MADS-MMS – Mathematics and Multivariate Statistics

Density-Based Clustering

Prof. Dr. Stephan Doerfel

FACHHOCHSCHULE KIEL
University of Applied Sciences

Moodle (WiSe 24/25)

Outline

Motivation

- k-means has various limitations:

Motivation

- k-means has various limitations:
  - fixed on Euclidean distance

Motivation

- k-means has various limitations:
  - fixed on Euclidean distance
  - cannot discover outliers/noise

Motivation

- k-means has various limitations:
  - fixed on Euclidean distance
  - cannot discover outliers/noise
  - discovers compact, round centered shapes

Motivation

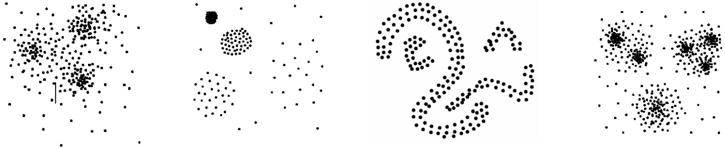
- k-means has various limitations:
  - fixed on Euclidean distance
  - cannot discover outliers/noise
  - discovers compact, round centered shapes
  - needs the number of clusters upfront

Motivation

- k-means has various limitations:
  - fixed on Euclidean distance
  - cannot discover outliers/noise
  - discovers compact, round centered shapes
  - needs the number of clusters upfront
- exploit the idea of density rather than of distance to a central entity

Examples

Clusters of different size, form, density, and hierarchical structure



Chapter Goals

- understand the basics of density based clustering

Chapter Goals

- understand the basics of density based clustering
- understand and apply parameters of DBSCAN and their influence on the resulting clustering

Chapter Goals

- understand the basics of density based clustering
- understand and apply parameters of DBSCAN and their influence on the resulting clustering
- understand the difference between clusters and noise

Chapter Goals

- understand the basics of density based clustering
- understand and apply parameters of DBSCAN and their influence on the resulting clustering
- understand the difference between clusters and noise
- understand and apply OPTICS to choose good DBSCAN parameters or to create OPTICS clusterings

Outline

Basics

Idea:

Basics

Idea:

- clusters are regions in space, that are densely populated with data instances

Basics

Idea:

- clusters are regions in space, that are densely populated with data instances
- clusters are separated by regions that are sparsely populated with data instances

Basics

Idea:

- clusters are regions in space, that are densely populated with data instances
- clusters are separated by regions that are sparsely populated with data instances

Handling Noise:

datasets are rarely clean and contain noise

→ noise detection

Idea: Density Based Clustering

Requirements:

Idea: Density Based Clustering

Requirements:

- each object that belongs to a cluster comes from a neighborhood whose data-density exceeds a fixed threshold.

Idea: Density Based Clustering

Requirements:

- each object that belongs to a cluster comes from a neighborhood whose data-density exceeds a fixed threshold.
- the set of instances belonging to one cluster is "connected".

Idea: Density Based Clustering

Requirements:

- each object that belongs to a cluster comes from a neighborhood whose data-density exceeds a fixed threshold.
- the set of instances belonging to one cluster is "connected".

Ingredients:

Idea: Density Based Clustering

Requirements:

- each object that belongs to a cluster comes from a neighborhood whose data-density exceeds a fixed threshold.
- the set of instances belonging to one cluster is “connected”.

Ingredients:

- distinction between noise and valid data

Idea: Density Based Clustering

Requirements:

- each object that belongs to a cluster comes from a neighborhood whose data-density exceeds a fixed threshold.
- the set of instances belonging to one cluster is "connected".

Ingredients:

- distinction between noise and valid data
- notion of density

Idea: Density Based Clustering

Requirements:

- each object that belongs to a cluster comes from a neighborhood whose data-density exceeds a fixed threshold.
- the set of instances belonging to one cluster is "connected".

Ingredients:

- distinction between noise and valid data
- notion of density
- neighborhoods

Idea: Density Based Clustering

Requirements:

- each object that belongs to a cluster comes from a neighborhood whose data-density exceeds a fixed threshold.
- the set of instances belonging to one cluster is “connected”.

Ingredients:

- distinction between noise and valid data
- notion of density
- neighborhoods
- selection of density threshold

Neighborhoods and Core Points

Definition 1 (Neighborhood)

Let D be a dataset and $\operatorname{dist}(\cdot,\cdot)$ a distance function. For an instance $o \in D$ and a given radius ε, the neighborhood of o w.r.t. ε is given as

$$N_\varepsilon(o) = \{p \in D \mid \operatorname{dist}(o,p) \leq \varepsilon\}.$$

Neighborhoods and Core Points

Definition 1 (Neighborhood)

Let D be a dataset and $\text{dist}(\cdot,\cdot)$ a distance function. For an instance $o \in D$ and a given radius ε, the neighborhood of o w.r.t. ε is given as

$$N_\varepsilon(o) = \{p \in D \mid \text{dist}(o,p) \leq \varepsilon\}.$$

Definition 2 (Core Point)

Let D, $\text{dist}(\cdot,\cdot)$, ε and o be as above. Then instance o is called a **core point** w.r.t. the radius ε and the threshold MinPts, if:

$$|N_\varepsilon(o)| \geq \text{MinPts}.$$

Neighborhoods and Core Points

Definition 1 (Neighborhood)

Let D be a dataset and $\mathrm{dist}(\cdot, \cdot)$ a distance function. For an instance $o \in D$ and a given radius ε, the neighborhood of o w.r.t. ε is given as

$$N_\varepsilon(o) = \{p \in D \mid \mathrm{dist}(o, p) \leq \varepsilon\}.$$

Definition 2 (Core Point)

Let D, $\mathrm{dist}(\cdot, \cdot)$, ε and o be as above. Then instance o is called a **core point** w.r.t. the radius ε and the threshold MinPts, if:

$$|N_\varepsilon(o)| \geq \mathrm{MinPts}.$$

With the parameter MinPts, we control the required minimum density of neighborhoods.

Definition 3 (Reachability)

For a dataset D, distance function $\mathrm{dist}(\cdot, \cdot)$, and parameters ε and MinPts:

- a data instance $p \in D$ is called **directly (density-)reachable** from $q \in D$ if $p \in N_\varepsilon(q)$ and q is a core point.

Definition 3 (Reachability)

For a dataset D, distance function $\text{dist}(\cdot, \cdot)$, and parameters ε and MinPts:

- a data instance $p \in D$ is called **directly (density-)reachable** from $q \in D$ if $p \in N_\varepsilon(q)$ and q is a core point.
- a data instance $p \in D$ is called **(density-)reachable** if there is a chain of directly reachable objects from q to p.

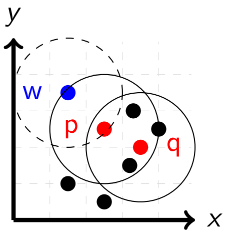
Definition 3 (Reachability)

For a dataset D, distance function $\text{dist}(\cdot, \cdot)$, and parameters ε and MinPts:

- a data instance $p \in D$ is called **directly (density-)reachable** from $q \in D$ if $p \in N_\varepsilon(q)$ and q is a core point.
- a data instance $p \in D$ is called **(density-)reachable** if there is a chain of directly reachable objects from q to p.
- two data instances p and q are called **(density-)connected**, if both are reachable from the same instance $o \in D$.

Examples: Reachability

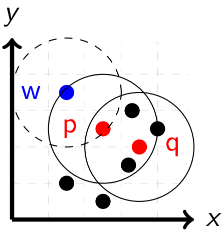
Use Euclidean distance and set $\varepsilon = 1.5cm$, MinPts $= 3$



Examples: Reachability

Use Euclidean distance and set $\varepsilon = 1.5cm$, MinPts $= 3$

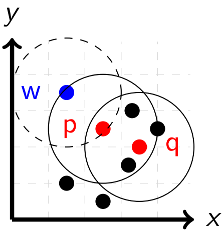
- p and q are core points



Examples: Reachability

Use Euclidean distance and set $\varepsilon = 1.5cm$, $\text{MinPts} = 3$

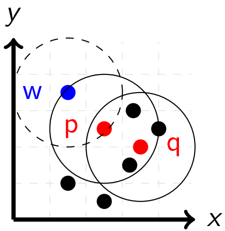
- p and q are core points
- w is not a core point



Examples: Reachability

Use Euclidean distance and set $\varepsilon = 1.5cm$, $\text{MinPts} = 3$

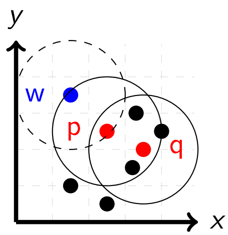
- p and q are core points
- w is not a core point
- w is directly reachable from p but not from q



Examples: Reachability

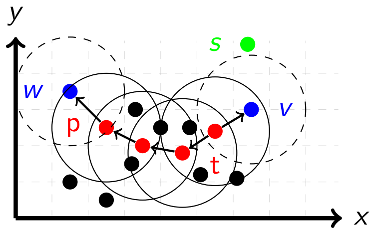
Use Euclidean distance and set $\varepsilon = 1.5cm$, MinPts $= 3$

- p and q are core points
- w is not a core point
- w is directly reachable from p but not from q
- w is reachable from q (via p)



Examples: Reachability and Connectedness

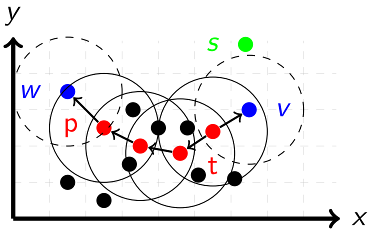
Use Euclidean distance and set $\varepsilon = 1.5cm$, MinPts $= 3$



Examples: Reachability and Connectedness

Use Euclidean distance and set $\varepsilon = 1.5cm$, $\text{MinPts} = 3$

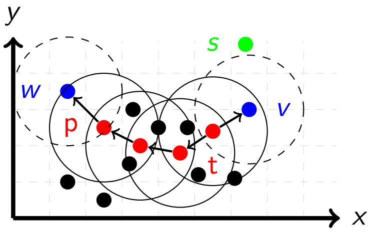
- s is not reachable from any instance



Examples: Reachability and Connectedness

Use Euclidean distance and set $\varepsilon = 1.5cm$, $\text{MinPts} = 3$

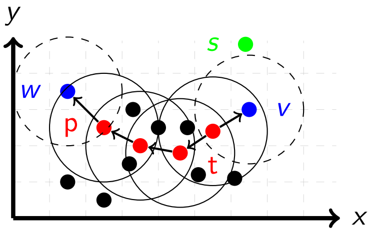
- s is not reachable from any instance
- w is reachable from t



Examples: Reachability and Connectedness

Use Euclidean distance and set $\varepsilon = 1.5cm$, MinPts $= 3$

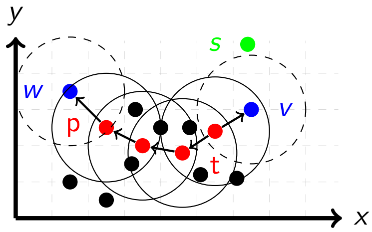
- s is not reachable from any instance
- w is reachable from t
- t is not reachable from w



Examples: Reachability and Connectedness

Use Euclidean distance and set $\varepsilon = 1.5cm$, $\text{MinPts} = 3$

- s is not reachable from any instance
- w is reachable from t
- t is not reachable from w
- v,w are connected



Remarks on Reachability

- reachability is a parametrized property – depending on the choices of ε and MinPts

Remarks on Reachability

- reachability is a parametrized property – depending on the choices of ε and MinPts
- works with any distance function

Remarks on Reachability

- reachability is a parametrized property – depending on the choices of ε and MinPts
- works with any distance function
- reachability is not symmetric!

Remarks on Reachability

- reachability is a parametrized property – depending on the choices of ε and MinPts
- works with any distance function
- reachability is not symmetric!
- reachability implies that all instances in the chain of direct reachability are core points

Remarks on Reachability

- reachability is a parametrized property – depending on the choices of ε and MinPts
- works with any distance function
- reachability is not symmetric!
- reachability implies that all instances in the chain of direct reachability are core points
- if p is reachable from q, then p and q are connected

Remarks on Reachability

- reachability is a parametrized property – depending on the choices of ε and MinPts
- works with any distance function
- reachability is not symmetric!
- reachability implies that all instances in the chain of direct reachability are core points
- if p is reachable from q, then p and q are connected
- reachability gives rise to a notion of noise: everything that is not reachable!

Density-based Clusters

Definition 4 (Density Cluster)

A **density cluster** C w.r.t. ε and MinPts is a non-empty subset of the data D for which hold:

maximality : $\forall p, q \in D : p \in C \wedge q$ reachable from $p \Rightarrow q \in C$

connectedness : $\forall p, q \in C : p$ and q are connected.

Density-based Clusters

Definition 4 (Density Cluster)

A density cluster C w.r.t. ε and MinPts is a non-empty subset of the data D for which hold:

maximality : $\forall p, q \in D : p \in C \wedge q$ reachable from $p \Rightarrow q \in C$

connectedness : $\forall p, q \in C : p$ and q are connected.

Definition 5 (Density Clustering)

A density clustering $\mathcal{C}$ of the data D w.r.t. ε and MinPts is the set of all density clusters.

Density-based Clusters

Definition 4 (Density Cluster)

A **density cluster** C w.r.t. ε and MinPts is a non-empty subset of the data D for which hold:

maximality : $\forall p, q \in D : p \in C \wedge q$ reachable from $p \Rightarrow q \in C$

connectedness : $\forall p, q \in C : p$ and q are connected.

Definition 5 (Density Clustering)

A **density clustering** $\mathcal{C}$ of the data D w.r.t. ε and MinPts is the set of all density clusters.

Definition 6 (Noise)

With the above definitions, $Noise_{\mathcal{C}}$ is defined as the set of all instances of D that belong to none of the clusters $C \in \mathcal{C}$.

Outline

Basics for DBSCAN

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

Basics for DBSCAN

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

Lemma 7 (Characterization)

Let $p \in D$ be a core point. Then the density cluster C that contains p can be constructed as

$$C = \{q \in D \mid q \text{ is reachable from } p\}.$$

DBSCAN: For each core-point, add all reachable instances to the same cluster.

DBSCAN: Main Loop

```
DBSCAN (D: dataset, ε: float, MinPts: int):
  mark each instance as unvisited
  C = 0 # ClusterID
  for each instance p in D:
    if (p has been visited):
      continue
    mark p visited
    if (|N_ε(p)| < MinPts):
      mark p NOISE
    else:
      C++;
      expand_cluster(p, N_ε(p) \ p, C, ε, MinPts)
```

DBSCAN: expand_Cluster

```
expand_cluster(p: instance, L: list of instances,
ε: float, MinPts: int):
  mark p with  C
  for each point q in L:
    if (q not visited):
      mark q visited
      if (|N_ε(q)| >= MinPts):
      L = N_ε(q) ∪ L
    if (q is not yet member of any cluster):
      unmark q as NOISE
      mark q with C
```

DBSCAN: expand_Cluster

```
expand_cluster(p: instance, L: list of instances,
ε: float, MinPts: int):
  mark p with  C
  for each point q in L:
    if (q not visited):
      mark q visited
      if (|N_ε(q)| >= MinPts):
      L = N_ε(q) ∪ L
    if (q is not yet member of any cluster):
      unmark q as NOISE
      mark q with C
```

Notebook 09_1_density_synthetic, Cells 1–12

Discussion of DBSCAN

positive:

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes
- solution is unique (except for cluster enumeration and non-core-point memberships)

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes
- solution is unique (except for cluster enumeration and non-core-point memberships)
- works with any distance function

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes
- solution is unique (except for cluster enumeration and non-core-point memberships)
- works with any distance function
- noise is separated from data

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes
- solution is unique (except for cluster enumeration and non-core-point memberships)
- works with any distance function
- noise is separated from data

negative:

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes
- solution is unique (except for cluster enumeration and non-core-point memberships)
- works with any distance function
- noise is separated from data

negative:

- worst case complexity $O(n^2)$

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes
- solution is unique (except for cluster enumeration and non-core-point memberships)
- works with any distance function
- noise is separated from data

negative:

- worst case complexity $O(n^2)$
- computing ε-neighborhoods is expansive

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes
- solution is unique (except for cluster enumeration and non-core-point memberships)
- works with any distance function
- noise is separated from data

negative:

- worst case complexity $O(n^2)$
- computing ε-neighborhoods is expansive
- suitable ε, MinPts must be chosen

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes
- solution is unique (except for cluster enumeration and non-core-point memberships)
- works with any distance function
- noise is separated from data

negative:

- worst case complexity $O(n^2)$
- computing ε-neighborhoods is expansive
- suitable $\varepsilon, \text{MinPts}$ must be chosen
- does not allow for differently dense clusters

Discussion of DBSCAN

positive:

- number of clusters must not be set upfront
- clusters can obtain arbitrary geometric shapes
- solution is unique (except for cluster enumeration and non-core-point memberships)
- works with any distance function
- noise is separated from data

negative:

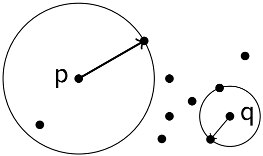
- worst case complexity $O(n^2)$
- computing ε-neighborhoods is expansive
- suitable ε, MinPts must be chosen
- does not allow for differently dense clusters

Exercises 1–2

Outline

Parameters

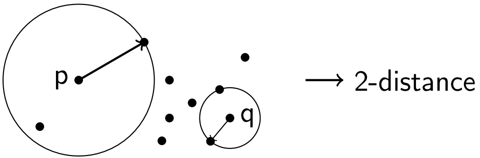
- determine good values for ε, MinPts



$\longrightarrow$ 2-distance

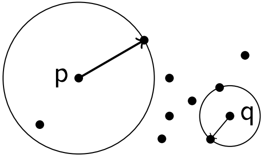
Parameters

- determine good values for ε, MinPts
- heuristic approach: order instances by their distance to their *k*-nearest neighbors



Parameters

- determine good values for ε, MinPts
- heuristic approach: order instances by their distance to their k-nearest neighbors



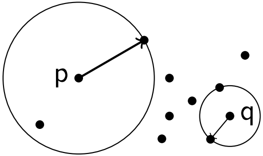
$\longrightarrow$ 2-distance

Definition 8 (k-distance)

In the above setting, the **k-distance** of an instance p is: $\mathrm{dist}_k(p) \coloneqq \min\{\varepsilon \in \mathbb{R}_{\geq 0} \mid |N_\varepsilon(p)| > k\}$. (Note: p is not counted as its own neighbor.)

Parameters

- determine good values for ε, MinPts
- heuristic approach: order instances by their distance to their k-nearest neighbors



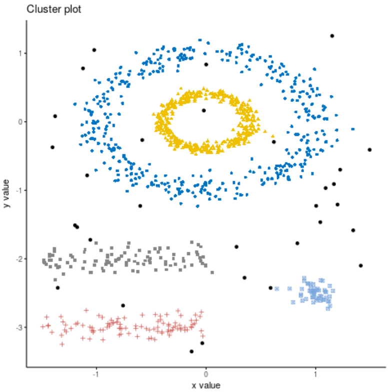
$\longrightarrow$ 2-distance

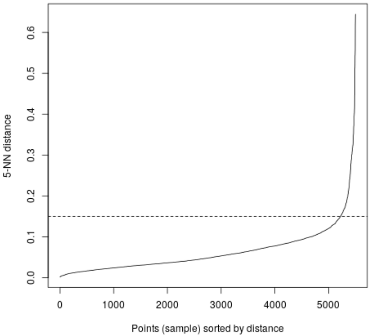
Definition 8 (k-distance)

In the above setting, the **k-distance** of an instance p is:
$\mathrm{dist}_k(p) := \min\{\varepsilon \in \mathbb{R}_{\geq 0} \mid |N_\varepsilon(p)| > k\}$. (Note: p is not counted as its own neighbor.)

- k-distance diagram: the ordered k-distances of all objects

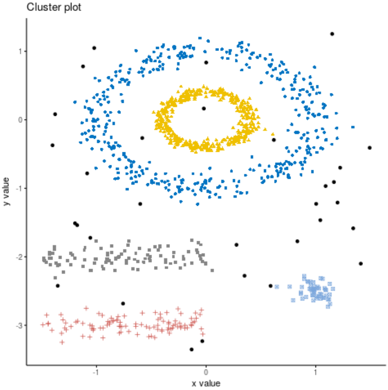
Heuristic Choice of Parameters

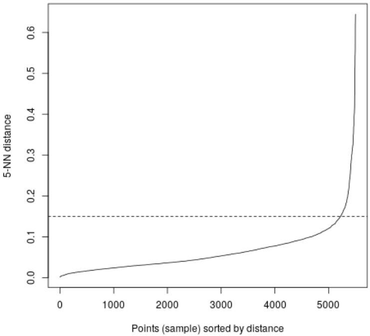




- choose k (rule of thumb $k > d$), and MinPts $:= k + 1$

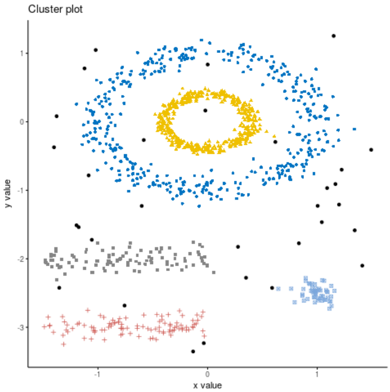
Heuristic Choice of Parameters

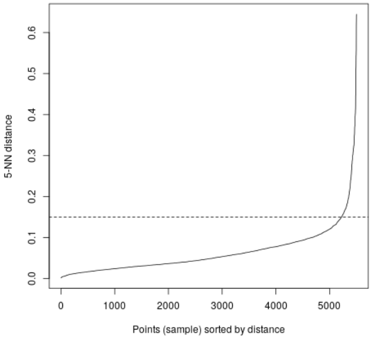




- choose k (rule of thumb $k > d$), and MinPts $:= k + 1$
- compute k-distance diagram

Heuristic Choice of Parameters

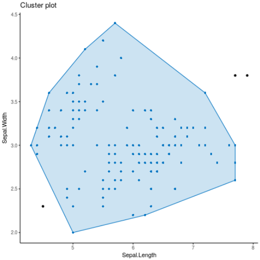


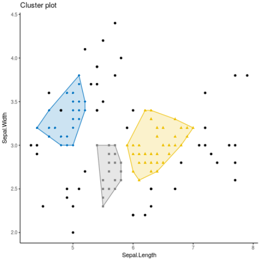


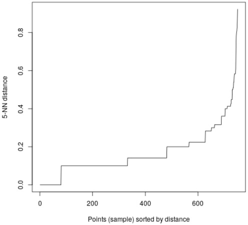
- choose k (rule of thumb $k > d$), and MinPts $:= k + 1$
- compute k-distance diagram
- choose instance o as border object (elbow in graph) and set $\varepsilon := \mathrm{dist}_k(o)$

Problem: Hierarchical Clusters

An example from IRIS



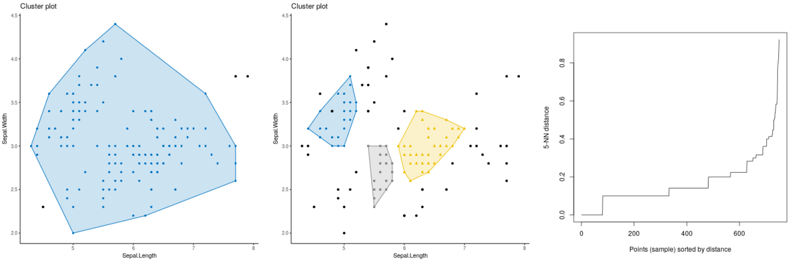




- hierarchical clusters

Problem: Hierarchical Clusters

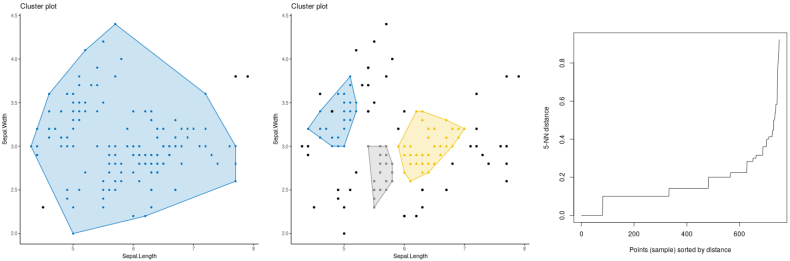
An example from IRIS



- hierarchical clusters
- strong differences in density in different parts of the space

Problem: Hierarchical Clusters

An example from IRIS



- hierarchical clusters
- strong differences in density in different parts of the space
- clusters and noise are not well separated

Outline

Observation

Theorem 9 (Cluster Monotony)

Let D be a dataset and $\mathcal{C}$ a density based clustering with ε and MinPts. Let $\mathcal{C}'$ be another such clustering with MinPts and $\varepsilon' \geq \varepsilon$. Then, for each $C \in \mathcal{C}$ there exists a $C' \in \mathcal{C}'$ such that $C \subseteq C'$.

Observation

Theorem 9 (Cluster Monotony)

Let D be a dataset and $\mathcal{C}$ a density based clustering with ε and MinPts. Let $\mathcal{C}'$ be another such clustering with MinPts and $\varepsilon' \geq \varepsilon$. Then, for each $C \in \mathcal{C}$ there exists a $C' \in \mathcal{C}'$ such that $C \subseteq C'$.

Remark:

Observation

Theorem 9 (Cluster Monotony)

Let D be a dataset and $\mathcal{C}$ a density based clustering with ε and MinPts. Let $\mathcal{C}'$ be another such clustering with MinPts and $\varepsilon' \geq \varepsilon$. Then, for each $C \in \mathcal{C}$ there exists a $C' \in \mathcal{C}'$ such that $C \subseteq C'$.

Remark:

- lowering ε makes the density constraint harder to meet

Observation

Theorem 9 (Cluster Monotony)

Let D be a dataset and $\mathcal{C}$ a density based clustering with ε and MinPts. Let $\mathcal{C}'$ be another such clustering with MinPts and $\varepsilon' \geq \varepsilon$. Then, for each $C \in \mathcal{C}$ there exists a $C' \in \mathcal{C}'$ such that $C \subseteq C'$.

Remark:

- lowering ε makes the density constraint harder to meet
- thus, clusters decompose into smaller subsets and noise

Outline

- Motivation
- Basics
- DBSCAN
- Choosing the Hyperparameters
- Density-based Hierarchical Clustering
- **OPTICS**

Density based Hierarchical Clustering

Idea

Density based Hierarchical Clustering

Idea

- create a data structure from which one can obtain DBSCAN clusterings for different choices of ε in **linear(!)** time

Density based Hierarchical Clustering

Idea

- create a data structure from which one can obtain DBSCAN clusterings for different choices of ε in **linear(!)** time
- Ordering Points To Identify the Clustering Structure ➔ OPTICS

Density based Hierarchical Clustering

Idea

- create a data structure from which one can obtain DBSCAN clusterings for different choices of ε in **linear(!)** time
- Ordering Points To Identify the Clustering Structure ➔ OPTICS

Steps

Density based Hierarchical Clustering

Idea

- create a data structure from which one can obtain DBSCAN clusterings for different choices of ε in **linear(!)** time
- Ordering Points To Identify the Clustering Structure ➔ OPTICS

Steps

- fix MinPts

Density based Hierarchical Clustering

Idea

- create a data structure from which one can obtain DBSCAN clusterings for different choices of ε in **linear(!)** time
- Ordering Points To Identify the Clustering Structure ➔ OPTICS

Steps

- fix MinPts
- fix an upper bound for ε

Density based Hierarchical Clustering

Idea

- create a data structure from which one can obtain DBSCAN clusterings for different choices of ε in **linear(!)** time
- Ordering Points To Identify the Clustering Structure ➔ OPTICS

Steps

- fix MinPts
- fix an upper bound for ε
- find a clever ordering of points

Density based Hierarchical Clustering

Idea

- create a data structure from which one can obtain DBSCAN clusterings for different choices of ε in **linear(!)** time
- Ordering Points To Identify the Clustering Structure $\rightarrow$ OPTICS

Steps

- fix MinPts
- fix an upper bound for ε
- find a clever ordering of points
- compute the minimum ε for which a point is still reachable from its predecessors

Density based Hierarchical Clustering

Idea

- create a data structure from which one can obtain DBSCAN clusterings for different choices of ε in **linear(!)** time
- Ordering Points To Identify the Clustering Structure → OPTICS

Steps

- fix MinPts
- fix an upper bound for ε
- find a clever ordering of points
- compute the minimum ε for which a point is still reachable from its predecessors
- the upper bound is used to determine whether an instance can be a core point at all (for this or lower values of ε)

The OPTICS Algorithm

Principle: Given fix MinPts and ε, in a DBSCAN-like procedure yield a data structure with three components:

The OPTICS Algorithm

Principle: Given fix MinPts and ε, in a DBSCAN-like procedure yield a data structure with three components:

1. the core distance for each instance

The OPTICS Algorithm

Principle: Given fix MinPts and ε, in a DBSCAN-like procedure yield a data structure with three components:

1. the core distance for each instance
2. an order on the dataset

The OPTICS Algorithm

Principle: Given fix MinPts and ε, in a DBSCAN-like procedure yield a data structure with three components:

1. the core distance for each instance
2. an order on the dataset
3. a reachability distance for each instance

The OPTICS Algorithm

Principle: Given fix MinPts and ε, in a DBSCAN-like procedure yield a data structure with three components:

1. the core distance for each instance
2. an order on the dataset
3. a reachability distance for each instance

Visual Representation: a reachability distance diagram, which is readable even for very large datasets with high dimensionality

Core Distance

Definition 10 (Core Distance)

For an instance o of a dataset D with distance dist, the **core distance** of o, given ε and MinPts is

$$\underset{\varepsilon,\text{MinPts}}{core-\text{dist}}(o) := \begin{cases} \textit{undefined} & |N_\varepsilon(o)| < \text{MinPts} \\ \text{dist}_{\text{MinPts}-1}(o) & \text{else} \end{cases}$$

Core Distance

Definition 10 (Core Distance)

For an instance o of a dataset D with distance dist, the **core distance** of o, given ε and MinPts is

$$\underset{\varepsilon,\mathrm{MinPts}}{core - \mathrm{dist}}(o) := \begin{cases} \textit{undefined} & |N_\varepsilon(o)| < \mathrm{MinPts} \\ \mathrm{dist}_{\mathrm{MinPts}-1}(o) & \text{else} \end{cases}$$

In a clustering with MinPts and $\varepsilon' \leq \varepsilon$, o will be a core point if $core - \mathrm{dist}_{\varepsilon,\mathrm{MinPts}}(o) \leq \varepsilon'$.

Reachability Distance

Definition 11 (Reachability Distance)

For an instances p and o of a dataset D with distance dist, the reachability distance of p from o, given ε and MinPts is

$$\underset{\varepsilon,\mathrm{MinPts}}{\mathit{reach} - \mathrm{dist}}(o, p) := \begin{cases} \mathit{undefined} \text{ if } |N_\varepsilon(o)| < \mathrm{MinPts} \\ \max\{\mathit{core} - \mathrm{dist}_{\varepsilon,\mathrm{MinPts}}(o), \mathrm{dist}(o, p)\} \text{ else.} \end{cases}$$

Reachability Distance

Definition 11 (Reachability Distance)

For an instances p and o of a dataset D with distance dist, the reachability distance of p from o, given ε and MinPts is

$$\underset{\varepsilon,\mathrm{MinPts}}{\mathit{reach}} - \mathrm{dist}(o,p) := \begin{cases} \mathit{undefined} \text{ if } |N_\varepsilon(o)| < \mathrm{MinPts} \\ \max\{\mathit{core} - \mathrm{dist}_{\varepsilon,\mathrm{MinPts}}(o), \mathrm{dist}(o,p)\} \text{ else.} \end{cases}$$

The reachability distance of p from o tells us, for which $\varepsilon' \leq \varepsilon$ is directly reachable, meaning

Reachability Distance

Definition 11 (Reachability Distance)

For an instances p and o of a dataset D with distance dist, the reachability distance of p from o, given ε and MinPts is

$$\underset{\varepsilon,\mathrm{MinPts}}{\mathit{reach} - \mathrm{dist}}(o, p) := \begin{cases} \mathit{undefined} \text{ if } |N_\varepsilon(o)| < \mathrm{MinPts} \\ \max\{\mathit{core} - \mathrm{dist}_{\varepsilon,\mathrm{MinPts}}(o), \mathrm{dist}(o, p)\} \text{ else.} \end{cases}$$

The reachability distance of p from o tells us, for which $\varepsilon' \leq \varepsilon$ is directly reachable, meaning

- o must be a core point and

Reachability Distance

Definition 11 (Reachability Distance)

For an instances p and o of a dataset D with distance dist, the reachability distance of p from o, given ε and MinPts is

$$\underset{\varepsilon,\mathrm{MinPts}}{\mathit{reach}-\mathrm{dist}}(o,p) := \begin{cases} \mathit{undefined} \text{ if } |N_\varepsilon(o)| < \mathrm{MinPts} \\ \max\{\mathit{core}-\mathrm{dist}_{\varepsilon,\mathrm{MinPts}}(o), \mathrm{dist}(o,p)\} \text{ else.} \end{cases}$$

The reachability distance of p from o tells us, for which $\varepsilon' \leq \varepsilon$ is directly reachable, meaning

- o must be a core point and
- p must be in the neighborhood $N_\varepsilon(o) = \{p \in D \mid \mathrm{dist}(o,p) \leq \varepsilon\}$ of o.

Cluster Order

1. start with an arbitrary object

Cluster Order

1. start with an arbitrary object
2. as soon as the first object p with defined core distance is found, start an ordered priority list

Cluster Order

1. start with an arbitrary object
2. as soon as the first object p with defined core distance is found, start an ordered priority list
3. for each neighbor q of p do:

Cluster Order

1. start with an arbitrary object
2. as soon as the first object p with defined core distance is found, start an ordered priority list
3. for each neighbor q of p do:
   - if q is not on the list, add it with its reachability distance from p

Cluster Order

1. start with an arbitrary object
2. as soon as the first object p with defined core distance is found, start an ordered priority list
3. for each neighbor q of p do:
   - if q is not on the list, add it with its reachability distance from p
   - if q is on the list and the reachability distance from p is smaller than the previous reachability distance, replace the previous reachability distance

Cluster Order

1. start with an arbitrary object
2. as soon as the first object p with defined core distance is found, start an ordered priority list
3. for each neighbor q of p do:
   - if q is not on the list, add it with its reachability distance from p
   - if q is on the list and the reachability distance from p is smaller than the previous reachability distance, replace the previous reachability distance
4. repeat 3 with the first object from from the ordered priority list until the list is empty.

Cluster Order

1. start with an arbitrary object
2. as soon as the first object p with defined core distance is found, start an ordered priority list
3. for each neighbor q of p do:
   - if q is not on the list, add it with its reachability distance from p
   - if q is on the list and the reachability distance from p is smaller than the previous reachability distance, replace the previous reachability distance
4. repeat 3 with the first object from from the ordered priority list until the list is empty.
5. repeat 2 with an arbitrary unprocessed instance.

Cluster Order

1. start with an arbitrary object
2. as soon as the first object p with defined core distance is found, start an ordered priority list
3. for each neighbor q of p do:
   - if q is not on the list, add it with its reachability distance from p
   - if q is on the list and the reachability distance from p is smaller than the previous reachability distance, replace the previous reachability distance
4. repeat 3 with the first object from from the ordered priority list until the list is empty.
5. repeat 2 with an arbitrary unprocessed instance.

Note that the ordered priority list is updated while it is processed (recursive algorithm).

Cluster Order

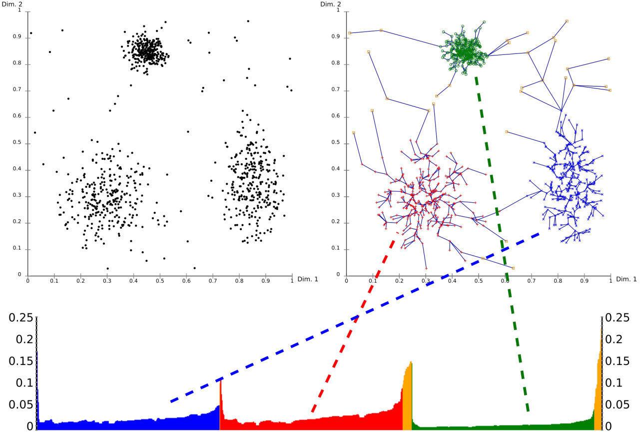
1. start with an arbitrary object
2. as soon as the first object p with defined core distance is found, start an ordered priority list
3. for each neighbor q of p do:
   - if q is not on the list, add it with its reachability distance from p
   - if q is on the list and the reachability distance from p is smaller than the previous reachability distance, replace the previous reachability distance
4. repeat 3 with the first object from from the ordered priority list until the list is empty.
5. repeat 2 with an arbitrary unprocessed instance.

Note that the ordered priority list is updated while it is processed (recursive algorithm).

Notebook 09_2_optics_toy_example

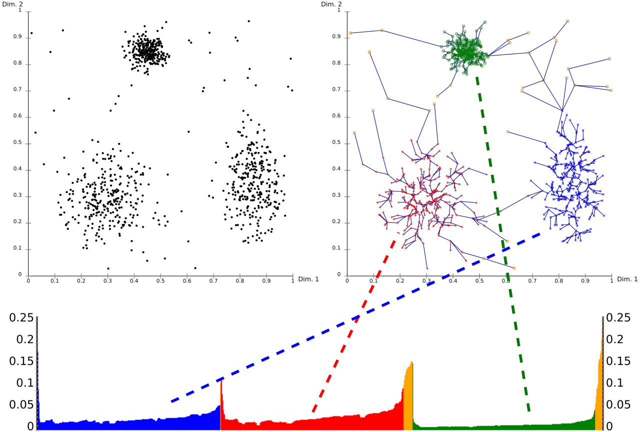
Reachability Diagram

- visualize reachability distances of objects, next to each other, ordered by the cluster order



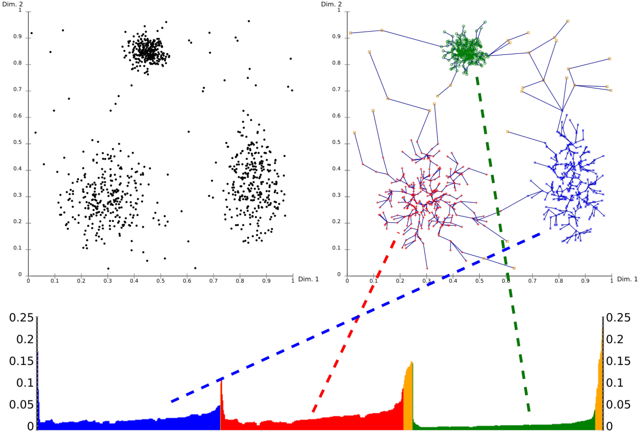
Reachability Diagram

- visualize reachability distances of objects, next to each other, ordered by the cluster order
- valleys are clusters

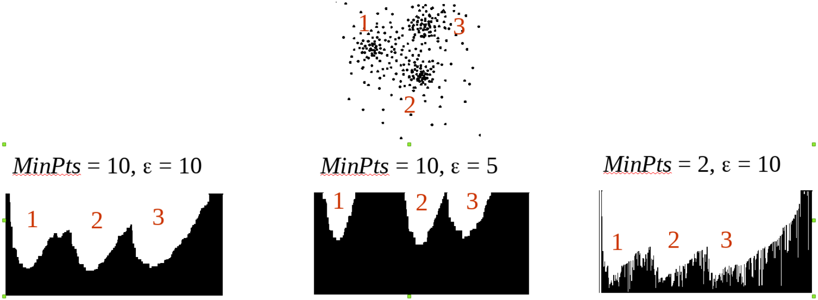


Reachability Diagram

- visualize reachability distances of objects, next to each other, ordered by the cluster order
- valleys are clusters
- the deeper the valley, the more dense the cluster



Example: Parameter Sensitivity



Extracting Clusterings

How to determine "a valley"?

Extracting Clusterings

How to determine "a valley"?

Distance Cut :

Extracting Clusterings

How to determine “a valley”?

Distance Cut :

- select a reachability distance (y-axis) in the reachability diagram

Extracting Clusterings

How to determine "a valley"?

Distance Cut :

- select a reachability distance (y-axis) in the reachability diagram
- points between two intersections form a DBSCAN cluster

Extracting Clusterings

How to determine "a valley"?

Distance Cut :

- select a reachability distance (y-axis) in the reachability diagram
- points between two intersections form a DBSCAN cluster

OPTICS ξ method : (sketch)

Extracting Clusterings

How to determine "a valley"?

Distance Cut :

- select a reachability distance (y-axis) in the reachability diagram
- points between two intersections form a DBSCAN cluster

OPTICS ξ method : (sketch)

- go through cluster order

Extracting Clusterings

How to determine "a valley"?

Distance Cut :

- select a reachability distance (y-axis) in the reachability diagram
- points between two intersections form a DBSCAN cluster

OPTICS ξ method : (sketch)

- go through cluster order
- start cluster when in an area reachability distance falls ξ-steep

Extracting Clusterings

How to determine "a valley"?

Distance Cut :

- select a reachability distance (y-axis) in the reachability diagram
- points between two intersections form a DBSCAN cluster

OPTICS ξ method : (sketch)

- go through cluster order
- start cluster when in an area reachability distance falls ξ-steep
- end cluster when in an area reachability rises ξ-steep

Extracting Clusterings

How to determine "a valley"?

Distance Cut :

- select a reachability distance (y-axis) in the reachability diagram
- points between two intersections form a DBSCAN cluster

OPTICS ξ method : (sketch)

- go through cluster order
- start cluster when in an area reachability distance falls ξ-steep
- end cluster when in an area reachability rises ξ-steep
- clusters contain at least MinPts instances

Notebook 09_1_density_synthetic, Cells 13–21

Extracting Clusterings

How to determine "a valley"?

Distance Cut :

- select a reachability distance (y-axis) in the reachability diagram
- points between two intersections form a DBSCAN cluster

OPTICS ξ method : (sketch)

- go through cluster order
- start cluster when in an area reachability distance falls ξ-steep
- end cluster when in an area reachability rises ξ-steep
- clusters contain at least MinPts instances

Notebook 09_1_density_synthetic, Cells 13–21

Exercises 3