

Detection and Visualization of Subspace Cluster Hierarchies

Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, Ina Müller-Gorman,
Arthur Zimek

Institute for Informatics, Ludwig-Maximilians-Universität München, Germany
{achtert,boehm,kriegel,kroegerp,muellerg,zimek}@dbs.ifi.lmu.de
WWW home page: <http://www.dbs.ifi.lmu.de>

Abstract. Subspace clustering (also called projected clustering) addresses the problem that different sets of attributes may be relevant for different clusters in high dimensional feature spaces. In this paper, we propose the algorithm DiSH (Detecting Subspace cluster Hierarchies) that improves in the following points over existing approaches: First, DiSH can detect clusters in subspaces of significantly different dimensionality. Second, DiSH uncovers complex hierarchies of nested subspace clusters, i.e. clusters in lower-dimensional subspaces that are embedded within higher-dimensional subspace clusters. These hierarchies do not only consist of single inclusions, but may also exhibit multiple inclusions and thus, can only be modeled using graphs rather than trees. Third, DiSH is able to detect clusters of different size, shape, and density. Furthermore, we propose to visualize the complex hierarchies by means of an appropriate visualization model, the so-called subspace clustering graph, such that the relationships between the subspace clusters can be explored at a glance. Several comparative experiments show the performance and the effectivity of DiSH.

1 Introduction

The well-known curse of dimensionality usually limits the applicability of traditional clustering algorithms to high-dimensional feature spaces because different sets of features are relevant for different (subspace) clusters. To detect such lower-dimensional subspace clusters, the task of subspace clustering (or projected clustering) has been defined recently. Existing subspace clustering algorithms usually either allow overlapping clusters (points may be clustered differently in varying subspaces) or non-overlapping clusters, i.e. points are assigned uniquely to one cluster or noise. Algorithms that allow overlap usually produce a vast amount of clusters which is hard to interpret. Thus, we focus on algorithms that generate non-overlapping clusters. Those algorithms in general suffer from two common limitations. First, they usually have problems with subspace clusters of significantly different dimensionality. Second, they often fail to discover clusters of different shape and densities, or they assume that the tendencies of the subspace clusters are already detectable in the entire feature space.

A third limitation derives from the fact that subspace clusters may be hierarchically nested, e.g. a subspace cluster of low dimensionality is embedded within several larger subspace clusters of higher dimensionality. None of the existing algorithms is

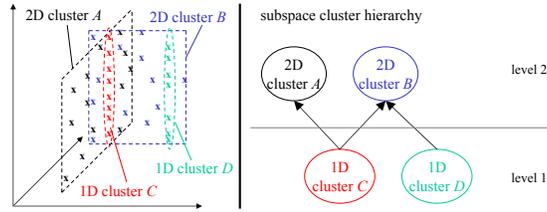


Fig. 1. Hierarchies of subspace clusters with multiple inheritance.

able to detect such important hierarchical relationships among the subspace clusters. An example of such a hierarchy is depicted in Figure 1 (left). Two one-dimensional (1D) cluster (C and D) are embedded within one two-dimensional (2D) cluster (B). In addition, cluster C is embedded within both 2D clusters A and B . Detecting such relationships of subspace clusters is obviously a hierarchical problem. The resulting hierarchy is different from the result of a conventional hierarchical clustering algorithm (e.g. a dendrogram). In a dendrogram, each object is placed in a singleton cluster at the leaf level, whereas the root node represents the cluster consisting of the entire database. Any inner node n represents the cluster consisting of the points located in the subtree of n . Dendrograms are limited to single inclusion, i.e. a lower dimensional cluster can only be the child cluster of one higher dimensional cluster. However, hierarchies of subspace clusters may exhibit multiple inclusions, e.g. cluster C in Figure 1 is a child of cluster A and B . The concept of multiple inclusions is similar to that of “multiple inheritance” in software engineering. To visualize such more complex relationships among subspace clusters, we need graph representations rather than tree representations. Such a graph representation which we will call *subspace clustering graph* (cf. Figure 1(right)) consists of nodes at different levels. These levels represent the dimensionality of the subspace in which the cluster is found (e.g. the level of cluster A in the graph of Figure 1 is 2). Each object p is assigned to a unique node in that hierarchy representing the lowest dimensional subspace cluster in which p is placed. In addition, an edge between a k -dimensional cluster C and an l -dimensional cluster B , where $l > k$, (e.g. cf. Figure 1) indicates that all points of cluster C are also members of cluster B .

In this paper, we propose the algorithm DiSH (Detecting Subspace cluster Hierarchies) that improves in the following aspects over the state-of-the-art subspace clustering approaches: First, DiSH uncovers complex hierarchies of nested subspace clusters including multiple inclusions. Second, DiSH can detect clusters in subspaces of significantly different dimensionality. Third, DiSH is able to detect clusters of different size, shape, and density. Furthermore, we propose the subspace clustering graph to visualize the resulting complex hierarchies by means of an appropriate visualization model. Using this visualization method the relationships between the subspace clusters can be explored at a glance.

The rest of the paper is organized as follows. We discuss related work in Section 2. Section 3 describes our new algorithm DiSH. The concepts of the clustering graph visualization are outlined in Section 4. An experimental evaluation is presented in Section 5. Section 6 concludes the paper.

2 Related Work

Many subspace clustering algorithms, e.g. [1–4], aim at finding all clusters in all subspaces of the feature space producing overlapping clusters, i.e. one point may belong to different clusters in different subspaces. In general, these methods also produce some sort of subspace hierarchy. However, those hierarchies are different from the hierarchy addressed in this paper because points are allowed to be placed in clusters such that there are no relationships between the subspaces of these clusters. Thus, the resulting “hierarchy” is much more complex and usually hard to interpret.

Other subspace clustering algorithms, e.g. [5–7], focus on finding non-overlapping subspace clusters. These methods assign each point to a unique subspace cluster or noise. Usually, those methods do not produce any information on the hierarchical relationships among the detected subspaces. The only approach to find some special cases of subspace cluster hierarchies introduced so far is HiSC [8]. However, HiSC is limited by the following severe drawbacks. First, HiSC usually assumes that if a point p belongs to a projected cluster C , then C must be visible in the local neighborhood of p in the *entire* feature space. Obviously, this is a quite unrealistic assumption. If p belongs to a projected cluster and the local neighborhood of p in the entire feature space does not exhibit this projection, HiSC will not assign p to its correct cluster. Second, the hierarchy detected by HiSC is limited to single inclusion which can be visualized by a tree (such as a dendrogram). As discussed above, hierarchies of subspace clusters may also exhibit multiple inclusions. To visualize such more complex relationships among subspace clusters, we need graph representations rather than tree representations. Third, HiSC uses a Single-Linkage approach for clustering and, thus, is limited to clusters of particular shapes. DiSH applies a density-based approach similar to OPTICS [9] to the subspace clustering problem that avoids Single-Link effects and is able to find clusters of different size, shape, and densities.

We do not focus on finding clusters of correlated objects that appear as arbitrarily oriented hyperplanes rather than axis-parallel projections (cf. e.g. [10–13]) because obviously, these approaches are orthogonal to the subspace clustering problem and usually demand more cost-intensive solutions.

3 Hierarchical Subspace Clustering

Let $\mathcal{D} \subseteq \mathbb{R}^d$ be a data set of n feature vectors and \mathcal{A} be the set of attributes of \mathcal{D} . For any subspace $S \subseteq \mathcal{A}$, $\pi_S(o)$ denotes the projection of $o \in \mathcal{D}$ into S . Furthermore, we assume that DIST is a distance function applicable to any $S \subseteq \mathcal{A}$, denoted by DIST^S , e.g. when using the Euclidean distance, $\text{DIST}^S(p, q) = \sqrt{\sum_{a_i \in S} (\pi_{\{a_i\}}(p) - \pi_{\{a_i\}}(q))^2}$.

Our key idea is to define the so-called *subspace distance* that assigns small values if two points are in a common low-dimensional subspace cluster and high values if two points are in a common high-dimensional subspace cluster or are not in a subspace cluster at all. Subspace clusters with small subspace distances are embedded within clusters with higher subspace distances.

For each point $o \in \mathcal{D}$ we first compute the subspace dimensionality representing the dimensionality of that subspace cluster in which o fits best. Thereby, we assume that

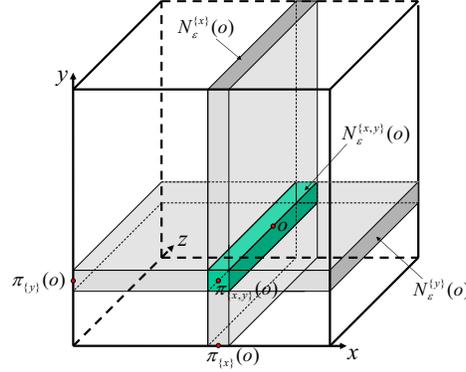


Fig. 2. Subspace selection for a point o (see text for details).

the “best” projection for clustering o is the subspace with the highest dimensionality (providing the most information), or in case of tie-situations, which provides the larger subspace cluster (containing more points in the neighborhood of o w.r.t. the subspace). The subspace dimensionality of a point o is determined by searching for dimensions of low variance (high density) in the neighborhood of o . An attribute-wise ε -range query ($\mathcal{N}_\varepsilon^{\{a_i\}}(o) = \{x \mid \text{DIST}^{\{a_i\}}(o, x) \leq \varepsilon\}$ for each $a_i \in \mathcal{A}$) yields a simple way to assign a predicate to an attribute for a certain object o . If only few points are found within the ε -neighborhood in attribute a_i the variance around o in attribute a_i will be relatively high. For this attribute we will assign 0 as predicate for the query point o , indicating that this attribute does not participate in a subspace that is relevant to any cluster to which o could possibly belong. Otherwise, if $\mathcal{N}_\varepsilon^{\{a_i\}}(o)$ contains at least μ objects, the attribute a_i will be a candidate for a subspace containing a cluster including object o .

From the variance analysis the candidate attributes that might span the best subspace S_o for object o are determined. These attributes need to be combined in a suitable way. This combination problem is equivalent to frequent itemset mining due to the monotonicity $S \subseteq T \Rightarrow |\mathcal{N}_\varepsilon^T(o)| \leq |\mathcal{N}_\varepsilon^S(o)|$. Thus, we can use any frequent itemset mining algorithm (e.g. the Apriori-algorithm [14]) in order to determine the best subspace of an object o .

Definition 1 (subspace preference vector/dimensionality of a point). Let S_o be the best subspace determined for object $o \in \mathcal{D}$. The subspace preference vector $w(o) = (w_1, \dots, w_d)^T$ of o is defined by

$$w_i(o) = \begin{cases} 1 & \text{if } a_i \in S_o \\ 0 & \text{if } a_i \notin S_o \end{cases}$$

The subspace dimensionality $\lambda(o)$ of $o \in \mathcal{D}$ is the number of zero-values in the subspace preference vector $w(o)$.

In the example in Figure 2 the ε -neighborhoods of the 3D point p in attributes x and y are shown by gray-shaded areas. If we assume that both of these areas contain at

least μ points whereas the ε -neighborhood of o along z (not shown) contains less than μ points, o may participate in a subspace cluster that is projected into the subspace $\{x, y\}$. If $|\mathcal{N}_\varepsilon^{\{x,y\}}(o)| \geq \mu$, then $w(o) = (1, 1, 0)^T$ and $\lambda(o) = 1$. Otherwise, none of the 1D subspace clusters containing o can be merged to form a higher dimensional subspace cluster, i.e. we assign o to the subspace containing more points.

Obviously, using any frequent itemset mining algorithm is rather inefficient for high-dimensional data sets, especially when the dimensionality of the subspace clusters are also high-dimensional. Thus, we further propose a heuristics for determining the best subspace S_o for an object o which scales linearly in the number of dimensions. We simply use a best-first search:

1. Determine the candidate attributes of o : $C(o) = \{a_i \mid a_i \in \mathcal{A} \wedge |\mathcal{N}_\varepsilon^{a_i}(o)| \geq \mu\}$.
2. Add $a_i = \arg \max_{a \in C(o)} \{|\mathcal{N}_\varepsilon^a(o)|\}$ to S_o and delete a_i from $C(o)$.
3. Set current intersection $I := \mathcal{N}_\varepsilon^{a_i}(o)$.
4. Determine attribute $a_i = \arg \max_{a \in C(o)} \{|I \cap \mathcal{N}_\varepsilon^a(o)|\}$.
 - (a) If $|I \cap \mathcal{N}_\varepsilon^{a_i}(o)| \geq \mu$ then:
Add a_i to S_o , delete a_i from $C(o)$, and set $I := I \cap \mathcal{N}_\varepsilon^{a_i}(o)$.
 - (b) Else: stop.
5. If $C \neq \emptyset$ continue with Step 4.

Using these heuristics to compute S_o for $o \in \mathcal{D}$, we can determine $w(o)$ as in Definition 1. Overall, we assign a d -dimensional preference vector to each point. The attributes having predicate “1” span the subspace where to find a cluster containing the point, whereas the remaining attributes are irrelevant.

We define a similarity measure between points which assigns a distance of 1, if these two points share a common 1D subspace cluster. If they share a common 2D subspace cluster, they have a distance of 2, etc. This similarity measure is integrated into the algorithm OPTICS [9]. Sharing a common k -dimensional subspace cluster may mean different things: Both points may be associated to the same k -dimensional subspace cluster, or both points may be associated to different $(k-1)$ -dimensional subspace clusters that intersect or are parallel (but not skew). Intuitively, the distance measure between two points corresponds to the dimensionality of the data space which is spanned by the “combined” subspace preference vector of the two points. We first give a definition of the *subspace dimensionality of a pair of points* $\lambda(p, q)$ which follows the intuition of the spanned subspace and then define our subspace distance measure.

Definition 2 (subspace dimensionality of a point pair). *The subspace preference vector $w(p, q)$ of a pair of points $p, q \in \mathcal{D}$ representing the combined subspace of p and q is computed by an attribute-wise logical AND-conjunction of $w(p)$ and $w(q)$, i.e. $w_i(p, q) = w_i(p) \wedge w_i(q)$ ($1 \leq i \leq d$). The subspace dimensionality between two points $p, q \in \mathcal{D}$, denoted by $\lambda(p, q)$, is the number of zero-values in $w(p, q)$.*

We cannot directly use the subspace dimensionality $\lambda(p, q)$ as the subspace distance because points from parallel subspace clusters will have the same subspace preference vector. Thus, we check whether the preference vectors of two points p and q are equal or one preference vector is “included” in the other one. This can be done by computing

the subspace preference vector $w(p, q)$ and checking whether $w(p, q)$ is equal to $w(p)$ or $w(q)$. If so, we determine the distance between the points in the subspace spanned by $w(p, q)$. If this distance exceeds $2 \cdot \varepsilon$, the points belong to different, parallel clusters. The threshold ε , playing already a key role in the definition of the subspace dimensionality (cf. Definition 1), controls the degree of jitter of the subspace clusters.

Since $\lambda(p, q) \in \mathbb{N}$, we usually have many tie situations when merging points/clusters during hierarchical clustering. These tie situations can be solved by considering the distance within a subspace cluster as a second criterion. Inside a subspace cluster the points are then clustered in the corresponding subspace using the traditional OPTICS algorithm and, thus, the subspace clusters can exhibit arbitrary sizes, shapes, and densities.

Definition 3 (subspace distance). *Let w be an arbitrary preference vector. Then $S(w)$ is the subspace defined by w and \bar{w} denotes the inverse of w . The subspace distance SDIST between p and q is a pair $\text{SDIST}(p, q) = (d_1, d_2)$, where $d_1 = \lambda(p, q) + \Delta(p, q)$ and $d_2 = \text{DIST}^{S(\bar{w}(p, q))}(p, q)$, and $\Delta(p, q)$ is defined as*

$$\Delta(p, q) = \begin{cases} 1 & \text{if } (w(p, q) = w(p) \vee w(p, q) = w(q)) \wedge \text{DIST}^{S(w(p, q))}(p, q) > 2\varepsilon \\ 0 & \text{else,} \end{cases}$$

We define $\text{SDIST}(p, q) \leq \text{SDIST}(r, s) \iff \text{SDIST}(p, q).d_1 < \text{SDIST}(r, s).d_1$ or $(\text{SDIST}(p, q).d_1 = \text{SDIST}(r, s).d_1 \text{ and } \text{SDIST}(p, q).d_2 \leq \text{SDIST}(r, s).d_2)$.

As suggested in [9], we introduce a smoothing factor μ to avoid the Single-Link effect and to achieve robustness against noise points. The parameter μ represents the minimum number of points in a cluster and is equivalent to the parameter μ used to determine the best subspace for a point. Thus, instead of using the subspace distance $\text{SDIST}(p, q)$ to measure the similarity of two points p and q , we use the *subspace reachability* $\text{REACHDIST}_\mu(p, q) = \max(\text{SDIST}(p, r), \text{SDIST}(p, q))$, where r is the μ -nearest neighbor (w.r.t. subspace distance) of p . DiSH uses this subspace reachability and computes a “walk” through the data set, assigning to each point o its smallest subspace reachability with respect to a point visited before o in the walk. The resulting order of the points is called *cluster order*. In a so-called reachability diagram for each point (sorted according to the cluster order along the x -axis) the reachability value is plotted along the y -axis. The valleys in this diagram represent the clusters. The pseudo-code of the DiSH algorithm can be seen in Figure 3.

4 Visualizing Subspace Cluster Hierarchies

The reachability plot is equivalent to tree-like representations and, thus, is not capable of visualizing hierarchies with multiple inclusions. This is illustrated in Figures 4(a) and 4(d): When exploring the reachability plots of the two different data sets A and B, one can see that they look almost the same (cf. Figures 4(b) and 4(e)). Thus, taking only the reachability plots into account, it is impossible to detect the obviously different kind of hierarchy of the second data set. This phenomenon is due to the fact that in data set B we face a subspace cluster hierarchy with multiple inclusion (the 1D cluster is embedded within both 2D clusters).

```

algorithm DiSH( $\mathcal{D}, \mu, \varepsilon$ )
   $co \leftarrow$  cluster order; // initially empty
   $pq \leftarrow$  empty priority queue ordered by REACHDIST $_{\mu}$ ;
  foreach  $p \in \mathcal{D}$  do
    compute  $w(p)$ ;
     $p$ .REACHDIST $_{\mu} \leftarrow \infty$ ;
    insert  $p$  into  $pq$ ;
  while ( $pq \neq \emptyset$ ) do
     $o \leftarrow pq$ .next();
     $r \leftarrow \mu$ -nearest neighbor of  $o$  w.r.t. SDIST;
    foreach  $p \in pq$  do
       $new\_sr \leftarrow \max(\text{SDIST}(o, r), \text{SDIST}(o, p))$ ;
       $pq$ .update( $p$ ,  $new\_sr$ );
    append  $o$  to  $co$ ;
  return  $co$ ;

```

Fig. 3. The DiSH algorithm.

This limitation of the reachability plot leads to our contribution of representing the relationships between cluster hierarchies as a so-called subspace clustering graph such that the relationships between the subspace clusters can be explored at a glance. The subspace clustering graph displays a kind of hierarchy which should not be confused with a conventional (tree-like) cluster hierarchy usually represented by dendrograms. The subspace clustering graph consists of nodes at several levels, where each level represents a subspace dimension. The top level represents the highest subspace dimension, which has the dimensionality of the data space. It consists of only one root node representing all points that do not share a common subspace with any other point, i.e. the noise points. Let us note that this is different to dendrograms where the root node represents the cluster of all objects. The nodes in the remaining levels represent clusters in the subspaces with the corresponding dimensionalities. They are labeled with the preference vector of the cluster they represent. For emphasizing the relationships between the clusters, every cluster is connected with its parents and its children. In contrast to tree representations, like e.g. dendrograms, a graph representation allows multiple parents for a cluster. This is necessary, since hierarchical subspace clusters can belong to more than one parent cluster. Consider e.g. data set B, where the objects of the intersection line are embedded in the horizontal plane as well as in the vertical plane, i.e. the cluster forming the intersection line belongs to two parents in the hierarchy. The subspace clustering graphs of the two data sets A and B are depicted in Figures 4(c) and 4(f). The line of data set A is represented by the cluster with the preference vector $[1,0,1]$. This cluster is a child of cluster $[1,0,0]$ representing the plane in data set A (cf. Figure 4(c)). The more complex hierarchy of data set B is represented in Figure 4(f), where the cluster $[1,0,1]$ belongs to two parent clusters, the cluster of the horizontal plane $[0,0,1]$ and the cluster of the vertical plane $[1,0,0]$.

In contrast to dendrograms, objects are not placed in singleton clusters at the leaf level, but are assigned to the lowest-dimensional subspace cluster they fit in within

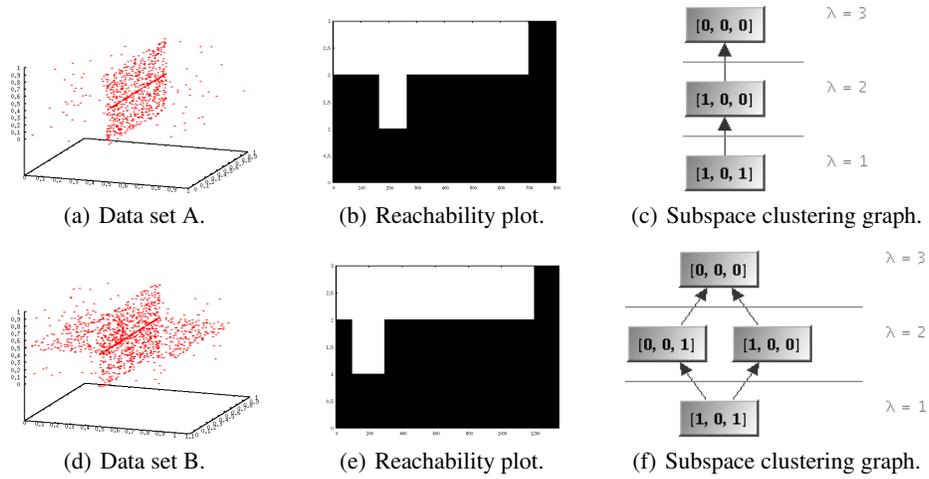


Fig. 4. Different hierarchies in 3-dimensional data.

```

method extractCluster(ClusterOrder co)
  cl ← empty list; // cluster list
  foreach o ∈ co do
    p ← o.predecessor;
    if (∄ c ∈ cl with  $w(c) = w(o, p) \wedge \text{dist}_{w(o, p)}(o, c.\text{center}) \leq 2 \cdot \varepsilon$ ) then
      create a new c;
      add c to cl;
      add o to c;
  return cl;

```

Fig. 5. The method to extract the clusters from the cluster order.

the graph. Similar to dendrograms, an inner node n of the subspace clustering graph represents the cluster of all points that are assigned to n and of all points assigned to its child nodes.

To build the subspace clustering graph, we extract in a first step all clusters from the cluster order. For each object o in the cluster order the appropriate cluster c has to be found, where the preference vector $w(c)$ of cluster c is equal to the preference vector $w(o, p)$ between o and its predecessor p . Additionally, since parallel clusters share the same preference vector, the weighted distance between the centroid of the cluster c and object o with $w(o, p)$ as weighting vector has to be less or equal to 2ε . The complete method to extract the clusters from the cluster order can be seen in Figure 5.

After the clusters have been derived from the cluster order, the second step builds the subspace cluster hierarchy. For each cluster we have to check, if it is part of one or more (parallel) higher-dimensional clusters, whereas each cluster is at least the child of the noise cluster. The method to build the subspace hierarchy from the clusters is depicted in Figure 6.

```

method buildHierarchy(c1)
  d ← dimensionality of objects in D;
  foreach ci ∈ c1 do
    foreach cj ∈ c1 do
      if ( $\lambda_{c_j} > \lambda_{c_i}$ ) then
        d ←  $\text{dist}_{w(c_i, c_j)}(c_i.\text{center}, c_j.\text{center})$ ;
        if ( $\lambda_{c_j} = d \vee (d \leq 2 \cdot \varepsilon \wedge \nexists c \in c1 : c \in c_i.\text{parents} \wedge \lambda_c < \lambda_{c_j})$ ) then
          add ci as child to cj;

```

Fig. 6. The method to build the hierarchy of subspace clusters.

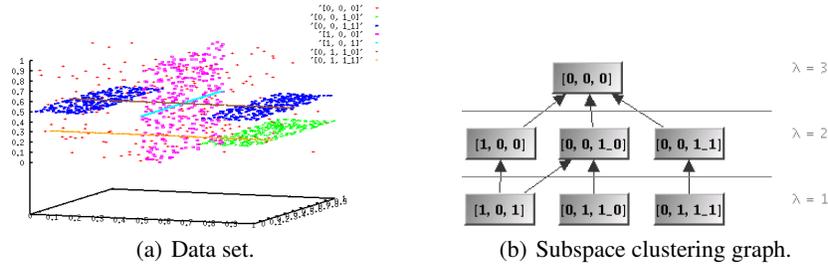


Fig. 7. Results on synthetic dataset DS1.

Table 1. Runtime, precision and recall w.r.t. the strategy for preference vector computation.

	APRIORI			BEST-FIRST		
	DS1	DS2	DS3	DS1	DS2	DS3
runtime [sec]	147	32	531	76	14	93
precision [%]	99.7	99.5	99.7	99.7	99.5	99.5
recall [%]	99.8	99.6	99.8	99.8	99.6	99.5

5 Experimental Evaluation

We first evaluated DiSH on several synthetic data sets. Exemplary, we show the results on three data sets named “DS1”, “DS2”, and “DS3”.

We evaluated the precision, recall and the runtime of our DiSH algorithm w.r.t. the strategies used for determination of the preference vectors. The strategy using the Apriori-algorithm [14] is denoted with “APRIORI”, the heuristics using the best-first search is denoted with “BEST-FIRST”. The results of the runs with both strategies on the three data sets are summarized in Table 1. Since the heuristics using best-first search outperforms the strategy using the Apriori-algorithm in terms of runtime and has almost equal precision and recall values, we used in all further experiments the heuristics to compute the preference vectors rather than the Apriori-based approach.

Data set “DS1” (cf. Figure 7(a)) contains 3D points grouped in a complex hierarchy of 1D and 2D subspace clusters with several multiple inclusions and additional noise

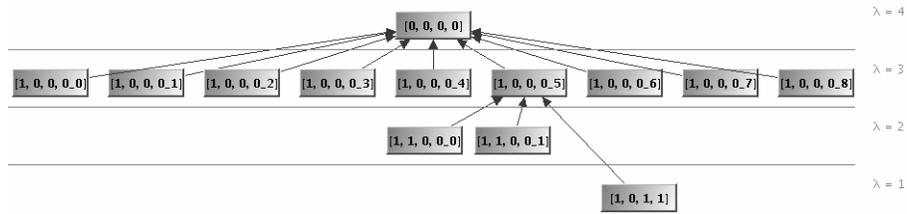


Fig. 8. Subspace clustering graph of the Forest data.

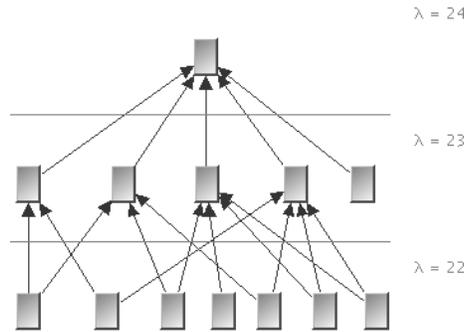


Fig. 9. Subspace clustering graph of the Gene data.

points. The results of DiSH applied to DS1 are depicted in Figure 7(b). As it can be seen, the complete hierarchical clustering structure can be obtained from the resulting subspace clustering graph. In particular, the complex nested clustering structure can be seen at a glance. Data set “DS2” is a 5D data set containing ten clusters of different dimensionality and noise: one cluster is embedded in a 4D subspace, four clusters are 3D, three clusters are 2D and two clusters are 1D subspace clusters. The resulting subspace clustering graph (not shown due to space limitations) produced by DiSH exhibits all ten subspace clusters of considerably different dimensionality correctly. Similar observations can be made when evaluating the subspace clustering graph obtained by DiSH on data set “DS3” (not shown due to space limitations). The 16D data set DS3 contains noise points, one 13 dimensional, one 11 dimensional, one 9 dimensional, one 7 dimensional cluster, and two 6 dimensional clusters. Again, DiSH found all six subspace clusters correctly.

We also applied HiSC, PreDeCon and PROCLUS on DS1 for comparison. Neither PreDeCon nor PROCLUS are able to detect the hierarchies in DS1 and the subspace clusters of significantly different dimensionality. HiSC performed better in detecting simple hierarchies of single inclusion but fails to detect multiple inclusions.

In addition, we evaluate DiSH using several real-world data sets. Applied to the Wisconsin Breast Cancer Database (original) from the UCI ML Archive¹ ($d = 9, n = 569$, objects labeled as “malignant” or “benign”) DiSH finds a hierarchy containing

¹ <http://www.ics.uci.edu/~mllearn/MLSummary.html>

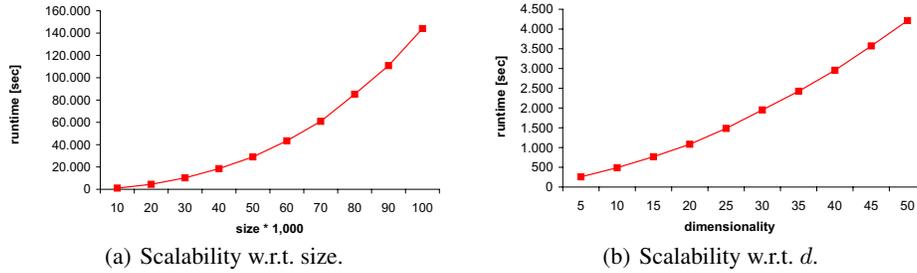


Fig. 10. Scalability results.

several low dimensional clusters and one 7D cluster ($\epsilon = 0.01, \mu = 15$). An additional 9D cluster contains the noise points. It is worth mentioning that the reported clusters are pure. In particular, the seven low dimensional clusters only contain objects labeled as “benign”, whereas the 7D cluster only contains objects marked as “malignant”.

We applied DiSH on the Wages data set² ($d = 4, n = 534$). Since most of the original attributes are not numeric, we used only 4 dimensions (YE=years of education, W=wage, A=age, and YW=years of work experience) for clustering. The resulting subspace cluster hierarchy (using $\epsilon = 0.001, \mu = 9$) is visualized in Figure 8. The nine parallel clusters having a subspace dimensionality of $\lambda = 3$ consist of data of people having equal years of education, e.g in cluster $[1, 0, 0, 0_0]$ YE=17 and in cluster $[1, 0, 0, 0_5]$ YE=12. The two clusters labeled with $[1, 1, 0, 0_0]$ and $[1, 1, 0, 0_1]$ in the 2D subspace are children of cluster $[1, 0, 0, 0_5]$ and have (in addition to equal years of education, YE=12) equal wages values (W=7.5 and W=5, respectively). The 1-dimensional cluster $[1, 0, 1, 1]$ is a child of $[1, 1, 0, 0_0]$ and has the following properties: YE=12, A=26, and YW=8.

Last but not least, we applied DiSH to the yeast gene expression data set of [15] ($d = 24, n \approx 4,000$). The result of DiSH (using $\epsilon = 0.01, \mu = 100$) on the gene expression data is shown in Figure 9. Again, DiSH found several subspace clusters of different subspace dimensionalities with multiple inclusions.

The scalability of DiSH w.r.t. the data set size is depicted in Figure 10(a). The experiment was run on a set of 5D synthetic data sets with increasing number of objects ranging from 10,000 to 100,000. The objects are distributed over equally sized subspace clusters of subspace dimensionality $\lambda = 1, \dots, 4$ and noise. As parameters for DiSH we used $\epsilon = 0.001$ and $\mu = 20$. As it can be seen, DiSH scales slightly superlinear w.r.t. the number of tuples. A similar observation can be made when evaluating the scalability of DiSH w.r.t. the dimensionality of the data set (cf. Figure 10(b)). The experiments were obtained using data sets with 5,000 data points and varying dimensionality of $d = 5, 10, 15, \dots, 50$. For each data set the objects were distributed over $d - 1$ subspace clusters of subspace dimensionality $\lambda = 1, \dots, d - 1$ and noise. Again, the result shows a slightly superlinear increase of runtime when increasing the dimensionality of the data set. The parameters for DiSH were the same as in the evaluation of the scalability of DiSH w.r.t. the data set size ($\epsilon = 0.001$ and $\mu = 20$).

² http://lib.stat.cmu.edu/datasets/CPS_85_Wages

6 Conclusions

In this paper, we presented DiSH, the first subspace clustering algorithm for detecting complex hierarchies of subspace clusters. DiSH is superior to the state-of-the-art subspace clustering algorithms in several aspects: First, it can detect clusters in subspaces of significantly different dimensionality. Second, it is able to determine hierarchies of nested subspace clusters containing single and multiple inclusions. Third, it is able to detect clusters of different size, shape, and density. Fourth, it does not assume that the subspace preference of a point p is exhibited in the local neighborhood of p in the entire data space. We have shown by performing several comparative experiments using synthetic and real data sets that DiSH has a superior performance and effectivity compared to existing methods.

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. SIGMOD. (1998)
2. Cheng, C.H., Fu, A.W.C., Zhang, Y.: Entropy-based subspace clustering for mining numerical data. In: Proc. KDD. (1999) 84–93
3. Kailing, K., Kriegel, H.P., Kröger, P.: Density-connected subspace clustering for high-dimensional data. In: Proc. SDM. (2004)
4. Kriegel, H.P., Kröger, P., Renz, M., Wurst, S.: A generic framework for efficient subspace clustering of high-dimensional data. In: Proc. ICDM. (2005)
5. Aggarwal, C.C., Procopiuc, C.M., Wolf, J.L., Yu, P.S., Park, J.S.: Fast algorithms for projected clustering. In: Proc. SIGMOD. (1999)
6. Procopiuc, C.M., Jones, M., Agarwal, P.K., Murali, T.M.: A Monte Carlo algorithm for fast projective clustering. In: Proc. SIGMOD. (2002)
7. Böhm, C., Kailing, K., Kriegel, H.P., Kröger, P.: Density connected clustering with local subspace preferences. In: Proc. ICDM. (2004)
8. Achtert, E., Böhm, C., Kriegel, H.P., Kröger, P., Müller-Gorman, I., Zimek, A.: Finding hierarchies of subspace clusters. In: Proc. PKDD. (2006) To appear.
9. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proc. SIGMOD. (1999)
10. Yang, J., Wang, W., Wang, H., Yu, P.S.: Delta-Clusters: Capturing subspace correlation in a large data set. In: Proc. ICDE. (2002)
11. Wang, H., Wang, W., Yang, J., Yu, P.S.: Clustering by pattern similarity in large data sets. In: Proc. SIGMOD. (2002)
12. Böhm, C., Kailing, K., Kröger, P., Zimek, A.: Computing clusters of correlation connected objects. In: Proc. SIGMOD. (2004)
13. Aggarwal, C.C., Yu, P.S.: Finding generalized projected clusters in high dimensional space. In: Proc. SIGMOD. (2000)
14. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. SIGMOD. (1994)
15. Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., Futcher, B.: "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast *Saccharomyces Cerevisiae* by Microarray Hybridization." *Molecular Biology of the Cell* **9** (1998) 3273–3297