RC 22103 (99082) 29 June 2001 **Computer Science**

IBM Research Report

A Human-Computer Cooperative System For Effective High-Dimensional Clustering

Charu C. Aggarwal
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A Human-Computer Cooperative System For Effective High-Dimensional Clustering

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
charu@watson.ibm.com

Abstract

Clustering problems are well known in the database literature for their use in numerous applications such as customer segmentation, classification and trend analysis. High dimensional data has always been a challenge for clustering algorithms because of the inherent sparsity of the points. Recent research results have shown that it is often not meaningful to find clusters in high dimensional data with traditional measures of proximity: since such measures themselves become questionable in high dimensionality. Therefore, techniques have recently been proposed to find clusters in hidden subspaces of the data. However, since the behavior of the data may vary considerably in different subspaces, it is often difficult to define the notion of a cluster with the use of simple mathematical formalizations. In fact, the meaningfulness and definition of a cluster is best characterized with the use of human intuition. In this paper, we propose a system which performs high dimensional clustering by effective cooperation between the human and the computer. The complex task of cluster creation is accomplished by a combination of human intuition and the computational support provided by the computer. The result is a system which leverages the best abilities of both the human and the computer in order to create very meaningful sets of clusters in high dimensionality.

1 Introduction

The clustering problem is formally defined as follows: Given a set of points in multidimensional space, find a partition of the points into clusters so that the points within each cluster are similar to one another. Various distance functions may be used in order to make a quantitative determination of similarity. In addition, an objective function may be defined with respect to this distance function in order to measure the overall quality of a partition. The clustering problem is used for similarity search, customer segmentation, pattern recognition, trend analysis and classification. The method has been widely studied by both the statistics and database communities for different domains of data [10, 11, 13, 18, 22, 23]. Detailed surveys on clustering methods can be found in [16].

Most clustering algorithms do not work efficiently in higher dimensional spaces because of the inherent sparsity of the data. This problem has been traditionally referred to as the ***dimensionality*** curse. Recent theoretical results [8] have shown that in high dimensional space, the distance between every pair of points is almost the same for a wide variety of data distributions and distance functions. Under such circumstances, even the meaningfulness of proximity or clustering in high dimensional data is questionable. An interesting fact about high dimensional data is that even though the data is sparse in full dimensionality, certain projections of the data reveal clearly separated clusters [3]. Most real data contains different kinds of skews in which some subsets of dimensions are related

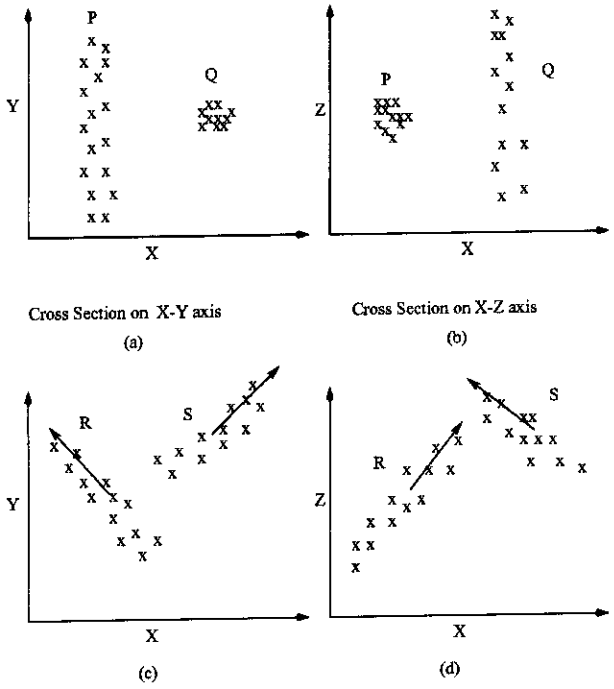

Figure 1: Illustrations of Lower Dimensional Clusters

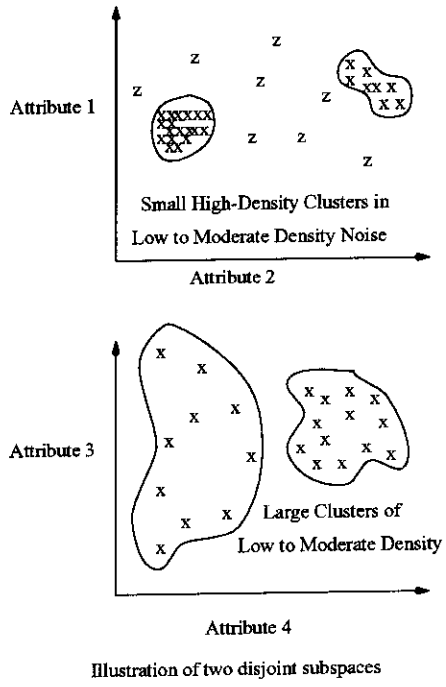

Figure 2: Variation in cluster shape, size and density across data localities and subspaces

to one another. Furthermore, these subsets of correlated dimensions may vary with data locality [2, 9]. Correlated sets of dimensions lead to points getting aligned along arbitrary shapes in lower dimensional space. Such distributions create clusters in lower dimensional projections and are referred to as *projected clusters*. Techniques for finding projected clusters in lower dimensional spaces have been discussed in [1, 2, 3]. Such clusters may exist either in projections of the original set of attributes or in arbitrary lower dimensional subspaces. In Figures 1(a) and (b), we have illustrated a data set in which two clusters P and Q exist in projections of the data which are parallel to the original set of attributes. In Figures 1(c) and (d), we have illustrated a different case in which the clusters R and S exist in completely arbitrary projections of the data. We note that the formalization for finding clusters in projections of the original set of attributes provides greater interpretability [3], whereas that of picking arbitrary projections is more flexible in discovering clusters created by inter-attribute correlations [2].

It may often be the case that the density, distribution, and shapes of the clusters may be quite different in different data localities and subspaces. We have illustrated examples of such cases in Figure 2. In many cases, a region of low density can be clearly distinguished as a separate cluster in one subspace, but regions with similar density correspond to noise in another subspace. Often, even within a subspace, the clusters can be distinguished from one another only on a case-by-case basis. Such clusters are difficult to isolate using fully automated methods in which simple mathematical formalizations are used as the only criterion in order to define all clusters. Since there is so much variation across different data localities and projections, it is difficult to reconcile these differences

without the use of human intervention. At the same time, since there are a very large[1] number of subspaces in the high dimensional case, human involvement necessitates the exploration of only a small fraction of the subspaces. Thus, computational support is required in order to minimize the effort in finding clusters in optimally chosen subspaces. Clustering of massive high dimensional data sets is a problem which requires both computational power and intuitive understanding; therefore, a natural solution is to divide the clustering task in such a way that each entity performs the task that it is most well suited to. In the system thus devised, the computer performs the high dimensional data analysis which is used in order to provide the user with summary feedback; this feedback is given in a way so that the human is facilitated in his intuitive task of characterizing the clusters. The result of this cooperative technique is a system which can perform the task of high dimensional clustering better than either a human or a computer.

In recent years, considerable progress has been made on the incorporation of human interaction and exploration into the data mining process. Several methods for interactive classification, visualization, and high dimensional data exploration and mining have been developed [6, 7, 15, 19, 24]. Other recent work deals with the problem of incorporating user constraints into various data mining problems [12, 21]. We especially note the interesting technique in [15], which describes a graphical tool for users to interact with clusters in lower dimensional views of the data. The job of picking these views is largely left to the user; a task which becomes more difficult (and time-constrained) with increasing dimensionality.

In this paper, we describe a human-computer cooperative system for high dimensional clustering. The difficult process of determining the subspaces in which the clusters are best revealed is performed by the computer, and presented visually to the user. The user then applies his intuitive judgement in separating out the clusters in this subspace. This process is repeated iteratively until it is determined that most points have been covered in one or more clusters. The reactions of the user are utilized in order to determine and quantify the meaningfulness of the final set of clusters which are highly interpretable in terms of the user-reactions.

This paper is organized as follows. In the remainder of this section, we will introduce the notations and definitions and a formal description of the contributions. In section 2, we will describe the interactive system for high dimensional clustering. The performance of the algorithm on several real data sets is discussed in section 3. Section 4 contains the conclusions and summary.

1.1 Basic Definitions

In order to describe our algorithm further, we shall introduce a few notations and definitions. Let N denote the total number of data points. Assume that the dimensionality of the overall space is d. We assume that the data set is denoted by $\mathcal{D}$ and the full dimensional space by $\mathcal{U}$. Let $\mathcal{C} = \{\overline{x_1}, \overline{x_2}, \ldots, \overline{x_t}\}$ be the set of points in a cluster.

Let $\overline{y} = (y_1, \ldots y_d)$ be a point in the d-dimensional space, and let $\mathcal{E} = \{\overline{e_1} \ldots \overline{e_l}\}$ be a set of $l \leq d$ orthonormal vectors in this d-dimensional space. These orthonormal vectors define a subspace. We note that the set of vectors $\overline{e_1} \ldots \overline{e_l}$ may be drawn either from the original set of atrribute directions, or they may be arbitrary vectors. The advantage of finding clusters in projections of

[1]There are infinitely many if arbitrary subspaces of the data are picked.

the original attributes is better interpretability, whereas that of choosing arbitrary projections is greater flexibility. In this paper. we will discuss methods for determining clusters of both kinds.

The projection $\mathcal{P}(\overline{y}, \mathcal{E})$ of point $\overline{y}$ in subspace $\mathcal{E}$ is an l-dimensional point $(\overline{y} \cdot \overline{e_1} \ldots \overline{y} \cdot \overline{e_l})$. Here $\overline{y} \cdot \overline{e_i}$ denotes the dot-product of $\overline{y}$ and $\overline{e_i}$. Let $\overline{y_1}$ and $\overline{y_2}$ be two points in the original d-dimensional space. Then, the projected distance between the points $\overline{y_1}$ and $\overline{y_2}$ in subspace $\mathcal{E}$ is denoted by $Pdist(\overline{y_1}, \overline{y_2}, \mathcal{E})$ and is equal to the euclidean distance between the projections $\mathcal{P}(\overline{y_1}, \mathcal{E})$ and $\mathcal{P}(\overline{y_2}, \mathcal{E})$ in the l-dimensional space represented by $\mathcal{E}$. The *subspace momentum* of the cluster $\mathcal{C} = \{\overline{x_1}, \overline{x_2}, \ldots, \overline{x_t}\}$ in subspace $\mathcal{E}$ about the point $\overline{y}$ is denoted by $R(\mathcal{C}, \mathcal{E}, \overline{y})$ and is given by the mean square distance of the points to $\overline{y}$, when all points in $\mathcal{C}$ are projected to the subspace $\mathcal{E}$. Thus, we have:

$$R(\mathcal{C}, \mathcal{E}, \overline{y}) = \sum_{i=1}^{t} \{Pdist(\overline{x_i}, \overline{y}, \mathcal{E})\}^2 / t \tag{1}$$

Note that the momentum of the cluster $\mathcal{C}$ about a given point $\overline{y}$ is less in a subspace $\mathcal{E}$ than that in the full dimensional space. When the data is polarized tightly around $\overline{y}$, this momentum is likely to be significantly lower. For example, for the case of Figures 1(c) and (d), if the data point $\overline{y}$ was chosen from the cluster R, then the subspace momentum for the points in R is likely to be low when projected onto the 2-dimensional plane normal to the arrow illustrating R in the same figure. In this paper, we will see that such subspaces are very helpful to the user in separating out the prominent clusters.

1.2 Contributions of this paper

In this paper, we discuss a method for high dimensional clustering which works by effective cooperation between the human and the computer. Methods are discussed for finding projections of the data in which certain clusters are clearly distinguished from the rest of the data. The user is provided with well chosen visual cross-sections of the data using these projections. He is then provided the ability to visually isolate the clusters in these cross-sectional views. The views may be arbitrary projections or be chosen from a combination of the original set of attributes in the interests of greater interpretability. We note that although these views are 2-dimensional because of the inherent limits of human visual perception, the final clusters are created by a combination of different views. Therefore, clusters are discovered in higher dimensional projections as well. Methods are proposed to use the entire set of user reactions in order to construct a set of clusters which exist as distinctly separated sets in multiple projections. The meaningfulness of the final clusters is quantified using the statistical behavior of the number of views in which a set of points is classified by the user into the same cluster. Thus, even the final effectiveness of clustering is measured purely from the user perspective. This approach has the added advantage that since the user is directly involved in the process of visualizing and creating the clusters, he has a clear understanding of the significance and interpretability of the final set of clusters.

2 The Interactive Clustering Algorithm

We will first provide a high level overview of the clustering algorithm; in a later section, we will provide a detailed description of the various steps. The idea behind the Interactive Projected Clustering algorithm (IPCLUS) is to provide the user with a set of meaningful visualizations in

Algorithm *IPCLUS(Data Set: $\mathcal{D}$, Minimum Support: s, Number of Polarization Points: k)*;
begin
{ The Id Vector $\mathcal{I} = \{I_1, \ldots I_N\}$ has one entry string
for each record in the database and is initialized to null }
while not(termination_criterion) **do**
begin
Randomly sample k points $y_1 \ldots y_k$ from the database;
$\mathcal{E}$ = *ComputePolarizedProjection*($\mathcal{D}$, $y_1 \ldots y_k$, s);
$\mathcal{K}$ = *InteractiveClusterSeparation*($\mathcal{D}$, $\mathcal{E}$, $y_1 \ldots y_k$);
$\mathcal{I}$ = *UpdateIdStrings*($\mathcal{I}$, $\mathcal{K}$);
end;
$(C_1^F \ldots C_K^F, Q_1 \ldots Q_K)$ = *FinalClusterCreation*($\mathcal{I}$, s);
$(IR_1, \ldots IR_K)$ = *EvaluateMeaningfulness*($\mathcal{I}, C_1^F \ldots C_K^F, Q_1 \ldots Q_K$);
return$(IR_1 \ldots IR_K, C_1^F \ldots C_K^F, Q_1 \ldots Q_K)$;
end;

Figure 3: The IPCLUS Algorithm

lower dimensional projections together with the ability to decide how to separate the clusters. We assume that as input to the system we provide a user-defined *support*, which is the minimum fraction of database points in a cluster for it to be considered statistically significant.

The interactive clustering algorithms works in a series of iterations in each of which a projection is determined in which there are distinct sets of points which can be clearly distinguished from one another. We refer to such projections as *well polarized*. In a well polarized projection, it is easier for the user to clearly distinguish a set of clusters from the rest of the data. In order to create the polarizations, we pick a set of k records from the database which are referred to as the *polarization anchors*. A subspace of the data is determined such the data is clustered around each of these polarization anchors. The data is repeatedly sampled for polarization anchors in an iterative process, so that the most dominant subspace clusters containing these anchors are discovered.

In each iteration, when the projection subspace has been found, kernel density estimation techniques [20] can be used in order to determine the data density at each point in this projection. A visual profile of the data density is provided to the user, who uses this aid in order to find the most intuitive separation of the data into clusters. The cluster separation by the user is then recorded in the form of a set of N Identity Strings (*IdStrings*), where N is the total number of records in the database. These set of identity strings are denoted by $\mathcal{I} = (I_1 \ldots I_N)$. We assume that the rth string I_r corresponds to the rth data record. In the nth iteration, the value of the nth position in the identity string is recorded. If the rth data point does not belong to any of the clusters corresponding to the polarization points, then this position value is set at * (don't care), otherwise, we set the value to the index of the corresponding cluster in that particular projection. We will provide a more detailed discussion on this slightly later. As the termination criterion, we ensure that most points in the data are included in a cluster in some view. Specifically, we define the *coverage* of the data set as the average number of views in which a data point occurs in some cluster. The algorithm is terminated when the average coverage of each data point is above a user-defined threshold.

Algorithm *ComputePolarizedProjection(Data Set: $\mathcal{D}$, Polarization Anchors: $y_1 \ldots y_k$, Minimum Support: s);*
begin
$l_c = d/2$; $\mathcal{E}_c = \mathcal{U}$;
while $l_c > 2$ **do**
begin
for each data point $x \in \mathcal{D}$ and polarization anchor y_i compute $Pdist(y_i, x, \mathcal{E}_c)$;
Let the set of $N \cdot s$ data points closest to polarization anchor y_i be denoted by $\mathcal{M}_i$;
$\mathcal{N} = \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_k$;
Let z_i be the centroid of $\mathcal{M}_i$;
for $i = 1$ **to** k let $\mathcal{O}_i(z_i)$ be computed by subtracting
z_i from each point in $\mathcal{M}_i$;
Let c_{ij} be the covariance between any pair of dimensions (i, j) for set $\cup_{q=1}^{k} \mathcal{O}_q(z_q)$;
Let Δ be the $d * d$ matrix $[c_{ij}]$ for $[i, j] \in [dXd]$; }
Determine the eigenvectors $\overline{e_1} \ldots \overline{e_d}$ of matrix Δ;
$\mathcal{E}_c$ = Set of l_c vectors from $\overline{e_1} \ldots \overline{e_d}$ with least eignevalues;
$l_c = \max\{2, [l_c/2]\}$;
end
return$(\mathcal{E}_c)$;
end

Figure 4: Computing Polarized Projections

Algorithm *InteractiveClusterSeparation(Data Set: $\mathcal{D}$, Projection Subspace: $\mathcal{E}$, Polarization Anchors: $y_1 \ldots y_k$);*
begin
$i = 1$;
while (iflag==true) **do**
begin
Let the 2-dimensional hyperplane corresponding to the subspace $\mathcal{E}$ be $\mathcal{H}$;
Compute the density profile of the data along the hyperplane $\mathcal{H}$ using kernel density estimation;
Display the density profile along the hyperplane $\mathcal{H}$;
User visually sets the value of the noise threshold η or sets iflag=false;
Superpose the density profile with an axis-parallel hyperplane at user-defined noise threshold of η;
User decides whether current value of noise threshold η should be
stored as κ_i and increments i by 1 in that case;
end
$\mathcal{K}$ = All values of noise threshold specified by user = $\{\kappa_1 \ldots \kappa_i\}$;
return$(\mathcal{K})$;
end

Figure 5: Interactive Separation of Clusters

Algorithm *UpdateIdStrings(IdStrings: $\mathcal{I}$, Density Thresholds: $\mathcal{K}$)*;
{ We assume that K contains r noise thresholds $\{\kappa_1 \ldots \kappa_r\}$ }
begin
  for $i = 1$ **to** r **do**
  begin
    Find the sets of data points $\mathcal{C}_1 \ldots \mathcal{C}_{K_i}$ for noise threshold of κ_i;
    { Detailed Description of the separation process in section 2.3 };
    for each data point x **do**
      if x belongs to some cluster (say $\mathcal{C}_q$) **then**
          concatenate q at the end of I_q;
          else concatenate $*$ at the end of I_q;
  end;
end

Figure 6: Updating the Identity Vector from User Interaction

Algorithm *FinalClusterCreation(IdStrings: $\mathcal{I}$, Support: s)*;
begin
  Determine all maximal subpatterns of non-* alphabets with support at least s;
  { We assume that there are a total of K itemsets $Q_1 \ldots Q_K$ at support of s; }
  Let $\mathcal{C}_r^F$ be the set of points supported by the itemset $\mathcal{Q}_r$;
  return$(\mathcal{C}_1^F \ldots \mathcal{C}_K^F, Q_1 \ldots Q_K)$;
end;

Figure 7: Creation of final set of clusters

Algorithm *EvaluateMeaningfulness(IdStrings: $\mathcal{I}$, Clusters: $\mathcal{C}_1^F \ldots \mathcal{C}_K^F$, Cluster Templates: $Q_1 \ldots Q_k$)*;
begin
  for $i = 1$ **to** K **do**
  begin
    Let $S = Q_i$ be the itemset corresponding to cluster $\mathcal{C}_i^F$;
    Let the l items in S be denoted by $\{m_1 \ldots m_l\}$;
    for each $j \in \{1, \ldots l\}$ set β_j to the support of m_j;
    Set θ to the fractional support of cluster $\mathcal{C}_i^F$;
    $IR_i = \theta/\beta_1 \cdot \beta_2 \ldots \beta_l$;
  end;
    return$(IR_1, \ldots IR_K)$;
end

Figure 8: Evaluation of cluster meaningfulness

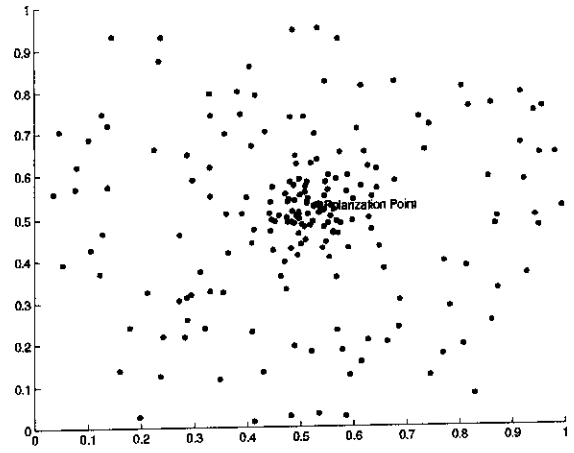

Figure 9: Determination of a well polarized projection (one polarization point)

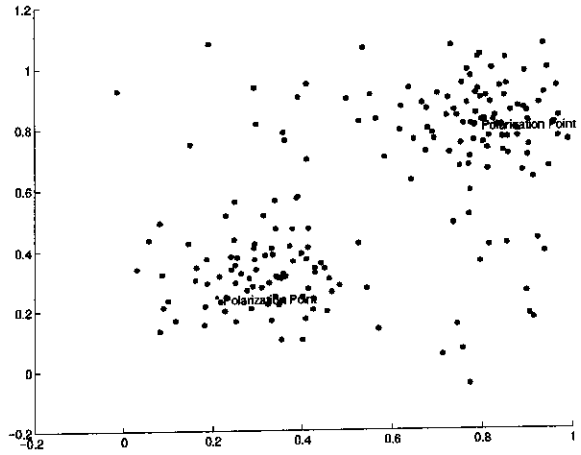

Figure 10: Determination of a well Polarized Projection (two polarization points)

At this stage all the user responses have been encoded in the identity strings which are postprocessed in order to create the final set of clusters. The final quality of the clustering is quantified in terms of the consistency of the user behavior across different projections. The overall framework of the algorithm is discussed in Figure 3. We note that the algorithm can be divided into two parts; the first part is iterative which involves the repeated user-interaction; the later step involves the determination of the final clusters and the quantification of the meaningfulness of these clusters. In the next few sections, we will describe the various components of the algorithm in detail, including the following iterative steps:

(1) Determination of subspaces in which the data is well polarized.

(2) User Interaction in order to visually separate the clusters in different views.

(3) Storing the user interactions in the form of *IdStrings*;

After the description of the iterative steps, we will also explain the final creation of the clusters from user responses and the quantification of meaningfulness from these responses.

2.1 Determination of Polarization Subspaces

The procedure for determination of the polarization step is described in the *ComputePolarizedProjection* procedure of Figure 4. The motivation of the procedure is to find sets of projections in which the data is well clustered and can be perceived in a clear visual way by the user. To this effect, in each iteration we sample a small number k of points from the database. The points are denoted by $y_1 \ldots y_k$, and are the polarization points around which we would like to find clusters. Since $y_1 \ldots y_k$ are sampled randomly, many of them may lie in a well defined cluster in a carefully chosen projection of the data. If a small number of polarization points are chosen, then it is reasonable that a projection can be found in which the data shows distinct clusters containing a majority of the polarization points. Of course, clusters may also exist in that projection which do not contain any of the polarization points. However, once a good projection is determined, all relevant clusters are used by the algorithm.

It is an interesting problem to find the projection subspace $\mathcal{E}$ in which the data shows this polarized

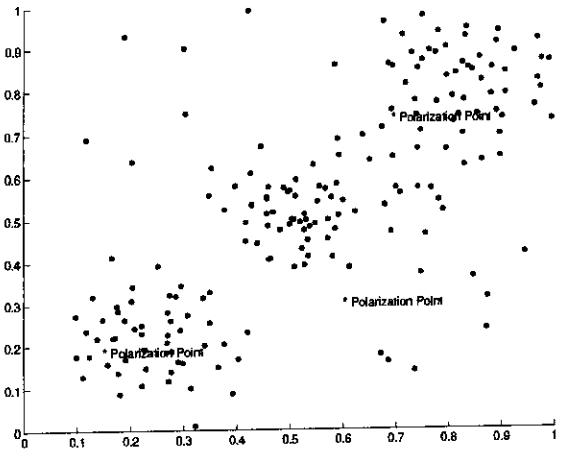

Figure 11: Partially Polarized Projection

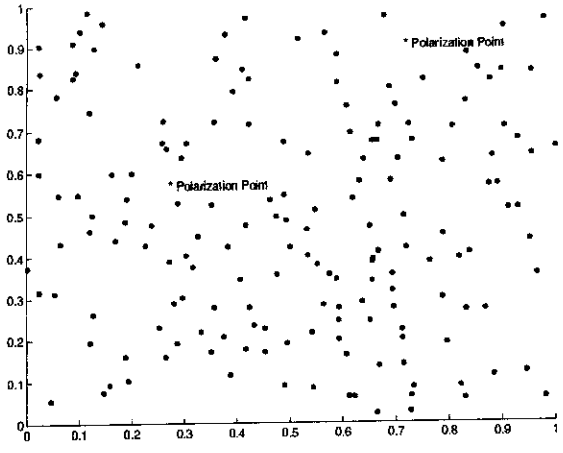

Figure 12: Poorly Polarized Projection

behavior. In Figures 9, 10, 11 and 12, we have illustrated a number of different possibilities for the relationship between the data and chosen polarization points in different projections. In Figures 9 and 10 the data is nicely separated out into different clusters using one and two polarization points respectively. On the other hand, in Figure 12, the projection is poorly chosen; so the data does not show clustered behavior around any polarization point. We note that it is often possible that no projection may contain well defined clusters around the data points which have been picked for the purpose of polarization. The larger the number of polarization points, the more likely that such a situation may occur. We have illustrated an example in Figure 11 in which there are three polarization points, but there are well defined clusters in the vicinity of only two points. In general, if high dimensional data shows projected clusters of the kind illustrated in Figure 1, then it is expected that most views will reveal only one or two of the clusters effectively as illustrated in Figure 9. This tends to suggest that it may be valuable to use only a small number of polarization points. We emphasize that the main purpose of randomly sampling the data for polarization points is to discover projected clusters (if any) which contain these points. If enough number of points are sampled, it is expected that the user would have the opportunity to visualize the most prominent projected clusters in the data. Furthermore, even if some clearly separated clusters have low density in all possible projections, the corresponding subspaces would still be discovered when a polarization point from the cluster is sampled.

In order to actually find the projection subspace we use an iterative process in which we start off with the universal d-dimensional subspace $\mathcal{U}$. In each iteration, we maintain a current subspace $\mathcal{E}_c$ of dimensionality l_c which is gradually reduced to a 2-dimensional projection. In each iteration, we find the set of data points $\mathcal{M}_i$ which is closest to the polarization anchor y_i based on the distance measurements in the subspace $\mathcal{E}_c$. We assume that the centroid of $\mathcal{M}_i$ is z_i. The number of data points in each subset $\mathcal{M}_i$ is determined by the user-defined threshold s. In turn, the data points in $\cup_{i=1}^{k}\mathcal{M}_i$ are used to determine the subspace in which these points form well distinguished clusters around the polarization points. This can be done only be finding the subspace in which the momentum of these sets about their centroids is minimal. Some principal component analysis

techniques exist [17] which can find the optimum subspace in which the momentum of a *single* set of points about their centroid is minimal. However, here we have k sets $\mathcal{M}_1 \ldots \mathcal{M}_k$, and we need to minimize $\sum_{q=1}^{k} \mathcal{R}(\mathcal{M}_q, \mathcal{E}, z_i)$ over all subspaces $\mathcal{E}$. Therefore, we will use a simple transformation trick in order to use the method in [17].

Let $\mathcal{O}(z_i)$ be the set of points obtained by subtracting z_i from each point in $\mathcal{M}_i$. Thus, each of the sets $\mathcal{O}(z_i)$ is now centered at the origin $\overline{0}$. Since this transformation is a simple translation, the covariance and momentum about the centroid is preserved by the transformation. Therefore, we can show that $\mathcal{R}(\mathcal{M}_i, \mathcal{E}, z_i) = \mathcal{R}(\mathcal{O}_i(z_i), \mathcal{E}, \overline{0})$. But we also know that $\sum_{q=1}^{k} \mathcal{R}(\mathcal{O}_i(z_i), \mathcal{E}, \overline{0}) = \mathcal{R}(\cup_{i=1}^{k} \mathcal{O}_i(z_i), \mathcal{E}, \overline{0})$. Now all we need to do is to use the method of [17] in order to find the subspace in which the momentum of the single set of points $\cup_{i=1}^{k} \mathcal{O}_i(z_i)$ about their centroid is minimal.

In order to do, we first determine the covariance matrix Δ of the data points in $\cup_{i=1}^{k} \mathcal{O}_i(z_i)$. Specifically, the covariance matrix is a $d * d$ matrix, in which the entry (i, j) is equal to the covariance between the dimensions i and j of the points in $\cup_{i=1}^{k} \mathcal{O}_i(z_i)$. This matrix is positive semi-definite and can be diagonalized in order to determine a set of eigenvectors which will be orthogonal to one another. It has been shown in [17] that the eigenvectors corresponding to the smallest eigenvalues define the subspace with the least momentum about the centroid. Therefore, the subspace $\mathcal{E}_c$ corresponds to the eigenvectors for the l_c smallest eigenvalues. Correspondingly, this means that the data points in $\mathcal{M}_i$ are likely to be clusters in the projection $\mathcal{E}_c$.

The process of determining the data points $\mathcal{N}$ and the subspace $\mathcal{E}_c$ of dimensionality l_c is continued iteratively while reducing the value of l_c by a factor of 2 in each iteration, until the final 2-dimensional projection $\mathcal{E}_c$ is determined. We note that the data set $\mathcal{N}$ and the subspace $\mathcal{E}_c$ are alternately determined from one another in this iterative process. The motivation behind this iterative approach is the gradual refinement of the projection subspace and corresponding cluster, so that the sparse subspaces are gradually eliminated. As our empirical section will show, this results in well distinguished clusters, many of which are centered at the polarization points. We have so far discussed the case when the data is visualized in arbitrary projections of the data. In some cases, it may also be desirable to find the clusters in combinations of the original set of attributes in the interests of interpretability. In such cases, we simply replace the eigenvectors with the original axis-directions in all the above iterative calculations for choosing the directions of greatest polarization.

2.2 User Interaction for Creation of Different Clusters

The algorithm for interactive cluster separation is described in Figure 5. In order to facilitate user-interaction, effective ways must be provided to understand and visualize the clusters in each projection. A convenient way of doing so is to provide the user with an idea of which regions of the data are dense or sparsely populated in a given projection. Density based methods [10, 22] are quite useful in order to determine the variations in data behavior. To this effect, we use the method of kernel density estimation [20]. In this technique, in order to calculate the density estimate at x, we sum up the *kernel function values* for each data point x_i:

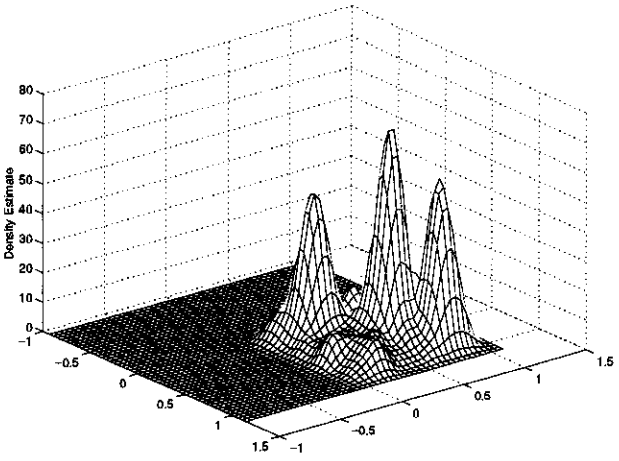

Figure 13: A plot of the density profile (Well Polarized Projection)

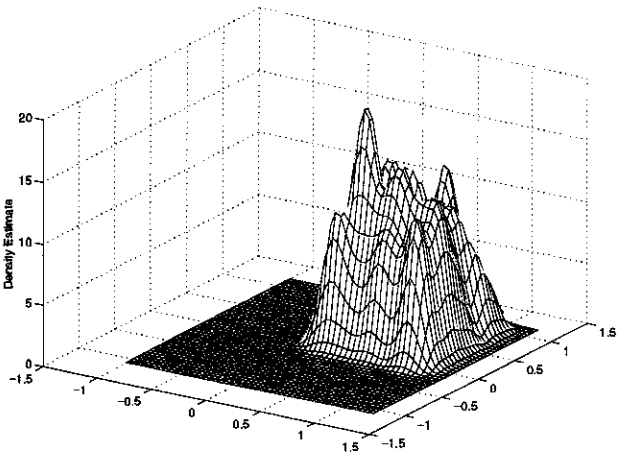

Figure 14: Density Profile (Poorly Polarized Projection)

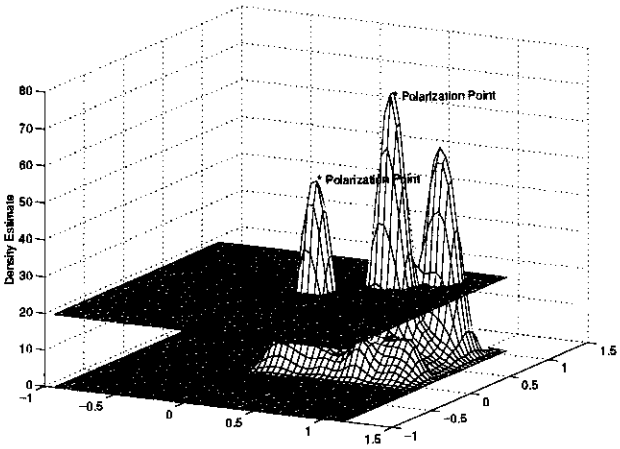

Figure 15: Density Based Cluster Separation (Two Clusters with $\eta = 20$)

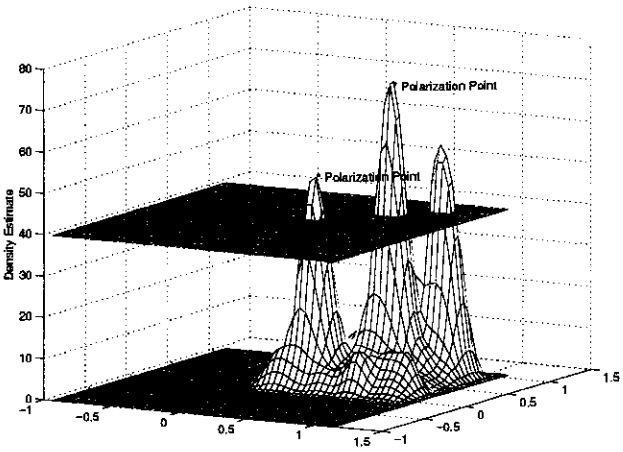

Figure 16: Density Based Cluster Separation (Three Clusters with $\eta = 40$)

Definition 2.1 *Given the data points $x_1 \ldots x_N$, the kernel density estimate at x is given by:*

$$\overline{f(x)} = 1/N \cdot \sum_{i=1}^{N} K_h(x - x_i) \tag{2}$$

Here K_h is a smooth, unimodal density function which is dependent on the smoothing parameter h.

A widely used value of the kernel is the gaussian function:

$$K_h(x - x_i) = (1/\sqrt{2\pi} \cdot h) \cdot e^{-(x-x_i)^2/(2h^2)} \tag{3}$$

For N data points with variance σ^2, the smoothing parameter h is chosen to be $1.06 \cdot \sigma \cdot N^{-1/5}$ in accordance with the Silverman approximation rule [20].

Since the density at every point in the continuous space cannot be calculated, we pick a set of $p * p$ grid-points at which the density of the data is estimated. The density values at these grid points are used in order to create surface plot of the data density. An example of such a surface plot is illustrated in Figure 13. Since clusters correspond to dense regions in the data, they are represented by peaks in the density profile. Similarly, the regions which separate out the different clusters have low density values and are represented by valleys in the density profile.

In order to actually separate out the clusters, the user can visually specify density value thresholds which correspond to noise level at which clusters can be separated from one another. Specifically, a cluster may be defined to be a connected region in the space with density above a certain noise threshold η, which is specified by the user. In order to provide the visual perspective of this separation, a hyperplane at a density value of η can be superposed on the density profile. We shall refer to this as the *density separator plane*. The intersection between the separator plane and the density profile creates a number of connected regions at which the density is above the specified noise threshold. The contours of the intersections between the separator planes and the density profiles are also the contours of the clusters in the data. All the data points which lie within a contour correspond to the same cluster in a given projection. For example, in Figure 15, we have illustrated a case in which by specifying the noise threshold $\eta = 20$, we find two clusters above the noise threshold. Note that the resulting clusters may be of arbitrary shape. Furthermore, by specifying different values of the noise threshold η one may have different number, sizes and shapes of clusters. For example, in Figure 15, there are two clusters at the specified noise threshold of 20, whereas in Figure 16, at the higher noise threshold of 40, there are three clusters. If we increased the noise threshold further, then one or more of the clusters may not be revealed at all. We note that both Figures 15 and 16 are reasonable separations of the data into clusters of different levels of granularity. This is another interesting aspect of the cluster creation process in which it is often difficult to settle on the use of single density, since clusters in different localities will have different densities. In order to handle this, we allow the user the flexibility to specify multiple values of η in a single projection. The smaller values of the density will reveal even the low density clusters in the data but will not reveal the finer separations among different clusters. The larger values of the density will reveal the fine grained separations into different clusters, but will not reveal the low density clusters. We assume that the final set of densities picked by the user is denoted by

$\mathcal{K} = \kappa_1 \ldots \kappa_r$. In some projections, the data may not be amenable to clustering. An example of such a projection is illustrated in Figure 14. In such cases, the user may choose not to specify any value of the noise threshold η, and the set $\mathcal{K}$ containing the noise threshold is null. This will not happen too often if the subspace determination procedure of the previous section is effective in finding well polarized projections. We note that the number of different separations r may depend upon the nature of a given projection and a user's understanding of the data behavior. Such intuition cannot be matched by any fully automated system effectively; this is an example of the criticality of the user in the cooperative process of high dimensional clustering. We note that in the Figures 15 and 16, the polarization points occur at the peaks of local optima in the density profiles. It is typical that polarization points occur at high elevations in the density profile, because the subspace determination algorithm actively tries to finds a view in which the sampled polarization points occur in the interior of some cluster. We note that the specification of the noise threshold η need not be done directly by value; rather the density separator hyperplane can be visually superposed on the density profile with the help of a mouse.

2.3 Updating the Identity Strings of the Different Data Points

The procedure for updating the *IdStrings* is illustrated in Figure 6. The behavior of the user in a given projection is used in order to update the set of *IdStrings* $\mathcal{I}$. The first step is the determination of the contours in the different clusters. Since the data densities have been calculated using a set of $p * p$ grid points, we can use them in order to obtain an approximation of the cluster contours. The first step is to find all of the elementary grid squares which lie approximately within a cluster contour. A grid square is assumed to approximately lie within a cluster contour, if at least three of its corners have a density value above the user-specified noise threshold κ. We refer to such grid squares as *dense*. Once we have determined all the dense grid squares, we need to consolidate them into connected components. A variety of graph search algorithms are available for this purpose. In order to use these techniques we assume that each grid square is a node, and they are connected by an edge if they share a common side. Now, the simple breadth first search technique discussed in [5] may be used in order to find all the connected components. For each of the clusters, we determine the data points which lie in the corresponding grid squares. These are the clusters which the user has separated out in that particular projection.

After all the clusters have been found, as assign each of them an Id number. (The Ids of the clusters are numbered from 1 through the total number of clusters which have been determined in that particular projection.) If a data point lies inside a cluster, then its identity is equal to that of the corresponding cluster, otherwise it is "*". We concatenate the identity of each data point to the corresponding *IdString*. We note that for each different noise threshold specified by the user in any projection, we concatenate exactly one element to the end of each *IdString*. Therefore, at the end of the process, the length of each string is equal to the total number of acceptable cluster separations specified by the user.

Once the interactive phase has been terminated by multiple iterations of subspace determination and user interaction, these *IdStrings* are used to create the final set of clusters.

2.4 Final Creation of Clusters

The final step of creation of the clusters is denoted by *FinalClusterCreation*, and is described in Figure 7. We note that even though a set of points may be perceived as a cluster by the user in a given projection, they may be separated out into different clusters in a different projection. Therefore, in order to find the final set of clusters we would like to isolate sets of points which have been classified by the user into the same projection in as many views as possible. At the same time, we would like to guarantee that a cluster contains at least the minimum fractional support s. Since the user characterization of clusters is captured with the set of identity strings in $\mathcal{I}$, we need to find subpatterns in these strings which occur throughout the data.

Consider an *IdString* I_r and pattern S of the same length l. A pattern S is a subpattern of I_r if and only if for each position $i \in \{1, \ldots l\}$ in S which is not *, the ith position in I_r also has the same value. Thus, the string *2*5*** is a subpattern of the string *2*5*4*, but it is not a subpattern of the string *2*3*4*. The *support* of the pattern S is equal to the percentage of the *IdStrings* in $\mathcal{I}$, for which the string S is a subpattern. The larger the number of fixed positions in S, the smaller the support of the string. We note that the minimum support s provided by the user is a measure of the minimum number of data points which a cluster must contain for it to be considered useful. Therefore we find all the *maximal* subpatterns which have support greater than the user defined threshold s. We also refer to such subpatterns as *itemsets*. Methods for finding such itemsets have been proposed[2] in [4]. Let us denote the final patterns found by $\mathcal{Q}_1 \ldots \mathcal{Q}_K$. Note that each of these patterns $\mathcal{Q}_i$ can be mapped onto all the data points $\mathcal{C}_i^F$ whose *IdStrings* are supersets of these subpatterns. We shall refer to the subpattern Q_i for cluster $\mathcal{C}_i^F$ as the *cluster template*. A position value m' on this template Q_i which is not * (a fixed position) corresponds to a projection in which the points of $\mathcal{C}_i^F$ belong to cluster id m' separated out by the user in that view. We note that when the projections are chosen from the original set of dimensions, it is possible to interpret the clusters using the cluster template. This is because the cluster template Q_i of the cluster $\mathcal{C}_i^F$ provides the different combinations of dimensions in which the user always classified all of these points to belong to the same cluster. This provides an intuitive interpretation of how the final clusters relate both to the attributes in the data and the history of user interaction. The subspace in which the set of points $\mathcal{C}_i^F$ is a cluster corresponds to the union of the all the 2-dimensional subspaces for fixed positions in Q_i. It now remains to discuss how the meaningfulness of each of these clusters is quantified.

2.5 Quantification of Meaningfulness by User Responses

Since the final clusters are created by determination of the sets of points which occur together as clusters in multiple projections, it is useful to evaluate the consistency of the user behavior across these different views in order to evaluate meaningfulness. In order to do so, we calculate the *interest ratios* of the patterns which define the clusters. The interest ratio of a pattern is the ratio of its actual support to the expected support based on the assumption of statistical independence. Let $S = m_1 \ldots m_l$ be a given cluster template, created by the l iterations. Let θ be the fraction of database points supported by S. Let β_i be the fractional support of the number of points

[2]Note that the method in [4] is for binary market basket data. The above string problem can be transformed to the binary market basket problem by defining an item for each position-value pair.

Data Set (Dimensionality)	Polarization Points	Views	Avg. Interest Ratio	Cluster Class Purity (IPCLUS)	Cluster Class Purity (ORCLUS)
Ionosphere (34)	1	22	71.4	86.3%	73.1%
	2	24	79.1	85.7%	
	3	29	64.3	82.3%	
Segmentation(19)	1	21	63.7	75.3%	51.2%
	2	20	66.9	77.8%	
	3	27	56.7	70.1%	

Table 1: Statistics of Cooperative Clustering for some Real Data Sets

Data Set	IPCLUS (Cluster Class Purity)	IPCLUS^A (Cluster Class Purity)	ORCLUS (Cluster Class Purity)
Ionosphere	86.3%	81.7%	73.1%
Segmentation	75.3%	63.8%	51.2%

Table 2: Axis-Parallel versus Arbitrary Projections

corresponding to m_i. (Specifically, β_i is the fraction of points that belong to cluster Id m_i for the visual projection i. When m_i is "*", then the value of β_i is 1.) Then the interest ratio $IR(S)$ of the cluster template S is the ratio of the support of template S to its support assuming statistical independence.

$$IR(S) = \theta/(\beta_1 \cdot \beta_2 \cdot \beta_3 \ldots \beta_l) \tag{4}$$

When the user behavior does not show any meaningful correlation across the different projections, the interest ratio for the clusters discovered will be close to one. An interest ratio larger than 1 is indicative of a cluster which reveals significantly greater affinity among the different data points based on the user behavior. This effectively means that the set of points occur together in one cluster based on the user observations in a larger number of projections than can be justified by random or statistically independent behavior across the different views. The overall procedure for quantification of meaningfulness is illustrated in Figure 8. Thus, the IPCLUS algorithm finally returns all the clusters, their templates (which provide interpretability) and the interest ratios (which quantify meaningfulness).

3 Empirical Results

In this section, we will illustrate the behavior of the IPCLUS algorithm with a variety of real data sets. All the data sets used in this paper were obtained from the UCI machine learning[3] repository. Our aim in this section is to provide a flavor of how the cooperative process for high dimensional clustering works both in terms of the kind of projections discovered and the final quality of clustering.

Unless otherwise mentioned, for each data set that we tested, support of a cluster was set at 5%. We ran the algorithm to termination, until the average coverage of each point in the data set was 10. The first data set on which we tested the algorithm was the ionosphere data set. This

[3]http://www.cs.uci.edu/~mlearn

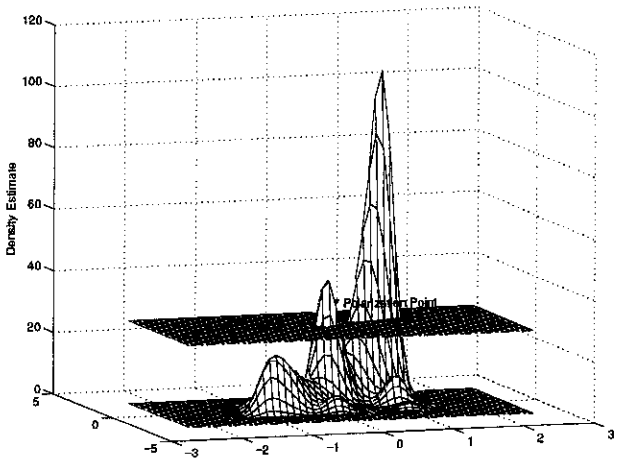

Figure 17: Ionosphere Data Set (Example Projection with Noise Threshold $\eta = 27$)

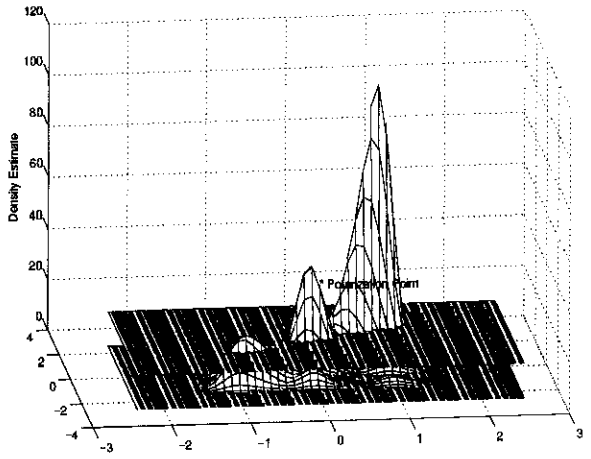

Figure 18: Ionosphere Data Set (Example Projection with Noise Threshold $\eta = 13.5$)

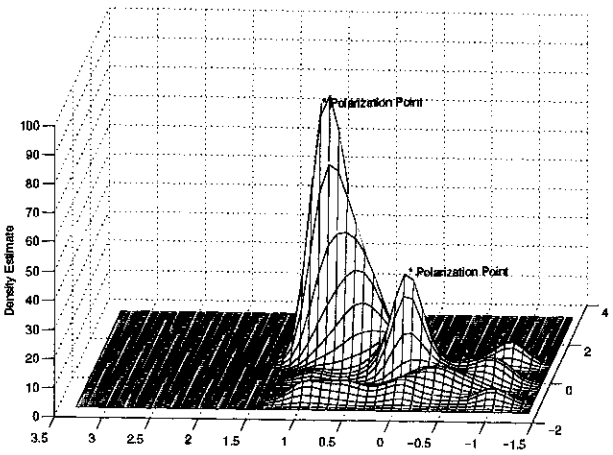

Figure 19: Example Projection (Ionosphere Data Set, two polarization points)

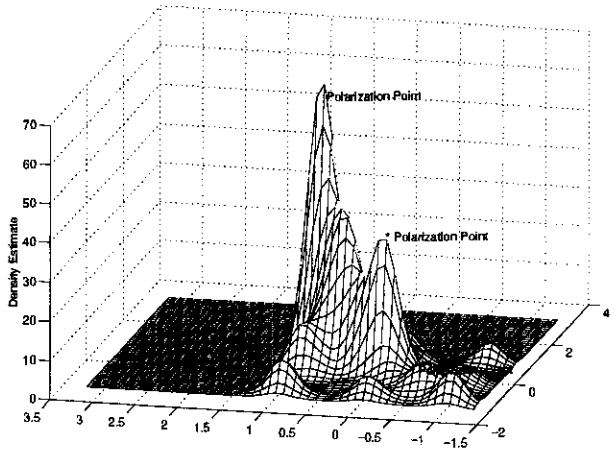

Figure 20: Example Projection (Ionosphere Data Set, three polarization points)

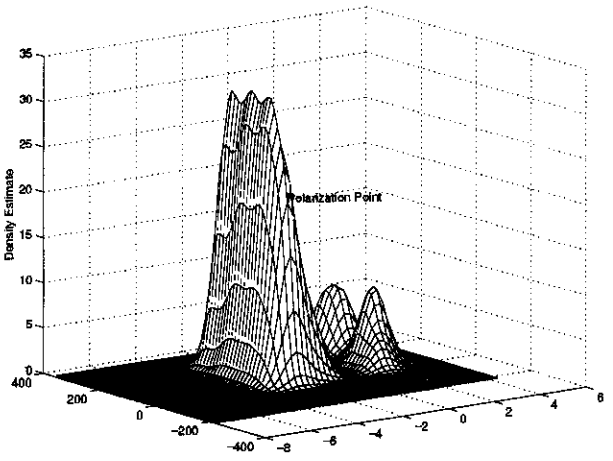

Figure 21: Example Projection (Segmentation Data Set)

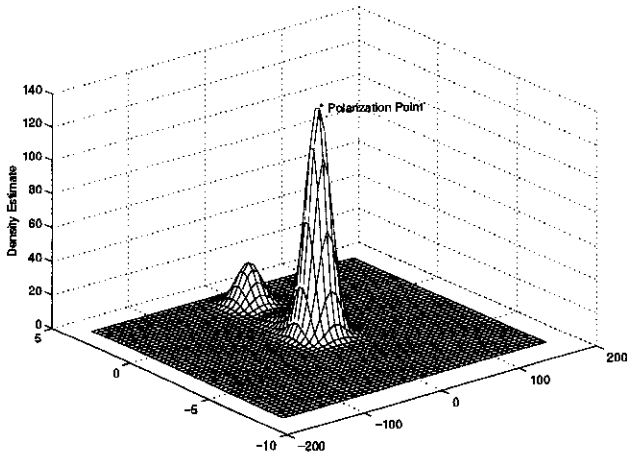

Figure 22: Example Projection (Segmentation Data Set)

data set contained 34 dimensions corresponding to radar returns from the atmosphere. The radar returns were classified as "good" or "bad" depending upon whether or not they showed some kind of structure in the ionosphere. We tested the algorithm by using one, two and three polarization points. We will first discuss the case when one polarization point was used. In order to achieve the termination criterion of an average coverage of 10, the user was presented about 22 different views of the data, since each view covered between $10 - 75\%$ of the data points. We found that the subspace determination procedure of our algorithm always determined projections in which the data was well polarized. An example of the density profile in such a projection is illustrated in Figure 17. A number of natural clusters are revealed in the data, one of which contains the polarization point. We have used two different noise thresholds ($\eta = 27$ and $\eta = 13.5$ - Figures 17 and 18) in order to illustrate the different kinds of clusters found. In the former case, two high density clusters are revealed, whereas one low density cluster is ignored. In the latter case, the low density cluster is found, but the two high density clusters are treated like a single cluster. Thus, in many projections, we were able to choose multiple separations of points into clusters each of which provided a different understanding of the clusters in the data. The total length of the id strings was 29, which is different from the number of views (22) since multiple density separators were used for some of the views. When the cluster templates were determined at the user defined support of 5%, the average interest ratio was found to 71.4, which is significantly greater than 1. In order to provide an intuitive understanding of the actual quality of the clusters found, we use a measure which we refer to as the *cluster purity*. Each record in the data sets had a class label was associated with it. The cluster purity was defined as the average percentage of the dominant class label in each cluster. Since the class label was not used in the interactive clustering process, this provides an intuitive idea of the final quality of the clusters. For the case of the ionosphere data set, the average cluster purity was found to 86.3%. As a baseline, we have also illustrated the cluster class purity from using the ORCLUS method [2], whose effectiveness is significantly lower. The summary results are indicated in Table 1.

We also tested the algorithm when more than one polarization point was used. Examples of sample projections on the ionosphere data set with two and three polarization points are illustrated in Figures 19 and 20 respectively. The interactive experience and final result for the case of two polarization points was similar to the case when one point was used. On the other hand, when three polarization points were used the algorithm did not work quite as effectively, and the quality degraded further with increase in the number of polarization points.

In the Table 1, we have also illustrated the results from the segmentation data set. Examples of some sample density profiles with the use of one polarization point are illustrated in Figures 21 and 22 respectively. Again we find that the use of one or two polarization points are more effective than the use of three polarization points. In fact our common experience with a wide variety of data sets was that the best results were obtained with one or two polarization points. This is because in these cases, the subspace determination subroutine is able to optimize the projection which reveals the cluster to which the polarization points belong. When there are many polarization points, the subspace determination procedure tries to find a projection in which all these points lie in a distinct

cluster. This may sometimes be difficult; consequently a projection is often chosen which does not provide the user a very clear view of the clusters in the data. For example, in Figure 20, even though there are three polarization points, only two of them lie in a distinct cluster. The lower quality of the projections with the use of more than two polarization points also explains the large number of views (see Table 1) that a user has to browse through to termination; many views either have to be discarded or do not contain a significant percentage of clustered data. With the use of one polarization point, at least one distinct cluster (containing the polarization point) is usually revealed; typically additional clusters may also be revealed. Thus, when a large enough number of views have been browsed through, most of the clusters have already been revealed from the data.

The results above have been presented for the case when arbitrary projections of the data were used in order to determine the polarization subspaces. We also tested the technique by using the axis-parallel version (denoted by IPCLUS^A) of the algorithm. The results are indicated in Table 2 for the use of one polarization point. It is clear that arbitrary projections are more effective than the axis-parallel version in obtaining higher cluster class purity. On the other hand, the axis-parallel version of the IPCLUS algorithm was better than the fully automated ORCLUS algorithm. This conclusively shows the advantage of a human-computer cooperative approach, since even the axis-parallel version of the IPCLUS algorithm performs better than the fully automated (arbitrary projection based) ORCLUS algorithm.

3.1 When meaningful clusters are absent from the data...

Since the focus of the method is to develop clusters which are well discriminated from one another in several combinations of dimensions, it is an interesting question as to how the algorithm behaves when the data itself is poorly behaved. The techniques of this paper use the fact that even though high dimensional data is sparse, there are often considerable correlations in lower dimensional projections which can be exploited in order to create clusters which are meaningful to a user. Thus, even though the overall implicit dimensionality of the data set is high, the *local* implicit dimensionality is low in the neighborhood of the polarization point. Our experiments with real data sets tend to show this behavior. Some data sets may not even satisfy this weak assumption. Such data sets correspond to those in which the local implicit dimensionality is high for each data point. An example of such a data set is the uniformly distributed data set. If the data is not amenable to clustering, then it is desirable for the IPCLUS algorithm to provide some indication of this fact.

We applied the IPCLUS algorithm on a uniformly distributed data set with $N = 5000$ data points. We used a support of $s = 5\%$ and one polarization point. An immediate indication of the poor behavior of the data is provided by the density profiles in which the clusters were not well discriminated from one another. An example of such a density profile in a projection with this data is illustrated in Figure 14. We note that even in the uniformly distributed case, the subspace determination algorithm was able to find a projection in which there is some level of discrimination because of the local variations in data density. Therefore, in a particularly well chosen projection it is still possible to find some clusters. However, the clusters found in a given projection have very little correlation with the clusters determined in a different projection. This will be detected

by the final phase of the algorithm in which we find cluster templates. The algorithm is usually unable to find a cluster template with as few as 2 fixed positions from different views which satisfy the user-defined support threshold. This corresponds to the fact that the user-behavior in two *distinct* views never classified any set of points (with the required support) into the same cluster. By lowering the support further, we were able to force the algorithm to find some cluster templates with the required support. Typically, when the support was gradually lowered, the algorithm suddenly found a very large number of cluster templates, all of which had an interest ratio which was very close to one. This was evidence of the fact that there are no meaningful clusters in this high dimensional data set. Thus, when the data itself is poorly behaved, the IPCLUS algorithm is able to effectively diagnose this and provide the user with a better understanding of the data.

4 Conclusions and Summary

High dimensional clustering is a very challenging problem because of the sparsity of the data. In this paper we discussed the problem of discovering meaningful clusters in high dimensional space with the utilization of a cooperative process between the human and the computer. The clusters are discovered in different subspaces with the use of polarization points and a visual profile of the data is provided to the user. This visual profile may be used in order separate out the different clusters with the use of human intervention. The attraction behind such an interactive process is that the problem of high dimensional clustering requires both the computational power of a computer and the intuition of a human; consequently a system which allocates the tasks according to the abilities of each entity is more effective than a fully automated process.

References

[1] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. S. Yu, J.-S. Park. Fast algorithms for projected clustering. *SIGMOD Conference* , 1999.

[2] C. C. Aggarwal, P. S. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces. *SIGMOD Conference* , 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *SIGMOD Conference* , 1998.

[4] R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *VLDB Conference Proceedings*, 1994.

[5] A. V. Aho, J. Hopcroft, J. D. Ullman. *Data Structures and Algorithms.* Addison-Wesley Publishing Company, 1987.

[6] M. Ankerst, C. Elsen, M. Ester, H.-P. Kriegel. Visual Classification: An Interactive Approach to Decision Tree Construction. *KDD Conference Proceedings*, 1999.

[7] M. Ankerst, M. Ester, H.-P. Kriegel. Towards an Effective Cooperation of the User and the Computer for Classification. *KDD Conference Proceedings*, 2000.

[8] K. Beyer, R. Ramakrishnan, U. Shaft, J. Goldstein. When is nearest neighbor meaningful? *ICDT Conference Proceedings*, 1999.

[9] K. Chakrabarti, S. Mehrotra. Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces. *VLDB Conference Proceedings*, 2000.

[10] M. Ester, H.-P. Kriegel, J. Sander, X. Xu. A Density Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *KDD Conference*, 1996.

[11] S. Guha, R. Rastogi, K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. *ACM SIGMOD Conference Proceedings*, 1998.

[12] J. Han, L. Lakshmanan, R. Ng. Constraint Based Multidimensional Data Mining. *IEEE Computer*, Vol. 32, no. 8, 1999, pp. 46-50.

[13] A. Hinneburg, D. A. Keim. Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering. *VLDB Conference Proceedings*, 1999.

[14] A. Hinneburg, C. C. Aggarwal, D. A. Keim. What is the nearest neighbor in high dimensional spaces? *VLDB Conference Proceedings*, 2000.

[15] A. Hinneburg, D. A. Keim, M. Wawryniuk. HD-Eye: Visual Mining of High Dimensional Data. *IEEE Computer Graphics and Applications*, 19(5), pages 22-31, 1999.

[16] A. Jain, R. Dubes. Algorithms for Clustering Data, *Prentice Hall,* New Jersey, 1998.

[17] I. T. Jolliffe. *Principal Component Analysis*, Springer-Verlag, New York, 1986.

[18] R. Ng, J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining, *VLDB Conference Proceedings*, 1994.

[19] S. Sarawagi. User-Adaptive Exploration of Multidimensional Data. *VLDB Conference Proceedings*, pp 307-316, 2000.

[20] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, 1986.

[21] A. Tung, J. Han, L. Lakshmanan, R. Ng. Constraint Based Clustering in Large Databases. *ICDT Conference Proceedings*, 2001.

[22] X. Xu, M. Ester, H.-P. Kriegel, J. Sander. A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases. *ICDE Conference Proceedings*, 1998.

[23] T. Zhang, R. Ramakrishnan, M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *ACM SIGMOD Conference Proceedings*, 1996.

[24] L. Yang. Interactive Exploration of Very Large Relational Databases through 3D dynamic projections. *ACM SIGKDD Conference Proceedings*, 2000.