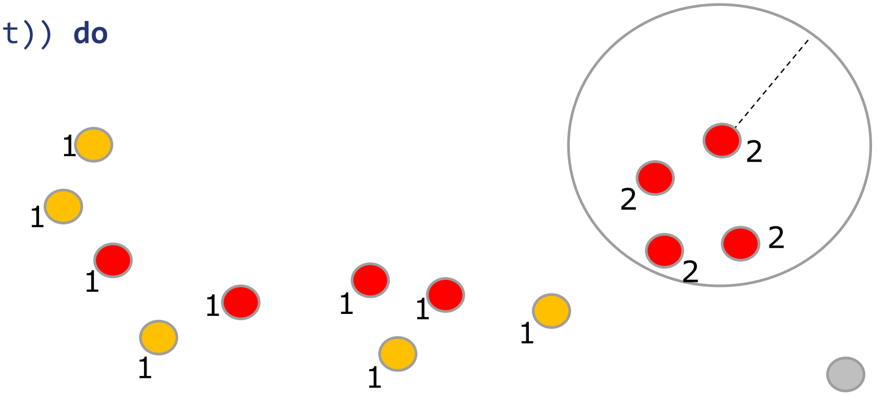
🟡 border points
🔴 core points
⚪ noise points

Example

Eps = 1 cm, MinPTS = 3

🟡 border points
🔴 core points
⚪ noise points

```
if core point has no cluster_label then
    cluster_label ← cluster_label + 1
    assign cluster_label to core point
for all points in eps-neighborhood(core point)) do
    if point has no cluster_label then
      assign cluster_label to point
    endif
  endfor
```

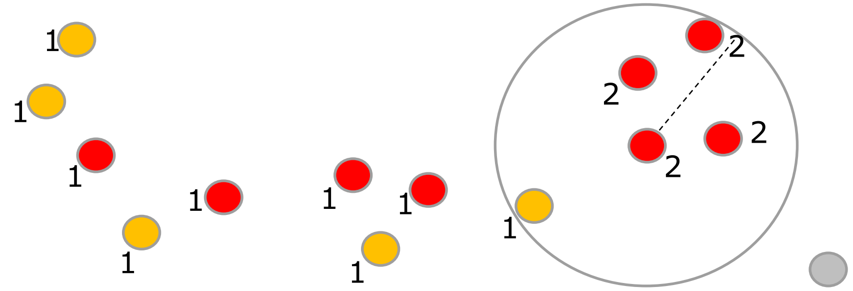


Example

Eps = 1 cm, MinPTS = 3

```
if core point has no cluster_label then
       cluster_label ← cluster_label + 1
       assign cluster_label to core point
for all points in eps-neighborhood(core point)) do
       if point has no cluster_label then
           assign cluster_label to point
       endif
   endfor
```

🟡 border points
🔴 core points
⚪ noise points

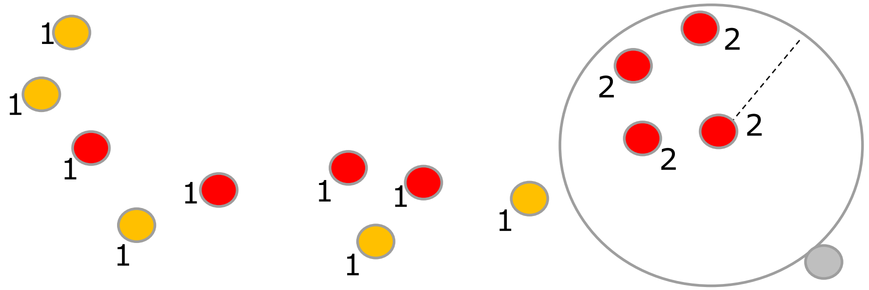


Example

Eps = 1 cm, MinPTS = 3

🟡 border points
🔴 core points
⚪ noise points

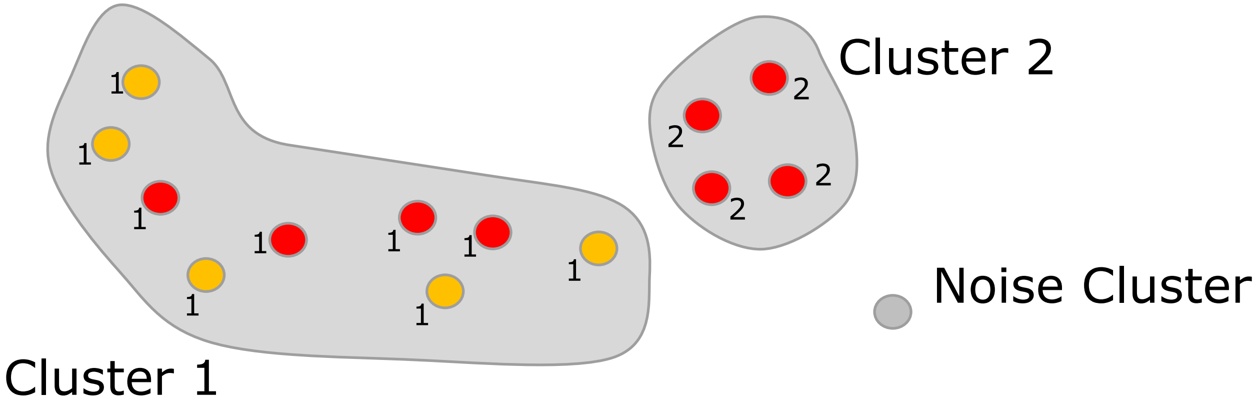
```
if core point has no cluster_label then
        cluster_label ← cluster_label + 1
        assign cluster_label to core point
for all points in eps-neighborhood(core point)) do
        if point has no cluster_label then
            assign cluster_label to point
        endif
    endfor
```



Example

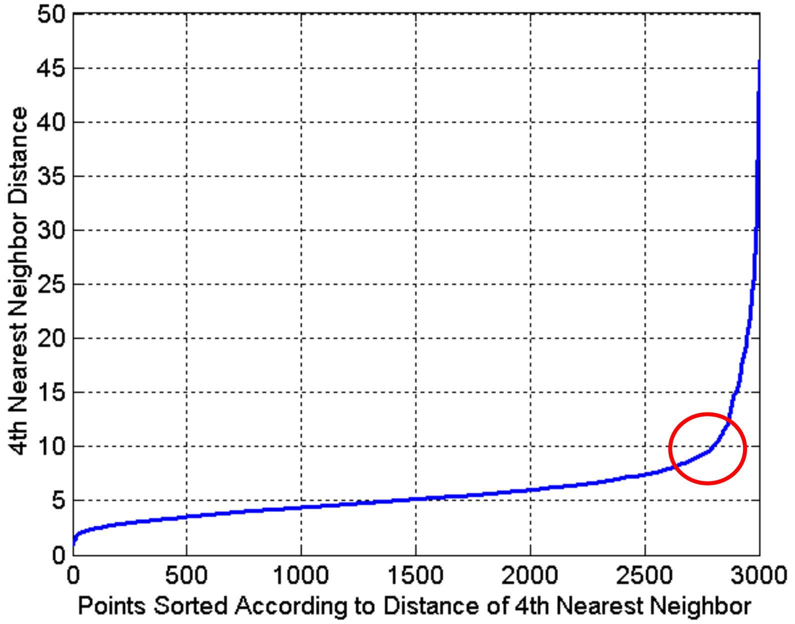
Eps = 1 cm, MinPTS = 3

🟡 border points
🔴 core points
⚪ noise points



DBSCAN: Determining EPS and MinPts

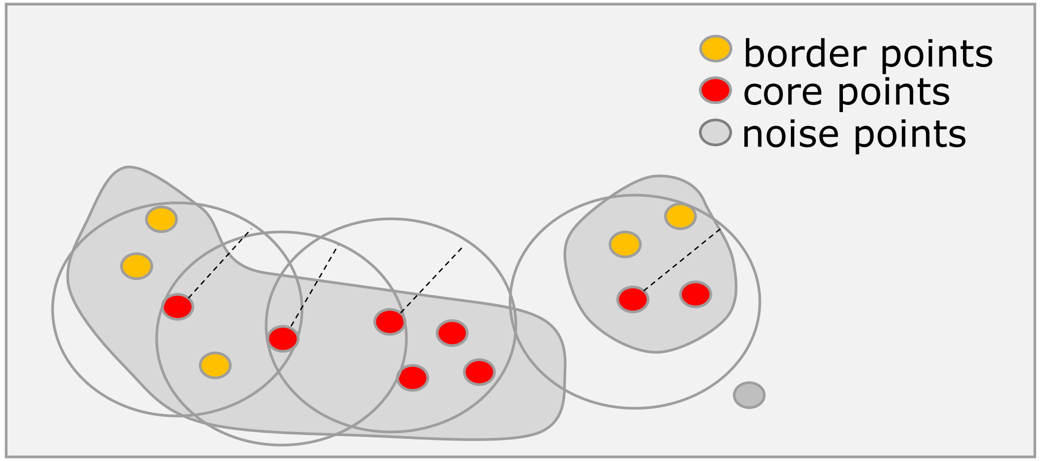
- MinPts
  - Conceptually it translates to the min. desired cluster size
  - MinPts = 1 does not make sense
  - Rule-of-Thumb: **minPts ~= num_features * 2**
  - In any case: minPts >= num_features + 1
- Eps
  - Calculate the average of the distances of every point to its *k-nearest neighbors* (with K = MinPts).
  - Next, these k-distances are plot in ascending order.
  - A knee corresponds to a threshold where a sharp change occurs along the k-distance curve. Hence it indicates the optimal eps parameter.



Example: MinPts = 4 → Eps ~= 10

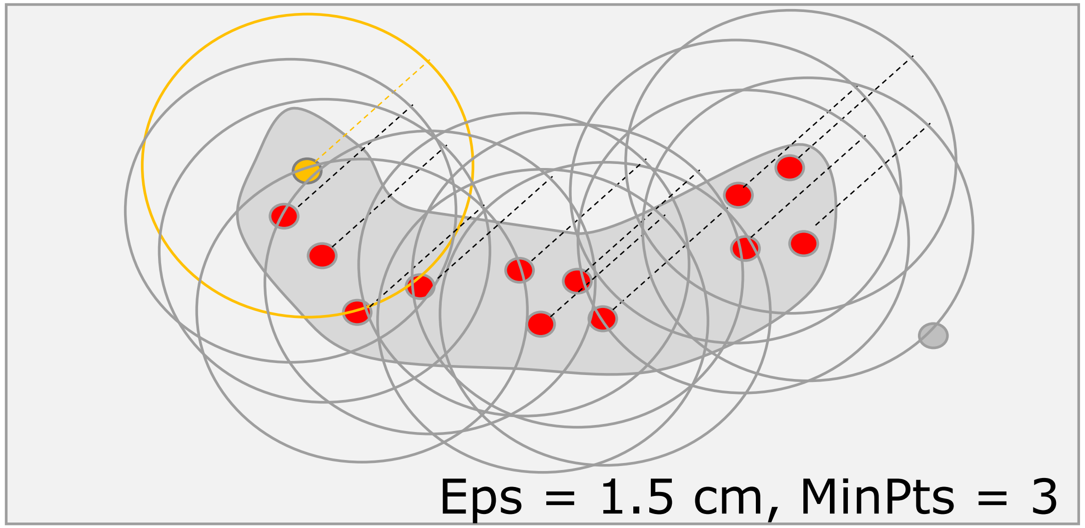
Example

Effect of varying Eps



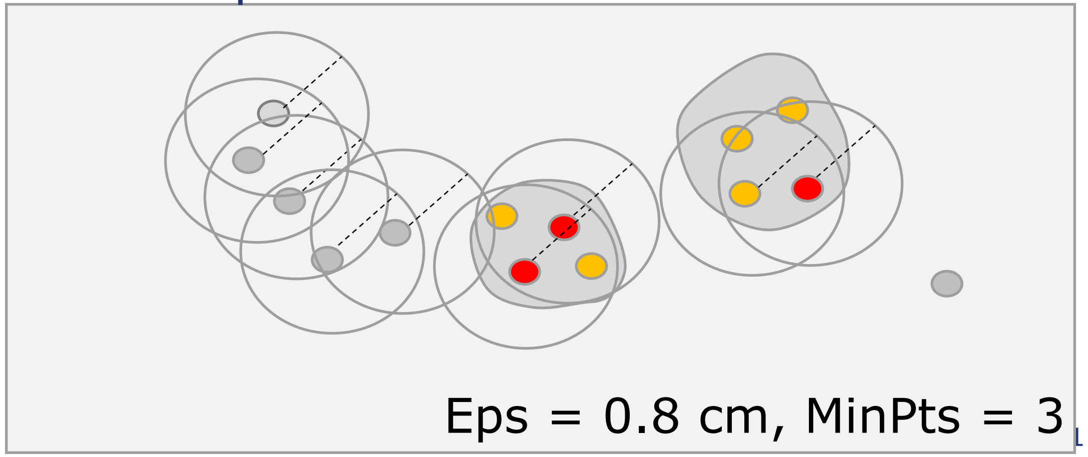
Eps = 1 cm, MinPts = 3

Larger eps → larger clusters



Eps = 1.5 cm, MinPts = 3

Smaller eps → more noise



Eps = 0.8 cm, MinPts = 3

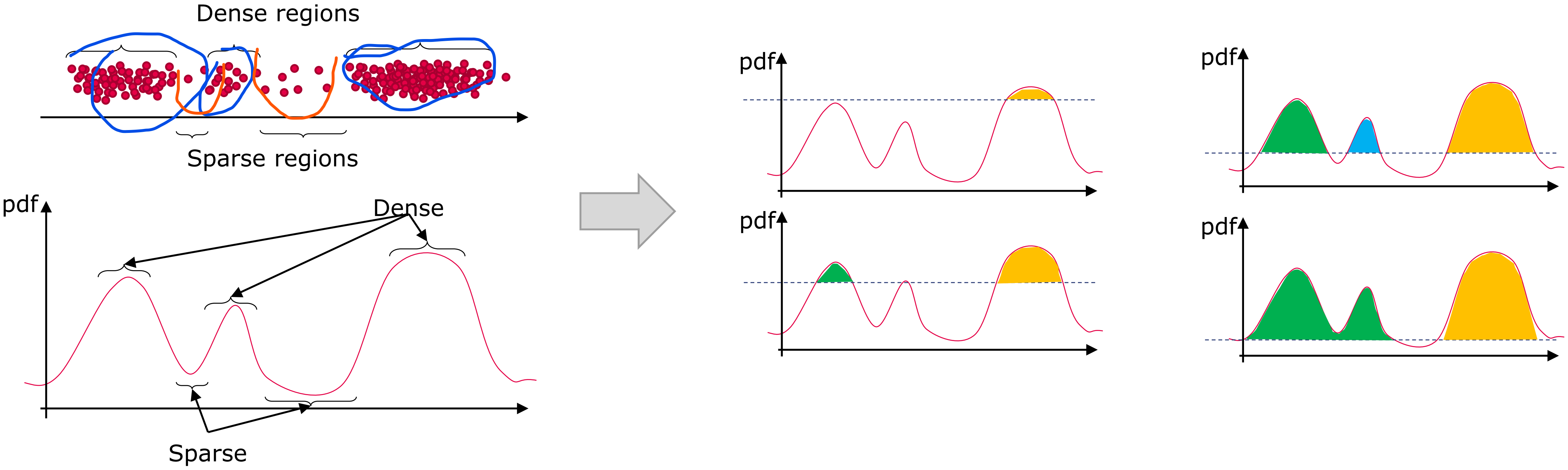
Coding session

fpc::dbscan(…)

dbscan::dbscan(…)

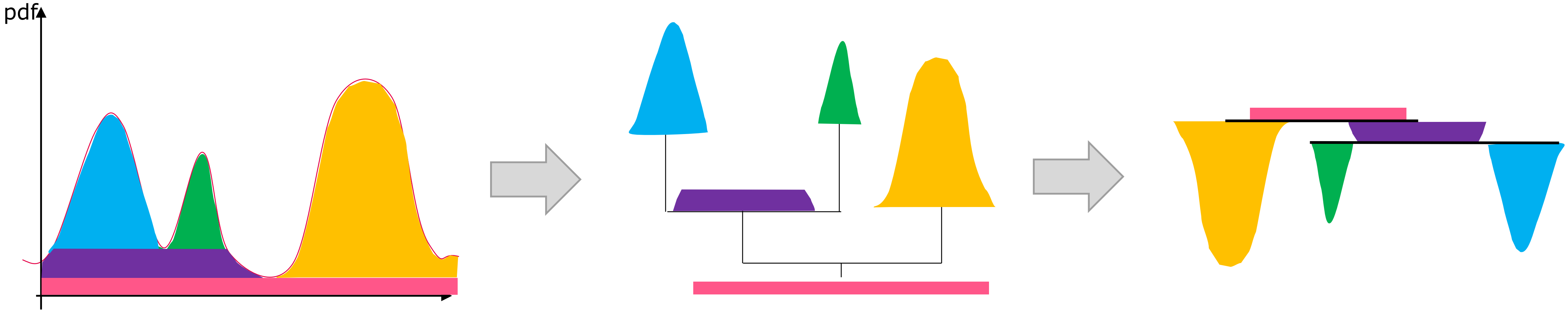
Hdbscan

- To understand Hdbscan we are going to take a synthetic example in **one dimension**



Hdbscan

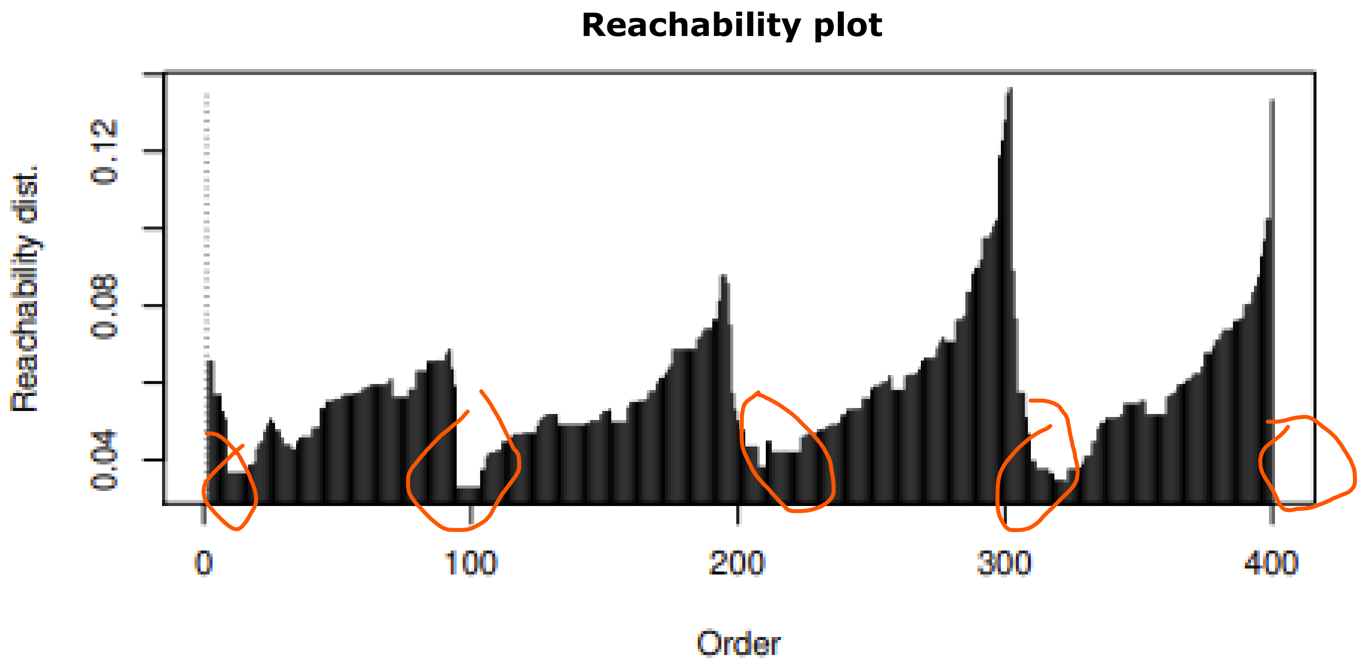
Hierarchical representation of densities

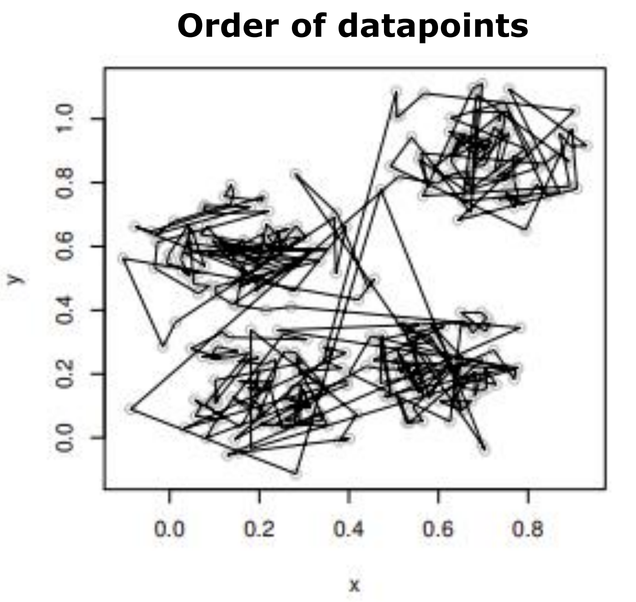


OPTICS (just a short mention)

Main concept

looking for valleys

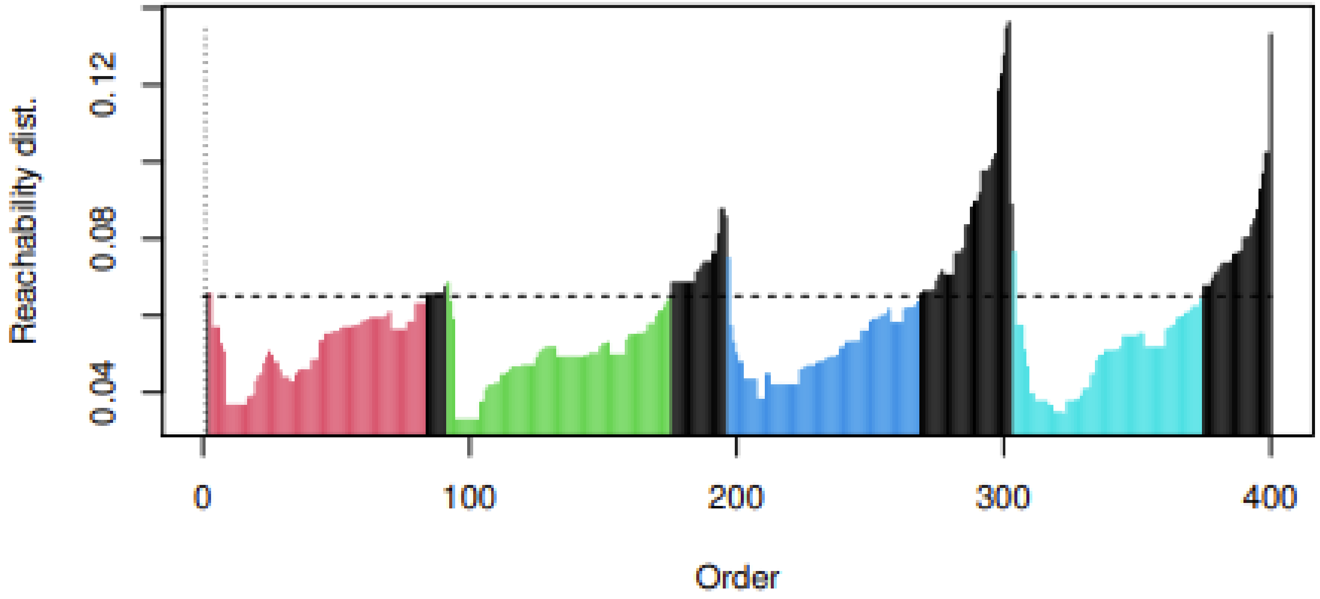




OPTICS

Extracting clusters

- Static threshold (eps)

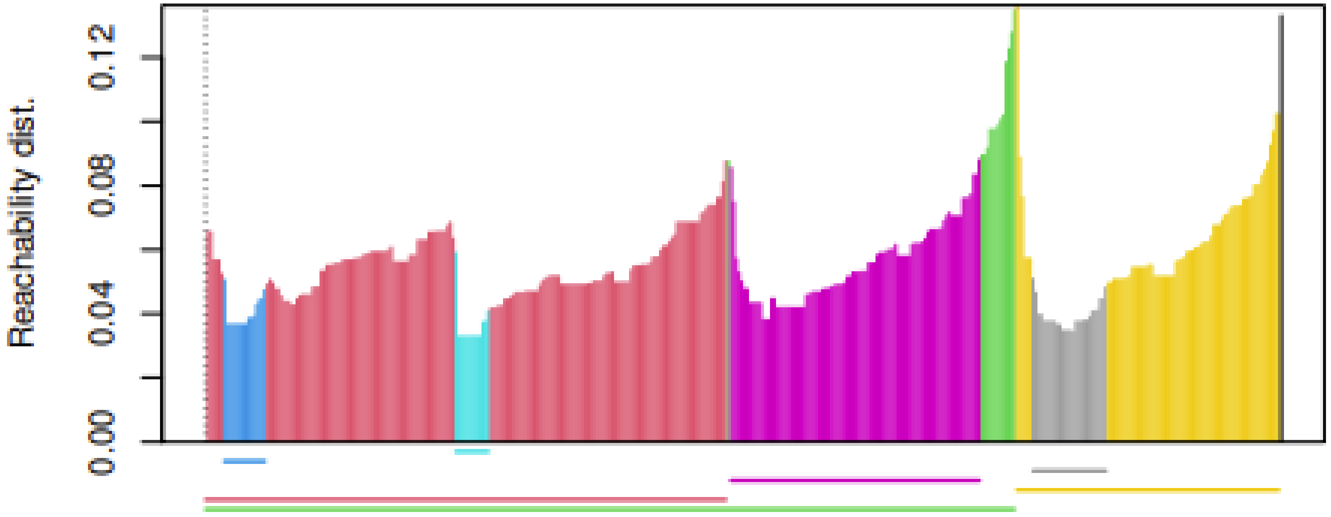


- Extract clusters by setting an eps threshold.
- Result is equivalent to dbscan, with the exception of border points (here marked as noise)

OPTICS

Extracting clusters

- Dynamic threshold (Xi)



- Extract clusters hierarchically based on the steepness of the reachability plot.
- Xi = Change in relative cluster density. T

Coding session

dbscan::hdbscan(…)

dbscan::optics(…)

One last word on clustering

On the similarity measure

- The choice of the similarity measure has a large impact in the clustering results
- The choice is not easy and at the beginning there will be a lot of trial & error.
  - Do not use the same similarity measure if features are of different type (unless it was designed for this purpose). One-hot-encoding must be used only with extreme caution!
  - ***Euclidean*** and ***Manhattan*** are very good for compact and isolated clusters. They are sensible to outliers and to the number of dimensions, they should not be used for many dimensions.
  - ***Cosine*** is very popular for very large number of dimensions (documents, webpages, trajectories)
  - ***Jaccard*** or ***Dice*** are very popular and almost always a good choice for binary features.
  - ***Correlation****-based* (**Pearson**, **Spearman**) measures can also be used when we are not interested in the geometrical distance but rather in their correlation.
  - ***Gower*** is a popular distance for mixed data types.
  - For very large number of dimensions, other clustering techniques exist (e.g. ***subspace*** clustering)

Coding session

daisy::gower()