Mining Projected Clusters in High-Dimensional Spaces

Mohamed Bouguessa and Shergrui Wang

The authors are with the Department of Computer Science, University of Sherbrooke, Sherbrooke, Quebec, J1K 2R1, Canada. E-mail:{mohamed.bouguessa, shengrui.wang}@usherbrooke.ca.

Abstract

Clustering high-dimensional data has been a major challenge due to the inherent sparsity of the points. Most existing clustering algorithms become substantially inefficient if the required similarity measure is computed between data points in the full-dimensional space. To address this problem, a number of projected clustering algorithms have been proposed. However, most of them encounter difficulties when clusters hide in subspaces with very low dimensionality. These challenges motivate our effort to propose a robust partitional distance-based projected clustering algorithm. The algorithm consists of three phases. The first phase performs attribute relevance analysis by detecting dense regions in each attribute, thereby allowing irrelevant attributes to be identified. Starting from the results of the first phase, the goal of the second phase is to eliminate outliers, while the third phase aims to discover clusters in different subspaces. The clustering process is based on the K-means algorithm, with the computation of distance restricted to subsets of attributes where object values are dense. Our algorithm is capable of detecting projected clusters of low dimensionality embedded in a high-dimensional space and avoids the computation of the distance in the full-dimensional space. The suitability of our proposal has been demonstrated through an empirical study using synthetic and real datasets.

Index Terms

Data mining, clustering, high dimensions, projected clustering.

I. INTRODUCTION

Data mining is the process of extracting potentially useful information from a dataset [1]. Clustering is a popular data mining technique which is intended to help the user discover and understand the structure or grouping of the data in the set according to a certain similarity measure [2]. Clustering algorithms usually employ a distance metric (e.g., Euclidean) or a similarity measure in order to partition the database so that the data points in each partition are more similar than points in different partitions.

The commonly used Euclidean distance, while computationally simple, requires similar objects to have close values in all dimensions. However, with the high-dimensional data commonly encountered nowadays, the concept of similarity between objects in the full-dimensional space is often invalid and generally not helpful. Recent theoretical results [3] reveal that in high-dimensional data, the distance between any two data points becomes almost the same, making it difficult to differentiate similar data points from dissimilar ones. These results explain the poor

performance of conventional distance-based clustering algorithms on such data sets.

Feature selection techniques are commonly utilized as a preprocessing stage for clustering, in order to overcome the curse of dimensionality. The most informative dimensions are selected by eliminating irrelevant and redundant ones. Such techniques speed up clustering algorithms and improve their performance [4]. Nevertheless, in some applications, different clusters may exist in different subspaces spanned by different dimensions. In such cases, dimension reduction using a conventional feature selection technique may lead to substantial information loss [5].

The following example provides an idea of the difficulties encountered by conventional clustering algorithms and feature selection techniques. Figure 1 illustrates a generated data set composed of 1000 data points in 10-dimensional space, with four clusters that have their own relevant dimensions (e.g., cluster 1 exists in dimensions A_1, A_4, A_8, A_{10}). By relevant dimensions, we mean dimensions that exhibit cluster structure. For each relevant dimension of a cluster, points in the cluster are distributed according to a normal distribution, while in the remaining dimensions, the points are distributed sparsely. In our example, there are also three irrelevant dimensions A_3, A_5 and A_7 in which all the data points are sparsely distributed, i.e. no cluster structure exist in these dimensions.

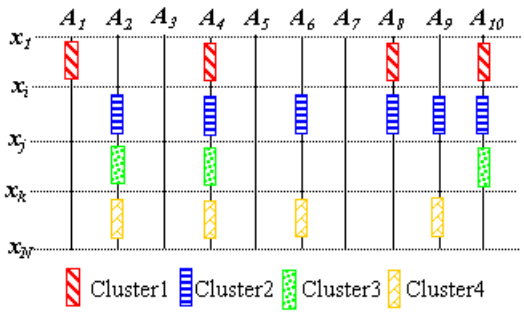

Fig. 1. Example of dataset containing projected clusters.

For such an example, a traditional clustering algorithm is likely to fail to find the four clusters. This is because the distance function used by these algorithms gives equal treatment to all dimensions, which are, however, not of equal importance. While feature selection techniques can reduce the dimensionality of the data by eliminating irrelevant attributes such as A_3, A_5 and A_7, there is an enormous risk that they will also eliminate relevant attributes such as A_1. This

is due to the presence of many sparse data points in A_1, where a cluster is in fact present. To cope with this problem, new classes of projected clustering have emerged.

Projected clustering exploits the fact that in high-dimensional datasets, different groups of data points may be correlated along different sets of dimensions. The clusters produced by such algorithms are called "projected clusters". A projected cluster is a subset SP of data points, together with a subspace SD of dimensions, such that the points in SP are closely clustered in SD [5]. For instance, the third cluster in the dataset presented in Figure 1 is $(SP_3, SD_3) = (\{x_j, \ldots, x_k\},\ \{A_2, A_4, A_{10}\})$. Recent research has suggested the presence of projected clusters in many real-life datasets [6].

A number of projected clustering algorithms have been proposed in recent years. Although these previous algorithms have been successful in discovering clusters in different subspaces, they encounter difficulties in identifying very low-dimensional projected clusters embedded in high-dimensional space. Yip et al. [7] observed that current projected clustering algorithms provide meaningful results only when the dimensionalities of the clusters are not much lower than that of the dataset. For instance, some partitional projected clustering algorithms, such as PROCLUS [5] and ORCLUS [8], make use of a similarity function that involves all dimensions in order to find an initial approximation of the clusters. After that, relevant dimensions of each cluster are determined using some heuristics and the clustering is refined based on the relevant dimensions previously selected. Here, it is clear that a similarity function that uses all dimensions misleads the relevant dimensions detection mechanism and adversely affect the performance of these algorithms. Another example is HARP [9], a hierarchical projected clustering algorithm based on the assumption that two data points are likely to belong to the same cluster if they are very similar to each other along many dimensions. However, when the number of relevant dimensions per cluster is much lower than the dataset dimensionality, such an assumption may not be valid. In addition, some existing projected clustering algorithms, such as PROCLUS [5] and ORCLUS [8], require the user to provide the average dimensionality of the subspaces, which is very difficult to establish in real-life applications.

These observations motivate our effort to propose a novel projected clustering algorithm, called PCKA (Projected Clustering based on the K-means Algorithm). PCKA is composed of three phases: attribute relevance analysis, outlier handling and discovery of projected clusters. Our algorithm is partitional in nature and able to automatically detect projected clusters of very

low dimensionality embedded in high-dimensional space, thereby avoiding computation of the distance in the full-dimensional space.

The remainder of this paper is organized as follows. In Section 2, we provide a brief overview of recent projected clustering algorithms and discuss their strengths and weaknesses. Section 3 describes our projected clustering algorithm in detail. Section 4 presents experiments and performance results on a number of synthetic and real datasets. Our conclusion is given in Section 5.

II. Related Work

The problem of finding projected clusters has been addressed in [5]. The partitional algorithm PROCLUS, which is a variant of the K-medoid method, iteratively computes a good medoid for each cluster. With the set of medoids, PROCLUS finds the subspace dimensions for each cluster by examining the neighboring locality of the space near it. After the subspace has been determined, each data point is assigned to the cluster of the nearest medoid. The algorithm is run until the sum of intracluster distances ceases to change. ORCLUS [8] is an extended version of PROCLUS that looks for non-axis-parallel clusters, by using Singular Value Decomposition (SVD) to transform the data to a new coordinate system and select principal components. PROCLUS and ORCLUS were the first to successfully introduce a methodology for discovering projected clusters in high-dimensional spaces, and they continue to inspire novel approaches.

A limitation of these two approaches is that the process of forming the locality is based on the full dimensionality of the space. However, it is not useful to look for neighbors in datasets with very low-dimensional projected clusters. In addition, PROCLUS and ORCLUS require the user to provide the average dimensionality of the subspace, which also is very difficult to do in real life applications.

In [10], Procopiuc et al. propose an approach called DOC (Density-based Optimal projective Clustering) in order to identify projected clusters. DOC proceeds by discovering clusters one after another, defining a projected cluster as a hypercube with width $2w$, where w is a user-supplied parameter. In order to identify relevant dimensions for each cluster, the algorithm randomly selects a seed point and a small set, Y, of neighboring data points from the dataset. A dimension is considered as relevant to the cluster if and only if the distance between the projected value of the seed point and the data point in Y on the dimension is no more than w. All data points

that belong to the defined hypercube form a candidate cluster. The suitability of the resulting cluster is evaluated by a quality function which is based on a user-provided parameter β that controls the trade-off between the number of objects and the number of relevant dimensions. DOC tries different seeds and neighboring data points, in order to find the cluster that optimizes the quality function. The entire process is repeated to find other projected clusters. It is clear that since DOC scans the entire dataset repetitively, its execution time is very high. To alleviate this problem, an improved version of DOC called FastDOC is also proposed in [10].

DOC is based on an interesting theoretical foundation and has been successfully applied to image processing applications [10]. In contrast to previous approaches, (i.e. PROCLUS and ORCLUS), DOC is able to automatically discover the number of clusters in the dataset. However, the input parameters of DOC are difficult to determine and an inappropriate choice by the user can greatly diminish its accuracy. Furthermore, DOC looks for clusters with equal width along all relevant dimensions. In some types of data, however, clusters with different widths are more realistic.

Another hypercube approach called FPC (Frequent-Pattern-based Clustering) is proposed in [11] to improve the efficiency of DOC. FPC replaces the randomized module of DOC with systematic search for the best cluster defined by a random medoid point p. In order to discover relevant dimensions for the medoid p, an optimized adaptation of the frequent pattern tree growth method used for mining itemsets is proposed. In this context, the authors of FPC illustrate the analogy between mining frequent itemsets and discovering dense projected clusters around random points. The adapted mining technique is combined with FastDOC to discover clusters. However, the fact that FPC returns only one cluster at a time adversely affects its computational efficiency. In order to speed up FPC, an extended version named CFPC (Concurrent Frequent-Pattern-based Clustering) is also proposed in [11]. CFPC can discover multiple clusters simultaneously, which improves the efficiency of the clustering process.

It is shown in [11] that FPC significantly improves the efficiency of DOC/FastDOC and can be much faster than the previous approaches. However, since FPC is built on DOC/FastDOC it inherits some of their drawbacks. FPC performs well only when each cluster is in the form of a hypercube and the parameter values are specified correctly.

A recent paper [9] proposes a hierarchical projected clustering algorithm called HARP (a Hierarchical approach with Automatic Relevant dimension selection for Projected clustering).

The basic assumption of HARP is that if two data points are similar in high-dimensional space, they have a high probability of belonging to the same cluster in lower-dimensional space. Based on this assumption, two clusters are allowed to merge only if they are similar enough in a number of dimensions. The minimum similarity and minimum number of similar dimensions are dynamically controlled by two thresholds, without the assistance of user parameters. The advantage of HARP is that it provides a mechanism to automatically determine relevant dimensions for each cluster and avoid the use of input parameters, whose values are difficult to set. In addition to this, the study in [6] illustrates that HARP provides interesting results on gene expression data.

On the other hand, it has been shown in [3] that as dimensionality increases, the distance to the nearest data point approaches the distance to the farthest data point. Based on these results, the basic assumption of HARP will be less valid when projected clusters have few relevant dimensions. In such situations the accuracy of HARP deteriorates severely. This effect on HARP's performance was also observed by Yip et al. in [7].

In order to overcome the limitation encountered by HARP and other projected clustering algorithms, the authors of HARP propose in [7] a semi-supervised approach named SSPC (Semi-Supervised Projected Clustering). This algorithm is partitional in nature and similar in structure to PROCLUS. As in semi-supervised clustering, SSPC makes use of domain knowledge (labeled data points and/or labeled dimensions) in order to improve the quality of a clustering. As reported in [7], the clustering accuracy can be greatly improved by inputting only a small amount of domain knowledge. However, in some applications, domain knowledge in the form of labeled data points and/or labeled dimensions is very limited and not usually available.

A density-based algorithm named EPCH (Efficient Projective Clustering by Histograms) is proposed in [12] for projected clustering. EPCH performs projected clustering by histogram construction. By iteratively lowering a threshold, dense regions are identified in each histogram. A "signature" is generated for each data point corresponding to some region in some subspace. Projected clusters are uncovered by identifying signatures with a large number of data points [12]. EPCH has an interesting property in that no assumption is made about the number of clusters or the dimensionality of subspaces. In addition to this, it has been shown in [12] that EPCH can be fast and is able to handle clusters of irregular shape. On the other hand, while EPCH avoids the computation of distance between data points in the full-dimensional space, it suffers from the curse of dimensionality. In our experiments [13], we have observed that when

the dimensionality of the data space increases and the number of relevant dimensions for clusters decreases, the accuracy of EPCH is affected.

A field that is closely related to projected clustering is subspace clustering. CLIQUE [1] was the pioneering approach to subspace clustering, followed by a number of algorithms in the same field such as ENCLUS [14], MAFIA [15] and SUBCLU [16]. The idea behind subspace clustering is to identify all dense regions in all subspaces, whereas in projected clustering, as the name implies, the main focus is on discovering clusters that are projected onto particular spaces [5]. The outputs of subspace clustering algorithms differ significantly from those of projected clustering [5]. Subspace clustering techniques tend to produce a partition of the dataset with overlapping clusters [5][9]. The output of such algorithms is very large, because data points may be assigned to multiple clusters. In contrast, projected clustering algorithms produce disjoint clusters with a single partitioning of points [5][9][10][11][12]. Depending on the application domain, both subspace clustering and projected clustering can be powerful tools for mining high-dimensional data. Since the major concern of this paper is projected clustering, we will focus only on such techniques. Further details and a survey on subspace clustering algorithms and projected clustering algorithms can be found in [17] and [18].

III. THE ALGORITHM PCKA

Let DB be a dataset of d-dimensional points, where the set of attributes is denoted by $A = \{A_1, A_2, \ldots, A_d\}$. Let $X = \{x_1, x_2, \ldots, x_N\}$ be the set of N data points, where $x_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{id})$. Each x_{ij} $(i = 1, \ldots, N; j = 1, \ldots, d)$ corresponds to the value of data point x_i on attribute A_j. In what follows, we will call x_{ij} a 1-d point. We assume that each data point x_i belongs either to one projected cluster or to the set of outliers. Given the number of clusters nc, which is an input parameter, a projected cluster C_s, $s = 1, \ldots, nc$, containing N_s data points is defined in a d_s-dimensional subspace formed by the set A_s, where $(A_s \subseteq A)$, of its relevant attributes. The remaining set $A - A_s$ represents the irrelevant attributes of C_s. The main focus of the approach that we propose in this paper is to discover low-dimensional projected clusters which are parallel to axes. To achieve this PCKA proceeds in three phases:
1- Attribute relevance analysis: The goal is to identify attributes which exhibit some cluster structure by discovering dense regions and their location in each attribute. Such attributes are considered as relevant to the clustering process since they contain the projected values of one

or more clusters.

2- Outlier handling: Based on the results of the first phase, the aim is to identify and eliminate outlier points from the dataset.

3- Discovery of projected clusters: The goal of this phase is to identify clusters and their relevant dimensions. The clustering process is based on the K-means algorithm and the computation of distance is restricted to subsets where the data point values are dense.

The algorithms of these phases are described in the following subsections.

A. Attribute Relevance Analysis

In the context of projected clustering, irrelevant attributes contain noise/outliers and sparse data points, while relevant ones may exhibit some cluster structure [5]. By cluster structure we mean a region that has a higher density of points than its surrounding regions. Such dense region represents the 1-d projection of some cluster. Hence, it is clear that by detecting dense regions in each dimension we are able to discriminate between dimensions that are relevant to clusters and irrelevant ones.

In order to detect densely populated regions in each attribute we compute a sparseness degree y_{ij} for each 1-d point x_{ij} by measuring the variance of its k nearest (1-d point) neighbors $(k-nn)$.

Definition 1. *The sparseness degree of x_{ij} is defined as $y_{ij} = \frac{\sum_{r \in p_i^j(x_{ij})} (r - c_i^j)^2}{k+1}$, where $p_i^j(x_{ij}) = \{nn_k^j(x_{ij}) \cup x_{ij}\}$ and $|p_i^j(x_{ij})| = k+1$. nn_k^j denotes the set of $k-nn$ of x_{ij} in dimension A_j and c_i^j is the center of the set $p_i^j(x_{ij})$, i.e., $c_i^j = \frac{\sum_{r \in p_i^j(x_{ij})} r}{k+1}$.*

Intuitively, a large value of y_{ij} means that x_{ij} belongs to a sparse region, while a small one indicates that x_{ij} belongs to a dense region.

Calculation of the k nearest neighbors is, in general, an expensive task, especially when the number of data points N is very large. However, since we are searching for the k nearest neighbors in a one-dimensional space, we can perform the task in an efficient way by pre-sorting the values in each attribute and limiting the number of distance comparisons to a maximum of $2k$ values.

The major advantage of using the sparseness degree is that it provides a relative measure on which the dense regions are more easily distinguishable from sparse regions. On the other hand, when a dimension contains only sparse regions, all the estimated y_{ij} for the same dimension

tend to be very large. Our objective now is to determine whether or not dense regions are present in a dimension.

In order to identify dense regions in each dimension, we are interested in all sets of x_{ij} having a small sparseness degree. In the preliminary version of PCKA described in [13], dense regions are distinguished from sparse regions using a user pre-defined density threshold. However, in such an approach an inappropriate choice of the value of the density threshold by the user may affect the clustering accuracy. In this paper, we develop a systematic and efficient way to discriminate between dense and sparse regions. For this purpose we propose to model the sparseness degree y_{ij} of all the 1-d points y_{ij} in a dimension as a mixture distribution. The probability density function (*PDF*) is therefore estimated and the characteristics of each dimension are identified.

1) PDF *estimation:* Since we are dealing with one-dimensional spaces, estimating the histogram is a flexible tool to describe some statistical properties of the data. For the purpose of clarification, consider the dataset presented in Figure 1. The histograms of the sparseness degrees calculated for dimensions A_1, A_2, A_3 and A_4 are illustrated in Figure 2. Here k is chosen to be $\sqrt{N}$ and the calculated y_{ij} are normalized in the interval $]0, 1]$. The histograms presented in Figure 2 suggest the existence of components with different shape and/or heavy tails, which inspired us to use the gamma mixture model.

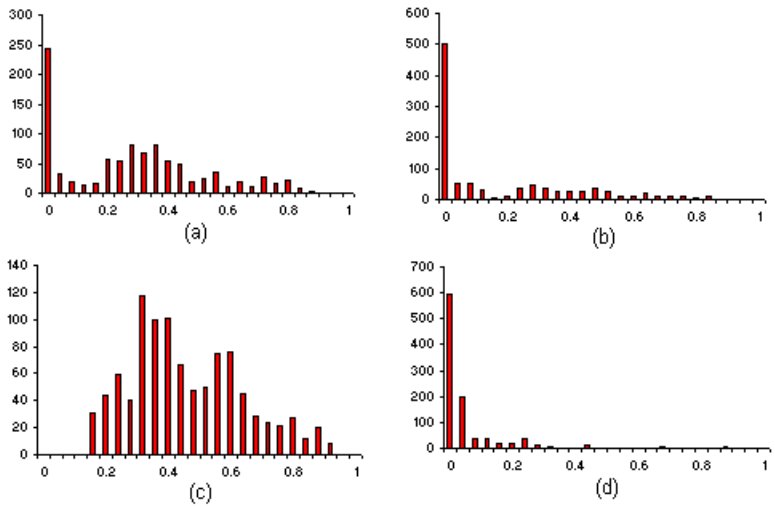

Fig. 2. Histograms of the sparseness degree of: (a) A_1, (b) A_2,(c) A_3 and (d) A_4.

Formerly, we expect that the sparseness degrees of a dimension d follows a mixture density of the form:

$$G(y) = \sum_{l=1}^{m} \gamma_l G_l(y, \alpha_l, \beta_l) \tag{1}$$

where $G_l(.)$ is the *l*th gamma distribution with parameters α_l and β_l which represent, respectively, the shape and the scale parameters of the *l*th component; and $\gamma_l (l = 1, \ldots, m)$ are the mixing coefficients, with the restriction that $\gamma_l > 0$ for $l = 1, \ldots, m$ and $\sum_{l=1}^{m} \gamma_l = 1$.
The density function of the *l*th component is given by

$$G_l(y, \alpha_l, \beta_l) = \frac{\beta_l^{\alpha_l}}{\Gamma(\alpha_l)} y^{\alpha_l - 1} \exp(-\beta_l y) \tag{2}$$

where $\Gamma(\alpha_l)$ is the gamma function given by $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} \exp(-t) dt; \;\; t > 0.$

As mentioned above, each gamma component G_l in equation (2) has two parameters: the shape parameter α_l and the scale parameter β_l. The shape parameter allows the distribution to take on a variety of shapes, depending on its value [19][20]. When $\alpha_l < 1$, the distribution is highly skewed and is L-shaped. When $\alpha_l = 1$, we get the exponential distribution. In the case of $\alpha_l > 1$, the distribution has a peak (mode) in $(\alpha_l - 1)/\beta_l$ and skewed shape. The skewness decreases as the value of α_l increases. This flexibility of the gamma distribution and its positive sample space make it particularly suitable to model the distribution of the sparseness degrees.

A standard approach for estimating the parameters of the gamma components G_l is the maximum likelihood technique [21]. The likelihood function is defined as

$$\begin{aligned} L_{G_l}(\alpha_l, \beta_l) &= \textstyle\prod_{y \in G_l} G_l(y, \alpha_l, \beta_l) \\ &= \frac{\beta_l^{\alpha_l N_l}}{\Gamma^{N_l}(\alpha_l)} \prod_{y \in G_l} y^{\alpha_l - 1} \exp(-\beta_l \sum_{y \in G_l} y) \end{aligned} \tag{3}$$

where N_l is the size of the *l*th component. The logarithm of the likelihood function is given by

$$\log(L_{G_l}(\alpha_l, \beta_l)) = N_l \log(\beta_l) - N_l \log(\Gamma(\alpha_l)) + (\alpha_l - 1) \sum_{y \in G_l} \log(y) - \beta_l \sum_{y \in G_l} y \tag{4}$$

To find the values of α_l and β_l that maximize the likelihood function, we differentiate $\log(L_{G_l}(\alpha_l, \beta_l))$ with respect to each of these parameters and set the result equal to zero:

$$\tfrac{\partial}{\partial \beta_l} \log(L_{G_l}(\alpha_l, \beta_l)) = N_l \log(\beta_l) - N_l \tfrac{\Gamma'(\alpha_l)}{\Gamma(\alpha_l)} + \textstyle\sum_{y \in G_l} \log(y) = 0$$

$$\Rightarrow -\log(\beta_l) + \frac{\Gamma'(\alpha_l)}{\Gamma(\alpha_l)} = \frac{1}{N_l}\sum_{y\in G_l}\log(y) \tag{5}$$

and

$$\frac{\partial}{\partial\beta_l}\log(L_{G_l}(\alpha_l,\beta_l)) = \frac{N_l\alpha_l}{\beta_l} - \sum_{y\in G_l} y = 0$$

$$\Rightarrow \beta_l = \frac{N_l\alpha_l}{\sum_{y\in G_l} y} \tag{6}$$

This yields the equation

$$\log(\alpha_l) - \Psi(\alpha_l) = \log(\frac{1}{N_l}\sum_{y\in G_l} y) - \frac{1}{N_l}\sum_{y\in G_l}\log(y) \tag{7}$$

where $\Psi(.)$ is the digamma function given by $\Psi(\alpha) = \frac{\Gamma\prime(\alpha)}{\Gamma(\alpha)}$.
The digamma function can be approximated very accurately using the following equation [22]:

$$\Psi(\alpha) = \log(\alpha) - \frac{1}{2\alpha} - \frac{1}{12\alpha^2} + \frac{1}{120\alpha^4} - \frac{1}{252\alpha^6} + \dots \tag{8}$$

The parameter $\hat{\alpha_l}$ can be estimated by solving equation (7) using the Newton-Raphson method. $\hat{\alpha_l}$ is then substituted into equation (6) to determine $\hat{\beta_l}$.

The use of a mixture of gamma distributions allows us to propose a flexible model to describe the distribution of the sparseness degree. To form a such model, we need to estimate m, the number of components, and the parameters for each component. One popular approach to estimating the number of components m is to increase m from 1 to m_max and to compute some particular performance measures on each run, until partition into an optimal number of components is obtained. For this purpose, we implement a standard two-step process. In the first step, we calculate the maximum likelihood of the parameters of the mixture for a range of values of m (from 1 to m_max). The second step involves calculating an associated criterion and selecting the value of m which optimizes the criterion. A variety of approaches have been proposed to estimate the number of components in a dataset [23] [24]. In our method, we use a penalized likelihood criterion, called the Bayesian Information Criterion (BIC). BIC was first introduced by Schwartz [25] and is given by

$$BIC(m) = -2L_m + N_p\log(N) \tag{9}$$

where L is the logarithm of the likelihood at the maximum likelihood solution for the mixture model under investigation and N_p is the number of parameters estimated. The number of components that minimizes $BIC(m)$ is considered to be the optimal value for m.

Typically, the maximum likelihood of the parameters of the distribution is estimated using the EM algorithm [26]. This algorithm requires the initial parameters of each component. Since EM is highly dependent on initialization [27], it will be helpful to perform initialization by mean of a clustering algorithm [27] [28]. For this purpose we implement the FCM algorithm [29] to partition the set of sparseness degrees into m components. Based on such a partition we can estimate the parameters of each component and set them as initial parameters to the EM algorithm. Once the EM algorithm converges we can derive a classification decision about the membership of each sparseness degree in each component. The procedure for estimating the PDF of the sparseness degrees of each dimension is summarized in Algorithm 1.

Input : A_j, m_max, k
Output: $m, \alpha_l, \beta_l, \gamma_l$

```
begin
    Based on Definition 1, compute the sparseness degree y_ij;
    Normalize y_ij in the interval ]0, 1];
    for m ← 1 to m_max do
        if m=1 then
            Estimate the parameters of the gamma distribution based on the likelihood formula using equations (6), (7) and (8);
            Compute the value of BIC(m) using equation (9);
        else
            Apply the FCM algorithm as an initialization of the EM algorithm;
            Apply the EM algorithm to estimate the parameters of the mixture using equations (6), (7) and (8);
            Compute the value of BIC(m) using equation (9);
        end
    end
    Select the number of components m̂, such that m̂ = arg_min BIC(m);
end
```

Algorithm 1: PDF estimation of the sparseness degrees of each dimension

Let us focus now on the choice of the value of m_max. In our experiments on different datasets, we observed that the sparseness degrees are well fitted, in general, by one or two gamma components. Based on this, we believe that setting $m\_max = 3$ is, in general, a practical choice. The reader should be aware, however, that the choice of m_max is not limited to three and the user can set other values. However, setting values of $m\_max > 3$ can unnecessarily increase the execution time. Hence, we suggest using $m\_max = 3$ as a default value. The estimated $PDFs$ of the sparseness degrees of A_1, A_2, A_3 and A_4 are illustrated in Figure 3.

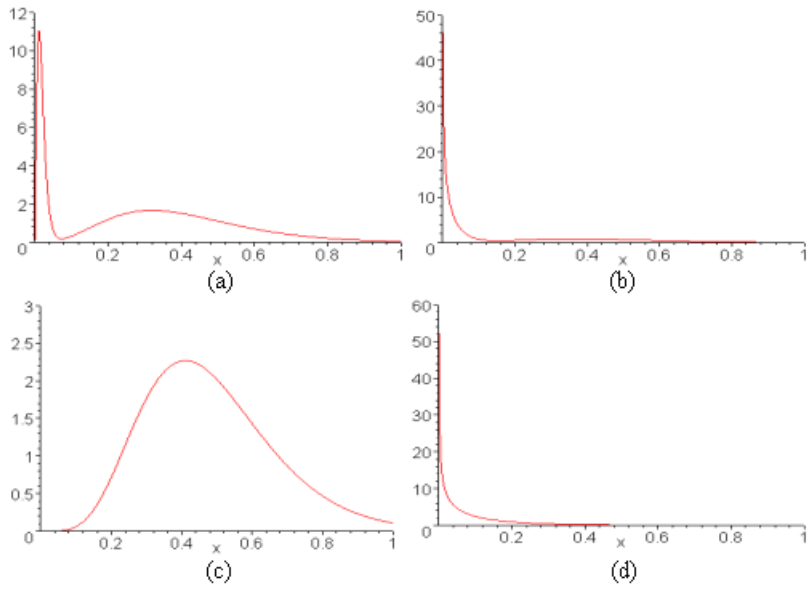

Fig. 3. $PDFs$ of the sparseness degrees of: (a) A_1, (b) A_2,(c) A_3 and (d) A_4. The sparseness degrees of A_1 and A_2 are well fitted by two gamma components, where the first component represents a dense regions while the second one represents a sparse regions. In the case of A_3 and A_4 the estimated $PDFs$ contain only one gamma component, which represents a sparse regions for A_3 and a dense region for A_4.

2) Detection of dense regions: Once the PDF of the sparseness degrees y_{ij} of each dimension is estimated, we turn to the problem of how to detect dense regions and their location in a dimension. For this purpose, we make an efficient use of the properties of the estimated PDF. As illustrated in Figure 3, the locations of the components which represent dense regions are close to zero in comparison to those that represent sparse regions. Based on this observation, we propose a method to examine the location of each of the components in all the dimensions in order to find a typical value that best describes the sparseness degrees of each component. Let loc_q denote the location of the component q, where $q = 1, \ldots, m_{total}$ and m_{total} is the total number of all the components over all the dimensions in the dataset. Intuitively, a large value of loc_q means that the component q corresponds to a sparse regions, while a small one indicates that this component corresponds to a dense regions.

The most commonly used measures of location for univariate data are the mean, the mode and the median. The most appropriate measure in our case is the median, due to the variable shape of the gamma distribution. For instance, when the shape parameter $\alpha_l < 1$, the distribution is L-

shaped with a heavy tail. In this case, the distribution has no mode and the extreme value present in the tail affects the computation of the mean. When $\alpha_l > 1$, the distribution is skewed, which implies that the mean will be pulled in the direction of the skewness. In all these situations, the median provides a better estimate of location than does the mean or the mode, because skewed distribution and extreme values do not distort the median, whose computation is based on ranks.

In order to identify dense regions in each dimension, we are interested in all components with small values of loc_q. We therefore propose to use the MDL principle [30] to separate small and large values of loc_q. The MDL-selection technique that we use in our approach is similar to the MDL-pruning technique described in [1]. The authors in [1] use this technique to select subspaces with large coverage values and discard the rest. We want to use the MDL principle in similar way but in our case we want to select small values of loc_q and their corresponding components. The fundamental idea behind the MDL principle is to encode the input data under a given model and select the encoding that minimizes the code length [30].

Let $LOC = \{loc_1, \ldots, loc_q, \ldots, loc_{m_{total}}\}$ be the set of all loc_q values for each dimension in the entire dataset. Our objective is to divide LOC into two groups E and F, where E contains the highest values of loc_q and F, the low values. To this end, we implement the model described in Algorithm 2.

Input : LOC
Output: E, F
begin
  Sort the values in LOC in descending order;
  for $q \leftarrow 2$ **to** m_{total} **do**
    $E = \{loc_i,\ i = 1, \ldots, q\}$ and the mean of E is μ_E;
    $F = \{loc_j,\ j = q, \ldots, m_{total}\}$ and the mean of F is μ_F;
    Calculate the code length $CL(q) = \log_2(\mu_E) + \sum_{loc_q \in E} \log_2(|loc_q - \mu_E|) + \log_2(\mu_F) + \sum_{loc_q \in F}(|loc_q - \mu_F|)$
  end
  Select the best partitioning given by the pair (E, F), i.e., the one for which the corresponding $CL(q)$ is the smallest;
end

Algorithm 2: MDL-based selection technique

Based on the result of this partitioning process, we selected the components corresponding to each of the loc_q values in F. In Figure 4, we use the dataset described in Section 1 to provide an illustration of this approach. As shown in this figure, there is a clear cutoff point which allows us to select the components with the smallest loc_q values.

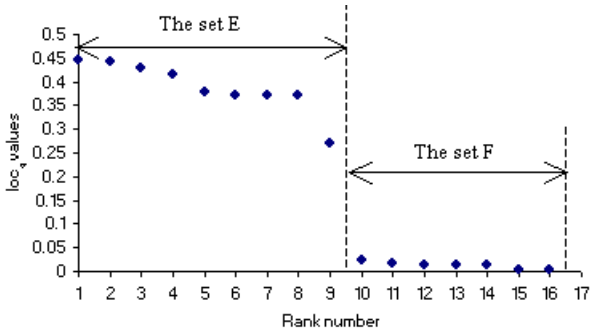

Fig. 4. Partitioning of the set LOC into two sets E and F.

Based on the components selected by means of Algorithm 2, we can now definitively discriminate between dense and sparse regions in all the dimensions in the dataset.

Definition 2. *Let $z_{ij} \in \{0, 1\}$, where z_{ij} is a binary weight.*
If y_{ij} belongs to one of the selected components then $z_{ij} = 1$ and x_{ij} belong to a dense region; else z_{ij}= 0 and x_{ij} belongs to a sparse region.

From Definition 2, we obtain a binary matrix $Z_{(N*d)}$ which contains the information on whether each data point falls into a dense region of an attribute. For example, Figure 5 illustrates the matrix Z for the data used in the example in Section 1. Phase 1 of PCKA is summarized in Algorithm 3.

It is clear that the computation of z_{ij} depends on the input parameter k (the number of nearest neighbors of 1-d point). Although it is difficult to formulate and obtain optimal values for this parameter, it is possible for us to propose guidelines for its estimation. In fact, the role of the parameter k is intuitively easy to understand and it can be set by the user based on specific knowledge of the application. In general, if k is too small, the sparseness degrees y_{ij} are not meaningful, since a 1-d point in a dense region might have a similar sparseness degree value to a 1-d point in a sparse region. Obviously, the parameter k is related to the expected minimum cluster size and should be much smaller than the number of objects N in the data. To gain a clear idea of the sparseness of the neighborhood of a point we have chosen to set $k = \sqrt{N}$ in our implementation.

		A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
Cluster 1	x_1	1	0	0	1	0	0	0	1	0	1
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
		1	0	0	1	0	0	0	1	0	1
Cluster 2	x_i	0	1	0	1	0	1	0	1	1	1
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
		0	1	0	1	0	1	0	1	1	1
Cluster 3	x_j	0	1	0	1	0	0	0	0	0	1
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
		0	1	0	1	0	0	0	0	0	1
Cluster 4	x_k	0	1	0	1	0	1	0	0	1	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	x_N	0	1	0	1	0	1	0	0	1	0

Fig. 5. The matrix $Z_{(N*d)}$.

Input : DB, k, m_max
Output: Z
begin
  $LOC \leftarrow \emptyset$;
  Choose m_max;
  for $j \leftarrow 1$ **to** d **do**
    Apply Algorithm 1: pdf estimation (A_j, m_max, k, m, α_l, β_l, γ_l);
    Partition the sparseness degree in dimension j into m components; //this partitioning process is done by mean of the EM algorithm
    for $i \leftarrow 1$ **to** m **do**
      $LOC \leftarrow LOC \cup median(component_i)$; //median(component i) return the value of the median of component i
    end
  end
  Apply Algorithm 2: MDL-based selection technique(LOC,E,F);
  Based on the loc_q values in F, select the components which represent dense regions;
  Based on Definition 2 compute the matrix Z;
end

Algorithm 3: Phase 1 of PCKA

B. Outlier Handling

In addition to the presence of irrelevant dimensions, high-dimensional data are also characterized by the presence of outliers. Outliers can be defined as a set of data points that are considerably dissimilar, exceptional, or inconsistent with respect to the remaining data [31]. Most of the clustering algorithms, including PCKA, consider outliers as points that are not located in clusters and should be captured and eliminated because they hinder the clustering process.

A common approach to identify outliers is to analyze the relationship of each data point with the rest of the data, based on the concept of proximity [32] [33]. However, in high-dimensional spaces, the notion of proximity is not straightforward [3]. To overcome this problem, our outlier

handling mechanism makes an efficient use of the properties of the binary matrix Z. In fact, the matrix Z contains useful information about dense regions and their locations in the dataset DB. It is obvious that outliers do not belong to any of the identified dense regions; they are located in sparse regions in DB. Such points are highly dissimilar from the rest of the dataset and can be identified by using the binary weights z_{ij}. For this purpose, we use binary similarity coefficients to measure the similarity between binary data points z_i for $(i = 1, \ldots, N)$ in the matrix Z.

Given two binary data points z_1 and z_2, there are four fundamental quantities that can be used to define similarity between the two [34]: $a = |z_{1j} = z_{2j} = 1|, b = |z_{1j} = 1 \wedge z_{2j} = 0|, c = |z_{1j} = 0 \wedge z_{2j} = 1|$ and $d = |z_{1j} = z_{2j} = 0|$, where $j = 1, \ldots, d$. One commonly used similarity measure for binary data is the Jaccard coefficient. This measure is defined as the number of variables that are coded as 1 for both states divided by the number of variables that are coded as 1 for either or both states. In our work, we require a similarity measure that can reflect the degree of overlap between the binary data points z_i in the identified dense regions in the matrix Z. Since dense regions are encoded by 1 in the matrix Z, we believe that the Jaccard coefficient is suitable for our task because it considers only matches on 1's to be important. The similarity Jaccard coefficient is given as:

$$JC(z_1, z_2) = \frac{a}{a + b + c} \tag{10}$$

The Jaccard coefficient has values between 0 (not at all similar) and 1 (completely similar). A pair of points is considered similar if the estimated Jaccard coefficient between them exceeds a certain threshold ε. In all our experiments on a number of datasets, we observed that setting $\varepsilon = 0.70$, for the task of our study, represents an acceptable degree of similarity between two binary vectors.

Our outlier handling mechanism is based on the following definition:

Definition 3. *Let λ be a user-defined parameter, z_i a binary vector from Z, and $SV_i = \{z_j |\ JC(z_i, z_j) < \varepsilon \ and \ j = 1, \ldots, N\}$ the set of similar binary vectors of z_i. A data point x_i is an outlier with respect to parameters λ and ε if $|SV_i| < \lambda$.*

The above definition has intuitive appeal since in essence it exploits the fact that in contrast to outliers, points which belong to dense regions (clusters) generally have a large number of similar points. Based on this, the value of the parameter λ should not be larger than the size of

the cluster containing the ith data point x_i and should be much smaller than N, the size of the dataset DB. Hence, setting $\lambda \approx \sqrt{N}$ is, in general, a reasonable choice. On the other hand, it is clear that when all of the binary weights for a binary data point z_i in the matrix Z are equal to zero, the related data point x_i is systematically considered as an outlier because it does not belong to any of the discovered dense regions.

The identified outliers are discarded from DB and stored in the set OUT, while their corresponding rows are eliminated from the matrix Z. Thus Phase 2 of PCKA yields a reduced data set RDB with size $N_r = N - |OUT|$ and its new associated matrix of binary weights $T_{(N_r * d)}$. Our method for eliminating outliers is described in Algorithm 4.

```
    Input : DB, Z, ε, λ
    Output: RDB, T, OUT
 1  begin
 2      OUT ← ∅;
 3      Let count be a table of size N;
 4      for i ← 1 to N do
 5          count[i] ← 0;
 6      end
 7      for i ← 1 to N do
 8          if Σ_{j=1}^{d} z_ij == 0 then
 9              OUT ← OUT ∪ {x_i};
10          else
11              for j ← i + 1 to N do
                    // estimate the number of similar binary vector of z_i and z_j
12                  if JC(z_i, z_j) > ε then
13                      count[i] ← count[i] + 1;
14                      count[j] ← count[j] + 1;
15                  end
16              end
17              if count[i] < λ then
18                  OUT ← OUT ∪ {x_i};
19              end
20          end
21      end
22      RDB ← DB − OUT;
23      Based on RDB and OUT extract T from Z;
24  end
```

Algorithm 4: Phase 2 of PCKA

C. Discovery of Projected Clusters

The main focus of Phase 3 of PCKA is to perform projected clustering and detect relevant dimensions for each cluster. For this purpose we use the K-means algorithm and exploit the properties of the matrix T. The K-means partitions the data into a number of clusters, each of

which is represented by a center. A data point is assigned to a cluster using a distance function, e.g., Euclidian distance, to calculate its distance from the center of the cluster. However, as could be expected from the discussion in Section 1, this is not an effective approach with high-dimensional data, because each dimension is equally weighted in computing the distance between two points. To address this problem, we associate the binary weights t_{ij} $(i = 1, \ldots, N_r; j = 1, \ldots, d)$ in the matrix T to the Euclidian distance. This makes the distance measure more effective because the computation of distance is restricted to subsets (i.e., projections) where the object values are dense.

Formally, this weighted Euclidean distance between a point x_i and the cluster center v_s $(s = 1, \ldots, nc)$ is defined as

$$dist(x_i, v_s) = \sqrt{\sum_{j=1}^{d} t_{ij} \times (x_{ij} - v_{sj})^2} \tag{11}$$

The use of the matrix T in the K-means algorithm to compute 1) the distance between cluster centers and data points and 2) the centroid coordinates (see Algorithm 5) avoids the computation of the distance in the full-dimensional space and clusters the data points in a more efficient way.

Once the data points are clustered, we turn to the problem of detecting relevant dimensions for each cluster. For this purpose, we make use of the density information stored in the matrix T to determine how well a dimension contributes to the formation of the obtained clusters. In fact, the sum of the binary weights of the data points belonging to the same cluster over each dimension gives us a meaningful measure of the relevance of each dimension to the cluster. Based on this observation, we propose a relevance index W_{sj} for each dimension in cluster C_s. The index W_{sj} for the dimension j $(j = 1, \ldots, d)$ in cluster C_s is defined as follows:

$$W_{sj} = \frac{\sum_{t_i \in C_s} t_{ij}}{|C_s|} \tag{12}$$

The value of the index is always between 0 and 1. The index gives a large value (close to 1) when the dimension is relevant to the cluster. On the other hand, an irrelevant dimension receives a very small index value (close to 0).

Definition 4. *Let $\delta \in]0, 1]$. A dimension A_j is considered δ-relevant for the cluster C_s if $W_{sj} > \delta$.*

In the above definition, δ is a user-defined parameter that controls the degree of relevance of the dimension A_j to the cluster C_s. It is clear that the more relevant the dimension to the cluster, the larger the value of the relevance index. Since W_{sj} is a relative measure, it is not difficult

to choose an appropriate value for δ. In general, setting $\delta \geq 0.8$ is a good practical choice to ensure a high degree of relevance. Phase 3 of PCKA is summarized in Algorithm 5.

Input : RDB, T, nc, δ
Output: $v_s, U_{(N_r * nc)}, A_s$
// v_s: cluster centers where $s = 1, \ldots, nc$
// $U_{(N_r * nc)}$: matrix of the membership degrees of each data point in each cluster
// A_s: set of the relevant dimensions of each cluster
. **begin**
  Choose the cluster centers v_s^0 $(s = 1, \ldots, nc)$ randomly from RDB;
  repeat
    // compute the membership matrix $U_{(N_r * nc)}$
    for $i \leftarrow 1$ **to** N_r **do**
      for $j \leftarrow 1$ **to** nc **do**
        if $dist(x_i, v_s) < dist(x_i, v_j)$ **then**
          $u_{ij} = 0$;
        else
          $u_{ij} = 1$;
        end
      end
    end
    // compute the cluster center
    $v_s^1 = \frac{\sum_{i=1}^{N_r} (u_{is} \times t_i \times x_i)}{\sum_{i=1}^{N_r} u_{is}}$ $(s = 1, \ldots, nc)$;
  until *convergence, i.e., no change in centroid coordinates*;
  Based on Definition 4, detect the set A_s of relevant dimensions for each cluster C_s;
end

Algorithm 5: Phase 3 of PCKA

IV. Empirical Evaluation

In this section we devise a series of experiments designed to evaluate the suitability of our algorithm in terms of:

1) **Accuracy**: the aim is to test whether our algorithm, in comparison with other existing approaches, is able to correctly identify projected clusters.
2) **Efficiency**: the aim is to determine how the running time scales with 1) the size and 2) the dimensionality of the dataset.

We compare the performance of PCKA to that of SSPC [7], HARP [9] , PROCLUS [5] and FASTDOC [10]. The evaluation is performed on a number of generated data sets with different characteristics. Furthermore, experiments on real data sets are also presented. All the experiments reported in this section were run on a PC with Intel Core 2 Duo CPU of 2.4GHz and 4GB RAM.

A. Performance Measures

A number of new metrics for comparing projected clustering algorithms and subspace clustering algorithms were recently proposed in [35]. The performance measure used in our paper is the Clustering Error (CE) distance for projected/subspace clustering. This metric performs comparisons in a more objective way since it takes into account the data point groups and the associated subspace simultaneously. The CE distance has been shown to be the most desirable metric for measuring agreement between partitions in the context of projected/subspace clustering [35]. A deeper investigation of the properties of the CE distance and other performance measures for projected/subspace clustering can be found in [35].

Assume that GP is a collection $\{C_1, \ldots, C_s, \ldots, C_{nc}\}$ of nc generated projected clusters and RP is a collection $\{C'_1, \ldots, C'_s, \ldots, C'_{nc}\}$ of nc real projected clusters. To describe the CE distance, we need to define the union of projected clusters. Let U denote the union of the projected clusters in GP and RP. Formally, $U = U(GP, RP) = supp(GP) \bigcup supp(RP)$, where $supp(GP)$ and $supp(RP)$ are the support of clustering GP and RP, respectively. The support of clustering RP is defined as $supp(GP) = \bigcup_{s=1,\ldots,nc} supp(C_s)$, where $supp(C_s)$ is the support of projected cluster C_s, given by $supp(C_s) = \{x_{ij} | x_i \in C_s \wedge A_j \in A_s\}$. Let $M = (m_{ij})$ denote the confusion matrix, in which m_{ij} represents the number of 1-d points shared by the projected clusters C_s and C'_s, i.e., $m_{ij} = |supp(C_i) \bigcap supp(C'_j)|$. The matrix M is transformed, by using of the Hungarian method [36], in order to find a permutation of cluster labels such that the sum of the diagonal elements of M is maximized. D_{max} denotes this maximized sum. The generalized CE distance for projected/subspace clustering is given by

$$CE(RP, GP) = \frac{|U| - D_{max}}{|U|} \tag{13}$$

The value of CE is always between 0 and 1. The more similar the two partitions GP and RP, the smaller the CE value. When GP and RP are identical, the CE value will be zero.

B. Synthetic Data Generation Method

Since our focus is on axis-parallel clusters, we used the data generator model described in the PROCLUS paper [5] in order to simulate various situations. This data generation model has been used by most researchers in their studies [7] [9] [10] [11] [12] to evaluate the performance

of projected clustering algorithms. The parameters used in synthetic data generation are the size of the dataset N; the number of clusters nc; the dataset dimensionality d; the average cluster dimensionality l_{real}; the domain of the values of each dimension $[min_j, max_j]$; the standard deviation value range $[SD_{min}, SD_{max}]$, which is related to the distribution of points in each cluster; and the outlier percentage OP. Using these parameters, clusters and their associated subspaces are created randomly. Projected values of cluster points are generated according to the normal distribution in their relevant dimension, with the mean randomly chosen from $[min_j, max_j]$ and the standard deviation value from $[SD_{min}, SD_{max}]$. For irrelevant dimensions and outliers, the dimensional values are randomly selected from $[min_j, max_j]$

C. Parameters for Comparing Algorithms

As mentioned above, we compared the performance of PCKA to that of SSPC, HARP, PROCLUS and FASTDOC. In all the experiments on synthetic datasets, the number of clusters nc was set to the true number of projected clusters used to generate the datasets. PROCLUS requires the average cluster dimensionality as a parameter; it was set to the true average cluster dimensionality. Several values were tried for the parameters of FastDOC and SSPC, following the suggestions in their respective papers, and we report results for the parameter settings that produced the best results. HARP requires the maximum percentage of outliers as a parameter; it was set to the true percentage of outliers present in the datasets. In order to obtain satisfactory results and avoid initialization bias, non-deterministic algorithms such as SSPC, PROCLUS and FastDOC were run more than 10 times on each dataset and we consider only the best result for each of them. In all of the experiments, SSPC was run without any semi-supervision. For PCKA, in all the experiments we set $k = \lambda = \sqrt{N}$, $\epsilon = 0.7$ and $\delta = 0.8$.

D. Quality of Results

The performance of projected clustering algorithms is primarily affected by the average cluster dimensionality and the amount of outliers in the dataset. The main goal of the experiments presented in this subsection was to evaluate the capability of projected clustering algorithms to correctly identify projected clusters in various situations.

1) Robustness to the average cluster dimensionality: The main concern of the first set of experiments was to analyze the impact of cluster dimensionality on the quality of clustering.

For this purpose, we generated sixteen different datasets with $N = 3000$ data points, number of dimensions $d = 100$, $min_j = -100$ and $max_j = 100$. In each dataset there were 5 clusters with sizes varying between 10% of N to 30% of N. For each relevant dimension of a cluster, the values of SD_{min} and SD_{max} were set to 1% and 6% of the global standard deviation on that dimension, respectively. The average cluster dimensionality varied from 2% to 70% of the data dimensionality d. Since our goal in this first set of experiments was to analyze the impact of cluster dimensionality, no outliers were generated. Therefore, the outlier detection mechanism of PCKA, SSPC and PROCLUS was disabled. For FastDOC, since we could not disable its outlier elimination option, we chose the results with the highest accuracy after several runs. Figure 6 illustrates the CE distance between the output of each of the five algorithms and the true clustering.

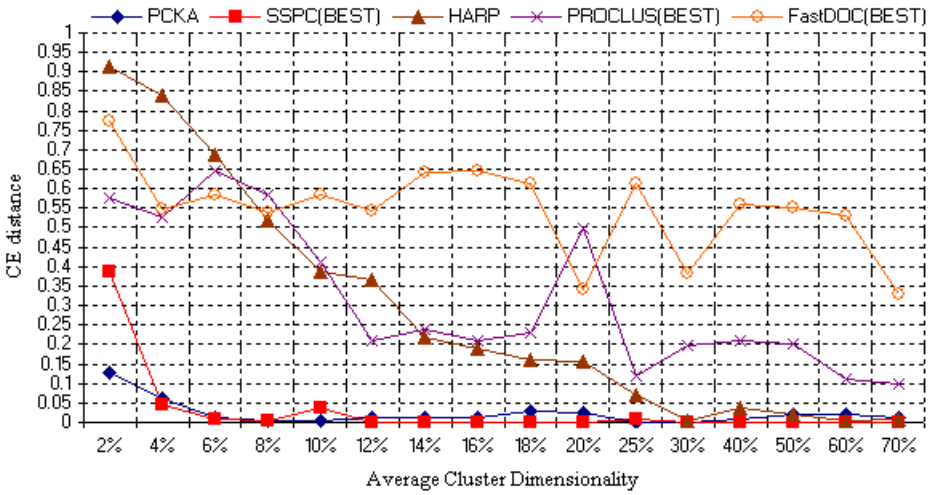

Fig. 6. CE distance between the output of each of the five algorithms and the true clustering.

The values of the CE distance in the above figure invite several comments.

PCKA is able to achieve highly accurate results and its performance is generally consistent. As we can see from Figure 6, PCKA is more robust to variation of the average cluster dimensionality than the other algorithms. For instance, when the average cluster dimensionality is very low ($l_{real} = 2\%$ of d), which is a difficult case, only PCKA yields an acceptable results. The experiments show that our algorithm is able to detect clusters and their relevant dimensions accurately in various situations. PCKA successfully avoids the selection of irrelevant dimensions

in all the datasets used in these experiments. This can be explained by the fact that PCKA starts by identifying dense regions and their locations in each dimension, enabling it to restrict the computation of the distance to subsets of dimensions where the projected values of the data points are dense.

SSPC encounters difficulties when the average cluster dimensionality is low as 2% of d. In other situations, the best results of SSPC are similar to those of PCKA. SSPC provides accurate results and is able to correctly identify projected clusters. This may be due to the fact that SSPC makes use of an objective function that combines data point clustering and dimension selection into a single optimization problem [7].

HARP performs well when $l_{real} \geq 30\%$ of d, displaying performance comparable to that of PCKA and SSPC. On the other hand, when $l_{real} < 30\%$ of d, the performance of HARP is affected and its results are less competitive with those of PCKA and SSPC. We have observed that when $l_{real} < 20\%$ of d, HARP encounters difficulties in correctly identifying clusters and their relevant dimensions. When $l_{real} \in [20\% \, of \, d, \, 30\% \, of \, d]$, HARP clusters the data points well but tends to include many irrelevant dimensions, which explains the values of the CE distance in Figure 6. This can be attributed to the fact that datasets with low-dimensional projected clusters mislead HARP's dimension selection procedures. In such situations, the basic assumption of HARP - i.e., that if two data points are similar in high-dimensional space, they have a high probability of belonging to the same cluster in lower-dimensional space - becomes less valid.

The results of PROCLUS are less accurate than those given by PCKA and SSPC. When datasets contain projected clusters of low dimensionality, PROCLUS performs poorly. When $l_{real} \geq 40\%$ of d, we have observed that PROCLUS is able to cluster the data points well but its dimension selection mechanism is not very accurate because it tends to include some irrelevant dimensions and discard some relevant ones. This behavior of PROCLUS can be explained by the fact that its dimension selection mechanism, which is based on a distance calculation that involves all dimensions by detecting a set of neighboring objects to a medoid, severely hampers its performance.

As we can see from Figure 6, FastDOC encounters difficulties in correctly identifying projected clusters. In our experiments, we have observed that although the dimensionality of projected clusters is high, (i.e., $l_{real} = 70\%$ of d), FastDOC tends to select a small subset of relevant dimensions for some clusters. All the experiments reported here seem to suggest that in the

presence of relevant attributes with different standard deviation values, FastDOC's assumption that projected clusters take the form of a hypercube appears to be less valid. In such situations, it is difficult to set correct values for the parameters of FastDOC.

2) Outlier immunity: The aim of this set of experiments was to test the effect of the presence of outliers on the performance of PCKA in comparison to SSPC, HARP, PROCLUS and FastDOC. For this purpose, we generated three groups of datasets, each containing five datasets with $N = 1000$, $d = 100$ and $nc = 3$. In each dataset in a group, the percentage of outliers varied from 0% to 20% of N. We set $l_{real} = 2\%$ of d for the first group, $l_{real} = 15\%$ of d for the second and $l_{real} = 30\%$ of d for the third. In all the datasets in the three groups, SD_{min} and SD_{max} were set to 1% and 6% of the global standard deviation, respectively. Figure 7 illustrates the CE distance for the five algorithms.

As we can see from the figure, PCKA display consistent performance, as was observed for the first set of experiments on datasets without outliers. In difficult cases, (i.e., when $l_{real} = 2\%$ of d), PCKA provides much better results than all the other algorithms. All the results reported in Figure 7 suggest that PCKA is less sensitive to the percentage of outliers in datasets. In addition to this variations in the average cluster dimensionality in the presence of different percentages of outliers have no major impact on the performance of PCKA. This can be explained by the fact that the outlier handling mechanism of PCKA makes an efficient use of the density information stored in the matrix Z, giving it high outlier immunity.

The performance of SSPC is greatly affected when $l_{real} = 2\%$ of d, with different values of OP. When $l_{real} \geq 4\%$ of d, the performance of SSPC is greatly improved and is not much affected by the percentage of outliers in the datasets. HARP is less sensitive to the percentage of outliers in datasets when $l_{real} \geq 30\%$ of d. Otherwise, (i.e., when $l_{real} < 30\%$ of d, as in the case illustrated in Figure 7), the performance of HARP is severely affected. This behavior of HARP is consistent with the analysis given in the previous subsection. On the other hand, we have observed in our experiments that HARP is sensitive to its maximum outlier percentage parameter. An incorrect choice of the value of this parameter may affect the accuracy of HARP.

Figure 7 illustrate that PROCLUS and FastDOC encounter difficulties in correctly identifying projected clusters. PROCLUS tends to classify a large number of data points as outliers. Similar behavior of PROCLUS was also observed in [8] and [9]. The same phenomenon occurs for FastDOC. We found that FastDOC performs well on datasets that contain dense projected clusters

with the form of a hypercube.

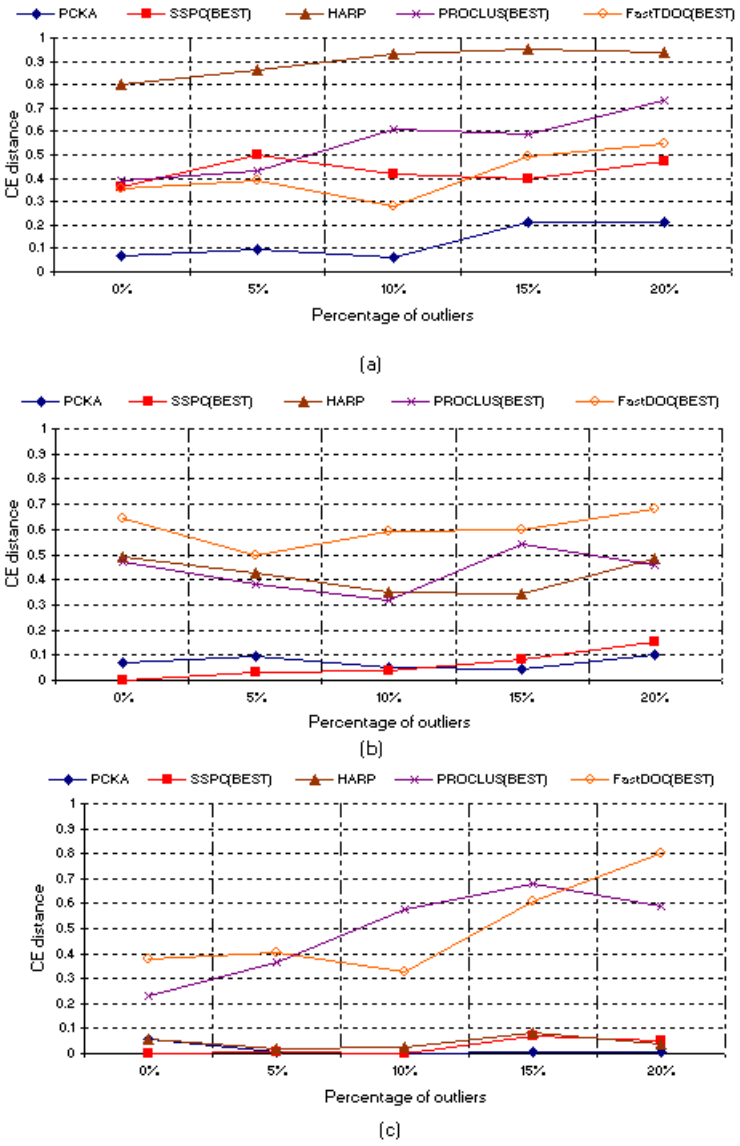

Fig. 7. Immunity to outliers. (a) Datasets with $l_{real} = 2\%$ of d. (b) Datasets with $l_{real} = 15\%$ of d. (c) Datasets with $l_{real} = 30\%$ of d.

E. Scalability

In this subsection, we study the scalability of PCKA with increasing data set size and dimensionality. In all of the following experiments, the quality of the results returned by PCKA was similar to that presented in the previous subsection.

Scalability with data set size: Figure 8 shows the results for scalability with the size of the dataset. In this experiment we used 50-dimensional data and varied the number of data points from 1000 to 100000. There were 4 clusters with $l_{real} = 12\%$ of d and 5% of N were generated as outliers. As we can see from Figure 8, PCKA scales quadratically with increase in dataset size. In all our experiments, we observed that PCKA is faster than HARP. For $N \leq 25000$, the execution time of PCKA is comparable to that of SSPC and PROCLUS when the time used for repeated runs is also included.

Scalability with dimensionality of data: In Figure 9 we see that PCKA scales linearly with the increase in the data dimension. The results presented are for data sets with 5000 data points grouped in 4 clusters, with $l_{real} = 12\%$ of d and 5% of N generated as outliers. The dimensionality of the data sets varies from 100 to 1000. As in the scalability experiments w.r.t. the data set size, the execution time of PCKA is usually better than that of HARP and comparable to those of PROCLUS and SSPC when the time used for repeated runs is also included.

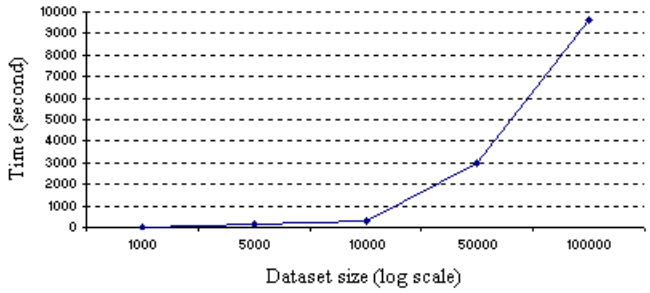

Fig. 8. Scalability of PCKA w.r.t. the dataset size

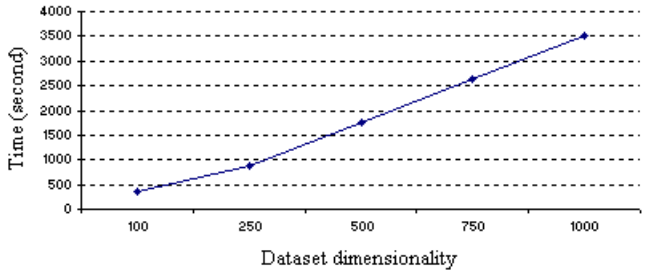

Fig. 9. Scalability of PCKA w.r.t. the dataset dimensionality

F. Experiments on Real-World Data

In addition to our experiments on synthetic data, the suitability of our algorithm was also tested on three real-world data sets. A simple illustration of each of these is given bellow.

Wisconsin Diagnostic Breast Cancer Data (WDBC): This data can be obtained from the UCI Machine Learning Repository (http://www.ics.uci.edu/ mlearn/MLSummary.html). The set contains 569 samples, each with 30 features. The samples are grouped into two clusters: 357 samples for benign and 212 for malignant.

Saccharomyces Cerevisiae Gene Expression Data (SCGE): This data set, available from http://cs.wellesley.edu/ btjaden/CORE, represents the supplementary material used in Brian Tjaden's paper [37]. The Saccharomyces Cerevisiae data contains the expression level of 205 genes under 80 experiments. The data set is presented as a matrix. Each row corresponds to a gene and each column to an experiment. The genes are grouped into four clusters.

Multiple Features Data (MF): This data set is available from the UCI Machine Learning Repository. The set consists of features of handwritten numerals $(\text{"0"} - \text{"9"})$ extracted from a collection of Dutch utility maps. 200 patterns per cluster (for a total of 2,000 patterns) have been digitized in binary images. For our experiments, we used five feature sets (files):

1. mfeat-fou: 76 Fourier coefficients of the character shapes;
2. mfeat-fac: 216 profile correlations;
3. mfeat-kar: 64 Karhunen-Love coefficients;
4. mfeat-zer: 47 Zernike moments;
5. mfeat-mor: 6 morphological features.

In summary, we have a data set with 2000 patterns, 409 features, and 10 clusters. All the values in each feature were standardized to mean 0 and variance 1.

We compared the performance of our algorithm, in terms of clustering quality, with that of SSPC, HARP, PROCLUS and FastDOC. For this purpose, we used the class labels as ground truth and measured the accuracy of clustering by matching the points in input and output clusters. We adopted the classical definition of accuracy as the percentage of correctly partitioned data points. We want to mention that the value of the average cluster dimensionality parameter in PROCLUS was estimated based on the number of relevant dimensions identified by PCKA. In addition to this, since there are no outliers in any of the data described above, the outlier

TABLE I

ACCURACY OF CLUSTERING

Data Set	PCKA	SSPC (Best)	HARP	PROCLUS (Best)	FastDOC (Best)
WDBC	91.56	93.84	62.91	71.00	83.59
SCGE	98.53	96.09	88.29	86.34	45.85
MF	90.45	–	58.35	83.50	10.00

detection option of PCKA, SSPC, HARP and PROCLUS was disabled. Table I illustrates the accuracy of clustering for the four algorithms.

As can be seen from the above table, PCKA is able to achieve highly accurate results in different situations involving data sets with different characteristics. These results confirm the suitability of PCKA previously observed on the generated data. The behavior of SSPC is also characterized by high clustering accuracy. However, with multiple feature data, the clustering process is stopped and SSPC returns an error. HARP yields moderate accuracy on WDBC and MF data, while its performance on SCGE data is acceptable. FastDOC yields acceptable accuracy on WDBC data, while it performs poorly on SCGE and MF data. The poor results may be due to inappropriate choice of parameters, although we did try different sets of parameters in our experiments. Finally, from the experiments on the three data sets, we can see that the accuracy achieved by PROCLUS is reasonable.

V. CONCLUSION

We have proposed a robust distance-based projected clustering algorithm for the challenging problem of high-dimensional clustering, and illustrated the suitability of our alogorithm in tests and comparisons with previous work. Experiments show that PCKA provides meaningful results and significantly improves the quality of clustering when the dimensionalities of the clusters are much lower than that of the dataset. Moreover, our algorithm yields accurate results when handling data with outliers. The performance of PCKA on real data sets suggests that our approach could be an interesting tool in practice. The accuracy achieved by PCKA results from its restriction of the distance computation to subsets of attributes, and its procedure for the initial selection of these subsets. Using this approach, we believe that many distance-based clustering

algorithms could be adapted to cluster high dimensional data sets.

There are still many obvious directions to be explored in the future. The interesting behavior of PCKA on generated datasets with low dimensionality suggests that our approach can be used to extract useful information from gene expression data that usually have a high level of background noise. From the algorithmic point of view, we believe that an improved scheme for PCKA is possible. One obvious direction for further study is to extend our approach to the case of arbitrarily oriented clusters. Another interesting direction to explore is to extend the scope of Phase 1 of PCKA from attribute relevance analysis to attribute relevance and redundancy analysis. This seems to have been ignored by all of the existing projected clustering algorithms.

ACKNOWLEDGMENT

We gratefully thank Kevin Yuk-Lap Yip of Yale University for providing us with the implementations of projected clustering algorithms used in our experiments. This work is supported by research grants from the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data," *Data Mining and Knowledge Discovery*, vol. 11, no. 1, pp. 5–33, 2005.

[2] A. K. Jain, M. N. Mutry, and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[3] K. Beyer, J. Goldstein, R. Ramakrishan, and U. Shaft, "When Is Nearest Neighbor Meaningful?," *Proc. of the 7th International Conference on Database Theory*, pp. 217–235, 1999.

[4] H. Liu and L. Yu, "Toward Integrating Feature Selection Algorithms for Classification and Clustering," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 3, pp. 1–12, 2005.

[5] C.C. Aggarwal, C. Procopiuc, J. L. Wolf, P.S. Yu, and J.S. Park, "Fast Algorithm for Projected Clustering," *Proc. ACM SIGMOD Conf.*, pp. 61–72, 1999.

[6] K.Y.L. Yip, D. W. Cheung, M. K. Ng and K. Cheung, "Identifying Projected Clusters from Gene Expression Profiles," *Journal of Biomedical Informatics*, vol. 37, no. 5, pp. 345–357, 2004.

[7] K.Y.L. Yip, D.W. Cheng and M.K. Ng, "On Discovery of Extremely Low-Dimensional Clusters using Semi-Supervised Projected Clustering," *Proc. ICDE*, pp. 329–340, 2005.

[8] C.C. Aggarwal and P.S. Yu, "Redefining Clustering for High Dimensional Applications," *IEEE Trans. Knowledge and Data Eng.*, vol. 14, no. 2, pp. 210–225, 2002.

[9] K.Y.L. Yip, D.W. Cheng and M.K. Ng, "HARP: A Practical Projected Clustering Algorithm," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 11, pp. 1387–1397, 2004.

[10] C.M. Procopiuc, M. Jones, P.K. Agarwal, and T.M. Murali, "Monte Carlo Algorithm for Fast Projective Clustering," *Proc. ACM SIGMOD*, pp. 418 – 427, 2002.

[11] M. Lung and N. Mamoulis, "Iterative Projected Clustering by Subspace Mining," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 2, pp. 176–189, 2005.

[12] E.K.K. Ng, A.W. Fu, and R.C. Wong, "Projective Clustering by Histograms," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 32, pp. 369–383, 2005.

[13] M. Bouguessa, S. Wang, Q. Jiang, "A K-means-based Algorithm for Projective Clustering," *Proc. 18th IEEE International Conference on Pattern Recognition*, pp. 888–891, 2006.

[14] C.H. Cheng, A.W. Fu, Y. Zhang, "Entropy-based Subspace Clustering for Mining Numerical Data," *Proc. ACM SIGMOD*, pp. 84–93, 1999.

[15] S. Goil, H. Nagesh, and A. Choudhary, "MAFIA: Efficient and scalable subspace clustering for very large data sets," *Technical Report CPDC-TR-9906-010, Northwestern University*, 1999.

[16] K. Kailing, H.-P. Kriegel, and P. Kroger, "Density-Connected Subspace Clustering for High-Dimensional Data," *Proc. Fourth SIAM Int'l Conf. Data Mining*, pp. 246–257, 2004.

[17] L. Parsons, E. Haque, and H. Liu, "Subspace Clustering for High Dimensional Data: A Review," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 90–105, 2004.

[18] K.Y.L. Yip, "HARP: A Practical Projected Clustering Algorithm for Mining Gene Expression Data," Master's thesis, The Univ. of Hong Kong, Hong Kong, 2004.

[19] K. Bury, *Statistical Distributions in Engineering*. Cambridge University Press, 1998.

[20] N. Balakrishnan and V.B. Nevzorov, *A Primer on Statistical Distributions*. John Wiley and Sons, 2003.

[21] R.V. Hogg, J.W. McKean and A. T. Craig, *Introduction to Mathematical Statistics*. Pearson Prentice Hall, sixth ed., 2005.

[22] J.F. Lawless, *Statistical Models and Methods for Lifetime Data*. John Wiley and Sons, 1982.

[23] M. Bouguessa, S. Wang and H. Sun, "An Objective Approach to Cluster Validation," *Pattern Recognition Letters*, vol. 27, no. 13, pp. 1419–1430, 2006.

[24] J.J. Oliver, R.A. Baxter and C.S. Wallace, "Unsupervised Learning Using MML," *Proc. of the 13th International Conference on Machine Learning*, pp. 364–372, 1996.

[25] G. Schwarz, "Estimating the Dimension of a Model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[26] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of Royal Statistical Society, (Series B)*, vol. 39, pp. 1–37, 1977.

[27] M.A.T. Figueiredo and A.K. Jain, "Unsupervised Learning of Finite Mixture Models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 381–396, 2002.

[28] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. John Wiley and Sons, 1997.

[29] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[30] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*. World Scientific Publ. Co., 1989.

[31] J. Han and M. Kamber, *Data Mining, Concepts and Techniques*. Morgan Kaufman, 2001.

[32] F. Angiulli and C. Pizzuti, "Outlier Mining in Large High-Dimensional Data Sets," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 2, pp. 369–383, 2005.

[33] E.M. Knorr, R.T. Ng and V. Tucakov, "Distance-Based Outliers: Algorithms and Applications," *The VLDB Journal*, vol. 8, no. 3-4, pp. 237–253, 2000.

[34] T. Li, "A Unified View on Clustering Binary Data," *Machine Learning*, vol. 62, no. 3, pp. 199–215, 2006.

[35] A. Patrikainen and M. Meila, “Comparing Subspace Clusterings,” *IEEE Trans. Knowledge and Data Eng.*, vol. 18, no. 7, pp. 902–916, 2006.

[36] C.H. Papadimitrio and K. Steiglizt, *Combinatorial Optmization, Algorithms and Complexity*. Prentice-Hall, 1982.

[37] Brian Tjaden, “An approach for clustering gene expression data with error information,” *BMC Bioinformatics*, vol. 7, no. 17, 2006.