Grace – Limiting the Number of Grid Cells for Clustering High-Dimensional Data

Anna Beer, Daniyal Kazempour, Julian Busch,
Alexander Tekles, and Thomas Seidl

Ludwig-Maximilians-Universität München, Munich, Germany
{beer,kazempour,busch,seidl}@dbs.ifi.lmu.de
alexander.tekles@campus.lmu.de

Abstract. Using grid-based clustering algorithms on high-dimensional data has the advantage of being able to summarize datapoints into cells, but usually produces an exponential number of grid cells. In this paper we introduce Grace (using a *Gr*id which is *a*daptive for *c*lust*e*ring), a clustering algorithm which limits the number of cells produced depending on the number of points in the dataset. A non-equidistant grid is constructed based on the distribution of points in one-dimensional projections of the data. A density threshold is automatically deduced from the data and used to detect dense cells, which are later combined to clusters. The adaptive grid structure makes an efficient but still accurate clustering of multidimensional data possible. Experiments with synthetic as well as real-world data sets of various size and dimensionality confirm these properties.

Keywords: Grid-based, Clustering, High-dimensional

1 Introduction

Clustering is one of the most important and well investigated unsupervised data mining tasks. Nevertheless, some problems related to the curse of dimensionality are still not solved. Grid based approaches suffer not only from the exponentially increasing number of cells in relation to the number of dimensions, but also from the incoherence between data and grid structure. As many real-world datasets have high dimensional feature spaces, being able to handle many dimensions is quite important for clustering algorithms.

Even though subspace clustering algorithms focus on high-dimensional data, they assume clusters to be in a low dimensional subspace of the data and are thus not suitable to find clusters lying in the full-dimensional space. Density based approaches on the other hand find clusters in full-dimensional space where all dimensions are equally important, but cannot handle high-dimensional data. Thus, efficient clustering of high-dimensional data without any a priori knowledge is still a huge challenge.

Hence we developed a grid-based clustering approach for high-dimensional data. It works fully automatically and finds clusters in full-dimensional data space without creating an exponential number of cells. The grid adapts itself to the data in regards of cell size as well as individual number of cells per dimension by using information from one-dimensional projections of the data. This leads not only to an adequate quality of the clustering results, but at the same time facilitates high efficiency of the algorithm.

Our main contributions are as follows:

– We develop Grace, a new grid-based clustering algorithm.
– By constructing the adaptive grid gradually, we are, to the best of our knowledge, the first ones to circumvent an exponential number of grid cells in relation to the number of dimensions.
– Grace is efficient and detects clusters of arbitrary shape in high-dimensional space.

The rest of the paper is structured as follows. Section 2 provides an overview of related work in the field of density-based and grid-based clustering. The algorithm itself is described in Section 3 and evaluated theoretically as well as empirically in Section 4. A brief conclusion is finally provided in Section 5.

2 Related Work

Since there exists a wealth of literature in the field of density-based as well as gird-based clustering, this section aims to provide an overview on some of the existing methods. We shall provide a brief elaboration on the core ideas behind each of the methods revealing the distinctive properties of our method in contrast to the competitors.

2.1 Density-based Clustering Approaches

Density-based clustering methods detect dense regions which are enclosed and separated by sparse regions and are thus suitable to find arbitrarily shaped clusters. Most density-based methods rely on local densities based on distances in the full-dimensional data space. The most common density-based approach is DBSCAN [7], which considers points with at least $minPts$ points in their ε-range as core points. Core points are connected if their distance is lower than ε and form a cluster. Not-core points either lie in the ε-range of a core point and get assigned to the cluster of the core point, or are declared noise. DBSCAN is quite sensitive to ε and $minPts$, which are two parameters hard to guess for a user without detailed knowledge of the data. OPTICS [3] improves DBSCAN by introducing a reachability plot based on $minPts$, on which users can see the cluster structure and choose appropriate ε.

DENCLUE [11] uses local densities to compute an overall density function, the maxima of which constitute density attractors. Every object is connected to such a density attractor by means of a hill-climbing procedure. A threshold ξ gives the minimum density level for a density-attractor which allows to find noise. To accelerate the calculation of local densities, a simple grid with the same cell width 2σ in all dimensions is used, where σ is a user given parameter.

Other density-based clustering algorithms usually build on the approaches presented so far and aim to improve or extend them. HDBSCAN [5] for example extends DBSCAN to a hierarchical approach allowing different levels of density that can detect clusters of different density or nested clusters, overcoming the aforementioned issue of one global hyperparameter setting. DeLiClu [1] and SOPTICS [15] represent algorithms with purposes similar to OPTICS, but with improvements regarding their efficiency. Likewise, DENCLUE 2.0 [9] is a straightforward improvement of DENCLUE with reduced runtime complexity.

2.2 Grid-based and Subspace Clustering Approaches

Grid-based clustering methods generally partition the data into cells of different densities by dividing each dimension into several intervals. Those cells can then be connected into clusters without having to look at each data point again, which decreases the runtime. The results are often highly dependent on the structure of the constructed grid and most algorithms require users to set the defining parameters. STING [14] proposes a quite interesting hierarchical grid structure based on statistical information, but is neither used for clustering, nor does it deliver exact values, but rather approximations. Also, the distribution type of the data has to be known or ascertained by hypothesis tests.

Most grid-based clustering techniques produce subspace clusterings i.e. they detect subspaces of a high-dimensional data space which contain clusters of the given data. Grid-based approaches are well-suited for this task because they can easily exploit the monotonicity of the clustering criterion regarding dimensionality. This criterion implies that a k-dimensional cell is dense only if every (k - 1)-dimensional projection of the cell is also dense, given a constant density threshold for the number of objects in a cell.

One of the first clustering algorithms to implement a grid-based subspace clustering was CLIQUE [2]. After constructing a grid with the same number of equidistant intervals in each dimension and identifying the dense cells, CLIQUE employs a bottom-up approach to find subspaces with dense regions by joining cells in k-dimensional spaces to candidate cells in $(k + 1)$ dimensions. If the number of objects in such a candidate cell exceeds a given density threshold, the corresponding $(k+1)$-dimensional space is considered a relevant subspace. After detecting the relevant subspaces, CLIQUE connects adjacent dense cells in their corresponding subspaces.

However, CLIQUE does not take the data distribution into account for generating the grid structure or for finding the dense regions. One subspace clustering approach that considers the data distribution beforehand is FIRES, which detects clusters on the basis of a greedy heuristics merging one-dimensional clusters in order to find approximations of subspace clusters. This significantly reduces the runtime complexity of FIRES compared to CLIQUE.

While FIRES does not employ a grid structure at all, MAFIA [8] incorporates data distribution by using an *adaptive grid* in order to produce a better partitioning of the dimensions and reduce the number of grid cells. MAFIA determines the intervals in each dimension on the basis of one-dimensional histograms. Adjacent bins of a histogram are joined if they have approximately the same frequency. This yields larger intervals in the dimensions, each with roughly constant (one-dimensional) density. Nevertheless, MAFIA requires two parameters that may have a significant impact on the results and there is no guaranteed bound on the number of grid cells. On its adaptive grid, MAFIA proceeds like CLIQUE.

A general framework for clustering high-dimensional data on the basis of an adaptive grid is provided by OptiGrid [10] which recursively splits the data set by means of separating hyperplanes which should cut through low-density regions and separate high-density regions. The resulting cells already represent clusters, given a sufficiently high density. Though different approaches exist for selecting suitable hyperplanes [10, 6], these methods are not able to detect arbitrarily shaped clusters since generated cells already represent clusters and are not combined. Further, these methods require setting parameters whose impact on the result is difficult to assess a priori.

Further grid-based methods include SCHISM [17] which addresses the question of how to define and detect statistically interesting subspaces in high-dimensional data. As a measure for interestingness, the authors rely on the Chernoff-Hoeffding bound and use it for pruning. WaveCluster [18] relies on discrete wavelet transformation. The data is mapped to the frequency domain where clusters are then found by detecting dense regions. The method is insensitive to outliers and has a runtime complexity linear in the number of data objects. Among the most recent grid-based methods, ITGC [4] is an information theoretic approach regarding clustering as a data compression task. As such, neighboring grid cells are merged if it is beneficial with respect to compression costs.

3 Efficient Grid-based Clustering of Multi-dimensional Spaces

In this section we describe Grace in detail. In 3.1 we explain how the non-equidistant regular grid is generated dependent on the respective dataset. Section 3.2 shows how dense grid cells are combined to form clusters.

3.1 Generation of Adaptive Grids

The grid generation process is designed to limit the number of generated cells depending on the number of data points N and still allow for an accurate detection of clusters. For that we first estimate the density of each dimension separately and then split the data space iteratively based on these estimations until the number of cells exceeds $N \cdot \log(N)$. To reduce runtime for calculating local changes in one-dimensional densities, we consider a histogram with $b = max(50, \sqrt{N/d})$ equi-width bins for every dimension instead of all points separately. A maximum of 50 bins for each histogram has shown to produce appropriate grid structures for various data distributions and different numbers of points N. Higher N imply possibly more complex shaped clusters requiring a higher granularity of the histogram. A higher number of dimensions d in contrast results in a lower number of bins, since high dimensionality implies less expressiveness of distances and less bins allow for higher deviations.

Estimation of Local Changes in One-Dimensional Densities Next, we compute for each bin in all one-dimensional histograms a *local change indicator* to express the local change of the one-dimensional density. To this end, we measure local changes in density as differences of frequencies between areas left and right to a particular bin and additionally set them in relation to the frequencies in their respective areas to distinguish random variations from relevant density shifts. The relevant neighborhood left and right of a particular bin is determined dynamically based on the frequency f covered by this area. The idea is to add less bins if the local density is already high, such that the separating hyperplanes will be lying closer together. Adjacent bins left and right are added iteratively until f exceeds a threshold t which is adjusted after each step. The threshold primarily depends on f and N such that a certain fraction of all objects needs to lie within the neighborhood range to stop expanding it. To avoid building large low-density cells, the neighborhood frequency is further weighted with the width of the current neighborhood range h, leading to a threshold

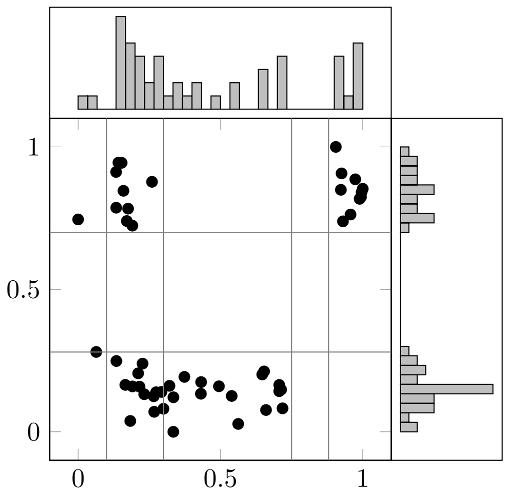


Fig. 1: Two-dimensional data with histograms as approximations of the one-dimensional data projections and edges chosen respectively

$$t = \frac{1}{h \cdot (1/b) \cdot (N/d)} \cdot N. \tag{1}$$

These weighted frequencies are used both for computing the differences between left and right areas as well as for determining the neighborhood area. As a consequence, the density close to a bin has more impact on these computations than more distant bins. Figure 1 illustrates the generation of an adaptive grid based on the one-dimensional data projections as described so far.

Selection of Intervals After the one-dimensional histograms are computed and a change indicator is assigned to each bin, separating hyperplanes are selected iteratively until the number of cells generated by these planes is greater than or equal to $N \cdot \log(N)$. If d is high, not all dimensions are considered in order to limit the number of cells, i.e. some dimensions are not split by a separating plane. A dimension is only considered if at least two edges are chosen for this dimension. With only one edge in a dimension, i.e. two intervals, all data points would lie in the same interval or in adjacent intervals with respect to this dimension. In both cases, the corresponding dimension would have no informative content. This grid generation method yields at max $3 \cdot N \cdot \log(N)$ cells (see Theorem 1).

Theorem 1. *Given a d-dimensional data set with N objects. A regular grid in the d-dimensional space that is constructed by iteratively adding cutting hyperplanes, consists of at most 3N cells, if the grid generation process is stopped as soon as the number of cells is greater than or equal to N.*

Proof. Suppose h_{dim} edges are already chosen for dimension $dim \in 1, ..., d$. If $h_k \leq 1$, dimension k is not yet considered due to the minimum of two edges in a dimension for it to be considered. Let $D_c \subseteq 1, ..., d$ be the set of indices of the dimensions already considered. Dimension k is divided into $b_k = (h_k + 1)$ intervals. The number of cells c in the grid is computed by multiplying the number of intervals in all dimensions:

$$c = \prod_{dim \in D_c} b_{dim}$$

When adding a hyperplane separating dimension k to increase the number of cells from c to c', three cases can occur. Before adding the separating plane, $c < N$ holds.

Case 1: $h_k > 1$

If $h_k > 1$, the number of intervals on the corresponding coordinate axis increases by one as well. This yields:

$$\begin{aligned} c' &= \prod_{dim \in D_c} b'_{dim} = (b_k + 1) \cdot \prod_{dim \in D_c \setminus k} b_{dim} \\ &= \left[b_k \cdot \prod_{dim \in D_c \setminus k} b_{dim} \right] + \left[1 \cdot \prod_{dim \in D_c \setminus k} b_{dim} \right] \\ &\leq c + c = 2 \cdot c < 2 \cdot N < 3 \cdot N \end{aligned}$$

Case 2: $h_k = 0$

If $h_k = 0$, the number of cells does not increase at all, as dimension k is not considered yet and will not after this iteration. For the relationship between c and c' it holds, that: $c' = c < N < 3N$

Case 3: $h_k = 1$

If $h_k = 1$, k is a new dimension to be considered, as $h'_k = h_k + 1 = 2$ and $b'_k = b_k + 1 = 3$ respectively. The number of cells therefore increases by the factor 3. $h'_k = 2 \Rightarrow k \in D_c$

$$c' = \prod_{dim \in D_c} b'_{dim} = b_k \cdot \prod_{dim \in D_c \setminus k} b_{dim} = 3 \cdot \prod_{dim \in D_c} b_{dim} = 3 \cdot c < 3 \cdot N$$

□

Every bin of the initial histogram represents a potential edge for splitting. In every iteration, the bin with the maximum local change indicator is chosen as the next edge. To choose edges closer to cluster borders, we make a small adjustment after selecting an edge: The edge is shifted in one direction as long as two successive bins have a frequency difference below 5% and we choose the direction to which the edge needs to be shifted less. After selecting an edge, we discard all bins within the neighborhood if the selected bin from the set of potential edges. This step avoids high granularity of the grid in areas of density changes. Algorithm 1 summarizes the grid creation.

Algorithm 1 CreateGrid

```
$b \leftarrow max(50, \sqrt{N/d})$
for all dimensions do
  generate histogram with b bins
end for
for all dimensions do
  for all bins of histogram do
    determine neighborhood range
    compute local change indicator
  end for
end for
sort the bins of all histograms w.r.t. the local change indicator
$moreEdgesNeeded \leftarrow true$
$selectedEdges \leftarrow \{\}$
while $moreEdgesNeeded$ do
  select the bin with highest local change indicator
  shift the bin if the adjacent bins have approximately similar frequencies
  add the edge to $selectedEdges$
  discard the bins within the neighborhood range of the selected bin from the set of
  potential edges
  if grid generated by $selectedEdges$ contains more than $N \cdot \log(N)$ cells then
    $moreEdgesNeeded \leftarrow false$
  end if
end while
```

3.2 Simple Connection of Dense Grid Cells to Clusters

Given the generated grid, the next steps involve detection of dense grid cells and subsequent combination of adjacent dense cells. For Grace, we apply wider notion of adjacency than existing works: Coordinates may differ at most by one in all dimensions – compared to a narrow notion, where the coordinates of adjacent bins may differ by one only in exactly one dimension.

A cell of volume V is considered dense, if it contains more than $minPts/V$ points for a $minPts$ given by the user. To avoid setting the density threshold too high for clusters with lower density, we first determine the most dense cells. To this end, we identify all cells containing more points that they would in expectation assuming a uniform distribution. In a second step, we discard these cells and detect the remaining dense cells with lower density using a new threshold.

Detection of dense cells is straightforward. After discretizing all data points to the grid, the number of data points in each grid cell is counted and compared to the threshold of the particular grid cell. Given an adequate grid structure, the number of dense cells p is usually much lower than N. Adjacent dense cells are now connected to form clusters by extracting connected components from the graph represented by the symmetric $p \times p$ adjacency matrix M with $M_{i,j}$ if and only if cells i and j differ by exactly one dimension. Adjacent cells can be found by sorting the dataset in every dimension and then iterating over the dimensions.

3.3 Connection of Diagonally Adjacent Grid Cells

So far, only adjacent cells in the narrower sense have been connected. To connect cells adjacent in the wider sense, i.e., diagonally adjacent cells, we add additional helper cells next to dense cells. Given a cell $\tilde{a}$, that is either a dense cell or a previously added helper cell with coordinates $(a_1, a_2, ..., a_d)$ and the order of dimensions considered for the current sorting of the coordinates $d_{i_1}, d_{i_2}, ..., d_{i_d}$, a new helper cell $(b_1, b_2, ..., b_d)$ with $b_k = a_k \forall k \in \{i_1, ..., i_{d-1}\}$ and $b_{i_d} = a_{i_d} + 1$ is now added to the set of helper cells if it has not yet been added before. New helper cells are not considered in the same iteration they were added, but in subsequent iterations they are treated just like the original dense cells. This ensures to find exactly all connections between originally adjacent dense cells in the wider sense.

Theorem 2. *Given a d-dimensional grid with a set P of dense cells and an initially empty set of helper cells H, both of whom are iterated d times. In every iteration i, a cell b with coordinates $b_k = a_k \forall k \in \{1, ..., d\} \setminus i$ for each cell $a \in P \cup H$ that has no adjacent cell with the coordinates of b is added to the set of helper cells after the current iteration. With this procedure, two dense cells that are adjacent in the wider sense can be connected, either directly or indirectly with the help of other cells in $P \cup H$, by just applying the notion of adjacency in the narrow sense.*

Proof. Given two dense cells a and b with coordinates $(a_1, a_2, \dots, a_d)$ and $(b_1, b_2, ..., b_d)$, respectively.

Case 1: a and b are adjacent in the narrow sense
Adjacency in the narrow sense implies adjacency in the wider sense by definition, thus the two cells are also adjacent in the wider sense.

Case 2: $a_i \in \{b_i, b_i - 1\}, \forall i \in \{1, ..., d\}$
Suppose the cells' coordinates differ in dimensions $\{j_1, j_2, ..., j_m\} \subseteq \{1, 2, ..., d\}$ with $j_s < j_t \ \forall s < t$. In iteration j_1, the cell $a_{(1)}$ with coordinates $(a_1, ..., a_{j_1} + 1, ..., a_d)$ is either added as helper cell or already a dense cell. Since this cell differs by one from a in exactly one dimension, it is adjacent to a in the narrow sense. In iteration j_2, the cell $a_{(2)}$ with coordinates $(a_1, ..., a_{j_1} + 1, ..., a_{j_2} + 1, ..., a_d)$ is again either added as helper cell or already a dense cell. The new cell is now adjacent to the previously added cell $a_{(1)}$. This step is then repeated for all $j \in \{j_1, j_2, ..., j_{m-1}\}$. Finally, the cell $a_{(m-1)}$ with coordinates $(a_1, ..., a_{j_1} + 1, ..., a_{j_2} + 1, ..., a_{j_{m-1}}, ..., a_d)$ is either added as helper cell or already a dense cell. $a_{(m-1)}$ differs in exactly one dimension from b, so that $a_{(m-1)}$ and b are adjacent in the narrow sense and thus a and b.

Case 3: $a_i \in \{b_i, b_i + 1\}, \forall i \in \{1, ..., d\}$
Switching a and b converts this case to the same as case 2.

Case 4: $a_i \in \{b_i, b_i - 1\}, \forall i \in I \subset \{1, ..., d\}$ and $a_j \in \{b_j, b_j + 1\} \forall j \in J = \{j_1, j_2, ..., j_m\} \subset \{1, ..., d\} \setminus I$
In this case, two chains of adjacent cells can be constructed, each following the same idea as in case 2 or in case 3 respectively. The first chain starts from cell a, considering all dimensions with $a_i = b_i + 1$, which corresponds to case 2. The second chain starts from cell b, considering all dimension with $a_i = b_i - 1$, which corresponds to case 3. Finally, without loss of generality, the coordinates of the last cell $\tilde{a}$ in the first chain are of the form $(\tilde{a}_1, \tilde{a}_2, ..., \tilde{a}_d)$ with $\tilde{a}_i = \tilde{a}_i + 1 = b_i \forall i \in I$ and $\tilde{a}_k = b_k \forall k \in \{1, ..., d\} \setminus I$. The coordinates of the last cell $\tilde{b}$ in the second chain are of the form $\tilde{b}_1, \tilde{b}_2, ..., \tilde{b}_d$ with $\tilde{b}_j = \tilde{b}_j + 1 = a_j \forall j \in \{j_1, ..., j_{m-1}\}$, $\tilde{b}_{j_m} = a_{j_m} + 1$ and $\tilde{b}_k = a_k \forall k \in \{1, ..., d\} \setminus J$. Now, these two last cells of both chains differ only in dimension j_m by one, so that they are connected in iteration j_m.

□

4 Evaluation

We investigate Grace from a theoretical as well as from an empirical point of view. In Section 4.1 we calculate the runtime complexity and in 4.2 we present and discuss several experiments based on synthetic as well as real world data sets and compare the results with DBSCAN and CLIQUE.

4.1 Complexity Analysis

The histograms for each dimension can be computed in time $O(N \cdot d)$. For each of the $b \cdot d$ histogram bins, the local change indicator can be computed in constant time, leading to $O(b \cdot d)$. Having $b \leq \sqrt{N \cdot d}$, we get $O(\sqrt{N \cdot d} \cdot d) = O(\sqrt{N} \cdot d^2)$. Finding separating hyperplanes requires sorting the $b \cdot d$ local change indicators, which can be done in time $O(b \cdot d \cdot \log(b \cdot d)) = O(\sqrt{N \cdot d} \cdot d \cdot \log(\sqrt{N \cdot d} \cdot d)) = O(N \cdot d^2)$.

The dense cells are determined by counting for every point the occurrences of each coordinate combination, which is in $O(N \cdot d)$. Dense cells can be combined efficiently by sorting them in every dimension to identify adjacent grid cells in that dimension. Since the maximum number of dense cells is $N \cdot \log(N)$, the complexity of sorting the dense cells is $O(N \cdot \log(N) \cdot \log(N \cdot \log(N)) \cdot d) = O(N \cdot \log^2(N) \cdot d)$. Adjacent dense cells can be identified in each dimension by comparing consecutive cells in the sorted order with complexity $O(N \cdot d)$. Thus, all connections between dense grid cells can be detected in time $O(N \cdot \log^2(N) \cdot d)$. Connected components can be identified in time $O(N^2 \cdot \log^2(N))$. In total, the complexity is thus

$$O(N \cdot d + \sqrt{N} \cdot d^2 + N \cdot d^2 + N \cdot \log^2(N) \cdot d + N^2 \cdot \log^2(N)) = O(N^2 \cdot \log^2(N) \cdot d^2).$$

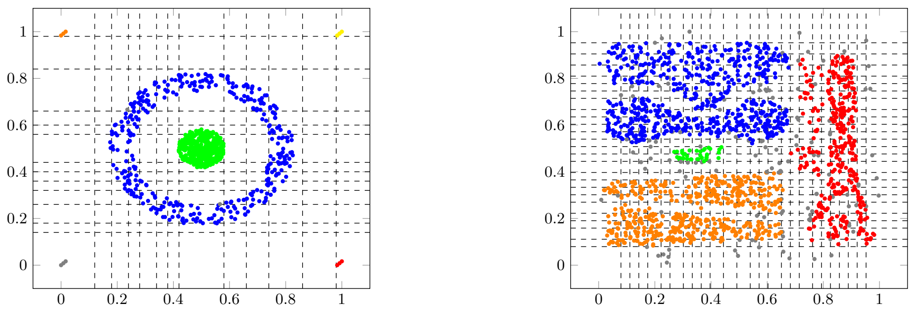
Note, that the number of dense cells p which is responsible for the $N^2 \cdot \log^2(N)$ part is usually far lower than N.

4.2 Empirical

The following experiments have been conducted on a Linux machine with a commodity hardware featuring a 2.0 GHz CPU with two cores and 3.6 GB RAM. As Grace uses elements from density-based as well as grid-based clustering, we compare our results to those DBSCAN and CLIQUE. Where Grace works fully automatically, DBSCAN and CLIQUE both need two parameters, for which those yielding the best results were chosen. For DBSCAN we used the scikit-learn Python library implementation and for CLIQUE the implementation from the data mining framework ELKI [16].

For a first visual interpretation, we show the effectivity of Grace working on two simple two-dimensional synthetic data sets containing density-based clusters and compare the results to those of DBSCAN. Figure 2 shows the clustering result for “TARGET” [19] with $N = 770$, consisting of four small clusters distributed at opposing corners as seen in Figure 2, one of them being detected as noise (bottom left cluster). Apart from the detected noise cluster, all other clusters are detected correctly by Grace. With a proper parameter setting, DBSCAN is capable of detecting all clusters, too. “CLUTO-T8-8K” contains 8000 two-dimensional objects [12]. Applied to this data set, the Grace found some, but not all clusters similar to DBSCAN.

Another synthetic data set with $N = 101,000$ and $d = 9$ containing three spherical clusters and $1,000$ noise objects has been created to show the scalability of Grace. The clusters of this data set are found in mere 4.5 seconds, where, due to excessive memory consumption, neither DBSCAN nor CLIQUE could be applied to this data set with the machine used here. To compare at least CLIQUE with the proposed approach in high dimensions, another data set with



(a) TARGET dataset (b) CLUTO-T8-8K

Fig. 2: Clustering results using an adaptive grid (represented by the dashed lines). Noise is colored gray.

$N = 100,000$ and $d = 8$, also containing three spherical clusters, is clustered by the two algorithms. They both detect all three clusters, where Grace was almost three times as fast as CLIQUE (1.7 vs 4.8 seconds).

Finally, a data set derived from the "VICON" data set containing physical action data measuring human activity [13] has been tested. The actions are measured by means of nine sensors on different body parts, each emitting three-dimensional spatial data. In summary, this yields a data set with 27 dimensions. For this experiment, the two actions punch and handshake have been merged into one data set with 5045 objects. Again, neither the DBSCAN implementation nor the CLIQUE implementation used can be applied to this data set on the machine used due to excessive memory consumption. Grace detected an accurate clustering within 232ms, with one cluster being detected 100% and the other cluster being split with 92% of it being grouped in one cluster.

5 Conclusion

Grace finds clusters in multidimensional data spaces where points build a cluster if they are close in all dimensions. It generates an adaptive grid structure that makes it possible to reduce the runtime complexity significantly for multidimensional data spaces compared to similar grid-based approaches. The experimental evaluation has shown that the algorithm outperforms DBSCAN and CLIQUE for large data sets and high dimensions. Grace works fully automatically and can be applied to datasets of various sizes, dimensionalities, and cluster densities. For clustering high-dimensional data with all dimensions being relevant for forming the clusters, it is an efficient alternative to established algorithms. It is moreover a possibility to get some first insights if no information about the data is available yet, since no expert knowledge about the data is needed beforehand due to the absence of any parameters. In future work noise should be handled separately and we are also investigating the suitability for anytime results. The grid construction is promising for many other applications, and could, e.g., be applied in context of arbitrarily oriented correlation clusters.

Acknowledgments

This work has been partially funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

References

1. Achtert, E., Böhm, C., Kröger, P.: Deliclu: Boosting robustness, completeness, usability, and efficiency of hierarchical clustering by a closest pair ranking. In: PAKDD (2006)
2. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD (1998)
3. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: Ordering points to identify the clustering structure. In: SIGMOD (1999)
4. Behzadi, S., Hinterhauser, H., Plant, C.: Itgc: Information-theoretic grid-based clustering. In: EDBT (2019)
5. Campello, R., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pei, J., Tseng, V., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD (2013)
6. Chang, J.W., Jin, D.S.: A new cell-based clustering method for large, high-dimensional data in data mining applications. In: SAC (2002)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD (1996)
8. Goil, S., Nagesh, H., Choudhary, A.: Mafia: Efficient and scalable subspace clustering for very large data sets. In: KDD (1999)
9. Hinneburg, A., Gabriel, H.H.: Denclue 2.0: Fast clustering based on kernel density estimation. In: IDA (2007)
10. Hinneburg, A., Keim, D.: Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In: VLDB (1999)
11. Hinneburg, A., Keim, D.: An efficient approach to clustering in multimedia databases with noise. In: KDD (1998)
12. Karypis, G., Han, E.H., Kumar, V.: Chameleon: A hierarchical clustering algorithm using dynamic modeling. IEEE Computer (1999)
13. Lichman, M.: UCI machine learning repository (2013), http://archive.ics.uci.edu/ml
14. Muntz, R., Wang, W., Yang, J.: Sting: A statistical information grid approach to spatial data mining. In: VLDB (1997)
15. Schneider, J., Vlachos, M.: Scalable density-based clustering with quality guarantees using random projections. DMKD (2017)
16. Schubert, E., Zimek, A.: Elki: A large open-source library for data analysis-elki release 0.7. 5" heidelberg". arXiv preprint arXiv:1902.03616 (2019)
17. Sequeira, K., Zaki, M.: Schism: A new approach for interesting subspace mining. In: ICDM (2004)
18. Sheikholeslami, G., Chatterjee, S., Zhang, A.: Wavecluster: A multi-resolution clustering approach for very large spatial databases. In: VLDB (1998)
19. Ultsch, A.: Clustering with som, u*c. In: WSOM (2005)