MAFIA: An Adaptive Grid-Based Subspace Clustering Approach

Henrik Daniel Christensen[hench13@student.sdu.dk]

University of Southern Denmark, SDU
Department of Mathematics and Computer Science

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: High-Dimensional Subspace Clustering · MAFIA · Critical Review · Comparative Study.

1 Introduction

Clustering is one of the main techniques within data mining. This technique is a descriptive method that tries to discover unknown patterns within a data set, by partitioning the data objects into subsets (*clusters*). Here, each object in a cluster is similar to one another, but different from objects in other clusters. Clustering is widely used in many applications, such as biology, web search and business intelligence [5, p. 444].

A simple clustering example is in the context of customer data, where it is useful to group similar customers together for the purpose of targeted advertising or placement of products within a store [7, p. 5].

The task of clustering data is challenging task, first of all as the data sets typical is large in size, which means that the clustering algorithm must be *scalable*. Additionally, data sets often contains numerous features (*attributes*), which introduces the problem of *curse of dimensionality*, which refers to a several challenges related to high-dimensional data spaces:

First, there is the issue of *concentration of distances*, a phenomenon in high-dimensional spaces where the distances between objects become increasingly similar as dimensionality increases. This means that data points tend to become nearly equidistant from one another, making it difficult for traditional distance-based algorithms to discover clusters.

Secondly, there is the problem of *local feature relevance* and *local feature correlation*, where only a subset of features or different combinations of feature correlations may be relevant for clustering. Consequently, feature reduction techniques like Principal Component Analysis (PCA), which project the original space onto a lower-dimensional subspace, are inadequate because they typically identify only one global subspace that best fits the entire dataset. Similarly, clustering algorithms that evaluate the entire feature space, such as DBSCAN, struggle to address this issue effectively. [7, p. 43–46]

Instead of relying on a global approach to feature selection, a local approach that addresses the issues of local feature relevance and local feature correlation is necessary. However, when clustering high-dimensional data we encounter two separate problems, which, however, both needs to be solved simultaneously. First, is the problem of finding the relevant subspaces of each cluster. Second, is the problem of finding the clusters in each relevant subspace. In the first problem, notice that the search space is in general infinite and for the second problem, to find the best partitioning of the objects is NP-complete. To solve them simultaneously, we need to employ heuristics into the clustering algorithms. [7, p. 6–7]

For many applications, it is reasonable to focus only on clusters in axis-parallel subspaces, thus restricting the search space to $O(2^d)$ dimensions. These algorithms are often called *projected clustering* or *subspace clustering* algorithms. Furthermore, these can be divided into two categories: *top-down*- and *bottom-up* approaches. In the top-down approach, the relevant subspaces for the clusters are determined starting from the full-space either using the so called *locality assumption* or using a random sampling. In contrast, bottom-up approaches, finds the relevant subspaces for the clusters from the original space starting from one-dimensional using the *downward closure property* (also called monotonicity property), that says, *if a subspace contains a cluster, then a superspace must also contain a cluster*, which can be used to prune (exclude) subspaces. [7, p. 8, 11]

1.1 Contributions

This paper examines three different bottom-up subspace clustering algorithms, with a primary focus on the MAFIA algorithm [9], a grid-based method that partitions the data space into adaptive grids using histograms. Since MAFIA extends the pioneering subspace clustering algorithm called CLIQUE [2], a comparative analysis between the two will be conducted. Since the two algorithms both are being grid-based, which may limit clusters to hyper-rectangles, a more flexible SUBCLU algorithm [8], which uses density-connected sets to allow clusters with arbitrary shapes, will also be analyzed and evaluated.

The remainder of the paper is organized as follows. In Section 2, first the two grid-based algorithms CLIQUE and MAFIA are analyzed and how they relate. Additionally, the density-based algorithm approach of SUBCLU will be analyzed and how it differ from the grid-based approach, as well as how it differ from the full-dimensional density-based approach of the well-known DBSCAN algorithm [3]. Section 3 gives a more detailed description of the MAFIA algorithm. Section 4 evaluate the performance of the three algorithms in terms of scalability, considering both data dimensionality and cluster dimensionality, as well as their overall cluster quality. Section 5 discusses the pros and cons of the three algorithms. Finally, Section 6 draws conclusions and points out future work.

2 Subspace Clustering

The main idea of subspace clustering is to identify subspaces of a high dimensional space to allow better clustering than the original (full) space. This is opposed to e.g. PCA, which projects the original space onto a new subspace, which may can be hard to interpret for the user.

Two different subspace clustering approaches will be discussed. First, the grid-based approach will be discussed, after which the density-based approach will be discussed. Note that, only bottom-up approaches will be considered in this paper.

We will adopt the following notation: Let $\mathcal{A} = \{A_1, \ldots, A_d\}$ be a set of domains, and $\mathcal{S} = A_1 \times A_2 \times \ldots A_d$ a d-dimensional numerical space containing n points $p_1, \ldots, p_n$. Let $A_1, \ldots, A_d$ be the dimensions (attributes) of $\mathcal{S}$.

2.1 Grid-based approach

The key idea of grid-based subspace clustering is to partition the $\mathcal{S}$ into axis-parallel grid structure starting in 1-dimensional space. The grids forms hyper-rectangular *units* (cells) for which we find the number of points in each. Only the units that exceeds a certain threshold are retained, called *dense units.* Next, adjacent dense units will be merged to form so called *candidate dense units* (CDUs), which will be used to find clusters in higher dimensional subspaces. The goal is then to describe the clusters using a minimal description in the form of DNF (*Disjunctive Normal Form*) expressions.

CLIQUE CLIQUE is a bottom-up grid-based subspace clustering algorithm that uses the monotonicity property as the clustering criterion, which is similar to the well-known Apriori algorithm.

Lemma 1. *If a collection of points S is a cluster in a k-dimensional subspace, then S is also a part of a cluster in any $(k-1)$-dimensional projections of this space.*

Proof can be found in [2].

First, $\mathcal{S}$ is partitioned into equal-sized ε (input parameter) intervals, creating axis-parallel rectangular units. Hereafter, the dense units are found in the 1-dimensional space. The dense units are then merged to form CDUs in the 2-dimensional space, thus reducing the search space using Lemma 1. That is, CDUs in any k dimensions are obtained by merging the dense cells in $(k-1)$ dimensions which share the *first* $(k-2)$ dimensions. The procedure to merge dense cells continues unitil no more CDUs are generated. However, if a dense unit exists in k dimensions then all of its projections in a subset of the k dimensions that is $O(2^k)$ different combinations, are also dense. Therefore the running time of the algorithm is exponential in the highest dimensionality of any dense unit. However, using a pruning technique suggested in [2], where the authors suggest to prune subspaces with low coverage, that is the number of points that the dense units cover in a subspace. However, this comes with a cost as it may prune subspaces that may contain clusters.

A cluster is a maximal set of connected dense units in k-dimensions. Two k-dimensional units u_1, u_2 are *connected* if they have a common face in the k-dimensional space, that is the two units share $k-1$ dimensions. or if they are connected by a common cell. That is, there exists another k-dimensional unit, say u_3, such that u_1 is connected to u_3 and u_2 is connected to u_3.

A region in k dimensions is an axis-parallel rectangular k-dimensional set. We are only interested in those regions that can be expressed as unions; henceforth all referencs to a region mean such unions. A region can be expressed as a DNF expression on intervals of the domains A_i.

We aim to make a minimal description of a cluster which will be a non-overlapping covering of the cluster. An example is given in Figure 1, where the grid size given by ε is 0.05, where the minimal description of the cluster is $A \cup B$ and has the DNF expression: $((0.05 \leq A_1 < 0.15) \wedge (0.10 \leq A_2 < 0.20)) \vee ((0.10 \leq A_1 < 0.20) \wedge (0.05 \leq A_2 < 0.15))$.

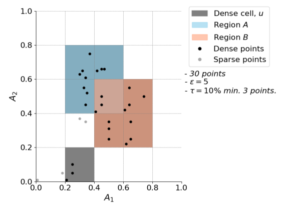


Fig. 1. Illustration of a dense unit u and two overlapping dense regions A and B.

Initialization:

The algorithm begins by making a pass over the dataset to identify dense units in 1 dimension (1D). These are the grid intervals in each individual dimension where the number of points meets or exceeds the density threshold (). Once the 1D dense units are identified, the algorithm moves on to higher dimensions, incrementing the dimension level by one at each step.

Level-by-Level Approach:

The algorithm proceeds in stages, going from 1D to 2D, then 2D to 3D, and so on, until no more candidates for dense units can be generated at a given level. Each stage is focused on determining kk-dimensional dense units using the (k1)(k1)-dimensional dense units identified in the previous stage.

Candidate Generation Procedure:

The core of the algorithm lies in its candidate generation procedure, which uses a self-join operation on the set of (k1)(k1)-dimensional dense units (denoted as Dk1Dk1). Join Condition: The join operation is performed to form potential kk-dimensional units (denoted as CkCk). For this join, units must share the same intervals in the first k2k2 dimensions, and the intervals in the (k1)(k1)-th dimension must differ. For example, if we have two (k1)(k1)-dimensional units u1u1 and u2u2 with intervals: u1.a1=u2.a1,u1.a2=u2.a2,. . . ,u1.ak2=u2.ak2u1.a1=u2.a1,u1.a2=u2.a2,. . . ,u1.ak2=u2.ak2 u1.ak1<u2.ak1u1.ak1<u2.ak1 These conditions ensure that the units share the

same intervals in the first k2k2 dimensions but differ in the last dimension, forming a candidate for a kk-dimensional unit.

Forming CkCk:

The candidate kk-dimensional units are formed by combining the intervals from u1u1 and u2u2, specifically adding the intervals from the k1k1-th dimension where they differ. The result is a superset CkCk of potential kk-dimensional dense units.

Filtering Candidate Dense Units:

The algorithm then performs a filtering step to identify which of these candidate units in CkCk are actually dense. This is done by making a pass over the data again to count the number of points in each candidate unit. Any candidate unit that meets or exceeds the density threshold () is retained as a dense kk-dimensional unit.

Projection Checking:

The algorithm further checks each dense unit from CkCk to ensure that all its projections onto (k1)(k1) dimensions are also dense (as per Lemma 1). If a kk-dimensional unit has any (k1)(k1)-dimensional projection that is not in Dk1Dk1, it is discarded. This ensures that a unit is only marked as dense if all its lower-dimensional projections are also dense.

Termination:

The algorithm continues this process, incrementing the dimensionality level, until no more candidates can be generated. At this point, the algorithm terminates.

2.2 MAFIA

MAFIA is an extension to CLIQUE where the grid sizes are adaptive meaning that the grid sizes are not fixed but are automatically determined by an algorithm. The idea is to cosim-2012ncentrate the portions of the data space, as having more points a more likely to be part of a cluster region enabling minimal length DNF expressions. The overall intention is not to rely on the input parameters CLIQUE uses do not use on the pruning technique as noted in citeclique, this could result in lost information. The algorithm is described in more detail in Section **??**.

2.3 Density-based approach

A big drawback of grid-based methods is that the rely on the grids. In Figure 1 we see that the due to the ragid grid structure we might miss cluster points closely to the dense cells due to non-rectangular cluster shape. Furthermore, This is a limitation of grid-based methods. Density-based methods do not have this limitation, as they do not rely on grids. Instead, they rely on the density of the data points. A well-known density-based clustering algorithm is DBSCAN [3].

Description of SUBCLU and describe how it relates to DBSCAN.

3 MAFIA

3.1 Adaptive Grids

3.2 MAFIA Algorithm

- Simplified version of algorithm

3.3 Candidate Dense Units (CDUs)

- Why "any" dense unit

4 Evaluation

The evaluation of the clustering algorithms was performed on a Intel i7 1.70 GHz processor (12th gen.) with 16 GB of RAM running Windows 11.

The evaluation of MAFIA was performed using *GPUMAFIA* [1], which was installed on a virtual machine (VM) running Ubuntu. The VM was configured with 4 CPUs and 4 GB of RAM.

CLIQUE and SUBCLU were evaluated on the main machine using *ELKI* [10]. Thus, one should be careful to compare the results of the three algorithms directly, as the execution environment and the implementation may affect the results. However, the growth rate and their clustering of the data sets can be compared.

Throughout the different experiments, a range of input parameters for the three algorithms were tested. The best found were selected. The complete evaluation project can be found in the GitHub repository: https://github.com/henrikdchristensen/SDU-Data-Mining-Exam. Here, additional experiments as well as detailed descriptions of how to generate the data sets and how to install and use GPUMAFIA is described.

4.1 Data set generation

The aim for the synthetic data generation was to be able to produce similar data sets as discussed in [2,9]. That is, hyper-rectangles (axis-parallel) clusters in different subspaces. This was achieved by using *MDCGen*, where different sizes and different densities of the clusters could be determined, as well as which attributes for each cluster that are noise for which values at random is selected from a uniform distribution over the entire range of the attribute.

However, to see the one of the main advantages of SUBCLU compared to CLIQUE and MAFIA, a data set containing a Bezier-shaped cluster was created using *Artificial Cluster* (AC). In addition, a self-populated data set containing a plus-shaped cluster was created as discussed in [9].

Finally, two real-world data sets were selected to evaluate the algorithms in a more realistic setting.

All the data sets were normalized such that each attribute was in the range [0, 1].

4.2 Experimental Results

Scalability with Data Set Size To evaluate the scalability of the algorithms a data set containing 20 dimensions with 5 clusters in 5 different subspaces with 10% noise records was used. The data set size ranges from 10k to 1mio records. The results can be seen in Figure 2. As can be seen, MAFIA is the most scalable algorithm of the three. CLIQUE could handle up to 500k records, while SUBCLU was only able to handle up to 100k records. Similar results in terms of growth rate were reported in all three papers [9,2,8]. That is, linear growth for MAFIA and CLIQUE, and quadratic growth for SUBCLU. The linear behaviour of CLIQUE and MAFIA comes from the fact that the number of passes over the data set depends only on the dimensionality of the embedded cluster. An increase in the size of the data set just means that more data has to be scanned on every pass over the data set while finding dense units resulting in a linear growth rate. SUBCLU, on the other hand, has a quadratic growth rate, as it relies on the DBSCAN algorithm which needs partial range queries that can be costly in terms of running time.

Better running time of all three is achieved compared to the three papers. This is probably, due to better hardware.

Note that, the `minpts` in SUBCLU were scaled linearly with the data set size, as the clusters in the dateset were of a fixed size, which means, by increasing the data set size, the density of the clusters increases linearly. However, for the other two algorithms, there was no need to scale any of input parameters, as they relies on the density of the units for the total amount of records.

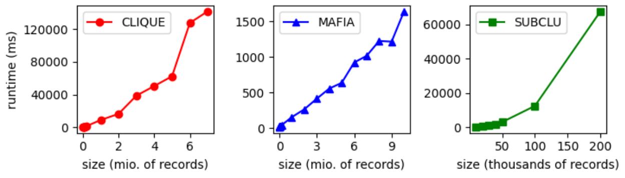


Fig. 2. Scalability with increasing data set size.

Clustering Accuracy Many different synthetic data sets were generated to test the clustering accuracy of the algorithms.

The first data set were a 2-dimensional data set containing a single cluster formed as a plus with 10% noise added, as discussed in [9]. MAFIA, was able to detect the 2-dimensional cluster almost completely, whereas CLIQUE only partly detects the cluster and the clusters overlaps. The accuracy of MAFIA comes however with a cost, as it also reports some lower-dimensional clusters as well. The results can be seen in Figure 3. The effect of adaptive grid sizes, clearly shows that MAFIA is more flexible than CLIQUE in such cases.

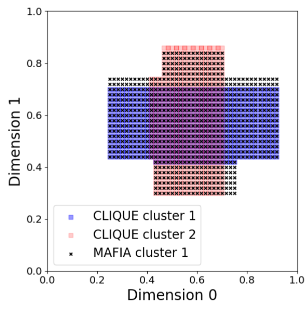
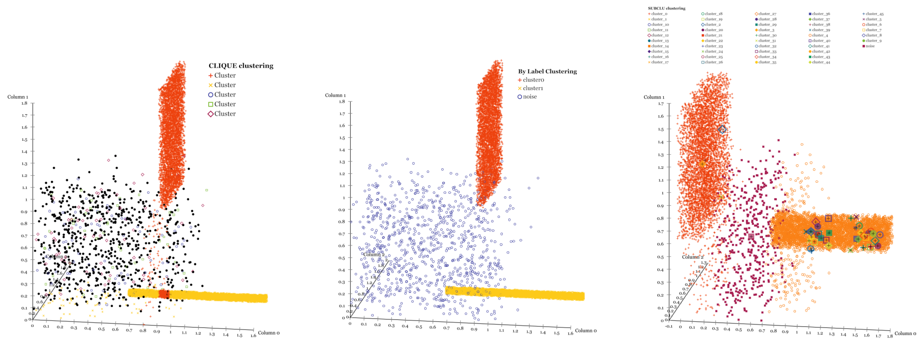


Fig. 3. A single plus-shaped cluster.

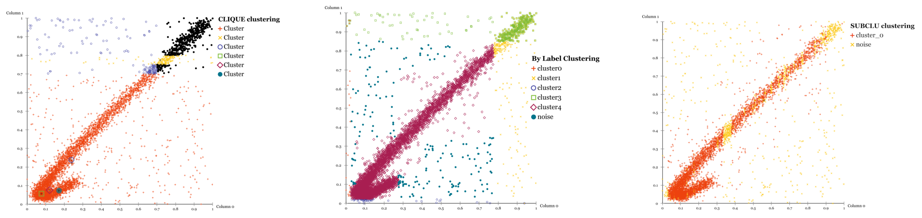
The second data set contains 10 dimensions with contains 2 clusters embedded in a different 4 dimensional subspace. 10% of the data was added as noise records. Similar data set as the one used in [9]. The results can be seen in Figure 4. Here, MAFIA was able to detect the clusters without any additional lower-dimensional clusters. In contrast, CLIQUE reports some overlapping clusters and some of the noise records as clusters. SUBCLU detects also the two clusters, but reports many lower-dimensional clusters and noise records as clusters as well.



(a) CLIQUE **(b)** MAFIA **(c)** SUBCLU

Fig. 4. Two clusters in 4 different subspaces.

The third data set were a 2-dimensional data set containing a single cluster formed as a Bezier curve with 10% noise added to the data set. As expected, SUBCLU outperforms the other two algorithms as can be seen in Figure 5. MAFIA and CLIQUE only partly detects the cluster and reports additional clusters and noise records as clusters. The result of SUBCLU demonstrates its locality assumption and its use of DBSCAN.



(a) CLIQUE **(b)** MAFIA **(c)** SUBCLU

Fig. 5. A single bezier-shaped cluster.

Scalability with Data Dimensionality and Cluster Dimensionality Figure 6 shows the scalability of the CLIQUE and MAFIA with increasing data set dimensionality. The data set contains 1 mio. records with 3 clusters in 5 different subspaces and 10% noise records, similar to the data set in [9]. The data set dimensionality ranges from 10 to 100 dimensions. However, CLIQUE was only able to handle half of the dimensions, as the PC constantly freezes – probably due to the high memory consumption. In [2] it is noted that CLIQUE has a quadratic growth rate in terms of data set dimensionality. To investigate this even further, one could try to reduce the memory consumption by having a smaller amount of records in the data set. Nevertheless, MAFIA is the most scalable algorithm of the two.

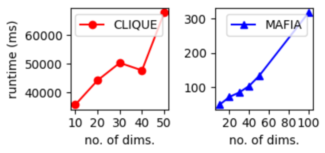


Fig. 6. Scalability with increasing data set dimensionality.

Figure 7 shows the scalability of the CLIQUE and MAFIA with increasing cluster dimensionality. A 20-dimensional data set, containing 500k records with a single cluster were used. The cluster was embedded in increasing number of dimensions starting from 10 to 100 dimensions. 10% of the records was added as noise. The data set is similar to the one used in [9]. Both algorithms heavily suffers from the increasing cluster dimensionality, however, MAFIA is able to handle a higher dimensional cluster than CLIQUE. The reason why both algorithms depends on the cluster dimensionality is due to the fact that a higher cluster dimensionality results in a large subspace coverage and a large number of CDUs. In other words, each pass on the data needs to populate a large number of CDUs, adn increase in cluster dimensionality also increases the number of passes over the data set.

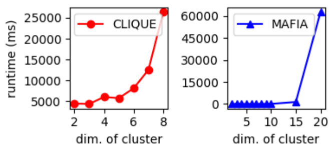


Fig. 7. Scalability with increasing cluster dimensionality.

The results reported for SUBCLU in [8] were also tried to be replicated, however, similar it was not possible evaluate in different scales of data set dimensionality and cluster dimensionality, as the PC constantly freezes – probably due to the high memory consumption. In future work, it would be interesting to investigate this further.

4.3 Sensitive analysis for MAFIA

From [9], MAFIA should not be sensitive to the choice of α. Therefore, a 20-dimensional data set with 1mio data points with 10% noise were set to evaluate this. β was fixed to 35%, and α was ranging from 0.8 to 5.2 in step size of 0.4. The results can be seen in Figure 8. From the results one could again draw the conclusion that MAFIA is not very sensitive to the α parameter. However, during evaluation of different types of data sets, many different values α was used to detect the right clusters. Also, different β values as well as maximum number of windows was needed to be set properly to achieve meaningful results.

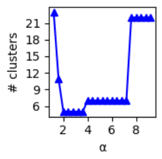
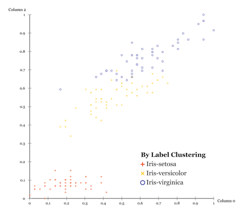


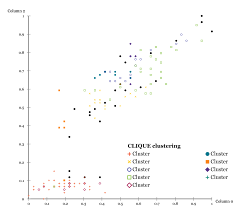
Fig. 8. Sensitivity of α for MAFIA.

Real Data Sets Two real-world data sets are used to evaluate the algorithms in a more realistic setting. Small data sets were selected for both performance and visualization reasons. Both data sets were normalized such that each attribute was in the range [0, 1].

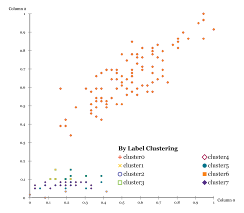
The first data set is the well-known *Iris* data set [4], which contains 150 records of 3 different iris types (e.g. Setosa, Versicolor, Virginica). The data set contains 4 features (sepal length, sepal width, petal length and petal width). All three algorithms produced some meaningful clusters, however, as can be seen in Figure 9. It is hard to determine which algorithm performs the best in this case.



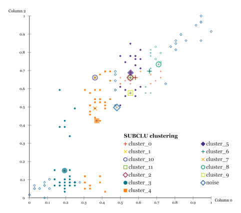
(a) Original



(b) CLIQUE



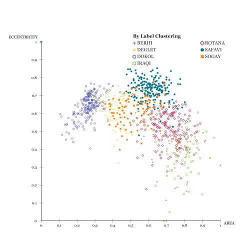
(c) MAFIA



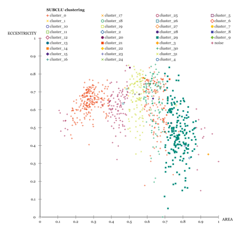
(d) SUBCLU

Fig. 9. Iris data set.

The second data set is the *Date Fruit* data set [6], which contains 898 records of 7 different date fruit types (e.g. Barhee, Deglet Nour, Sukkary, Rotab). These were obtained via a computer vision, where 34 features (e.g shape and color) was extracted. Only SUBCLU were able to produce meaningful results, see Figure 10. CLIQUE and MAFIA either produced way too many clusters or close to one single cluster.



(a) Original



(b) SUBCLU

Fig. 10. Date Fruit data set.

5 Discussion

- Discussion of the different approaches
- Discussion of results
- Limitations and Strengths
- Model: Input parameters; Assumptions on number, size, and shape of clusters; Noise
- Determinism
- Independence w.r.t. order of objects/attributes

- Assumptions on overlap/non-overlap of clusters/subspaces
- Effiency

Interpretability and usability: Clustering results should be easy to interpret and algorithms should produce clusters that are meaningful and comprehensible, making the results practical for decision-making.

Discovery of arbitrary-shaped clusters: Many clustering algorithms are limited to detecting spherical clusters based on distance measures (e.g. Euclidean distance). However, clusters in real-world data often take arbitrary shapes.

Minimize dependence on domain knowledge: Many clustering algorithms require users to provide specific input parameters, such as the number of clusters, which can be challenging to determine a prior. Reducing such parameters not only simplifies the process for users but also improve the reliability of the results.

Robust to noisy data: Real-world data is often noisy or contains outliers, which can distort clustering results. Clustering algorithms should be robust enough to handle noisy data, missing values, and outliers without degrading the quality of the clusters.

[5, p. 446-447]

6 Conclusion

Main findings

References

1. Adinetz, A., Kraus, J., Meinke, J., Pleiter, D.: Gpumafia: Efficient subspace clustering with mafia on gpus pp. 838–849 (08 2013). https://doi.org/10.1007/978-3-642-40047-6_83
2. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications p. 94–105 (1998). https://doi.org/10.1145/276304.276314, https://doi.org/10.1145/276304.276314
3. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: 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. p. 226–231. KDD'96, AAAI Press (1996)
4. Fisher, R.A.: Iris. UCI Machine Learning Repository (1936). https://doi.org/https://doi.org/10.24432/C56C76
5. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and techniques. Elsevier (6 2011)
6. Koklu, M., Kursun, R., Taspinar, Y.S., Cinar, I.: Classification of date fruits into genetic varieties using image analysis. Mathematical Problems in Engineering **2021**(1), 4793293 (2021). https://doi.org/https://doi.org/10.1155/2021/4793293, https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/4793293
7. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans. Knowl. Discov. Data **3**(1) (Mar 2009). https://doi.org/10.1145/1497577.1497578, https://doi.org/10.1145/1497577.1497578

8. Kröger, P., Kriegel, H.P., Kailing, K.: Density-connected subspace clustering for high-dimensional data **246-257** (04 2004). https://doi.org/10.1137/1.9781611972740.23
9. Nagesh, H., Goil, S., Choud, A., Choudhary, P.: Adaptive grids for clustering massive data sets (01 2002). https://doi.org/10.1137/1.9781611972719.7
10. Schubert, E.: Automatic indexing for similarity search in ELKI **13590**, 205–213 (2022). https://doi.org/10.1007/978-3-031-17849-8_16, https://doi.org/10.1007/978-3-031-17849-8_16