Towards Effective and Interpretable Data Mining by Visual Interaction

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

ABSTRACT

The primary aim of most data mining algorithms is to facilitate the discovery of concise and interpretable information from large amounts of data. However, many of the current formalizations of data mining algorithms have not quite reached this goal. One of the reasons for this is that the focus on using purely automated techniques has imposed several constraints on data mining algorithms. For example, any data mining problem such as clustering or association rules requires the specification of particular problem formulations, objective functions, and parameters. Such systems fail to take the user's needs into account very effectively. This makes it necessary to keep the user in the loop in a way which is both efficient and interpretable. One unique way of achieving this is by leveraging human visual perceptions on intermediate data mining results. Such a system combines the computational power of a computer and the intuitive abilities of a human to provide solutions which cannot be achieved by either. This paper will discuss a number of recent approaches to several data mining algorithms along these lines.

1. INTRODUCTION

Typical data mining tasks often involve massive amounts of computation on large data sets using particular formulations and models. In many cases, these formulations and models are not sufficiently sensitive to user needs. As a result, the final results may often differ considerably from the original intentions of the user. For example, in the case of the clustering problem, even the choice of the objective function and input parameters can have significant qualitative implications on the final results. It is not easy to create formalizations and make parametric choices which result in the most intuitive sets of clusters.

In recent years, the importance of incorporating human interaction into several data mining problems has been well understood and appreciated [3; 10; 16; 18; 23; 24; 28; 34]. Many tools have recently been proposed for interactive clustering and nearest neighbor search [3; 5; 15; 26; 27; 29; 34]. The importance of human interaction in data mining algorithms process arises from the ability of a user to make intuitive judgements that are outside the capabilities of fully automated systems. Simply speaking, a computer cannot match the visual insight, understanding and intuition of a human in distinguishing useful patterns in the data. On the other hand, a human needs computational support in order to determine effective summaries of the data which can be used to derive this intuition and understanding. Therefore, a natural strategy would be to devise a system which is centered around a human-computer interactive process. In such a system, the particular data mining task can be divided between the human and the computer in such a way that each entity performs the task that it is most well suited to. The active participation of the user has the additional advantage that he has a better understanding of the final results.

In this paper, we will illustrate the power of visual interactive data mining problems on the clustering and nearest neighbor problems. The advantages of visual interaction lie in their natural appeal to the intuitive abilities of the human. However, in most cases, it requires considerable amount of computational analysis to design the exact nature of the visual feedback which is most helpful to the user. This paper will also discuss some of these issues including interpretability and enhanced data understanding.

This paper is organized as follows. In the next section, we will discuss visual algorithms for clustering. In section 3, we will discuss interactive methods for the nearest neighbor problem. Extensions to other potential data mining problems are discussed in section 4. The conclusion and summary is discussed in section 5.

2. THE CLUSTERING ALGORITHM

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. Detailed surveys on clustering methods can be found in [20].

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 [12] 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 [9]. Most real data contains different kinds of skews in which some subsets of dimensions are related to one another. Furthermore, these subsets of correlated dimensions may vary with data locality [7; 13]. 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 [2; 7; 9]. 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 [9], whereas that of picking arbitrary projections is more flexible in discovering clusters created by inter-attribute correlations [7].

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.

We note one interesting technique in [18], 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 dimension-

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

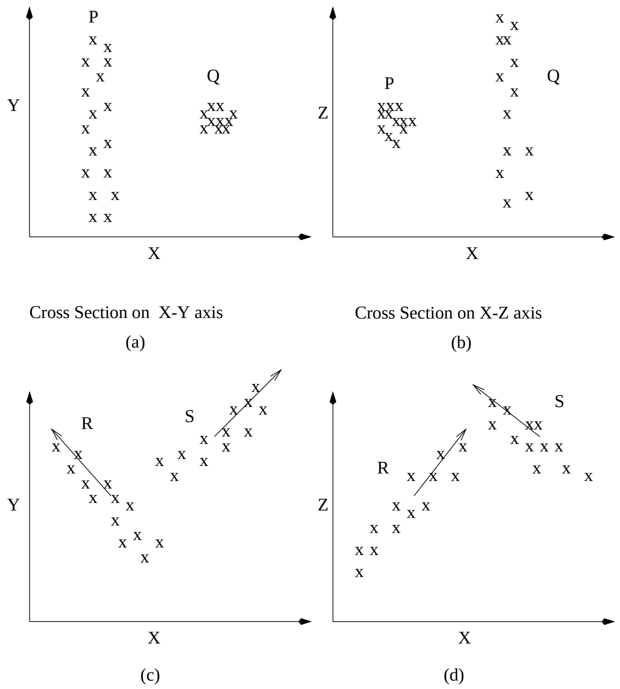


Figure 1: Illustrations of Lower Dimensional Clusters

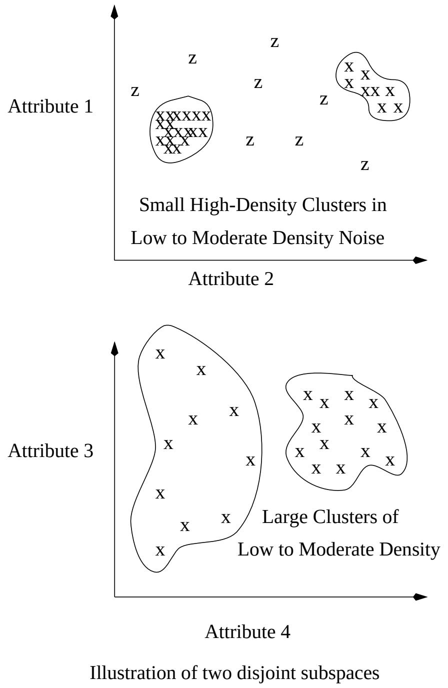


Figure 2: Variation in cluster shape, size and density across data localities and subspaces

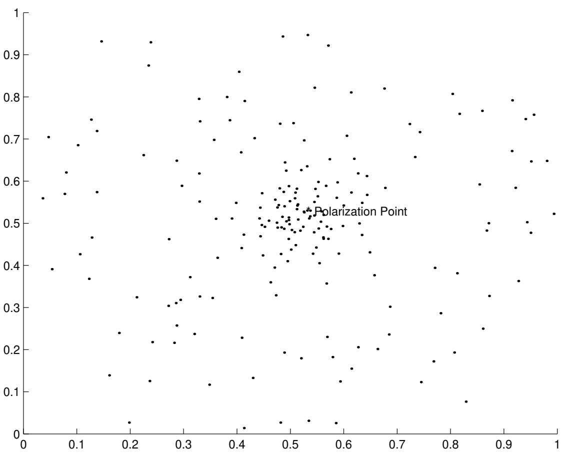


Figure 3: Determination of a well polarized projection (one polarization point)

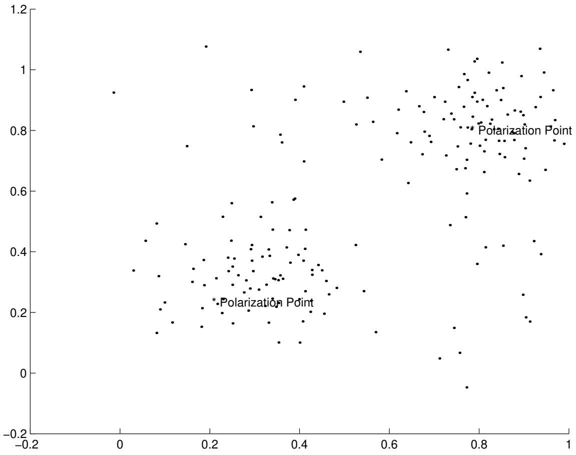


Figure 4: Determination of a well Polarized Projection (two polarization points)

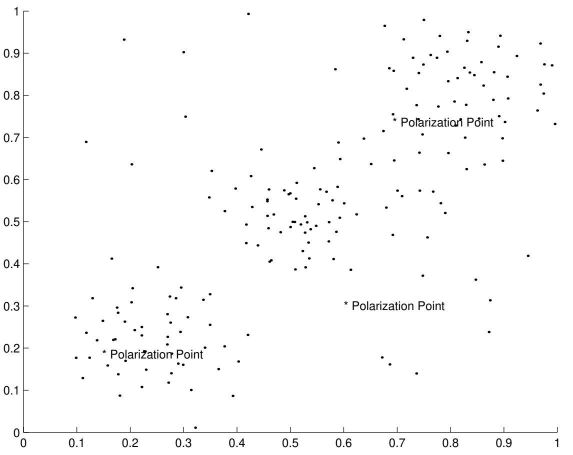


Figure 5: Partially Polarized Projection

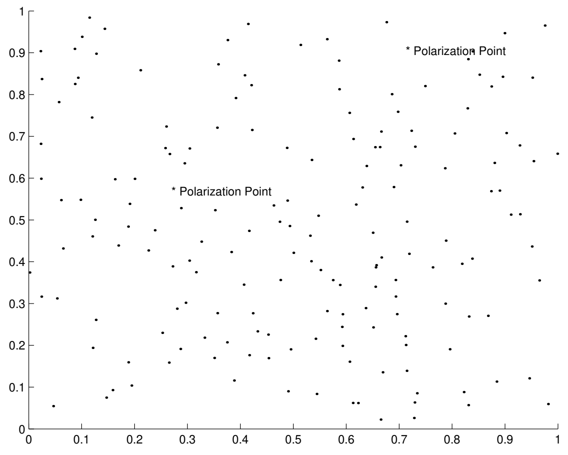


Figure 6: Poorly Polarized Projection

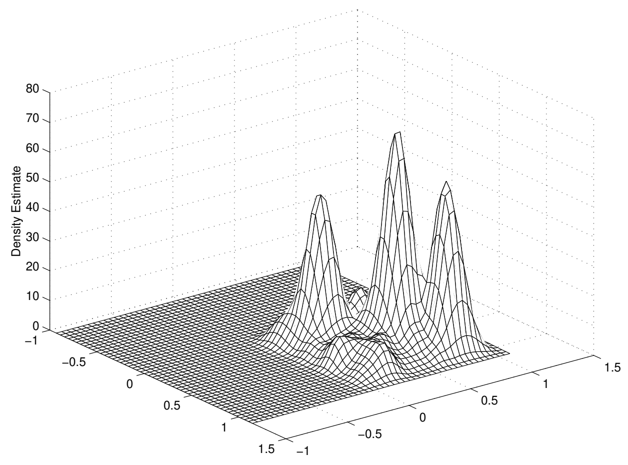


Figure 7: A plot of the density profile (Well Polarized Projection)

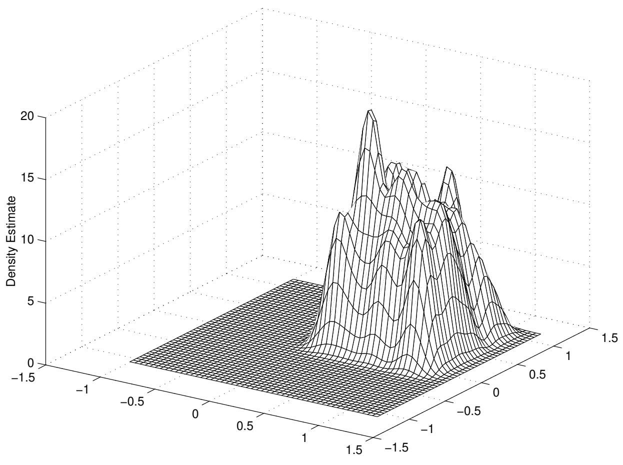


Figure 8: Density Profile (Poorly Polarized Projection)

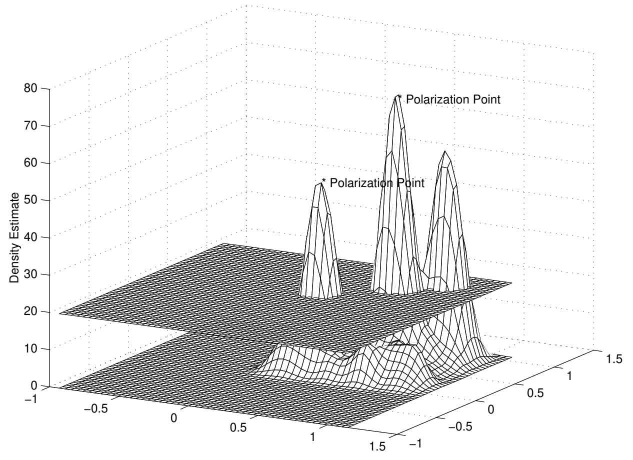


Figure 9: Density Based Cluster Separation (Two Clusters with $\eta = 20$)

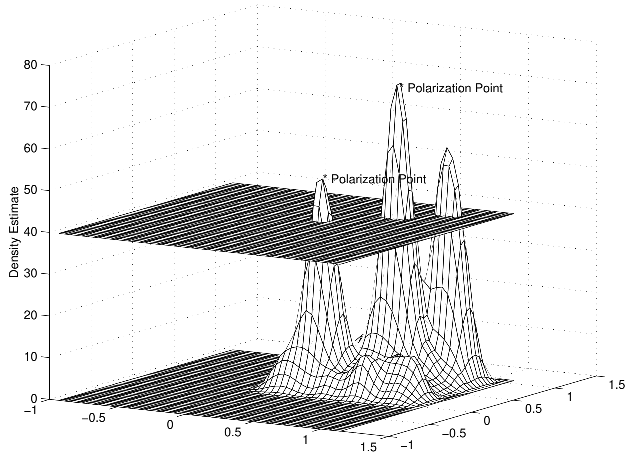


Figure 10: Density Based Cluster Separation (Three Clusters with $\eta = 40$)

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;
  $(\mathcal{C}_1^F \ldots \mathcal{C}_K^F, Q_1 \ldots Q_K)$ =*FinalClusterCreation(*$\mathcal{I}$*,* s*)*;
  $(IR_1, \ldots IR_K)$ =*EvalMeaningfulness(*$\mathcal{I}, \mathcal{C}_1^F \ldots \mathcal{C}_K^F, Q_1 \ldots Q_K$*)*;
  return$(IR_1 \ldots IR_K, \mathcal{C}_1^F \ldots \mathcal{C}_K^F, Q_1 \ldots Q_K)$;
end;

Figure 11: The IPCLUS Algorithm

ality. An interesting system developed recently is the IPCLUS system [3] which is able to leverage the computational power more effectively. 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.

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.

It is an interesting problem to find the projection subspace $\mathcal{E}$ in which the data shows this polarized behavior. In Figures 3, 4, 5 and 6, we have illustrated a number of different possibilities for the relationship between the data and chosen polarization points in different projections. In Figures 3 and 4 the data is nicely separated out into different clusters using one and two polarization points respectively. On the other hand, in Figure 6, 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 5 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 3. 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 each iteration, when the projection subspace has been found, kernel density estimation techniques [30] 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 $\mathcal{I} = (I_1 \dots I_N)$. 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.

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 11. 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. Thus, the basic algorithm includes 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 $IdStrings$;

The procedure for determination of the polarization step is denoted by the *ComputePolarizedProjection* procedure of Figure 11. Details of the method may be found in [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.

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. To this effect, we use the method of kernel density estimation [30]. 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:

Definition 2.1. *Given the data points $x_1 \dots x_N$, 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 [30].

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 7. 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 9, 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 9, there are two clusters at the specified noise threshold of 20, whereas in Figure 10, 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 9 and 10 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 8. 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 9 and 10, 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.

The behavior of the user in a given projection is used in order to update the set of *IdStrings* $\mathcal{I}$. 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 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. 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 [31]. 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.

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 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. VISUAL METHODS FOR HIGH DIMENSIONAL NEAREST NEIGHBOR SEARCH

[2]Note that the method in [31] 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.

The nearest neighbor search problem is defined as follows: For a given query point Q, find the data points which are closest to it based on a pre-defined distance function. Examples of application domains in which this problem arises are similarity search in geometric databases, multimedia databases, and data mining applications such as fraud detection and information retrieval. Typical domains such as data mining contain applications in which the dimensionality of the representation is very high. For example, a typical supermarket application will contain hundreds of dimensions. Consequently a wide variety of access methods and data structures have been proposed for high dimensional nearest neighbor search [11; 13; 21; 25; 33].

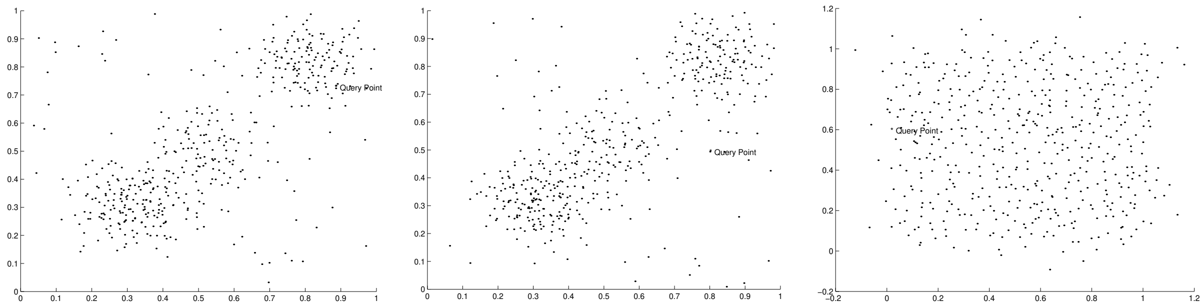
It has been questioned in recent theoretical work [12] as to whether the nearest neighbor problem is meaningful for the high dimensional case. These results have characterized the data distributions and distance functions for which all pairs of points are almost equidistant from one another in high dimensional space and have illustrated the validity of the results on a number of real work loads. We note that these results do not necessarily claim that nearest neighbor is not meaningful in every high dimensional case, but that one must be careful in interpreting the significance of the results. For example, a lack of *contrast* in the distribution of distances implies that a slight relative perturbation of the query point away from the nearest neighbor could change it into farthest neighbor and vice versa. In such cases, a nearest neighbor query is said to be *unstable*. Furthermore, the use of different distance metrics can result in widely varying ordering of distances of points from the target for a given query. This leads to questions on whether a user should consider such results meaningful.

Recent work [17] has shown that by finding discriminatory projections in the neighborhood of a query point, it is possible to improve the quality of the nearest neighbors. This approach uses the fact that even though high dimensional data is sparse in full dimensionality, certain projections of the space may contain meaningful patterns. These meaningful patterns are more closely related to the query point than the ones determined by using the full dimensionality. Related techniques [6; 8] design distance functions in a data driven manner in order to find the most meaningful nearest neighbors. In these techniques, the statistical properties of high dimensional feature vectors are used in order to obtain meaningful measures of the distances between the points. This is often a difficult task, since the most effective method of measuring distances may vary considerably with the data set and application domain.

Since the issue of meaningfulness is connected to the instability in measurement of distances, a natural guiding principle in these methods is to find data projections and distance functions in which the distances of the nearest neighbors from the query point have high contrast from the rest of the data. It has been shown in [17] that such a strategy leads to improvement in search quality. It has also been independently confirmed for the multimedia domain that *distinctiveness sensitive* nearest neighbor search [22] leads to higher quality of retrieval. At the same time, it is quite difficult for a fully automated system to always find nearest neighbors which would be considered valuable and meaningful by the user. Furthermore, even if the neighbors found are valuable, a user would have little idea about the quality of the neighbors found without being actively involved in the process. The fully automated systems discussed in [8; 17] are incomplete in their characterization of the data in terms of a single best projection or distance function. Different projections can provide different views of the data, all of which may be informative to a human in understanding the relationship between the query point and the rest of the data.

In light of these issues, it is especially advantageous to develop a human-computer interactive system for high-dimensional nearest neighbor search. In this method, the distribution of the data in carefully chosen projections are presented visually to the user in order to repeatedly elicit his preferences about the relationships between the data patterns and the query point. Specifically, these projections are chosen in such a way that the natural data patterns containing the query point can be visually distinguished easily. It is evident from our earlier discussion about Figure 1 that even though it is difficult to define clusters for sparse high dimensional data, it is often possible to find clusters in certain lower dimensional projections. Many of these clusters may contain the query point. We refer to such projections as *query centered projections* and the corresponding clusters as *query clusters*. These projections may exist either in the original sets of dimensions or in an arbitrarily oriented subspace of the data. For each such projection determined, the user is provided with the ability to visually separate out the data patterns which are most closely related to the query point. In a given view, a user may choose to pick some or no points depending upon the nature and distribution of the data. At the end of the process, these reactions are utilized in order to determine and quantify the meaningfulness of the nearest neighbors found from the *user perspective*.

In addition, the sparsity of the data creates interpretability problems for the meaningfulness of the nearest neighbor problem. However, it has also been shown in recent work [17] that even though meaningful contrasts cannot be determined in the full dimensional space, it is often possible to find lower dimensional projections of the data in which tight clusters of points exist. In this spirit, the projected nearest neighbor technique [17] finds a single optimal projection of the data in which the closest neighbors are determined in an automated way using the euclidean metric. In reality, no single projection provides a complete idea of the distribution of the data in the locality of the query point. The full picture may be obtained by using multiple projections, each of which provides the user with different insights about the distribution of the points. For example, Figure 12(a) illustrates a projection in which there is a cluster near the query point which is well separated from the other data points. Such a projection is very useful, since it provides a distinct set of points which can be properly distinguished from the rest of the data. Figures 12(b) and 12(c) are examples of projections in which the closest records to the query point are not well distinguished from the rest of the data. For example, in the case of Figure 12(b), the query point is located in a region which is sparsely populated; this is not a good query centered projection, since one cannot identify a distinct subset of points in proximity to the query. In the case of Figure 12(c), the projection is a poor one irrespective of the nature of the query point, since the points are uniformly distributed and do not separate out well into clusters. It is often a subjective issue to determine whether or not a given projection distinguishes the query cluster well. There may



(a) (b) (c)

Figure 12: (a) Good Query Centered Projection (b) Poor Query Centered Projection (Query Point in Sparse Region) (c) Noisy projection

also be many cases in which query points may be located at the fringes of a cluster, and it may be difficult to determine the query cluster in an automated way. In all such cases, it becomes important to use the visual insight and feedback of a user in diagnosing the query clusters.

We note that it is possible that a good query-centered projection may be difficult to find in a combination of the original set of attributes. In such cases, it may be necessary to determine arbitrary projections created by vectors which are not parallel to the original axis directions. On the other hand, the use of axis-parallel projections has the advantage of greater interpretability to the user. In this paper, we propose methods for determining projections of both kinds i.e. axis-parallel and arbitrary projections.

Since the system works by determining the distribution of the nearest records to the query points in a given projection, we introduce a parameter called the *support*. This is the number of database points that are candidates for the nearest neighbor in a given projection, and whose distribution relative to the rest of the data set is analyzed. The value of this support parameter can either be chosen by the user or the system. In most real applications, users are not looking for a single nearest neighbors, but a group of nearest neighbors all of which can be considered to be close matches for a target query. Therefore, the number of database points to be retrieved by the user is the *support* s used for the analysis. We note that in order to perform a proper analysis of the directions in the data of greatest discrimination, this support should at least be equal to the dimensionality d. Therefore, in cases when the user-specified support is less than d, we set it equal to d. We also note that in many cases, there may be a certain number of points which are inherently more closely related to the query as compared to the rest of the database. This number may be different from s, and cannot be known a-priori. In such cases, we will see that our approach is able to provide some guidance in returning the natural sets of points related to the query.

The system works in an iterative loop in which a set of $d/2$ mutually orthogonal projections are presented to the user in a given iteration. Each of these projections is carefully chosen such that it brings out the contrast between the points closest to the query and the remaining points. Once such a projection has been found, the user separates out the points which belong to the query cluster. The selection statistics of each data point are maintained in the set of counts $v(1) \dots v(N)$ which are initialized to zero, and incremented whenever a set of points is picked. After each iteration of $d/2$ projections, the set of choices made by the user $v(\cdot)$ are utilized determine his level of perception as to the level of closeness of each data point to the query point Q. This number lies in the range $(0, 1)$, and is referred to as the meaningfulness probability. This number defines the user-reaction probability that the data point can be distinguished as significantly more closely related to the query point as compared to the average record in the data. The meaningfulness probability is calculated independently for each iteration of $d/2$ projections, and the values over multiple iterations are aggregated in order to determine a final value. At the end of each iteration, those points are removed from the data set which were not picked even once in any projection. Thus, the user behavior in an iteration influences the later profiles which are presented to him. The process continues until it is determined that the current ordering of meaningfulness probabilities reliably matches user's intent based on his reaction to all views which have been presented to him so far. The details of the meaningfulness quantification and termination criterion are described a later section.

Each iteration (henceforth referred to as a major iteration) is divided into a set of $d/2$ minor iterations, in each of which a projection is determined and presented visually to the user for his feedback. The set of $d/2$ projections which are determined in each major iteration are mutually orthogonal. This is because we would like to present the user with several independent perspectives of the data, which together span the full dimensional space. In order to achieve this, we maintain a current data set $\mathcal{D}_c$, and a current subspace $\mathcal{E}_c$. Let us say that a total of $r < d/2$ projections have already been presented to the user in the current major iteration. Let the subspaces corresponding to these projections be denoted by $\mathcal{E}_{proj(1)} \dots \mathcal{E}_{proj(r)}$. Then, the current subspace $\mathcal{E}_c$ is the $d - 2 \cdot r$ dimensional subspace which is orthogonal to all of these projections, and is given by $\mathcal{U} - \cup_{i=1}^{r} \mathcal{E}_{proj(i)}$. Here $\mathcal{U}$ is the full dimensional space. The data set $\mathcal{D}_c$ is the projection of the data set $\mathcal{D}$ onto the subspace $\mathcal{E}_c$. Thus, each data point $x_i \in \mathcal{D}$ is represented by the data point $Proj(x_i, \mathcal{E}_c)$ in $\mathcal{D}_c$. The next projection $\mathcal{E}_{proj(r+1)}$ is determined by using the data set $\mathcal{D}_c$.

In each minor iteration, we perform the following steps:

(1) Finding the most discriminatory projection which is centered around the query point. This discriminatory projection is picked out of $\mathcal{E}_c$.

(2) Interactive determination of the query cluster by the

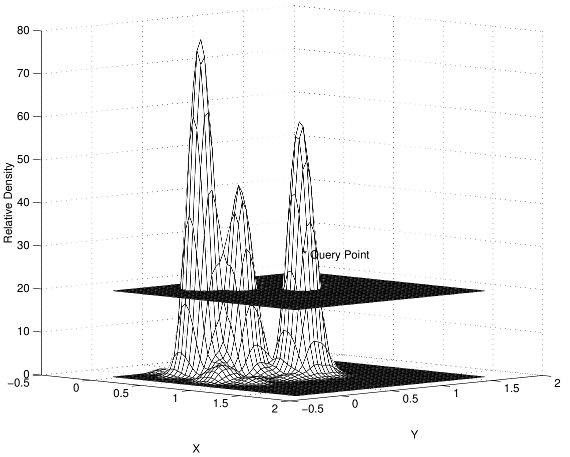


Figure 13: Query Cluster Separation with Noise Threshold $\phi = 20$

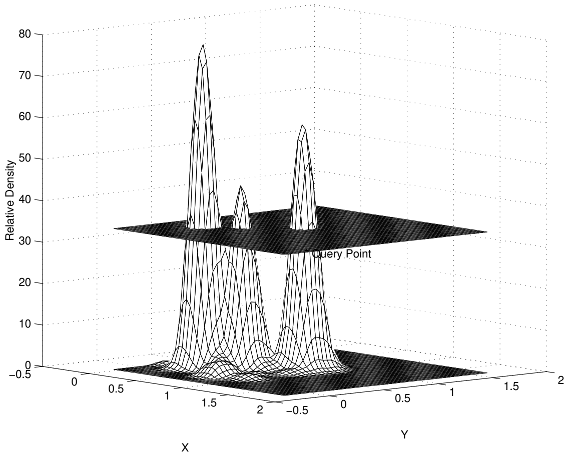


Figure 14: Query Cluster Separation with Noise Threshold $\phi = 34$ (Null Query Cluster)

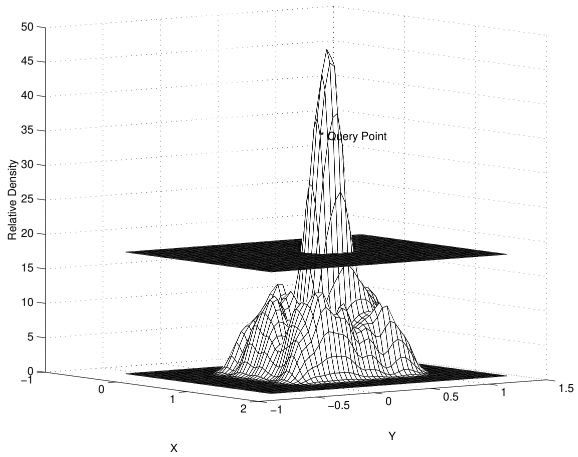


Figure 15: Query Cluster Separation (Second Case $\phi = 17.4$)

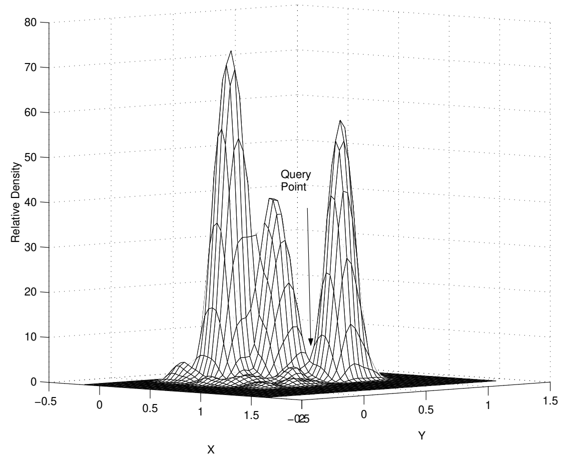


Figure 16: Density Profile for poor projection

user based on the visual separation of the query point from the remaining data.
(3) Updating the counts for the points in the query cluster.
In the remaining part of this section, we will discuss each of these steps in detail.
Once a discriminatory projection has been determined, then it is valuable to rely on user-interaction in order to separate the query cluster from the remaining data points. In order to achieve this objective, it is desirable to find ways of providing the user with a visual idea of the nature of the probabilistic distribution of the data. To this effect, we again use the kernel density estimation technique [30] as for the clustering problem. In order to actually construct the density profiles, we estimate the probability density of the data at a set of $p * p$ grid-points, which are used to create surface plots. Examples of such density profiles are illustrated in Figures 13, 14, 15 and 16. Note that there is a sharp and well separated peak containing the query point. This corresponds to the highly dense cluster near the query point. This behavior is typical of a well chosen projection which discriminates the data patterns near the query point well. A second way of providing the user with a visual understanding of the data[3] is to provide a *lateral* density plot, in which we have a scatter plot of fictitious points which are generated in proportion to their density. We note that all of Figures 12(a), 12(b), and 12(c) are lateral scatter plots of 500 points generated from synthetic data sets.
Once the user is provides with this visual profile then it is possible for him to separate the query cluster from the remaining points by using either of the two visual profiles. A convenient way of separating the query cluster visually is by using density separators of a certain height. In this technique the user specifies the density ϕ which is the noise threshold. This threshold is used in order to determine the set of points which are the user-defined nearest neighbors in that projection by using the concept of density connectivity. The concept of density connectivity [14; 18] is defined as follows:

DEFINITION 3.1. *A data point x is density connected to the query point Q at noise threshold ϕ, if there exists as path P from x to Q such that each point on P has density larger than the noise threshold ϕ.*

Thus, for a given noise threshold ϕ and query point Q, it is possible to uniquely determine the set of points in the database which are density connected to the query point. For example, in Figures 13 and 14, we have shown the density profile of a data set along with *density separator planes* for two different values of the noise threshold ϕ. We note that the contour of intersection of the density separator plane with the density profile of the data is a set of closed regions. Each such closed region corresponds to the contour of the cluster in the projection. However, only one of these contours is relevant; the one that contains the query point Q. All data points contained within this contour are relevant answers to the query point for this particular projection. We shall henceforth refer to the contour containing the query point Q at noise threshold ϕ as the (ϕ, Q)-contour.

[3]Note that a density scatter plot is significantly easier to comprehend than the scatter plot of the original data points which shows considerable overlap among the individual points. Also, the number of points in a lateral scatter plot can be chosen in order to provide the best visual profile.

Such a contour is not restricted to be of any particular shape, and is dependent only upon the distribution of the points in the data. For example, in Figure 13 there are two density connected regions above the noise threshold ϕ, whereas in the second case, there are three such regions, when a higher noise threshold is used. All the data points which lie in the same region as the query cluster are the set of preferences for that particular projection. In this particular case, the difference between two views at noise threshold $\phi = 34$ and $\phi = 20$ is that in the former case, the query point is not included in the contour of intersection of the peak containing the query point and the density separator hyperplane. Therefore, at that value of the noise threshold $\phi = 34$, there are no points in the query cluster. At a noise threshold of $\phi = 20$, a distinct cluster of points containing the query point are created; by reducing ϕ further, more and more points from the fringes of the cluster are included. Here, the intuition of a user is very useful, since an accurate delineation of the related data pattern is often not possible by fully automated methods. We also note that if the query point had belonged to one of the other two peaks, then for different values of the noise threshold, different number of peaks would have been included in the query cluster. By using $\phi = 0$, all points are included in the query cluster. We refer to such views created by this process as *density separated views*, since they clearly show the various clusters in the data based on the density profile and the noise threshold supplied by the user. We note that it is not necessary for the user to supply the noise threshold after just one view of the data. Rather, the user can look at density separated views for many different values of the noise threshold ϕ in order to interactively converge at the most intuitively appropriate value.
The user reactions over multiple iterations can then be aggregated into the final preference counts. These preference counts indicate the value and meaningfulness of the nearest neighbor from the user perspective.

4. EXTENSIONS TO OTHER DATA MINING PROBLEMS

The methodologies discussed in this paper can be extended to a number of data mining problems. Some important problems in which visual interaction can be leveraged effectively are as follows:
(1) Classification: High Dimensional Data is a difficult problem for decision tree construction because of the large number of combinations of dimensions which have discriminatory power. Therefore, it is difficult to effectively build a decision tree for high dimensional data without using a large number of nodes or complex split criteria. In an interactive application, a user may find it more valuable to develop a diagnostic decision support method which can reveal significant classification behavior of exemplar records. Such an approach has the additional advantage of being able to optimize the decision process for the individual record in order to design more effective classification methods. In the work discussed in [4], we propose a method which provides the user with the ability to interactively explore a small number of nodes of a hierarchical decision process so that the most significant classification characteristics for a given test instance are revealed. Therefore, this method combines the abilities of the human and the computer in creating an effec-

tive diagnostic tool for instance-centered high dimensional classification.

(2) Association Rules: The problem of association rules is an expecially important one for the case of online and interactive data mining. This is because in most cases, association rules are generated by batch processing algorithms result in large number of rules which cannot be assimilated easily for users. There is a greater need for techniques which provide better assimilation and interactive abilities in discovering rules of various types. One important technique proposed recently is discussed in [1] in which online capabilities for finding association rules are propsed. The system also provides the user the ability to determine rules containing particular items and constraints. To provide even better understanding visual techniques are needed. To this effect, the method of [19] proposes moasic plots for interactive visual association rule mining. We believe that this technique can be extended to further ease several limits of the data mining process:

• In many applications, it is difficult to provide support and confidence as hard numbers without having an intuitive idea of the nature of the underlying rules. On the other hand, the technique in [19] has the capacity of abstracting out such parameters from the data mining process.

• The technique in [19] can also be used in order to determine which items are the most valuable for the association rule discovery process. This lays the groundwork for effective exploratory mining of association rules.

5. CONCLUSIONS AND SUMMARY

In this paper, we discussed the merits of interactive visual approaches to a number of data mining problems. The primary aim of the paper is to show that interactive visual data mining is a technique which has powerful implications in leveraging the intuitive abilities of the human for data mining problems. This leads to solutions which can model data mining problems in a more intuitive and unrestricted way that methods which make use excessive use of messy formalizations and parameters. The additional advantage of such techniques is that the user also has much better understanding of the final quality of the solution.

6. REFERENCES

[1] C. C. Aggarwal, P. S. Yu. Online Generation of Association Rules. *ICDE Conference*, 1998.

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

[3] C. C. Aggarwal. A Human-Computer Cooperative System for Effective High Dimensional Clustering. *ACM KDD Conference*, 2001.

[4] C. C. Aggarwal. Towards Exploratory Instance Centered Classification of High Dimensional Data. *IBM Research Report*, 2002.

[5] C. C. Aggarwal. Towards Meaningful High Dimensional Nearest Neighbor Search by Human-Computer Interaction. *ICDE Conference*, 2002.

[6] C. C. Aggarwal, A. Hinneburg, D. A. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. *ICDT Conference Proceedings*, 2001.

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

[8] C. C. Aggarwal, P. S. Yu. The IGrid Index: Reversing the Dimensionality Curse for Similarity Indexing in High Dimensional Space. *ACM SIGKDD Conference Proceedings*, 2000.

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

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

[11] S. Berchtold, D. A. Keim, H.-P. Kriegel: The X-Tree: An Index Structure for High-Dimensional Data, *VLDB Conference Proceedings*, 1996.

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

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

[14] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu. Density-Connected Sets and their Application for Trend Detection in Spatial databases. *Proceedings of the KDD Conference*, 1997.

[15] C. Faloutsos et al. Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems*, Vol 3, pp. 231-262, 1994.

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

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

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

[19] H. Hofman, A. Siebes, A. Wilhelm. Visualizing Association Rules with Interactive Mosaic Plots. *ACM KDD Conference*, 2000.

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

[21] N. Katayama, S. Satoh: The SR-Tree: An Index Structure for High Dimensional Nearest Neighbor Queries. *ACM SIGMOD Conference*, pages 369–380, 1997.

[22] N. Katayama, S. Satoh. Distinctiveness Sensitive Nearest Neighbor Search for Efficient Similarity Retrieval of Multimedia Information. *Proceedings of the ICDE Conference*, 2001.

[23] D. A. Keim. *Visual Support for Query Specification and Data Mining.* Shaker Publishing Company, Aachen, Germany 1995.

[24] D. A. Keim, H.-P. Kriegel, T. Seidl. Supporting Data Mining of Large Databases by Visual Feedback Queries. *ICDE Conference*, 1994.

[25] K.-I. Lin, H. V. Jagadish, C. Faloutsos The TV-tree: An Index Structure for High Dimensional Data. *VLDB Journal*, Volume 3, Number 4, pages 517–542, 1992.

[26] Y. Rui, T. S. Huang, S. Mehrotra, Content-based image retrieval with relevance feedback in MARS. *Proceedings of the IEEE Conference on Image Processing*, 1997.

[27] G. Salton. *THE SMART Retrieval System - Experiments in Automatic Document Processing*, Prentice Hall, Englewood Cliffs, NJ, 1971.

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

[29] T. Seidl, H.-P. Kriegel: Efficient User-Adaptable Similarity Search in Large Multimedia Databases. *VLDB Conference Proceedings*, 1997.

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

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

[32] A. K. H. Tung, R. Ng, L. V. S. Lakshmanan, J. Han. Constraint-based clustering in large databases. *ICDT Conference*, 2001.

[33] R. Weber, H.-J. Schek, S. Blott: A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces, *VLDB Conference Proceedings*, 1998.

[34] L. Wu, C. Faloutsos, K. Sycara, T. Payne. FALCON: Feedback Adaptive Loop for Content-Based Retrieval. *VLDB Conference Proceedings*, 2000.