CoExDBSCAN: Density-based Clustering with Constrained Expansion

Benjamin Ertl[1][a], Jörg Meyer[1][b], Matthias Schneider[2] and Achim Streit[1][c]

[1] *Steinbuch Centre for Computing (SCC), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany*

[2] *Institute for Meteorology and Climate Research (IMK-ASF), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany*

Keywords: Data Mining, Machine Learning, Pattern Recognition, Clustering, Correlation Clustering, Constrained Clustering, DBSCAN, Spatio-temporal Data, Climate Research.

Abstract: Full space clustering methods suffer the curse of dimensionality, for example points tend to become equidistant from one another as the dimensionality increases. Subspace clustering and correlation clustering algorithms overcome these issues, but still face challenges when data points have complex relations or clusters overlap. In these cases, clustering with constraints can improve the clustering results, by including a priori knowledge into the clustering process. This article proposes a new clustering algorithm CoExDBSCAN, density-based clustering with constrained expansion, which combines traditional, density-based clustering with techniques from subspace, correlation and constrained clustering. The proposed algorithm uses DBSCAN to find density-connected clusters in a defined subspace of features and restricts the expansion of clusters to a priori constraints. We provide verification and runtime analysis of the algorithm on a synthetic dataset and experimental evaluation on a climatology dataset of satellite observations. The experimental dataset demonstrates, that our algorithm is especially suited for spatio-temporal data, where one subspace of features defines the spatial extent of the data and another correlations between features.

1 INTRODUCTION

Mining datasets has become more challenging with the increasing amount of high dimensional data that is available today through new technologies, higher processing power, bigger storage capacities and data-driven research. Finding clusters in such datasets can reveal interesting patterns and dependencies often caused by complex correlations. However, traditional full space clustering algorithms suffer the curse of dimensionality, for example points tend to become equidistant from one another as the dimensionality increases (Friedman, 1994). For this purpose, different subspace and correlation clustering algorithms have been proposed, which extend traditional clustering algorithms to detect correlations in subsets of features (Agrawal et al., 1998) (Aggarwal and Yu, 2000). While some correlation algorithms are able to find arbitrarily oriented subspace clusters, for example CASH (Achtert et al., 2008), or can identify local subgroups of data objects sharing a uniform but arbitrarily complex correlation, for example 4C (Böhm et al., 2004a), these algorithms still face challenges for instance with overlapping clusters or uncorrelated complex relations between features. For this reason, there has been a growing interest in semi-supervised clustering methods, in particular constrained clustering, where additional information or domain knowledge is included into the clustering process (Pourrajabi et al., 2014) (Basu et al., 2008) (Dinler and Tural, 2016). Incorporating a priori knowledge into the clustering process can improve the clustering results, lead to better performance and align the outcome of the cluster analysis with knowledge of domain experts. In this paper, we propose a new density-based clustering algorithm with constrained cluster expansion, CoExDBSCAN, that combines different techniques from subspace, correlation and constrained clustering. The proposed algorithm uses DBSCAN (Ester et al., 1996) to find density-connected clusters in a defined subspace of features and restricts the expansion of clusters to a priori constraints. The validation of the algorithm on an experimental, real-world dataset demonstrates, that our algorithm is especially suited for spatio-temporal data, where one subspace of features defines the spatial extent of the data and another correlations between features.

[a] https://orcid.org/0000-0003-1431-2243
[b] https://orcid.org/0000-0003-0861-8481
[c] https://orcid.org/0000-0002-5065-469X

Ertl, B., Meyer, J., Schneider, M. and Streit, A.
CoExDBSCAN: Density-based Clustering with Constrained Expansion.
DOI: 10.5220/0010131201040115
In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Volume 1: KDIR*, pages 104-115
ISBN: 978-989-758-474-9

Specifically, our contributions with this work can be summarized as follows:

- We introduce two user-defined parameters to the original DBSCAN algorithm. One to define the dimensions of the subspace to be used to discover density-based clusters, and one to define the dimensions of the subspace to be used to apply constraints to the cluster expansion of DBSCAN.
- We modify the cluster expansion step in the original DBSCAN algorithm to be restricted to user-defined constraints.
- We propose a generic constraint to discover correlated structures in large datasets.
- Finally, we provide results of thorough experimental studies on synthetic and real-world datasets and demonstrate, that our algorithm is especially suited for spatio-temporal data, where one subspace of features defines the spatial extent of the data and another correlations between features.

The reminder of the paper is organized as follows: Section 2 compares our proposed algorithm to related work while Section 3 presents the proposed algorithm in detail. The evaluation is provided in Section 4 with verification and runtime analysis of the algorithm on a synthetic dataset and experimental evaluation on a climatology dataset of satellite observations. In Section 5 we give a discussion on the results while Section 6 provides the conclusions and outlooks.

All datasets together with the code for this paper is publicly available[1].

2 RELATED WORK

Numerous clustering algorithms have been developed and studied over time. A well received survey is provided by Anil K. Jain in his article ”Data Clustering: 50 Years Beyond K-Means” (Jain, 2010). In this paper we focus on most related and relevant work in the areas of density-based clustering, correlation clustering and constrained clustering.

Martin Ester et al. presented the DBSCAN algorithm in 1996 as a density-based clustering algorithm for discovering clusters in large spatial databases with noise (Ester et al., 1996). The authors introduced a until then new notion of clusters, based on the density of point neighbourhoods. The algorithm considers data point by data point. If the distance from an initial point to at least a *minPts* defined number of other points is smaller than a defined distance ε, these points form a cluster. The initial point is considered a core point and the remaining points the ε-neighbourhood of that point. If a cluster can be formed, this cluster is expanded by applying the initial step to all points in the ε-neighbourhood. If the initial point is not a core point, the point is considered to be a noise point and the algorithm moves to the next point in the dataset. Noise points can become border points and therefore be associated to a different cluster later on, if they are density-reachable from some other data point. The two parameters *minPts* and ε determine the outcome of the DBSCAN algorithm. While the purpose of *minPts* is to smooth the density estimate and is recommended to be chosen according to the dimensionality of the dataset, the radius parameter ε depends on the distance function, and should ideally be based on domain knowledge (Schubert et al., 2017). Schubert et al. discuss further the advantages and disadvantages of DBSCAN. Most notably, Schubert (Schubert et al., 2017) states, that DBSCAN continues to be relevant even for high-dimensional data, but becomes difficult to use:

> ”Independent of the algorithm, the parameter ε of DBSCAN becomes hard to choose in high- dimensional data due to the loss of contrast in distances (Beyer et al., 1999; Houle et al., 2010; Zimek et al., 2012). Irrespective of the index, it therefore becomes difficult to use DBSCAN in high-dimensional data because of parameterization; other algorithms such as OPTICS and HDBSCAN*, that do not require the ε parameter are easier to use, but still suffer from high dimensionality.”

The OPTICS (Ankerst et al., 1999) and HDBSCAN* (Campello et al., 2013) algorithm mentioned by Schubert are examples of DBSCAN variants that focus on finding hierarchical clustering results (Schubert et al., 2017). Although our work is based on the original DBSCAN algorithm, it is not restricted to it and can be used in combination with any variant where the cluster expansion step can be restricted to user-defined constraints.

To overcome the issues of conventional, full space clustering methods in high-dimensional data, algorithms in the area of subspace clustering and correlation clustering have attracted more and more attention recently (Achtert et al., 2008). For example CLIQUE (Agrawal et al., 1998) for axis-parallel subspaces or ORCLUS (Aggarwal and Yu, 2000) and CASH (Achtert et al., 2008) for arbitrarily oriented subspaces. Elke Achtert et al. introduced the CASH algorithm as an efficient and effective method to find arbitrarily oriented subspace clusters. The main idea of the algorithm is to transform every data point from

[1]https://github.com/bertl4398/kdir2020

data space to parameter space, the space of all possible subspaces, by using the ideas of the Hough transformation (Hough, 1962; Duda and Hart, 1972). Every data point in data space is mapped onto the corresponding sinusoidal curve in parameter space. Hypercuboids in parameter space with many intersecting sinusoidal curves indicate points in data space that are located on, or near, a common hyperplane, and therefore are considered to form a subspace cluster (Achtert et al., 2008). According to Achtert et al.

> ”[CASH is able to] find subspace clusters of different dimensionality even if they are sparse or are intersected by other clusters within a noisy environment.”

Since our research interest and real-world data belongs manly into this categorization of data, we choose to compare our clustering results to this algorithm as well. Moreover, an open source implementation within the ELKI (Schubert and Zimek, 2019) data mining software is available and has been used in our experiments. CASH is also outperforming other correlation clustering algorithms such as ORCLUS or 4C (Böhm et al., 2004b) on datasets with highly overlapping clusters (Achtert et al., 2008), which is the case for our synthetic and real-world datasets.

Incorporating additional information or domain knowledge about the underlying cluster structure of the data is known as constrained clustering and has been the subject of extensive research recently (Dinler and Tural, 2016). Wagstaff and Cardie (Wagstaff and Cardie, 2000) introduced the notion of using constraints that express information about the underlying class structure in the clustering process by considering two general types of constraints, called instance level constraints; (1) *must-link* constraints that specify that two instances have to be in the same cluster and (2) *cannot-link* constraints that specify, that two instances cannot be in the same cluster. The most relevant related work to our paper in this area is the C-DBSCAN algorithm by Ruiz et al. (Ruiz et al., 2007), a density-based clustering algorithm with constraints. C-DBSCAN extends DBSCAN in three steps:

1. Partitioning the data space into dense partitions by applying a k-d tree (Bentley, 1975)
2. Creating local clusters under cannot-link constraints
3. Merging local clusters under must-link and cannot-link constraints

Ruiz et al. chose a random percentage of points under the must-link constraint and derived the cannot-link constraint interdependently. The authors could demonstrate, that even those randomly chosen constraints improve the clustering quality substantially, notably on datasets where the original DBSCAN performs poorly (Ruiz et al., 2007). Compared to our approach, we do not follow the method of instance level constraints expressed as must-link and cannot-link constraints. Our modification to DBSCAN restricts the cluster expansion to user-defined constraints, which has been proven to be very flexible and is explained in detail in the next section.

3 ALGORITHM

First, we are going to recap the main definitions of the original DBSCAN algorithm by Ester et al. before giving a detailed description of our proposed extensions.

3.1 DBSCAN Recap

In the original paper from Martin Ester et al. presented at the International Conference on Knowledge Discovery and Data Mining (KDD) in 1996, Ester gives six main definitions essential for the DBSCAN algorithm (Ester et al., 1996), recapitulated in the following.

Definition 1. ε-neighbourhood of a Point.
Let *DB* be a database of points. The ε-*neighbourhood* of a point p, denoted by $N_\varepsilon(p)$, is defined by

$$N_\varepsilon(p) = \{q \in DB | dist(p,q) \leq \varepsilon\}$$

Definition 2. Directly Density-reachable.
A point p is *directly density-reachable* from a point q wrt. ε and *minPts* if

1. $p \in N_\varepsilon(q)$ and
2. $|N_\varepsilon(q) \geq minPts|$ (core point condition).

Definition 3. Density-reachable.
A point p is *density-reachable* from a point q wrt. ε and *minPts* if there is a chain of points $p_1,...,p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

Definition 4. Density-connected.
A point p is *density-connected* to a point q wrt. ε and *minPts* if there is a point o such that both, p and q are *density-reachable* from o wrt. ε and *minPts*.

Definition 5. Cluster.
A *cluster C* wrt. ε and *minPts* is a non-empty subset of *DB* satisfying the following conditions:

1. $\forall p,q$: if $p \in C$ and q is *density-reachable* from p wrt. ε and *minPts*, then $q \in C$. (Maximality)
2. $\forall p,q \in C$: p is *density-connected* to q wrt. ε and *minPts*. (Connectivity)

Definition 6. Noise.
Let $C_1,...,C_k$ be the clusters of the database *DB* wrt. parameters ε_i and $minPts_i$, $i = 1,...,k$. Then we define the *noise* as the set of points in the database *DB* not belonging to any cluster C_i, i.e. $noise = \{p \in DB | \forall i : p \notin C_i\}$

In a nutshell, clusters extracted by the DBSCAN algorithm comprise points that have at least a specific number of other points (*minPts*) within a specific distance (ε) or otherwise are considered to be noise.

While DBSCAN is able to find clusters of arbitrary shape, one problem with the global density parameter ε is, that DBSCAN can not identify clusters with different densities. Although metrics exist to determine good, global ε and *minPts* parameters, for example analysing the *k-distance* graph as proposed in the original paper (Ester et al., 1996), finding good, global parameters becomes even more challenging for high-dimensional data. An intuitively accessible example, specifically why the Euclidean distance does not work well in high-dimensional data, is given by Ertöz et al.(Ertöz et al., 2003), who introduced a new notion of similarity and density for their shared nearest neighbour clustering algorithm (SNN). Another problem with DBSCAN arises with datasets that contain overlapping clusters. As long as points are density-connected, these points will end up in the same cluster, with the rare exception of a point being a border point in two clusters. For overlapping clusters, this means, that DBSCAN will merge them or will consider the lesser dense true cluster points as noise.

Our solution addresses these problems by modifying the original DBSCAN algorithm as described in the following section.

3.2 CoExDBSCAN

Our density-based clustering algorithm with constrained expansion (CoExDBSCAN) modifies the original DBSCAN clustering algorithm in two ways. First, we introduce a user-defined parameter to define the dimensions of the (sub)space to be used to discover density-based clusters. Second, we restrict the cluster expansion step in the DBSCAN algorithm to user-defined constraints, applied to a user-defined (sub)space of the dataset.

According to these extensions, we can redefine the ε-neighbourhood definition from the original DBSCAN algorithm introduced in the previous section, Definition 1.

Definition 7. CoExDBSCAN ε-neighbourhood of a Point.
Let *DB* be a database of points. The ε-*neighbourhood* of a point *p*, denoted by $N_\varepsilon(p)$, is defined by

$$N_\varepsilon(p) = \{q \in DB | dist(p_S, q_S) \leq \varepsilon \\ \wedge constraints(p_R, q_R)\}$$

where p_S, q_S are the subspace representations of point *p* and *q* of the user-defined spatial subspace *S*, p_R, q_R are the subspace representations of point *p* and *q* of the user-defined constraint subspace *R* and the $constraints$ function evaluates $true$ for each constraint T_i in a user-defined set of constraints $T = \{T_1, T_2, ..., T_m\}$.

The pseudo code representation is given in Algorithm 1 and has been adopted from the pseudo code representation by Schubert et al. (Schubert et al., 2017) of the original, sequential DBSCAN algorithm. Modifications to the algorithm are coloured red and marked by star (*). Each object in the database that has not been processed already is labeled as noise, if the core point condition is violated, see Definition 2, i.e. there are not at least *minPts* objects in the ε-neighbourhood of the object under consideration. Otherwise, the object is a core point and forms a new cluster, while iteratively expanding and adding the core point neighbours to the cluster.

Our first modification is in the `RangeQuery` function in line 3 and line 13 of Algorithm 1 that accepts a user-defined parameter *sDim*, allowing to define the dimensions of the space for the range query. In the original DBSCAN algorithm, the range query is always executed on the full space of the dataset, unless the algorithm operates on a precomputed distance matrix that takes these restrictions into account. However, the parametrization of the spatial dimensions makes this restriction more explicit and easier accessible to the user. Especially for data with explicit spatial dimensions, excluding certain dimensions or all non-spatial dimensions from the range query can improve the quality of the clustering, as we will demonstrate in the experimental evaluation of the algorithm.

Our second modification is in the expansion step. While the original DBSCAN algorithm expands a cluster by starting at one core point and adding all of its neighbours to a set of seeds that are iteratively expand on if they satisfy the core point condition itself, we allow for additional user-defined constraints to be considered before adding these points as seeds. Therefore, even if the neighbour points satisfy the core point condition as expressed in line 15 of the algorithm, if the `PointConstraint` function can not be satisfied in line 17, they will not be added to the set of seeds, i.e. the algorithm will not expand on their respective neighbours. The user can define one or multiple constraints (*cFunc*) and the dimensions

($cDim$) that the constraints should be applied to. This approach can significantly improve the quality of the clustering and allows to incorporate a priori knowledge into the clustering process expressed in the form of sub- or full-space constraints.

Algorithm 1: Pseudo Code of CoExDBSCAN Algorithm.

```
input  : database $DB$
input  : radius $\varepsilon$
input  : density threshold $minPts$
input  : distance function $dist$
input  : spatial dimensions $sDim$ *
input  : user-defined constraints $cFunc$ *
input  : constraint dimensions $cDim$ *
output: point labels $label$ initially $undefined$
1  foreach $point\ p\ in\ database\ DB$ do
2      if $label(p) \neq undefined$ then continue;
3      Neighbours $N \leftarrow$
         RangeQuery($DB, dist, sdim, p, \varepsilon$) *;
4      if $|N| < minPts$ then
5          $label(p) \leftarrow Noise$;
6          continue;
7      $c \leftarrow$ next cluster label;
8      $label(p) \leftarrow c$;
9      Seed set $S \leftarrow N \setminus \{p\}$;
10     foreach $q\ in\ S$ do
11         if $label(q) = Noise$ then
             $label(q) \leftarrow c$;
12         if $label(q) \neq undefined$ then
             continue;
13         Neighbours $N \leftarrow$
             RangeQuery($DB, dist, sdim, q, \varepsilon$) *;
14         $label(q) \leftarrow c$;
15         if $|N| < minPts$ then continue;
16         foreach $s\ in\ N$ do
17             if PointConstraint
                 ($cFunc, cDim, s$) $is\ false$ then
                 continue *;
18             $S \leftarrow S \cup s$;
```

The PointConstraint function in line 17 of the pseudo code takes a set of user-defined constraints in the form of functions and returns true if all of the constraints can be satisfied or false otherwise. This behaviour can be relaxed, for example by applying a threshold of number of constraints that needs to be satisfied instead of the all-true behaviour. Also finding good constraints is a challenging task, depending on the data to analyse, the analysis to be conducted and the expected outcome. Moreover, it is up to the user to define non-mutually exclusive and sensible constraints. However, constraining the DBSCAN cluster expansion introduces a variety of interesting studies that can provide valuable insights into different datasets, as we will demonstrate in the next section.

4 EVALUATION

In this section, we provide the runtime analysis of our proposed algorithm first and verify the improved quality of the clustering results in the subsequent sections. For the runtime analysis, we consider the average complexity of the original DBSCAN algorithm with the additional complexity of user-defined constraints.

We compared the results of our algorithm with the results from DBSCAN and CASH for different existing popular reference datasets, for example the Iris flower dataset (Fisher, 1936) and the artificial datasets used for the verification of the CURE algorithm (Guha et al., 2001), but chose to base the presented verification in this paper on our own generated synthetic dataset. All results, including the results for existing popular reference datasets, can be found in the GitHub repository[2]. Generating our own synthetic dataset allows us to fully control the properties of the clusters, so that we can create a dataset that is especially challenging for density-based clustering methods, and proofs to be very challenging for subspace and correlation clustering methods as well. In addition, we provide verification of the algorithm on a real-world dataset within the domain of spatio-temporal data and climate research.

Since the true labels are known for the synthetic dataset, we use the Rand index adjusted for chance (Rand, 1971; Hubert and Arabie, 1985) to evaluate our clustering results. The Rand index is a measure of similarity between two data clusterings and can be computed as following (Rand, 1971):

Definition 8. Rand Index.
Given a set of n elements $S = \{o_1, \ldots, o_n\}$, a partition $X = \{X_1, \ldots, X_r\}$ of S into r subsets and a partition $Y = \{Y_1, \ldots, Y_s\}$ of S into s subsets, the Rand index is:

$$R = \frac{a+b}{n(n-1)/2} \qquad (1)$$

with a, the number of pairs of elements in S that are in the same subset in X and Y, with b, the number of pairs of elements in S that are in the same subset in X and in different subsets in Y, and the total number of pairs $\binom{n}{2}$ in the denominator.

[2]https://github.com/bertl4398/kdir2020

The adjusted Rand index, which is bounded above by 1 and takes on the value of 0 when the index equals its expected value $\mathbf{E}(R)$, can be expressed in general form of an index corrected for chance the following (Hubert and Arabie, 1985):

Definition 9. Adjusted Rand Index.

$$ARI = \frac{R - \mathbf{E}(R)}{max(R) - \mathbf{E}(R)} \quad (2)$$

Moreover, we use the clustering accuracy (ACC) to evaluate our clustering results, which finds the best match between the true labels and the cluster labels. The greater the clustering accuracy, the better the clustering performance (Role et al., 2019).

Definition 10. Clustering Accuracy (ACC).

$$ACC(y, \hat{y}) = \max_{perm \in P} \frac{1}{n} \sum_{i=0}^{n-1} 1(perm(\hat{y}_i) = y_i) \quad (3)$$

where P is the set of all permutations in $[1;K]$ where K is the number of clusters. The set of all permutations can be efficiently computed using the Hungarian algorithm (Papadimitriou and Steiglitz, 1998).

However, our experiments have shown, that "good" clusterings, in terms of number of clusters and correct labels, often have a lower adjusted Rand index than "bad" clusterings. Therefore, in addition to the adjusted Rand index and clustering accuracy, we evaluate our cluster results based on the total number of clusters assigned, the minimum and maximum clustered data points and to some extend the standard deviation of number of points per cluster, as well as with the aid of visual analysis. For our real-world dataset, we assess the quality of the clustering primarily according to expert opinion.

4.1 Runtime Analysis

The average runtime complexity of the original DBSCAN algorithm is $O(n \log n)$ (Ester et al., 1996). The authors argue, that distance queries can be supported efficiently by spatial access methods such as *R*-trees*, that have the height of $O(\log n)$, and because for each of n points only one query has to be executed, the average run time complexity is consequently $O(n \log n)$. Schubert et al. (Schubert et al., 2017) state, that the DBSCAN runtime complexity can be $\Theta(n^2 \cdot D)$, with cost D of computing the distance of two points, if implementing the range query with a linear scan. In general however:

"[...] DBSCAN remains a method of choice even for large n because many alternatives are in $\Theta(n^2)$ or $\Theta(n^3)$. [...] In the general case of arbitrary non-metric distance measures, the worst case remains $O(n^2 \cdot D)$ [...]"

Introducing a set of constraints $T = \{T_1, T_2, ..., T_m\}$ to the expansion step of the DBSCAN algorithm adds the complexity of $O(n \cdot max(T))$ to check the set of constraints for each point. The complexity of constraints can vary greatly, for example from hash table searches with average time complexity $O(1)$ (Cormen et al., 2009) to linear regression complexity $O(w^2 n + w^3)$ for n number of observations and w number of weights (Mohri et al., 2012), as in our demonstrated examples in the following sections. In total, the runtime complexity of CoExDBSCAN, depending on the user-defined constraints, is therefore on average the runtime complexity of DBSCAN plus the maximum complexity of the user-defined constraints, $O(n \cdot max(T) + n \cdot \log n)$.

4.2 Verification

Our synthetic dataset contains 3,000 points with three dimensions and three classes, 1,000 points per class. Table 1 lists the generation method and interval range as well as the linear dependencies of the variables. Figure 1a) illustrates the overlapping nature and linear dependencies of the three classes.

Table 1: Value range and dependencies for the synthetic dataset.

Points	x	y	z
1,000	evenly [0,1]	$-0.5x + 0.2 + \xi$	$-0.5x + 0.2 + \xi$
1,000	uniform [0,1)	uniform [0,1)	$0.1x + 0.1y$
1,000	uniform [0,1)	uniform [0,1)	$0.4x + 0.2y$

The values for the x and y variables of cluster 0, blue color in Figure 1a), are generated by sampling the random uniform distribution in the half-open interval $[0, 1)$; the values for the z variable are computed using the linear equation $0.1x + 0.1y$. For cluster 1, orange color in the figure, the values for the x and y variables are generated also by sampling the random uniform distribution in the half-open interval $[0, 1)$; the values for the z variable are computed using the linear equation $0.4x + 0.2y$. The green coloured cluster 2 in the figure is generated by evenly spaced x values in the closed interval $[0, 1]$ and the values for the y and z variables following the linear equation $-0.5x + 0.2 + \xi$, where ξ is some random variation with $\xi \sim \mathcal{N}(0, 0.01)$. This dataset poses a challenge to all clustering algorithms that we evaluated for this study. First, the clusters have different densities, with a very dense, line shaped cluster and two looser, plane shaped clusters. Second, all clusters are overlapping with some proportions. The plane shaped clusters have an overlapping edge, which is crossed by parts of the line shaped cluster.

We implemented the CoExDBSCAN algorithm

in Python and conducted our experiments with the scikit-learn (Pedregosa et al., 2011) machine learning package implemented in Python and the ELKI (Schubert and Zimek, 2019) data mining software written in Java. Our first objective was to find the best DBSCAN clustering according to the adjusted Rand index and the most accurate clustering, i.e. three clusters with the same amount of data points each and the fewest noise. In order to find suitable clusterings, we conducted a grid search for DBSCAN with ε in the range of $[0.01, 0.2]$ with a step size of 0.01 and *minPts* in the range of $[3, 100]$ with a step size of 1.

The DBSCAN clustering result with the highest adjusted Rand index for the parameters $\varepsilon = 0.03$ and $minPts = 39$ in three-dimensional space has only one single cluster, not depicted here. The line shaped, dense cluster has been identified almost perfectly with 1,018 data points in the cluster. However, with the specified ε radius and number of minimum ε-neighbourhood points, the algorithm was not able to expand into the plane shaped point structures. Although the adjusted Rand index ($ARI = 0.562$) is the highest in the explored parameter space, according to the accuracy and qualitative assessment, the clustering result is worse than the qualitative best clustering result. Figure 1b) shows the qualitative best DBSCAN clustering result, with three clusters, fewest noise points and the maximum amount of points in each cluster. It becomes apparent, that with a higher radius ε, the DBSCAN algorithm is now able to expand into the whole dataset, while the number of ε-neighbourhood points determines the amount of noise and closeness of the clusters. A lower number of *minPts* relaxes the condition on the cluster expansion up to a point, where all data can be clustered into one single cluster. Whereas a higher number of *minPts* restricts the cluster expansion more, resulting in clusters that are further apart but also increases the number of noise points. With our proposed modification in the expansion step of the DBSCAN algorithm, we can keep the ε parameter at a value that allows to expand into the whole dataset and lower the number of *minPts* at the same time, while avoiding the degenerated case of one single cluster.

One advantage of our proposed algorithm is the flexibility of how users can provide constraints to the clustering process. In our verification experiments, we tested several constraints that include the a priori knowledge of correlated structures in the dataset. Experiments have shown, that a suitable constraint that expresses this information in a generic way is to put a threshold on the change of the mean squared error regression loss by including the next core point into the current set of cluster points. This constraint allows the algorithm to expand clusters on arbitrarily correlated structures and changing correlation up to a certain degree.

Another advantage over the original DBSCAN algorithm is the user-defined selection of the spatial dimensions and the constraint dimensions. We refer to those features of the dataset as spatial dimensions that are used to calculate the `RangeQuery`, i.e. the pairwise distance between data points. Likewise, we refer to the features of the dataset that are used in the `PointConstraint` function as constraint dimensions, see Algorithm 1. In our verification experiments, we chose to include all features as spatial dimensions and only the x and z features as correlation dimensions. The threshold for the change of the mean squared error regression loss has been set to $9 \cdot 10^{-6}$, according to empirical tests. The ε and *minPts* parameters have been determined by a grid search on the parameter space for ε in the interval $[0.01, 0.2]$ with a step size of 0.01 and *minPts* in the range of $[3, 100]$ with a step size of 1.

Figure 1c) shows the best qualitative result, with three clusters, fewest noise points and the maximum amount of points in each cluster. By comparing the CoExDBSCAN results visually to the original DBSCAN results, it is apparent, that the CoExDBSCAN clustering result better captures the inherent structure of the dataset. Apart from the overlapping area, the correlated data points in each of the three generated point samples have been assigned to three distinct clusters. It should be noted, that we are using a generic constraint here that includes only the information of some arbitrary correlated structures in the dataset and that even permits gradually changes to the linear regression of the cluster points. Still, the clustering of CoExDBSCAN compared to DBSCAN shows an improvement in our qualitative measures, although the algorithm was not able to separate the data points in the overlapping area. As we will discuss in Section 5, constraints more specific to the data could further improve the result, but may lead to overfitting the algorithm to this one particular dataset.

In addition, we compared our algorithm to the CASH algorithm (Böhm et al., 2004b). According to Achtert et al., CASH is significantly outperforming other correlation clustering algorithms, such as ORCLUS or 4C, on datasets with highly overlapping clusters in terms of robustness and effectiveness. CASH requires the user to specify three parameter. (1) The minimum number of sinusoidal curves that need to intersect a hypercuboid in parameter space to be considered a dense area, i.e. the minimum number of points in a cluster (*minPts*); (2) the maximal number of splits along a search path (*maxLevel*), i.e the

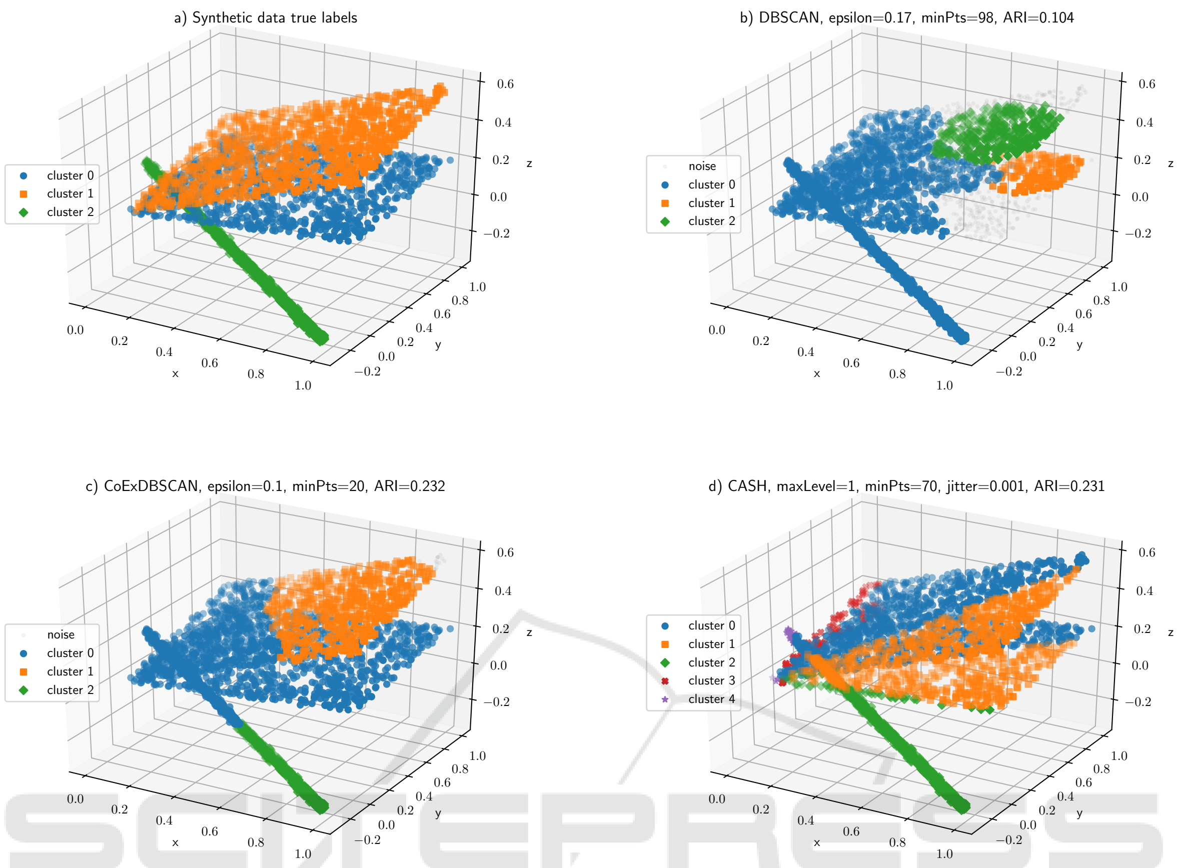

Figure 1: Clustering results for the synthetic data: a) original data in three dimensional space; best qualitative clustering results: b) DBSCAN, c) CoExDBSCAN, d) CASH.

maximal deviation from the hyperplane of the cluster in terms of orientation and jitter; and (3) the amount of *jitter*. We performed again a grid search on the parameter space for *minPts* in the interval of $[3, 100]$ with a step size of 1, for *maxLevel* in the range of $[1, 10]$ with a step size of 1 and *jitter* in the interval of $[0.001, 0.005]$ with a step size of 0.001. Figure 1d) illustrates the qualitative best clustering result for CASH on our synthetic dataset with the parameters $minPts = 70$, $maxLevel = 1$ and $jitter = 0.001$, with five clusters and the maximum amount of points in each cluster. The lowest number of clusters in the parameter space is five, and while the best qualitative results captures proportions of the correlated structures well, it performs poorer than CoExDBSCAN in terms of our evaluation metrics, which are summarized in Table 2. Table 2 lists the respective algorithm with the number of clusters identified, the minimum number of points in any cluster, the maximum number of points in any cluster and the number of noise points, i.e. points that were not assigned to any cluster. CoExDBSCAN outperforms the original DBSCAN algorithm and CASH in terms of the correct number of

Table 2: Summary of Best Qualitative Clustering Results for the Synthetic Dataset.

Algorithm	Clusters	ARI	ACC	Min	Max	Noise
CoExDBSCAN	**3**	**0.232**	**0.669**	**468**	**1,947**	**17**
DBSCAN	3	0.104	0.555	155	2,132	421
CASH	5	0.231	0.543	51	1,157	NA

clusters identified, the cluster accuracy, and therefore closest similarity to the true labels of the data. The adjusted Rand index provides an indifferent result in this comparison.

4.3 Real World Example

To demonstrate the significance of our algorithm for real-world data and provide additional verification, we applied DBSCAN and CoExDBSCAN to a dataset within the domain of spatio-temporal data and climate research. We chose data from this particular domain, since the development of the algorithm is part of an interdisciplinary research project between computer scientists and climate researchers, with the aim to develop methods and algorithms for data-driven climate data analysis. Our real-world dataset consists of spec-

Table 3: Summary statistics for the real-world dataset.

	lon	lat	H_2O	δD
count	38,734	38,734	38,734	38,734
mean	11.40	15.92	3749.84	-231.17
std	32.08	21.66	2057.17	66.15
min	-44.99	-25.00	526.53	-475.08
25%	-18.95	-3.01	2217.36	-279.89
50%	16.86	17.36	3301.84	-230.05
75%	38.80	35.65	4723.93	-179.05
max	59.99	50.00	14376.72	-70.47

tral data gathered from Metop-A and Metop-B satellites that have been processed for the water vapour H_2O mixing ratio and water isotopologue δD depletion for air masses at $5km$ height with most sensitivity. The water isotopologue in question is HDO, which differs only in the isotopic composition compared to H_2O. Isotopologues of atmospheric water vapour can make a significant contribution for a better understanding of atmospheric water transport, because different water transport pathways leave a distinctive isotopologue fingerprint (Schneider et al., 2017). The paired analysis of water vapour H_2O mixing ratio and water isotopologue δD depletion allows to identify different processes in the atmosphere, for example air mass mixing, precipitation and condensation (Noone, 2012). Our goal is to develop data-driven methods to identify and analyse such processes for a better understanding of atmospheric water transport and to evaluate the moisture pathways as simulated by different state-of-the-art atmospheric models. These methods and algorithms have to scale and cope with the amount of data that is continuously produced by the remote sensing instruments onboard the satellites, where global measurements of our data for one year aggregate to 20 Terabyte.

Our dataset in this example comprises 38,734 satellite observations with the geographical coordinates longitude (*lon*) and latitude (*lat*) and the spectral data processed for the water vapour H_2O mixing ratio and water isotopologue δD depletion. This dataset corresponds to measurements for one global morning overpass of both satellites in a region of interest over the Atlantic and West Africa. The measurements have been filtered for cloud free conditions and highest sensitivity; their summary statistics are listed in Table 3. We applied different transformations to align the value range of each variable according to domain experts, given in Table 4. After scaling the data we executed multiple runs with DBSCAN and CoExDBSCAN and compared the clustering results.

Figure 2 illustrates a qualitative good example using DBSCAN, with parameters $\varepsilon = 10$ and $minPts = 200$. DBSCAN has identified three clusters that are

Table 4: Transformation of Features.

variable	transformed variable
lon	$lon_{transformed} = lon \cdot cos(lat \frac{\pi}{180})/0.5$
lat	$lat_{transformed} = lat/0.5$
H_2O	$H_2O_{transformed} = log(H_2O)/0.2$
δD	$\delta D_{transformed} = \delta D/20$

dense in the full space of the scaled variables lat, lon, H_2O and δD. Moreover, two of the identified clusters (green and orange color in the figure) indicate some correlation in the $\{log(H_2O), \delta D\}$ and $\{H_2O, H_2O \cdot \delta D\}$ value space, while at the same time are located geographically close. Using the same parameters $\varepsilon = 10$ and $minPts = 200$ with CoExDBSCAN, we can additionally expresses a priori knowledge of correlated structures in the dataset by providing user-defined constraints to the expansion of clusters, as well as define the spatial and constraint dimensions. In the same way as in the verification tests with the synthetic dataset, we used again a threshold for the change of the mean squared error regression loss as constraint, which has been set to $1 \cdot 10^{-5}$ in this example and has been empirically determined. As spatial dimensions we chose the geographical coordinate variables longitude (*lon*) and latitude (*lat*), and as constraint dimensions we chose the remaining two variables H_2O and δD. Figure 3 shows the results using the CoExDBSCAN algorithm. Compared to DBSCAN we have identified a significant higher number of clusters, while keeping the same ε radius and *minPts* neighbourhood points. All clusters are geographically close and correlated in either the $\{log(H_2O), \delta D\}$ or $\{H_2O, H_2O \cdot \delta D\}$ value space. By incorporating a threshold for the change of the mean squared error regression loss as constraint in the cluster expansion step and explicitly defining spatial and constraint dimension, we can keep the DBSCAN parameters that allow to explore the full dataset, but still be able to explore more fine-grained structures, even with highly overlapping data points. This level of granularity is very important to distinguish and to identify the different processes in the atmosphere, which emphasizes the value of the algorithm for data analysis in this particular domain.

5 DISCUSSION

The main concept of our presented CoExDBSCAN algorithm, to allow users to constrain the expansion of clusters in specific subspaces of the data, can significantly improve the clustering results, as demonstrated in sections 4.2 and 4.3 for synthetic and real-world data. However, finding and expressing suitable con-

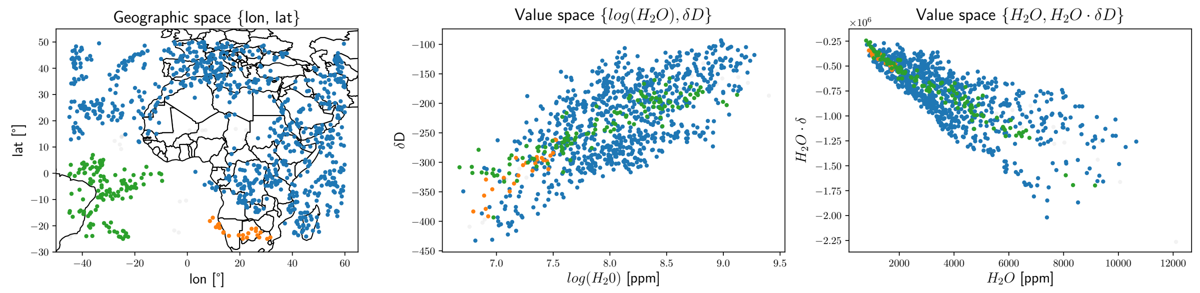

Figure 2: $\{H_2O, \delta D\}$ cluster analysis with DBSCAN, $\varepsilon = 10, minPts = 200$. Only 1,000 randomly sampled points are shown for better visibility.

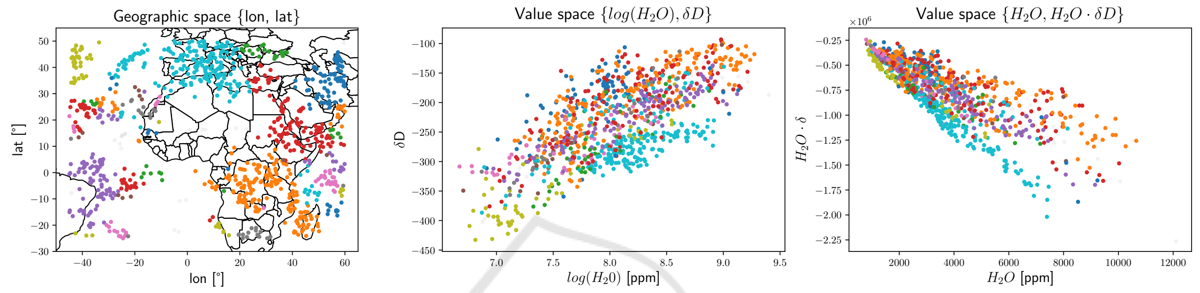

Figure 3: $\{H_2O, \delta D\}$ cluster analysis with CoExDBSCAN, $\varepsilon = 10, minPts = 200, \delta = 1e^{-5}$. The geographic space (longitude and latitude) has been used as spatial dimensions and the $\{log(H_2O), \delta D\}$ has been used as constraint dimensions. Only 1,000 randomly sampled points are shown for better visibility.

straints is a challenging task. In our presented verification and throughout multiple experimental runs, we applied mainly generic constraints that allow the algorithm to expand clusters for arbitrarily correlated data points. With generic constraints we can avoid overfitting the cluster algorithm, i.e. avoid to constrain the cluster expansion to the generating process. Whereas with specially tailored constraints, for example if we express the information about the functions that generate the dependent y and z variables in our synthetic data example as constraints, we can achieve a perfect match to the true label of the dataset, but would loose the generality of the algorithm. To simplify the process of defining constraints, methods from the field of active learning could be included into the data analysis workflow (Zhu, 2005; Settles, 2009) that provide appropriate constraints to the CoExDBSCAN algorithm. Furthermore, a machine learning based selection of suitable constraints could additionally aid the user in applying the algorithm to new datasets, which is part of our ongoing research.

Beyond the presented low-dimensional verification and evaluation datasets, our algorithm remains relevant even for high-dimensional data. This derives from the fact that Schubert's evaluation of the DBSCAN algorithm for high-dimensional data (Schubert et al., 2017) shows that that DBSCAN continues to be relevant even for high-dimensional data, although the parameter ε of DBSCAN becomes hard to choose in high-dimensional data due to the loss of contrast in distances. CoExDBSCAN is based on DBSCAN and, moreover, overcomes the issue of loss of contrast in distances by utilizing a user-defined subspace for the distance measure.

However, besides the challenge of finding and expressing suitable constraints, finding the right parameters for the algorithm still remains another challenge, especially for high-dimensional data. In addition to the parameters of the DBSCAN algorithm, the dimensions for the spatial- and constraint-subspace have to be determined by the user. For the parameters of the DBSCAN algorithm we usually rely on hyperparameter optimization techniques, for example grid search, while varying the selected dimensions based on domain knowledge and the expected outcome of the analysis. We expect to provide a more general guidance on the selection of parameters with future cluster analysis findings based on CoExDBSCAN.

6 CONCLUSION

In this article we propose a new density-based clustering algorithm with constrained cluster expansion, CoExDBSCAN. The proposed algorithm uses DBSCAN to find density-connected clusters in a defined subspace of features and restricts the expansion of clusters to a priori constraints. Incorporating a priori knowledge into the clustering process can significantly improve the clustering results and can align the outcome of the clustering process with the objective of the data analysis, as demonstrated in Section 4. Our approach combines different techniques form subspace, correlation and constrained clustering. Specifically, we introduce two user-defined parameters to the original DBSCAN algorithm, one to define the dimensions of the subspace to be used to discover density-based clusters, and one to define the dimensions of the subspace to be used to apply constraints to the cluster expansion. Further, we modify the cluster expansion step in the original DBSCAN algorithm to be restricted to these user-defined constraints. Our validation of the algorithm on an experimental and real-world dataset demonstrates, that our algorithm is especially suited for spatio-temporal data, where one subspace of features defines the spatial extent of the data and another correlations between features.

In the future, we plan to evaluate different constraints in terms of their feasibility and added overhead compared to the improvement of the clustering results, as well as propose a machine learning based selection of suitable constraints, according to the inherent structure of the data. In addition, we plan to work on an optimized implementation of the algorithm that allows us to provide additional runtime measurements and detailed comparison studies with other algorithms in the field of subspace, correlation and constrained clustering.

REFERENCES

Achtert, E., Böhm, C., David, J., Kröger, P., and Zimek, A. (2008). Robust clustering in arbitrarily oriented subspaces. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, ICDM '08, pages 763–774, Philadelphia, PA. Society for Industrial and Applied Mathematics.

Aggarwal, C. C. and Yu, P. S. (2000). Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 70–81, New York, NY, USA. ACM.

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 94–105, New York, NY, USA. ACM.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: Ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60.

Basu, S., Davidson, I., and Wagstaff, K. (2008). *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, Boca Raton, Florida.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful? In Beeri, C. and Buneman, P., editors, *Database Theory — ICDT'99*, pages 217–235, Berlin, Heidelberg. Springer Berlin Heidelberg.

Böhm, C., Kailing, K., Kröger, P., and Zimek, A. (2004a). Computing clusters of correlation connected objects. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 455–466, New York, NY, USA. ACM.

Böhm, C., Kailing, K., Kröger, P., and Zimek, A. (2004b). Computing clusters of correlation connected objects. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 455–466, New York, NY, USA. Association for Computing Machinery.

Campello, R. J. G. B., Moulavi, D., and Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In Pei, J., Tseng, V. S., Cao, L., Motoda, H., and Xu, G., editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT Press, Cambridge, Massachusetts.

Dinler, D. and Tural, M. K. (2016). *A Survey of Constrained Clustering*, pages 207–235. Springer International Publishing, Cham.

Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15.

Ertöz, L., Steinbach, M., and Kumar, V. (2003). *Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data*, pages 47–58. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD '96, pages 226–231, Palo Alto, California. AAAI Press.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

Friedman, J. H. (1994). An overview of predictive learning and function approximation. In Cherkassky, V., Friedman, J. H., and Wechsler, H., editors, *From Statistics to Neural Networks*, pages 1–61, Berlin, Heidelberg. Springer Berlin Heidelberg.

Guha, S., Rastogi, R., and Shim, K. (2001). Cure: an efficient clustering algorithm for large databases. *Information Systems*, 26(1):35 – 58.

Hough, P. V. (1962). Method and means for recognizing complex patterns. US Patent 3,069,654.

Houle, M. E., Kriegel, H.-P., Kröger, P., Schubert, E., and Zimek, A. (2010). Can shared-neighbor distances defeat the curse of dimensionality? In Gertz, M. and Ludäscher, B., editors, *Scientific and Statistical Database Management*, pages 482–500, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. MIT Press, Cambridge, Massachusetts.

Noone, D. (2012). Pairing measurements of the water vapor isotope ratio with humidity to deduce atmospheric moistening and dehydration in the tropical midtroposphere. *Journal of Climate*, 25(13):4476–4494.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

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

Pourrajabi, M., Moulavi, D., Campello, R. J. G. B., Zimek, A., Sander, J., and Goebel, R. (2014). Model selection for semi-supervised clustering. In Amer-Yahia, S., Christophides, V., Kementsietsidis, A., Garofalakis, M. N., Idreos, S., and Leroy, V., editors, *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014*, pages 331–342, Konstanz. OpenProceedings.org.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Role, F., Morbieu, S., and Nadif, M. (2019). Coclust: A python package for co-clustering. *Journal of Statistical Software, Articles*, 88(7):1–29.

Ruiz, C., Spiliopoulou, M., and Menasalvas, E. (2007). C-dbscan: Density-based clustering with constraints. In An, A., Stefanowski, J., Ramanna, S., Butz, C. J., Pedrycz, W., and Wang, G., editors, *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, pages 216–223, Berlin, Heidelberg. Springer Berlin Heidelberg.

Schneider, M., Borger, C., Wiegele, A., Hase, F., García, O. E., Sepúlveda, E., and Werner, M. (2017). Musica metop/iasi {H_2O,δD} pair retrieval simulations for validating tropospheric moisture pathways in atmospheric models. *Atmospheric Measurement Techniques*, 10(2):507–525.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3).

Schubert, E. and Zimek, A. (2019). ELKI: A large open-source library for data analysis - ELKI release 0.7.5 ”heidelberg”. *CoRR*, abs/1902.03616:1–134.

Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Wagstaff, K. and Cardie, C. (2000). Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML ’00, pages 1103–1110, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Zhu, X. J. (2005). Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Zimek, A., Schubert, E., and Kriegel, H.-P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387.