Density-Connected Subspace Clustering for High-Dimensional Data *

Karin Kailing Hans-Peter Kriegel Peer Kröger

Institute for Computer Science
University of Munich, Germany
{kailing,kriegel,kroegerp}@dbs.informatik.uni-muenchen.de

Abstract

Several application domains such as molecular biology and geography produce a tremendous amount of data which can no longer be managed without the help of efficient and effective data mining methods. One of the primary data mining tasks is clustering. However, traditional clustering algorithms often fail to detect meaningful clusters because most real-world data sets are characterized by a high dimensional, inherently sparse data space. Nevertheless, the data sets often contain interesting clusters which are hidden in various subspaces of the original feature space. Therefore, the concept of subspace clustering has recently been addressed, which aims at automatically identifying subspaces of the feature space in which clusters exist. In this paper, we introduce SUBCLU (density-connected *Sub*space *Clu*stering), an effective and efficient approach to the subspace clustering problem. Using the concept of density-connectivity underlying the algorithm DBSCAN [EKSX96], SUBCLU is based on a formal clustering notion. In contrast to existing grid-based approaches, SUBCLU is able to detect arbitrarily shaped and positioned clusters in subspaces. The monotonicity of density-connectivity is used to efficiently prune subspaces in the process of generating all clusters in a bottom up way. While not examining any unnecessary subspaces, SUBCLU delivers for each subspace the same clusters DBSCAN would have found, when applied to this subspace separately.

Keywords

Data mining, clustering, high dimensional data, subspace clustering

1 Introduction

Modern methods in several application domains such as molecular biology, astronomy, geography, etc. produce a tremendous amount of data. Since all this data can no longer be managed without the help of automated analysis tools, there is an ever increasing need for efficient and effective data mining methods to make use of the information contained implicitly in that data. One of the primary data mining tasks is clustering which is intended to help a user discovering and understanding the natural structure or grouping in a data set. In particular, clustering is the task of partitioning objects of a data set into distinct groups (clusters) such that two objects from one cluster are similar to each other, whereas two objects from distinct clusters are not.

A lot of work has been done in the area of clustering. Nevertheless, clustering real-world data sets is often hampered by the so called curse of dimensionality since many real-world data sets consist of a very high dimensional feature space. In general, most of the common algorithms fail to generate meaningful results because of the inherent sparsity of the objects. In such high dimensional feature spaces, data does not cluster anymore. But usually, there are clusters embedded in lower dimensional subspaces. In addition, objects can often be clustered differently in varying subspaces.

Gene expression data is a prominent example: Microarray chip technologies enable a user to measure the expression level of thousands of genes simultaneously. Roughly speaking, the expression level of a gene is a measurement for the frequency the gene is expressed (i.e. transcribed into its mRNA product). The expression level of a gene allows conclusions about the current amount of the protein in a cell the gene codes for. Usually, gene expression data appears as a matrix where the rows represent genes, and the columns represent samples (e.g. different experiments, time slots, test persons, etc.). The value of the i-th feature of a particular gene is the expression level of this gene in the i-th sample.

It is interesting from a biological point of view to cluster both the rows (genes) and the columns (samples) of the matrix, depending on the research scope. Clustering the genes is the method of choice if one searches for co-expressed genes, i.e. genes, whose expression levels are similar. Co-expression usually

*Supported by the German Ministry for Education, Science, Research and Technology (BMBF) under grant no. 031U112F within the BFAM (Bioinformatics for the Functional Analysis of Mammalian Genomes) project which is part of the German Genome Analysis Network (NGFN).

indicates that the genes are functionally related. If one searches for homogeneous groups in the set of samples, the problem is to cluster the samples. For example in cancer diagnostics, the samples may represent test persons. The ultimate goal of the clustering is then to distinguish between healthy and ill patients.

When clustering the genes to detect co-expressed genes, one has to cope with the problem, that usually the co-expression of the genes can only be detected in subsets of the samples. In other words, different subsets of the attributes (samples) are responsible for different co-expressions of the genes. When clustering the samples to identify e.g. homogeneous groups of patients this situation is even worse. As various phenotypes (e.g. hair color, gender, cancer, etc.) are hidden in varying subsets of the genes, the samples could usually be clustered differently according to these phenotypes, i.e. in varying subspaces.

A common approach to cope with the curse of dimensionality for data mining tasks such as clustering are methods to reduce the dimensionality of the data space. In general, dimensionality reduction methods map the whole feature space onto a lower-dimensional subspace of relevant attributes in which clusters can be found. The feature selection is usually based on attribute transformations by creating functions of attributes. Examples of such functions are principal component analysis (PCA) – also called Karhunen-Loève transformation (KLT) – used in multivariate statistics, e.g. [Jol86], methods based on singular value decomposition (SVD) used in information retrieval, e.g. [BDL95], and statistics, e.g. [Fuk90], and other transformations, for example based on wavelets [KCP01] or low frequency Fourier harmonics in conjunction with Parseval´s theorem [AFS93].

However, dimensionality reduction methods have major drawbacks: First, the transformed attributes often have no intuitive meaning any more and thus the resulting clusters are hard to interpret. Second, in some cases, dimensionality reduction does not yield the desired results (e.g. [AGGR98] present an example where PCA/KLT does not reduce the dimensionality). Third, using dimensionality reduction techniques, the data is clustered only in a particular subspace. The information of objects clustered differently in varying subspaces is lost.

A second approach for coping with clustering high-dimensional data is projected clustering, which aims at computing k pairs $(C_i, S_i)_{(0 \leq i \leq k)}$ where C_i is a set of objects representing the i-th cluster, S_i is a set of attributes spanning the subspace in which C_i exists (i.e. optimizes a given clustering criterion), and k is a user defined integer. Representative algorithms include

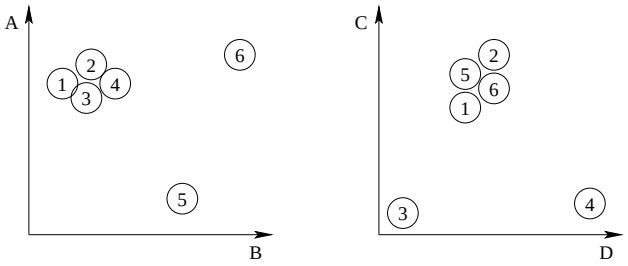


Figure 1: Drawback of the projected clustering approach.

the k-means related PROCLUS [AP99] and ORCLUS [AY00]. While the projected clustering approach is more flexible than dimensionality reduction, it also suffers from the fact that the information of objects which are clustered differently in varying subspaces is lost. Figure 1 illustrates this problem using a feature space of four attributes A,B,C, and D. In the subspace AB the objects 1 and 2 cluster together with objects 3 and 4, whereas in the subspace CD they cluster with objects 5 and 6. Either the information of the cluster in subspace AB or in subspace CD will be lost.

In recent years, the task of subspace clustering was introduced to overcome these problems. Subspace clustering is the task of automatically detecting clusters in subspaces of the original feature space. In this paper, we introduce a density-connected approach to subspace clustering overcoming the problems of existing approaches mentioned beneath. SUBCLU (density-connected *Sub*space *Clu*stering) is an effective answer to the problem of subspace clustering.

The remainder of the paper is organized as follows. In Section 2, we review current subspace clustering algorithms and point out our contributions to subspace clustering. The density-connected clustering notion and its application to subspace clustering is presented in Section 3. Section 4 describes our algorithm SUBCLU in full details. A broad experimental evaluation of SUBCLU based on artificial as well as on real-world data sets is presented in Section 5. Section 6 draws conclusions and points out future work.

2 Related Work and Contributions

2.1 Discussion of Recent Approaches for Subspace Clustering

Recent work has been done to tackle the problem of subspace clustering. In the following, current approaches are reviewed with no claim on completeness.

One of the first approaches to subspace clustering is CLIQUE (CLustering In QUEst) [AGGR98]. CLIQUE is a grid-based algorithm using an *apriori*-like method

to recursively navigate through the set of possible subspaces in a bottom-up way. The data space is first partitioned by an axis-parallel grid into equi-sized blocks of width ξ called *units*. Only units whose densities exceed a threshold τ are retained. Both ξ and τ are the input parameters of CLIQUE. The bottom-up approach of finding such dense units starts with 1-dimensional dense units. The recursive step from $(k-1)$-dimensional dense units to k-dimensional dense units takes $(k-1)$ dimensional dense units as candidates and generates the k-dimensional units by self-joining all candidates having the first $(k-2)$-dimensions in common. All generated candidates which are not dense are eliminated. For efficiency reasons, a pruning criterion called *coverage* is introduced to eliminate dense units lying in less "interesting" subspaces as soon as possible. For deciding whether a subspaces is interesting or not, the Minimum Description Length principle is used. Naturally this pruning bears the risk of missing some information. After generating all "interesting" dense units, clusters are found as a maximal set of connected dense units. For each k-dimensional subspace, CLIQUE takes all dense units of this subspace and computes disjoint sets of connected k-dimensional units. These sets are in a second step used to generate minimal cluster descriptions. This is done by covering each set of connected dense units with maximal regions and then determining the minimal cover.

A slight modification of CLIQUE is the algorithm ENCLUS (ENtropy-based CLUStering) [CFZ99]. The major difference is the criterion used for subspace selection. The criterion of ENCLUS is based on entropy computation of a discrete random variable. The entropy of any subspace S is high when the points are uniformly distributed in S whereas it is lower the more closely the points in S are packed. Subspaces with an entropy below an input threshold ω are considered as good for clustering. A monotonicity criterion is presented to be used for a similar bottom-up algorithm as in CLIQUE [CFZ99].

A more significant modification of CLIQUE is presented in [GNC99, NGC01] introducing the algorithm called MAFIA (Merging of Adaptive Finite IntervAls). MAFIA uses adaptive, variable-sized grids in each dimension. A dedicated technique based on histograms which aims at merging grid cells is used to reduce the number of bins compared to CLIQUE. An input parameter α is used as a so called *cluster dominance factor* to select bins which are α-times more densely populated (relative to their volume) than average. The algorithm starts to produce such one-dimensional dense units as candidates and proceeds recursively to higher dimensions. In contrast to CLIQUE, MAFIA uses any two k-dimensional dense units to construct a new $(k+1)$-dimensional candidate as soon as they share an arbitrary $(k-1)$-face (not only first dimensions). As a consequence, the number of generated candidates is much larger compared to CLIQUE. Neighboring dense units are merged to form clusters. Redundant clusters, i.e. clusters that are true subsets of higher dimensional clusters, are removed.

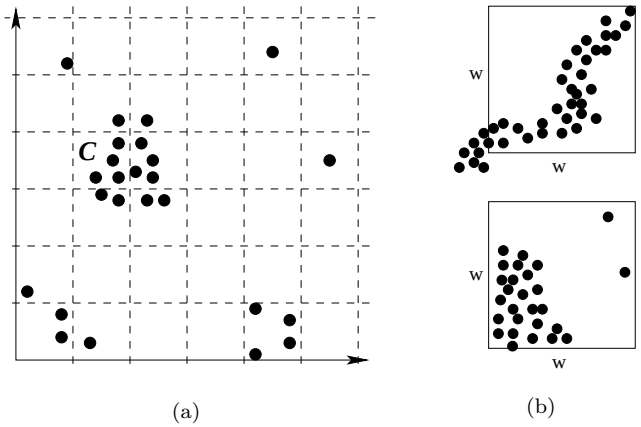


Figure 2: Drawbacks of existing subspace clustering algorithms (see text for explanation).

A big drawback of all these methods is caused by the use of grids. In general, grid-based approaches heavily depend on the positioning of the grids. Figure 2(a) illustrates this problem for CLIQUE: Each grid by itself is not dense, if $\tau > 4$, and thus, the cluster C is not found. On the other hand if $\tau = 4$, the cell with four objects in the lower right corner just above the x-axis is reported as a cluster. Clusters may also be missed if they are inadequately oriented or shaped.

Another recent approach called DOC [PJAM02] proposes a mathematical formulation for the notion of an optimal subspace cluster, regarding the density of points in subspaces. DOC is not grid-based but as the density of subspaces is measured using hypercubes of fixed width w, it has similar problems drafted in Figure 2(b). If a cluster is bigger than the hypercube, some objects may be missed. Furthermore, the distribution inside the hypercube is not considered, and thus it need not necessarily contain only objects of one cluster.

2.2 Contributions

In this paper, we propose a new approach which eliminates the problems mentioned above and enables the user to gain all the clustering information contained in high-dimensional data. Instead of using grids, we

adopt the notion of density-connectivity presented in [EKSX96] to the subspace clustering problem. This has the following advantages:

- Our algorithm SUBCLU is able to detect arbitrarily shaped and positioned clusters in subspaces.
- In contrast to CLIQUE and its successors, the underlying cluster notion is well defined.
- Since SUBCLU does not use any pruning heuristics like CLIQUE, it provides for each subspace the same clusters as if DBSCAN is applied to this subspace.

3 Density-Connected Subspace Clustering

3.1 Preliminary Definitions

Let DB be a data set of n objects. We assume, that DB is a database of d-dimensional feature vectors ($DB \subseteq \mathbb{R}^d$). All feature vectors have normalized values, i.e. all values fall into $[0, attrRange]$ for a fixed $attrRange \in \mathbb{R}^+$. Let $\mathcal{A} = \{a_1, \ldots, a_d\}$ be the set of all attributes a_i of DB. Any subset $S \subseteq \mathcal{A}$, is called a subspace. The cardinality of S ($|S|$) is called the dimensionality of S. The projection of an object o into a subspace $S \subseteq \mathcal{A}$ is denoted by $\pi_S(o)$. The distance function is denoted by $dist$. We assume that $dist$ is one of the L_p-norms.

3.2 Clusters as Density-Connected Sets

The density-based notion is a common approach for clustering used by various algorithms such as DBSCAN [EKSX96], DENCLUE [HK98], and OPTICS [ABKS99]. All these methods search for regions of high density in a feature space that are separated by regions of lower density. Our approach SUBCLU is particularly based on the formal definitions of density-connected clusters underlying the algorithm DBSCAN. The original formal definition of the clustering notion for the entire feature space is presented and discussed in full details in [EKSX96]. In the following, we adopt these definitions for the problem of subspace clustering. Let us note, that the density-connected clustering notion is well defined, and capable of finding clusters of arbitrary shapes. Using this clustering notion, SUBCLU is much more effective and soundly founded than recent approaches.

Definition 1. (ε-neighborhood)
Let $\varepsilon \in \mathbb{R}$, $S \subseteq \mathcal{A}$ and $o \in DB$. The ε-neighborhood *of o in S, denoted by $\mathcal{N}_\varepsilon^S(o)$, is defined by*

$$\mathcal{N}_\varepsilon^S(o) = \{x \in DB \mid dist(\pi_S(o), \pi_S(x)) \leq \varepsilon\}.$$

Based on two input parameters (ε and m), dense regions can be defined by means of core objects:

Definition 2. (core object)
Let $\varepsilon \in \mathbb{R}$, $m \in \mathbb{N}$, and $S \subseteq \mathcal{A}$. An object $o \in DB$ is called core object *in S, denoted by $\text{CORE}_{\varepsilon,m}^S(o)$, if its ε-neighborhood in S contains at least m objects, formally:*

$$\text{CORE}_{\varepsilon,m}^S(o) \Leftrightarrow |\mathcal{N}_\varepsilon^S(o)| \geq m.$$

Usually clusters contain several core objects located inside a cluster and border objects located at the border of the cluster. In addition, objects within a clusters should be "connected". These observations led to the following concepts.

Definition 3. (direct density-reachability)
Let $\varepsilon \in \mathbb{R}$, $m \in \mathbb{N}$, and $S \subseteq \mathcal{A}$. An object $p \in DB$ is directly density-reachable *from $q \in DB$ in S if q is a core object in S and p is an element of $\mathcal{N}_\varepsilon^S(q)$, formally:*

$$\text{DIRREACH}_{\varepsilon,m}^S(q,p) \Leftrightarrow \text{CORE}_{\varepsilon,m}^S(q) \wedge p \in \mathcal{N}_\varepsilon^S(q).$$

Definition 4. (density-reachability)
Let $\varepsilon \in \mathbb{R}$, $m \in \mathbb{N}$, and $S \subseteq \mathcal{A}$. An object $p \in DB$ is density-reachable *from $q \in DB$ in S if there is a chain of objects $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i, formally:*

$$\begin{aligned}&\text{REACH}_{\varepsilon,m}^S(q,p) \Leftrightarrow \\ &\exists p_1, \ldots, p_n \in DB : p_1 = q \wedge p_n = p \wedge \\ &\forall i \in \{1 \ldots n-1\} : \text{DIRREACH}_{\varepsilon,m}^S(p_i, p_{i+1}).\end{aligned}$$

Definition 5. (density-connectivity)
Let $\varepsilon \in \mathbb{R}$, $m \in \mathbb{N}$, and $S \subseteq \mathcal{A}$. An object $p \in DB$ is density-connected *to an object $q \in DB$ in S if there is an object o such that both p and q are density-reachable from o, formally:*

$$\begin{aligned}&\text{CONNECT}_{\varepsilon,m}^S(q,p) \Leftrightarrow \\ &\exists o \in DB : \text{REACH}_{\varepsilon,m}^S(o,q) \wedge \text{REACH}_{\varepsilon,m}^S(o,p).\end{aligned}$$

Definition 6. (density-connected set)
Let $\varepsilon \in \mathbb{R}$, $m \in \mathbb{N}$, and $S \subseteq \mathcal{A}$. A non-empty subset $C \subseteq DB$ is called a density-connected set *in S if all objects in C are density-connected in S, formally:*

$$\text{CONSET}_{\varepsilon,m}^S(C) \Leftrightarrow \forall o,q \in C : \text{CONNECT}_{\varepsilon,m}^S(o,q).$$

Finally, a density-connected cluster is defined as a set of density-connected objects which is maximal w.r.t. density-reachability [EKSX96]. This definition can easily be adopted to clusters in a particular subspace, analogously.

3.3 Monotonicity of Density-Connected Sets

A straightforward approach would be to run DBSCAN in all possible subspaces to detect all density-connected clusters. The problem is, that the number of subspaces is 2^d. A more effective strategy would be to use the clustering information of previous subspaces in the process of generating all clusters and drop all subspaces that cannot contain any density-connected clusters.

Unfortunately, density-connected clusters are not monotonic, i.e. if $C \subseteq DB$ is a density-connected cluster in subspace $S \subseteq \mathcal{A}$, it need not be a density-connected cluster in any $T \subseteq S$. The reason for this is that in T the density-connected cluster C need not be maximal w.r.t. density-reachability any more. There may be additional objects which are not in C but are density-reachable in T from an object in C.

However, density-connected sets are monotonic. In fact, if $C \subseteq DB$ is a density-connected set in subspace $S \subseteq \mathcal{A}$ then, C is also a density-connected set in any subspace $T \subseteq S$.

LEMMA 3.1. (MONOTONICITY)
Let $\varepsilon \in \mathbb{R}$, $m \in \mathbb{N}$, $o, q \in DB$, $C \subseteq DB$, where $C \neq \emptyset$ and $S \subseteq \mathcal{A}$. Then the following monotonicity properties hold:

$\forall T \subseteq S$:

(1) $\text{CORE}^S_{\varepsilon,m}(o) \Rightarrow \text{CORE}^T_{\varepsilon,m}(o)$

(2) $\text{DIRREACH}^S_{\varepsilon,m}(o,q) \Rightarrow \text{DIRREACH}^T_{\varepsilon,m}(o,q)$

(3) $\text{REACH}^S_{\varepsilon,m}(o,q) \Rightarrow \text{REACH}^T_{\varepsilon,m}(o,q)$

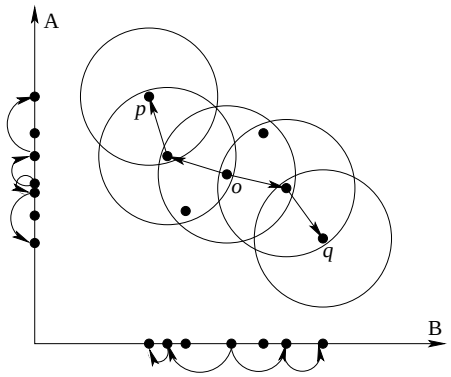
(4) $\text{CONNECT}^S_{\varepsilon,m}(o,q) \Rightarrow \text{CONNECT}^T_{\varepsilon,m}(o,q)$

(5) $\text{CONSET}^S_{\varepsilon,m}(o,q) \Rightarrow \text{CONSET}^T_{\varepsilon,m}(o,q)$

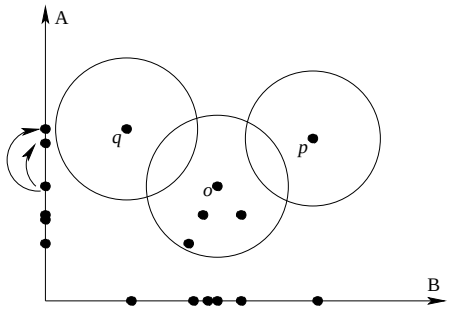
Proof. See Appendix A

The monotonicity of density-connectivity is illustrated in Figure 3. In Figure 3(a), p and q are density-connected via o in the subspace spanned by attributes A and B. Thus, p and q are also density-connected via o in each subspace A and B of AB. The inverse conclusion is depicted in Figure 3(b): p and q are not density-connected in subspace B. Thus they are also not density-connected in the superspace AB although they are density-connected in subspace A via o.

The inversion of Lemma 3.1(5) is the key idea for an efficient bottom-up algorithm to detect the density-connected sets in all subspaces of high dimensional data. Due to this inversion, we do not have to examine any subspace S if at least one $T_i \subset S$ contains no cluster (i.e. a density connected set). On the other hand, we have to test each subspace S if all $T_i \subset S$ contain clusters whether those clusters are conserved.



(a) p and q are density-connected via o



(b) p and q are not density-connected

Figure 3: Monotonicity of density-connectivity (the circles indicate the ε-neighborhoods, $m = 4$).

4 The Algorithm SUBCLU

SUBCLU is based on a bottom-up, greedy algorithm to detect the density-connected clusters in all subspaces of high dimensional data. The algorithm is presented in Figure 4. The following data structures are used (c.f. Figure 4):

- $\mathcal{C}^S$ denotes the set of all density-connected clusters of DB in subspace S (w.r.t. ε and m) and can be computed by the procedure $DBSCAN(DB, S, \varepsilon, m)$ (the input parameters ε and m are fixed), i.e.
  $\mathcal{C}^S := DBSCAN(DB, S, \varepsilon, m)$.

- $\mathcal{S}_k$ denotes the set of all k-dimensional subspaces containing at least one cluster, i.e.
  $\mathcal{S}_k := \{S \,|\, |S| = k \text{ and } \mathcal{C}^S \neq \emptyset\}$.

- $\mathcal{C}_k$ denotes the set of sets of all clusters in k-dimensional subspaces, i.e.
  $\mathcal{C}_k := \{\mathcal{C}^S \,|\, |S| = k\}$.

```
SUBCLU(SetOfObjects DB, Real ε, Integer m)
    /* STEP 1 Generate all 1-D clusters */
    S_1 := ∅    // set of 1-D subspaces containing clusters
    C_1 := ∅    // set of all sets of clusters in 1-D subspaces
    FOR each a_i ∈ A DO
        C^{a_i} := DBSCAN(DB, {a_i}, ε, m)    // set of all clusters in subspace a_i;
        IF C^{a_i} ≠ ∅ THEN    // at least one cluster in subspace {a_i} found
            S_1 := S_1 ∪ {a_i};
            C_1 := C_1 ∪ C^{a_i};
        END IF
    END FOR
    /* STEP 2 Generate (k+1)-D clusters from k-D clusters */
    k := 1;
    WHILE C_k ≠ ∅
        /* STEP 2.1 Generate (k+1)-dimensional candidate subspaces */
        CandS_{k+1} := GenerateCandidateSubspaces(S_k);
        /* STEP 2.2 Test candidates and generate (k+1)-dimensional clusters */
        FOR EACH cand ∈ CandS_{k+1} DO
            // Search k-dim subspace of cand with minimal number of objects in the clusters
            bestSubspace := min_{s∈S_k ∧ s⊆cand} Σ_{C_i∈C^s} |C_i|
            C^{cand} := ∅;
            FOR EACH cluster cl ∈ C^{bestSubspace} DO
                C^{cand} = C^{cand} ∪ DBSCAN(cl, cand, ε, m);
                IF C^{cand} ≠ ∅ THEN
                    S_{k+1} := S_{k+1} ∪ cand;
                    C_{k+1} := C_{k+1} ∪ C^{cand};
                END IF
            END FOR
        END FOR
        k := k + 1
    END WHILE
```

Figure 4: The SUBCLU algorithm.

We begin with generating all 1-dimensional clusters by applying DBSCAN to each 1-dimensional subspace (STEP 1 in Figure 4).

For each detected cluster we have to check, whether this cluster is (or parts of it are) still existent in higher dimensional subspaces. Due to Lemma 3.1 no other clusters can exist in higher dimensional subspaces. Thus, for each k-dimensional subspace $S \in \mathcal{S}_k$, we search all other k-dimensional subspaces $T \in \mathcal{S}_k$ ($T \neq S$) having $(k-1)$ attributes in common and join them to generate $(k+1)$-dimensional candidate subspaces (STEP 2.1.1 of the procedure GenerateCandidates in Figure 5). The set of $(k+1)$-dimensional candidate subspaces is denoted by $CandS_{k+1}$.

Due to Lemma 3.1, for each candidate subspace $S \in CandS_{k+1}$, $\mathcal{S}_k$ must contain each k-dimensional subspace $T \subset S$ ($|T| = k$), we can prune these candidates having at least one k-dimensional subspace not included in $\mathcal{S}_k$ (STEP 2.1.2 of procedure GenerateCandidates in Figure 5). This reduces the number of $(k+1)$-dimensional candidate subspaces.

In the last step (STEP 2.2 in Figure 4) we generate the $(k+1)$-dimensional clusters and the corresponding $(k+1)$-dimensional subspaces containing these clusters using the k-dimensional subclusters and the list of $(k+1)$-dimensional candidate subspaces. For that purpose, we simply have to do the following: for each candidate subspace $cand \in CandS_{k+1}$ we take one k-dimensional subspace $T \subset cand$ and simply call the procedure $DBSCAN(cl, cand, \varepsilon, m)$ for each cluster cl in T ($cl \in \mathcal{C}^T$) to generate $\mathcal{C}^{cand}$. To minimize the cost of the runs of DBSCAN in $cand$, we choose that subspace $bestSubspace \subset cand$ from $\mathcal{S}_k$ in which a minimum number of objects are in the cluster, i.e.

$$bestSubspace := \min_{s \in \mathcal{S}_k \wedge s \subseteq cand} \sum_{C_i \in \mathcal{C}^s} |C_i|$$

These heuristics minimize the number of range queries necessary during the runs of DBSCAN in S. If $\mathcal{C}^S \neq \emptyset$, we add it to $\mathcal{C}_{k+1}$ and add S to $\mathcal{S}_{k+1}$.

Steps 2.1 to 2.3 are recursively executed as long as the set of k-dimensional subspaces containing clusters is not empty.

```
GenerateCandidates(SetOfSubspaces S_k)
    /* STEP 2.1.1 Generate (k+1)-dimensional candidate subspaces */
    Cand S_{k+1} := ∅;
    FOR each s_1 ∈ S_k DO
        FOR each s_2 ∈ S_k DO
            IF s_1.attr_1 = s_2.attr_1 ∧ ... ∧ s_1.attr_{k-1} = s_2.attr_{k-1} ∧ s_1.attr_k < s_2.attr_k THEN
                insert {s_1.attr_1, ..., s_1.attr_k, s_2.attr_k} into Cand S_{k+1};
            END IF
        END FOR
    END FOR
    /* STEP 2.1.2 Prune irrelevant candidates subspaces */
    FOR EACH cand ∈ Cand S_{k+1} DO
        FOR EACH s ⊂ cand with |s| = k DO
            IF s ∉ S_k THEN delete cand from Cand S_{k+1};
            END IF
        END FOR
    END FOR
```

Figure 5: Procedure GenerateCandidates.

The most time consuming part of our algorithm are all the partial range queries (range queries on arbitrary subspaces of the data space) necessary for the DBSCAN algorithm. As DBSCAN is applied to different subspaces, an index structure for the full-dimensional data space is not applicable. Therefore we apply the approach of inverted files. Our algorithm provides an efficient index support for range queries on each single attribute in logarithmic time. For range queries on more than one attribute, we apply the range query to each separate attribute (index structure) and generate the intersection of all intermediate results to obtain the final result.

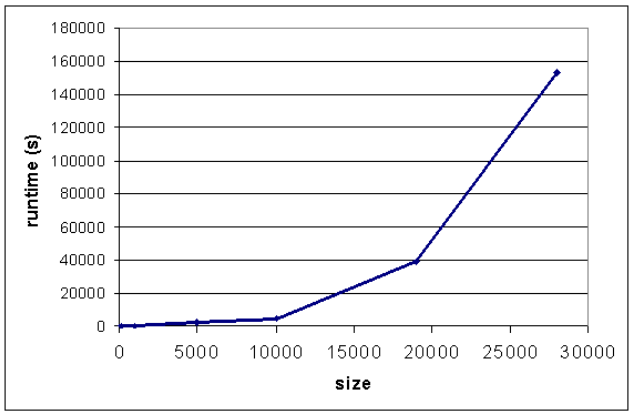


Figure 6: Scalability of SUBCLU against the size of the data set.

5 Performance Evaluation

We tested SUBCLU using several synthetic data sets and a real world gene expression data set. All experiments were run on a workstation with a 1.7 GHz processor and 2 GB RAM.

5.1 Data Sets

We tested SUBCLU using synthetic as well as real world gene expression data sets. The synthetic data sets were generated by a self-implemented data generator. It permits to control the size and structure of the generated data sets through parameters such as number and dimensionality of subspace clusters, dimensionality of the feature space and density parameters for the whole data set as well as for each cluster. In a subspace that contains a cluster, the average density of data points in that cluster is much larger than the density of points not belonging to the cluster in this subspace. In addition, it is ensured, that none of the synthetically generated data sets can be clustered in the full dimensional space. The gene expression data set [SSZ+98] studies the yeast mitotic cell cycle. We used only the data set of the CDC15 mutant. The expression level of 6000 genes was measured at 24 different time slots. Since some genes have missing expression values and the handling of missing values in gene expression analysis is a non-trivial task, we eliminated those genes from our test data set. The resulting data set contains around 4000 genes expressed at 24 different time slots.

5.2 Efficiency

We evaluated the efficiency of SUBCLU using several synthetic data sets. All tests were run with $MinPts = 8$ and $\varepsilon = 2.0$.

The scalability of SUBCLU against the size of the data set, the dimensionality of the data set and the dimensionality of the hidden subspace clusters are depicted in Figures 6, 7, and 8, respectively. In all three

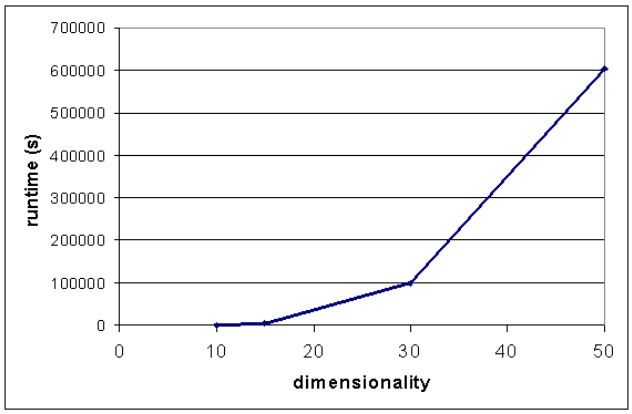


Figure 7: Scalability of SUBCLU against the dimensionality of the dataset.

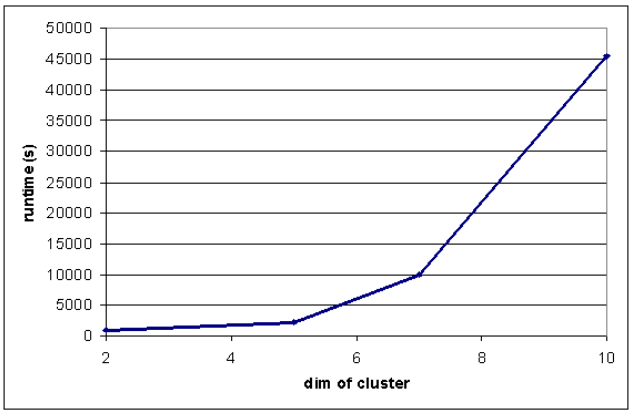


Figure 8: Scalability of SUBCLU against the maximum dimensionality of the hidden subspace clusters.

cases, SUBCLU grows with an at least quadratic factor. The reason for this scalability w.r.t. the size of the data set is that SUBCLU performs multiple range queries in arbitrary subspaces. As mentioned above, we can only support these queries using inverted files, since there is no index structure that can support partial range queries in average case logarithmic time. The scalability to the dimensionality of the data set and of the hidden subspaces can be explained by the *Apriori*-like bottom-up greedy algorithm underlying SUBCLU to navigate through the space of all possible subspaces.

5.3 Accuracy

To evaluate the effectivity of SUBCLU we compared it with CLIQUE [AGGR98]. Since CLIQUE is a product of IBM and its code is not easy to obtain, we re-implemented CLIQUE according to [AGGR98]. In all accuracy experiments, we run CLIQUE with a broad range of parameter settings and took only the best results.

We applied SUBCLU and CLIQUE to several synthetic data sets which we generated as described above. In each data set, several clusters are hidden in subspaces of varying dimensionality. The results are depicted in Table 1. In almost all cases, SUBCLU computed the artificial clusters whereas CLIQUE had difficulties in detecting all patterns properly. In addition, CLIQUE split usually connected clusters into several distinct clusters (not mentioned in the table).

We also used SUBCLU to find co-expressed genes based on gene expression data.

SUBCLU found many interesting clusters in several subspaces of this data set. The most interesting clusters were found in the subspaces spanned by time slots 90, 110, 130, and 190 as well as time slots 190, 270, and 290. The functional relationships of the genes in the resulting clusters were investigated using the public yeast genome database at Stanford University (Saccharomyces Genome Database, SGD: http://www.yeastgenome.org/). Let us note, that each cluster contains additional genes with yet unknown function.

The contents of four sample clusters in two different subspaces are depicted in Table 2. The first cluster (in subspace spanned by time slots 90, 110, 130, 190) contains several genes which are known to play a role during the cell cycle such as DOM34, CKA1, CPA1, and MIP6. In addition, the products of two genes in that cluster are part of a common protein complex. The second cluster contains the gene STE12, identified by [SSZ+98] as an important transcription factor for the regulation of the mitotic cell cycle. In addition, the genes CDC27 and EMP47 which have possible STE12-sites and are most likely co-expressed with STE12 are in that cluster. The third cluster consists of the genes CDC25 (starting point for mitosis), MYO3 and NUD1 (known for an active role during mitosis) as well as various other transcription factors (e.g. CHA4, ELP3) required during the cell cycle. The fourth cluster contains several mitochondrion related genes which have similar functions. For example, the genes MRPL17, MRPL31, MRPL32, and MRPL33 are four mitochondrial large ribosomal subunits, the genes UBC1 and UBC4 are subunits of a certain protease, and the genes SNF7 and VPS4 are direct interaction partners. This indicates a higher mitochondrial activity at these time spots, which might be explained by a higher demand of biological energy during the cell cycle (the energy metabolism is located in mitochondrion).

Let us note, that the described four clusters are only a representative glance at the results SUBCLU yields

Table 1: Comparative evaluation of SUBCLU and CLIQUE: summary of the results on synthetic data sets.

Data set	d	dim. of subspace cluster	N	# generated clusters	true clusters found by	
					SUBCLU	CLIQUE
DS01	10	4	18999	1	1	1
DS02	10	4	27704	1	1	1
DS03	15	5,5	3802	3	3	1
DS04	15	3,5,7	4325	3	2	1
DS05	15	5,5,5	4057	3	3	1
DS06	15	4,4,6,7,10	2671	6	5	2

when applied on the gene expression data set. The resulting clusters contained functional related genes and thus, their co-expression is biological meaningful. Since most clusters also consists genes which have not yet any annotated function, the results of SUBCLU might propose a biologically interesting prediction for these unknown genes.

We also applied CLIQUE to the gene expression data set. We again tested a broad range of parameter settings and compared SUBCLU to the best results of CLIQUE. Since the parameter ξ of CLIQUE (width of grid cells) affects the runtime of CLIQUE heavily, we were forced to run CLIQUE with rather low values for ξ. As a consequence, CLIQUE was not able to find any reasonable clusters in the gene expression data set. Let us note, that a more efficient implementation of CLIQUE would enable a better parameter setting (i.e. higher values for ξ) and would thus also detect some of the clusters computed by SUBCLU. On the other hand, in real world data sets such as gene expression data, it is most likely that the clusters are not axis-parallel hypercubes. Thus, SUBCLU is much more suitable than CLIQUE due to the fact, that the density-connected clustering notion underlying SUBCLU is able to detect arbitrarily shaped (subspace) clusters.

6 Conclusions

In this paper, we presented SUBCLU, a density-based subspace clustering algorithm for detecting clusters in high dimensional data. Built on an adaption of the density-connected notion of clusters underlying the algorithm DBSCAN, we developed an efficient greedy-algorithm to compute all density-connected sets hidden in subspaces of high dimensional data. A comparison with CLIQUE empirically showed, that SUBCLU outperforms state-of-the-art subspace clustering algorithms in quality. An application of SUBCLU to real-world gene expression data yields biologically interesting and meaningful results, and thus demonstrates the very usefulness of SUBCLU.

An approach for future work is the development of an efficient index structure for partial range queries (range queries in arbitrary subspaces of the original data space). Since the inverted files used by SUBCLU are less effective the more dimensions are relevant to the range query, a better index support could further improve the efficiency of SUBCLU. Moreover, a parallel version of SUBCLU is envisioned for further improving its scalability.

Acknowledgments

We want to acknowledge the work of Alexandra Boshammer and Elke Schumm supporting the implementation of SUBCLU and CLIQUE, respectively.

References

[ABKS99] M Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. "OPTICS: Ordering Points to Identify the Clustering Structure". In *Proc. ACM SIGMOD Int. Conf. on Management of Data, Philadelphia, PA*, pages 49–60, 1999.

[AFS93] R Agrawal, C Faloutsos, and A Swami. "Efficient Similarity Search in Sequence Databases". In *Proc. 4th Int. Conf. on Foundations of Data Organization and Algorithms, Chicago, IL*, 1993.

[AGGR98] R Agrawal, J Gehrke, D Gunopulos, and P Raghavan. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications". In *Proc. ACM SIGMOD Int. Conf. on Management of Data, Seattle, WA*, 1998.

[AP99] C C Aggarwal and C Procopiuc. "Fast Algorithms for Projected Clustering". In *Proc. ACM SIGMOD Int. Conf. on Management of Data, Philadelphia, PA*, 1999.

[AY00] C.C. Aggarwal and P.S. Yu. "Finding Generalized Projected Clusters in High Dimensional Space". In *Proc. ACM SIGMOD Int. Conf. on Management of Data, Dallas, TX*, 2000.

[BDL95] M Berry, S Dumais, and T Landauer. "Using Linear Algebra for Intelligent Information Retrieval". *SIAM Review*, 37(4):573–595, 1995.

[CFZ99] C.-H. Cheng, A.W.-C. Fu, and Y. Zhang. "Entropy-Based Subspace Clustering for Mining Numerical Data". In *Proc. ACM SIGKDD Int. Conf.*

Gene Name	Function
Cluster 1 (subspace 90, 110, 130, 190)	
RPC40	subunit of RNA pol I and III, builds complex with CDC60
CDC60	tRNA synthesase, builds complex with RPC40
FRS1	tRNA synthesase
DOM34	protein synthesis, mitotic cell cycle
CKA1	mitotic cell cycle control
CPA1	control of translation
MIP6	RNA binding activity, mitotic cell cycle
Cluster 2 (subspace 90, 110, 130, 190)	
STE12	transcription factor (regulation of cell cycle)
CDC27	regulation of cell cycle, possible STE12-site
EMP47	Golgi membrane protein, possible STE12-site
XBP1	Transcription factor
Cluster 3 (subspace 90, 110, 130, 190)	
CDC25	starting control factor for mitosis
MYO3	control/regulation factor for mitosis
NUD1	control/regulation factor for mitosis
Cluster 4 (subspace 190, 270, 290)	
RPT6	protein catabolism; builds complex with RPN10
RPN10	protein catabolism; builds complex with RPT6
UBC1	protein catabolism; subunit of 26S protease
UBC4	protein catabolism; subunit of 26S protease
MRPL17	component of mitochondrial large ribosomal subunit
MRPL31	component of mitochondrial large ribosomal subunit
MRPL32	component of mitochondrial large ribosomal subunit
MRPL33	component of mitochondrial large ribosomal subunit
SNF7	direct interaction with VPS2
VPS4	mitochondrial protein; direct interaction with SNF7

Table 2: Contents of four sample clusters in different subspaces.

on Knowledge Discovery in Databases, San Diego, FL, 1999.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR*, pages 291–316, 1996.

[Fuk90] K Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA, 1990.

[GNC99] S. Goil, H.S. Nagesh, and A. Choudhary. "MAFIA: Efficiant and Scalable Subspace Clustering for Very Large Data Sets". Tech. Report No. CPDC-TR-9906-010, Center for Parallel and Distributed Computing, Dept. of Electrical and Computer Engineering, Northwestern University, 1999.

[HK98] A Hinneburg and D A Keim. "An Efficient Approach to Clustering in Large Multimedia Databases with Noise". In *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, New York City, NY*, pages 224–228, 1998.

[HK99] A. Hinneburg and D. Keim. "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering". In *Proc. 25th Int. Conf. on Very Large Databases, Edinburgh, Scotland*, 1999.

[Jol86] I Joliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[KCP01] E Keogh, K Chakrabarti, and M Pazzani. "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data, Santa Barbara, CA*, 2001.

[NGC01] H.S. Nagesh, S. Goil, and A. Choudhary. "Adaptive Grids for Clustering Massive Data Sets". In *1st SIAM Int. Conf. on Data Mining, Chicago, IL*, 2001.

[PJAM02] Cecilia M Procopiuc, Michael Jones, Pankaj K Agarwal, and T M Murali. "A Monte Carlo Algorithm for Fast Projective Clustering". In *Proc. ACM SIGMOD Int. Conf. on Management of Data, Madison, WI*, pages 418–427, 2002.

[SSZ+98] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P Brown, D. Botstein, and B. Futcher. "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization.". *Molecular Biolology of the Cell*, 9:3273–3297, 1998.

A Formal Proofs

Proof of Lemma 3.1

(1) $\textsc{Core}^S_{\varepsilon,m}(o)$

$$\begin{aligned}
&\Leftrightarrow |\mathcal{N}^S_\varepsilon(o)| \geq m \\
&\Leftrightarrow |\{x \,|\, dist(\pi_S(o), \pi_S(x)) \leq \varepsilon\}| \geq m \\
&\Leftrightarrow |\{x \,|\, \sqrt[p]{\sum_{a_i \in S} (\pi_{a_i}(o) - \pi_{a_i}(x))^p} \leq \varepsilon\}| \geq m \\
&\overset{(T \subseteq S)}{\Rightarrow} |\{x \,|\, \sqrt[p]{\sum_{a_i \in T} (\pi_{a_i}(o) - \pi_{a_i}(x))^p} \leq \varepsilon\}| \geq m \\
&\Leftrightarrow |\{x \,|\, dist(\pi_T(o), \pi_T(x)) \leq \varepsilon\}| \geq m \\
&\Leftrightarrow |\mathcal{N}^T_\varepsilon(o)| \geq m \\
&\Leftrightarrow \textsc{Core}^T_{\varepsilon,m}(o)
\end{aligned}$$

(2) $\textsc{DirReach}^S_{\varepsilon,m}(o,q)$

$$\begin{aligned}
&\Leftrightarrow \textsc{Core}^S_{\varepsilon,m}(o) \wedge q \in \mathcal{N}^S_\varepsilon(o) \\
&\Leftrightarrow \textsc{Core}^S_{\varepsilon,m}(o) \wedge dist(\pi_S(o), \pi_S(q)) \leq \varepsilon \\
&\Leftrightarrow \textsc{Core}^S_{\varepsilon,m}(o) \wedge \sqrt[p]{\sum_{a_i \in S} (\pi_{a_i}(o) - \pi_{a_i}(q))^p} \leq \varepsilon \\
&\overset{(T \subseteq S)}{\Longrightarrow}{}^{(1)} \textsc{Core}^T_{\varepsilon,m}(o) \wedge \sqrt[p]{\sum_{a_i \in T} (\pi_{a_i}(o) - \pi_{a_i}(q))^p} \leq \varepsilon \\
&\Leftrightarrow \textsc{Core}^T_{\varepsilon,m}(o) \wedge dist(\pi_T(o), \pi_T(q)) \leq \varepsilon \\
&\Leftrightarrow \textsc{Core}^T_{\varepsilon,m}(o) \wedge q \in \mathcal{N}^T_\varepsilon(o) \\
&\Leftrightarrow \textsc{DirReach}^T_{\varepsilon,m}(o,q)
\end{aligned}$$

(3) $\textsc{Reach}^S_{\varepsilon,m}(o,q)$

$$\begin{aligned}
&\Leftrightarrow \exists \mathrm{p}_1, \ldots, \mathrm{p}_\mathrm{n} \in DB : p_1 = o \wedge p_n = q \wedge \forall i \in \{1 \ldots n-1\} : \textsc{DirReach}^S_{\varepsilon,m}(p_i, p_{i+1}) \\
&\overset{(T \subseteq S)}{\Longrightarrow}{}^{(2)} \exists \mathrm{p}_1, \ldots, \mathrm{p}_\mathrm{n} \in DB : p_1 = o \wedge p_n = q \wedge \forall i \in \{1 \ldots n-1\} : \textsc{DirReach}^T_{\varepsilon,m}(p_i, p_{i+1}) \\
&\Leftrightarrow \textsc{Reach}^T_{\varepsilon,m}(o,q)
\end{aligned}$$

(4) $\textsc{Connect}^S_{\varepsilon,m}(o,q)$

$$\begin{aligned}
&\Leftrightarrow \exists x \in DB : \textsc{Reach}^S_{\varepsilon,m}(x,o) \wedge \textsc{Reach}^S_{\varepsilon,m}(x,q) \\
&\overset{(T \subseteq S)}{\Longrightarrow}{}^{(3)} \exists x \in DB : \textsc{Reach}^T_{\varepsilon,m}(x,o) \wedge \textsc{Reach}^T_{\varepsilon,m}(x,q) \\
&\Leftrightarrow \textsc{Connect}^T_{\varepsilon,m}(o,q)
\end{aligned}$$

(5) $\textsc{ConSet}^S_{\varepsilon,m}(C)$

$$\begin{aligned}
&\Leftrightarrow \forall o,q \in C : \textsc{Connect}^S_{\varepsilon,m}(o,q) \\
&\overset{(T \subseteq S)}{\Longrightarrow}{}^{(4)} \forall o,q \in C : \textsc{Connect}^T_{\varepsilon,m}(o,q) \\
&\Leftrightarrow \textsc{ConSet}^T_{\varepsilon,m}(C)
\end{aligned}$$