In Proc. 8th SIAM Int. Conf. on Data Mining (SDM'08), Atlanta, GA, 2008.
- Best Paper Honorable Mention Award Winner -

Robust Clustering in Arbitrarily Oriented Subspaces

Elke Achtert, Christian Böhm, Jörn David, Peer Kröger, Arthur Zimek

Department Institute for Informatics
Ludwig-Maximilians Universität München
http://www.dbs.ifi.lmu.de
{achtert,boehm,david,kroegerp,zimek}@dbs.ifi.lmu.de

Abstract

In this paper, we propose an efficient and effective method to find arbitrarily oriented subspace clusters by mapping the data space to a parameter space defining the set of possible arbitrarily oriented subspaces. The objective of a clustering algorithm based on this principle is to find those among all the possible subspaces, that accommodate many database objects. In contrast to existing approaches, our method can find subspace clusters of different dimensionality even if they are sparse or are intersected by other clusters within a noisy environment. A broad experimental evaluation demonstrates the robustness, efficiency and effectivity of our method.

1 Introduction

Subspace clustering is a data mining task which has attracted considerable attention during the last years. There are two main reasons for this popularity. Firstly, conventional (full space) clustering algorithms often fail to find useful clusters when applied to data sets of higher dimensionality, because typically many of the attributes are noisy, some attributes may exhibit correlations among another, and only few of the attributes really contribute to the cluster structure. Secondly, the knowledge gained from a subspace clustering algorithm is much richer than that of a conventional clustering algorithm. It can be used for interpretation, data compression, similarity search, etc. as we will discuss in the next paragraph.

We can distinguish between subspace clustering algorithms for axis-parallel subspaces [5, 12, 3, 15, 6] and those for subspaces which are arbitrarily oriented (called *oriented clustering*, *generalized subspace clustering*, or *correlation clustering*, e.g., [4, 7, 17]). In both cases, the data objects which are grouped into a common subspace cluster, are very dense (i.e., the variance is small) when projected onto the hyperplane which is perpendicular to the subspace of the cluster (called the *perpendicular space plane*). The objects may form a completely arbitrary shape with a high variance when projected onto the hyperplane of the subspace in which the cluster resides (called the *cluster subspace plane*). This means, that the objects of the subspace cluster are all *close* to the cluster subspace plane. The knowledge, that all data objects of a cluster are close to the cluster subspace plane is valuable for many applications: If the plane is axis-parallel, this means that the values of some of the attributes are, more or less, constant for all cluster members. The whole group is characterized by this constant attribute value, an information which can definitely be important for the interpretation of the cluster. This property may also be used to perform a dedicated dimensionality reduction for the objects of the cluster and may be useful for data compression (because only the higher-variance attributes must be stored at high precision individually for each cluster member) and similarity search (because only the high-variance attributes need to be individually considered for the search and an index needs only be constructed for the high-variance attributes).

If the cluster subspace plane is arbitrarily oriented, the knowledge is even more valuable. In this case, we know that the attributes which define the cluster subspace plane, have a complex dependency among each other. This dependency defines a rule, which again characterizes the cluster and which is potentially useful for cluster interpretation. Similarly to the case of axis-parallel clusters, this dependency rule may also be used for dimensionality reduction, data compression, similarity search, and indexing. Consider, for example, Figure 1 which contains two general subspace clusters in a very noisy environment. For each of the subspace clusters, we know that the x and y coordinates are approximately linearly dependent from each other ($y \approx m_i \cdot x + t_i$), and, therefore, only one of them needs to be stored at full precision, indexed, etc. Furthermore, the knowledge of the degree of dependency, as well

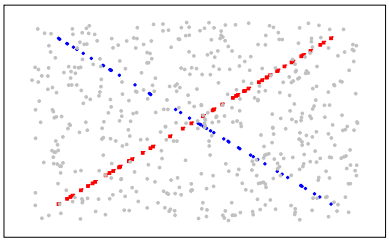


Figure 1: Data set with two non-dense general subspace clusters in a noisy environment

as the slope and intercept may be important for the interpretation of the cluster in the context of the application.

One well-known effect of the "curse of dimensionality" is the correlation among attributes in high-dimensional data. While full-dimensional clustering approaches are easily misled by these correlations, generalized subspace clustering approaches, hence also called *correlation clustering*, make use of this effect to identify clusters in subspaces of arbitrary dimensionality. However, finding axis-parallel or generally oriented subspace clusters is not a trivial task. The number of possible axis parallel subspaces is exponential in the number of dimensions, and the number of general subspaces is even infinite. Therefore, a complete enumeration of all possible subspaces to be checked for clusters is not feasible. Consequently, all previous solutions rely on specific assumptions and heuristics, and try to find promising subspaces during the clustering process, for instance in an iterative optimization. We will see that this previous approach of *learning* suitable subspaces works well if (but only if) subspace clusters are locally well separated and no outlier objects (belonging to no cluster) exist. In the presence of outliers in the local neighborhood of cluster points or cluster representatives in the entire feature space, most previous subspace clustering algorithms fail to detect subspace clusters, because the algorithms try to find suitable subspaces for each cluster from the local neighborhood of cluster points or cluster representatives in the entire feature space. Outliers in the neighborhoods, that do not belong to the corresponding cluster prevent the algorithms from finding suitable subspaces, and the absence of a precise subspace prevents the algorithm from effectively filtering out the outliers. In high dimensional spaces, where distances cannot be used to differentiate between near and far points, the concept of local neighborhoods is meaningless. Consequently, the neighborhoods of cluster points or cluster representatives will contain a large number of outliers that do not belong to the corresponding cluster. However, those problems arise even if the number of outliers is very small (e.g. 5-10 outliers in the complete data set). Thus, an environment of heavy noise such as that of Figure 1 is completely out of the scope of previous subspace clustering methods even in lower dimensional data spaces, as we will discuss more deeply in Section 2 for locally optimizing approaches such as ORCLUS [4] and for density-based approaches such as 4C [7].

To escape from this circular dependency of subspace finding and outlier filtering, we propose in this paper to reconsider the problem of finding generally oriented subspace clusters from a new perspective. The main idea is to transform every object into a new space, the space of *all possible subspaces* in which this object is contained. Since the number of all such subspaces is infinite, we do neither enumerate these subspaces nor represent each object by an infinite (or very high) number of transformed objects. Instead, we consider a *continuum* of many possible subspaces represented by a small number of parameters. This continuum is split up on demand during the clustering process to set limits to the allowed cluster subspace planes and finally identify subspace clusters. We will describe this method in detail in Section 3, and then experimentally evaluate our method in Section 4. Section 5 concludes our paper.

2 Related Work

Existing approaches for subspace clustering rely on certain heuristics that use specific assumptions to shrink down the search space and thus to reduce the runtime complexity. However, if these assumptions are not true for a given data set, the methods will either fail to detect any suitable patterns or exhibit an exponential runtime.

Many subspace clustering algorithms (e.g. [5, 12, 3, 15, 6]) assume that the subspace clusters are axis-parallel. Otherwise, they will not find any pattern. Pattern-based subspace clustering algorithms (e.g. [19, 18, 14, 13]) are limited to find only clusters that represent pairwise positive correlations in the data set. In contrast, arbitrarily oriented hyperplanes (subspace clusters) may also represent more complex or negative correlations.

In this paper we focus on the generalized problem of finding arbitrarily oriented subspace clusters. All existing algorithms for this problem assume that the cluster structure is significantly dense in the local neighborhood of the cluster centers or other points that participate in the cluster. In the context of high-dimensional data this "locality assumption" is rather optimistic. Theoretical considerations [10] show that concepts like "local neighborhood" are not meaningfull in high dimensional spaces because distances can no longer be used to dif-

ferentiate between points. This is a consequence of the well-known curse of dimensionality.

ORCLUS [4] is based on k-means and iteratively learns the similarity measure capturing the subspace containing a given cluster from the points assigned to the cluster in each iteration by applying PCA on these points. Since the algorithm starts with the Euclidean distance, the algorithm learns the subspaces from the local neighborhood of the initial cluster centers. However, if this local neighborhood contains some noise or the clustering structure is too sparse within this local neighborhood, the learning heuristic will be misled because PCA is rather sensitive to outliers. In those cases, ORCLUS will fail to detect meaningful patterns. These considerations accordingly apply to the method proposed in [8] which is a slight variant of ORCLUS designed for enhancing multi-dimensional indexing.

4C [7] integrates PCA into density-based clustering. It evaluates the Euclidean neighborhood of each point p to learn the subspace characteristics in which p can be clustered best. Similar to ORCLUS, 4C thus relies on the assumption that the clustering structure is dense in the entire feature space. Otherwise 4C will also fail to produce meaningful results. The same holds true to some variations of 4C like COPAC [2] and ERiC [1].

The method CURLER [17] merges the clusters computed by the EM algorithm using the so-called cosharing level. The resulting clusters need not to represent linear correlations. Rather, any dense pattern in the data space is found that may represent a more complex, not necessarily linear correlation. CURLER also relies on the assumption that the subspace clustering structure is dense in the entire feature space because both the generation as well as the merging of microclusters uses local neighborhood information.

3 Algorithm CASH

Obviously, the locality assumption that the clustering structure is dense in the entire feature space and that the Euclidean neighborhood of points in the cluster or of cluster centers does not contain noise is a very strict limitation for high dimensional real-world data sets. In [10] the authors show that in high dimensional spaces, the distance to the nearest neighbor and the distance to the farthest neighbor converge. As a consequence, distances can no longer be used to differentiate between points in high dimensional spaces and concepts like the neighborhood of points become meaningless. Usually, although many points share a common hyperplane, they are not close to each other in the original feature space. In those cases, existing approaches will fail to detect meaningful patterns because they cannot learn the correct subspaces of the clusters. In addition, as long as the correct subspaces of the clusters cannot be determined, obviously outliers and noise cannot be removed in a preprocessing step.

In this paper, we propose to use the ideas of the Hough transform [11, 9] to develop an original principle for characterizing the subspace containing a cluster. The Hough transform was originally designed to map the points from a 2-dimensional data space (also called picture space) of Euclidean coordinates (e.g. pixels of an image) into a parameter space. The parameter space represents all possible 1D lines in the original 2D data space. In principle, each point of the data space is mapped into an infinite number of points in the parameter space which is not materialized as an infinite set but instead as a trigonometric function in the parameter space. Each function in the parameter space represents all lines in the picture space crossing the corresponding point in data space. The intersection of two curves in the parameter space indicates a line through both the corresponding points in the picture space. The objective of a clustering algorithm is to find intersections of many curves in the parameter space representing lines through many database objects. The key feature of the Hough transform is that the distance of the points in the original data space is not considered any more. Objects can be identified as associated to a common line even if they are far apart in the original feature space. As a consequence, the Hough transform is a promising candidate for developing a principle for subspace analysis that does not require the locality assumption and, thus, enables a global subspace clustering approach.

In the following, we will first present a novel principle for subspace analysis inspired by the ideas of the Hough transform (cf. Section 3.1). This principle enables us to transform the task of subspace clustering (in data space) into a grid-based clustering problem (in parameter space). Unlike grid-based methods operating directly in the data space, our method does not suffer from grid resolution and grid positioning problems. In order to perform this transformation, we first need to define the boundaries of the grid (cf. Section 3.2). Then we will show how to identify dense grid cells that represent potential subspace clusters (cf. Section 3.3). Since the parameter space is d-dimensional for a d-dimensional data space, finding dense grid cells becomes rather costly for higher dimensional data sets. Thus, we will propose a more efficient search strategy for finding regions of interest in the parameter space (cf. Section 3.4). We will also summarize our subspace clustering algorithm CASH (Clustering in Arbitrary Subspaces based on the Hough transform) and discuss some of its properties (cf. Section 3.5).

3.1 Subspace Analysis: a Novel Principle

Our novel principle for subspace analysis is based on a generalized description of spherical coordinates. Generalized spherical coordinates combine $d-1$ independent angles $\alpha_1, \ldots, \alpha_{d-1}$ with the norm r of a d-dimensional vector $x = (x_1, \ldots, x_d)^T$ to completely describe the vector x w.r.t. the given orthonormal basis $e_1, \ldots, e_d$.

DEFINITION 1. (SPHERICAL COORDINATES) *Let e_i, $1 \leq i \leq d$, be an orthonormal basis in a d-dimensional feature space. Let $x = (x_1, \ldots, x_d)^T$ be a d-dimensional vector on the hypersphere of radius r with center at the origin. Let u_i be the unit vector in the direction of the projection of vector x onto the manifold spanned by $e_i, \ldots, e_d$. For the $d-1$ independent angles $\alpha_1, \ldots, \alpha_{d-1}$, let α_i, $1 \leq i \leq d-1$, be the angle between u_i and e_i. Then the* generalized spherical coordinates *of vector x are defined by:*

$$x_i = r \cdot \left(\prod_{j=1}^{i-1} \sin(\alpha_j) \right) \cdot \cos(\alpha_i), \quad \text{where } \alpha_d = 0.$$

For any point $p \in \mathcal{D} \subseteq \mathbb{R}^d$ there exists an infinite number of hyperplanes containing p. The spherical coordinates are utilized to define the normal vector of the Hessian normal form for any of those hyperplanes, i.e., each hyperplane is uniquely defined by a point p and $d-1$ angles $\alpha_1, \ldots, \alpha_{d-1}$, with $\alpha_i \in [0, \pi)$, defining the normal vector. Thus, any point p together with any tupel of angles $\alpha_1, \ldots, \alpha_{d-1}$, can be mapped by the following *parametrization function* to the distance of the corresponding hyperplane to the origin.

DEFINITION 2. (PARAMETRIZATION FUNCTION) *Let $p = (p_1, \ldots, p_d)^T \in \mathcal{D} \subseteq \mathbb{R}^d$ be a d-dimensional vector, and let $n = (n_1, \ldots, n_d)^T$ be a d-dimensional unit vector specified by $d-1$ angles $\alpha_1, \ldots, \alpha_{d-1}$ according to Definition 1. Then the* parametrization function $f_p : \mathbb{R}^{d-1} \to \mathbb{R}$ *of vector p denotes the distance of the hyperplane defined by the point p and the normal vector n to the origin:*

$$\begin{aligned} f_p(\alpha_1, \ldots, \alpha_{d-1}) &= \langle p, n \rangle \\ &= \sum_{i=1}^{d} p_i \cdot \left(\prod_{j=1}^{i-1} \sin(\alpha_j) \right) \cdot \cos(\alpha_i) \end{aligned}$$

Based on Definition 2, we can map any point $p \in \mathbb{R}^d$ to a function in a d-dimensional parameter space $\mathcal{P}$ representing all possible hyperplanes containing p. This parameter space is spanned by the $d-1$ angles $\alpha_1, \ldots, \alpha_{d-1}$ of the normal vectors defining the hyperplanes in Hessian normal form and their distances $\delta = f_p(\alpha_1, \ldots, \alpha_{d-1})$ to the origin.

By means of the parametrization function (Definition 2), we can also extend the properties of the original Hough transform as stated in [9] for the mapping of 2-dimensional points to d-dimensional data spaces and the corresponding parameter spaces:

PROPERTY 1. *A point $p \in \mathcal{D} \subseteq \mathbb{R}^d$ in data space is represented by a sinusoidal curve $f_p : \mathbb{R}^{d-1} \to \mathbb{R}$ in parameter space $\mathcal{P}$.*

Figure 2 illustrates a 3-dimensional example of this property. Three points p_1, p_2, and p_3 in data space are mapped onto the corresponding sinusoidal curves f_{p_1}, f_{p_2}, and f_{p_3}, respectively, in parameter space.

PROPERTY 2. *A point $(\alpha_1, \ldots, \alpha_{d-1}, \delta) \in \mathcal{P}$ in parameter space corresponds to a $(d-1)$-dimensional hyperplane in data space.*

In Figure 2, the point $(\alpha_1^s, \alpha_2^s, \delta^s)$ in parameter space represents the 2-dimensional plane s with

$$\delta^s = \cos(\alpha_1^s) \cdot x_1 + \sin(\alpha_1^s) \cdot \cos(\alpha_2^s) \cdot x_2 + \sin(\alpha_1^s) \cdot \sin(\alpha_2^s) \cdot x_3$$

in data space.

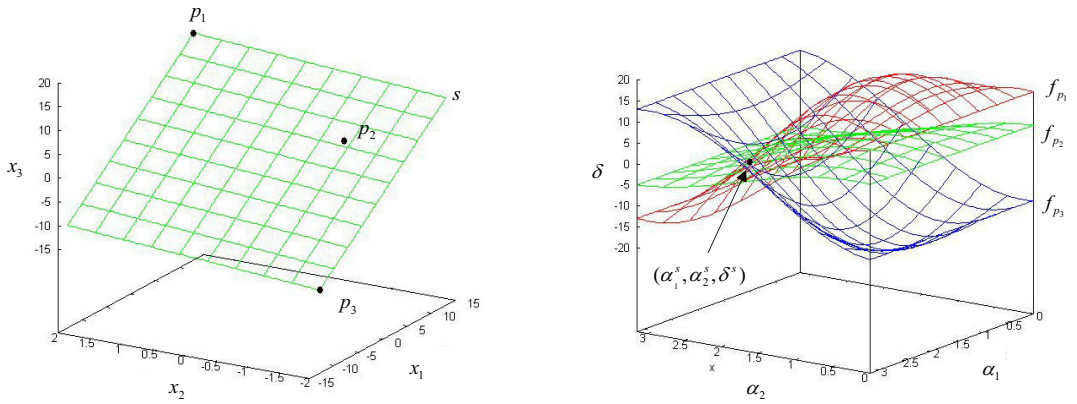
PROPERTY 3. *Points that are located on a $(d-1)$-dimensional hyperplane in data space correspond to sinusoidal curves through a common point in parameter space.*

The three points $p_1, p_2, p_3 \in \mathcal{D}$ (Figure 2) are located on the 2-dimensional plane s. Their corresponding sinusoidal curves $f_{p_1}, f_{p_2}, f_{p_3}$ intersect in the point $(\alpha_1^s, \alpha_2^s, \delta^s) \in \mathcal{P}$, where α_1^s, α_2^s and δ^s are the parameters of plane s as given above (cf. Property 2).

PROPERTY 4. *Points lying on the same sinusoidal curve in parameter space represent $(d-1)$-dimensional hyperplanes through the same point in data space.*

For example, in Figure 2, f_{p_1} in parameter space represents all 2-dimensional planes through p_1 in data space. Thus, any point on f_{p_1} in parameter space represents a given 2-dimensional plane in data space that passes through p_1.

Properties 1 – 4 induce that an intersection point in the parameter space indicates points in the data space that are located on a common $(d-1)$-dimensional hyperplane. In order to detect those linear hyperplanes in the data space, the task is to search for points in the parameter space where many sinusoidal curves intersect. Since computing all possibly interesting intersection points is computationally too expensive, we discretize the parameter space by some grid and search for grid cells with which many sinusoidal curves intersect. For that purpose, for each grid cell the



(a) Three points p_1, p_2, p_3 on a plane $s \subseteq \mathbb{R}^3$.

(b) Corresponding parametrization functions.

Figure 2: Transform of a 3-dimensional data space into a 3-dimensional parameter space.

number of intersecting sinusoidal curves is aggregated. Due to this discretization of the parameter space, exact intersections are no longer considered. Rather, a slight impreciseness is allowed modelling a certain degree of jitter given by the grid resolution. The higher the grid resolution is, the lower is the allowed degree of jitter, i.e. the more accurate the recognition of the line segments.

With the proposed concepts, we transform the original subspace clustering problem (in data space) into a grid-based clustering problem (in parameter space).

3.2 Specifying the Boundaries of the Grid

To define a discretization of the parameter space, the range of the axes must be known. The axes for the angle-parameters $\alpha_1, \ldots, \alpha_{d-1}$, are bounded by $[0, \pi)$. The δ-axis ranges from the minimum of all minima of all parametrization functions to the maximum of all their maxima within $[0, \pi)^{d-1}$. Each f_p is a sinusoid with a period of 2π. Thus, any f_p has exactly one global extremum in the interval $[0, \pi)^{d-1}$. If the extremum of f_p is a maximum, the minimal value for f_p in the given interval has to be determined and *vice versa*.

To find the global extremum of a parametrization function f_p in the interval $[0, \pi)^{d-1}$, those angles $\alpha_1, \ldots, \alpha_{d-1}$ need to be determined where all the first order derivatives of f_p are zero, and the Hessian matrix of f_p is either positive or negative definite. As noted above, f_p is guaranteed to have exactly one global extremum $f_p(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_{d-1})$ in $[0, \pi)^{d-1}$. The values for the angles $\tilde{\alpha}_n$ $(n = 1, \ldots, d-1)$ of the global extremum of f_p are given by (cf. Appendix A for details):

$$\tilde{\alpha}_n \quad = \quad \arctan\left(\frac{\sum_{j=n+1}^{d} p_j \cdot \left[\prod_{k=n+1}^{j-1} \sin(\tilde{\alpha}_k)\right] \cdot \cos(\tilde{\alpha}_j)}{p_n}\right)$$

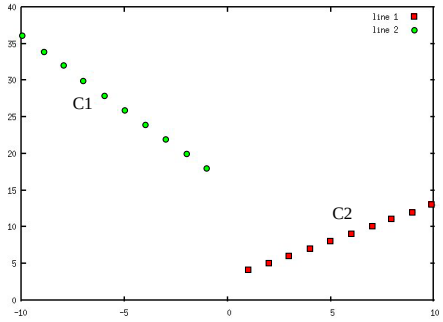
Given the global extremum of a parametrization function f_p in the interval $[0, \pi)^{d-1}$, we have to distinguish several cases to determine the opposite value, i.e., to determine the maximum of f_p if the global extremum of f_p is a minimum, or, to determine the minimum of f_p if the global extremum is a maximum. In the following, we describe how to determine the point $\alpha^{\min}$ where the parametrization function f_p has a minimum in interval $[0, \pi)^{d-1}$ given that the global extremum is a maximum. In the opposite case, the point $\alpha^{\max}$ where the parametrization function f_p has a maximum in interval $[0, \pi)^{d-1}$ given the global extremum is a minimum can be determined analogously. Please refer to Appendix B for a detailed formalization of this step.

We determine the point $\alpha^{\min} = (\alpha_1^{\min}, \ldots, \alpha_{d-1}^{\min})$ where the parametrization function f_p has a minimum in interval $[0, \pi)^{d-1}$ as follows: First, the angle $\overline{\alpha}_{d-1}$ on axis $(d-1)$ is determined where f_p has an extremum on this axis. Dependent on the type of the extremum in $\overline{\alpha}_{d-1}$ and the location of $\overline{\alpha}_{d-1}$ in the interval $[0, \pi)$, the minimum angle $\alpha_{d-1}^{\min}$ on axis $(d-1)$ in interval $[0, \pi)$ is determined. In the next step, axis $(d-2)$ will be considered: Now, the angle $\overline{\alpha}_{d-2}$ will be determined, where f_p has an extremum on this axis under the constraint of the known minimum on axis $d-1$, which is given by $\alpha_{d-1}^{\min}$. Analogously to the first step, dependent on the type of the extremum in $\overline{\alpha}_{d-2}$ and the location of $\overline{\alpha}_{d-2}$ in the interval $[\check{\alpha}_{d-2}, \hat{\alpha}_{d-2})$, the minimum angle $\alpha_{d-2}^{\min}$ is determined. In this way, all minimum angles are determined under the constraint of the known minima on the already processed axes.

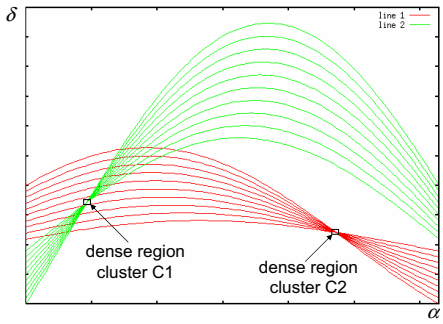
In summary, given for each parametrization function f_p its minimal and maximal value $\alpha_p^{\min}$ and $\alpha_p^{\max}$ in interval $[0, \pi)^{d-1}$, the δ-axis of the parameter space $\mathcal{P}$ is bounded by

$$[\delta_{min}, \delta_{max}] = [\min_{p \in \mathcal{D}}(f_p(\alpha_p^{\min})), \max_{p \in \mathcal{D}}(f_p(\alpha_p^{\max}))]$$

and $\mathcal{P} = [\delta_{min}, \delta_{max}] \times [0, \pi)^{d-1}$.



(a) Two lines in data space.



(b) Dense regions in parameter space.

Figure 3: Dense regions in parameter space capturing two lines in data space.

3.3 Identifying Dense Grid Cells

Given a discretized parameter space, now those grid cells (hypercuboids) have to be found that are intersected by parametrization functions of a minimum number m of functions. Hypercuboids containing at least m parametrization functions are called dense regions of the parameter space. Those dense regions represent arbitrarily oriented subspaces in the data space accommodating at least m points. This is illustrated in Figure 3. The two subspace clusters forming lines in the data space (cf. Figure 3(a)) are represented by two distinct dense regions in the parameter space (cf. Figure 3(b)).

To find those dense regions in the parameter space, for each grid cell or hypercuboid the number of parametrization functions which intersect this hypercuboid has to be counted. This can be done conveniently by determining those values $\alpha_p^{\min}$ and $\alpha_p^{\max}$ in a given interval $[\check{\alpha}, \hat{\alpha}) \subseteq [0, \pi)^{d-1}$ that minimize and maximize a parametrization function f_p. Then, all hypercuboids based on this interval and positioned between $f_p(\alpha_p^{\min})$ and $f_p(\alpha_p^{\max})$ are intersected by f_p. The values $\alpha_p^{\min}$ and $\alpha_p^{\max}$ in a given interval $[\check{\alpha}, \hat{\alpha}) \subseteq [0, \pi)^{d-1}$ that minimize and maximize f_p can be determined analogously to the algorithm specified in Section 3.2 where the given interval was assumed to be $[0, \pi)^{d-1}$.

3.4 Efficiently Finding Regions of Interest

A region qualifying as a dense region, but containing exclusively one cluster, possibly need to be defined by a rather small interval of angles and also a rather small interval of distances from the origin because otherwise the same interval could also contain functions representing points of other clusters (cf. the dense region of cluster C1 in Figure 3). For that purpose, a rather high number of intervals in each dimension of the parameter space is needed, resulting in a huge number of grid cells possibly qualifying as dense regions. Thus, searching the parameter space with a predefined grid in the range $[0, \pi)$ for each angle and $[\delta_{min}, \delta_{max}]$ for the distance from the origin, is not feasible for high dimensional data in terms of space and time complexity.

To avoid exponential complexity, the following search strategy for the parameter space is proposed:

1. The axes (distance and angles) are divided successively in a static order given by $\delta, \alpha_1, \ldots, \alpha_{d-1}$. After dividing one axis, from the resulting 2 hypercuboids the one containing most points is selected for refinement. If both hypercuboids contain an equal amount of points, the first one is selected (arbitrarily). The selected hypercuboid is divided recursively by splitting the next axis. The neglected hypercuboid is kept in a queue.

2. If both children of a divided hypercuboid contain less than m points, the search in the corresponding path is discontinued. Unless the queue is empty, the next hypercuboid in the queue is examined using the same procedure. In the queue, hypercuboids are ordered descendingly by the amount of points contained by a hypercuboid. If two hypercuboids contain an equal amount of points, the smaller one is preferred, since a smaller interval containing an equal amount of data points is a more promising candidate.

3. At a predefined depth (i.e. a given number s of successive splits), a hypercuboid (i.e. the corresponding interval) is considered to be sufficiently small to define a hyperplane containing a subspace cluster. If the number of points within the hypercuboid exceeds a predefined number m of points, these points are considered to build a subspace cluster. The corresponding subspace is treated as a new data space containing all the points accounted for in the hypercuboid. This new data space of dimensionality $d-1$ undergoes the same procedure recursively, while $d > 2$, i.e., CASH is called for the points in the hypercuboid using the corresponding subspace as data space. If no subspace cluster of lower dimensionality is found in this $(d-1)$-dimensional space, all the points in this subspace are supposed to build a $(d-1)$-dimensional subspace cluster.

4. All points participating at a $(d-1)$-dimensional subspace cluster derived at a search path are removed

from the d-dimensional data space. The queue is reorganized and hypercuboids are removed, if they contain now less than m points. A new search path based on the next hypercuboid in the queue is pursued.

5. The search is complete, if in the d-dimensional space no interval is found containing at least m points.

This search strategy determines clusters of at least m points in any arbitrarily oriented subspace and provides a description with an accuracy regarding the orientation α and the distance δ from the origin as defined by the predefined number s of splits.

Unlike traditional grid-based clustering approaches, CASH has no problems if a region of interest (i.e., a cluster) is located at the boundary of two connected grid cells, g_1 and g_2. In that case, the functions will intersect both neighboring grid cells and both grid cells, g_1 and g_2, will be dense. CASH will refine one of these grid cells (e.g. g_1 – cf. step 1) until the cluster is found. After that, CASH eliminates the participating points (i.e., functions) and, thus, the second grid-cell (g_2) will not be dense anymore (step 4).

Due to the recursive search in an obtained cluster (step 3), a cluster hierarchy is gained along the way, i.e., a subspace cluster may in turn contain nested subspace clusters of lower dimensionality. In that case, it may be interesting to report all nested clusters and the information of the "contained-in" relationships.

3.5 Properties of the Algorithm

The algorithm CASH transforms the data objects from $\mathcal{D} \subseteq \mathbb{R}^d$ into a corresponding parameter space (based on radius and angles) $\mathcal{P} = [\delta_{min}, \delta_{max}] \times [0, \pi)^{d-1}$. After that, CASH identifies dense regions in that parameter space using the search strategy proposed above. These dense regions represent arbitrarily oriented subspace clusters in the data space.

Complexity. Let N be the number of data points in a d dimensional data space. When bisecting the parameter space of α_i and δ, we need to determine those database points, whose parameter functions intersect with the generated cells in the parameter space. This is done by the maximization and minimization of δ given the constraints on α_i (i.e., $\check{\alpha}_i \leq \alpha_i < \hat{\alpha}_i$ for all $1 \leq i \leq d-1$) requiring $O(d^3)$ time per object and cell.

The CASH algorithm performs a recursive bisection of the data space where all bisections with fewer than m associated database points are discarded. Since bisection cells which do not belong to any cluster are only randomly associated to a few arbitrary points, the bisection process for those cells stops at a high level of the bisection tree. Only cells belonging to actual subspace clusters are bisected until the defined maximum number s of bisection levels is reached. Therefore, for a data set containing c clusters, a number $O(s \cdot c)$ of nodes in the bisection tree are encountered, each causing $O(N \cdot d^3)$ work to find all subspace clusters. Together, we have a time complexity in $O(s \cdot c \cdot N \cdot d^3)$.

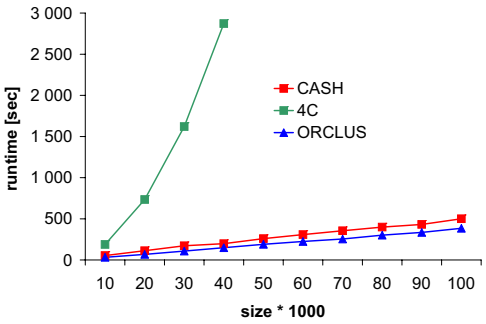


Figure 4: Scalability w.r.t. size.

Input Parameters. CASH requires the user to specify two input parameters: The first parameter m specifies the minimum number of sinusoidal curves that need to intersect a hypercuboid in the parameter space such that this hypercuboid is regarded as a dense area. Obviously, this parameter represents the minimum number of points in a cluster and thus is very intuitive. The second parameter s specifies the maximal number of splits along a search path (splitlevel). Thus, it controls the maximal allowed deviation from the hyperplane of the cluster in terms of orientation and jitter. We show in our experiments, that CASH is rather robust w.r.t. s. Since CASH does not require parameters that are hard to guess like the number of clusters, the average dimensionality of the subspace clusters, or the size of the Euclidean neighborhood based on which the similarity of the subspace clusters is learned, it is much more usable and stable than its competitors.

4 Evaluation

4.1 Efficiency

To evaluate the scalability of CASH w.r.t. the size of the data set, we created ten data sets containing four, equally sized one dimensional clusters in a 5 dimensional data space with an increasing number of points ranging from 10,000 to 100,000. CASH performs comparably well to ORCLUS. Both outperform 4C significantly (cf. Figure 4). As a fair setting, we gave as parameter k to ORCLUS the exact number of clusters in the data set (i.e. $k = 4$), and parameter l has been set to the correct correlation dimensionality of the clusters (i.e. $l = 1$). For 4C, the parameters have been set to $k = \mu = 100$, $\varepsilon = 0.1$, $\lambda = 1$, and $\delta = 0.01$, reflecting the actual cluster structure in the synthetic data sets. The

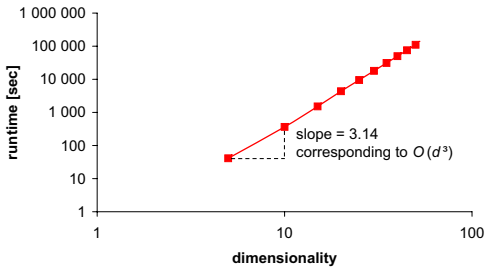


Figure 5: Scalability of CASH w.r.t. dimensionality.

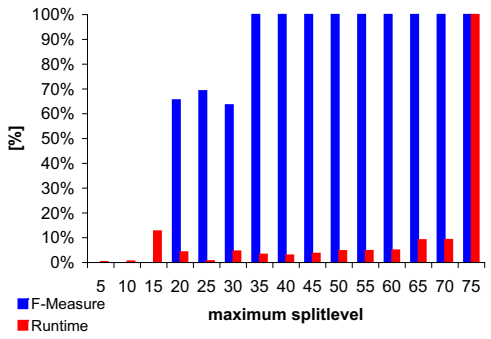


Figure 6: F-measure and runtime of CASH w.r.t. maximum split level.

parameter setting for CASH was $s = 40$ and $m = 2,500$.

To assess the impact of the dimensionality of the data space on the runtime of CASH, we created 10 data sets ranging in dimensionality from 5 to 50, each data set containing a one dimensional cluster of 10.000 points. The parameters were set to $s = 50$ and $m = 5,000$. Figure 5 shows the scalability of CASH logarithmically on both axes, dimensionality and runtime. The graph is a line with slope 3.14, approximately corresponding to the expected runtime behavior.

In both test scenarios, the objective was to find 1-dimensional clusters in a d-dimensional data space, since this is the most complex task for CASH, requiring a maximal recursive descent from $d - 1$ until subspace dimensionality 1 is reached.

4.2 Effectivity

The parameter s clearly influences the runtime behavior to a certain degree. However, CASH reaches satisfying behavior in terms of effectivity for even relatively low values for s. Figure 6 illustrates the effect of s on runtime and effectivity simultaneously. On a 5-dimensional data set containing two 1-dimensional clusters, each containing 500 points, and 500 points of noise, CASH reaches an F-measure of 100% already for $s = 35$, while the runtime remains relatively low with

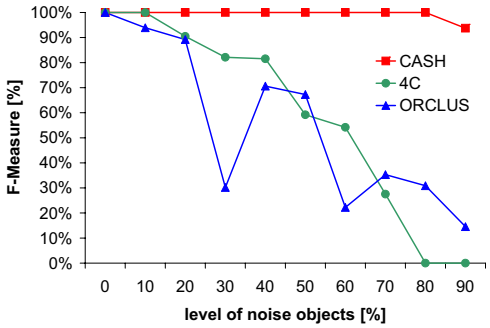


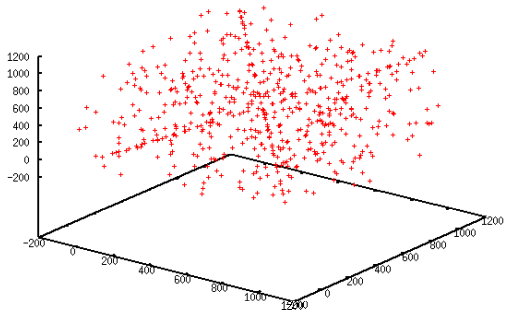
Figure 7: F-Measure w.r.t. noise level.

3.29% compared to the maximum runtime for $s = 75$.

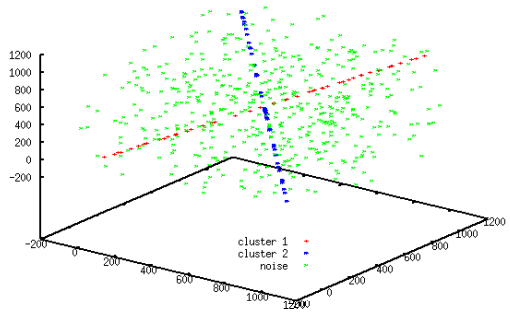
To assess the robustness of CASH against noise, we created ten data sets containing an increasing level of noise objects ranging from 0 to 90% of the complete data set. Figure 7 shows the comparison in robustness with ORCLUS and 4C. The parameter setting for ORCLUS has been $l = 1$ and $k = 2$, reflecting the true number of clusters and their dimensionality. For 4C, the optimal parameter setting has been used with $k = \mu = 5$, $\varepsilon = 0.12$, $\lambda = 1$, and $\delta = 0.01$. For CASH, the parameters have been chosen as $s = 30$ and $m = 50$. Let us note that CASH did not require any efforts for optimization of parameter settings. While both 4C and ORCLUS performed relatively well for very low levels of noise objects, their performance deteriorates for a higher degree of noise. CASH remains constantly on an F-measure of 100% up to a noise level of 80%. Even for an extremely high level of noise (90%), CASH still reaches an F-measure of 94%.

We illustrate the robustness of CASH against noise on an exemplary 3-dimensional data set depicted in Figure 8(a). CASH finds the 2 1-dimensional subspace clusters (each of size 50) embedded in 500 noise points exactly (cf. Figure 8(b)). The results of ORCLUS (with optimal parameter setting $l = 1$ and $k = 2$) are shown in Figures 8(c) and 8(d). As it can be seen, the clusters found by ORCLUS do not reflect the real cluster structure at all. For 4C, we tried several parameter settings. Unfortunately, 4C was never able to find a meaningful cluster structure at all.

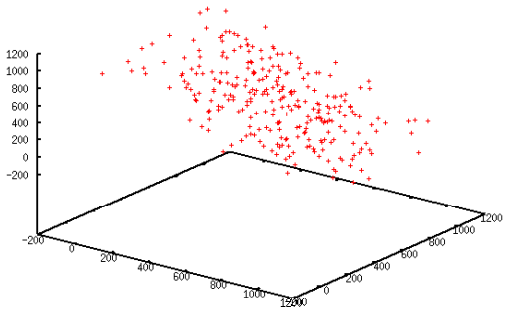
Further experiments on high dimensional data sets have been performed with CASH, 4C, and ORCLUS. The data sets contained complex subspace cluster structures with sparse clusters, including subspace clusters of significantly differing dimensionality, subspace clusters hierarchically embedded in higher dimensional subspaces, and noise objects. In none of the performed experiments, 4C or ORCLUS were able to find meaningful clusters, while CASH exactly detected the cluster



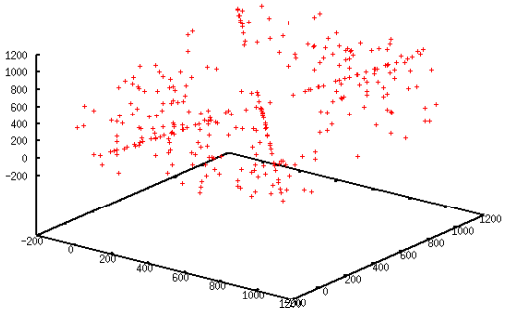
(a) Synthetic data set DS1.



(b) CASH - Clustering.

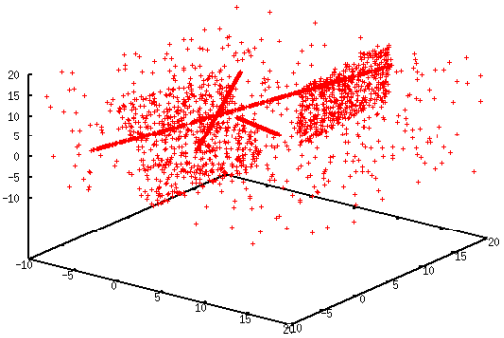


(c) ORCLUS - Cluster 1.

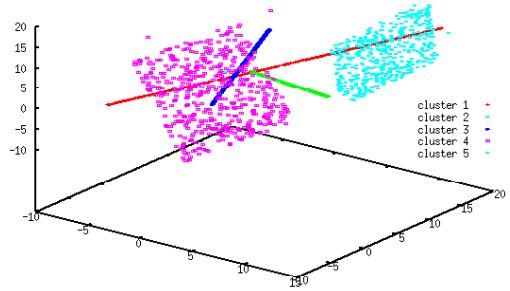


(d) ORCLUS - Cluster 2.

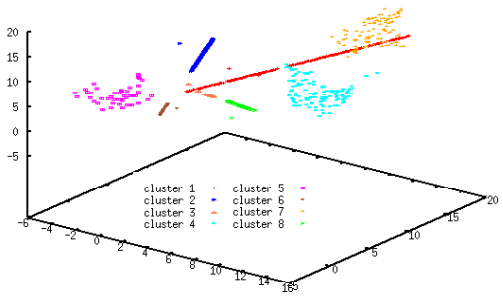
Figure 8: Clustering synthetic data set DS1.



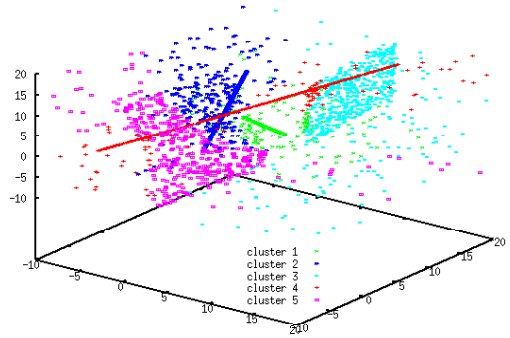
(a) Data set DS2.



(b) CASH – Cluster 1 - 5.

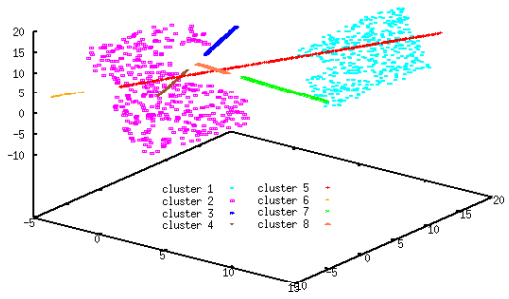


(c) 4C – Cluster 1 - 8.

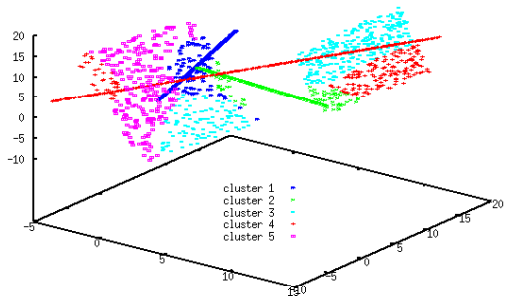


(d) ORCLUS – Cluster 1 - 5.

Figure 9: Clustering results on synthetic data set DS2.



(a) 4C – Cluster 1 - 8.



(b) ORCLUS – Cluster 1 - 5.

Figure 10: Clustering results on DS2 after noise removal.

Table 1: CASH clustering on Wages data.

c ID	dim	# objects	Description
1	2	215	YE = 12; A - YW = 18
2	2	70	YE = 16; A - YW = 22
3	3	247	YE + YW = A - 6

Table 2: CASH clustering on NBA data.

c ID	dim	Description
1	1	"go-to-guys"
2	2	shooting guards
3	2	point guards
4	2	starting centers
5	8	point guards
6	9	power forwards
7	9	small forwards
8	10	well-known rebounder
9	12	role players/reserves

structures in most cases. As an example, we present the results on a complex 3-dimensional data set shown in Figure 9(a), containing three 1-dimensional clusters each of 500 points, two 2-dimensional planes each containing 500 points, and 500 points of noise. One of the planes is intersected by two lines, the other plane is intersected by one line. Here, CASH is able to identify the cluster structure of all 5 clusters exactly (Figure 9(b)). In contrast, 4C (cf. Figure 9(c), parameters optimized to $k = \mu = 20$, $\varepsilon = 0.1$, $\lambda = 2$, and $\delta = 0.01$) and ORCLUS (cf. Figure 9(d), parameters $k = 5$ and $l = 2$ reflect the cluster structure exactly) could not compete. Both missed very large and important parts of the clustering structure. This bad behaviour of the two competitors can partly be explained by the high degree of noise present in the data set. The influence of noise on the existing approaches can be observed in Figure 10. Omitting the noise points, 4C is able to detect the cluster structure relatively well (cf. Figure 10(a)) but cannot handle intersecting clusters. Even on the data set without noise points, ORCLUS was not able to identify the 5 clusters correctly (cf. Figure 10(b)). This again illustrates the superiority of CASH over existing methods especially in terms of noise robustness.

4.3 Real-World Data

We applied CASH on the Wages data set[1], a data set containing average career statistics of current and former NBA players[2] and a gene expression data set [16]. The Wages data consist of 534 4D observations (A=age, YE=years of education, YW=years of work experience, and W=wage) from the 1985 Current Population Survey. As parameters for CASH we used $m = 70$ and $s = 40$. The results are summarized in Table 1: CASH detected three pure subspace clusters in this data set, two data objects have been identified as noise objects. The first cluster consists only of people having 12 years of education and having started their working life at the age of 18. The second cluster consists only of people having 16 years of education and having started their working life at the age of 22. In the third cluster only those employees are grouped, which started school in the age of 6 years and after graduation immediately began working. Thus, the sum of years of education and work experience equals the age minus 6.

The NBA data contains 15 statistical measures such as "games played" (G), "games started" (GS), "minutes played per game" (MPG), "points per game" (PPG), etc. for 413 former and current NBA players. As parameters for CASH we used $m = 30$ and $s = 45$. CASH detected 9 interesting clusters of very different dimensionality each containing players of similar characteris-

[1] `http://lib.stat.cmu.edu/datasets/CPS_85_Wages`

[2] obtained from `http://www.nba.com`

tics (cf. Table 2). In addition, several players were noise. The detected correlations confirmed basketball fundamentals. For example, in cluster 1 containing superstars like Michael Jordan, Larry Bird, Shaquile O'Neal, and James Worthy, PPG of all players were negatively dependent on G and MPG. On the other hand, the more games the players were in (G), the higher the number of starting line-up appearances (GS). Let us note that this cluster also contains less well-known players that had similar characteristics such as Rik Smits, Dan Majerle, and Rick Fox. The three clusters containing guards all showed correlations between G and MPG on the one hand, and the number of assists and steals per game on the other hand. For the guards in cluster 3, this correlation was positive, whereas for the guards in cluster 5, this correlation was negative. On the other hand, cluster 3 exhibits a positive correlation between the G and GS. In cluster 5 these two attributes are also correlated but in a negative fashion. This indicates that the coaches in the NBA usually decided to start with the better point guards.

In the gene expression data set (24 dimensions, 4,000 genes) CASH found several clusters of functionally related genes that are biologically interesting and relevant (details are omitted due to space limitations).

Neither ORCLUS nor 4C were able to detect meaningful clusters in our real-world data sets. One reason for this may be that the found clusters are highly overlapping. Thus, neither ORCLUS nor 4C can learn the appropriate similarity measure capturing the subspaces of the clusters from the local neighborhood.

5 Conclusions

Existing subspace clustering methods only find clusters that are dense and not noisy in the original feature space. In this paper, we overcome this severe limitation by introducing CASH, a novel approach for detecting any oriented subspace cluster regardless of its density even in a very noisy environment. Our experimental evaluation confirms that CASH significantly outperforms competing approaches in terms of robustness and effectivity.

References

[1] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. On exploring complex relationships of correlation clusters. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM), Banff, Canada*, 2007.

[2] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. Robust, complete, and efficient correlation clustering. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM), Minneapolis, MN*, 2007.

[3] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA*, 1999.

[4] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional space. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX*, 2000.

[5] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Seattle, WA*, 1998.

[6] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *Proceedings of the 4th International Conference on Data Mining (ICDM), Brighton, U.K.*, 2004.

[7] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing clusters of correlation connected objects. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Paris, France*, 2004.

[8] K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt*, 2000.

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

[10] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt*, 2000.

[11] P. V. C. Hough. Methods and means for recognizing complex patterns. U.S. Patent 3069654, December 18 1962.

[12] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM), Orlando, FL*, 2004.

[13] J. Liu and W. Wang. OP-Cluster: Clustering by tendency in high dimensional spaces. In *Proceedings of the 3th International Conference on Data Mining (ICDM), Melbourne, FL*, 2003.

[14] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu. MaPle: A fast algorithm for maximal pattern-based clustering. In *Proceedings of the 3th International Conference on Data Mining (ICDM), Melbourne, FL*, 2003.

[15] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD),*

Madison, WI, 2002.

[16] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[17] A. K. H. Tung, X. Xu, and C. B. Ooi. CURLER: Finding and visualizing nonlinear correlated clusters. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Baltimore, ML*, 2005.

[18] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Madison, WI*, 2002.

[19] J. Yang, W. Wang, H. Wang, and P. S. Yu. δ-clusters: Capturing subspace correlation in a large data set. In *Proceedings of the 18th International Conference on Data Engineering (ICDE), San Jose, CA*, 2002.

A Global Extremum

Each parametrization function f_p is a sinusoid with a period of 2π. Thus, any f_p has exactly one global extremum in the interval $[0,\pi)^{d-1}$. To find the global extremum of f_p, those angles $\alpha_1,\ldots,\alpha_{d-1}$ need to be determined where all the first order derivatives of f_p are zero, and the Hessian matrix is either positive or negative definite. The *first order partial derivatives* of the parametrization function f_p are given by:

$$\frac{\partial f_p}{\partial \alpha_n}(\alpha) = \prod_{i=1}^{n-1}\sin(\alpha_i) \cdot \Big(-p_n\cdot\sin(\alpha_n) + \textstyle\sum_{j=n+1}^{d} p_j\cdot\cos(\alpha_n) \cdot \Big[\prod_{k=n+1}^{j-1}\sin(\alpha_k)\Big]\cdot\cos(\alpha_j)\Big)$$

For any first order partial derivative $\frac{\partial f_p}{\partial \alpha_n}(\tilde{\alpha}) \doteq 0$, $(1 \le n \le d-1)$, one of the following conditions holds:

$$\begin{aligned}\sin(\tilde{\alpha}_1) &= 0\\ &\vdots\\ \sin(\tilde{\alpha}_{n-1}) &= 0\\ \tan(\tilde{\alpha}_n) &= \frac{\sum_{j=n+1}^{d} p_j\cdot\big[\prod_{k=n+1}^{j-1}\sin(\tilde{\alpha}_k)\big]\cdot\cos(\tilde{\alpha}_j)}{p_n}\end{aligned}$$

Since the first $n-1$ conditions yield an indefinite Hessian matrix, according to the last condition, a point $\tilde{\alpha} = (\tilde{\alpha}_1,\ldots,\tilde{\alpha}_{d-1})$ can be an extremum point of parametrization function f_p only if

$$\tilde{\alpha}_n = \arctan\left(\frac{\sum_{j=n+1}^{d} p_j\cdot\big[\prod_{k=n+1}^{j-1}\sin(\tilde{\alpha}_k)\big]\cdot\cos(\tilde{\alpha}_j)}{p_n}\right)$$

As noted above, f_p is guaranteed to have exactly one global extremum $f_p(\tilde{\alpha}_1,\ldots,\tilde{\alpha}_{d-1})$ in $[0,\pi)^{d-1}$. The values for the angles $\tilde{\alpha}_n, n = 1,\ldots,d-1$ of the global extremum are given by the equation above.

B Minimum and Maximum Value

Let $\check{\alpha} = (\check{\alpha}_1,\ldots,\check{\alpha}_{d-1})$ and $\hat{\alpha} = (\hat{\alpha}_1,\ldots,\hat{\alpha}_{d-1})$ for a given interval $[\check{\alpha},\hat{\alpha}) \subseteq [0,\pi)^{d-1}$, $1 \le i \le d-1$. To determine the point $\alpha^{min} = (\alpha_1^{min},\ldots,\alpha_{d-1}^{min})$ where the parametrization function f_p has a minimum in interval $[\check{\alpha},\hat{\alpha})$ the following steps for each dimension $n = d-1,\ldots,1$ have to be performed:

1. Let $\overline{\alpha}_n = \arctan\left(\frac{\sum_{j=n+1}^{d} p_j\cdot\big[\prod_{k=n+1}^{j-1}\sin(\alpha_k^{\min})\big]\cdot\cos(\alpha_j^{\min})}{p_n}\right)$ be the value where f_p has an extremum on the n-th axis under the constraint of known minimum angles $\alpha_{n+1}^{\min},\ldots,\alpha_{d-1}^{\min}$.
2. Given $\alpha = (c_1,\ldots,c_{n-1},\overline{\alpha}_n,\alpha_{n+1}^{\min},\ldots,\alpha_{d-1}^{\min}) \in [\check{\alpha},\hat{\alpha})^{n-1}\times[0,\pi)\times[\check{\alpha},\hat{\alpha})^{d-1-n}$, where c_i are arbitrarily chosen values in $[\check{\alpha},\hat{\alpha})$, we differentiate the following cases:
   i. f_p has a *maximum* in α:
      A. $\check{\alpha}_n \le \overline{\alpha}_n \le \hat{\alpha}_n$:
         A1. $\overline{\alpha}_n - \check{\alpha}_n \le \hat{\alpha}_n - \overline{\alpha}_n$: $\alpha_n^{\min} \to \hat{\alpha}_n$.
         A2. $\overline{\alpha}_n - \check{\alpha}_n > \hat{\alpha}_n - \overline{\alpha}_n$: $\alpha_n^{\min} = \check{\alpha}_n$.
      B. $\overline{\alpha}_n < \check{\alpha}_n$: $\alpha_n^{\min} \to \hat{\alpha}_n$.
      C. $\overline{\alpha}_n > \hat{\alpha}_n$: $\alpha_n^{\min} = \check{\alpha}_n$.

      As illustrated in Figure 11, if $\overline{\alpha}_n$ is inside the interval and nearer to the left boundary (A1), the minimum value $\alpha_n^{\min}$ is located at the right boundary and *vice versa* (A2). If $\overline{\alpha}_n$ is outside the interval (B and C), the minimum value $\alpha_n^{\min}$ is located at the opposite boundary.
   ii. f_p has a *minimum* in α: The same principle of reasoning has to be applied contrarilywise.
      A. $\check{\alpha}_n \le \overline{\alpha}_n \le \hat{\alpha}_n$: $\alpha_n^{\min} = \overline{\alpha}_n$.
      B. $\overline{\alpha}_n < \check{\alpha}_n$: $\alpha_n^{\min} = \check{\alpha}_n$.
      C. $\overline{\alpha}_n > \hat{\alpha}_n$: $\alpha_n^{\min} \to \hat{\alpha}_n$.

The maximum $\alpha^{max} = (\alpha_1^{max},\ldots,\alpha_{d-1}^{max})$ of f_p in a given interval $[\check{\alpha},\hat{\alpha}) \subseteq [0,\pi)^{d-1}$ can be determined analogously.

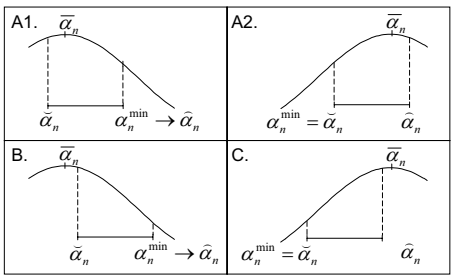


Figure 11: Different cases for finding the minimum of a parametrization function in a given interval.