Programa de Verão FGV EMAp 2019

Introduction to Machine Learning with Python

CLUSTERING TECHNIQUES

Prof. Luis Gustavo Nonato

University of São Paulo - São Carlos - SP

Clustering: learns a model that groups "similar" observations. The similarity criterion is predefined and application dependent. Input data is typically not annotated - unsupervised task.

Learning Strategies

Clustering: learns a model that groups "similar" observations. The similarity criterion is predefined and application dependent. Input data is typically not annotated - unsupervised task.

Applications were data is partially annotated are also common (semi-supervised tasks, more common for classification).

Learning Strategies

Clustering: learns a model that groups "similar" observations. The similarity criterion is predefined and application dependent. Input data is typically not annotated - unsupervised task.

Applications were data is partially annotated are also common (semi-supervised tasks, more common for classification).

(a) Original points.

(b) Two clusters.

(c) Four clusters.

(d) Six clusters.

Introduction

There are many clustering techniques:

Introduction

There are many clustering techniques:

- Hierarchical
  - Agglomerative
  - Divisive
  - $\vdots$
- Partitional
  - K-means
  - Mixture Resolving
  - Spectral Clustering
  - Density-based
  - $\vdots$

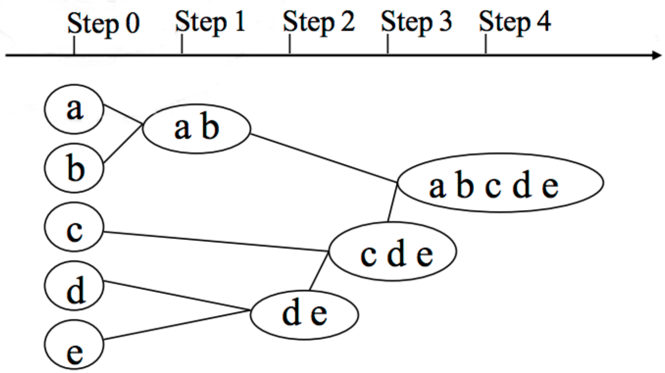
Introduction

There are many clustering techniques:

- Hierarchical
  - Agglomerative
  - Divisive
  - $\vdots$
- Partitional
  - K-means
  - Mixture Resolving
  - Spectral Clustering
  - Density-based
  - $\vdots$

Hierarchical Clustering

Hierarchical Clustering



Hierarchical Clustering: Agglomerative Method

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)
2. Find the most similar pair of clusters C_i and C_j and merge them into a single cluster

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)
2. Find the most similar pair of clusters C_i and C_j and merge them into a single cluster
3. Update distances between clusters

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)
2. Find the most similar pair of clusters C_i and C_j and merge them into a single cluster
3. Update distances between clusters
4. Repeat steps 2-3 until a single cluster (or the desired number of clusters) is obtained

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)
2. Find the most similar pair of clusters C_i and C_j and merge them into a single cluster
3. Update distances between clusters
4. Repeat steps 2-3 until a single cluster (or the desired number of clusters) is obtained

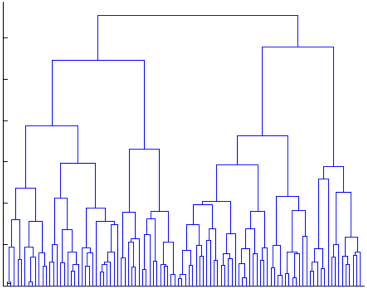
Step 3 can assume different forms:

$$d(C_a, C_b) = \min_{i \in C_a, j \in C_b} \{d(i,j)\} \quad \text{Single Link}$$

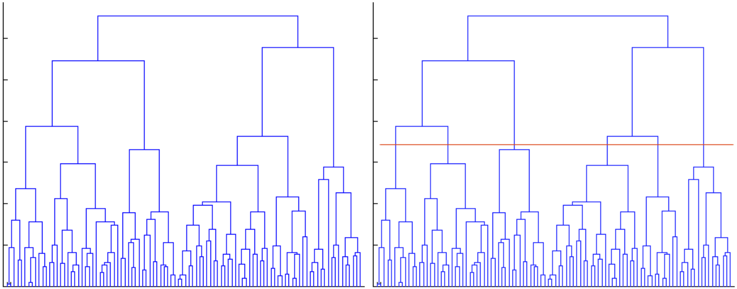
$$d(C_a, C_b) = \max_{i \in C_a, j \in C_b} \{d(i,j)\} \quad \text{Complete Link}$$

$$d(C_a, C_b) = \frac{1}{n_a n_b} \sum_{i \in C_a, j \in C_b} \{d(i,j)\} \quad \text{Average Link}$$

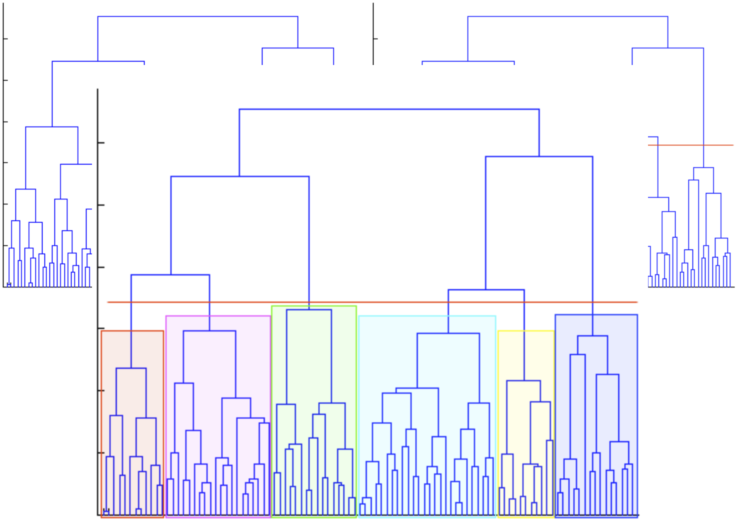
Dendogram



Dendogram



Dendogram



K-means

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

Mathematically this idea can be state as:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2$$

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

Mathematically this idea can be state as:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} \boxed{r_{ij}} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$$

$r_{ij} = 1$ if $\mathbf{x}_i$ in cluster j, 0 otherwise, and

K-means

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

Mathematically this idea can be state as:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} \boxed{r_{ij}} \|\mathbf{x}_i - \boxed{\boldsymbol{\mu}_j}\|^2$$

$r_{ij} = 1$ if $\mathbf{x}_i$ in cluster j, 0 otherwise, and
$\boldsymbol{\mu}_j$ is a "prototype" associated to $cluster_j$.

K-means

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

Mathematically this idea can be state as:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} \boxed{r_{ij}} \|\mathbf{x}_i - \boxed{\boldsymbol{\mu}_j}\|^2$$

$r_{ij} = 1$ if $\mathbf{x}_i$ in cluster j, 0 otherwise, and
$\boldsymbol{\mu}_j$ is a "prototype" associated to $cluster_j$.

The goal is to find $\{r_{ij}\}$ and $\{\boldsymbol{\mu}_j\}$ so as to minimize J.

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$$

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \| \mathbf{x}_i - \boldsymbol{\mu}_j \|^2$$

If the $\{\boldsymbol{\mu}_j\}$ are fixed then the minimum of J is reached when

$$r_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_s \|\mathbf{x}_i - \boldsymbol{\mu}_s\|^2 \\ 0 & \text{otherwise} \end{cases}$$

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$$

If the $\{\boldsymbol{\mu}_j\}$ are fixed then the minimum of J is reached when

$$r_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_s \|\mathbf{x}_i - \boldsymbol{\mu}_s\|^2 \\ 0 & \text{otherwise} \end{cases}$$

If $\{r_{ij}\}$ is fixed then the minimum can be obtained by setting the derivative of J w.r.t. $\boldsymbol{\mu}_j$ to zero, resulting in

$$\boldsymbol{\mu}_j = \frac{\sum_i r_{ij}\mathbf{x}_j}{\sum_i r_{ij}}$$

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \| \mathbf{x}_i - \boldsymbol{\mu}_j \|^2$$

If the $\{\boldsymbol{\mu}_j\}$ are fixed then the minimum of J is reached when

$$r_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_s \|\mathbf{x}_i - \boldsymbol{\mu}_s\|^2 \\ 0 & \text{otherwise} \end{cases}$$

If $\{r_{ij}\}$ is fixed then the minimum can be obtained by setting the derivative of J w.r.t. $\boldsymbol{\mu}_j$ to zero, resulting in

$$\boldsymbol{\mu}_j = \frac{\sum_i r_{ij} \mathbf{x}_j}{\sum_i r_{ij}}$$

$\boldsymbol{\mu}_j$ is simply the average of the $\mathbf{x}_i \in cluster_j$.

Algorithm

Algorithm

1. Initialize $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$

Algorithm

1. Initialize $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
2. Assign instances to their closest prototype $\boldsymbol{\mu}_j$

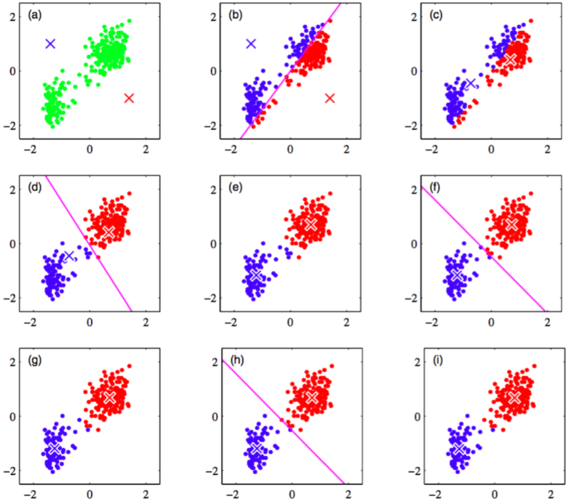
Algorithm

1. Initialize $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
2. Assign instances to their closest prototype $\boldsymbol{\mu}_j$
3. $\boldsymbol{\mu}_j = \frac{\sum_i r_{ij} \mathbf{x}_j}{\sum_i r_{ij}}$

Algorithm

1. Initialize $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
2. Assign instances to their closest prototype $\boldsymbol{\mu}_j$
3. $\boldsymbol{\mu}_j = \frac{\sum_i r_{ij} \mathbf{x}_j}{\sum_i r_{ij}}$
4. Repeat 2-3 until there are no changes in the prototypes

K-means



(figure extracted from Bishop's book)

- K-means always converges to a local minimum

K-means

- K-means always converges to a local minimum
- The algorithm can be adapted to deal with “distances” (dissimilarities) other than Euclidean

- K-means always converges to a local minimum
- The algorithm can be adapted to deal with “distances” (dissimilarities) other than Euclidean
- Unstable as to the initial centroids

- K-means always converges to a local minimum
- The algorithm can be adapted to deal with “distances” (dissimilarities) other than Euclidean
- Unstable as to the initial centroids
- Outliers can unduly influence the clusters

K-means

- K-means always converges to a local minimum
- The algorithm can be adapted to deal with "distances" (dissimilarities) other than Euclidean
- Unstable as to the initial centroids
- Outliers can unduly influence the clusters
- K-means generates spherically shaped clusters

- K-means always converges to a local minimum
- The algorithm can be adapted to deal with "distances" (dissimilarities) other than Euclidean
- Unstable as to the initial centroids
- Outliers can unduly influence the clusters
- K-means generates spherically shaped clusters
- K-means is a particular case of a more general method called *Mixture Resolving*

Bisecting K-means

Initialize the list of clusters to contain the cluster consisting of all points.
repeat
    Remove a cluster from the list of clusters.
    {Perform several “trial” bisections of the chosen cluster.}
    for $i = 1$ to *number of trials* **do**
        Bisect the selected cluster using basic K-means.
    end for
    Select the two clusters from the bisection with the lowest total SSE.
    Add these two clusters to the list of clusters.
until Until the list of clusters contains K clusters.

Mixture Resolving

Given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, a mixture solving algorithm aims to find parameters c_i, $\boldsymbol{\mu}_i$, Σ_i and a "responsibility" (membership) function γ_{ij} so as to maximize the likelihood

$$p(\mathbf{X}|\boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} \left(\sum_{j=1}^{k} c_j N(\mathbf{x}_i | \boldsymbol{\mu}_j, \Sigma_j) \right)$$

Given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, a mixture solving algorithm aims to find parameters c_i, $\boldsymbol{\mu}_i$, Σ_i and a "responsibility" (membership) function γ_{ij} so as to maximize the likelihood

$$p(\mathbf{X}|\boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} \left(\sum_{j=1}^{k} c_j N(\mathbf{x}_i | \boldsymbol{\mu}_j, \Sigma_j) \right)$$

the γ_{ij} should be such that $N(\mathbf{x}_i|\boldsymbol{\mu}_j, \Sigma_j) > N(\mathbf{x}_i|\boldsymbol{\mu}_s, \Sigma_s)$, $j \neq s$.

Mixture Resolving

Given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, a mixture solving algorithm aims to find parameters c_i, $\boldsymbol{\mu}_i$, Σ_i and a "responsibility" (membership) function γ_{ij} so as to maximize the likelihood

$$p(\mathbf{X}|\boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} \left(\sum_{j=1}^{k} c_j N(\mathbf{x}_i | \boldsymbol{\mu}_j, \Sigma_j) \right)$$

the γ_{ij} should be such that $N(\mathbf{x}_i|\boldsymbol{\mu}_j, \Sigma_j) > N(\mathbf{x}_i|\boldsymbol{\mu}_s, \Sigma_s)$, $j \neq s$.

Such optimization can be accomplish via an Expectation Maximization strategy (chapter 9 of Bishop's book).

The optimization works as follows:

The optimization works as follows:

1. (E-step) Fixing c_i, $\boldsymbol{\mu}_i$, Σ_i we can compute the probability of a Gaussian with parameters $\boldsymbol{\mu}_j, \Sigma_j$ generates to point $\mathbf{x}_i$ as:

$$\gamma_{ij} = \frac{c_j N(\boldsymbol{\mu}_j, \Sigma_j)}{\sum_{s=1}^{k} c_s N(\boldsymbol{\mu}_s, \Sigma_s)}$$

The optimization works as follows:

1. (E-step) Fixing c_i, $\boldsymbol{\mu}_i$, Σ_i we can compute the probability of a Gaussian with parameters $\boldsymbol{\mu}_j, \Sigma_j$ generates to point $\mathbf{x}_i$ as:

$$\gamma_{ij} = \frac{c_j N(\boldsymbol{\mu}_j, \Sigma_j)}{\sum_{s=1}^{k} c_s N(\boldsymbol{\mu}_s, \Sigma_s)}$$

2. (M-step) Fixing γ_{ij} the parameters can be obtained by setting to zero the derivative of the likelihood, resulting in:

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma_{ij} \mathbf{x}_i$$

$$\hat{\Sigma}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^\top$$

$$\hat{c}_j = \frac{\sum_{i=1}^{n} \gamma_{ij}}{n}$$

The optimization works as follows:

1. (E-step) Fixing c_i, $\boldsymbol{\mu}_i$, Σ_i we can compute the probability of a Gaussian with parameters $\boldsymbol{\mu}_j, \Sigma_j$ generates to point $\mathbf{x}_i$ as:

$$\gamma_{ij} = \frac{c_j N(\boldsymbol{\mu}_j, \Sigma_j)}{\sum_{s=1}^{k} c_s N(\boldsymbol{\mu}_s, \Sigma_s)}$$

2. (M-step) Fixing γ_{ij} the parameters can be obtained by setting to zero the derivative of the likelihood, resulting in:

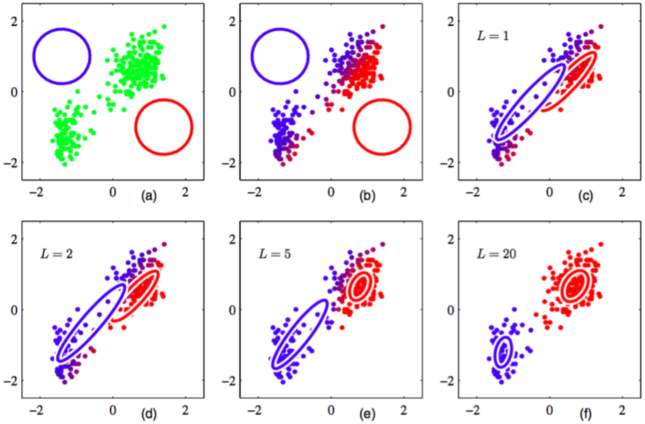
$$\hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma_{ij} \mathbf{x}_i$$

$$\hat{\Sigma}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^\top$$

$$\hat{c}_j = \frac{\sum_{i=1}^{n} \gamma_{ij}}{n}$$

Steps E and M are repeated until convergence.

Mixture Resolving



(figure extracted from Bishop's book)

Mixture Resolving

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

- variances $\Sigma_i = \epsilon \mathbf{I}$ for all Gaussians (spherically shaped Gaussians)

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

- variances $\Sigma_i = \epsilon \mathbf{I}$ for all Gaussians (spherically shaped Gaussians)
- $\gamma_{ij} \to r_{ij}$ when $\epsilon \to 0$.

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

- variances $\Sigma_i = \epsilon \mathbf{I}$ for all Gaussians (spherically shaped Gaussians)
- $\gamma_{ij} \to r_{ij}$ when $\epsilon \to 0$.

Therefore, K-means tends to generate spherically shaped clusters !!

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

- variances $\Sigma_i = \epsilon \mathbf{I}$ for all Gaussians (spherically shaped Gaussians)
- $\gamma_{ij} \rightarrow r_{ij}$ when $\epsilon \rightarrow 0$.

Therefore, K-means tends to generate spherically shaped clusters !!

The convergence of Mixture Resolving is slower than the convergence of K-means.

K-means is typically used to set initial conditions !!

DBSCAN

- DBSCAN is density-based clustering algorithm.

DBSCAN

- DBSCAN is density-based clustering algorithm.
- A cluster is primarily defined by points whose neighbourhood within a given radius contains at least a minimum number of points.

- DBSCAN is density-based clustering algorithm.
- A cluster is primarily defined by points whose neighbourhood within a given radius contains at least a minimum number of points.
- The algorithm demands two parameters
  - ϵ: the radius defining the neighbourhood area
  - *npt*: the minimum number of points within in the ϵ-neighbourhood.

- DBSCAN is density-based clustering algorithm.
- A cluster is primarily defined by points whose neighbourhood within a given radius contains at least a minimum number of points.
- The algorithm demands two parameters
  - ϵ: the radius defining the neighbourhood area
  - *npt*: the minimum number of points within in the ϵ-neighbourhood.
- The clustering process is based on the classification of the points as *core points*, *border points*, and *noise points*.

- **Core Point**: a point whose ϵ-neighbourhoo contains at least *npt* points.

- **Core Point**: a point whose ϵ-neighbourhoo contains at least *npt* points.
- **Border Point**: a point whose ϵ-neighbourhoo contains less than *npt* points but it belongs to the ϵ-neighborhood of some core point.

- **Core Point**: a point whose ϵ-neighbourhoo contains at least *npt* points.
- **Border Point**: a point whose ϵ-neighbourhoo contains less than *npt* points but it belongs to the ϵ-neighborhood of some core point.
- **Noisy Point**: a point that is neither a core nor a border point.

Algorithm

Algorithm

- Find the points in the ϵ-neighborhood of every point and identify the core points with more than *npt* neighbors.

Algorithm

- Find the points in the ϵ-neighborhood of every point and identify the core points with more than *npt* neighbors.
- Find the ϵ-nearest neighbor graph of core points ignoring all non-core points and label each connected component of the graph as being a cluster

Algorithm

- Find the points in the ϵ-neighborhood of every point and identify the core points with more than *npt* neighbors.
- Find the ϵ-nearest neighbor graph of core points ignoring all non-core points and label each connected component of the graph as being a cluster
- Assign each non-core point to a nearby cluster if the non-core point is in the ϵ-neighbor of a core point of the cluster, otherwise assign it to noise.

Properties

Properties

- dthe number of clusters is not specified

Properties

- dthe number of clusters is not specified
- can find arbitrarily shaped clusters
- has a notion of noise while being robust to outliers

Properties

- dthe number of clusters is not specified
- can find arbitrarily shaped clusters
- has a notion of noise while being robust to outliers
- not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed

Properties

- dthe number of clusters is not specified
- can find arbitrarily shaped clusters
- has a notion of noise while being robust to outliers
- not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed
- cannot cluster data sets well with large differences in densities

Properties

- dthe number of clusters is not specified
- can find arbitrarily shaped clusters
- has a notion of noise while being robust to outliers
- not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed
- cannot cluster data sets well with large differences in densities
- for $npt \leq 2$ the result tends to be the same as of hierarchical clustering with the single link metric

Properties

- dthe number of clusters is not specified
- can find arbitrarily shaped clusters
- has a notion of noise while being robust to outliers
- not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed
- cannot cluster data sets well with large differences in densities
- for $npt \leq 2$ the result tends to be the same as of hierarchical clustering with the single link metric
- there are several methods for estimating ϵ and npt automaticall (for instance, using histograms)