Towards Meaningful High-Dimensional Nearest Neighbor Search by Human-Computer Interaction

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

Abstract

Nearest Neighbor search is an important and widely used problem in a number of important application domains. In many of these domains, the dimensionality of the data representation is often very high. Recent theoretical results have shown that the concept of proximity or nearest neighbors may not be very meaningful for the high dimensional case. Therefore, it is often a complex problem to find good quality nearest neighbors in such data sets. Furthermore, it is also difficult to judge the value and relevance of the returned results. In fact, it is hard for any fully automated system to satisfy a user about the quality of the nearest neighbors found unless he is directly involved in the process. This is especially the case for high dimensional data in which the meaningfulness of the nearest neighbors found is questionable. In this paper, we address the complex problem of high dimensional nearest neighbor search from the user perspective by designing a system which uses effective cooperation between the human and the computer. The system provides the user with visual representations of carefully chosen subspaces of the data in order to repeatedly elicit his preferences about the data patterns which are most closely related to the query point. These preferences are used in order to determine and quantify the meaningfulness of the nearest neighbors. Our system is not only able to find and quantify the meaningfulness of the nearest neighbors, but is also able to diagnose situations in which the nearest neighbors found are truly not meaningful.

1. Introduction

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. Consequently a wide variety of access methods and data structures have been proposed for high dimensional nearest neighbor search [9, 11, 18, 21, 27].

It has been questioned in recent theoretical work [10] 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 [15] 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 [3, 6] 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 [15] 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 [19] 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 [6, 15] 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 recent years, the importance of incorporating human interaction into several data mining problems has been well understood and appreciated [5, 8, 14, 16, 20, 24, 28]. For particular domains of data such as multimedia, image databases and information retrieval, application-specific methods have been devised [13, 22, 23, 25, 28] in order to incorporate user feedback into the retrieval process. The importance of human interaction in the nearest neighbor search 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 work of finding nearest neighbors 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 quality of the nearest neighbors found.

In this paper, we will describe 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. Recent work [4, 7, 15] has shown 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*.

This paper is organized as follows. The remainder of this Section discusses issues of nearest neighbor search in high dimensional data, and formalizes the contributions of the paper. In section 2, we discuss the interactive system for nearest neighbor search by data exploration. In section 3, we discuss the quantification of nearest neighbor meaningfulness. The empirical section is discussed in section 4, whereas section 5 contains the conclusions and summary.

1.1. On the Nature of High Dimensional Data

Recent theoretical results [10] have shown that in high dimensional space, the nearest and furthest neighbors are of the same relative distance to the query point for a wide variety of data distributions and distance functions. However, it has also been shown in recent work [4, 7, 15] 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 [15] 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 1(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 1(b) and 1(c) are examples of pro-

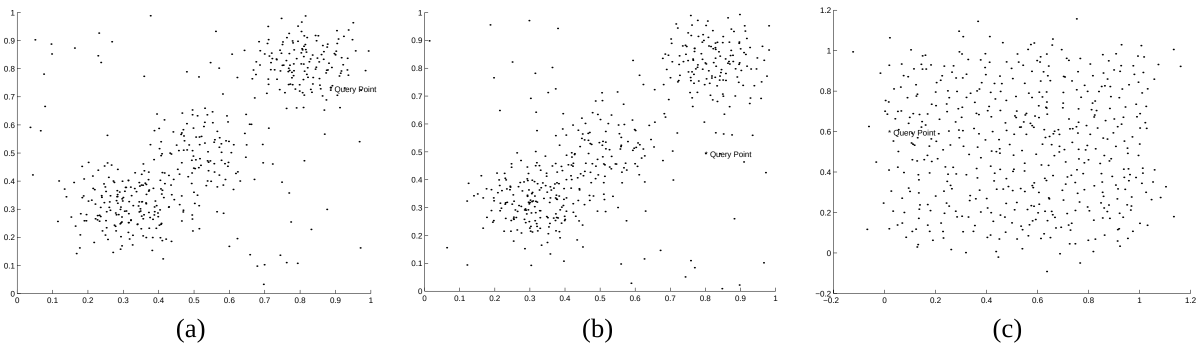


Figure 1. (a) Good Query Centered Projection (b) Poor Query Centered Projection (Query Point in Sparse Region) (c) Noisy projection

jections 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 1(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 1(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 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.

1.2. Contributions of this paper

This paper discusses a interactive system between the user and the computer for high dimensional nearest neighbor search. The advantage behind such a system is its ability to share the best skills of a human and a computer in finding the most meaningful nearest neighbors. Effective techniques are proposed for choosing projections of the data in which the natural data patterns containing the query point are well distinguished from the entire data set. An interesting method discussed in this paper is a *graded subspace determination process* so that most of the data discrimination is hidden in a small subset of orthogonal projections. The visual profiles of these projections is used to provide feedback. The repeated feedback of the user over multiple iterations is used to determine a set of neighbors which are statistically significant and meaningful. In the event that the data does not have any consistently interesting patterns in different lower dimensional projections, then this is detected by the meaningfulness quantification method and reported. Thus, an additional advantage of this approach is that it is also able to detect and report cases when the data is truly not amenable to meaningful nearest neighbor search.

1.3. Notations and Terminology

We assume that the total number of points is N and the dimensionality of the data space is d. The query point is denoted by Q. The data set is denoted by $\mathcal{D}$ and the universal d-dimensional space by $\mathcal{U}$. Let $\mathcal{E}$ be the l-dimensional subspace defined by a set of $l \leq d$ orthogonal vectors $\{\overline{e_1} \ldots \overline{e_l}\}$. In the general case, the vectors $\overline{e_1} \ldots \overline{e_l}$ may be arbitrary directions in the space which are not parallel to the axes directions representing the attributes. The projection of a point $\overline{y}$ onto the subspace $\mathcal{E}$ is the l-dimensional point $(\overline{y} \cdot \overline{e_1} \ldots \overline{y} \cdot \overline{e_l})$ and is denoted by $Proj(\overline{y}, \mathcal{E})$. The projected distance between two points $\overline{x_1}$ and $\overline{x_2}$ in subspace $\mathcal{E}$ is denoted by $Pdist(x_1, x_2, \mathcal{E})$ and is equal to the distance between $Proj(\overline{x_1}, \mathcal{E})$ and $Proj(\overline{x_2}, \mathcal{E})$ in subspace $\mathcal{E}$.

2. The Interactive Nearest Neighbor System

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

```
Algorithm InteractiveNNSearch(Data Set: $\mathcal{D}$,
                    QueryPoint: $Q$, Support: $s$);
begin
  for each data point $i$ do set $\mathcal{P}(i) = 0$;
  { $\mathcal{P} = \{\mathcal{P}(1) \dots \mathcal{P}(N)\}$ denotes the probability of each of
  the $N$ points in the database being a meaningful nearest neighbor;}
  while not(termination_criterion) do
  begin
    Set count $v(i)$ if each data point $i$ to zero;
    $\mathcal{D}_c = \mathcal{D}$; $\mathcal{E}_c = \mathcal{U}$; { $\mathcal{U}$ is universal subspace }
    { User counts are $\mathcal{V} = \{v(1) \dots v(N)\}$ };
    for $i = 1$ to $d/2$ do
    begin
      $(\mathcal{E}_{proj}, \mathcal{E}_{new}, \mathcal{D}_{new})$ = FindQueryCenteredProjection($\mathcal{D}_c$, $\mathcal{E}_c$, $Q$, $s$);
      DisplayVisualProfile($\mathcal{D}_c$, $\mathcal{E}_{proj}$, $\infty$);
      $\phi$ = AdjustDensitySeparator($\mathcal{D}_c$, $\mathcal{E}_{proj}$, $\phi$);
      $\mathcal{V}$ = UpdateCounts($\mathcal{D}_c$, $\mathcal{E}_{proj}$, $Q$, $\phi$, $\mathcal{V}$);
      $\mathcal{E}_c = \mathcal{E}_{new}$; $\mathcal{D}_c = \mathcal{D}_{new}$;
    end;
    $\mathcal{P}$ = QuantifyMeaningfulness($\mathcal{V}$, $\mathcal{P}$);
    Remove any point $i$ from $\mathcal{D}$ for which $v(i) = 0$;
    end;
  return $s$ points with highest value of $\mathcal{P}(i)$;
end
```

Figure 2. The Nearest Neighbor Algorithm

```
Algorithm FindQueryCenteredProjections(Data Set: $\mathcal{D}_c$,
          Current Subspace: $\mathcal{E}_c$, QueryPoint: $q$, Support: $s$)
begin
  $l_p$ = Dimensionality of $\mathcal{E}_c$; $\mathcal{E}_p = \mathcal{E}_c$;
  while $l_p > 2$ do
  begin
    Compute the value of $Pdist(q, x, \mathcal{E}_p)$ for each data point $x \in \mathcal{D}_c$;
    Find the nearest $s$ points to $q$ with smallest value of $Pdist(q, x, \mathcal{E}_p)$
              and denote by $\mathcal{N}$;
    $\mathcal{E}_p = QueryClusterSubspace(\mathcal{N}_p, l_p)$; $l_p = \max\{2, [l_p/2]\}$;
  end
  $\mathcal{E}_{new} = \mathcal{E}_c - \mathcal{E}_p$;
  Compute $\mathcal{D}_{new}$ as the projection of the set of points in $\mathcal{D}_c$ onto $\mathcal{E}_{new}$;
  return($\mathcal{E}_p$, $\mathcal{E}_{new}$, $\mathcal{D}_{new}$);
end
```

Figure 3. Finding Query Centered Projections

```
Algorithm QueryClusterSubspace(Query Cluster: $C$,
          Dimensionality of Subspace: $l_p$);
begin
  Let $\delta_{ij}^{\mathcal{C}}$ be covariance between dimensions $i$ and $j$ using the points in $\mathcal{C}$;
  { We denote the corresponding covariance matrix by $\Delta = [\delta_{ij}]$ };
  Determine the eigenvectors of matrix $\Delta$ with eigenvalues $\lambda_i$;
  Let $\gamma_i$ be the variance of entire data set along eigenvector $i$;
  $\mathcal{E}$ = Set of $l_p$ eigenvectors with least value of $\lambda_i/\gamma_i$;
  return($\mathcal{E}$);
end;
```

Figure 4. Determining Query Cluster Subspaces

```
Algorithm DisplayVisualProfile(Data Set: $\mathcal{D}_c$,
     Projection Subspace: $\mathcal{E}_{proj}$, Noise Threshold Density: $\phi$ )
begin
  for each data point $x_i \in \mathcal{D}_c$ compute $y_i = Proj(x_i, \mathcal{E}_{proj})$;
  $\mathcal{Y} = \{y_1, \dots y_N\}$;
  Divide the 2-dimensional hyperplane for $\mathcal{E}_{proj}$ into a $p * p$ grid;
  { We denote the coordinate points on the grid to be $\{z_1 \dots z_{p^2}\}$; }
  Use $\mathcal{Y}$ to compute kernel density $\tau(z_i)$ on the $p^2$ grid points;
  Display the density profile $\tau(\cdot)$ for the subspace $\mathcal{E}_{proj}$;
  Superpose the density profile with a hyperplane at density value $\phi$;
end
```

Figure 5. Computing Visual Profiles

```
Algorithm AdjustDensitySeparator(Data Set: $\mathcal{D}_c$,
          Query Cluster Subspace: $\mathcal{E}_{proj}$);
begin
  interaction-flag=true;
  while (interaction-flag==true) do
  begin
    User inputs new height of density separator $\phi$;
    DisplayVisualProfile($\mathcal{D}_c$, $\mathcal{E}_{proj}$, $\phi$);
    User specifies interaction-flag;
  end;
  return($\phi$);
end
```

Figure 6. Visually Separating Query Clusters

```
Algorithm UpdateCounts(Data Set:$\mathcal{D}_c$, Query Cluster Subspace: $\mathcal{E}_{proj}$,
        Query Point: $Q$, Separator Density: $\phi$, Counts: $\mathcal{V}$);
begin
  for each data point $x_i \in \mathcal{D}_c$ do
  begin
    Compute density at $x_i$ in subspace $\mathcal{E}_{proj}$;
    if $x_i$ is density-connected to $Q$ and has density at least
      $\phi$ then add one to count $v(i)$ of record $i$;
  end;
  return($\mathcal{V}$);
end
```

Figure 7. Updating User Preference Counts

```
Algorithm QuantifyMeaningfulness(Counts: $\mathcal{V}$,
          Meaningfulness Vector: $\mathcal{P}$);
begin
  { $n_i$ is the number of points picked by
  user in projection $i$, $i \in \{1, \dots [d/2]\}$ }
  for $j = 1$ to $N$ do
  begin
    $E[Y_j] = \sum_{i=1}^{d/2} n_i/N$; $var(Y_j) = \sum_{i=1}^{d/2} (n_i/N) \cdot (1 - n_i/N)$;
    $p = \frac{v(j) - E[Y_j]}{\sqrt{var(Y_j)}}$; $\mathcal{P}(j) = \mathcal{P}(j) + p$;
  end;
  return($\mathcal{P}$);
end;
```

Figure 8. Quantification of Meaningfulness

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 overall framework of the algorithm is illustrated in Figure 2. 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 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.

2.1. Finding Good Query Centered Projections

The overall process for determining a discriminatory projection is illustrated in Figure 3. A discriminatory projection is defined as one which distinguishes the natural lower dimensional clusters containing the query point from the rest of the data. In order to find the most highly discriminating projection, it may often be desirable to use projections which are created by arbitrary sets of vectors $\{\overline{e_1}\dots\overline{e_l}\}$ which are not parallel to the original axis system. In other cases, it may be desirable to pick projections from the original set of attributes for reasons of better interpretability. Our system can support both versions. In the following discussion, we will first discuss the general case, and then discuss the minor changes required for the particular case of axis-parallel projections.

In order to find the most discriminatory projections, we compare the distribution of the nearest s points, as compared to the rest of the data set. Our goal is to pick a projection in which this small fraction of points shows a well distinguished cluster around the query point. Since the current data set $\mathcal{D}_c$ is represented in the space $\mathcal{E}_c$, the projection subspace needs to be a subspace of $\mathcal{E}_c$. The process of finding the query cluster and query subspace is an iterative one. We start with the subspace $\mathcal{E}_p = \mathcal{E}_c$ from which the 2-dimensional projection $\mathcal{E}_{proj}$ needs to be found. In each iteration, we reduce the dimensionality of the subspace $\mathcal{E}_p$ in which there is a distinct cluster $\mathcal{N}_p$ surrounding the query point which is also well separated from the data. In the first iteration, we start off with $\mathcal{N}_p$ being the set of points which are closest to the query point Q in the subspace $\mathcal{E}_c$. Then, we find a subspace $\mathcal{E}_p$ in which this "query cluster" $\mathcal{N}_p$ is a tightly-knit cluster as compared to the variance of the remaining data set. In order to do so, we find the principle component directions [17] of the set of points in $\mathcal{N}_p$. The principle component directions are helpful in finding those projections of the data in which $\mathcal{N}_p$ is tightly clustered. In order to find the principle components, we determine the covariance matrix Δ of the set of points in $\mathcal{N}_p$. Since the data points in $\mathcal{N}_p$ have dimensionality $|\mathcal{E}_c|$, the covariance matrix is an $|\mathcal{E}_c| * |\mathcal{E}_c|$ matrix in which the entry (i, j) denotes the covariance between dimensions i and j. This covariance matrix is positive semi-definite and can be expressed as $\Delta = P \cdot D \cdot P^T$, where D is a diagonal matrix containing the eigenvalues, and the columns of P contain the eigenvectors which form an orthonormal axis-system. These eigenvectors are the principal components and represent the directions in the data along which the second order covariances of the points in $\mathcal{N}_p$ are zero. The eigenvalue λ_i along the direction i denotes the variance of the set of points in $\mathcal{N}_p$ along the direction i. Therefore, if γ_i be the variance of the entire data set $\mathcal{D}_c$ along the eigenvector i, then the ratio λ_i/γ_i denotes the ratio of the variances between the query cluster and remaining data when projected onto the eigenvector i. Therefore, by picking the l_p directions with the smallest variance ratio, we are able to determine the directions in which the query cluster is well distinguished from the rest of the data. The procedure for determining the query cluster subspace is illustrated in Figure 4.

Once the query subspace has been determined, then it will be used in the next iteration in order to determine a new query cluster $\mathcal{N}_p$. Specifically, the set of points $\mathcal{N}_p$ in the next iteration is determined by finding those points which are closest to the query point when projected into the newly found subspace $\mathcal{E}_p$. The value of the dimensionality of the subspace $\mathcal{E}_p$ is denoted by l_p and is reduced by factor of 2 in each iteration. The process continues till the value of l_p is 2. The reason for the iterative methodology used by the algorithm is that both $\mathcal{N}_p$ and $\mathcal{E}_p$ are dependent on one another and the gradual reduction in the dimensionality ensures an effective refining process in which we find query cluster $\mathcal{N}_p$ and a corresponding subspace $\mathcal{E}_{proj}$ in which this cluster is well distinguished from the rest of the data. When it is desirable to use clusters only from axis parallel projections, then a minor modification needs to the query subspace determination subroutine of Figure 4. Here instead of using the principal components of the set of data points in $\mathcal{N}_p$, we use the original set of axis directions.

Once the projection subspace $\mathcal{E}_p$ is determined, we compute the new subspace and data set $\mathcal{E}_{new}$ and $\mathcal{D}_{new}$ which will be used in order to determine the projection in the next iteration. We wish to ensure that later iterations find only subspaces which are orthogonal to those found so far. Therefore, $\mathcal{E}_{new}$ is chosen as the complementary subspace to $\mathcal{E}_p$, assuming that $\mathcal{E}_c$ is the entire subspace. The data set $\mathcal{D}_c$ is also projected onto this new subspace in order to create $\mathcal{D}_{new}$.

2.2. Interactive Separation of Query Cluster

Once a discriminatory projection has been determined, human interaction is used in order to separate the query cluster from the remaining data points. In order to maximize the use of human intuition, we use a visual profile of the probabilistic data distribution. To this effect, we use kernel density estimation techniques [26]. In this technique the probability density at a given point is estimated as the sum of the smoothed values of kernel functions $K_h(\cdot)$ associated with each point in the data set. Each kernel function is associated with a kernel width h which determines the level of smoothing created by the function. The kernel estimation $\overline{f}(x)$ based on N data points $x_1 \dots x_N$ and kernel function $K_h(\cdot)$ is defined as follows:

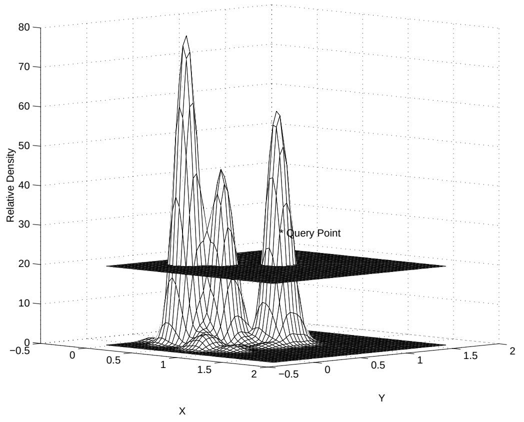
$$\overline{f}(x) = (1/N) \cdot \sum_{i=1}^{N} K_h(x - x_i) \tag{1}$$

Thus, each discrete point x_i in the data set is replaced by a continuous function $K_h(\cdot)$ which peaks at x_i and has a variance which is determined by the smoothing parameter h. An example of such a distribution would be a gaussian kernel with width h.

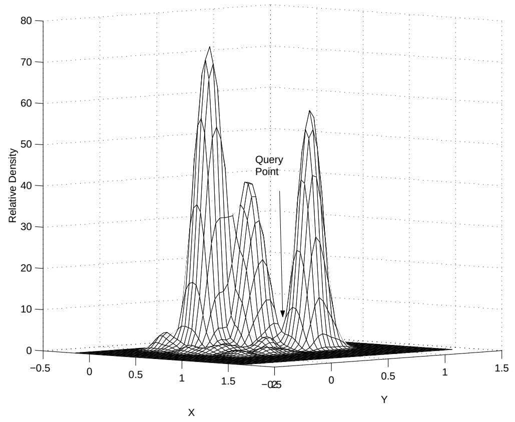
$$K_h(x - x_i) = (1/\sqrt{2\pi} \cdot h) \cdot e^{-(x-x_i)^2/2h^2} \tag{2}$$

The error in density estimation is determined by the bandwidth h. One well known approximation formula [26] for determining the bandwidth is $h = 1.06 \cdot \sigma \cdot N^{-1/5}$ for a data set with N points and standard deviation σ.

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 two such density profiles are illustrated in Figures 9(a) and 9(b). Note that in the case of Figure 9(a) 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 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 1(a), 1(b), and 1(c) are lateral scatter plots of 500 points generated from synthetic data sets.



(a) A Good Query Centered Projection



(b) A Poor Query Centered Projection

Figure 9. Illustration of the Qulaity of Projections

Once the user is provided 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 is discussed in [12, 16], and is defined as follows:

Definition 2.1 *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 9(a), we have shown the density profile of a data set along with a *density separator planes* which separates the data out into different clusters. 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. 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 9(a), there are two density connected regions above the noise threshold ϕ. All the data points which lie in the same region as the query cluster are the set of preferences for that particular projection. 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.

As discussed in an earlier section, not all views are equally informative in understanding the relationship of the data to the query point. For example, the projection for Figure 9(a) is significantly better than the similar profile of Figure 9(b), since in the former case the query point is located on a peak of the density profile, whereas in the latter case the query point is in a sparse region of the data. In such cases, it is difficult to find a coherent cluster of points in the projection, which are related to the query point. The user can choose to ignore this projection by specifying an arbitrarily high value of the noise threshold ϕ.

An alternative way of separating the query cluster is by using the lateral density plot in which the user visually specifies the separating hyperplanes (lines) in order to divide the space into a set of polygonal regions. The set of points in the same polygonal region as the query point is the user response to the query for that particular projection. However, using a density separator tends to be a more attractive option, since it can separate out clusters of arbitrary shapes with the specification of a single noise threshold. The algorithms for displaying visual profiles and performing user interaction are illustrated in Figures 5 and 6 respectively.

2.3. Updating the Preference Counts

After each minor iteration, we need to update the preference counts for the points which have been determined to lie in the query cluster. An important problem is to discover all the points which are density connected to the query point, without having to calculate the density value at each individual data point. It is possible to use the density values calculated across the $p * p$ grid structure in order to approximate the points which are density connected to the query point. The first step is to find all the elementary rectangles in the grid structure which approximately lie within the (ϕ, Q)-contour. We shall denote this set of elementary rectangles by $\mathcal{R}(\phi, Q)$. We define $\mathcal{R}(\phi, Q)$ as follows:

Definition 2.2 *An elementary rectangle $\mathcal{L}$ is defined to be a member of $\mathcal{R}(\phi, Q)$, if and only if there exists some sequence of adjacent rectangles $\mathcal{L} = \mathcal{L}_0, \mathcal{L}_1 \ldots \mathcal{L}_k$ such that (i) $\mathcal{L}_k$ contains Q, and (ii) at least three corners of each rectangle $\mathcal{L}_i$ have density above the noise threshold ϕ.*

Two rectangles are said to be adjacent, when they share a common side. In effect, the set $\mathcal{R}(\phi, Q)$ is the set of high density rectangles which are connected to the rectangle containing Q by some adjacent sequence of high-density rectangles. In order to find $\mathcal{R}(\phi, Q)$, we use a simple graph search algorithm in which we start at the rectangle containing Q and keep searching adjacent rectangles until we have determined all rectangles which lie in $\mathcal{R}(\phi, Q)$.

Once the points inside all these rectangles have been determined, we increment their counts by one unit. It is also possible to weight different query clusters by importance. Specifically, the weight for each point may be increased by w_i corresponding to the projection i. This procedure is illustrated in Figure 7. In this paper, we always assume that $w_i = 1$; therefore each query cluster is considered equally important. It is important to understand that since the user has the ability to pick only those projections in which there are meaningful data patterns surrounding the query point, the preference counts $v(\cdot)$ will not be affected by those noisy combinations of dimensions which are not useful for the nearest neighbor search process. At the end of each major iteration, the count $v(i)$ for each point i quantified into a meaningfulness value, and is used in order to update a corresponding probability value for that data point. This process is outlined in Figure 8, and is described in detail in the next section.

2.4. Iterative User Preference Quantification

The iterative process discussed above continues until a sufficient amount of feedback has been calculated to find and quantify the meaningfulness of the nearest neighbors. In order to do so, we convert the preference counts $v(\cdot)$ of the user into *meaningfulness probabilities* in each major iteration. These probabilities quantify the level of coherence in the behavior of the user in classifying a certain point into the query cluster across different projections. The variation in these probabilities from iteration to iteration is used in order to decide whether the process should terminate. The process for conversion of the user preference counts in each major iteration into a probability vector $\mathcal{P}(\cdot)$ and subsequent termination is discussed in the next section.

3. User Quantification of Meaningfulness

If the user coherently picks similar points across the different orthogonal projections in a given major iteration, then such behavior can be used to quantify the meaningfulness of the technique. After each major iteration, the set of preferences provided by the user $v(\cdot)$ are used to update a meaningfulness probability vector $\mathcal{P}(\cdot)$. This procedure is illustrated in Figure 8. First, we will analyze the coherence of a set of reactions by the user in a sequence of $d/2$ projections. Let X_{ij} be a random variable which denotes whether ($X_{ij} = 1$) or not ($X_{ij} = 0$) the user picks the point j in projection i. We note that X_{ij} is a bernoulli random variable, and if n_i be the number of points that a user picks in projection i, then the probability that $X_{ij} = 1$ for the data point j is given by n_i/N. Let w_i be the weight for each preference count in projection i. Then, the random variable Y_j indicating the total user preference for point j is given by:

$$Y_j = \sum_{i=1}^{d/2} w_i \cdot X_{ij} \tag{3}$$

The corresponding expected value $E[Y_j]$ is given by:

$$E[Y_j] = \sum_{i=1}^{N} w_i \cdot n_i/N \tag{4}$$

The value of $E[Y_j]$ is the same for every data point j, and is simply the sum of the (weighted) fractions of points picked by the user in the different projections. Since this system tries to detect the relationships across preference patterns in different projections (because of correlations among the different attributes), it is instructive to look at the case when the data is completely uncorrelated. If such were the case, then the preference values of the users in the different projections in an iteration would be uncorrelated to one another. This would mean that the random variables $X_{1j} \ldots X_{d/2,j}$ are also independent. Consequently, we can compute the variance of Y_j as the sum of the variances of the individual components $w_i \cdot X_{ij}$. Since X_{ij} is a bernoulli random variable with probability n_i/N, we have:

$$var(Y_j) = \sum_{i=1}^{d/2} w_i^2 \cdot (n_i/N) \cdot (1 - n_i/N) \qquad (5)$$

Again, $var(Y_j)$ is independent of j. Let $v(j)$ be the true number of preference counts that a user has given to a data point x_j. When the data is distributed in a noisy way, then the preference pattern across different projections will not show any meaningful consistency. Therefore, the value of $v(j)$ will not vary significantly from $E[Y_j]$. In order to quantify this notion, we define the *meaningfulness coefficient* $M(j)$ for the point x_j as follows:

$$M(j) = (v(j) - E[Y_j])/\sqrt{var(Y_j)} \qquad (6)$$

The meaningfulness coefficient is a numerical estimate of the level of confidence with which the nearest neighbor found is closer to the target than the average. Note that when the value of d is high, the distribution of $M(j)$ is approximately normal. Let $\Phi(\cdot)$ be the cumulative distribution of the normal distribution with zero mean and unit variance. In such a case, we can compute the *meaningfulness probability* $\mathcal{P}(j)$ as follows:

$$\mathcal{P}(j) = \max\{2 \cdot \Phi(M(j)) - 1, 0\} \qquad (7)$$

This value is equal to the probability that the preference count for data point x_j is larger than the expected preference count $E[Y_j]$ purely by chance. Note that when the value of x_j is smaller than $E[Y]$, this value is equal to zero. When the user preference patterns shows considerable consistency across different projections, we expect $\mathcal{P}(j)$ to be almost one for some of the points, whereas for the other points, the value of $\mathcal{P}(j)$ is significantly less than one. This is a very desirable situation, since some of the data points can be clearly distinguished as the nearest neighbors.

In a single iteration, we obtain the user preference patterns from a set of $d/2$ mutually orthogonal projections. The above analysis for calculation of $\mathcal{P}(j)$ is based on such an iteration of mutually orthogonal projections. However, as illustrated in Figure 2, the process is repeated for multiple iterations in order to obtain feedback for many sets of $d/2$ projections. In such a case, the overall meaningfulness is calculated as the arithmetic average over multiple iterations. Let us say that the values of $\mathcal{P}(j)$ for data point x_j for each of the κ iterations are given by $p_j^1, \ldots p_j^\kappa$. Then, the overall meaningfulness probability for the data point j over multiple iterations is given by the following arithmetic average:

$$\mathcal{P}(j) = \sum_{i=1}^{\kappa} (p_j^i)/\kappa \qquad (8)$$

We note that the values of the meaningfulness probability vector $\mathcal{P}(\cdot)$ in two successive iterations will be highly correlated with one another and the level of this correlation will increase with the value of κ. Therefore we compare the set of s points with highest value of $\mathcal{P}(\cdot)$ in the iteration $\kappa - 1$ and κ. When the percentage of common points between these two iterations is larger than a certain threshold t, then we terminate. At the end of the procedure, we return the s data points which have the highest value of $\mathcal{P}(j)$. Note that in Figure 8 for each data point j we maintain the value of $\sum_{i=1}^{\kappa}(p_j^i)$ as opposed to the average. Therefore, the true value of the meaningfulness probability may be obtained by dividing this value by κ.

Data Set	Precision	Recall
Synthetic 1	87%	98%
Synthetic 2	91%	96%

Table 1. Accuracy on Synthetic Data Sets

4. Empirical Results

In this section, we will discuss the results obtained by using our user-adaptable system for a variety of real and synthetic data sets. For the case of synthetic data sets, we show that the nearest neighbors indeed lie within the natural projected clusters which are created for testing purposes. We also show some interesting examples of high dimensional data in which the data is truly distributed in a noisy and meaningless way. In these cases, we show that the technique is effectively able to predict the meaninglessness of applying a nearest neighbor search process.

4.1. Synthetic Data Sets

We generated a set of sparse synthetic data sets in high dimensionality, such that projected clusters were embedded in lower dimensional subspaces. We generated two data sets with $N = 5000$ points using this technique. We shall refer to these data sets as Case 1 and Case 2 respectively with the same parameters used in [4], except for the number of points. These data sets contain 6-dimensional projected clusters embedded in 20 dimensional data.

For the purpose of testing, we adopted the policy of isolating a cluster with the query point containing about $0.5 - 5\%$ of the data. Correspondingly, the value of the support used in order to determine a discriminatory projection

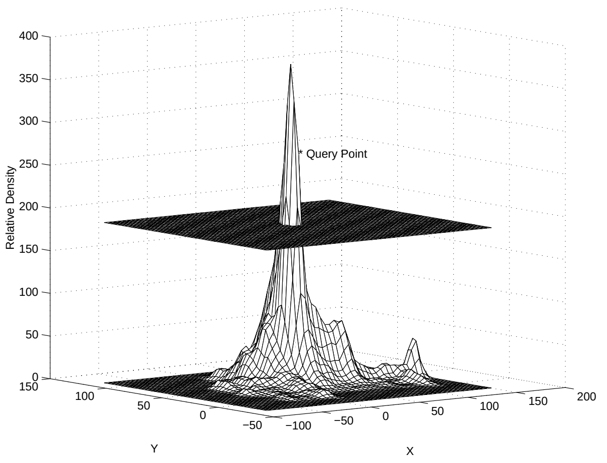


Figure 10. Density Profile (Syn. 1: Early Minor Iteration)

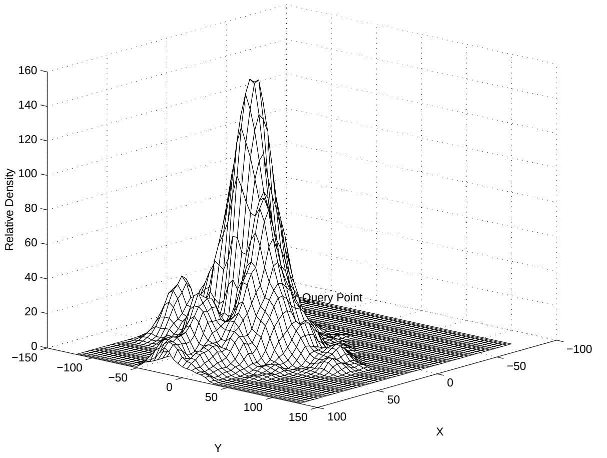


Figure 11. Density Profile (Syn. 1: Late Minor Iteration)

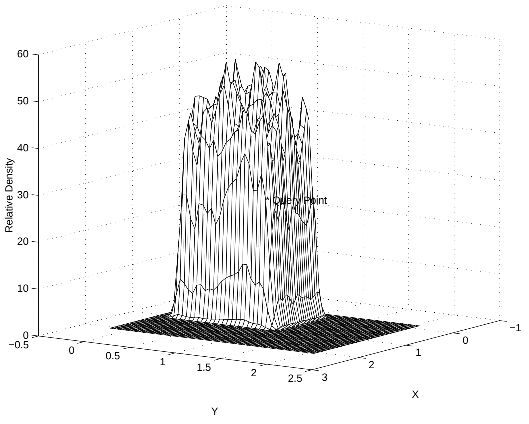


Figure 12. Density Profile (Uniform)

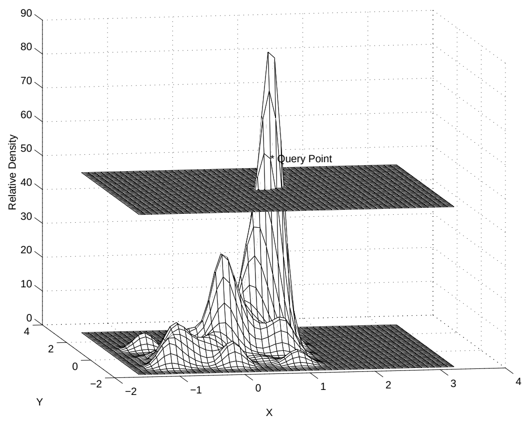


Figure 13. Density Profile (Ionosphere)

was set at 0.5%. However, in many cases, when the visual profile was constructed, the actual cluster was often either much smaller or larger than this threshold. In such cases, the interactive query cluster separation process was able to correct for any discrepancies. Because of the careful method in which the subspace is determined, we found that in most cases a clear cluster could be found near the query point. In each major iteration, the visual profiles obtained during the first few minor iterations were the most discriminatory. For example, a visual profile obtained during an early (first) minor iteration for the first case is illustrated in Figure 10, whereas the visual profile in the last minor iteration is illustrated in Figure 11. It is clear that the former profile is one in which the query cluster can be more clearly distinguished from the remaining data points. This is because in the first few minor iterations, the subspace determination subroutine has considerable flexibility in choosing a subspace which results in the best query centered projection. This does not continue to be the case in the last iteration in which the algorithm is forced to pick from the subspace which is complementary to the union of all the subspaces already chosen. *This gradation in the quality of the projections has an important influence on the nearest neighbor search process.* Since the user can choose to discard the projections determined in the last few minor iterations, only the nicely coherent behavior of the data is reflected in the user preference counts $v(\cdot)$. Thus, the graded quality of the projections ensures that most of the noise in the data is pushed into the last few projections, and the user is able to use his intuition in order to easily pick out the clearly coherent projections (and data patterns) in the first few minor iterations. In other words, at the end of each major iteration, the user preference counts implicitly define a relevance value for each data point in which the noise/sparsity effects of high dimensionality have been filtered with the use of human intuition. However, this intuition could not have been harnessed without the use of a carefully graded subspace determination process. This interdependence between the user and the computer reflects the nature of the interaction between the two entities. The visual profile for the first and last minor iterations in the second data set were similar to those obtained in the first. In each case, we determined the meaningfulness probability of each data point at the termination of the process. We sorted the data in order of meaningfulness probability and found that a few of the data points had meaningfulness probability in the range of 0.9 to 1, after which there was a steep drop. This steep drop corresponds to the distinct projected cluster to which the query point belongs. By using the threshold which occurs just before this steep drop, it is possible to isolate the natural set of points related to the query. Note that the corresponding cardinality may be quite different from the user-specified support. In this case, the corresponding value was 0.9. This corresponds to about 520 neighbors in Case 1, which had a meaningfulness probability higher than this threshold. This also compares well with the cardinality (562) of the projected cluster containing the query point. Of the 520 neighbors recovered in case 1, 508 belonged to the same cluster as the query point. In order to illustrate the effectiveness of the technique, we have illustrated some summary results for the case 1 and case 2 data sets in Table 1 over a set of 10 examples on which we ran the method. We have illustrated the precision and recall of the two techniques in the Table using the natural number of nearest neighbors found by the thresholding technique. Typically, the natural number of nearest neighbors are often a slight overestimate (about 5 to 15% over the correct value), and hence the recall values are higher than the precision. The recall value indicates that very few of the true nearest neighbors are actually missed. It is clear that in each case, since both the precision and recall are so high, the nearest neighbor search technique was not only accurate, but was able to determine the "natural" number of nearest neighbors effectively. This is very useful for nearest neighbor applications in which finding a natural number of nearest neighbors is as important as the quality of the records returned.

4.2. Results on Poorly Behaved Data

The meaningfulness issue of the nearest neighbor problem in high dimensionality critically depends upon the fact that often the local implicit dimensionality of the data is significantly lower than the full dimensionality [1, 2, 4, 11]. The technique discussed in this paper exploits this fact in conjunction with the user interaction. However, for some data sets even the local implicit dimensionality is high. An example of such a case is uniformly distributed data. In such a situation, the problem of nearest neighbor search is indeed not very meaningful in high dimensionality. Therefore, it is interesting to test what happens in such a system for uniformly distributed data.

Data Set (Dimensionality)	Accuracy (L_2)	Accuracy (Interactive)
Ionosphere (34)	71%	86%
Segmentation (19)	61%	83%

Table 2. Accuracy on Real Data Sets

We tested a case with $N = 5000$ uniformly distributed points in $d = 20$ dimensions. In this case, we found that it was difficult to find views in which the points were well discriminated from one another. A typical example of a view obtained from such a case is illustrated in Figure 12. We note that the discrimination of the the data surrounding the query cluster is very poor in such a case. This is valuable information for a user, since it tends to indicate the poor selectivity of the data even in carefully chosen projections. Therefore, a user can infer that the data is not very prone to meaningful nearest neighbor search in high dimensional space. Even further evidence may be obtained from the distribution of the meaningfulness probability values. Even though it was difficult to pick out the query cluster because of the poor discrimination behavior, we were able to find a few query-clusters in some of the projections because of local variations in density. When the process was completed, we found that there was very little coherence in the preference counts across the different views, and they got evenly distributed among the different data points. In this case, the meaningfulness values do not show the kind of steep drop which is visible in the synthetic data sets. Consequently, it is difficult to isolate a well defined query cluster based on a similar threshold. The conclusion from these numerous observations is that in those high dimensional cases in which meaningful nearest neighbors do not exist, the technique is able discover this valuable information.

4.3. Results on Real Data

We also tested the results on a number of real data sets obtained from the UCI machine learning[1] repository. An example of a visual profile from a query-centered projection obtained from the ionosphere data set is illustrated in Figures 13. The meaningfulness probability values also showed a similar steep drop as in the case of the clustered synthetic data. Thus, both the visual profiles and the meaningfulness behavior for this real data set is similar to the clustered synthetic data set as opposed to the uniformly distributed case. We tested the results on the ionosphere and segmentation data sets from the UCI machine learning

[1]http://www.cs.uci.edu/˜mlearn.

repository. We have illustrated the nearest neighbor classification accuracy for using 10 query points (and as many nearest neighbors as determined by the natural query cluster size) for a number of data sets. The results are illustrated in Table 2. As evident from Table 2, it is clear that the interactive nearest neighbor process was more effective than the full dimensional method. In the particular case of the segmentation data set, the improvement was 32% over the full dimensional method, which was quite significant.

5. Conclusions and Summary

In this paper, we discussed a user-adaptive method for meaningful high dimensional similarity search. High dimensional data has always been a challenge to similarity search methods from the meaningfulness perspective because of its peculiar property of getting distributed sparsely throughout the data space. Therefore, we propose an interactive method to present a user with meaningful query-centered projections which can distinguish the nearest points to the query from the remaining points. The user is provided with a visual perspective which he can use in order to separate out these points. Such choices made by the user over multiple iterations are used to quantify the meaningfulness of the discovered nearest neighbors. In cases when the nearest neighbor is truly not meaningful, our system is able to detect and report the poor behavior of the data. Thus, our system provides a unique understanding by either solving or diagnosing the poor behavior of high dimensional nearest neighbor search from the user perspective.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[16] 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.

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

[18] N. Katayama, S. Satoh: The SR-Tree: An Index Structure for High Dimensional Nearest Neighbor Queries. *SIGMOD Conference*, 1997.

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

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

[21] 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.

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

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

[24] S. Sarawagi. User-adaptive Exploration of Multidimensional Data. *VLDB Conference*, 2000.

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

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

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

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