# An Adaptive and Efficient Dimensionality Reduction Algorithm for High-Dimensional Indexing

Hui Jin[1] Beng Chin Ooi[2] Heng Tao Shen[2] Cui Yu[3] Ao Ying Zhou[4]

[1]Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, USA

[2]Department of Computer Science
National University of Singapore, Singapore

[3]Department of Computer Science
Monmouth University, NJ 07726, USA

[4]Department of Computer Science
Fudan University, China

## Abstract

*The notorious "dimensionality curse" is a well-known phenomenon for any multi-dimensional indexes attempting to scale up to high dimensions. One well known approach to overcoming degradation in performance with respect to increasing dimensions is to reduce the dimensionality of the original dataset before constructing the index. However, identifying the correlation among the dimensions and effectively reducing them is a challenging task. In this paper, we present an adaptive Multi-level Mahalanobis-based Dimensionality Reduction (MMDR) technique for high-dimensional indexing.*

*Our MMDR technique has three notable features compared to existing methods. First, it discovers elliptical clusters using only the low-dimensional subspaces. Second, data points in the different axis systems are indexed using a single $B^+$-tree. Third, our technique is highly scalable in terms of data size and dimensionality.*

*An extensive performance study using both real and synthetic datasets was conducted, and the results show that our technique not only achieves higher precision, but also enables queries to be processed efficiently.*

## 1 Introduction

Many database applications, such as multimedia retrieval, exploratory data analysis, market basket application and time-series matching, involve high-dimensional data. Indexing high-dimensional data has been an area of active research for a long time and many indexing techniques have been proposed [13]. However, the performance of these indexes degrades rapidly with increasing dimensionality [3].

One approach to minimizing the effect of "dimensionality curse" is to reduce the number of dimension of the high-dimensional data before indexing on the reduced dimension [10, 5]. Data is first transformed into a much lower dimensional space using dimensionality reduction methods and then an index is built on it.

Transforming data from a high-dimensional space to a lower dimensional space without losing critical information is not a trivial task. In this paper, we propose a dimensionality reduction technique called *Multi-level Mahalanobis-based Dimensionality Reduction* (MMDR) for indexing based on the following two observations. First, elliptical shaped (correlated) clusters are more suitable for dimensionality reduction than spherical shaped clusters. Second, we observe that certain level of the lower dimensional subspaces may contain sufficient information for correlated cluster discovery in the high-dimensional space. In the MMDR, *Principal Component Analysis*(PCA) [8] is employed to find the lower dimensions for dimension reduction. Most of the information in the original space can be condensed into a few dimensions along which the variances in the data distribution are the largest. We make use of the *Mahalanobis distance* (MahaDist) in our approach instead of the standard well-known L-norm distance functions.

Mahalanobis distance could be applied to find ellipsoidal correlated data, by taking local elongation into account. Instead of equally treating all values, MahaDist weights the differences by the range of variability in the dimension of the data points. It weights the variation along the axis of elongation less than that in the shorter axis of the ellipse. It can be shown that the surfaces on which MahaDist is a constant are ellipses.

Euclidean distance-based clustering algorithms are not meant to discover elliptical shape since the clusters identified are in circular shape. Figure 1 illustrates two clusters, one obtained using Euclidean distance and the other obtained by Mahalanobis distance. Point A is a valid point and point B is a noise in the cluster of the circle if Euclidean distance is employed. However, in terms of Mahalanobis

measurements, point B has a substantially smaller distance to the centroid than point A since it lies along the direction of the group that has the largest variance. Thus point A is a noise while point B is valid. Therefore, while Euclidean distance based algorithms produces circular subsets as shown in Figure 1, Mahalanobis distance based algorithms will produce elliptical clusters where data points are well correlated and more natural for dimensionality reduction, as dimensions with large variance of data are kept and dimensions with small variance of data are eliminated.

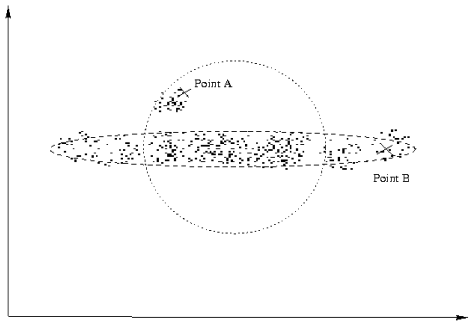



**Figure 1. Mahalanobis vs. Euclidean**

Based on multi-level low-dimensional projections produced by PCA and the Mahalanobis distance function, the MMDR can quickly identify highly correlated elliptical clusters. After the dimensionality reduction, each cluster of data is in a different axis system. Instead of creating one index for each cluster, we build one index for all the clusters for K nearest neighbor (KNN) queries. We extend a recently proposed $B^+$-tree based index - iDistance[13, 14], to index the data projections from the different reduced-dimensionality spaces. The extended iDistance allows us to index data points from different axis systems in a single index efficiently. Performance studies using real and synthetic datasets were conducted to evaluate the effectiveness and precision of the technique. The results show significant performance gain over an existing method [5]. Experiments on datasets with very high dimensionality (up to 200 dimensions) show that the proposed method is scalable in terms of both size and dimensionality.

The rest of the paper is organized as follows. In Section 2, we present some related works. In Section 3, we provide the definitions for using Principal Component Analysis and Mahalanobis distance in dimensionality reduction. We present our MMDR algorithm and its variant in Section 4. In Section 5, we propose an extended iDistance for indexing data points in reduced-dimensionality spaces. Experiments are presented in Section 6 and conclusion is drawn in Section 7.

## 2 Related Work

In dimension reduction for indexing, [5] proposed two strategies. In the first strategy, called the *Global Dimensionality Reduction* (GDR), all the data is reduced as a whole down to a suitable dimension on which search time and access costs are optimized. This strategy is unable to handle datasets that are not globally correlated. The other strategy, called the *Local Dimensionality Reduction* (LDR), divides the whole dataset into separate clusters based on correlation of the data and then indexes each cluster separately. Unfortunately, the LDR is not able to detect all the correlated clusters effectively, because it does not consider correlation nor dependency between the dimensions.

Clustering algorithms have been studied recently in the domain of data mining and pattern discrimination. Methods proposed for high-dimensional data clustering are related to our work. PROCLUS [2] clusters the data based on the correlation among the data along certain original dimensions. OptGrid [7] finds clusters in a high-dimensional space by projecting the data onto each axis and partitioning the data by using cutting planes at low-density points. Wavelet transform [12] and discrete cosine transform [9] based techniques rely on the partitioning of the data space into grids similar to OptGrid. These approaches do not work well when well-separated clusters in the actual space overlap after they are projected onto certain axis.

[1] presents various results of qualitative behaviors of L-norm distance matrices for measuring the proximity in high-dimensional spaces, and examines the meaningfulness of similarity in such spaces. They show that the clustering quality and answer sets vary from one distance metric to another. In this paper, we examine a different distance function, Mahalanobis function [6], to explore the local intrinsic cluster shape for elliptical clusters discovery (which cannot be detected by L-norm functions). Mahalanobis distance has been used in face detection to discover actual non-isotropic face patterns among thousands of face images using a k-means like algorithm called the elliptical k-means method [11]. It is a nested loop algorithm, where the inner loop is to perform k-means using Mahalanobis distance and the outer loop is to re-compute the covariance matrix of each cluster. Both loops stop when there is no change to the cluster membership.

## 3 Definitions

In this section, we provide the basic definitions.

**Definition 3.1 Ellipticity**

*Ellipticity*(e) is the deviation of an ellipse or an ellipsoid from the form of a circle or a sphere, which is the ratio of the difference of the two sub-axes to the minor axis.

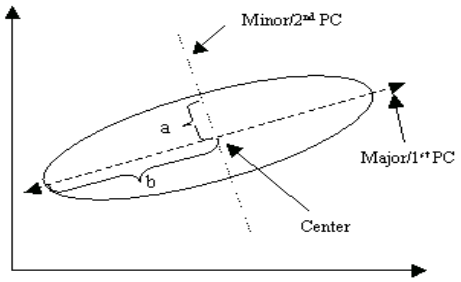



**Figure 2. Illustration of Ellipticity**

$$e = \frac{b-a}{a}$$

where *b* is the radius along the major axis and *a* is the radius along the minor axis, as shown in Figure 2.

Assuming all the points are clustered inside the ellipse. To reduce the dimensionality, all the points are projected onto the major axes. Thus, the minor axes can be eliminated. Obviously, the larger the *e* is, the more effective dimensionality reduction can be obtained. When *e* =0, the data points form a circle, and the dimensionality reduction technique becomes ineffective.

**Definition 3.2 Mahalanobis Distance**

The *covariance* of data in two feature spaces measures their tendency to vary together. In a multi-dimensional space, the variance measures the relative 'radius' of a cluster along each dimension, and the covariance indicates the orientation of the cluster. Both the variance and the covariance co-determine the shape of the cluster. Collecting them together, we get the covariance matrix **C**. Now let us look at the distance function called *Mahalanobis Distance* by using the inverse of covariance matrix.

Given a cluster centred at $O$, the *Mahalanobis Distance* between a point $P$ and $O$ is given as follows:

$$MahaDist(P,O) = (P-O)^T C^{-1}(P-O)$$

where **C** is covariance matrix describing cluster's shape.

From Mahalanobis Distance, we get a normalized measure: *Normalized Mahalanobis Distance*.

$$MahaDist_n(P,O) = \frac{1}{2}(d\ln(2\Pi\cdot \mid C \mid) + (P-O)^T C^{-1}(P-O))$$

where $d$ is the dimensionality, $\Pi$ is the trigonometric number 3.14 and $\mid C \mid$ is the determinant of **C**. Notice that given a spatial displacement between a point and an ellipsoid, the standard Mahalanobis Distance tends to be smaller for long clusters with large covariance matrices than that for small clusters. With standard Mahalanobis Distance, the larger cluster will keep increasing in size and eventually overwhelm the smaller clusters. Normalized Mahalanobis Distance avoids such situation [11].

**Definition 3.3 Multi-level Projections**

Principal Component Analysis (PCA) [8] examines the variance structure in the dataset and determines the directions along which the data exhibits high variance. The first principal component is the eigenvector corresponding to the largest eigenvalue of the dataset's covariance matrix **C**, the second component corresponds to the eigenvector with the second largest eigenvalue and so on. It is interesting to note that the Principal Components in PCA are just the eigenvectors of the covariance matrix in Mahalanobis Distance which describes the dataset's shape. An example is shown in Figure 3, where the preserved dimension is the first principal component, and the eliminated dimension is the second principal component. In dimensionality reduction, given a point $P$ in a dataset, it has two projections. One is the projection on the preserved subspace $P'$ that we are interested in; the other is the projection on the eliminated subspace $P''$. The $d_r$-dimensional projection $P'_{d_r}$ can be defined as:

$$P'_{d_r} = P \cdot \Phi_{d_r}$$

where $\Phi_{d_r}$ represents the matrix containing $1^{st}$ to $d_r^{th}$ principal components. Change $d_r$ with different value, we can generate multi-level projections of the data for cluster discovery purpose in *Multi-level Mahalanobis-based Dimensionality Reduction* algorithm.

**Definition 3.4 Projection Distance**

From the above two projections, $ProjDist_r$ measures the distance from $P$ to $P'$ and $ProjDist_e$ measures the distance from $P$ to $P''$ on the eliminated subspace. More specifically, $ProjDist_r$ is the information lost from original representation $P$ to its reduced $d_r$-dimensional representation $P'$. $ProjDist_e$ is the information retained. Figure 3 illustrates the two projection distances. In the following paragraphs, $ProjDist$ represents $ProjDist_r$.

Based on the above two projection distances, we extend the definition of **ellipticity** to multidimensional space as:

$$e = \frac{Max(ProjDist_e) - Max(ProjDist_r)}{Max(ProjDist_r)}$$

where $Max(ProjDist_e)$ is the radius along the remained subspace, and $Max(ProjDist_r)$ is the radius along the eliminated subspace. The cluster's Mahalanobis radius *r* is $Max(ProjDist_r)$. For dimensionality reduction, the larger the *ellipticity* value, the more effective dimensionality reduction can be performed.

**Definition 3.5 Mean $ProjDist_r$ Error (MPE)**

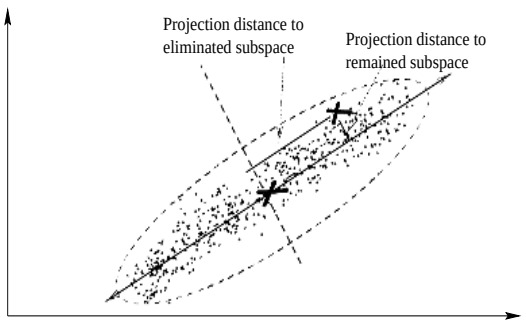



**Figure 3. Two projection distances**

| $Symbols$ | $Descriptions$ | $Value$ |
|---|---|---|
| N | Data Size | |
| d | Original Dimensionality | |
| $d_r$ | Optimal Dimensionality | |
| $s\_dim$ | Subspace Dimensionality | |
| e | Ellipsoid's Ellipticity | |
| r | Mahalanobis Radius | |
| C | Covariance Matrix | |
| $ProjDist_r$ | Dist to remained subspace | |
| $ProjDist_e$ | Dist to eliminated subspace | |
| MPE | Mean Projection Error | |
| $\sigma$ | Outlier Set | |
| $\beta$ | $ProjDist_r$ Threshold | 0.1 |
| MaxMPE | Max MPE Allowed | 0.05 |
| EC | Elliptical Cluster | |
| MaxEC | Max EC allowed | 10 |
| MaxDim | Max Remained Dim allowed | 20 |
| $\varepsilon$ | Data Stream Percentage | 0.005 |
| $\xi$ | Outlier Percentage | 0.005 |
| $k$ | Num of IDs in lookup table | 3 |

**Table 1. Table of Symbols and default values**

Mean $ProjDist_r$ Error is defined to be the average representation error when points are mapped from original space to eliminated subspace.

$$MPE = \frac{\sum_{i=1}^{N} ProjDist_r(P_i, O, L)}{N}$$

Table 1 gives a summary of the symbols and their respective description with default values used in experiments.

# 4 Multi-level Mahalanobis-based Dimensionality Reduction

## 4.1 MMDR Algorithm

The Multi-level Mahalanobis-based Dimensionality Reduction (MMDR) algorithm, which is outlined in Figure 4, consists of two major steps, namely: *Generate Ellipsoid* and *Dimensionality Optimization*.

**MMDR Algorithm**

**Generate Ellipsoid (GE)**
Variable: ellipsoid_array, subspaces;
GE(data, d, s_dim)
1. projections ← getProj(data, s_dim);
2. semi_ellip ← ellip_k_means(projections,s_dim);
3. // process each semi_ellips
4. for each semi_ellip with size > 0
5. semi_ellip_data ← restoreData(semi_ellip);
6. semi_ellip ← getProj(semi_ellip_data, s_dim);
7. MPE ← getMPE(s_dim);
8. if MPE > MaxMPE and 2*s_dim > d
9. GE((data, d, 2*s_dim);
10. else
11. add semi_ellip_data into ellipsoid_array

**Dimensionality Optimization**
12. for each ellipsoid_array[i]
13. $d_r$ ← min(MaxDim, s_dim);
14. MPE ← getMPE($d_r$);
15. while change of MPE < threshold
16. $d_r$ −− ;
17. MPE ← getMPE($d_r$);
18. projections ← getProj(ellipsoid_array[i],$d_r$);
19. for each projection
20. ProjDist ← getProDist();
21. if ProjDist ≤ $\beta$
22. add it to this subspace;
23. else
24. add it to noise set

**Figure 4. MMDR Algorithm**

In *Generate Ellipsoid*, we recursively apply multi-level projections from low to high dimensionality until ellipsoids are fully discovered. At each level, Mahalanobis distance is applied to detect possible ellipsoids. Any unqualified ellipsoid is passed to *Generate Ellipsoid* with a higher subspace dimensionality so that more information can be used for clustering. We adopt this divide-lower-before-conquer-upper approach based on the following observations. First, in high dimensional space, some dimensions may contain little information, which may not be very helpful when it comes to identify the cluster membership. Second, for the well-separated clusters in the subspace, their correspondences in the higher dimensional space are usually well separated because of the property of PCA. In our algorithm, MPE indicates how much information is lost during the projection process. It is used as the parameter to determine if

the subspace projections carry enough information to reflect the shape of their correspondence in the original space.

The *Generate Ellipsoid* is invoked with a small subspace dimensionality - $s\_dim$. In line 1 of Figure 4 the low dimensional projections are produced from the original $d$-dimensional space, followed by elliptical k-means clustering in this low dimensional subspace, line 2. The data are then partitioned into semi-ellipsoids at $s\_dim$-dimensional subspace. We call this $semi-ellipsoid$ since we have not decided yet whether it properly indicates the shape of its correspondence in the original space. From line 3 to line 11, each semi-ellipsoid is handled individually. For each semi-ellipsoid discovered above, its corresponding shape is restored in the original dimensional space (line 5), and its local $s\_dim$-dimensional subspace is generated(line 6). The newly produced projections are local to individual semi-ellipsoid and different from the projections produced in line 1. At line 7, the MPE to $s\_dim$-dimensional subspace is computed.

If a semi-ellipsoid has smaller MPE than the maximum error allowed, it suggests that the $s\_dim$-dimensional subspace can approximately represent its original data. Otherwise, there are two possible reasons for the big MPE. First, it could be due to the overlap of several clusters in the subspace such that each point did not project to its local subspace. Higher subspace dimensionality should be retained in order to distinguish each cluster. Second, though it is a single cluster, the $s\_dim$ could be too small for a subspace to represent original dimensional data. To further discover ellipsoids in each semi-ellipsoid, we increase the $s\_dim$ twofold without losing generality and recursively call *Generate Ellipsoid* (line 9). Therefore, the semi-ellipsoid is repeatedly partitioned locally. This step produces possible ellipsoids. It should be noted that the process of discovering ellipsoids in the subspaces is the first step of dimensionality reduction, where the remained subspace dimensionality of each ellipsoid at this stage is their respective $s\_dim$, and further dimensionality optimization is performed in the next step.

Since the above step produces possible ellipsoids in their respective $s\_dim$-dimensional subspace, and ellipsoids are effective for dimensionality reduction, the $s\_dim$ of the subspaces discovered in *Generate Ellipsoid* can be further reduced (it should be noted that each ellipsoid may correspond to a different $s\_dim$ value). That is, if the change of MPE is less than the pre-set threshold, we decrease the dimensionality by 1 and the process is repeated till the above condition is false (line 15-17). The final dimensionality is treated as the 'optimal' one and denoted as $d_r$. The points are projected into this $d_r$-dimensional subspace (line 18). A threshold value $\beta$ is employed to determine whether a point belongs to a cluster. If the projection distance on $(d - d_r)$-dimensional eliminated subspace for a point is greater than $\beta$, this point is taken as an outlier (line 23-24). Otherwise, it is classified as a member of the subspace (line 21-22).

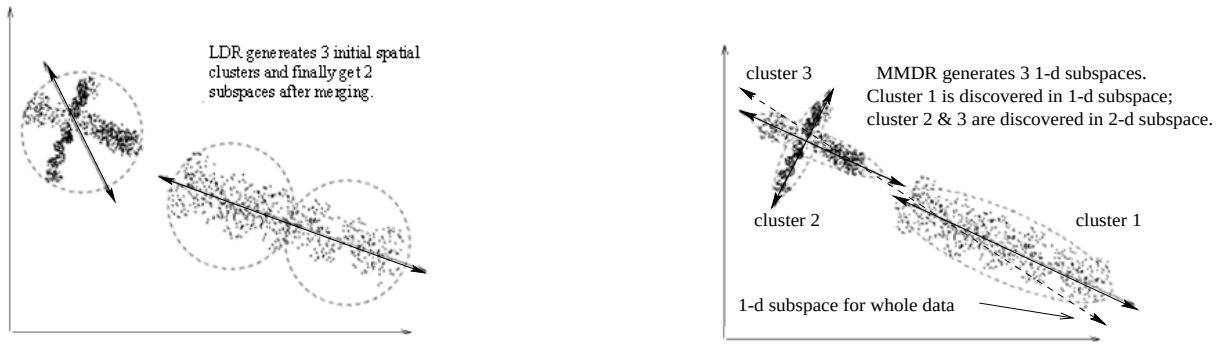
The final output of the algorithm is a set of subspaces and outliers. Each subspace may have a different optimal number of reduced dimensions. The outlier set remains in the original space since its data points are not well correlated.

In a dataset, some clusters are elongated along certain directions and yet they are locally correlated. Such elongation may be detected in its lower dimensional subspaces. Given a 2-dimensional subspace as shown in Figure 5 projected from a higher dimensional space (say 4-dimensional). This 2-dimensional subspace can represent the original dimensional space with very little information being lost. The LDR technique [5] is able to discover correlated clusters on the original 4-dimensional space and produces two 1-dimensional subspaces as shown in Figure 5a. In order to partition the bigger shape cluster for dimensionality reduction, the clustering radius must be suitably large. However, this will result in smaller clusters being grouped together as one. Obviously, substantial information is lost for the smaller shaped clusters. The situation is even worse for small shaped clusters with a high density.

Figure 5b shows three 1-dimensional subspaces produced by MMDR algorithm. MMDR first projects the original dimensional space into 1-dimensional subspace, then elliptical k-means method partitions 1-dimensional projections of whole data into two partitions: cluster 1's 1-dimensional subspace and cluster 2 and 3's 1-dimensional subspace. After restoring cluster 1's 1-dimensional subspace back to original dimensional space and performing local 1-dimensional projections (line 5-6), MMDR detects that it is an ellipsoid since its MPE is small. The 1-dimensional subspace projected from clusters 2 and 3 overlaps heavily with the high MPE and thus its corresponding full dimensional shape/data are passed to *Generate Ellipsoid* by increasing the subspace dimensionality to be 2-dimensional. At the 2-dimensional subspace, these 2 ellipsoids can be discovered by the Mahalanobis Function. Dimensionality reduction is further performed in *Dimensionality Optimization* so that both can be reduced to 1-dimensional subspaces with less information lost than using LDR method.

In summary, MMDR has the following advantages. First, the ellipsoids can be effectively discovered at data's subspace level, rather than at the original space. Second, the ellipsoids are able to be discovered as soon as the shapes can be identified. Third, the cost to perform clustering using Mahalanobis distance can be reduced dramatically since it is performed in the low dimensional subspace.

The cost of MMDR comes mainly from elliptical k-means method (line 2), which takes $O(Iter_{out} * Iter_{inn} * d\_sim^2 * N * MaxEC)$, where $Iter_{out}$ and $Iter_{inn}$ is the number of iterations for outer and inner loop respec-




(a) Clusters generated by LDR using Euclidean distance

(b) Clusters generated by MMDR using Mahalanobis distance

**Figure 5. LDR vs MMDR**

tively, $d\_sim^2$ comes from distance computation. However, in MMDR, the input dimensionality $s\_dim$ is very small compared to the original $d$. The input data size $N$ becomes smaller as $s\_dim$ increases, which leads to $Iter_{out}$ and $Iter_{inn}$ being reduced also. We further reduce the cost in the next subsection.

## 4.2 Optimization on Distance Computation

We note that the most time consuming step of the MMDR algorithm is the Mahalanobis distance computation between centroids and data points in elliptical k-means method. In this subsection, we reduce the computational cost by using the following techniques. The factor *MaxEC* can be reduced to a small number by avoiding computing all the distances between *MaxEC* centroids and a data point. We only need to re-compute the distance between the *k* most closest centroids which might change the membership of a point, where $k << MaxEC$. This is based on the following observations. First, if a data point is to be re-assigned to another cluster, that cluster is most probably the one with the closest distance except the current assignment. Second, in each iteration, only a small portion of data points might change their membership. As the converging process continues, the number of data points changing memberships decreases quickly. Third, some data points may never change their membership.

A lookup table is designed to store the *k* most closest centroids' IDs computed in the previous iteration for each data point. In the next iteration, only those centroids whose IDs are stored in the lookup table are taken to compute and find the closest centroid. A data point entry in the lookup table is updated only when its membership is changed. By doing so, the factor *MaxEC* is removed from the overall cost.

To further reduce the cost for large datasets, we introduce one additional field called $Activity$ to the lookup table to indicate how frequently a data point changes its membership. It records the number of iterations that a data point does not change its membership. If the value of $Activity$ is larger than a threshold, we say this data point is $inactive$, otherwise $active$. $Inactive$ data points need not make any further distance computation and re-assignment unless the number of clusters is changed. This reduces the value of $N$ dramatically at each iteration. Assume that at each iteration, only $\frac{1}{Iter_{inn}}$ of the dataset change their memberships, the factor of $N$ is replaced by $\frac{N}{Iter_{inn}}$. As the converging process continues, the number of points which change their membership decreases dramatically. Therefore, the time complexity becomes $O(Iter_{out} * d\_sim^2 * N)$. Comparing with the LDR's time complexity of $O(N * d^2 * MaxEC)$, MMDR has a smaller dimensionality of $d\_sim$ than $d$, but with a larger factor of $Iter_{out}$ than $MaxEC$.

## 4.3 Scalability for Large Datasets

For a very large dataset which cannot be completely loaded into the main memory buffer, the data scan at each iteration is extremely expensive. To make the MMDR scalable for very large datasets, we divide the dataset into a number of data streams, which is defined as a sequence of data points read in order of indices, and we process one data stream at a time. We set the size of a data stream to be $\varepsilon$ percent of the data size. A temporary array called Ellipsoid Array is created to store the ellipsoids' centroids generated for each data stream. Scalable MMDR loads a single data stream at a time and performs *Generate Ellipsoid* operation to generate small size ellipsoids. These small ellipsoids' centroids are stored in the Ellipsoid Array. After all the data streams have been processed, only the Ellipsoid Array is in the buffer. By calling *Generate Ellipsoid* on Ellipsoid

Array, Scalable MMDR forms bigger size of ellipsoids by merging smaller ellipsoids whose centroids are stored in Ellipsoid Array.

The size of data stream is much smaller than original data size N. Empirically, it is reasonable to expect that $Iter_{datastream} < Iter_{originaldata}$. Hence, the total time required to cluster $\frac{1}{\varepsilon}$ data streams of size $\varepsilon * N$ is generally less then the time required to cluster N data points.

## 5 Indexing Reduced Subspaces by Extended iDistance

After dimensionality reduction, the projections in reduced dimensionality subspaces have to be indexed using efficient indexes. Instead of using an index for each subspace, we want all the projections to be indexed in a single structure for ease of maintenance. We selected the iDistance [13] as our base index due to its efficiency and its $B^+$-tree base structure.

The design of iDistance was motivated by two factors. One, the triangular inequality relationships enable the (dis)similarity between a query point and a data point to be derived with reference to a chosen reference point. Two, data points can be ordered based on their distances to a reference point, and indexed based on such distance value. This enables one to represent high-dimensional data in a single dimensional space and use an existing $B^+$-tree. However, the iDistance has to be extended to index subspaces in different axis systems and handle dynamic insertion of data points.

The data partitioning strategy and reference point selection are straight forward, as the data partitions are determined by the MMDR algorithm and the centroid of each cluster is the ideal choice as the reference point. For each subspace (outliers as a subspace in its original dimensionality), all data points in subspaces are represented in a single dimensional space with reference to its centroid of cluster. This is achieved by the following mapping function:

$$y = i \times c + dist(P, O_i)$$

where the P is a data point in the subspace of $i^{th}$ ellipsoid $EC_i$, and $O_i$ is its centroid. $dist(P, O_i)$ is the distance function that returns distance between $O_i$ and P. y is the index key for P. c is some constant to stretch the data range so that distance values are range partitioned based on reference points. That is, it serves to partition the single dimension space into regions so that points in the $i^{th}$ cluster will be mapped to the range [i $\times$ c, (i+1) $\times$ c].

Extended iDistance employs three data structures:

- A $B^+$-tree is used to index the transformed single value points to facilitate speedy retrieval.
- An array is required to store the centroids and Principal Components of ellipsoids, and their respective nearest and farthest radius that define the subspace. This array is used for searching purpose.
- An array is required to store covariance matrices of ellipsoids, Mahalanobis radius, and the dimensionality retained. This array is used for the purpose of dynamic insertion (due to page limit, we omit the algorithm for dynamic insertion and its experiments).

To search for the $K$ nearest neighbors of a query point $q$, the distance of the $K$th nearest neighbor to $q$ defines the minimum radius required for retrieving the complete answer set. Such a distance cannot be predetermined, and hence, an iterative approach that examines increasingly larger sphere in each iteration has to be employed.

The algorithm works as follows. Given a query point $q$, finding $K$ nearest neighbors (NN) begins with a query sphere defined by a *relatively small* radius $R$ around $q$. For each cluster $EC_i$, the query point is mapped into $q_i$, which is the projection of $q$ on the $i^{th}$ subspace.

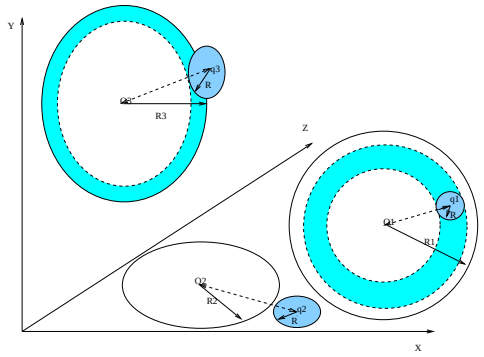



**Figure 6. Searching for NN queries $q_1$, $q_2$ and $q_3$**

Figure 6 shows an example with 3 clusters' max radius ranges in the different axis systems, where $EC_1$ is in XY plane, $EC_2$ in XZ plane and $EC_3$ in YZ plane. Here, for a query point $q$, its projection on 3 subspaces are $q_1$, $q_2$, and $q_3$ respectively. The shaded regions are the areas that need to be checked.

Searching in extended iDistance begins by scanning the auxiliary structure to identify the centroids whose data space (sphere area of cluster) overlaps with the query sphere defined by $q_i$ and $R$. The search starts with a small global radius $R$ for all subspaces, and step by step, the radius is increased to form a bigger query sphere. For each enlargement, there are three main cases to consider.

1. The data space $EC_i$ contains $q_i$. In this case, we want to traverse the data space sufficiently to determine the

$K$ nearest neighbors. This is done by first locating the leaf node where $q_i$ may be stored. Since this node does not necessarily contain points whose distance are closest to $q_i$ compared to its sibling nodes, we need to search left and right (inward and outward of data space) from the reference point accordingly. This situation is illustrated by the subspace $EC_1$ and $q_1$.

2. The data space intersects the query sphere. In this case, we only need to search leftward (inward) since the query point is outside the data space. This situation is illustrated by the subspace $EC_3$ and $q_3$.

3. The data space does not intersect the query sphere. Here, we do not need to examine the data space. This situation is illustrated by the subspace $EC_2$ and $q_2$.

The search stops when the distance of the $K^{th}$ NN object to $q$ is less than search radius $R$. The search is correct as the distance between image query point and data point always lower bounds the actual distance between the actual query point and data point in the original space. The searching subspace can be fast pruned by using triangle inequality property.

$$\|Q-P\| \geq \|Q_j - P_j\| \geq \|Q_j - O_j\| - \|P_j - O_j\| \geq \|Q_j - O_j\| - R_j$$

Where Q is query, P is original data point, $Q_j$ is the projection in $j^{th}$ subspace, $P_j$ is the projection of P in $j^{th}$ subspace, $O_j$ is the reference point in $j^{th}$ subspace, and $R_j$ is the max radius in $j^{th}$ subspace. $\|Q_j - O_j\| - R_j$ specifies the tightest searching bound for $j^{th}$ subspace.

# 6 Performance Study

In this section, we present the performance study to evaluate the effectiveness of MMDR and the efficiency of extended iDistance. For the experiments, we use the default values as shown in table 1, and all experiments were done with Ultra-10 SunOS 5.7 processor (333 MHz CPU and 256 MB RAM).

We have two categories of test data.

1. Real life datasets: The real life dataset consists of 64-dimensional color histogram extracted from 70,000 color images from Corel Database, used in LDR[5].

2. Synthetic datasets: We have four sets of synthetic datasets. One small synthetic dataset contains 100,000 points in 64-dimensional space. Three large synthetic datasets with 1,000,000 points are in 50-, 100-, and 200-dimensional spaces respectively. For each synthetic dataset, we use the algorithm (Appendix A) to generate correlated clusters in different subspaces with different distensibilities. Each subspace has different size, orientation and ellipticity.

We used 100 queries to obtain the mean precision and query cost on 10NN, and $L_2$ distance was used for searching (Note that the Mahalanobis distance is used for discovering intrinsic ellipsoids, not for searching). The query precision is defined as follows:

$$Precision = \frac{R_{d_r} \bigcap R_d}{R_d}$$

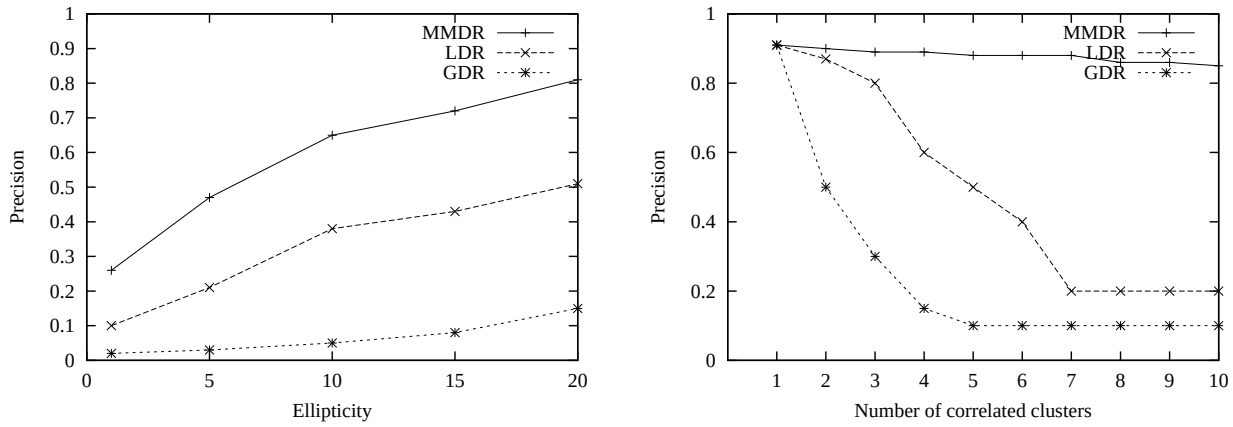
where $R_d$ and $R_{d_r}$ are the results respectively returned from the original space and reduced subspaces.

## 6.1 Query Precision

Here dimensionality reduction methods serve for the purpose of efficient indexing. However, they are lossy in nature. When a dimensionality method tries to reduce more, it may cause bigger loss of information and hence query precision. The query precision is also affected by correlation between data points and number of correlated clusters. Here we use the small dataset with 100,000 points.

Figure 7a shows the query precision with respect to increasing ellipticity. As we can see, the MMDR method performs much better than the LDR and GDR methods. The GDR method can achieve at most 15% of precision as the dataset is not globally correlated. As ellipticity decreases, LDR drops faster than MMDR. Obviously, less correlation has more negative effect on the query precision of LDR then MMDR. In the next experiment, we varied the number of correlated clusters to test its effect on query precision. The results in Figure 7b show that all MMDR, LDR and GDR perform equally well when there is only one correlated cluster. But as the number of correlated clusters increases, the MMDR is able to locate all correlated clusters effectively and maintains its query precision. However, the query precision of the LDR drops rapidly, and so does the GDR method. It indicates that when clusters intersect and have different ellipticities and scales, LDR can not discover all of them. As more such clusters exist, LDR performs worse. In contrast, the MMDR can discover the intrinsic number of correlated cluster based on Mahalanobis distance and thus is independent of the number of correlated cluster.

To see the effect of the number of eliminated dimensions on the effect of query precision, we conducted experiments using the small synthetic dataset and color histogram dataset. In this experiment, we set the maximum remained subspace dimensionality $MaxDim$ to be 20. Figure 8 presents the effect of the number of dimensions retained after dimensionality reduction on the query precision on two datasets. All the three methods show increasing precision as the remained dimensionality increases for three datasets. MMDR achieves much higher precision. As




(a) Effect of ellipticity

(b) Effect of number of clusters

**Figure 7. Effect on precision**

shown in Figure 8a for the synthetic dataset, at 20 dimensions, LDR only can achieve at most 60% of precision. and GDR cannot achieve more than 25% of precision due to uncorrelated property. Figure 8b shows the effect of retained dimensionality on the query precision using the color histogram dataset. It is interesting to note all three methods are not performing as well as before. Nevertheless, the MMDR method performs the best and is least affected. The higher precision obtained by the MMDR method confirms two important observations. First, there exist some local elongated clusters. Second, some intrinsic local elongated $shapes/correlations$ cannot be detected by LDR. Compare to the synthetic dataset (Figure 8a), the precision of the methods on color histogram dataset are much worse. One reason could be that the real dataset may have clusters that are highly uncorrelated. Too many outliers may be another reason. This is possible, as for each image in the real dataset, the color histograms tend to be very skewed towards a small set of colors, with many attributes being 0.

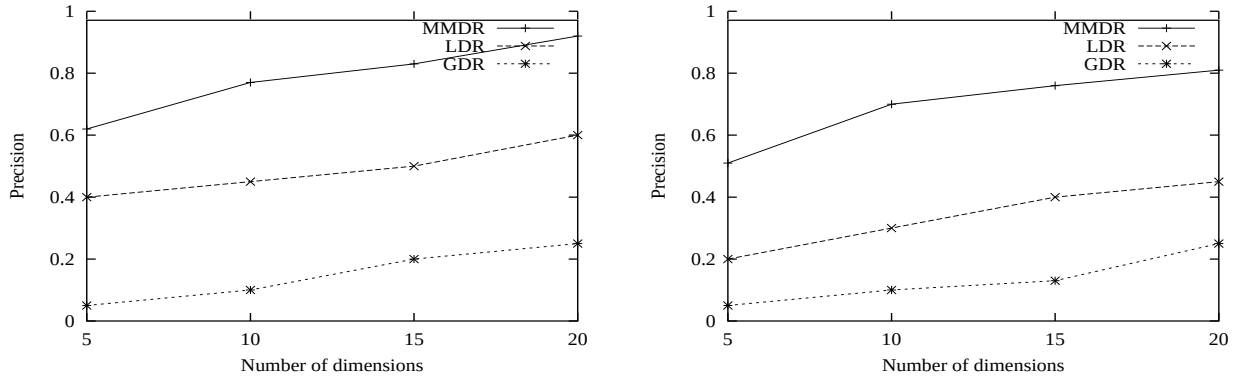
The above experiments confirm that the MMDR method is a much more effective dimensionality reduction technique in correlated environments with lower loss of distance information, as it can achieve the better reduction performance with higher precision, which should lead to faster searching and retrieval.

## 6.2 Query Efficiency

In this experiment, we examine the query performance of the index methods on reduced dimensionality data points. Note that the final purpose of performing effective dimensionality reduction by using MMDR is to improve the query performance, as it is well known that existing multi-dimensional indexing structures are not be able to index very high (30 or greater) dimensional data space. Here we have three indexing schemes to compare: extended iDistance on MMDR data (iMMDR), extended iDistance on LDR data (iLDR) and Global indexing method [5] on LDR data (gLDR). The Global indexing method makes use of one Hybrid tree [4] for each cluster, and maintains the information about each cluster and index in an array. Here we use the same datasets as in the last sub section.

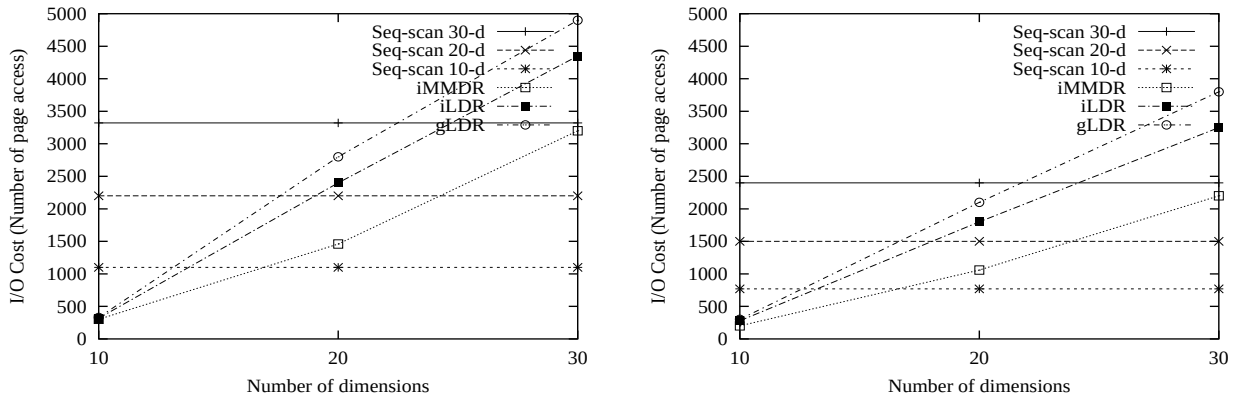
Figure 9 shows the I/O cost for three indexing schemes, and sequential scan in reduced subspaces, when the subspace dimensionality varies from 10 to 30. Figure 9a shows that for synthetic dataset, as the dimensionality increases, the iMMDR has much lower I/O cost than the iLDR, which confirms that *a more effective dimensionality reduction method leads to an overall improved query efficiency*. We also notice that the gLDR is worse than the iLDR, and when the dimensionality reaches 20, its cost is higher than that of direct sequential scan. The extended iDistance is more efficient in terms of I/O cost as it has to traverse only one index, and this index is smaller since only the 1-dimensional distance values are used in the internal nodes. Figure 9b shows the similar trends for color histogram.

Figure 10 provides the CPU cost of three indexing schemes for three datasets. From Figure 10a, we can see that as the dimensionality increases, the gap becomes wider between iLDR and gLDR. iMMDR is the best. Performance difference between iMMDR and iLDR is relatively small. When the dimensionality reaches 30, the CPU cost for gLDR is an order of magnitude higher than that for iMMDR and iLDR. The main reason is clear. In gLDR indexing structure, tree nodes contain multi-dimensional data



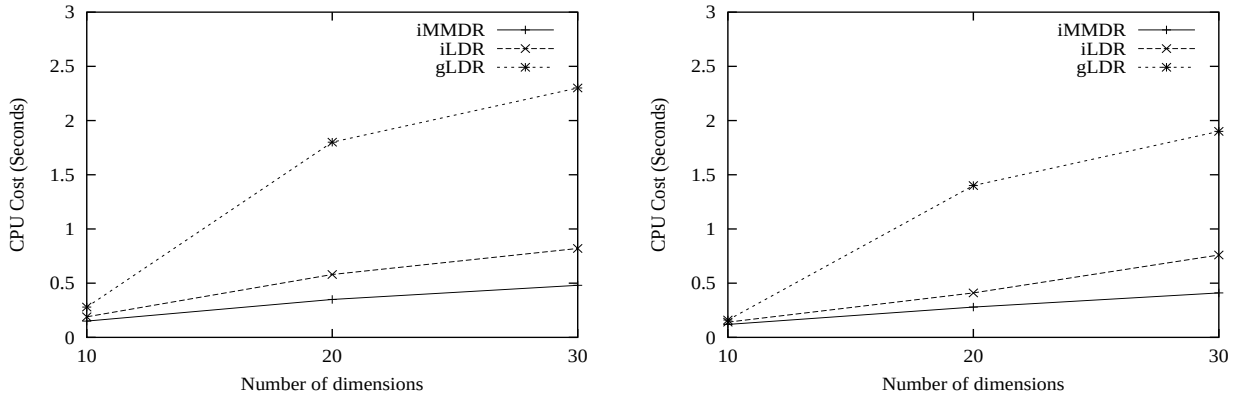

(a) Synthetic data (b) Color histogram

**Figure 8. Effect of dimensionality on query precision**



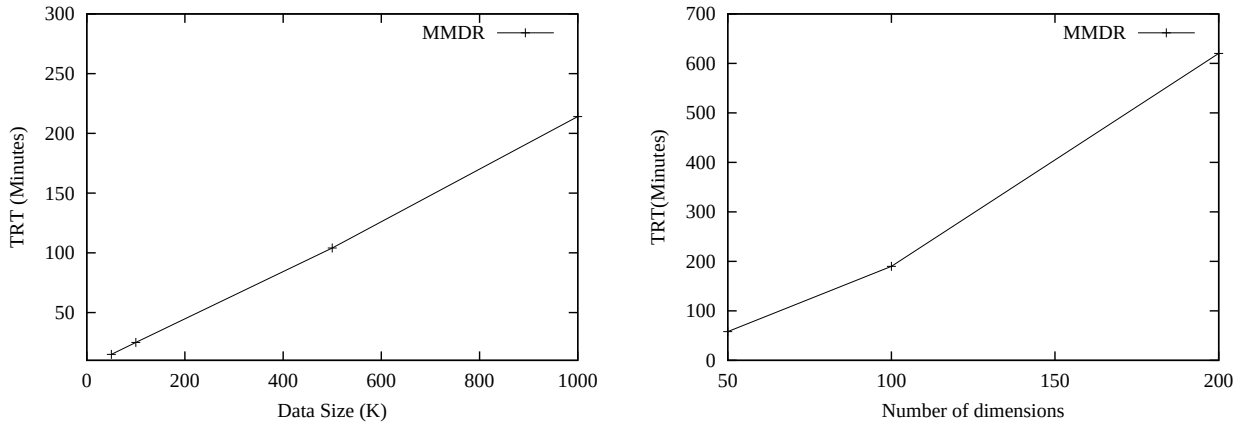

(a) Synthetic data (b) Color histogram

**Figure 9. Effect of dimensionality on I/O cost**




(a) Synthetic data (b) Color histogram

**Figure 10. Effect of dimensionality on CPU cost**




(a) Effect of data size

(b) Effect of dimensionality

**Figure 11. Effect on total response time**

points. However, in extended iDistance structure, tree nodes contain 1-dimensional key values. Extended iDistances (iMMDR and iLDR) incur single dimensional value comparison in searching while L-norm computation is involved in the Global structure's Hybrid-Tree. Thus computation in gLDR is much more expensive. Again, Figure 10b for color histogram dataset shows the similar trend as Figure 10a. In terms of both CPU and I/O cost, the single dimensional extended iDistance index outperforms the Global indexing structure significantly. Furthermore, more effective dimensionality reduction method leads to more efficient indexing.

### 6.3 Scalability

All high-dimensional indexes are affected by the data size and number of dimensions. In this experiment, we look at scalability of MMDR. We set the data stream ratio $\varepsilon$ as 0.005, and the $k$ value in the lookup table to be 3 and the number of iterations that indicates a point as $inactive$ as 10. The parameter we used here is the total response time (TRT) for MMDR to generate the optimal subspaces from the original data.

Figure 11a describes the effect of data size on the total response time. We keep the number of dimensions fixed at 100, while we vary the data size from 50,000 to 1,000,000. From Figure 11a we make the following observation. The response time increases linearly to the data size. When the data size reaches the limit of buffer - 500K, there is no jump in response time for scalable MMDR since we need only scan the whole dataset once. Figure 11b shows the effect of the number of dimensions on the total response time. For this experiment, we used 1,000,000 data points and varied the number of dimensions from 50 to 200. As expected, the total response time is nearly quadratic to the dimensionality. The results again exhibit that the limited buffer has no effect on the total response time.

## 7 Conclusions

In this paper, we have presented an effective and fast dimensionality reduction algorithm – Multi-level Mahalanobis-based Dimensionality Reduction, which is able to reduce the number of dimensions while keeping the precision high, and able to effectively handle large datasets. We used an extended iDistance to index the data points in different reduced subspaces. We conducted extensive experimental studies using both real and synthetic datasets to compare the algorithm with existing approaches. The results show that the proposed technique, as a whole, is very effective and efficient in supporting KNN search in very high-dimensional space. Furthermore, it is scalable for very large databases.

**Acknowledgment**

We would like to thank Kaushik Chakrabarti for providing us the source codes of LDR and Hybrid-tree.

## References

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Spaces. In *ICDT*, pages 420–434, 2001.

[2] C.C. Aggarwal, J.L. Wolf, P.S. Yu, C. Procopiuc, and J. S. Park. Fast Algorithms for Projected Clustering. In *SIGMOD*, pages 61–72, 1999.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful? In *ICDT*, 1999.

[4] K. Chakrabarti and S. Mehrotra. The Hybrid Tree: An Index Structure for High Dimensional Feature Spaces. In *ICDE*, pages 322–331, 1999.

[5] K. Chakrabarti and S. Mehrotra. Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces. In *VLDB*, pages 89–100, 2000.

[6] R. Duda. Pattern Recognition for HCI. In *http://www.engr.sjsu.edu/k̃napp/*.

[7] A. Hinneburg and D.A. Keim. An Optimal Grid-Clustering: Towards Breaking the Curse of Diminsionality in High Dimensional Clustering. In *VLDB*, 1999.

[8] I.T. Jolliffe. *Principle Componet Analysis*. Springer-Verlag, 1986.

[9] J.H. Lee, D.H. Kim, and C.W. Chung. Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. In *SIGMOD*, pages 205–214, 1999.

[10] B.C. Ooi, K.L. Tan, C. Yu, and S. Bressan. Indexing the Edges - A Simple and Yet Efficient Approach to High-Dimensional Indexing. In *PODS*, pages 166–174, 2000.

[11] K.K. Sung and T. Poggio. Example-Based Learning for View-Based Human Face Detection. In *PAMI*, pages 20(1):39–51, 1998.

[12] J.S. Vitter and M. Wang. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In *SIGMOD*, pages 193–204, 1999.

[13] C. Yu. *High-dimensional indexing*. Lecture Notes in Computer Science 2341, Springer-Verlag, 2002.

[14] C. Yu, B.C. Ooi, K.L. Tan, and H. V. Jagadish. Indexing the Distance: An Efficient Method to KNN Processing. In *VLDB*, 2001.

**Appendix A: Generate Synthetic Datasets**

In order to generate the local correlated datasets, we use the algorithm outlined in Figure 12 to generate different clusters in different subspaces with different orientations and distensibilities retained.

In this algorithm, array s_dim[i] contains the dimensions which should be remained for each cluster.We can randomly choose which dimension should be retained. For simplicity, we make remained dimensions continuous starting with s_r_dim[i]. For example, if s_r_dim[i]=6, then the remained dimensions for $i^{th}$ cluster starts from $6^{th}$ to $(6 + s\_dim)^{th}$ dimensions. Specifying the different values for each cluster allows each reduced subspace in different axis systems. Method gen_float() will return a random float value in [lb,lb+variance]. It can also return a value based on other distribution functions, such as Zipfian. For each cluster, we also specify their different lower bound values, which can be used to control the positions of centers of each cluster together with its variance. Along each of the remaining s_dim[i] dimensions, we assign a randomly chosen value falling in range of [ lb[i], lb[i]+variance_r[i]] to all the points in the cluster. Along each of the reduced (d-s_dim[i]) dimensions, we assign a randomly chosen value falling in range of [lb[i], lb[i]+variance_e[i]] to all the points in the cluster. The ratio between variance_r[i] and variance_e[i] in fact specifies the ratio between the energy carried by remained and reduced dimensions for each cluster, or the degree of correlation/ellipticity. Both values can be adjusted for different clusters in order to have different level of correlation. To make the subspace arbitrarily oriented, we can generate a random orthonormal rotation matrix (generated using MATLAB) and rotate the cluster by multiplying the data matrix with the rotation matrix.

**Generate Correlated Dataset (GCD)**

input: N,d,EC,EC_size[EC],s_dim[EC],s_r_dim[EC],
variance_e[EC],variance_r[EC], lb[EC]
Output: datasets[EC]
Algorithm:

```
1.  for i from 0 to EC-1 do
2.          for j from 0 to EC_size[i]-1 do
3.          for k from 0 to s_remained_dim[i]-1
4.              datasets[i][j*d+k]= gen_float(lb[i], variance_e[i])
5.          for k from s_r_dim[i] to s_r_dim[i]+s_dim[i]-1
6.              datasets[i][j*d+k]= gen_float(lb[i], variance_r[i])
7.          for k from s_r_dim[i]+s_dim[i] to d-1
8.              datasets[i][j*d+k]= gen_float(lb[i], variance_e[i])
9.  rotate datasets[i] to be arbitrarily oriented
```

**Figure 12. Synthetic Datasets Generation**