HKU CS Tech Report TR-2004-08

On Discovery of Extremely Low-Dimensional Clusters using Semi-Supervised Projected Clustering

Kevin Y. Yip, David W. Cheung
Department of Computer Science
University of Hong Kong
{ylyip,dcheung}@cs.hku.hk

Michael K. Ng
Department of Mathematics
University of Hong Kong
mng@maths.hku.hk

Abstract

Recent studies suggest that projected clusters with extremely low dimensionality exist in many real datasets. A number of projected clustering algorithms have been proposed in the past several years, but few can identify clusters with dimensionality lower than 10% of the total number of dimensions, which are commonly found in some real datasets such as gene expression profiles. In this paper we propose a new algorithm that can accurately identify projected clusters with relevant dimensions as few as 5% of the total number of dimensions. It makes use of a robust objective function that combines object clustering and dimension selection into a single optimization problem. The algorithm can also utilize domain knowledge in the form of labeled objects and labeled dimensions to improve its clustering accuracy. We believe this is the first semi-supervised projected clustering algorithm. Both theoretical analysis and experimental results show that by using a small amount of input knowledge, possibly covering only a portion of the underlying classes, the new algorithm can be further improved to accurately detect clusters with only 1% of the dimensions being relevant. The algorithm is also useful in getting a target set of clusters when there are multiple possible groupings of the objects.

1. Introduction

Recently many studies have suggested the presence of low dimensional clusters in high-dimensional real datasets. For example, in a typical microarray gene expression dataset that contains the expression values of several thousands of genes in different samples, it is common to find that only several tens of genes have expression patterns unique to each cluster of samples [15]. The genes are called the relevant genes, as opposed to the irrelevant genes that do not help much in identifying the cluster members. Due to the large number of genes being irrelevant to each cluster, two samples in the same cluster could have low similarity when measured by a similarity function that considers the expression values of all genes. The clusters are thus undetectable by traditional clustering algorithms.

The projected clustering problem [1] is defined for such a scenario. Each projected cluster is a set of member objects with an associated set of relevant dimensions such that the member objects are similar to each other in the subspace formed by the relevant dimensions, but dissimilar to objects outside the cluster. In this paper we measure object similarity based on Euclidean distance, so a dimension is more relevant to a cluster if the projections of its members on the dimension are closer to each other, but more remote from the projections of other objects. The goal of a projected clustering algorithm is to identify clusters of objects and their relevant dimensions such that a certain objective function (e.g. average within-cluster similarity along the relevant dimensions) is optimized.

While the clusters in real datasets can contain an extremely low percentage of relevant dimensions (less than 10% of all genes in the previous example), it has been reported that most current projected clustering algorithms are unable to identify clusters with such low dimensionality [25]. This is mainly due to their use of objective functions that highly rely on the accuracy of some input parameters, and the use of similarity calculations that involve all dimensions, which may not reflect the real similarity between different objects [25].

In addition, being unsupervised methods, the algorithms make little use of domain knowledge in the clustering process, despite the fact that a small amount of domain knowledge is usually available in some applications. For example, in gene expression datasets, the functions of a small number of genes are usually known to the biologists.

In order to better utilize domain knowledge in the clustering process, a number of *semi-supervised* clustering algorithms [9] have been proposed, which we will review in the next section. The basic idea is to use some domain knowledge to guide the clustering process. For example, in a semi-supervised k-means algorithm [21], domain knowledge about the relationships between some objects is used to force the assignment of some object pairs to the same cluster or to different clusters. As reported in many studies (e.g. [8,20]), the clustering accuracy can be greatly improved by inputting only a small amount of domain knowledge.

It should be noted that while some domain knowledge is being used, semi-supervised clustering is different from classification (supervised-learning) in that the knowledge being used in semi-supervised clustering may not be suitable or sufficient for classification. For instance, the knowledge needs not be in the form of objects with class labels as required by classification. The amount of knowledge for each class can be so small that is statistically insignificant to build a classifier that captures the general properties of the class, and the input knowledge can be biased towards one side of the class. The input knowledge of semi-supervised clustering also needs not cover all classes, but it is still possible to produce all the corresponding clusters.

Previous studies on semi-supervised clustering have been focused on non-projected clustering, which does not consider the relevance of dimensions. The semi-supervised approach is in fact very useful in projected clustering. For instance, given a small amount of example objects of a cluster (the "labeled objects", e.g. tumor samples known to be of a certain type), the relevant dimensions of the cluster can be estimated by the dimensions along which the objects are significantly close to each other. Similarly, having a dimension specified as relevant to a cluster (a "labeled dimension", e.g. a gene known to be relevant to a tumor type), the cluster members can be estimated from regions with unexpectedly high object densities.

Semi-supervised clustering also has an important application in handling datasets that have multiple possible groupings. For example, in cancer study, patients can be grouped by their response to a certain treatment, or by the risk of having cancer recurrence. Unsupervised methods can only produce a single set of clusters, which may correspond to only one of the groupings, or even none of them. Using the semi-supervised approach, by supplying different input knowledge, a single clustering algorithm can be guided to produce both kinds of clusters in different runs.

All the above observations motivate the current study, which has three major contributions:

- Proposing a robust objective function for projected clustering that naturally involves dimension selection in the optimization process.
- Proposing the use of domain knowledge (labeled objects and labeled dimensions) to improve the accuracy of projected clustering.
- Developing a new algorithm that can (theoretically and empirically) detect clusters of extremely low dimensionality, and whose accuracy can be further improved by incorporating some domain knowledge in the clustering process.

In the next section we will review some related work in projected clustering and semi-supervised clustering. In Section 3 we will formally define the semi-supervised projected clustering problem, as well as the assumptions being made in this study. In Section 4 we will describe our new algorithm. Experimental results will be presented in Section 5, together with some observations and discussions. In Section 6 we will summarize the whole study and discuss some future extensions.

2. Related Work

2.1. Projected Clustering

Existing projected clustering algorithms can be classified into three categories: partitional, one cluster at a time, and hierarchical. The partitional approach PROCLUS [1] is one of the earliest projected clustering algorithms. It is based on the traditional k-medoids approach [13], with a goal of minimizing the average within-cluster dispersion. It differs from traditional k-medoids in that the distance between different cluster members is computed in the relevant subspace of the cluster. The relevant subspace of a cluster is determined by measuring the average distance

between the medoid and a set of "neighboring objects" that are close to it when all dimensions are considered. The dimensions with the smallest average distances to the medoid of each cluster are selected as the relevant dimensions of the cluster, which form its relevant subspace.

Another partitional algorithm ORCLUS [2] improves PROCLUS by selecting principal components so that clusters not parallel to the original dimensions can also be detected. It also adds a hierarchical part that can potentially reduce the errors due to inaccurate initial object assignments.

A limitation of these partitional methods is the determination of neighboring objects based on similarity calculations that involve all dimensions. If the number of relevant dimensions of each cluster is small, different members of a cluster may appear to be dissimilar when all dimensions are considered. In these methods, therefore, the neighboring objects of a medoid need not be come from the same real cluster and the relevant dimensions suggested by them could be wrong. Also, as the approaches use an objective function that tends to give better scores when fewer dimensions are regarded as relevant to a cluster [25, 26], they require users to supply the average number of relevant dimensions per cluster, which is usually unknown to users. If incorrect parameter values are used, the clustering accuracy can be seriously affected.

The Monte Carlo method DOC [16] and its variant, FastDOC, identify projected clusters one after another. To find a cluster, an object is randomly selected as the seed, and a number of other objects are randomly sampled to determine the relevant subspace of the cluster. A dimension is regarded as relevant to the cluster if all the objects are within a distance ω from the seed along the dimension. A cluster is thus in the form of a hypercube of width 2ω. The quality of a cluster is evaluated by an objective function that takes into account both the number of objects falling in the hypercube, and the number of relevant dimensions of the cluster. The more objects and relevant dimensions a cluster has, the less likely it is formed by random chance, and thus it receives a better score. The relative importance between the number of objects and relevant dimensions is controlled by a user parameter β. The algorithm repeatedly tries different seeds and neighboring objects until a certain number of iterations is reached, when the cluster with the highest score will be returned and the algorithm will try to find a new cluster.

The algorithms perform well when each cluster is in the form of a hypercube and the parameter values are specified correctly, but in many cases these requirements cannot be met and the clustering results are quite unsatisfactory [25]. The number of seeds and neighboring objects required to try can also be so large that causes the algorithms to run for a long time.

The hierarchical algorithm HARP [25] makes use of the dynamic threshold loosening mechanism to avoid the problems of the previous methods. The basic assumption is that two objects are likely to belong to the same cluster if they are very similar to each other along many dimensions. Clusters are allowed to merge only if they are similar enough in a number of dimensions, where the minimum similarity and minimum number of similar dimensions are controlled by two internal threshold variables. Instead of asking users to supply fixed values for the two variables, HARP automatically adjusts their values during the clustering process. At the beginning, the thresholds are set to some harsh values such that only merges that are very likely to group objects belonging to the same real cluster are allowed to be performed. As the relevant dimensions of each cluster becomes more apparent, the threshold values are loosened to allow more cluster merges. The process repeats until the thresholds reach their baseline values, or a target number of clusters is reached.

The method successfully avoids extensive distance calculations that involve all dimensions and user parameters whose values are hard to determine. However, due to the hierarchical nature, the algorithm is intrinsically slow. Also, if the number of relevant dimensions per cluster is extremely low (e.g. 5% of the dataset dimensionality), the accuracy of HARP may drop as the basic assumption will become less valid due to the presence of large amount of noise values in the dataset.

In summary, most of the existing projected clustering algorithms make use of objective functions whose effectiveness rely greatly on the accuracy of some parameter values that are hard for users to determine. Some of them involve similarity calculations that consider all dimensions, which can be quite misleading when cluster dimensionality is small. They also do not make explicit use of domain knowledge in the common forms such as labeled objects and labeled dimensions.

A more thorough survey of the algorithms, as well as the algorithms proposed for two related problems, namely subspace clustering [3] and biclustering [7], can be found in [26].

2.2. Semi-supervised Clustering

A recent trend in machine learning research is to combine the techniques developed for unsupervised learning and supervised learning to handle datasets with partial external information. One of the foci is semi-supervised clustering, which actively uses the available domain knowledge in guiding the clustering process. These methods can be categorized according to the following three groupings: what knowledge is being input, when the knowledge is input, and how the knowledge affects the clustering process.

The simplest type of input is labeled objects [4, 9]. In some cases, users do not know the exact class labels of objects, but they have some knowledge on which objects should be/should not be put into the same cluster, which can be specified by must-links and cannot-links [5, 6, 11, 20, 21]. Some other studies propose the input of classification rules [19], examples of similar objects [23], or even general comments like which cluster a particular object should not be put into [8].

The knowledge can be supplied at different time. It can be supplied before clustering to guide the clustering process [4, 5, 6, 9, 11, 20, 21, 23], or after clustering to evaluate the clusters and guide the next round of clustering [8]. Some algorithms can also actively request users to supply some specific information at the most appropriate time [5, 11].

There are various ways to use the input knowledge, such as guiding the formation of seed clusters [4, 5, 6], forcing or recommending some objects to be put in the same cluster or different clusters [20, 21], and modifying the objective function [5, 6, 9], similarity function [8, 23] or distance matrix [11].

3. Problem Definition

We now formally define the semi-supervised projected clustering problem. We start with the data model. Given a dataset D with n objects and d dimensions, the objects can be partitioned into k clusters $\{C_i\}_{i=1}^{k}$ and a possibly empty set of outliers. We assume each cluster is a random sample of the corresponding hidden class, each of which is associated with a set of relevant dimensions that form a relevant subspace. Denote D_j and C_{ij} as the projections of D and C_i on a dimension v_j respectively. Suppose v_j is relevant to a subset R_j of the clusters, then for each cluster $C_i \in R_j$, C_{ij} is a random sample of a local Gaussian population with a small variance σ_{ij}^2. The set of all other projected values on v_j, $D_j - \bigcup_{C_i \in R_j} C_{ij}$, is a random sample of a global population with a variance σ_j^2 much larger than the local Gaussians.

Intuitively, in the relevant subspace of a cluster, the cluster members are on average close to each other, but remote from other objects not in the cluster. Alternatively, given a cluster with an unknown relevant subspace, a dimension is more likely to be relevant to the cluster if the projection of the cluster on the dimension has a smaller variance.

To distinguish the actual clusters due to the hidden classes and the clusters produced by a clustering algorithm, in the remaining of this paper we will call the former the "real clusters" and the later simply the "clusters". We will also call the dimensions determined by a clustering algorithm as relevant to a cluster the "selected dimensions" of it.

The inputs to our semi-supervised projected clustering algorithm are:

- The dataset D
- The target number of clusters k
- A (possibly empty) set I^o of labeled objects (<obj. ID, class label> pairs), each indicates that the object is a member of the class. The set may or may not cover all classes.
- A (possibly empty) set I^v of labeled dimensions (<dim. ID, class label> pairs), each indicates that the dimension is relevant to the class. Each dimension can be specified as relevant to multiple classes. The set may or may not cover all classes.

The outputs of the algorithm are k clusters and their selected dimensions, and a (possibly empty) set of outliers. The goal is to optimize an objective function whose value (the objective score) reflects the quality of the clusters. In the non-projected clustering algorithm k-means [10], the objective function is defined as the total within-cluster squared error. It can be shown that the partition of objects that minimize the function corresponds to the maximum likelihood hypothesis of the above model when there are no irrelevant dimensions [12]. In [1], the objective function is modified for projected clustering such that only relevant dimensions are involved in the distance

calculations, and the part of objective score from each cluster is normalized by the number of selected dimensions. Due to the normalization, the function tends to gives better (i.e., smaller) scores for clusters with fewer selected dimensions [25, 26], which forces the algorithm to request users to supply the average cluster dimensionality in order not to select only one dimension per cluster. Also, as the function is based on the summation of variances among different dimensions, a worse dimension (one with larger variance) constitutes more to the objective score. Since from our data model, the variance of a cluster is much smaller along a relevant dimension than an irrelevant dimension, if some irrelevant dimensions are accidentally selected, the objective score can be dominated by the constituents from the irrelevant dimensions, and it can remain virtually unchanged if some relevant dimensions are deselected.

We therefore designed a new objective function with three goals: 1) it should facilitate the selection of dimensions based on the particular data properties of different clusters and dimensions, 2) its value should be constituted more by relevant dimensions, and 3) it should be relatively robust. The function, denoted as ϕ, is defined as follows:

$$\phi = \frac{1}{nd}\sum_{i=1}^{k}\phi_i \quad (1)$$

$$\phi_i = \sum_{v_j \in V_i}\phi_{ij} \quad (2)$$

$$\phi_{ij} = n_i - 1 - \frac{1}{\hat{s}_{ij}^2}\sum_{x \in C_i}(x_j - \tilde{\mu}_{ij})^2 \quad (3)$$

$$= (n_i - 1)(1 - \frac{s_{ij}^2 + (\mu_{ij} - \tilde{\mu}_{ij})^2}{\hat{s}_{ij}^2}), \quad (4)$$

where V_i is the set of selected dimensions of cluster C_i, n_i is the size of (number of objects in) C_i, x_j is the projection of an object x on dimension v_j, $\tilde{\mu}_{ij}$, μ_{ij} and s_{ij}^2 are the sample median, sample mean and sample variance of the projection of C_i on v_j respectively, and $\hat{s}_{ij}^2$ is the selection threshold whose meaning will be explained later. The objective function ϕ is composed of the score components ϕ_i of each cluster, which in turn is the sum of the score components ϕ_{ij} of each selected dimension. ϕ_{ij} is basically the negation of the sample variance s_{ij}^2 adjusted to the range $(0, n_i - 1]$, such that if all members of C_i have the same projected value on v_j, ϕ_{ij} takes the maximum value of $n_i - 1$.

There are three major differences between ϕ and the objective function defined in [1]. First, ϕ_i is not normalized by the number of selected dimensions of C_i, but by the threshold $\hat{s}_{ij}^2$ instead. As to be discussed later in this section, this allows the dimension selection mechanism to be based on the data characteristics of C_i along v_j (design goal #1). It also avoids the existence of trivial best score when each cluster selects only one dimension. Second, by setting $\hat{s}_{ij}^2$ to a value that is always larger than the sample variance s_{ij}^2 of each selected dimension, ϕ_{ij} is always positive, and a better dimension (a dimension with smaller s_{ij}^2) has a larger constituent to ϕ_i (design goal #2). A good clustering is reflected by a large ϕ value, so the goal is to maximize the objective score. Third, within-cluster dispersion is measured by the distance from the cluster median rather than centroid, which makes the function less affected by outliers. We will discuss the robustness of ϕ later.

The definition of the objective function leads to the following lemma:

Lemma 1 *Given a set of clusters $\{C_i\}_{i=1}^k$, the objective function ϕ is maximized when all dimensions with $s_{ij}^2 + (\mu_{ij} - \tilde{\mu}_{ij})^2$ smaller than $\hat{s}_{ij}^2$ are selected and all other dimensions are not selected.*

Proof: To maximize ϕ, a dimension v_j should be selected if ϕ_{ij} is positive, and should not be selected if ϕ_{ij} is negative, which correspond to the cases where $s_{ij}^2 + (\mu_{ij} - \tilde{\mu}_{ij})^2$ is smaller than and larger than $\hat{s}_{ij}^2$ respectively. □

The following dimension selection procedure follows directly from the lemma:

$\hat{s}_{ij}^2$ is thus called the selection threshold, whose values are critical to the accuracy of the clustering algorithm. In the next section we will show that there is a simple scheme to determine the values of $\hat{s}_{ij}^2$, and a more advanced probabilistic-based scheme when certain conditions are satisfied. In both cases, only one user parameter is required, and whose value is not critical to the clustering accuracy. So even the objective function is defined in the most generic way with a different $\hat{s}_{ij}^2$ for each cluster and each dimension, this does not create a tremendous amount of parameters for the algorithm.

Procedure SelectDim(C_i: target cluster)
1 Foreach dimension v_j do
2 Select v_j for C_i if and only if $s_{ij}^2 + (\mu_{ij} - \tilde{\mu}_{ij})^2 < \hat{s}_{ij}^2$
End

Listing 1: The dimension selection procedure.

In this study we confine the scope by a number of assumptions, most of which are also implicitly or explicitly made in previous studies on projected clustering or semi-supervised clustering. The possibility of relaxing some of them will be discussed in Section 6.

1. The determination of relevant dimensions is mainly to help identify the object clusters (as opposed to biclustering [7] where both rows and columns of a dataset are treated equally).
2. Clusters are disjoint (as opposed to subspace clustering [3]) and axis-parallel (as opposed to ORCLUS [2]).
3. Object similarity is based on the difference between projected values (as opposed to pattern-based clustering [14, 22]).
4. All objects of a class are close to each other in a certain subspace, so that one class corresponds to only one real cluster (as opposed to decision trees [17] where the objects of one class can form multiple clusters).
5. The input knowledge is correct yet can be biased. For instance, if an object is input as the member of a cluster, it must really belong to the cluster, although it may have projections on the relevant dimensions highly deviated from the cluster mean.

4. The New Algorithm

The outline of the new algorithm SSPC (Semi-Supervised Projected Clustering) is shown in Listing 2. It is a partitional method similar to the k-medoids algorithms [13]. At the beginning it determines some seeds (potential medoids) and each cluster draws a medoid from them. Every object in the dataset is then assigned to the cluster that gives the greatest improvement to the objective score, where the value of $\tilde{\mu}_{ij}$ in Equation 4 is temporarily substituted by the projection of the medoid on v_j. If an object does not improve the ϕ_i score of any cluster, it will be put on the outlier list. After assigning all objects, the selected dimensions of each cluster are redetermined and the overall objective score is recomputed using the actual medians. If the new score is the best one encountered so far, the clusters will be recorded. Otherwise, the best clusters will be restored. A bad cluster is then identified from the current best set of clusters, and a new medoid is selected for it with an attempt to improve the objective score in the next iteration. The medoid of each other cluster is replaced by the cluster median (the virtual object with projected value along each dimension equal to the median of the cluster members) as a medoid could have projected values deviated from the cluster center along some relevant dimensions according to the data model in Section 3. In the next iteration, the values of $\tilde{\mu}_{ij}$ will be substituted by these medians. We will use the term "cluster representative" to call the medoid or median that represents a cluster. After replacing the old cluster representatives, the members of each cluster are removed, and a new iteration of object assignment, score comparison and cluster representatives replacement is carried out. The process repeats until the best objective score remains unchanged for a certain number of iterations.

Algorithm SSPC
1 Initialization: determine the seeds and relevant dimensions of each cluster
2 For each cluster, draw a medoid from the seeds
3 Assign every object in the dataset to the cluster (or outlier list) that gives the greatest improvement to the objective score
4 Call SelectDim(C_i) for each cluster C_i, and calculate the overall objective score
5 Record the clusters if they give the best objective score so far, restore the best clusters otherwise
6 Replace the cluster representative of each cluster, then remove its members
7 Repeat 3-6 until no score improvements are observed for a certain number of iterations
End

Listing 2: The outline of the SSPC algorithm.

There are three main differences between SSPC and the previous partitional approaches for projected clustering, PROCLUS and ORCLUS. First, the seeds are determined based on some domain knowledge (labeled objects and labeled dimensions) if supplied, which is potentially more accurate. Second, unlike PROCLUS and ORCLUS where the selected dimensions are determined based on some distance calculations that involve all dimensions, the seeds of each cluster are associated with an estimated set of relevant dimensions determined during initialization. When a seed is picked as the medoid of a cluster, either initially or to replace an old cluster representative, the associated dimensions become the selected dimensions of the cluster. As to be seen later, this process does not rely on distance calculations that involve all dimensions or any user parameters that are critical to the clustering accuracy but whose values are hard to determine. This allows SSPC to identify low-dimensional clusters more accurately. Third, after each iteration, besides replacing a bad cluster representative by a new medoid, other cluster representatives are also replaced by the cluster medians to avoid problems due to the potential biased projected values of the medoids as discussed before.

In the coming subsections some core mechanisms of SSPC will be discussed in detail, followed by an overall analysis of the whole algorithm.

4.1. Determining $\hat{s}_{ij}^2$

In order to use the SelectDim procedure and calculate the objective score ϕ, we need to determine the values of $\hat{s}_{ij}^2$. As mentioned in Section 3, $\hat{s}_{ij}^2$ should be greater than the sample variance s_{ij}^2 for all selected dimensions. Since the expected variance of a random sample of a population is the variance of the population, if s_{ij}^2 is no smaller than the variance of the global population, σ_j^2, the members of C_i are no more similar to each other along v_j than a group of random objects. So, σ_j^2 can be viewed as a baseline (maximum) value of $\hat{s}_{ij}^2$, whose value can be estimated by the sample variance of D_j, hereafter denoted as s_j^2.

We propose two schemes to set the actual value of $\hat{s}_{ij}^2$. The first scheme is to set it to ms_j^2, where $m \in (0, 1]$ is a user parameter. A smaller m tightens the selection criterion. Although this scheme is very simple, the resulting clustering accuracy is very stable across a wide range of m values as to be shown later by the experimental results.

The second scheme is based on a probabilistic reasoning. Users are asked to specify a value p that bounds the maximum probability that a dimension irrelevant to a cluster is selected by chance. Suppose C_i is a cluster to which dimension v_j is irrelevant. We can write $p = Pr(s_{ij}^2 < \hat{s}_{ij}^2)$. If the sampling distribution of s_{ij}^2 has a known probability density function (PDF), the value of $\hat{s}_{ij}^2$ can be computed accordingly.

For example, suppose the global populations are Gaussian. Then the random variable $\frac{(n_i-1)s_{ij}^2}{\sigma_j^2}$ has a chi-square distribution with $n_i - 1$ degrees of freedom. With p specified and σ_j^2 approximated by s_j^2, the value of $\hat{s}_{ij}^2$ can be computed from the inverse of the cumulative chi-square distribution.

The first scheme is more generic as it does not need to assume the properties of the global populations. But in case the sampling distribution of s_{ij}^2 has a known PDF, the second scheme is more recommended as parameter p has a stronger intuitive meaning than m.

Notice that although $\hat{s}_{ij}^2$ can take different values for different C_i (when parameter p is used) and v_j (in both cases), the objective function and the SelectDim procedure involves only one user parameter whose value is not critical to the clustering accuracy, which makes SSPC quite robust. According to the experimental results to be presented in Section 5, the range of values of m and p that give good clustering results is usually much wider than the range of l values (average cluster dimensionality) used by PROCLUS. In general, some reasonable values (e.g. $0.3 \leq m \leq 0.7$, $0.01 \leq p \leq 0.2$) can be used when the user has no ideas on the proper value to use. The value can be tuned down if the clusters appear to have too many selected dimensions, or tuned up if too many objects are discarded as outliers.

4.2. Initialization

In the initialization step, SSPC needs to determine the seeds. Unlike most k-medoids methods like CLARANS [13] where each cluster draws its medoid from a single pool of seeds, SSPC puts the seeds into different seed groups. Each seed group contains a set of seeds that are expected to come from a single real cluster. From the seeds, the set of relevant dimensions of the cluster are estimated. Each time a seed of a seed group is picked as the medoid of a cluster, the set of dimensions of the seed group is used as its selected dimensions.

SSPC creates two kinds of seed groups: private and public. For a cluster with input knowledge, it is easier to form an accurate seed group (in terms of both the seeds and the estimated relevant dimensions). A private seed group is thus created, which is solely used by the cluster. All other clusters share some public seed groups, where the number of groups is larger than the number of such clusters, so that medoids can be drawn from different seed group combinations. Whenever a cluster needs to draw a new medoid, it is randomly drawn its private seed group if it has, or otherwise one of the public seed groups not currently used by other clusters.

The order of seed group creation is important. Having created the seed groups of some clusters accurately, it becomes easier to accurately create the remaining seed groups. This is because objects that are close to the seeds of the previous seed groups in the corresponding subspaces are likely members of those clusters, which need not be considered when determining the seeds of the new seed groups. We will show in Section 4.5 that it is easier to create seed groups accurately for clusters with more input knowledge, which suggests that clusters with more input knowledge should have their seed groups created earlier. Based on the ease of creating accurate seed groups, SSPC creates seed groups in the following order: 1) clusters with inputs in both I^o and I^v, 2) clusters with inputs in I^o only, 3) clusters with inputs in I^v only, 4) clusters with no inputs. Within each category, clusters with larger amount of inputs are initialized earlier.

We now discuss the details of the seed group creation process for the four cases separately. For the first three cases, a private seed group is created, and we will use C_i to denote the target cluster, G_i to denote the resulting seed group, and I_i^o and I_i^v as the sets of labeled objects and labeled dimensions for C_i respectively. Since G_i contains both a set of objects (the seeds) and a set of estimated relevant dimensions, we will sometimes treat it as a cluster for the sake of discussion.

4.2.1. Clusters with both kinds of inputs In this case, we have several pieces of important information. First, the center of C_i is likely to be located near the median of I_i^o. While the objects in I_i^o can be biased towards one side of C_i, they should be still relatively close to the center of C_i as compared to most other objects. Second, if the set of objects in I_i^o is viewed as a temporary cluster $C_{i'}$, the dimensions with larger $\phi_{i'j}$ values are more likely to be the relevant dimensions of C_i. Third, as we assume all input knowledge is correct, all dimensions in I_i^v should be included in the set of relevant dimensions of the seed group.

This leads to a two-step process of seed group creation. We first identify the seeds in G_i to be the objects in the dataset that are close to the median of I_i^o along dimensions with large $\phi_{i'j}$ values. Then we set the relevant dimensions of the seed group to be those selected by SelectDim(G_i) plus the ones in I_i^v. We use the dimensions from SelectDim(G_i) rather than those from SelectDim($C_{i'}$) as the number of seeds can be controlled to be larger than the number of objects in I_i^o, so the former can probably give a more accurate estimate of the relevant dimensions.

We developed a mechanism for the first step based on a simple idea. If we form a grid (multi-dimensional histogram) of the whole dataset using a fixed number of dimensions, then if the dimensions are all relevant to C_i, a cell will be found to contain a large number of objects, which correspond to the center of C_i in the subspace. If some of the dimensions are irrelevant to C_i, the peak density (the highest number of objects among the cells) will be much lower. So, multiple grids are built using different sets of dimensions, and the objects in the peak cell with the highest density are chosen as the members of G_i.

There are several issues to consider. First, the number of dimensions used to build each grid should not be too large as the number of cells increases exponentially as the number of building dimensions increases, which makes each cell to have too few objects and creates a heavy computational overhead. Normally a three-dimensional grid serves the purpose quite well.

The second issue is which dimensions should be used to build the grids. The basic principle is to allow dimensions with greater chance of being relevant to C_i to have higher probabilities of being involved in grid building. Therefore we define a candidate set that includes the dimensions selected by SelectDim($C_{i'}$) as well as those in I_i^v, where each dimension v_j in the set has a probability proportional to the $\phi_{i'j}$ value of being selected as a building dimension of a grid.

The third issue is that a set of grid-building dimensions could be simultaneously relevant to multiple clusters, in which case the grid will contain multiple peaks. Since the cluster center is expected to be close to the median of I_i^o, instead of looking for the absolute peak of the whole grid we perform a localized hill-climbing search starting from the cell that contains the median of I_i^o. The search also solves the problem that the median may be biased towards one side of C_i.

As $\phi_{i'j}$ involves the computation of the sample variance $s_{i'j}^2$, I_i^o should contain at least two objects.

4.2.2. Clusters with labeled objects only The seed group creation process is almost the same as in the previous case, except that only dimensions from SelectDim($C_{i'}$) are involved in grid building and only those from SelectDim(G_i) are set as the relevant dimensions of the seed group.

4.2.3. Clusters with labeled dimensions only In this case, the temporary cluster $C_{i'}$ cannot be formed. Only dimensions in I_i^v are involved in grid building, and each of them has the same probability of being involved. Without $C_{i'}$, there is no starting point for the localized hill-climbing search, so the seeds are chosen from the objects in the absolute peak of the whole grid.

4.2.4. Clusters with no inputs In this case, we cannot build the grids directly due to the lack of input knowledge. An alternative mechanism that makes use of the information of other seed groups is developed. It is similar to the max-min mechanism of PROCLUS [1], which identifies an object whose minimum distance to all the seeds already picked by other seed groups is maximum. Distance calculations are performed in the subspace defined by the relevant dimensions of the seed groups, normalized by the number of dimensions. The identified object is remote from all picked seeds, so there is a good chance that it belongs to one of the clusters whose seed group has not been created. It is thus used to replace the median of I_i^o in the previous cases as the starting point of the hill-climbing search.

An one-dimensional histogram is then constructed for each dimension of the dataset to measure the object density around the identified object along the dimension. A dimension with high object density around the identified object is likely to be relevant to a cluster that centers around the object along the dimension. The dimension is thus given a high probability of being involved in grid building. With these probabilities determined, the seed group creation process for the case with labeled objects only can be performed.

In the extreme case, if all clusters receive no input knowledge, the object of the first seed group is randomly drawn from the whole dataset instead of using the max-min mechanism.

4.3. Cluster Representatives Replacement

To improve the objective score, SSPC needs to identify a bad cluster and replace its cluster representative appropriately. The most common situation where a bad cluster exists is that the medoids drawn by two clusters belong to the same real cluster. As a result, the two clusters will be quite similar, and they compete for the ϕ_i score of the real cluster. On the other hand, one of the other real clusters will not be represented by any cluster, and most of its members will be put on the outlier list.

There are several ways to detect the occurrence of this situation. One is to check whether there exists a cluster with a very low ϕ_i score, either absolutely (compared to the maximum possible score) or relatively (compared to the score of other clusters). The cluster is likely the loser of the two competing clusters. The other way is to check if there exists two clusters that are very similar by calculating the change of ϕ_i score if two clusters are merged into one. If two clusters have medoids belonging to the same real cluster, the resulting score will approach the sum of the two original scores. Otherwise the resulting score will be much lower, since few dimensions can be selected if the members of two different real clusters are mixed. In either case, a bad cluster can be pinpointed and its cluster representative can be replaced by a new medoid randomly drawn from one of the associated seed groups currently not used by other clusters.

For each other cluster, although the medoid may really be a member of the cluster, it may not be close to the cluster center along some relevant dimensions. As a result some real members located at the other side of the cluster may not be attracted to this cluster. The cluster can be improved by replacing the medoid with the median of the cluster, which is probably closer to the center of the real cluster.

4.4. Complexity

We divide the complexity analysis of SSPC into two parts: seed groups creation and actual clustering. A substantial amount of time in seed groups creation is spent on the building of grids, which involve projecting $O(n)$ objects into the cells. With the number of building dimensions and the number of bins per dimension fixed at constant, the time for building a grid and searching for the cell with peak density is $O(n)$. With the number of grids to build per cluster also fixed at constant, the total time spent in grid building and searching is $O(kn)$. To estimate the relevant dimensions, the ϕ_{ij} scores are to be computed, which take $O(n_i d)$ for each cluster C_i (with

the selection thresholds precomputed when parameter p is used). The total time complexity for seed groups creation is thus $O(knd)$. With the size of each grid fixed by constants, the space complexity depends on the size of the dataset, which is $O(nd)$.

In actual clustering, assigning each object involves computing the resulting scores of k cluster, which take a total time of $O(kd)$ (since the local means and variances can be computed in an incremental fashion by using cluster features (CF) [27]). Each iteration thus takes $O(knd)$ time in object assignment and objective score calculations. Cluster refinements require the computation of medians, each of which can be done in $O(n)$ time in a brute force way when there are $O(n)$ projected values. The total time of an iteration is therefore $O(knd)$. The overall time complexity of the whole algorithm, depends on the number of iterations, which is an unknown value that can vary in different runs. As in most partitional clustering algorithms, we observe from experimental results that it does not grow more than linearly with k, n or d. So we suggest that the overall time complexity of the whole SSPC algorithm is also $O(knd)$. The space complexity of actual clustering is $O(nd)$, which is also the space complexity of the whole algorithm.

The linear time and space complexities make it more practical to cluster large datasets as compared to some other projected clustering algorithms such as DOC, HARP, and ORCLUS.

4.5. How much input is needed?

In many real situations the amount of available domain knowledge is very limited, or it is very costly to obtain such knowledge. It is therefore important to predict the relationship between the amount of input knowledge and the resulting clustering accuracy, so as to minimize the amount of input knowledge while getting a satisfactory accuracy.

We begin with the case where only labeled objects are available. Suppose a certain cluster C_i receives $|I_i^o|$ labeled objects. The objects form a temporary cluster, which is used to determine the grid-building dimensions. We want to estimate the probability that at least one grid is built from dimensions that are really relevant to C_i only, which is crucial to the accuracy of the seed group and the clustering accuracy in turn.

Assume the global distributions are Gaussian, and parameter p is used in computing the values of $\hat{s}_{ij}^2$. By definition, the probability that a dimension irrelevant to C_i being selected by the temporary cluster is p. As discussed in Section 4.1, the value of $\hat{s}_{ij}^2$ can be computed from the inverse of the cumulative chi-square distribution. Now, suppose dimension $v_{j'}$ is relevant to C_i. The probability that $v_{j'}$ is selected by the temporary cluster, $q = Pr(s_{ij'}^2 < \hat{s}_{ij'}^2) = Pr(\frac{(|I_i^o|-1)s_{ij'}^2}{\sigma_{j'}^2} < \frac{(|I_i^o|-1)\hat{s}_{ij'}^2}{\sigma_{j'}^2})$, where $\sigma_{j'}^2$ is the variance of the local population of the underlying class of C_i along $v_{j'}$. If $\sigma_{j'}^2$ can be estimated (e.g. from the labeled objects and a histogram on $v_{j'}$), q can be estimated accordingly from the cumulative chi-square distribution.

Let d_i be the number of relevant dimensions of C_i, the number of "true positives" (selected dimensions that are actually relevant to C_i) is thus a binomial random variable with d_i trials and probability q. Similarly, the number of "false positives" (selected dimensions that are actually irrelevant to C_i) is a binomial random variable with $d-d_i$ trials and probability p. Denote t and f as the number of true positives and false positives respectively. Suppose each grid is built from c dimensions, and each dimension selected by the temporary cluster has the same chance of being involved in building a grid, the probability that a grid is built with all c dimensions being relevant to C_i is:

$$\alpha = \sum_{t=c}^{d_i} \sum_{f=0}^{d-d_i} \binom{d_i}{t} q^t (1-q)^{d_i-t} \binom{d-d_i}{f} p^f (1-p)^{d-d_i-f} \frac{\binom{t}{c}}{\binom{t+f}{c}}, \tag{5}$$

where the notation $\binom{n}{r}$ represents the number of ways of picking r unordered outcomes from n possibilities. If g grids are built for each cluster, the probability that at least one of them involves relevant dimensions only is $1-(1-\alpha)^g$. To visualize the change of this value with different input sizes, let us consider some real values to be used in the experiments in Section 5. Suppose $d = 3000$, $p = 0.01$, $c = 3$, $g = 20$, and the variance ratio of a local population to the corresponding global population is 0.15. We vary $|I_i^o|$ from 2 to 15, and the ratio $\frac{d_i}{d}$, from 1% to 20%. Figure 1 shows the estimated probabilities that at least one grid is formed by relevant dimensions only.

The figure shows that for a fixed $\frac{d_i}{d}$ ratio, having more input objects increases the probability of forming a grid by relevant dimensions only. In addition, each curve is observed to have a sharp increase followed by a flattened region. This means users can estimate the smallest amount of input that can lead to a near maximal accuracy. It is an exciting result to see that when $\frac{d_i}{d} = 5\%$, only 5 inputs are enough to have an almost 100% guarantee

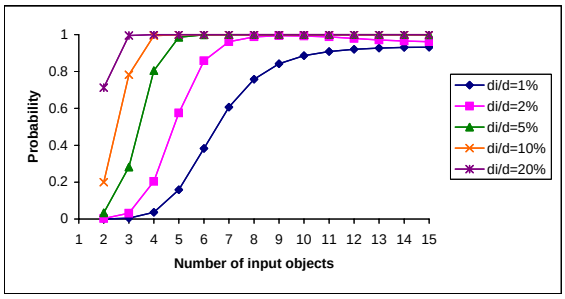

Figure 1: The probability that at least one grid is formed by relevant dimensions only, when only labeled objects are available.

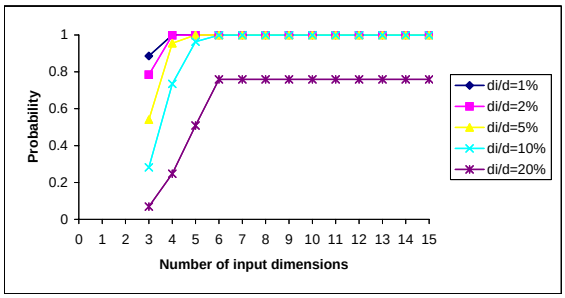

Figure 2: The probability that at least one grid with all c building dimensions being relevant to C_i only is formed, when only labeled dimensions are available.

that a grid will be formed by relevant dimensions only. The figure also shows that for a fixed amount of input, the probability increases as $\frac{d_i}{d}$ increases, which suggests that input objects work better when the clusters have more relevant dimensions.

Next, we consider the case where only labeled dimensions are available. Suppose dimension v_j is specified as relevant to a cluster C_i. If an one-dimensional histogram is built from v_j, we expect to find a peak at the center of C_i. If the cell with the highest object density is not close to the center of C_i, most probably v_j is also relevant to another cluster. Suppose there are k clusters, and on average each cluster has d_i relevant dimensions. Then the probability that v_j is also relevant to one or more other clusters is $1-(1-\frac{d_i}{d})^{k-1}$. Denote this probability as γ, the probability of forming a c-dimensional grid with all c dimensions being relevant to C_i only is $(1-\gamma)^c$. We want to estimate the probability of forming at least one such grid when g grids are built, which is a good indicator of the chance of forming an accurate seed group.

Suppose there are $|I_i^v|$ input dimensions, and no two grids are allowed to use exactly the same set of building dimensions. When $\binom{|I_i^v|}{c} \leq g$, the required probability is $1-(1-(1-\gamma)^c)^{\binom{|I_i^v|}{c}}$. When $\binom{|I_i^v|}{c} > g$, it becomes $1-(1-(1-\gamma)^c)^g$. Using the same parameter values as before, and setting $k=5$, the estimated probabilities at various $|I_i^o|$ and $\frac{d_i}{d}$ values are shown in Figure 2.

In general, the more labeled dimensions being supplied, the higher is the chance of forming a grid with all building dimensions being relevant to C_i only. The figure also reveals an interesting phenomenon: while labeled objects work better when $\frac{d_i}{d}$ is large, labeled dimensions work better when it is small as the chance for a single dimension to be relevant to multiple clusters is small. This suggests that when trying to identify clusters with extremely low dimensionality, which is the main focus of this study, it is more effective to use labeled dimensions as input knowledge.

Both inferences show that a very small amount of input knowledge would enhance the accuracy a lot. Finally, since the two kinds of input complement each other, there is a synergy when they are supplied at the same time, provided the amount of input objects is not so small that causes a large amount of irrelevant dimensions to be

used in building the grids. Some empirical results will be presented in the next section.

5. Experiments

In this section we present various experimental results on SSPC and some comparing algorithms. We first describe the algorithms involved, the general setting, and the performance metric. Then we present the details of the experiments in subsequent subsections.

We compare the accuracy of three projected clustering algorithms HARP [25], PROCLUS [1] and SSPC, using the non-projected k-medoids clustering algorithm CLARANS [13] as reference. We do not include ORCLUS since we consider only axis-parallel clusters, and it has a long execution time on large datasets. We have also tried to include FastDOC in the experiments, but there have been some great difficulty in setting its parameter values. In some preliminary experiments, FastDOC rapidly moved from putting all objects to a single cluster to discarding a lot of outliers (objects that are not included in the first k clusters) with a slight change of the value of the parameter β. We therefore focused on the four algorithms listed above, and left the more thorough comparisons between different projected clustering algorithms to a future study.

HARP always produces the same clusters when clustering a dataset multiple times using the same set of parameter values. This is not true for the other three algorithms, which all use random numbers in some procedures. We therefore repeated each experiment ten times, and report only the result that gives the best algorithm-specific objective score.

The primary performance metric used in evaluating the quality of a clustering result is the Adjusted Rand Index (ARI) [24] that validates the produced clusters by the known real clusters. It is based on the Rand Index [18], with the expected index value also taken into account. It measures how similar are the partition of objects according to the real clusters (U) and the partition in a clustering result (V). Denote a, b, c and d as the number of object pairs that are in the same cluster in both U and V, in the same cluster in U but not V, in the same cluster in V but not U, and in different clusters in both U and V respectively, ARI is defined as follows:

$$ARI(U,V) = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)} \tag{6}$$

The more similar are the two partitions (larger a and d, smaller b and c), the larger will be the ARI value. When U and V are identical, the index value will be one. When V is only as good as a random partition, the index value will be zero.

If input knowledge is involved in a run of SSPC, the labeled objects are removed from the resulting clusters before computing the ARI values in order to eliminate the direct performance gain due to the input objects.

5.1. Raw accuracy

In the first set of experiments we compared the raw accuracy of the algorithms, i.e., the accuracy without using any input knowledge. A series of synthetic datasets were generated with $n = 1000$, $d = 100$ and $k = 5$. The actual average dimensionality of the clusters, l_{real}, varies from 5 to 40, accounting for $5\% - 40\%$ of the dataset dimensionality. The datasets were generated according to the data model described in Section 3, with the global distributions being uniform and the local distributions having variances ranging from $1\% - 10\%$ of the value range of the global distributions.

We set k to 5 for all algorithms, used default parameter values for HARP and CLARANS, and tried different values of the critical parameters of PROCLUS and SSPC. For PROCLUS, we tried 9 different values of l for each dataset. For SSPC, 5 different values of m and p were used for each dataset. We present the results in two different figures. In Figure 3, the best results (the results with the highest ARI values) after trying different parameter values are shown. In Figure 4, the best and average results of PROCLUS and SSPC are shown for the comparison of their relative robustness.

Figure 3 shows that all projected clustering algorithms performed well when the cluster dimensionality is high. In comparison, CLARANS, being a non-projected algorithm, has a much lower accuracy. When the dataset dimensionality is as low as 10% of d, the best performance of PROCLUS begins to drop. At 5%, the performance of

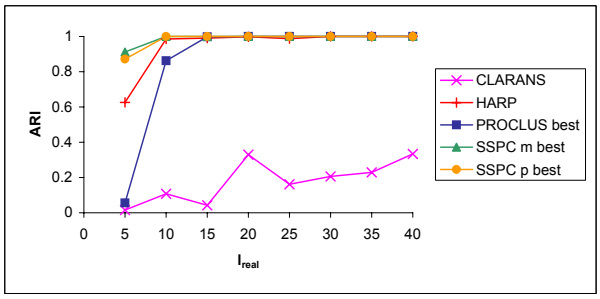

Figure 3: The best raw accuracies of the algorithms on datasets with various average cluster dimensionality.

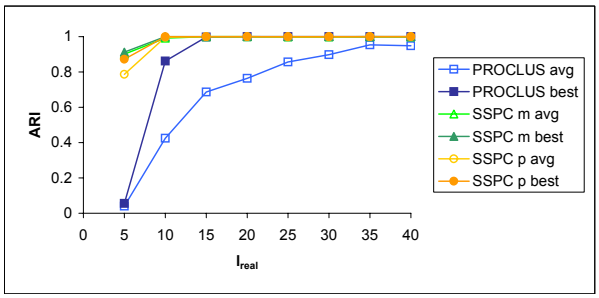

Figure 4: A comparison of the best and average raw accuracies of PROCLUS and SSPC on datasets with various average cluster dimensionality.

all three projected clustering algorithms went down, but SSPC has the mildest performance drop, no matter parameter m or parameter p is used. It is somewhat unexpected that the raw performance of SSPC when parameter p is used is close to the performance when parameter m is used, given the global distributions are actually non-Gaussian. This may due to the fact that except the dimension selection mechanism, SSPC makes no assumptions on the global distribution. The performance of the other parts of the algorithm may compensate for the invalid assumptions being made when parameter p is used.

Figure 4 shows that SSPC is more robust than PROCLUS in that the average accuracies when different parameter values are used are much closer to the best accuracies. The individual clustering results when $l_{real} = 10$ are shown in Figure 5. PROCLUS performed well when the value of l was supplied correctly, but the performance went down as the input moved away from the true value. In contrast, SSPC performed well with the various parameter values being tried, and is thus more robust.

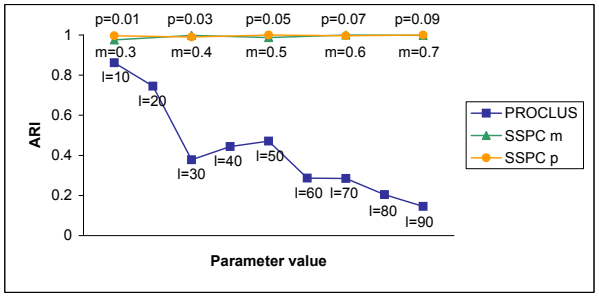

Figure 5: The raw accuracy of PROCLUS and SSPC on the dataset with $l_{real} = 10$ using various parameter values.

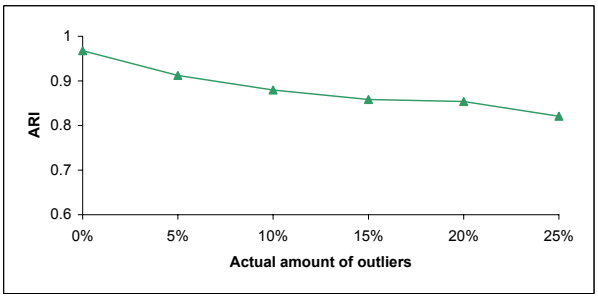

Figure 6: The accuracy of SSPC on datasets with various amount of outliers.

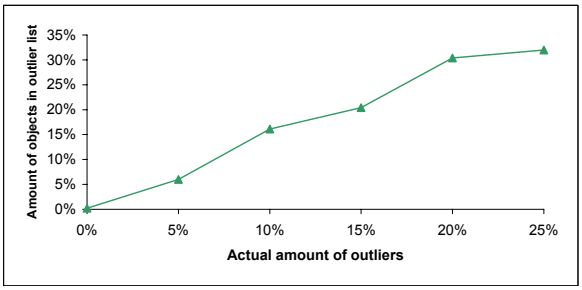

Figure 7: The amount of objects put in the outlier list by SSPC when clustering the datasets with various amount of outliers.

5.2. Outlier Immunity

In this set of experiments we studied how SSPC is affected by outliers. A series of synthetic datasets were generated with $n = 1000$, $d = 100$, $k = 5$, and $l_{real} = 5$. The amount of outliers varies from 0% to 25%. We set m to the arbitrary value of 0.5. The clustering accuracies and the amount of objects on the outlier lists are shown in Figure 6 and Figure 7 respectively.

The results show that SSPC has a high noise-immunity, with only moderate accuracy decrease as the amount of outliers increases. The amount of objects detected as outliers also highly resembles the actual amount of outliers in the datasets.

5.3. Performance with input knowledge

Although SSPC has an encouraging raw accuracy, its performance also drops when l_{real} is very small. In this set of experiments, we further lower the average cluster dimensionality and see if the accuracy of SSPC can be improved by input knowledge. We generated a dataset with $n = 150$, $d = 3000$, $k = 5$ and $l_{real} = 30$, i.e., 1% of d. The configuration highly resembles a real gene expression dataset when the goal is to cluster the samples, and the number of relevant genes of each sample class is as low as 1% of d. We set $m = 0.5$, and tried 5 different coverage ratios (fraction of clusters receiving inputs), 4 input categories (no inputs, I^o only, I^v only, both), and 8 different input sizes. For example, when coverage=0.6, both kinds of inputs are supplied and input size=4, $0.6 \times 5 = 3$ clusters receive input knowledge, each with 4 labeled objects and 4 labeled dimensions. No inputs are supplied for the other 2 clusters.

The inputs are drawn randomly from the real cluster members and relevant dimensions. Each point in the coming figures is the median of 10 repeated runs with 10 independent sets of inputs.

Figure 8 shows the accuracy of SSPC when coverage=1. For reference, the ARI values of HARP and PROCLUS (with correct l value supplied) are 0.17 and 0.08 respectively, which are much lower than the raw accuracy of SSPC (at input size 0). In general, SSPC has a larger accuracy improvement when more inputs are supplied. The

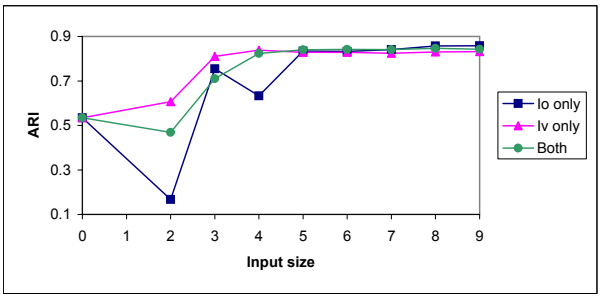

Figure 8: The accuracy of SSPC with various amount of input knowledge when coverage is 1.

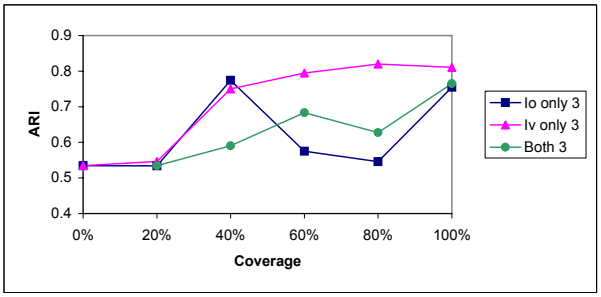

Figure 9: The accuracy of SSPC with various coverage of input knowledge when input size is 3.

accuracy becomes stable with 5 objects and 3 dimensions (which is equal to the default value of c, the number of building dimensions per grid). All these observations are consistent with our earlier analysis in Section 4.5. The accuracy of SSPC appears to be more stable with labeled dimensions as inputs. In particular, an accuracy lower than the raw accuracy is observed when only 2 labeled objects are supplied to each cluster. This is due to the large probability that the two objects are close to each other along many irrelevant dimensions, which seriously misleads the dimension selection mechanism (see Figure 1 for reference). On the other hand, the probability for a pair of dimensions to be relevant to multiple clusters is much lower due to the low average cluster dimensionality, which results in an observable accuracy improvement when 2 labeled dimensions are supplied.

Figures 9 and 10 show the accuracy of SSPC with changing coverage, when the input size is 3 and 6 respectively. Again, there is a general trend of increasing accuracy when the coverage increases, and the increasing trend is more stable at the larger input size of 6. An interesting observation from Figure 10 is that the peak performance is reached at 60% coverage, which suggests that it is not necessary to input domain knowledge to every cluster. By using the max-min mechanism (Section 4.2.4), clusters with no input knowledge could also locate their cluster centers provided the seed groups of the other clusters are created accurately. During object assignment, if members of the clusters with input knowledge are assigned correctly, the chance for the remaining clusters to identify their members and relevant dimensions correctly is also increased.

5.4. Datasets with multiple possible groupings

As discussed in Section 1, an important application of semi-supervised clustering is to produce different desired clusters based on different input knowledge. In this set of experiments we verify the capability of SSPC in achieving this. We generated two datasets, each with $n = 150$, $d = 1500$, $k = 5$ and $l_{real} = 30$. The members and relevant dimensions of the clusters in the two datasets are independent. We then combined the two datasets to produce a dataset with 3000 dimensions, where the first 1500 come from the first original dataset and the last 1500 come from the second. The average cluster dimensionality thus remains at 1% of d. We then tested the accuracy of HARP, PROCLUS and SSPC on the dataset, with correct l value supplied to PROCLUS. For SSPC, we tested its accuracy in three different scenarios: without inputs (raw accuracy), input based on the knowledge of the first

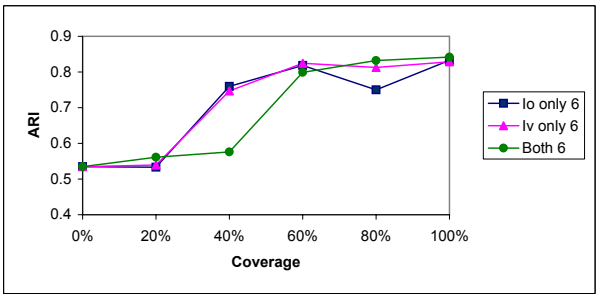

Figure 10: The accuracy of SSPC with various coverage of input knowledge when input size is 6.

Table 1: The accuracy of the algorithms on the dataset with two possible groupings. The two sets of ARI values are computed from the actual clusters of the two original datasets respectively.

Algorithm	ARI (set 1)	ARI (set 2)
HARP	0.054	0.012
PROCLUS	0.280	0.027
SSPC (no inputs)	0.446	0.001
SSPC (I^o only, size=3)	0.329	0.397
SSPC (I^o only, size=6)	0.841	0.847
SSPC (I^v only, size=3)	0.826	0.818
SSPC (I^v only, size=6)	0.833	0.834
SSPC (I^o and I^v, size=3)	0.750	0.669
SSPC (I^o and I^v, size=6)	0.842	0.846

original dataset, and input based on the knowledge of the second original dataset. The ARI values of the algorithms computed from the actual clusters of the two original datasets are shown in Table 1.

The performance of HARP is seriously affected by the simultaneous existence of two possible groupings. Objects not in the same cluster can be close to each other along many dimensions (due to the fact that they do belong to the same cluster in the other grouping), which ruin the threshold loosening mechanism of HARP. The performance of PROCLUS is better, but is still not very encouraging. The raw accuracy of SSPC is better than HARP and PROCLUS when evaluated by the first set of clusters, but worse when evaluated by the second set. This shows that without any external inputs, SSPC tends to form clusters that are more similar to the first set. But as some external inputs were supplied, the accuracy of SSPC was significantly improved in both cases. The results confirm the importance of external inputs in guiding the formation of some desired clusters when there are multiple possible groupings.

5.5. Scalability

In this set of experiments we tested the scalability of SSPC with respect to an increasing dataset size (n) and dimensionality (d). We generated two series of datasets, the first with $d = 100$, and n varies from 1000 to 100000. The second has $n = 150$, and d varies from 100 to 10000. We measured the execution time of 10 repeated runs of each experiment, using the execution time of PROCLUS as reference. We prefer to report the time of 10 repeated runs in order to reduce the random variation of execution time in each run. Also, in real situations, repeated runs are usually required to obtain satisfactory results. The execution time of the two algorithms are shown in Figures 11 and 12.

The figures confirm the linear time complexity of SSPC with respect to both n and d. Its speed is comparable to PROCLUS in our implementations. With input knowledge, the execution time of SSPC can be further reduced as fewer histograms are built, the histogram searches are localized, and fewer iterations are required before the (locally) optimal objective score is reached.

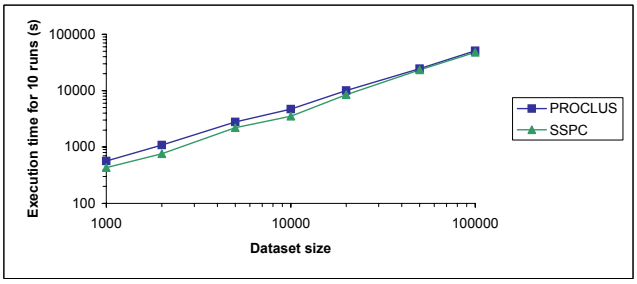

Figure 11: The execution time of PROCLUS and SSPC with increasing dataset size.

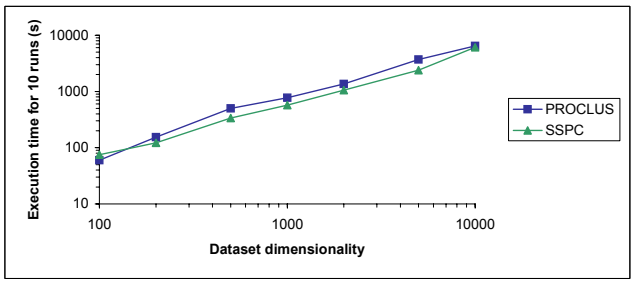

Figure 12: The execution time of PROCLUS and SSPC with increasing dataset dimensionality.

6. Summary and Future Extensions

In this paper, we have discussed some potential limitations of some existing projected clustering algorithms, including their inability to detect clusters with very low dimensionality, the use of user parameters whose proper values are hard to determine, and the potential accuracy drop when improper parameter values are supplied.

We have proposed a new projected clustering algorithm that is robust and is able to detect clusters of extremely low dimensionality as it uses a robust objective function and avoids distance calculations that involve all the dimensions. In addition, we have proposed ways to utilize any available domain knowledge in the form of labeled objects and labeled dimensions. Experimental results show that there is a clear accuracy improvement when some input knowledge is incorporated in the clustering process. The peak performance is readily reached when only a small amount of knowledge is supplied, and when the knowledge covers only some of the classes.

There are some obvious directions for further study. The most important one is to test the new algorithm on some real datasets that are expected to contain projected clusters, such as gene expression profiles. When applying to complex, noisy real data, the data model and objective function may have to be revised according to the observed data properties.

Another direction is to allow incorrect inputs. When inputs could be incorrect, they have to be validated before being used to guide the clustering process, for example by comparing the assumed data model and the observed data values. It is also possible to study fuzzy inputs, each of which contains a confidence level that indicates its chance of belonging to a cluster, and/or a quality level that specifies the chance for the object to be of a certain distance from the cluster center.

The current study is focused on distance-based clustering, i.e., the similarity between different objects is measured by a distance function. There are more and more applications where similarity is measured by other means, such as the rise and fall pattern of values. Since most semi-supervised clustering algorithms are based on some distance-based methods such as k-means and k-medoids, some adaptations may be required when trying to incorporate domain knowledge in pattern-based clustering.

It is also interesting to study the case where one class corresponds to multiple clusters. This can occur when, for example, some samples from the same type of tumor actually belong to different subtypes. The samples are

best modeled by multiple clusters with different properties, some of which being common to all the clusters. The simplest way to handle such datasets is to assign a different sub-class label to each cluster, and treat each cluster as originating from a different class. But if the corresponding domain knowledge is not available, one may resort to forming a single cluster for each class, based on the common properties. One interesting algorithm for non-projected clustering that follows this direction is described in [11], which modifies the distance matrix to artificially move objects of the same class towards each other. A more general approach is to allow the formation of multiple clusters per class, which is currently not yet fully studied.

References

[1] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *ACM SIGMOD International Conference on Management of Data*, 1999.

[2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD International Conference on Management of Data*, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD International Conference on Management of Data*, 1998.

[4] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[5] S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the SIAM International Conference on Data Mining*, 2004.

[6] S. Basu, M. Bilenko, and R. J. Mooney. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.

[7] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 2000.

[8] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback, 2000. Unpublished.

[9] A. Demiriz, K. P. Bennett, and M. J. Embrechts. Semi-supervised clustering using genetic algorithms. In *Artificial Neural Networks In Engineering*, 1999.

[10] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28, 1979.

[11] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[12] T. M. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[13] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, 1994.

[14] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu. MaPle: A fast algorithm for maximal pattern-based clustering. In *IEEE International Conference on Data Mining*, 2003.

[15] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. H. Kim, L. C. Goumnerovak, P. M. Black, C. Lau, J. C. Allen, D. Zagzag, J. M. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. P. Mesirov, E. S. Lander, and T. R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415:436–442, 2002.

[16] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *ACM SIGMOD International Conference on Management of Data*, 2002.

[17] J. R. Quinlan. *C4.5 Programs for Machine Learning.* Morgan Kaufmann, 1993.

[18] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[19] L. Talavera and J. Bejar. Integrating declarative knowledge in hierarchical clustering tasks. In *International Symposium on Intelligent Data Analysis*, 1999.

[20] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[21] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[22] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *ACM SIGMOD International Conference on Management of Data*, 2002.

[23] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, 2003.

[24] K. Yeung and W. Ruzzo. An empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[25] K. Y. Yip, D. W. Cheung, and M. K. Ng. HARP: A practical projected clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 2004. to appear.

[26] K. Y. L. Yip. HARP: A practical projected clustering algorithm for mining gene expression data. Master's thesis, The University of Hong Kong, December 2003. http://www.cs.hku.hk/~ylyip/papers/thesis.pdf.

[27] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *ACM SIGMOD International Conference on Management of Data*, 1996.