A Human-Computer Cooperative System for Effective High Dimensional Clustering

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

ABSTRACT

High dimensional data has always been a challenge for clustering algorithms because of the inherent sparsity of the points. 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. Most clustering algorithms do not work efficiently in higher dimensional spaces because of the inherent sparsity of the data [1, 3, 4, 6]. Consequently, techniques for finding projected clusters in lower dimensional spaces have been discussed in [3, 4, 6]. Such clusters may exist either in projections of the original set of attributes or in arbitrary lower dimensional subspaces. We note that the formalization for finding clusters in projections of the original set of attributes provides greater interpretability [3, 6], whereas that of picking arbitrary projections is more flexible in discovering clusters created by local correlations [4].

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

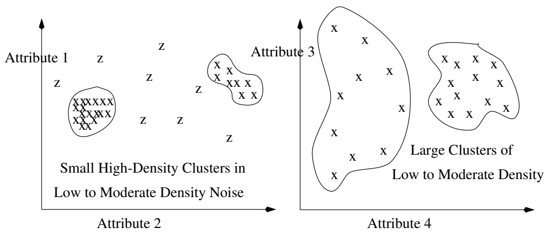

Figure 1: Variation in cluster shape, size and density across data localities and subspaces

of such cases in Figure 1. 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. 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 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. 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. We note an interesting technique discussed in [7], which describes a visual tool for users to interact with clusters in lower dimensional views of the data. However, our system actively finds views of the data in which projected clusters exist, whereas the technique in

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

Algorithm *IPCLUS(Data Set: $\mathcal{D}$, Min. 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 and is initialized to null }
while not(termination_criterion) **do**
begin
Randomly sample k points $y_1 \ldots y_k$ from $\mathcal{D}$;
$\mathcal{E} =$ *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;
$(\mathcal{C}_1^F \ldots \mathcal{C}_K^F, Q_1 \ldots Q_K) =$ *FinalClusterCreation($\mathcal{I}$, s)*;
$(IR_1, \ldots IR_K) =$ *EvalMeaningful($\mathcal{I}, \mathcal{C}_1^F \ldots \mathcal{C}_K^F$, $Q_1 \ldots Q_K$)*;
return$(IR_1 \ldots IR_K, \mathcal{C}_1^F \ldots \mathcal{C}_K^F, Q_1 \ldots Q_K)$;
end;

Figure 2: The IPCLUS Algorithm

Algorithm *ComputePolarizedProjection(Data Set: $\mathcal{D}$, Polarization Anchors: $y_1 \ldots y_k$, Min. 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 $\mathcal{M}_i$ denote the set of $N \cdot s$ points closest to polarization anchor y_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 compute $\mathcal{O}_i(z_i)$ by subtracting z_i from each point in $\mathcal{M}_i$;
Let c_{ij} be covariance of (i, j) for data $\cup_{q=1}^k \mathcal{O}_q(z_q)$;
{ Let Δ be the $d * d$ matrix $[c_{ij}]$ for $[i, j] \in [dXd]$; }
Determine eigenvectors $\mathcal{V} = \{\overline{e_1} \ldots \overline{e_d}\}$ of matrix Δ;
$\mathcal{E}_c =$ Set of l_c vectors from $\mathcal{V}$ with least eigenvalues;
$l_c = \max\{2, [l_c/2]\}$;
end
return$(\mathcal{E}_c)$;
end

Figure 3: Computing Polarized Projections

[7] is somewhat more passive in this respect by preselecting projections and then combining them over time along with user interaction. Since it becomes a more difficult problem to find good projections with increasing dimensionality, it is valuable to combine active subspace determination techniques with user interaction in order to reduce the number of possibilities during subspace exploration. We also provide techniques for quantification of meaningfulness and value of the clusters.

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 $\mathcal{E} = \{\overline{e_1} \ldots \overline{e_l}\}$ be a set of $l \leq d$ orthonormal vectors in this d-dimensional space. These orthonormal vectors define an l-dimensional subspace. 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 of the points $\overline{y_1}$ and $\overline{y_2}$ 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 $R(\mathcal{C}, \mathcal{E}, \overline{y}) = \sum_{i=1}^t \{Pdist(\overline{x_i}, \overline{y}, \mathcal{E})\}^2/t$.

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. In this paper, we will see that such subspaces are very helpful to the user in separating out the prominent clusters.

2. INTERACTIVE PROJECTED CLUSTERING

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 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 overall algorithm is illustrated in Figure 2. The interactive projected clustering algorithm 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 that 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 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

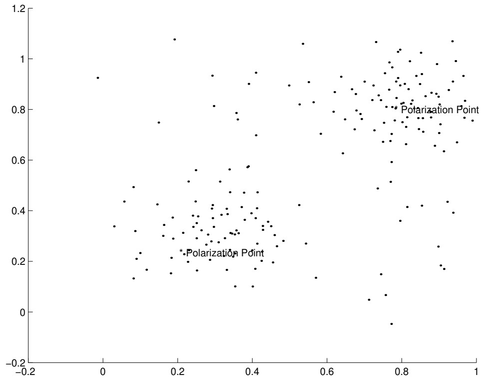

Figure 4: Determination of a well Polarized Projection (two polarization points)

some cluster. The algorithm is terminated when the average coverage of each data point is above a user-defined threshold.

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.

Thus, 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. (Procedure *ComputePolarizedProjection*)

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

(3) Storing the user interactions in the form of $IdStrings$ (Procedure *UpdateIdStrings*).

The procedures for final creation of the clusters and the quantification of meaningfulness from user responses. are respectively denoted by procedures *FinalClusterCreation* and *EvalMeaningful* in Figure 2. In this paper, we provide pseudocodes of only the IPCLUS algorithm and the process for creation of polarized subspaces. Word descriptions of the procedures *InteractiveClusterSeparation*, *UpdateIdStrings*, *FinalClusterCreation*, and *EvalMeaningful* are provided in subsections 2.2, 2.3, 2.4 and 2.5 respectively. Pseudocodes may be found in [2].

2.1 Determination of Polarization Subspaces

The procedure for determination of the polarization step is described in the *ComputePolarizedProjection* procedure of Figure 3. 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 $y_1 \dots y_k$, and are the polarization points around which we would like to find clusters. Since $y_1 \dots y_k$ 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 behavior. An example of such a projection is illustrated in Figure 4, in which the data is nicely separated out into different clusters using two 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. 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 this sense, polarization points serve a similar function to medoids in many clustering algorithms [3]. As in medoid based algorithms, it is expected that the user would sample at least one point from the prominent (projected) clusters in the data, if enough number of points are sampled for polarization.

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 by finding the subspace in which the momentum of these sets about their centroids is minimal. Some principal component analysis techniques exist [8] 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 $\mathcal{M}_1 \dots \mathcal{M}_k$, and we need to minimize $\sum_{i=1}^{k} \mathcal{R}(\mathcal{M}_i, \mathcal{E}, z_i)$ over all subspaces $\mathcal{E}$. Therefore, we will use a simple transformation trick in order to use the method in [8].

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 $\sum_{i=1}^{k} \mathcal{R}(\mathcal{O}_i(z_i), \mathcal{E}, \overline{0}) = \mathcal{R}(\cup_{i=1}^{k}\mathcal{O}_i(z_i), \mathcal{E}, \overline{0})$. Now all we need to do is to use the method of [8] 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 [8] 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

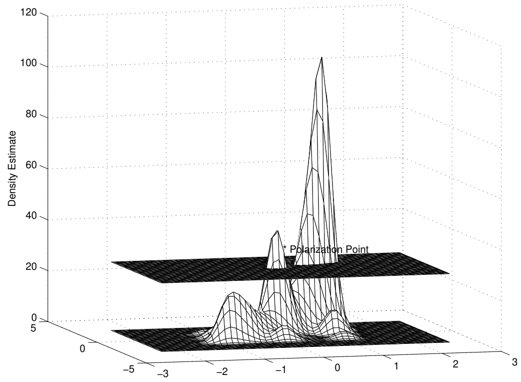

Figure 5: Density Based Cluster Separation (Ionosphere Data Set, $\eta = 27$)

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.

2.2 User Interaction for Cluster Creation

The procedure for interactive separation of clusters is denoted by *InteractiveClusterSeparation* in Figure 2. 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 [7] are quite useful in order to determine the variations in data behavior. 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. Given the data points $x_1 \ldots x_N$, the kernel density estimate at x is given by:

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

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:

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

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.

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 a surface plot. An example of such a plot is illustrated in Figure 5. 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 5, we have illustrated a case in which by specifying the noise threshold $\eta = 27$, 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 5, there are two clusters at the specified noise threshold of 27, whereas if we reduced the noise threshold to 15, there are three clusters. This is because in this case the cluster corresponding to the low density peak is also found. If we reduced the noise threshold further, then two of the clusters may get merged, and a larger portion of the low density cluster is found. Thus, different noise thresholds provide a division of the data into clusters of different levels of granularity. 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. In such cases, the user may choose not to specify any value of the noise threshold η, and the set $\mathcal{K}$ 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 Figure 5, the polarization point occurs at the peak of a local optimum 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 Identity Strings

The procedure for updating the *IdStrings* is denoted by *UpdateIdStrings* in Figure 2. The set of user reactions $\mathcal{K}$ in a given projection is used in order to update the *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_i. 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, a simple breadth first search technique 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, we 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 $\kappa_i \in \mathcal{K}$ 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* in Figure 2. 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 cluster 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 in [5]. 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 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

The procedure for evaluation of meaningfulness of clusters is denoted by *EvalMeaningful* in Figure 2. 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 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.

$$IR(S) = \theta / (\beta_1 \cdot \beta_2 \cdot \beta_3 \ldots \beta_l) \tag{3}$$

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

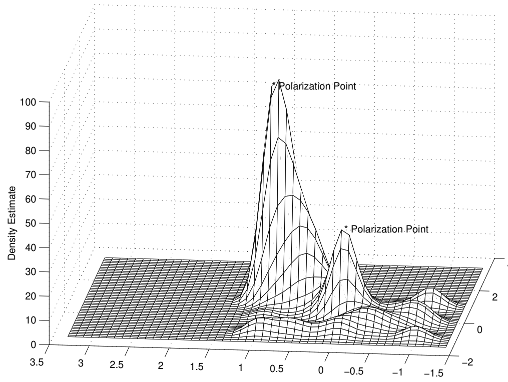

Figure 6: Example Projection (Ionosphere Data Set, two polarization points)

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. Thus, the IPCLUS algorithm finally returns all the clusters, their templates (which provide interpretability) and the interest ratios (which quantify meaningfulness).

3. EMPIRICAL RESULTS

The data sets for testing the IPCLUS algorithm were obtained from the UCI machine learning[2] 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 data set contained feature vectors for 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 will 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 5. A number of natural clusters are revealed in the data, one of which contains the polarization point. As discussed in an earlier section, it was possoble to isolate clusters of different granularity with different noise thresholds. 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 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 be 86.3%. We also tested the algorithm when more than one polarization point was used. Example of a sample projection on the ionosphere data set with two polarization points is illustrated in Figure 6. The interactive experience and final result for the case of two polarization points was similar to the case when one point was used. The algorithm was also tested on the segmentation data set from the UCI machine learning repository. The overall clustering experience was quite similar to the ionosphere data set. A detailed description of these empirical results and the effect of using different number of polarization points is provided in a long version of this paper [2].

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

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.

5. REFERENCES

[1] C. C. Aggarwal. Re-designing distance functions and distance based applications for high dimensional data. *ACM SIGMOD Record*, March 2001.

[2] C. C. Aggarwal. A Human-Computer Cooperative System for Effective High Dimensional Clustering, *IBM Research Report*, 2001.

[3] C. C. Aggarwal et al. Fast algorithms for projected clustering. *ACM SIGMOD Conference*, 1999.

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

[5] R. Srikant, R. Agrawal. Mining Quantitative Association Rules in Large Relational Tables. *ACM SIGMOD Conference*, 1996.

[6] R. Agrawal et al. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *ACM SIGMOD Conference*, 1998.

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

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