Finding, Evaluating and Exploring Clustering Alternatives Unsupervised and Semi-supervised

by

Davoud Moulavi

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science
University of Alberta

Abstract

Clustering aims at grouping data objects into meaningful clusters using no (or only a small amount of) supervision. This thesis studies two major clustering paradigms: density-based and semi-supervised clustering. Density-based clustering algorithms seek partitions with high-density areas of points (clusters that are not necessarily globular) separated by low-density areas that may contain noise objects. Semi-supervised clustering algorithms use a small amount of information about data to guide the clustering task.

In the context of density-based clustering, we study *(a)* the validation of density-based clustering and *(b)* hierarchical density-based clustering.

The validation of density-based clustering, i.e., the objective and quantitative assessment of clustering results, is one of the most challenging aspects of clustering. Numerous different relative validity criteria have been proposed for the validation of *globular* clusters. Not all data, however, are composed of globular clusters. We propose a relative density-based validation index, DBCV, that assesses the quality of an arbitrarily-shaped clustering based on the relative density connection between pairs of objects. Our index is formulated on the basis of a new kernel density function, which is used to compute the density of objects and to evaluate the within- and between-cluster density connectedness

of clustering results.

In addition to the DBCV, we make several major contributions in the area of hierarchical density-based clustering. We improve on the AUTO-HDS framework for automated clustering and visualization of biological data sets by removing a parameter thereby making the cluster extraction stage simpler and more accurate. We also propose a theoretically and practically improved general hierarchical density-based clustering, called GHDBSCAN, which generalizes the density-based clustering by recognizing its essential components and based on this generalization we propose two algorithms, GHDBSCAN(NMRD) and GHDBSCAN(NMRD+PF), which improve over previous state-of-the-art methods both theoretically and practically.

Regarding semi-supervised clustering, we use the knowledge available about a dataset in the form of constraints to guide the clustering algorithm. In this context, we provide two approaches for model selection that allow the user to select the best model based on few constraints and/or the DBCV value and also discuss a framework for extracting a partitional clustering from a hierarchical clustering tree.

Preface

Part of the research conducted in this thesis was done in a research collaboration led by my PhD supervisor Professor Jörg Sander at the University of Alberta.

The main part of Chapter 3 has been published as D. Moulavi, P.A. Jaskowiak, R.J.G.B. Campello, A. Zimek, J. Sander. "Density-Based Clustering Validation," Proceedings of the 14^{th} SIAM International Conference on Data Mining (SDM), Philadelphia, PA, USA, 2014. I was the main contributor of the key ideas and writing the manuscript in this paper, and participated in conducting experiments and analyzing the results. P.A. Jaskowiak was responsible for part of the experiments. P.A. Jaskowiak, R.J.G.B. Campello and J. Sander, were involved in developing ideas and writing the manuscript. A. Zimek was involved in discussing the ideas and editing the manuscript.

Parts of Chapter 4 are described in two published works.

The work described in Section 4.2 has been published in R.J.G.B. Campello, D. Moulavi, J. Sander. "A Simpler and More Accurate AUTO-HDS Framework for Clustering and Visualization of Biological Data," IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), Vol. 9, No. 6, 1850-1852, 2012. I was involved in the discussions of the main ideas of the paper, and in particular I was the main contributor for both the formulation and the mathematical proof of the idea of eliminating a parameter from the cluster extraction stage of AUTO-HDS; furthermore, I was responsible for conducting and analyzing the experiments and participated in writing the manuscript. R.J.G.B. Campello was the main contributor of the ideas for computing the AUTO-HDS hierarchy in a more efficient way and in writing the paper and analyzing the results. J. Sander was involved in the discussions and refinement

of the ideas, analyzing the results and editing the manuscript.

The algorithm HDBSCAN, described in Section 4.3.1, which forms the basis and starting point for my own, original extensions of this work in Chapter 4, was published in R.J.G.B. Campello, D. Moulavi, J. Sander. "Density-Based Clustering Based on Hierarchical Density Estimates," Proceeding Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), LNAI 7819, Gold Coast, Australia, 2013, 160-172. For this paper, I participated in the discussions of the main ideas, conducted all the experiments, analyzed the results with R.J.G.B. Campello and J. Sander, and participated in the writing of the manuscript. R.J.G.B. Campello was the main contributor in developing the key ideas and in writing the manuscript. J. Sander was also involved in developing the ideas and writing the manuscript.

Parts of Chapter 5 are part of two published works.

One of the methods described in Section 5.4 was published in M. Pourrajabi, D. Moulavi, R.J.G.B. Campello, A. Zimek, J. Sander, R. Goebel. "Model Selection for Semi-Supervised Clustering," Proceeding of the 17^{th} International Conference on Extending Database Technology (EDBT), Athens, Greece, 2014. For this paper, M. Pourrajabi and I were the main contributors to the development of the method, conducting the experiments, and analyzing the results, and I also participated in writing the manuscript. The other authors were all involved in discussing and refining the ideas, as well as in analyzing the results and writing/editing the manuscript.

The concepts and method presented in Sections 5.2 and 5.3 were published in R.J.G.B. Campello, D. Moulavi, A. Zimek, J. Sander. "A Framework for Semi-Supervised and Unsupervised Optimal Extraction of Clusters from Hierarchies," Data Mining and Knowledge Discovery (DMKD), Vol. 27, 344-371, 2013. For this paper, I conducted all experiments, analyzed the results with the other authors, and participated in the discussions and refinements of the ideas and the writing of the manuscript. R.J.G.B. Campello was the main contrib-

utor to the ideas and the writing of the manuscript; A. Zimek and J. Sander contributed to the refinement of the ideas and participated in the writing of the manuscript.

Acknowledgements

There are many people that I would like to thank for their help and support during the past years. While I cannot mention all of their names here, I am nonetheless grateful to all of them. Without them, many accomplishments in this thesis would have been much more difficult to achieve. I would like, however, to express my gratitude to a few people who have influenced my study in the past few years.

I would like to thank my supervisor, Jörg Sander for his constant help and support, insightful discussions and, constant feedback. He was always available for discussion, even at times when he was busy with deadlines, courses, and his graduate chair duties. He made this work possible by providing a supportive milieu and team. I learned a great deal from him during these years. Thank you Jörg!

I'm also thankful to Ricardo J.G.B. Campello, who has been supportive during the past few years. This work would not have been initiated without the cooperation of Ricardo. I've learned many things by working with him. In addition, I have collaborated with several great researchers, benefiting from their wisdom and learning from them. Along with Jörg and Ricardo, I would like to thank my collaborators and coauthors, including Russell Greiner, Arthur Zimek and Pablo A. Jaskowiak. I also extend thanks to the other members of my committee, Osmar R. Zaïane and Myra Spiliopoulou.

I have had a chance to see many wonderful friends and enjoy their friendship since the beginning of my stay in Edmonton. I would like to thank all of them, and due to space limitation I only name a few of them here: Nicholas M. Boers, Xiye Wang, Hesam Izakian, Mohammad Raoufi, Jonathan Snook, Reza Sadod-

din, Pablo A. Jaskowiak, Arthur Zimek, Koosha Golmohammadi, Amin Jorati, Hossein Azari, Parisa Naeimi, Pirooz Chubak, Farzad Sangi, AmirAli Sharifi, Amir-massoud Farahmand, MohammadAli Salehian, Sajib Barua, Mojgan Pourrajabi, Arash Karimi, Arash Afkanpour, Ashkan Zarnani, Ali Shahbazi, Mostafa Vafadoost, Himanshu Vashishta, Shahab Jabbari, Shahin Jabbari, Siamak Ravanbakhsh and many other friends. I am glad I have met you all.

I would also like to thank my family for their love and support throughout my life. I especially thank my parents, who always provided me with an environment in which I could pursue my dreams and achieve my goals.

Thank you all!

Contents

List of Tables

List of Figures

List of Abbreviations

MST	Minimum Spanning Tree
EM	Expectation Maximization
EMGM	Expectation Maximization for Gaussian Mixtures
DBCV	Density-Based Clustering Validation
HMA	Hierarchical Mode Analysis
OPTICS	Ordering Points To Identify the Clustering Structure
AUTO-HDS	Automated Hierarchical Density Shaving
GHDBSCAN	Generalized Hierarchical Density-Based Clustering
SL	Single-Linkage
MRD	Mutual Reachability Distance
NMRD	New Mutual Reachability Distance
PF	Parameter Free
FOSC	Framework for Optimal Extraction of Clusters from Hierarchies
ARI	Adjusted Rand Index
ALOI	Amsterdam Library of Object Images
GSS-MS	General Semi-Supervised Model Selection

Chapter 1

Introduction and Motivation

Over the past decades a variety of high dimensional data that contain different types of data have been collected at an increasing rate. These data can be from different applications, including genetic information, medical images and geographical image data. Many of these data sets contain valuable information in their structure that can be revealed using data mining techniques.

One of the primary unsupervised learning data mining tasks is clustering. Although there is no single consensus on the definition of a cluster, the clustering procedure can be characterized as the organization of data into a finite set of categories by abstracting their underlying structure, either by grouping objects in a single partition or by constructing a hierarchy of partitions to describe data according to similarities or relationships among its objects [1, 2, 3]. Clustering methods have broad application in many areas, including medicine and bioinformatics, financial markets, anomaly detection, image segmentation, web mining, and education, to mention just a few [4, 3, 5, 6].

In recent decades, different clustering definitions have given rise to a number of clustering algorithms, showing significant field development, and clustering techniques have become the subject of active research in several fields such as statistics, pattern recognition and machine learning [7, 8, 9]. It is useful to distinguish between the unsupervised classification of data, which normally

occurs in clustering, and the supervised classification. In supervised classification labels of objects are available, and are used in the process of the class's description learning, which in turn is used for the problem of finding the new object's class label. However, in clustering there is little or no prior information available about the object labels, and the problem is to categorize the unlabeled data into clusters. In the case of semi-supervised clustering, the problem is to categorize the data into clusters using partially labeled objects or other kinds of constraints (e.g., pairwise should-link and should-not-link constraints) [10, 11]. Unsupervised learning is also used in other data analysis approaches such as dimensionality reduction methods which try to discover compact representations of high-dimensional data, e.g., locally linear embedding (LLE) is an unsupervised learning method that computes low-dimensional, neighborhood-preserving representations of high dimensional data [12].

In this study we examine two major paradigms for clustering algorithms. First, in the context of density-based clustering, we study an approach to validating arbitrarily-shaped clustering solutions. This approach enables users to compare algorithms and select parameters. We also study hierarchical density-based clustering to generate a complete hierarchy describing data in different granularity or density thresholds. Second, in the context of semi-supervised clustering, we study semi-supervised extraction of flat clusterings from local cuts through cluster hierarchies by considering cluster quality/ constraint satisfaction as an objective function. We also study semi-supervised model selection, by considering constraint satisfaction and cluster quality, to select the most appropriate model for a given problem. In the following sections (1.1 and 1.2), we discuss density-based cluster analysis and semi-supervised clustering.

1.1 Density-Based Cluster Analysis

Density-based clustering is a popular clustering paradigm, and the notion of density is an important topic in statistics and many data mining tasks such

as anomaly detection and image segmentation. The main idea behind density-based techniques for data analysis is that the data set represents a sample from an unknown probability density function (p.d.f.). The estimation of such a p.d.f. for a data set can be tackled using parametric or non-parametric approaches for density-estimation. In parametric approaches the assumption is that the data are drawn from a known parametric family of distributions; for example, the normal distribution of the density f of the underlying data can be estimated by estimating the mean and variance from the data. In non-parametric approaches the assumptions are less rigid and the data can speak for themselves in determining the estimates of f [13].

In cluster analysis, there is also a contrast between parametric and non-parametric approaches. Some popular algorithms, such as k-means and EMGM (expectation maximization for Gaussian mixtures) that correspond to a parametric approach, produce a predetermined number of clusters that tend to be of convex (hyper-spherical or hyper-elliptical) shape. From a statistical point of view, these algorithms produce a predetermined number of clusters of convex shape because an unknown p.d.f. is assumed to be composed of a mixture of k Gaussian distributions, each of which is associated with one of the k clusters supposed to exist in the data (where k typically must be provided by the analyst) [14].

The limitation to convex-shaped clusters is also present in other traditional clustering algorithms, such as average-linkage, Ward's, and related techniques [1], which do not make explicit use of parametric models. In common among these methods there is an underlying principle of "minimum variance," in the sense that all of them directly or indirectly seek to minimize a given measure of variance within clusters. The limitations of such methods, including their inability to find clusters of arbitrary shapes, have encouraged the development of alternative clustering paradigms and related algorithms that allow for more complex structures to be found in the data [15, 16], including

density-based clustering methods. Many density-based clustering algorithms have been proposed in the literature, such as [17, 18, 19, 20, 21] that are popular algorithms that explicitly or implicitly incorporate elements from the theory of non-parametric density estimation [22].

However, existing methods have a number of limitations and pose difficulties for users, who not only have to select the clustering algorithm best suited for a particular task, but also have to properly tune its parameters. The limitations are as follows: (i) There are many density-based clustering methods, so it is hard to compare the results of these methods because usually there is no ground truth available in any given clustering task. (ii) Most approaches have some parameters and the quality of the resulting clustering depends on an appropriate choice of parameters (e.g., ϵ and *MinPts* in DBSCAN [23], smoothness h and the noise level ξ in DENCLUE [24] and r_{shave} in AUTO-HDS [25], along with other parameters in [26, 27, 28]). (iii) Some methods (e.g., DBSCAN [23] and DENCLUE [24]) can only provide a "flat" (i.e. non-hierarchical) labeling of the data objects, based on a global density threshold. Using a single density threshold can often improperly characterize common data sets with clusters of very different densities and/or nested clusters. (iv) Among the methods that provide a clustering hierarchy, some (e.g., gSkeletonClu [29]) are not able to automatically simplify the hierarchy into an easily interpretable representation involving only the most significant clusters. (v) Many hierarchical methods, including OPTICS [30] and gSkeletonClu, suggest only how to extract a flat partition by using a global cut/density threshold, which may not result in the most significant clusters if these clusters are characterized by different density levels. (vi) Some methods are limited to specific classes of problems, such as networks (gSkeletonClu), point sets in the real coordinate space (e.g., DECODE [26], and Generalized Single-Linkage [27]). (vii) Some methods are defined based on only a specific density-estimation method which usually have some limitations.

We discuss the aforementioned issues in more detail in the following sections 1.1.1 and 1.1.2.

1.1.1 Density-Based Clustering Validation

In recent decades, different clustering definitions have given rise to a number of clustering algorithms, resulting a significant development in the field. The variety of clustering algorithms, however, poses difficulties to users, who not only have to select the clustering algorithm best suited for a particular task, but also have to properly tune its parameters. To make such decisions, one requires procedures capable of assessing clustering results in a quantitative and objective fashion, which would allow the user to select which is the best suited result among a collection of results. Such choices are closely related to clustering validation, one of the most challenging topics in the clustering literature, as stated by Jain and Dubes [1]: "*without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage*". More striking than the statement itself is the fact that it still holds true after 26 years, despite all the progress that has been made.

Clustering is an unsupervised learning task and usually there is no ground truth knowledge about data structure; therefore, determining the significance of clustering results remains a large challenge. A common approach to evaluating the quality of clustering solutions involves the use of internal validity criteria [1], which is the validation of results using only the information intrinsic to the data alone. Many of such measures allow one to rank solutions according to their quality and are hence called relative validity criteria. Because internal validity criteria measure the clustering quality based solely on information intrinsic to the data, they have great practical appeal in contrast to the external validity criteria which need ground truth information about data. Thus numerous internal validity criteria have been proposed in the lit-

erature [31, 1, 32, 33]. The vast majority of relative validity criteria are based on the idea of computing the ratio of within-cluster scattering (compactness) to between-cluster separation. Measures that follow this definition have been designed for the evaluation of convex shaped clusters[1] (e.g. globular clusters) and fail when applied to validate arbitrarily-shaped, non-convex clusters. They are also not defined for noise objects. These measures are primarily suitable for partitions with convex shaped clusters that have minimum variance within clusters and maximum separation in terms of the criterion definition. To this extent, such measures are more appropriate in the context of parametric clustering, i.e., when clusters in a data set are drawn from a p.d.f. that is composed, for example, of mixtures of k Gaussian distributions, in which each Gaussian distribution is associated with one of the k clusters in the data [14].

Density-based clusters are relevant to various contexts such as geographical applications, which have clusters of points belonging to rivers, roads, power lines, or any connected shape in image segmentation [20]. Some attempts have been made to develop relative validity measures for arbitrarily-shaped clusters [35, 36, 37, 38, 39]. As we shall see, however, these measures have serious drawbacks that limit their practical applicability.

Only a few attempts so far in the context of arbitrarily-shaped clustering account for adaptations of relative clustering validity criteria that were originally developed for evaluation of convex clusterings. Such measures do not directly take density-based assumptions into account, however. They provide only rough estimates of cluster quality based on the distribution of several representative points from the clusters, such as their cohesiveness and separation, and introduce several parameters that are undesirable to the validation task. For instance, the validation index $CDbw$ [36] that was proposed for assessing arbitrarily-shaped clusters defines the compactness and separation of clus-

[1]That such measures were developed for the evaluation of globular clusters does not necessarily mean that they perform well on such a task [33, 32, 34].

ters similar to clustering validation indices that are used for spherical shaped clusters, and to handle the arbitrarily-shaped density-based clustering, it uses multiple points rather than a single point as representatives in each cluster.

To overcome the lack of appropriate measures for the validation of density-based clusters, we propose a measure called the Density-Based Clustering Validation index (DBCV) that is suitable for validating arbitrarily-shaped clusters. DBCV employs the concept of Hartigan's model of density-contour trees [3] to compute the least dense region inside a cluster and the most dense region between the clusters, which are then used to measure the within- and between-cluster density connectedness of clusters.

In the following Section 1.1.2 we introduce and expand in more detail upon hierarchical density-based clustering and its respective challenges that we briefly discussed in Section 1.1.

1.1.2 Hierarchical Density-Based Clustering and Extraction of a Partition from a Hierarchy

An important paradigm in cluster analysis is hierarchical clustering[2], in which a clustering solution is represented as a tree describing hierarchical relationships between nested clusters [1, 40]. This paradigm has been of particular interest in a large variety of application areas. The reasons are manifold. First, it is a property of real data sets that clusters may be nested. Typically, nested clusters are characterized by having different densities, which is essentially the rationale behind Hartigan's classic definition of *density-contour trees* [3] as a conceptual hierarchical model for data clustering. In addition, in some areas, such as biology, domain experts may prefer tree representations, as they are more accustomed to them. In fact, sometimes the underlying application domain is hierarchically structured in its nature, as it is usually the case in areas

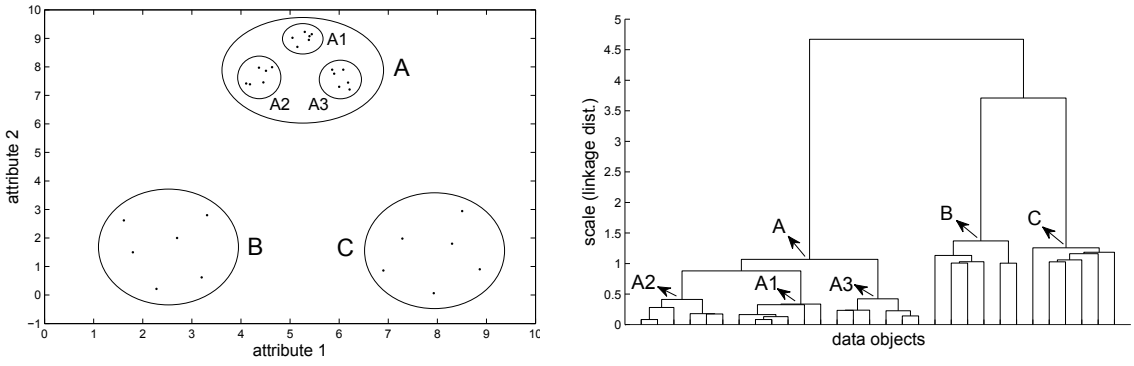
[2]If clusters are permitted to have subclusters, the hierarchical clustering is obtained, which is a set of nested clusters organized as a tree

such as biological taxonomy and document categorization, just to mention a few examples [41]. In conceptual clustering, for example, the hierarchical structure can also provide an attribute-value description that allows interpretation of the resulting clusters [42, 43, 44]. Furthermore, hierarchical models are useful tools for visualization of high-dimensional data, e.g., in the form of a traditional dendrogram[3] [1, 40], a compacted cluster tree [28, 27], a reachability-like plot [30, 45], or a silhouette-like plot [25]. For all these reasons, hierarchical models often represent a natural choice to describe clustering structures.

Hierarchical models are able to provide richer descriptions of clustering structures than those provided by "flat" models, in which a given label (possibly null, representing "noise") is assigned to each object of the data set. In spite of that the hierarchies provide richer descriptions of clustering structures, applications in which the user also (or even only) needs a flat solution are common, either for further manual analysis by a domain expert or in automated KDD processes in which the output of a clustering algorithm is the input of a subsequent data mining procedure — e.g., pattern recognition based on image segmentation. In this context, the extraction of a flat clustering from a hierarchy may be advantageous when compared to the extraction directly from data by a partitioning-like (i.e. non-hierarchical) algorithm. One reason is that hierarchical models describe data from multiple levels of specificity/generality, providing a means for exploration of multiple possible solutions from different perspectives while having a global picture of the cluster structure available. For example, as we will discuss in Chapter 4 and Chapter 5, for a multitude of types of hierarchies we can effectively evaluate the quality of clusters according to their behavior along different hierarchical levels.

The usual approach to extract a flat solution from a hierarchical clustering is by (manually or automatically) choosing one of the levels of the hierarchy. How

[3]A dendrogram is a tree diagram that is used to illustrate the arrangement of the clusters produced by a hierarchical clustering (e.g., Figure1.1(b))



(a) data set (b) dendrogram

Figure 1.1: (a) Illustrative data set and (b) its single-linkage dendrogram (Euclidean distance).

to choose the most appropriate level is the well-known problem of performing a horizontal cut through a dendrogram, which has been widely studied in classic cluster analysis [31, 1]. In spite of its widespread use in practice, this approach has a major limitation, as it cannot provide solutions composed of clusters described at different levels of abstraction or granularity [46, 47]. The toy example in Figure 1.1 illustrates one possible consequence of such a limitation. Specifically, it is clear that there is no horizontal cut through the single-linkage[4] dendrogram in Figure 1.1(b) that would be able to simultaneously provide clusters $A1$, $A2$, $A3$, B, and C as a flat clustering solution for the data set in Figure 1.1(a), even though this is a discernibly valid alternative.

Situations like the one described above and illustrated in Figure 1.1 can also occur when considering hierarchical clustering algorithms other than single-linkage [48]. In particular, in the case of hierarchical algorithms based on density estimates, such as OPTICS [30, 28], AUTO-HDS [49, 25], and gSkeletonClu [29], a horizontal cut corresponds to a clustering solution induced by a single, global density threshold. As has been discussed in the literature, it is

[4]Single-linkage clustering is a hierarchical clustering algorithm in which at the beginning each element is in cluster of its own. Then at each step two clusters separated by the shortest distance are combined into a larger clusters, until all objects end up being in the same cluster.

often not possible to simultaneously detect clusters of varied densities by using such a threshold [30, 50, 17, 20], which is also a major shortcoming of many density-based non-hierarchical algorithms, such as DBSCAN [23] and DENCLUE [51], among others. Devising appropriate means to perform local cuts at different hierarchical levels for different branches (subtrees) of a cluster tree or dendrogram is therefore an important problem.

Performing local cuts through a clustering hierarchy is equivalent to adopting different density thresholds for different subsets of the data and allows one to get flat solutions that cannot be obtained by the traditional horizontal, global cut. Although there is a plethora of different methods for hierarchical clustering, very few are (either directly or indirectly) able to automatically perform some sort of local cut in the resulting hierarchy [28, 50, 52, 27, 25]. For example, the most recently proposed method, "Automated Hierarchical Density Shaving (AUTO-HDS)" [25], which is hierarchical density-based clustering in the context of clustering biological data, automatically selects a flat partition through local cuts from the hierarchy by first selecting the cluster that is present in more hierarchical levels and then looking for the second most prominent cluster that does not have any overlap with the first cluster and so on.

In the following we will introduce semi-supervised clustering and more specifically motivate and discuss semi-supervised extraction of a flat partition from a hierarchy along with semi-supervised model selection.

1.2 Semi-Supervised Clustering

In many different application scenarios, a certain but usually small amount of information, about the data may be available. This information, which can be from various sources such as information provided by the user or topological features extracted from gene interaction networks [53], allows one to perform

clustering in a semi-supervised way, using it to guide the clustering task so the final clustering can be more in accordance with background knowledge. These *external*, *explicit* constraints are provided by the user or analyst or that come from a different source of information in addition to the *internal*, *implicit* constraints that follow from the inductive bias of the adopted clustering model. Here we discuss a common type of external constraints used in literature and known as "Instance-level Constraints."

- **Instance-Level Constraints:** Instance-level constraints provide information about the assignment of the objects to clusters. Two common types of instance-level constraints are should-link and should-not-link constraints.

  - **"Should-link" Constraints:** Should-link constraints encourage pairs of objects to be grouped together in the same cluster. Should-link(x,y) implies that objects x and y should be in the same cluster.
  - **"Should-not-link" Constraints:** Should-not-link constraints encourage pairs of objects to be separated apart into different clusters. Should-not-link(x,y) implies that x and y should not be in the same cluster.

  Should-link and Should-not-link constraints have transitive and closure properties as follows:

  For each three objects x,y,z if:

  - should-link(x,y) and should-link(y,z) then should-link(x,z).
  - should-link(x,y) and should-not-link(y,z) then should-not-link(x,z).

As illustrated in example in Figure 1.2, if we consider each constraint as an edge between two objects, the transitivity of should-link constraints form iso-

lated complete graphs[5] of should-link constraints, e.g., two complete component graphs with green edges shown in Figure 1.2, meaning that if some should-link constraints form a connected component graph[6], this connected component should be a complete graph of should-links. If an object involves a should-not link with one of the objects in the complete should-link graph, it forms a should-not link with all other objects in that graph (see Figure 1.2), forming a bipartite graph of should-not-links[7]. Therefore if the should-link and should-not-link constraint must be satisfied, we need to check only a fraction of these constraints to ensure that they are satisfied, e.g. checking a Minimum Spanning Tree (MST) in a complete graph of should-link constraints[8].

When considering these types of constraints, it is important to note that they can be used as hard (must be satisfied) or soft (can be violated) constraints [11]. Only soft constraints may be violated in a final solution, given that the aim of soft constraints is usually to minimize the number of constraint violations or maximize the number of constraint satisfactions. With hard constraints, by contrast, all constraints must be satisfied in a final solution. Furthermore, there are three possible approaches to utilizing constraints to improve the clustering task. The first approach is using the constraints in the preprocessing step of the data [54]. The second approach is using the constraints in the process of clustering. There are several methods in the literature that use constraints in this way, including those that use constraints to change the similarity matrix in the process of clustering or satisfying the constraint in the process of constructing the cluster tree, e.g., by incorporating propositional logic solvers into

[5]A complete graph is an undirected graph, in which every pair of vertices is connected by an edge.

[6]A connected component of an undirected graph is a subgraph in which every two vertices are connected to each other through at least one path.

[7]A bipartite graph is a graph whose vertices can be divided into two disjoint set of vertices V and U such that every edge connects a vertex in V to a vertex in U. E.g., in Figure 1.2, $V = \{A, B\}$ and $U = \{C, D, E\}$ connected by should-not-link constraints (red edges).

[8]Given a weighted connected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together, and a minimum spanning tree is a spanning tree that has the minimum sum of the weights of edges among all spanning trees.

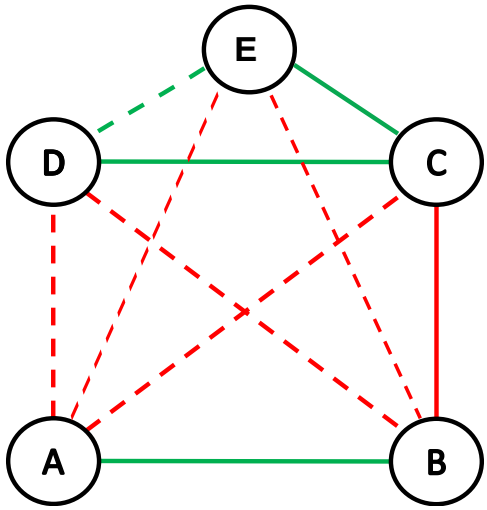


Figure 1.2: Example of transitive closure for some given constraints: with given constraints should-link(A,B), should-link(C,D), should-link(C,E) and should-not-link(B,C), the should-link(D,E) (dotted green link) and should-not-link constraints, (A,C), (A,D), (A,E), (B,D) and (B,E) (dotted red links), can be induced.

the construction loop [55]. Finally, it is possible to use the constraints after the clustering task to choose between different clustering choices [56].

In this manuscript we use instance-level constraints in a soft way to extract a flat partition from a hierarchy and also to select models in a semi-supervised way. We discuss them further in the following, Section 1.2.1.

1.2.1 Semi-Supervised Extraction of a Flat Partition from a Hierarchy and Semi-Supervised Model Selection

As we noted in Section 1.1.2, hierarchical models are able to provide richer descriptions of clustering structures than those provided by flat models, in which a given label (possibly null, representing noise) is assigned to every object of the data set. However, there are also applications in which the user requires a flat solution in addition to hierarchical solution. Along with unsupervised approaches for extracting the flat solution from the hierarchy, there has been a growing interest in semi-supervised hierarchical clustering meth-

ods, i.e., methods that produce cluster trees using partial supervision in the form of labels or constraints that represent previous knowledge about the data [57, 43, 58, 59, 60, 55]. These methods should be distinguished from those that use hierarchical clustering and partially labeled data to categorize unlabeled data into *predefined categories* (semi-supervised categorization; e.g., see [61, 62]).

The area of semi-supervised clustering has earned particular attention in recent years [10], and the branch of hierarchical approaches has followed a similar trend. Formulations of the problem have been discussed from a theoretical perspective [63, 59] and semi-supervised hierarchical clustering algorithms have been developed to deal with constraints in different ways. These algorithms are mostly based on some form of distance learning [64, 57, 65, 66, 58, 60] or on the adaptation of agglomerative [63, 59, 55, 67] and divisive [68, 69, 70] hierarchical methods[9] to enforce constraint satisfaction during the construction of the cluster tree.

In spite of these advances, the focus has only been on constructing hierarchies only, by satisfying constraints completely or as much as possible. To the best of our knowledge, the problem of selecting clusters to compose a flat solution from a cluster tree, when such a solution is needed or desired, has been virtually untouched in the semi-supervised clustering literature. The only option is the traditional horizontal cut approach, which imposes an arbitrary and unnecessary additional constraint on the problem. In fact, by requiring that all extracted clusters must lie on the same level of the hierarchy, one would never be able to fully satisfy user-specified constraints that suggest the existence of $A1$, $A2$, $A3$, B, and C as clusters in the example of Figure 1.1.

As regards semi-supervised model selection, in spite of the advances in semi-supervised clustering that we discussed in this section, the focus has been

[9]Hierarchical clustering is agglomerative (bottom-up) if it starts with single objects and aggregates them into clusters and is divisive (top-down) if it starts with the complete data set and breaks it up into smaller clusters.

only on how to obtain potentially better clustering solutions through semi-supervised guidance. The problem of semi-supervised model selection has, notably, been overlooked.

As we discussed in Section 1.1.1, different clustering algorithms or even the same algorithm with different configurations for its parameters (e.g., the number of clusters k when this quantity is required as an input) may come up with significantly different solutions when applied to the same data. Selecting the best solution is the fundamental problem of *model selection*, i.e., choosing a particular algorithm and/or a particular parameterization of the algorithm amongst a diverse collection of alternatives. As we discussed in Section 1.1.1, one can use relative clustering evaluation criteria as quantitative, commensurable measures of clustering quality [31, 1, 71]. This approach, however, has a few drawbacks [32], such as the well-known fact that the evaluations and performance of different existing criteria are highly data-dependent, in a way that makes it very difficult to choose one specific criterion for a particular data set. To overcome these drawbacks we study semi-supervised model selection using small amount of information in the form of instance-level constraints.

1.3 Contributions

In this thesis, we make several contributions to density-based clustering and semi-supervised clustering.

First, as regards density-based clustering:

1. In the context of density-based clustering validation, we propose Density Based Clustering Validation (DBCV) for density-based, arbitrarily-shaped clusters. DBCV assesses clustering quality based on the relative density connection between pairs of objects. This work has been published in [71]. As regards DBCV, we make five important contributions:

    - We propose a new core distance definition, which evaluates the den-

sity of objects with respect to other objects in the same cluster; these distances are also comparable to distances between objects inside the cluster.

- We propose a novel approach to estimate the density inside and between clusters by constructing the Minimum Spanning Tree using symmetric reachability distances, which, in turn, are computed using our definition of core distance.
- We propose a new relative validity measure based on our concept of core distance, which allows for the validation of arbitrarily-shaped clusters (along with noise, if present).
- We propose a novel approach that makes other relative validity criteria proposed in the literature capable of handling noise.
- We provide theoretical proofs of the properties of the DBCV method and we also present extensive experimental evaluation results of clustering validity criteria using a variety of real-world and synthetic data sets that demonstrate the effectiveness of our method.

2. In the context of hierarchical density-based clustering, we propose several methods that improve upon previous hierarchical clustering approaches. In detail:

   - First, we propose an improvement to the state-of-the-art algorithm AUTO-HDS by removing its parameter r_{shave}, thereby making the cluster extraction stage of AUTO-HDS simpler and more accurate. This is joint work with Campello and Sander and has been published in [72].
   - Second, we review a hierarchical clustering algorithm, called HDBSCAN. HDBSCAN provides a clustering hierarchy whose extracted flat partition from a hierarchy consists of the most significant clusters that can be obtained. The HDBSCAN algorithm was proposed

by Campello, Moulavi and Sander [73] with participation of the author.

- Third, we propose a general density-based clustering approach called GHDBSCAN, which is the generalization of the DBSCAN and HDBSCAN algorithms. We recognize two essential components of density-based clustering, density at each data point and density of a path between two points, and show that it is possible to replace these two components to define new density-based clustering algorithms.

- Fourth, we propose two methods to improve on each of these components that are used in state-of-the-art density-based clustering algorithms. We build two new algorithms based on these two new proposed methods, namely, GHDBSCAN(NMRD) and GHDBSCAN (NMRD+PF). GHDBSCAN(NMRD) improves on HDBSCAN by providing a better estimate of the density along paths between objects. GHDBSCAN(NMRD+PF) incorporates our new parameter-free kernel for estimating the density at objects, which makes it a hierarchical density-based clustering method without parameters.

- Fifth, we propose a new measure of cluster stability, called Bounded Excess of Mass, that can be used in extracting a flat partition of significant clusters from possibly different levels of a hierarchy produced by GHDBSCAN method.

Second, as regards semi-supervised clustering we make the following contributions:

- We propose a framework for semi-supervised model selection that allows the user to select the best model based on few constraints along with the DBCV relative validation index, called GSS-MS. We also provide a framework for semi-supervised model selection based on cross-validation

technique called CVCP (the framework CVCP was proposed by Pourrajabi and Moulavi et al. [54] with participation of the author). We also review a framework for semi-supervised Optimal Extraction of Clusters from hierarchies, called FOSC (the framework FOSC was proposed by Campello, Moulavi, Zimek and Sander [56] with participation of the author);

- We combine FOSC and GHDBSCAN (FOSC-GHDBSCAN) and use it along with our GSS-MS model selection approach, then present extensive experimental evaluation of results for each method and for combined scenarios on a variety of real-world data sets. These experiments show that all of our approaches outperform state-of-the-art algorithms from the literature. We also demonstrate that the combination of our approaches even further improves the quality of clustering results.

Outline:[10]

The remainder of this thesis is organized as follows: In Chapter 2 we review background material and related work, including the problem of density-based clustering validation and density-based hierarchical clustering. In this chapter we also review studies on semi-supervised clustering related to semi-supervised hierarchical clustering and semi-supervised model-selection. In Chapter 3, we propose a density-based clustering validation method to evaluate arbitrarily-shaped density-based clustering methods. We also present experiments on synthetic and real-world data to show the effectiveness of our method for the evaluation and selection of clustering algorithms and their respective parameters.

In Chapter 4, we propose different hierarchical methods in chronological and progressive order. First, in Section 4.2 we propose a new approach to

[10] A substantial amount of the text and content from the works published in [71, 72, 73, 56, 54] has been used in this manuscript.

improve the algorithm AUTO-HDS proposed by [25] by removing the user-defined parameter r_{shave}, making the cluster extraction stage of AUTO-HDS simpler and more accurate. Second, in section 4.3 we propose a method called GHDBSCAN, which is the generalization of density-based clustering algorithms that are based on Hartigan's model of density-based clustering. We recognize two essential components of density-based clustering and then we propose two methods to improve on each of these components in Sections 4.4.1 and 4.4.2. We provide two new algorithms based on these two new proposed methods, namely, GHDBSCAN(NMRD) and GHDBSCAN(NMRD+PF). GHDBSCAN(NMRD) improves over HDBSCAN by providing a better estimate of the density along paths between objects. GHDBSCAN(NMRD+PF) incorporates our new kernel parameter free kernel for estimating the density at objects, which makes GHDBSCAN(NMRD+PF) a parameter-free hierarchical density-based clustering. We also propose a new measure of the stability that can be used to extract the most prominent clusters from a hierarchy. A collection of experiments is presented that involves clustering hierarchies of different natures, a variety of real-world data sets, and comparisons with state-of-the-art methods from the literature. Through these experiments we show that all of our hierarchical density-based clustering algorithms outperform the state-of-the-art algorithms from the literature.

In Chapter 5, we discuss the use of a small amount of knowledge in the form of constraints to guide the clustering task and provide a framework for optimal extraction of flat clusterings from local cuts through cluster hierarchies, then provide two frameworks for semi-supervised model selection. We also present extensive experimental results using FOSC with the GHDBSCAN hierarchy and use our model selection approaches on a variety of real-world data sets to compare our methods with state-of-the-art algorithms. Finally in Chapter 6 we give some final remarks and propose further research directions to extend the works proposed in this thesis.

Chapter 2

Background and Related Work

In this chapter we survey sets of literature that are relevant to this thesis. First, Section 2.1 reviews the related work on density-based cluster analysis; Subsection 2.1.1 reviews the related work on density-based clustering validation, and Subsection 2.1.2 reviews the work related to hierarchical density-based clustering. Second, in Section 2.2, we discuss related work on semi-supervised clustering, focusing specifically on semi-supervised extraction of clusters from a hierarchy and semi-supervised model selection in Subsection 2.2.1.

2.1 Density-Based Cluster Analysis

The notion of density is a major concept in statistics and many data mining tasks. As described in Section 1.1, two approaches called parametric and non-parametric density estimation are utilized in cluster analysis techniques. If the distribution of the underlying unknown p.d.f. in the data set is known *a priori* for some or all of the clusters, the parametric models for data clustering can be utilized. For example, well-known clustering algorithms, such as k-means and EM (Expectation Maximization), correspond to parametric approaches in which an unknown p.d.f. is assumed to be composed of a mixture of k Gaussian distributions. From a procedural view, these clustering algorithms

find a predetermined number of clusters that tend to be of convex shape, so that within-cluster similarities and between cluster dissimilarities are maximized. Notions of within-cluster similarity and between-cluster dissimilarity are defined using the given dissimilarity (distance) function. The tendency to produce convex-shaped clusters is also intrinsic to other traditional clustering algorithms such as average-linkage, complete-linkage and Ward's [1, 74], which do not use the parametric density estimation explicitly.

In contrast to these methods, density-based clustering methods are non-parametric approaches, where the clusters are high density areas separated from other clusters with lower density areas [3]. Density-based clustering algorithms, such as the well-known DBSCAN [23] and DENCLUE [51], do not make assumptions about the underlying density f of data. They also do not require number of clusters as a priori knowledge. Thus density-based clusters do not necessarily have high within-cluster similarity as measured by dissimilarity function, but they can have arbitrary shape in the feature space, and thus these clusters are sometimes referred to as "natural clusters" [3].

Wishart [74] adopted the term "minimum-variance" in the late 1960s to refer to clustering methods that produce convex-shaped clusters, and listed 13 clustering algorithms as just a few representatives of the plethora of methods in this category. In his paper, Wishart [74] raised some objections about these algorithms and described why they may fail when applied to real-world data sets, while also elaborating on the limitations of single-linkage clustering. Wishart [74] then proposed a non-parametric density-based clustering called *One Level Mode Analysis*, followed by its hierarchical version, the *Hierarchical Mode Analysis* (HMA), which can be seen as a first attempt at non-parametric density-based clustering. To estimate the density of the objects, Wishart proposed using a distance threshold r and a frequency threshold k. He noted, however, that these thresholds must be chosen by the user, which is a major limitation and has become a point of criticism of the *One Level Mode Analysis*.

To reduce the problem of selecting distance threshold r and frequency threshold k by the user, he proposed the *Hierarchical Mode Analysis* algorithm, which is based on the order of the distances in which points become dense. Estimating density by using parameters r and k also have other disadvantages, as the density of an object depends only on the distance of the object to a single point in a data set. *Hierarchical Mode Analysis* may also overcome the chaining problem that has affected some clustering methods, such as the well-known clustering algorithm single-linkage[1].

Wishart's method searches for "modes" by using a density threshold $r_{inverse}$ of the underlying multivariate distribution without restricting this distribution to any particular shape. He observed that " ... if probability density function has two or more modes at the level of probability threshold $r_{inverse}$, then the covering will be partitioned into two or more disjoint connected subsets of points". Indeed, the methods proposed by Wishart [74] include a number of ideas that have also been utilized by more recent density-based clustering algorithms.

Hartigan [3] generalized and extended the definition of density-based clustering, describing a density-based cluster as a higher density region separated from other clusters with lower density regions. He specifically defined density-contour clusters and density-contour trees. A density contour cluster $C \in \mathbb{R}^d$ at density level f_1 is a *maximally connected* set of objects x for which $f(x) \geq f_1$, where $f(x)$ is the density at each object. The density-contour tree is the tree of nested clusters that is conceptually conceived by varying the density threshold f_1.

Forty-five years after Wishart's proposal and 40 years after Hartigan's proposal for practical arbitrarily-shaped density-based clustering, challenges still exist in this area. To explore the major challenges: *a)* we study related work

[1]In single-linkage clustering, a cluster grows gradually by adding one element at a time. This may result in heterogeneous straggly clusters that is known as chaining phenomenon.

on evaluation of the arbitrarily-shaped clusterings in Section 2.1.1, and *b)* we study related work on hierarchical density-based clustering and extraction of flat partition from these hierarchies in Section 2.1.2.

2.1.1 Density-Based Clustering Validation

One of the major challenges in clustering is the validation of results, which has been described as one of the most difficult and frustrating steps in cluster analysis [1, 31]. Clustering validation can be divided into three scenarios: external, internal, and relative [1].

External clustering validity approaches such as the Adjusted Rand Index [75] compare clustering results with a pre-existing clustering (or class) structure, i.e., a ground truth solution. Although arguably useful for algorithm comparison and evaluation [76], external measures do not have practical applicability, since, by definition, clustering is an unsupervised task, with no ground truth solution available *a priori*.

In real-world applications, internal and relative validity criteria are preferred, and widely applied. Internal criteria measure the quality of a clustering solution using only the data themselves. Relative criteria are internal criteria that can compare two clustering structures and point out which one is better in relative terms. Although most external criteria can also meet this requirement, the term "relative validity criteria" usually refers to internal criteria that are also relative, and this convention is adopted hereafter. There are many relative clustering validity criteria proposed in the literature [33, 32]. Such measures take as input the results of a clustering algorithm and provide a quantitative evaluation of the results, based on intrinsic information of the data alone. These measures are based on the general idea of computing the ratio of within-cluster scattering to between-cluster separation, with differences arising from different formulations of these two concepts.

Although relative validity measures have been successfully employed in the

evaluation of globular clustering results, they are not suitable for the evaluation of arbitrarily-shaped clusters, as obtained by density-based algorithms. In the density-based clustering paradigm, clusters are defined as dense areas separated by sparse regions [3]. Therefore, clustering results can contain arbitrarily-shaped clusters and noise, which such measures cannot handle properly, given they are originally defined to evaluate the globular clustering. One approach to handling the noise is de-noising the data [77]; however, this method changes the original data set by removing the objects defined as noise and does not address the arbitrarily-shaped clustering. In spite of the extensive literature on relative validation criteria, little attention has been paid to density-based clustering validation; indeed, only a few preliminary approaches are described in the literature. In an effort to capture arbitrary shapes of clusters, some authors have incorporated concepts from graph theory into clustering validation. Pal and Biswas [35] build graphs for each clustering solution, e.g., Minimum Spanning Trees, and use information from their edges to reformulate relative measures such as the Dunn Index [78]. Although the use of a graph can, in principle, capture arbitrary cluster structures, the measures introduced by the authors still compute compactness and separation based on a Euclidean view, favoring globular clusters. Moreover, separation is still based on cluster centroids, which is not appropriate for arbitrarily-shaped clusters. Yang and Lee [79] employ a Proximity Graph to detect cluster borders and develop tests in order to verify whether a clustering is invalid, possibly valid or good. The problem with such an approach is that it does not result in a relative measure. Moreover, the tests employed by the authors require three different parameters from the user. Finally, in both [35] and [79], graphs are obtained directly from distances; therefore, no density concept is employed.

Density concepts have also been taken directly into account during clustering validation. Chou et al. [37] discuss the advantages and disadvantages of Dunn's [78] and Davies-Bouldin's [80] measures, showing that each cap-

tures some aspects of clusters while missing other aspects. The authors [37] then introduce a measure that combines concepts from Dunn [78] and Davies-Bouldin [80] and is aimed at dealing with clusters of different densities. The measure cannot, however, handle arbitrarily-shaped clusters; as shown by the authors, it is only appropriate for spherical-shaped clusters with different densities.

Of the four measures [38, 81, 82, 36] that try to capture the arbitrary shape and density-based properties of the methods simultaneously, $CDbw$ [36] is by far the most employed method and is in fact a enhancement of the SD and S_Dbw measures [82, 81]. The Pauwels and Frederix [38] index was an attempt in the context of image segmentation that did not become popular at all. We discuss them in the following.

Pauwels and Frederix [38] introduced two validity indices to help in selecting the proper clustering in the context of image segmentation. The first index is the "Isolation" measure, which is defined as the average of *k-nearest norm*s of objects inside the cluster, where *k-nearest norm* of object x is defined as a fraction of the k nearest neighbors of x that have the same cluster label as x. As Pauwels and Frederix [38] mentioned in their paper, the major drawback of this measure is that it does not notice when two clusters are merged. To deal with this drawback they propose the "Connectivity" measure, which tries to determine whether a cluster is connected. To quantify the Connectivity, they choose several pairs of random points (e.g., a and b) in the same cluster and connect them using a straight line, then picking the testpoint t halfway along this connecting line in order to seek its local density maximum. Finally, they average these values to achieve the connectivity indicator value. To come up with a single measure that deals with "Isolation" and "Connectivity" simultaneously, they combine these two measures.

Even though the combination approach occasionally solves the problem of its indices, it still suffers from many of the disadvantages of each of its indices

which we will discuss them in the following. First, all three measures (Isolation, Connectivity and combination) have critical parameters that are difficult to set, such as the parameter k in the "Isolation" measure and a number of representative pairs of objects in the "Connectivity" measure and both parameters in the combination approach. Second, even if two clusters are not density-separated (having overlapping objects), the "Isolation" index can be inconclusive because *(a)* most of the objects can still be in non-overlapping areas and *(b)* this index can vary significantly when different values of k are chosen. Third, as the authors noted, it is possible that in the "Connectivity" measure, the testpoints may be attracted to one of the objects in the initial pair. Thus, in the case of two separated clusters, not all of these testpoints get stuck in the void between the high density regions, if they do not, it is not possible to measure the separation properly. Fourth, because each of the two indices depends on parameters and constraints, the value of the indices is non-deterministic, making it hard to compare two clustering solutions properly. Finally, the authors fail to propose an approach to handling noise, which is intrinsic to the definition of density-based clustering.

The SD index [81] is based on two concepts: average scattering within clusters and separation between clusters considering the variance and distance between the centroids of the clusters. The measure is defined as the sum of these two terms. Motivated by taking into account density variations among clusters, the S_Dbw measures [82] have definition similar to the SD index [81], in terms of the concept of scattering within clusters and separation between clusters considering the variance and distance between the centroids of the clusters. Then the measure is defined as the sum of this two terms. Neither measure can, however, deal with arbitrarily-shaped clusters, given that they consider the center of clusters within their definitions, which is not a representative point in density-based arbitrarily-shaped clusters.

The measure called $CDbw$ [36] reflects an attempt to overcome several of the

drawbacks of the SD and S_Dbw measures. $CDbw$ is, as far as we know, the most frequently employed relative measure for density-based validation. Similar to SD and S_Dbw, validity index $CDbw$ also assesses the compactness and separation of clusters defined by a clustering algorithm. However, the approach adopted by $CDbw$ is to consider multiple representative points per cluster, rather than one per cluster, and thereby to capture the arbitrary shape of a cluster based on the spatial distribution of such points. $CDbw$ has, however, several major drawbacks related to the multiple representatives it employs.

The first of these drawbacks is that the measure does not specify how to determine the number of representative points for each cluster. Given that clusters of different sizes, densities and shapes are under evaluation, employing a fixed number of representatives for all clusters does not seem the best approach. Even if a single number of representative points is employed for all clusters (as the authors suggest), this number can still be critical to the performance of the measure and consists of a parameter, which is, at the very least, undesirable. Second, assuming that a reasonable number of representative points can be defined, the representative points themselves have to be determined. Different approaches can be employed in such a task for $CDbw$, as suggested by the authors. The adoption of different approaches in order to find representative points can be seen not only as another parameter, but as a significant source of instability, given that two different sets of representatives containing the *same* number of points, but generated by *different* approaches, which can lead to different evaluations. A latter minor modification of $CDbw$ [83] suffers from the same drawbacks as the original measure [36].

2.1.2 Hierarchical Density-Based Clustering

Apart from methods aimed at finding approximate estimates of level sets and density-contour trees for continuous-valued p.d.f. — e.g., see [27] and references therein — not much attention has been given to hierarchical density-based

clustering in more general data spaces. The works most closely related to ours in Hierarchical Density-Based Clustering are those in [30, 28, 29, 25].

In [30], the authors proposed an algorithm, "Ordering Points To Identify the Clustering Structure" (OPTICS), which provides a graphical and interactive method of creating an augmented ordering of the database. This algorithm represents the density-based clustering structure of the database and its reachability plot, a one-dimensional plot which can display the structure of clusters, with respect to parameters ε and m_{pts} (parameters to find density in DBSCAN algorithm). A post-processing procedure to extract a simplified cluster tree from the reachability plot produced by the algorithm was also proposed. This procedure has not become as popular as OPTICS itself, probably because it is very sensitive to the choice of a critical parameter, ξ, that cannot easily be determined or understood. Moreover, no automatic method to extract a flat clustering solution based on local cuts in the obtained tree has been described. In addition both methods estimate the density based on a distance to a single point (the k_{th} nearest neighbor); this density is not as robust as density estimates that consider more objects from the neighborhood [71].

Sander et al. [28] proposed an improved method of extracting trees of significant clusters from reachability plots which is less sensitive to the user settings than the original method in [30]. In approach [28] there are two different proposals. The first one is an algorithm to transform an OPTICS reachability plot into an equivalent density-based dendrogram. The second one is a method to derive from those plots a compacted tree containing only significant clusters. However, this method is based on heuristics with embedded threshold values that can strongly affect the results. In addition the problem of extracting a flat solution from local cuts in the cluster tree was practically untouched; the only (*ad-hoc*) approach mentioned by the authors was to arbitrarily take all the leaf clusters and discard the others.

In [29], the original findings from [30, 28, 84] were recompiled in the par-

ticular context of community discovery in complex networks. However, no mechanism to extract a simplified cluster tree from the resulting single-linkage-like clustering dendrogram was adopted, and only a method producing a global cut through the dendrogram was described. Parameter setting is still critical in this algorithm, e.g., in small-scale networks with big communities, an example mentioned by the authors [29].

The algorithm AUTO-HDS proposed in [25], like our method, is based on a principle used to simplify clustering hierarchies, which in part refers back to the work of [85], and also to the concept of a "rigid cluster" introduced by Hartigan [3] and the method for providing compact hierarchy suggested in [74]. The clustering hierarchy obtained by AUTO-HDS is typically a subset of the one obtained by Wishart's Hierarchical Mode Analysis (*HMA*) method [74]. Conceptually, clustering stage at AUTO-HDS is equivalent to a sampling of the *HMA* full hierarchical levels, from top to bottom, at a geometric rate controlled by a user-defined parameter, r_{shave}. Such a sampling can lead to an underestimation of the stability of clusters (clusters can appear before and disappear after sampled levels) or even to missed clusters, and these side effects can only be prevented if $r_{shave} \to 0$. In this case, however, the asymptotic running time of AUTO-HDS is $O(n^3)$ [49] (in contrast to $O(n^2 \log n)$ for "sufficiently large" values of r_{shave}), equal to the asymptotic running time of Wishart's *HMA* method.

AUTO-HDS also attempts to perform local cuts through the hierarchy in order to extract a flat clustering solution from density-based hierarchies produced by the HDS algorithm [49], but it uses a greedy heuristic procedure guided by the stabilities of the candidate clusters to select clusters that may give suboptimal results in terms of overall stability. It also suffers from the problem, discussed above, of estimating the density using distance directly. In addition, the stability measure used in AUTO-HDS has a few undesirable properties: first, it depends on the parameter r_{shave}; second, in calculating stability,

only the approximation of levels in which clusters appear in or disappear from the hierarchy is considered; and third, the stability value for a cluster in one branch of the hierarchy can be affected by the density and cardinality of other clusters lying on different branches.

In the following we review the related works in semi-supervised clustering and more specifically the works related to semi-supervised extraction of a flat partition from a hierarchy and semi-supervised model selection.

2.2 Semi-Supervised Clustering

In [86], the authors propose a semi-supervised clustering algorithm that through modification of the clustering objective function satisfies the constraints. In 2001, shortly after the first semi-supervised clustering attempt [86], Wagstaff et al. showed that using instance-level should-link and should-not-link constraints in the clustering task produces more desirable clusters when clustering GPS trace data from automobiles using the k-means algorithm [87]. Since the initial work by Demiritz et al. [86] and Wagstaff et al. [87], many studies have been done in the area of semi-supervised clustering. We study these in Section 2.2.1, with the focus on the works most related to ours: semi-supervised extraction of a flat partition from a hierarchy and semi-supervised model selection.

2.2.1 Semi-Supervised Extraction of a Flat Partition from a Hierarchy and Semi-Supervised Model Selection

Many studies have addressed the topic of partitional semi-supervised clustering; these include the works proposed in [88, 89, 90, 91, 92] just to mention a few. These clustering algorithms have proved to be effective when a small amount of information is available along with a large amount of unlabelled data. Other important studies have addressed the semi-supervised construction of clustering hierarchies [64, 57, 63, 68, 65, 66, 69, 43, 58, 59, 93, 94, 60, 70, 55, 67, 95], but

we are not aware of any approach in the literature for the semi-supervised extraction of clusters based on optimal cuts through a generic cluster tree.

The works of [96, 84, 97, 91] are most closely related to our work in the area of semi-supervised extraction of clusters from a hierarchy. The HISSCLU [96] algorithm is an algorithm based on OPTICS that can be described in two steps. In the first step, HISSCLU starts the OPTICS expansion from all given labeled objects. During the expansion, they use a distance learning method to change the distance between objects. The reachability plots are then concatenated and reordered to produce one single reachability plot. In the second step, horizontal cuts are made to extract the clusters. The density threshold cuts that are done by the user-provided parameter are a disadvantage of this approach. The algorithm SS-DBSCAN [84] uses semi-supervision in the form of partially labeled objects and implicitly provides as a result a collection of clusters that would be equivalent to local cuts through the OPTICS hierarchy [30] performed so as to ensure that clusters are maximal and pure. SS-DBSCAN is an improvement over HISSCLU [84], which, as noted earlier, obtains clusters based on concatenation of horizontal cuts through a related hierarchy and whose cut levels depends on a user-specified parameter [84]. An underlying, restrictive assumption of SS-DBSCAN, however, is that there should be at least one labeled object from each data category in order to be discovered as a cluster; the method cannot discover natural clusters whose objects are not involved in the partial information provided a priori by the user. The C-DBSCAN [91] is a modified version of DBSCAN that can use instance-level constraints. The two main disadvantages of C-DBSCAN are *a)* that it uses a single global density threshold (like DBSCAN) and *b)* that the algorithm deals with instance-level constraints in a hard sense. Therefore, in order to completely satisfy the constraints, the algorithm violates the density-based assumption, i.e. it must merge two clusters if the objects inside these clusters are involved in should-link constraints. In [97] a heuristic algorithm

partially inspired by the OPTIC idea of cluster expansion was proposed as an alternative to incorporating should-link constraints in the process of clustering biological data, but it requires that three user-defined parameters be set in order to handle the should-not-link constraints.

Apart from the methods described in Section 2.1.2, which extract a flat partition from hierarchies in unsupervised domain [30, 28, 29, 25], a few other methods are able to explicitly or implicitly provide local cuts through some kind of clustering hierarchy. The most related of such works to ours are those found in [46, 47, 52, 50, 27].

Boudaillier and Hebrail [46, 47] described an interactive tool for manual local cuts in traditional dendrograms based on exploratory visualization. In regard to automated methods, Ferraretti et al. [52] proposed a greedy heuristic approach to iteratively select clusters to be split, top-down through a traditional dendrogram, attempting to improve the Dunn's validity index as a global measure of cluster quality; the remaining clusters are extracted as the final flat solution. Stuetzle [50] and Stuetzle and Nugent [27] proposed algorithms to detect clusters of spatial point sets as modes of continuous-valued density estimates. These algorithms are equivalent to performing local cuts through a single-linkage-like dendrogram, but in both cases the cuts are based on criteria that are critically dependent upon a user-specified threshold.

As regards semi-supervised model selection, the evaluation of semi-supervised clustering results may involve two different problems, *external evaluation* and *internal, relative evaluation*[2] of results provided by multiple candidate clustering models (algorithms and/or parameters) using only the data and labels or constraints available, particularly to help users select the best solution for their application.

Regarding the external evaluation problem, the main challenge is dealing

[2]Similar problems have been discussed in unsupervised scenario in Sections 1.1.1 and 2.1.1.

with objects involved in the partial information (labels or constraints) used by the semi-supervised algorithm to be assessed. Indeed, without a suitable setup for the evaluation, this process can actually mislead the assessment of the clustering results. The literature contains a variety of approaches for the external evaluation of semi-supervised clustering, which can be divided into four major categories: (i) *use all data*: in this naïve approach, all data objects, including those involved in labels or constraints, are used when computing an external evaluation index between the clustering solution at hand and the ground truth. This approach is not recommended, as it clearly violates the basic principle that a learned model should not be validated using supervised training data. Some authors [91, 98, 60, 99] do not mention the use of any particular approach to address this issue in their external evaluations, which suggests that they might have used all the data both for training and for validation; (ii) *set aside*: in this approach all the objects involved in labels or constraints during the training stage are just ignored when computing an external index [57, 68, 96, 84, 56]. Obviously, this approach does not have the drawback of the first approach; (iii) *holdout*: in this approach, the database is divided into training and test data, then labels or constraints are generated exclusively from the training data (using the ground truth). Clustering takes place with respect to all data objects as usual, but only the test data is used for evaluation [100, 61]. In practice, this is similar to the second (*set aside*) approach described above in that both prevent the drawback of the first approach (*use all data*), but a possible disadvantage of holdout is that objects in the training fold that do not happen to be selected for producing labels or constraints will be neglected during evaluation; (iv) *n-fold cross validation*: in this approach the data set is divided into n (typically 10) folds and labels or constraints are generated from $(n-1)$ training folds combined together. The whole database is then clustered but the external evaluation index is computed using only the test fold that was left out. As usual in classification tasks, this process is repeated

n times using a new fold as test fold each time [101, 87, 102, 103, 90, 104]. Note that this latter procedure alleviates the dependence of the evaluation results on a particular collection of labels or constraints. For the other three approaches, this can be achieved by conducting multiple trials in which labels or constraints are randomly sampled from the ground truth in each trial; then, summary statistics such as mean can be computed, as it has been done in most of the references cited above.

Apart from the aforementioned external evaluation scenario, a more practical problem is how to evaluate the results provided by semi-supervised clustering algorithms in real applications where ground truth is unavailable, i.e., when all we have is the data themselves and a subset of labeled objects or a collection of clustering constraints. In particular, given that different parameterizations of a certain algorithm or even different algorithms can produce quite diverse clustering solutions, a critical practical issue is how to select a particular candidate amongst a variety of alternatives. This is the classic problem of *model selection*, which aims at discriminating between good and not-as-good clustering models by some sort of data-driven guidance. Notably, to the best of our knowledge, this problem has not been discussed in the literature on semi-supervised clustering.

Chapter 3

Density-Based Clustering Validation

3.1 Introduction

Hartigan's model of Density Contour Trees [3] defines density-based clusters as regions of high density separated from other such regions by regions of low density. Considering such a model, we can expect a good density-based clustering solution to have clusters in which the lowest density area inside each cluster is still denser than the highest density area surrounding clusters.

Relative validity measures, deemed as "traditional," take into account distances to quantify cluster variance that, combined with their separation, then amounts for clustering quality. Minimizing cluster variance and separation, however, is not the objective in density-based clustering. Therefore, a relative measure for evaluation of density-based clustering should be defined by means of densities rather than by distances. Examples of using density-based validation index include recognizing the proper arbitrarily-shaped clustering solution in geographical or image domains.

Below we introduce the Density Based Clustering Validation (DBCV) which considers both density and shape properties of clusters. To formulate DBCV,

we define the notion of all-points-core-distance ($a_{pts}coredist$) which is the inverse of the density of each object with respect to all other objects inside its cluster. Using $a_{pts}coredist$, we define a symmetric reachability distance (similar to the definition by Lelis and Sander [84]) which is then employed to build a Minimum Spanning Tree (MST) inside each cluster. The MST captures both the shape and density of a cluster, since it is built on the transformed space of symmetric reachability distances. Using such MSTs (one for each cluster), DBCV finds the lowest density region in each cluster and the highest density region between pairs of clusters. In the following Section, we discuss our proposed DBCV validation measure.

3.2 Density-Based Clustering Validation

In the definitions of our concepts we use the following notations. Let $\mathbf{O} = \{\mathbf{o}_1, \cdots, \mathbf{o}_n\}$ be a data set containing n objects in the $\mathbb{R}^d$ feature space where d is the dimensionality of the feature space. Let **Dist** be an $n \times n$ matrix of pairwise distances $d(\mathbf{o}_p, \mathbf{o}_q)$, where $\mathbf{o}_p, \mathbf{o}_q \in \mathbf{O}$, for a given metric distance $d(\cdot, \cdot)$. Let $KNN(\mathbf{o}, i)$ be the distance between object $\mathbf{o}$ and its i^{th} nearest neighbor. Let $C = (\{C_i\}, N)$ $1 \leq i \leq l$ be a clustering solution containing l clusters and (a possibly empty) set of noise objects N, for which n_i is the size of the i^{th} cluster and n_N is the cardinality of noise.

To estimate the density of an object within its cluster, a traditional approach is to take the inverse of the threshold distance necessary to find K objects within this threshold [3, 73]. This way, however, the density of an object is based on the distance to a single point (the k^{th} nearest neighbor). As such, this density is not as robust as density estimates that consider more objects from the neighborhood, such as Gaussian kernel density estimate. Moreover, this definition introduces a parameter (K), which is not desirable in validation.

In the following we aim to propose a new, more robust, and parameterless

definition of a core distance that can be interpreted as the inverse of a density estimate and be used in the definition of a mutual reachability distance. To achieve this goal such a core distance should have the following properties: first to act as a more robust density estimate it should not depend on a single point, but rather consider all the points in a cluster in a way that closer objects have a greater contribution to the density than farther objects. This is a common property in density estimate methods such as Gaussian kernel density estimation. Second, since in the definition of a mutual reachability distance [84], the core distance of an object is compared to the distances of the object to other objects in the cluster, the core distance should be comparable to these distances. Third the core distance of an object should be approximately to the distance of a K^{th} nearest neighbor where K is adjusted automatically based on the size of the data set, representing a neighborhood of the object.

We define the core distance of an object $\mathbf{o}$ with respect to all other objects inside its cluster ($a_{pts}\,coredist$) as follows.

Definition 3.2.1 (Core Distance of an Object)**.** The *all-points-core-distance* (inverse of the density) of an object $\mathbf{o}$, belonging to cluster C_j with respect to all other $n_j - 1$ objects in C_j is defined as:

$$a_{pts}\,coredist(\mathbf{o}) = \left(\frac{\sum_{\substack{o_i \in C_j \\ o_i \neq o}} \left(\frac{1}{d(\mathbf{o}, o_i)} \right)^d}{n_j - 1} \right)^{-\frac{1}{d}} \tag{3.1}$$

In the following we show that our definition of $a_{pts}\,coredist$ has the three aforementioned properties.

The first property holds because we calculate the inverse of the distances to the power of dimensionality in $a_{pts}\,coredist$, resulting in higher weight of the contribution of closer objects. Note that this effect could be made stronger by using the squared Euclidean distance instead of Euclidean distance as dissimilarity.

In Proposition 3.2.1 we show that the second property holds for $a_{pts}coredist$, i.e., we prove that the $a_{pts}coredist$ has values between the second and the last (n^{th}) *KNN* distances of the objects.

Proposition 3.2.1. *The all-points-core-distance of each object* $\mathbf{o}$*,* $a_{pts}coredist(\mathbf{o})$*, with respect to all other* $n-1$ *objects in a d-dimensional data set* X *is between the second and last nearest neighbor distance of that object, i.e.,*

$$KNN(\mathbf{o}, 2) \leq a_{pts}coredist(\mathbf{o}) \leq KNN(\mathbf{o}, n)$$

Proof. Proof is provided in Appendix A on page 144. □

Finally, in Propositions 3.2.2 and 3.2.3 we show that the third property holds for our definition of $a_{pts}coredist$ in uniform distribution using Euclidean distance and similarly we prove for Squared Euclidean distance in Propositions 3.2.4 and 3.2.5.

Proposition 3.2.2. *Let* n *objects be uniformly distributed random variables in a d-dimensional unit hypersphere and* $\boldsymbol{o}$ *be an object in the center of this hypersphere. For the all points core distance of* $\boldsymbol{o}$ *we have:*

$$a_{pts}coredist(\mathbf{o}) = ((ln(n-1) + \gamma + \varepsilon)^{-\frac{1}{d}} \approx \ln(n)^{-\frac{1}{d}} \tag{3.2}$$

where $\gamma \approx 0.5772$ *and* $\varepsilon \approx \frac{1}{2n}$ *which approaches to zero as* n *goes to infinity.*

Proof. Proof is provided in Appendix A on page 145. □

Proposition 3.2.3. *For calculated* $a_{pts}coredist(\mathbf{o})$ *in Proposition 3.2.2, we have:*

$$a_{pts}coredist(\mathbf{o}) \approx \ln(n)^{-\frac{1}{d}} \approx KNN(\mathbf{o}, j), \tag{3.3}$$

with j *being the closest natural number to* $\frac{n}{\ln(n)}$ *and* $KNN(\mathbf{o}, j)$ *being the expected value of* j^{th} *nearest neighbor distance to object* o*.*

Proof. Proof is provided in Appendix A on page 147. □

This proposition shows that the core distance of the object **o** approximating distance to the same K^{th} nearest neighbor independent of the dimensionality of the data space.

Although the core distance of object **o**, $a_{pts}coredist(\mathbf{o})$, is approximately equal to $KNN(\mathbf{o}, j)$ for an uniform distribution for some $j \approx \frac{n}{\ln(n)}$ (Propositions 3.2.2 and 3.2.3), note that, when we have a distribution other than the uniform distribution, its behavior follows our first desired property. If most of the objects are close to **o**, $a_{pts}coredist$ tends to be a smaller value. Contrarily if most of the objects are distributed far away from **o**, $a_{pts}coredist$ tends to be a greater value.

In Propositions 3.2.2 and 3.2.3, Euclidean distance is assumed as dissimilarity, however, the conclusions are similar for Squared Euclidean distance and are shown in Propositions 3.2.4 and 3.2.5.

Proposition 3.2.4. *If the dissimilarity measure in Proposition 3.2.2 is Squared Euclidean distance the all points core distance of* **o** *is:*

$$a_{pts}coredist(\mathbf{o}) \approx (1.645 * n)^{-\frac{1}{d}} \tag{3.4}$$

Proof. Proof is provided in Appendix A on page 147. □

Proposition 3.2.5. *For* $a_{pts}coredist(\mathbf{o})$ *(Proposition 3.2.4), we have:*

$$a_{pts}coredist(\mathbf{o}) \approx (1.645 * n)^{-\frac{1}{d}} \approx KNN(\mathbf{o}, j), \tag{3.5}$$

with j *being the closest natural number to* $\sqrt{(}n/1.645)$ *and* $KNN(\mathbf{o}, j)$ *being the expected value of* j^{th} *nearest neighbor distance to object* o.

Proof. Proof is provided in Appendix A on page 148. □

Similar to Proposition 3.2.3 this proposition also shows that the core distance of the object **o** is approximately equal to the same nearest neighbor

distance of **o** independent of the dimensionality of the data space. Comparing propositions 3.2.3 and 3.2.5 confirms that by applying Squared Euclidean distance the effect of the first property becomes stronger and the core distance represents smaller neighborhood of the objects. We prove this property in Proposition 3.2.6.

Proposition 3.2.6. *For the core distances of object o calculated in Propositions 3.2.2 and 3.2.4, we have:* $a_{pts}coredist(\mathbf{o})_{Sq-Euclid.} \leq a_{pts}coredist(\mathbf{o})_{Euclid.}$

Proof. Proof is provided in Appendix A on page 149. □

$a_{pts}coredist$ is used to calculate the symmetric mutual reachability distances in Definition 3.2.2, which can be seen as the distance between objects considering their density properties. In Definition 3.2.4 we define the minimum spanning tree using mutual reachability distances to capture the shape of the clusters together with density properties. These definitions are then used to find the lowest density area (density sparseness) within—and highest density area (density separation) between—clusters in Definitions 3.2.5 and 3.2.6, which are then used to define the relative validity index DBCV in Definitions 3.2.7 and 3.2.8.

The following proposition shows that to calculate the $a_{pts}coredist$, we do not need to raise values to power of dimensionality, and instead we can use the *log* properties to calculate $a_{pts}coredist$.

Proposition 3.2.7. *For the core distance of an object* **o** *calculated with respect to all other* $n-1$ *objects in data set* X*, we have:*

$$a_{pts}coredist(o) = \left(\frac{\sum_{\substack{o_i \in X \\ o_i \neq o}} \left(\frac{1}{d(\mathbf{o},o_i)} \right)^d}{n-1} \right)^{-\frac{1}{d}} = b^P$$

where $P = -\frac{1}{d} \times (log_b{}^{1+\sum_{o_i \in X_1} b^{d \times (log_b{}^{d(\mathbf{o},o_c)} - log_b{}^{d(\mathbf{o},o_i)})}} - log_b{}^{n-1}) + log_b{}^{d(\mathbf{o},o_c)}$,

o_c can be any arbitrary object in X different than o, $X_1 = X - \{o, o_c\}$ and b can be any Real number greater than 1.

Proof. Proof is provided in Appendix A on page 150. □

Definition 3.2.2 (Mutual Reachability Distance)**.** The *mutual reachability distance* between two objects $\mathbf{o}_i$ and $\mathbf{o}_j$ in $\mathbf{O}$ is defined as $d_{mreach}(\mathbf{o}_i, \mathbf{o}_j) = \max\{a_{pts}\,coredist(\mathbf{o}_i), a_{pts}\,coredist(\mathbf{o}_j), d(\mathbf{o}_i, \mathbf{o}_j)\}$.

Note that the comparison of $a_{pts}\,coredist$ and $d(\mathbf{o}_i, \mathbf{o}_j)$ in Definition 3.2.2 is meaningful because of the properties of $a_{pts}\,coredist$ shown in Propositions 3.2.1, 3.2.3 and 3.2.5.

Definition 3.2.3 (Mutual Reach. Dist. Graph)**.** The *Mutual Reachability Distance Graph* is a complete graph with objects in $\mathbf{O}$ as vertices and the mutual reachability distance between the respective pair of objects as the weight of each edge.

Definition 3.2.4 (Mutual Reach. Dist. MST)**.**
Let O be a set of objects and G be a mutual reachability distance graph. The *minimum spanning tree* (MST) of G is called MST_{MRD}.

We present, in brief, the overall idea behind DBCV. Considering a single cluster C_i and its objects, we start by computing the $a_{pts}\,coredist$ of the objects within C_i, from which the Mutual Reachability Distances (MRDs) for all pairs of objects in C_i are then obtained. Based on the MRDs, a Minimum Spanning Tree (MST_{MRD}) is then built. This process is repeated for all the clusters in the partition, resulting in l minimum spanning trees, one for each cluster.

Based on the MSTs obtained in the previous steps, we define a density-based clustering validation index based on the following notions of density sparseness and density separation. The density sparseness of a single cluster is defined as the maximum edge of its corresponding MST_{MRD}, which can be interpreted as the area with the lowest density inside the cluster. We define the density

separation of a cluster with respect to another cluster as the minimum MRD between its objects and the objects from the other cluster, which can be seen as the maximum density area between the cluster and the other cluster. These two definitions are then finally combined into our validity index DBCV.

Let the set of internal edges in the MST be all edges except those with one ending vertex of degree one. Let the set of internal objects (vertices) be all objects except those with degree one. The density sparseness and separation of clusters are given by Definitions 3.2.5 and 3.2.6.

Definition 3.2.5 (Density Sparseness of a Cluster)**.** The *Density Sparseness of a Cluster* (DSC) C_i is defined as the maximum edge weight of the internal edges in MST_{MRD} of the cluster C_i, where MST_{MRD} is the minimum spanning tree constructed using a_{pts}*coredist* considering the objects in C_i.

Definition 3.2.6 (Density Separation)**.**
The *Density Separation of a Pair of Clusters* (DSPC) C_i and C_j, $1 \leq i, j \leq l, i \neq j$, is defined as the minimum reachability distance between the internal nodes of the MST_{MRD}s of clusters C_i and C_j.

Now we can compute the density-based quality of a cluster as given by Definition 3.2.7. Note that, if a cluster has better density sparseness than the density separation, then we obtain positive values of the validity index. If the density inside a cluster is lower than the density that separates it from other clusters, then the index is negative.

Definition 3.2.7 (Validity Index of a Cluster)**.** *We define the validity of a cluster* $C_i, 1 \leq i \leq l$, *as:*

$$V_C(C_i) = \frac{\min\limits_{1 \leq j \leq l, j \neq i} \Big(DSPC(C_i, C_j) \Big) - DSC(C_i)}{\max \Big(\min\limits_{1 \leq j \leq l, j \neq i} \Big(DSPC(C_i, C_j) \Big), DSC(C_i) \Big)} \tag{3.6}$$

The density-based clustering validity, DBCV, is given by Definition 3.2.8. Note that, although noise is not explicitly present in below formulation, it is

implicitly considered by the weighted average that takes into account the size of the cluster ($|C_i|$) and the total number of objects under evaluation, including noise, given by $|O|$ in Eq. (3.7).

Definition 3.2.8 (Validity Index of a Clustering)**.** The *Validity Index of the Clustering Solution* $C = \{C_i\}, 1 \leq i \leq l$ is defined as the weighted average of the Validity Index of all clusters in C.

$$DBCV(C) = \sum_{i=1}^{i=l} \frac{|C_i|}{|O|} V_C(C_i) \tag{3.7}$$

It is easy to verify that our index produces values between -1 and $+1$, with greater values of the measure indicating better density-based clustering solutions.

3.3 Experimental Setup

The evaluation of a relative validity index is usually performed as follows [32, 105]: (i) several partitions are generated with different clustering algorithms; (ii) for each clustering algorithm the ability of the new measure to identify the *correct* number of clusters, as defined by the ground truth partition of each data set, is verified. Although commonly employed, this evaluation procedure has drawbacks [32]. In brief, it quantifies the accuracy of a given relative validity criterion according to whether or not it identifies the correct *number* of clusters for a data set, ignoring completely relative qualities of the partitions under evaluation. Although a partition may have the correct *number* of clusters, it can present an unnatural clustering, misleading the evaluation.

Since we use data sets with a known ground truth, we choose to employ a methodology that takes full advantage of external information. This methodology was introduced by Vendramin et al. [32] and has been subsequently employed successfully [34]. It assesses the accuracy of relative criteria by com-

paring their scores against those provided by an external criterion, such as the Adjusted Rand Index (ARI) [1]. A relative criterion is considered to be better the more similar its scores are to those provided by an external criterion. Similarity, in this case, is measured by the Pearson correlation. Although this procedure is far from perfect [76], it probably is the best procedure available. The methodology is summarized as follows:

1. Given a data set with known ground truth, generate n_π partitions with different properties by varying the parameters of one or more clustering methods.
2. Compute the values of the relative and external validity criteria for each one of the n_π partitions.
3. Compute the correlation between the vectors with the n_π relative validity measure values and the n_π external validity measure values. This correlation quantifies the accuracy of the relative validity criterion with respect to the external validity measure (ARI).

An important aspect in the evaluation of the relative measures for density-based clustering is how to deal with noise objects, given that partitions generated with density-based clustering algorithms may contain noise. As far as we know, DBCV is the first relative validity measure capable of handling noise. Since other relative indices do not have this capability, noise has to be handled *prior* to their application for a fair comparison. To the best of our knowledge, there is no established procedure in the literature defining how to deal with noise objects in a partition when applying a relative validity index. We see at least five possible alternatives: (i) assign all noise to a single cluster, (ii) assign each noise point to its closest cluster, (iii) assign each noise point to a singleton cluster, (iv) remove all noise points, and (v) remove all noise points with a proportional penalty.

Following approach (i), *real clusters* end up embedded in an unique cluster of noise. Approach (ii) modifies the solutions under evaluation, causing other

relative indices to evaluate clustering solutions different than the ones evaluated by our measure. In approach (iii), singleton clusters become close to most of the *real clusters*, resulting in a poor overall separation, which degrades the results of all measures. Just removing the noise without any penalty in approach (iv) is not a good strategy because the coverage is not considered. For instance, a solution which has one object from each cluster and all other objects as noise results in a perfect score. However penalizing lack of coverage as in approach (v) allows the measures to deal with noise in an well behaved way. Therefore we adopt this approach in our evaluation, i.e., we evaluate measures only on points in clusters and multiply the resulting score with $(|O| - |N|)/|O|$, where $|N|$ being the cardinality of noise objects and $|O|$ being the cardinality of all objects in the data set.

Note that this is the same approach adopted for DBCV (Equation 3.7) and we prove this in the following Proposition 3.3.1

Proposition 3.3.1. *Here we show that the weighted averaging approach from DBCV, as shown in Equation 3.7, is exactly the same as penalizing the other relative validity measures based on the proportion of the noise objects in the data set.*

Proof. Let $|N|$ be the cardinality of noise objects and $|O|$ be the cardinality of all objects in the data set.

$$\begin{aligned} DBCV(C) &= \sum_{i=1}^{i=l} \frac{|C_i|}{|O|} V_C(C_i) \\ &= \frac{|O|-|N|}{|O|} \sum_{i=1}^{i=l} \frac{|C_i|}{|O|-|N|} V_C(C_i) \end{aligned} \tag{3.8}$$

Note that $\sum_{i=1}^{i=l} \frac{|C_i|}{|O|-|N|} V_C(C_i)$ is equal to removing the noise objects and calculating DBCV, whereas $\frac{|O|-|N|}{|O|}$ is penalizing the resulting value proportional to the number of noise objects that are left out from the partition. □

In our implementation we use the Squared Euclidean distance since it ampli-

fies the effect of Property 1, which helps to better score solutions with clusters at largely different scales of separation.

3.3.1 Relative Measures

We compare our measure against five well-known relative measures from the literature, namely, Silhouette Width Criterion (SWC) [106], Variance Ratio Criterion (VRC) [107], Dunn [78], and Maulik-Bandyopadhyay (MB) [108]. We also evaluate *CDbw* [36], which is, as far as we know, the most employed measure for density-based validation. All measures are available in the Cluster Validity Library [109].

Note that different studies use different methodologies and set of data sets to compare validation measures (e.g., see [31, 32]). It is noticed that conclusions and results of different studies hold for a particular collection of data set that were used which are mostly for volumetric data sets [33, 32]. However, here we focus on evaluating the arbitrarily-shaped clustering, thus we use well-known and representative validation measures that are most employed in the literature.

3.3.2 Clustering Algorithms

During the evaluation of our measure we consider three different density-based clustering algorithms for generating partitions: (i) the well-known DBSCAN algorithm [23] (ii) the heuristic method by Sander et al. [28], referred to here as *OPTICS-AutoCluster*, which consists of the extraction of the leaf nodes of a density-based cluster tree constructed from an OPTICS reachability plot [30] also used in [56] and, (iii) HDBSCAN [73], which produces a hierarchy of all possible DBSCAN* partitions, each of which is evaluated by the aforementioned relative validity measures.

Considering parameters for such algorithms, we simulate a scenario in which the user has no idea about which values to choose, i.e., a scenario in which a

relative density-based validation index is useful.

Two different parameters are needed as input for DBSCAN, *MinPts* and ϵ. In the case of *MinPts* we choose $MinPts \in \{4, 6, \ldots, 18, 20\}$. For ϵ, we obtain the minimum and maximum values of pairwise distances for each data set and employ 1000 different values of ϵ equally distributed within this range. *OPTICS-AutoCluster* also demands *MinPts*, which was set equally to *MinPts* of DBSCAN. The speed-up control value ϵ in OPTICS was not used (ϵ = Infinity). For minimum cluster ratio we use 250 different values from 0.001 to 0.5 with steps of 0.002. Finally, for HDBSCAN we set m_{pts} equally to *MinPts* of DBSCAN, and use, $MinClSize = m_{pts}$ as employed by its authors [73].

3.3.3 Data Sets

We employ real and synthetic data sets during our evaluation. We use real data from gene expression data sets and the well-known UCI Repository [110]. We use three gene expression data sets: (i) Cell Cycle 237 (Cell237), with 237 objects, 17 features and 4 clusters; (ii) Cell Cycle 384 (Cell384), with 384 objects, 17 features and 5 clusters both Cell237 and Cell384 were made public by [111]; and (iii) Yeast Galactose (Yeast), with 205 objects, 20 features and 4 clusters used in [112]. From UCI Repository [110], we use four data sets: (i) Iris, with 150 objects, 4 features and 3 clusters; (ii) Wine, with 178 objects, 13 features and 3 clusters; (iii) Glass, with 214 objects, 9 features and 7 clusters; and (iv) Control Chart (KDD), with 600 objects, 60 features and 6 clusters.

Besides the real data sets, which are multidimensional, we also employ four 2D synthetic data sets, with different numbers of objects, clusters and noise, as depicted in Figure 3.1. We generated these synthetic arbitrarily-shaped data sets and added noise objects because such data sets are useful to illustrate the behavior of our measure for arbitrarily-shaped clusters.

Index	Data set						
	Cell237	Cell384	Yeast	Iris	Wine	Glass	KDD
DBCV	**0.62**	**0.39**	**0.96**	**0.60**	0.24	**0.29**	**0.56**
SWC	0.52	0.33	0.90	0.57	**0.29**	0.28	0.37
VRC	0.40	0.33	0.73	0.21	0.01	0.28	0.37
Dunn	0.35	0.16	0.38	0.13	0.01	0.28	**0.56**
CDbw	0.55	0.30	0.75	0.55	0.23	0.28	0.54
MB	0.43	0.15	0.73	0.23	0.01	0.28	**0.56**

Table 3.1: Best ARI found by each relative measure.

Index	Data set						
	Cell237	Cell384	Yeast	Iris	Wine	Glass	KDD
DBCV	**0.76**	**0.79**	**0.87**	**0.97**	0.65	**0.81**	**0.84**
SWC	0.72	0.75	0.81	0.93	**0.67**	0.78	0.57
VRC	0.25	0.17	0.34	0.11	0.00	0.19	0.66
Dunn	0.64	0.29	0.65	0.25	0.10	0.62	0.51
CDbw	-0.37	-0.39	-0.06	0.83	0.59	0.09	0.01
MB	0.40	0.14	0.41	0.15	0.06	0.35	0.52

Table 3.2: Correlation between relative indices and ARI.

3.4 Results and Discussion

3.4.1 Real Data Sets

Results for real data sets are shown in Tables 3.1 and 3.2, for which the best values for each data set are highlighted. Table 3.1 shows the Adjusted Rand Index (ARI) of the *best* partition selected by each relative validity criterion. Note that DBCV outperforms its five competitors in most of the data sets. Considering the results for the Wine data set, in which SWC provides the best result, DBCV is a close second. For the Glass data set, DBCV provides the best ARI value, which is the maximum obtained by all three clustering algorithms employed in the evaluation. Therefore, DBCV recognizes the best solution that is available to it. This also holds for other data sets, given that the relative measures can only find partitions as good as the ones generated by the clustering algorithms, which explains the low ARI in some cases. Table 3.2 shows the correlation between each relative measure and ARI. In all but one case, DBCV outperforms its competitors. Again, for Wine, in which the best correlation is obtained by SWC, DBCV provides close results to SWC.

One interesting aspect that can be observed in this evaluation is that some

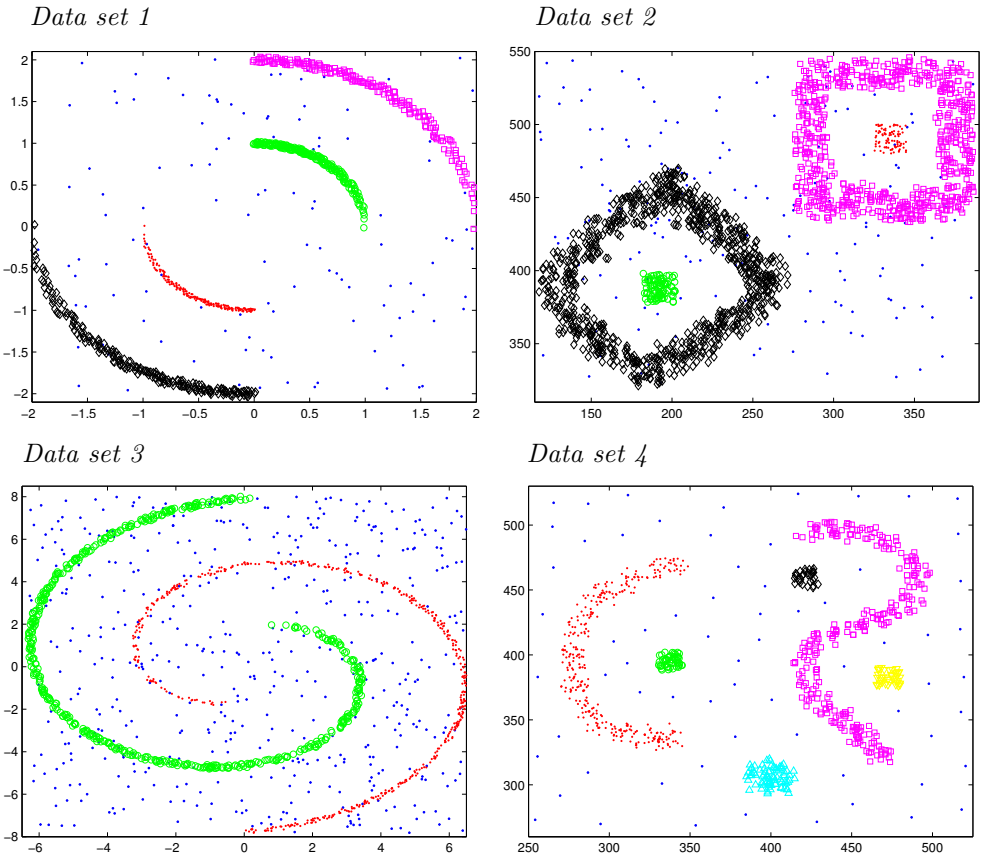


Figure 3.1: Synthetic 2D Data Sets.

relative measures developed for globular clusters perform relatively well. In fact, Silhouette even provides the best results for one data set. This is easily explained by three facts. The first one is the adaptation we introduced to deal with noise for such measures, making them capable of handling partitions with noise, which can be considered an additional contribution of this dissertation. The second is the fact that some of the data sets employed have *globular* clusters. The third is that in some of the data sets ground truth labeling does not follow the density-based structure of the data, e.g., although ground truth consist of three globular clusters in the Iris data set, two of these clusters are overlapping and therefore form a single cluster from a density-based perspective. In such cases, DBCV prefers 2 clusters whereas traditional measures prefer 3. The combination of these three factors makes such measures capable

Index	Data set			
	Data set 1	Data set 2	Data set 3	Data set 4
DBCV	**0.91**	**0.90**	**0.74**	**0.99**
SWC	0.72	0.21	0.19	0.31
VRC	0.51	0.01	0.02	0.01
Dunn	0.20	0.01	0.01	0.01
CDbw	0.84	0.71	0.04	0.92
MB	0.51	0.01	0.01	0.01

Table 3.3: Best ARI found by each relative measure.

of recognizing good *globular* partitions in the presence of noise, as generated by density-based clustering algorithms. Note that we emphasize *globular*, since our adaptation of such measures is useful only for such data sets. In case of *arbitrarily-shaped* data sets, such measures are still not appropriate, as we illustrate in the following with synthetic 2D data.

3.4.2 Synthetic 2D Data Sets

To show how the relative measures perform in the presence of noise in *arbitrarily-shaped* clusters, we consider the four synthetic data sets shown in Figure 3.1. Results for this data sets are shown in Figures 3.2 and 3.3. We show plots of the best partitions selected by four relative measures, i.e., DBCV, SWC, *CDbw*, and VRC. Results for other relative measures designed for the evaluation of globular clustering solutions follow the same trend as the ones shown here. In all figures, noise points are denoted by blue dots. For all data sets, DBCV is the only measure capable of recognizing the true structure present in the data. Other relative measures, like SWC and VRC, often find a large number of clusters, *breaking* the true clusters into multiple small sub-clusters of small size.

As shown in Figures 3.2 and 3.3, *CDbw* is the only competitor measure that finds some arbitrarily-shaped structure in the data sets, although it has some flaws. Considering data set 1, for instance, the best partition it finds has a large portion of the clusters assigned to noise (blue dots). In data set 2, it

Index	Data set			
	Data set 1	Data set 2	Data set 3	Data set 4
DBCV	**0.66**	**0.76**	**0.37**	**0.86**
SWC	0.39	-0.25	-0.31	-0.35
VRC	-0.15	-0.05	-0.14	-0.43
Dunn	-0.21	-0.05	-0.31	-0.32
CDbw	0.49	0.71	0.15	**0.86**
MB	-0.14	-0.16	-0.12	-0.21

Table 3.4: Correlation between relative indices and ARI.

recognizes a clustering solution with merged clusters which is clearly not the best solution. This is also the case in data set 4. In data set 3 it is simply not capable of recognizing the best partition, which is composed of two spirals and random noise.

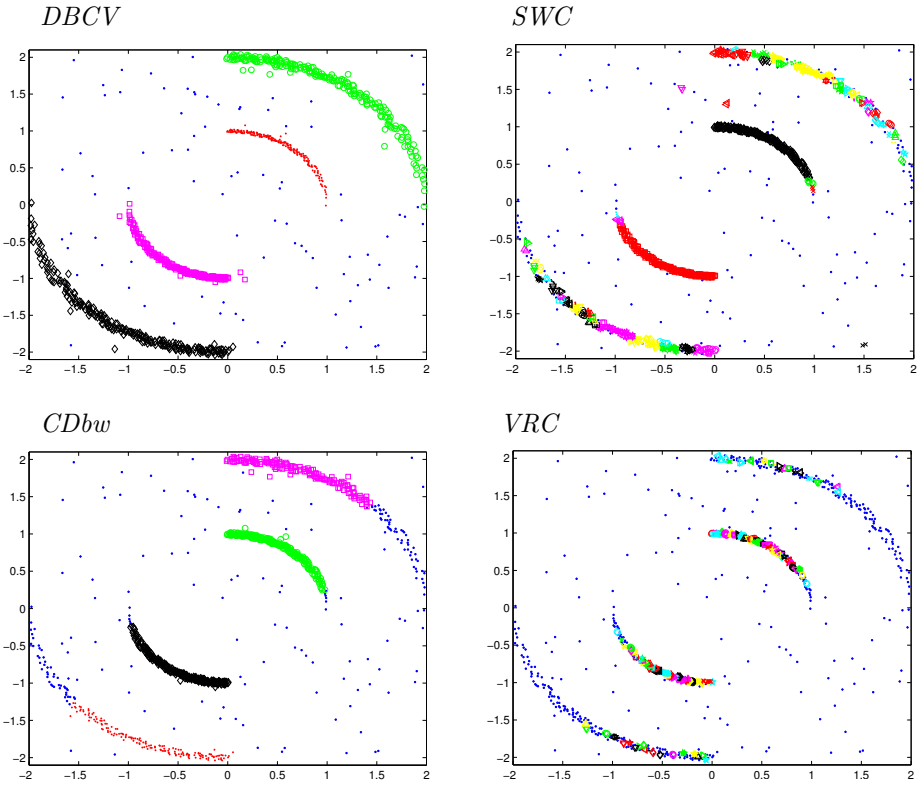
Finally, we show in Tables 3.3 and 3.4 the best ARI values found with each measure and their respective correlation with ARI, for each data set. For all the 2D data sets, DBCV finds the best solution. It also displays the best correlation with ARI for all data sets (along with *CDbw* in data set 4). In brief, this shows that only DBCV can properly find arbitrarily-shaped clusters.

3.5 Summary

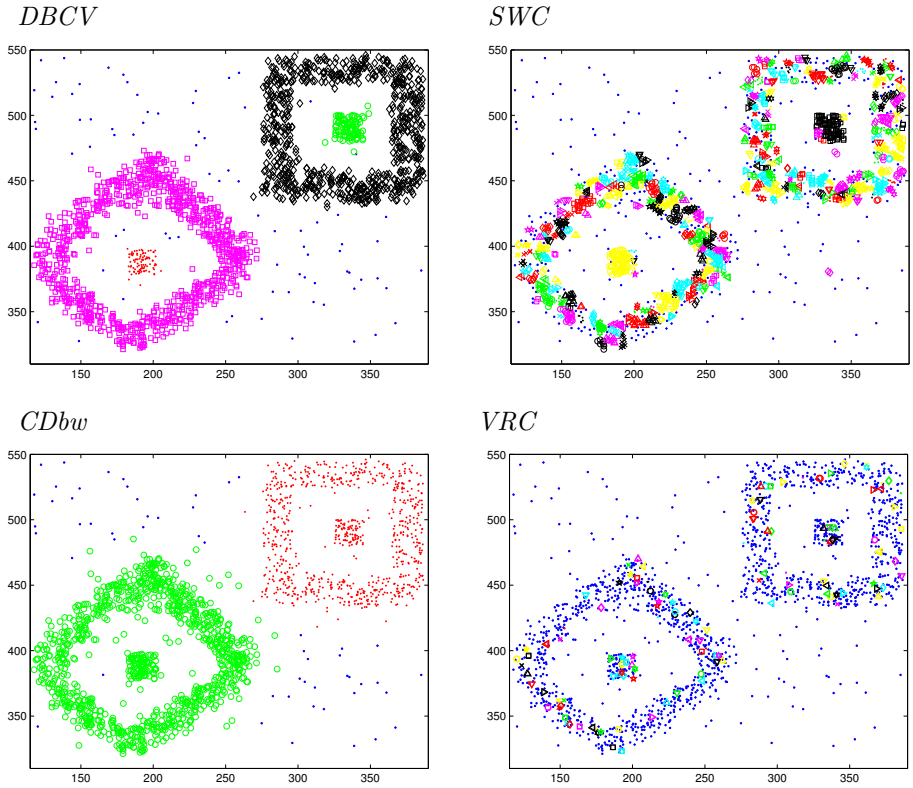
We have introduced a novel relative Density-Based Clustering Validation index, DBCV. Our new index is formulated on the basis of a new kernel density function, which is used to estimate the density inside and between clusters, which in turn used to compute within-cluster density connectedness and between cluster density separation of clustering results. Unlike other relative validity indices, our method not only directly takes into account density and shape properties of clusters but also properly deals with noise objects, which are intrinsic to the definition of the density-based clustering.

We also propose an adaptation to make other relative measures capable of handling noise. We conducted extensive experiments to evaluate the performance of DBCV and five other state-of-the-art validation methods. Both

DBCV and our noise adaptation approach showed promising results, confirming their efficacy and applicability to clustering validation.

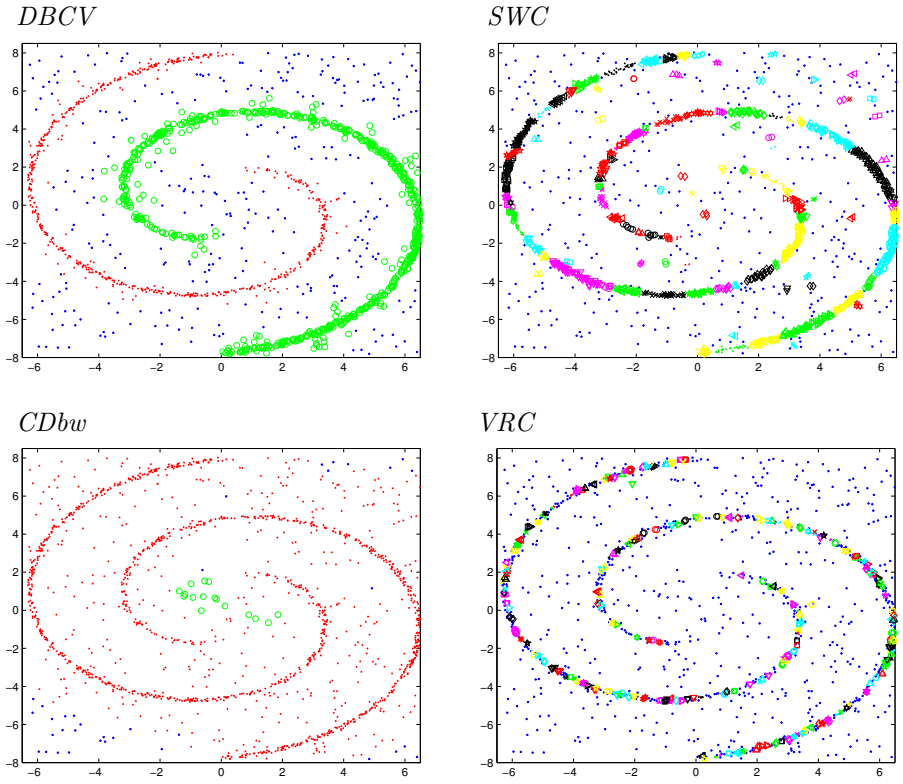


(a) Data set 1, best partitions found

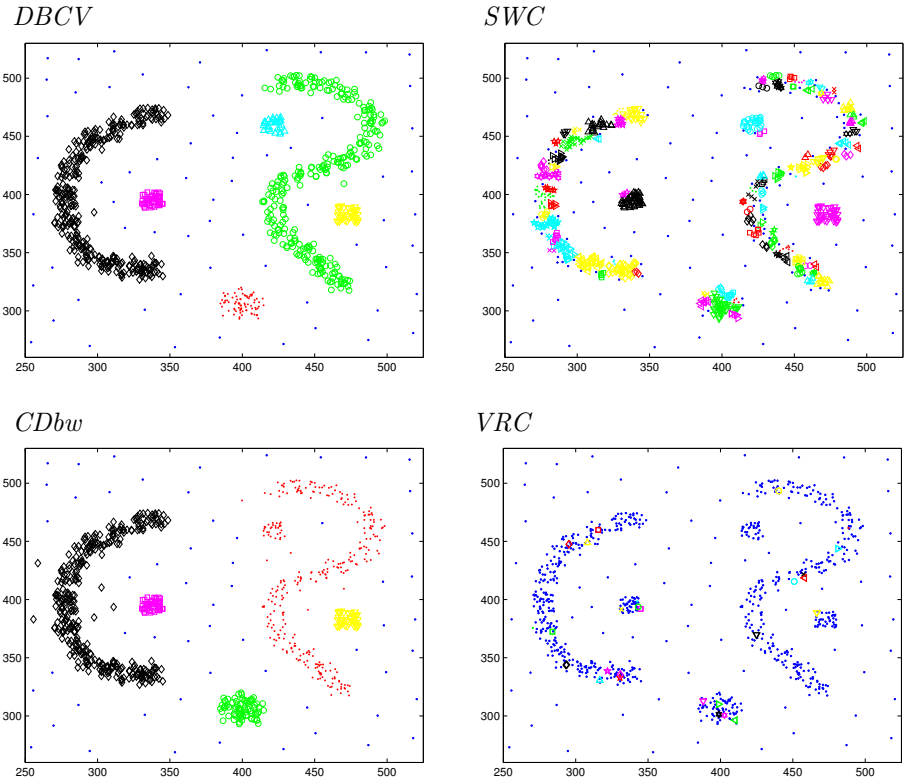


(b) Data set 2, best partitions found

Figure 3.2: Best partitions found for datasets.



(c) Data set 3, best partitions found



(d) Data set 4, best partitions found

Figure 3.3: Best partitions found for datasets.

Chapter 4

Hierarchical Density-Based Clustering

4.1 Introduction

In this chapter, we provide an extended description and discussion of our several hierarchical density-based clustering methods that are discussed in chronological and progressive order. Each approach utilizes some concepts from the previous approach and improves on it both theoretically and practically.

Gupta et al. [25] proposed a framework called *AUTO-HDS* for the automated clustering and visualization of biological data sets. In our method, called *Improved AUTO-HDS* (see Section 4.2), we extend that framework by showing that it is possible to get rid of the user-defined parameter r_{shave}, making the cluster extraction stage of AUTO-HDS simpler and more accurate. The asymptotic running time of AUTO-HDS is $O(n^3)$ [25]; we show that it is possible to reduce this asymptotic running time, while increasing the accuracy of the clustering solution. In addition, the cluster stability measure used in AUTO-HDS has a few undesirable properties: (i) it depends on the parameter r_{shave}, and (ii) the stability of a cluster in one branch of a hierarchy can be affected by characteristics (such as cardinality and density properties) of the

clusters in other branches of the hierarchy, thus affecting the final selection of a flat clustering solution. We discuss this stability in Section 4.2 and propose the Bounded Excess of Mass stability, which does not have any parameters and is not affected by clusters in other branches of a hierarchy, in Section 4.3.

Then we propose our generalized hierarchical density-based clustering method, called *GHDBSCAN*, in Section 4.3. GHDBSCAN is the generalization of the clustering algorithms that use Hartigan's model of density-based clustering, including DBSCAN and HDBSCAN algorithms. In this section, we first describe the modified version of the algorithm DBSCAN [23] and then we review the algorithm HDBSCAN, which was proposed in 2013 by Campello, Moulavi and Sander [73] with participation of the author. There are two essential components of Hartigan's model of density-based clustering, namely, density at each data point and density of a path between two data points. It is possible to replace these two components in a density-based algorithm to define a new density-based clustering algorithm. First we propose an improvement over the method used in HDBSCAN to estimate the density of a path between pairs of objects and show that our new approach estimates the density of a path between two objects in a more appropriate way. We then discuss the estimation of density at each data point. The previous methods including DBSCAN, Improved AUTO-HDS and HDBSCAN, use distance threshold (ε) and number-of-objects threshold (m_{pts}) to estimate the density of objects[1]. Such estimations have two undesirable properties: *a)* the estimated density of an object is based on the distance to a single point (the k^{th} nearest neighbor) and *b)* setting the value of the user-defined parameter m_{pts} is challenging and depends on the characteristics of each data set, such as the size of the data set and amount of background noise[2]. We propose a new approach to calcu-

[1]Other, similar methods (e.g., HMA [74] and OPTICS [30]) also estimate the density in a similar way.

[2]As Gupta et al. [25] noted and argued, different choices of m_{pts} can give dramatically different clusterings; choosing these parameters is thus a potential pitfall for the user.

late the density of the objects based on all other objects in a data set, and we show how this density estimation can be replaced with the one that estimates the density with the k^{th} nearest neighbor to resolve the aforementioned problems and make the HDBSCAN algorithm more accurate. We also propose two novel hierarchical density-based clustering methods GHDBSCAN(NMRD) and GHDBSCAN(NMRD+PF) by replacing the two essential components that have been noted.

4.2 Improved Automated Hierarchical Density Shaving (Improved AUTO-HDS)

AUTO-HDS is a useful clustering framework, proposed by Gupta et al. [25], that can be used to discover relevant data clusters from biological data sets. It is composed of a clustering stage, a cluster ranking and selection stage, and a visualization stage. The clustering stage is based on the HDS algorithm, proposed by the same authors [49]. HDS is a density-based hierarchical clustering algorithm that performs a sampling of the possible hierarchical levels (each of which represents a particular density threshold that discriminates between dense objects and noise) by using a geometric sampling rate controlled by a user-defined parameter, r_{shave}. The complete hierarchy will be obtained as $r_{shave} \to 0$. In this case, however, the asymptotic running time of the method is the same as the worst case running time of an analogous method (HMA) proposed by Wishart [74], namely $O(n^3)$, where n is the number of data objects [49, 113] and is prohibitive for large data sets. The use of sufficiently large values of r_{shave} allows for the sampling of a logarithmic number of hierarchical levels, reducing this complexity to $O(n^2 log(n))$ [25, 49, 113]. Further gains have been shown to be possible by using parallel computing techniques, but only for very low dimensional spaces [114]. However, the sampling of hierarchical levels performed by HDS represents a loss of information that may affect the results

provided by the subsequent stages of AUTO-HDS, i.e., the ranking/selection of clusters based on their stability and the visualization tool. In fact, when not all hierarchical levels are calculated, the birth and/or death of clusters cannot be precisely captured, so their stability cannot be exactly computed. In the worst case, a cluster may even be born and then disappear in between two sampled levels in such a way that it is not detected or presented to the user. Therefore, r_{shave} represents a trade-off between accuracy and the computational burden of AUTO-HDS.

In Section 4.2.1, we discuss that the complete hierarchy that would be obtained as $r_{shave} \to 0$ can actually be computed faster than $O(n^3)$ without any need for sampling. In Section 4.2.2, we discuss how the same procedure for ranking and selection of clusters used by AUTO-HDS can still be applied to the complete hierarchy regardless of r_{shave}. We also discuss some implications of our observations for the AUTO-HDS visualization tool.

4.2.1 Complete Hierarchical AUTO-HDS

Gupta et al. [25] have proposed a framework for clustering and visualization of biological data, the constituent parts of which are presumed to be replaceable. In order to replace the HDS clustering algorithm with another one capable of producing a fully compatible yet complete hierarchy, we need first to recall the discussions by the authors in [25] on the connections between HDS and other related density-based clustering algorithms of particular interest. In [25], when referring to the DBSCAN algorithm [23] and particularly to the choice of its parameters (m_{pts} and ε), Gupta et al. argued that different choices of ε and m_{pts} can give dramatically different clusterings; choosing these parameters is a potential pitfall for the user. While this is true concerning the combination of the two parameters, one should note that m_{pts} is fully equivalent to the parameter n_ε of HDS, which is a classic smoothing factor found in different density-based clustering algorithms [84, 73, 25, 23, 30, 26, 29, 27].

As concerns ε, the OPTICS algorithm [30] is known to produce a bar plot, called a reachability plot, that, for a given value of m_{pts}, encodes in a nested way all possible DBSCAN-like clusterings with respect to ε, except for eventual differences in the assignment of border objects. In [28], it was shown that a hierarchical dendrogram can be extracted from a reachability plot such that each level of the resulting hierarchy corresponds to a horizontal cut through the plot. This horizontal cut, in turn, corresponds to a DBSCAN-like clustering (with possible differences in the assignment of border objects) for a specific value of ε [30]. At this point, one should notice that the only difference between a DBSCAN clustering with respect to ε and the HDS clustering at density level r_ε is the presence of border objects in DBSCAN. As Gupta et al. [25] observe, "in DBSCAN, it is possible to label points that are not dense but rather on the periphery of a dense neighborhood."

However, removing the border objects from OPTICS and, accordingly, from the DBSCAN-like hierarchy that can be extracted from it, can be easily done by simply redefining the reachability distances in a symmetric way, as described in [84]. Therefore, as observed in [84], it follows that OPTICS reduces to an MST algorithm in a transformed space of symmetric reachability distances. This means that it produces a complete hierarchy in which the hierarchical levels are fully equivalent to those of HDS with respect to all density thresholds r_ε.

This means that the clustering procedure of AUTO-HDS can be replaced with the complete hierarchical dendrogram obtained from OPTICS reachability plot or the later procedure. Using OPTICS this can be implemented in $O(n^2 \log(n))$. The complete density-based hierarchy can even be computed faster in $O(n^2)$ as first proposed in Campello, Moulavi and Sander [72] and later elaborated in the HDBSCAN algorithm which is described in Section 4.3.1. Once the complete hierarchy is available, the relabeling and smoothing (particle removal) procedures described in [25] can be normally applied; this procedure is discussed in more detail in Section 4.4.3.

4.2.2 Ranking, Selection, and Visualization

Gupta et al. [25] defined the stability of a cluster as the number of shavings between the first and the last iteration of HDS in which the cluster appears:

$$Stab(C) = \frac{log(n_c^e) - log(n_c^{s-1})}{log(1 - r_{shave})}$$

where n_{c_i} is the total number of objects inside cluster C at the ith hierarchical level, e is the level where cluster C first appears, and s is the last level at which C survives. The selection of clusters described in [25] depends only on the ranking (relative ordering) of the clusters. As the authors of [25] observe, the denominator of *Stab(C)* is a constant for all clusters C. This means that the ranking (and selection) of clusters can be performed using only the total number of objects inside a cluster when the cluster appears (n_c^e) and disappears (n_c^{s-1}). In HDS, however, the total number of objects inside a cluster implicitly depend on r_{shave}. By contrast, in the complete hierarchy that can be produced as described in Section 4.2.1, they depend only on the (ranks of the) density thresholds r_ε associated with their respective hierarchical levels. These terms can thus be readily derived from the hierarchy or even pre-computed and stored during its construction, if desired. Hence, r_{shave} is not needed when the complete hierarchy is considered. In the following proposition we prove this property.

Proposition 4.2.1. *Consider two clusters C_i and C_j, in a hierarchy. The relative stability of clusters C_i and C_j is not dependent on parameter r_{shave} and can be computed using the following formula:*

$$RelativeStability(C_i, C_j) = \log \frac{\frac{n_{c_i}^e}{n_{c_i}^{s-1}}}{\frac{n_{c_j}^e}{n_{c_j}^{s-1}}}$$

Proof. Proof is provided in Appendix A on page 151. □

Visualization

One side effect of the geometric sampling controlled by r_{shave} is a further compaction of the hierarchy allowing for a log-scale visualization that enables us to emphasize smaller clusters with respect to bigger ones, as observed by the authors in [25]. A direct consequence of having the complete hierarchy available is that, if desired, these sampled hierarchical levels can be pre-computed and stored during the construction of a hierarchy, or can be derived later from the final hierarchy.

This means that such a visualization is now possible for any value of r_{shave} without the need for re-clustering. In other words, one can still use r_{shave} for visualization, if desired, and results for different values can be produced simply by sampling the levels of the complete hierarchy.

4.3 Generalized Hierarchical Density-Based Clustering

In this section, we introduce a generalized hierarchical clustering method, GHDBSCAN, that can be seen as a conceptual and algorithmic improvement over OPTICS and HDBSCAN. Before describing the hierarchical density-based clustering methods, we first describe a slightly modified version of the DBSCAN algorithm [73] and its properties. Our definitions follow the standard definitions of DBSCAN [23] and the only slight difference as described in [84], is in the way that density connection between two objects is calculated.

The Algorithm DBSCAN*. Let $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ be a data set of n objects, and let D be an $n \times n$ matrix containing the pairwise distances $d(\mathbf{x}_p, \mathbf{x}_q)$, $\mathbf{x}_p, \mathbf{x}_q \in \mathbf{X}$, for a metric distance $d(\cdot, \cdot)$.[3] We define density-based clusters based on *core objects* alone:

[3]The matrix D is not required if distances $d(\cdot, \cdot)$ can be computed from $\mathbf{X}$ on demand.

Definition 4.3.1. *(Core Object Condition): The condition in which the ε neighborhood of an object contains more than or equal to m_{pts} objects, i.e., $|\mathbf{N}_\varepsilon(\mathbf{x}_p)| \geq m_{pts}$, is called core object condition for an object $\mathbf{x}_p$ where $\mathbf{N}_\varepsilon(\mathbf{x}_p) = \{\mathbf{x} \in \mathbf{X} \mid d(\mathbf{x}, \mathbf{x}_p) \leq \varepsilon\}$ and $|\cdot|$ denotes cardinality.*

Definition 4.3.2. *(Core Object and Noise Object): An object $\mathbf{x}_p \in \mathbf{X}$ is called a core object with respect to parameter ε if the core object condition holds for object $\mathbf{x}_p$. An object is called* noise *if it is not a core object.*

Definition 4.3.3. *(Core Distance): The* core distance *of an object $\mathbf{x}_p \in \mathbf{X}$ with respect to m_{pts}, $d_{core}(\mathbf{x}_p)$, is the distance from $\mathbf{x}_p$ to its m_{pts}-nearest neighbor (including $\mathbf{x}_p$). It can be seen as minimum possible ε that core condition holds for the object.*

Definition 4.3.4. *(ε-Reachable): Two objects $\mathbf{x}_p$ and $\mathbf{x}_q$ are* ε-reachable *if $\mathbf{x}_p \in \mathbf{N}_\varepsilon(\mathbf{x}_q)$ and $\mathbf{x}_q \in \mathbf{N}_\varepsilon(\mathbf{x}_p)$ and core object condition holds for both objects with respect to ε and m_{pts}.*

Definition 4.3.5. *(Density-Connected): Two* core *objects $\mathbf{x}_p$ and $\mathbf{x}_q$ are* density-connected *with respect to ε and m_{pts} if they are directly or transitively ε-Reachable.*

Definition 4.3.6. *(Cluster): A* cluster $\mathbf{C}$ *with respect to ε and m_{pts} is a non-empty maximal subset of $\mathbf{X}$ such that every pair of objects in $\mathbf{C}$ is density-connected.*

Based on these definitions, we can devise an algorithm DBSCAN* (similar to DBSCAN) that conceptually finds clusters as the connected components of a graph in which the objects of $\mathbf{X}$ are vertices and every pair of vertices is adjacent if and only if the corresponding objects are ε-Reachable with respect to user-defined parameters ε and m_{pts}. Non-core objects are labeled as noise.

Note that the density of each object and each path connecting two objects inside such a connected component graph are greater than the density threshold

$\frac{1}{\varepsilon}$. Because, every pair of vertices is adjacent if and only if the corresponding objects are ε-Reachable, thus, there is always a path between the objects in a connected component (a cluster) in such a graph with all edge weights (mutual reachability distances) less than or equal to ε. In the following proposition we prove that core distance of each object inside such a connected component is less than or equal to ε. This means that density of objects and a path between pair of objects inside a connected component (a cluster) is greater than or equal to density threshold $\frac{1}{\varepsilon}$ ($d_{core} \leq \varepsilon$).

Proposition 4.3.1. *Let X be a set of objects in the connected components (clusters) of graph described above. Core distances of all such objects are less than or equal to ε.*

Proof. Let x_p be any arbitrary object and in X. This means that there is at least one edge in the connected component graph with weight less than or equal to ε that is connected to x_p, let x_q be the vertex in the other side of this edge. Thus based on Definition 4.3.4, $d_{core}(x) \leq \varepsilon$. □

Note that the original definitions of DBSCAN also include the concept of *border* objects, i.e., non-core objects that are within the ε-neighborhood of a core object (border objects) are also included inside the cluster. Our new definitions are more consistent with a statistical interpretation of clusters as connected components of a level set of a density (as defined, e.g., in [3])[4], since border objects do not technically belong to the level set as their estimated density is below the threshold. The new definitions also allow us to observe a precise relationship between DBSCAN* and its hierarchical version.

[4]Hartigan noted that "For a cluster C at level f_o, the density inside C is no less than f_o, but for every path connecting x in C to y outside C the density somewhere on the path is less than f_o." This holds true for all objects belonging to a cluster C that obtained by the DBSCAN* algorithm, and not true for border objects in DBSCAN clusters.

4.3.1 The HDBSCAN Clustering Algorithm

Here we describe hierarchical clustering method, HDBSCAN [73], which has as its single input parameter a value for m_{pts}, which is a classic smoothing factor in density estimates. Different density levels in the resulting density-based cluster hierarchy will thus correspond to different values of the radius ε.

For a proper formulation of the density-based hierarchy, we first define the notions of symmetric reachability distance (following the definition used in [84]) and the notion of a *conceptual*, transformed proximity graph.

Definition 4.3.7. *(Mutual Reachability Distance): The* mutual reachability distance *between two objects* $\mathbf{x}_p$ *and* $\mathbf{x}_q$ *in* $\mathbf{X}$ *with respect to* m_{pts} *is defined as* $d_{mreach}(\mathbf{x}_p, \mathbf{x}_q) = \max\{d_{core}(\mathbf{x}_p), d_{core}(\mathbf{x}_q), d(\mathbf{x}_p, \mathbf{x}_q)\}$.

Definition 4.3.8. *(Mutual Reachability Graph): The mutual reachability graph is a complete graph,* G_{mrd}*, in which the objects of* $\mathbf{X}$ *are vertices and the weight of each edge is the mutual reachability distance between the respective pair of objects.*

Let $G_{mrd,\varepsilon} \subseteq G_{mrd}$ be the graph obtained by removing all edges with weights greater than ε from G_{mrd}. From Definitions 4.3.2, 4.3.6, and 4.3.8, it can be inferred that DBSCAN* clusters (with m_{pts} and ε) are the connected components of core objects in $G_{mrd,\varepsilon}$ while the remaining objects are noise. Consequently, all DBSCAN* partitions for $\varepsilon \in [0, \infty)$ can be produced in a *hierarchical* way by removing edges in decreasing order of weight from G_{mrd}.

Proposition 4.3.2. *Let* $\mathbf{X}$ *be a set of* n *objects described in a metric space by* $n \times n$ *pairwise distances. The partition of this data obtained by DBSCAN* with respect to* m_{pts} *and* ε *is identical to the one obtained by first running Single-Linkage over the transformed space of mutual reachability distances, then cutting the resulting dendrogram at level* ε *of its scale, and treating all resulting singletons with* $d_{core}(\mathbf{x}_p) > \varepsilon$ *as a single class representing "Noise."*

Proof. Proof sketch as per discussion above, after Definition 4.3.8. □

Algorithm 1: HDBSCAN main steps

1. Compute the core distance with respect to m_{pts} for all data objects in $\mathbf{X}$.
2. Compute an extended MST (MST_{ext}) of Mutual Reachability Graph (G_{mrd}), by first constructing an MST of G_{mrd} and then adding for each vertex a "self-edge" with the core distance of the corresponding object as weight.
3. Extract the HDBSCAN hierarchy as a dendrogram from MST_{ext}:
    3.1 For the root of the tree assign all objects the single cluster.
    3.2 Iteratively remove all edges from MST_{ext} in decreasing order of weights (in case of ties, edges must be removed simultaneously):
        3.2.1 Before each removal, set the dendrogram scale of the current hierarchical level as the weight of the edge(s) to be removed.
        3.2.2 After each removal, assign labels to the connected component(s) that contain(s) the end vertex(-ices) of the removed edge(s), to obtain the next hierarchical level: assign a new cluster label to a component if it still has at least one edge, else assign it a null label ("noise").

Proposition 4.3.2 states that a hierarchical version of DBSCAN* can be implemented by first computing a single-linkage hierarchy on the space of *transformed* distances (i.e., mutual reachability distances) and then processing this hierarchy to identify connected components and noise objects at each level. A more efficient and elegant equivalent solution is described in the following.

A density-based cluster hierarchy has to represent the fact that an object o is considered as noise below the level l that corresponds to o's core distance. To represent this in a dendrogram, consider including an additional dendrogram node for o at level l representing the cluster containing o at that level and higher. To directly construct such a hierarchy, consider an extension of a an MST of the Mutual Reachability Graph G_{mrd}, from which the extended dendrogram by removing edges in decreasing order of weight can be constructed. More precisely, the MST extended with edges connecting each vertex o to itself (self-edges), where the edge weight is set to the core distance of o. These "self-edges" will then be considered when removing edges.

Algorithm 1 shows the pseudo-code for HDBSCAN, which has as inputs a value for m_{pts} and the data set $\mathbf{X}$. It produces a clustering tree that contains all partitions obtainable by DBSCAN* (with respect to m_{pts}) in a hierarchical, nested way we call it the *HDBSCAN hierarchy*.

4.4 Generalized Hierarchical Density-Based Clustering, GHDBSCAN

In this section, we discuss the essential components of the density-based clustering methods and try to generalize those approaches. Before introducing the formal definitions required to describe this approach, we first give basic definitions and an overview of the approach. It is based on Hartigan's [3] informal definition of density-based clusters, which "may be thought of as regions of high density separated from other such regions by regions of low density." Based on this informal definition he described the concept of a density-contour cluster and density-contour tree.

According to Hartigan's model [3], a *density-contour cluster* is a subset $\mathbf{C} \subset \Re$ at density level λ, such that: (i) every object $x \in \mathbf{C}$ satisfies $f(x) \geq \lambda$, (ii) $\mathbf{C}$ is connected, and (iii) $\mathbf{C}$ is maximal. f(x) being the density function defined for each x as a value proportional to the number of points per unit volume at x.

Consider the example in Figure 4.1, in which there are two clusters C_1 and C_2 at density level $\lambda = 2$. This means that: (i) For every object $x \in C_1 \cup C_2$, we have $f(x) \geq 2$. (ii) For every pair of objects $x_i^{C_1}, x_j^{C_1} \in C_1$ there is a path $x_i^{C_1}, x_{i+1}^{C_1}, x_{i+2}^{C_1}, \ldots, x_j^{C_1}$, between the two objects such that $x_l^{C_1}$ is linked to $x_{l+1}^{C_1}$ for each $i \leq l \leq j$. This link can be seen as an approximation of the density of the area between objects $x_l^{C_1}$ and $x_{l+1}^{C_1}$, and it should be greater than $\lambda = 2$. This holds true as well for all pairs of objects in cluster C_2. In addition, for every path connecting two objects that are not in the same cluster, density drops below $\lambda = 2$ somewhere along the path[5]. (iii) For every object x with $f(x) \geq 2$, if there is a path with density greater than or equal to two that connects x to any object inside cluster C_1, then $x \in C_1$ (this also holds for

[5]This pair of objects can consist of objects from separate clusters C_1 and C_2 or noise objects in the areas outside these two clusters.

similar object connected to cluster C_2).

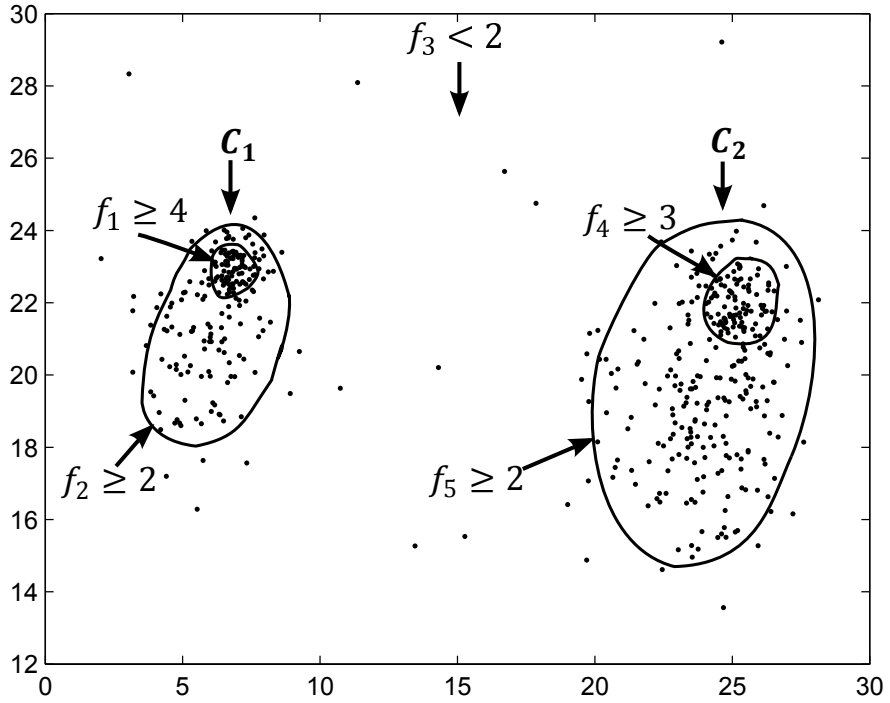


Figure 4.1: At density level 2, two clusters C_1 and C_2 can be found. Objects inside each cluster have densities greater than 2, and there is a path with density greater than 2 that connects objects inside each cluster. On every path connecting objects that are not in the same cluster, density drops below 2 somewhere along the path.

As noted above, Hartigan's model has two essential components: *(a)* density f defined at each object, and *(b)* density f defined at each path between the objects in the space. We argue that clustering algorithms based on Hartigan's concept of density-based clustering, including density-contour tree, DBSCAN, OPTICS and HDBSCAN, contain these two major components. We then show that it is possible to obtain a new density-based clustering algorithm by replacing these two components in an algorithm.

Consider that the density of objects is estimated by parameters ε and min_{pts} and that we use the HDBSCAN algorithm. As noted in Section 4.3.1, let $G_{mrd,\varepsilon} \subseteq G_{mrd}$ be the graph obtained by removing all edges from G_{mrd} having weights greater than ε (density less than $\frac{1}{\varepsilon}$). Based on Definitions 4.3.2, 4.3.6,

and 4.3.8, it is straightforward to infer that clusters defined by DBSCAN* with respect to m_{pts} and ε are the connected components of core objects in $G_{mrd,\varepsilon}$; the remaining objects are noise.

Each connected component in the $G_{mrd,\varepsilon}$ graph has the following properties: (i) every object has a density ($\frac{1}{d_{core}}$) greater than or equal to $\frac{1}{\varepsilon}$ (see Proposition 4.3.1); (ii) for every pair of objects (nodes in $G_{mrd,\varepsilon}$) inside each cluster, meaning that both objects are in the same connected component graph, there is a path between these objects in the $G_{mrd,\varepsilon}$ graph such that the edge weights (d_{mreach}) on this path are always less than or equal to ε (meaning that $\frac{1}{d_{mreach}} \geq \frac{1}{\varepsilon}$) ; (iii) every $G_{mrd,\varepsilon}$ graph component is maximal.

As we can easily see, the essential components of HDBSCAN are similar to those of Hartigan's model: *(a)* density f defined at each object, which is at least $\frac{1}{\varepsilon}$, and *(b)* density f defined on a path between the objects in the space, which essentially depends on the density between pairs of objects and which is equal to $\frac{1}{d_{mreach}}$. In the following we discuss the properties of these components and propose a replacement for each of them.

For the sake of simplicity and without loss of generality, the inverse of density values have been utilized in the DBSCAN and HDBSCAN algorithms (see Definitions 4.3.2 and 4.3.7), for two main reasons: (i) it is easier to discuss clustering using distances instead of densities, and (ii) the final solution is similar in both cases. We will follow the same trend in the following.

4.4.1 Estimation of Density on Each Path in Data Space

To estimate the density area between two objects by considering their density and distance, we defined *mutual reachability distance* in Section 4.3.1 (Definition 4.3.7) as follows:

Mutual Reachability Distance: The mutual reachability distance *between two objects* $\mathbf{x}_p$ *and* $\mathbf{x}_q$ *in* $\mathbf{X}$ *with respect to* m_{pts} *is defined as* $d_{mreach}(\mathbf{x}_p, \mathbf{x}_q) = \max\{d_{core}(\mathbf{x}_p), d_{core}(\mathbf{x}_q), d(\mathbf{x}_p, \mathbf{x}_q)\}$.

Our definition of *mutual reachability distance* is similar to the definition of reachability distance in the OPTICS algorithm [30], but the difference is that our definition is symmetric (we do not include border objects that were originally defined in DBSCAN, which makes our definition symmetric). Although the clustering created by a cut level through the OPTICS plot tries to include border objects defined in DBSCAN, it also misses some border objects. This is described in [30], which states that the clustering created through a cut: "is nearly indistinguishable from a clustering created by DBSCAN. Only some border objects may be missed when [clusters are] extracted ...".

Using the symmetric mutual reachability distance allows us to build an undirected graph of mutual reachability distances: however, like the definition of reachability distance in OPTICS [30], it has unsmooth behaviour, and in many cases it does not differentiate between the density of areas with different density characteristics[6]. We illustrate this problem in Figure 4.2, in which the circles represent the core distance of the objects o_1, o_2, o_3, o_4 and we have $(d_{core}(o_2) = d_{core}(o_3)) \leq (d_{core}(o_1) = d_{core}(o_4))$. Based on our definition of mutual reachability distance, $d_{mreach}(o_1, o_2) = d_{mreach}(o_1, o_3) = d_{mreach}(o_1, o_4)$. However, the density at points o_2 and o_3 is estimated to be higher than the density at point o_4, and thus the mutual reachability distances $d_{mreach}(o_1, o_2)$ and $d_{mreach}(o_1, o_3)$ should be smaller than $d_{mreach}(o_1, o_4)$. In addition, the proximity of point o_2 to o_1 is closer than the proximity of o_3 to o_1; therefore, the mutual reachability distance of o_2 and o_1 should be smaller than that of o_3 and o_1.

We define a new mutual reachability distance, based on the core distances of objects, that is symmetric and consider the differences (as discussed above) as follows:

[6]Mutual reachability graphs and MSTs created using these mutual reachability distances have many edges with the same weight, and usually there are many possible MSTs for each data set.

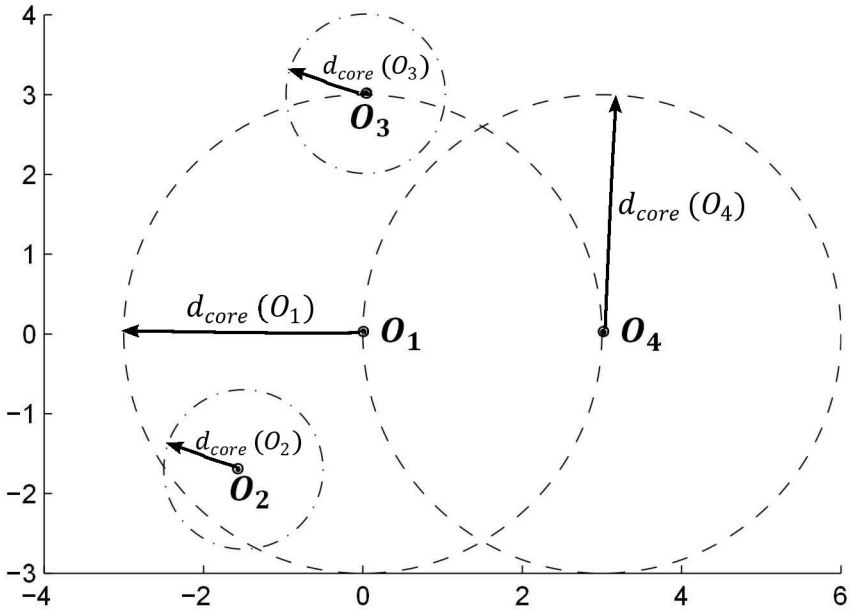


Figure 4.2: Objects o_1, o_2, o_3, o_4 and their core distances.

Definition 4.4.1. *(New Mutual Reachability Distance): The* new mutual reachability distance *between two objects* $\mathbf{x}_p$ *and* $\mathbf{x}_q$ *in* $\mathbf{X}$ *with respect to core distance and distance is defined as* $d_{nmreach}(\mathbf{x}_p, \mathbf{x}_q) = \frac{(d_{core}(\mathbf{x}_p)+d_{core}(\mathbf{x}_q))}{2} + d(\mathbf{x}_p, \mathbf{x}_q)$

We can easily see that with the new definition, we have $d_{nmreach}(o_1, o_2) < d_{nmreach}(o_1, o_3) < d_{nmreach}(o_1, o_4)$, as desired. Our new mutual reachability distance also has the properties that have so far been discussed for mutual reachability distance. In the following we state the property that we stated in Proposition 4.3.1 for our new mutual reachability distance.

Proposition 4.4.1. *Let* $n_{G_{nmrd,\varepsilon}}$ *be a set of objects in the connected components (clusters) in* $G_{nmrd,\varepsilon}$[7]*. Core distance of all such objects are less than or equal to* ε*.*

Proof. The proof is a bit trickier than that provided for Proposition 4.4.1. Let o_1 and o_2 be any arbitrary pair of objects whose mutual edge belongs to the set of edges in $G_{nmrd,\varepsilon}$. This means that the weight of the edge (o_1, o_2) is less than

[7] G_{nmrd} is a graph similar to G_{mrd} and $G_{nmrd,\varepsilon}$ is a graph similar to $G_{mrd,\varepsilon}$ that were built using our new mutual reachability distances

or equal to ε ($d_{nmreach}(o_1, o_2) \leq \varepsilon$). Without loss of generality let's consider that $d_{core}(o_2) \leq d_{core}(o_1)$. If $d_{core}(o_1) \leq d(o_1, o_2)$ we can easily infer that $d_{core}(o_1) \leq d_{nmreach}(o_1, o_2) \leq \varepsilon$. If alternatively, $d(o_1, o_2) < d_{core}(o_1)$, as we can see in Figure 4.2, the object o_2 is inside the $d_{core}(o_1)$ neighborhood of the object o_1, and as we can also see in the Figure 4.2, the $d_{core}(o_1)$ neighborhood of the object o_1 cannot contain the whole neighborhood of the object o_2. Otherwise, the $d_{core}(o_1)$ must be smaller than the current $d_{core}(o_1)$, which is not possible because $d_{core}(o_1)$ is defined as the smallest neighborhood that contains min_{pts} objects. Therefore, $d_{core}(o_1) \leq d_{core}(o_2) + d(o_1, o_2)$

$$
\begin{aligned}
&\Rightarrow d_{core}(o_1) \leq d_{core}(o_2) + 2 \times d(o_1, o_2) \\
&\Rightarrow \frac{d_{core}(o_1)}{2} \leq \frac{d_{core}(o_2)}{2} + d(o_1, o_2) \\
&\Rightarrow \frac{d_{core}(o_1)}{2} + \frac{d_{core}(o_1)}{2} \leq \frac{d_{core}(o_1)}{2} + \frac{d_{core}(o_2)}{2} + d(o_1, o_2) \\
&\Rightarrow d_{core}(o_1) \leq \frac{d_{core}(o_1) + d_{core}(o_2)}{2} + d(o_1, o_2) = d_{nmreach}(o_1, o_2) \\
&\Rightarrow d_{core}(o_1) \leq \varepsilon \Rightarrow d_{core}(o_2) \leq \varepsilon
\end{aligned}
$$

□

With this new definition of mutual reachability distance, the problem of border objects is also solved, because when calculating mutual reachability distance, we distinguish whether an object with low density is close to a dense object or the same object with low density is close to a non-dense object.

By utilizing the new concept of mutual reachability distance, we can define our algorithm GHDBSCAN(NMRD). Algorithm 2 gives the pseudo-code for GHDBSCAN(NMRD), similar to the standard pseudo-code for SL algorithm and also HDBSCAN, with m_{pts} value and the data set $\mathbf{X}$ as an input. Note that, in contrast to HDBSCAN, we did not include the self-edges in this algorithm meaning that a cluster as in Definition 4.3.6 needs to have at least two objects. However one can consider isolated core object similar to the HDBSCAN as a

cluster by adding the self-edges in step 3 of the algorithm.

In the following Section (4.4.2), we discuss a new possible way of estimating the density of each object.

4.4.2 Estimation of Density at Each Object

To estimate the density of an object, a traditional approach is to take the inverse of the threshold distance necessary to find m_{pts} objects within this threshold [73, 23, 30, 3]; this approach is used in the Improved-AUTO-HDS and HDBSCAN algorithms. This approach, however, has its disadvantages. First, the estimated density of an object is based on the distance to a single point (the k^{th} nearest neighbor). As such, this estimate is not as robust as density estimates that consider more objects from the neighborhood, as does the Gaussian kernel density estimate. Second, this definition introduces a parameter m_{pts}. Setting user-defined parameter m_{pts} is challenging as it depends on the characteristics of each data set, such as size of data set and amount of background noise.

In the following, we provide a new, more robust, and parameter-less definition of a density that can be used in the mutual reachability distance and new mutual reachability distance[8]. To achieve this goal, the density definition should have the following properties:

(1) To be more robust, a density estimate should not depend on a single point, but should rather consider all of the points in such a way that closer objects make a greater contribution to the density than do farther objects. This is a common property in density estimate methods such as Gaussian kernel density estimation.

(2) In the definition of mutual reachability distance [84] and new mutual reachability distance, which we introduced in Section (4.4.1) and which we

[8]The definition of core distance is explored in more detail in the context of density-based clustering validation in Chapter 3. Here we give a concise and self-contained overview of it and its properties and repeat some of those material and discuss them in the context of the density-based clustering.

apply in our methods, the core distance of an object is compared to the distance of the object to other objects. Therefore the core distance should be comparable to those distances.

(3) The core distance of an object should be approximately that of the k^{th} nearest neighbor from the object, where k is not too large (hence the core distance represents a small neighborhood of the object).

We define the core distance of an object $\mathbf{o}$ ($a_{pts}\,coredist$) with respect to all other objects in a data set as follows.

Definition 4.4.2 (Core Distance of an Object)**.** The *all-points-core-distance* of an object $\mathbf{o}$, belonging to the d-dimensional data set X with respect to all other $n-1$ objects inside that data set is defined as:

$$a_{pts}\,coredist(\mathbf{o}) = \left(\frac{\sum_{\substack{o_i \in X \\ o_i \neq o}} \left(\frac{1}{d(\mathbf{o}, o_i)} \right)^d}{n-1} \right)^{-\frac{1}{d}} \tag{4.1}$$

In Chapter 3, Section 3.2 we proved that properties one and two hold for this definition of core distance.

In Propositions 4.4.2 and 4.4.3 we show again that the third property holds for our definition of $a_{pts}\,coredist$ in uniform distribution using Euclidean distance and we describe the behaviour of this density in other distributions intuitively based on its behaviour in uniform distribution.

Proposition 4.4.2. *Let n objects be uniformly distributed random variables in a d-dimensional unit hypersphere and $\boldsymbol{o}$ be an object in the center of this hypersphere. For the all points core distance of $\boldsymbol{o}$ we have:*

$$a_{pts}\,coredist(\mathbf{o}) = ((ln(n-1) + \gamma + \varepsilon)^{-\frac{1}{d}} \approx \ln(n)^{-\frac{1}{d}} \tag{4.2}$$

where $\gamma \approx 0.5772$ and $\varepsilon \approx \frac{1}{2n}$ which approaches to zero as n goes to infinity.

Proof. Proof is provided in Appendix A on page 145. □

Note that when we have a distribution other than the uniform distribution, its behavior follows our first desired property. If most of the objects are close to $\mathbf{o}$, *density* and a_{pts}*coredist* tend to be larger and smaller values respectively. By contrast, if most of the objects are distributed far away from $\mathbf{o}$, *density* and a_{pts}*coredist* tend to assume smaller values.

Proposition 4.4.3. *For calculated* $a_{pts}coredist(\mathbf{o})$ *in Proposition 4.4.2, we have:*

$$a_{pts}coredist(\mathbf{o}) \approx \ln(n)^{-\frac{1}{d}} \approx KNN(\mathbf{o}, j), \tag{4.3}$$

with j *being the closest natural number to* $\frac{n}{\ln(n)}$ *and* $KNN(\mathbf{o}, j)$ *being the expected value of* j^{th} *nearest neighbor distance to object* o*.*

Proof. Proof is provided in Appendix A on page 147. □

This proposition shows that the core distance of the object $\mathbf{o}$ approximating distance to the same K^{th} nearest neighbor independent of the dimensionality of the data space.

By using a_{pts}*coredist* value for each object as the core distance of the object and using these core distance values to calculate the new mutual reachability distance as defined in Definition 4.4.1, we can define our algorithm GHDBSCAN(NMRD+PF). Algorithm 2 gives the pseudo-code for GHDBSCAN(NMRD+PF) with the data set $\mathbf{X}$ as an input. As described in Section 4.4.1, one can consider isolated core objects as a cluster by adding the self-edges in step 3 of the algorithm.

4.4.3 Hierarchy Simplification

The hierarchies of density-based clustering algorithms introduced so far can easily be visualized as a traditional dendrogram or as related representations. However, these plots are not easy to interpret or process for large and "noisy" data sets, so it is a fundamental problem to extract from a dendrogram a

Algorithm 2: GHDBSCAN (NMRD+PF) and GHDBSCAN (NMRD) main steps.

1. Compute the a_{pts}*coredist* core distance of each object with respect to all other objects (in case of GHDBSCAN(NMRD) compute core distance with respect to m_{pts}).
2. Compute the new mutual reachability distance between pairs of objects using calculated core distances of the objects.
3. Compute an MST of New Mutual Reachability Graph (G_{nmrd}), by constructing an MST of G_{nmrd}.
4. Extract the GHDBSCAN hierarchy as a dendrogram from MST:
    4.1 For the root of the tree assign all objects the single cluster.
    4.2 Iteratively remove each edge from MST in decreasing order of weights (in case of ties, edges must be removed simultaneously):
        4.2.1 Before each removal, set the dendrogram scale of the current hierarchical level as the weight of the edge(s) to be removed.
        4.2.2 After each removal, assign labels to the connected component(s) that contain(s) the end vertex(-ices) of the removed edge(s), to obtain the next hierarchical level: assign a new cluster label to a component if it still has at least one edge, else assign it a null label ("noise").

summarized tree of only "significant" clusters. We provide a simplification of these hierarchies based on a fundamental observation about estimates of the level sets of continuous-valued probability density functions (p.d.f.), which refers back to Hartigan's concept of *rigid clusters* [3], and which has been employed similarly by Gupta et al. in [25]. For a given p.d.f., there are only two possibilities for how the connected components of a continuous density level set evolve when increasing the density level [85] in which the size of the component reduces but remains connected, up to a density threshold at which either *(i)* the component is divided into smaller ones, or *(ii)* it disappears. This observation can be applied to our hierarchies to select only those hierarchical levels in which new clusters arise by a "true" split of a cluster, or in which clusters disappear; these are the levels at which the most significant changes in the clustering structure occur. When increasing the density level, the ordinary removal of noise objects from a cluster is not a true split; only the size of the cluster reduces in this case, so it should keep the same label.

As in [73] we can generalize this idea by setting a minimum cluster size, a commonly used practice in real cluster analysis (see, e.g., the notion of a *particle*

Algorithm 3: Hierarchy Simplification Using Parameter $m_{clSize} \in \mathbb{N}$

After removing each edge to obtain the next hierarchical level in a hierarchy as described before, process each cluster that contained the edge(s) just removed, by relabeling its resulting connected subcomponent(s):

Label *spurious* subcomponents as noise by assigning them the null label. If all subcomponents of a cluster are *spurious*, then the **cluster has disappeared**.

Else, if a single subcomponent of a cluster is *not spurious*, keep its original cluster label (**cluster just reduced the size**).

Else, if two or more subcomponents of a cluster are *not spurious*, assign new cluster labels to each of them (**"true" cluster split**).

in AUTO-HDS [25]). With a minimum cluster size, $m_{clSize} \geq 1$, components with fewer than m_{clSize} objects are disregarded, and their disconnection from a cluster does not establish a true split. We can adapt the produced hierarchies accordingly by changing labelling Steps of algorithms 1 and 2 (step 3.2.2 in 1 and step 4.2.2 in 2) as shown in Algorithm 3. In simplification process a connected component is deemed *spurious* if it has fewer than m_{clSize} objects. Any spurious component is labeled as noise and its removal from a larger component is not considered as a cluster split. In practice, this can reduce the size of the hierarchy dramatically. The optional parameter m_{clSize} represents an independent control for the smoothing of the resulting cluster tree, in addition to m_{pts} where needed. To make the hierarchies produced by algorithms that have m_{pts} parameter more similar to previous density-based approaches and to simplify their use, we can optionally set $m_{clSize} = m_{pts}$, which turns m_{pts} into a single parameter that acts as both a smoothing factor and a threshold for the cluster size.

4.4.4 Model Selection

In many applications, a user is interested in extracting a flat partition of the data, consisting of the best clusters. Those clusters, however, may have very different local densities and may not be detectable by a single, global density

threshold (i.e., a global cut through a hierarchical cluster representation). We describe here methods of automatically selecting the number and location of the best clusters. To select these clusters, we introduce our novel stability criterion, called Bounded Excess of Mass, in order to rank the clusters.

4.4.5 Bounded Excess of Mass Cluster Stability

Using a simplified hierarchy generated by the process discussed in Section 4.4.3, makes it easier to extract a flat clustering solution from a hierarchy. However to select a flat clustering solution from a hierarchy, we need to define the stability criterion for each cluster. Using stability values for each cluster makes it possible to extract a flat clustering solution from a hierarchy with maximum overall stability. In this section we describe our proposed stability criterion, called "Bounded Excess of Mass". Without loss of generality, let us initially consider that the data objects are described by a single continuous-valued attribute x. Following Hartigan's model [3], the density-contour clusters of a given density $f(x)$ on $\Re$ at a given density level λ are the maximal connected subsets of the level set defined as $\{x \,|\, f(x) \geq \lambda\}$. Most density-based clustering algorithms are, to some extent, based on this concept. As we described in Section 4.4, the differences between them lie in the way the density f at each object and density of a path between two objects are estimated.

Each of our described hierarchical density-based clustering algorithms produces a hierarchy with respect to all possible density thresholds λ. Intuitively, when increasing λ, clusters get smaller and smaller, until they disappear or break into sub-clusters; more prominent clusters will "survive" longer after they appear. To formalize this concept, we adapt the notion of *excess of mass* [115]. Imagine increasing the density level λ, and assume that a density-contour cluster $\mathbf{C}_i$ appears at level $\lambda_{min}(\mathbf{C}_i)$. The excess of mass of $\mathbf{C}_i$ is defined as:

$$E(\mathbf{C}_i) = \int \Big(f(x) - \lambda_{min}(\mathbf{C}_i) \Big) \, dx \tag{4.4}$$

For our density-based clustering hierarchies where we have a finite data set, thus having finite density thresholds, we can write Equation 4.4 as:

$$E(\mathbf{C}_i) = \sum_{x \in \mathbf{C}_i} \Big(f(x) - \lambda_{min}(\mathbf{C}_i) \Big) \tag{4.5}$$

Excess of Mass, as introduced above, has a few properties that make it inappropriate to use as a cluster stability: (i) It exhibits a monotonic behaviour along any branch of the hierarchical cluster tree. As a consequence, the excess of mass of a parent cluster is always greater than the sum of excess of mass of its children, thus Excess of Mass can not directly be used to compare the stabilities of nested clusters. (ii) If a few objects inside a cluster are very close to each other, the quality of that cluster goes to infinity. This causes two undesirable problems: *(a)* other objects inside that cluster will not have any effect on the quality of that cluster, and *(b)* it will not be meaningful to compare values close to infinity to find the best cluster. Although, the Relative Excess of Mass that was originally proposed in HDBSCAN method [73] resolves the first problem, it still suffers from the two problems mentioned in (ii). To resolve these problems we propose *Bounded Excess of Mass* by adapting the Excess of Mass and its relative version as a stability measure for a cluster as follows:

$$\textit{Bounded-}E(\mathbf{C}_i) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} \Big(\frac{\lambda_{max}(\mathbf{x}_j, \mathbf{C}_i) - \lambda_{min}(\mathbf{C}_i)}{\lambda_{max}(\mathbf{x}_j, \mathbf{C}_i) + \lambda_{min}(\mathbf{C}_i)} \Big)^d \tag{4.6}$$

where $\lambda_{min}(\mathbf{C}_i)$ is the minimum density level at which $\mathbf{C}_i$ exists, $\lambda_{max}(\mathbf{x}_j, \mathbf{C}_i)$ is the density level beyond which object $\mathbf{x}_j$ no longer belongs to cluster $\mathbf{C}_i$, and d is dimensionality of the data space. We can easily see that our proposed bounded excess of mass for cluster C_i is a number between 0 and $|C_i|$ where $|C_i|$ is the cardinality of the cluster C_i and our proposed stability criterion does not suffer from the aforementioned problems and has following properties: First, it

is local, which means that it can be computed independently for each cluster and that the stability of one cluster does not depend on the stability of the other clusters. Second, the sum of the stabilities of child clusters is comparable to the stability of their parent cluster. These two properties are required for a stability measure to be eligible for use in the optimal extraction of a flat solution from a hierarchy, which we describe it in Chapter 5 in detail. Third, the value of such stability is bounded, which is a desired though not necessary property. We define the stability of a clustering partition as the sum of the stabilities of its clusters. For example, in Figure 4.3, the bounded excess of mass stabilities for each cluster are illustrated. It is possible to extract several non-overlapping partitions from this cluster tree, e.g., the partition including clusters C_2 and C_3 which has a stability of 11 (Bounded-E(C_2) + Bounded-E(C_3) = 6 + 5 = 11). The partition including the clusters C_2, C_7, C_8 and C_9 is the most prominent flat partition, with a maximum stability of 25 among all possible partitions (Bounded-E(C_2) + Bounded-E(C_7) + Bounded-E(C_8) + Bounded-E(C_9)= 6 + 8 + 5 + 6 = 25). It is possible to find the most prominent flat partitions in every cluster tree by using a method analogous to the one used in error-based pruning of decision-tree classifiers. We apply this method to extract the most prominent partition from a cluster tree (not selecting the partition that contains only the root cluster) in our algorithms, see Chapter 5.

4.5 Complexity Analysis

In this section, we analyze the complexity of our methods. Consider first a scenario where the data set X with n d-dimensional objects is given as input. Provided that the distance between objects can be computed in $O(d)$ time, the core distances of all objects can be computed in $O(dn^2)$ time in the worst case (for both m_{pts} and $a_{pts}coredist$ core distances). When the core distances have

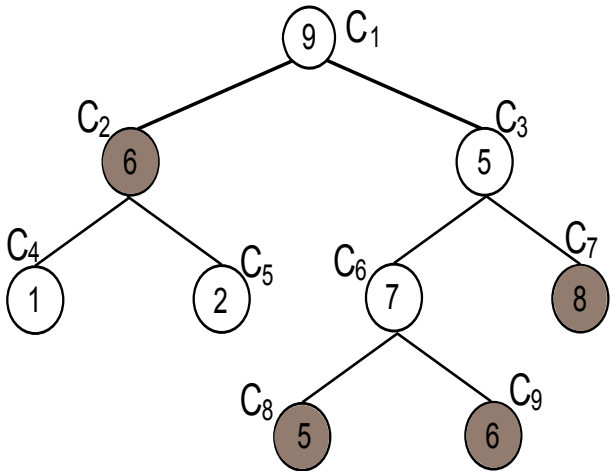


Figure 4.3: Bounded excess of mass stability values for each cluster in a cluster tree. The partition including the clusters C_2, C_7, C_8 and C_9 has maximum sum of stabilities.

been pre-computed, each mutual reachability distance can be computed in $O(d)$ time when the SL algorithm demands it. One of the fastest ways to compute SL is by using a divisive method based on the Minimum Spanning Tree (MST) [1]. It is possible to construct an MST in $O(dn^2)$ time by using an implementation of Prim's algorithm based on an ordinary list search. The divisive extraction of the SL hierarchy from the MST does not exceed this complexity. Once the hierarchy is available, the relabeling and smoothing procedures (removal of clusters smaller than m_{clSize}) can be normally applied. These procedures likewise do not exceed this complexity. The extraction of a flat solution from the hierarchy is $O(n)$ (see Section 5.3.2 for more detail).

In summary, then, the overall worst-case asymptotic time complexity of the algorithms, when the data set X is given as input, is $O(dn^2)$. Regarding main memory requirements, one needs $O(dn)$ space to store the data set X and $O(n)$ space to store the core distances (for both m_{pts} and $a_{pts}coredist$ core distances). The MST requires $O(n)$ space to be stored. In the divisive extraction stage, only the currently processed hierarchical level, which requires $O(n)$ space, is needed at any point in time. Hence, the overall space complexity of the algorithm is $O(dn)$. If the distance matrix M_S is given instead of the data

set X, the only change affecting the runtime is that one can directly access any distance $d_S(o_i, o_j)$ from M_S in constant time. In such a case, the computations no longer depend on the dimension d of the objects, and, as a consequence, the worst case time complexity decreases to $O(n^2)$. On the other hand, this requires that matrix M_S be stored in main memory, which results in $O(n^2)$ space complexity.

4.6 Experiments and Discussion

Data Sets. We report the performance on 6 individual data sets plus the average performance on 2 collections of data sets, representing a large variety of application domains and data characteristics (number of objects, dimensionality, number of clusters, and distance function). The first two data sets ("CellCycle-237", and "CellCycle-384") represent gene-expression data. CellCycle-237 and CellCycle-384 were made public by Yeung et al. [111]; they contain 237 respectively 384 objects (genes), 4 respectively 5 known classes, and have both 17 dimensions (conditions). For these data sets we used Euclidean distance on the z-score normalized objects, which is equivalent to using Pearson correlation on the original data. The next two data sets are the Glass, and Iris from the UCI Repository [116], containing 214, respectively 150 objects in 9, respectively 4 dimensions, with 7, respectively 3 classes. For these data sets we used Euclidean distance. The last two individual data sets consist of very high dimensional representations of text documents. In particular, "Articles-1442-5", made available upon request by Naldi et al. [117], is formed by 253 articles represented by 4636 dimensions. "Cbrilpirivson" [118] is composed of 945 articles represented by 1431 dimensions and is available at http://infoserver.lcad.icmc.usp.br/infovis2/PExDownload. The number of classes in all two document data sets is 5, and we used the Cosine measure as dissimilarity function.

In addition to individual data sets we also report *average* performance on two collections of data sets, which are based on the Amsterdam Library of Object Images (ALOI) [119]. Image sets were created by randomly selecting k ALOI image categories as class labels 100 times for each number $k = 2, 3, 4, 5$, then sampling (without replacement), each time, 25 images from each of the k selected categories, thus resulting in 400 sets, each of which contains 2, 3, 4, or 5 clusters and 50, 75, 100, or 125 images (objects). The images were represented using six different descriptors: color moments (144 attributes), texture statistics from the gray-level co-occurrence matrix (88 attributes), Sobel edge histogram (128 attributes), 1st order statistics from the gray-level histogram (5 attributes), gray-level run-length matrix features (44 attributes), and gray-level histogram (256 attributes). We report *average* clustering results for the texture statistics, denoted by "ALOI-TS88", which is typical for the individual descriptors. We also show results for a 6-dimensional representation combining the first principal component extracted from each of the 6 descriptors using PCA, denoted by "ALOI-PCA". We used Euclidean distance in both cases.

Algorithms and settings. In first set of experiments we compare the performance of the HDBSCAN (see Section 4.3.1) with original settings [73], with: *(i)* AUTO-HDS [25]; and *(ii)* a method referred to here as "OPTICS(AutoCl)," which involves first running OPTICS, and then using the method proposed by Sander *et al.* [28] to extract a flat partitioning. We set m_{pts} (n_ε in AUTO-HDS, m_{pts} in OPTICS) equal to 4 in all experiments. The speed-up control value ε in OPTICS is set to "infinity," thereby eliminating its effect. For AUTO-HDS, we set the additional parameters r_{shave} to 0.03 (following the authors' suggestion to use values between 0.01 and 0.05) and *particle size*, n_{part}, to $m_{pts} - 1$. In this set of experiments we show that HDBSCAN outperforms other algorithms. We then compare performance of HDBSCAN with performance of two new algorithms provided in Section 4.3, GHDBSCAN(NMRD) and GDBSCAN(NMRD+PF). To extract flat partition from the hierarchies the stability

is measured based on bounded excess of mass. For text data sets for calculating stabilities d is set to one. Minimum cluster size, m_{clSize}, was set equivalently to 4 in all algorithms.[9]

Quality Measures. The measure we report is the Adjusted Rand Index [75], denoted by "ARI," which is the measure commonly used in the literature. In addition, we also report the fraction of objects assigned to clusters (as opposed to noise), denoted by "%covered."

Clustering Results. The results obtained in our experiments are shown in bar plots in Figure 4.4 and Figure 4.5. Each sub-figure has two groups of bars; the group on the left side represents the ARI for each method, and the group on the right side represents %covered. We first compare HDBSCAN algorithm with other state-of-the-art algorithms, OPTICS(AutoCl) and AUTO-HDS, in Figure 4.4.

Note that HDBSCAN outperforms the other two methods in a large majority of the data sets (in many cases by a large margin) and, in almost all cases, it covers a larger fraction of objects while having high ARI values. A large fraction of clustered objects is only good when also the clustering quality is high. E.g., for CellCycle-384, OPTICS(AutoCl) covers 100% of the data, but results in an ARI of 0, a meaningless clustering. In the only case where HDBSCAN does not perform best, CellCycle-237, its ARI and %covered is much higher than ARI and %covered of AUTO-HDS and is close to the "winner," OPTICS(AutoCl). The collections of image data sets, ALOI-TS88 and ALOI-PCA, allowed us to perform paired t-tests with respect to ARI, confirming that the observed differences in performance between all pairs of methods is statistically significant at the $\alpha = 0.01$ significance level. This means that the methods are indeed doing different things, and, in particular, that HDBSCAN significantly outperforms the others on these data set collections.

Second, we compare GHDBSCAN(NMRD) and GHDBSCAN(NMRD+PF)

[9]We also tried other values of m_{pts}, r_{shave}, and n_{part}/m_{clSize}, with similar results.

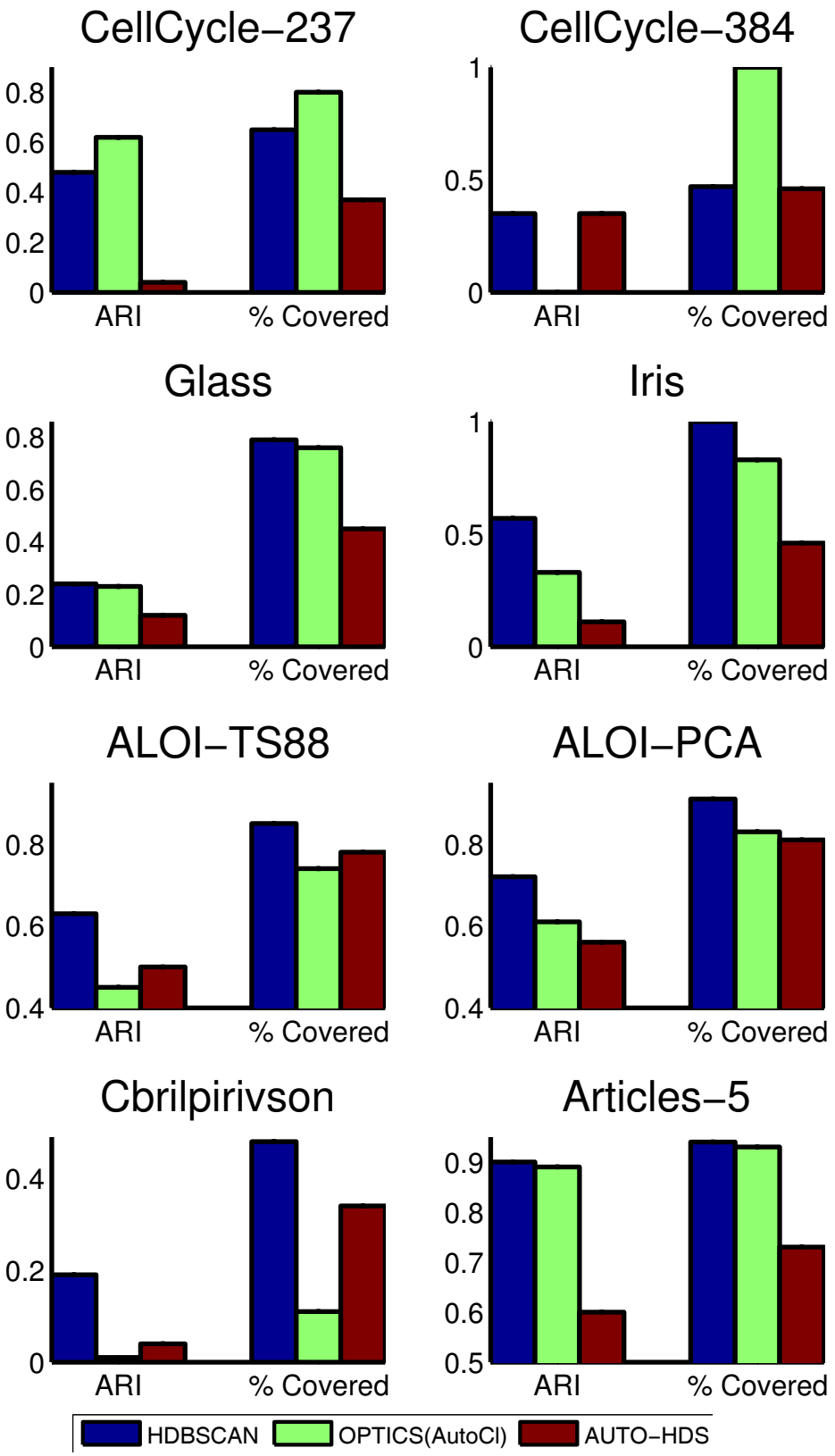


Figure 4.4: ARI results and percentage of the data clustered for algorithms HDBSCAN, OPTICS(AutoCl) and AUTO-HDS.

with HDBSCAN algorithm in Figure 4.5. Note that HDBSCAN outperforms both of the other methods only in one data set, CellCycle-384, where its ARI is slightly better than both of these two methods. GHDBSCAN(NMRD+PF) outperforms HDBSCAN in four out of eight data sets and GHDBSCAN(NMRD) outperforms in CellCycle-237 and two collections of image data sets, ALOI-TS88 and ALOI-PCA, by a large margin, and its results are quite similar to that of HDBSCAN in four data sets (Glass, Iris, Cbrilpirvison and Ariticle-5). We performed paired t-tests with respect to ARI on two collections of image data sets confirming that the large margin differences in performance are statistically significant at the $\alpha = 0.01$ significance level.

4.7 Summary

In this chapter we proposed and discussed in several novel density-based clustering approaches. First, we discussed the AUTO-HDS framework and show that the particular algorithm adapted in the clustering stage of the framework can be replaced so that a user-defined parameter is eliminated and the clustering stage and extraction can be performed in a simpler, faster and more accurate way. Then, we introduced a novel generalized density-based clustering approach by recognizing the two essential components of density-based clustering. We proposed two methods for replacing these components by first proposing a novel, theoretically sound approach to estimating the density of the objects along the path, and second proposing a novel approach to estimating the density of the objects without having any parameters and only based on the distances to all other objects in the data set that improve on HDBSCAN which produces a complete density-based clustering hierarchy representing all possible DBSCAN* solutions for all possible density thresholds.

In an extensive set of experimental evaluations using a wide variety of real-world data sets, we have demonstrated that our density-based cluster-

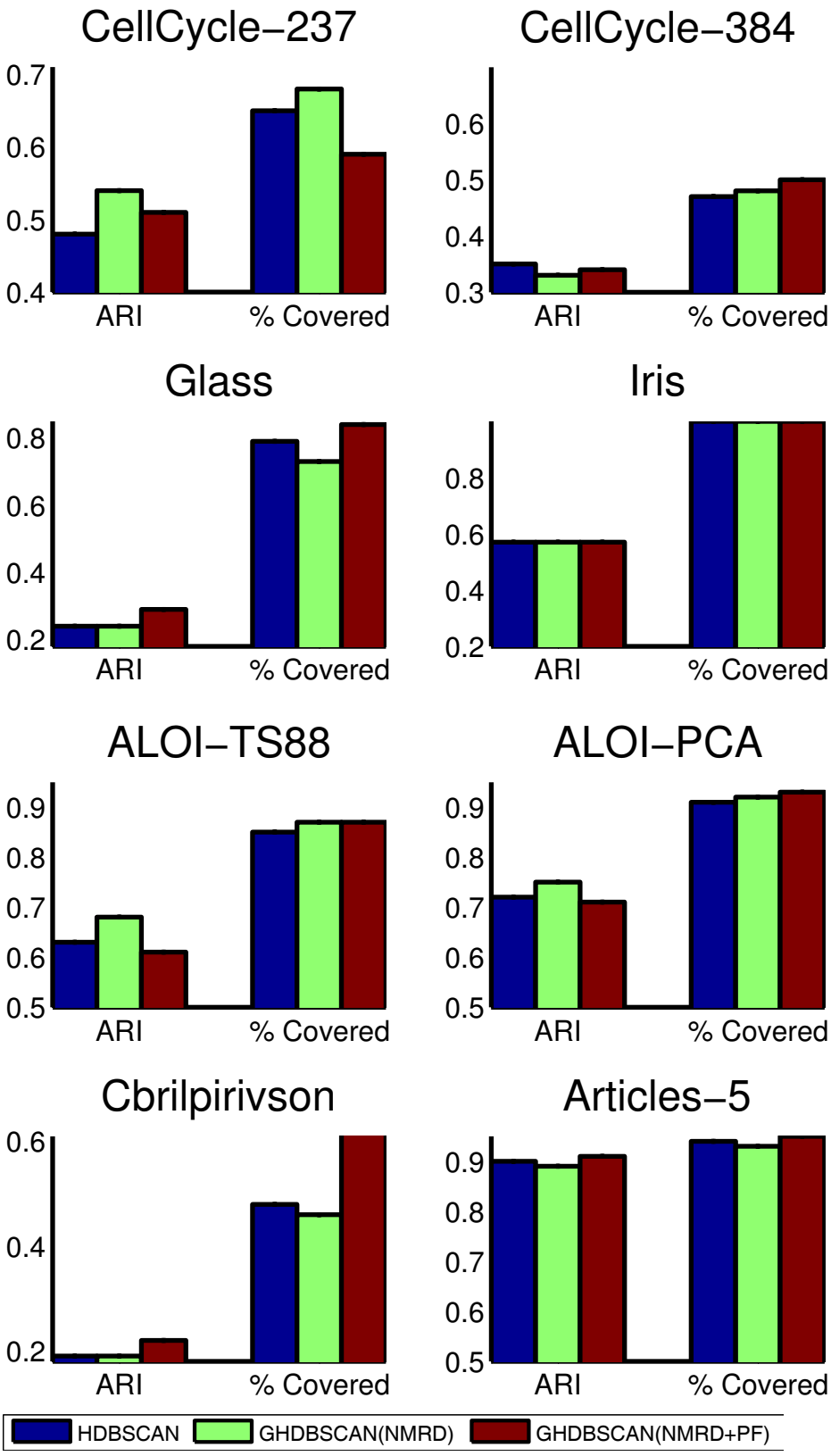


Figure 4.5: ARI results and percentage of the data clustered for algorithms HDBSCAN, GHDBSCAN(NMRD) and GHDBSCAN(NMRD+PF).

ing algorithms perform in general significantly better and more robustly than state-of-the-art methods. There are several interesting challenges for possible future research such as utilizing other density estimation methods in our general framework or to investigate other alternative stability measures. One can also investigate combination of different stability measures that capture different aspects of the quality of the clusters to improve on extraction of flat clustering solution. One can also use a small amount of information about data in the form of constraints in the selection process; one type of integration of semi-supervision to the clustering procedure is discussed in Chapter 5.

Chapter 5

Semi-Supervised Extraction of a Flat Partition from a Hierarchy and Model Selection

5.1 Introduction

In this chapter, in order to show practical application of semi-supervised clustering, we first describe the Framework for Optimal Selection of Clusters framework, called FOSC, which is the semi-supervised and unsupervised extraction of flat partition from a hierarchy (FOSC was proposed by Campello, Moulavi, Zimek and Sander [56] with participation of the author). Then we instantiate that framework in the context of density-based clustering for our proposed GHDBSCAN method, and show that this combination approach outperforms other state-of-the-art semi-supervised and unsupervised clustering algorithms. We then propose a semi-supervised model selection framework, General Semi-Supervised Model Selection (GSS-MS), which works based on a small amount of instance-level constraints and also on our proposed density-based clustering validation measure, DBCV. We use GSS-MSS along with GHDBSCAN to show that combining all the methods that have been proposed in this thesis further

improves the quality of the clustering results. Then we provide a model selection method for semi-supervised clustering algorithm, CVCP, based on a sound cross validation procedure (the framework CVCP was proposed by Pourrajabi and Moulavi et al. [54] with participation of the author).

The remainder of this chapter is organized as follows. In the following section, Section 5.2, we define and formulate the extraction of a flat partitional clustering from a cluster tree as an optimization problem. In Section 5.3 we present a globally optimal solution to this problem along with an algorithm that can efficiently compute the globally optimal solution in the semi-supervised and unsupervised scenarios. In Section 5.4 we discuss the problem of semi-supervised model selection and provide two semi-supervised model selection frameworks, GSS-MS and CVCP. In Section 5.5 we provide an extensive experimental evaluation of our approaches. In Sections 5.5.2 and 5.5.3 particularly, we show that the combination of the approaches that we introduced in this thesis even further improves the quality of the clustering results.

5.2 Semi-Supervised Extraction of a Flat Partition from a Hierarchy Problem Statement

5.2.1 Preliminaries

Let $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ be a data set containing n data objects and $\{\mathbf{C}_1, \cdots, \mathbf{C}_\kappa\}$ be the collection of clusters in a clustering hierarchy of $\mathbf{X}$ from which a flat solution has to be extracted. We represent the clustering hierarchy as a cluster tree in which each node is associated with a particular cluster. Let $\mathbf{C}_1$ be the root of the tree, which represents the all-inclusive "cluster" composed of the entire data set (i.e. $\mathbf{C}_1 = \mathbf{X}$). Without any loss of generality, we consider that the cluster tree is binary, for two main reasons: (i) it is simpler to formalize the problem for binary trees; and (ii) the solution of the problem can be straight-

forwardly generalized to deal with non-binary trees. Formally, the tree is such that each internal node, associated with a given cluster $\mathbf{C}_i$, has two child nodes associated with nested sub-clusters of $\mathbf{C}_i$. These nodes are denoted as $\mathbf{C}_{i_l}$ and $\mathbf{C}_{i_r}$, in reference to the left and right child of $\mathbf{C}_i$, respectively. The leaf nodes of the tree are associated with clusters that do not have sub-clusters.

Problem Overview

Before running into possible complications associated with the different aspects of clustering that will be considered in this chapter, we first provide the reader with a more abstract overview of the problem and outline the basic solution strategy that will be further elaborated in Section 5.3. Let us assume that we are given a cluster tree as described above and a numerical value associated with each cluster, which quantifies a certain property of interest related to the clusters. The fundamental problem we want to solve is to select a collection of clusters from the tree that represents a flat clustering solution and for which a predefined form of aggregation of the corresponding numerical values is maximized. For instance, if the meaningful aggregation for the problem at hand is the sum, then we want to maximize the sum of the numerical values associated with selection of clusters that do not overlap. Possible meanings for such numerical values are not relevant at this point.

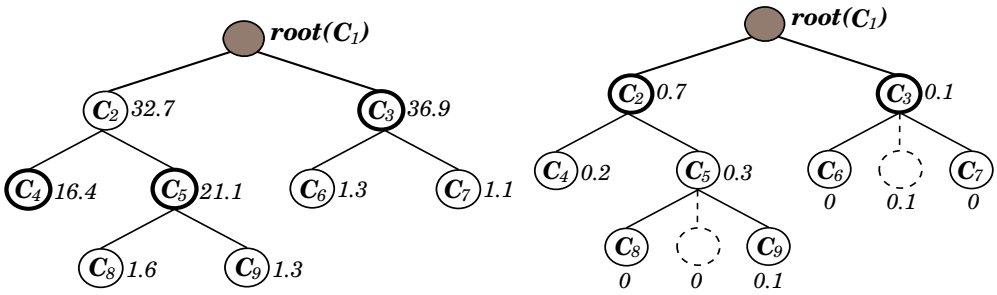
Consider the example cluster tree in Figure 5.1(a). To obtain a flat solution, no other cluster can be selected in the subtree rooted at a given selected cluster. Among the solutions that satisfy this condition, the one composed of $\{\mathbf{C}_3, \mathbf{C}_4, \mathbf{C}_5\}$, is optimal as it maximizes the sum of the values displayed beside the clusters in the figure. To find such a solution efficiently, it is worth noting that the sub-selection of clusters in any subtree represents a sub-problem of the same nature of the original problem (i.e., the one that refers to the complete tree). Based on this observation, a dynamic programming strategy can be applied that incrementally solves sub-problems (subtrees) of increasing sizes,

starting from the leaves and aggregating the intermediate solutions upwards in the tree [56]. At this very high level of abstraction, this strategy is analogous to the one used in error-based pruning of decision-tree classifiers.

In the example of Figure 5.1(a), starting from the deepest leaves, $\mathbf{C}_8$ and $\mathbf{C}_9$ must be discarded as their values sum up to 2.9, which is worse than $\mathbf{C}_5$ (21.1). At the level above, $\mathbf{C}_5$, as the best sub-selection in its subtree, and $\mathbf{C}_4$, as a subtree on its own, are together (37.5) better than $\mathbf{C}_2$ (32.7), which is then discarded. In the other branch, clusters $\mathbf{C}_6$ and $\mathbf{C}_7$ are also discarded as they are together (2.4) worse than $\mathbf{C}_3$ (36.9). In the end, clusters $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$ remain, which is the optimal global solution, with a total value of 74.4. In the following, this strategy will be specialized, extended, and formalized as an algorithm for applications in different contexts of clustering.

Hierarchies with Noise Objects

It is worth remarking that many clustering hierarchies are not only able to represent parent-children relationships involving groups of objects with valid cluster labels, but they can also model the fact that some objects in a given parent cluster may become noise before this cluster is split into its children (from a top-down perspective on the cluster tree), thus no longer belonging to any cluster below the respective hierarchical level. Noise objects are a natural consequence when using density-based hierarchical algorithms [e.g. 30, 28, 25], as such algorithms produce hierarchies in which the hierarchical levels are associated with different density thresholds, and objects whose density is below the threshold are deemed noise at the corresponding levels. The concept of noise, however, can be easily extended to other types of hierarchies, for instance, by using a *minimum cluster size*, m_{clSize}, commonly used in real cluster analysis (see, e.g., the notion of a *particle* in AUTO-HDS [25]) and that we discussed it in Chapter 4 and here we extend our discussion in the context of semi-supervised clustering in which we consider as in [56] the concept of virtual



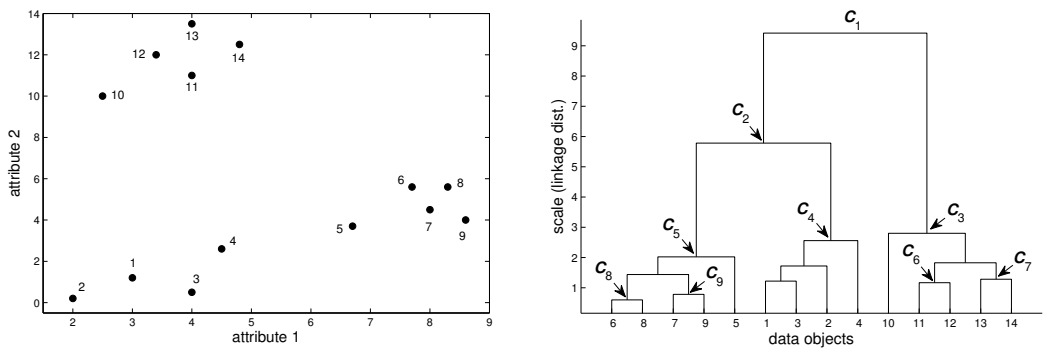
(a) unsupervised case (b) semi-supervised case 1

Figure 5.1: Simplified cluster trees of the hierarchy in Table 5.1. Figures beside the nodes denote: (a) cluster quality (stability); (b) fractions of constraint satisfactions. Boldfaced solid nodes indicate optimally selected clusters. Dashed circles in (b) are virtual nodes representing noise objects.

nodes for the objects that become noise along the hierarchical levels.

As an example, Figure 5.2 shows a simple data set and its average-linkage dendrogram, where there are 27 clusters (including 14 singletons), 9 of which have been highlighted. Table 5.1 shows the idea described above for the dendrogram in Figure 5.2(b), with $m_{clSize} = 2$. When a single object, as a subcomponent with fewer than m_{clSize} objects, is disconnected from a cluster top-down along the hierarchy (which reads from left to right along the table), the original cluster is regarded only as having decreased in size, so it keeps its original label. This procedure reduces the 27 clusters in the dendrogram to 9 more prominent ones (labeled "1" to "9" in Table 5.1)[1], which however may exhibit different configurations along different hierarchical levels (e.g., cluster "4" with and without objects $\mathbf{x}_4$ and $\mathbf{x}_2$). In our framework, such different configurations can be modeled as parent-child nodes in the tree of candidate clusters, which in this case would not be strictly binary. Optionally, the cluster tree and the corresponding cluster extraction problem can be simplified by considering as significant hierarchical levels only those in which a cluster is truly split, giving rise to new sub-clusters. The clusters at such hierarchical levels are those highlighted in Figure 5.2(b) ($\mathbf{C}_1, \cdots, \mathbf{C}_9$) and refer back to Hartigan's concept of

[1]Note that such a reduction may be even more noticeable for higher values of m_{clSize}.



(a) data set (b) dendrogram

Figure 5.2: (a) Sample data set and (b) its average-linkage dendrogram.

rigid clusters [3], which is adopted in this chapter and has also been explored, e.g., by [85, 25]. The corresponding cluster tree is displayed in Figure 5.1(a).

$\mathbf{x}_1$	1	2	4	4	4	4	4	4	4	4	0	0	0	0
$\mathbf{x}_2$	1	2	4	4	4	4	4	0	0	0	0	0	0	0
$\mathbf{x}_3$	1	2	4	4	4	4	4	4	4	4	0	0	0	0
$\mathbf{x}_4$	1	2	4	4	0	0	0	0	0	0	0	0	0	0
$\mathbf{x}_5$	1	2	5	5	5	0	0	0	0	0	0	0	0	0
$\mathbf{x}_6$	1	2	5	5	5	5	5	5	8	8	8	8	8	0
$\mathbf{x}_7$	1	2	5	5	5	5	5	5	9	9	9	9	0	0
$\mathbf{x}_8$	1	2	5	5	5	5	5	5	8	8	8	8	8	0
$\mathbf{x}_9$	1	2	5	5	5	5	5	5	9	9	9	9	0	0
$\mathbf{x}_{10}$	1	3	3	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{x}_{11}$	1	3	3	3	3	3	6	6	6	6	6	0	0	0
$\mathbf{x}_{12}$	1	3	3	3	3	3	6	6	6	6	6	0	0	0
$\mathbf{x}_{13}$	1	3	3	3	3	3	7	7	7	0	0	0	0	0
$\mathbf{x}_{14}$	1	3	3	3	3	3	7	7	7	0	0	0	0	0
Scale	9.42	5.78	2.8	2.56	2.02	1.83	1.72	1.44	1.28	1.22	1.17	0.78	0.6	0

Table 5.1: Hierarchy for the data in Figure 5.2(a) with $m_{clSize} = 2$. Higher (lower) hierarchical levels are on the left (right). Scale values (bottom bar) are the average-linkage distances. The remaining values are labels: a non-0 value i in the jth row means that object $\mathbf{x}_j$ belongs to cluster $\mathbf{C}_i$ at the corresponding level, whereas a 0 value denotes noise.

Note that the cluster tree, as represented in Figure 5.1(a), does not explicitly account for the fact that object $\mathbf{x}_{10}$ of cluster $\mathbf{C}_3$ is not part of either of $\mathbf{C}_3$'s children, namely, $\mathbf{C}_6$ ($\mathbf{C}_{3_l}$) and $\mathbf{C}_7$ ($\mathbf{C}_{3_r}$), as $\mathbf{x}_{10}$ is already deemed noise at the level at which $\mathbf{C}_6$ and $\mathbf{C}_7$ appear as clusters and below. The same holds true with respect to object $\mathbf{x}_5$ and clusters $\mathbf{C}_5$, $\mathbf{C}_8$ ($\mathbf{C}_{5_l}$), and $\mathbf{C}_9$ ($\mathbf{C}_{5_r}$). We will see later that these objects can affect the optimal extraction of clusters

in the semi-supervised case. For this reason and for the sake of generality, we assume that the information about objects that are part of a cluster yet not part of its sub-clusters (if any) is available in the hierarchy to be processed. This information can be represented in the cluster tree by means of *"virtual" nodes*, like the ones displayed in dashed lines in Figure 5.1(b).

To simplify the formulation, we assume that every internal node of the cluster tree has one virtual child node, even though only some of them are actually associated with noise objects and remaining ones store no information at all.[2] The virtual child node of a cluster $\mathbf{C}_i$ in the cluster tree will be denoted hereafter as $\mathbf{C}_i^{\emptyset}$, no matter whether it actually stores information or not.

5.2.2 Problem Formulation as a Programming Task

Assumed that we are given a cluster tree of a data set $\mathbf{X}$ and also a collection of should-link and should-not-link constraints that being satisfied or violated to different degrees by the clusters in the cluster tree (when such a collection of constraints is empty will be discussed later as a unsupervised extraction case).

Our primary objective is to extract a flat solution from the cluster tree that is globally optimal with respect to the constraints, that is, to extract a collection of clusters that altogether maximize the number of constraint satisfactions computed over the whole data set $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$. Unsupervised measures of cluster quality such as bounded excess of mass that we discussed in Chapter 4 or the one described in Section 5.2.3 will in principle be considered as a secondary objective. The primary objective function can be written as:

$$J = \frac{1}{2n_c} \sum_{j=1}^{n} \gamma(\mathbf{x}_j) \tag{5.1}$$

where n_c is the number of available should- and should-not-link constraints and $\gamma(\mathbf{x}_j)$ is the number of constraints involving object $\mathbf{x}_j$ that are satisfied (not

[2] In the example of Figure 5.1(b), these nodes would be virtual children of $\mathbf{C}_1$ and $\mathbf{C}_2$, which have been omitted for the sake of clarity.

violated). The scaling constant ½ is due to the fact that a single constraint involves a pair of objects and, as such, it is taken into account twice in the sum. Term n_c in the denominator is used to normalize J in the unit interval. Thus, J is the fraction of constraints that are satisfied or, equivalently, the complement of the fraction of constraints that are violated. Evidently, maximizing J is equivalent to minimizing the number of constraint violations.

The candidate clusters are all the clusters in the cluster tree. However, in the following, we exclude $\mathbf{C}_1$ from the collection of eligible clusters, which is therefore $\mathbf{E} = \{\mathbf{C}_2, \cdots, \mathbf{C}_\kappa\}$ which does not change the essence of the problem.

When selecting clusters for a flat clustering solution, nested clusters in the tree must be mutually exclusive, i.e., each object can only be assigned a single label (possibly null, representing noise). This condition can be formulated as a set of constraints for the problem of maximizing J in Equation (5.1). By noticing that terms $\gamma(\cdot)$ in Equation (5.1) are a function of the clusters that will be selected, the optimization problem can be written as:

$$\begin{aligned} &\underset{\delta_2, \ldots, \delta_\kappa}{\text{maximize}} \quad J \text{ in Eq. (5.1)} \\ &\text{subject to} \quad \begin{cases} \delta_i \in \{0,1\}, \quad i = 2, \cdots, \kappa \\ \sum_{j \in \mathbf{I}_h} \delta_j = 1, \quad \forall h \text{ such that } \mathbf{C}_h \text{ is a leaf cluster} \end{cases} \end{aligned} \tag{5.2}$$

where δ_i $(i = 2, \cdots, \kappa)$ is an indicator that denotes whether cluster $\mathbf{C}_i$ is included in the flat solution ($\delta_i = 1$) or not ($\delta_i = 0$) and $\mathbf{I}_h$ is the set of indexes for those clusters on the path from leaf cluster $\mathbf{C}_h$ (included) to the root (excluded). The constraints in Problem (5.2) enforce that exactly one cluster will be selected along any branch from the root to a leaf. Accordingly, they ensure that each data object will be either assigned to a single cluster or labeled as noise for not being part of any selected cluster. It is worth remarking that, since a noise object is, by definition, not clustered with any other object,

then a should-not-link constraint involving one or both objects labeled as noise is deemed satisfied, while a should-link constraint is deemed violated.

In problem (5.2) if the objects of a parent cluster are not involved in any constraints, then it is not possible to discriminate between this cluster and its sub-clusters in terms of constraint satisfiability. This is a particular case of a tie between nested (and therefore "competing") clusters, which can also occur in more general scenarios, even for clusters involved in constraints (as we will see in an example in Section 5.3.1). In these cases, there will be multiple valid cluster selections with the same globally optimal value for the objective function in Problem (5.2). In practice, this means that, one might return as a result a clustering that is only partially flat, possibly including subtrees of nested clusters whose selection is undecidable. If the user does require a truly flat solution, the ties can be decided arbitrarily or a secondary objective must be considered to solve such subtrees. As a secondary objective, we consider an overall aggregation of the individual qualities of the composing clusters. Specifically, let $S(\mathbf{C}_i)$ be a given unsupervised measure of quality for cluster $\mathbf{C}_i$. Then, an overall aggregation of the qualities of those clusters selected from the cluster tree to compose a flat solution can be written as:

$$J = \sum_{i=2}^{\kappa} \delta_i \, S(\mathbf{C}_i) \tag{5.3}$$

Using J given by Equation (5.3) in lieu of Equation (5.1) allows for resolving ties in subtrees. By doing so, when the whole tree is indistinguishable with respect to constraint satisfiability, because it has been successfully constructed by enforcing that all clusters satisfy all the constraints or because there are no constraints at all, then the problem falls into the particular, unsupervised case and clusters are purely extracted based on their qualities $S(\mathbf{C}_i)$. For example, Bounded excess of mass proposed in chapter 4 or the measure that we describe in Section 5.2.3 can be used as an unsupervised measure of quality.

5.2.3 Unsupervised Measures of Cluster Quality

A suitable unsupervised measure of cluster quality depends on the characteristics of the cluster tree at hand. Cluster trees can be of varied natures and be produced by algorithms based on different paradigms. We have discussed two other appropriate measures for cluster trees of all possible types in Chapter 4. Here, we will describe a measure that can be applied to any algorithm that produces hierarchies of the same nature as that illustrated in Table 5.1. This includes, for instance, density-based methods [e.g. 28, 49] and all the methods that produce traditional dendrograms as a result [e.g., see 1, 40], and to which the hierarchy simplification technique based on minimum cluster size and noise labels described in Section 5.2.1 can be applied.

The measure considered here [56] is based on the premise that more prominent clusters will survive longer after they appear, which is the rationale behind the definition of *cluster lifetime* as a measure of cluster stability in classic cluster analysis [1]. The lifetime of a given cluster in a dendrogram is defined as the length of the dendrogram scale along the hierarchical levels that cluster exists. For hierarchies like the one in Table 5.1, this concept can be adjusted to account for different lifetimes of the objects of a cluster. The adjustment we consider here, which will be used in all related experiments reported in Section 5.5, is summing up the lifetimes of the objects belonging to the cluster.

For example, from a bottom-up perspective of the hierarchy in Table 5.1, cluster $\mathbf{C}_4$ appears with 2 objects at level 1.22 of the scale and disappears when it is merged with $\mathbf{C}_5$, giving rise to $\mathbf{C}_2$ at level 5.78. In between these levels, a third and fourth objects join the cluster at levels 1.72 and 2.56 respectively. Then, the stability of this cluster is given by $S(\mathbf{C}_4) = 2*(5.78-1.22)+(5.78-1.72)+(5.78-2.56) = 16.4$. The values for the other clusters can be computed analogously and are displayed beside the nodes of the tree in Figure 5.1(a).

Cluster stability should have two important properties that are required for a quality measure to qualify for use in the framework proposed in this

chapter. First, it is *local* in the sense that it can be computed for each cluster independently of the clusters that will be selected to compose the final flat solution, which are obviously unknown beforehand. Second, it is *additive* with respect to the objects that compose the cluster. This means that adding the values corresponding to the clusters to be selected is meaningful and compatible with the aggregation operator used in the objective function of the optimization problem, which is the sum operator in the case of Equation (5.3).

5.3 Optimal Cluster Extraction

5.3.1 Problem Solution and Algorithm

Recalling that $\mathbf{E} = \{\mathbf{C}_2, \cdots, \mathbf{C}_\kappa\}$ is the whole collection of eligible clusters, let $\mathbf{F} \subseteq \mathbf{E}$ be any candidate flat clustering solution satisfying the structural constraints of Problem (5.2). Furthermore, let $\mathbf{X}_L \subseteq \mathbf{X}$ be the subset of data objects that have a non-null label in such a candidate flat solution, i.e., $\mathbf{X}_L = \{\mathbf{x}_j \mid \exists\, \mathbf{C}_i \in \mathbf{F} :\ \mathbf{x}_j \in \mathbf{C}_i\}$. Finally, let $\overline{\mathbf{X}}_L$ be the subset of data objects that have a null label (noise) in this candidate solution, i.e., $\overline{\mathbf{X}}_L = \mathbf{X} - \mathbf{X}_L$. Then, the objective function in Equation (5.1) can be rewritten as:

$$\begin{aligned} J &= \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \mathbf{X}_L} \gamma(\mathbf{x}_j) \ + \ \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \overline{\mathbf{X}}_L} \gamma(\mathbf{x}_j) \\ &= \sum_{i=2}^{\kappa} \delta_i \, \Gamma(\mathbf{C}_i) \ + \ \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \overline{\mathbf{X}}_L} \gamma(\mathbf{x}_j) \end{aligned} \tag{5.4}$$

where $\Gamma(\mathbf{C}_i) = \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \mathbf{C}_i} \gamma(\mathbf{x}_j)$ is the fraction of constraint satisfactions involving the objects of cluster $\mathbf{C}_i$ (which is zero for clusters whose objects are not involved in constraints). For example, let us consider three should-link constraints, $(\mathbf{x}_1, \mathbf{x}_6)$, $(\mathbf{x}_2, \mathbf{x}_5)$, and $(\mathbf{x}_5, \mathbf{x}_8)$, as well as two should-not-link constraints, $(\mathbf{x}_4, \mathbf{x}_9)$ and $(\mathbf{x}_3, \mathbf{x}_{10})$, for the data set in Figure 5.2(a). Given the clusters in Table 5.1, we can compute $\Gamma(\mathbf{C}_9) = \frac{1}{2\times 5} \left(\gamma(\mathbf{x}_7) + \gamma(\mathbf{x}_9)\right) = \frac{1}{10}(0 + 1) =$

0.1. The values for the other clusters can be computed analogously and are displayed beside the (non-virtual) nodes of the tree in Figure 5.1(b).

Similarly to the first term of Equation (5.4), we can also decompose its second term, which refers to objects labeled as noise, into a sum of fractions, each of which is associated with a virtual node of the cluster tree. Every object labeled as noise in a flat solution extracted from the tree is associated with one of the virtual nodes. Hence, we can rewrite Equation (5.4) as:[3]

$$J \;=\; \sum_{i=2}^{\kappa} \delta_i \, \Gamma(\mathbf{C}_i) \;+\; \sum_{l=1}^{m} \varphi_l \, \Gamma(\mathbf{C}_l^{\emptyset}) \tag{5.5}$$

where m is the number of virtual nodes in the tree ($m = \frac{\kappa-1}{2}$ for binary trees, i.e., the number of internal nodes), φ_l is an indicator that denotes whether the objects associated with the virtual node $\mathbf{C}_l^{\emptyset}$ (the virtual child of $\mathbf{C}_l$) are labeled as noise in the flat solution ($\varphi_l = 1$) or not ($\varphi_l = 0$), and $\Gamma(\mathbf{C}_l^{\emptyset}) = \frac{1}{2n_c} \sum_{\mathbf{x}_j \in \mathbf{C}_l^{\emptyset}} \gamma(\mathbf{x}_j)$ is the fraction of constraint satisfactions involving the noise objects associated with $\mathbf{C}_l^{\emptyset}$. For instance, considering the same set of constraints as in the example above (and recalling that a should-not-link constraint involving one or both objects labeled as noise is deemed satisfied, while a should-link constraint is deemed violated), one has $\Gamma(\mathbf{C}_3^{\emptyset}) = \frac{1}{10}\gamma(\mathbf{x}_{10}) = 0.1$ and $\Gamma(\mathbf{C}_5^{\emptyset}) = \frac{1}{10}\gamma(\mathbf{x}_5) = 0$, which are the values displayed beside the respective virtual nodes in Figure 5.1(b). For virtual nodes not associated with any noise object at all (those omitted in Figure 5.1(b)), $\Gamma(\mathbf{C}_l^{\emptyset})$ is defined as zero.

Notice that the objects associated with a virtual node $\mathbf{C}_l^{\emptyset}$ will end up being labeled as noise ($\varphi_l = 1$) if and only if any descendant of $\mathbf{C}_l$ is included into the final solution. This means that φ_l ($l = 1, \cdots, m$) is a function of the original

[3]Notice that Γ has the same properties as S discussed in Sections 4.4.5 and 5.2.3: it can be computed locally for each node and it is additive with respect to the objects in the node, so it is compatible with the aggregation operator (sum) used in the objective function in Equations (5.1) and (5.5).

decision variables δ_i of Problem (5.2), which can then be rewritten as:

$$\begin{array}{ll} \underset{\delta_2,\,\ldots,\,\delta_\kappa}{\text{maximize}} & J \text{ in Eq. (5.5)} \\ \text{subject to} & \left\{ \begin{array}{l} \text{the same constraints as in (5.2)} \\ \varphi_l = \left\{ \begin{array}{ll} 1 & \text{if } l \in \mathbf{A} \\ 0 & \text{otherwise} \end{array} \right. \end{array} \right. \end{array} \tag{5.6}$$

where $\mathbf{A} = \{l \mid \exists\, i \neq l :\ \delta_i = 1 \ \wedge\ \mathbf{C}_l \text{ is an ancestor of } \mathbf{C}_i\}$ are the indexes of clusters that are ancestors of any cluster $\mathbf{C}_i$ to be selected as part of the final flat solution ($\delta_i = 1$). In brief, Problem (5.6) formulates the maximization of the constraint satisfactions, written as a sum of fractions associated with the original and virtual nodes of the tree, provided that: (a) clusters located along a common branch of the tree (nested clusters) must be mutually exclusive; and (b) if a cluster is selected, then the objects associated with all virtual nodes of this cluster's ancestors must be labeled as noise.

To solve Problem (5.6), we process every cluster node except the root, starting from the leaves (bottom-up), deciding at each node $\mathbf{C}_i$ whether $\mathbf{C}_i$ or the best-so-far selection of nodes in $\mathbf{C}_i$'s subtrees should be selected. To be able to make this decision locally at $\mathbf{C}_i$, we propagate and update the sum of constraint satisfactions $\hat{\Gamma}(\mathbf{C}_i)$ of nodes provisionally selected in the subtree rooted at $\mathbf{C}_i$ in the following, recursive way:

$$\hat{\Gamma}(\mathbf{C}_i) \ = \ \left\{ \begin{array}{ll} \Gamma(\mathbf{C}_i), & \text{if } \mathbf{C}_i \text{ is a (non-virtual) leaf node} \\ \max\{\Gamma(\mathbf{C}_i), \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})\}, & \\ & \text{if } \mathbf{C}_i \text{ is an internal node} \end{array} \right. \tag{5.7}$$

Algorithm 4 gives the pseudo-code [56]. Note that Step 3.3 does not specify a particular criterion to resolve ties. Different decisions when resolving a tie will lead to different flat solutions, but the value of the primary objective func-
where $\mathbf{C}_{i_l}$ and $\mathbf{C}_{i_r}$ are the sub-clusters of $\mathbf{C}_i$, while $\mathbf{C}_i^{\emptyset}$ is its virtual child.

Algorithm 4: Solution to Problem (5.6)

Input: Cluster tree and fraction of constraint satisfactions for each node, $\Gamma(\cdot)$.

1. Initialize $\delta_2 = \cdots = \delta_\kappa = 1$.
2. Initialize the $\hat{\Gamma}$ values of the leaf nodes as $\hat{\Gamma}(\mathbf{C}_h) = \Gamma(\mathbf{C}_h)$.
3. For every internal node $\mathbf{C}_i$ (except the root $\mathbf{C}_1$), starting from the deepest levels and going up the tree, do:
   - 3.1 If $\Gamma(\mathbf{C}_i) < \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$:
     Set $\hat{\Gamma}(\mathbf{C}_i) = \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$ and remove $\mathbf{C}_i$ from the list of candidate clusters by setting $\delta_i = 0$.
   - 3.2 Else If $\Gamma(\mathbf{C}_i) > \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$:
     Set $\hat{\Gamma}(\mathbf{C}_i) = \Gamma(\mathbf{C}_i)$ and remove from the list of candidates (by setting their $\delta_{(\cdot)}$ values to 0) all the clusters in $\mathbf{C}_i$'s subtrees.
   - 3.3 Otherwise, resolve the tie by performing either the actions in Step 3.1 or those in Step 3.2, according to a secondary criterion.
4. **Return**: $\delta_2, \cdots, \delta_\kappa$.

tion in Equation (5.5) will necessarily be the same for any of these solutions. In other words, the choice of the secondary objective does not affect the optimality of the final solution with respect to the primary objective. Therefore, in order to provide a single optimal solution to Problem (5.6), ties can be resolved arbitrarily. However, a more justified approach is to consider a suitable measure of cluster quality as a secondary objective. Specifically, a tie can be decided analogously to Steps 3.1 — 3.2, but replacing $\Gamma(\cdot)$ with an unsupervised measure of cluster quality $S(\cdot)$ and considering it to be null for virtual nodes ($S(\mathbf{C}_i^{\emptyset}) = 0$). Under these conditions, J in Equation (5.5) becomes structurally equivalent to the unsupervised objective function given by Equation (5.3). In this case, the solution of a sub-problem that refers to a subtree involved in a tie will therefore be optimal with respect to this secondary, unsupervised objective function. The detailed pseudo-code is given in Algorithm 5.

When there are no constraints, Algorithm 5 processes the whole tree in an unsupervised way, as in the example displayed in Figure 5.1(a). We have shown in Section 5.2.1 that the optimal solution in that example consists of clusters $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$, with J in Equation (5.3) equal to 74.4.

When the collection of five constraints is considered (Figure 5.1(b)), the

Algorithm 5: Solution to Problem (5.6) with quality-based tiebreaker

Input: Cluster tree, the fraction of constraint satisfactions for each node, $\Gamma(\cdot)$, and the unsupervised quality of each non-virtual node, $S(\cdot)$ (null for virtual nodes).

1. Initialize $\delta_2 = \cdots = \delta_\kappa = 1$.
2. Initialize $\hat{\Gamma}$ and $\hat{S}$ for the leaf nodes as $\hat{\Gamma}(\mathbf{C}_h) = \Gamma(\mathbf{C}_h)$ and $\hat{S}(\mathbf{C}_h) = S(\mathbf{C}_h)$.
3. For every internal node $\mathbf{C}_i$ (except the root $\mathbf{C}_1$), starting from the deepest levels and going up the tree, do:
   - 3.1 If $\Gamma(\mathbf{C}_i) < \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$:
     Set $\hat{\Gamma}(\mathbf{C}_i) = \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$ and remove $\mathbf{C}_i$ from the list of candidate clusters by setting $\delta_i = 0$. Also, set $\hat{S}(\mathbf{C}_i) = \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$.
   - 3.2 Else If $\Gamma(\mathbf{C}_i) > \hat{\Gamma}(\mathbf{C}_{i_l}) + \hat{\Gamma}(\mathbf{C}_{i_r}) + \Gamma(\mathbf{C}_i^{\emptyset})$:
     Set $\hat{\Gamma}(\mathbf{C}_i) = \Gamma(\mathbf{C}_i)$ and remove from the list of candidates (by setting their $\delta_{(\cdot)}$ values to 0) all the clusters in $\mathbf{C}_i$'s subtrees. Also, set $\hat{S}(\mathbf{C}_i) = S(\mathbf{C}_i)$.
   - 3.3 Otherwise, resolve the tie in an unsupervised manner:
     3.3.1 If $S(\mathbf{C}_i) < \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$, then do the same as if the condition in Step 3.1 had been satisfied.
     3.3.2 If $S(\mathbf{C}_i) >= \hat{S}(\mathbf{C}_{i_l}) + \hat{S}(\mathbf{C}_{i_r})$, then do the same as if the condition in Step 3.2 had been satisfied.
4. **Return**: $\delta_2, \cdots, \delta_\kappa$.

solution changes to $\mathbf{C}_2$ and $\mathbf{C}_3$, with J in Equation (5.5) equal to 0.8. In details, $\mathbf{C}_8$ and $\mathbf{C}_9$ are discarded as they are together (along with $\mathbf{C}_5^{\emptyset}$) worse than $\mathbf{C}_5$. At the level above, the pair $\mathbf{C}_4$ and $\mathbf{C}_5$ is also discarded as $\mathbf{C}_2$ is better. In the other subtree of the root, there is constraint satisfiability tie: either $\mathbf{C}_3$ or its children ($\mathbf{C}_6$, $\mathbf{C}_7$, and $\mathbf{C}_3^{\emptyset}$) would add the same amount of 0.1 to the objective function, but $\mathbf{C}_3$ is kept in this case because its unsupervised quality (36.9) is higher than that of $\mathbf{C}_6$ and $\mathbf{C}_7$ together (2.4), which are then discarded. In what concerns the secondary objective in Equation (5.3), this sub-selection is optimal for the subtree rooted at $\mathbf{C}_3$.

5.3.2 Efficient Implementation and Complexity Analysis

Note that Step 3.2 of Algorithm 4 (or Algorithm 5) can be implemented in a more efficient way by not setting $\delta_{(\cdot)}$ values to 0 for discarded clusters down

in the subtrees. Instead, in a simple post-processing procedure, the tree can be traversed top-down in order to find, for each branch, the shallowest cluster that has not been discarded ($\delta_{(\cdot)} = 1$) by Step 3.1. This way, the solution can be found with two traversals of the tree, one bottom-up and another one top-down. This means that the complexity of the algorithm, is $O(\kappa)$, i.e., *it is linear with respect to the number of nodes (or candidate clusters) in the tree, both in terms of running time and memory space.*

It is worth noticing that, if the cluster tree results from a hierarchy simplification procedure like the one described in Section 5.2.1, κ is typically much smaller than the number of data objects ($\kappa \ll n$). It is theoretically possible, however, that a cluster split is observed at each of the n hierarchical levels, leading to the worst case in which $\kappa = 2n - 1$, i.e., κ is $O(n)$.

The pre-computation of the input values $\Gamma(\cdot)$ and $S(\cdot)$ depends on the nature of the hierarchy. $S(\cdot)$ also depends on the particular cluster quality measure to be adopted. For the type of hierarchies in the examples and experiments provided in this chapter, which are dendrogram-like hierarchies possibly modeling noise, it follows that: (i) it is straightforward to compute $\Gamma(\cdot)$ associated with both the original and virtual nodes of the tree with a single pass through the hierarchical levels for each constraint, checking the labels of the pair of objects involved in that constraint. So, the complexity to compute $\Gamma(\cdot)$ for all nodes of the tree is $O(n_c\, n)$, where n_c is the number of constraints; and (ii) similarly, it is also straightforward to compute $S(\cdot)$ for all cluster nodes of the tree by means of a single bottom-up screening of the hierarchy, which is $O(n^2)$ (i.e., no greater than the complexity of computing the clustering hierarchy itself). In the Section we describe our framework for semi-supervised model selection.

5.4 Semi-Supervised Model Selection

Typical clustering algorithms will find different results depending on input parameters. Relative cluster validity criteria [32] could help to distinguish

those parameters that fit better to the data set at hand from those that are not a good fit. However, this approach has few drawbacks. Using relative criteria gives only a relative comparison of solutions, and the selection of the validity criterion is a highly subjective choice. In the presence of additional criteria such as partly labeled data or some constraints such as instance-level should-link and should-not-link constraints, the best model for the task at hand can be selected in a much more objective manner.

Given that many clustering algorithms are parameter dependent, our goal is to provide the basis for selecting the best of a set of possible models. In the following we discuss two semi-supervised approaches for model selection.

5.4.1 General Semi-Supervised Model Selection GSS-MS

Let's consider that we have a set of instance-level, should-link and should-not-link constraints. We run a specific clustering algorithm with parameter p by setting n different values to its parameter to produce the set of clustering partitions $\{\mathbf{C}_{p_1}, \cdots, \mathbf{C}_{p_n}\}$.

Our primary goal is to find a clustering solution that maximizes the number of constraint satisfactions (see Equation 5.1). In our framework, the results of a clustering algorithm will be evaluated based on the fraction of the constraint that are satisfied for the given set of parameters. Then the partition with the maximum fraction of constraint satisfactions will be reported.

If there is more than one partition with the maximum number of constraint satisfaction among given $\{\mathbf{C}_{p_1}, \cdots, \mathbf{C}_{p_n}\}$ clustering partitions, we then consider as a secondary objective another semi-supervised measure of cluster quality (e.g., number of clusters in each partition) or an unsupervised measure of cluster quality such as the value of the DBCV cluster validation measure that we proposed in Chapter 3. Algorithm 6 gives the pseudo-code for this

Algorithm 6: GSS-MS main steps

Input: The data set **X**, range of parameter values.

1. Run a general clustering algorithm with parameter value p to generate a partition, in the case of semi-supervised clustering with the given set of constraints.
2. For the generated partition, determine the fraction of the constrain satisfied and set this value as the quality of the partition.
3. Repeat **(Step 1)** and **(Step 2)** for a different parameter setting.
4. Select the partition with the highest quality value. If more than one partition has the highest quality value:
    4.1 Resolve ties according to a secondary criterion.

framework. In Step 4.1 any particular criterion can be used to resolve ties that can give different clustering solutions. However, the number of the constraint satisfaction for these different solutions will be the same. It means that the choice of the secondary criterion will not affect the amount of constraint satisfactions of the final solution.

However, this model selection approach can also be used with some semi-supervised clustering algorithms (such as FOSC) that treat constraints as soft constraints, in the case of semi-supervised clustering algorithms it is crucial that the information used in the semi-supervised clustering process not be used in the error estimation of the final clustering solution. We discuss this issue and our approach to semi-supervised model selection, which has been customized for semi-supervised clustering, further in Section 5.4.2.

5.4.2 Cross-Validation for Finding Clustering Parameters, CVCP

The following framework describes our second model selection approach.

step 1: Determine the quality of a parameter value p for a semi-supervised clustering algorithm using n-fold cross-validation by treating the generated partition as a classifier for constraints. A single step in the n-fold cross-validation is illustrated in Figure 5.3.

step 2: repeat **(step 1)** for different parameter settings

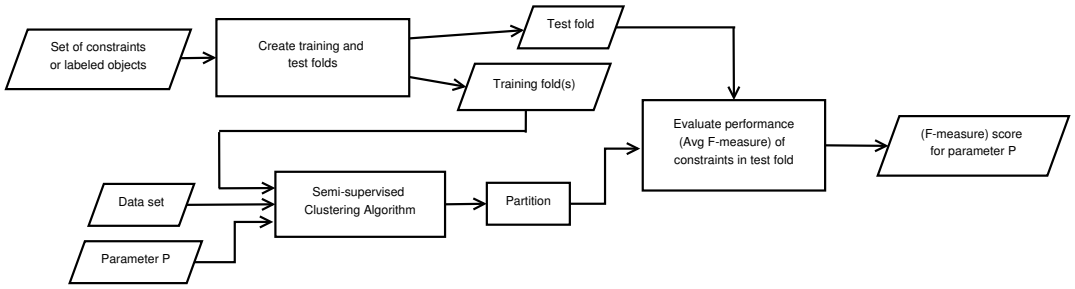


Figure 5.3: Illustration of a single step in an n-fold cross validation to determine the quality score of a parameter value p in step 1 of our framework. This step is repeated n times and the average score for p is returned as p's quality.

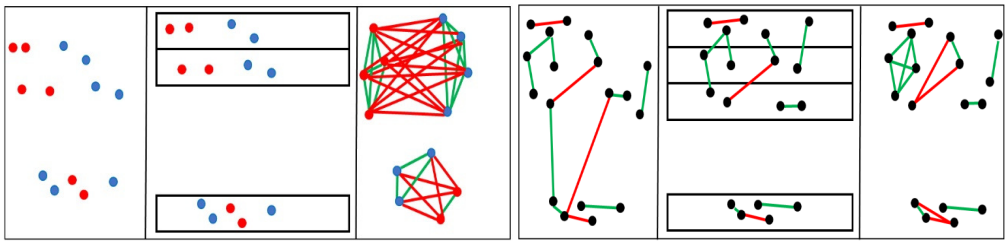
step 3: select the parameter p^* with the highest score

step 4: run the semi-supervised clustering algorithm with parameter value p^* using all available information (labels or constraints) given as input to the clustering algorithm.

The crucial, non-trivial questions for this general framework are how to evaluate **(step 1)** and how to compare **(step 3)** the performance of different models. The question of what constitutes appropriate evaluation in the context of semi-supervised clustering involves several different issues.

First, as we described above it is crucial to *not* use the same information (e.g., labels or constraints) twice in both the learning process (running the clustering algorithm) and in the estimation of the classification error of the learned clustering model. Otherwise, the classification error is likely to be underestimated. We discuss this problem and a solution in Section 5.4.3

Second, we will have to elaborate on how to actually estimate the classification error. For measuring and comparing the performance quantitatively, we will transform the semi-supervised clustering problem to a classification problem over the constraints — which are originally available or that have been extracted from labels — and then use the well-established F-measure. We provide further details on this step in Section 5.4.4.



(a) Scenario I: Labeled objects are provided. (b) Scenario II: Constraints are provided.

Figure 5.4: Objects are distributed in *n - 1* training folds and *1* test fold. (a) Constraints are derived from the labeled objects in *n - 1* folds for the training set and from 1 remaining fold for test set. (b) The transitive closure of constraints are computed for all objects in *n - 1* training folds for training set and for the objects in the test fold for the test set.

Finally, we explain the selection of the best model, based on the previous steps, in Section 5.4.5.

5.4.3 Ensuring Independence Between Training and Test Folds

We suggest the use of cross-validation for the evaluation step and provide a description for cross-validation that ensures independence between training and test folds in the following.

The problem associated with cross-validation, or any evaluation procedure based on splitting the available information into training and test partitions, can be most easily seen by considering the transitive closure of constraints as we discussed in Chapter 1. Let us consider the available objects and the available constraints (whether given directly or derived from the labels of some objects) as a graph where the data objects are the vertices and the constraints are the edges, e.g., with weight 0 (should-not-link) and weight 1 (should-link). The transitive closure provides all edges that can be induced from the given edges, an example of transitive closure of constraints was demonstrated in Introduction chapter in Figure 1.2.

We partition the available information into different folds, to use some part

for training and some part for testing. The transitive closure of pairwise instance level constraints, whether explicitly computed or not, can lead unintentionally to the indirect presence of information in some fold or partition. For example, suppose a training fold contains the constraints should-link(A,B) and should-not-link(B,C). If the test fold contains the constraint should-not-link(A,C), this is information that was, implicitly, already available during the clustering process even though only the explicit constraints in the training folds were given. Therefore, an ordinary setup for cross-validation for semi-supervised clustering evaluation can lead to significantly underestimating the true classification error with respect to the constraints. A more sophisticated cross-validation procedure, for example, would have to split the graph of constraints, possibly cutting some of the edges, in order to identify truly non-overlapping folds. This graph-based approach can provide a solution to avoid this pitfall at an abstract level. In the following, we provide a more detailed description of two scenarios for a proper cross-validation procedure, (I) using labeled objects, and (II) using pairwise instance-level should-link and should-not-link constraints. In both scenarios, we implement an efficient procedure that essentially results in correctly cutting the graph of constraints, to ensure independence between training and test folds.

Scenario I: Providing Labeled Objects

First consider the simpler and more widely applicable scenario where the user provides a certain percentage of labeled objects. This scenario is more widely applicable because, from labeled objects, we can derive instance level pairwise constraints (should-link and should-not-link constraints), and so use algorithms that require labeled objects as input as well as those that require a set of instance level constraints. This setup of the framework is as follows.

We partition the set of all labeled objects into the desired number of folds n as illustrated in Figure 5.4(a). As usual in cross-validation, one of the folds

is left out each time as a test set and the union of the remaining $n-1$ folds provides the training set. Instance level constraints can then be derived from the labels, independently for the training set ($n-1$ folds together) and for the test set. When two objects have the same label, this results in a should-link constraint; different labels for two objects result in a should-not-link constraint. If the framework is applied with an algorithm that uses labels directly, then we do not need to derive the constraints for the training set, only for the test set. In either case, only the labels or constraints coming from the union of the $n-1$ training folds are used in the clustering process. For the test fold, constraints are necessarily derived and they will obviously not have any overlap with the information contained in the training folds. Only these constraints are used for the estimation of the classification error for the clustering result.

The procedure is repeated n times, using each of the n folds once as the test fold.

Scenario II: Providing Pairwise Instance-Level Constraints

If we are directly given a set of (should-link/should-not-link) constraints, we extend this set by computing the transitive closure (e.g., if we have a should-link(A,B) and a should-link(B,C), we can derive a should-link(A,C)).

To ensure our cross-validation procedure avoids the pitfall of using the same information for training and testing, we partition the *data objects involved in any pairwise constraint* in training folds and test fold, then *delete all constraints that involve an object from the training fold and an object from the test fold.* For n-fold cross validation, we partition the objects into n folds and use, in turn, $n-1$ folds as training set and the remaining fold as test set, as illustrated in Figure 5.4(b). This way, when provided with pairwise instance-level constraints, the cross-validation procedure essentially reduces to the approach of Scenario I.

5.4.4 Transforming the Evaluation of Semi-Supervised Clustering to Classification Evaluation

Regardless of whether the clustering algorithm uses the labels or constraints, we can use the constraints to estimate the quality of a partition produced by the clustering algorithm. We can consider a produced partition as a classifier that distinguishes the class of should-link (class 1) from should-not-link (class 0) constraints. In other words we evaluate for pairs of objects in the test fold whether their constraint has been "recognized" by the clustering procedure (as opposed to evaluating the performance at an object level where we would consider if a single object is a member of an appropriate cluster in some clustering solution). A given clustering solution provides the basis to assess the degree to which the constraints in the test fold are satisfied or violated. As a consequence, we can use the well established F-measure to estimate the constraint satisfaction of a given solution.

The semi-supervised clustering problem can then be considered as a classification problem as follows: for each test fold, we have a set of should-link constraints (class 1) and should-not-link constraints (class 0). The clustering solution satisfies a certain number of these constraints: pairs of objects that are involved in a should-link constraint are either correctly paired in the same cluster (true positive for class 1 and true negative for class 0) or not (false negative for class 1 and false positive for class 0); likewise, pairs of objects that are involved in a should-not-link constraint are either correctly separated in two clusters (true positive for class 0 and true negative for class 1) or paired in the same cluster (false negative for class 0 and false positive for class 1). Based on these numbers, precision and recall, and the F-measure can be computed for each class. The average F-measure for both classes is the criterion for the overall constraint satisfaction of one test fold (see again Figure 5.3).

5.4.5 Model Selection

So far, we have noted a possible problem in evaluating semi-supervised clustering based on pairwise constraints when using some partition-based (hold-out) evaluation such as cross-validation, and we have elaborated how cross-validation can avoid this problem. Based on this improved formulation of a cross-validation framework for semi-supervised clustering (depending on the nature of the provided data, according to scenario I or scenario II), we can now discuss the process of model selection.

Cross-validation is suitable for estimating the classification error of a semi-supervised clustering algorithm on some given data set and given labels or pairwise constraints based on using n times a certain fraction of the available information for clustering ($\frac{n-1}{n}$) and, in each case, the remaining fraction (i.e., $\frac{1}{n}$) for evaluation. The average of the average F-measure over all n test folds is the criterion for the constraint satisfaction of some cluster model.

Based on this overall error estimation, we can compare the performance of some semi-supervised clustering algorithm when using different parameters. Users who apply this framework can now select the best available model for clustering their data. To do so, any algorithm is evaluated in cross-validation for each parameter setting that the user would like to consider, resulting in different cluster models of different quality.

Picking the best model based on the error estimate from a cross-validation procedure is still a guess, assuming that the error estimation can be generalized to when complete information is available. In what follows, we provide an outline of how well this assumption works for a variety of clustering algorithms applied to different data sets.

5.5 Experimental Evaluation

In the following, after describing the data sets, we first provide a brief evaluation of our model selection approaches in Section 5.5.1. We then demonstrate the results for FOSC framework applied to our hierarchical clustering methods and compare them with those of other state-of-the-art clustering algorithms in Section 5.5.2. Finally in Section 5.5.3 we further evaluate the performance of semi-supervised model selection when applied on our proposed hierarchical clustering algorithm.

Data Sets. We used real data sets with a variety of characteristics (no. of objects, dimensionality, and no. of clusters) and from different domains, namely, biology, text, image, and UCI data sets. Two data sets, "Articles-5" and "Cbrilpirivson", consist of very high dimensional representations of text documents. They are formed by 253 and 945 articles represented by 4636 and 1431 dimensions, respectively, both with 5 classes. Articles-5 is made available upon request by [117] and Cbrilpirivson [118] is available at `http://infoserver.lcad.icmc.usp.br/infovis2/PExDownload`. We used the Cosine measure as dissimilarity function for these data sets. Two data sets, "CellCycle-237" (made public by [111]) and "YeastGalactose" (used by [112]) represent gene-expression data and contain 237 respectively 205 objects (genes), 17 respectively 20 dimensions (conditions), and 4 known classes. For these data sets we used Euclidean distance on the z-score normalized objects, which is equivalent to using Pearson correlation on the original data. Two data sets, "Wine" and "Ecoli", are from the UCI Repository [116]. They contain 178 respectively 336 objects in 13 respectively 7 dimensions, with 3 respectively 8 classes. For these data sets we used Euclidean distance.

In addition to individual data sets, we also report average performance on two data set collections based on the Amsterdam Library of Object Images (ALOI) [119]. Image sets were created by randomly selecting k ALOI image

categories as class labels 100 times for each $k = 2, 3, 4, 5$, then sampling (without replacement), each time, 25 images from each of the k selected categories, thus resulting in 400 sets, each of which contains 2, 3, 4, or 5 clusters and 50, 75, 100, or 125 images (objects). The images were represented using 6 different descriptors: color moments, texture statistics from the gray-level co-occurrence matrix, Sobel edge histogram, 1st order statistics from the gray-level histogram, gray-level run-length matrix features, and gray-level histogram, with 144, 88, 128, 5, 44, and 256 attributes, respectively. We report results for the texture statistics (as a typical case), denoted by "ALOI-TS88", and for a 6-dimensional representation combining the first principal component extracted from each of the 6 descriptors using PCA, denoted by "ALOI-PCA". We used Euclidean distance in both cases.

5.5.1 Experimental Results for Semi-Supervised Model Selection

In the following, we first provide a brief evaluation of our model selection approaches (more results are shown in [54]) and also further discuss model selection experimental results in Section 5.5.3. Here we report the performance of GSS-MS and CVCP compared to the expected performance when having to guess the right parameter from the given range for Wine, Ecoli, YeastGalactose and a subset of ALOI-TS88, which contains 5 categories, ALOI-TS88-k5.

Experimental Setting. We apply GSS-MS and CVCP using a method referred to here as FOSC-OPTICSDend [56], which is FOSC applied to the equivalent dendrogram that can be constructed from an OPTICS reachability plot by using the transformation algorithm described in [28][4]. FOSC-OPTICSDend is a density-based clustering method that requires a parame-

[4]We also report the results for the semi-supervised k-means algorithm, called MPCK-means, in our publication [54]. Here we only report the results for density-based clustering algorithms because our research in this thesis is focused on the density-based paradigm.

ter *MinPts* that specifies the minimum number of points required in the ε-neighborhood of a dense (*core*) point. GSS-MS and CVCP select the best parameter values from a range of considered values. This range for *MinPts* values were set to $\{3, 6, 9, 12, 15, 18, 21, 24\}$.

We evaluate the performance of the clustering algorithms for different amounts of instance-level constraints. We first use the ground truth to generate a candidate pool of constraints by randomly selecting 10% of the objects from each class and generating all constraints between these objects. From this pool of constraints, we then randomly select subsets of 20% and 50% as input to the clustering method. All reported values are the average values computed over 50 independent experiments, each using a new set of constraints. We report "Overall F-Measure" [120] as an external evaluation measure by considering only the objects that are not involved in the constraints given to the semi-supervised clustering method (see Section 2.2.1). We report the performance of FOSC-OPTICSDend with the parameter values selected by GSS-MS and CVCP and compare it with the expected performance when having to guess the right parameter from the given range. The expected performance is defined as the average Overall F-Measure for FOSC-OPTICSDend, measured over all *MinPts* parameter values.

Clustering Results. In Table 5.2 we show the results for the scenario when constraints are provided directly as input to the semi-supervised clustering methods (the results for the label scenario were similar to those for the constraint scenario, giving the same overall picture, see [54] for further results). The values shown are the mean of the performance when selecting $MinPts$ using GSS-MS and CVCP, and the mean of the expected performance for the $MinPts$ in the given range. In all data sets, using GSS-MS and CVCP always results in much better performance than the mean of the expected performance. Also, the results improve when using larger numbers of constraints in a large number of the cases. For the ALOI data set collection, we did the test for all

Data Set	20 percent constraints			50 percent constraints		
	GSS-MS Mean	CVCP Mean	Expected Mean	GSS-MS Mean	CVCP Mean	Expected Mean
ALOI-TS88-k5	0.851	0.846	0.721	0.854	0.852	0.723
Wine	0.627	0.617	0.553	0.643	0.576	0.525
Ecoli	0.695	0.644	0.596	0.714	0.602	0.558
YeastGalactose	0.96	0.971	0.897	0.976	0.97	0.898

Table 5.2: Results for FOSC-OPTICSDend average performance using 20 and 50 percent constraints from the pool of constraints. 100/100 and 100/100 of GSS-MS results and 99/100 and 99/100 of CVCP results in the ALOI data set were significantly better than expected mean results for 20 and 50 percent constraint scenarios, respectively.

of the data sets in the collection; the number of data sets for which a difference is statistically significant is given in the caption of Table 5.2.

As we can see, GSS-MS in general has better performance than does CVCP. This can be due to a few reasons. First, the FOSC framework is a post-processing approach in which the constraints do not have any effect on the construction of the hierarchy. Thus the GSS-MS framework combined with the FOSC framework can be seen as the way to find the optimal solution to the constraint satisfaction problem over all input parameters. Second, when using GSS-MS we have more constraints from which to select the best parameter, as in CVCP where some constraints were lost when enforcing the independence of training and test sets. In the following we demonstrate the experimental evaluation of FOSC used along with GHDBSCAN. We further study the performance of our GSS-MS semi-supervised model selection method in Section 5.5.3.

5.5.2 Experimental Results for Semi-Supervised Extraction of a Flat Partition from GHDBSCAN Hierarchies

In this section and also in Section 5.5.3 we report the Adjusted Rand Index (ARI) [75] in all experiments in which it is calculated by removing the objects

that were involved in the constraints. We have also computed the Overall F-measure [120], but the results are omitted for the sake of compactness as the conclusions that can be drawn are the same as for ARI. Here we used "FOSC", on the hierarchies produced by both of our proposed generalized density-based clustering algorithms, namely, our density-based clustering algorithm with new mutual reachability distance, GHDBSCAN(NMRD), and our parameter-free density-based clustering algorithm, GHDBSCAN(NMRD+PF), (see Chapter 4). Utilizing the sum of the lifetimes stability that was reviewed in Section 5.2.3, we refer to these approaches as "$FOSC_{sum}$-GHDBSCAN(NMRD)" and "$FOSC_{sum}$-GHDBSCAN(NMRD+PF)." $FOSC_{sum}$-GHDBSCAN(NMRD) and $FOSC_{sum}$-GHDBSCAN(NMRD+PF) are compared with the following algorithms: *(i)* Semi-Supervised DBSCAN (SS-DBSCAN) [84], which is a semi-supervised density-based clustering algorithm based on OPTICS; *(ii)* the heuristic method by [28], referred to here as "OPTICS-AutoCl," which consists of the extraction of the leaf nodes of a compacted, density-based cluster tree from an OPTICS reachability plot.

Experimental Settings. We performed our experiments in the semi-supervised scenario when some constraints were available. The methods based on OPTICS demand a parameter *MinPts*, which is a smoothing factor also used by other density-based clustering algorithms e.g., [23, 25, 29, 73]. We set $MinPts = 4$ in all experiments in this section, which is a value commonly used in the literature. The speed up control value ϵ in OPTICS was not used ($\epsilon =$ "infinity"). The parameters required by OPTICS-AutoCl is set as suggested by the authors. Finally, in all the experiments, we applied the procedure from Section 5.2.1 with $m_{clSize} = MinPts = 4$, also as a smoothing factor.

The partial information provided to the semi-supervised methods was obtained in the form of labeled objects randomly selected from the data sets. These objects were not considered when assessing the quality of the results. SS-DBSCAN uses the labels explicitly. Our methods use should-link and should-

not-link constraints that can be derived from the labels. We set the number of labeled objects to 2, 4, 6, 8, and 10 percent of the number of objects in each data set. The results reported are averages over 100 random selections of label sets. Note that OPTICS-AutoCl is an unsupervised method, therefore their results do not change with the number of labeled objects available.

Clustering Results. The results are shown in Figures 5.5. $FOSC_{sum}$-GHDBSCAN(NMRD) and $FOSC_{sum}$-GHDBSCAN(NMRD+PF) outperform unsupervised OPTICS(AutoCl) in most of the data sets and usually by large margin. In one of the only two cases, YeastGalactose, where OPTICS-AutoCl is better, its performance is very close to that of $FOSC_{sum}$-GHDBSCAN(NMRD) and this small difference in results vanishes gradually by adding some constraints. Also note that all algorithms except SS-DBSCAN perform close to perfect on this data set and there is not much room for improvement. In the Article-5 data set, the results are quite high and overall results of our algorithms $FOSC_{sum}$-GHDBSCAN(NMRD) and $FOSC_{sum}$-GHDBSCAN(NMRD+PF) are quite similar to that of OPTICS-AutoCl. In all other data sets, both of our proposed approaches outperform the unsupervised method of OPTICS-AutoCl by a large margin as expected.

In regards to comparison of semi-supervised scenarios, both $FOSC_{sum}$-GHDBSCAN(NMRD) and $FOSC_{sum}$- GHDBSCAN(NMRD+PF) outperform SS-DBSCAN in all cases and in most of the cases by a large margin. Experimental results show that, both $FOSC_{sum}$-GHDBSCAN (NMRD) and $FOSC_{sum}$-GHDBSCAN (NMRD+PF) perform well even by using a very small number of constraints. In many cases this can be explained by knowing that FOSC operates based on an unsupervised selection method using cluster quality as a secondary objective, which can help the algorithm in the absence of constraints or ties to decide between clusters in the cluster tree. Furthermore, both of our proposed methods improve faster than SS-DBSCAN as the number of labeled objects increases. For example, as demonstrated

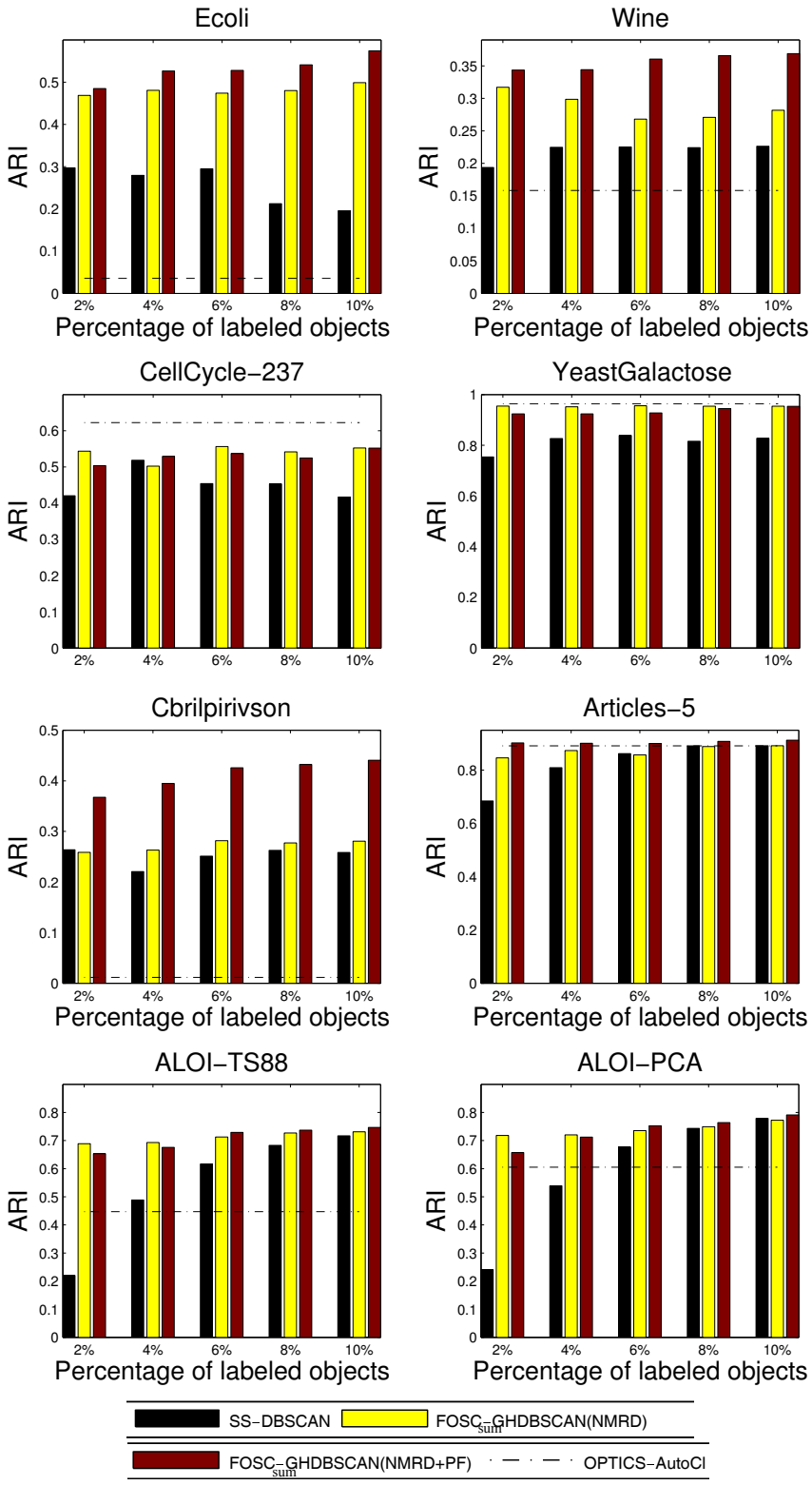


Figure 5.5: ARI results for SS-DBSCAN, $FOSC_{sum}$-GHDBSCAN(NMRD), $FOSC_{sum}$-GHDBSCAN(NMRD+PF) and OPTICS-AutoCl methods. OPTICS-AutoCl is an unsupervised method, thus its results are shown with a horizontal dashed line.

in Figure 5.5, in ALOI-PCA and Article-5, SS-DBSCAN can only achieve the same quality results as $FOSC_{sum}$-GHDBSCAN(NMRD) and $FOSC_{sum}$-GHDBSCAN(NMRD+PF) when the amount of labeled objects available is about 10%.

In some of the data sets such as CellCycle-237 and Ecoli, the performance of SS-DBSCAN deteriorates when larger amounts of constraints were added. This effect of adding constraints possibly decreasing the performance of clustering algorithms has been observed and discussed in the semi-supervised clustering literature [121]. In our experiments, we have excluded the objects involved in constraints when computing the evaluation measures, thus occurrence of this effect is particularly plausible. In the case of our proposed methods, however, such a behavior is rarely observed, and in these experimental results only $FOSC_{sum}$-GHDBSCAN(NMRD) shows such behaviour for the Wine data set. We further discuss this in the next section. In most of the data sets, $FOSC_{sum}$-GHDBSCAN(NMRD) and $FOSC_{sum}$-GHDBSCAN(NMRD+PF) with 2% constraints outperform SS-DBSCAN even when SS-DBSCAN uses 10% of labeled objects as supervision.

In regards to comparison of our two proposed approaches, we can see that in most of the data sets $FOSC_{sum}$-GHDBSCAN(NMRD+PF) outperforms or gives a comparable results to that of $FOSC_{sum}$-GHDBSCAN(NMRD). This can be explained because in contrast to $FOSC_{sum}$-GHDBSCAN(NMRD), which uses the same k in KNN distance density estimation to estimate the density, $FOSC_{sum}$-GHDBSCAN(NMRD+PF) uses a self-adjusting density estimation. In the following section we will use our proposed GSS-MS model selection framework along with our proposed DBCV clustering validation index to select a more appropriate parameter for the $FOSC_{sum}$-GHDBSCAN(NMRD) algorithm.

5.5.3 Experimental Results for Combination of Model Selection, FOSC and GHDBSCAN

In this section we provide an evaluation of our proposed method GSS-MS, which finds the parameters of our clustering method GHDBSCAN(NMRD) when used along with FOSC framework.

After discussing the experimental setup we demonstrate the performance of our clustering approach $FOSC_{sum}$-GHDBSCAN(NMRD) when the parameter values selected by GSS-MS compared to the expected performance when the user has to guess the right parameter from the given range of parameters to use with our $FOSC_{sum}$-GHDBSCAN(NMRD) algorithm. The expected performance is defined as the average ARI values for the $FOSC_{sum}$-GHDBSCAN(NMRD) results measured over all parameter values.

Experimental Setting. Our method $FOSC_{sum}$-GHDBSCAN(NMRD) requires parameter m_{pts} which specifies the minimum number of objects required in the ε-neighborhood of dense objects (similar to the *MinPts* parameter of OPTICS described in Section 5.5.2). We use our GSS-MS framework, which finds the partitions with maximum constraint satisfaction, with our density-based clustering validation method, DBCV (see Chapter 3) as a secondary criterion by applying to on our method $FOSC_{sum}$-GHDBSCAN(NMRD) to select the best parameter values from a range of 8 parameters from [3, 6, 9, 12, 15, 18, 21, 24] for m_{pts} values. In all the experiments, we applied the procedure in Section 5.2.1 with $m_{clSize} = 4$. We set the number of labeled objects to 2, 4, 6, 8, and 10 percent of the number of objects in each data set. All reported results are average values computed over 50 random selections of label sets.

Clustering Results. The clustering results are shown in Figure 5.6. In all of the cases the clustering results with GSS-MS selected parameters (GSS-MS Mean in Figure 5.6) outperform the expected performance of the $FOSC_{sum}$-GHDBSCAN(NMRD) algorithm results (Expected Mean in Figure 5.6) by a

large margin. Note that in the Wine data set, even though the expected performance deteriorates by adding more constraints (similar to the results of the previous section), this effect is not seen in the GSS-MS average results. One reason for this effect is the distribution of the constraints over different classes, which along with a specific m_{pts} parameter deteriorates the clustering results. However, GSS-MS finds the appropriate parameter to prevent this effect and finds the clustering that better matches the ground truth. Note that almost in all the cases the performance of $FOSC_{sum}$-GHDBSCAN(NMRD) with GSS-MS selected parameter using only 2% constraints is much better than the performance of $FOSC_{sum}$-GHDBSCAN(NMRD) using 10% constraints when the right parameter selected randomely from the range of paramters.

Furthermore, in two collections of image data sets, ALOI-TS88 and ALOI-PCA we performed paired t-tests with respect to ARI confirming that the large differences in performance are statistically significant at the $\alpha = 0.01$ significance level.

5.6 Summary

In this chapter we have introduced several approaches for the semi-supervised clustering.

In the first approach we study extracting flat clusterings from cluster hierarchies in a semi-supervised way. We described an optimal extraction of a flat clustering from a hierarchy, called FOSC [56], which can use unsupervised measures of cluster quality as well as semi-supervised measures such as constraint satisfaction. Then, we combined our proposed hierarchical density-based clustering algorithms GHDBSCAN(NMRD) and GHDBSCAN(NMRD+PF) with FOSC to select partitions consisting of most prominent clusters. Unlike most non-hierarchical (partitioning-like) algorithms for clustering, FOSC provides not only an optimal solution with respect to the different criteria it optimizes,

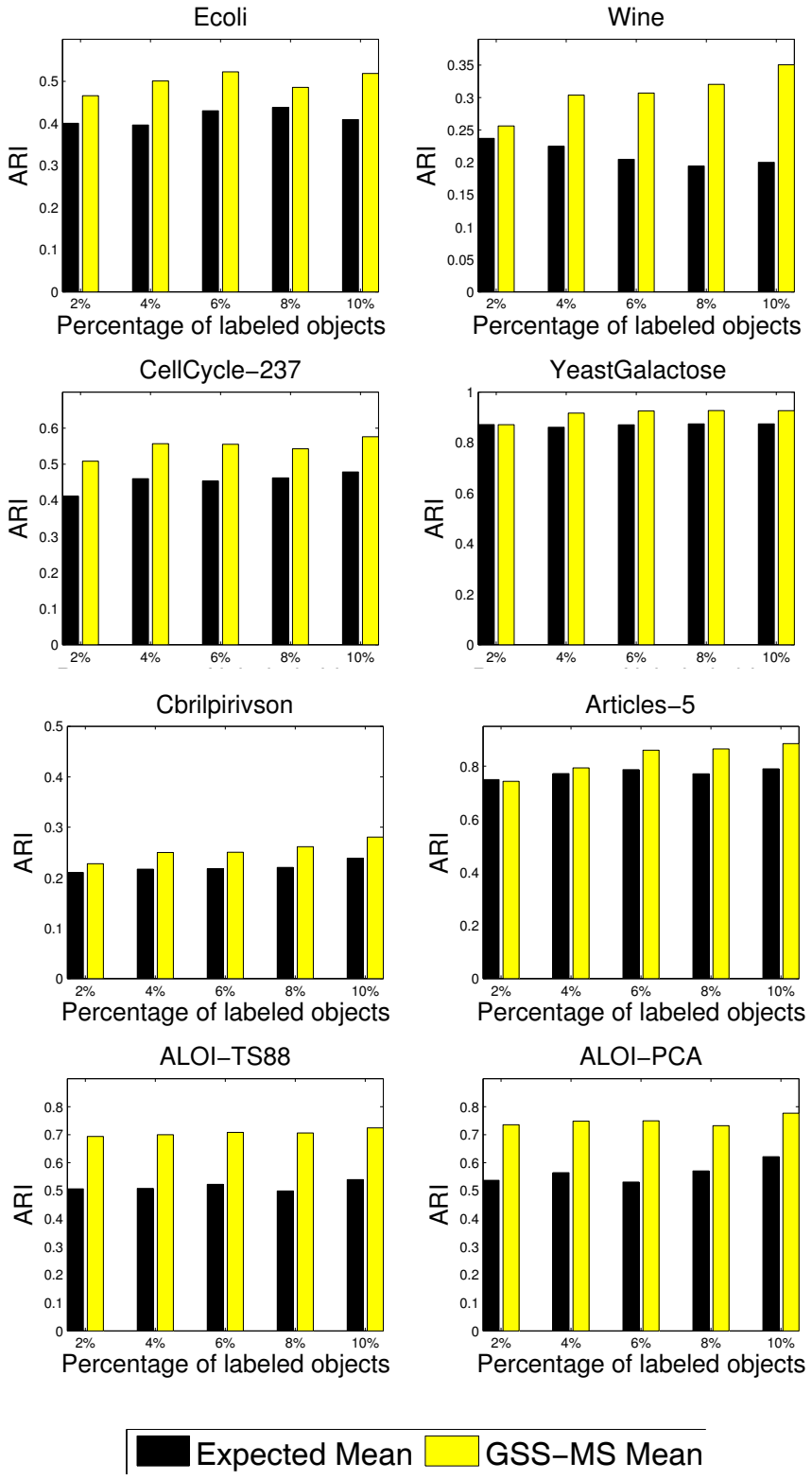


Figure 5.6: ARI results of $FOSC_{sum}$-GHDBSCAN(NMRD) with GSS-MS selected parameter (GSS-MS Mean) compared to the expected performance of $FOSC_{sum}$-GHDBSCAN(NMRD) when the user has to guess the right parameter from the range of 8 given parameters.

but also the number of clusters as a byproduct, rather than as an explicit or implicit input parameter. FOSC can, in principle, be applied to hierarchies of varied natures, so it is not hooked on a particular clustering inductive bias. Our experimental results of using FOSC along with our proposed hierarchical density-based clustering algorithms on a wide variety of real-world data sets shows that our approaches outperform other algorithms from the literature.

In the semi-supervised extraction case, only a cluster tree and the degree to which the clusters in the tree satisfy instance-level constraints provided by the user is needed. We have not considered weighted constraints, but all the approaches in this chapter can be straightforwardly generalized to do so. When the user provides no constraints at all (unsupervised case) or part of the clusters in the hierarchy cannot be distinguished in terms of constraint satisfactions/violations, then unsupervised measures of cluster quality can be applied. An important topic for future research is the study of quality measures suitable for clustering hierarchies of varied natures, eventually constructed in a semi-supervised way by using constraints of different types possibly other than should-link and should-not-link, e.g., see [59, 55, 60].

In the second and third approach to semi-supervised clustering, we study two methods for model selection. First, we provided a General Semi-Supervised Model Selection framework, GSS-MS, which can be used to find the most appropriate model (e.g., density-parameters) for a given problem. GSS-MS provides an optimal solution with respect to the constraint-satisfaction criterion. Second, we provided a semi-supervised model-selection method, CVCP, based on a sound cross-validation procedure. The future work for our model selection approaches includes the investigation of how such approaches could be extended to also allow the comparison and selection of different clustering methods. In regards to GS-MSS one can also investigate the weighted combination of primary and secondary criteria to select the best model.

Chapter 6

Conclusion and Future Work

In this thesis, we have presented a number of advances in two major clustering paradigms, density-based and semi-supervised clustering. In particular, in the context of density-based clustering, we have studied both the validation of density-based clustering, and the hierarchical density-based clustering.

Our proposed Density-Based Clustering Validation index, DBCV, provides an effective evaluation for arbitrarily-shaped clustering. Our index is formulated based on new kernel density function, which is used to compute the density of the objects and the density of the path between pairs of objects that in turn are used to evaluate the within- and between-cluster density connectedness and separation of clustering results. DBCV directly takes into account density and shape properties of clusters, and also properly deals with noise objects, which are intrinsic to the definition of density-based clustering. We also adapted our approach of dealing with noise to other relative validity criteria in order to allow them handle noise properly.

Our method allows the user to select the most appropriate clustering model and set proper parameters, e.g., selecting the most appropriate density-based clustering algorithm among several density-based clustering algorithms and also finding the proper density-parameters or number of clusters for such algorithms. By experimenting on results of various density-based clustering algo-

rithms with many different parameter settings on a variety of real-world and synthetic datasets, we showed that DBCV outperforms specialized methods from the literature and that our noise adaptation approach also shows promising results.

In addition to DBCV, in the density-based clustering context we make major contributions in the area of hierarchical density-based clustering methods. We proposed several hierarchical density-based approaches that were introduced and discussed in chronological and progressive order because each approach has utilized some concepts from the previous algorithm and improved on it both theoretically and practically.

We proposed an improved version of AUTO-HDS by getting rid of its user-defined parameter r_{shave} in such a way that the cluster extraction stage of AUTO-HDS can be implemented in a simpler and more accurate way. We reviewed HDBSCAN algorithm, a complete density-based clustering hierarchy representing all possible DBSCAN-like solutions for an infinite range of density thresholds. We then proposed a novel generalized density-based clustering approach, GHDBSCAN: *(i)* having recognized the two essential components of the DBSCAN algorithm, we discussed the possibilities of replacing these two components to improve on the current state-of-the-art methods; *(ii)* we proposed a novel, theoretically sound approach to estimating the density of the objects along the path, and we proposed an approach to estimating the density of the objects based on all other objects in the data set, which can be used to replace the k nearest neighbor density-estimate, which has certain limitations and *(iii)* we proposed two novel hierarchical density-based clustering methods GHDBSCAN(NMRD) and GHDBSCAN(NMRD+PF).

By undertaking an extensive experimental evaluation on a wide variety of real-world data sets, we have shown that our methods for density-based clustering perform, in general, significantly better and more robustly than state-of-the-art methods.

In the context of semi-supervised clustering, we first described FOSC [56], a framework that enables the user to use a small amount of information in the form of instance-level should-link and should-not-link constraints to extract a flat clustering solution from optimal local cuts through cluster hierarchies. Second we combined FOSC with our proposed hierarchical density-based clustering approaches GHDBSCAN(NMRD) and GHDBSCAN(NMRD+PF) to further improve their performance, and we also provided two semi-supervised model selection frameworks to select the best parameter values for clustering algorithms.

In the FOSC framework, the extraction of a flat clustering from a cluster tree has been formulated as an optimization problem and also a linear complexity algorithm and with respect to time and memory it provides the globally optimal solution to this optimization problem. We applied FOSC to the hierarchies obtained by our proposed density-based clustering methods and on varied real-world data sets. The results of these experiments show that our approach outperforms other state-of-the-art unsupervised and semi-supervised clustering algorithms.

In regards semi-supervised model selection, we have provided two approaches, GSS-MS and CVCP. First we provided a General Semi-Supervised Model Selection framework, GSS-MS, that can be used to find the most appropriate model (e.g., density-parameters) for a given problem. GSS-MS provides an optimal solution with respect the constraint-satisfaction criterion and considering a secondary unsupervised criteria. Second we provided a semi-supervised model-selection method, CVCP, based on a sound cross-validation procedure.

6.1 Future Work

There are several interesting challenges for possible future research for all our methods discussed here.

Regarding DBCV, based on our new view in density-based clustering as we discussed in Chapter 4, there are two essential components for density estimation; density estimation at each point and density estimation at each path between pairs of objects. Each of these components can affect the quality of density estimation and thus would affect our final evaluation process. One can investigate the effect of each component on the density-based validation index separately and propose new ways to calculate each of these components to improve on DBCV. We also used the estimates of minimum density area inside the cluster and maximum density area between clusters to measure the cluster quality; another future study would be investigating the use of the average density of paths between pairs of objects.

The hierarchical clustering methods described in this thesis are based on using the k nearest neighbor and all-points core distance for density-estimation that were used in estimating the density at each object and the paths between objects. Our generalization of the density-based clustering algorithms opened up a vista for future development of density-based clustering that can allow use of other density estimations. One can also investigate applying other types of density estimation to improving different aspects of our methods; one aspect that can be improved is the quality of the clustering results in certain scenarios. Another direction for future research is to investigate other alternative stability measures in order to measure the quality of clusters in the selection process. We introduced bounded excess of mass and sum of life times stability measures in this thesis; one can investigate other types of quality measures and/or combine them with current measures to capture different aspects of the quality of the clusters.

In the context of semi-supervised clustering approaches, all our methods can lead to much new research. In regards to semi-supervised extraction of flat solution from a hierarchy, research can be conducted on the extraction of the flat solution in a semi-supervised way by using different types of should-

and should-not-link constraints (e.g. see [59, 55]) or other types of constraints. Also, one can study applying similar strategies in different stages of hierarchical clustering, e.g., with a similar strategy constraints can be used in a similar way during the clustering stage instead of post-processing stage of hierarchical clustering algorithms. Similar approaches can also be applied in semi-supervised model selection. Future work in regards to semi-supervised model selection also includes the investigation of how such approaches could be extended to allow the comparison and selection of different clustering methods. In regards to GS-MSS, one can also investigate the weighted combination of primary and secondary criteria to select the best model.

Bibliography

[1] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[2] B. S. Everitt, S. Landau, M. Leese, and D. Stahl. *"Cluster Analysis"*. West Sussex, UK: Wiley, 2011.

[3] J. A. Hartigan. *Clustering Algorithms*. New York: Wiley & Sons, 1975.

[4] M. R. Anderberg. *"Cluster Analysis for Applications"*. New York: Academic Press, 1973.

[5] D. Jiang, C. Tang, and A. Zhang. "cluster analysis for gene expression data: A survey". *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, 2004. doi: 10.1109/TKDE.2004.68.

[6] A. K. Jain, M. N. Murty, and P. J. Flynn. "data clustering: A review". *ACM Computing Surveys*, 31(3):264–323, 1999. doi: 10.1145/331499.331504.

[7] L. J. Hubert P. Arabie and. "an overview of combinatorial data analysis". *World Scientific Publication*, 1996.

[8] K. Fukunaga. *"Introduction to Statistical Pattern Recognition"*. Boston, MA: Academic Press, 2nd edition, 1990.

[9] Y. Kodratoff and R. S. Michalski. *"Machine Learning: An Artificial Intelligence Approach"*, volume 3. San Mateo, CA: Morgan Kaufmann, 1990.

[10] S. Basu, I. Davidson, and Eds K. Wagstaff, editors. *"Constrained Clustering: Advances in Algorithms, Applications and Theory"*. New York: CRC Press, 2008.

[11] K. L. Wagstaff. *"Intelligent Clustering with Instance-Level Constraints"*. PhD thesis, Dept of Computer Science, Cornell University, 2002.

[12] S. T. Roweis and L. K. Saul. "nonlinear dimensionality reduction by locally linear embedding". *Science*, 290(5500):2323–2326, 2000.

[13] B. W. Silverman. "density estimation for statistics and data analysis.". *Monographs on Statistics and Applied Probability, B. W. Silverman, Ed. London: Chapman and Hall*, 26, 1986.

[14] C. M. Bishop. *"Pattern Recognition and Machine Learning"*. New York: Springer, 2006.

[15] D. Xu, R. Wunsch. "survey of clustering algorithms". *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[16] D. Wunsch R. Xu. *"Clustering"*, volume 10. Hoboken, NJ: John Wiley & Sons, 2008.

[17] P. N. Tan, M. Steinbach, and V. Kumar. *"Introduction to Data Mining"*. Boston, MA: Addison Wesley, 2006.

[18] M. Ester. "density-based clustering". In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, pages 795–799. New York: Springer, 2009. doi: 10.1007/978-0-387-39940-9_605.

[19] J. Sander. "density-based clustering". In *in Encyclopedia of Machine Learning*, pages 270–273. New York: Springer, 2010.

[20] H.P. Kriegel, P. Kröger, J. Sander, and A. Zimek. "density-based clustering". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[21] J. Sander, M. Ester, H. P. Kriegel, and X. Xu. "density-based clustering in spatial databases: The algorithm GDBSCAN and its applications". *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998. doi: 10.1023/A:1009745219419.

[22] J. N Hwang, S. R. Lay, and A. Lippman. "nonparametric multivariate density-estimation: a comparative study.". *IEEE Transactions on Signal Processing*, 42:2975–2810, 1994.

[23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "a density-based algorithm for discovering clusters in large spatial databases with noise". In *in KDD96*, pages 226–231, 1996.

[24] A. Hinneburg and D. A. Keim. "a general approach to clustering in large databases with noise". *Knowl. and Info. Sys.*, 5(4):387–415, 2003.

[25] G. Gupta, A. Liu, and J. Ghosh. "automated hierarchical density shaving: A robust automated clustering and visualization framework for large biological data sets". *IEEE/ACM Trans. Computational Biology and Bioinformatics*, 7(2):223–237, Apr–June 2010.

[26] T. Pei, A. Jasra, D. J. Hand, A. X. Zhu, and C. Zhou. "DECODE: a new method for discovering clusters of different densities in spatial data". *Data Mining and Knowledge Discovery*, 18(3):337–369, 2009. doi: 10.1007/s10618-008-0120-3.

[27] W. Stuetzle and R. Nugent. "a generalized single linkage method for estimating the cluster tree of a density". *J. Comp. and Graph. Stat.*, 19 (2):397–418, 2010.

[28] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky. "automatic extraction of clusters from hierarchical clustering representations". In *in Pacific-Asia Conf. of Adv. in Knowl. Discovery and Data Mining*, pages 75–87, 2003.

[29] H. Sun, J. Huang, J. Han, H. Deng, P. Zhao, and B. Feng. "gskeletonclu: Density-based network clustering via structure-connected tree division or agglomeration". *in: IEEE Int. Conf. Data Min. (ICDM)*, 2010.

[30] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. "OPTICS: Ordering points to identify the clustering structure". In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA*, pages 49–60, 1999.

[31] G. W. Milligan and M. C. Cooper. "an examination of procedures for determining the number of clusters in a data set". *Psychometrika*, 50(2): 159–179, 1985.

[32] L. Vendramin, R. J. G. B. Campello, and E. R. Hruschka. "relative clustering validity criteria: A comparative overview". *Statistical Analysis and Data Mining*, 3(4):209–235, 2010. doi: 10.1002/sam.10080.

[33] L. Vendramin, R. J. G. B. Campello, and E. R. Hruschka. "on the comparison of relative clustering validation criteria". In *in SDM09*, pages 733–744, 2009.

[34] L. Vendramin, P. A. Jaskowiak, and R. J. G. B. Campello. "on the combination of relative clustering validity criteria". In *Proceedings of the 25th SSDBM*, page 4. ACM, 2013.

[35] N. R. Pal and J. Biswas. "cluster validation using graph theoretic concepts ". *Pattern Recognition*, 30(6):847 – 857, 1997. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/S0031-3203(96)00127-6. URL http://www.sciencedirect.com/science/article/pii/S0031320396001276.

[36] M. Halkidi and M. Vazirgiannis. "a density-based cluster validity approach using multi-representatives". *Pattern Recognition Letters*, 29(6): 773–786, 2008.

[37] C. H. Chou, M. C. Su, and E. Lai. "a new cluster validity measure and its application to image compression". *Pattern Analysis and Applications*, 7(2):205–220, 2004. ISSN 1433-7541. doi: 10.1007/s10044-004-0218-1. URL http://dx.doi.org/10.1007/s10044-004-0218-1.

[38] E. J. Pauwels and G. Frederix. "finding salient regions in images: Nonparametric clustering for image segmentation and grouping ". *Computer Vision and Image Understanding*, 75(1–2):73 – 85, 1999. ISSN 1077-3142. doi: http://dx.doi.org/10.1006/cviu.1999.0763. URL http://www.sciencedirect.com/science/article/pii/S1077314299907634.

[39] K. Rizman Žalik and Borut Žalik. "validity index for clusters of different sizes and densities". *Pattern Recognition Letters*, 32(2):221 – 234, 2011. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2010.08.007. URL http://www.sciencedirect.com/science/article/pii/S0167865510002928.

[40] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. London, UK: Arnold, 2001.

[41] Y. Zhao and G. Karypis. "hierarchical clustering algorithms for document datasets". *in Data Mining and Knowledge Discovery*, 10(2):141–168, 2005.

[42] H. Blockeel, L. De Raedt, and J. Ramon. "top-down induction of clustering trees". *in arXiv preprint cs/0011032*, 2000.

[43] J. Struyf and S. Džeroski. "clustering trees with instance level constraints". *in: Eur. Conf. Mach. Learning (ECML)*, page 359–370, 2007.

[44] R. S. Michalski and R. E. Stepp. "learning from observation: Conceptual clustering". In *Machine learning*, pages 331–363. Springer, 1983.

[45] S. Brecheisen, Hans-Peter Kriegel, Peer Kröger, and Martin Pfeifle. "visually mining through cluster hierarchies". In *SDM*, pages 400–411. SIAM, 2004.

[46] E. Boudaillier and G. Hebrail. "interactive interpretation of hierarchical clustering". *in: Principles of Data Min. and Knowl. Disc.*, pages 288–298, 1997.

[47] E. Boudaillier and G. Hebrail. "interactive interpretation of hierarchical clustering". *Intell Data Anal*, (2):229–244, 1998.

[48] J. R. Kettenring. "the practice of cluster analysis". *Journal of Classification*, 23(1):3–30, 2006. doi: 10.1007/s00357-006-0002-6.

[49] G. Gupta, A. Liu, and J. Ghosh. "hierarchical density shaving: A clustering and visualization framework for large biological data sets". *Proc. IEEE ICDM Workshop Data Mining in Bioinformatics (DMB)*, pages 89–93, 2006.

[50] W. Stuetzle. "estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample". *Journal of Classification*, 20(1): 25–47, 2003. doi: 10.1007/s00357-003-0004-6.

[51] A. Hinneburg and D. A. Keim. "an efficient approach to clustering in large multimedia databases with noise". In *in KDD98*, pages 58–65, 1998.

[52] D. Ferraretti, G. Gamberoni, and E. Lamma. "automatic cluster selection using index driven search strategy". *in: Int. Conf. of the Italian Assoc. Artif. Intell. (AI*IA)*, 2009.

[53] D. Moulavi, M. Hajiloo, J. Sander, P. F. Halloran, and R. Greiner. "combining gene expression and interaction network data to improve kidney lesion score prediction". *International Journal of Bioinformatics Research and Applications*, 8(1):54–66, 2012.

[54] M. Pourrajabi, D. Moulavi, R.J.G.B. Campello, A. Zimek, J. Sander, and R. Goebel. "model selection for semi-supervised clustering.". In *EDBT*, pages 331–342, 2014.

[55] S. Gilpin and I. Davidson. "incorporating SAT solvers into hierarchical clustering algorithms: An efficient and flexible approach". *in: ACM SIGKDD Int. Conf. Knowl. Disc. and Data Min. (KDD)*, pages 1136–1144, 2011.

[56] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander. "a framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies". *in: Data Mining and Knowledge Discovery*, 27(3):344–371, 2013.

[57] D. Klein, S. D. Kamvar, and C. D. Manning. "from instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering". In *in ICML02*, pages 307–314, 2002.

[58] K. Bade and A. Nürnberger. "creating a cluster hierarchy under constraints of a partially known hierarchy". *in: SIAM Int. Conf. Data Min. (SDM)*, pages 13–23, 2008.

[59] I. Davidson and S. Ravi. "using instance-level constraints in agglomer-

ative hierarchical clustering: Theoretical and empirical results". *Data Mining and Knowledge Discovery*, 18:257–282, 2009.

[60] L. Zheng and T. Li. "semi-supervised hierarchical clustering". In *in ICDM11*, pages 982–991, 2011.

[61] A.G. Skarmeta, A. Bensaid, and N. Tazi. "data mining for text categorization with semi-supervised agglomerative hierarchical clustering". *International Journal of Intelligent systems*, 15(7):633–646, 2000.

[62] M. Benkhalifa, A. Mouradi, and H. Bouyakhf. "integrating wordnet knowledge to supplement training data in semi-supervised agglomerative hierarchical clustering for text categorization". *International Journal of Intelligent Systems*, 16(8):929–947, 2001.

[63] I. Davidson and S. Ravi. "agglomerative hierarchical clustering with constraints: Theoretical and empirical results". *in: Eur. Conf. Principles and Practice of Knowl. Disc. in Databases (PKDD)*, 2005.

[64] K. Hj and L. Sg. "an effective document clustering method using user-adaptable distance metrics". *in: ACM Symp. Applied Comp. (SAC)*, 2002.

[65] K. Bade and A. Nürnberger. "personalized hierarchical clustering". In *in Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on*, pages 181–187. IEEE, 2006.

[66] K. Bade, M. Hermkes, and A. Nürnberger. "user oriented hierarchical information organization and retrieval". *in: Eur. Conf. Mach. Learning (ECML)*, page 518–526, 2007.

[67] Y. Hamasuna, Y. Endo, and S. Miyamoto. "on agglomerative hierarchical clustering using clusterwise tolerance based pairwise constraints". *J Advanced Comp Intell and Intell Informatics*, 16(1):174–179, 2012.

[68] H. Kestler, J. Kraus, G. Palm, and F. Schwenker. "on the effects of constraints in semisupervised hierarchical clustering". *in: IAPR Work. Artif. Neural Net. Patt. Rec. (ANNPR)*, 2006.

[69] J. M. Kraus, G. Palm, and H. A. Kestler. "on the robustness of semi-supervised hierarchical graph clustering in functional genomics". *in: 5th Int. Work. Min. and Learning with Graphs (MLG)*, page 1–4, 2007.

[70] T. Xiong, S. Wang, A. Mayers, and E. Monga. "semi-supervised parameter-free divisive hierarchical clustering of categorical data". *in: Pacific-Asia Conf. Knowl. Disc. and Data Min. (PAKDD)*, page 265–276, 2011.

[71] D. Moulavi, P. A. Jaskowiak, R.J.G.B. Campello, A. Zimek, and J. Sander. "density-based clustering validation". *in Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA*, 2014.

[72] R. J. G. B. Campello, D. Moulavi, and J. Sander. "a simpler and more accurate auto-hds framework for clustering and visualization of biological data". *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(6):1850–1852, 2012.

[73] R. J. G. B. Campello, D. Moulavi, and J. Sander. "density-based clustering based on hierarchical density estimates". In *in PAKDD13*, volume 27, pages 344–371, 2013.

[74] D. Wishart. "mode analysis: A generalization of nearest neighbor which reduces chaining effects". In *in Numerical Taxonomy: Colloquium in Numerical Taxonomy, University of St. Andrews*, 1968.

[75] L. Hubert and P. Arabie. "comparing partitions". *J. Classification*, 2(1): 193–218, 1985.

[76] I. Färber, S. Günnemann, H.-P. Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek. "on using class-labels in evaluation of clusterings". In *in MultiClust10*, 2010.

[77] A. Dasgupta and A. E. Raftery. "detecting features in spatial point processes with clutter via model-based clustering". *Journal of the American Statistical Association*, 93(441):294–302, 1998.

[78] J. C. Dunn. "well separated clusters and optimal fuzzy partitions". *Journal of Cybernetics*, 4(1):95–104, 1974.

[79] J. Yang and I. Lee. "cluster validity through graph-based boundary analysis". In *in IKE*, pages 204–210, 2004.

[80] D. Davies and D. Bouldin. "a cluster separation measure". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[81] M. Halkidi, M. Vazirgiannis, and Y. Batistakis. "quality scheme assessment in the clustering process". *in: Eur. Conf. Principles and Practice of Knowl. Disc. in Databases (PKDD)*, page 265–276, 2000.

[82] M. Halkidi and M. Vazirgiannis. "clustering validity assessment: Finding the optimal partitioning of a data set". *Proceedings 2001 IEEE International Conference on Data Mining*, pages 187–194, 2001.

[83] S. Wu and T. W.S. Chow. Clustering of the self-organizing map using a clustering validity index based on inter-cluster and intra-cluster density. *Pattern Recognition*, 37(2):175 – 188, 2004. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/S0031-3203(03)00237-1. URL `http://www.sciencedirect.com/science/article/pii/S0031320303002371`.

[84] L. Lelis and J. Sander. "semi-supervised density-based clustering". In

Proceedings of the 9th IEEE International Conference on Data Mining (ICDM), Miami, FL, pages 842–847, 2009.

[85] M. Herbin, N. Bonnet, and P. Vautrot. "estimation of the number of clusters and influence zones". *Patt. Rec. Letters*, 22(14):1557–1568, 2001.

[86] A. Demiriz, K. P. Bennett, and M. J. Embrechts. "semi-supervised clustering using genetic algorithms". *Artificial Neural Networks in Engineering (ANNIE-99)*, pages 809–814, 1999.

[87] K. Wagstaff, C. Claire, S. Rogers, and S. Schrödl. "constrained k-means clustering with background knowledge". In *in ICML*, volume 1, pages 577–584, 2001.

[88] O. R. Zaïane and C. H. Lee. "clustering spatial data when facing physical constraints". In *in Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 737–740. IEEE, 2002.

[89] K. Basu, A. Banerjee, and R. Mooney. "semi-supervised clustering by seeding". *Proceedings of the 19th International Conference on machine learning (ICML 2002)*, pages 19–26, 2002.

[90] S. Basu, M. Bilenko, and R. J. Mooney. "active semi-supervision for pairwise constrained clustering". *in Proceedings of the 2004 SIAM International Conference on Data Mining (SDM2004)*, pages 333–344, 2004.

[91] C. Ruiz, M. Spiliopoulou, and E. Menasalvas. "density-based semi-supervised clustering". *Data mining and knowledge discovery*, 21(3):345–370, 2010.

[92] I. Davidson and Z. Qi. "finding alternative clusterings using constraints". In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), Pisa, Italy*, pages 773–778, 2008. doi: 10.1109/ICDM.2008.141.

[93] H. Zhao and Z. Qi. "hierarchical agglomerative clustering with ordering constraints". In *Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on*, pages 195–199. IEEE, 2010.

[94] S. Miyamoto and A. Terami. "semi-supervised agglomerative hierarchical clustering algorithms with pairwise constraints". In *in Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.

[95] E. B. Ahmed, A. Nabli, and F. Gargouri. "shacun: Semi-supervised hierarchical active clustering based on ranking constraints". In *in Advances in Data Mining: Applications and Theoretical Aspects: 10th Industrial Conference, Berlin, Germany*, pages 194–208. 2012.

[96] C. Böhm and C. Plant. "HISSCLU: a hierarchical density-based method for semi-supervised clustering". In *Proceedings of the 11th International Conference on Extending Database Technology (EDBT), Nantes, France*, pages 440–451, 2008. doi: 10.1145/1353343.1353398.

[97] S. Hang, Y. Zhou, and L. Y. Chun. "incorporating biological knowledge in to density- based clustering analysis of gene expression data". *in: 6th Int. Conf. on Fuzzy Systems and Knowledge Discovery*, pages 52–56, 2009.

[98] C. Ruiz, M. Spiliopoulou, and E. Menasalvas. "c-dbscan: Density-based clustering with constraints". In *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, pages 216–223. Springer, 2007.

[99] R. Jin J. Zhu L. Wu, S. C. H. Hoi and N. Yu. "learning bregman distance functions for semi-supervised clustering.". 24(3):478–491, 2012.

[100] A. Topchy M. H. C. Law and A. K. Jain. "clustering with soft and group constraints.". pages 662–670, 2004.

[101] K. Wagstaff and C. Cardie. "clustering with instance-level constraints". *AAAI/IAAI, Proceedings of the Seventeenth International Conference on Machine Learnin*, 1097:1103–1110, 2000.

[102] C. Antunes. A. Silva. "semi-supervised clustering: A case study.". pages 252–263, 2012.

[103] M. Bilenko S. Basu and R. J. Mooney. "a probabilistic framework for semi-supervised clustering". In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Seattle, WA*, pages 59–68. ACM, 2004.

[104] C. Campbell P. Li, Y. Ying. "a variational approach to semi-supervised clustering". 2009.

[105] A. B. Baya and P. M. Granitto. "how many clusters: A validation index for arbitrary shaped clusters". *in IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 401–414, 2013. doi: doi.ieeecomputersociety.org/10.1109/TCBB.2013.32.

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

[107] R. B. Calinski and J. Harabasz. "a dendrite method for cluster analysis". *Communications in Statistics*, 3(1):1–27, 1974.

[108] U. Maulik and S. Bandyopadhyay. "performance evaluation of some clustering algorithms and validity indices". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(12):1650–1654, 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1114856.

[109] E. Sivogolovko and B. Novikov. "validating cluster structures in data mining tasks". In *EDBT/ICDT Workshops*, pages 245–250, 2012.

[110] K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[111] K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery, and W. L. Ruzzo. "model-based clustering and data transformations for gene expression data". *Bioinformatics*, 17(10):977–987, 2001.

[112] K. Y. Yeung, M. Medvedovic, and R. E. Bumgarner. "clustering gene expression data with repeated measurements". *Genome Biol.*, 4(5):R34.1–R34.17, Apr-2003.

[113] G. Gupta, A. Liu, and J. Ghosh. "automated hierarchical density shaving and gene diver". Technical report, IDEAL-2006-TR05, Dept. of Electrical and Computer Eng., Univ. of Texas at Austin, 2006.

[114] S. Dhandapani, G. Gupta, and J. Ghosh. *"Design and implementation of scalable hierarchical density based clustering"*. PhD thesis, IDEAL-2010-06, Dept. Electrical and Computer Eng., University of Texas at Austin, Tech. Rep.,, 2010.

[115] D. W. Muller and G. Sawitzki. "excess mass estimates and tests for multimodality". *J. Amer. Stat. Association*, 86(415):738–746, 1991.

[116] A. Frank and A. Asuncion. "*UCI Machine Learning Repository*", 2010. URL `http://archive.ics.uci.edu/ml`.

[117] M. C. Naldi, R. J. G. B. Campello, E. R. Hruschka, and A. C. P. L. F. Carvalho. "efficiency issues of evolutionary k-means". *Applied Soft Computing*, 11(2):1938–1952, 2011.

[118] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. "least square projection: A fast high-precision multidimensional projection technique and its application to document mapping". *IEEE Trans. Visual. & Comp. Graphics*, 14(3):564–575, 2008.

[119] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. "the Amsterdam library of object images". *Int. J. of Computer Vision*, 61(1): 103–112, 2005.

[120] B. Larsen and C. Aone. "fast and effective text mining using linear-time document clustering.". In *in Int. Conf. Knowl. Discovery and Data Mining*, pages 16–22, 1999.

[121] I. Davidson, K. L. Wagstaff, and S. Basu. *"Measuring constraint-set utility for partitional clustering algorithms."*. Springer, 2006.

Appendix A

Appendix

A.1 Proofs

In this section we present concise proofs related to the propositions presented in Chapters 3 and 4.

Proposition A.1.1. *The all-points-core-distance of each object* $\mathbf{o}$, $a_{pts}coredist(\mathbf{o})$, *with respect to all other* $n-1$ *objects in a d-dimensional data set* X *is between the second and last nearest neighbor distance of that object, i.e.,*

$$KNN(\mathbf{o},2) \leq a_{pts}coredist(\mathbf{o}) \leq KNN(\mathbf{o},n)$$

Proof. For $i \in \{2,3,\ldots,n\}$, we have:

$$0 < KNN(\mathbf{o},2) \leq KNN(\mathbf{o},i) \rightarrow \left(\frac{1}{KNN(\mathbf{o},i)}\right)^d \leq \left(\frac{1}{KNN(\mathbf{o},2)}\right)^d \quad \text{(A.1)}$$

Therefore,

$$\sum_{i=2}^{n}\left(\frac{1}{KNN\,(\mathbf{o},i)}\right)^d \leq (n-1)\left(\frac{1}{KNN\,(\mathbf{o},2)}\right)^d \quad \text{(A.2)}$$

$$\left((n-1)\frac{\frac{1}{KNN(\mathbf{o},2)}^{d}}{n-1}\right)^{-\frac{1}{d}} \leq \left(\frac{\sum_{i=2}^{n}\left(\frac{1}{KNN(\mathbf{o},i)}\right)^{d}}{n-1}\right)^{-\frac{1}{d}}$$

We have $\left((n-1)\frac{\frac{1}{KNN(\mathbf{o},2)}^{d}}{n-1}\right)^{-\frac{1}{d}} = KNN(\mathbf{o},2)$ and also

$$a_{pts}coredist(\mathbf{o}) = \left(\frac{\sum_{\substack{o_i \in X \\ o_i \neq o}}\left(\frac{1}{d(\mathbf{o},o_i)}\right)^{d}}{n-1}\right)^{-\frac{1}{d}} = \left(\frac{\sum_{i=2}^{n}\left(\frac{1}{KNN(\mathbf{o},i)}\right)^{d}}{n-1}\right)^{-\frac{1}{d}} \tag{A.3}$$

leading to Equation A.4.

$$KNN(\mathbf{o},2) \leq a_{pts}coredist(\mathbf{o}). \tag{A.4}$$

The upper bound inequality can be similarly proved. □

Proposition A.1.2. *Let n objects be uniformly distributed random variables in a d-dimensional unit hypersphere and $\boldsymbol{o}$ be an object in the center of this hypersphere. For the all points core distance of $\boldsymbol{o}$ we have:*

$$a_{pts}coredist(\mathbf{o}) = (ln(n-1)+\gamma+\varepsilon)^{-\frac{1}{d}} \approx \ln(n)^{-\frac{1}{d}} \tag{A.5}$$

where $\gamma \approx 0.5772$ and $\varepsilon \approx \frac{1}{2n}$ which approaches to zero as n goes to infinity.

Proof. First we prove the following:

$$\forall 1 \leq i \leq j \leq n, \frac{KNN(\mathbf{o},i)}{KNN(\mathbf{o},j)} = \left(\frac{i-1}{j-1}\right)^{\frac{1}{d}} \tag{A.6}$$

where $KNN(\mathbf{o},i)$ and $KNN(\mathbf{o},j)$ being the expected values of the i^{th} and j^{th} nearest neighbors of the center object $\mathbf{o}$ in the hypersphere respectively.

For $\mathbf{o} \in \mathbb{R}^d$, let $\|\mathbf{o}\|$ denote its Euclidean norm. Let $E(\mathbf{o},r) = \{x : \|x-\mathbf{o}\| \leqslant r\} \subseteq \mathbb{R}^d$ be a hypersphere centered at $\mathbf{o}$ with radius r.

Let

$$V = \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}+1\right)} r^d.$$

denote the volume of the hypersphere.

Consider V_i be the average minimum volume of the hypersphere centered at $\mathbf{o}$ with radius r_i that contains the i^{th} nearest neighbor to object $\mathbf{o}$ and $KNN(\mathbf{o}, i)$ being expected distance of i^{th} nearest neighbors from object $\mathbf{o}$. Therefore we have $KNN(\mathbf{o}, i) = r_i$. Also in uniform distribution the ratio of the average volumes of such hyperspheres that contain the i^{th} and j^{th} nearest neighbors to object $\mathbf{o}$ is $\frac{V_i}{V_j} = \frac{i-1}{j-1}$. Therefore from above discussion we have:

$$\frac{V_i}{V_j} = (\frac{r_i}{r_j})^d \tag{A.7}$$

$$\frac{r_i}{r_j} = \frac{KNN(\mathbf{o}, i)}{KNN(\mathbf{o}, j)} = \left(\frac{i-1}{j-1}\right)^{\frac{1}{d}} \tag{A.8}$$

Thus we have:

$$\frac{1}{KNN(\mathbf{o}, i)} = \left(\frac{n-1}{i-1}\right)^{\frac{1}{d}} \frac{1}{KNN(\mathbf{o}, n)} \tag{A.9}$$

also we have $KNN(\mathbf{o}, n) \approx 1$ therefore,

$$\begin{aligned}
a_{pts}coredist(\mathbf{o}) &= \left(\frac{\sum_{\substack{o_i \in X \\ o_i \neq o}} \left(\frac{1}{d(\mathbf{o}, o_i)}\right)^d}{n-1}\right)^{-\frac{1}{d}} = \left(\frac{\sum_{i=2}^{n} \left(\frac{1}{KNN(\mathbf{o}, i)}\right)^d}{n-1}\right)^{-\frac{1}{d}} \\
&= \left(\frac{\sum_{i=2}^{n} \left(\frac{n-1}{i-1}^{\frac{1}{d}}\right)^d}{n-1}\right)^{-\frac{1}{d}} = \left((n-1)\frac{\sum_{i=2}^{n} \left(\frac{1}{i-1}\right)}{n-1}\right)^{-\frac{1}{d}} \\
&= \left(\sum_{i=2}^{n} \left(\frac{1}{i-1}\right)\right)^{-\frac{1}{d}} = (\ln(n-1) + \gamma + \epsilon)^{-\frac{1}{d}} \\
&\approx \ln(n)^{-\frac{1}{d}}
\end{aligned}$$

where $\gamma \approx 0.5772$ is the Euler-Mascheroni constant and $\varepsilon \approx \frac{1}{2n} - \frac{1}{12n^2} + \frac{1}{120n^4} < \frac{1}{2}$

which approaches to 0 as n goes to infinity. We used the properties of the harmonic series ($\sum_{i=1}^{n} \frac{1}{i}$) in the above proof. □

Proposition A.1.3. *For calculated $a_{pts}coredist(\mathbf{o})$ in Proposition A.1.2, we have:*

$$a_{pts}coredist(\mathbf{o}) \approx \ln(n)^{-\frac{1}{d}} \approx KNN(\mathbf{o}, j), \tag{A.10}$$

with j being the closest natural number to $\frac{n}{\ln(n)}$ and $KNN(\mathbf{o}, j)$ being the expected value of j^{th} nearest neighbor distance to object $\mathbf{o}$.

Proof. We know that $KNN(\mathbf{o}, n) \approx 1$, therefore considering Equation (A.6) we have:

$$\frac{KNN(\mathbf{o}, j)}{KNN(\mathbf{o}, n)} \approx \left(\frac{\frac{n}{\ln(n)} - 1}{n - 1}\right)^{\frac{1}{d}} \approx \left(\frac{\frac{n}{\ln(n)}}{n}\right)^{\frac{1}{d}} = \ln(n)^{-\frac{1}{d}} \tag{A.11}$$

thus we have, $a_{pts}coredist(\mathbf{o}) \approx KNN(\mathbf{o}, j)$. □

Proposition A.1.4. *If the dissimilarity measure in Proposition A.1.2 is Squared Euclidean distance the all points core distance of $\mathbf{o}$ is:*

$$a_{pts}\, coredist(\mathbf{o}) \approx (1.645 * n)^{-\frac{1}{d}} \tag{A.12}$$

Proof. If the dissimilarity measure is Squared Euclidean distance, we have:

$$\forall 1 \leq i \leq j \leq n, \frac{KNN(\mathbf{o}, i)}{KNN(\mathbf{o}, j)} = \left(\frac{i - 1}{j - 1}\right)^{\frac{2}{d}} \tag{A.13}$$

Therefore:

$$\frac{1}{KNN(\mathbf{o}, i)} = \left(\frac{n - 1}{i - 1}\right)^{\frac{2}{d}} \frac{1}{KNN(\mathbf{o}, n)} \tag{A.14}$$

Thus,

$$
\begin{aligned}
a_{pts}coredist(\mathbf{o}) &= \left(\frac{\sum_{\substack{o_i \in X \\ o_i \neq o}} \left(\frac{1}{d(\mathbf{o},o_i)} \right)^d}{n-1} \right)^{-\frac{1}{d}} = \left(\frac{\sum_{i=2}^{n} \left(\frac{1}{KNN(\mathbf{o},i)} \right)^d}{n-1} \right)^{-\frac{1}{d}} \\
&= \left(\frac{\sum_{i=2}^{n} \left((\frac{n-1}{i-1})^{\frac{2}{d}} \right)^d}{n-1} \right)^{-\frac{1}{d}} = \left(\frac{\sum_{i=2}^{n} \left(\frac{n-1}{i-1} \right)^2}{n-1} \right)^{-\frac{1}{d}} \\
&= \left((n-1)^2 \frac{\sum_{i=2}^{n} \left(\frac{1}{i-1} \right)^2}{n-1} \right)^{-\frac{1}{d}} = \left((n-1) \sum_{i=2}^{n} \left(\frac{1}{i-1} \right)^2 \right)^{-\frac{1}{d}} \\
&\approx (1.645 * n)^{-\frac{1}{d}}
\end{aligned}
$$

using well-known Basel problem it can be easily proved that:

$$1 \leq \sum_{i=1}^{n} \left(\frac{1}{i} \right)^2 < 1.645$$

For i = 5 the sum is equal 1.4636 and for i =10 the sum is equal to 1.5498 □

Proposition A.1.5. *For calculated $a_{pts}coredist(\mathbf{o})$ in Proposition A.1.4, we have:*

$$a_{pts}coredist(\mathbf{o}) \approx (1.645 * n)^{-\frac{1}{d}} \approx KNN(\mathbf{o}, j), \tag{A.15}$$

with j being the closest natural number to $\sqrt{}(n/1.645)$ and $KNN(\mathbf{o}, j)$ being the expected value of j^{th} nearest neighbor distance to object $\mathbf{o}$.

Proof. We know that $KNN(\mathbf{o}, n) \approx 1$, therefore, considering Equation (A.13), we have:

$$\begin{aligned}\frac{KNN\,(\mathbf{o},j)}{KNN(\mathbf{o},n)} &\approx \left(\frac{\sqrt{\frac{n}{1.645}}-1}{n-1}\right)^{\frac{2}{d}} \\ &\approx \left(\frac{\sqrt{\frac{n}{1.645}}}{n}\right)^{\frac{2}{d}} \\ &\approx (1.645*n)^{-\frac{1}{d}}\end{aligned}$$

thus we have, $a_{pts}coredist(\mathbf{o}) \approx KNN(\mathbf{o},j)$. □

Proposition A.1.6. *For the core distances of object* **o** *calculated in Propositions A.1.2 and A.1.4, we have:*

$$a_{pts}coredist(\mathbf{o})_{Sq-Euclid.} \leq a_{pts}coredist(\mathbf{o})_{Euclid.} \tag{A.16}$$

Proof. For $i \in \{2,3,\ldots,n\}$, we have:

$$1 \leq \frac{n-1}{i-1}$$

Therefore:

$$\sum_{i=2}^{n}(\frac{1}{i-1}) \leq \sum_{i=2}^{n}(\frac{n-1}{i-1}*\frac{1}{i-1})$$

Thus:

$$\left(\sum_{i=2}^{n}\left(\frac{n-1}{i-1}*\frac{1}{i-1}\right)\right)^{-\frac{1}{d}} \leq \left(\sum_{i=2}^{n}\left(\frac{1}{i-1}\right)\right)^{-\frac{1}{d}}$$

Considering that $KNN(o,n) \approx 1$ and using Equation A.9 we can easily prove that: $a_{pts}coredist(\mathbf{o})_{Euclid.} = (\sum_{i=2}^{n}(\frac{1}{i-1}))^{-\frac{1}{d}}$ also using Equation A.14 we can easily prove that: $a_{pts}coredist(\mathbf{o})_{Sq-Euclid.} = \left(\sum_{i=2}^{n}\left(\frac{n-1}{i-1}*\frac{1}{i-1}\right)\right)^{-\frac{1}{d}}$, thus:

$$a_{pts}coredist(\mathbf{o})_{Sq-Euclid.} \leq a_{pts}coredist(\mathbf{o})_{Euclid.}$$

It can be easily seen that equality only holds for $n = 2$. □

Proposition A.1.7. *For the core distance of an object* $\mathbf{o}$ *calculated with respect to all other* $n-1$ *objects in data set* X*, we have:*

$$a_{pts}coredist(o) = \left(\frac{\sum_{\substack{o_i \in X \\ o_i \neq o}} \left(\frac{1}{d(\mathbf{o}, o_i)} \right)^d}{n-1} \right)^{-\frac{1}{d}} = b^P$$

where $P = -\frac{1}{d} \times ({log_b}^{1+\sum_{o_i \in X_1} b^{d \times ({log_b}^{d(\mathbf{o},o_c)} - {log_b}^{d(\mathbf{o},o_i)})}} - {log_b}^{n-1}) + {log_b}^{d(\mathbf{o},o_c)}$*,* o_c *can be any arbitrary object in* X *different than* $\mathbf{o}$*,* $X_1 = X - \{o, o_c\}$ *and* b *can be any Real number greater than 1.*

Proof. Using log properties for $0 < a_i$ we can easily prove the following:

$${log_b}^{\sum_{i=1}^{n} a_i} = {log_b}^{a_1} + {log_b}^{1+\sum_{i=2}^{n} b^{({log_b}^{a_i} - {log_b}^{a_1})}} \tag{A.17}$$

also we have:

$$a_{pts}coredist(o) = b^{{log_b}^{a_{pts}coredist(o)}} \tag{A.18}$$

we can also calculate the following:

$${log_b}^{a_{pts}coredist(o)} = -\frac{1}{d} \times ({log_b}^{\sum_{o_i \in X - \{o\}} \left(\frac{1}{d(\mathbf{o}, o_i)} \right)^d} - {log_b}^{n-1}) \tag{A.19}$$

considering equation A.17 we have:

$$\begin{aligned} log_b{}^{\sum_{o_i \in X-\{o\}} \left(\frac{1}{d(\mathbf{o},o_i)}\right)^d} =& log_b{}^{(\frac{1}{d(\mathbf{o},o_c)})^d} + log_b{}^{1+\sum_{o_i \in X_1} b^{(log_b{}^{(\frac{1}{d(\mathbf{o},o_i)})^d} - log_b{}^{(\frac{1}{d(\mathbf{o},o_c)})^d})}} \\ =& d \times log_b{}^{(\frac{1}{d(\mathbf{o},o_c)})} + log_b{}^{1+\sum_{o_i \in X_1} b^{d \times (log_b{}^{(\frac{1}{d(\mathbf{o},o_i)})} - log_b{}^{(\frac{1}{d(\mathbf{o},o_c)})})}} \\ =& log_b{}^{1+\sum_{o_i \in X_1} b^{d \times (log_b{}^{d(\mathbf{o},o_c)} - log_b{}^{d(\mathbf{o},o_i)})}} - d \times log_b{}^{(d(\mathbf{o},o_c))} \end{aligned} \tag{A.20}$$

where $X_1 = X - \{o, o_c\}$, therefore by substituting the value obtained in Equation A.20 in Equation A.19 we have:

$$\begin{aligned} log_b{}^{a_{pts}coredist(o)} =& -\frac{1}{d} \times (log_b{}^{1+\sum_{o_i \in X_1} b^{d \times (log_b{}^{d(\mathbf{o},o_c)} - log_b{}^{d(\mathbf{o},o_i)})}} \\ & - d \times log_b{}^{(d(\mathbf{o},o_c))} - log_b{}^{n-1}) \\ =& -\frac{1}{d} \times (log_b{}^{1+\sum_{o_i \in X_1} b^{d \times (log_b{}^{d(\mathbf{o},o_c)} - log_b{}^{d(\mathbf{o},o_i)})}} - log_b{}^{n-1}) \\ & + log_b{}^{(d(\mathbf{o},o_c))} \end{aligned} \tag{A.21}$$

considering Equations A.18 and A.21 we have:

$$a_{pts}coredist(o) = b^P \tag{A.22}$$

where $P = -\frac{1}{d} \times (log_b{}^{1+\sum_{o_i \in X_1} b^{d \times (log_b{}^{d(\mathbf{o},o_c)} - log_b{}^{d(\mathbf{o},o_i)})}} - log_b{}^{n-1}) + log_b{}^{d(\mathbf{o},o_c)}$, o_c can be any arbitrary object in X different than $\mathbf{o}$, $X_1 = X - \{o, o_c\}$ and b can be any Real number greater than 1.

□

Proposition A.1.8. *Consider two clusters C_i and C_j, in a hierarchy. The relative stability of clusters C_i and C_j is not dependent on parameter r_{shave} and can be computed using the following formula:*

$$RelativeStability(C_i, C_j) = \frac{Stab(C_i)}{Stab(C_j)} = \log \frac{\frac{n_{c_i}{}^{s-1}}{n_{c_i}{}^{e}}}{\frac{n_{c_j}{}^{s-1}}{n_{c_j}{}^{e}}}$$

Proof. We have:

$$Stab(C) = \frac{log(n_c^e) - log(n_c^{s-1})}{log(1 - r_{shave})}$$

thus,

$$\begin{aligned} Stability(C_i, C_j) =& \frac{Stab(C_i)}{Stab(C_j)} \\ =& \frac{log(n_{c_i}^e) - log(n_{c_i}^{s-1})}{log(n_{c_j}^e) - log(n_{c_j}^{s-1})} \\ =& \frac{log\frac{n_{c_i}^e}{n_{c_i}^{s-1}}}{log\frac{n_{c_j}^e}{n_{c_j}^{s-1}}} = \log_{\frac{n_{c_j}^e}{n_{c_j}^{s-1}}} \frac{n_{c_i}^e}{n_{c_i}^{s-1}} \end{aligned}$$

□