Otto von Guericke University of Magdeburg

Department of Computer Science
Institute for Knowledge and Language Engineering

Master Thesis

Hierarchical Extensions and Cluster Validation Techniques for DBSCAN

Author:
Alexander Dockhorn

September 24, 2015

Advisers:

Prof. Dr. habil.
Rudolf Kruse

Department of Computer Science
Otto von Guericke University
Universitätsplatz 2
39106 Magdeburg, Germany

Prof. Dr.-Ing.
Andreas Nürnberger

Department of Computer Science
Otto von Guericke University
Universitätsplatz 2
39106 Magdeburg, Germany

Dockhorn, Alexander:
Hierarchical Extensions and
Cluster Validation Techniques for DBSCAN
Master Thesis, Otto von Guericke University
Magdeburg, 2015.

Contents

Kurzfassung

DBSCAN ist einer der weitverbreitetsten dichte-basierter Clustering Algorithmen. Trotz seiner Effizienz in unterschiedlichen Clustering Szenarien stellt das Finden von geeigneten Parametern eine nicht-triviale Aufgabe dar. Eine Vielzahl an Parameterschätzungs- und Eliminationsverfahren wurden vorgeschlagen. Dennoch sind angeführte Algorithmen oft nicht intuitiv in ihrer Handhabung und ungeeignet für die Erkennung von Clustern unterschiedlicher Dichte. In dieser Arbeit wird ein Algorithmus basierend auf alternatierender Optimierung der Eingabeparameter zur Bestimmung einer lokal optimalen Parameterkombination für DBSCAN beschrieben. Dieser basiert auf einer Kombination zweier hierarchischer Erweiterungen von DBSCAN, welche durch das Festsetzen eines Parameters und Iterieren des zweiten freien Parameters erstellt werden. Durch die Monotonie des Parameterraums können aufeinanderfolgende Ebenen der Hierarchie effizient berechnet werden. Die erstellten Hierarchien werden weiter analysiert, um eine geeignete Abschätzung des freien Parameters zu wählen oder Cluster mit unterschiedlicher Dichte durch nicht horizontale Schnitte der Clusterhierarchien zu extrahieren. Geeignete horizontale Schnitte werden durch die Verwendung von internen Clustervalidierungsmaßen gefunden. In dieser Arbeit werden zur Verfügung stehende interne Validierungsmaße verglichen und eine dichte-basierte Version des Silhouetten Koeffizienten beschrieben. Unsere Ergebnisse zeigen, dass dieser gut geeignet ist, um von DBSCAN generierbare nicht-konvexe Cluster zu bewerten. Mit Hilfe des dichte-basierten Silhouetten Koeffizient gelang es durch alternierende Optimierung zuverlässig geeignete Parameterkombinationen für DBSCAN in einer Reihe von unterschiedlichen Clustering Szenarien zu finden. Zudem zeigten sich nicht horizontale Schnitte besonders nützlich zur Extraktion von Clustern unterschiedlicher Dichte.

Abstract

DBSCAN is one of the most commonly used density-based clustering algorithms. While it performs good in various clustering scenarios, finding appropriate parameters for the algorithm is a non-trivial task. Multiple works proposed parameter estimation or elimination techniques. However, most of the resulting algorithms suffer from usability issues and the incapability of finding clusters of different densities. In this work we propose an alternating optimization algorithm to find locally optimal parameter combinations for DBSCAN. It combines two hierarchical versions of DBSCAN, which are generated by fixing one parameter and iterating through the parameter space of the other. Due to the monotonicity of the parameter space, successive hierarchy levels can efficiently be computed. Hierarchies generated this way can be analyzed to find an appropriate estimate for the free parameter or finding clusters with different densities by the use of non-horizontal cuts. An internal validation criterion is used to find an appropriated horizontal. Throughout this work we compare multiple internal validation measures and propose a density-based interpretation of the silhouette coefficient. Our results show that the proposed density-based silhouette coefficient adapts well to non-convex clusters produced by DBSCAN. Also, the alternating optimization approach automatically detects a good parameter combination in a variety of clustering scenarios. Additionally, non-hierarchical cuts performed especially well in the detection of clusters with different densities.

The validation of clustering structures is
the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction,
cluster analysis will remain a black art accessible only to those
true believers who have experience and great courage.

Anil K. Jain and Richard C. Dubes

1 Introduction and Motivation

1.1 Motivation

The automatic extraction of information and knowledge from data is a well-studied task in the field of digital data analysis. Clustering is a common problem of this field and describes the task of extracting groups of similar objects in data sets. A large variety of algorithms for clustering is openly available and many real world applications make use of them. However, the user base is still restricted to experts, who can adapt the methods to a certain data setting. The non-trivial choice which algorithm to use and how to configure parameters can have a large influence on the clustering results. Finally, obtained results can often difficult to be validate or interpret by the user.

As the introductory quote by JAIN und DUBES (1988) warns, the knowledge gap between clustering experts and users is about to grow. Especially, further effort is needed in the field of cluster validation. As was already mentioned, methods for parameter estimation are needed as well to support non-expert users in the application of clustering algorithms. This thesis will specifically focus on the elimination of parameters and the validation of obtained results.

Easier applicable clustering algorithms and understandable feedback of the clustering structure can benefit users and experts. Additionally, overcoming the knowledge gap can help to establish clustering solutions in new business areas by providing confidence in the methods' results.

1.2 Aim of this thesis

A widely used clustering paradigm is the class of density-based methods. They divide the data set into sets of dense areas, which are separated by sparse areas. Throughout the years a variety of methods were developed to cover multiple applications and try to avoid problems of previous algorithms. However, each of the current methods has its limitations.

Methods like DBSCAN and DENCLUE can only provide a flat clustering. Whereas hierarchical methods like gSkeletonClu are not able to condense the hierarchy of clusterings and can hinder the user from detecting most significant structures. Still, multiple methods, even hierarchical methods like OPTICS, cannot extract clusters of differing density. Additionally, each of the named methods depends on one or more critical input parameters, which have significant influence in the results of the clustering algorithm. However, most algorithms lack of methods for parameter estimation and, therefore, make it difficult to adjust the algorithm to a specific dataset (CAMPELLO et al., 2013).

Referring to the No-free-Lunch-theorem (WOLPERT, 1995), it will not be possible to develop one single method, which is successful in every clustering scenario. However, this work will try to tackle listed limitations as best as possible and aims for the development of suitable extensions of the DBSCAN algorithm. Further, the topics user interaction and reporting interpretable results will be covered in this thesis to facilitate the user experience of proposed extensions.

1.3 Structure of this thesis

The thesis on hand will be structured as follows: Chapter 2 will review general clustering concepts and present common clustering algorithms. Special focus will lie on the class of density-based clustering algorithms as well as clustering validation techniques discussed at the end of this chapter.

Further on, enhancements of the DBSCAN algorithm will be covered in Chapter 3. Based on monotonicity characteristics of both DBSCAN parameters we will propose two hierarchical versions of the DBSCAN algorithm. These can be combined to an alternating optimization approach.

Thus, using an objective function for the optimization process can substitute the parameter choice. Furthermore, we will analyze characteristics of obtained clustering hierarchies. Finally, multiple visualizations will be presented to support the user in interpreting the results.

A general comparison of the new developed algorithm and its alternatives will be covered in Chapter 4. External cluster validation indices will be used for a comparison of the clustering accuracy in a variety of scenarios.

Final remarks on the proposed enhancements, algorithms and visualizations will be included in Chapter 5. Since the scope of this thesis is limited propositions for feature work on the topic of density-based hierarchical clustering algorithms will be summarized at the end of this work.

2

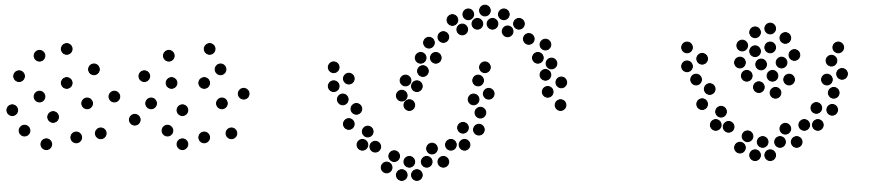
Introduction to Clustering Analysis

Cluster analysis describes the task of finding sets of similar objects in data. The discovered sets represent a compressed model of the original data. This can include prototypes per cluster, hierarchical relations about the data or found clusters and visualization of the dataset, e.g. using the clustering for dimensionality reduction (BERKHIN, 2010).

This chapter will first review common cluster characteristics followed by a description of the general clustering analysis process. Section 2.3 will summarize the desired outcome types of this process. We will further review common clustering algorithms in Section 2.4 to review common representatives and for later analysis of the results in comparison to proposed methods. The group of density-based clustering algorithms will be discussed in more detail in Section 2.5. Limitations and disadvantages of those methods will be summarized in Subsection 2.5.6. In Chapter 3 we will propose extensions for DBSCAN to cope with those limitations. Furthermore, we will review cluster validation techniques in Section 2.6. These will be used as basis for the evaluation chapter and internal rating of interim results of proposed methods.

2.1 Definition of a cluster

While typical notions of clustering algorithms include the categorization in hierarchical, partitioning and density based algorithms, this does only describe the algorithms' clustering strategy. Another way to describe clusters is by their inherent properties. Three fundamental categories were summarized by HANDL et al. (2005):



(a) Compactness (b) Connectedness (c) Spatial separation

Figure 2.1: Visualization of the three inherent properties of clusters.

Compactness: Objects in the same cluster should be similar to each other. Algorithms searching for compact clusters will prefer clusters of spherical shape, since the average variation would be the smallest compared to other shapes.

Connectedness: A local approach is to assign spatially close or similar objects to the same cluster. A more concrete description would be the following: a point is closer or more similar to one or more points in a cluster than to any point not in the cluster. In contrast to algorithms based on compactness, algorithms optimizing connectedness can adapt to various cluster shapes. However, they are easily influenced by noise and cannot separate two clusters if they are not spatially separated.

Spatial separation: While both other measures try to sort similar objects into the same cluster, spatial separation demands dissimilar objects to be assigned to different clusters as well as clusters being well separated from each other.

Figure 2.1 shows example structures for each cluster property. Figure 2.1a depicts two clusters of spherical shape, which are connected in the middle. The second example shows two spatially separated, but in itself connected sickles. Figure 2.1c shows both previous cluster shapes, which are spatially separated, such that the distance to at least one point in the same cluster is always less than the distance to any point of the other cluster.

This work will compare density-based methods, which focus on the extraction of connected and spatial separated clusters. Algorithms in section 2.3 will be introduced with their correspondent clustering objectives. Clustering results will be compared within the evaluation chapter.

2.2 Process model of cluster analysis

After clarifying what the process of cluster analysis tries to reveal in a dataset, we will take a closer look at the process itself. As HANDL et al. (2005) proposed, the process of clustering analysis can be divided into three phases as shown in Figure 2.2.

First of all the data hast to be preprocessed. This can contain adjusting attribute ranges, filtering noise, calculating distance or similarity of data instances along with others. Each step can have an influence on the final clustering result. The work of BONDER et al. (2012) showed that in their experiments the applied preprocessing had a higher influence on the final clustering result than the choice of the clustering algorithm. Another study suggests that special care has to be taken of the choice of the distance measure, since the decision which distance measure should be used can change the meaning and result of the obtained clustering (FINCH, 2005). Nonetheless, preprocessing methods will not be covered in this work. The interested reader will find further information on the topic in BERTHOLD et al. (2010).

Second phase of Handl's process model is the cluster analysis phase, in which the algorithm has to be chosen and applied. Methods differ in the clustering in- and output. The user needs to find an algorithm that fits the inherent properties of the dataset. Additional information like estimates of the datasets distribution can improve the clustering process and its outcome. However, choosing the correct input parameters is not a trivial task and covered in many research papers (XU und WUNSCH II, 2005). An introduction on clustering algorithms will be given in section 2.3.

The final phase of Handl's clustering process model covers the analysis of the clustering results. A drawback of most clustering algorithms is that they will always enforce a clustering even if the data does not support any such structure. However, multiple methods test the integrity of found clusters and try to support the user in the validation of given results. Further, the results can be used to iteratively improve preceding phases. For example, additional attribute scaling or filtering can be applied or input parameters of the second phase can be adjusted. Clustering validation will be covered in more detail in section 2.6.

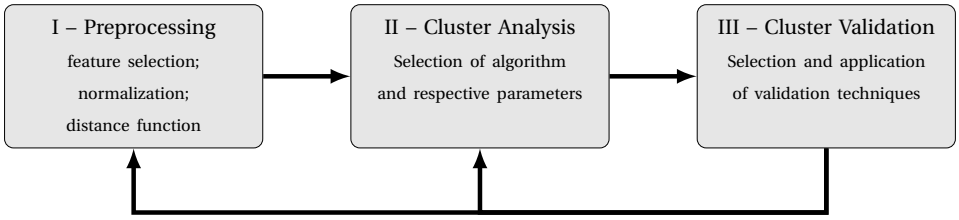


Figure 2.2: Three step process model of cluster analysis by HANDL et al. (2005).

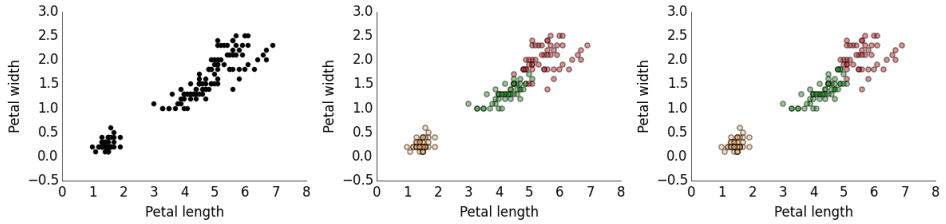
The more general data mining model *Cross Industry Standard Process for Data Mining* (CRISP-DM) (SHEARER et al., 2000) suggests further phases, namely project understanding, data understanding, and deployment. All of these phases focus on industrial applications and are not needed to evaluate the general characteristics of a clustering algorithm. The remaining phases data preparation, modeling and evaluation are covered by the three step process model by HANDL et al. (2005).

ROUSSEEUW (1987) proposed the extend the cluster validation by making the attempt to interpret the results in context of the dataset. Visualizations and general characteristics of the clustering algorithm can help the user to interpret clustering results. Since the ultimate goal is to obtain interpretable results in the sense of additional insights of the data, the resulting clustering can be compared with existing expert knowledge or used to validate previous constructed hypotheses. Chapter 5 of this thesis will summarize the clustering results and possible implications for the area of application of the proposed clustering techniques.

2.3 Types of clustering algorithms

While previous sections gave insights in the process of clustering analysis and the definition of a cluster, this section will cover the definition of the desired output of a clustering algorithm.

Given a set of input patterns $\mathbf{X} = \{x_1, \ldots, x_j, \ldots, x_N\}$, where each point $x_j = (x_{j1}, x_{j2}, \ldots, x_{jd}) \in \mathbb{R}^d$ is made of features x_{ji} (also referred to as attributes or dimensions) clustering algorithms have different specifications of the desired type of clustering. XU und WUNSCH II (2005) differentiated the following types of a clustering:



(a) Two dimensions of the Iris dataset

(b) true partition of the dataset

(c) clustered dataset using k-means

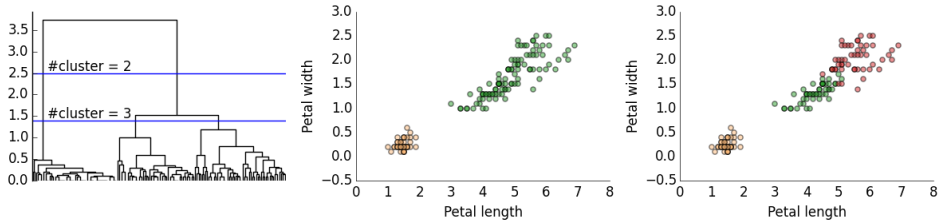
Figure 2.3: Flat clustering example of the Iris dataset using k-means.

Definition 1 ((hard) flat / partitional clustering) *A flat or partitional clustering is a partition of dataset* $\mathbf{X}$ *to* $\mathcal{C} = \{C_1, \cdots, C_K\}$ *with* $K \leq N$*, such that all sets* C_1 *to* C_K *(clusters) are non-empty, pairwise disjunct and their union is equal to the original dataset* $\mathbf{X}$*.*

Flat clusterings can be used to find and describe groups in datasets. A well known example in the field of clustering and classification analysis is the taxonomy of plants in the iris dataset (FISHER, 1936). It consists of the three types of the iris plant *Iris setosa, Iris versicolor* and *Iris virginica*, which can be partitioned by using lengths and widths of their petals and sepals. Figure 2.3 presents a partitional clustering of the iris dataset compared to the unclustered dataset and the true partition. Except for few mis-assignments in between the two clusters on the top right corner, the clustering matches the true partition.

Nevertheless, does a flat partition not always suffice to represent the group structure. Looking at the iris dataset two groups can be identified through spatial separation. Using additional information about the number of data points per group, we can assume that the top group has to be further divided. In such a scenario a hierarchy of clusterings can be much more appropriate and is defined by:

Definition 2 ((hard) hierarchical clustering) *A hierarchical clustering is a tree-like nested structured partition of* $\mathbf{X}$ *denoted by* $\mathcal{H} = \{\mathcal{C}_1, \ldots, \mathcal{C}_Q\}$ *with* $Q \leq N$*, where each hierarchy level represents a flat clustering. Clusterings are ordered such that* $C_i \in \mathcal{C}_m, C_j \in \mathcal{C}_l$*, and* $m > l$ *implies that either* $C_i \subseteq C_j$ *or* $C_i \cap C_j = \emptyset$*. The hierarchy* $\mathcal{H}$ *can be visualized using a tree, in which each*



(a) Average linkage dendrogram of the Iris dataset

(b) Average linkage producing 2 clusters

(c) Average linkage producing 3 clusters

Figure 2.4: Hierarchical clustering example of the Iris dataset using average linkage and multiple horizontal cuts of the dendrogram.

subset relation $C_i \subseteq C_j$ between consecutive hierarchy levels is represented as an edge from C_j to C_i.

In comparison to the flat clustering of Figure 2.3, a hierarchical representation of the Iris dataset can be determined using hierarchical agglomerative clustering with the average linkage criterion. The plots in Figure 2.4 show the full dendrogram of the Iris dataset and two possible clusterings with two and three clusters. It can be seen how the clusters in the top right corner are merged in consecutive levels.

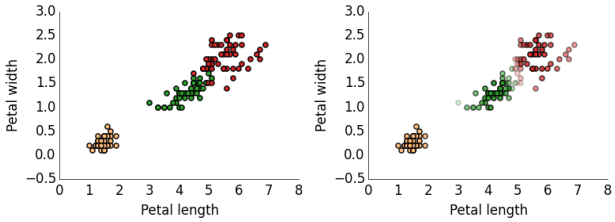
Both previous clustering definitions force all points to be assigned to one specific cluster (per hierarchy level). Soft or also called fuzzy clustering loosens this assumption and assigns a cluster membership degree to each pair of points and clusters, such that a point can be part of multiple clusters.

Definition 3 (soft / fuzzy clustering) *Fuzzy clustering assigns a degree of membership to each pair of points $x \in \mathbf{X}$ and $C \in \mathcal{C}$. A point x_i may belong to the cluster C_j with a degree of membership $u_{i,j} \in [0,1]$, such that:*

$$\sum_{j=1}^{|\mathcal{C}|} u_{i,j} = 1, \quad \forall i$$

The total clustering can be represented as membership matrix $U[|\mathbf{X}|, |\mathcal{C}|]$, in which each cell $U(i,j)$ corresponds to the membership degree $u_{i,j}$.

This definition allows us to have points to participate in multiple clusters. A flat clustering can be obtained by assigning each point to the cluster with



(a) True flat clustering of the Iris dataset

(b) Fuzzy clustering of the Iris dataset

Figure 2.5: Fuzzy clustering example of the Iris dataset using fuzzy-c-means (c=3). Transparency values were related to the membership degree.

its highest degree of membership. Figure 2.5 shows a fuzzy clustering of the Iris dataset.

In the following we will present common cluster algorithms. This work will focus on density-based methods, which will be explained in more detail in Section 2.5.

2.4 Exemplary clustering algorithms

2.4.1 K-Means

K-means is one of the standard partitioning algorithms (MACQUEEN, 1967). In an iterative approach the algorithm determines cluster centers such that the average distance of each point to its nearest cluster center is minimized. This is done by randomly initializing k cluster centroids. In every iteration each point will be assigned to the nearest cluster center. After this assignment phase all cluster centroids will be repositioned to the center of all its assigned points. This process will be repeated until the positions of the centroids converge.

The algorithm aims for minimizing the sum of squared errors (SSE) of all points to their assigned cluster centroids:

$$SSE(\mathcal{C}) = \sum_{k=1}^{K} \sum_{x_i \in C_k} |x_i - c_k|^2, \tag{2.1}$$

where K is the number of clusters and c_k the centroids of cluster C_k. The described optimization process is a greedy algorithm, which is known to

always converge and leading to a local optimum. Finding a global optimum for its score function is known to NP-Hard (KIM et al., 2012).

2.4.2 Fuzzy C-Means

An extension of the k-means algorithm is fuzzy c-means (BEZDEK et al., 1984). In contrary to the hard cluster assignment of the k-means algorithm, fuzzy c-means assigns a membership degree u_{ij} as it was defined in Definition 3.

The target function of fuzzy c-means can be stated as:

$$J(U,V) = \sum_{i=1}^{|\mathcal{C}|} \sum_{k=1}^{N} u_{i,k}^{m} d(c_i, x_k)^2 \tag{2.2}$$

Parameter m is known as fuzzifier. In case of $m = 1$ fuzzy c-means is equal to standard k-means. The target function needs to be minimized under the additional constraints for the membership degrees $u_{i,j}$. Optimizing the target function under those constraints leads to an update formula for the cluster centers c and the membership degrees.

$$c_i = \frac{\sum_{k=1}^{N} u_{i,k}^{m} x_k}{\sum_{k=1}^{N} u_{i,k}^{m}} \qquad u_{i,k} = \left(\sum_{j=1}^{|\mathcal{C}|} \left(\frac{d(c_i,x_k)}{d(c_j,x_k)} \right)^{\frac{2}{m-1}} \right)^{-1} \tag{2.3}$$

The clustering starts by randomly choosing positions for the initial cluster centers. In an iterative process the membership degrees and the cluster centers can be recomputed using the formulas in Equation 2.3. This recomputation can be repeated for a maximal number of iterations or until the overall change of membership degrees is smaller than a fixed threshold. Final membership degrees can be transformed to a flat clustering by assigning each point to the cluster of its highest cluster membership degree.

2.4.3 Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering describes a class of bottom-up algorithms. The algorithms start by assigning each point to an individual cluster and iteratively merging most similar clusters until every point belongs to the same cluster. This way a hierarchy of clusters is generated.

Table 2.1: Distance measures for hierarchical agglomerative clustering.

Linkage type	distance measure
single linkage	$d_{\text{single}}(C_i, C_j) = \min_{a \in C_i, b \in C_j}\{d(a,b)\}$
complete linkage	$d_{\text{complete}}(C_i, C_j) = \max_{a \in C_i, b \in C_j}\{d(a,b)\}$
average linkage	$d_{\text{average}}(C_i, C_j) = \frac{1}{\lvert C_i \rvert \lvert C_j \rvert} \sum_{a \in C_i, b \in C_j}\{d(a,b)\}$

Complex similarity measures were proposed to match certain cluster structures. Most commonly known are single, complete and average linkage. Single linkage measures the distance of two clusters C_i and C_j by using the minimal distance between any pair of points $a \in C_i, b \in C_j$. On the contrary complete linkage uses the maximal distance between any pair of those points. Furthermore, average linkage defines the distance of two clusters by the average distance of all pairwise distances between points of both clusters. Table 2.1 summarizes the distance measures used in all three clustering algorithms. Another type of distance criterion is wards minimum variance (WARD, 1963), where the distance of two clusters is calculated by the increase of SSE (Equation 2.1) this merge would result in.

While single linkage optimizes for connectedness, complete and average linkage both focus on compact clusters. The effects of each merging strategy will be discussed in more details throughout the evaluation chapter.

2.5 Density Based Clustering

Clustering algorithms often do implicit assumptions about the shape of clusters. Where the shape is known this can be a desired effect, since applying expert knowledge can increase the accuracy of the result. However, in cases where the underlying model is unknown, methods enforcing certain cluster shapes can produce undesired results.

Another notion of clusters can be made through the estimation of density. As presented in section 2.1 cluster properties as connectedness and spatial separation can be used to justify the division of the data space into sparse and dense regions, where the latter represents clusters.

In physics density is known as mass per unit volume. Specifically number density is an intensive quantity for the degree of concentration of countable objects in physical space. We will see that density based methods loosen this definition by removing the condition of space being three-dimensional, since number of dimensions most often exceeds three.

2.5.1 DBSCAN

A common representative of the class of density based clustering algorithms is DBSCAN, which was proposed in ESTER et al. (1996). This thesis will focus on enhancements of DBSCAN. Therefor, it will be described in more detail than other representatives of this group of algorithms.

The DBSCAN algorithms searches for dense regions in spatial datasets. A region is called dense if the region around a point with a radius of size ϵ contains at least m_{Pts} objects. We use the following mathematical definitions through the rest of this work:

Definition 4 (ϵ-neighborhood of a point) *The ϵ-neighborhood of a point consists of all points with a maximal distance of ϵ:*

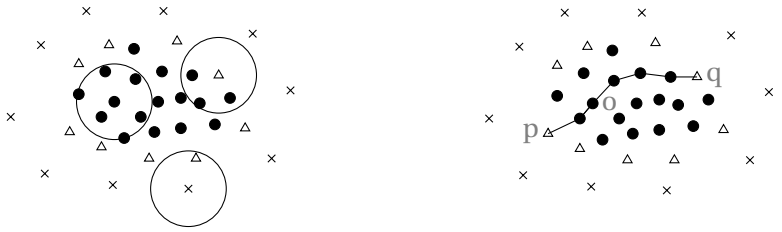
$$N_\epsilon(p) = \{ q \in \mathbf{X} \mid d(p,q) \leq \epsilon \} \tag{2.4}$$

Definition 5 (core points) *A point is denoted as core point if its ϵ-neighborhood consists of at least m_{Pts} objects (including itself). The set of all core points for a given pair of parameters will be referred to as $cores_{\epsilon,m_{Pts}}$ and be defined as:*

$$cores_{\epsilon,m_{Pts}} = |\{ p \in \mathbf{X} \mid m_{Pts} \leq |N_\epsilon(p)| \}| \tag{2.5}$$

With the set of cores and their respective ϵ-neighborhoods we already have a description of dense regions of the dataset. Further definitions are needed to combine dense regions to a set of clusters. The characteristic of directly density-reachability describes the range of a dense region, when density-reachability is further extended by allowing transitivity for a chain of core-points.

Definition 6 ((directly) density-reachable) *A point q is directly density-reachable from point p, if $q \in N_\epsilon(p)$ and p is a core-point. Note that the*



(a) Types of points and their encircled neighborhood sets: core points = •, border points = △, noise points = ×.

(b) Density-reachability and density-connectedness of core and border points. Points p and q are density-reachable by o and therefor density-connected to each other.

Figure 2.6: Visualization of a DBSCAN clustering using $\epsilon = 0.5$ and $m_{\text{Pts}} = 5$.

conditions $p \in N_\epsilon(q)$ and $q \in N_\epsilon(p)$ are equivalent. Furthermore, two points p, q are density-reachable if a chain of points $p_1, \ldots, p_n$ exists with $p_1 = p$ and $p_n = q$ such that for each $1 \leq i < n$, p_{i+1} is directly density-reachable from p_i.

A point which is density-reachable by a point p and is not a core point is called border point, since it is part of a dense region. Points not belonging to a dense region are called noise points.

Further on clusters are formed by strongly overlapping dense regions, which is characterized by the density-connected property:

Definition 7 (density-connected) *Two points p, q are density connected to each other if there exists a point o from which both points are density-reachable.*

Previous definitions are exemplified in Figure 2.6. First, Sub-Figure 2.6a shows representatives of each class of points in DBSCAN. While the number of points in the depicted ϵ-neighborhood of the core points exceeds the m_{Pts} threshold, the ϵ-neighborhood sets of the remaining points are not large enough. All points depicted as border points are density-reachabil from at least one core point. The second part of the graphic, Sub-Figure 2.6b, shows the reachability of the two border points p and q from the core point o. For this reason both border points are density-connected to each other. The remaining points which are marked as noise

points do neither fulfill the core condition nor do they lie in any of the core's ϵ-neighborhood sets.

2.5.2 DENCLUE

The DENCLUE algorithm by HINNEBURG und KEIM (1998) generalizes the density estimation of DBSCAN using influence functions. Let the influence of a point be defined by a function $f_B : \mathbf{X} \rightarrow \mathbb{R}$. Two examples for influence functions are the square wave and the gaussian influence function:

$$f_{square}(x,y) = \begin{cases} 1 & d(x,y) \leq \sigma \\ 0 & otherwise \end{cases} \qquad f_{gauss}(x,y) = \exp^{-\frac{d(x,y)^2}{2\sigma^2}}$$

The total density of a point can be calculated by the sum of influence functions to all objects of the dataset. Let the total density of a point be defined by its density function:

$$f_B^D(x) = \sum_{i=1}^{N} f_B(x, y_i)$$

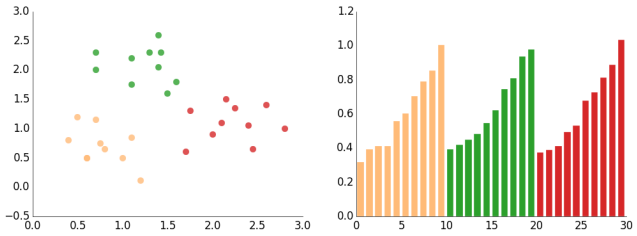
Local density maximums can be found by using a gradient function and applying a local search. Equal to DBSCAN, DENCLUE defines core points which exceed a density threshold ξ. Additionally the points representing a local density maximum are used to define a cluster center.

Using the square wave function and setting the radius $\sigma = \epsilon$ and the density threshold $\xi = m_{\text{Pts}}$ produces equal results compared to DBSCAN. While the problem of fixing two parameters remains the additional choice of a density function is added in comparison to DBSCAN.

2.5.3 OPTICS

Both DBSCAN and DENCLUE produce flat clusterings based on one parameter combination defining the density threshold. OPTICS (ANKERST et al., 1999) enhances the ideas of DBSCAN by determining an cluster order, which describes nested structures of higher density in clusters of a specified maximum ϵ value and a constant value of m_{Pts}.

This can be achieved by ordering the points in a cluster corresponding to the minimal distance they would be density-reachabil by a core point.



(a) Sample dataset (b) Reachability plot

Figure 2.7: Example dataset of convex clusters and respective reachability plot (based on an illustration of ANKERST et al. (1999)).

We will later refer to this concept by reachability distance and will review it throughout Subsection 3.2.1. Additionally the user is supported by a new visualization technique called reachability-plot, which describes the nested structure. One example is presented in Figure 2.7. Valleys in the plot correspond to clusters in the dataset shown on the left.

A drawback of the OPTICS method is that their is no simplification of the hierarchy of nested structures in the reachability-plot. No specification exists for determining a flat clustering or reduce the hierarchy to a manageable amount of levels.

2.5.4 CLIQUE

Another clustering technique called CLIQUE (AGRAWAL et al., 1998) is based on subspace density. Clustering in high-dimensional datasets can often be subdivided to multiple lower-dimensional subspaces to compute the clustering.

CLIQUE utilizes this idea by checking for low-dimensional dense units in a grid-based structure and expanding them by adding additional dimensions. The density cannot increase by adding dimensions, since the grid structure needs to be refined further subdivided by this dimension. If a unit of k dimensions is dense, then all of its projections in subsets containing $k-1$ dimensions are dense as well. For this reason the complexity of CLIQUE is exponential in the highest dimensionality of any dense unit, which discards it for the use in high dimensional datasets.

2.5.5 HDBSCAN*

HDBSCAN* is a hierarchical version of DBSCAN* based on the iteration of ϵ for a fixed value of m_{Pts}. It was initially proposed by CAMPELLO et al. (2013). DBSCAN* ignores the existence of border points and classifies points as either core or noise point. Clusters are composed of density-connected core points. An efficient calculation is supported by the use of a reachability graph and the minimum spanning tree algorithm.

We will aim for a similar approach in Section 3.2 but include border points in the calculation. Additionally, we will show that same principle can be applied for iterating values of m_{Pts}. The detailed algorithm and differences to the version proposed by CAMPELLO et al. (2013) will be discussed later throughout Chapter 3.

2.5.6 Limitations and drawbacks of current density-based algorithms

Methods like DBSCAN and DENCLUE can only provide a flat clustering. The hierarchical algorithm HDBSCAN* copes with this problem by implementing a hierarchical version of DBSCAN* but ignores the presence of border points. While the number of border points contained in multiple clusters is neglectable small in low dimensional datasets it grows with increased dimensionality and value of m_{Pts}. Therefore, generated hierarchy levels miss a lot of possible DBSCAN clusterings.

Another known disadvantage of previously listed density-based methods is the dependency to critical input parameters. DBSCAN and DENCLUE need an estimate for the ϵ and the m_{Pts} parameter. The first represents an estimate for the spatial separation of the dataset. This can be problematic in the case of high dimensional datasets, since the difference between maximal and minimal distance is known to vanish:

$$\lim_{dim \to \text{inf}} \frac{d_{\text{max}} - d_{\text{min}}}{d_{\text{min}}} \to 0$$

This effect is also called curse of dimensionality. The OPTICS algorithm as well as the HDBSCAN* algorithm avoid the estimation of ϵ and are only dependent on the m_{Pts} parameter. While the influence of this parameter can easily be visualized in low dimensional datasets, its estimation becomes more difficult with growing number of dimensions.

Additionally, neither DBSCAN, DENCLUE nor OPTICS are able to extract clusters of differing density. HDBSCAN* reduces the hierarchy by rating hierarchy levels by the amount of points being stable in comparison to the levels before and after. This approach performed well in the evaluation of CAMPELLO et al. (2013). However, the hierarchy generation is limited to an appropriate choice of m_{Pts}. Adjusting m_{Pts} can greatly change the structure of the hierarchy and therefore obtained results.

In Chapter 3 we propose extensions of DBSCAN to cope with mentioned problems of density-based methods.

2.6 Cluster validation

A general problem of applying cluster analysis to a dataset is validating the outcome. Independent from the analyzed dataset, algorithms will always return a clustering result, even if no intrinsic structure is present.

Validation techniques can be used to rate the outcome of the clustering algorithm and for example compare it to a known correct labeling. AGGARWAL und REDDY (2013) sorted available evaluation measures into the two categories external and internal measures. Additionally, visual evaluation measures were emphasized by HALKIDI et al. (2002). External and internal validation measures will be reviewed in the following subsections. For both categories we will present a selection of common measures, which will later be used for the evaluation in Chapter 4. Visual evaluation measures will be picked up again in Section 3.5, where we will discuss adaptations for several visualization techniques.

2.6.1 External evaluation measures

External validation measures compare the outcome of the clustering algorithm to external information, which was not used during the clustering process. Clustering is an unsupervised task, which means that true labels per point are not available throughout the analysis. However, if the dataset contains a true partition we can use it to validate obtained results.

Common approaches use a contingency matrix to compare the label assigned by the clustering algorithm with the true label. Let a contingency matrix be defined by:

Table 2.2: General contingency matrix

	C_1	C_2	$\cdots$	$C_{K'}$	$\sum$
P_1	n_{11}	n_{12}	$\cdots$	$n_{1K'}$	$n_{1\cdot}$
P_2	n_{21}	n_{22}	$\cdots$	$n_{2K'}$	$n_{2\cdot}$
$\cdot$	$\cdot$	$\cdot$	$\cdots$	$\cdot$	$\cdot$
P_K	n_{K1}	n_{K2}	$\cdots$	$n_{KK'}$	$n_{K\cdot}$
$\sum$	$n_{\cdot 1}$	$n_{\cdot 2}$	$\cdots$	$n_{\cdot K'}$	n

Definition 8 (Contingency matrix) *Given a dataset* **X** *containing n objects, with a desired partition* $\mathbf{P} = \{P_1, P_2, \ldots, P_K\}$ *and the clustered partition* $\mathcal{C} = \{C_1, C_2, \ldots, C_{K'}\}$. *The contingency matrix counts the number of occurrences* n_{ij}*, where a point was labeled as cluster* C_j *but lies in the true partition* P_i.

Table 2.2 represents a general contingency matrix of the clustering $\mathcal{C}$ and the true partition **P**. For the calculation of external validation measures we will make use of the following probabilities described by the occurrences in the contingency matrix:

$$p_{ij} = \frac{n_{ij}}{n}; \qquad p_i = \frac{n_{\cdot i}}{n}; \qquad p_j = \frac{n_{j\cdot}}{n} \tag{2.6}$$

Entropy (E)

Entropy is a concept of information theory, where it is used to describe the information contained in a message received. A clustering $\mathcal{C}$ can be seen as information source about the true structure **P** of the dataset and can be tested as predictor for exactly the same. In this sense the formula for the entropy of a clustering can be formulated as:

$$E(\mathcal{C}, \mathbf{P}) = -\sum_i p_i \left(\sum_j \frac{p_{ij}}{p_i} \log \frac{p_{ij}}{p_i} \right) \tag{2.7}$$

Entropy has a range of $[0, \log K']$. Values near 0 describe a approximately perfect clustering, where each cluster C_i is congruent with a partition P_j. The maximal value of $\log K'$ states that any point in a cluster C_i is equiprobable to be in any partition P_j.

Purity (P)

A closely related concept to entropy is purity, which is described by the extent in which a cluster contains objects of a single class (TAN et al., 2005).

$$P(\mathcal{C},\mathbf{P}) = \sum_i p_i \left(\max_j \frac{p_{ij}}{p_i} \right) \tag{2.8}$$

The range for the purity measure is $(0,1]$. An optimal clustering assigns all nodes of a cluster to the same partition and therefor achieves a value of 1.

F-Measure (F)

The f-measure is based on the harmonic mean of the two concepts precision and recall from the information retrieval community. The combination of both measures the extent in which clusters contain only objects of one true partition (precision) and the extent in which a cluster contains all elements of this partition (recall). The measure can be described by the following formula:

$$F(\mathcal{C},\mathbf{P}) = \sum_j p_j \max_i \left(\frac{2\frac{p_{ij}}{p_i}\frac{p_{ij}}{p_j}}{\frac{p_{ij}}{p_i}+\frac{p_{ij}}{p_j}} \right) \tag{2.9}$$

As it is the case for purity, the f-measure shares the range of $(0,1]$, where an optimal clustering achieves a value of 1.

Mutual information (MI)

Mutual information measures the degree of in which two random variables are mutual dependent on each other (BELLAMY et al., 1991). It is based on a comparison of the variables joint distribution and the products of their marginal distributions. Mutual information can be calculated by:

$$MI(\mathcal{C},\mathbf{P}) = \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{p_i p_j} \right) \tag{2.10}$$

The MI-score ranges from $(0, logK']$. Testing two independent random variables will result in values near 0, whereas dependent variable pairs result in higher values.

Homogeneity (Hom)

Homogeneity shares the general concept with purity and measures the extent in which every cluster only contains elements of one single class. The homogeneity of a clustering $\mathcal{C}$ and the true partition $\mathbf{P}$ can be calculated by applying the following formula:

$$Hom(\mathcal{C},\mathbf{P}) = \begin{cases} \frac{MI(\mathcal{C},\mathbf{P})}{E(P,P)} & ,E(P,P) > 0 \\ 1.0 & ,else \end{cases} \tag{2.11}$$

It results in score in a range of $[0,1]$, whereat a value of 1 represents are perfect homogeneous clustering.

Completeness (Compl)

In contrast to homogeneity, completeness tests for all data points of one class being part of the same cluster. The completeness score of a clustering $\mathcal{C}$ and a true partition $\mathbf{P}$ is given by the formula:

$$Compl(\mathcal{C},\mathbf{P}) = \begin{cases} \frac{MI}{MI(\mathcal{C},\mathbf{P})(\mathcal{C},\mathcal{C})} & ,E(\mathcal{C},\mathcal{C}) > 0 \\ 1.0 & ,else \end{cases} \tag{2.12}$$

The range of this score is [0,1], whereat values near 1 describe a perfect clustering in the sense of completeness.

V-measures (V)

The v-measure is based on the harmonic mean of homogeneity and completeness. This combination follows a similar concept as the f-measure, but showed superior results in experiments by ROSENBERG und HIRSCHBERG (2007). The v-measure can be calculated by:

$$V(\mathcal{C},\mathbf{P}) = \frac{2 \cdot Hom(\mathcal{C},\mathbf{P}) \cdot Compl(\mathcal{C},\mathbf{P})}{Hom(\mathcal{C},\mathbf{P}) + Compl(\mathcal{C},\mathbf{P})} \tag{2.13}$$

2.6.2 Internal evaluation measures

A drawback of external measures is that true labels are most often to complicated to receive or simply not available. Internal measures try to rate

the outcome of the clustering by comparing the structure of the dataset to the clustering.

Some clustering algorithms, as k-means, already use internal validation measures like the SSE as objective functions to find an optimal clustering. In the following we will review multiple internal validation measures, which will later be discussed for the use as an optimization criterion for proposed algorithms. Validation measures were chosen based on a summary in AGGARWAL und REDDY (2013).

Silhouette coefficient (S)

The silhouette coefficient ROUSSEEUW (1987) compares the tightness of a cluster in comparison to its separation to other clusters.

Let the silhouette of point i be defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2.14}$$

where $a(i)$ is the average distance to points in the same cluster as point i and $b(i)$ the minimum distance to points of other clusters. The silhouette of a point ranges from $(-1, 1)$. Negative values indicate that points of other clusters are nearer than points of the cluster the point was assigned to. Points in between two clusters will result in a silhouette near 0, whereas points of well separated clusters will score values near 1.

The silhouette coefficient is defined as the arithmetic mean of all silhouettes of points in a cluster C_i.

$$s_{C_i} = \frac{1}{n_{C_i}} \sum_{o \in C_i} s(o) \tag{2.15}$$

Accordingly the silhouette score is the mean of the silhouette coefficients of each cluster.

$$s_{\mathcal{C}} = \frac{1}{n_{\mathcal{C}}} \sum_{C_i \in \mathcal{C}} s_{C_i} \tag{2.16}$$

Since the silhouette coefficient is the mean of silhouettes, the range of the silhouette coefficient is $(-1, 1)$.

Dunn's index (D)

The internal validation criterion proposed by DUNN (1974) is a compactness oriented criterion, which measures the intercluster distance by the minimum pairwise distance between points of different clusters and sets it in relation to the intracluster distance determined by maximum diameter along all clusters.

$$D_{\mathcal{C}} = \min_{ij} \frac{\min_{x \in C_i, y \in C_j} d(x,y)}{\max_k \{\max_{a,b \in C_k} d(a,b)\}} \tag{2.17}$$

Dunn's index is a non-negative score, which needs to be maximized for optimal results.

The index as well as the I index (MAULIK und BANDYOPADHYAY, 2002) and the Calinski-Harabasz index (CALINSKI und HARABASZ, 1974) are indices based on a weighted comparison of cluster compactness and cluster separation. For this reason only Dunn's index will be used for a comparison with the other validation indices.

Clustering Validation Index based on Nearest Neighbor (*CVNN*)

Other than the previous measures the clustering validation index based on nearest neighbors evaluates intercluster separation based on the neighborhood of each point (LIU et al., 2013). Points with neighborhoods that were mainly assigned to the same cluster do only little influence the assessment of intercluster separation. Whereas points, whose neighborhoods include points assigned to other clusters have a higher weighted influence on the estimation of intercluster separation.

The *CVNN* score of a clustering $\mathcal{C}$ from a set of alternative clusterings $\Gamma = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ and a neighborhood set size of k is determined by the following formulas:

$$CVNN_{\mathcal{C}}(\Gamma, k) = \frac{Sep_{\mathcal{C}}(k)}{\max_{\mathcal{C}' \in \Gamma}\left(Sep_{\mathcal{C}'}(k)\right)} + \frac{Com_{\mathcal{C}}(k)}{\max_{\mathcal{C}' \in \Gamma}\left(Com_{\mathcal{C}'}(k)\right)} \tag{2.18}$$

$$Com_{\mathcal{C}}(k) = \sum_{C_i \in \mathcal{C}} \left(\frac{1}{n_i(n_i-1)} \sum_{x,y \in C_i} d(x,y) \right) \quad \text{with } n_i = |C_i| \tag{2.19}$$

$$Sep_{\mathcal{C}}(k) = \max_{C_i \in \mathcal{C}} \left(\frac{1}{n_i} \sum_{j=1,2,\ldots,n_i} \frac{q_i}{k} \right) \tag{2.20}$$

where q_i is equal to the number of nearest neighbors belonging to another cluster than the j-th point of C_i.

CVNN compares the current clustering with a range of other possible clusterings. Results for separation (Sep) and compactness (Com) need to be normalized through the range of clusterings to compare to. For optimal results the score needs to be minimized.

Edge Correlation (ρ)

Another way to measure a clusterings goodness of fit to a dataset is achieved by calculating a correlation coefficient between a cluster matrix and the similarity matrix. As the cluster attributes connectedness and spatial separation propose, data points assigned to the same cluster should be more similar than points assigned to different clusters.

For the objective function we adapted the method edge correlation, which was originally considered for validating graph clustering (BANSAL et al., 2004) and used by TAN et al. (2005) for the evaluation of spatial data clustering . Therefor we have to calculate the Pearson-correlation ρ of a $n \times n$ cluster matrix $L_{\mathcal{C}}$ and a similarity matrix *sim*.

Cluster matrix $L_{\mathcal{C}}$ is defined by:

$$L_{\mathcal{C}}(i,j) = \begin{cases} 1 & \text{, if } i \text{ and } j \text{ are in the same cluster} \\ 0 & \text{, else} \end{cases}$$

such that each entry (i,j) of the matrix $L_{\mathcal{C}}$ is 1 if i and j are in the same cluster referring to $\mathcal{C}$ or 0 if not. For better visualization points can be reordered by their clustering assignment. This results in a block diagonal matrix:

$$\begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & A_m \end{bmatrix}$$

where A_k is a square matrix corresponding to cluster k.

The similarity matrix can be inferred from the distance matrix, which was already calculated for the clustering process. We used $1 - d(i,j)$ as a similarity measure. In case cells where reordered during the calculation of L, the same transformation has to be applied to *sim*.

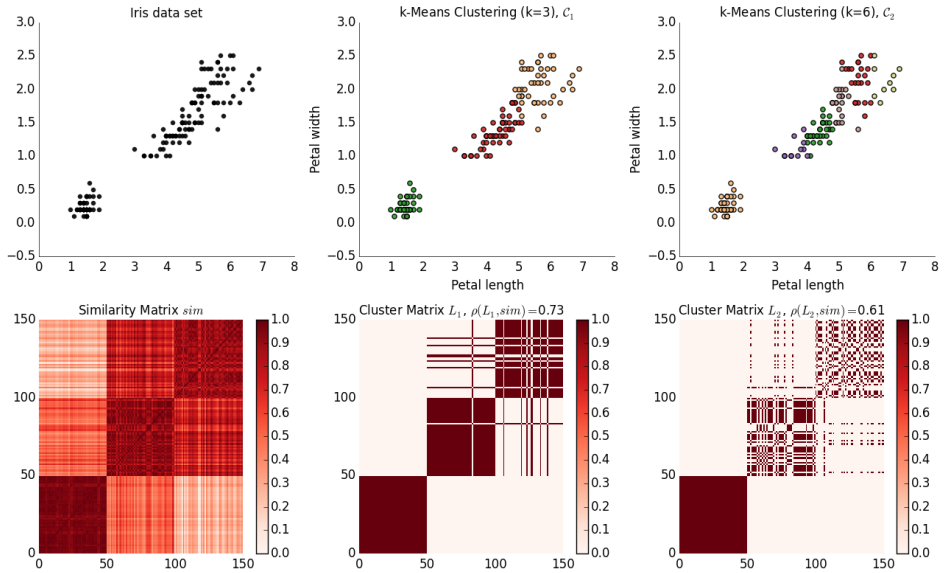


Figure 2.8: Visualization of clusterings and corresponding edge correlation scores of the Iris dataset. top row: Visualization of the Iris dataset and two clusterings calculated using the k-means algorithm. bottom row: similarity and cluster matrices with their corresponding edge correlation values.

The edge correlation $\rho_{\mathcal{C}}$ of a clustering $\mathcal{C}$ can be calculated as seen in:

$$\rho_{\mathcal{C}} = \rho(L_{\mathcal{C}}, sim) \tag{2.21}$$

The Pearson-correlation coefficient ranges from $[-1,1]$. Values near 1 indicate a desirable high correlation of the cluster matrix and the similarity matrix.

A heatmap can be used for visual comparison of the matrices $L_{\mathcal{C}}$ and *sim*. Figure 2.8 shows the similarity matrix of the data of the Iris dataset, with two clusterings and their corresponding heat similarity and cluster matrices. Rating both clusterings using the edge correlation cluster matrix L_1 is better than L_2, since it bas a higher correlation with the similarity matrix ($\rho(L_1, sim) = 0.73 > 0.61 = \rho(L_2, sim)$).

3

Hierarchical Extensions of DBSCAN

This chapter will propose extensions of DBSCAN, which try to cope with known disadvantages of density based clustering algorithms, such as interpretability of parameters, compression of a clustering hierarchy, and handling of differing density levels per cluster.

First, we will discuss the concept of monotonicity in context of the standard DBSCAN algorithm in Section 3.1. In the subsequent sections we will propose two hierarchical density based clustering algorithms, namely m_{Pts}-HDBSCAN and ϵ-HDBSCAN, which are dependent on only one parameter of DBSCAN. Multiple ways to handle produced hierarchies are discussed in Section 3.3. Additionally, both algorithms were combined in an alternating optimization approach, which will be introduced in Section 3.4. The chapter is closed with proposals for appropriate visualizations of discussed algorithms.

Parts of this chapter's content were already discussed in the recent paper of DOCKHORN et al. (2015). It should be noted that these parts were developed during the work on this thesis.

3.1 Monotonicity

A well known property in function theory is monotonicity. Frequently it can be exploited to reuse results for consecutive time steps. A typical example is the a-priori property in the field of frequent item set mining. The support of a set, the number of times a set occurs in different transactions of the database, cannot increase if further elements are added to the set. A logical implication for the search of frequent item sets is to stop the search for supersets if the current set does not fulfill the minimum support con-

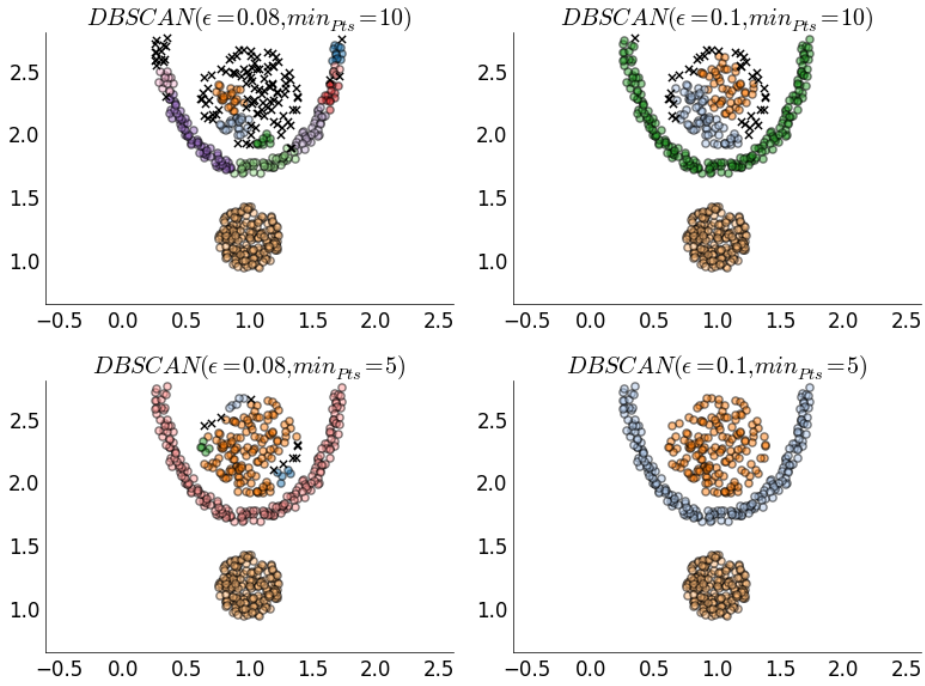


Figure 3.1: Comparison of DBSCAN results for various parameter settings. Noise points are marked with ×. row-wise: monotonic behavior related to ϵ, column-wise: monotonic behavior related to m_{Pts}.

dition, since further extending them cannot increase their support. This observation was made by AGRAWAL und SRIKANT (1994) and used in their corresponding a-priori algorithm.

The frequent item set condition can be viewed as monotonic criterion. Decreasing the minimum support value can only add further elements to the set of frequent item sets. This is because every element fulfilling the support condition of the original minimum support value will also fulfill the new minimum support value. For that reason consecutive runs of the a-priori algorithm with decreasing support value only need to extend the resulting frequent item sets of preceding runs.

Similar monotonicity observations can be made regarding the DBSCAN algorithm. Adjusting the parameters ϵ and m_{Pts} monotonically influences the reported cluster sizes (see Figure 3.1). This observation suggest to use a similar approach as it was used in the a-priori algorithm to create consecutive clusterings of DBSCAN with varying parameters.

The following subsections will prove monotonicity features in context of the DBSCAN algorithm. We will later exploit these to speed up the calculation of DBSCAN clusterings with varying parameter combinations.

3.1.1 Monotonicity of the neighborhood set

DBSCAN includes the two parameters ϵ and m_{Pts}, which can be adjusted for the search of differing clusterings. We will first observe how the change of ϵ influences the clustering result.

For two radii $\epsilon_1 > \epsilon_2$ we can show that the neighborhood-set of each point can only increase, since:

$$\begin{aligned} |\{q \in \mathbf{X} | \, dist(p,q) \leq \epsilon_1\}| &\geq |\{q \in \mathbf{X} | \, dist(p,q) \leq \epsilon_2\}| \\ |N_{\epsilon_1}(p)| &\geq |N_{\epsilon_2}(p)| \end{aligned} \tag{3.1}$$

The possible increase of the ϵ-neighborhood also influences the number of cores. For a fixed value of m_{Pts} we can infer:

$$\begin{aligned} |\{p \in \mathbf{X} | \, m_{\text{Pts}} \leq |N_{\epsilon_1}(p)|\}| &\geq |\{p \in \mathbf{X} | \, m_{\text{Pts}} \leq |N_{\epsilon_2}(p)|\}| \\ |cores_{\epsilon_1, m_{\text{Pts}}}| &\geq |cores_{\epsilon_2, m_{\text{Pts}}}| \end{aligned} \tag{3.2}$$

As shown in the previous formula, the core sets cannot decrease if the size of the ϵ-neighborhood is increased. This can lead to multiple changes of the cluster structure. For example when the neighborhood set of a point increases it could become a core and either form a new cluster or connect to one or more existing clusters. Additionally, new points can become density-reachable in context of a core-point and therefore be added to the cluster of the core-point.

3.1.2 Monotonicity of the core-condition

Instead of changing the size of neighborhood sets we can also directly influence the core-condition and their resulting cores by adjusting the value of m_{Pts}.

Similar to the approach used before, we observe the influence of a change of m_{Pts} on the two conditions. By definition the neighborhood set is uninfluenced by a change of m_{Pts}. However, it has a direct influence on

the core-condition. For two values $m_{\text{Pts}1} < m_{\text{Pts}2}$ the following inequality holds:

$$\begin{aligned} |\{p \in \mathbf{X} |\, m_{\text{Pts}1} \leq |N_\epsilon(p)|\}| &\geq |\{p \in \mathbf{X} |\, m_{\text{Pts}2} \leq |N_\epsilon(p)|\}| \\ |cores_{\epsilon, m_{\text{Pts}1}}| &\geq |cores_{\epsilon, m_{\text{Pts}2}}| \end{aligned} \tag{3.3}$$

As the previous inequality states, new cores can arise by lowering the minimum amount of points in the neighborhood set. This can lead to an increase in the number of clusters, adding cores to an existing cluster or merging two clusters in the case that a shared border point becomes a core point. Because the neighborhood set is uninfluenced by the parameter change, existing clusters cannot increase in size without an increase of cores in this cluster.

Figure 3.1 shows multiple clusterings of the same dataset with varying parameter combinations. An adjustment of the m_{Pts} value is depicted column-wise. An decrease of the m_{Pts} value leads to bigger clusters and the development of new clusters. It has to be noted that any further decrease would lead to the sphere on the top being connected to the surrounding sickle.

3.2 Hierarchical DBSCAN — HDBSCAN

Hierarchical algorithms can be divided into agglomerative and divisive clustering algorithms. While the first type starts with a set of unclustered objects and merges most similar pairs till only one group is remaining, divisive hierarchical clustering starts by assigning all elements to one cluster and splits this iteratively.

In case of $m_{\text{Pts}} = 1$, DBSCAN acts similar to single linkage. This is due to every point fulfilling the core-condition and being linked to points in its ϵ-neighborhood. In this particular case the parameter ϵ is equal to the cut height of an horizontal dendrogram cut. The following chapters will show, how a similar principle can be adopted for other parameter combinations.

First, we will introduce the algorithm m_{Pts}-HDBSCAN in Subsection 3.2.1, which iterates through all possible values of ϵ to create a dendrogram for a fixed value of m_{Pts}. Subsection 3.2.2 will propose a similar approach for a fixed value of ϵ.

3.2.1 HDBSCAN for a fixed $m_{\mathbf{Pts}}$ value — $m_{\mathbf{Pts}}$-HDBSCAN

The first algorithm we want to propose is m_{Pts}-HDBSCAN. It follows the idea of iterating through all possible values for ϵ, while using a fixed value of m_{Pts}. We will present two implementations of this algorithm. The first one can directly be inferred by the recent description and the second one follows a more sophisticated approach by changing the data-structure to a graph containing information about pairwise reachability-distances.

DBSCAN does not describe a general estimate for the ϵ value. An iterative method could fix an interval of ϵ values $[\epsilon_{min}, \epsilon_{max}]$ and calculate the DBSCAN clustering for a number of values contained in this range. A disadvantage of this approach is the number of possible values for ϵ. Since the spatial data is processed using a distance matrix, the number range for valid cut-heights in the dendrogram is equal to the number range of data points and their according distance metric. In general this is not limited to a countable set, which would need us to iterate through an infinite number of ϵ values.

We can reduce the amount of possible ϵ values, if we focus on all radii, which possibly change the clustering. Reasonable ϵ values can be directly inferred from entries in the distance matrix. This limits the number range to all values present in the matrix, which is N^2 for a dataset of size N.

Relating to the monotonicity of the neighborhood set, we first define the minimal distance in which the size of the neighborhood set is greater or equal to the specified minimum number of points. We will further refer to this using the term core distance of a point and define it by:

Definition 9 (core distance) *Let the core distance $d_{core,m_{Pts}}(x_i)$ of a point $x_i \in \mathbf{X}$ be the distance to its m_{Pts}-nearest neighbor (*ANKERST *et al., 1999).*

At each core distance it is possible that either existing clusters are merged, clusters grow in size, or emerge from new core points. However, further hierarchy levels in between two core distances d_i, d_j might possibly exist, since new points can get in range of an existing core with distance d_k, whereat $d_i < d_k < d_j$.

From this observation it can be inferred that all pairwise distances of two points have to be processed to ensure that each different DBSCAN hierar-

Algorithm 1 m_{Pts}-HDBSCAN (Clustering-tree)

Input: m_{Pts}, *Dist* = pairwise distance matrix of **X**

$\mathcal{C} \leftarrow \{\}$
clust_hierarchy $\leftarrow$ *initialize_clustertree*($\mathcal{C}$)
dist_list $\leftarrow$ sort *Dist* in priority list as (r, c, d)
(row-index, column-index, distance)

for all (r, c, d) in dist_list **do**
  add c to neighborhood of r
  if $r \notin$ cores **and** $m_{\text{Pts}} \leq |N(r)|$ **then**
    add r to cores
  end if
  update_density_reachability(r)
  $\mathcal{C}_{dist} \leftarrow$ *update_clustering*($\mathcal{C}$)
  if $\mathcal{C}_{dist} \neq \mathcal{C}$ **then**
    clust_hierarchy.*add_clustering*($\mathcal{C}_{dist}$)
    $\mathcal{C} \leftarrow \mathcal{C}_{dist}$
  end if
end for

return clust_hierarchy

chy level is included. The process can be stopped as soon as every point is in the same cluster. The m_{Pts}-HDBSCAN algorithm is summarized in Algorithm 1.

Further improvements of the proposed algorithm can result from using other data structures to process the distance matrix and store the hierarchy of DBSCAN clusterings. As the work of GOWER und ROSS (1969) proposed, single linkage can be implemented using a minimum spanning tree. However, DBSCAN allows to choose a arbitrary value for m_{Pts}, which directly influences the density-reachability of two points. For this reason distances in the distance matrix have to be adjusted, since density-reachability requires at least one of the points to be a core point. Following distance transformation has to be applied to all pairwise distances and is later referred to as reachability-distance:

Definition 10 (reachability distance) *Let the reachability distance of two points $x_i, x_j \in X$ be the distance, at which either x_i is density reachable by x_j or the other way around.*

$$d_{reach,m_{Pts}}(x_i, x_j) = \max\left\{\min\{d_{core,\, m_{Pts}}(x_i),\, d_{core,\, m_{Pts}}(x_j)\},\, d(x_i, x_j)\right\} \quad (3.4)$$

Next, we define a reachability-graph by:

Definition 11 (reachability graph) *Let the reachability graph be a complete graph $G_{m_{Pts}} = (V, E)$. Each point $x_i \in X$ is represented by a vertex $v_i \in V$ and the weight of an edge $(v_i, v_j) \in E$ is equal to the reachability distance of corresponding points $w(v_i, v_j) = d_{reach,m_{Pts}}(x_i, x_j)$.*

We further filter edges of the graph using Kruskal's algorithm for building a minimum spanning tree KRUSKAL (1956). Each edge in the final tree represents a hierarchy level similar to the previous algorithm. A clustering for the parameters m_{Pts}, ϵ can be obtained by removing all edges with a weight greater than the ϵ threshold and calculating all connected components of the graph. It has to be noted that for the latter step only nodes with core distance equal or lower than ϵ are allowed to be expanded. Otherwise, border points which have edges to two or more clusters would merge those. Each core-distance can be added to the generated tree as a self-edge $(v_i, v_i) \in E$ with a weight of $w(v_i, v_i) = d_{\text{core},m_{\text{Pts}}}(v_i)$. The total number of edges in the enriched minimum spanning tree is $2n - 1 + b$, consisting of $n - 1$ edges of the generated minimum spanning tree, n self-edges and b edges of border points being part of multiple clusters. The number of border points participating in multiple clusters can be seen as marginal ($b << n$) in comparisons to the number of points in the dataset n. Even though exceptions are theoretically possible, they are not practically relevant. For this reason, the number of valid hierarchy levels generated by this method can be estimated with $2n$. The m_{Pts}-HDBSCAN based on building a minimal spanning tree is summarized in Algorithm 2.

A similar approach was discussed in CAMPELLO et al. (2013). Nevertheless their work excluded the handling of border points, For this reason they changed the definition of reachability distance to:

$$d_{\text{reach},m_{\text{Pts}}}(x_i, x_j)' = \max\{d_{\text{core},m_{\text{Pts}}}(x_i),\, d_{\text{core},m_{\text{Pts}}}(x_j),\, d(x_i, x_j)\} \quad (3.5)$$

Therefore, obtained clusters only contain core points and the result is different to the results of the algorithm discussed above.

Algorithm 2 m_{Pts}-HDBSCAN (Minimum spanning tree)

Input: m_{Pts}, *Dist* = pairwise distance matrix of **X**,
Optional: max_ϵ (default = ∞)

$G \leftarrow$ *empty_graph()*
$d_{\text{core},m_{\text{Pts}}} \leftarrow$ *calculate_core_distances(Dist, m_{Pts})*

for i **from** 1 **to** N **do**
  for j **from** $(i+1)$ **to** N **do**
    if $d_{\text{reach},m_{\text{Pts}}}(x_i,\ x_j) < max_\epsilon$ **then**
      $G.add\_edge(v_i,\ v_j,\ d_{\text{reach},m_{\text{Pts}}}(x_i,\ x_j))$
    end if
  end for
end for

$T \leftarrow$ *minimum_spanning_tree(G)*
for i **from** 1 **to** N **do**
  $T.add\_edge(v_i,\ v_i,\ d_{\text{core},m_{\text{Pts}}}(v_i))$
end for

return T

3.2.2 HDBSCAN for a fixed ϵ value — ϵ-HDBSCAN

Building a hierarchy based on a fixed value for m_{Pts} has been widely discussed in the previous section. Further on we will address necessary adjustments for an algorithm, which creates a hierarchy based on a fixed ϵ value.

Since the value for ϵ is fixed the neighborhood sets of each node can be computed beforehand. Our observations for the monotonicity of the core condition (see Equation 3.3) suggests to iteratively decrease the value of m_{Pts} for an agglomerative clustering. Starting point for the iterative process is setting m_{Pts} to the biggest neighborhood set size in the dataset and decreasing it stepwise until every node fulfills the core-condition. As shown in Subsection 3.1.2 a further increase of any cluster is impossible. The algorithm ϵ-HDBSCAN is summarized in Algorithm 3.

Algorithm 3 ϵ-HDBSCAN (Clustering-tree)

Input: ϵ, *Dist* = pairwise distance matrix of **X**

$\mathcal{C} \leftarrow \{\}$
clust_hierarchy $\leftarrow$ *initialize_clustertree*($\mathcal{C}$)
calculate neighborhood sets based on ϵ
$m_{\text{Pts}} \leftarrow \max(|N_\epsilon(p)|)$

while $|cores| < N$ **do**
  new_core_nodes = $\{x_i \in X \,|\, |N_\epsilon(x_i)| = m_{\text{Pts}}\}$
  add new_core_nodes to *cores*
  update_density_reachability(new_core_nodes)
  $\mathcal{C}_{m_{\text{Pts}}} \leftarrow$ *update_clustering*($\mathcal{C}$)
  if $\mathcal{C}_{m_{\text{Pts}}} \neq \mathcal{C}$ **then**
    clust_hierarchy.*add_clustering*($C_{m_{\text{Pts}}}$)
    $\mathcal{C} \leftarrow \mathcal{C}_{m_{\text{Pts}}}$
  end if
  $m_{\text{Pts}} \leftarrow m_{\text{Pts}} - 1$
end while

return clust_hierarchy

In the previous discussed m_{Pts}-HDBSCAN algorithm the hierarchy levels were bound to pairwise distances. In contrast to this, the set of hierarchy levels for a fixed ϵ can be determined by:

$$\{|N_\epsilon(x_i)\,|\,x_i \in X\} \tag{3.6}$$

As it was the case for m_{Pts}-HDBSCAN the ϵ-HDBSCAN algorithm can also be implemented using a spanning tree. Since the value of m_{Pts} needs to be decreased to increase the size of the cluster, either a maximum spanning tree needs to be computed or the values have to be rescaled to match a minimum spanning tree. The distance measure used for the computation of a minimum spanning tree is defined by:

$$d_{\text{reach},\epsilon}(i,j) = \begin{cases} \infty & ,\text{if } d(x_i,x_j) > \epsilon \\ N - \max\{|N_\epsilon(x_i)|, |N_\epsilon(x_j)|\} & ,\text{else} \end{cases} \tag{3.7}$$

The pseudo code in Algorithm 4 summarizes the minimum spanning tree based implementation of ϵ-HDBSCAN.

Algorithm 4 ϵ-HDBSCAN (Minimum spanning tree)

Input: ϵ, *Dist* = pairwise distance matrix of **X**,
Optional: $\min_{m_{\text{Pts}}}$ (default = 0)

G ← *empty_graph()*
$N_\epsilon \leftarrow$ *calculate_neighborhood_sets*(*Dist*, ϵ)

for source **from** 1 to N **do**
  if $|N_\epsilon(\text{source})| > \min_{m_{\text{Pts}}}$ **then**
    for target **in** N_ϵ(source) **do**
      if target > source **then**
        G.*add_edge*(source, target, $N - \max(|N_\epsilon(\text{source})|, |N_\epsilon(\text{target}|))$
      end if
    end for
  end if
end for

T ← *maximum_spanning_tree(G)*
for source **from** 1 to N **do**
  T.*add_self_edge*(source, d_{core}(source))
end for

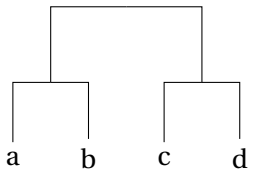
return T

3.2.3 Complexity of proposed HDBSCAN algorithms

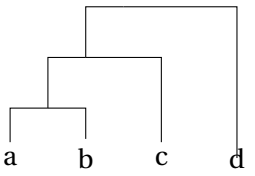
Provided implementations of proposed algorithms are far from optimal complexity. This chapter is used to discuss the complexity of each processing step. Table 3.1 lists the overall complexity of each algorithm.

Clustertree based implementation: The initialization of the clustertree has a complexity of $O(1)$. Calculating the neighborhood sets and the core distance of each point takes $O(N^2)$. Sorting the distance matrix using heap-sort equals a complexity of $O(N^2 \log N)$. Due to symmetry of the distance matrix the number of sorted entries can be halved by only sorting the upper triangle matrix. Note that sorting entries of the distance matrix is only necessary in m_{Pts}-HDBSCAN.

The complexity of updating density reachability and clustering labels depends on the structure of the resulting hierarchy. Figure 3.2 depicts two



(a) Merging clusters of equal size
levels = $\log_2(N)$, updates per level = $N/2$



(b) Adding single nodes
levels = $N-1$, updates per level = 1

Figure 3.2: Hierarchie structures and corresponding updates per level.

Table 3.1: Complexity of proposed clustering algorithms.

Algorithm	Clustertree	Minimum Spanning Tree
m_{Pts}-HDBSCAN	$O(N^2 \log(N))$	$O(N^2 \log(N))$
ϵ-HDBSCAN	$O(N^2)$	$O(N^2 \log(N))$

prime example structures. In the left subfigure the structure for a maximal number of hierarchy levels is depicted. Here, for each of the N levels the label of one point has to be updated with a complexity of $O(1)$. This sums up to a complexity of $O(N)$. The right subfigure depicts a hierarchy of equal merges with $\log_2(N)$ merges for a dataset of size N. Each level half ($N/2$) of the points labels have to be updated. Adding the hierarchy level to the clustertree has a complexity of $O(1)$.

Overall the complexity of the clustertree based implementation of m_{Pts}-HDBSCAN is bounded to the complexity of sorting the distance matrix ($O(N^2 \log(N))$). ϵ-HDBSCAN does not require to sort the distance matrix and has a overall complexity of $O(N^2)$.

Minimum spanning tree based implementation: Calculating neighborhood sets and core distances has a complexity of $O(N^2)$. The calculation of the minimum spanning tree has a complexity of $O(N^2 \log(N))$ for complete graphs. Reducing the number of edges in the reachability graph by fixing a maximum edge weight can be used to shorten the minimum spanning tree construction. Inserting remaining self edges has a cost of $O(N)$. The total complexity of both minimum spanning tree variants is dominated by the complexity of calculating the minimum spanning tree. Therefore, the overall complexity is $O(N^2 log(N))$.

3.3 Processing the clustering hierarchy

The algorithms in the previous sections both produce a hierarchy of clusterings. A monotonic hierarchy can be depicted as a dendrogram or clustertree (see Section 3.5 for further explanations). In the following we will discuss multiple cutting strategies from KIM et al. (2012) and review those in the context DBSCAN hierarchies. A simplified example dataset and corresponding hierarchy cuts are depicted in Figure 3.3.

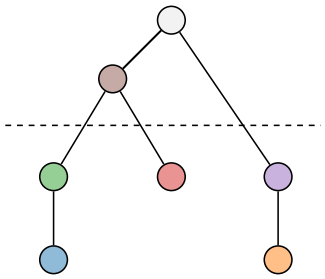
Horizontal cut by height: Each level of the dendrogram represents one parameter combination of $(\epsilon, m_{\text{Pts}})$. Cutting the dendrogram at a specific height results in the DBSCAN clustering using said parameters. The choice of the cut-height can be based on a minimum degree of spatial separation. Since this information is not always known it can be useful to choose the cut in between two consecutive clustering levels with a large height difference. The height difference is an indicator for the stability of the resulting clustering, since it describes the required parameter change to yield a different result. However, single outliers could lead to a maximal height difference by themselves, thus rendering this process ineffective. Figure 3.3a represents a simplified dendrogram and a horizontal cut determined by maximal height difference of consecutive levels.

Horizontal cut by number of clusters: As it is the case for HAC algorithms the estimated number of clusters is a valid criterion to fix a horizontal cut level of the dendrogram. Different to dendrograms produced by HAC, HDBSCAN dendrograms are not strictly monotonic on the number of clusters depending on the height level. New clusters can emerge or existing clusters can grow by increasing ϵ or decreasing m_{Pts}. For this reason multiple levels of the hierarchy are allowed to have the same number of clusters and therefore are potential candidates for a clustering of k clusters. Choosing which level to report can either be done by rating each clustering with an internal validation measure or using the highest level, which fulfills the k-cluster condition. The latter represents the maximal clustering regarding the number of points assigned to k-clusters. Figure 3.3b represents a simplified dendrogram and two cuts corresponding to a clustering of two clusters.

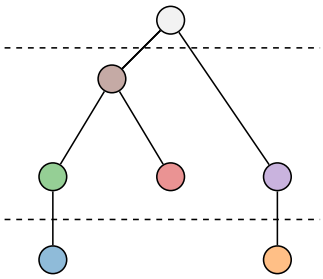
Non-horizontal cut by edge weight: The weight of a dendrogram edge connecting cluster C_i and C_j, such that $C_i \subset C_j$ can be measured by the height difference of the connected nodes. Experiments showed that the distribution of edge weights $\Phi(w)$ seems to follow one or multiple normal distributions. We can transfer the quantiles z_α of a standard normal distribution to specify a maximal weight for filtering edges by:

$$weight_{max} = z_\alpha \cdot \sigma(\Phi(w)) + \mu(\Phi(w))$$

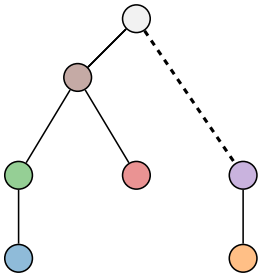
We insert cluster C_i to our final clustering hierarchy if its edge has a weight larger than the $weight_{max}$. Finally, nodes are assigned to the cluster of minimal height that contains them. Thus, the algorithm returns clusters of differing density. An example is provided in Figure 3.3c, in which the largest edge was cut, producing one small cluster contained in a supercluster.



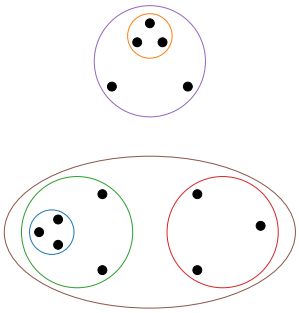
(a) Horizontal cut by height



(b) Horizontal cut by number of clusters



(c) Non-horizontal cut by edge weight



(d) Hierarchical example dataset

Figure 3.3: Visualization of dendrogram cuts for the dataset in (d)

3.4 Alternating Optimization for DBSCAN — aoDBSCAN

To further automatize the clustering process both hierarchical algorithms of DBSCAN can be combined in an alternating optimization approach. For this purpose an objective function on each clustering per hierarchy level has to be defined, which will be mini- or maximized during the search for an optimal parameter combination.

In each of the alternating optimization steps a hierarchical DBSCAN clustering will be carried out to produce a clustering dendrogram for either a fixed value of ϵ or m_{Pts}. This will return a clustering dendrogram, which contains all possible clusterings depending on the second parameter. Each horizontal cut implicates a valid parameter combination for which the respective clustering occurs. Rating cuts by an objective function lets us determine an appropriate value for the second parameter given the fixed first parameter. In the algorithm we choose the parameter value which yields the minimal or maximal value of the objective function. The next step of the optimization process would be to fix the second parameter at the optimal level found in the first step and initialize the respective algorithm to find the next optimal value pair. The whole process is summarized in Algorithm 5.

Algorithm 5 aoDBSCAN

Input: *Dist* = pairwise distance matrix of **X**

$m_{\text{Pts}} \leftarrow$ *random()* or set by user
repeat
  clust_hierarchy $\leftarrow$ m_{Pts}-HDBSCAN (m_{Pts}, *Dist*)
  ϵ = *get_best_*ϵ (clust_hierarchy)

  clust_hierarchy $\leftarrow$ ϵ-HDBSCAN (ϵ, *Dist*)
  m_{Pts} = *get_best_*m_{Pts}(clust_hierarchy)
until convergence of m_{Pts} and ϵ

return m_{Pts}, ϵ

In certain scenarios it can be beneficial to swap the order of m_{Pts}-HDBSCAN and ϵ-HDBSCAN. Since the initial estimation influences the found local optimum we recommend to use either ϵ-HDBSCAN or m_{Pts}-HDBSCAN,

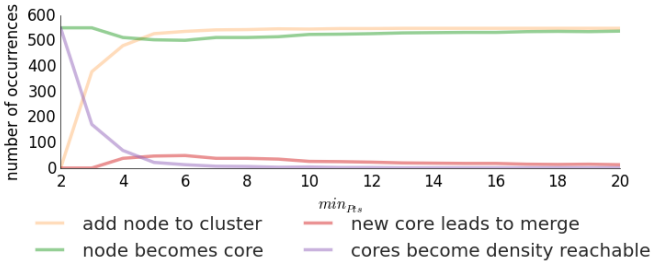


Figure 3.4: Clustering change distributions depending on parameter m_{Pts}.

depending on which parameter is easier to estimate. In our experiments it proved beneficial to start the optimization process using m_{Pts}-HDBSCAN, since in average the number of hierarchy levels is smaller than for ϵ-HDBSCAN.

Applying the objective function can be computationally costly if done for each hierarchy level. For this reason, additional filtering of the hierarchy levels can be applied for the clustering of large datasets. Figure 3.4 shows the frequency of clustering changes depending on the parameter m_{Pts} for the BlobsMoon dataset. As the graphs suggest the distribution of type of changes is dependent on the used value of m_{Pts}. Higher values of m_{Pts} lead to a decrease in border points ("new core leads to merge") and two clusters becoming density reachable to each other ("cores become density reachable"). Both types indicate larger changes of the underlying graph structure. Experiments showed that the clustering levels before and after the merge are good candidates for being rated by the objective function. However, Figure 3.5 shows that levels chosen by the filtering process do not contain all locally optimal hierarchy levels. For this reason, filtering should only be used for time critical tasks. Later on, a local search around the found optimum could be applied to find near optima.

3.4.1 Objective Functions

Alternating optimization handles the issue of finding an optimal pair of parameters, but leaves the choice of a suitable objective function. In this section we will review internal measures presented in Subsection 2.6.2 for the use as objective function in aoDBSCAN.

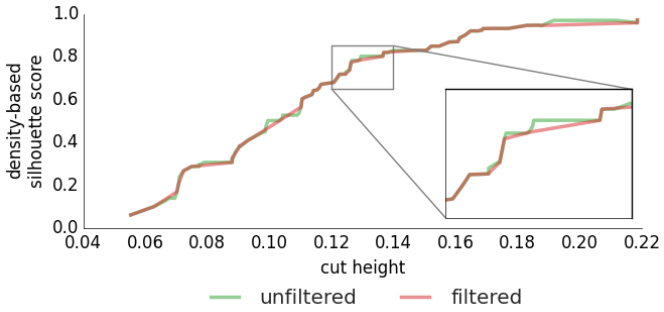


Figure 3.5: Horizontal cuts and their respective density-based silhouette score for filtered and unfiltered reporting of the "Moons" dataset.

(Density) Silhouette coefficient

Silhouette coefficient as proposed by ROUSSEEUW (1987) is a compactness based cluster validation measure. Experiments revealed that it can be used for the optimization process as long as the intercluster distance is relatively large in comparison to the average intracluster distance. However, DBSCAN can produce clusters of arbitrary shape, which do not necessary fulfill this condition. In the following we will discuss how standard silhouette coefficient can be adjusted for connectedness-based clusters.

The inclusion of the average distance to each point ($a(i)$) in the cluster leads the silhouette score to favor convex clusters. This property suits well for clustering algorithms searching for convex shapes, but limits the area of application for the DBSCAN algorithm.

Referring to the cluster attributes summarized in Section 2.1 the silhouette coefficient tries to optimize for compactness and spatial-separation. In the following we propose a density-based interpretation of the silhouette score, which changes the definition of $a(i)$. Aiming for measuring the difference of connectedness and spatial separation, a straightforward method would be to estimate the average density of the cluster and compare it to the minimal distance to another cluster. A natural way for the density estimation of a DBSCAN cluster would be to take the average necessary distance of all pairwise points of a cluster such that those are density-connected.

Let $G_{m_{\text{Pts}}}$ be the reachability graph and $G_{m_{\text{Pts}}}(C_i)$ the subgraph of $G_{m_{\text{Pts}}}$ only containing points assigned to cluster C_i. We define the cost of a path (cost(p)) to be the largest edge weight on the path.

Let $a(i)$ be:

$$a(i) = \frac{1}{n_{C_i} - 1} \sum_{j \in C_i,\, i \neq j} \arg\min_{p}(\text{cost}(p(i,j))) \tag{3.8}$$

Since this formula is too complex to evaluate for every hierarchy level, we suggest to use the ϵ value, for which the cluster C_i first emerged, which is equal to the largest edge-weight of $G_{m_{\text{Pts}}}(C_i)$. This results in an upper-bound $\bar{a}(i) \geq a(i)$, which is included in the density based calculation of the silhouette score. Based on the changed calculation of the silhouette score the silhouette coefficient of a cluster and a clustering are similar to the Equations 2.15 and 2.16. We later refer to the density based silhouette score of a clustering by $s^d_{\mathcal{C}}$.

$$\begin{aligned} \bar{a}(i) &= \arg\min_{\epsilon}(\text{for } C_i \text{ to exist}) \\ &= \max(\{\text{weight}(e), e \in E \mid G_{m_{\text{Pts}}}(C_i) = (V,E)\}) \end{aligned} \tag{3.9}$$

$$s^d(i) = \frac{b(i) - \bar{a}(i)}{\max\{\bar{a}(i), b(i)\}}; \quad s^d_{C_i} = \frac{1}{n_{C_i}} \sum_{o \in C_i} s^d(o); \quad s^d_{\mathcal{C}} = \frac{1}{n_{\mathcal{C}}} \sum_{C_i \in \mathcal{C}} s^d_{C_i} \tag{3.10}$$

In cases where all clusters are disjunct sets, the value of $b(i)$ will always be larger than the value of $\bar{a}_i$. As a result the density based silhouette coefficient of a clustering of disjunct sets will be in the range of $[0,1]$. In case border points are participating in in multiple clusters the value of $b(i)$ can be smaller than $\bar{a}(i)$. Since the denominator of the silhouette score is always larger than zero, the presence of such border points changes the range of a clusters density-based silhouette score to $(-1,1]$. This is especially a problem because of border points influencing multiple clusters. Because of this we suggest to remove border points from the calculation of the density-based silhouette coefficient.

The evaluation chapter will compare the performance of the silhouette coefficient and its density based interpretation. We will use the abbreviation $s_{\mathcal{C}}$ for results based on the original silhouette coefficient and $s^d_{\mathcal{C}}$ for the density based interpretation of silhouette coefficient, where a(i) is replaced by $\bar{a}(i)$ and border points are excluded from the calculation.

Dunn's index

As the study by AGGARWAL und REDDY (2013) suggests the validation measure proposed by Dunn is highly impacted by subclusters. In their experiments Dunn's index favored subclusters to be grouped together in one supercluster. While this can be a desired characteristic for other algorithms the application in our hierarchical scenario highly limits the range of cluster structures to be detected. It is to be expected that lower levels of the hierarchy will be rated generally lower than higher hierarchy levels. For this reason we omit Dunn's index from further evaluation as objective function for aoDBSCAN.

Clustering Validation Index based on Nearest Neighbor

While *CVNN* focuses on connectedness based clusters it seems unsuitable for the comparison of clusterings generated by DBSCAN. Looking at the separation (*Sep*) and compactness (*Com*) component we can see that *Sep* weights separation by the share of nearest neighbors assigned to other clusters than the point currently looked on. In case of DBSCAN this value converges to zero, because only a low number of border points participating in multiple cluster can be expected. For this reason, only the compactness component of *CVNN* influences the rating of a clustering. Since we already decided to use standard silhouette coefficient, which optimizes for compactness as well, we will omit *CVNN* from further evaluation as an objective function for aoDBSCAN. Therefore, we will not further evaluate it in Chapter 4.

Edge correlation

Edge correlation sticks out of the previous functions by using the whole similarity matrix as basis for the rating. The comparison of the labeling and the similarity tends to favor compact clusterings, because those better map block matrices. We will test the capabilities of using edge correlation as objective function criterion in our evaluation. It will show how intense compact cluster structures will be favored over connectedness based cluster shapes.

3.5 Visualization

To a large degree the practicability of a clustering algorithm is determined by suitable visualizations. While hierarchies generated by HDBSCAN contain a vast amount of information about the dataset and its structure, they can be difficult to interpret without appropriate visualization techniques. The following sections will review visualization techniques for HDBSCAN and aoDBSCAN to fully leverage both algorithms.

3.5.1 Dendrograms and clustertrees

Results of hierarchical methods are typically displayed using dendrograms. Links in the dendrogram mark merges of connected clusters. The horizontal height in which a merge occurs is equal to the distance of the affected clusters.

Ordinary dendrograms assume each node to be in a separate cluster at zero height. While this is true for HAC algorithms and various linkage criteria it does not transfer to DBSCAN using m_{Pts} values higher than two. Here, a cluster is first added to the hierarchy when the core-condition is fulfilled. Node pairs becoming part of each others neighborhood sets are not marked in the hierarchy if the core-condition is not fulfilled by at least one of them. For better interpretability we chose a clustertree to be an appropriate visualization method for hierarchies produced by HDBSCAN. Clusters changed due to an increase of the iterated parameter are marked using a new node in the hierarchy. As in the dendrogram subset relations are designated using edges between corresponding nodes. Later chapters will use the terms dendrogram and clustertree, when referring to hierarchies produced by HDBSCAN.

To further ease the interpretability of clusters in the clustertree, we add additional information to each node in the tree. A pie chart visualization containing information about the type of points in the cluster and their quantity is depicted in Figure 3.6. The outer ring can be added, by calculating objective function score for each cluster. However, this was not implemented yet due to the higher computational effort before drawing the hierarchy.

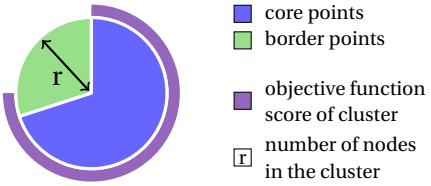


Figure 3.6: Pie chart visualization for nodes of a clustertree.

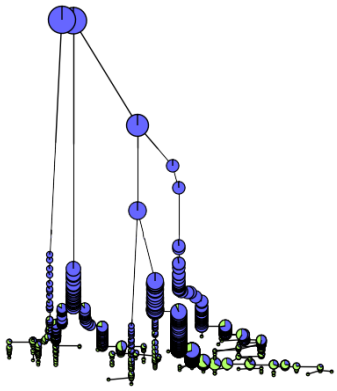


Figure 3.7: Visualization of a clustertree.

Figure 3.7 depicts the final version of a clustertree used for the visualization of HDBSCAN.

3.5.2 Interactive visualizations

DBSCAN parameters are easily interpretable in low dimensional clustering scenarios. Nevertheless, the choice of an appropriate ϵ value for high dimensional datasets can be complicated. Figure 3.8 depicts an interactive visualization of the parameter space. The left plot is a binned representation of the parameter space. Each cell contains the clustering of the according parameter combination. A cell's color is determined by the objective function score of the corresponding clustering. By clicking on a cell the scatter plot on the right displays the associated DBSCAN clustering.

The gridded visualization is fairly limited, due to the choice of the grid-size and a range of the parameter space to be explored. Consecutive grid cells do not necessary contain diverging clusterings of the dataset. For example, Figure 3.8 contains a large blue area on the right half of the parameter

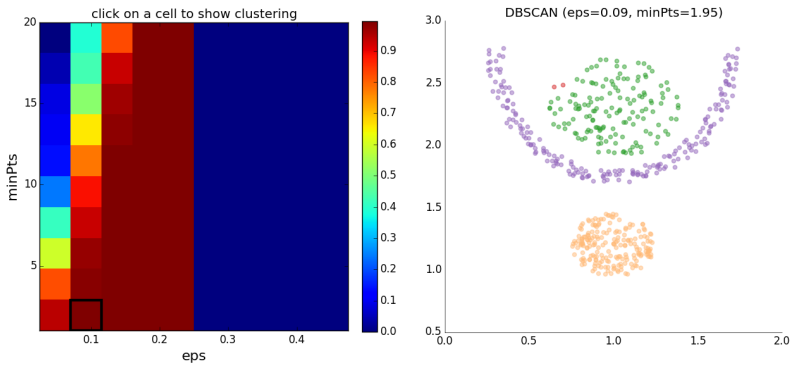


Figure 3.8: Interactive visualization of the parameter space and the density-based silhouette score to respective parameter combinations. A click on the gridded plot on the left updates the scatter plot on the right regarding the chosen parameter combination.

space. Each of these cells contain the same clustering in which only one large cluster exists.

Hierarchies generated by HDBSCAN already represent a condensed version of the parameter space, in which levels of differing clusterings are marked by new nodes. We provide an interactive version for horizontal and non-horizontal cuts for a user guided exploration of the parameter space. Figure 3.9 depicts a clustertree and a corresponding horizontal cut adjustable by the vertical slider on the left. The scatter plot on the right updates every time a new cut-height was chosen by the user. For interactive versions of the non-horizontal cuts by edge weight we chose to use a color scale for the visualization of edge weights. Figure 3.10 pictures the interactive version of non-horizontal cuts for HDBSCAN. As it was the case for horizontal cuts, a vertical slider on the left determines the maximal weight of an edge. The scatter-plot on the right updated every time a new cut was chosen by the user.

With aoDBSCAN we proposed a method which automatically explores the parameter space with the help of an objective function. Usually, the only feedback the user receives is the final state of the optimization process. However, it can be of interest how the result was obtained. Figure 3.11 depicts the learning process of the algorithm and the final result on the right side of the plot. Comparing this for multiple initializations can give insights in characteristics of the used objective function.

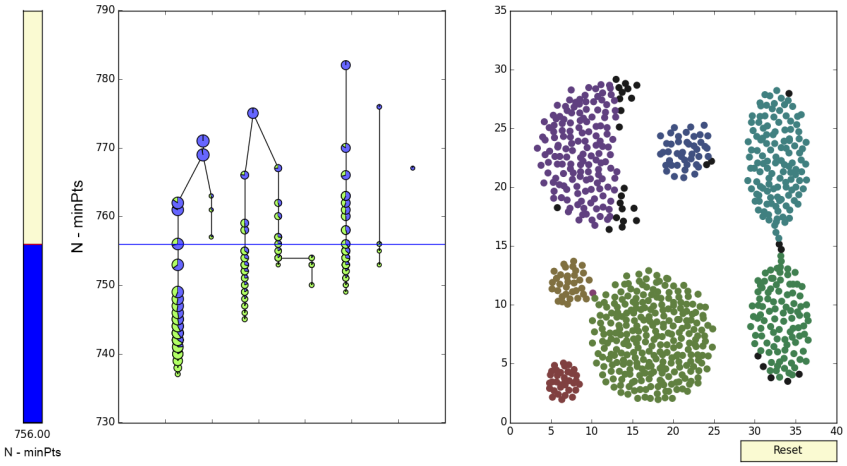


Figure 3.9: Interactive visualization for horizontal dendrogram cuts.

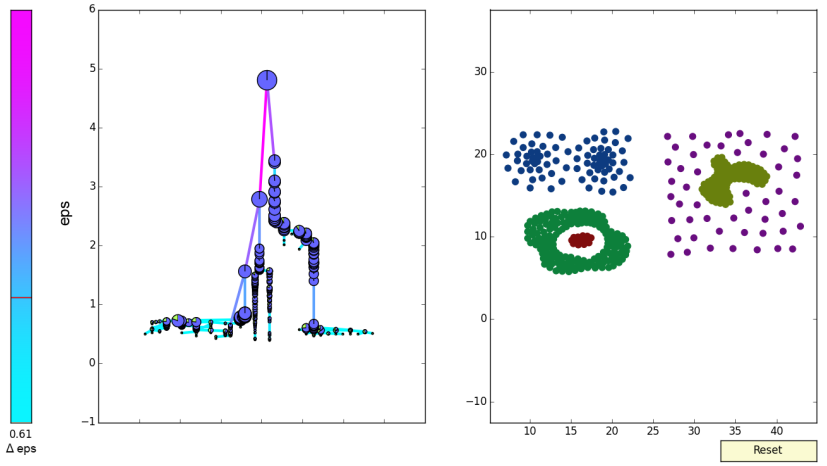


Figure 3.10: Interactive visualization for non-horizontal cuts by edge weight.

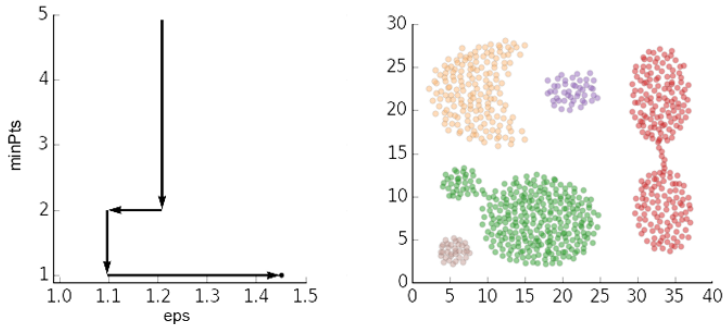


Figure 3.11: Learning process of aoDBSCAN for the depicted dataset on the right.

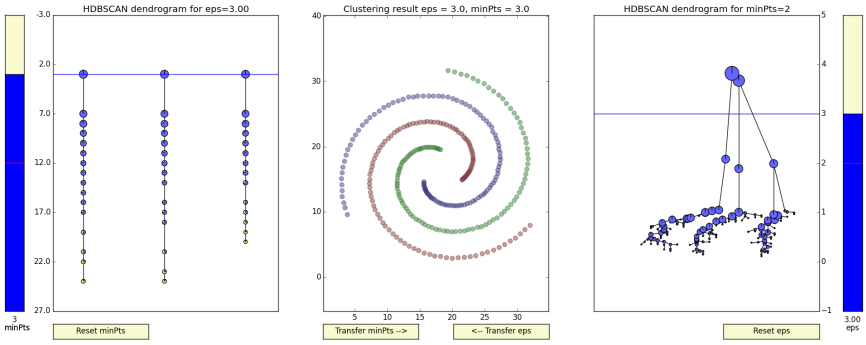


Figure 3.12: Interactive alternating optimization of DBSCAN. left: ϵ-HDBSCAN dendrogram; middle: scatter plot of the clustering for the current parameter combination; right: m_{Pts}-HDBSCAN dendrogram.

It can be of interest to execute the aoDBSCAN process by hand. For this purpose we created an interactive version of aoDBSCAN depicted in Figure 3.12. The left side contains an interactive hierarchy of ϵ-HDBSCAN, whereas the right side contains m_{Pts}-HDBSCAN. Each hierarchy can be cut horizontal by choosing a cut-height at the corresponding sliders on the side of both plots. Every time a cut-height is changed the scatter plot in the middle is updated for the corresponding value combination. Hierarchies can be recalculated based on the current cut-height by clicking one of the buttons below the scatter plot.

3.6 Summary

In the course of this chapter we proposed two hierarchical extensions for DBSCAN. Both use monotonic observations of the neighborhood set and the core-condition for efficient computation of the full hierarchy. However, the clustertree-based implementations still maintain a computational overhead by first sorting and then traversing the entries of the distance matrix. The concept of reachability distance and the corresponding reachability graph were introduced to further optimize the hierarchy generation process. This is based on minimum spanning tree implementation of HDBSCAN, which is computational more efficient and stores all necessary information about the clustering hierarchies.

Further processing of the hierarchy was discussed for creating flat or compressed hierarchical clusterings. While the first one was used to automatize the clustering process by implementing an alternating optimization approach, the second introduces a novel way for the extraction of clusters of differing density. AoDBSCAN eliminates the choice of both DBSCAN parameters and replaces it by the optimization of an internal validation criterion. For this reason, we reviewed multiple internal validation measures in the context of being used as objective function for aoDBSCAN. While we could cancel out Dunn's index and the *CVNN* score, we also proposed a density-based interpretation of the silhouette score and selected the silhouette coefficient, its density-based interpretation, and edge correlation as appropriate candidates.

Since we were already able to remove the parameter dependency from DBSCAN, we also wanted to improve the usability of proposed algorithms. For this reason, we introduced interactive visualizations, which help the user to get a grasp of the parameter space and information contained in returned hierarchies. Plots of the learning process and an interactive version of aoDBSCAN support the user throughout the clustering analysis and help to understand behavior and outcome of the applied clustering algorithm. These interactive elements help users, whom are unfamiliar with the topic, to fully utilize the capabilities of discussed clustering methods.

The following evaluation chapter will compare the performance of proposed algorithms with common clustering algorithms, while assessing the capabilities of selected objective functions.

4 Evaluation

4.1 Experiment setup

The following evaluation will compare clustering results of multiple algorithms in context of various cluster scenarios. Included algorithms and their used configuration are:

K-means: The k-means algorithm was used to generate a flat partition of k clusters. Parameter k was set to the true number of clusters in each dataset. To avoid the initialization problem of the cluster centers, each dataset was clustered 100 times and only the result with minimal sum of squared errors was reported.

Fuzzy-c-means: For fuzzy-c-means we used the same configuration as for the k-means algorithm. Fuzzifier w was set to 2.

HAC single linkage: The single linkage criterion was used to build a full clustering hierarchy of the dataset using hierarchical agglomerative clustering. A flat clustering was obtained using a horizontal dendrogram cut for k clusters, such that k is the true number of clusters.

HAC complete linkage: For the complete linkage criterion we used the same approach as for single linkage.

HAC wards minimum variance: The same applies to the wards minimum variance criterion.

OPTICS: Our evaluation of the OPTICS algorithm is based on the parameter settings $m_{\text{Pts}} = 4$ and 20 values of ϵ equidistantly distributed on a range of [0.1, 1.00]. Before applying the OPTICS algorithm the dataset was rescaled to the range of [0.1, 1.00] to match the iterated

ϵ values. The result with the best density based silhouette coefficient was reported.

CLIQUE: The CLIQUE algorithm was initialized several times using grid-sizes of (10×10), (15×15), (20×20) and (25×25). Results reported here are based on a density threshold of 4, which averaged as the best threshold value. The clustering with the best modified silhouette coefficient is recorded in the following evaluations.

HDBSCAN (edge quantile): m_{Pts}-HDBSCAN with $m_{\text{Pts}} = 4$ was used to produce a hierarchy of clusters, which was postprocessed using $\alpha \in \{0.1, 0.075, 0.05, 0.025, 0.01, 0.005\}$ for an estimate of the non-horizontal cut length. Edges (C_i, C_j), for which $C_i \subset C_j$, with higher length than the estimated cut length were cut and C_i was added to the reduced hierarchy. Nodes were assigned to the smallest cluster of the final hierarchy. Clustering results were rated using density-based silhouette coefficient and the best result was reported.

aoDBSCAN ($s_{\mathcal{C}}^d$): The alternating optimization was initialized by first using m_{Pts}-HDBSCAN with an initial value of $m_{\text{Pts}} = 5$. Each hierarchy level was rated using density-based silhouette coefficient to find an optimal horizontal cut of the hierarchy. The maximal number of iterations was set to 10. Plots of the resulting clustering use the heading "aoDBSCAN (modsil)".

aoDBSCAN ($s_{\mathcal{C}}$): The same approach as in aoDBSCAN ($s_{\mathcal{C}}^d$) was applied, with the change of using silhouette coefficient as rating criterion for the levels of the hierarchy. Plots of the resulting clustering use the heading "aoDBSCAN (sil)".

aoDBSCAN ($\rho_{\mathcal{C}}$): The same applies to aoDBSCAN ($\rho_{\mathcal{C}}$), with the change of using edge correlation as optimization criterion. Plots of the resulting clustering use the heading "aoDBSCAN (edgecore)".

Implementations of k-means and HAC algorithms were taken from the Python packages SciPy (JONES et al., 2001) and Scikit-learn (PEDREGOSA et al., 2011). For the fuzzy-c-means algorithm we used an implementation from the sci-kit-fuzzy package (WARNER et al., 2015). Additionally, an publicly available OPTICS Python implementation was provided by GRIGSBY

(2013). Remaining algorithms were self-implemented and are based on the descriptions provided in previous chapters.

The external validation measures entropy (E), purity (P), F-measure (F), and V-measure (V) were recorded for each clustering result and are shown in the corresponding table for each dataset. The best results of each column were written in bold and worst values were highlighted using italics. Note that in contrast to the other measures entropy has to be minimized for an optimal result.

4.2 Dataset results

Results will be discussed on the basis of the dataset characteristics in each scenario. Available datasets were divided into two groups. The first focuses on the clustering algorithms ability to find compact clusters and the second focuses on clusters based on varying connectedness. Both groups contain datasets with varying spatial separation of the clusters to test the algorithms sensitivity to spatial separation.

In the following each dataset will be shortly discussed and external validation results will be presented in a table. The clustering results of proposed algorithms are visualized in each section. A visualization of each algorithms clustering result per dataset can be found in Appendix A. In Section 4.3 we summarize our results of both categories and compare the overall accuracy of proposed algorithms.

4.2.1 Cluster scenarios based on compactness

R15

The R15 dataset contains 15 spherical shaped clusters with varying degrees of spatial separation. Table 4.1 summarizes the results of each clustering process. An almost correct clustering was achieved using k-means, fuzzy-c-means, complete linkage, and wards minimum variance. Few misassignments in the inner clusters are recorded, which is due to the missing spatial separation of few cluster pairs.

Single linkage assigns few points to singleton clusters. This is due to the larger distance of single noise points to the cluster they belong to. The behavior could have been avoided by fixing a minimum cluster size. In this case respective points would either be marked as noise or assigned to another cluster.

Proposed DBSCAN variants and common algorithms as OPTICS and CLIQUE detect all spatially well separated clusters of the outer circle, but perform bad in the clustering of the inner spheres. All 6 algorithms assign the 8 clusters in the middle to one large cluster. Therefore, purity values of those algorithms are competitive with the partitioning and HAC algorithm, but V and F-scores are much lower. The results of the proposed algorithms are depicted in Figure 4.1. A visualization of all clustering algorithms' results can be found at Figure A.1 in the appendix.

It could be argued that the inner circles form a region of higher density than the outer circles or their regions are strongly overlapping. A correct partition of the dataset could be achieved by fixing a high value for m_{Pts}. This would alter the reachability distance of each point and increase the reachability distance differences between points near to a clusters center and points at the outer border of a cluster.

Table 4.1: Evaluation results for the "R15" dataset.

Algorithm	E	P	V	F
Mini-Batch-K-Means	0.02	**1.00**	**1.00**	**0.99**
Fuzzy-C-Means	0.02	**1.00**	**1.00**	**0.99**
Single Linkage	0.08	0.99	0.76	0.88
Complete Linkage	0.06	0.99	0.99	0.98
Wards Minimum Variance	0.05	0.99	0.99	**0.99**
OPTICS	**0.00**	**1.00**	0.59	0.74
CLIQUE	*0.25*	*0.94*	*0.43*	*0.58*
HDBSCAN(edge quantile)	**0.00**	**1.00**	0.59	0.74
aoDBSCAN($s_{\mathcal{C}}^{d}$)	**0.00**	**1.00**	0.59	0.74
aoDBSCAN($s_{\mathcal{C}}$)	**0.00**	**1.00**	0.59	0.74
aoDBSCAN($\rho_{\mathcal{C}}$)	**0.00**	**1.00**	0.59	0.74

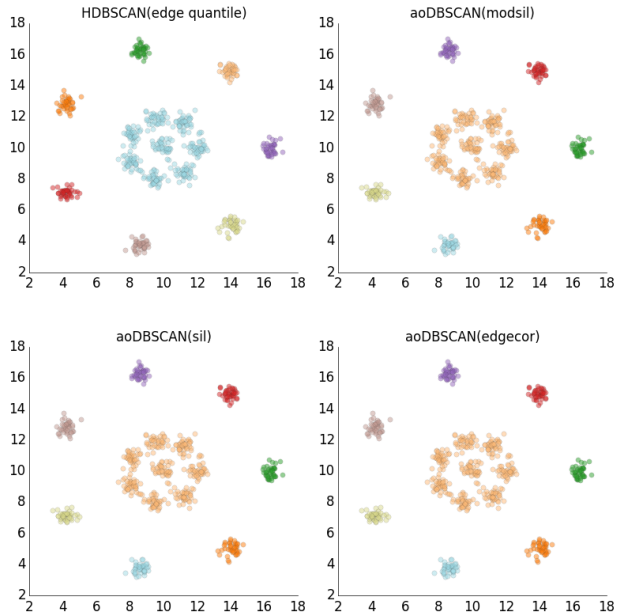


Figure 4.1: Clustering results for the dataset "R15" of proposed hierarchical DBSCAN variants.

Aggregation

The second dataset named Aggregation contains multiple ellipsoid clusters and one broad sickle in the top right corner. Two pairs of clusters are connected through a short bridge of points. Table 4.2 contains our experimental results for the Aggregation dataset.

Best clustering results are achieved by density-based algorithms. However, none of them correctly split both connected cluster pairs. For this reason single linkage, OPTICS, HDBSCAN (edge quantile), and aoDBSCAN using modified silhouette coefficient or edge correlation achieve a purity of $P = 1.00$, and V-measure or F-measure scores of about $V \approx 0.88, F \approx 0.88$.

The alternating optimization process using silhouette coefficient results in an overestimate of the parameter ϵ. Therefore, the clusters on the left side of the dataset are merged, which results in the worst score for V and F. Figure 4.2 depicts clustering results obtained by proposed algorithms.

Partitioning algorithms such as k-means and fuzzy-c-means perform well for clusters of equal size and spatially well separated clusters. While the two connected spheres on the right are separated correctly, the connected big and small sphere on the lower right corner is not separated due to different cluster sizes.

Complete linkage and wards minimum variance perform well in the split of the connected clusters on the right. Nonetheless, both do not split the clusters on the lower left corner correctly. On the contrary, CLIQUE correctly separates clusters on the left, except a few outliers at the clusters borders. Results of all clustering algorithms are visualized in Figure A.2.

Table 4.2: Evaluation results for the "Aggregation" dataset.

Algorithm	E	P	V	F
Mini-Batch-K-Means	1.26	0.83	0.85	0.89
Fuzzy-C-Means	**0.91**	*0.68*	0.74	0.76
Single Linkage	1.65	**1.00**	0.89	0.88
Complete Linkage	1.31	0.84	0.85	**0.91**
Wards Minimum Variance	1.36	0.88	0.87	0.90
OPTICS	*1.69*	**1.00**	0.89	0.89
CLIQUE	1.45	0.96	*0.66*	*0.51*
HDBSCAN(edge quantile)	*1.69*	**1.00**	0.89	0.89
aoDBSCAN($s_{\mathcal{C}}^{d}$)	1.63	0.99	0.88	0.88
aoDBSCAN($s_{\mathcal{C}}$)	1.59	0.98	**0.93**	**0.91**
aoDBSCAN($\rho_{\mathcal{C}}$)	*1.69*	**1.00**	0.89	0.89

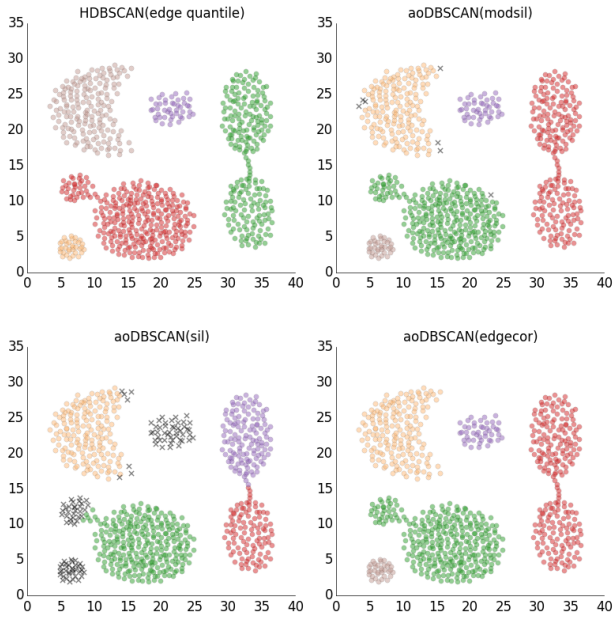


Figure 4.2: Clustering results for the dataset "Aggregation" of proposed hierarchical DBSCAN variants.

Blobs-1000D

The third dataset in the compactness category tests the clustering algorithms ability to cope with a higher number of dimensions. The dataset Blobs-1000D consists of 3 spherical shaped clusters of each 100 points in an 1000 dimensional space. It was expected that the curse of dimensionality already leads to problems in distinguishing the clusters, however only CLIQUE and OPTICS perform bad in this scenario. This could be due to the experimental setup and the configuration of the algorithms. Probably neither a valid grid-size nor an appropriate ϵ-value was used, such that the clustering of both algorithms fail to distinguish the groups.

Remaining algorithms perform well and result in the correct clustering. The visualizations in Figure A.3 and Figure 4.3 show a scatter-plot of the first two dimensions. Table 4.3 summarizes the recorded validation measures for all algorithms.

Table 4.3: Evaluation results for the "Blobs-1000D" dataset.

Algorithm	E	P	V	F
Mini-Batch-K-Means	**0.03**	**1.00**	**1.00**	**1.00**
Fuzzy-C-Means	**0.03**	**1.00**	**1.00**	**1.00**
Single Linkage	**0.03**	**1.00**	**1.00**	**1.00**
Complete Linkage	**0.03**	**1.00**	**1.00**	**1.00**
Wards Minimum Variance	**0.03**	**1.00**	**1.00**	**1.00**
OPTICS	**0.03**	**1.00**	*0.51*	*0.00*
CLIQUE	*0.17*	*0.97*	0.74	0.60
HDBSCAN(edge quantile)	**0.03**	**1.00**	**1.00**	**1.00**
aoDBSCAN($s_{\mathcal{C}}^{d}$)	0.09	0.99	0.99	0.98
aoDBSCAN($s_{\mathcal{C}}$)	0.09	0.99	0.99	0.98
aoDBSCAN($\rho_{\mathcal{C}}$)	0.09	0.99	0.99	0.98

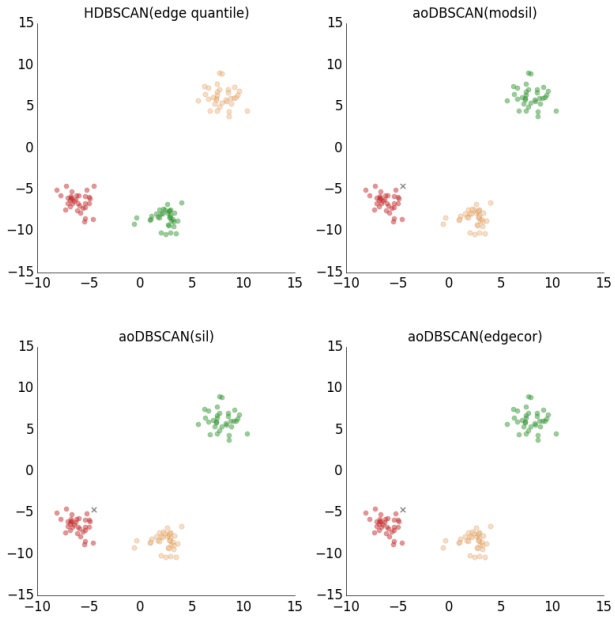


Figure 4.3: Clustering results for the dataset "Blobs-1000D" of proposed hierarchical DBSCAN variants.

4.2.2 Cluster scenarios based on connectedness

Moons

The Moons dataset is the first representative of clusters based on connectedness. It consists of two interlocked crescent shaped clusters with a small degree of noise. In our evaluation single linkage and aoDBSCAN using silhouette coefficient perform best. Both yield a perfect clustering according to the true partition of the dataset.

Other aoDBSCAN variants result in a nearly correct clustering. All exclude few noise points, which could be argued to be outliers. Figure 4.4 shows the clustering result of proposed algorithms.

Surprisingly, OPTICS and CLIQUE do not return a correct clustering. Both clustering results show two gaps in the lower sphere. This could be due to the preset m_{Pts} value of 4 and no check of an appropriate ϵ or grid-size. Possibly, the number of points per cell at the ends of the bottom sickle is to small for the fixed threshold of 4. Changing the location of the grid or lowering the threshold could improve the clustering result.

The remaining algorithms k-means, fuzzy-c-means, complete linkage, and wards minimum variance do not reflect the correct shape of both clusters. While k-means and fuzzy-c-means are forced to find a linear separation of the dataset, the hierarchical algorithms tend to produce spherical shaped clusters. Both strategies are not sufficient to result in a correct clustering. Their respective clustering results can be found in Figure A.4 in the appendix. External validation measures of each clustering result are recorded in Table 4.4.

Table 4.4: Evaluation results for the "Moons" dataset.

Algorithm	E	P	V	F
Mini-Batch-K-Means	0.81	0.75	0.75	0.19
Fuzzy-C-Means	0.81	0.75	0.75	0.19
Single Linkage	**0.00**	**1.00**	**1.00**	**1.00**
Complete Linkage	0.77	0.76	0.76	0.22
Wards Minimum Variance	0.38	0.89	0.89	0.60
OPTICS	0.49	0.80	0.81	0.42
CLIQUE	1.02	0.67	0.70	0.58
HDBSCAN(edge quantile)	0.09	0.99	0.99	0.96
aoDBSCAN($s_{\mathcal{C}}^{d}$)	0.15	0.98	0.97	0.92
aoDBSCAN($s_{\mathcal{C}}$)	**0.00**	**1.00**	**1.00**	**1.00**
aoDBSCAN($\rho_{\mathcal{C}}$)	*1.18*	*0.65*	*0.64*	*0.15*

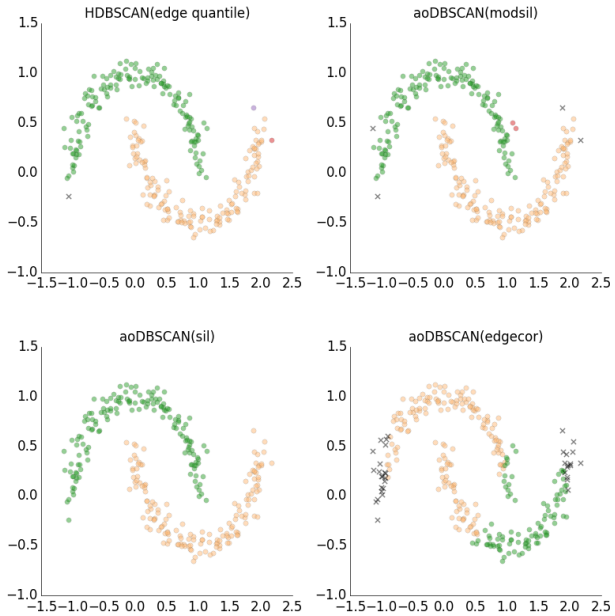


Figure 4.4: Clustering results for the dataset "Moons" of proposed hierarchical DBSCAN variants.

BlobsMoon

The BlobsMoon dataset consists of two spheres of different size and density. The upper sphere is encased by a semicircle shaped cluster. All aoDBSCAN variants as well as HAC using single linkage result in a correct clustering.

The remaining density based algorithms merge both clusters on the top to one bigger cluster. Clique additionally separates few points at the border of the cluster into smaller clusters. It is expected that this behavior can be avoided with another localization of the underlying grid or by changing the density threshold. Examples for the correct classification of the aoDBSCAN variants and mistakes made by HDBSCAN can be seen in Figure 4.5.

K-means and fuzzy-c-means have the lowest purity values. This is due to a vertical split of both upper clusters. Wards minimum variance and complete linkage result in similar mistakes, but to a lower extent than both partitioning algorithms.

Results of used clustering algorithms can be found at Figure A.5. Table 4.5 contains corresponding evaluation measures.

Table 4.5: Evaluation results for the "BlobsMoon" dataset.

Algorithm	E	P	V	F
Mini-Batch-K-Means	*0.87*	*0.69*	0.70	*0.52*
Fuzzy-C-Means	0.82	*0.69*	*0.69*	0.55
Single Linkage	**0.11**	**1.00**	**1.00**	**1.00**
Complete Linkage	0.38	0.93	0.76	0.71
Wards Minimum Variance	0.72	0.76	0.72	0.61
OPTICS	**0.11**	**1.00**	0.83	0.75
CLIQUE	0.59	0.88	0.73	0.62
HDBSCAN(edge quantile)	**0.11**	**1.00**	0.83	0.75
aoDBSCAN($s_{\mathcal{C}}^{d}$)	**0.11**	**1.00**	**1.00**	**1.00**
aoDBSCAN($s_{\mathcal{C}}$)	**0.11**	**1.00**	**1.00**	**1.00**
aoDBSCAN($\rho_{\mathcal{C}}$)	**0.11**	**1.00**	**1.00**	**1.00**

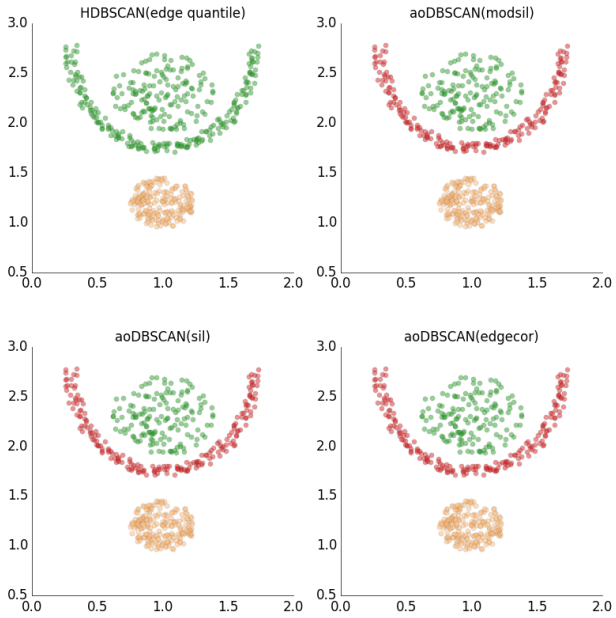


Figure 4.5: Clustering results for the dataset "BlobsMoon" of proposed hierarchical DBSCAN variants.

Spiral

The Spiral dataset consists of three entwined spirals with slightly decreasing density at the outer ends. This dataset is a typical counterexample for cluster oriented on compactness. As the result shows, algorithms like single linkage perform very well on such kinds of datasets, while k-means and fuzzy-c-means completely fail to capture the clusters structure. Both partitioning algorithms result in a F-measure of about $F = 0.00$.

As it was already stated earlier, DBSCAN is equal to single linkage for $m_{\text{Pts}} = 1$. The basic initialization of $m_{\text{Pts}} = 5$ would not be sufficient to cluster the dataset correctly using DBSCAN. However, the alternating optimization process for density-based and standard silhouette coefficient result in an appropriate parameter combination for DBSCAN. Table 4.6 contains recorded values of the evaluation process. In contrast to those, edge correlation does not lead to a suitable parameter combination and results in an F-measure score of $F = 0.04$. HDBSCAN has a high accuracy in the clustering process but omits the last point of each spiral. This can be seen in Figure 4.6, which depicts clustering results of proposed algorithms.

The CLIQUE algorithm performs especially bad regarding the purity and entropy measures. This is due to the thin spirals diagonally crossing the grid cells, while the used implementation only considered adjacent vertical and horizontal cells for the connection of clusters.

For a detailed comparison clustering results of all used clustering algorithms are depicted in Figure A.6.

Table 4.6: Evaluation results for the "Spiral" dataset.

Algorithm	E	P	V	F
Mini-Batch-K-Means	1.54	0.38	0.35	*0.00*
Fuzzy-C-Means	1.54	*0.35*	*0.34*	*0.00*
Single Linkage	**0.04**	**1.00**	**1.00**	**1.00**
Complete Linkage	1.51	0.43	0.39	0.01
Wards Minimum Variance	0.91	0.71	0.63	0.27
OPTICS	**0.04**	**1.00**	**1.00**	**1.00**
CLIQUE	*1.84*	0.42	0.36	0.08
HDBSCAN(edge quantile)	**0.04**	0.99	0.99	0.98
aoDBSCAN($s_{\mathcal{C}}^{d}$)	**0.04**	**1.00**	**1.00**	**1.00**
aoDBSCAN($s_{\mathcal{C}}$)	**0.04**	**1.00**	**1.00**	**1.00**
aoDBSCAN($\rho_{\mathcal{C}}$)	0.68	0.78	0.51	0.04

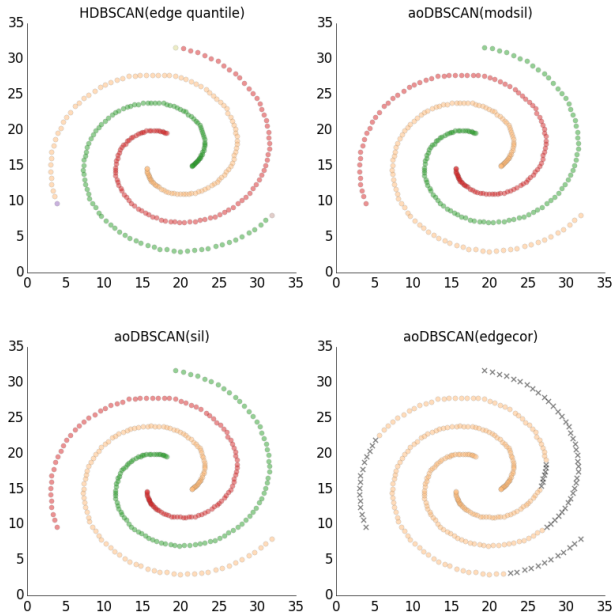


Figure 4.6: Clustering results for the dataset "Spiral" of proposed hierarchical DBSCAN variants.

Compound

The Compound dataset is used to check for the algorithms capabilities of adapting to different cluster shapes and densities. It consists of two gaussian distributed clusters on the top left blending into each other, one dense cluster on the right encompassed in a sparse area, and two clusters on the lower left including one sphere surrounded by a spatially separated circular cluster.

Density-based algorithms as DBSCAN, CLIQUE and OPTICS are generally not capable of returning clusters of varying density. The alternating optimization approach fails to extract most clusters with all three optimization criteria. Their high purity values are a result of merging clusters on the left and right, due to the large gap between those. OPTICS and Clique further divide the groups, but overall could not divide both various cluster pairs.

Both partitioning algorithms perform well for the gaussian clusters on the top left corner. However, they fail to capture the cluster structure of remaining clusters, since those are not compact in shape.

The clustering using single linkage fails, due to the sparse cluster on the right. Single Nodes are isolated in the hierarchy, which is a result of the well known chaining effect of single linkage. Complete linkage and wards minimum variance suffer from the same problem as the algorithms k-means and fuzzy-c-means.

HDBSCAN outperformed previous clustering algorithms by finding a nearly perfect non-horizontal cut of the hierarchy. As it can be seen in Table 4.7 HDBSCAN performs best in the V-measure and the F-measure with scores of approximately 0.99. Just a single point on the top-right corner is excluded from the surrounding points. Figure 4.7 depicts the clustering result of HDBSCAN and aoDBSCAN variants. A full comparison of clustering results can be found in Figure A.7.

Table 4.7: Evaluation results for the "Compound" dataset.

Algorithm	E	P	V	F
Mini-Batch-K-Means	0.05	0.68	0.72	0.72
Fuzzy-C-Means	0.08	*0.66*	*0.70*	0.71
Single Linkage	0.69	0.99	0.84	0.80
Complete Linkage	0.50	0.91	0.84	0.81
Wards Minimum Variance	**0.01**	0.70	0.71	0.73
OPTICS	*0.74*	**1.00**	0.84	0.81
CLIQUE	0.45	0.93	0.88	0.77
HDBSCAN(edge quantile)	0.68	0.99	**0.99**	**0.98**
aoDBSCAN($s_{\mathcal{C}}^{d}$)	*0.74*	**1.00**	0.77	*0.59*
aoDBSCAN($s_{\mathcal{C}}$)	*0.74*	**1.00**	0.77	*0.59*
aoDBSCAN($\rho_{\mathcal{C}}$)	*0.74*	**1.00**	0.77	*0.59*

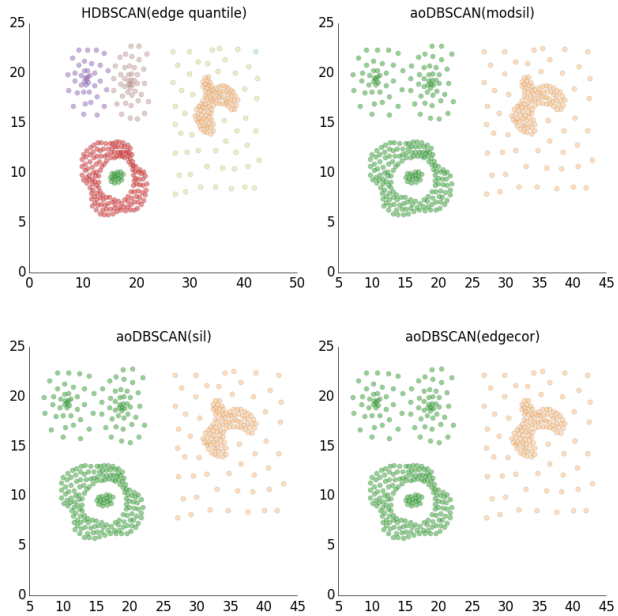


Figure 4.7: Clustering results for the dataset "Compound" of proposed hierarchical DBSCAN variants.

Flame

The Flame dataset contains two clusters connected by a sparse area. Two outliers were added to the upper left corner.

The alternating optimization approaches using standard and density-based silhouette coefficient show to be highly influenced by the two outliers. The large gap between the two outliers and the remaining points result in a higher minimal distance between two clusters than the division of both clusters through the sparse area. The edge quantile cut of HDBSCAN suffers from the same problem. CLIQUE correctly identifies the mass centers of both clusters, but excludes sparser areas at the clusters borders. Same behavior could probably be achieved for the edge quantile cut by increasing m_{Pts}. Clustering results of proposed algorithms are depicted in Figure 4.8.

Neither k-means nor fuzzy-c-means are able to find a linear separation of both clusters. However, both chose a cut through the sparse area, since the mass centers of the remaining clusters are in the middle of the desired clusters. This approach results in the best values regarding the V-measure. Similar problems could be observed for complete linkage and wards minimum variance.

Figure A.8 shows a full comparison of clustering results. Corresponding scores per evaluation measure are recorded in Table 4.8.

Table 4.8: Evaluation results for the "Flame" dataset.

Algorithm	E	P	V	F
Mini-Batch-K-Means	0.02	0.85	**0.85**	**0.48**
Fuzzy-C-Means	0.05	0.85	**0.85**	0.44
Single Linkage	0.46	**0.99**	0.78	0.02
Complete Linkage	**0.00**	0.86	0.63	0.07
Wards Minimum Variance	0.11	0.72	0.72	0.33
OPTICS	0.46	**0.99**	0.78	0.02
CLIQUE	*0.80*	*0.52*	*0.55*	0.38
HDBSCAN(edge quantile)	0.45	**0.99**	0.78	0.02
aoDBSCAN($s_{\mathcal{C}}^{d}$)	0.46	**0.99**	0.78	0.02
aoDBSCAN($s_{\mathcal{C}}$)	0.46	**0.99**	0.78	0.02
aoDBSCAN($\rho_{\mathcal{C}}$)	0.22	0.79	0.62	*0.00*

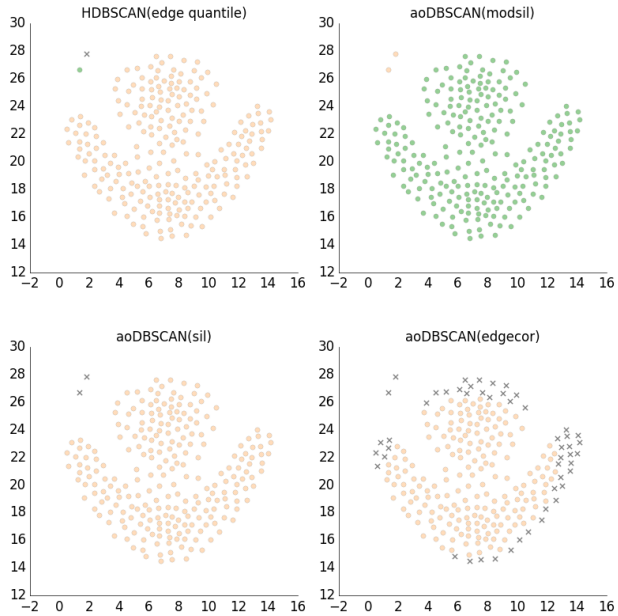


Figure 4.8: Clustering results for the dataset "Flame" of proposed hierarchical DBSCAN variants.

Moons-DifDense

As the Compound dataset the dataset Moons-DifDense consists of two clusters of differing density. Clusters are of similar shape and positioning as in the Moons dataset. Since only two density levels are included, it is also possible to rate the sparse cluster as noise. The influence of the changes can be observed in Table 4.9.

The performance of k-means and fuzzy-c-means is the same compared to the Moons dataset with equal density levels. Complete linkage and wards minimum variance show no big changes as well.

In contrast to partitioning algorithms, density-based methods and single linkage are largely affected by the change. HDBSCAN and OPTICS perform well, whereas all aoDBSCAN variants fail to capture the desired cluster structure. HDBSCAN chose to include three hierarchy levels and one noise point. The same noise point is also excluded using aoDBSCAN. Excluding the noise point could be attributed to the used internal validation measure for HDBSCAN. THe clustering results of proposed algorithm are depicted in Figure 4.9. For a full comparison of clustering results see Figure A.9 in the appendix.

Table 4.9: Evaluation results for the "Moons-DifDense" dataset.

Algorithm	E	P	V	F
Mini-Batch-K-Means	*1.09*	*0.76*	*0.74*	0.34
Fuzzy-C-Means	1.07	0.77	0.76	0.35
Single Linkage	0.61	0.93	0.85	0.25
Complete Linkage	0.58	0.95	0.95	0.70
Wards Minimum Variance	0.91	0.86	0.85	0.51
OPTICS	**0.39**	**1.00**	**1.00**	**1.00**
CLIQUE	0.85	0.84	0.84	0.22
HDBSCAN(edge quantile)	0.54	0.98	0.97	0.90
aoDBSCAN($s_{\mathcal{C}}^{d}$)	0.41	**1.00**	0.85	*0.01*
aoDBSCAN($s_{\mathcal{C}}$)	0.72	0.89	0.90	0.44
aoDBSCAN($\rho_{\mathcal{C}}$)	0.69	0.89	0.90	0.46

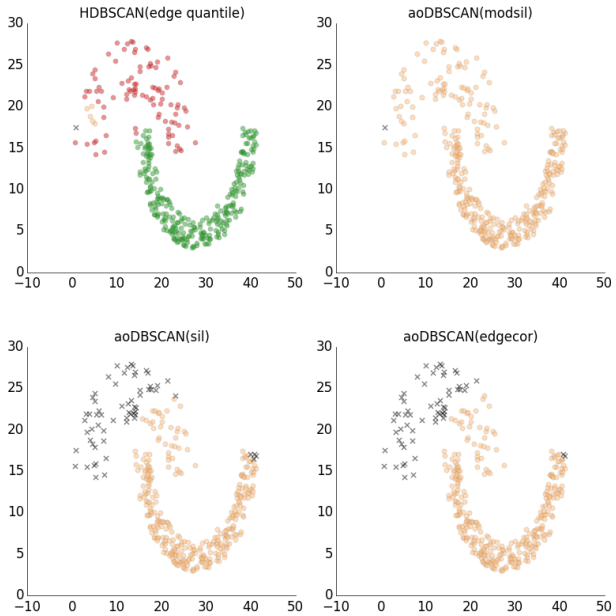


Figure 4.9: Clustering results for the dataset "Moons-DifDense" of proposed hierarchical DBSCAN variants.

4.3 Summary

The performed evaluation tested proposed and common standard algorithms in a set of clustering scenarios. In the following, we will shortly summarize the results of the experiments and their possible implications.

Compactness-based clustering datasets were used to test for the influence of dimensionality, degree of compactness, and spatial separation of individual clusters on the clustering accuracy. Dimensionality did not influence the accuracy of proposed algorithms as well as of standard algorithms. HDBSCAN as well as aoDBSCAN were able to find a correct clustering for the dataset. We expected the curse of dimensionality to take effect for various algorithms, however the large degree of spatial separation of the clusters seems to prevented this effect. Results of the "R15" dataset showed that compact clusters, which are loosely connected and form a dense area of clusters, could not be clustered correctly by proposed algorithms. This could have been prevented by providing the true number of clusters in advance and choosing an appropriate hierarchy cut. However, this is an unreasonable assumption for the clustering task, since one of the advantages of DBSCAN is the automatic detection of the number of clusters. Still, the returned clustering is reasonable interpretation of the dataset concerning the higher density in the center of the dataset. The "Aggregation" dataset consisted of clusters of differing size and shape. Most algorithms failed to identify a correct partition of connected clusters. It is questionable whether the bridges between affected cluster-pairs are negligible in the context of the connectedness criterion of clusters. All used objective functions led to nearly similar results. Standard silhouette coefficient performed negligibly better than both concurrents for the "Aggregation" dataset by correctly separating the cluster pair on the right. A merge of both clusters would move the center of the resulting dataset to the middle of the bridge, which leads to a large decrease of the respective silhouette score due to the contained average function. Generally, no preference for a particular objective function can be justified for compactness-based cluster scenarios.

In the second part we investigated clustering accuracy for connectedness-based clusters. In four of six edge correlation did not perform as well as the two silhouette score variants. Neither the "Moons" datasets nor the "Spi-

ral" dataset were clustered correctly. Whereas, standard silhouette score and its density-based interpretation resulted in appropriate parameters for cluster scenarios containing clusters of equal density. Nevertheless, due to the limitations of DBSCAN, it is not possible to find an appropriate parameter combination for clustering the "Compound" dataset. Here, HDBSCAN in combination with non-horizontal cuts exclusively resulted in a nearly perfect clustering.

Overall proposed methods leaded to comparative results in most clustering scenarios and even surpassed other clustering algorithms in the context of connectedness-based clusters. Silhouette coefficient and density-based silhouette coefficient outperformed edge correlation and seem to be an appropriate choice for the optimization. Nevertheless, HDBSCAN is the only method capable of detecting clusters of differing density.

5 Conclusions and Future Work

5.1 Conclusions

In the present thesis we improved usability and performance of density-based clustering algorithms. Using DBSCAN as one common representative clustering algorithm of this paradigm, we proposed the two hierarchical extensions m_{Pts}-HDBSCAN and ϵ-HDBSCAN. However, checking every possible parameter of DBSCAN would be computational inefficient without optimization. By exploiting monotonicity of neighborhood sets and the core condition, the hierarchy generation process of proposed algorithms was improved. Thus, only necessary parameter values are processed and added to a hierarchy of clusterings. Furthermore, the hierarchical clustering process of HDBSCAN was enhanced by interactive visualizations of both proposed algorithms. Those increase the usability of proposed hierarchical variations of DBSCAN and can help to adjust parameters for the desired clustering.

Since adjusting DBSCAN parameters by hand is only applicable for low dimensional datasets, we proposed a greedy optimization process for determining an appropriate value combination. We combined both hierarchical clustering algorithms in aoDBSCAN and replaced the necessity of fixing two parameters with the introduction of an optimization criterion. The optimization process guarantees to find a local optimal parameter combination by maximizing the objective function.

Proposed algorithms were compared to various partitioning, hierarchical and density-based algorithms. Algorithms were compared on various clustering scenarios based on either compact or connected structures. HDBSCAN, which used non-horizontal cuts, as well as aoDBSCAN proved

to produce comparable clustering results to other standard algorithms. HDBSCAN using non-horizontal cuts even surpassed other methods in detecting clusters, which differ in density. AoDBSCAN showed to be capable of determining appropriate parameter combinations for DBSCAN by choosing an optimization criterion. These enhancements increased the usability of DBSCAN and make it applicable for high dimensional datasets, for which visual methods fail to provide interpretable feedback for the choice of parameters.

The development of proposed algorithms was accompanied by the planning and implementation of interactive visualizations. Those can help to close the gap between experts and non-expert users by visualizing the parameter space and corresponding clusterings. The following section will conclude this work with final remarks about possible future developments.

5.2 Future Work

In the thesis at hand we proposed multiple algorithms and described their current state of development. However, the evaluation showed that the performance of proposed algorithms still needs to be improved to successfully cluster various scenarios. In the following, we will summarize possible extensions and enhancements, which could be subject of future research.

Developing new objective functions: The aoDBSCAN approach shifted the preparation for the clustering algorithm from choosing appropriate parameters to selecting an internal validation criterion. Our experiments revealed that different local optima are chosen per cluster validation criterion. We suggest to investigate the influence of existing rating criteria and developing new density-based internal validation measures.

Weighted density-based silhouette coefficient: In our experiments density-based silhouette coefficient was highly influenced by outliers (see Flame dataset). Weighting the silhouette score of every cluster by the size of the respective cluster could cope with this problem. The behavior of a weighted mean density-based silhouette coeffi-

cient used as an objective function for aoDBSCAN could be subject of future experiments.

Further improving non-hierarchical cuts: The proposed method for non-hierarchical cuts is based on height differences of consecutive hierarchy levels. Large differences in cluster densities would decrease the efficiency of this approach, because the edge length distribution would contain multiple normal distributions. A local cutting criterion would enable us to detect a transition from one to another normal distribution. ZAHN (1971) proposed a consistency rating of a hierarchy level by comparing it to the average height of surrounding cluster levels. Implementing Zahn's consistency rating for dendrograms produced by HDBSCAN could improve the detection rate of clusters with differing densities.

Non-hierarchical cuts as stability check: Up to this point only horizontal cuts can be used for aoDBSCAN. Single HDBSCAN hierarchies support non-horizontal cuts as well. They can be used to find stable clusters in the dataset by cutting hierarchies of multiple m_{Pts} values and comparing the clustering results. In addition, the stability of a cluster could be measured by the number of density levels it occurs in.

Search strategies for appropriate hierarchy levels: In Section 3.2 we discussed that a large number of hierarchy levels have to be rated to find the level with maximal internal validation score. A solution for minimizing the amount of hierarchy levels is filtering appropriate candidates for rating. Another approach would be to apply search algorithms to the hierarchy levels. For example, gradient-descent could be applied in case the search space is continuous.

Streaming algorithm for HDBSCAN: The minimum spanning tree implementation proved to be faster because it generated less hierarchy levels. However, the large number of hierarchy levels ($2n$) can still render the spanning tree generation unpractical for large datasets. The thesis of RIBICHINI (2007) summarized graph algorithms for streaming scenarios and reviewed parallel processing techniques for speeding up the generation of minimum spanning trees. The implementation of parallel processing techniques could reduce the run-time

of the algorithm and, therefore, speed up the generation of the full hierarchy for large datasets.

Other hierarchical algorithms: The discussed monotonicity behavior can also be exploited for other density-based clustering algorithms including a fixed density-threshold. For example the CLIQUE algorithm could be extended by storing density values of every cell and building up a hierarchy of clusterings as it was done for HDBSCAN. Since in our experiments CLIQUE often suffered from the problem of finding a correct density threshold this approach could automatically detect needed density thresholds per cluster.

A

Detailed Results

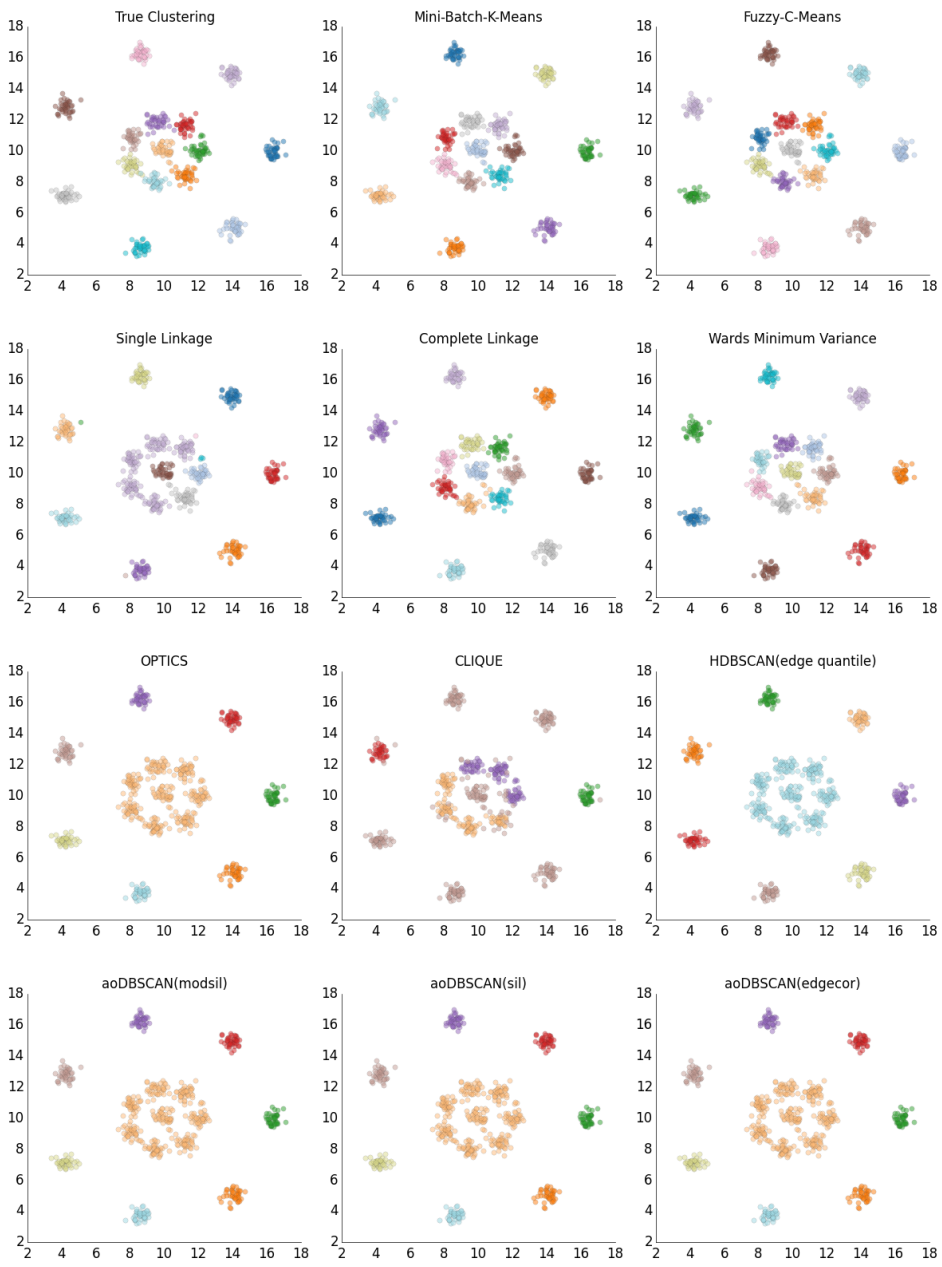


Figure A.1: Clustering results of the R15 dataset.

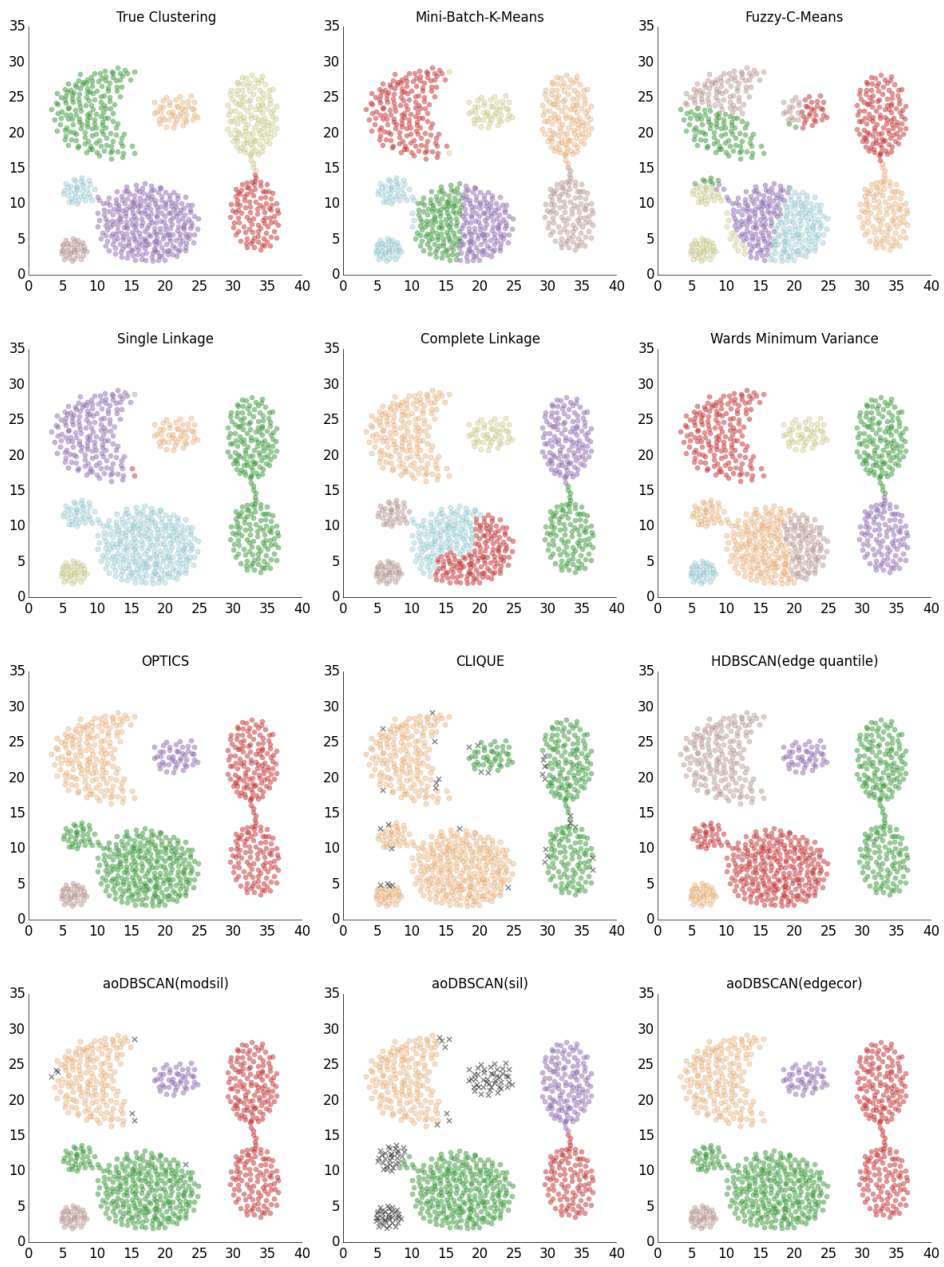


Figure A.2: Clustering results of the Aggregation dataset.

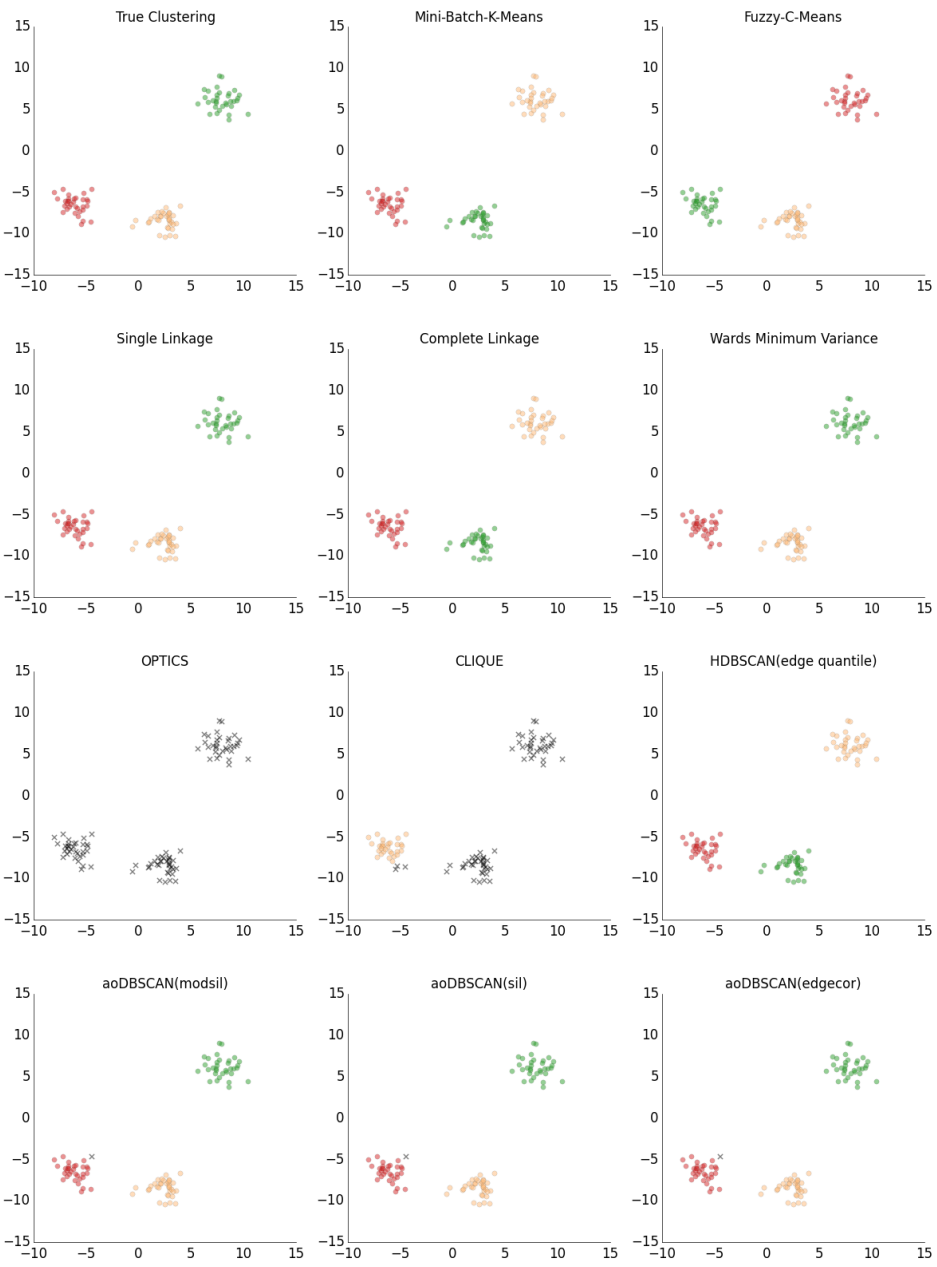


Figure A.3: Clustering results of the Blobs-1000D dataset.

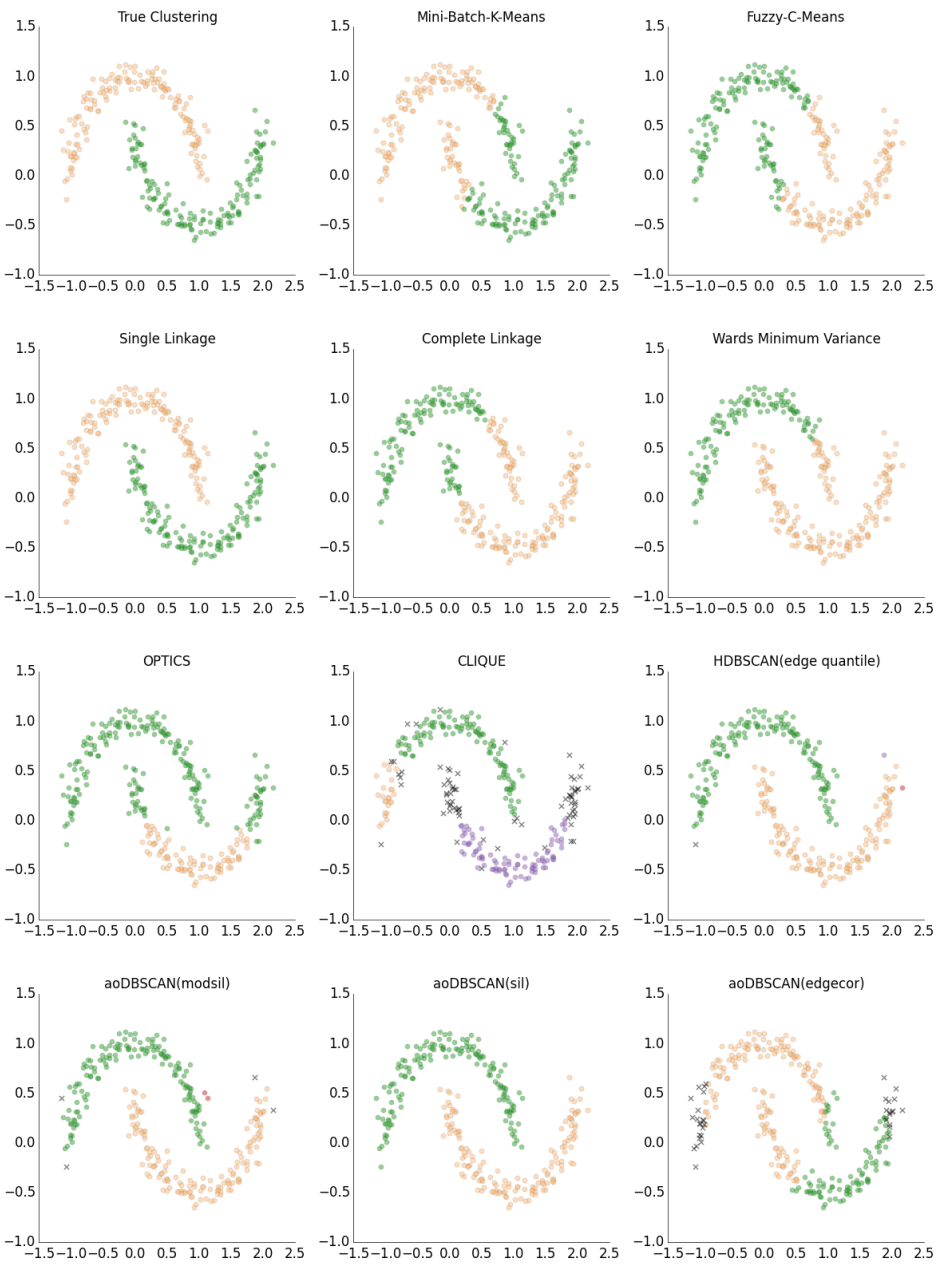


Figure A.4: Clustering results of the Moons dataset.

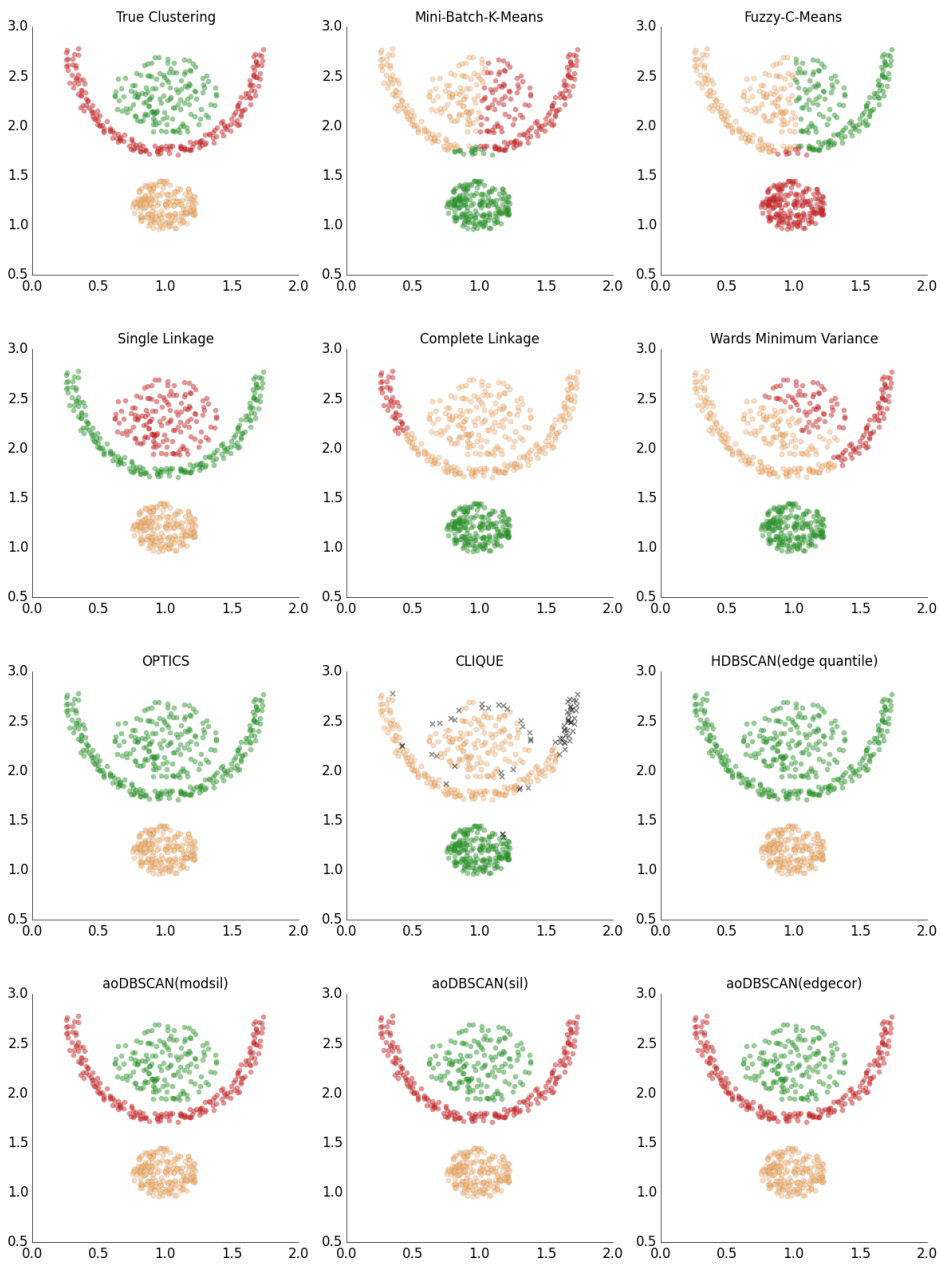


Figure A.5: Clustering results of the BlobsMoon dataset.

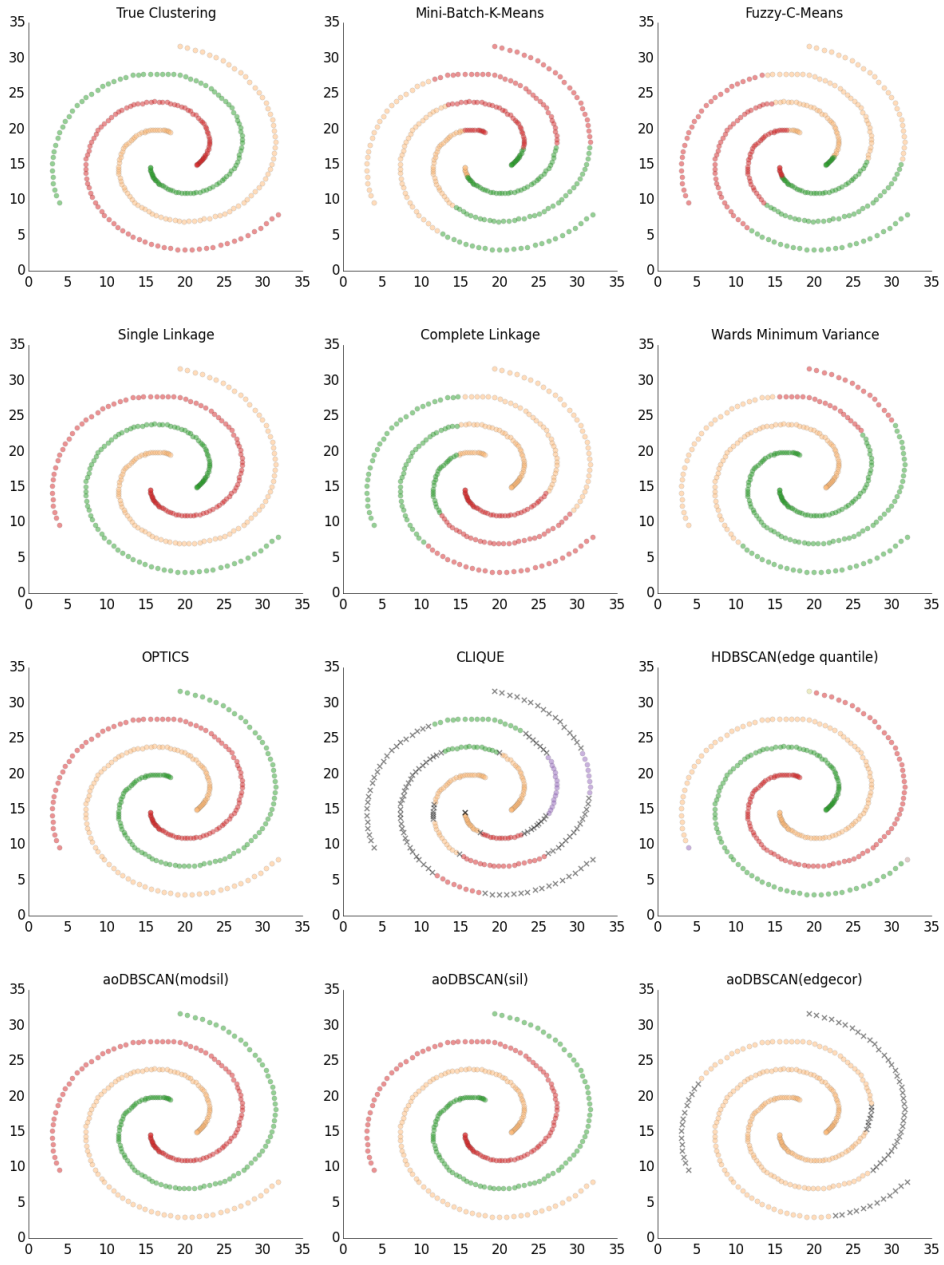


Figure A.6: Clustering results of the Spiral dataset.

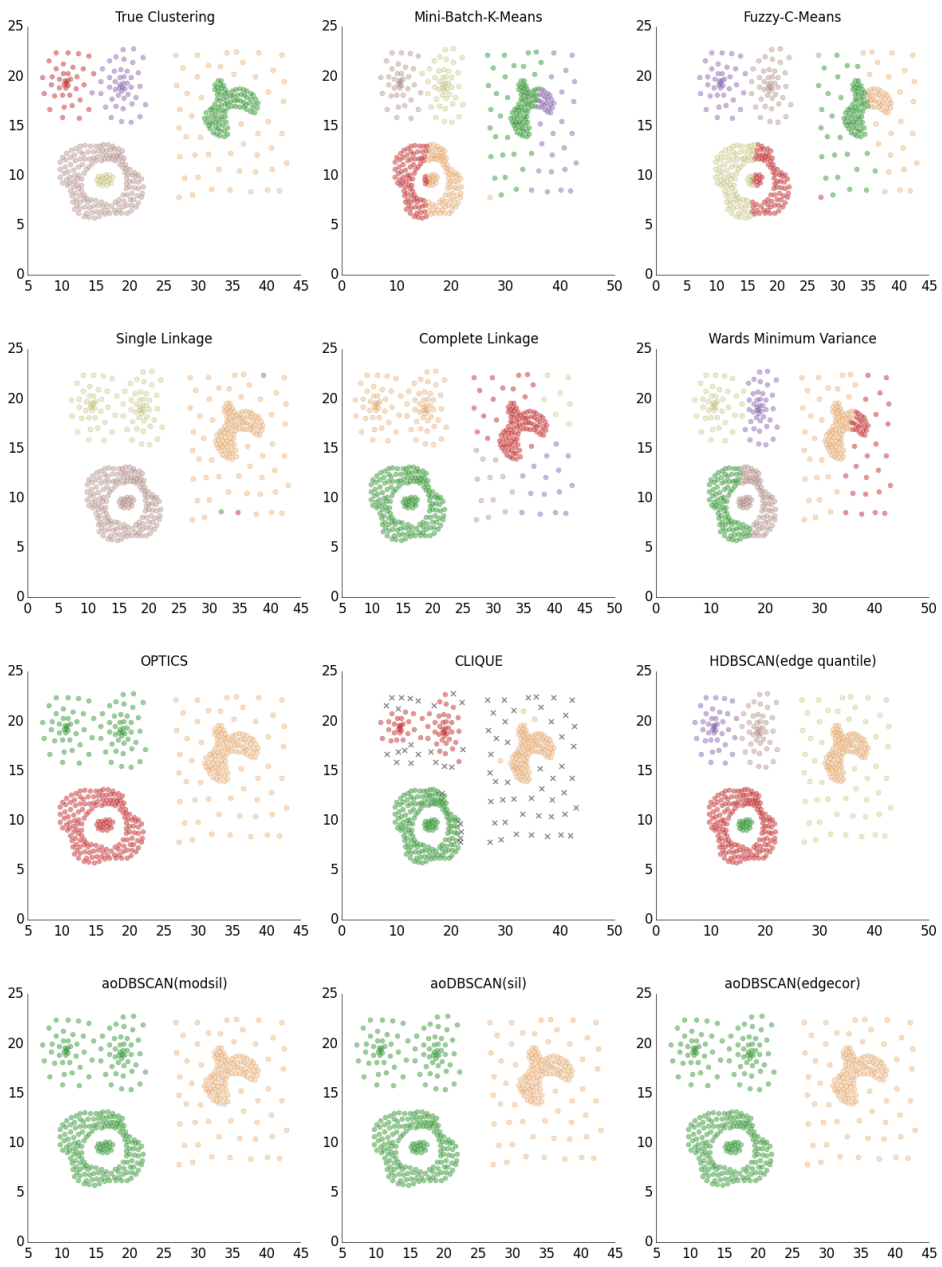


Figure A.7: Clustering results of the Compound dataset.

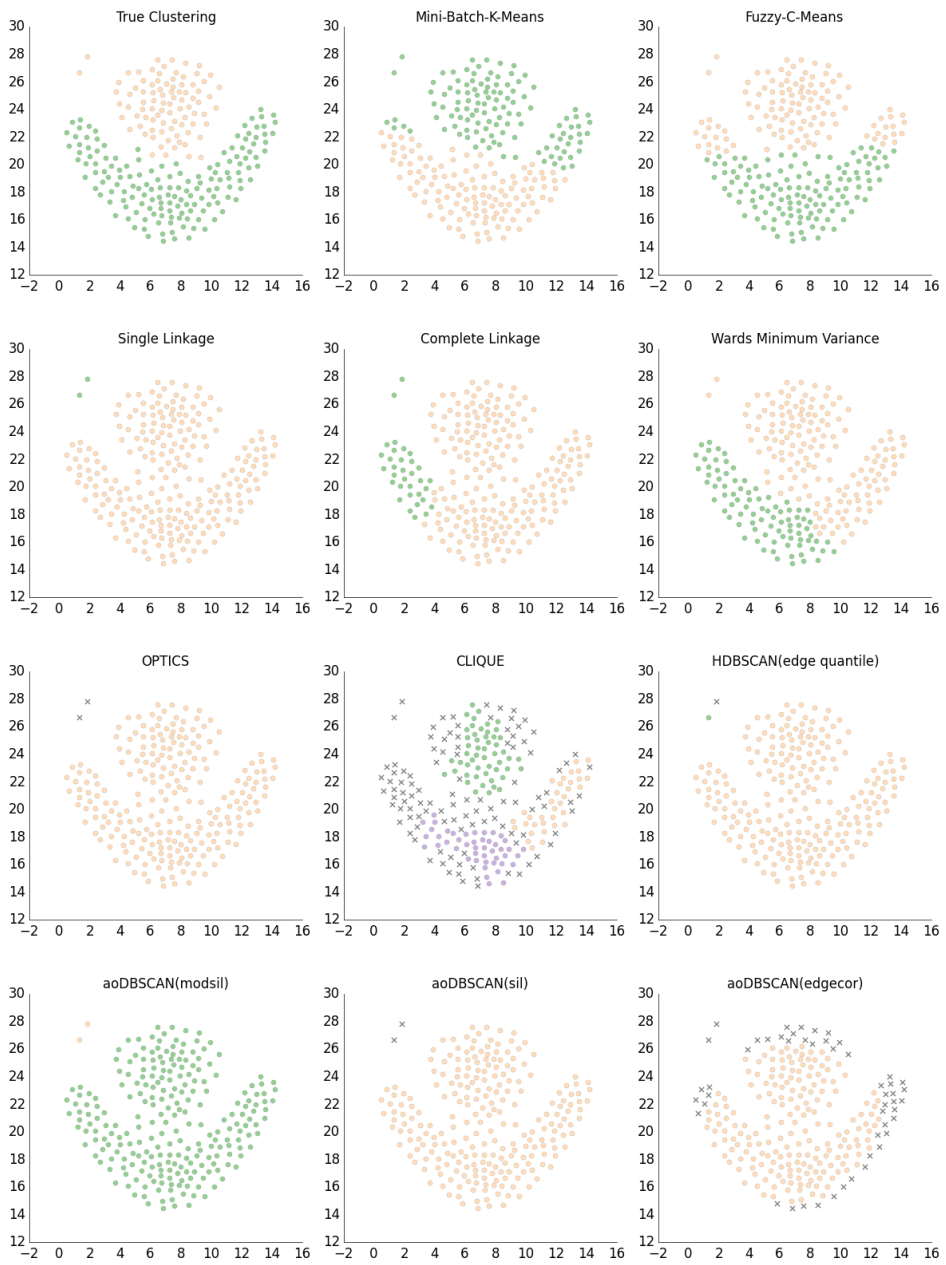


Figure A.8: Clustering results of the Flame dataset.

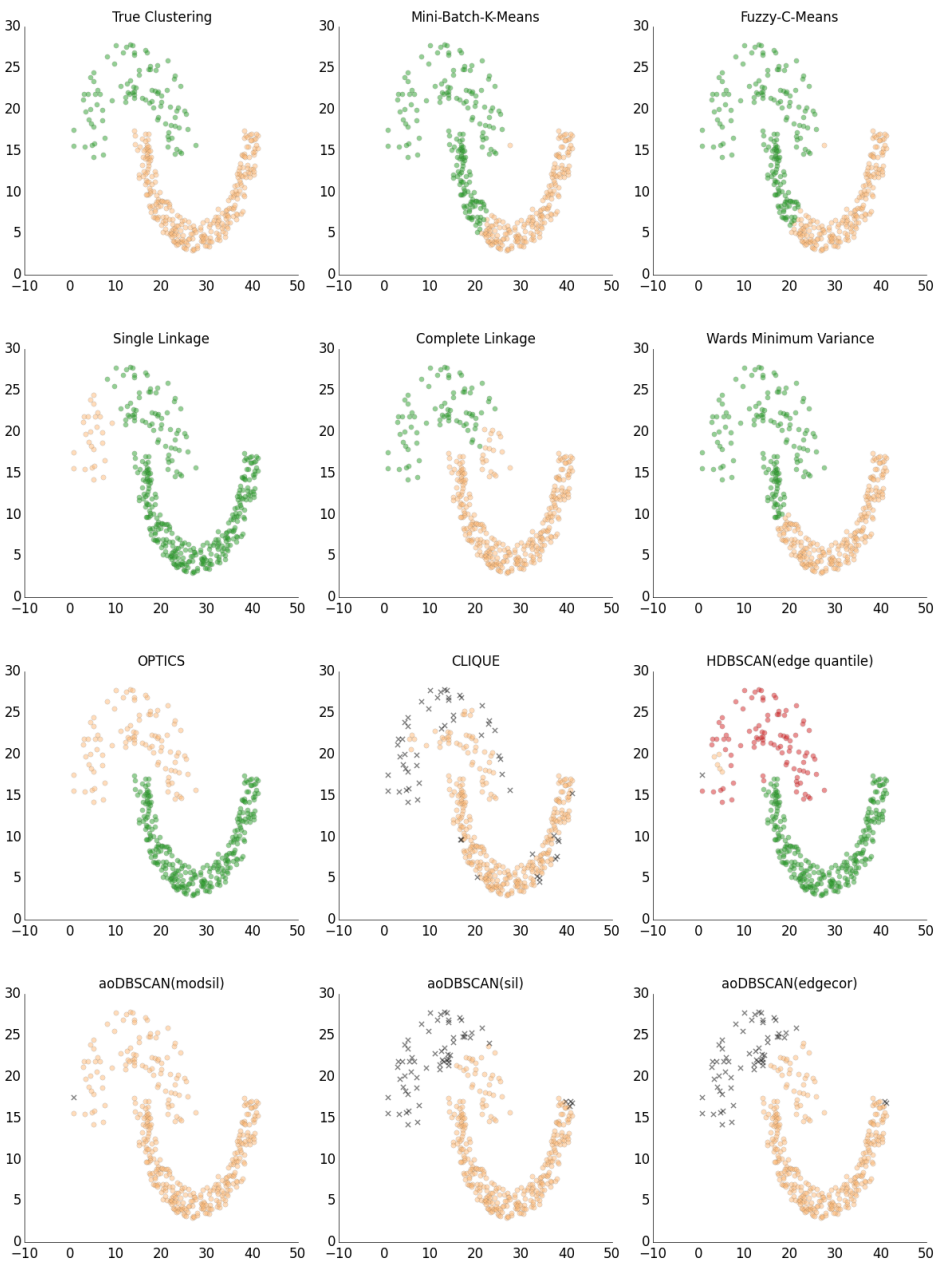


Figure A.9: Clustering results of the Moons-DifDense dataset

B

Abbreviations and Notations

Dataset and clustering acronyms

Acronym	Meaning
$\mathbf{X}$	dataset
x_i	i'th point of the dataset $\mathbf{X}$
x_{ij}	j'th attribute of point x_i
N	number of points in the dataset $\mathbf{X}$
$\mathbf{P}$	true partition of dataset $\mathbf{X}$
$\mathcal{C}$	clustering
C_i	i'th cluster of clustering $\mathcal{C}$
c_i	center of cluster C_i
K	number of clusters in clustering $\mathcal{C}$
$\mathcal{H}$	hierarchy of clusterings
Q	number of hierarchy levels
U	membership matrix
u_{ij}	membership degree of x_i and C_j

DBSCAN acronyms

Acronym	Meaning
m_{Pts}	minimal number of points to fulfill the core-condition
ϵ	radius of the neighborhood set
$N_\epsilon(p)$	neighborhood set of p with radius ϵ
$cores_{\epsilon,m_{\text{Pts}}}$	set if cores depending on ϵ and m_{Pts}
$d(x_i,x_j)$	distance of points x_i and x_j
$d_{core,m_{\text{Pts}}}$	core distance regarding m_{Pts}
$d_{reach,m_{\text{Pts}}}$	reachability distance regarding m_{Pts}

Validation measure acronyms

Acronym	Meaning
SSE	sum of squared errors
E	entropy of true partition **P** and clustering $\mathcal{C}$
P	purity of true partition **P** and clustering $\mathcal{C}$
F	f-measure score of true partition **P** and clustering $\mathcal{C}$
MIC	mutual information score of true partition **P** and clustering $\mathcal{C}$
Hom	homogeneity of true partition **P** and clustering $\mathcal{C}$
Com	completeness of true partition **P** and clustering $\mathcal{C}$
V	v-measure score of true partition **P** and clustering $\mathcal{C}$
$s_{\mathcal{C}}$	silhouette coefficient of $\mathcal{C}$
$s^d_{\mathcal{C}}$	density-based silhouette coefficient of $\mathcal{C}$
D	Dunn's index of $\mathcal{C}$
$CVNN$	Clustering Validation Index based on nearest neighbors of $\mathcal{C}$
$\rho_{\mathcal{C}}$	edge correlation score of $\mathcal{C}$

C

List of Figures

D

List of Tables

E

List of Algorithms

F

Bibliography

[AGGARWAL und REDDY 2013] C. C. Aggarwal und C. K. Reddy. **Data clustering: algorithms and applications**. CRC Press, Boca Raton, FL, 2013.

[AGRAWAL et al. 1998] R. Agrawal, J. Gehrke, D. Gunopulos und P. Raghavan. **Automatic subspace clustering of high dimensional data for data mining applications**. ACM SIGMOD Record, Vol. 27(2):94–105, 1998.

[AGRAWAL und SRIKANT 1994] R. Agrawal und R. Srikant. **Fast Algorithms for Mining Association Rules in Large Databases**. Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94), pp. 487–499, 1994.

[ANKERST et al. 1999] M. Ankerst, M. M. Breunig, H.-P. Kriegel und J. Sander. **OPTICS: Ordering Points To Identify the Clustering Structure**. ACM SIGMOD Record, Vol. 28(2):49–60, 1999.

[BANSAL et al. 2004] N. Bansal, A. Blum und S. Chawla. **Correlation Clustering**. Machine Learning, Vol. 56(1-3):89–113, 2004.

[BELLAMY et al. 1991] J. Bellamy, T. M. Cover, J. a. Thomas und R. L. Freeman. **Elements of Information Theory Telecommunication Transmission Handbook**. John Wiley and Sons, Inc., 3 Edn., 1991.

[BERKHIN 2010] P. Berkhin. **A Survey of Clustering Data Mining Techniques**. In: Grouping Multidimensional Data, pp. 25–71. 2010. Springer-Verlag, Berlin.

[BERTHOLD et al. 2010] M. R. Berthold, C. Borgelt, F. Höppner und F. Klawonn. **Guide to Intelligent Data Analysis**. Texts in Computer Science. Springer London, London, 2010.

[BEZDEK et al. 1984] J. C. Bezdek, R. Ehrlich und W. Full. **FCM: The fuzzy c-means clustering algorithm**. Computers & Geosciences, Vol. 10(2–3):191–203, 1984.

[BONDER et al. 2012] M. J. Bonder, S. Abeln, E. Zaura und B. W. Brandt. **Comparing clustering and pre-processing in taxonomy analysis**. Bioinformatics, Vol. 28(22):2891–2897, 2012.

[CALINSKI und HARABASZ 1974] T. Calinski und J. Harabasz. **A dendrite method for cluster analysis**. Communications in Statistics - Theory and Methods, Vol. 3(1):1–27, 1974.

[CAMPELLO et al. 2013] R. Campello, D. Moulavi und J. Sander. **Density-based clustering based on hierarchical density estimates**. Advances in Knowledge Discovery . . . , pp. 160–172, 2013.

[DOCKHORN et al. 2015] A. Dockhorn, C. Braune und R. Kruse. **An Alternating Optimization Approach based on Hierarchical Adaptations of DBSCAN**. In: 2015 IEEE Symposium Series on Computational Intelligence (SSCI), 2015, 2.

[DUNN 1974] J. C. Dunn. **Well-Separated Clusters and Optimal Fuzzy Partitions**. Journal of Cybernetics, Vol. 4(1):95–104, 1974.

[ESTER et al. 1996] M. Ester, H. P. Kriegel, J. Sander und X. Xu. **A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise**. Second International Conference on Knowledge Discovery and Data Mining, pp. 226–231, 1996.

[FINCH 2005] H. Finch. **Comparison of Distance Measures in Cluster Analysis with Dichotomous Data**. Journal of Data Science, Vol. 3:85–100, 2005.

[FISHER 1936] R. Fisher. **The Use of Multiple Measurements in Taxonomic Problems**. Annals of Eugenics, Vol. 7(2):179–188, 1936.

[GOWER und ROSS 1969] J. C. Gower und G. J. S. Ross. **Minimum Spanning Trees and Single Linkage Cluster Analysis**. Journal of the Royal Statistical Society. Series C (Applied Statistics), Vol. 18(1):54–64, 1969.

[GRIGSBY] **Optics Implementation**. `http://geog.ucsb.edu/~shane/optics/index.html`. Accessed September 2015.

[HALKIDI et al. 2002] M. Halkidi, Y. Batistakis und M. Vazirgiannis. **Cluster validity methods**. ACM SIGMOD Record, Vol. 31(2):40, 2002.

[HANDL et al. 2005] J. Handl, J. Knowles und D. B. Kell. **Computational cluster validation in post-genomic data analysis**. Bioinformatics, Vol. 21(15):3201–3212, 2005.

[HINNEBURG und KEIM 1998] A. Hinneburg und D. Keim. **An efficient approach to clustering in large multimedia databases with noise**. In Proceedings of 4th International Conference in Knowledge Discovery and Data Mining (KDD 98), pp. 58–65, 1998.

[JAIN und DUBES 1988] A. K. Jain und R. C. Dubes. **Algorithms for Clustering Data**. Prentice-Hall, Inc., Upper Saddle Ridder, New Jersey, 1988.

[JONES et al.] **{SciPy}: Open source scientific tools for Python**. `http://www.scipy.org/`. Accessed September 2015.

[KIM et al. 2012] J. Kim, S. Seo, D. Jung, J. S. Kim und J. Huh. **Parameter-aware I/O management for solid state disks (SSDs)**. IEEE Transactions on Computers, Vol. 61(5):636–649, 2012.

[KRUSKAL 1956] J. B. Kruskal. **On the shortest spanning subtree of a graph and the traveling salesman problem**. Proceedings of the American Mathematical Society, Vol. 7(1):48–48, 1956.

[LIU et al. 2013] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu und S. Wu. **Clustering Validation Measures**. IEEE Transactions on Cybernetics, Vol. 43(3):982–993, 2013.

[MACQUEEN 1967] J. B. MacQueen. **Kmeans Some Methods for classification and Analysis of Multivariate Observations**. 5th Berkeley Symposium on Mathematical Statistics and Probability 1967, Vol. 1(233):281–297, 1967.

[MAULIK und BANDYOPADHYAY 2002] U. Maulik und S. Bandyopadhyay. **Performance evaluation of some clustering algorithms and validity indices**. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24(12):1650–1654, 2002.

[PEDREGOSA et al. 2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot und E. Duchesnay. **Scikit-learn : Machine Learning in Python**. Journal of Machine Learning Research, Vol. 12(Oct):2825–2830, 2011.

[RIBICHINI 2007] A. Ribichini. **Streaming Algorithms for Graph Problems**. Ph.D. thesis, Sapienza University of Rome, 2007.

[ROSENBERG und HIRSCHBERG 2007] A. Rosenberg und J. Hirschberg. **V-measure: A conditional entropy-based external cluster evaluation measure**. Computational Linguistics, Vol. 1(June):410–420, 2007.

[ROUSSEEUW 1987] P. J. Rousseeuw. **Silhouettes: A graphical aid to the interpretation and validation of cluster analysis**. Journal of Computational and Applied Mathematics, Vol. 20:53–65, 1987.

[SHEARER et al. 2000] C. Shearer, H. J. Watson, D. G. Grecich, L. Moss, S. Adelman, K. Hammer und S. a. Herdlein. **The CRIS-DM model: The New Blueprint for Data Mining**. Journal of Data Warehousing14, Vol. 5(4):13–22, 2000.

[TAN et al. 2005] P.-N. Tan, M. Steinbach und V. Kumar. **Chap 8 : Cluster Analysis: Basic Concepts and Algorithms**. Introduction to Data Mining, p. Chapter 8, 2005.

[WARD 1963] J. H. Ward. **Hierarchical Grouping to Optimize an Objective Function**. Journal of the American Statistical Association, Vol. 58(301):236–244, 1963.

[WARNER et al.] **scikit-fuzzy: A fuzzy logic toolkit for SciPy**. `https://github.com/scikit-fuzzy/scikit-fuzzy`. Accessed September 2015.

[WOLPERT 1995] D. Wolpert. **No free lunch theorems for search**. Most, pp. 1–38, 1995.

[XU und WUNSCH II 2005] R. Xu und D. Wunsch II. **Survey of Clustering Algorithms**. IEEE Transactions on Neural Networks, Vol. 16(3):645–678, 2005.

[ZAHN 1971] C. Zahn. **Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters**. IEEE Transactions on Computers, Vol. C-20(1):68–86, 1971.

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Masterarbeit selbstständig, nur unter Zuhilfenahme der aufgeführten Quellen und Hilfsmittel, verfasst zu haben.

Die Arbeit wurde weder einer anderen Prüfungsbehörde vorgelegt noch veröffentlicht.

Datum: .....

(Unterschrift)