Skeleton-based Validation for Density-based Clustering

DISSERTATION

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von Christian Braune, M.Sc.

geb. am 10.08.1979 in Hamburg

Gutachterinnen/Gutachter

Prof. Dr. Rudolf Kruse
Prof. Dr. Marie-Jeanne Lesot
Prof. Dr. Richard Weber

Magdeburg, den 03.09.2018

Braune, Christian:
Skeleton-based Validation for Density-based Clustering

Dissertation, Otto von Guericke University
Magdeburg, 2018.

Skeleton-based Validation for Density-based Clustering

Abstract

Clustering is an important process in data analysis. It is the process of grouping previously unlabeled data and distinguishing noise or outliers from interesting data. Clustering algorithms can work in many different ways. These are, for example, centroid-based methods like k-means, hierarchical clustering or density-based methods. DBSCAN is the best-known representative of the latter. Since clustering is an unsupervised learning approach, validation measures are needed to assess whether a found result is good or not. In the case of centroid-based algorithms there exists a plethora of validation measures for the crisp and the fuzzy case. Only recently the validation of density-based clustering has made some progress. The majority of the hitherto used validation measures refer in some way or another to the centroids of the clusters. When using density-based clustering, these centroids to not exist or have no meaning. Therefore, these measures are only help- or meaningful under a limited scope of scenarios. They would favor clusterings that—to the human eye—are obviously sub-optimal.

This thesis suggest a way to make centroid-based cluster validation measures available for clusterings obtained by a density-based algorithm. For this the arithmetic mean as centroid is replaced by a cluster skeleton that provides more structural information for a cluster than a single point. It can then be used instead of the former centroid in the calculation of the validation scores. After discussing several different techniques of finding such an object, the behavior of the different validation scores is analyzed.

Zusammenfassung

Clustering ist ein wichtiger Prozess in der Datenanalyse. Innerhalb dieses Prozesses werden ungelabelte Datenpunkte zu sogenannten Clustern zusammengefasst. Einige Clusteringalgorithmen sind außerdem in der Lage, zwischen Rauschen, Ausreißen und interessanten Datenpunkten zu unterscheiden. Hierfür arbeiten Clusteringalgorithmen auf vielfältige Art und Weise. So gibt es zentroidbasierte Verfahren wie k-means, Hierarchisch-Agglomeratives Clustering oder dichtebasierte Methoden wie DBSCAN.

Clustering ist im Wesentlichen ein unüberwachtes Lernverfahren und bedarf daher einer sorgfältigen Validierung der Ergebnisse. Für die zentroidbasierten Verfahren gibt es bereits eine Vielzahl verschiedener Maße, die jeweils leicht andere Definitionen eines Clusters implizieren. Diese Maße erlauben es einzuschätzen, wie gut ein jeweils gefundenes Ergebnis mit dieser Clusterdefinition übereinstimmt. Viele dieser Maße können für hartes Clustering ebenso angewendet werden wie für fuzzy Clustering.

Für die Validierung dichtebasierter Clusteringverfahren gibt es jedoch erst seit kurzem erste Verfahren. Dies liegt unter anderem daran, dass sich die zentroidbasierten Validierungsmaße nicht ohne weiteres auf dichtebasierte Clusterings übertragen lassen, da der vielfach verwendete Mittelpunkt eines Clusters im Kontext des dichtebasierten Clusterings von geringerer Signifikanz ist. Würden diese Maße zur Bewertung dichtebasierter Clusterings herangezogen, so würden sie Ergebnisse bevorzugen, die – zumindest für das menschliche Auge – suboptimal sind.

In dieser Dissertation wird eine Möglichkeit vorgeschlagen, wie zentroidbasierte Clustervalidierungmaße auf dichtebasierte Clusterings angewandt werden können. Hierzu wird der klassischerweise verwendete Mittelpunkt eines Clusters durch ein Clusterskelett ersetzt. Es kann anstelle des Mittelpunktes für die Berechnung der einzelnen Validierungsmaßes genutzt werden. Nach einer Diskussion verschiedener Methoden zum Finden eines solchen Skeletts wird das Verhalten verschiedenen Validierungsmaße analysiert.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Data analysis has become a research area in its own right and revolves around topics such as statistics, classification, regression, data preprocessing, feature selection, and clustering [89]. Many of these methods are also used in machine learning, which in itself can be divided into supervised and unsupervised learning—although this is only a rough partitioning. Clustering algorithms fall into the domain of data analysis as well as they belong to the unsupervised learning methods.

Clustering algorithms aim at learning to distinguish different (kinds of) structures. The process is strictly hypothesis-driven in the sense that the choice of the clustering algorithm and its inherent hypotheses of cluster structure limits the search space and the types of cluster-shapes that can be found. For this these algorithms analyze the spatial relationships of different data points and categorize data into a previously unknown number of groups (although the desired number of groups might be part of the hypothesis used in the clustering process, cf. Section 1.3).

In some areas of neuro-science—for example—so-called spike trains (recordings of the firing behavior of individual neurons) are analyzed. Spike trains which exhibit similar, synchronous behavior are called ensembles. The data analysis task usually practiced here follows the scheme: Are there any neurons that exhibit synchronous firing activity? If so, in which ensemble do they belong? This is often enough treated as a classification task (e. g. [26, 32]) or as an association analysis (e. g. [22, 23, 33, 57]), despite

the fact that it could also be treated as a clustering problem as well (e. g. [21, 22, 27–29, 31]). A slightly different approach is used in [56]. Here a dataset is compressed by using different clustering algorithms to improve the runtime of a support vector machine classifier. In [46] the algorithms are used to cluster features of a dataset to reduce the overall dimensionality while in [103] clustering algorithms are used to create *codebooks* of images to analyze the behavior of different classification algorithms.

That clustering is a ubiquitous problem in natural sciences becomes clear if one takes a look at the study performed in e. g. [101]. Cell images taken from [39, 96] are analyzed in this study (a representative sample can be found in Figure 1.1). To get the right proportion of cells that are either in mitosis or otherwise interesting an accurate count of the cells is necessary. This is an example for a clustering application with initially unknown number of clusters. Even if performed manually by humans the cell count varies by up to 11%. Since sometimes the number of images in such a dataset can be larger than 50000, manual annotation or even validation of the results is very much inefficient. Ideally one would use a clustering algorithm within an automated process that is able to automatically detect the number of clusters without feeding it any form of prior knowledge about the possible number of cells.

Even with a dataset at hand and a properly working clustering algorithm, it is still unclear what clusters should actually be found (or how many [120]). A famous example is presented in [74]: a simple deck of cards is used as dataset. The question posed is then: How many clusters does this set of cards contain? The answer depends on what we see as a cluster. If we consider cards of the same suit as cluster, then there are obviously four clusters. If cards of the same rank form a cluster, then there are eight or thirteen (depending on whether you use an English or a French deck). If we look at the cards themselves, we will find two clusters: Those cards, that contain an image and those that do not. Or are the clusters those that are formed by the red and the black colors? All of these results would form valid clusterings or labellings of the dataset (cf. next Section). This example might serve as an argument for the futility of clustering itself [74], but it should

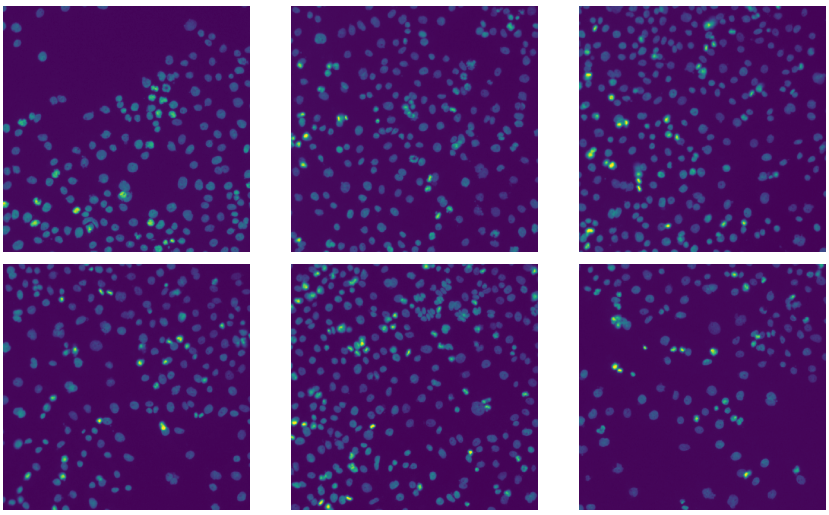


Figure 1.1: 6 different cell images showing colon cancer cells. An accurate count of the cells is needed for further studying the cells' activity.

rather serve as a reason for why clustering is an interesting problem and why the validation of clustering results is crucial for the data mining and data analysis process.

1.2 Fundamentals

Datasets come in various shapes and sizes. Often they are represented in a tabular format, in which each column represents an attribute (e. g. age, gender, height) and each row represents one datum. Attributes may be of different qualities, like categorical (or nominal), ordinal, or metric. Categorical attributes are in general non-orderable, like gender or hair color. Ordinal attributes may have a natural ordering (like perceived temperature; *cold*, *warm*, *hot*), or even be numeric (like *school grades*). However, the difference between numerical values has no meaning in itself except for its sign. Metric attributes encode numeric values such as *height* in meters, *temperature* in Kelvin or *mass* in kilograms. If all attributes are metric, each datum can be represented by a vector x with each component $x^{(i)}$

representing a single attribute. Such a vector $x \in \mathbb{R}^d$ is also called a data point with d attributes. The set of all data points available is referred to as $\mathcal{X} \subseteq \mathbb{R}^d$, with $n = \|\mathcal{X}\|$ being the cardinality of the dataset.

Similarity between different data points is measured either through similarity measures or by distance metrics. In the first case, points are said to be more similar to each other if the chosen similarity measure yields a higher value. In the second case, points are more similar to each other if the distance metric yields a lower value.

Definition 1.1 (*Metric*)

A metric is a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$ *which fulfills the following properties* $\forall x, y, z \in \mathcal{X}$*:*

1. $d(x, y) \geq 0$ *(non-negativity)*
2. $d(x, y) = 0 \Leftrightarrow x = y$ *(identity of indiscernibles)*
3. $d(x, y) = d(y, x)$ *(symmetry)*
4. $d(x, y) + d(y, z) \geq d(x, z)$ *(triangle inequality)*

In machine learning there are two categories for learning problems. Supervised learning is a form of learning problems where the dataset contains labeled data points. This means that there is a target attribute for which the relationship between all the other attributes should be learned. If these relationships have been properly learned, new and unlabeled data points can be *classified*, i.e. their respective target attribute's value can be predicted.

On the other hand, there are unsupervised learning problems. In these problems data points are usually unlabeled and class structure should be inferred from the spatial relations between different data points. Usually we use the distance or similarity of data points for this structural analysis. This process is called *clustering*. Clustering assigns labels to hitherto unlabeled data points to form groups (or *clusters*). Within these groups data points should be relatively more similar to either all or some members of the same group than to members of other groups and dissimilar to members of other groups.

Definition 1.2 (*Crisp Labeling*)

A (crisp) labeling of a dataset is a function $l : \mathcal{X} \to \{1, \ldots, m\}$*, i.e. each point* x *is assigned one out of* n *possible numeric labels.*

Definition 1.3 (*Cluster*)

A subset $C \subseteq \mathcal{X}$ is called a cluster if and only if $\forall x, y \in C : l(x) = l(y)$ holds. I. e., points belong to the same cluster C_i if and only if their label is equal.

Clusters are therefore formed by all the points that share the same label. Points with different labels belong to different clusters. To better access the index structure implied by such a labelling, a partition matrix can be formed:

Definition 1.4 (*Partition matrix*)

A crisp labeling of a dataset $\mathcal{X}$ into k clusters can also be represented by a partition matrix of the form $(u_{ij}) = U \in \{0, 1\}^{n \times k}$. If a point x_i belongs to cluster C_j, i.e. $l(x_i) = j$, then $u_{ij} = 1$ and $u_{ik} = 0$ for all $k \neq j$.

Sometimes data points cannot be reasonably assigned to a cluster. This might be the case for extreme outliers which are then considered as noise. If an algorithm is capable of detecting such noise points, by convention the additional label -1 is used for points belonging to no cluster. This ensures, that l is still surjective.

The same problem might occur for *inliers*, i. e., points that cannot be assigned unambiguously to a single cluster. In such cases fuzzy clustering can be used. Points now do not solely belong to a single cluster but spread their membership across different clusters to different degrees.

Definition 1.5 (*Fuzzy Labeling*)

A fuzzy labeling of a dataset $\mathcal{X}$ into c clusters is a vector-valued function $l : \mathcal{X} \to [0, 1]^c$ with the constraint $\sum_{i=1}^{c} l_i(x) = 1 \quad \forall x \in \mathcal{X}$. Each component l_i describes the membership of x to the cluster C_i.

If the constraint that all components of $l(x)$ have to sum up to 1 is omitted, we speak of possibilistic clustering.

Analogous to the crisp partition matrix, a fuzzy partition matrix can be defined.

Definition 1.6 (*Fuzzy Partition matrix*)

A fuzzy labeling of a dataset $\mathcal{X}$ into c clusters can also be represented by a fuzzy partition matrix of the form $(u_{ij}) = U \in [0, 1]^{n \times c}$ with column vectors $(u_{i\circ}) = l(x_i)$ (for a fuzzy labeling).

In probabilistic clustering each point belongs to exactly one cluster. The uncertainty about what that cluster might be, is expressed by the probabilisitc labeling. Numerically it is the same as a fuzzy labeling, but in fuzzy clustering each data point usually belongs to every cluster to a degree 0, 1, or some number in between that describes the degree of membership..

1.3 Clustering

Since neither the number of possible labels nor the shape of the clusters can usually be known in advance with absolute certainty, every labeling of the data might initially be possible. The number of ways of grouping points from an n-elemental set into k subsets can be described by a Sterling number of the second kind and is denoted as $S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$. The sum over all possible numbers of partitions is then the Bell number $B_n = \sum_{k=0}^{n} S(n,k)$, which grows at least exponentially fast [111].

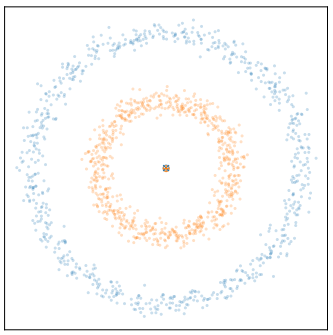


Figure 1.2: Two nested circles with their respective centroids. Clusters are indifferent when compared to the other cluster's representative.

Thus, the total number of possible clustering of a dataset of just 10 points is already $B_{10} = 115,975$ and for a dataset with 100 points it is already $B_{100} = 4.7585\ldots \times 10^{115}$. Datasets sometimes have several millions of data points.

As already described in Section 1.2, clustering is the process of finding groups of points. It is an unsupervised learning task in which a more or less consistent labeling of points should be found. While Figure 1.3 certainly shows two valid clusterings, the left one would probably never be the result of a well-designed clustering algorithm. In fact the labeling produced there is just random.

Different strategies have been developed over the last decades to find proper labelings of a given dataset. They all show different characteristics when it comes to how the clusters are found, which parameters are used and what constitutes a cluster in itself. Candidates from two of the most

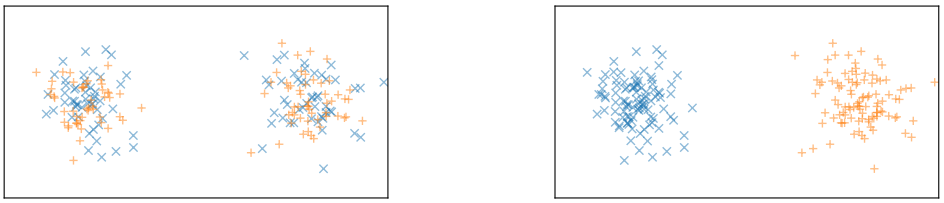


Figure 1.3: Two different labellings of the same dataset. Labels are randomly scattered in the left plot and intuitively assigned in the right plot. Though possible and a proper labelling, the left one does not produce natural cluster structure.

famous categories will be described in this section, explain the difference between crisp and fuzzy clustering and briefly summarize other clustering schemes for the sake of completeness.

1.3.1 Centroid-based Clustering

The family of centroid-based clustering methods encompasses all the (crisp) variants of the k-means algorithm [97]. The general procedure of this algorithms is shown in Algorithm 1. This algorithm will later also be used to determine the cluster skeleton which are the main focus of this thesis.

Formally the procedure described in Algorithm 1 optimizes the objective function

$$J(X, k; \mathcal{U}, \mathcal{M}) = \sum_{i=1}^{|X|} \sum_{j=1}^{k} u_{ij} \cdot d^2(x_i, \mu_j) \text{ w.r.t. } \sum_{j=1}^{k} u_{ij} = 1. \tag{1.1}$$

if d is the euclidean distance. Otherwise a different estimator for the centroid than the arithmetic mean is needed. By an alternating optimization scheme, one can derive the update formulas for the memberships stored in $\mathcal{U}$ and the centroids $\mu_j \in \mathcal{M}$. Since the centroids are recalculated as the arithmetic mean of all points they are representing, the only distance metric allowed here is the euclidean distance. For other distance measures, the arithmetic mean might not be an unbiased or consistent estimator.

Algorithm 1: Pseudocode of the k-means algorithm.

Require:
$\mathcal{X}$ set of n data points,
k number of clusters to find

1: **function** K-MEANS($\mathcal{X}$, k)
2: $\mathcal{M} = \{\mu_1, \ldots, \mu_k\} \leftarrow$ random but distinct points from $\mathcal{X}$
3: $\mathcal{U} \leftarrow \mathbf{0}_{x \times k}$ ▷ matrix with zeros only
4: **repeat**
5: **for all** $x_i \in \mathcal{X}$ **do**
6: $j \leftarrow \underset{l}{\operatorname{argmin}}\, d^2(x_i, \mu_l)$
7: $\mathcal{U}_{il} \leftarrow 1$ ▷ Membership is 1 for closest centroid
8: $\mathcal{U}_{il} \leftarrow 0$, for all $1 \leq l \leq k, l \neq j$ ▷ 0 otherwise
9: **for all** $\mu_j \in \mathcal{M}$ **do**
10: $C_j \leftarrow \{x_i \mid \mathcal{U}_{ij} = 1\}$
11: $\mu'_j \leftarrow \frac{\sum_{x \in C_j} x}{|C_j|}$ ▷ μ'_j is the center of all point assigned to μ_j.
12: $\mathcal{M} \leftarrow \{\mu'_1, \ldots, \mu'_k\}$
13: **until** convergence
14: **return** $\mathcal{M}$, $\mathcal{U}$

If for some reason one has to use the Manhattan distance instead of the euclidean distance, the centroid calculation in line 11 of Algorithm 1 has to be replaced by the component-wise median (as this is the appropriate estimator for the Manhattan distance).

In both aforementioned cases the clusters' centers might not be actual points in the dataset nor is guaranteed that any of its components actually exist in any data point. A clustering algorithm that solves this problem is the k-medoid algorithm. Here the point x_j is chosen as representative for a cluster if it minimizes the sum of within-cluster distances, i. e.,

$$\mu_j = \underset{x_j \in C_j}{\operatorname{argmin}} \left\{ \sum_{x \in C_j} d(x, x_j) \right\}, \tag{1.2}$$

where C_j is the set of all points that have been assigned to the cluster j in a previous step.

If we allow probabilistic membership degrees (which can be easily turned into a crisp labelling by assigning to each point the label that maximizes its membership to the respective cluster), gaussian mixture models can be seen as prototype-based clustering algorithms as well.

One of the major problems with prototype-based clustering is the proper choice of the number of clusters. Typically k is varied and so several different clusterings on the same dataset are generated. These clusterings are then compared with each other and the best choice is used as a labelling for the dataset (cf. Section 2.1.2). In contrast, density-based clustering usually chooses the number of clusters automatically. In the case of DBSCAN the number of clusters is equal to the number of equivalence classes given an algorithm-specific relation between points.

1.3.2 Crisp vs. Fuzzy Clustering

The previously described k-means algorithm suffers from two major drawbacks. Its usage of the arithmetic mean to establish cluster centers makes it susceptible to noise and outliers just like the arithmetic mean itself. This problem can be overcome by using a fuzzy variant of the k-means algorithm: fuzzy c-means [83]. And second, its crisp assignment of cluster labels may lead to misleading, non-deterministic or implausible results. Clusters are generated by finding the Voronoi cells [126] for the given cluster centers (cf. Figure 1.4) The Voronoi diagram visualizes the partitioning of space around a given set of points. It contains convex regions of points for each given point (here: cluster center) such that all points within the same region have that cluster center as their nearest neighbor. Each data point therefore belongs to the cluster center into whose cell it falls. However, in some cases points may lie on or near the intersection of two or more cells. In such cases a crisp assignment does not properly reflect the true ambiguity of the membership.

To cope with this problem the fuzzy-c-means algorithm [16] has been developed. Instead of assigning to each point the cluster's index it belong to (cf. Definition 1.2) the membership of a data point is inversely-proportional

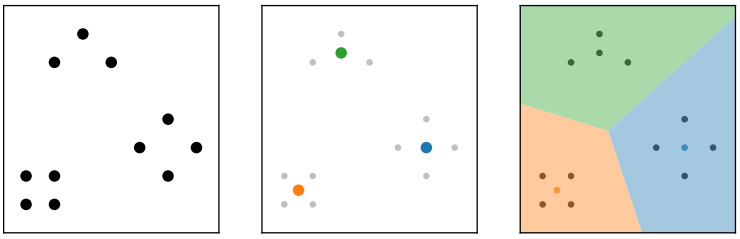


Figure 1.4: Voronoi diagram for the cluster centers of three simple clusters. Points are associated with the cluster center of the cell they fall into.

to their respective distance distributed across all cluster prototypes (see Equation 1.3).

$$\forall i, 1 \leq i \leq c : \forall j, 1 \leq j \leq n : u_{ij} = \frac{d_{ij}^{\frac{2}{1-\omega}}}{\sum_{k=1}^{c} d_{kj}^{\frac{2}{1-\omega}}} \tag{1.3}$$

Cluster centers are not anymore the arithmetic mean of their associated data points but rather the weighted mean, where the cluster membership acts as weight (see Equation 1.4).

$$\forall i, 1 \leq i \leq c : \mu_i = \frac{\sum_{j=1}^{n} u_{ij}^{\omega} \cdot x_j}{\sum_{j=1}^{n} u_{ij}^{\omega}} \tag{1.4}$$

Formally this optimizes the objective function

$$J(X, c; \mathcal{U}, \mathcal{M}) = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^{\omega} \cdot d^2(x_i, \mu_j), \text{ with } n = |\mathcal{X}|. \tag{1.5}$$

In all three formulas $\omega > 1, \omega \in \mathbb{R}$ acts as a fuzzifier. This constant determines the fuzziness of cluster memberships. Higher values lead to more smoothed out membership degrees and in the limit-case lead to a membership degree of $1/c$ for every point to every cluster. On the other hand, lower values lead to a crisper assignment of point (cf. Figure 1.5).

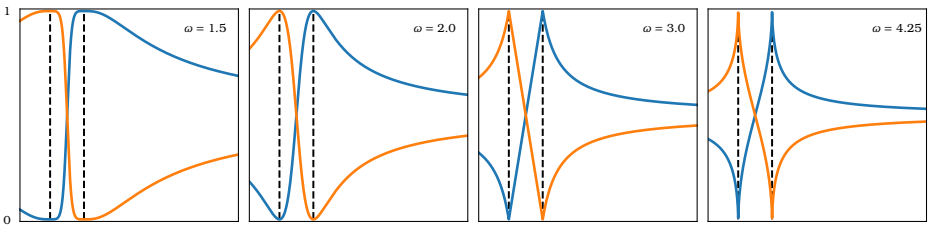


Figure 1.5: Development of membership degrees for two one-dimensional cluster centers (marked by vertical dashed lines) and different values for ω.

In general, fuzzy c-means tends to be more robust against outliers and noise than its crisp counterpart k-means due to the reduced influence that these points have on the calculation on the weighted mean [83].

As can be seen by comparison of Equation 1.5 and Equation 1.1, setting $\omega = 1$ leads to the crisp case of k-means. In that way, k-means might be seen as a special case of the fuzzy-c-means algorithm, or fuzzy-c-means as the generalization of k-means.

Both k-means and fuzzy-c-means still have the restriction of being only able to use the euclidean distance as metric. When we are not restricted to use metrics from the Minkowski family, but can rather use the Mahalanobis distance [98], we can derive the Gustafson-Kessel algorithm for clustering data [68]. This algorithm is similar to the k-means algorithm. During each iteration an additional covariance matrix per cluster is estimated which defines a cluster-specific metric (according to the Mahalanobis distance). The membership degrees are then calculated with respect to the individual within-cluster distances (i. e., for each point the covariance matrix for each cluster has to be considered). This has an advantage over the standard fuzzy-c-means algorithm. Clusters may have different shapes. Such clusters can be recognized and distinguished well, as long as their convex hulls do not overlap.

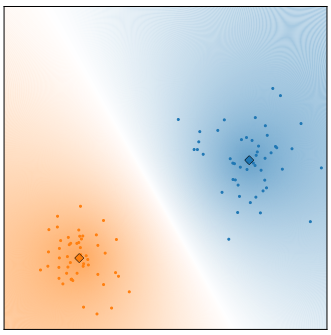


Figure 1.6: Membership contour plot for two cluster centers and $\omega = 2$.

However, a high dimensionality of the dataset can have detrimental effects on especially the fuzzy c-means algorithms as discussed in [129, 130].

In higher dimensional spaces the centers of clusters (calculated as weighted means of all data points) tend towards the center of gravity of the whole dataset. This happens since in high dimensions individual differences in distances tend to vanish or become non-significant. And if almost all points are almost equally far apart for each cluster center, the weighted mean has to be the mean of all data points (in the limit case).

1.3.3 Density-based Clustering

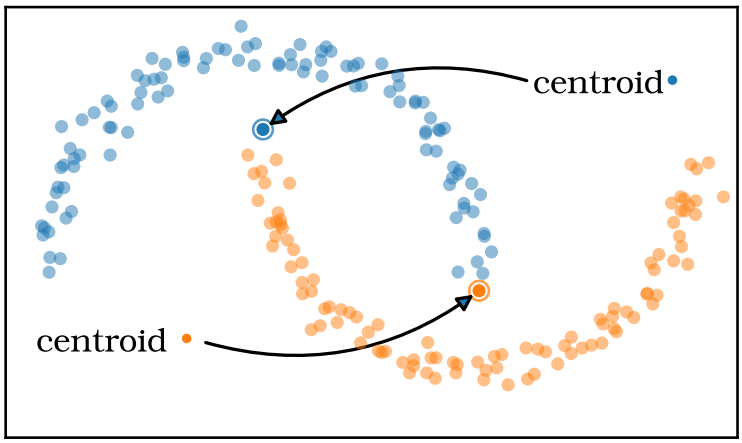


Figure 1.7: Two non-convex clusters with their respective centroids (calculated as the mean of all the clusters' points). The centroids do not lie within the vicinity of any of the cluster's points they are representing.

Density-based clustering does not suffer as much from this effect since only the closest point bear relevance for the clustering process. Additionally, for some datasets ordinary, prototype based clustering algorithms will not work properly. If the clusters' shapes are irregular, so that a single, simple point does not suffice to represent the cluster (as in Figure 1.2 or Figure 1.7), density-based clustering may be a solution.

Instead of parameterizing the algorithm with the desired number of clusters, these clustering algorithms require as parameters a notion of what is considered as *dense*. In the following a few selected density-based algorithms will be presented: DBSCAN, OPTICS and Black Hole Clustering.

These algorithms all use slightly different notions or ways of defining density or detecting dense regions. DBSCAN (and it successor HDBSCAN [38]) define density directly in the physical sense that a region is dense if it contains a minimum number of parts per volume. These two algorithms are without any doubt the most prominent examples for density-based clustering. E. g., in 2014 DBSCAN was awarded the *Test of Time Award* by the SIGKDD [115].

OPTICS inverses this by sorting points by their actual density (or the required parameter that would turn the region into a dense one) and Black Hole Clustering performs its labeling by transforming data into an observer space which is subsequently partitioned by only a single parameter.

DBSCAN

In the case of DBSCAN [55] these are ε, and *minPts*. These two parameters define a hyper-sphere and the minimal number of points required within such a sphere for its center point to become a cluster core. Following the original paper of Ester *et al.* an explanation of how clusters are formed within the DBSCAN algorithm is given by a series of definitions and relations beforehand.

Definition 1.7 (ε-*Neighborhood*)

The ε-neighborhood of a point $x \in \mathcal{X}$ is called $\mathcal{N}_\varepsilon(x)$ and is the set of all points $q \in \mathcal{X}$ that lie within a hypersphere of radius ε centered around x, i. e., $\mathcal{N}_\varepsilon(x) = \{q \in \mathcal{X} \mid d(x, q) \leq \varepsilon\}$.

Definition 1.8 (*Core Point*)

A point x is a core point if and only if the size of its ε-neighborhood is at least minPts, i. e., $\|\mathcal{N}_\varepsilon(x)\| \geq$ minPts $\Leftrightarrow$ x is a core point.

Definition 1.9 (*Directly Density Reachable*)

A point x is directly density reachable from a point q if and only if q is a core point and $x \in \mathcal{N}_\varepsilon(q)$.

Definition 1.10 (*Density Reachable*)

A point x is density reachable from a point q if and only if there exists a series of points $q_1, \ldots q_n$ with $q_1 = q$ and $q_n = x$ such that for any $i, i + 1$ holds that q_{i+1} is directly density reachable from q_i.

Note that here $q_1 \ldots, q_{n-1}$ are also required to be core points (by Definition 1.9).

Definition 1.11 (*Density Connected*)

A point x is density connected to a point y if and only if there exists a point q such that both x and y are density reachable from q.

Note that in this definition neither x nor y need to be core points.

With these definitions at hand, finding clusters becomes simply finding the transitive closure of the *density connected* relation.

Definition 1.12 (*Cluster in* DBSCAN)

The set of clusters found by DBSCAN *with respect to its parameters ε, and minPts is the transitive closure of the density-connected relation.*

Note that the clusters can be be labeled in any arbitrary order, depending on which point we start building the closure from. The definition also allows for some (non-core) point x to belong to more than one cluster in case it is density reachable from different core points which in return need not be density connected (since x is not a core point in this case). Such border cases are usually algorithmically decided by the *first come, first serve* principle.

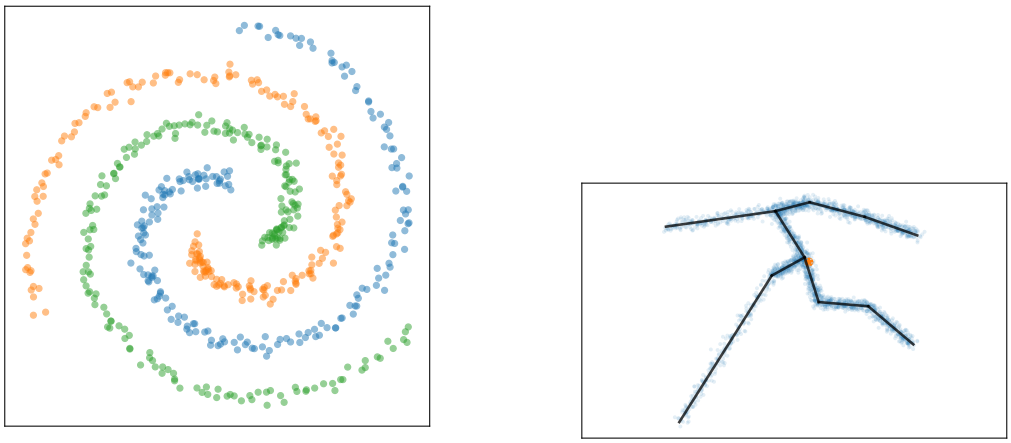


Figure 1.8: Examples of clusters for which density based clustering works better than centroid-based clustering

Definition 1.13 (*Noise in* DBSCAN)

Let $C = \{C_1, \ldots, C_m\}$ be the set of all clusters found by DBSCAN. *Then $\mathcal{X} \setminus \left(\bigcup_{i=1}^{m} C_i \right)$ is the set of noise points (i. e., all points that are not assigned to any cluster).*

Hierarchical extensions for DBSCAN have been proposed in [47, 48]. Monotonicity of the core point property in DBSCAN with respect to ε as well as *minPts* can be exploited to generate two distinct dendrograms by fixing one of the parameters and generating the dendrogram for the other. Non-constant cuts in the dendrogram can also be applied to generate cuts at different levels within one hierarchy, which allows to cluster datasets with varying density.

Figure 1.8 shows two examples of clusters that are typically better suted for density-based algorithms such as DBSCAN.

OPTICS

Another density-based clustering algorithm is OPTICS [6] which uses an inverted approach to DBSCAN. Its main difference lies in the role of the *minPts* parameter which here is used to determine the *core distance* for a given point x.

Definition 1.14 (*core distance*)

The core distance $cd(o)$ of a point o is the smallest ε for which the ε-neighborhood of o is large enough to make o a core point. I. e., $cd(o) = \inf_{\varepsilon} \{\|\mathcal{N}_\varepsilon(o)\| \geq minPts\}$.

With the help of the core distance a relative reachability distance can be defined (relative in the sense that it is relative to another point, here: o).

Definition 1.15 (*Reachability Distance*)

The reachability distance $rd_o(x)$ of a point x w.r.t. o is the maximum of either the core distance of x or the (euclidean) distance between x and o, i. e., $rd_o(x) = \max\{cd(x), d(x, o)\}$.

OPTICS now starts by taking a random point o and all of its ε neighbors. These $p \in \mathcal{N}_\varepsilon(o)$ are then ordered according to their reachability distance

and stored in a priority queue. The first point in the output of OPTICS is the randomly chosen point o while the next points are taken from the priority queue. Whenever a point is taken from this queue, all of its neighbors enter the queue. All point within the queue update their reachability distance with respect to the last taken point, if this improves their reachability distance.

This generates an ordering of the points from which the cluster structure can be extracted when plotting the ordering against the corresponding reachability distances.

Black Hole Clustering
Black-Hole Clustering [25] is another density-based clustering approach that is based on ordering (or sorting) points. By introducing $d + 1$ observers for $\mathcal{X} \subseteq \mathbb{R}^d$ placed on the edges of an equilateral polytope bounding the dataset and sorting the points iteratively according to the distances to these observers, the dataset can be split into smaller chunks until only the clusters remain. Each smaller chunk of data points is iteratively processed by the next observer. The advantage of this procedure is its very low runtime. Essentially it can find clusters (without the need to supply a number of clusters beforehand) in $O((d + 1) \cdot n \cdot \log n)$ which is at least a magnitude faster than any other clustering algorithm that automatically detects the number of clusters. It is—however—limited to only detect clusters with non-overlapping convex hulls. It can be considered a density-based clustering algorithm as it splits each sorted list of data points if two consecutive points are too far apart from each other (i. e., not dense enough anymore).

1.3.4 Other Clustering Algorithms

Besides density-based and prototypical clustering algorithms, there is a multitude of other clustering algorithms. Two methods should be mentioned that are extremely useful in applications.

Hierarchical Clustering If the number of clusters is unknown and hard to estimate and the data cannot be properly visualized, hierarchical clustering might be an option to use. Starting from singleton clusters (each point forms its own, individual cluster) a hierarchy of cluster merges is generated. For

each step in the hierarchy the two closest clusters are combined and form a new cluster. The distance between the clusters determines the height of that merge in a dendrogram.

Probabilistic Clustering As already mentioned in Section 1.3.1 clusters can be modeled as the result of a stochastic sampling from several different multivariate Gaussian distributions. In such a case the probability that a point was sampled from a specific distribution can be calculated by the inverse probability density function of that distribution. The iteration scheme is not significantly different from the k-means or the fuzzy-c-means scheme (which is why it will not be repeated here again). Instead of calculating the membership directly from the distance to the centroid, one needs to calculate a probability density estimate for each given point and the current cluster's parameters (note: these are also estimated in each iteration of the algorithm—just like the memberships and centers).

The most recent developments can always be found in the GfKI series on data analysis, machine learning and knowledge discovery [118].

1.4 Problem Definition

Choosing which labeling of the data points is the correct one depends not only on the algorithm used to find the labeling, but also on the (task-specific) understanding of what a cluster is. Cluster validation measures are measures that evaluate how much a given clustering corresponds with the idea of a cluster. While clustering algorithms can only find a certain subset of all the possible labelings (cf. Section 1.3), validation measures will only yield high scores for such clusterings that are close to their internal understanding of a cluster. If the inherent understanding of a cluster that is used by the validation measure does not line up with the implicit cluster definition of the clustering algorithm, an optimal clustering can hardly be found.

In addition to the already presented prototype-based clustering algorithms, there exist several (fuzzy) clustering algorithms, which do not use single points as cluster prototypes. A simple variation is the extension from points to hyperspheres as presented in [82]. In addition to the normal fuzzy

clustering approach, each prototype has a radius that is adapted during each iteration of the clustering process. Points that fall within such a hypersphere have a membership degree of 1.0 to that cluster. This results in a crispier partitioning of the data and thus in better validation results.

Many of the existing validation measures refer implicitly or explicitly to a cluster's center. This distinguished point (which is not necessarily a point belonging to the dataset) acts as a representative of the whole cluster. Some clustering algorithms extend the notion of a cluster's centroid even further. These algorithms are called fuzzy shell clustering algorithms [84] and use different shapes like circles [5], rectangles [72], or quadratic shells [87, 88] as prototype. These algorithms have been successfully applied in shape recognition on images. All of these algorithms have in common that they only implicitly change the shape of the prototype by defining alternative distance metrics. In case of the fuzzy-c-varieties clustering algorithm—for example—the distance between a point and any centroid is not euclidean anymore. Rather the distance is calculated by Equation 1.6:

$$d^2(x_j, c_i) = |x_j - c_i|^2 - \sum_{k=1}^{m} \left(x_j - c_i\right)^T e_{ik}, \tag{1.6}$$

where a cluster prototype is a tuple $C_i = (c_i, e_{i1}, \ldots, e_{im})$ and the different e_{ik} are pairwise orthogonal unit vectors spanning a subspace of the data space.

In the case of density-based clustering (cf. Section 1.3.3) one does not have such a centroid and thus can not and should not use most of the existing validation measures. While it would be possible to calculate a central point as the mean of all data points that are assigned to a cluster, such a centroid is not necessarily a good candidate for representing the cluster it is supposed to represent (see Figure 1.7). But even if the centroids are representing their respective clusters well, it might occur that the centroids do not allow to distinguish between clusters properly (see Figure 1.2). This leads directly to the first research question:

Q1 Can the arithmetic mean as a cluster representative be replaced by a more general structure?

If such a more general structure can be defined, one would still lack a proper way to find this structure (if its definition does not already imply

its calculation). The medial axis (cf. Section 3.2) can describe such a more general structure. Its definition implies that it may actually replace the center point of a cluster by a tree-like structure. However, its calculation in higher dimensions ($d \geq 3$) is neither simple nor can the resulting structure be guaranteed to be tree-like. This leads to the second research questions of this thesis:

Q2 How can the medial axis be extended onto higher dimensions and how can such a structure be efficiently calculated?

The latter part of this question is of special importance since validation takes place several times. Each algorithm's output would need to be validated and clustering algorithms suffer from almost cubic complexity in many cases.

When applying density-based clustering algorithms *noise* has also to be dealt with. In centroid-based clustering settings, noise is essentially ignored in the basic algorithms, as e. g. k-means simply assigns all the points to the clusters. Fuzzy clustering can deal with the problem of noise by assigning low membership values to such points. As will shown in Section 1.3.2, this only works well with a small workaround. Density-based clustering does not assign points to any cluster if they do not belong to it (according to the parametrization). These points might—however—still lie close to existing clusters. A validation measure that completely ignores the existence of noise points is certainly going to favor clusterings with very small clusters and a lot of noise. In such a case the (few) points assigned to a cluster would be represented well by whatever representative has been chosen, while the remaining points (which were assigned to the *noise cluster*) are completely ignored This over-simplifies the clustering problem and leads to wrong or inadequate results. This leads to the next research question:

Q3 How can noise points originating from the density-based clustering be dealt with in validation scores which ignore noise?

Finally some thoughts have to be given to the evaluation of the method itself. Of course one can think about many potential solutions to these problems. But at the end of the day the question which matters most is:

Q4 Does validation based on cluster skeletons actually lead to better results?

1.5 Structure of this Thesis

To answer the research question just posed existing validation measures are reviewed. (Chapter 2). Chapter 3 starts with an explanation of how the quality of the skeleton that will be generated is going to be validated. For this a modified version of the discrete Fréchet distance is used. In the second part of this chapter different methods of obtaining a cluster skeleton are presented and discussed. Chapter 4 starts with an evaluation of the previously presented methods, discusses their advantages and disadvantages w.r.t. the validation task and ends with a description if the evaluation setup and the results of the final experiments. At last, in Chapter 5 the thesis will be recapitulated, and the answers to the posed research questions will be summarized. Overall advantages but also problems with the proposed method are discussed here as well as open research questions that may lead to future work.

CHAPTER 2

Methods for Cluster Validation and Skeletonization

This chapter will give a brief state-of-the-art of different cluster validation techniques as well as different approaches to skeletonize datasets. The chosen validation measures give an idea of what different measures can achieve and also how that different optimization criteria might have to be met.

In Section 2.2 different methods of re-representing data are discussed. These are for once different skeletonization techniques that (usually) allow to represent a dataset by a thinned, piece-wise one-dimensional object: A skeleton. The presented algorithms cover the different categories of thinning and skeletonization methods available for image-based datasets.

Within this section, core sets are discussed as well. These allow a sparser representation of the dataset and might give insight into representation techniques for datasets.

2.1 Cluster Validation

Once the result of a clustering algorithm has been obtained, only in the rarest cases can one visually determine whether the results make sense or not. Visual inspection is, however, mostly limited to two- or three-dimensional datasets. If the performance of a clustering algorithm ought to be measured or the result's quality needs to be estimated, a way for validation or evaluation

is needed. Depending on the way a clustering has been computed, different measures may be suitable or not. These can be broadly categorized in two ways: external and internal cluster validation.

2.1.1 External Validation

External cluster validation generally uses external information that a clustering algorithm's result can be compared to. This may either be the result of another clustering algorithm, some kind of expert knowledge or simply a ground truth. The problem of externally validating a clustering result lies mainly in how such a measure is computed. It is therefore rather a computational problem.

All of the measures have in common that they can be used to compute a similarity between two clusterings from different algorithms and to benchmark algorithms against an existing ground truth. In the first case, the external validation computes the similarity between two clusterings. By counting the agreements and disagreements the general procedure can be compared to the computation of the Jaccard index [78]. In the second case, the ground truth and the algorithm's result can be compared by using an external validation measure. Which one is used depends on the use case. An overview over different distribution qualities of several external validation measures can be found in [131].

Set-based approaches can work on the label list and count the number of agreements and disagreements between two given labelings l_1 and l_2. Agreements here are usually understood as the number of times a pair of points, that lie in the same cluster in one labeling, also lie in the same cluster in the other labeling (i. e., $x, y \in \mathcal{X} : l_1(x) = l_1(y) \rightarrow l_2(x) = l_2(y)$). Or, that points which are not in the same cluster are placed in different clusters in both labelings (i. e., $x, y \in \mathcal{X} : l_1(x) \neq l_1(y) \rightarrow l_2(x) \neq l_2(y)$). For any pair x, y for which neither of the previous implications hold, it is said that the two labelings disagree. From the number of agreements and disagreements one can compute different statistics that are described in the following.

(Adjusted) Rand Index

The Rand Index (RI) [109] is defined as the ratio of the number of agreements between two labelings, and the total number of possible pairs (without ordering). Let n_{ij} denote the number of items that are shared by the clusters i under labeling l_1 and j under labeling l_2 respectively. If C_i^1 denotes the set of all points x for which $l_1(x) = i$ (or equivalently $C_i^1 = l_1^{-1}(i)$) and $C_j^2 = l_2^{-1}(j)$, then n_{ij} is can also be expressed by the cardinality of the set $C_i^1 \cap C_j^2$. For this the labelings do not need to be arranged such that cluster indices match for the same clusters. In fact the RI/ARI are completely independent of the order in which clusters are labeled (or the total number of clusters) since agreements are counted on a *per-pair* basis of points.

The Rand Index can now be computed for any number of clusters as:

$$RI = \frac{\sum_{i=1}^{l} n_{ii}}{\sum_{i=1}^{\max l_1} \sum_{j=1}^{\max l_2} n_{ij}}, \text{ where } l = \min\{\max l_1, \max l_2\}. \tag{2.1}$$

Since by randomly labeling the data one would expect to get at least some agreements purely by chance, the Adjusted Rand Index (ARI) [77] is defined as:

$$ARI = \frac{RI - E[RI]}{\max RI - E[RI]}. \tag{2.2}$$

The resulting index yields values between -1 and 1. In the case of the RI values around 0 indicate (nearly) complete randomness between the two labeling l_1 and l_2. In the case of the ARI such values indicate that the resulting labeling is as good as the expected value [125]. Higher values indicate a better agreement between the two labelings while lower, negative values rather indicate a disagreement between them.

Mutual Information

The Mutual Information (MI) [42] is the amount of information obtained about one random variable when observing the other. In terms of labelings it tells us how much can be known about the label of a data point x under

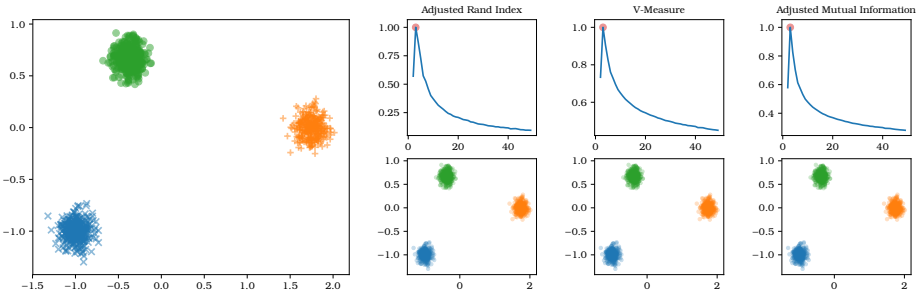


Figure 2.1: Original cluster structure shows three spherical clusters. All three shown external validation measures select the right number of clusters when used with k-means.

labeling l_2 once $l_1(x)$ is known. For discrete variables the Mutual Information is defined as:

$$MI = \sum_{i \in img(l_1)} \sum_{j \in img(l_2)} \frac{p(i,j)}{p(i) \cdot p(j)}, \tag{2.3}$$

where $p(i,j)$ is the joint distribution function of i and j, and $img(\cdot)$ is the image set of a function.

With the same notation as for the RI, an alternative formulation can be derived:

$$\begin{aligned} P(i) = \frac{n_{i\bullet}}{N}, P(j) = \frac{n_{\bullet j}}{N}, P(i,j) = \frac{n_{ij}}{N}, \\ MI = \sum_{i \in img(l_1)} \sum_{j \in img(l_2)} p(i,j) \log \frac{p(i,j)}{p(i) \cdot p(j)} \\ \text{with } n_{\bullet j} = \sum_i n_{ij} \text{ and } n_{i\bullet} = \sum_j n_{ij} \end{aligned} \tag{2.4}$$

which can equally be adjusted for chance to the Adjusted Mutual Information (AMI) as the RI:

$$AMI = \frac{MI - E[MI]}{\max MI - E[MI]}. \tag{2.5}$$

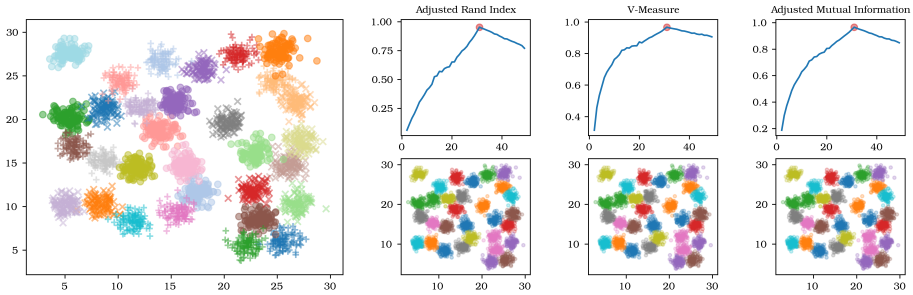


Figure 2.2: Original cluster structure shows 31 spherical and slightly overlapping clusters. All three shown external validation measures select the right number of clusters when used with k-means.

V-Measure

In [110] the V-Measure is defined as the harmonic mean between two different aspects of comparing to labelings:

Homogeneity: Only points that fall into the same cluster (under one of the labelings) have been placed into the same cluster (by the other labeling).

Completeness: All points that fall into the same cluster (under one of the labelings) have been placed into the same cluster (by the other labeling).

Homogeneity can be understood as an implication (i. e., *if a and b belong to the same cluster, they are placed in the same cluster*), while completeness measures the inverse direction (i. e., *if a and b have been placed in the same cluster, they belong to the same cluster*). In [12] it has been shown that the V-Measure and the Normalized Mutual Information (NMI) are identical when normalized by the sum of label entropies.

Human Assessment

Sometimes the number of clusters in a dataset is known it advance (e. g. if one wants to distinguish between sick and healthy patients, meaning $k = 2$). Sometimes the data analysis task requires a certain number of clusters (e. g. four different groups of customers). And sometimes the correct number of clusters that occur in a given dataset should be estimated. Beside the difficulties for estimating certain parameters for a clustering algorithm, this task can also be opened for humans to perform. Although in general slower than computers, humans excel at visual interpretation tasks. Simply

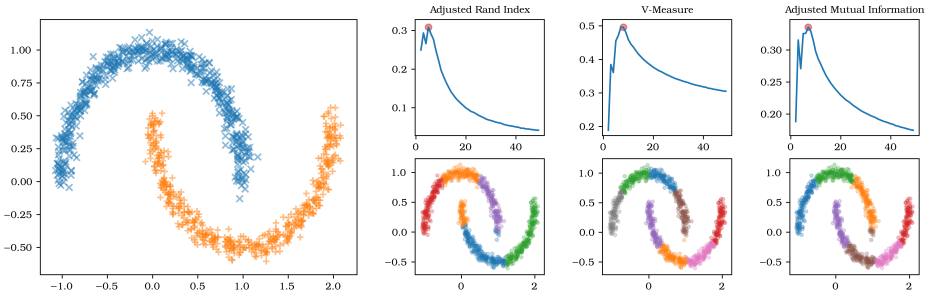


Figure 2.3: The original cluster structure shows two half-circle clusters. All three external validation measures fail at finding the right amount of clusters when used with k-means. This happens because the optimal clustering lies not in the range of possible results for the used clustering algorithm. The small peaks which can be seen for $k = 2$ for all measures do not correspond to a correct clustering but rather to one similar to that in Figure 2.7.

displaying the clustering result or the data itself is—however—infeasible in any interesting clustering scenario that goes beyond two-dimensional data. For such and similarly formulated tasks a complete family of visual techniques for cluster validation has been developed in the past years. It has to be noted though, that neither of the Visual Assessment of Cluster Tendency (VAT) methods presented in the following is strictly a validation technique in itself. It is rather a tool set to determine the number of clusters in a dataset if there are any at all.

All of the algorithms in the VAT family have in common that they create some form of graphical representation (the so-called Ordered Distance Image (ODI)) of the clustering result and allow the user to decide which results seems to be the most fitting. In case of the original VAT algorithm [17], the data is re-ordered according to an ordering of the points obtained by a Minimum Spanning Tree (MST) algorithm (Prim's [107] or Kruskal's [90] algorithm). The algorithm used for constructing an MST used in the different VAT algorithms uses a deterministic start: While the original MST may choose the initial node randomly the version used here always chooses one of those points that have the maximal distance. The rows and columns of the distance matrix are then re-ordered according to the sequential order in which the MST algorithm visits them and finally displayed as a gray-scale image (the so-called Ordered Distance Image (ODI)).

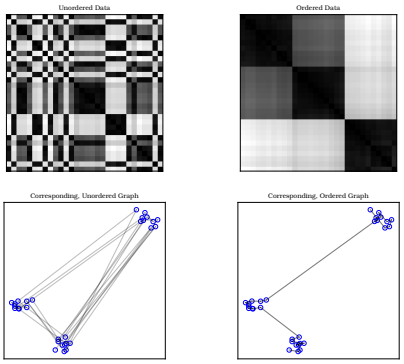
Extensions to this algorithm exist in the form of reVAT [76], which can be used for larger datasets and which uses profile graphs and sub-sampling; bigVAT [75] or scalable VAT [71] which both can be used for even larger datasets by using sub-sampling, co-VAT [18] which can be used in case of rectangular distance matrices for simultaneously finding row- and column-clusters; specVAT [127] where the re-ordering is based on the spectrum of the dataset's Laplacian matrix, and iVAT [128] which is an improved version of the original Visual Assessment of Cluster Tendency (VAT) algorithm that uses a path-based dissimilarity [59] instead of the Euclidean distance to improve the quality of the ODI.

The general process of the VAT algorithm can be seen in Figures 2.4a, 2.4b, 2.4c and 2.4d respectively. Especially Figure 2.4b shows an interesting problem of this approach. The eight clusters in the center of the dataset can appear as a single cluster in the ODI to one user, but another might recognize the individual clusters as darker squares along the principal diagonal inside the bigger square. Figure 2.4c shows, that for more complex cluster structures the VAT algorithm produces ODIs which do not allow to properly estimate the number of clusters. A similar effect can be seen in Figure 2.4d where one could derive that some kind of structure exists in the data, but a clear estimate such as in Figure 2.4a or Figure 2.4b is not evident.

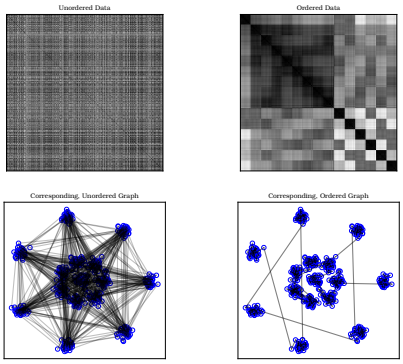
Another approach to involve humans into the clustering process has been described in [67] where the clustering results of several different algorithms are transformed into a *clustering space*. Within this space similar clustering results would be placed close to each other. Users can then choose different, dissimilar clusterings and decide which results best fit their purpose. An example of what such a visualization might look like can be found in Figure 2.5.

Summary

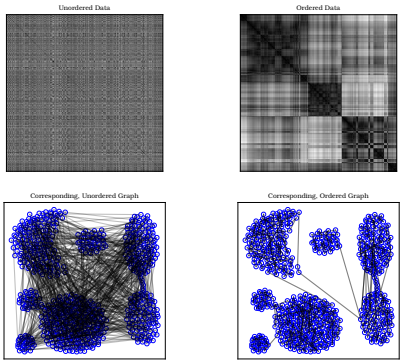
The selection process for the right number of clusters can be seen in Figure 2.1, Figure 2.2, and Figure 2.3. The process itself is fairly easy here, since the ground truth labeling was available to the clustering algorithm beforehand. Simply estimating the number of clusters in this scenario could be have been done by looking at the number of different labels in the



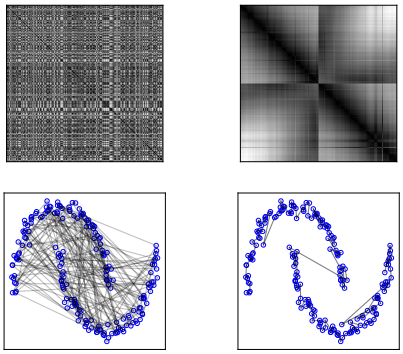
a) VAT results for small blobs dataset.



b) VAT results for `R15` dataset.



c) VAT results for `Aggregation` dataset.



d) VAT results for `moons` dataset.

Figure 2.4: The respective left figures show the unordered distance matrices and a graph visiting points according to this random ordering. The right figures show the distance matrices after they have been re-ordered and the resulting graphs.

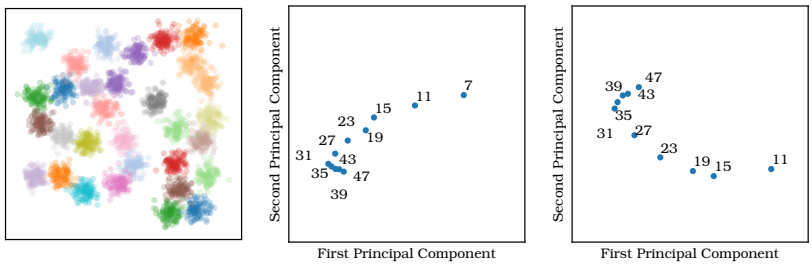


Figure 2.5: Projection of different clustering results onto the plane for `D31`. The similarities between different labelings have been computed with the V-Measure (center) and the ARI (right).

ground truth labeling as well. However, Figure 2.3 shows that even with the ground truth given, the correct clustering could not be found by the k-means algorithm.

The usage of external validation measures becomes more interesting when different clusterings on the same dataset are compared against each other (as seen in Figure 2.5). External validation measures allow a direct comparison of different labelings and may yield some further insight into which labelings are fundamentally different. In such a setting, visual analysis of the dataset to assess the number of clusters (and to some degree maybe even their structure) may be guided by algorithmic assistance. For this one may either use the VAT algorithms or by directly comparing clusterings in a projected space.

The presented numerical measures (RI, MI, and V-Measure) do not yield fundamentally different results for the same comparisons. Their major difference lies in the philosophy behind their calculation (combinatorial, information theoretical, or from an information-retrieval point-of-view respectively)

2.1.2 Internal Validation

In contrast to external validation, internal cluster validation cannot compare the labeling of a dataset against another one. Instead these measures have to rely on *internal* information about the dataset and the labeling itself. Usually this information can be categorized into measures of *compactness*, and *separation*. These can be designed in different ways, as can be seen by

the different validation scores described in the following. In general it can be said, that *compactness* is used as a measure of internal cluster similarity (cf. Section 1.3): points that belong to the same cluster should be similar to each other. *Separation*, on the other hand, measures how dissimilar points from different clusters are.

The overall selection process for the best parameter is rather simple. Every measure has to be either maximized, minimized or from a list of several values the optimal value has to be obtained via the *elbow method* [120]. After performing several runs of the centroid-based algorithm the resulting labeling (or partition matrix) can be converted into a single value via the selected validation measure. The optimal value (and therefore parametrization of the algorithm) can be determined by the optimization criterion associated with the validation measure. Since for centroid-based clustering methods like k-means and fuzzy c-means only a single parameter has to be estimated, such a process is sufficient.

Density-based clustering like DBSCAN required more sophisticated methods for validation since at least two parameters need to be estimated. In [47] it was shown that a similar process extended to a two-dimensional parameter space can lead to favorable results.

Centroid-based Measures

If an internal validation measure refers to the center of a cluster it is called a centroid-based measure. All of these measures prefer spherical clusters over non-spherical ones as their *compactness* definition leads to a minimization of the distance between data points and centroids. This naturally leads to spherical cluster structures.

Davies-Bouldin Index (DB) [43]
This validation index first measures the within-cluster scatter by summing up the distances between each data point assigned to a cluster C_i and its associated cluster center μ_i:

$$S_i = \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} d(x_j, \mu_j) \tag{2.6}$$

Calculating the ratio of the within-cluster scatters and the distances of their respective centroids yields an indicator of how well two specific clusters are separated from each other:

$$M_{i,j} = d(\mu_i, \mu_j) \qquad R_{i,j} = \frac{S_i + S_j}{M_{i,j}} \tag{2.7}$$

The rationale behind this is that if two clusters have a low within-cluster scatter *and* a high distance from each other, they can be easily separated, while highly scattered clusters that are close to each other yield high values and indicate worse separation.

$$D_i = \max_{j \neq i} R_{i,j} \tag{2.8}$$

Taking the maximum over all the $R_{i,j}$ for one cluster considers the worst-case scenario—for each cluster only that one cluster is considered, that it is the worst separated from.

$$DB = \frac{1}{k} \sum_{i=1}^{k} D_i \tag{2.9}$$

Finally, taking the average over all individual scores for each cluster yields the Davies-Bouldin Index (DB). Smaller values of DB indicate a better clustering. The selection process for some typical datasets can be seen in Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9, Figure 2.10, and Figure 2.11 respectively.

Silhouette Coefficient (SC) [112]

The calculation of the Silhouette Coefficient is initiated by calculation of an individual score for each point. For this the average distance from a point x_i to all the points in the same cluster is measured (Equation 2.10). The same is done for each other cluster (those x_i does not belong to) and the minimum is considered as b_i (Equation 2.11):

$$a_i = \frac{1}{|C_{l(x_i)}|} \sum_{y : l(x_i) = l(y)} d(x_i, y) \geq d(x_i, \mu_{l(x_i)}) \tag{2.10}$$

$$b_i = \min_{1 \leq j \leq k, j \neq l(x_i)} \frac{1}{|C_j|} \sum_{y \in C_j} d(x_i, y) \geq \min_{1 \leq j \leq k, j \neq l(x_i)} d(x_i, \mu_j) \tag{2.11}$$

The silhouette score is then the ratio between the difference of b_i and a_i, normalized by their maximal value to obtain scores between -1 and 1:

$$s_i = \frac{b_i - a_i}{\max\{b_i, a_i\}} \tag{2.12}$$

The final score is then the average of all obtained scores:

$$SC = \frac{1}{N}\sum_{i=1}^{N} s_i \tag{2.13}$$

Higher values indicate a better labeling than lower ones. The selection process for some typical datasets can be seen in Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9, Figure 2.10, and Figure 2.11 respectively.

Dunn's Index (DI), Dunn-like Index (DLI) [49]
Dunn's Index can be seen as a generalization of the Davies-Bouldin Index or the Silhouette Coefficient. By calculating the ratio of inter-cluster distance and intra-cluster distance a relatively simple measure can be derived, that needs to be maximized in order to find a good clustering. The generalization of the aforementioned lies in the fact, that Dunn's Index allows different notions of *compactness* and *separation* to be used.

Originally the index was defined as the ratio between the minimal distance between two separate clusters (defined as the minimal distance between any pair $(x, y) \in C_i \times C_j$) and the maximal diameter of any cluster (defined as the maximal distance between any pair from any cluster):

$$DI = \frac{\min\limits_{i \neq j} \min\limits_{x \in C_i, y \in C_j} d(x, y)}{\max\limits_{k} \max\limits_{x, y \in C_k} d(x, y)} \tag{2.14}$$

Nowadays a more relaxed form is used as Dunn's Index, which can lead to several different forms of this validation measure:

$$DLI = \frac{\min\limits_{i \neq j} \delta(C_i, C_j)}{\max\limits_{1 \leq i \leq k} \Delta_i} \tag{2.15}$$

where $\delta(\cdot, \cdot)$ is any kind of metric that measures the inter-cluster distance and Δ_i is a measure for the intra-cluster distance (or within-cluster scatter).

Originally Dunn used the minimal distance between two points from separate clusters for δ and the diameter of a set of points for Δ. As can be seen in Equation 2.7, the inter-cluster distance can also be measured as the distance between the clusters' centroids instead of the smallest pairwise distance.

Higher values of this index indicate a better clustering result. The selection process for some typical datasets can be seen in Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9, Figure 2.10, and Figure 2.11 respectively.

Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) computes the deviance of data points to their associated cluster centers. This closely ties into the concept of spherical clusters in which the distance between center and border would be limited by the cluster's radius. The denominator is adjusted by the number of dimensions m and the number of clusters k:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, \mu_i)}{m \cdot (N - k)}} \tag{2.16}$$

Lower values indicate a better clustering result. The selection process for some typical datasets can be seen in Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9, Figure 2.10, and Figure 2.11 respectively.

Calinski-Harabasz Index (CH) [37]

The Calinski-Harabasz Index is also known as variance ratio criterion and is mainly an F-value of a one-way ANOVA [60]. The enumerator shown in Equation 2.17 represents the variation between groups, while the denominator represents the variation within groups:

$$CH = \frac{\frac{1}{k-1} \sum_{i=1}^{k} |C_i| d^2(\mu_i, \mu)}{\frac{1}{N-k} \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, \mu_i)} \tag{2.17}$$

Higher values indicate a better clustering result. The selection process for some typical datasets can be seen in Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9, Figure 2.10, and Figure 2.11 respectively.

Fukuyama-Sugeno Index (FS) [65]
The Fukuyama-Sugeno cluster validity index is based on the difference of compactness of the clusters and the separation between the different clusters. Compactness is measured by the same expression that is also used in the k-means algorithm (cf. Algorithm 1 and Equation 1.1) as objective function, while separation is the minimal distance between two clusters' centroids. In this sense it is similar to the Dunn-like measures or the Silhouette coefficient. The main difference is that Fukuyama-Sugeno uses the sum of the obtained values instead of the respective maxima or minima:

$$FS = \left(\frac{1}{N} \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, \mu_i) \right) - \left(\min_{1 \leq i \neq j \leq k} d^2(\mu_i, \mu_j) \right) \tag{2.18}$$

The optimal values can be found by using the *elbow method* [120] The selection process for some typical datasets can be seen in Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9, Figure 2.10, and Figure 2.11 respectively.

Xie-Beni Index (XB) [132]
The Xie-Beni Index (XB) is structurally similar to the previously mentioned Fukuyama-Sugeno Index. It uses the same definitions for separation and compactness, but instead of looking at their difference, the Xie-Beni Index calculates their ratio.

$$XB = \frac{\sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, \mu_i)}{N \cdot \min_{1 \leq i \neq j \leq k} d^2(\mu_i, \mu_j)} \tag{2.19}$$

Lower values indicate a better clustering result. The selection process for some typical datasets can be seen in Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9, Figure 2.10, and Figure 2.11 respectively.

Partition Matrix-based Measures

Since fuzzy clustering does not yield a strict 0 or 1 labeling (cf. Equation 1.3) the previously mentioned validation measures do not apply to fuzzy clustering results. Although fuzzy clustering results could be easily defuzzified by assigning each point to the cluster it has the highest membership to, infor-

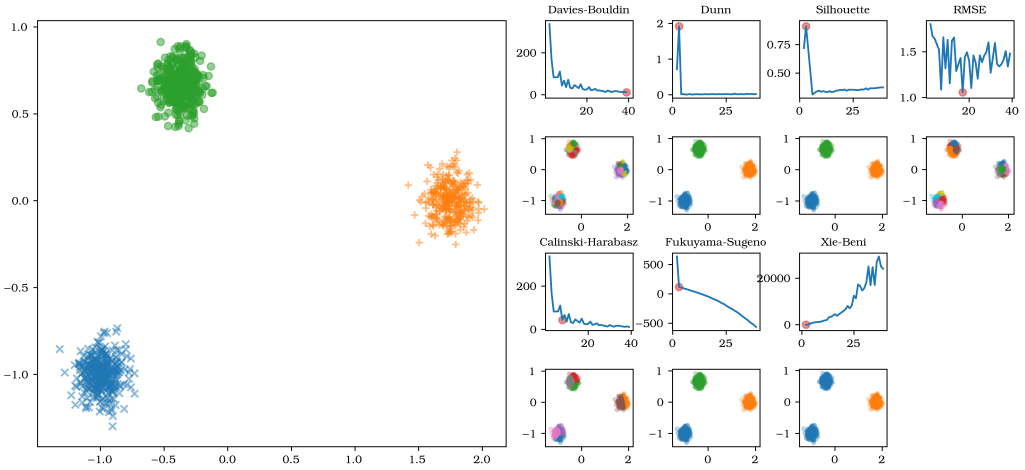


Figure 2.6: Original cluster structure shows three spherical clusters (`blobs`). Four out of seven shown validation measures select the right number of clusters when used with *k*-means.

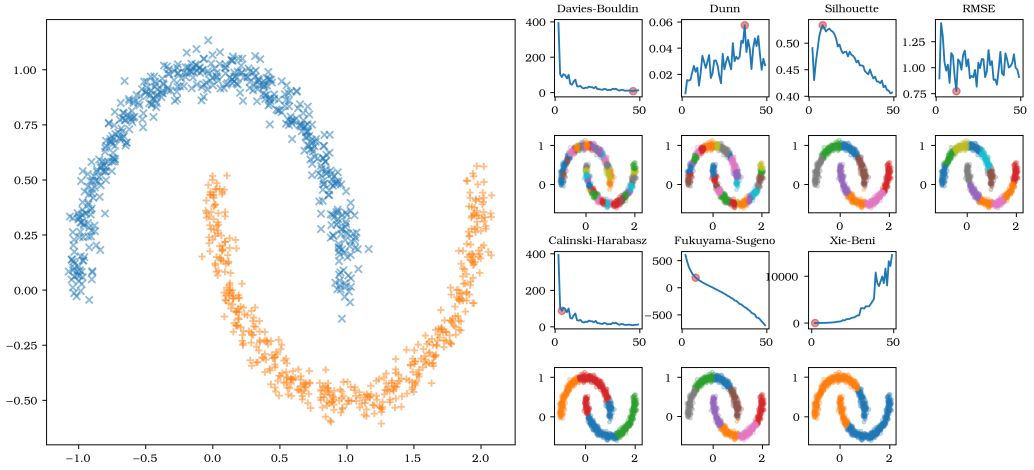


Figure 2.7: Original cluster structure shows two half-circle clusters (`moons`). Almost all shown validation measures select a wrong number of clusters when used with *k*-means. Respectively selected results shown below the measures' charts.

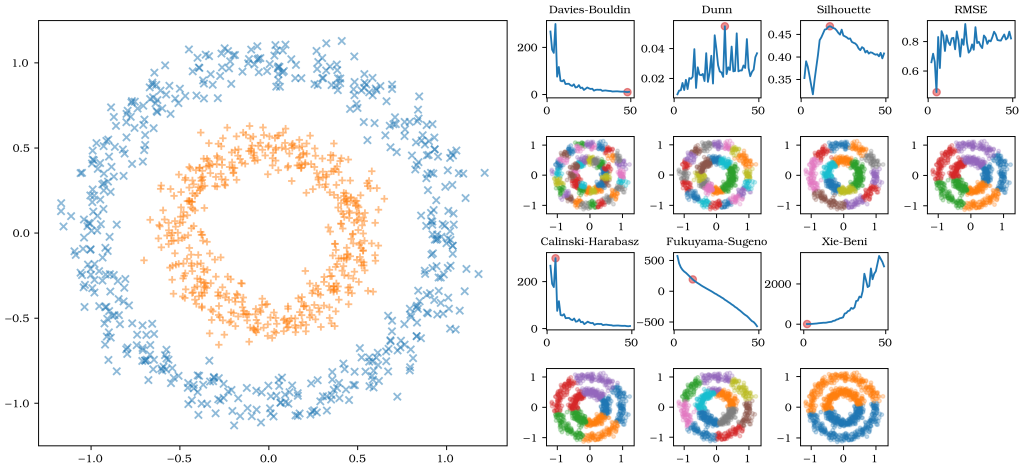


Figure 2.8: Original cluster structure shows two circular clusters (`circles`). Almost all shown validation measures select a wrong number of clusters when used with *k*-means. Respectively selected results shown below the measures' charts

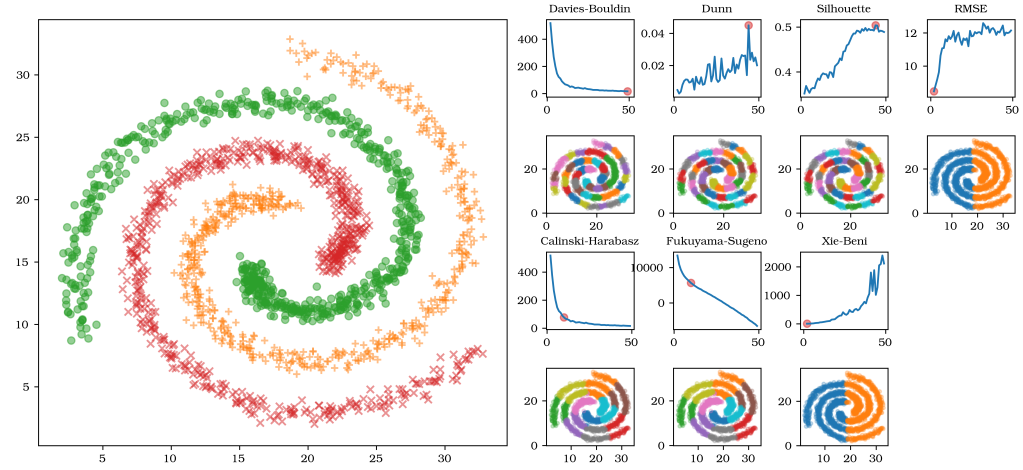


Figure 2.9: Original cluster structure shows three tubular clusters arranged in a spiral (`spirals`). All shown validation measures select a wrong number of clusters when used with *k*-means. Respectively selected results shown below the measures' charts

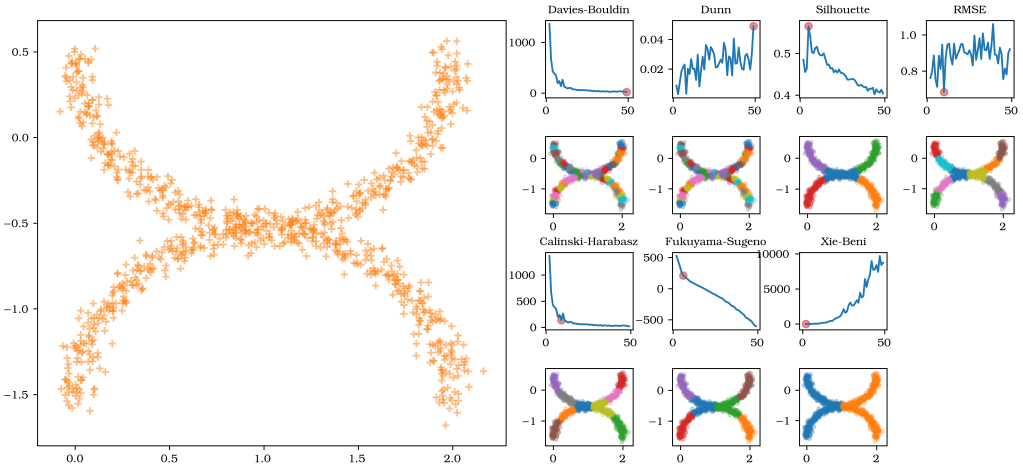


Figure 2.10: Original cluster structure shows a single χ-shaped cluster (`chishape`). All shown validation measures select a wrong number of clusters when used with k-means. Respectively selected results shown below the measures' charts

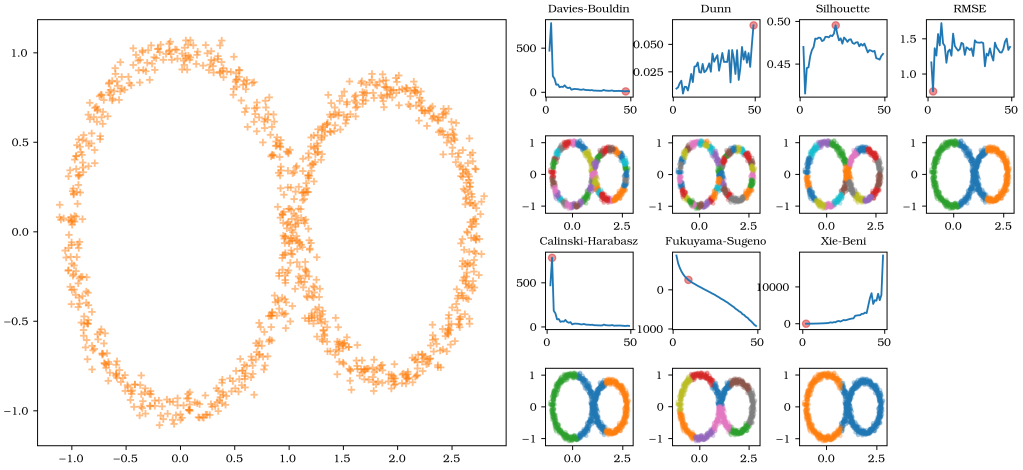


Figure 2.11: Original cluster structure shows a single 8-shaped cluster (`eightshape`). All shown validation measures select a wrong number of clusters when used with k-means. Respectively selected results shown below the measures' charts.

mation about the ambiguity of some cluster assignment (e. g. a point almost directly between two clusters) would be lost after defuzzification. Instead the degree of membership has to be taken into account in the validation process, too. Some measures that do so are presented in the following:

Xie-Beni-Index [132]
Originally designed for fuzzy clustering, the Xie-Beni index works as well for crisp clusterings (see above on page 34). Only the distances in the enumerator's sum are weighted by the membership degree and the fuzzifier.

$$XB_\omega = \frac{\sum_{i=1}^{k} \sum_{x_j \in C_i} u_{ij}^\omega d^2(x, \mu_i)}{N \cdot \min_{1 \le i \ne j \le k} d^2(\mu_i, \mu_j)} \tag{2.20}$$

With this adjustment the enumerator's terms equals that of the fuzzy-c-means objective function. Lower values indicate a better clustering result.

Partition Coefficient (PC) [14]
One of the many, purely partition matrix-based measures is the Partition Coefficient. It argues that the coupling between a fuzzy partition with itself can be measured by the average pairwise relative overlap of the corresponding fuzzy membership vectors. This is realized by first calculating $\mathcal{U}\mathcal{U}^T$, $\mathcal{U} \in [0, 1]^{c \times n}$, then calculating the trace of the resulting matrix, and dividing by the number of data points:

$$PC = \frac{1}{n} trace(\mathcal{U}\mathcal{U}^T). \tag{2.21}$$

Since it is possible to study this function analytically it can be shown, that an optimal partitioning is obtained when this function is maximized.

Actually a true maximum would be obtained for a crisp clustering. The off-diagonal entries in $\mathcal{U}\mathcal{U}^T$ can only be zero if there is no agreement between the corresponding membership vectors. However, since it is usually applied to fuzzy clustering results, it can be said that the partition coefficient favors labelings that are as crisp as possible. Higher values thus indicate a better clustering result.

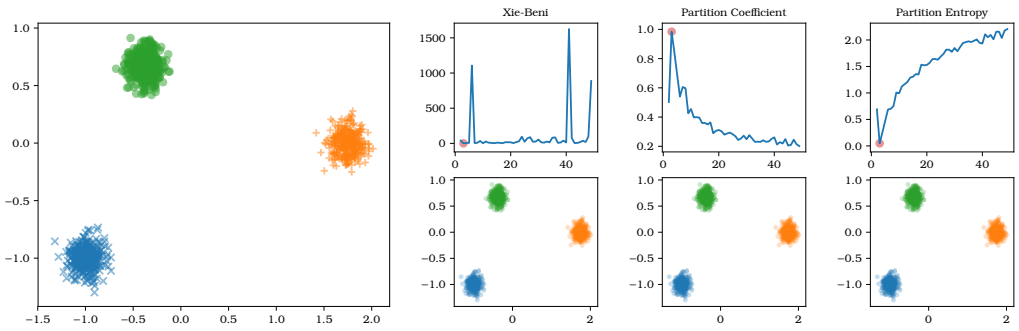


Figure 2.12: Original cluster structure shows three spherical clusters (`blobs`). All shown validation measures select the right number of clusters when used with k-means.

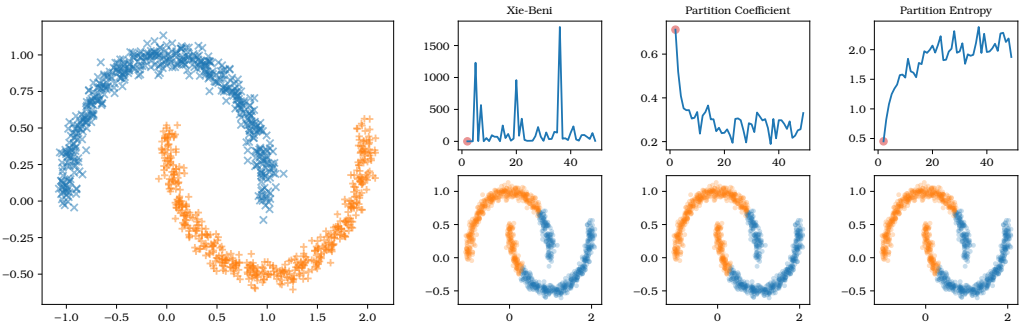


Figure 2.13: Original cluster structure shows two half-circle clusters (`moons`). All shown validation measures select a wrong number of clusters when used with k-means. Respectively selected results shown below the measures' charts.

Partition Entropy (PE) [15]

The partition entropy measures how orderly the entries within the partition matrix are. As with Shannon entropy [50], the partition entropy obtains its optimal value, when the entries in the matrix are only 0 or 1.

$$PE = \frac{1}{n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij} \cdot \log u_{ij} \tag{2.22}$$

Lower values indicate a better clustering result.

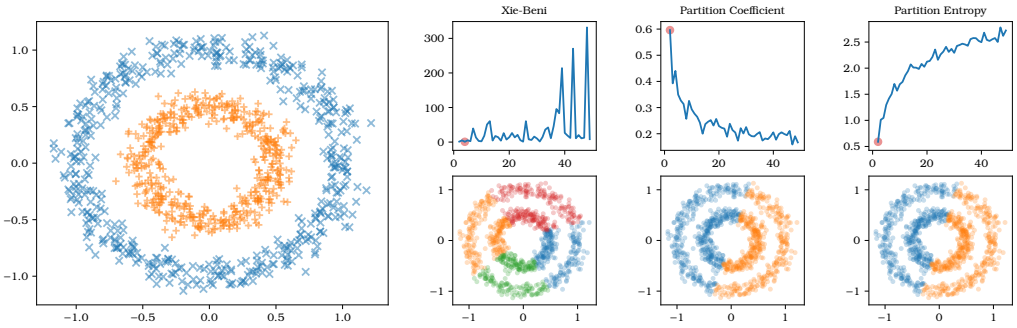


Figure 2.14: Original cluster structure shows two circular clusters (`circles`). All but one shown validation measures select the correct number of clusters when used with k-means. Respectively selected results shown below the measures' charts

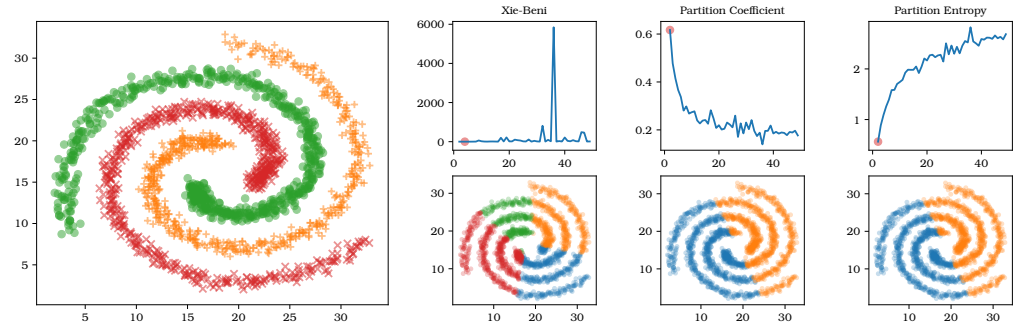


Figure 2.15: Original cluster structure shows three tubular clusters arranged in a spiral (`spirals`). All shown validation measures select a wrong number of clusters when used with k-means. Respectively selected results shown below the measures' charts

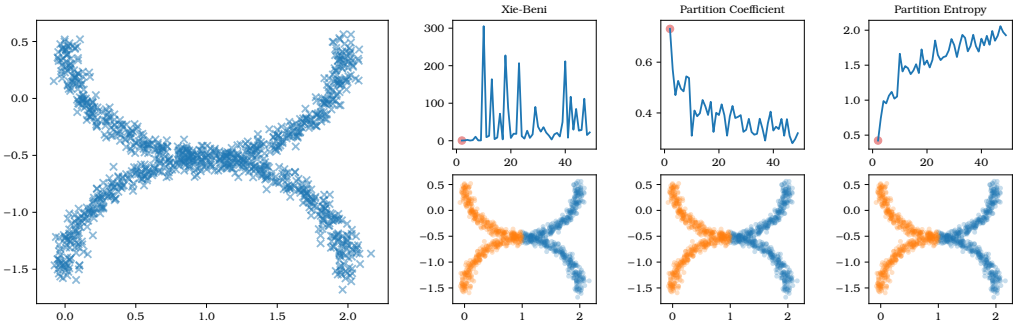


Figure 2.16: Original cluster structure shows a single χ-shaped cluster (`chishape`). All shown validation measures select a wrong number of clusters when used with k-means. Respectively selected results shown below the measures' charts

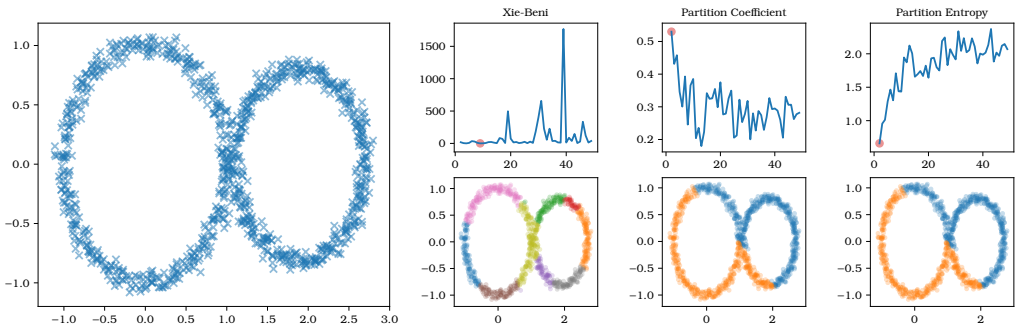


Figure 2.17: Original cluster structure shows a single 8-shaped cluster (`eightshape`). All shown validation measures select a wrong number of clusters when used with k-means. Respectively selected results shown below the measures' charts.

2.1.3 Density-based Cluster Validation

The validation of density-based clustering results is still a rather new and open research topic. One of the recent advancements is the development of the Clustering Validation Index based on Nearest Neighbors (CVNN) [94]. The index itself is calculated from the normalized separation and normalized compactness (the respective measures divided by their maximally obtained values). The main contribution here was that separation is not calculated globally for a cluster but rather locally for each point. If such a point has another point in its k-nearest-neighborhood and this other point does not belong to the same cluster, the score is increased. The lower the sum of this score is the better the separation between the different clusters. Problematic—however—is the introduction of a new, user-definable parameter into the validation process. Now, not only the clustering result but also the validation of this result are subject to the parameter choice of the user and can therefore not be judged completely independent from each other.

In [70] a method is proposed that tries to capture the geometric properties of a cluster by choosing a set of k representative points along its border. The first point is the one, which is farthest away from the center of the cluster, all additional points are chosen such that they maximize the distance to all points already selected. In this it acts quite similar to the k-means++ initialization method (cf. [8]). The clustering is then evaluated using these points as anchors for calculating the clusters' cohesion, compactness and

separation. A cluster has a high cohesion, if its density does not fluctuate very much. Compactness and separation are used in a similar way as inter-cluster and intra-cluster distances (which are used by the Davies-Bouldin-Index or the Dunn-Index). However, since the number of representatives selected is the same for each cluster, one may end up with too sparsely scattered points in one and too tightly spaced points in another cluster. This method also introduces an additional parameter into the validation process, which is generally undesired.

Another approach was to adopt the silhouette score to a density-based clustering scenario. In [47] an approach has been presented that directly adopts the separation measure of the silhouette score and redefines the compactness. Instead of using the distance to the cluster's centroid the longest edge in a minimum spanning tree over the data points within the cluster is used. By choosing the longest edge an upper bound for (non-)compactness is considered. The longer this longest edge was, the worse the cluster's coherence would be.

In [102] an inverse formulation of density is used to define the Mutual Reachability Distance (MRD). The MRD between two points is the maximum between their individual core distances and the true distance between them. If this distance is calculated for any two objects within a cluster, it can be used to construct a weighted, complete graph. The data structure used for validation is then calculated from a minimum spanning tree of this graph. The maximal weight within this minimum spanning tree is used as a sparseness measure (the inverse of compactness), while the separation between two clusters is measured by the minimal mutual reachability distance between objects of these clusters.

Both of the aforementioned approaches take the cluster shape better into consideration by using a minimum spanning tree for representation (w.r.t. the CVNN measure).

Because both of the aforementioned approaches use a minimum spanning tree to represent the clusters they take the shape of the clusters better into account than the CVNN measure. Using a generalized descriptor (or representative) of the clusters' shapes seems to be a natural approach for solving the problem of density-based cluster validation without introducing additional parameters.

2.2 Re-Representing Datasets

Ultimately, when using cluster skeletons instead of the arithmetic mean one chooses a different representation of the cluster itself for the validation process. There are many choices for finding a proper representation of a dataset and at least as many ways of obtaining them.

2.2.1 (Non-)Linear Transformations

One of the easiest ways to re-represent data is by a linear transformation. E. g., performing a Principal Component Analysis on the covariance matrix of a dataset allows a lower dimensional representation of the same data which preserves most of its variance. This applies not only to the whole dataset but to individual cluster as well. This can be seen either in the usage of the Mahalanobis distance [98], or the Gustafson-Kessel algorithm [68].

Different algorithms for linear transformations are the Linear Discriminant Analysis (LDA) [11, 61], and the Generalized Discriminant Analysis (GDA). The first finds a hyperplane separating two classes while the latter finds a general separation boundary. The parameters of the separating objects can also be used to transform the original dataset into a lower-dimensional version of itself. However, both methods need class information which is usually not available in clustering *a priori*.

A non-linear transformation is multi-dimensional scaling [13]. By minimizing the *stress* between the original dataset and its lower-dimension representation a good approximation can be achieved. *Stress* here is the sum of pairwise differences of the distance matrices in both the high- and low-dimensional space. If the pairwise differences as well as the overall sum are normalized by the (sum of) distance(s) in the original space, the scaling is also called Sammon's Mapping [114].

Another non-linear way of re-representing data is by using structure learning algorithms such as the Self-organizing Map (SOM) algorithm [85] or the Growing Neural Gas (GNG) [63]. The first method adopts an initial structure (often a grid network) to the dataset, while the latter starts with just a single edge and lets the network iteratively grow. This approach will be discussed in more detail in Section 3.3.5.

2.2.2 Core Sets

Another way of re-representing data is by finding a smaller sample that still captures all interesting features of the original dataset. *Core Sets* [1] are one way of describing such data summarizations. Several different understandings of core sets exist. Yet, they all have in common that they aim at generating a smaller dataset that can be used in calculations. Any calculation on this smaller sample is supposed to yield approximately the same result. The resulting centers of a k-means clustering can be seen as one possible core set of the underlying dataset $\mathcal{X}$ [35, 41, 58]. Any query of the form "Which structure is point x closest to?" can be answered with approximative correctness by only comparing x to the cluster centers obtained via k-means instead of comparing it to all points. The representation of a cluster used in [70] can actually be seen as such a core set approach. By capturing interesting and important structural properties the true cluster quality is approximated by the k representatives. These k points actually form a kind of core set and are used to (approximately) answer the question about the cluster's quality.

In a broader view core sets yield a sparser (or smaller) dataset which can be used to speed up otherwise costly calculations. Validating the labeling of a dataset is usually not that difficult or complex once the measure has been chosen and a (single) labeling has been obtained. Especially in the case of density-based clustering the problem lies more in a proper representation of a single cluster. For this task (as discussed earlier) and in many cases single points are not suitable. Core sets do not yield a structure that can be easily plugged into existing validation measures since the core set itself has to be seen as a dataset $\mathcal{X}'$ on its own—although it has been obtained from the original dataset $\mathcal{X}$.

Additionally, methods for generating core sets range in complexity from $O(n)$ to $O(n^5)$ (depending on the type of core set and the definition of core set used). As will be seen in Section 4.2, possibly all $O(n^2)$ entries of the distance matrix and a certain number m of possible values for *minPts* in DBSCAN might need to be tested. And for each of these $O(n^2 m)$ trials a core set representation would need to be found. This could in the very worst case lead to a runtime complexity beyond $O(n^8)$ rendering such an approach infeasible for any dataset of interesting size.

2.2.3 Skeletonization

Another group of algorithms that try to re-represent datasets are image skeletonization algorithms. In terms of computational geometry these algorithm try to find a topological skeleton [9] in various kinds of ways.

The group of *thinning* algorithms takes a binary image and removes pixel from the outside of a shape until a small line is remaining (e. g. [105]). The so-called *grassfire transformation* is probably the best known algorithm in this class.

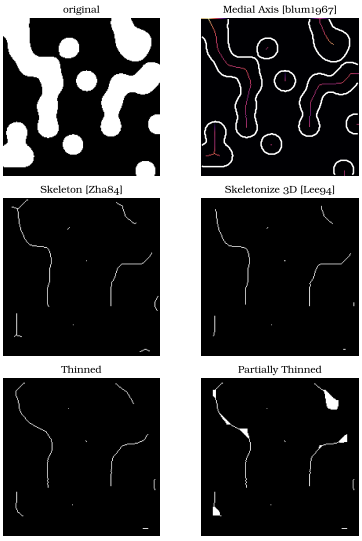


Figure 2.18: Different skeletonization methods applied to a binary image. Skeletons are the white pixels in the corresponding images, except in case of the medial axis, where white pixels denote the detected borders and red pixels the skeleton).

A strictly mathematical definition is given in [19]. The topological skeleton found here is the medial axis, which is the set of centers of bi-tangent circles. This algorithm requires that the exact boundary of the object to be skeletonized is known. This is certainly the case for binary images [135].

A third group is based on Discrete Curve Evolution (DCE) [91, 119]. The term *evolution* here is not to be understood in the same way as in *evolutionary algorithms*. Instead a shape describing curve (or polygon) is evolved over several steps and thus simplified until it can e. g. act as some kind of topological skeleton.

Lastly, the group of algorithms based on *distance transformations* aims at detecting ridges in the height profile of an image [20, 79]. Such a height profile is obtained by a discrete filtering or kernel function. Ridges are then those points in the (discrete) profile which have no neighboring points with a higher value.

Most of the aforementioned methods have in common that they work only on discrete (pixelized) binary data. This, of course, has limited use in data analysis, where datasets could easily have more than two dimensions and data points usually lie within a continuous space. Only the medial

axis transformation can be used without adaptation on continuous data. However, it is mostly limited to two-dimensional shapes. In three dimensions one could find medial surfaces and apply thinning algorithms on them [92] to generate skeletons in three dimensions.

For more than three dimensions these approaches all become infeasible since they are not directly applicable to arbitrary datasets in high-dimensional, continuous spaces. These algorithms can still be used as an inspiration for the algorithms used in Section 3.3.

CHAPTER 3

Generalized Centroids

In this chapter several different approaches to generate a cluster skeleton will be discussed. A cluster skeleton will be the object that acts as a representative for all the points of a cluster and replaces the otherwise commonly used centroid in the context of density-based clustering.

Though not exhaustive the presented methods will cover a wide variety of different approaches and give a good overview over what can be expected from which kind of method.

3.1 Necessary Terms and Definitions

In the usual setting described in Section 1.3, clustering algorithms refer to a centroid. This is a single point (which is not necessary part of the data set) that can be calculated by e. g. the arithmetic mean of all data points that are assigned to the cluster which the centroid should represent. In a more general notion, the centroid could be any kind of object suitably representing the points belonging to the cluster it stands for. In case of (hyper-)spherical cluster shapes the arithmetic mean of the cluster's members can represent the cluster appropriately (cf. Figure 3.1). As one can see, the arithmetic mean is a suitable centroid for spherical clusters. For more general cluster types like the half-circles shown earlier, the arithmetic mean (or even a single point) are no suitable representatives anymore.

In these cases a single point is not sufficient as a centroid and a different structure needs to be created. Thus a generalized version of the term centroid is needed which allows more complex structures to be equally well repre-

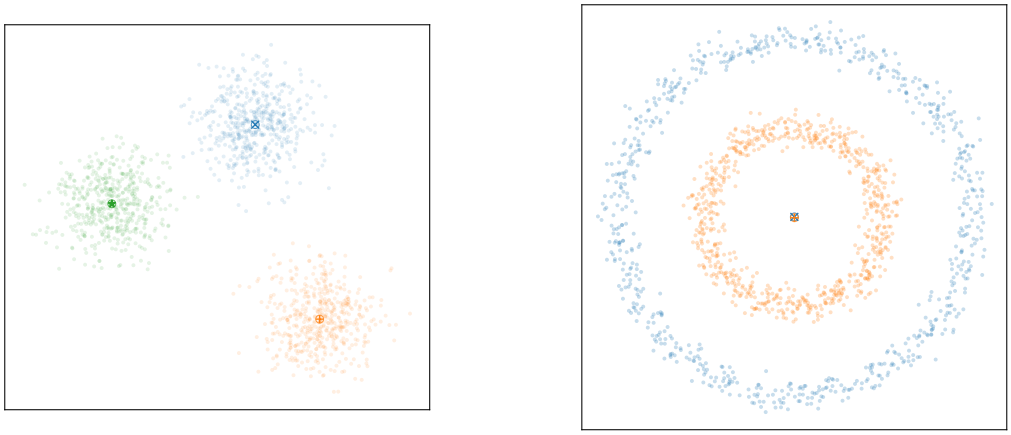


Figure 3.1: Three spherical clusters with their centroids in the left figure. The centroids represent their clusters well, in contrast to the centroids shown in the right figure.

sented as simple, spherical cluster structures. Such a generalized centroids is an object that stands for the cluster and represents the cluster's members appropriately while still providing some form of proper generalization.

The idea of generalizing cluster prototypes to more abstract concepts is not completely new. In one way the Mahalanobis distance [98] and later on the Gustafson-Kessel algorithm [68] can be seen as first step towards this direction. Within the Gustaffson-Kessel algorithm clusters need not be strictly spherical in the euclidean sense anymore. Instead the covariance matrix of each cluster provides a linear transformation from an elliptical cluster to a spherical one.

Later on different cluster prototypes have been proposed that extend this generalization to circles, rectangles or quadratic shells [5, 72, 82, 84, 87, 88] This thesis will go one step further by allowing any skeletal shape (i. e., a structure composed of several connected line segments) to be the centroid of a given cluster.

In the two-dimensional case the medial axis (see Section 3.2) can be seen as such a skeleton. For the more general case of arbitrary dimensionality one has to fall back to more approximative structures. Section 3.3 will give an overview of methods that can be used to find such structures.

3.1.1 Manifolds and Hausdorff Metrics

The search for a centroid can be seen as (a possibly non-linear) dimension reduction and lies therefore in the domain of manifold learning. A manifold is a topological structure which locally resembles euclidean space [121]. The surface of a three-dimensional sphere for example cannot be directly mapped onto a two-dimensional plane. However, certain mappings allow finding a two-dimensional representation that maintains some properties of the original space (e. g. Mercator projection [99, 117] which guarantees angular accuracy). Such a mapping can only be accurate in a local neighborhood of any given point such that a set of mappings (or charts) is needed to properly describe the complete topological structure. Such a collection of charts is called an atlas. Much like historical charts and atlases describe the surface of the earth, maps and atlases (as collections of maps) describe in mathematical terms how a topological space can be represented with several euclidean approximations.

Definition 3.1 (*r-neighborhood*)

The r-neighborhood $\mathcal{N}_r(p)$ of a point $p \in \mathbb{R}^d$ is defined as the set of points $x \in \mathbb{R}^d$ which are contained in a hypersphere with radius r around p.

Note that this is similar to the definition of $\mathcal{N}_\varepsilon(x)$ in Definition 1.7. The difference between the two here is that the r-neighborhood describes a subset of the space p is located in and $\mathcal{N}_\varepsilon(x)$ describes a subset of the data set $\mathcal{X}$, that x is member of. The canonical extension from the point-wise definition of an r-neighborhood to that of a set of points $\mathcal{P}$ is that the r-neighborhood of a set of points is the union of all the points' r-neighborhoods:

Definition 3.2 (*r-neighborhood of a set*)

The r-neighborhood $\mathcal{N}_r(\mathcal{P})$ of a set of points $p \in \mathcal{P} \subseteq \mathbb{R}^n$ is the union of the r-neighborhoods of the points contained in $\mathcal{P}$, i. e., $\mathcal{N}_r(\mathcal{P}) = \bigcup_{p \in \mathcal{P}} \mathcal{N}_r(p)$.

If $\mathcal{P}$ is convex, then the r-neighborhood is also called *tubular neighborhood*.

In order to introduce a topological structure on the set of subsets of a metric space, the Hausdorff metric is a good option.

Definition 3.3 (*Hausdorff distance*)

The Hausdorff distance $\delta_H(\mathcal{A}, \mathcal{B})$ between two non-empty, compact subsets $\mathcal{A}$ and $\mathcal{B}$ of euclidean space is defined as:

$$\delta_H(\mathcal{A}, \mathcal{B}) = \max\left\{\sup_{a\in\mathcal{A}}\left\{\inf_{b\in\mathcal{B}}\left\{d(a,b)\right\}\right\}, \sup_{b\in\mathcal{B}}\left\{\inf_{a\in\mathcal{A}}\left\{d(a,b)\right\}\right\}\right\}$$

The Hausdorff distance may of course be used to compare curves as well as polygons as well as sets with a non-zero diameter. However, if one restricts themselves to curves and polygons, the Fréchet distance δ_F [4, 62] is a more suitable choice, since it takes into account the flow of the curves. For polygonal representations of curves there exists a polynomial time algorithm [54] that computes the coupling distance (or *discrete* Fréchet distance) between two polygons.

In the following (V, d) denotes a metric space.

Definition 3.4 (*Curve*)

A curve is a continuous mapping f of the closed interval $[u, v]$ to V.

Definition 3.5 (*Reparametrization of a Curve*)

A reparametrization of a curve f is a continuous, non-decreasing surjection from $[u, v]$ to $[0, 1]$

Definition 3.6 (*Fréchet Distance [3]*)

Let f and g be two curves. Let a, β be reparametrizations of f and g respectively. The Fréchet distance between f and g is given by:

$$\delta_F(f, g) = \inf_{a,\beta}\left\{\max_{t\in[0,1]}\left\{d(f(a(t)), g(\beta(t)))\right\}\right\}.$$

If f and g are represented as polygons, then the computation can be performed faster and the distance measure is called *discrete* Fréchet distance δ_{dF} [54].

To represent clusters without too much over-fitting the term cluster skeleton should be introduced:

Definition 3.7 (*Cluster Skeleton*)

A cluster skeleton $\mathcal{S}_C$ of a cluster C is an object consisting of line segments (sections of a straight line limited by two points) such that the union of the tubular neighborhoods of the individual line segments of $\mathcal{S}_C$ contains all points of C for an adequately small r.

Such a cluster skeleton $\mathcal{S}_C$ could also be interpreted as a graph $G_{\mathcal{S}_C} = (V, E)$ with edges $e = (u, v) \in E$ for each line segment composing $\mathcal{S}_C$. The vertices are those points where each line segment ends and several line segments may end at a common vertex (a joint). Points at which only one edge ends are (from a graph-theoretical point-of-view) leaves.

In the following $G_{\mathcal{S}_C}$ and $\mathcal{S}_C$ will be used equivalently if the distinction is clear from the context.

3.1.2 Modified Fréchet Distance

To evaluate the different methods proposed to generate the cluster skeletons and compare their results with the underlying (and due to the data generation process known) ground truth *skeleton* of the clusters, a modified version of the discrete Fréchet distance will be used. Since one cannot expect the cluster skeletons that have been found to be strictly linear (in the sense that their graph structure contains exactly two leaves) one cannot directly apply the Fréchet distance to calculate the distance between the estimated cluster skeleton and the ground truth.

To cope with this problem the distance calculation is split into several parts. The main idea is to calculate all possible distances between pairs of paths. For this all possible pairs of leaves in the ground truth structure and all possible pairs of leaves within the found cluster skeleton are formed. Since the general procedure will yield a tree-like structure one can safely assume that only one path exists between each pair of leaves (and that this is also the shortest path). All pairwise discrete Fréchet distances between pairs of paths formed in such ways are then computed. Ideally each path in the found skeleton corresponds to one path in the original model and will therefore have the smallest distance to the original model's paths. All other

paths will in some point be dissimilar to the underlying model's paths. For each path within the original structure a corresponding path in the cluster skeleton and vice versa can be found.

From this point on there are different options of aggregating these pairwise distances into a single value. The optimal fit between two paths is obviously between the pair that yields the minimum of all pairwise Fréchet distances. Using the minimum of all pairwise distances leads to an optimistic interpretation of the path comparison. A single good fitting pair of paths would dominate this calculation—no matter how bad the fit between the other pairs of paths is. Taking the maximal distance between any two paths is not feasible, though. Testing all pairs necessarily also tests those parts of the structure against each other which do not correspond to each other. These parts must have a high distance to each other and lead to the most pessimistic interpretation of the obtained distance values possible. The last option considered is taking the maximum over all minimal distances. Since every path in the cluster skeleton should have a corresponding path in the original structure and the Fréchet distance between those two should be minimal for either of the two, one can select the best fitting partner for each path. The maximum over all distances obtained in that way now describes the worst fit of any of the corresponding pairs of paths.

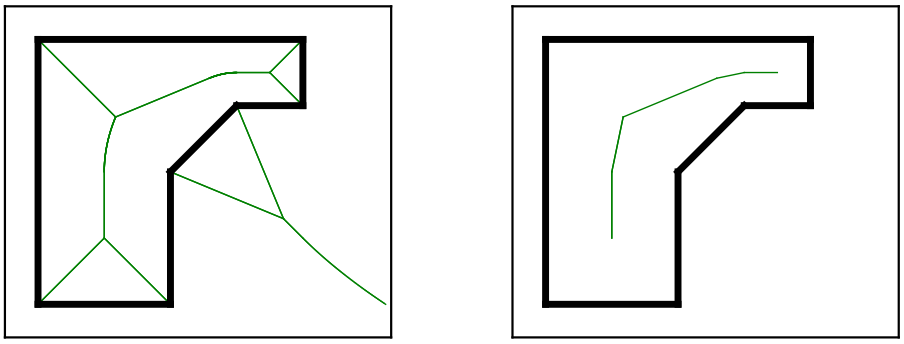
Formally, the modified Fréchet distance for trees can be defined as:

Definition 3.8 (*Discrete Fréchet Distance for Trees*)

Let (a, b) be a pair of leaves in $\mathcal{S}_C$ and let (u, v) be a pair of leaves in $\mathcal{S}_{C'}$. Let $\overbrace{a - b}^{\mathcal{S}}$ denote the path between a and b in $\mathcal{S}$.

The discrete Fréchet distance between the tree-like structures $\mathcal{S}_C$ and $\mathcal{S}_{C'}$ is then: $d_{dFt}(\mathcal{S}_C, \mathcal{S}_{C'}) = \max\left\{ \min_{(a,b)\in\mathcal{S}_C,(u,v)\in\mathcal{S}_{C'}} \left\{ d_{dF}(\overbrace{a - b}^{\mathcal{S}_C}, \overbrace{u - v}^{\mathcal{S}_{C'}}) \right\} \right\}.$

This final measure d_{dFt} will be used throughout the rest of this chapter to compare the found cluster skeletons to the (known) ground truth from which the sample clusters were generated.



a) Green: medial axis. Parts of the medial axis lie outside of the polygon due to it concavity.

b) Green: simplified medial axis.

Figure 3.2: A non-convex region bounded by a simple polygon (black).

3.2 Cluster Skeletons in Two Dimensions

If the dataset to be analyzed is only two-dimensional a construct known from computational geometry can be used to create a cluster skeleton. The medial axis [19] is defined as the set of all center points of maximal circles that touch but do not intersect the border of a region at least twice. An example of the medial axis for a non-convex but still fairly simple region can be seen in Figure 3.2a. As can be seen, not all parts of the medial axis are suitable to be used as a generalized centroid as smaller and smaller circles will fit into corners without being completely contained by other circles. Also, there are arched edges which run around corners of the border polygon that lie on the edges causing concavities. If those edges are removed that run towards the corners of the border polygon, and the arched edges are approximated by line segments and only those edges are kept which lie strictly inside the polygon, then the result will be a reduced medial axis (cf. Figure 3.2b).

3.2.1 Concave Hull and Medial Axis

Obtaining the medial axis for a set of points is not a straightforward problem. Cluster shapes might be arbitrarily complex and the region bounds are

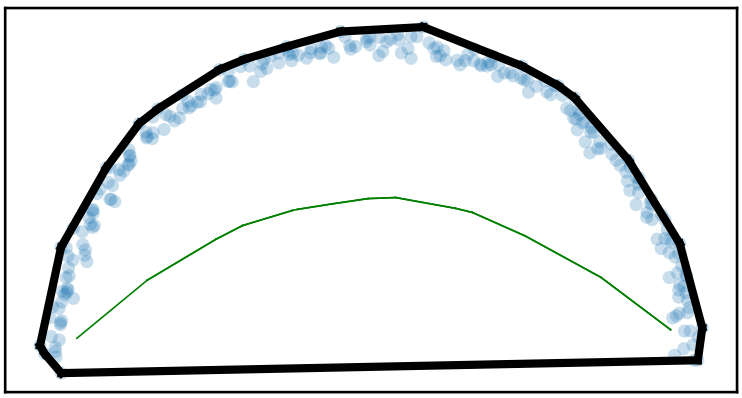


Figure 3.3: The convex hull (bold, black) is in general not a good boundary descriptor when one wants to use its simplified medial axis (green) as a generalized centroid.

usually not given. A simplistic approach would be using the convex hull of the point set. The resulting polygon certainly bounds the region where points are located but usually over-simplifies the shape (cf. Figure 3.3). If the structure is not convex then first a more complex but also more fitting description of the point cloud's shape needs to be constructed.

In [30] a way of obtaining a more detailed region bound by an iterative refinement scheme has been presented. The resulting boundary or concave hull can then be used to find a cluster's skeleton by using the simplified medial axis. Please note that there is no proper definition of what a concave

hull actually is. The algorithm presented here will refine the convex hull of a point set as long as there are still edges on the hull that are deemed too long.

The algorithm iteratively refines an existing hull by selecting the longest edge $e_l = (u, v)$. This edge is consecutively replaced by one ore more edges such that empty areas within the hull are reduced.

To do this, a point p is chosen which is contained in the hull and which will be used to replace e_l with the edges (u, p) and (p, v). This point can be either the point that lies closest to the center of e_l or the point that lies closest to the line segment defined by e_l itself. The former method intuitively allows to split the dataset into two separate regions and apply a divide-and-conquer approach to find the concave hull. However, there can be situations in which the new hull will not include all the points previously contained by the original hull.

To circumvent such situations the split point should be chosen as the point lying closest to e_l. Though one cannot apply the divide-and-conquer scheme anymore and the algorithm becomes purely iterative, the refined boundary is guaranteed to still contain all points formerly contained in the hull. This is easy to see, if p and e_l with its endpoints u and p are interpreted as a triangle. By assumption p is the closest point to e_l and the proposition is that after replacing the edge e_l with the edges (u, p) and (p, v) all points formerly contained within the hull are also still contained within. Assume there was a point q that after replacing is not contained within the hull. Since the only part of the hull that has changed was around the edge e_l, q has to lie within the triangle formed by p, u and v. Any points within the triangle can be constructed by moving from p on the perpendicular to e_l and then either move towards u or v parallel to e_l. However, by the intercept theorem [2] it follows, that any point constructed in such a way must lie closer to e_l than p. This is a contradiction to p is the closest point to e_l and thus a point q cannot exist within the triangle. If no such point exists, then the resulting hull muss still contain all points it formerly contained, q.e.d..

Edges are replaced until a termination criterion is met. Any edge longer than a user-defined parameter ∂ has to be replaced. To find an appropriate value for ∂, different dataset statistics that give an estimate of the density of the dataset can be used.

Using the Delaunay tessellation (or more specifically the lengths of the edges contained in the Delaunay tessellation) yields such a statistic. Intuitively it acts as an indicator of density in this scenario. By using the Delaunay tessellation one acknowledges that longer edges in the tessellation indicate less dense regions in the dataset while shorter edges indicate denser regions. Any occurring length that could be seen as an outlier in the distribution of the edges' length of the Delaunay tessellation indicates that this edge does not properly describe the underlying dataset. Such an edge is a candidate for the refining process. Of course outliers in this sense are only those edges that are significantly longer than the average edge even if outliers in the other direct, i. e., very short edges, may exist. Other dataset statistics that could be used instead of the Delaunay tessellation edge lengths are e. g. Minimum Spanning Tree edge lengths, or the distance of points to their k-nearest neighbors. Overall they are all only different means of defining the threshold.

To better scale the range of possible values from which ∂ can be chosen, two parameters are used. By default this algorithm find ∂ as the length of an edge that is $q = 3$ times longer than the p-th percentile if the distribution of edges' lengths within the Delaunay tessellation. One could argue that the algorithm now not only uses one but two parameters but the parameter combination (p, q) could always be replaced by an alternative combination $(p, 1)$ of p is chosen accordingly. Setting $q = 3$ relaxes the set of values from which p can be chosen to a wider range. This makes parametrization more intuitive. The difference between two hulls that result from different values for ∂ (and consequently from different values of p and q) is simple: Let $\partial_1 < \partial_2$. The hull originating from ∂_1 will be more ragged since more edges need to be removed. The resulting medial axis will therefore have more arched segment (from possibly more concavities in the hull) and also more segments then connect to the hull. However, since the simplified medial axis will be used, the latter segments are not of interest at all and will be pruned anyway while the arched segments will be interpolated by single line segments.

The result of applying Algorithm 2 to an artificial dataset can be seen in Figure 3.4 and Figure 3.5 respectively. These figures show the same dataset with their respective concave boundaries (see Appendix B.4 for details of

Algorithm 2: Creating a concave hull

Require:
$\mathcal{X}$ set of data points,
∂ threshold for the edges that need refinement

1: **function** ConcaveHull($\mathcal{X}$, ∂)
2: $hull \leftarrow$ convex hull of $\mathcal{X}$
3: $e_l \leftarrow$ longest edge in $hull$
4: **while** $|e_l| > \partial$ **do**
5: $hull \leftarrow$ REFINE($\mathcal{X}$, $hull$, e_l) ▷ cf. Algorithm 3
6: $e_l \leftarrow$ longest edge in $hull$
7: **return** $hull_{new}$

Algorithm 3: Edge refinement

Require:
$hull$ given as list of edges,
$e_l = (u, v)$ is the longest edge currently in $hull$

1: **function** Refine($\mathcal{X}$, $hull$, e_l)
2: $p \leftarrow$ point closest to e_l ▷ The split point
3: $hull_{new} \leftarrow hull[v \ldots u]$ ▷ Current hull without e_l
4: Append edge $u \ldots p$ to $hull_{new}$ ▷ Insert new edges, that
5: Append edge $p \ldots v$ to $hull_{new}$ ▷ replace e_l.
6: **return** $hull_{new}$

the data generation process). The difference between the two hulls lies only in the choice of the parameter ∂ which is chosen significantly smaller in Figure 3.5. The resulting simplified medial axis however remains nearly the same and does not vary very much. Since the simplified medial axis consists solely of line segments it can act as a cluster skeleton.

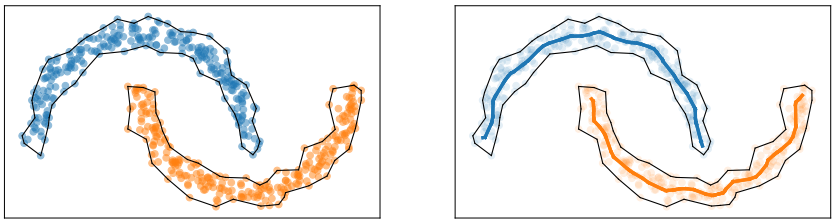


Figure 3.4: Concave hulls (left) and simplified medial axis (right) for two sets of data points whose convex hull would be intersecting each other.

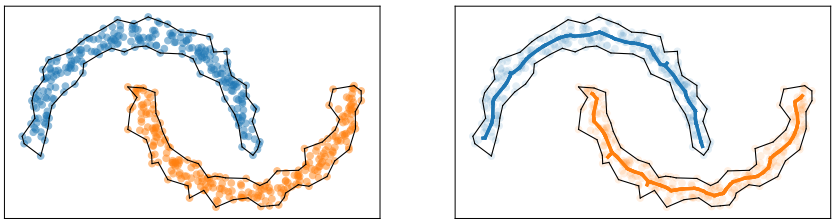


Figure 3.5: Concave hulls (left) and simplified medial axis (right) for the same data as in Figure 3.4 but generated with a lower ϑ. The hull is visibly more ragged, the simplified medial axis however changed only slightly.

With this method and the results already presented in [30, 34] Q1 can now be answered: Yes, for clusters which are not properly representable by a single point another representation in form of the medial axis is—at least in the two dimensional case—possible.

3.3 Cluster Skeletons in Higher Dimensions

On a more general level (i. e., more than two dimensions) the medial axis can neither be computed efficiently (i. e., its runtime complexity is exponential with the number of dimensions) nor would it necessarily be a piecewise linear construct. Since its computation implicitly constructs the Voronoi diagram of the border polygon and the construction of a (simple) Voronoi diagram for points already lies in $O\left(n \log n + n^{\lceil d/2 \rceil}\right)$ the computational effort rises quickly with increasing dimensionality of the data set. In fact the medial axis of a simple rectangular box (used as the hull that induces the medial axis) would not consist of line segments alone. Instead one would have to

deal with medial surfaces as well, since the set of all points containing more than just one nearest neighbor could possibly form a $d-1$-dimensional subspace. Furthermore, the generation of the concave hull requires at least one computation of the convex hull (cf. Algorithm 2). Even the fastest algorithms have a worst-case runtime that grows exponentially with the number of dimensions. Thus—for the general case—other methods that can be used to generate a cluster skeleton are introduced. All of the algorithm have in common, that they only get the data set itself as information. No additional knowledge is required.

In Section 2.2 several different skeletonization and techniques for finding alternate representations have been discussed. Some of these can be seen as inspiration for the methods described in the following.

Although Discrete Curve Evolution has nothing to do with evolutionary algorithms, the term itself sparked the idea, that a curve itself could be developed by an evolutionary algorithm. As such the evolutionary approach in Section 3.3.1 directly constructs a curve representation of the given dataset. The Glow Worm Swarm Optimization (GWSO) approach (presented in Section 3.3.2) on the other hand calculates a local density estimate and tries to find a skeleton along the different local maxima of that cumulated density function. This can be seen as a direct extension of the distance transformation approaches for image skeletonization to continuous spaces. Representing a dataset by a smaller sample is the one of the motivations behind coe sets. In combination with the way how the GWSO finds dense regions in a dataset, the idea was born to represent a single cluster by multiple cluster centers and using a MST on these centers as cluster skeleton. This approach is described in Section 3.3.3 . The alternative prototypes that have already been used in fuzzy clustering sparked the idea, that simpler yet more prototypes could also be used for constructing a cluster skeletons. An approach using that idea is presented in Section 3.3.4. And last but not least an a possible solution for finding cluster skeletons is described in Section 3.3.5 which uses automatic structure learning.

This chapter concludes with a description how the necessary distances between single points and their cluster's skeleton can be calculated and how

that can also lead to an alternative fuzzy membership calculation. Which of the presented methods works best on a controllable set of cluster shapes is validated in Section 4.1.

3.3.1 Evolutionary Approach

The first observation when comparing density- and prototype-based clustering is that clusters can become arbitrarily elongated and twisted (e. g. Figure 1.7). Such clusters could be described by a curve, e. g. a b-spline curve [44]. The advantage of such a representation is that with relatively few components (the control points) complex structures can be created. In [113] one can find a discussion on how such a curve representation can be found by a multi-objective evolutionary algorithm which aims at minimizing the distance of all points to the curve as well as the number of control points.

B-Spline Curve Fitting

There are different types of curve fitting methods that depend on the type of data the curve should be fitted to. Generally, a curve can be easily approximated if the points are ordered or such an ordering can be imposed on the points. The method in [53] can only be applied, if the ordering corresponds to one of the main axes. The dataset shown in Figure 1.7 may still fulfill this condition, but already a full circle could not be approximated this way. If, e. g. the curve winds back and forth, or the data points are placed along a circle, an algorithm that makes use of the flow of points can be used to find a local order. This can then in return be used to apply B-spline regression on the data [95]. However, these algorithm do not work properly, when the curve is self-intersecting at acute angles. The locality used by tangential flow algorithms may miss such intersections and thus represent only parts of the dataset. Another group of algorithm should only be mentioned for the sake of completeness. These expect the dataset to be absolutely noise-free. Thus every point is fit to the curve. Although certainly useful in their own right, these algorithms are not applicable to data analysis when we expect noise to be present.

Given a set of data points. To minimize the distance between all points in the dataset and a B-spline curve $C(t) = \sum_{i=0}^{n} N_{i,k}(t)p_i$ B-spline curve fitting

can be used. $\mathcal{P} = (p_0, p_1, \ldots, p_n)$ is called the set of control points; $\mathcal{N}_k(t) = (N_{0,k}(t), N_{1,k}(t), \ldots, N_{n,k}(t))$ are called B-spline basis functions with degree $k-1$. The knot vector is a non-decreasing sequence of real numbers: $\mathcal{T} = (t_0, t_1, \ldots, t_{n+k})$. A B-spline curve is said to be clamped uniform, if and only if the first and the last knot in the knot vector $\mathcal{T}$ each have multiplicity k and all for all other knots the absolute difference $|t_i - t_{i+1}|$ is constant (i. e., all remaining knots are evenly spaced). A uniform clamped curve has the property that it first and last point coincide with the first and last control point, respectively. Additionaly, a control point only influences the curve in the interval $[t_i, t_{i+k})$ due to the definition of the basis functions (local control property).

In this section a curve approximation approach is used. These algorithms are also known as curve reconstruction or curve extraction algorithms. These algorithms cannot directly calculate the control points. Thus an evolutionary algorithm based on NSGA-II [45] will be used.

Constraints

Out of all possible curves, the algorithm will be limited to a small subset. This makes fitting a curve to the dataset easier for an evolutionary algorithm.

The curves will be constrained to be cubic. I. e., the degree of the basis functions is constantly three. This guarantees C^2 continuity which makes curves visibly smoother [10]. In most cases C^2 continuity is a desired property except when the dataset contains very sharp corners. Here a less smoother curve would be a better fit to the data. Higher degrees of the curve may weaken the local control point property of the B-spline curve. The curve is only influenced within an interval of four knots by a changed control point..

Optimizing the fit of a curve to a dataset is a two-phase process. First in ideal set of control points should be found. Second, an ideal knot vector given the control points should be derived. Since the B-spline curves are uniformly clamped, the latter step can be omitted.

Finally, the output of the proposed algorithm is not a single candidate, but the first non-dominated Pareto front, i. e., a set of solutions. How a single individual is chosen as a solution is described in Section 3.3.1.

Objectives

Minimizing the distance from the B-spline curve to the data allows the curve to properly represent the data and can therefore be seen as one objective that needs to be optimized. However, calculating the distance from a point x to a B-spline curve is a not an easy task. The most common approach is to determine the closest point on the curve by Newton iteration [106]. Another approximative solution is to sample a large enough number of points on the B-spline and choose as the point-to-curve distance the smallest distance between x and any of the sampled points:

$$\begin{aligned} C &= \{C(t/n) \mid 0 \leq t \leq n\} \\ d(x, C(t)) &= \min_{c \in C} \left\{ d(x, c) \right\} \end{aligned} \tag{3.1}$$

Simply minimizing the distance between the curve and the points of the dataset may lead to an interpolation of the data instead of an approximation. For noisy data an interpolation is not desirable (and can be compared to polynomial regression with a polynom of a too high degree). An evolutionary algorithm may be negligent of the minimization goal in favor of a simpler, more generalizing model of the dataset. Properties of a curve that indicate such simplicity may be the number of control points, the total length of the curve, that runs parallel and close to another part of the curve (this would indicate that the same part of the dataset is approximated by the curve twice or more), or the length of segments outside of the dataset. For this section the number of control points is chosen as second objective. The total curve length has been used as an optional third objective for fitness in a later test. All of the three criteria are independent of each other (though more control points can lead to a longer curve easier) and can therefore be used as fitness functions for multi-objective optimization like Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [45]:

$$\mathcal{L}(C, \mathcal{X}) = \begin{pmatrix} |P| \\ \sum_x d(x, C(t)) \end{pmatrix} \tag{3.2}$$

Implementation

The algorithm described in this section has the goal to optimize the control points of a B-spline curve. The number of control points is not known or fixed beforehand. Although a minimum length of four control points can be given (due to the cubic degree of the B-spline), the total number of control points remains variable which favors the use of a vector as encoding structure for the control points. The search space of the algorithm that is induced by this encoding is closed except for the mutation and recombination operator. Both may construct an invalid encoding of the curve by constructing a set of control points that is too small. These corner cases are handled separately in the implementation.

Choosing the *optimal* candidate

NSGA-II only yields the Pareto-optimal candidates identified by the algorithm. These most certainly also contain candidates which only optimize a single objective and are not suitable as a general solution. Two ways of obtaining a single, Pareto-optimal candidate are the hypervolume indicator [136] and an adaptation of Pareto tournaments [73]. Given a (weak) Pareto domination relation $\succeq$ and a solution space $\mathcal{X}$, $x \in \mathcal{X}$ (weakly[1]) dominates $y \in \mathcal{X}$ if and only if $\forall 1 \leq i \leq n : f_i(x) \gtrdot f_i(y)$ (the direction of the relation depends on whether the objectives should be minimized or maximized), where the individual f_i are the different objectives (here: number of control points, overall distance to the spline, and curve length), normalized to the domain $[0, 1]$. A set $A \subseteq \mathcal{X}$ (weakly) dominates $B \subseteq \mathcal{X}$ ($A \succeq B$) if and only if $\forall y \in B \, \exists x \in A : x \succeq y$. With the help of an indicator function a_A

$$a_A(\{z\}) = \begin{cases} 1, A \succeq \{z\} \\ 0, \text{otherwise} \end{cases}$$

the hypervolume indicator I_H^* is defined as the integral over a

$$I_H^*(A) = \int_{(0,\ldots,0)}^{(1,\ldots,1)} a_A(\{z\}) \, \mathrm{d}z$$

[1] Weak domination occurs in case of equality.

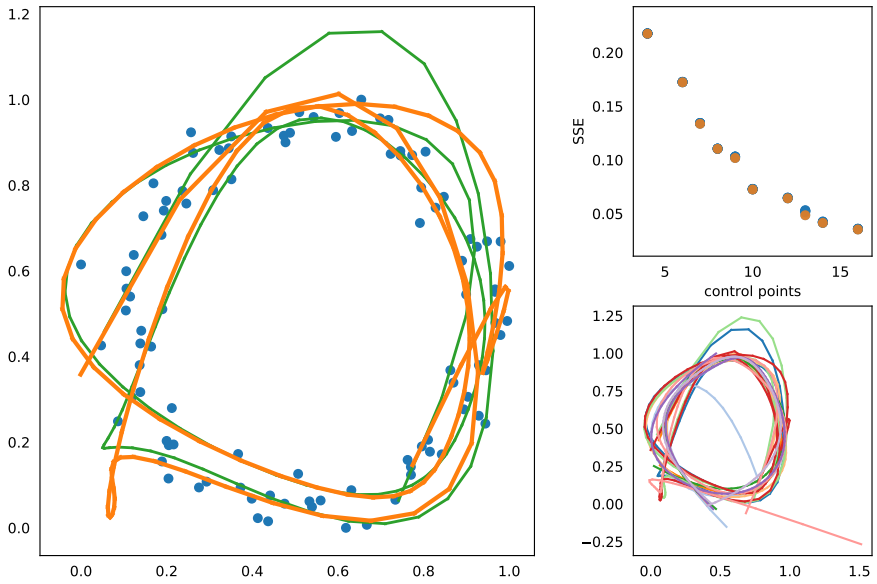


Figure 3.6: Circle data (left; hypervolume-optimal candidate in orange, Pareto-tournament winner in green) to which a set of splines is fitted (lower right). Almost all elements of the population (shown in objective space; upper right) are Pareto-optimal.

The solution x that maximizes $I_H(\{x\})$ is the one that will be accepted as single candidate when using the hypervolume indicator [24]. Please note, that in the case of single instances, there is no need to compute the integral over a but instead it is sufficient (and equivalent) to calculate $\prod_{i=1}^{n} f_i(x)$.

Though strictly speaking a selection operator for genetic algorithms, the Pareto tournament can also be applied to select a single *best* solution out of a set of Pareto-optimal solutions. By choosing two random samples of possible solutions and comparing them objective-wise, the more suitable candidate for reproduction can be found, i. e., the one that *wins* more of the individual comparisons. In this case, *all* pairwise comparisons across all objectives were performed and the single individual that dominated others in the most cases was chosen. One can also imagine this tournament like a soccer league season, where single objectives count as goals and comparisons between individuals are matches, that can be either won, lost or end in a draw. At the end the best candidate is the championship leader.

Figure 3.6 and Figure 3.7 show exemplary results obtained by the evolutionary approach. In both cases the population contained 100 individuals

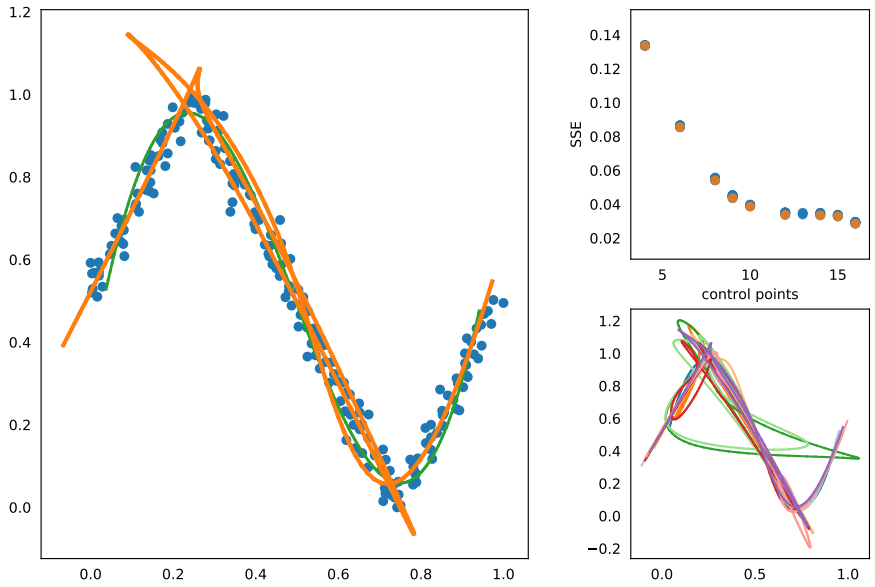


Figure 3.7: Sharp sinusoidal-like data (left; hypervolume-optimal candidate in orange, Pareto-tournament winner in green) to which a set of splines is fitted (lower right). Almost all elements of the population (shown in objective space; upper right) are Pareto-optimal.

per generation and the algorithm ran for 500 generations. In both cases, the hypervolume-optimal candidate (superposed onto the data set as orange curves) overfits the data a lot. Much of their low SSEs can be attributed to them traversing around the data multiple times. The green curves on the same figures show the Pareto tournament winners. These tend to be slightly better (especially in the case of Figure 3.7), but lack overall accuracy. Some of the individuals appear to be a good fit to the data. However, even for these fairly small data sets the runtime for these few generations was too long (approx. 25 minutes for each dataset). With more computing power and a better parallelization implemented, the slowness of the evolutionary approach might be compensated for. Within this thesis this approach cannot be followed any further.

3.3.2 Glow Worm Swarm Optimization

Clusters in general can be defined as regions with increased point density that are separated by regions of decreased density. Finding the regions of maximal density within a complete data set can be done by particle swarm

optimization, as shown in [7]. In that work regions of higher density are discovered by finding the several local maxima of a window density function, which is defined as the cardinality of points lying within a hypercube centered around the points. This is a special case of the uniform kernel, when the underlying support is not spherical (or the maximum norm is used instead of the euclidean). Local maxima in this case indicate clusters if the width of the hyper cube is chosen appropriately. In the general case the kernel density formula can be written as:

$$f_{\mathcal{K},\mathcal{X}} = \sum_{p \in \mathcal{X}} \mathcal{K}(p). \tag{3.3}$$

I. e., the sum of all kernel density estimates (cf. Figure 3.8) centered around each point in $\mathcal{X}$. This yields a function $f_{\mathcal{K},\mathcal{X}}(x)$ that yields a density estimate for any point x that lies in the same space as the original data points. Solving for the roots of the derivation of the window density function $f_{\mathcal{K},\mathcal{X}}$ is usually not feasible since closed form solutions may not exist. As a consequence, maxima have to be found in a different way. Since the density estimate can usually be easily evaluated for any given point, Particle Swarm Optimization (PSO) can be used.

PSO (cf. Algorithm 4, cf. [89]) belongs to the class of metaphorical meta-heuristics. Solution candidates are imagined as particles in $(d + 1)$-dimensional space that aim to find the global optimum of a given function. The particles within a swarm know about the location of the current optimal solution and their own (local) optimum. The movement of each particle depends on these two variables. Each particle moves to some degree toward its own local optimum and towards the known global optimum. Furthermore the movement can also be influenced by inertia, i. e., the movement of the previous iteration(s). PSO is a very powerful heuristic in finding a (possibly local) extremum of a given function.

To actually find a cluster's skeleton, a single extremum in density is not sufficient. Iteratively finding all the extrema could have unwanted side-effects. Instead a modified version of the PSO algorithm will be used. The so-called Glow Worm Swarm Optimization (GWSO) [86] (or Firefly Algorithm [133]) works in principle just as the normal PSO algorithm. But instead of having the same global optimum available to all swarm members,

Algorithm 4: Particle Swarm Optimization

Require:
$\mathcal{X}$ set of n data points p_i,
m number of particles
$\gamma = 0.7298$
$\mathcal{K}$ Kernel function for density estimation
a inertia

1: **function** PSO($\mathcal{K}, m, \gamma$)
2: ▷ the function to be optimized. Here: the sum of all kernels
3: $f \leftarrow \sum_{i=1}^{n} \mathcal{K}(p_i)$
4: Create a swarm $\mathcal{P}$ of m particles.
5: Initialize particle positions in $\mathcal{P}$ with some randomly chosen $x_i^{(0)}$.
6: Initialize all $x_i^{(local)}$ with $-\infty$.
7: **repeat**
8: **for all** particles $p_i \in \mathcal{P}$ **do**
9: ▷ local optimum might have changed, select the one, yielding the highest value for f.
10: $x_i^{(local)} \leftarrow \operatorname{argmax}_{x_i} \{x_i^{(local)}, f(x_i^{(t)})\}$
11: ▷ global optimum is the best of all local optima
12: $x^{(global)} \leftarrow \operatorname{argmax}_{x_i} f(x_i^{(local)})$
13: **for all** particles $p_i \in \mathcal{P}$ **do**
14: $\beta_1, \beta_2 \leftarrow$ uniform random variables in $[0, 1.49618]$
15: ▷ new velocity
16: $v_i^{(t+1)} \leftarrow \gamma \cdot (a \cdot v_i^{(t)} + \beta_1 \cdot (x_i^{(local)} - x_i) + \beta_2 \cdot (x^{(global)} - x_i))$
17: $x_i^{(t+1)} \leftarrow x_i^{(t)} + v_i^{(t)}$ ▷ new position = old position + velocity
18: **until** convergence
19: **return** $x^{(global)}$

the intensity with which a local optimum attracts other particles varies with the currently obtained function value and the distance between the particles. Simplified this means, that particles can only find an extremum in their local neighborhood and are only attracted to these. This difference can be seen by comparing the pseudo-code for PSO and GWSO (Algorithm 5) in lines 10 and 13. For each particle a neighborhood optimum is calculated (from the particles within some distance of the particle itself) and the attraction

Algorithm 5: Simplified Glowworm Swarm Optimization

Require:
$\mathcal{X}$ set of n data points p_i,
m number of particles
$\gamma = 0.7298$
ϵ visibility range
$\mathcal{K}$ Kernel function for density estimation
a inertia

function GWSO($\mathcal{K}$, m, γ)
  ▷ the function to be optimized. Here: the sum of all kernels
  $f \leftarrow \sum_{i=1}^{n} \mathcal{K}(p_i)$
  Create a swarm $\mathcal{P}$ of m particles.
  Initialize particle positions in $\mathcal{P}$ with some randomly chosen $x_i^{(0)}$.
  Initialize all $x_i^{(local)}$ with $-\infty$.
  repeat
    for all particles $p_i \in \mathcal{P}$ **do**
      $x_i^{(local)} \leftarrow \operatorname{argmax}_{x_i} \{x_i^{(local)}, f(x_i^{(t)})\}$
      $x^{(neighborhood)} \leftarrow \operatorname{argmax}_{x_i \in N_\epsilon(p_i)} f(x_i^{(local)})$ ▷ neighborhood optimum
    for all particles $p_i \in \mathcal{P}$ **do**
      $\beta_1, \beta_2 \leftarrow$ uniform random variables in $[0, 1.49618]$
      $v_i^{(t+1)} \leftarrow \gamma \cdot (a \cdot v_i^{(t)} + \beta_1 \cdot (x_i^{(local)} - x_i) + \beta_2 \cdot (x^{(neighborhood)} - x_i))$
      $x_i^{(t+1)} \leftarrow x_i^{(t)} + v_i^{(t)}$
  until convergence
  return $x^{(global)}$

that is calculated per particle incorporates this neighborhood optimum. The chosen values for $\gamma = 0.7298$ and $\beta_1 = \beta_2 = 1.49618$ stem from an empirical performance evaluation made in [51]. The parameter a acts as an inertia dampening factor so that the overall movement tends towards the local and global extrema over time.

What remains is to choose a proper kernel to create the density function. Figure 3.8 shows a choice of some kernels that appear to be suitable or are commonly used for this task. Often a Gaussian (or RBF) kernel is recommended for density estimates but since for large datasets the

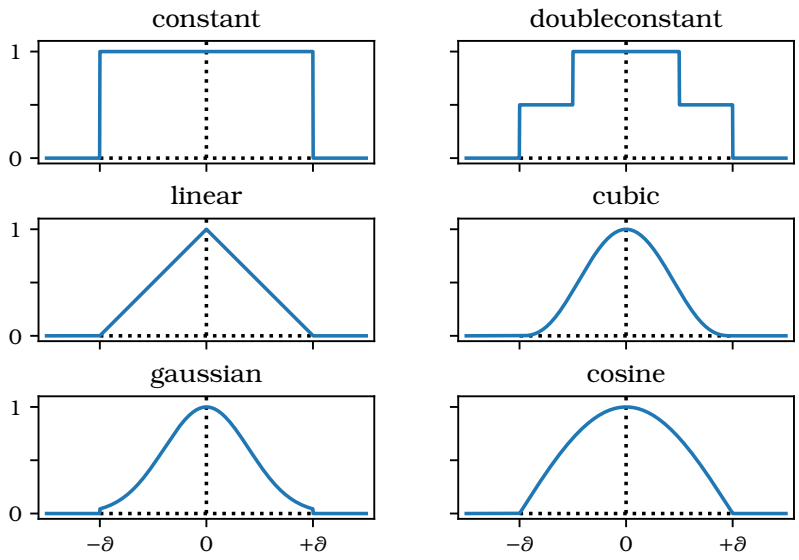
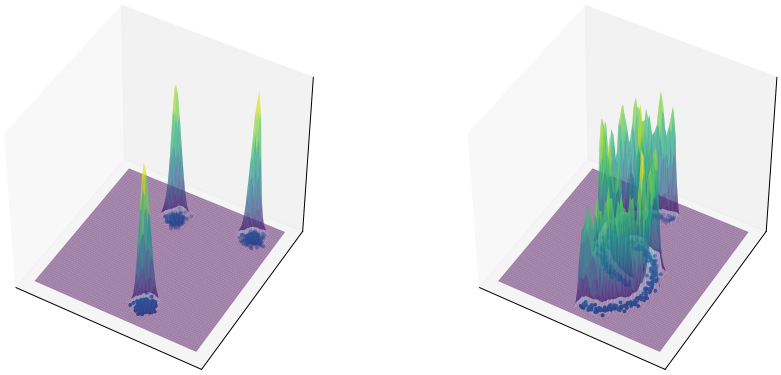


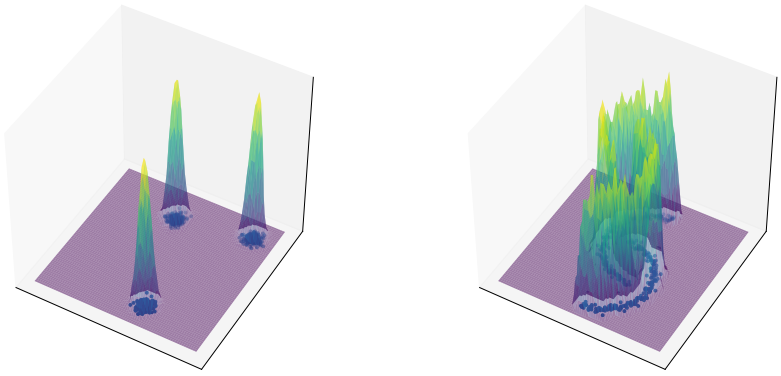
Figure 3.8: Kernels used to define the density function in Algorithm 4 and Algorithm 5.

probability density function of a multi-dimensional normal distribution has to be evaluated n times for each particle and iteration, approximations are more useful. In [108] a cosine approximation is suggested to speed up computation. This, however, can still negatively influence the speed of computation, so a cubic approximation is used. The resulting density function resembles a mountain range and the particles/glowworms are aiming towards the different mountain tops. The effects of choosing either the cubic or the Gaussian kernel can be seen in Figure 3.9a and Figure 3.9b respectively. Since several particles will gather near one local extremum, a simple clustering process that aggregates particles which are very close to each other ($d(p_1, p_2) < \epsilon$) is used to find unique representatives for each extremum. The remaining particles could then be used to create a minimum spanning tree which in turn can act as a generalized centroid.

Actually the process is very sensitive to the choice of parameters so that the out-of-the-box solution does not properly work here. Depending on the dataset itself, the kernel parameter ∂ (cf. Figure 3.8) has to be chosen properly. If it is chosen too small, every point will generate a singular extremum (in the worst case) or only small groups will lead to a density larger than one. On the other hand the initial number of particles, where



a) A dataset with three spherical clusters (left) and a data set with points scattered along to half circles. The z-axis shows the local density function when using a cubic kernel.



b) A dataset with three spherical clusters (left) and a data set with points scattered along to half circles. The z-axis shows the local density function when using a Gaussian kernel.

Figure 3.9: The choice of the kernel function is not as crucial as it might seem.

they are placed, and how far they can see also affects the performance in various ways. Lastly, the faster a decreases the less chance do particles have to move towards an actual extremum and they might get stuck in non-extremal areas of the density function.

For an illustration of the aforementioned effects, see Figure 3.10. Some of the particles have actually found the extrema, while other particles are still trapped in areas where the density function is constant zero. All points here will have the same function value and movement is not guided but totally random. One has to note, that the initial placement of 100 particles was done by a *best candidate sampling* (cf. B.6, [100]), so that the particles are placed in data space with nearly uniform density (as opposed to uniformly distributed sampling, which usually generates clumps of particles). The particles which where too far away from any data point (i. e., those that are almost guaranteed to get stuck in the constant-zero zone outside of the clusters' influence zones) where removed prior to running the optimization. Still some particles did not reach any extreme and so they have to be filtered before generating the skeleton (here: a point was filtered if its function value was less than 10% of the maximal value achieved by any other particle). Out of the remaining ones, not every particle reached a local extremum, but rather got stuck in a non-optimal location. Still, the resulting placement of the particles seems promising enough to pursue this approach.

3.3.3 k-means Skeletonization

Inspired by what some of the internal validation measures consider the best result (especially the R^2 measure, cf. Figures 2.6, 2.7, 2.8, 2.10, and 2.11) one can see that the higher the number of cluster centers in the k-means algorithm is chosen, the more do the clusters align along the clusters' flow. If a proper number k to describe a single cluster can be selected, the connection between the cluster's centers could yield a suitable generalized centroid. The idea to describe a single cluster via multiple cluster centers is not completely new. In e. g. [25] or [93] multi-centers are suggested either as an alternative to density-based clustering or to cope with the problem of imbalanced data distributions.

A general problem with k-means is that its objective function only reaches its true minimum, once k is set to n, i. e., the number of clusters is equal to

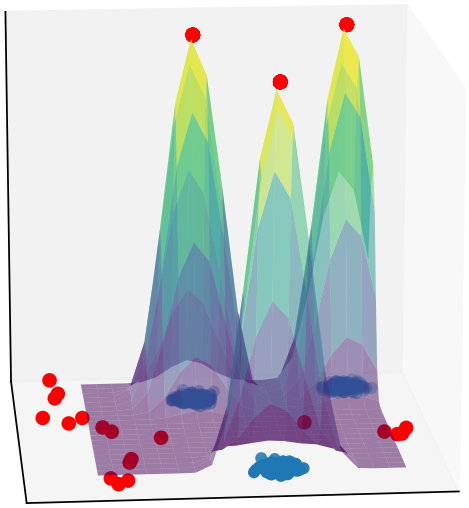


Figure 3.10: A dataset with three spherical clusters. The z-axis shows the local density function when using a cubic kernel, red dots indicate particle positions after the algorithm finished.

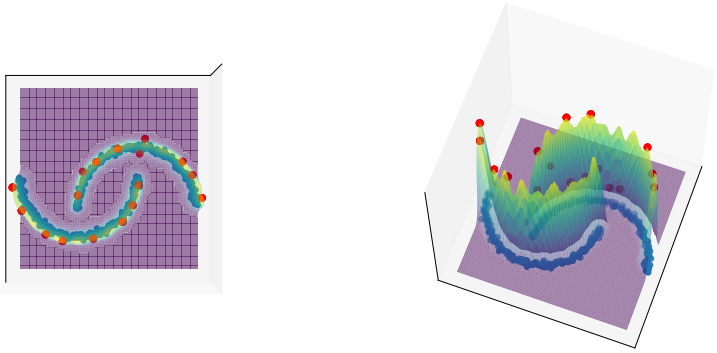


Figure 3.11: A dataset with points scattered along two half-circles and its density function. The red dots mark the particles remaining after filtering. Two different views of the same data set can be seen. Not all extrema are occupied by particles and not every particle actually occupies an extremum.

the number of data points. This makes numerical optimization sub-optimal for selecting a proper k. When choosing k too small, the approximation of the resulting cluster centers would be too coarsely spaced within the cluster, potentially losing important structural details. If k is chosen too large, however, the cluster centers will have only limited space and—similar to the pigeon hole principle—more than one cluster center will describe the same structural detail (cf. Figure 3.12).

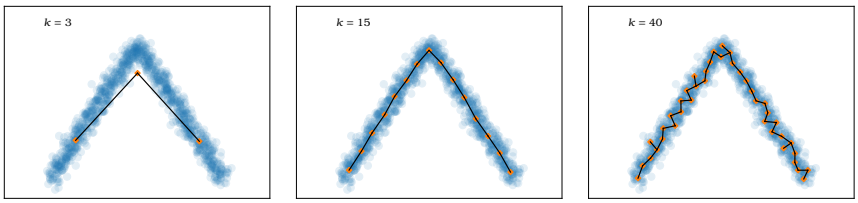


Figure 3.12: A simple dataset, where $k = 3$ (left) cluster centers are not sufficient to let the minimum spanning tree of the cluster centers describe the structure appropriately, $k = 15$ (middle) yields a good structural approximation and $k = 40$ (right) over-adapts to the cluster structure.

Figure 3.12 shows, that linear structures within a cluster can be approximated quite well, if the right parameter is chosen. Although the underlying prototype from which the data was generated contains only two line segments (and three joints), $k = 3$ is not sufficient to restore the structure. In fact approximately $l/2r$ joints per line segment are needed, if the segment has a length of l and a diameter $2r$. This can be easily understood by noting that k-means finds hyper-spherical clusters. Though—in theory—the cluster centers could be placed arbitrarily close together on the structure's generating prototype, local variations of data point placement will shift the centers from the true structure. This leads to an effect similar to the solution of the sphere packing problem [69] or covalent bonds between carbon atoms in organic chemistry [116]. An interesting observation here is, that contrary to the $k = 15$ solution, the angles between adjacent edges are normally not 180° anymore but rather close to 120°. This could—however—lead to a way to decide whether a solution is *good enough* or *bad*.

To achieve a mostly linear model for a cluster, angles between adjacent edges should be close to 180° which means that the arithmetic mean of the angles between two adjacent edges should be small. For sharp edges however there is the need to allow more acute angles as well. Simply checking

whether the average angle is close to 180° will therefore not suffice. Still *most* angles should be kept close to 180°, reducing the standard deviation or the variance of the angles is one of the desired properties of a cluster skeleton as well. Additionally, the arithmetic mean of the angles in the skeleton can be severely skewed by single edges with strongly deviating angles. Thus, the median of the angles will be taken into account as well.

Other objectives that could be considered are the overall distance between the points in the cluster and the cluster's skeleton or the number of joints in the skeleton. This first objective is monotonically decreasing as the number of skeleton joints (cluster centers) increases. The two are easily just different formulations of the same objective—but with opposite signs. If the number of joints is increased, more edges will be present in the cluster skeleton and with more edges, the sum of distances between points and edges decreases (as in the standard k-means algorithm). While both the sum of distances (better fit) and the number of joints (simpler model) should be minimized, this conflict seems to be unsolvable for such directly related objectives. Due to this only the mean, median, and the standard deviation (or variance) will be considered at this time. This turns the choice of the proper, best solution into a multi-objective optimization problem (cf. Section 3.3.1).

The complete procedure to obtain a skeleton via the k-means-based skeletonization can be seen in Algorithm 6 while Figure 3.13, Figure 3.14, Figure 3.15, and Figure 3.16 show the resulting skeletons for a variety of different cluster shapes.

3.3.4 Fuzzy-c-Line Segments (FCLS)

A clear extension of the k-means skeletonization approach is loosely based on the several different fuzzy shell clustering algorithms available, but especially inspired by the volume prototype clustering introduced in [82]. Volume prototypes extend the notion of a point prototype, to a hyper-spherical prototype (in the case of [82] also to ellipsoidal prototypes by adjusting the space-defining metric accordingly).

The centers used in the fuzzy-c-means algorithm (cf. Section 1.3.2) are no longer restricted to have zero radius, but instead can grow to include some points of the cluster. Fuzzy membership is set to 1.0 if and only if a

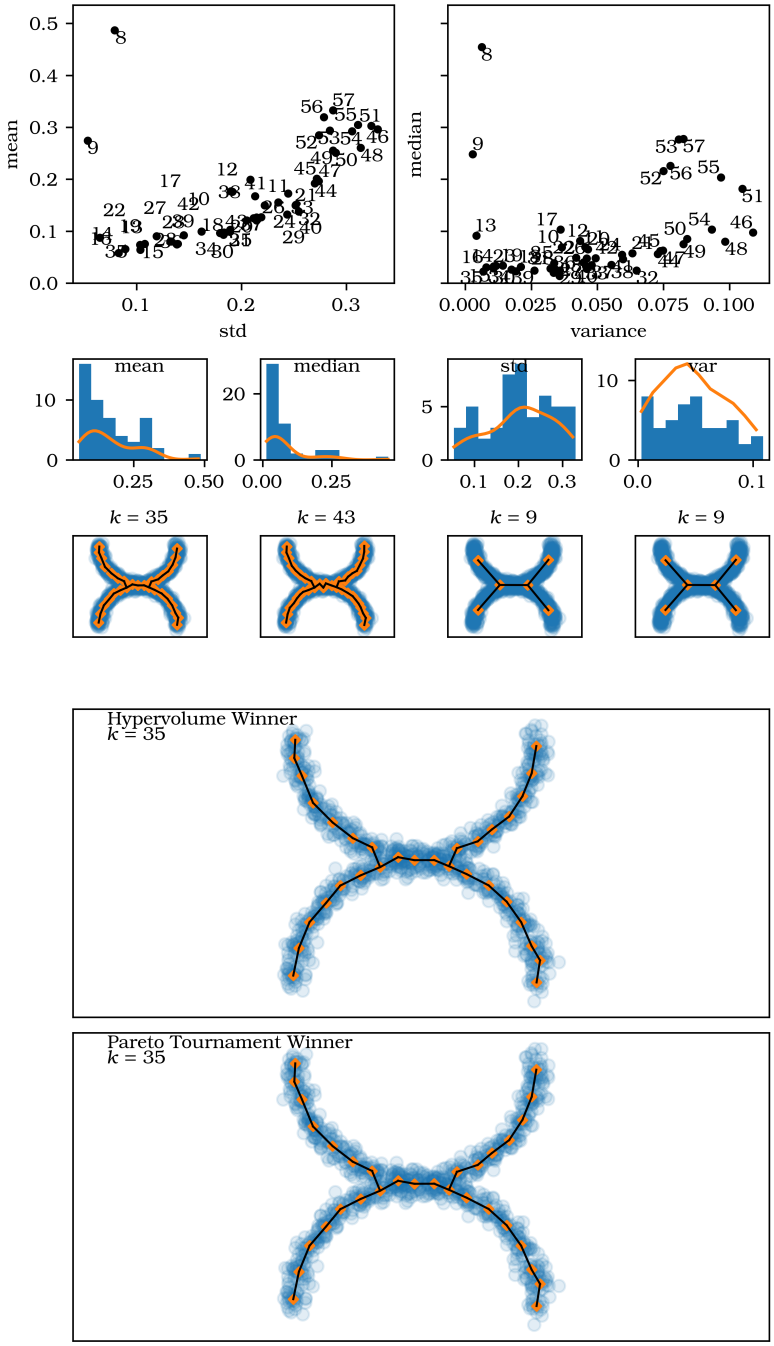


Figure 3.13: Skeletons obtained with the k-means skeletonization method. Top row shows the candidates in objective space, middle rows show the distribution of objective values per objective and the optimal candidate given each single objective. Bottom shows the winning candidates when using the hypervolume indicator or the Pareto tournament to determine a single best solution respectively.

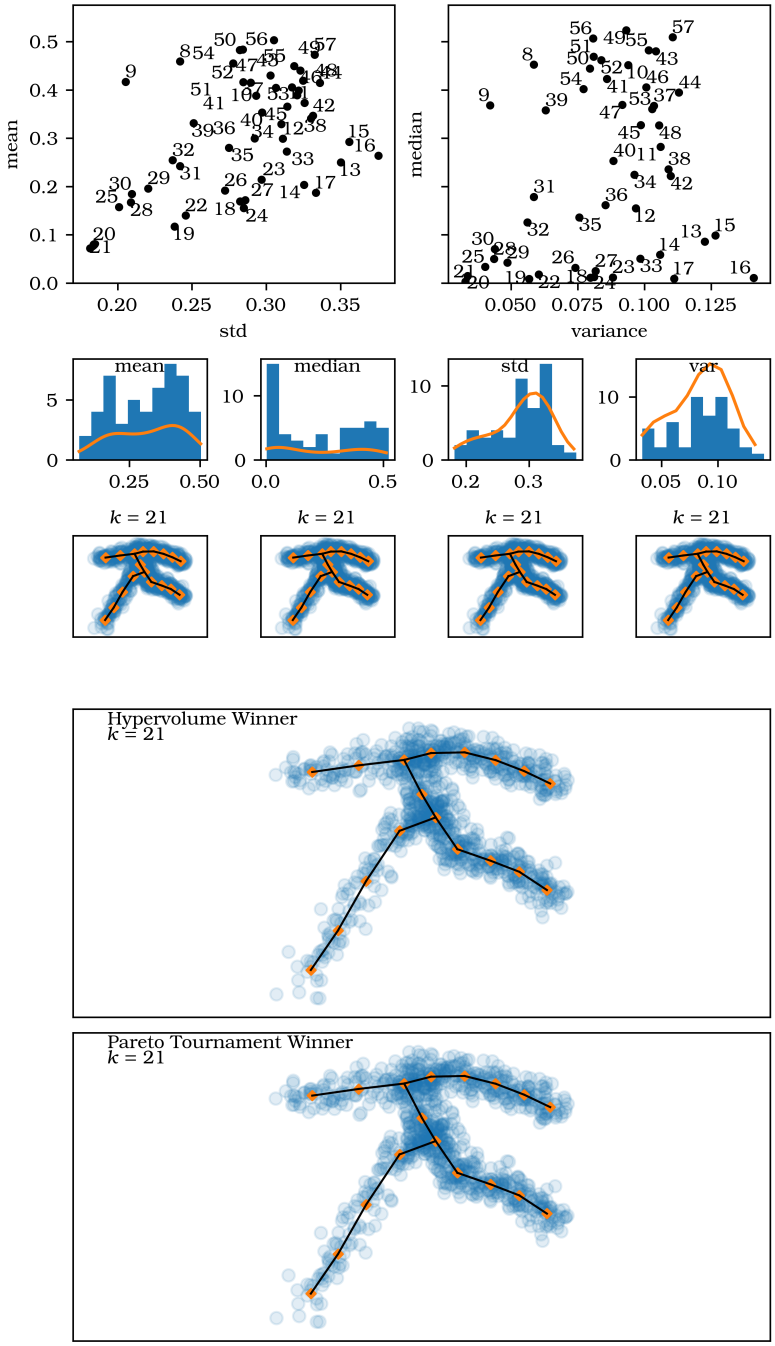


Figure 3.14: Skeletons obtained with the k-means skeletonization method. Top row shows the candidates in objective space, middle rows show the distribution of objective values per objective and the optimal candidate given each single objective. Bottom shows the winning candidates when using the hypervolume indicator or the Pareto tournament to determine a single best solution respectively.

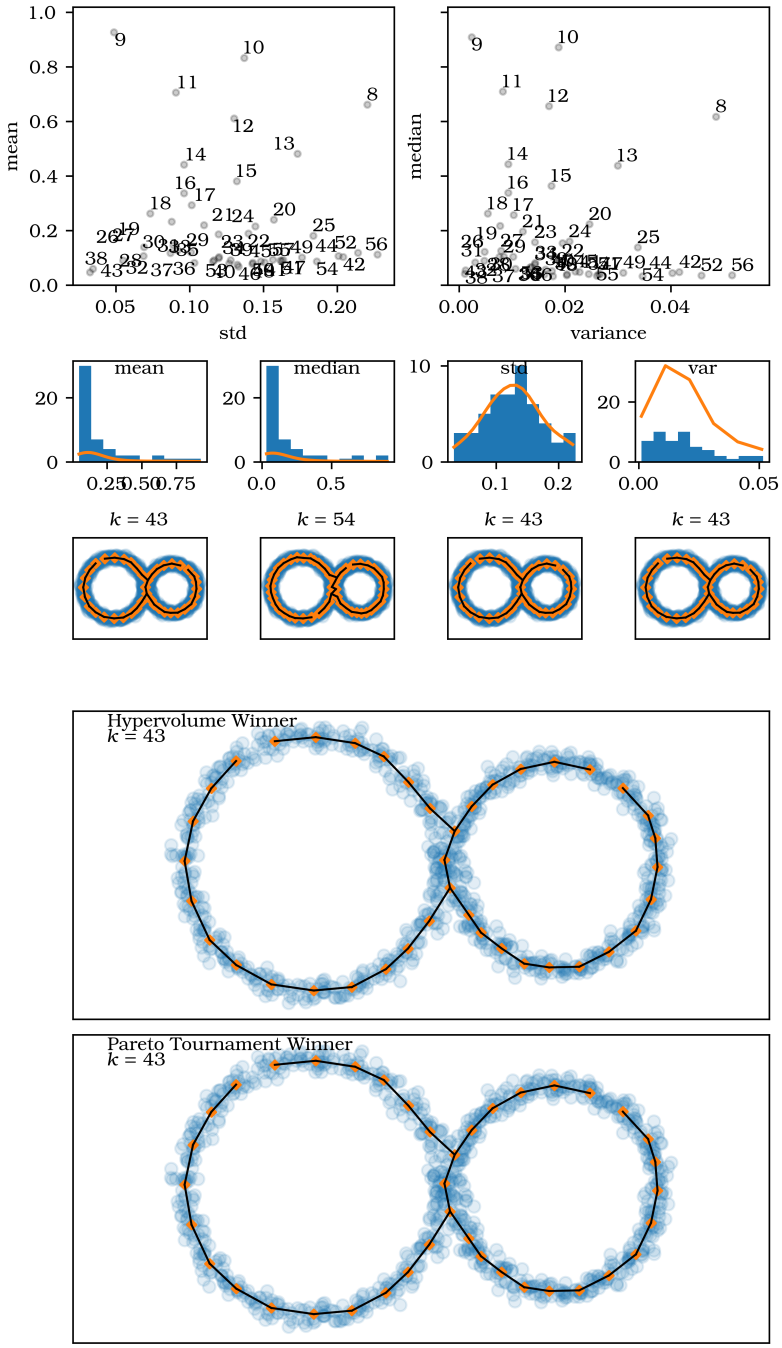


Figure 3.15: Skeletons obtained with the k-means skeletonization method. Top row shows the candidates in objective space, middle rows show the distribution of objective values per objective and the optimal candidate given each single objective. Bottom shows the winning candidates when using the hypervolume indicator or the Pareto tournament to determine a single best solution respectively.

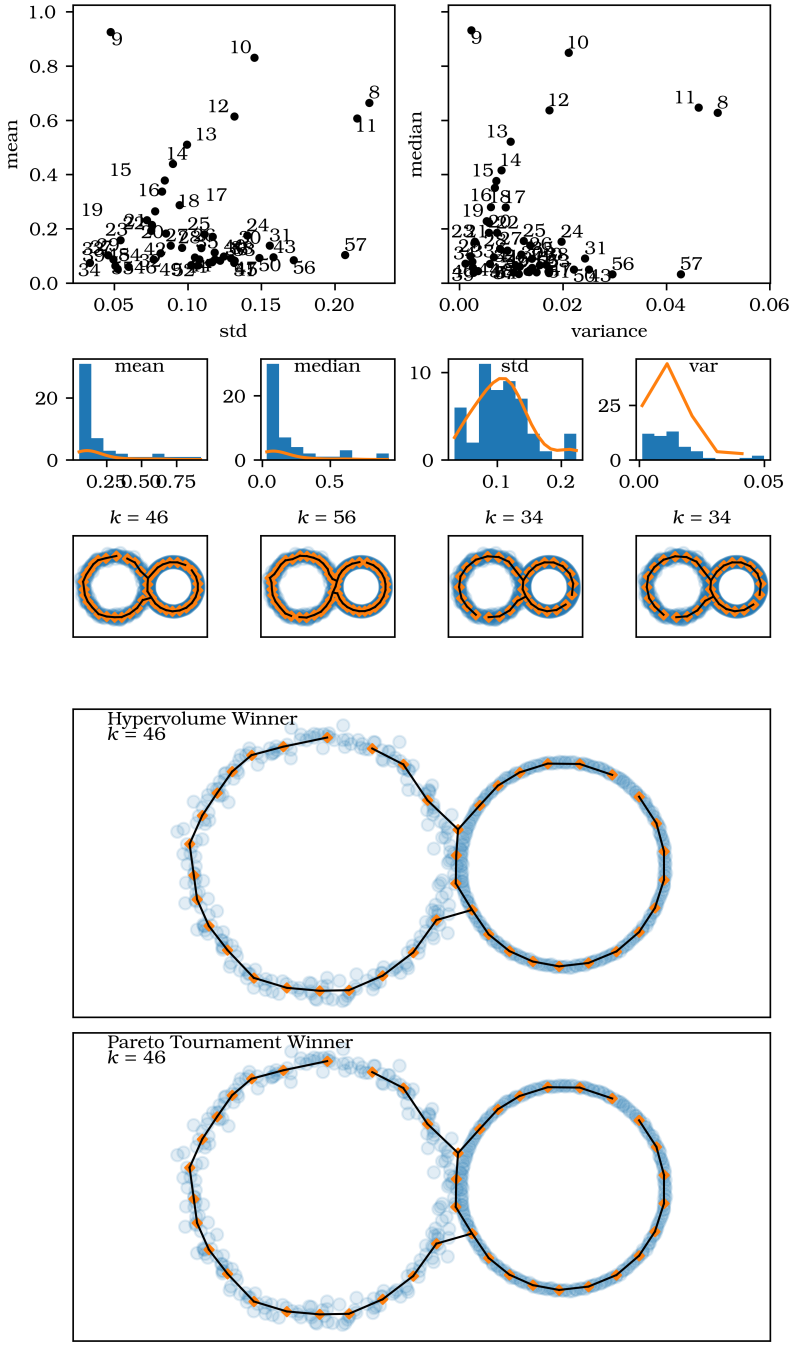


Figure 3.16: Skeletons obtained with the k-means skeletonization method. Top row shows the candidates in objective space, middle rows show the distribution of objective values per objective and the optimal candidate given each single objective. Bottom shows the winning candidates when using the hypervolume indicator or the Pareto tournament to determine a single best solution respectively.

Algorithm 6: K-Means-based Skeletonization

Require:
$\mathcal{X}$, a set of n data points p_i,
$\mathcal{T}$, a function to choose the *optimal* solution

function ANGULAR_INFORMATION(mst)
  $\mathcal{A} \leftarrow$ empty list
  for all adjacent edges e_1, e_2 in mst **do**
    Interpret e_1 and e_2 as vectors
    $a \leftarrow$ cosine distance between e_1 and e_2 ▷ $d_{cos}(x, y) = 1 - \frac{x \cdot y}{|x| \cdot |y|}$
    if $a > 1$ **then**
      $a \leftarrow |a - 2|$ ▷ $a \in [0, 1]$, a is now orientation-independent
    Append a to $\mathcal{A}$.
  return median($\mathcal{A}$), mean($\mathcal{A}$), std($\mathcal{A}$)

function k-MEANS-SKELETONIZATION($\mathcal{X}$, start, end)
  $\mathcal{A} \leftarrow$ empty list
  for $k \in [start, end]$ **do**
    $C = c_1, \ldots, c_k \leftarrow K$-means($\mathcal{X}, k$)
    mst $\leftarrow$ MST on the complete graph formed by C
    $\tilde{a}, \bar{a}, s_a \leftarrow$ ANGULAR_INFORMATION(mst)
    Append $(\tilde{a}, \bar{a}, s_a, \text{mst})$ to $\mathcal{A}$.
  Let mst^* be the MST of the best solution determined by $\mathcal{T}(\mathcal{A})$.
  return mst^* ▷ best skeleton

point is contained within the hypersphere (i. e., if its distance to the center is smaller than or equal to some r). Outside of the sphere the standard fuzzy membership calculation can be applied.

If prototypes within the fuzzy-c-means algorithm need not be restricted to points, one could also look for line segments as cluster prototypes as well. Theoretically the process of finding clusters with line segments works much like the fuzzy-c-means algorithm with only one change: After the update of the membership degrees, the lines' starting and ending points have to be determined.

To do this a (weighted) principal component analysis can be used. The first eigenvector obtained this way points into the direction of the largest variance within each cluster segment and could therefore be used as part of

the cluster skeleton if extended from the cluster segment's center in both directions. Since the length of each eigenvector corresponds to the variance in that direction, the vectors should be scaled to represent the standard deviation rather than the variance. This ensures that the line segments do not extend into regions outside of the cluster it represents.

The general procedure is given in Algorithm 7

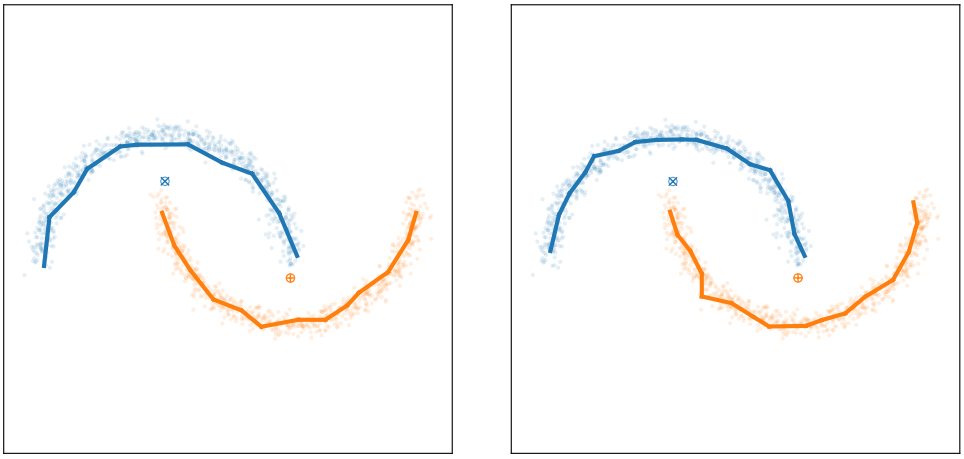


Figure 3.17: Resulting skeletons for the `moons` data set. Especially the blue (upper) cluster shows a slight shift towards the cluster's barycenter originating from the fuzzy process. $\omega = 1.05$ for the left figure, $\omega = 3$ for the right figure.

3.3.5 Kohonen Maps & Growing Neural Gas

A completely different approach to the ones previously presented falls into the class of automatic structure learning algorithms. Self-organizing Maps (SOMs) or Kohonen Maps [85]) deform a given neural network (the weight vectors associated with the neurons which represent the mapping location) so that it represents the topology of the target space. Since the output of a SOM depends on the initial network that unfolds onto the target space, a similar form of network will be considered here. This network starts with two randomly placed neurons and grows along the data: The Growing Neural Gas (GNG) [63] (cf. Algorithm 8).

This algorithm iteratively changes the size and shape of the initial graph supplied. Starting from a simple line the error (in form of distance between

Algorithm 7: Fuzzy-c-Line Segments

Require:
$\mathcal{X}$, a set of n data points x_i,
c, the number of clusters to find,
ω, the fuzzifier

function FUZZY c LINE SEGMENTS($\mathcal{X}$, c, ω)
  Initialize cluster prototypes p_i as points. ▷ start = end
  Calculate a $c \times n$ distance matrix (d_{ij}) between centers and points.
  Calculate membership degrees (u_{ij}) according to Equation 1.3.
  repeat
    for all clusters i **do** ▷ weighted PCA
      Calculate μ_i according to Equation 1.4.
      Subtract μ_i from each data point ▷ center data
      $W \leftarrow u_{i*} \cdot 1_c$ ▷ Turn membership degrees into diagonal matrix
      $A \leftarrow \mathcal{X}^T_{centered} \cdot W \cdot \mathcal{X}_{centered}$
      $R \leftarrow \frac{A}{\sum_{i=1}^{c} W_{ii}}$ ▷ Normalize
      Calculate largest eigenvalue λ and the its eigenvector e of R.
      $p_i \leftarrow (\mu_i - \sqrt{\lambda} \cdot e, \mu_i + \sqrt{\lambda} \cdot e)$ ▷ start ≠ end
      Calculate a $c \times n$ distance matrix (d_{ij}) between cluster prototypes (line segments) and data points.
      Calculate membership degrees (u_{ij}) ▷ Equation 1.3.
  until convergence

neurons and the data points) for the initial neurons is calculated. For each training sample the two closest neurons s_1, s_2 and all the direct neighbors of s_1 are located. All neighbors (in the graph structure) of s_1 are moved closer to the training sample (by a fraction of their respective distance). All edges starting or ending in s_1 increase their age except the edge between s_1 and s_2 (should it exist). If s_1 and s_2 were not connected, an edge between these two is added. The rationale behind this is that neurons that both could be used to represent a training point should also be connected. If there are edges that are *too old*, they are removed from the graph.

Whenever a certain amount of iterations has passed, the neuron n with the highest accumulated error is located. From all of this neurons neighbors the one m with the highest error is located as well and a new neuron

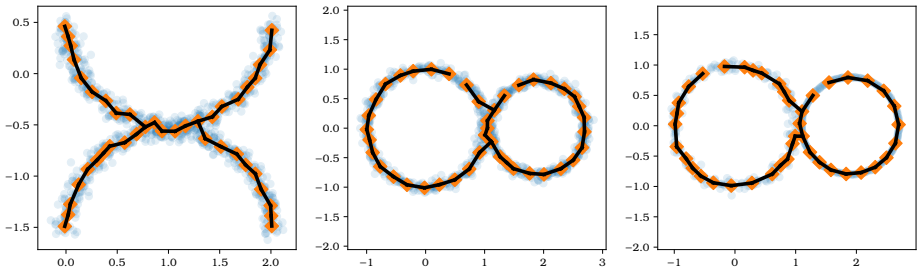


Figure 3.18: Results of the growing neural gas algorithm for some data sets. The resulting structures resemble the original data quite well although the density of the skeletons' nodes is unevenly distributed across the structure.

is inserted in the middle of n and m. After that, all accumulated errors are decreased and the procedure starts from the beginning. After some convergence criterion is met, the algorithm terminates and the final graph structure is returned.

Because randomly selected neurons move towards their neighbors, they tend to end in areas of high density. New neurons are inserted where the current graph structure does not represent the underlying data structure very well. Removing too old edges happens where the two connected nodes are no longer the two closest neighbors of a data point and eases the graph structure. The algorithms adaptability to local changes in the data density actually also allows it to find structure in non-static data.

Figure 3.18 shows the result of the skeletonization process based on the neural gas algorithm. The structures are well-preserved although the GNG's neurons are unevenly distributed across the complete structure. The parameters used here are the default parameters suggested in [63].

3.3.6 Summary

In this section several different ways of obtaining a cluster skeleton for a cluster have been described and explained. The methods originate from different areas and are partially inspired by the skeletonization methods known from image recognition.

The evolutionary approach for curve approximation was inspired by DCE and approximates a b-spline to the dataset. However, it is rather slow and

could therefore not be used for the final validation. Due to the long computing time too few generations could be iterated and this—of course—reduces the quality of the results.

The GWSO mimics the ridge detection of the distance transformation approaches by using a multi-modal swarm optimization. The choice of the correct kernel and its parameters for density optimization and—in consequence thereof—the proper convergence of this method make it difficult to use it in a parameter-free setting.

The k-means-based skeletonization finds cluster skeletons by using multi-objective optimization on small set of objectives that are geometrically motivated for cluster skeletons. It can be parallelized well and requires no additional parameters for finding well-fitting skeletons. The process itself tests several clusterings of a single cluster and can therefor be seen as an attempt to learn a sparser representation (or some kine of *core set*) of the cluster. A Minimum Spanning Tree on the cluster centers is then used as the cluster skeleton

The Fuzzy-c-Line Segments works in a similar fashion but uses line segments as cluster prototypes. Those can represent linear segments within a cluster already better than the centers used in the k-means-based approach. Due to it being a fuzzy clustering algorithm it is prone to suffering from the curse of dimensionality.

The Growing Neural Gas automatically learns the structure of the cluster by placing neurons inside the data. The update process is very similar to SOMs but with the benefit that no initial network structure has to be supplied. Instead the network structure is learned during the process.

With the results from Section 3.3 Q2 can be answered: The medial axis itself cannot be extended to higher dimensions easily or with the needed efficiency. However, approximations of the medial axis can be found in several different ways.

In the next chapter these methods will be evaluated and analyze which one should be used for the validation of density-based clustering.

3.4 Alternative Distance Calculation and Modified Fuzzy Membership Calculation

The motivation behind finding the cluster skeletons was—in the first place— to enable the centroids-based validation measures for the use-case of density-based clustering. Equation 2.6 or Equation 2.20, for example, use the distance between the cluster center μ and each individual data point x_i to calculate their scores. Equation 1.3 uses the distance to the clusters' centers to calculate the fuzzy membership.

3.4.1 Alternative Distance Calculation

Since the generalized centroids or cluster skeletons are no longer simple points from the same space as the data points but more complex structures, using the simple euclidean distance will not suffice anymore. Instead one has to use the distance between each data point individually and the edges that make up the skeleton.

Definition 3.9 (*Distance between a point and a cluster skeleton*)

The distance between a point x and a cluster skeleton $\mathcal{S}_C = \{e_1, \ldots, e_n\}$ is the smallest distance between x any edge contained in $\mathcal{S}_C$. Formally the distance is defined as

$$d(x, \mathcal{S}_C) = \min_{e \in \mathcal{S}_C} \Big\{ d(x, e) \Big\}.$$

The process of finding the nearest edge to which the actual distance should be calculated could be sped up by using the Voronoi diagram of the skeleton (i. e., its edges). By finding the Voronoi cell one point belongs to the corresponding edge could be easily found. However, calculating the Voronoi diagram for points is already in $O\left(n \log n + n^{\lceil d/2 \rceil}\right)$ and thus infeasible for high-dimensional spaces. Instead one has to fall back to the brute force solution for now.

The distance between a point and an edge (as part of a straight line can be easily computed by projecting the point x onto the line e defined by its start point v and end point w. Imagining that the edge e extends in both directions infinitely (i. e., e is the supporting vector of a line), then this line

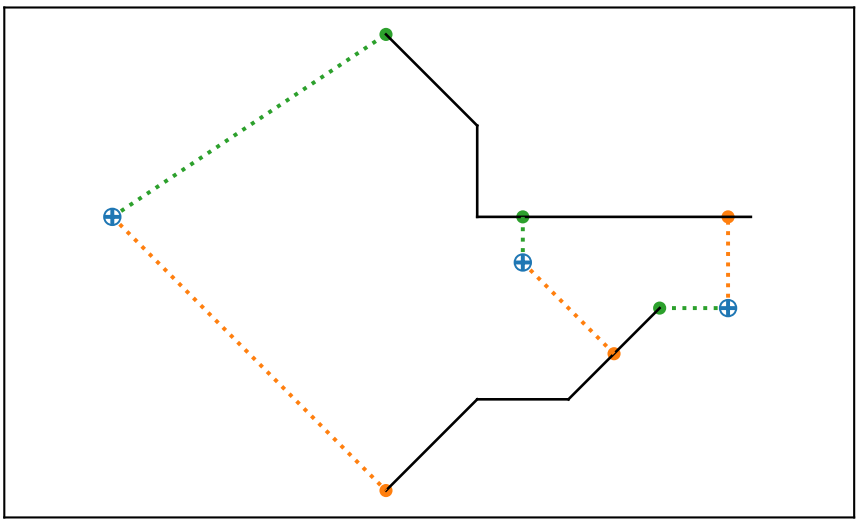


Figure 3.19: Illustration of the distance calculation for points and skeletons. For each point (blue) the corresponding shortest vector is drawn as a dotted line and the closest point on either skeleton is marked in green.

can be described parametrically as $l(t) = v + t \cdot (w - v), \forall t \in \mathbb{R}$. The orthogonal projection of x onto l is given by $t = \frac{((x-v)\cdot(w-v))}{|w-v|^2}$. Should $t \notin (0,1) \subset \mathbb{R}$ hold (i. e., the foot point does not lie on the line segment), then the shortest distance between x and e is the distance from x to v (if $t \leq 0$) or to w (if $t \geq 1$). Otherwise the shortest distance between x and e is $d(x, l(t))$. Please note that this projection is independent of the dimensionality of the underlying vector space and this the end points defining the edge. It can therefor be used to calculate the distance between points and cluster skeleton in arbitrary dimensionality.

This definition of distance between a point and a cluster's skeleton can replace the distance $d(x, \mu)$ between a point and a cluster's centroid in (almost) all cluster validation measures.

Some validation measures also require the distance between two centroids (cf. Equation 2.15, the Calinski-Harabasz Index (CH) on Page 33, or the

Xie-Beni Index (XB) on Page 34). The distance between two skeletons can be calculated with a rather simple procedure:

Definition 3.10 (*Distance between two cluster skeletons*)

The distance between a two cluster skeletons (or inter-cluster distance) $\mathcal{S}_C^1$ and $\mathcal{S}_{C'}^2$ is given by the smallest distance between any pair of edges $(e_1, e_2) \in \mathcal{S}_C^1 \times \mathcal{S}_{C'}^2$. Formally the distance between two cluster skeletons is defined as

$$d(\mathcal{S}_C^1, \mathcal{S}_{C'}^2) = \min_{e_1 \in \mathcal{S}_C^1} \left\{ \min_{e_2 \in \mathcal{S}_{C'}^2} \left\{ d(e_1, e_2) \right\} \right\},$$

where each $d(e_1, e_2)$ is calculated according to Algorithm 9.

Algorithm 9 [52] describes how the distance between two line segments in arbitrary dimensionality can be calculated. The algorithm simultaneously find the foot point on both line segments (similar to the point-to-line-segment distance described earlier in this section) and then calculates the euclidean distance between these two as the (shortest) distance between the two line segments.

The distance presented in Definition 3.10 is conceptually completely different from the modified Feéchet distance for trees presented in Definition 3.8. While the latter computes the similarity between to complete skeletons (or trees) $d(\mathcal{S}_C^1, \mathcal{S}_{C'}^2)$ computes the shortest distance between any two points that are part of the skeleton.

Algorithm 8: Growing Neural Gas

Require:
$\mathcal{X}$, a set of n data points x,
i, maximum number of iterations,
e_b, learning rate for best neurons,
e_n, learning rate for neighboring neurons,
λ, an iterator step-length,
a, local decay rate,
d, global decay rate

function GNG($\mathcal{X}, i, e_b, e_n, \lambda, a, d$)
  repeat
    for all x in random order **do**
      $s_1, s_2 \leftarrow$ two closest neurons to x
      for all edges e connected to s_1 **do**
        $e.age \leftarrow e.age + 1$ ▷ edge has been used, it grows older
      $s_1.error \leftarrow s_1.error + d(s_1, x)$
      $v \leftarrow x - s_1$
      $s_1 \leftarrow s_1 + e_b \cdot v$
      for all neighbors s_n of s_1 **do**
        $v \leftarrow x - s_n$
        $s_n \leftarrow s_n + e_n \cdot v$
      if $\exists e = (s_1, s_2) \in E$ **then**
        $e.age \leftarrow 0$
      else
        $E \leftarrow E \cup \{e\}$
      for all edges $e \in E$ **do**
        if $e.age > max - age$ **then**
          Remove e from E
          if Any neuron n has no edge anymore **then**
            remove n from gas
      if number of iteration $\mod \lambda \cong 0$ **then**
        $q \leftarrow \text{argmax}_n\, n.error$
        $f \leftarrow \text{argmax}_{m \in neighbors(n)}\, m.error$
        insert new neuron n halfway between q and f
        connect n with q and f
        remove (q, f) from E
        $q.error \leftarrow q.error \cdot a$
        $f.error \leftarrow q.error \cdot a$
        $n.error \leftarrow q.error$
      for all neurons n **do**
        $n.error \leftarrow n.error \cdot d$
  until required number of iterations i

Algorithm 9: Segment-Segment-Distance

Require:
$e_1 = (p_1, q_1), e_2 = (p_2, q_2)$, edges defined by start and end points p_i, q_i

function SSD(p_1, q_1, p_2, q_2)
  $u \leftarrow q_1 - p_1$
  $v \leftarrow q_2 - p_2$
  $w \leftarrow p_1 - p_2$
  $a, b, c, d, e \leftarrow u \cdot v, u \cdot v, v \cdot v, u \cdot w, v \cdot w$
  $D \leftarrow ac - b^2$
  $s_c, s_N, s_D \leftarrow D$
  $t_c, t_N, t_D \leftarrow D$
  if $D < \varepsilon$ **then** ▷ Lines are almost parallel
    $s_N, s_D, t_N, t_D \leftarrow 0.0, 1.0, e, c$
  else ▷ Assume infinite lines and get closest point
    $s_N, t_N \leftarrow be - cd, ae - bd$
    if $s_N < 0.0$ **then**
      $s_N, t_N, t_D \leftarrow 0.0, e, c$
    else if $s_N > s_D$ **then**
      $s_N, t_N, t_D \leftarrow s_D, e + b, c$
  if $t_N < 0.0$ **then** ▷ Adjust line parameters to cope for non-infiniteness
    $t_N \leftarrow 0.0$
    if $d > 0.0$ **then**
      $s_N \leftarrow 0.0$
    else if $d < a$ **then**
      $s_N \leftarrow s_D$
    else
      $s_N, s_D \leftarrow -d, a$
  else if $t_N > t_D$ **then**
    $t_N \leftarrow t_D$
    if $b - d < 0.0$ **then**
      $s_N \leftarrow 0.0$
    else if $b - d > a$ **then**
      $s_N \leftarrow s_D$
    else
      $s_N, s_D \leftarrow b - d, a$
  if $|s_N| < \varepsilon$ **then**
    $s_c \leftarrow 0.0$
  else
    $s_c \leftarrow {}^{s_N}/_{s_D}$
  if $|t_N| < \varepsilon$ **then**
    $t_c \leftarrow 0.0$
  else
    $t_c \leftarrow {}^{t_N}/_{t_D}$
  $d_P \leftarrow w + (s_c \cdot u) - (t_c \cdot v)$ ▷ shortest vector between line segments
  return $\|d_P\|$ ▷ Length of shortest vector

3.4.2 Density-based Fuzzy Membership

With the alternative in calculating the distance between a cluster skeleton (in lieu of the center point) and the cluster's points the same term can be replaced within the calculation of fuzzy membership degrees (see Equation 1.3). The resulting fuzzy partition will seem much more natural (given the arbitrary cluster shapes usually encountered in density-based clustering) than those obtained ordinarily. An exemplary fuzzy partitioning for two clusters is visualized in Figure 3.20; one for three clusters in Figure 3.22. The color encoding on the picture shows the degree of membership to either of the two clusters. The whiter the color is the more ambiguous is the membership assignment (pure white meaning that a point lies directly between two skeletons, i. e., $d(x, \mathcal{S}_{C_1}) = d(x, \mathcal{S}_{C_2})$ and thus $\mu(x) = (0.5, 0.5)^T$). Once compared with Figure 3.21 it becomes obvious, that the partitioning obtained with respect to the cluster skeleton resembles the original data better than the centroid-based version.

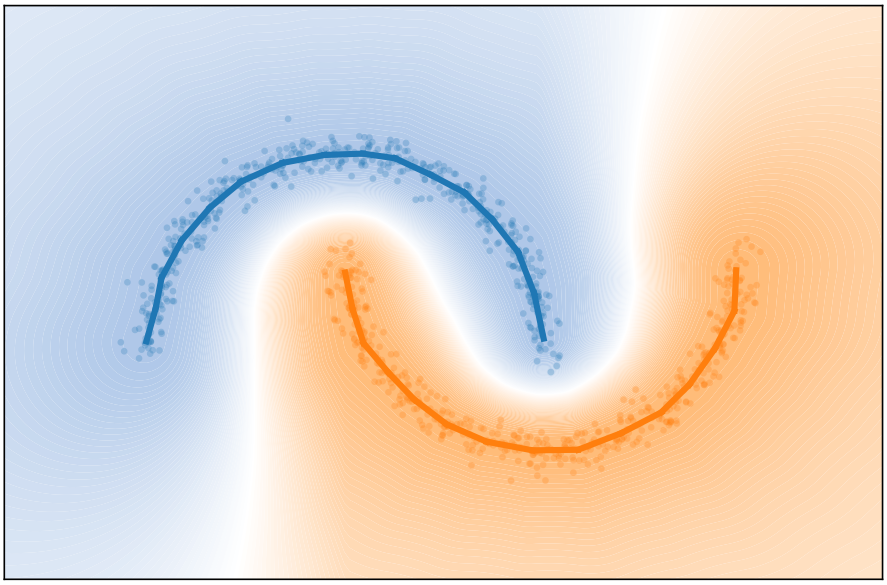


Figure 3.20: `halfcircles` dataset with $n = 1000$ and $\sigma = 0.05$. The resulting fuzzy partitioning is obtained with $\omega = 2$ and with respect to the clusters' skeletons.

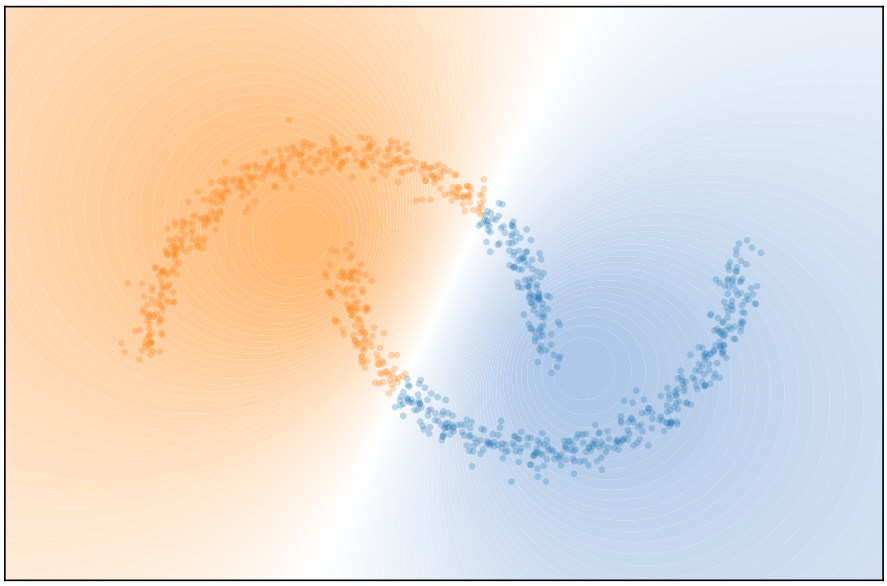


Figure 3.21: `halfcircles` dataset with $n = 1000$ and $\sigma = 0.05$. The resulting fuzzy partitioning is obtained with $\omega = 2$ and with respect to the clusters' center points.

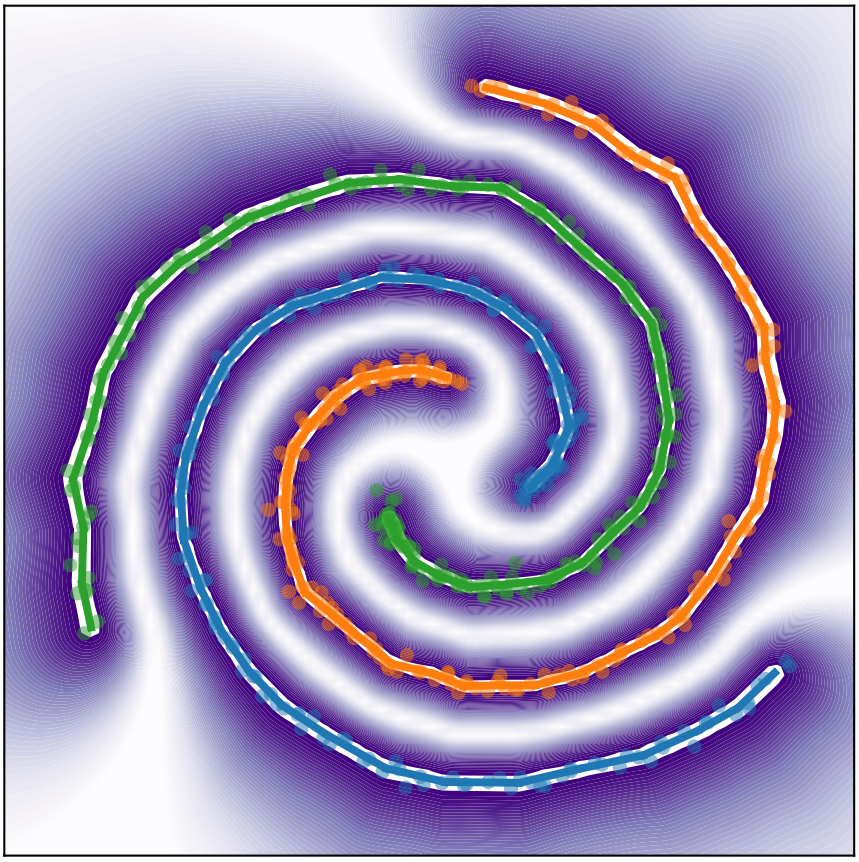


Figure 3.22: `spirals` dataset with $n = 624$ and $\sigma = 0.25$. The resulting fuzzy partitioning is obtained with $\omega = 2$ and with respect to the clusters' skeletons. The purple contour only encodes the membership to the closest cluster.

CHAPTER 4

Validation & Experiments

Within this chapter the previously described methods that can be used to obtain a cluster skeleton in any arbitrary dimension are validated against some useful and desired properties. Especially the results for high dimensional data sets are interesting, since no other skeletonization method is supposed to work on more than three dimensions. For this several different clusters with known and controllable characteristics were generated. Using the Modified Fréchet Distance derived in Section 3.1.2 the produced skeletons can be compared against the known ground truth. After this one method will be chosen for the next part of the experiments.

In the second part of this chapter the chosen method will be used to automatically detect proper parameters for DBSCAN. By using a large number of datasets different labelings can be obtained for DBSCAN (parameterized with the help of the chosen skeletonization method) and k-means as a easy-to-parameterize competitor. Each resulting labeling is then compared against the ground truth (known from the dataset generation) using an internal cluster validation measure. Both methods can then be compared with each other and it can be seen which produces better results. Without the use of such a ground truth-based validation it would be hard to assess the true quality of the clusterings obtained. Even the direct comparison of labelings of two different algorithms could only give a relative view on the result, e. g. *result A is similar to result B*, but one still would not know whether A or B are good or bad results.

4.1 Skeletonization Methods

Almost all of the methods presented in Section 3.3 are capable of generating a generalized centroid by creating a tree-like structure whose tubular neighborhood contains all data points of the cluster. Some methods may—however—be more suitable than others, either by speed, similarity to the original structure, or the capability of detecting branching cluster structures.

4.1.1 Branching

Since all methods except the evolutionary algorithm find distinguished points within the cluster structure (either as modal points in the density function or as centroids of another clustering algorithm) and build the cluster skeleton from the edges of a minimum spanning tree, this attribute can be obtained very easily. The evolutionary algorithm directly computes a set of candidate B-splines. These are smooth curves which are able to represent simple shapes like a circle but as soon as branching has to be taken into account, B-Splines cannot be used to properly describe the structure. Since this limits the usability of the method quite a lot, it will be omitted from future considerations.

4.1.2 Speed

The runtime of the algorithm has been tested on a single *type* of dataset. For this dataset 100 different instantiations were created with all parameters equal. The chosen dataset is the `chishape` because it needs a branching model to be properly captured. Unlike the `eightshape` dataset this dataset does not exhibit any ambiguity when it comes to finding the skeleton since it does not contain any closed circles.

Each instantiation contained $n = 2500$ points. The runtimes were measured on a Lenovo Y50-70 notebook (Intel(R) Core(TM) i7-4720HQ CPU, 2.6 GHz) with 16GB RAM. All implementation were done by myself and in Python 2.7. To get an estimate of the average runtime 100 trials were performed for each algorithm.

The fastest algorithm was the FCLS algorithm with an average runtime of $5.3s \pm 1.5s$. Approximately twice as much time was needed by the k-means

skeletonization ($12.0s \pm 3.5s$). The GNG algorithm took again almost three times this time ($34.3s \pm 12.2s$), while the GWSO algorithm took almost three minutes to skeletonize a single dataset ($160.6s \pm 191.0s$). For the last algorithms only 9 runs were performed since the runtime is already at least one order of magnitude longer than that of the k-means-based skeletonization or the FCLS.

Since validation usually occurs more than once per data set the high runtime makes the GWSO infeasible for cluster validation at the moment. The main problem lies in the repeated evaluation of the kernel function which take an enormous aount of time. Even the faster to compute approximations did not significantly reduce this problem. with more and faster CPUs the GWSO might become a suitable candidate again.

4.1.3 Accuracy

The accuracy of the resulting cluster skeleton is measured by its distance to the original model from which the data set had been generated. For this the Discrete Fréchet Distance for Trees is used (see Definition 3.8. The results can be seen in the following figures. In addition to the remaining three methods (k-means-based skeletonization, fuzzy-c-line segments-based skeletonization, growing neural gas-based skeletonization) the figures also show a the distance distribution for a purely random skeleton. For this between 11 and 36 points from within the bounding box of the original data set were sampled. A minimum spanning tree on these is constructed and used as a reference skeleton. Obviously this will not yield good results (although the smaller ones might yield good results since the Discrete Fréchet Distance for Trees uses the minimum over all infima that the discrete Fréchet distance yields). However, one can use this distribution to see how much of a methods quality might be attributed to chance.

The figures show that the k-means-based skeletonization yields the smallest error, followed closely by the growing neural gas-based method and the fuzzy-c-line-segments-based method. The k-means-based method usually performs better both in terms of distance and variation. Especially if only very few data points are available, this method performs much better than the other two.

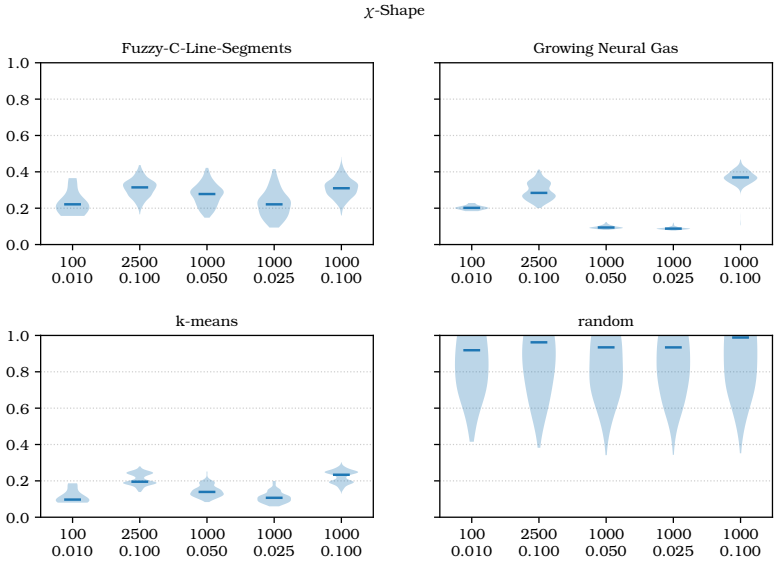


Figure 4.1: Distribution of distances between cluster skeletons and original cluster model as measured by the Discrete Fréchet Distance for Trees. (`chishape` ,n, σ)

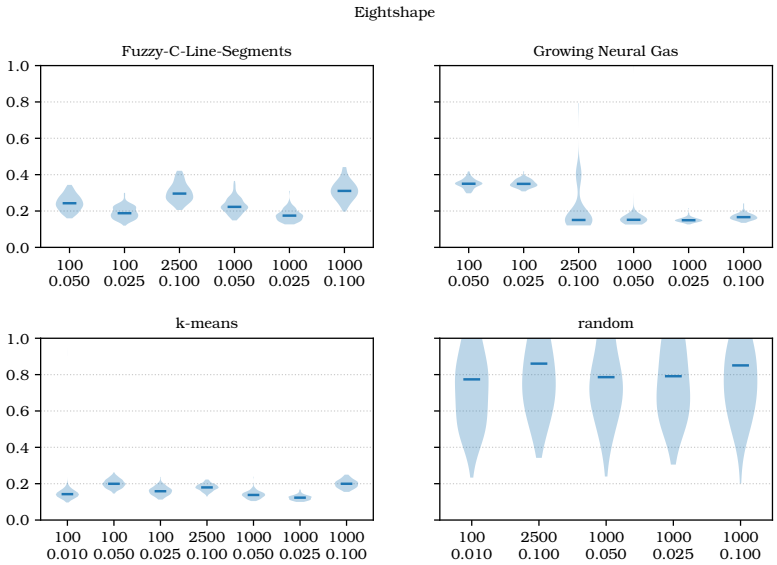


Figure 4.2: Distribution of distances between cluster skeletons and original cluster model as measured by the Discrete Fréchet Distance for Trees. (`figureeight` ,n, σ)

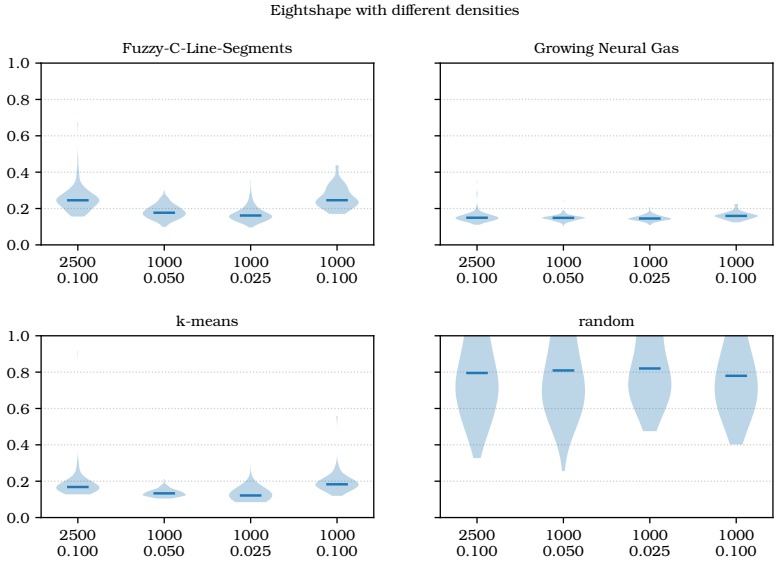


Figure 4.3: Distribution of distances between cluster skeletons and original cluster model as measured by the Discrete Fréchet Distance for Trees. (`figureeight` ,n, σ)

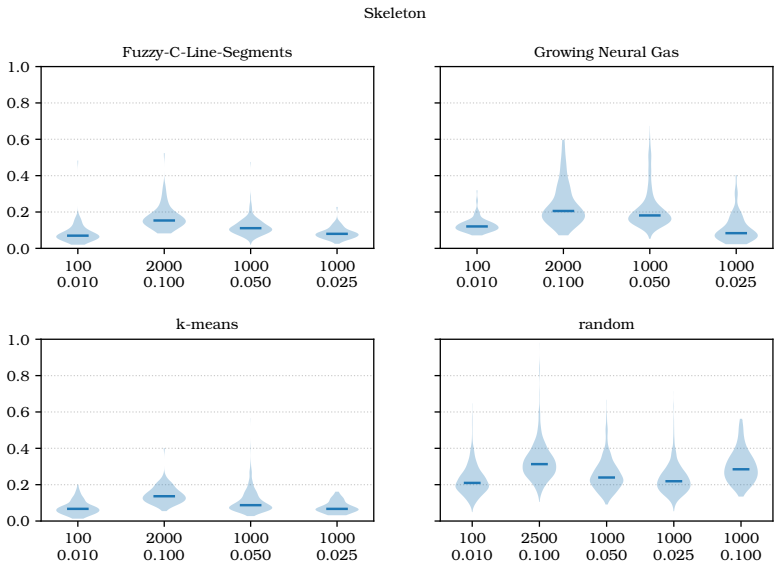


Figure 4.4: Distribution of distances between cluster skeletons and original cluster model as measured by the Discrete Fréchet Distance for Trees. (`skeleton` ,n, σ)

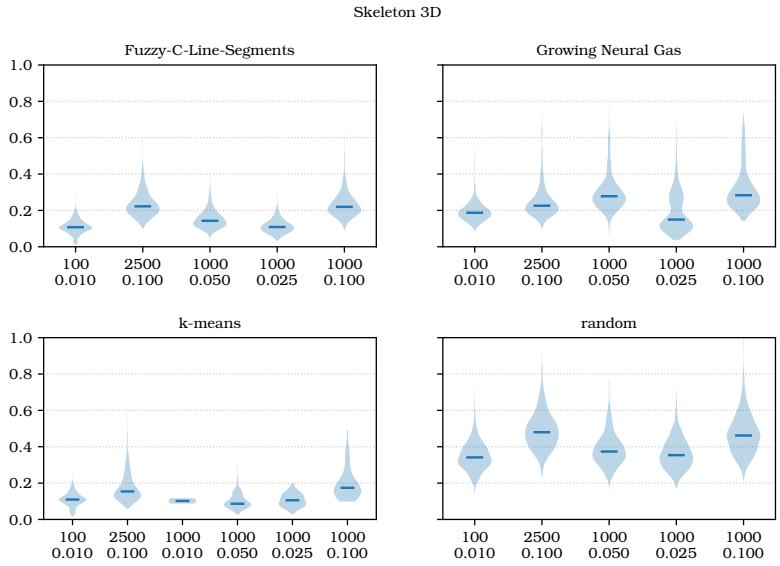


Figure 4.5: Distribution of distances between cluster skeletons and original cluster model as measured by the Discrete Fréchet Distance for Trees. (`skeleton3d` ,n, σ)

4.1.4 Summary

From an evaluation point-of-view the solution based in the evolutionary algorithms and B-splines was the least convincing approach. Neither was the accuracy nor the computational speed satisfying. The skeletons found lack the ability to find anything but single paths. The lack in accuracy—at least for the tested datasets—can certainly be attributed to the small number of generations that were used during testing. This, however, was necessary due to the overall runtime of the algorithm. Better implementations on faster computers may make this method an interesting candidate again, since the overall procedure is certainly capable of representing non-branching clusters in an appropriate way.

The same thing has to be said about the GWSO approach. Though it allows branching skeletons (and not only simple paths) it can be used in a wider range of scenarios. Yet, the repeated calculations of the kernel functions make this method rather slow. Additionally, one can see from the ground truth comparison, that the resulting skeletons do not have the same level of accuracy as the k-means-based skeletonization, the FCLS, or the GNG. This might also be attributed to the kernel parameters chosen

sub-optimally. The cumulated density function across which the particles move might not suitable to properly detect the cluster skeleton. However, more general problems arise in cases where the density varies a lot inside a cluster (e. g. in the `chishape` dataset where the two semi-circles meet). The multimodal approach might favor placing particles in this high density regions and too few inside the arms of the shape). So mainly due to its slowness—but not only—the GWSO approach will not be used any further.

The GNG algorithm achieves one of the highest accuracies of the selected algorithms, sometimes even surpassing the k-means-based skeletonization. Unfortunately it is a rather slow algorithm that furthermore uses a lot of hyper-parameters that can be tuned. The default parameters presented in [63] already yield results that were so good, that no hyper-parameter tuning was performed for this evaluation.

The fastest algorithm out of the five presented methods was the FCLS. Its only drawback was the inferior accuracy when compared to the k-means-based skeletonization or the Growing Neural Gas (GNG). The lack in accuracy can be traced back to what has already been observed in [129, 130]. The fuzzy centers tend to move towards the barycenter of the dataset. This effect is possibly strengthened by the use of a weighted PCA.

The best trade-off between the desired properties for the method is given by the k-means-based skeletonization. It yields the highest accuracy among all tested methods while not taking too much time for the skeletonization process. With a small set of objectives the selection process works towards mostly straight skeletons that can have the necessary angles to capture more complex structures. Furthermore it is relatively easy to incorporate prior knowledge about the kinds of clusters, which are expected, into the model selection process.

Due to this, the k-means-based skeletonization will be used for the rest of this thesis as the primary skeletonization procedure.

4.2 Automatic Selection of Cluster Parameters

From the previous sections it is now known that in terms of speed and accuracy the k-means-based skeletonization yields the most desirable combination of both criteria. Although not the fastest it outperforms every other

method presented in achieved accuracy. For the rest of this thesis this method for computing cluster skeletons will be used to calculate the different validation scores. In Section 3.4 it has been shown how the distance between a point and a cluster skeleton as well as the distance between two skeletons can be computed with relative ease. In Section 2.1.2 several different crisp and fuzzy clustering validation measures were introduced. In the following it will be evaluated how the skeletonization along with an automated parameter search for DBSCAN finds clusters.

4.2.1 Experimental Setup

To generate more challenging data sets to perform the evaluation on a combination of all aforementioned cluster patterns will be used. The individual clusters will be rotated, translated and scaled. An evaluation data set contains a random number of clusters chosen from the different patterns explained in Appendix A, randomly rotated around its individual center. Each of the clusters can be differently scaled and is then randomly placed inside a hypercube. In addition to that random, uniform noise is added throughout this hypercube (potentially lying next to or within a cluster). Figure 4.6 shows an exemplary data set that is used during the validation.

Please note that du to the random placement, and scaling clusters may actually overlap. They then become indistinguishable to a density-based clustering algorithm. Figure 4.7 shows such a case. In terms of density-based clustering the brown and the red cluster near the center are actually the same cluster and must not be separated. With increasing number of clusters, the likelihood of two clusters joining also increases-

The clustering itself is performed by a modified version of the algorithm presented in [47]. Instead of testing all possible values for ε a smaller sample of parameters is selected by performing a k-means clustering on the set of entries of the distance matrix (i. e., only distinct values will be clustered). This also allows a nearly constant runtime since not all $\mathcal{O}(n^2)$ possible values have to be tested. This is important, because a single skeletonization of a single cluster can already take up to two minutes if the cluster is sufficiently large. For performance reasons the range for *minPts* has been limited as well to $[5, 15] \subset \mathbb{N}$.

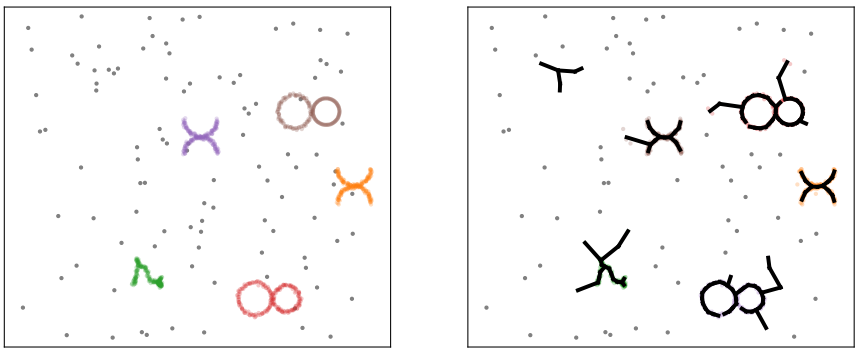


Figure 4.6: Example data set for final evaluation (left) and the resulting labeling (right).

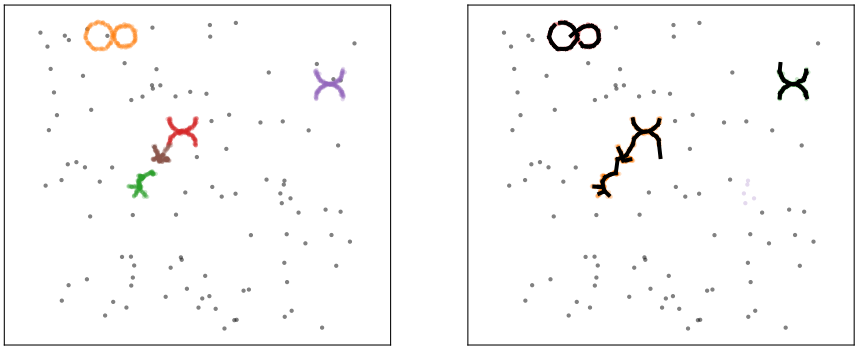


Figure 4.7: Example data set for final evaluation (left) and the resulting labeling (right). In this case the random placement of the cluster centers led to an overlap. Thus, the algorithm cannot separate the two central clusters (red and brown) properly.

4.2.2 Centroid-based Measures with Generalized Centroids

Q3 (Section 1.4) already mentioned that noise can actually become a problem for the validation process. In classical, centroid-based settings noise is simply assigned to its nearest cluster prototype. It then is not treated as noise anymore but just like any other point, possibly deteriorating the clusters' quality. Within a validation measure those noise points influence the calculation of the score since they are not recognizable anymore. When using a clustering algorithm that distinguishes noise from cluster points applying the skeletonization may lead to two different outcomes:

1. The algorithm chooses parameters in such a way that only a few, extremely dense points are assigned to clusters and the majority of points are assigned to the noise cluster. Since the calculation of the validation scores does not include term that cope with noise, they do not contribute the the overall score. The score would then be misleading in different ways. First, it would seem as if the score had been calculated for a much smaller data set. Direct comparison is then not feasible since the raw numbers do not say a lot about the true quality of a clustering (e. g. think about the SSE: Selecting those points closest to a cluster center and ignoring the remaining points is guaranteed to lower the score for this measure). Second, a clustering might appear good (or even yield the best value possible for the respective score), simply because everything that would lead to deduction has been filtered out by the clustering process (although it is still there, of course).

2. The algorithm might choose parameters in such a way that all points are placed into the same cluster. In the case of the fuzzy validation measure Partition Entropy and it is fairly easy to see why this is could be considered an optimal result. The Partition Entropy is maximized for a crisp partitioning (i. e., all memberships are either 0 or 1). If only a single cluster is present in the membership matrix, points spend all their potential membership (which has to sum up to 1.0 on that single cluster. The entropy measure is therefore maximized and the parameter selection procedure would have to choose this clustering as the optimal one.

4.2.3 Algorithmic Improvements

As shown points belonging to two different clusters may be placed into the same cluster if the edge connecting the two individual skeletons is not too long. The reason for this is that the majority of points will still have the same distance to its representative (or rather the edge within the cluster skeleton it is closest to). This distance does not change when another edge is added somewhere else in the skeleton or two skeletons are joined this way. For the membership (cf. Equation 1.3) this means that the enumerator stays

constant for most points while the denominator has one summand less for each point. Overall the membership to the single cluster will be slightly higher (depending on the other clusters) when fewer clusters are present. Since the fuzzy partition based validation measures prefer clusterings that have membership values closer to 0.0 or 1.0 such a clustering will be considered favorable.

For this reason an additional filtering step was added to the algorithm that will take care of such cases. When analyzing where points are projected onto an edge (for distance calculation, cf. Figure 3.19), one can notice that points are nearly uniformly distributed along those edges. However, for edges that connect two different clusters one can see that they only have foot points close to either end point and nearly no foot points in between. This observation gives a simple criterion when a skeleton should be discarded.

4.3 Results

To answer the final research question Q4 several different instances of the validation data sets have been generated. To assess the raw quality of the results the Adjusted Rand Index (cf. Section 2.1.1) was used. It was calculated against the ground truth known from the data generation process. This procedure of validating an algorithm against a ground truth using an external validation measure has already been proposed in [123] and successfully employed in [124]. On the first glance one can see that not all clusterings perfectly resemble the ground truth but that instead there are either smaller or bigger deviations.

Having a closer look at those instances one can see that the small deviations originate from two different sources:

1. The process of generating uniform background noise creates sometimes small clusters of only a few points which are then recognized by the clustering algorithm as dense region within an otherwise sparse region.

2. Sometimes noise points are simply placed within the clusters themselves (or nearby). Although they are technically noise they are also indistinguishable from the cluster (e. g. the same would happen, if two

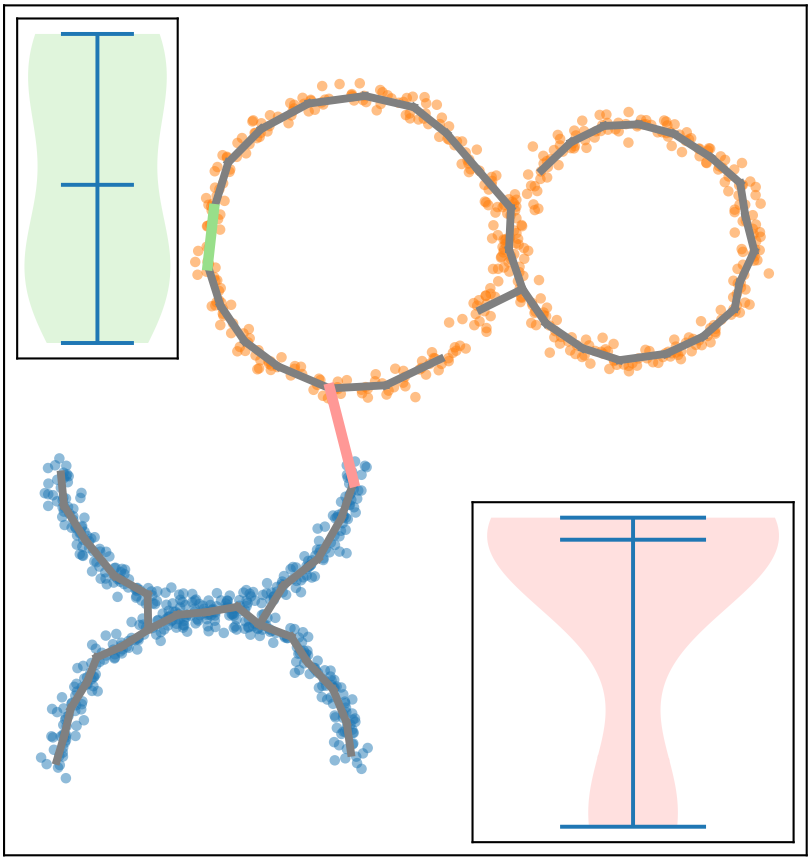


Figure 4.8: Two clusters joined by an unnecessary edge. The upper left violin plot shows a typical distribution for an intra-cluster edge (marked in green). The lower right violin plot shows a typical distribution of foot points for an edge (marked in red), that wrongfully connects two clusters

blob-like clusters were created with their centers very close to each other. These two clusters would appear as a single large and dense cluster, cf. Appendix B.1)

These errors are not rectifiable and have to be tolerated. Similar situations may occur in prototype-based clustering when a single outlier (not noise) for a cluster happens to be closer to another cluster prototype than its original. From a generative point of view the point belongs to the first cluster, from an uninformed, analytical point of view there would be no justification to perform such an assignment.

A similar problem occurs due to the random placement of the clusters. Since the data set generation process does not take the size of the cluster patterns into account it may happen, that the final cluster structures overlaps (see e. g. Figure 4.7). In such cases the algorithm *has* to conclude that the structures form a single cluster. But again, from a generative point of view the result would be plainly wrong and thus the external validation yields a suboptimal result.

Two clusters may also be joined if they are too close together or some noise point fall into the immediate vicinity of the connecting edge. This distorts the distribution of foot points that the closest point of an edge are projected onto. The usual distributions for a good and a bad edge in the skeleton of two wrongfully joined clusters can be seen in Figure 4.8.

The results for different sets of validation data can be found in Figure 4.9. The scores are overall very good. Variations around 1.0 can be explained by some noise points being added to the cluster if they are generated too close. Noise that accidentally forms clusters on its own also deteriorates the score. Table 4.1 shows the parameters used for data set generation.

The seemingly bad results around 0.75 and 0.5 can be explained by the phenomenon that can also be seen in Figure 4.8. Sometimes two clusters become joined by an edge and that edge cannot be removed due to noise points being projected onto the center of that edge. Even more common is the case that two or more clusters are joined because they are actually truly overlapping and have to be counted as a single cluster w.r.t. density-based clustering.

The same experiments have been performed for medium- and high-dimensional datasets, as well. The results can be seen in Figure 4.10 and Figure 4.11. It might seem surprising that the results actually seem to look better than in two dimensions, but in higher-dimensional spaces we have two adversing effects on cluster quality. First, the curse of dimensionality leads to a vanishing relative difference in distances. Essentially at some dimensionality points will always look like they are placed on the surface of a hyper sphere. On the other hand there is much more space to place clusters. Since density-based looks rather at the absolute differences of the distances than the relative difference, the adverse effect of the curse of dimensionality can be better coped with.

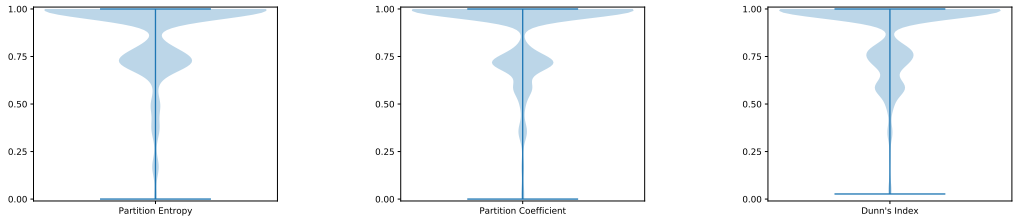


Figure 4.9: Violinplots show the the aggregated ARI scores for PE, PC, and DI. The parameters where chosen randomly and the range can be found in Table 4.1.

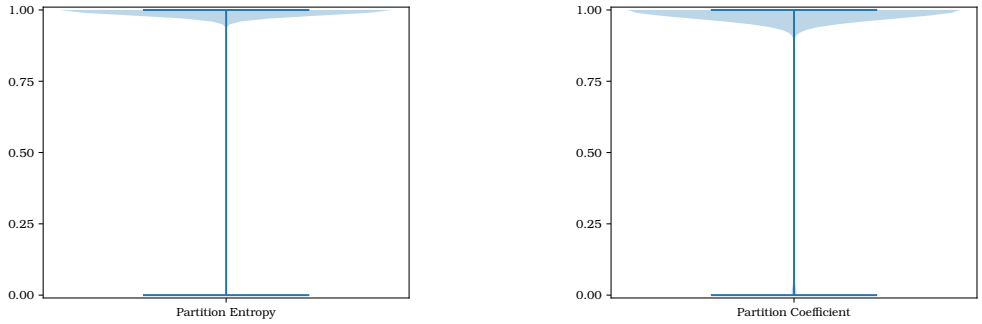


Figure 4.10: Violinplots show the the aggregated ARI scores for PE, and PE. The parameters where chosen randomly and the range can be found in Table 4.1 The number of features was set to 10.

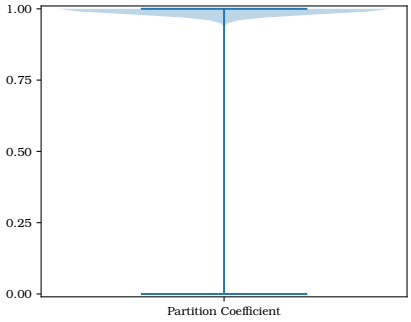


Figure 4.11: Violinplots show the the aggregated ARI scores for PE, and PE. The parameters where chosen randomly and the range can be found in Table 4.1 The number of features was set to 100.

Table 4.1: Parameters for generating Validation data sets.

Parameter	Range
# samples	2000–5000
# clusters	3–5
noisiness of clusters	0.01–0.1
additional uniform noise	10%
rotation	yes (0°–360°)

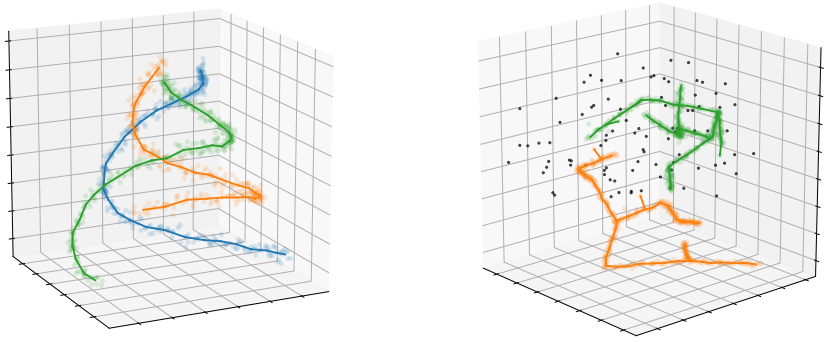


Figure 4.12: Two three-dimensional datasets clustered. Left: Spiral cluster with no additional noise. Right: Skeleton clusters with additional noise.

All of the (numerically) imperfect result can be attributed to one of the three above mentioned reasons. Since the scores from the external validation reflect the quality of the result in a less than optimal way,

4.3.1 Comparison with Other Methods

To reflect the superiority of the skeletonization-based cluster validation in the context of density-based clustering, the results were evaluated against other algorithms and some exemplary trials were run against the Clustering Validation Index based on Nearest Neighbors (CVNN) cluster validation method proposed in [94] (cf. Section 2.1.3).

First, all data sets used in the previous section were also clustered by k-means and a selection process as described in e. g. Figure 2.6 has been employed. The clusters used in this setting do not match the implicit cluster definition used by k-means. Additionally there is noise present, which

deteriorates the performance of k-means. So it might be expected, that the results will be slightly worse. However, the strict partitioning scheme should still be able to place clusters in separate Voronoi cells even if the clusters are overlapping (i. e., all cases were DBSCAN fails). The competitive results are shown in Figure 4.14. As one can see, the skeleton-based validation yields better results for most tested instances.

As for the other density-based validation measure, the same parameter search was used. Out of all possible, unique entries in the distance matrix of the used datasets a smaller sample was generated by using k-means. Using the CVNN the best parameter combination was chosen and compared to the same parameter set chosen by the skeletonization-based validation. The result can be seen in Figure 4.13. In case of the spiral clusters too many clusters are found, while for the skeleton clusters the two clusters which are in three-dimensional relatively close together are joined by the parameter search. In both cases an optimal clustering could have been found since the necessary parameters where well within the range of tested parameters. The problem CVNN might have is that it requires an additional parameter (the number of representatives) as its parameter (which was chosen as $k = 15$ in both cases). This might be a suboptimal value. Nonetheless, introducing a user-specifiable parameter into the validation process removes objectivity from the validation process.

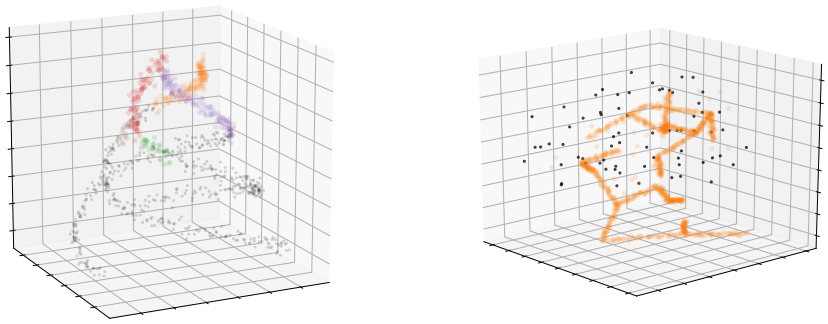


Figure 4.13: First two principal components plotted with resulting cluster structure when using CVNN to guide the parameter search.

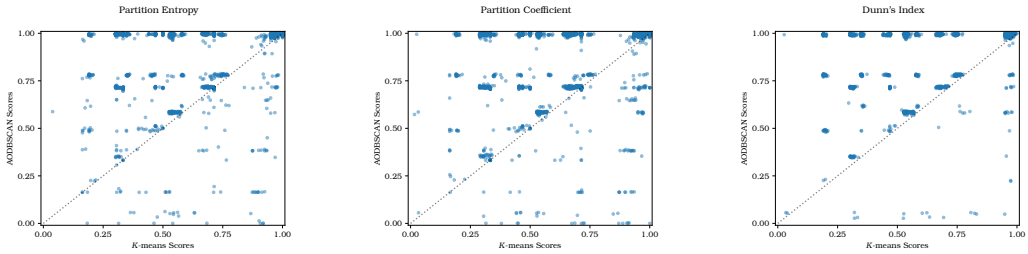


Figure 4.14: Results when comparing aoDBSCAN and k-means clustering using skeletonization-based parameter-selection for aoDBSCAN and choosing the best solution from $k = 2, \ldots, 20$ for k-means. Points above the dotted line indicate a better clustering result for the skeletonization-based method.

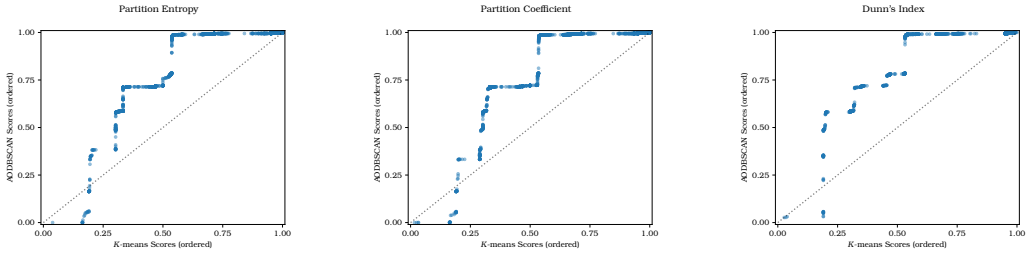


Figure 4.15: The same results as in Figure 4.14 but this time the results were ordered individually per axis. The resulting QQ plot can also be interpreted as a ROC curve.

Finally, this also answers research question Q4: Skeletonization-based validation yields better results than previously used methods.

4.3.2 Example: Biological Image Analysis

In Chapter 1 it was already motivated that especially in cases where the number of clusters is hard to know *a priori* density-based clustering can be one solution. The cell images used in this Section show an unspecified number of cells. The task is to obtain a reliable count of these cells which is needed for further analysis.

Since the cells shown in these image are usually circular and of a rather small diameter this prior knowledge can be exploited within the clustering process. By limiting the range of possible values ε and *minPts* can obtain, the clustering process can be sped up. Additionally, there is no need to look for cluster skeletons of a very high complexity. Nearly spherical cells can be described by skeletons with at most four bones or five joints (i. e., a

star graph with four edges that all meet in one central point). The PC was used as validation measure. Since the results for the different measures do not differ that much any other measure could have been used as well. The data points used as input to the clustering algorithm are those pixels of the (gray-scale) images that remain after applying Otsu's method [104] to determine a gray-level threshhold.

The resulting clustering can be seen in Figure 4.16. Individual clusters have different colors with border points habing a slightly darker shade of the same color. It becomes clear that with this method not only the individual cells can be found but also the cell walls can be made visible by connecting the border points of each cluster. This information may be further used to study some individual clusters which are not very circuly, e. g. the two yellow clusters in the lower center region of the image.

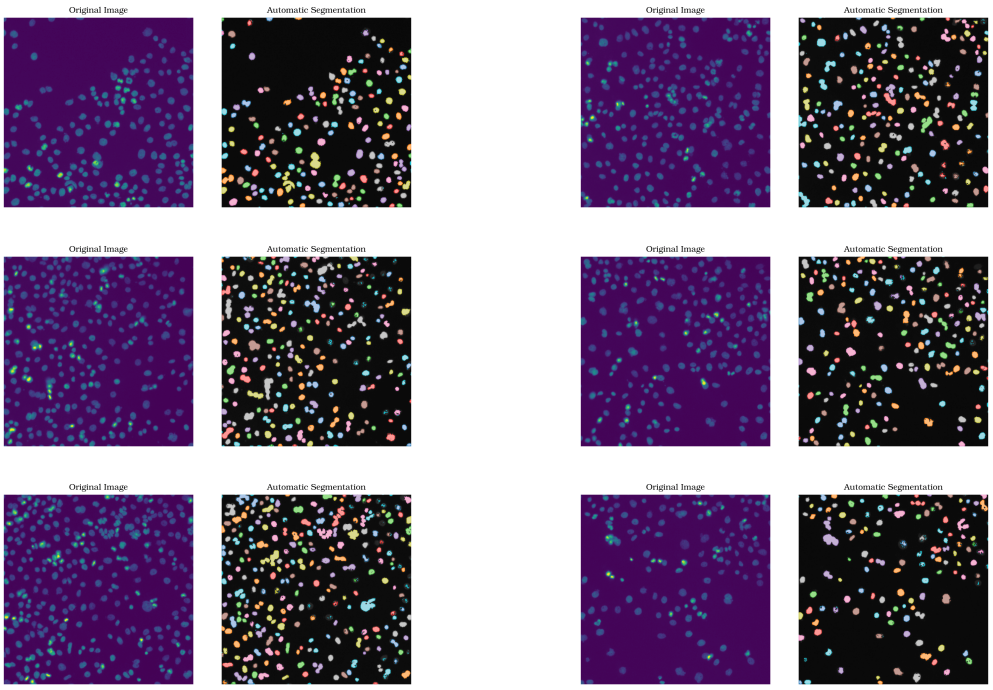


Figure 4.16: Left: Original image. Right: Clusters found by the skeletonization-based parameter selection.

The same procedure can obviously also performed by a simple k-means algorithm and the results are shown in Figure 4.17. Here the search space was limited to $2 \leq k \leq 50$. It can already be seen that not every validation measure is suitable for this task, e. g. Calinski-Harabasz Index favors an

equi-proportional partitioning of the dat space. Since the skeletonization uses fuzzy validation measures, a direct comparison with crisp validation measures does not yield a lot of information about the possible performance increase. Thus, Figure 4.19 shows the same process but this time using the fuzzy c-means algorithm in the background. The resulting clusterings are clearly worse than those obtained via k-means or the k-means-based skeletonization. This might be due to the overall high number of cluster in a relatively small vicinity. This makes a lot of assignment very fuzzy and less crisp which in return is accordingly penalized by the different fuzzy validation scores.

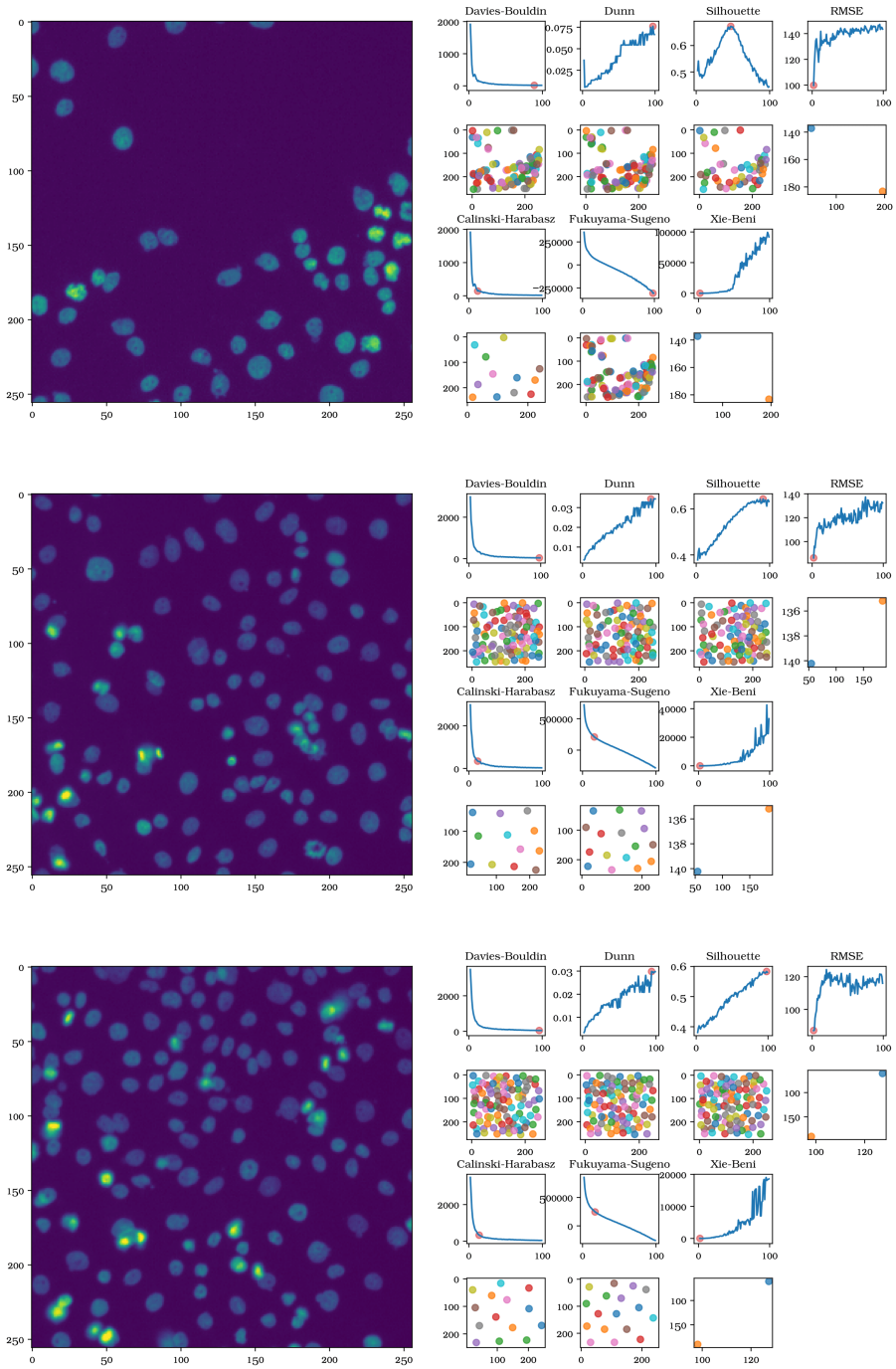


Figure 4.17: Cluster selection process for the first three colon cancer cell images using k-means. Due to memory limitations only the top-left quarter of the image (256×256px has been used).

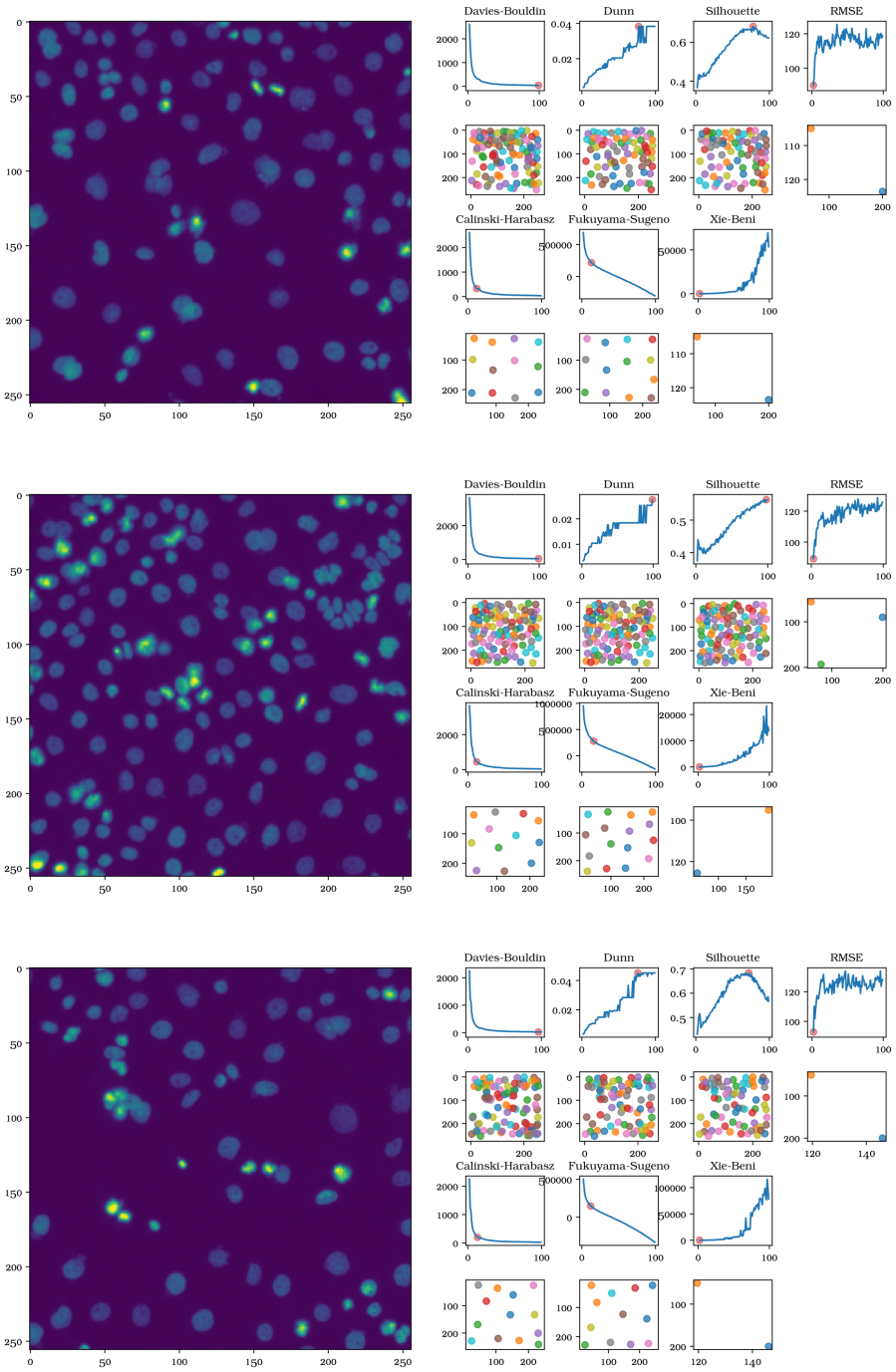


Figure 4.18: Cluster selection process for the last three colon cancer cell images using k-means. Due to memory limitations only the top-left quarter of the image (256×256px has been used).

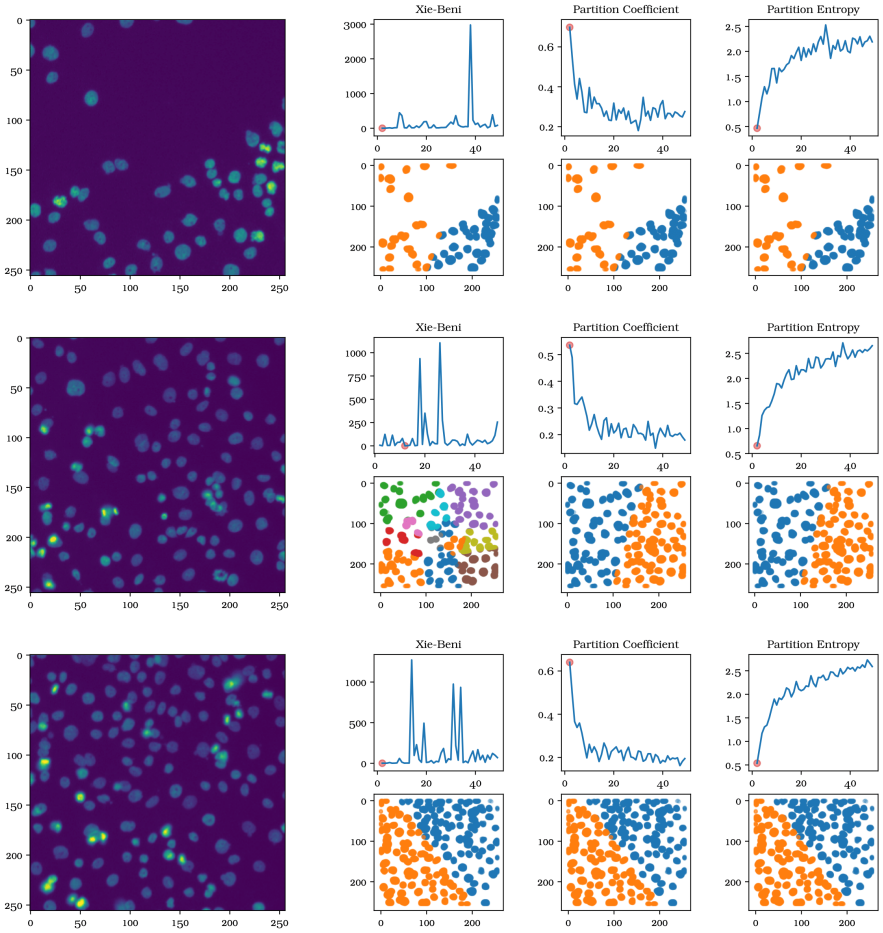


Figure 4.19: Cluster selection process for the first three colon cancer cell images using fuzzy c-means. Due to memory limitations only the top-left quarter of the image (256×256px has been used).

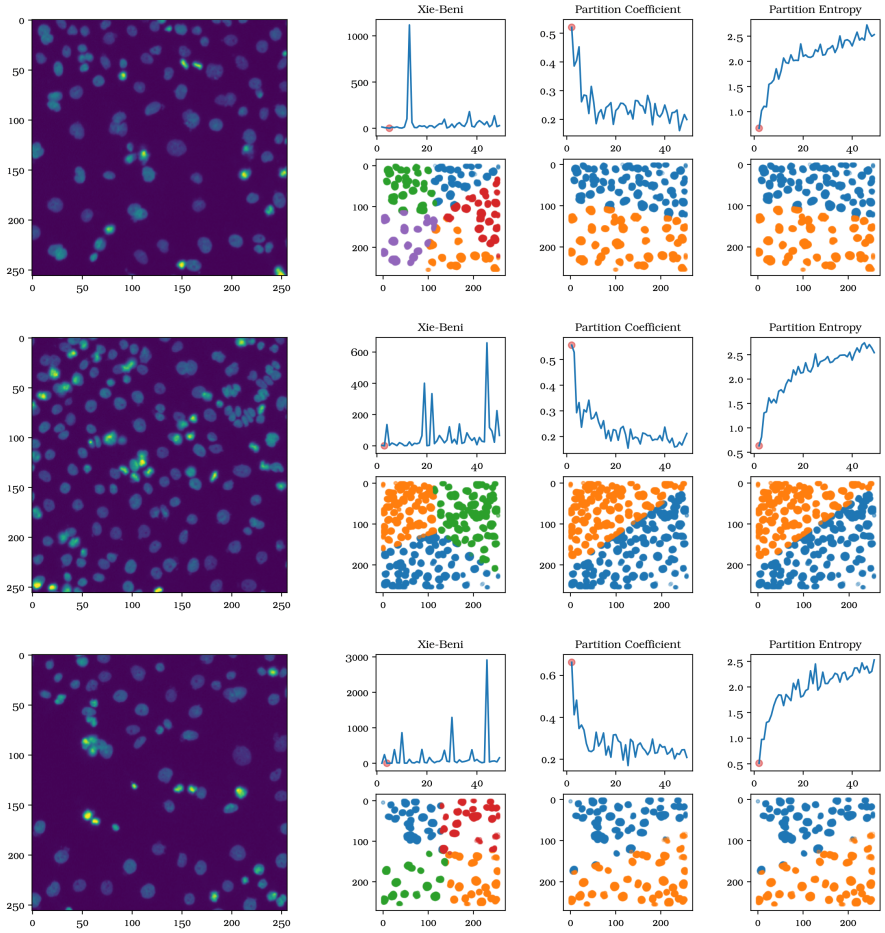


Figure 4.20: Cluster selection process for the first three colon cancer cell images using fuzzy *c*-means. Due to memory limitations only the top-left quarter of the image (256×256px has been used).

CHAPTER 5

Conclusion & Future Work

Within this thesis the problem of validating results from a density-based clustering algorithm was discussed. As a prime example for such an algorithm the DBSCAN algorithm was used. The problem with existing validation measures roots in the use of the arithmetic mean as a cluster's representative. However, the arithmetic mean is not always a suitable centroid for clusters that can be found by density-based algorithms.

To overcome this problem cluster skeletons were introduced. How such a skeleton can be found in two-dimensional space was shown and an analytical method was discussed. Since most data is not two-dimensional, different methods of obtaining an approximative solution to this problem were presented and analyzed. Those methods come from many different fields within computational intelligence, are motivated by existing algorithms for skeletonization of discrete or low-dimensional data, and have different benefits.

The resulting validation showed that an automated parameter selection process (imitating an exhaustive search) yields very good results when paired with validation measures that calculate their score based on a fuzzy partitioning.

5.1 Discussion & Research Questions

To summarize all the result of this thesis w.r.t. to the research questions initially posed, these will be recapitulated and a short discussion on each question's results be given in the following.

Q1 **Can the arithmetic mean as a cluster representative be replaced by a more general structure?**
Using the medial axis (and removing undesired parts of it) yields an object that better resembles a structured cluster. Performing necessary calculations w.r.t. this medial axis object instead of the arithmetic mean yields better results as can be seen in Figure 3.20. The resulting fuzzy partitioning of the space surrounding the two clusters seems much more intuitive and appropriate than the one resulting from using the arithmetic mean as centroids (cf. Figure 3.21).

A clear disadvantage here is that in order to calculate the medial axis a hull of the object has to be present from which the medial axis can be calculated. Since the convex hull is not suitable for this, an algorithm that either computes the medial axis of the data points indirectly is needed or some notion of a *concave hull* is required. This concave hull is neither easily nor uniquely defined. With Algorithm 2 however, some sort of shape descriptor that is suitable for the task at hand has been presented.

Q2 **How can the medial axis be extended onto higher dimensions and how can such a structure be efficiently calculated?**
The medial axis is only defined (as a locally one-dimensional object) in two-dimensional space. In three-dimensional space the resulting object would be called medial surface and calculation regarding this object (or its calculation itself) would become harder. For even higher dimensions the calculations of such objects becomes infeasibly complex (exponential in the number of dimensions). Though not strictly required for a generalized centroid, having a local restriction on the dimensionality eases some of the required calculations and makes the representation easier.

Throughout Section 3.3 several different methods that find cluster skeletons have been discussed. The selection of algorithms is not exhaustive but gives a good overview over different kinds of approaches, their limitations, and their benefits. With the help of a modified Fréchet distance, that can be used for tree structures and not only paths (or curves), the evaluation for a set of different cluster shapes

became possible. Thus it could be validated how well each methods fits skeletons to the original model from which the data has been generated.

A compromise between accuracy and speed had to be made here and overall the best candidate for further consideration is the *k*-means-based skeletonization. This methods yields the best accuracy by only making slight admissions to speed.

Q3 **How can noise points originating from the density-based clustering be dealt with in validation scores which ignore noise?**
Contrary to centroid-based clustering, one always has to consider that some points will not be assigned to clusters when using density-based clustering like DBSCAN. The original idea was to use cluster skeletons instead of the arithmetic mean. In crisp cluster validation this can lead to the (undesired) effect, that a clustering might assign almost every point to the noise cluster, and that this clustering will be assessed as a better result. This happens because noise points are commonly ignored in centroid-based cluster validation. [102] discusses several different approaches for handling noise:

1. **Noise is assigned to a single cluster**
   This way the noise cluster could dominate the validation (since noise should be expected to span the completely data space) and solution that assign all or most points to the noise cluster will be deemed better.

2. **Noise is assigned to the nearest cluster**
   This changes the structure of the clusters. The clusters that were found and the clusters that are validated are not the same anymore and thus the outcome of the internal cluster validation measures cannot be predicted.

3. **Noise could be assigned to a singleton cluster that is equi-distant to all points**
   The choice of the noise distance is crucial for this to lead to good results. In the best case the validation has a(nother) parameter; in the worst case the separation of clusters is lost (e. g. if the noise

cluster is less than ε from two core points of different clusters it may look like these clusters should have been joined and the clustering may be assessed as *bad*).

4. **Noise could be completely ignored**
   Just as the in the second approach this leads to a manipulation of the dataset such that the clustered and the validated dataset are not the same anymore. Internal cluster validation measures may produce results that do not allow to draw conclusions about the original dataset.

5. **Noise could be completely ignored (with penalty)**
   Adding a penalization to existing validation measures changes the inherent behavior of the measure. One cannot say anymore that the measures are identical. Although this is the path chosen in [102] one cannot say that this is ultimately the optimal way.

All these options contradict the goal of this thesis to make the existing terms available to density-based clustering and not introduce yet another validation measure in one way or another.

However, noise points are more gently handled in fuzzy clustering and the validation of fuzzy clusterings. Especially Partition Entropy recognizes points that are extreme out- or inliers very well since they will yield fuzzy memberships close to $1/c$ (in the case of outliers) or $1/2$ (in the case of inliers) and thus generate a lot of entropy. The results shown in Section 4.2 show that the only deviations from the ground truth labeling have one of the following reasons: 1. The points that were labeled incorrectly are actually noise points that were randomly placed inside or in the vicinity of a cluster. Thus, they cannot be distinguished from the cluster itself. Or, 2. The clusters were placed too close to each other. This leads to the situation that the clusters become density connected (cf. Definition 1.11) and by transitive closure belong to the same cluster. Points that fulfill the first condition lead to variance around the ideal scores; points that fulfill the latter lead to much worse scores (cf. Figure 4.9).

Q4 **Does validation based on cluster skeletons actually lead to better results?**

Cluster skeletons describe and represent a cluster's structure better than a single point. This holds true especially if the cluster is not spherical. The comparison between the two different approaches also shows that density-based clustering can be led within an automatic parameter selection process to find results that are better than those of a similar centroid-based procedure.

However, the way skeletonization represents clusters favors piecewise tubular clusters (like the skeleton clusters, cf. Appendix B.7). Spherical clusters or those of wildly varying density cannot be properly represented. This effect is demonstrated in Figure 5.1. Although a simpler model would suffice, the algorithm has to choose a more complex representation. This partially comes from the selected skeletonization method. Since the optimization process looks at the standard deviation of the angles between edges, at least three edges have to be present. Otherwise a proper estimation of either standard deviation or variance would not be possible (with only two edges one would have only a single angle and thus no way to properly estimate either value). This excludes the simplest models.

Yet, by limiting the search space of the automatic parameter search some form of prior knowledge can be introduced into the skeletonization process. The standard skeletonization process described in Section 3.3.3 evaluates cluster skeletons that have between 5 and 55 joints in total. If it is known in advance that the data set contains mostly spherical or simple tubular clusters (without a more complex structure) the range of skeletons can be easily restricted. The same holds true for the parameter space of the DBSCAN algorithm (described in Section 4.2). By limiting the possible range of values for ε and *minPts* certain clustering could be excluded from the validation *a priori*.

Last, if two spherical clusters are (partially) overlapping in their less dense regions (e. g. two clusters formed by Gaussian distributions), centroid-based clustering will label the bigger part of the points correctly. Especially fuzzy clustering allows a fine differentiation in these cases. Density-based clustering will—however—be forced to join these

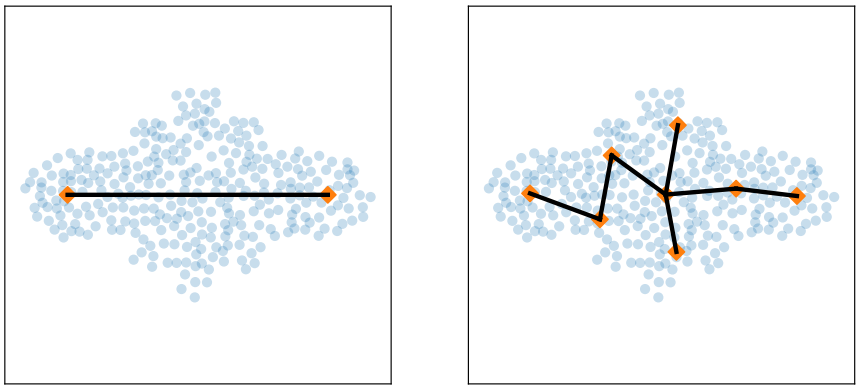


Figure 5.1: On the left an well-skeletonized line of points. On the right an overfitted skeleton of the same data

two clusters into one cluster if ε is chosen too large or miss many of the points by labeling them as noise if *minPts* is chosen too high. For such scenarios density-based clustering is less suitable in the first place and skeletonization will not make this better.

5.2 Future Work

The kind of clusters this method is applicable to is limited to tubular clusters, like blood vessels or streets. Gradual changes in thickness of these tubes are not handled very well (see Figure 5.1). If one applied a density estimate along each edge certain kinds of thickness changes might become detectable and the resulting skeletons may fit the data better.

Another improvement could be done in the way that edges are detected which connect to otherwise distinct clusters (cf. Section 4.2.3). While effective, the currently employed method appears a bit crude and a more sophisticated method might be more suitable. Especially when a single noise point lies directly at or at least near the center of an edge, it would currently be hard to detect this edge. Thus, a suboptimal skeleton would remain in the validation process and distort the result.

This edge exclusion procedure could—on the other hand— potentially lead to a completely new clustering algorithm. If the complete data set was properly skeletonized the edge exclusion could possibly split the skeleton at

the right places. The remaining connected components would then form the clusters. However, a validation of this algorithm should then be done with another method, since such a skeleton-based clustering would be strongly biased.

A rough analysis of the proposed approach reveals that the overall complexity is approximately in $O(n^{3.5})$: For each entry in the distance matrix (possible values of ε) perform several (reasonably only up to $\sqrt{n}$) calculations of k-means for the skeletonization. k-means itself has a runtime complexity of $O(n \cdot k \cdot d \cdot i)$, where i is the number of iterations needed and d the number of dimensions of the dataset. This accumulates to $O(n^2 \cdot (n \cdot k \cdot d \cdot i))$ because i and d can be considered constant and $k = \sqrt{n}$ in the worst case. A proper parallelization or use of an incremental k-means algorithm, that re-uses clustering information obtained from runs with lower k might speed up computations here slightly. Also a proper sampling on the distance matrix values to reduce the number of possible εs seems promising to reduce the overall runtime.

Currently all algorithms are implemented in pure Python without any further optimizations. C-based implementations will not reduce the complexity but certainly speed up computations.

Another challenge that occurs due to the usage of k-means in the skeletonization process is that k-means (as used here with k-means++ as initialization) is not deterministic. The effect of this is usually alleviated by running the same parametrization several times and then choosing the best result. However, this still does not guarantee that for every distinct run on identical datasets the same skeletons are constructed. Theoretically possible are therefore cases in which an optimal clustering has been achieved but cannot be found since the k-means step for the skeleton construction yielded a suboptimal result.

On the validation side the algorithm has been tested for several configurations of datasets. It was, however, only validated on a small sample of validation measures. The behavior of the other presented measures as well as those not discussed throughout Chapter 2 has not been tested and no statement about the algorithm's behavior can be made. One can only conjecture that they will show a similar behavior since the calculations are not fundamentally different.

Some of the proposed skeletonization methods have not been included in the final validation because they were either too slow or seemed to produce results that did not meet certain quality criteria. These might prove effective with a more suitable implementation or in the future with better hardware or more computing power respectively.

Yet, the current approach already shows that skeletonization-based cluster validation has the potential to improve clustering quality.

Bibliography

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, "Geometric approximation via coresets," *Combinatorial and computational geometry*, vol. 52, pp. 1–30, 2005 (cited on page 44).

[2] I. Agricola and T. Friedrich, *Elementary Geometry*, ser. Student Mathematical Library. Publications of the AMS, 2008, vol. 43 (cited on page 55).

[3] H. Alt and M. Godau, "Measuring the resemblance of polygonal curves," in *Proceedings of the eighth annual symposium on Computational geometry*, ACM, 1992, pp. 102–109 (cited on page 50).

[4] ——, "Computing the fréchet distance between two polygonal curves," *International Journal of Computational Geometry & Applications*, vol. 5, no. 01n02, pp. 75–91, 1995 (cited on page 50).

[5] I. Anderson, J. Bezdek, and R Dave, "Polygonal shape description of plane boundaries," *Systems science and science*, vol. 1, pp. 295–301, 1982 (cited on pages 18, 48).

[6] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *ACM Sigmod record*, ACM, vol. 28, 1999, pp. 49–60 (cited on page 15).

[7] G. S. Antzoulatos and M. N. Vrahatis, "a-clusterable sets," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2011, pp. 108–123 (cited on page 66).

[8] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035 (cited on page 41).

[9] X. Bai, L. J. Latecki, and W.-Y. Liu, "Skeleton pruning by contour partitioning with discrete curve evolution," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 3, 2007 (cited on page 45).

[10] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction To Splines For Use In Computer Graphics & Geometric Modeling*. Los Altos: Morgan Kaufmann, 1987 (cited on page 61).

[11] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural computation*, vol. 12, no. 10, pp. 2385–2404, 2000 (cited on page 43).

[12] H. Becker, "Identification and characterization of events in social media," PhD thesis, Columbia University, 2011 (cited on page 25).

[13] M. R. Berthold, C. Borgelt, F. Höppner, and F. Klawonn, *Guide to Intelligent Data Analysis, How to Intelligently Make Sense of Real Data*, 1st ed., ser. Texts in Computer Science. London: Springer-Verlag, 2010, XIII, 394, ISBN: 978-1-4471-2572-3 (cited on page 43).

[14] J. C. Bezdek, "Cluster validity with fuzzy sets," *Journal of Cybernetics*, vol. 3, no. 3, pp. 58–73, 1973 (cited on page 38).

[15] ——, "Mathematical models for systematics and taxonomy," in *Proceedings of eighth international conference on numerical taxonomy*, vol. 3, 1975, pp. 143–166 (cited on page 39).

[16] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2–3, pp. 191–203, 1984 (cited on page 9).

[17] J. C. Bezdek and R. J. Hathaway, "VAT: A tool for visual assessment of (cluster) tendency," in *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, IEEE, vol. 3, 2002, pp. 2225–2230 (cited on page 26).

[18] J. C. Bezdek, R. J. Hathaway, and J. M. Huband, "Visual assessment of clustering tendency for rectangular dissimilarity matrices," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 5, pp. 890–903, 2007 (cited on page 27).

[19] H. Blum, "A transformation for extracting new descriptors of shape," *Models for Perception of Speech and Visual Forms, 1967*, pp. 362–380, 1967 (cited on pages 45, 53).

[20] G. Borgefors, "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, vol. 34, no. 3, pp. 344–371, 1986, ISSN: 0734-189X (cited on page 45).

[21] C. Borgelt and C. Braune, "Prototype construction for clustering of point processes based on imprecise synchrony," in *8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-13)*, Atlantis Press, 2013 (cited on page 2).

[22] C. Borgelt, C. Braune, T. Kötter, and S. Grün, "New algorithms for finding approximate frequent item sets," *Soft Computing*, vol. 16, no. 5, pp. 903–917, 2012 (cited on pages 1, 2).

[23] C. Borgelt, C. Braune, K. Loewe, and R. Kruse, "Mining frequent parallel episodes with selective participation," in *2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15)*, Atlantis Press, 2015 (cited on page 1).

[24] L. Bradstreet, L. Barone, and L. While, "Maximising hypervolume for selection in multi-objective evolutionary algorithms," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, IEEE, 2006, pp. 1744–1751 (cited on page 64).

[25] C. Braune, S. Besecke, and R. Kruse, "Density based clustering: Alternatives to DBSCAN," in *Partitional Clustering Algorithms*, Springer International Publishing, 2015, pp. 193–213 (cited on pages 16, 71).

[26] ——, "Using changes in distribution to identify synchronized point processes," in *Strengthening Links Between Data Analysis and Soft Computing*, Springer, 2015, pp. 241–248 (cited on page 1).

[27] C. Braune, C. Borgelt, and S. Grün, "Finding ensembles of neurons in spike trains by non-linear mapping and statistical testing," *Advances in Intelligent Data Analysis X*, pp. 55–66, 2011 (cited on page 2).

[28] ——, "Assembly detection in continuous neural spike train data," *Advances in Intelligent Data Analysis XI*, pp. 78–89, 2012 (cited on page 2).

[29] C. Braune, C. Borgelt, and R. Kruse, “Behavioral clustering for point processes,” in *Advances in Intelligent Data Analysis XII*, Springer, 2013, pp. 127–137 (cited on page 2).

[30] C. Braune, M. Dankel, and R. Kruse, “Obtaining shape descriptors from a concave hull-based clustering algorithm,” in *International Symposium on Intelligent Data Analysis*, Springer International Publishing, 2016, pp. 61–72 (cited on pages 54, 58).

[31] C. Braune, M. Glauer, and R. Kruse, “Towards online detection of neural assemblies in parallel spike trains,” in *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, IEEE, 2015, pp. 1503–1511 (cited on page 2).

[32] C. Braune and R. Kruse, “Active learning-based identification of neuronal assemblies in parallel spike trains,” in *Proceedings. 24. Workshop Computational Intelligence, Dortmund, 27.-28. November 2014*, KIT Scientific Publishing, vol. 50, 2014, p. 155 (cited on page 1).

[33] ——, “Detecting parallel bursts in in silico generated parallel spike train data,” *BMC Neuroscience*, vol. 16, no. Suppl 1, P134, 2015 (cited on page 1).

[34] ——, “Fuzzy density based clustering with generalized centroids,” in *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, IEEE, 2016 (cited on page 58).

[35] V. Braverman, D. Feldman, and H. Lang, “New frameworks for offline and streaming coreset constructions,” *arXiv/CoRR*, vol. abs/1612.00889, 2016 (cited on page 44).

[36] R. Bridson, “Fast poisson disk sampling in arbitrary dimensions,” in *ACM SIGGRAPH 2007 Sketches*, ser. SIGGRAPH ’07, San Diego, California: ACM, 2007, ISBN: 978-1-4503-4726-6 (cited on page 163).

[37] T. Caliński and J. Harabasz, “A dendrite method for cluster analysis,” *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974 (cited on page 33).

[38] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*, Springer, 2013, pp. 160–172 (cited on page 13).

[39] A. E. Carpenter, T. R. Jones, M. R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. H. Chang, R. A. Lindquist, J. Moffat, P. Golland, and D. M. Sabatini, "Cellprofiler: Image analysis software for identifying and quantifying cell phenotypes," *Genome Biology*, vol. 7, no. 10, R100, 2006, ISSN: 1474-760X (cited on pages 2, 167).

[40] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 191–203, 2008 (cited on page 151).

[41] K. Chen, "On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications," *SIAM Journal on Computing (SICOMP)*, vol. 39, pp. 923–947, 2009 (cited on page 44).

[42] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. New York, NY, USA: Wiley-Interscience, 2006, ISBN: 0471241954 (cited on page 23).

[43] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 224–227, 1979 (cited on page 30).

[44] C. De Boor, *A practical guide to splines (applied mathematical sciences vol 27)*, 1978 (cited on page 60).

[45] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002 (cited on pages 61, 62).

[46] I. S. Dhillon, S. Mallela, and R. Kumar, "A divisive information-theoretic feature clustering algorithm for text classification," *Journal of machine learning research*, vol. 3, pp. 1265–1287, 2003 (cited on page 2).

[47] A. Dockhorn, C. Braune, and R. Kruse, "An alternating optimization approach based on hierarchical adaptations of dbscan," in *Computational Intelligence, 2015 IEEE Symposium Series on*, IEEE, 2015, pp. 749–755 (cited on pages 15, 30, 42, 100).

[48] ——, "Variable density based clustering," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016 (cited on page 15).

[49] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973 (cited on page 32).

[50] S. C. E., "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, (cited on page 39).

[51] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 Congress on Evolutionary Computation*, IEEE, vol. 1, 2000, pp. 84–88 (cited on page 68).

[52] D. H. Eberly, "Distance methods," in *3D Game Engine Design (Second Edition)*, ser. The Morgan Kaufmann Series in Interactive 3D Technology, D. H. Eberly, Ed., Second Edition, San Francisco: Morgan Kaufmann, 2007, pp. 639–679 (cited on page 86).

[53] P. H. Eilers and B. D. Marx, "Flexible smoothing with B-splines and penalties," *Statistical science*, pp. 89–102, 1996 (cited on page 60).

[54] T. Eiter and H. Mannila, "Computing discrete fréchet distance," Tech. Report CD-TR 94/64, Information Systems Department, Technical University of Vienna, Tech. Rep., 1994 (cited on page 50).

[55] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Second International Conference on Knowledge Discovery and Data Mining*, E. Simoudis, J. Han, and U. Fayyad, Eds., Portland, Oregon: AAAI Press, 1996, pp. 226–231 (cited on page 13).

[56] R. Evans, B. Pfahringer, and G. Holmes, "Clustering for classification," in *7th International Conference on Information Technology in Asia (CITA 11), 2011*, IEEE, 2011, pp. 1–8 (cited on page 2).

[57] S. Ezennaya-Gomez, C. Borgelt, C. Braune, K. Loewe, and R. Kruse, "Handling selective participation in neuron assembly detection," in *Computational Intelligence*, Springer International Publishing, 2015, pp. 386–406 (cited on page 1).

[58] D. Feldman and M. Langberg, "A unified framework for approximating and clustering data," in *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, ser. STOC '11, San Jose, California, USA: ACM, 2011, pp. 569–578, ISBN: 978-1-4503-0691-1 (cited on page 44).

[59] B. Fischer, T. Zöller, and J. Buhmann, "Path based pairwise data clustering with application to texture segmentation," in *Energy minimization methods in computer vision and pattern recognition*, Springer, 2001, pp. 235–250 (cited on page 27).

[60] R. A. Fisher, "On the probable error of a coefficient of correlation deduced from a small sample," *Metron*, vol. 1, pp. 3–32, 1921 (cited on page 33).

[61] ——, "The use of multiple measurements in taxonomic problems," *Annals of human genetics*, vol. 7, no. 2, pp. 179–188, 1936 (cited on page 43).

[62] M. M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, vol. 22, no. 1, pp. 1–72, 1906 (cited on page 50).

[63] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in neural information processing systems*, 1995, pp. 625–632 (cited on pages 43, 80, 82, 99).

[64] L. Fu and E. Medico, "Flame, a novel fuzzy clustering method for the analysis of dna microarray data," *BMC bioinformatics*, vol. 8, no. 1, p. 3, 2007 (cited on page 154).

[65] Y. Fukuyama, "A new method of choosing the number of clusters for the fuzzy c-mean method," in *Proc. 5th Fuzzy Syst. Symp., 1989*, 1989, pp. 247–250 (cited on page 34).

[66] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 4, 2007 (cited on page 149).

[67] J. Grimmer and G. King, "General purpose computer-assisted clustering and conceptualization," *Proceedings of the National Academy of Sciences*, vol. 108, no. 7, pp. 2643–2650, 2011 (cited on page 27).

[68] D. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on*, IEEE, 1979, pp. 761–766 (cited on pages 11, 43, 48).

[69] T. C. Hales, "Historical overview of the kepler conjecture," in *The Kepler Conjecture*, Springer, 2011, pp. 65–82 (cited on page 73).

[70] M. Halkidi and M. Vazirgiannis, "Clustering validity assessment using multi representatives," in *Proceedings of the Hellenic Conference on Artificial Intelligence, SETN*, 2002, pp. 237–249 (cited on pages 41, 44).

[71] R. J. Hathaway, J. C. Bezdek, and J. M. Huband, "Scalable visual assessment of cluster tendency for large data sets," *Pattern Recognition*, vol. 39, no. 7, pp. 1315–1324, 2006 (cited on page 27).

[72] F. Hoeppner, "Fuzzy shell clustering algorithms in image processing: Fuzzy c-rectangular and 2-rectangular shells," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 4, pp. 599–613, 1997 (cited on pages 18, 48).

[73] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, Ieee, 1994, pp. 82–87 (cited on page 63).

[74] J. Huband and J. Bezdek, "VCV2–visual cluster validity," *Computational Intelligence: Research Frontiers*, pp. 293–308, 2008 (cited on page 2).

[75] J. M. Huband, J. C. Bezdek, and R. J. Hathaway, "BigVAT: Visual assessment of cluster tendency for large data sets," *Pattern Recognition*, vol. 38, no. 11, pp. 1875–1886, 2005 (cited on page 27).

[76] J. Huband, J. Bezdek, and R. Hathaway, "Revised visual assessment of (cluster) tendency (reVAT)," in *Fuzzy Information, 2004. Processing NAFIPS'04. IEEE Annual Meeting of the*, IEEE, vol. 1, 2004, pp. 101–104 (cited on page 27).

[77] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985 (cited on page 23).

[78] P. Jaccard, "Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 241–272, 1901 (cited on page 22).

[79] A. K. Jain, "Fundamentals of digital image processing," *Prentice Hall Information and System Sciences series, NewJersey*, pp. 33–35, 1989 (cited on page 45).

[80] A. K. Jain and M. H. Law, "Data clustering: A user's dilemma," *PReMI*, vol. 3776, pp. 1–10, 2005 (cited on page 153).

[81] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999 (cited on page 149).

[82] U. Kaymak and M. Setnes, "Fuzzy clustering with volume prototypes and adaptive cluster merging," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 6, pp. 705–712, 2002 (cited on pages 17, 48, 74).

[83] F. Klawonn and F. Höppner, "Fuzzy cluster analysis from the viewpoint of robust statistics," in *Views on Fuzzy Sets and Systems from Different Perspectives*, Springer, 2009, pp. 439–455 (cited on pages 9, 11).

[84] F. Klawonn, R. Kruse, and H. Timm, "Fuzzy shell cluster analysis," in *Learning, networks and statistics*, Springer, 1997, pp. 105–119 (cited on pages 18, 48).

[85] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982 (cited on pages 43, 80).

[86] K. Krishnanand and D. Ghose, "Detection of multiple source locations using a glowworm metaphor with applications to collective robotics," in *Swarm intelligence symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, IEEE, 2005, pp. 84–91 (cited on page 66).

[87] R. Krishnapuram and C.-P. Freg, "Fitting an unknown number of lines and planes to image data through compatible cluster merging," *Pattern recognition*, vol. 25, no. 4, pp. 385–400, 1992 (cited on pages 18, 48).

[88] R. J. Krishnapuram, H. Frigui, and O. Nasraoui, "New fuzzy shell clustering algorithms for boundary detection and pattern recognition," in *Intelligent Robots and Computer Vision X: Algorithms and Techniques*, International Society for Optics and Photonics, vol. 1607, 1992, pp. 458–466 (cited on pages 18, 48).

[89] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher, *Computational intelligence: a methodological introduction*. Springer, 2016 (cited on pages 1, 66).

[90] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956 (cited on page 26).

[91] L. J. Latecki and R. Lakämper, "Application of planar shape comparison to object retrieval in image databases," *Pattern Recognition*, vol. 35, no. 1, pp. 15–29, 2002, Shape representation and similarity for image databases, ISSN: 0031-3203 (cited on page 45).

[92] T.-C. Lee, R. L. Kashyap, and C.-N. Chu, "Building skeleton models via 3-d medial surface axis thinning algorithms," *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462–478, 1994 (cited on page 46).

[93] J. Liang, L. Bai, C. Dang, and F. Cao, "The k-means-type algorithms versus imbalanced data distributions," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 4, pp. 728–745, 2012 (cited on page 71).

[94] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, and S. Wu, "Understanding and enhancement of internal clustering validation measures," *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 982–994, 2013 (cited on pages 41, 107).

[95] Y. Liu, H. Yang, and W. Wang, "Reconstructing b-spline curves from point clouds–a tangential flow approach using least squares minimization," in *Shape Modeling and Applications, 2005 International Conference*, IEEE, 2005, pp. 4–12 (cited on page 60).

[96] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter, "Annotated high-throughput microscopy image sets for validation," *Nat Methods*, vol. 9, no. 7, p. 637, 2012 (cited on pages 2, 167).

[97] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Berkeley, Calif.: University of California Press, 1967, pp. 281–297 (cited on page 7).

[98] P. C. Mahalanobis, "On the generalised distance in statistics," *Proceedings of the National Institute of Sciences of India, 1936*, pp. 49–55, 1936 (cited on pages 11, 43, 48).

[99] G. Mercator, *Nova et aucta orbis terræ descriptio ad usum navigantium emendate accomodata*, Map, Duisburg, 1569 (cited on page 49).

[100] D. P. Mitchell, "Spectrally optimal sampling for distribution ray tracing," in *ACM SIGGRAPH Computer Graphics*, ACM, vol. 25, 1991, pp. 157–164 (cited on pages 71, 163).

[101] J. Moffat, D. A. Grueneberg, X. Yang, S. Y. Kim, A. M. Kloepfer, G. Hinkle, B. Piqani, T. M. Eisenhaure, B. Luo, J. K. Grenier, A. E. Carpenter, S. Y. Foo, S. A. Stewart, B. R. Stockwell, N. Hacohen, W. C. Hahn, E. S. Lander, D. M. Sabatini, and D. E. Root, "A lentiviral RNAi library for human and mouse genes applied to an arrayed viral high-content screen," *Cell*, vol. 124, no. 6, pp. 1283 –1298, 2006, ISSN: 0092-8674 (cited on page 2).

[102] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM International Conference on Data Mining*, SIAM, 2014, pp. 839–847 (cited on pages 42, 119, 120).

[103] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," *Computer Vision–ECCV 2006*, pp. 490–503, 2006 (cited on page 2).

[104] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979 (cited on page 110).

[105] K. Palágyi, "A 3d fully parallel surface-thinning algorithm," *Theor. Comput. Sci.*, vol. 406, no. 1-2, pp. 119–135, Oct. 2008, ISSN: 0304-3975 (cited on page 45).

[106] L. Piegl and W. Tiller, *The NURBS Book*, 2nd. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996 (cited on page 62).

[107] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Labs Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957 (cited on page 26).

[108] D. H. Raab and E. H. Green, "A cosine approximation to the normal distribution," *Psychometrika*, vol. 26, no. 4, pp. 447–450, 1961 (cited on page 69).

[109] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971 (cited on page 23).

[110] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *EMNLP-CoNLL*, vol. 7, 2007, pp. 410–420 (cited on page 25).

[111] G.-C. Rota, "The number of partitions of a set," *The American Mathematical Monthly*, vol. 71, no. 5, pp. 498–504, 1964 (cited on page 6).

[112] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987 (cited on page 31).

[113] T. Sabsch, C. Braune, A. Dockhorn, and R. Kruse, "Using a multi-objective genetic algorithm for curve approximation," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2017 (cited on page 60).

[114] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on computers*, vol. 100, no. 5, pp. 401–409, 1969 (cited on page 43).

[115] (). SIGKDD News: 2014 SIGKDD TEST OF TIME AWARD, SIGKDD, [Online]. Available: `http://www.kdd.org/News/view/2014-sigkdd-test-of-time-award` (cited on page 13).

[116] M. Smith and J. March, *March's Advanced Organic Chemistry: Reactions, Mechanisms, and Structure*. Wiley, 2007, ISBN: 978-047-008-494-6 (cited on page 73).

[117] J. P. Snyder, *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997 (cited on page 49).

[118] M. Spiliopoulou, L. Schmidt-Thieme, and R. Janning, *Data Analysis, Machine Learning and Knowledge Discovery*, ser. Studies in Classification, Data Analysis, and Knowledge Organization. Cham: Springer International Publishing, 2014, p. 470, ISBN: 978-3-319-01594-1 (cited on page 17).

[119] A. Tannenbaum, "Three snippets of curve evolution theory in computer vision," *Mathematical and Computer Modelling*, vol. 24, no. 5, pp. 103–119, 1996, ISSN: 0895-7177 (cited on page 45).

[120] R. L. Thorndike, "Who belongs in the family?" *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953 (cited on pages 2, 30, 34).

[121] L. W. Tu, *An Introduction to Manifolds*. New York: Springer-Verlag New York, 2011, ISBN: 978-1-4419-7399-3 (cited on page 49).

[122] C. J. Veenman, M. J. T. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 9, pp. 1273–1280, 2002 (cited on pages 152, 153).

[123] L. Vendramin, R. J. Campello, and E. R. Hruschka, "Relative clustering validity criteria: A comparative overview," *Statistical analysis and data mining: the ASA data science journal*, vol. 3, no. 4, pp. 209–235, 2010 (cited on page 103).

[124] L. Vendramin, P. A. Jaskowiak, and R. J. Campello, "On the combination of relative clustering validity criteria," in *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, ACM, 2013, p. 4 (cited on page 103).

[125] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Is a correction for chance necessary?" In *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 1073–1080 (cited on page 23).

[126] G. Voronoï, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs.," *Journal für die reine und angewandte Mathematik*, vol. 134, pp. 198–287, 1908 (cited on page 9).

[127] L. Wang, X. Geng, J. Bezdek, C. Leckie, and R. Kotagiri, "SpecVAT: Enhanced visual cluster analysis," in *Eighth IEEE International Conference on Data Mining, 2008. ICDM'08*, IEEE, 2008, pp. 638–647 (cited on page 27).

[128] L. Wang, U. T. Nguyen, J. C. Bezdek, C. A. Leckie, and K. Ramamohanarao, "IVAT and aVAT: Enhanced visual analysis for cluster tendency assessment," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2010, pp. 16–27 (cited on page 27).

[129] R. Winkler, F. Klawonn, and R. Kruse, "Fuzzy c-means in high dimensional spaces," *International Journal of Fuzzy System Applications (IJFSA)*, vol. 1, no. 1, pp. 1–16, 2011 (cited on pages 11, 99).

[130] ——, "Prototype based fuzzy clustering algorithms in high-dimensional feature spaces," in *Enric Trillas: A Passion for Fuzzy Sets*, Springer, 2015, pp. 233–243 (cited on pages 11, 99).

[131] J. Wu, J. Chen, H. Xiong, and M. Xie, "External validation measures for k-means clustering: A data distribution perspective," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6050–6061, 2009 (cited on page 22).

[132] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, no. 8, pp. 841–847, 1991 (cited on pages 34, 38).

[133] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010 (cited on page 66).

[134] C. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *Computers, IEEE Transactions on*, vol. C-20, no. 1, pp. 68–86, Jan. 1971, ISSN: 0018-9340 (cited on page 151).

[135] T. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984 (cited on page 45).

[136] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration," in *Evolutionary multi-criterion optimization*, Springer, 2007, pp. 862–876 (cited on page 63).

List of Figures

List of Tables

Acronyms

AMI Adjusted Mutual Information. 24

ANOVA Analysis of Variance. 33

ARI Adjusted Rand Index. 23, 28, 101, 104

CH Calinski-Harabasz Index. 33, 85, 108

CVNN Clustering Validation Index based on Nearest Neighbors. 38, 42, 105, 106

DB Davies-Bouldin Index. 30, 31

DCE Discrete Curve Evolution. 45, 82

DI Dunn's Index. 32, 104

DLI Dunn-like Index. 32

FCLS Fuzzy-c-Line Segments. 73, 80, 82, 92, 93, 96, 97

FS Fukuyama-Sugeno Index. 33, 34

GDA Generalized Discriminant Analysis. 42, 43

GNG Growing Neural Gas. 43, 80–82, 93, 96, 97

GWSO Glow Worm Swarm Optimization. 59, 66, 67, 82, 93, 96, 97

LDA Linear Discriminant Analysis. 42

MI Mutual Information. 23, 24, 29

MRD Mutual Reachability Distance. 41

MST Minimum Spanning Tree. 26, 59, 78, 82

NMI Normalized Mutual Information. 25

NSGA-II Non-Dominated Sorting Genetic Algorithm II. 60, 62

ODI Ordered Distance Image. 26, 27

PC Partition Coefficient. 37, 38, 107

PCA Principal Component Analysis. 42, 80, 97

PE Partition Entropy. 38, 100, 104, 118

PSO Particle Swarm Optimization. 65–67

RBF Radial Basis Function. 67

RI Rand Index. 23, 24, 29

RMSE Root Mean Squared Error. 32, 33

ROC Receiver Operating Characteristic. 107

SC Silhouette Coefficient. 31

SOM Self-organizing Map. 43, 79, 82

SSE Sum of Squared Errors. 65, 100

VAT Visual Assessment of Cluster Tendency. 26, 27, 29

XB Xie-Beni Index. 34, 85

Benchmark Data Sets

This chapter shows some benchmark data sets commonly used for clustering. Every plot shows the ground truth labeling if a ground truth was available. Otherwise it will show a good fit resulting from a clustering algorithm (parameters given in the plot's caption).

A.1 T4.8k

The `t4.8k` data set [81] contains 8.000 two-dimensional data points structured into six clusters. A seventh cluster might be recognized as a sine curve in the background connecting the remaining clusters. This data set also contains a large amount of unstructured noise and two clusters with strongly overlapping convex hulls.

A.2 Aggregation

The `aggregation` data set [66] contains 788 two-dimensional data points structured into seven clusters. It contains connected clusters as well as two clusters which are almost indistinguishable close together. Two points within the moon-shaped cluster to the top left are irregularly far apart from the remaining points in the cluster. This sometimes leads to the detection of yet another cluster.

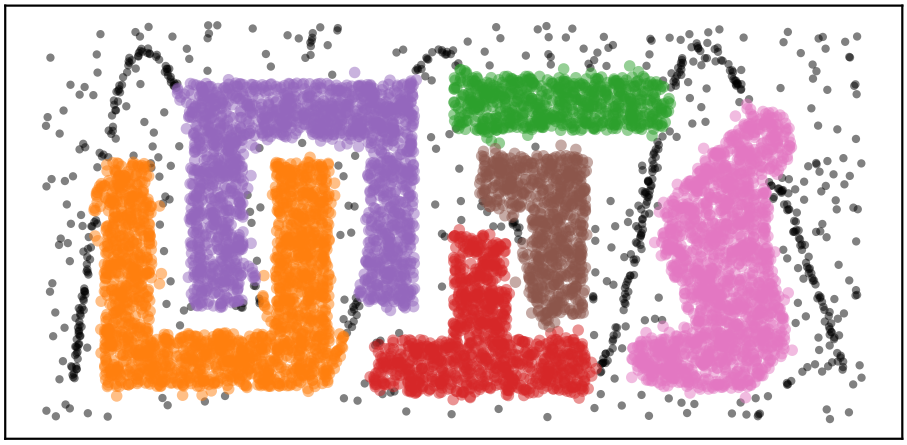


Figure A.1: The `t4.8k` data set. The labeling is the result of running DBSCAN with $\varepsilon = 11.5$ and $minPts = 30$, since no ground truth labeling was availbale fo this data set.

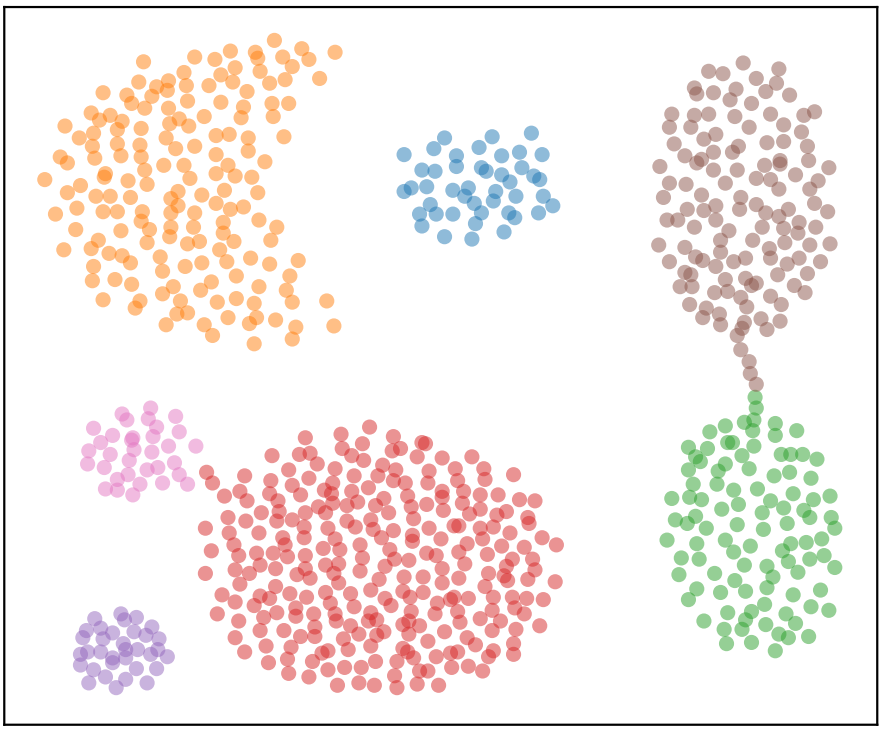


Figure A.2: The `aggregation` data set.

A.3 Compound

The `compound` data set [134] contains 399 two-dimensional data points structured into six clusters. One cluster is completely contained within the other, another one is surrounded by very structured noise (here counted as cluster) and the two remaining clusters are located close to each other with one point apparently mislabeled.

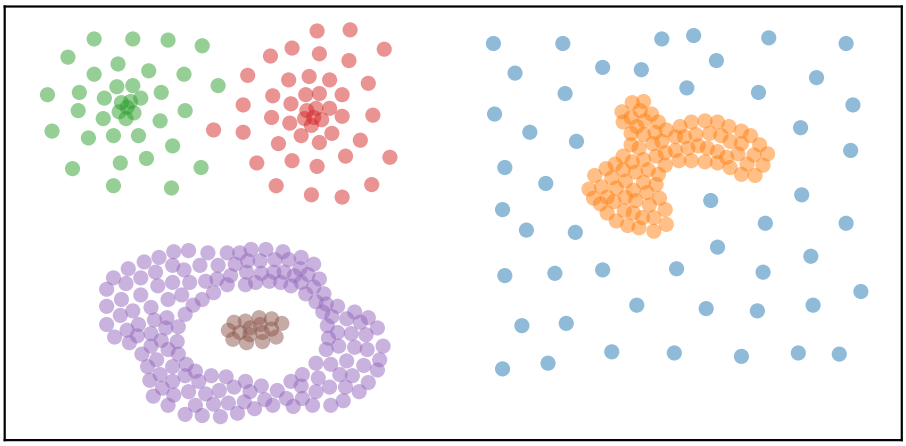


Figure A.3: The `compound` data set.

A.4 Path-Based

`path-based` [40] contains 300 two-dimensional data points structured into three clusters. One cluster forms almost a full circle in which two gaussian blobs are placed. These blobs lie very close to the circle, making it hard for density-based algorithms to distinguish them from each other.

A.5 Spiral

The `spiral` data set [40] contains 312 two-dimensional data points structured into three clusters. Originally the clusters hae zero width and are well separated. However by adding some gaussian noise onto each data

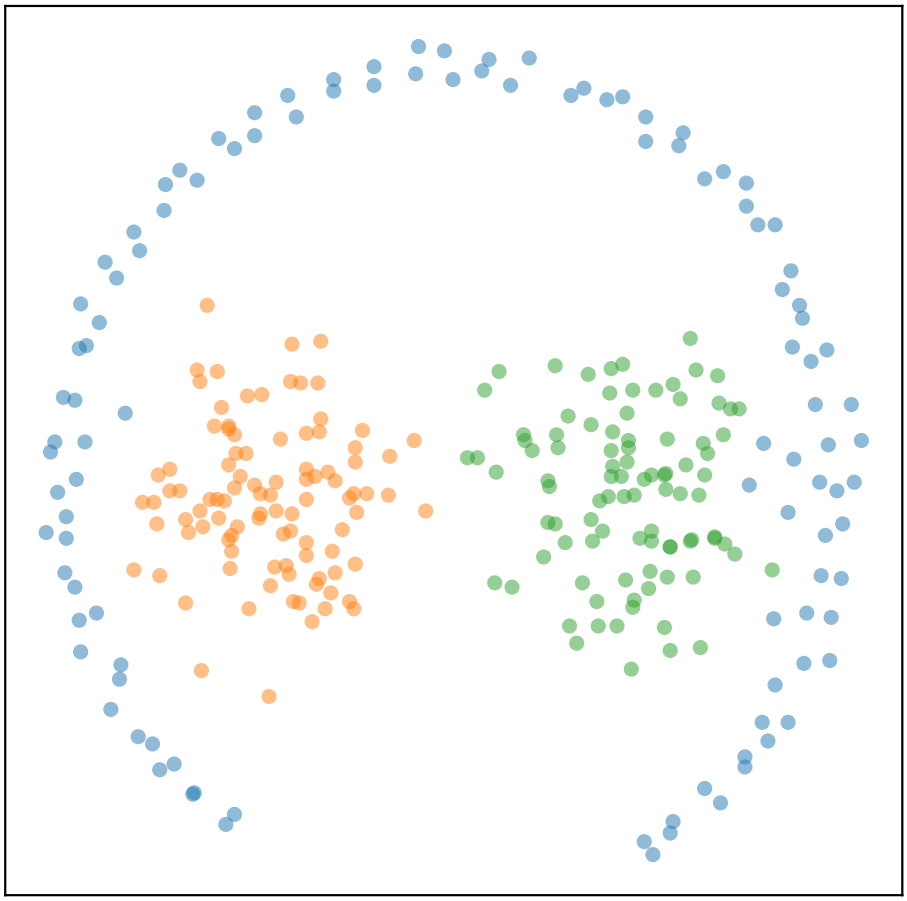


Figure A.4: The `pathbased` data set.

point (and drawing points with replacement from the original data set), three clusters with non-zero width can be generated. If the magnitude of the added noise is chosen too large, the clusters might become indistinguishable.

A.6 D31

The `d31` data set [122] contains 3100 two-dimensional data points structured into 31 clusters. All of the 31 clusters are formed by sampling 100 points from a bivariate normal distribution with equal covariance matrices.

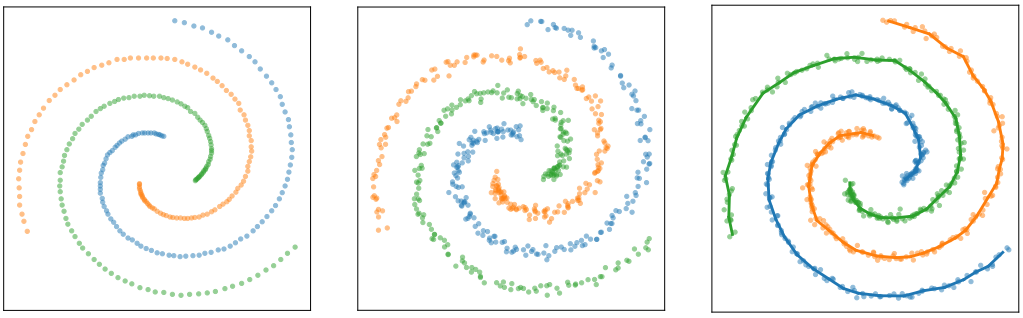


Figure A.5: The `spiral` dataset (left: without additional noise, center wiht noise, right: clustered, with skeletons).

The location of the cluster centers are chosen in such a way that the clusters overlap or are at least not well separated, thus making them almost indistinguishable for any density-based clustering algorithm.

A.7 R15

The `r15` data set [122] contains 600 two-dimensional data points structured into 15 clusters. Each cluster contains 40 samples drawn from a bivariate normal distribution and seven clusters each are placed along two nested circles. The last cluster is placed in the center of the two circles. Due to the radii of the clusters with respect to the inner circle, it might appear to some algorithms (or parametrizations) that the data set contains only eight clusters (the eight central ones might be considered a single cluster).

A.8 Jain's Toy Dat Set

The `jain` data set [80] contains 373 two-dimensional data points structured into two clusters. The two clusters are similar two the half-circle data set with the difference that the two clusters have varying density. One problem with this data set is that the lower density half circle contains a small region where no point at all is located. This is no problem for a human, but a lot of algorithm recognize this as two separate clusters.

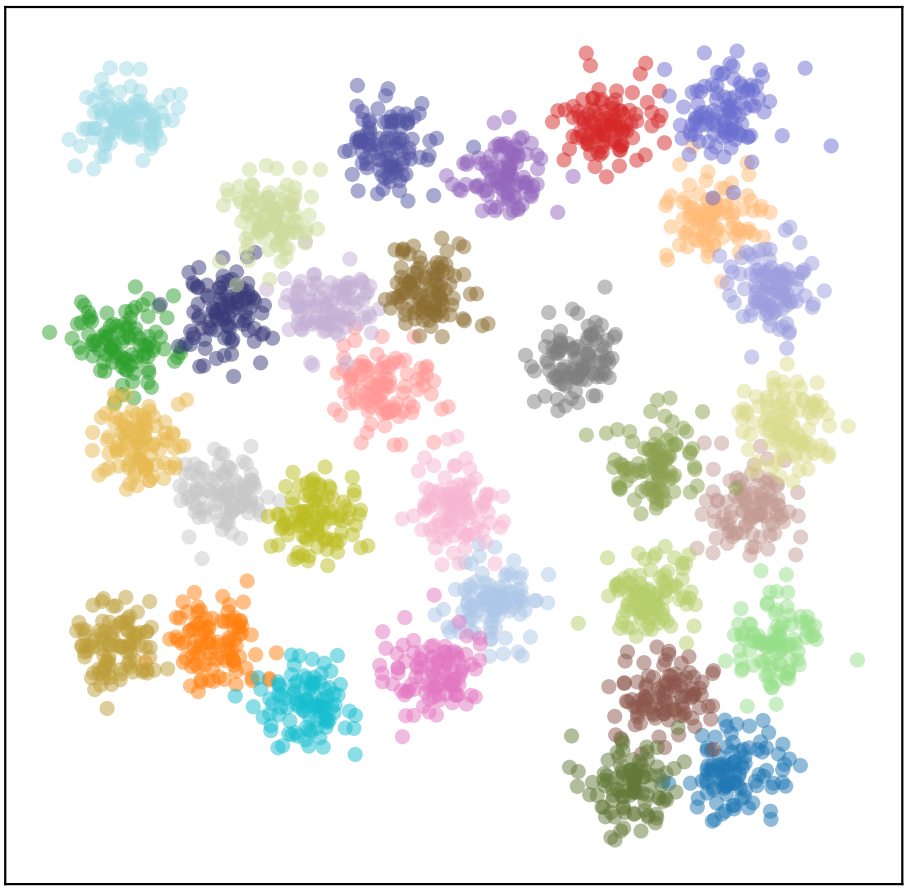


Figure A.6: The `d31` data set.

A.9 Flame

The `flame` data set [64] contains 240 two-dimensional data points structured into two clusters. The two clusters are not well separated and lie very close together. One of the clusters also contains two outlier points.

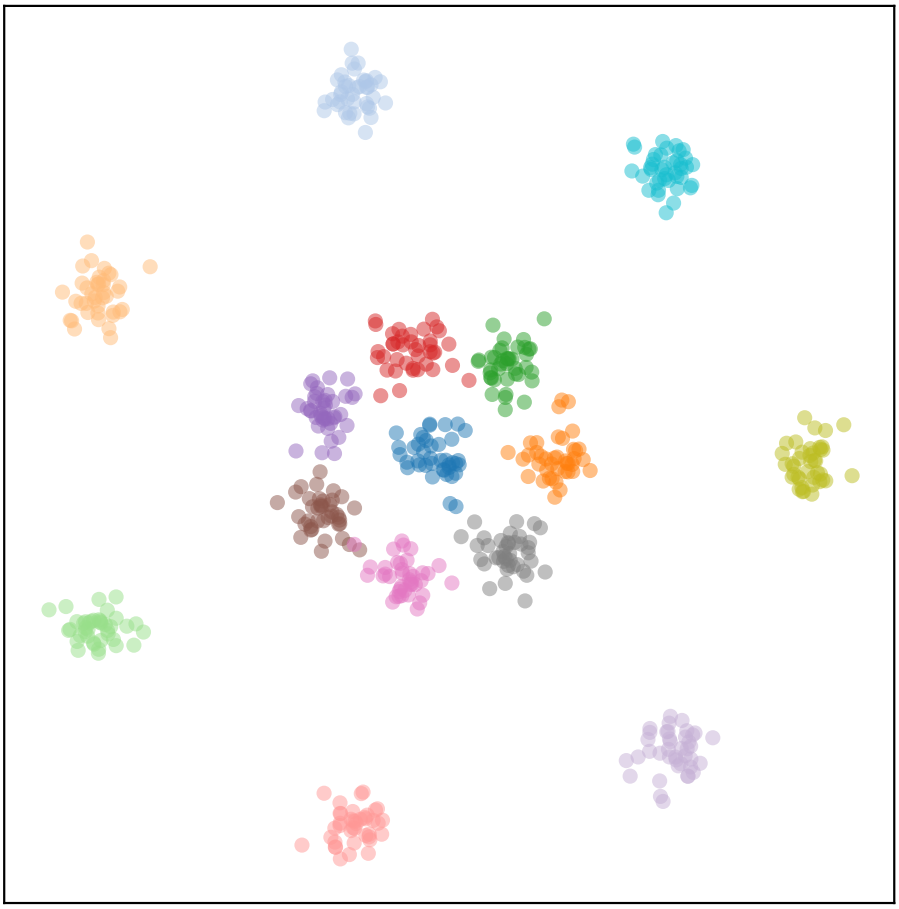


Figure A.7: The `r15` data set.

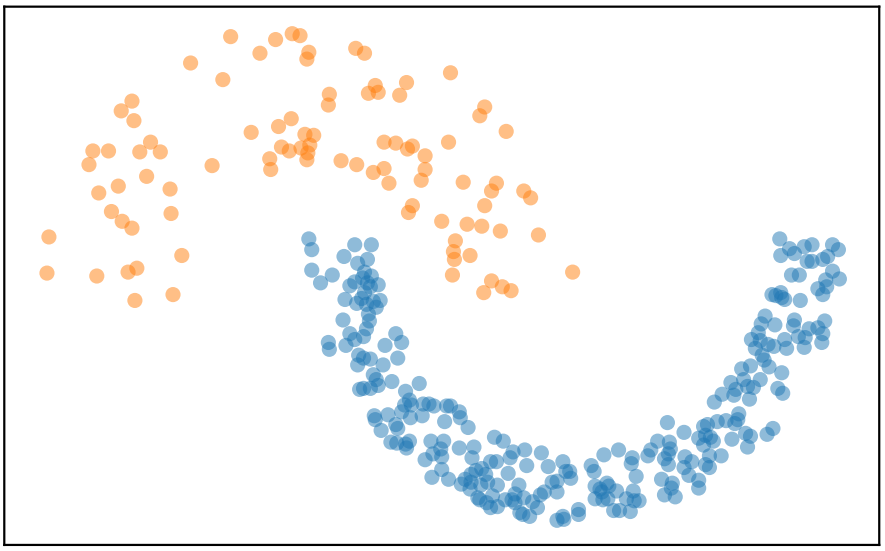


Figure A.8: The `jain` data set.

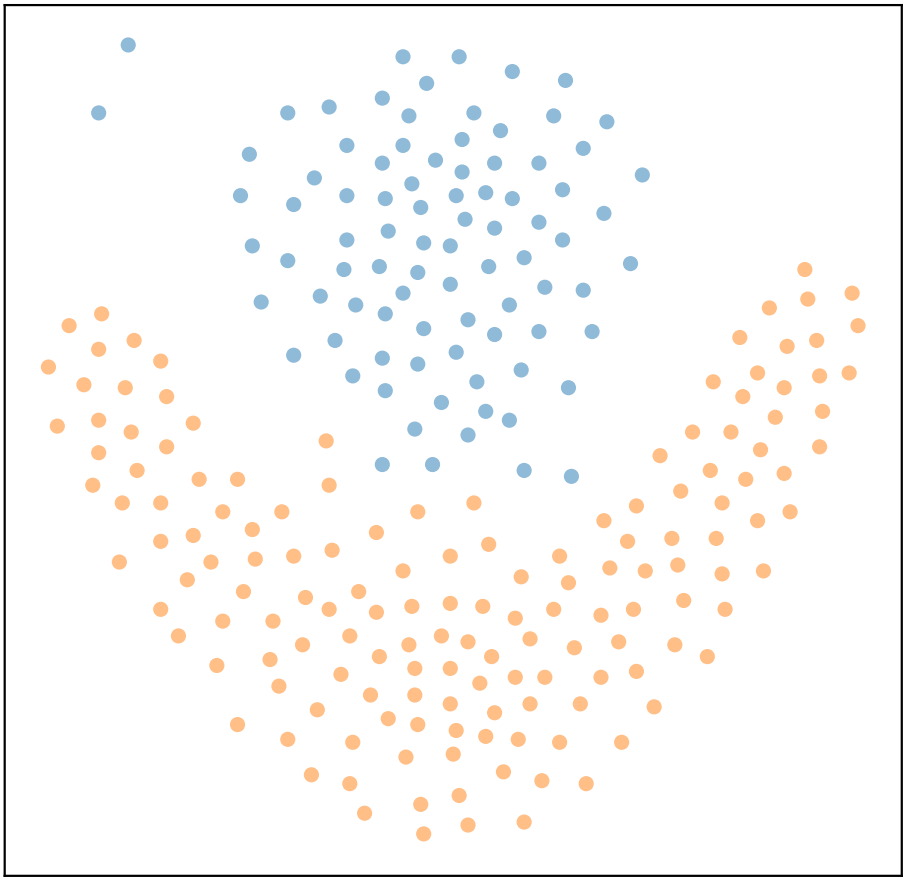


Figure A.9: The `flame` data set.

APPENDIX B

Dataset Generation

This chapter explains the different procedures according to which the data sets throughout this thesis have been generated. For the sake of completeness the general procedure of creating data sets will also be explained if the methods have been taken from publicly available libraries.

B.1 Blobs

Hyperspherical clusters are created by sampling center points from a uniform, d-dimensional distribution within the desired boundaries. These centers are then used in a multivariate gaussian distribution with specified variance. Depending on the intial location of the clusters' centers the clusters may overlap, making them indistinguishable for a (density-based) clustering algorithm.

B.2 Nested Circles

Circular clusters a generated by equiangular spaced point on a unit circle. For the inner circle, all a circle with smaller radius (by a user-defined factor, $0 < f \leq 1$) is used for sampling the corresponding data points. Gaussian noise is of a user-defined magnitude is then added to all the points in the data set. Depending on the magnitude of the noise and the factor for the inner circles radius the resulting clusters may overlap, making them indistinguishable for a (density-based) clustering algorithm.

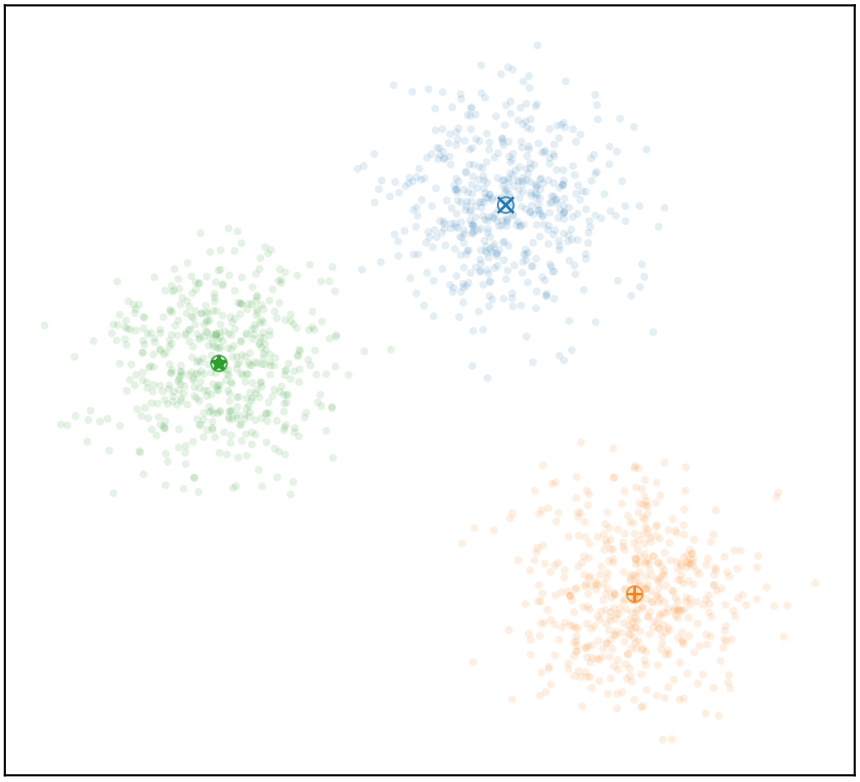


Figure B.1: Two-dimensional example of hyperspherical clusters and their respective center points.

B.3 Eightshape

The generation of clusters which follow this model is similar to that of the nested circles (see Section B.2). Instead of solely applying a factor to the main circles radius, the second circle is also translated in such a way, that the to circles touch at one point. The result is a single cluster structure which resembles the shape of an eight. Additionally, different noise setting can be applied to the individual circles to generate clusters with varying density.

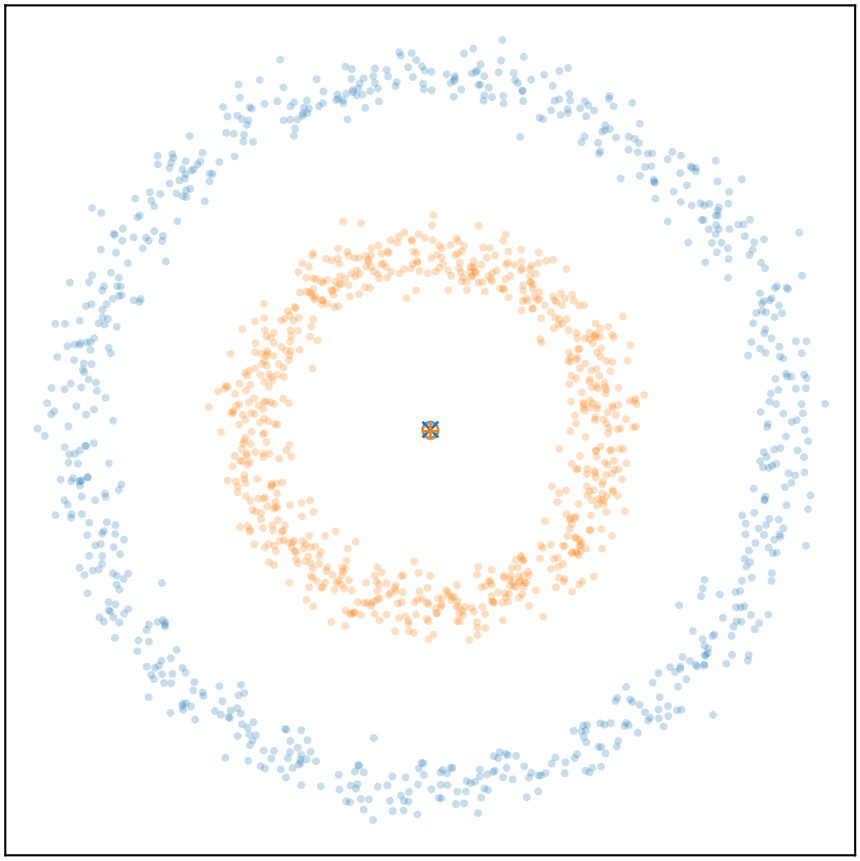


Figure B.2: Two nested circles with their respective centroids.

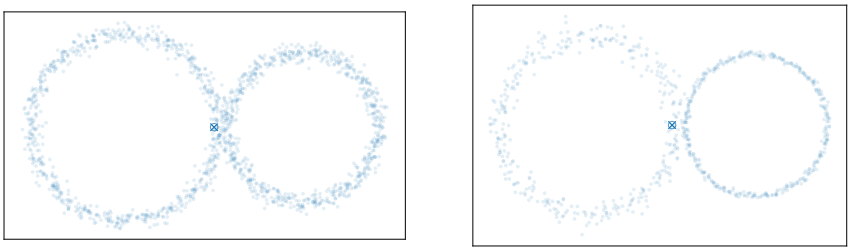


Figure B.3: Two versions of the eight-shaped cluster.

B.4 Half Circles

The moon-shaped data set is generated similar to the nested circles (see Section B.2). Instead of generating to separate circles, only a single unit circle is generated and split into a lower and upper half. The lower half is

then translated to the lower right so that one of its end points lies at the location of the initial circle's center. Gaussian noise is of a user-defined magnitude is then added to all the points in the data set. Depending on the magnitude of the noise the resulting clusters may overlap, making them indistinguishable for a (density-based) clustering algorithm.

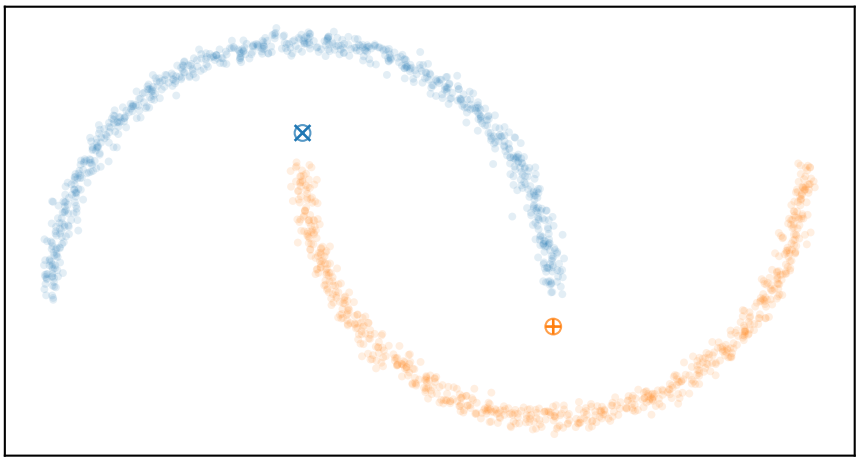


Figure B.4: Two nested half circles with their respective centroids.

B.5 Figure χ-Shape

The generation of clusters which follow the χ-shape model is similar to that of the half circles (see Section B.4). Instead of translating the lower half of the initial circle to the lower right, it is moved up by one unit. This makes the lowest point of the initial lower half coincide with the highest point of the initial upper halfcircle. The resulting structure is shaped like a χ and forms a single structure. Gaussian noise is of a user-defined magnitude is then added to all the points in the cluster.

B.6 B-Spline Cluster

To generate clusters that follow a B-Spline model, the control points are sampled from a uniform distribution in a way that the line is guaranteed to

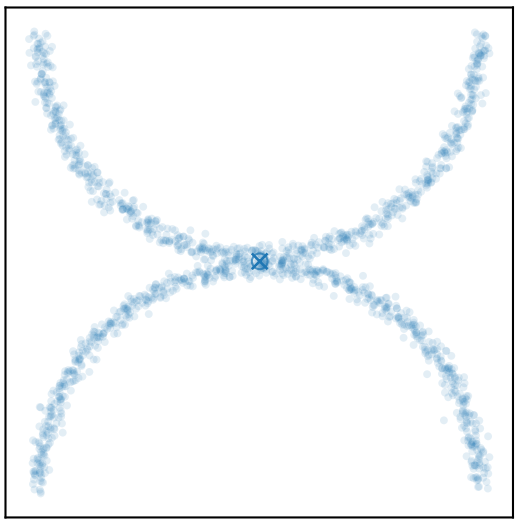


Figure B.5: Two nested half circles with their respective centroids.

be non self-intersecting by sampling a certain number of points uniformly from the desired bounding box and then sorting along the first axis. These points uniquely define the b-spline curve. Within the bounding box blue noise is created by either using Poisson disc sampling [36] or Mitchell's best candidate [100]. In contrast to white noise (points are randomly sampled from a uniform distribution), blue noise guarantees to have approximately equal density across every proper sub-neighborhood of the generated data. From the blue noise sample only the points closer than a threshold ∂ are kept in the data set.

B.7 Skeleton Cluster

The structural disadvantage of the χ-shape clusters is that they rarely show any pronounced branching. Clusters always follow a single line and only if two adjacent control vectors are very close to each other in one dimension something like branching might be observed. The skeleton clusters use a different model for cluster generation. In some sense this model is similar to the spherical cluster from Appendix B.1. But instead of a single point a cluster skeleton is sampled. This is done by choosing $b + 1$ points that are sampled from a multivariate uniform distribution.

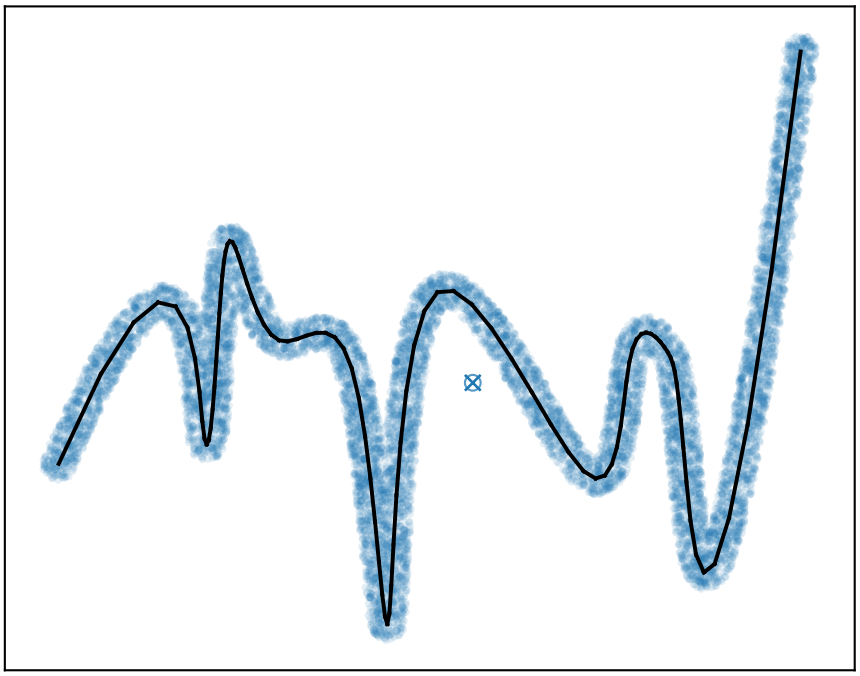


Figure B.6: Two-dimensional example of a b-spline cluster and the corresponding b-spline curve.

On these points a minimum spanning tree is calculated and each edge in this tree is subsampled into a specifiable number of points. Onto these point gaussian noise with some user-specified variance can be added. The resulting structure is usually highly dendritic in nature. Since all edges receive the same number of points, these clusters will also exhibit varying density along shorter edges and vertices where more than two edges join.

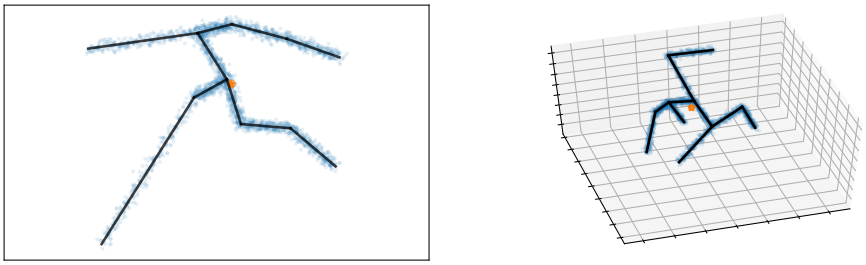


Figure B.7: Examples of skeleton clusters in 2d and 3d and the corresponding *bone structure*. For better visibility the clusters' arithmetic means have been highlighted in a different color.

Acknowledgements

The font used in this thesis is called *Kerkis* and is copyrighted by the Department of Mathematics, University of the Aegean.

This thesis used image set BBBC001v1 [39] from the Broad Bioimage Benchmark Collection [96].

Ehrenerklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, den 26.06.2018

Christian Braune