Neurocomputing 92 (2012) 133–144

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Clustering in applications with multiple data sources—A mutual subspace clustering approach

Ming Hua [a], Jian Pei [b,*]

[a] Facebook Inc., Palo Alto, CA, USA
[b] Simon Fraser University, Burnaby, BC, Canada

ARTICLE INFO

Available online 24 February 2012

Keywords:
Clustering
Multiple sources

ABSTRACT

In many applications, such as bioinformatics and cross-market customer relationship management, there are data from multiple sources jointly describing the same set of objects. An important data mining task is to find interesting groups of objects that form clusters in subspaces of the data sources jointly supported by those data sources.

In this paper, we study a novel problem of mining mutual subspace clusters from multiple sources. We develop two interesting models and the corresponding methods for mutual subspace clustering. The density-based model identifies dense regions in subspaces as clusters. The bottom-up method searches for density-based mutual subspace clusters systematically from low-dimensional subspaces to high-dimensional ones. The partitioning model divides points in a data set into k exclusive clusters and a signature subspace is found for each cluster, where k is the number of clusters desired by a user. The top-down method interleaves the well-known k-means clustering procedures in multiple sources. We use experimental results on synthetic data sets and real data sets to report the effectiveness and the efficiency of the methods.

1. Introduction

In many applications, there are multiple data sources. It is important to analyze data using the multiple data sources in an integrative way.

1.1. Motivation application examples and challenges

To develop effective therapies for cancers, both clinical data and genomic data have been accumulated for cancer patients. Examining clinical data or genomic data independently may not reveal the inherent patterns and correlations present in both data sets. Therefore, it is important to integrate clinical and genomic data and mining knowledge from both data sources.[1]

Clustering is a powerful tool for uncovering underlying patterns without requiring much prior knowledge about data. To discover phenotypes of cancer, subspace clustering has been widely used to analyze such data. However, in order to understand the clusters on clinical attributes well, and find out the genomic explanations, it is highly desirable to find clusters that are manifested in subspaces in both the clinical attributes and the genomic attributes. For a cluster mutual in a clinical subspace and a genomic subspace, we can use the genomic attributes to verify and justify the clinical attributes. The mutual clusters are more understandable and more robust. In addition, mutual subspace clustering is also helpful in integrating multiple sources.

As another example, consider cross-market customer relationship management. Customer behaviors in multiple markets (e.g., financial planning and investment, vacation expenditure, reading, entertainment and leisure expense) can be collected. Mutual subspace clustering can achieve more reliable customer segmentation. A mutual cluster which is a set of customers that are exclusively similar to each other in a subspace (i.e., some features) in each market is interesting, since we may use the features in different markets to explain their behaviors in the other markets. Mutual subspace clustering not only generates more robust clusters, but also integrates data from multiple sources and produces more understandable knowledge.

Recently, in a few applications such as bioinformatics, health-informatics and cross-market customer relationship management, attribute data about the same set of objects is collected from multiple aspects and/or sources. The availability of such data enables the verification and justification of learning from multiple sources, as demonstrated in recent research [14,15,26,27]. Particularly, joint clustering from multiple sources (e.g., [19,23,7]) which discovers clusters agreed by multiple sources has been found interesting and important in those applications.

* Corresponding author.
E-mail addresses: arceehua@fb.com (M. Hua), jpei@cs.sfu.ca (J. Pei).

[1] https://science.pfizer.com/content/we-link-genomics-and-clinical-data-for-enhancing-the-discovery-and-development-of-new-therapies/

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.08.032

In this paper, we study mining mutual subspace clusters for those applications with multiple data sources. In the clinical and genomic data analysis example, a mutual cluster is a subset of patients that form a cluster in both a subspace of the clinical data source and a subspace of the genomic data source. Such a mutual cluster may suggest the inherent connection between the genomic features and the clinical features.

Is mutual subspace clustering computationally challenging? One may consider the straightforward generate-and-test methods. For example, a simple method works in two steps. In the first step, we can find the complete set of possible subspace clusters in the first data source, say clinical data. Then, in the second step, we can check for each subspace cluster whether it is a subspace cluster in genomic data. Similarly, when clustering in the union space is feasible, we can first find all clusters in the union space with the constraint that the subspace of each cluster must contain at least one attribute from each clustering space. Then, we can check each cluster against the mutual clustering criteria.

However, such a two-step, generate-and-test method is problematic. Finding the complete set of possible subspace clusters in the clinical space or the union space of clinical data and genomic data is often very costly or even infeasible. For example, in the partitioning model (e.g., [1,2,25]), it is impossible to find all possible subspace clusterings. In some other models where clusters are not exclusive, there may be many subspace clusters in a large, high dimensional data set. Enumerating all possible subspace clusters explicitly and checking them one by one is often computationally expensive. In some models such as density-based clustering [3] and pattern-based clustering [24,20], enumerating all possible clusters is NP-hard.

1.2. Problem outline

While we will discuss the models of mutual subspace clustering in Sections 3.1 and 4.1, the problem can be generally described as follows.

We model a data source as a set of points in a clustering space. Let S_1 and S_2 be two *clustering spaces* where $S_1 \cap S_2 = \emptyset$, and O be a set of points in space $S_1 \cup S_2$ on which the subspace clustering analysis is applied. It is up to users to choose clustering spaces. The only requirement here is that each point appears in both clustering spaces.

A *mutual subspace cluster* is a triple (C,U,V) such that $C \subseteq O$, $U \subseteq S_1$, $V \subseteq S_2$, and C is a cluster in both U and V, respectively. U and V are called the *signature subspaces* of C in S_1 and S_2, respectively. To keep our discussion simple, we consider only two clustering spaces in this paper. However, our model can be easily extended to situations where more than two clustering spaces present.

What is the critical difference between mutual subspace clustering on multiple spaces and traditional subspace clustering in one space? Technically, one may think that we can find subspace clusters in the union space $S_1 \cup S_2$ with the constraint that the subspaces must contain attributes from both S_1 and S_2. Suppose C is a cluster in subspace $W \subseteq S_1 \cup S_2$. Then, we can assign $U = W \cap S_1$ and $V = W \cap S_2$ as the signature subspaces of C. Does this straightforward extension work?

Example 1 (*Mutual clustering*). Fig. 1 shows a synthetic data set. Let the clustering space S_1 be X and the clustering space S_2 be Y. The union space is the two-dimensional space as shown. There are three clusters (annotated as A, B and C in the figure) in the union space.

Mutual clustering from the clustering spaces S_1 and S_2 can help us to understand how the two attributes agree with each other in clusters. For example, cluster C is a good mutual cluster, since its projections on both S_1 and S_2 are also clusters. However, although

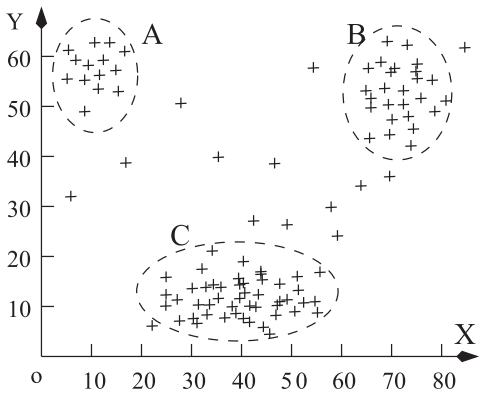

Fig. 1. An example of mutual clusters.

clusters A and B are clusters in the union space, each of them is not a distinguishing cluster in subspace S_2 (i.e., Y). They are mixed together in S_2. Thus, A and B are not good mutual clusters. □

Moreover, in real applications, different similarity measures and even clustering criteria may be adopted in different clustering spaces. In such a case, it is very difficult or even impossible to define an appropriate similarity measure and clustering criteria in the union space. Clustering in the union space becomes infeasible.

From the above example, we can see that mutual subspace clustering from multiple clustering spaces is critically different from subspace clustering in one (union) clustering space. A mutual cluster must be a cluster in a signature subspace of each clustering space. Mutual subspace clustering finds the common clusters agreed by subspace clustering in both clustering spaces, which cannot be handled by the traditional subspace clustering analysis.

In this paper, we study the mutual subspace clustering problem and make the following contributions. First, we identify the novel mutual subspace clustering problem, and elaborate its potential applications. Second, we develop two interesting models and the corresponding methods for mutual subspace clustering. The density-based model identifies dense regions in subspaces as clusters. The bottom-up method searches for density-based mutual subspace clusters systematically from low-dimensional subspaces to high-dimensional ones. Information from multiple sources is used to guide the search. The partitioning model divides points in a data set into k exclusive clusters and a signature subspace is found for each cluster, where k is the number of clusters desired by a user. The top-down method interleaves the well-known k-means clustering procedures in multiple sources. Third, we use experimental results on synthetic data sets and real data sets to report the effectiveness and the efficiency of the methods.

The rest of the paper is organized as follows. The related work is reviewed in Section 2. The density-based, bottom-up method and the partitioning, top-down method are developed in Sections 3 and 4, respectively. The experimental results are reported in Section 5. Section 6 discusses the related issues and concludes the paper.

2. Related work

Our work is generally related to subspace clustering, clustering from multiple sources, and multiview learning. In this section, we review those areas briefly.

2.1. Subspace clustering

Subspace clustering has attracted substantial interest due to its successful applications in some new domains such as

bioinformatics and image processing. Parsons et al. [18] provided a nice concise survey of the existing subspace clustering methods.

There are two major categories of subspace clustering methods, namely the top-down methods and the bottom-up methods. The top-down methods find an initial clustering in the full space, and then iteratively discover subspaces for the clusters and improve the clustering. PROCLUS [1], ORCLUS [2], FINDIT [25], δ-clusters [28], and COSA [11] are examples of the top-down methods.

The bottom-up methods identify dense regions in low-dimensional subspaces and extend those dense regions to form clusters. CLIQUE [3], ENCLUS [6], MAFIA [16], CBF [4], CLTree [13], DOC [21], and pattern-based clustering [24,20] are some examples.

In addition, some other clustering algorithms tackle the problem from different angles. For example, Ng et al. [17] developed a spectral clustering method base on the top eigenvectors derived from the pair-wise distance matrix among clustering objects. Zhou and Tao [29] proposed a fast gradient clustering approach by transforming the problem into a maximum eigenvalue minimization problem.

2.2. Clustering from multiple sources

Jointly mining clusters from multiple data sources has been emerging as a novel direction in the domain of clustering analysis. Some examples include [23,19,12,8,5]. Particularly, in [19], clusters on graphs are mined from multiple graphs. In [12], clustering is conducted in a space S_1, while some constraints specified in another space S_2 must be satisfied. Ester et al. [8] studied the problem of mining clusters in an integrated space consisting of both attribute data and relationship data (e.g., a graph). More recently, [5] studies the problem of clustering from multiple views (of the same set of objects), by projecting the data in each view to a lower dimensional subspace using Canonical Correlation Analysis (CCA).

Critically different from [23,19,12,8,5], mutual subspace clustering proposed in this paper searches clusters in *different* signature subspaces of multiple sources.

2.3. Multiview learning

Data with multiple representations from different feature spaces (also known as views) is inherent in various data mining applications, such as bioinformatics, computer vision, and social network analysis. Learning from multiview data has attracted extensive interests from research community. Long et al. [14] developed a mapping function that makes different feature spaces comparable and proposes a general framework for unsupervised learning from data with multiple views. Muslea et al. [15] presented an active learning framework from multiview data. Xia et al. [26] considered spectral embedding from multiple views. Xie et al. [27] studied multiview stochastic neighbor embedding that integrates heterogeneous features into a unified representation.

Our study in this paper falls into the category of unsupervised learning from multiple views. The general framework developed in [14] cannot be directly applied to the problem of mutual subspace clustering studied in this paper, since a clustering in the joint space of two feature spaces is critically different from mutual subspace clustering as explained in Example 1.

3. A density-based method

As indicated by the previous studies (e.g., [9,10]), a cluster can be modeled as a dense region in space. In this section, we present a density-based model for mutual subspace clusters, and develop a bottom-up method to find such clusters.

3.1. A density-based model

To keep our discussion simple, we assume that each attribute has the same range. As it will become clear soon, our method can be straightforwardly extended to the general case by setting one locality threshold for each attribute. By default we consider a set of points O. For a point $x \in O$ and an attribute D, let $x.D$ be the projection of x on D.

First of all, which distance/similarity measure should be used? The extensively used Euclidian distance [9,10] or any l-norm distance $l < \infty$ may not be appropriate since it biases towards low-dimensional subspaces. Suppose two points x and y are of the same distance d in each dimension, the Euclidian distance between x and y in an l-d subspace is $\sqrt{l} \cdot d$, which becomes larger and larger as the dimensionality increases. It is unfair to fix a distance/density threshold for subspaces of different dimensionality to identify dense regions.

Interestingly, the Chebyshev distance can avoid this problem. The Chebyshev distance between two points x and y in space $U = (D_1, \ldots, D_n)$ is defined as $dist_U(x,y) = \max_{i=1}^{n}\{|x.D_i - y.D_i|\}$. It is well known that $dist_U(x,y) = \lim_{k\to\infty}(\sum_{i=1}^{n}|x.D_i - y.D_i|^k)^{(1/k)}$. From the definition, it is easy to see that the Chebyshev distance does not bias in favor of any subspaces, and thus is suitable for the subspace cluster mining. However, our model also generally works for other unbias distance measures.

We measure the local density of a point using a user-defined *locality threshold* $\gamma > 0$. For a point x and a subspace $U = (D_{i_1}, \ldots, D_{i_l})$, we define the *$\gamma$-neighborhood* of x in U as $N_\gamma(x,U) = \{y \in O | dist_U(x,y) \le \gamma\}$. When γ is clear from context, we also write $N(x,U)$.

Clearly, our model can be easily extended to the general case where different dimensions have different ranges by simply setting one locality threshold for each attribute.

Let $\alpha > 0$ be a user-specified *density threshold*. A point x is called *dense* in subspace U if $|N_\gamma(x,U)| \ge \alpha$. Two points x and y are called *α-connected* in subspace U if they are connected by a series of dense points, i.e., there exist dense points $z_1, \ldots, z_q$ in U such that $x \in N(z_1,U)$, $z_i \in N(z_{i+1},U)$ $(1 \le i < q)$, and $y \in N(z_q,U)$.

A subset $C \subseteq O$ is a *density-based subspace cluster* with respect to locality threshold γ and density threshold α, if (1) for any $x,y \in C$, x and y are α-connected, and (2) there does not exist any points $z \notin C$ and $p \in C$ such that z and p are α-connected. In other words, a cluster is a maximal dense region that cannot be extended anymore. We also write the cluster as (C,U).

The above density-based model of subspace clusters is an extension of the density-based full-space clusters in [9,10]. Now, let us consider how mutual subspace clusters are formed.

Ideally, if (C,U,V) is a mutual subspace cluster in clustering spaces S_1 and S_2, then (C,U) and (C,V) should be subspace clusters in S_1 and S_2, respectively. However, the ideal case may not always happen in practice. Instead, we have to tolerate some noise in the cluster.

Consider subspace clusters (C_1,U) in S_1 (i.e., $U \subseteq S_1$) and (C_2,V) in S_2 (i.e., $V \subseteq S_2$). If C_1 and C_2 are almost identical, then we can regard $(C_1 \cap C_2,U,V)$ as a mutual subspace cluster. Technically, we can define the *purity* of $C_1 \cap C_2$ as $\rho = |C_1 \cap C_2|/|C_1 \cup C_2|$. A user can set a *purity threshold* ϱ to control the purity of mutual clusters.

Now, we are ready to define the problem of density-based mutual subspace clustering.

The density-based model: Given a set of points O in clustering spaces S_1 and S_2 $(S_1 \cap S_2 = \emptyset)$, a locality threshold γ, a density threshold α, a purity threshold ϱ, and a population threshold ξ, we want to find the mutual subspace clusters (C,U,V) such that there exist subspace clusters (C_1,U) and (C_2,V) such that $U \subseteq S_1$, $V \subseteq S_2$, $C = C_1 \cap C_2$, $\rho(C) \ge \varrho$, and $|C| \ge \xi$. The population threshold is to filter out insignificant clusters. □

3.2. Finding one-dimensional clusters

Density-based clusters have the following properties.

Lemma 1 (*Monotonicity*). *If C is a density-based cluster in subspace* $U=(D_{i_1},\ldots,D_{i_l})$, *then for each attribute* D_{i_j} $(1\le i\le l)$, *there exists a subspace cluster* (C',D_{i_j}) *such that* $C'\supseteq C$. □

An essential step to find density-based subspace clusters is to find 1-*d* clusters. The following conditional transitivity property of α-connectivity helps the search.

Lemma 2 (*Conditional transitivity*). *On attribute D, suppose x and y are α-connected, andy and z are also α-connected. Then, x and z are α-connected provided that y is dense.* □

We can find all 1-*d* subspace clusters in an attribute *D* as elaborated in the following example.

Example 2 (*Finding 1-d clusters*). Consider the set of points in Fig. 2. Let the locality threshold $\gamma=10$ and the density threshold $\alpha=3$. Let us find the 1-*d* clusters on attribute *Y*.

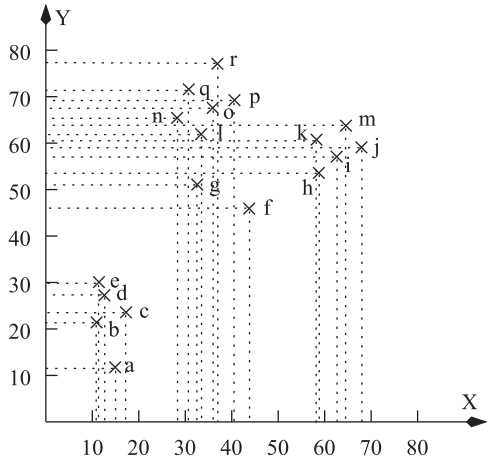

Fig. 2. A set of points as an example of finding clusters.

First, we sort the projections of all the points in *Y* in the value ascending order. The sorted list is *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, *i*, *j*, *k*, *l*, *m*, *n*, *o*, *p*, *q*, *r*.

We scan the sorted list. We find the first dense point *b*, and assign *b* to the first cluster C_1. We also include the points preceding *b* of distance up to $\gamma=10$, i.e., *a* in this example, into C_1. We set the state of C_1 to open meaning later points can be added into the cluster.

The next point, *c*, is dense and 3-connected to *b*, we add it to C_1 as well. Similarly, we add points *d* and *e* into C_1. Now, we consider point *f*, which is dense but not 3-connected to *e*, the point just added to C_1. This indicates that *f* as well as any points following *f* in the sorted list do not belong to C_1. Thus, we set the state of C_1 to closed meaning it cannot be extended further, and initiate a new cluster C_2 for *f*.

Points *g*,*h*,*i*,*j*,*k*,*l*,*m*,*n*,*o*,*p*,*q* are dense and 3-connected to the ones preceding them. They are added to C_2 in sequence. Point *r* is not dense, but is in $N(q,Y)$, and thus is also added to C_2. The state of C_2 is set to closed. Since there are no more points, the search terminates.

After all, we find two clusters on *Y*: $C_1=\{a,b,c,d,e\}$ and $C_2=\{f,g,h,i,j,k,l,m,n,o,p,q\}$. We can apply the above procedure to find 1-*d* clusters on attribute *X*. Three clusters can be found: $C_3=\{b,e,d,a,c\}$, $C_4=\{n,q,g,l,o,r,p,f\}$, and $C_5=\{k,h,i,m,j\}$. □

The algorithm is summarized in Fig. 3. The correctness of the algorithm follows with the density-based model. Limited by space, we omit the formal proof.

The complexity of finding all 1-*d* clusters in all attributes is $O((|S_1|+|S_2|)n\log n)$ where *n* is the total number of points in the data set and $|S_1|$ and $|S_2|$ are the dimensionality of the two clustering spaces. Due to the population threshold, we can prune the clusters that contain less than ξ points.

3.3. Finding all subspace clusters

A straightforward, generate-and-test method for mining mutual subspace clusters is to firstly find all subspace clusters

Input: a set of points O, an attribute D;
Output: the 1-d subspace clusters on D;
Method:

```
1:  sort all points in the ascending order of their projections on D,
    let x_1, ..., x_n be the sorted list;
2:  let j = 0; // C_j is the cluster under construction
3:  FOR i = 1 TO n DO
4:    IF x_i is dense THEN
5:      IF C_j = ∅ THEN
6:        add x_i to C_j; l = j − 1;
7:        WHILE l > 0 AND x_l ∈ N(x_i, D) DO
8:          add x_l to C_j; l = l − 1;
          END-WHILE
9:      ELSE IF i > 1 AND x_i ∈ N(x_{i−1}, D) AND x_{i−1} is also dense THEN add x_i to C_j;
10:     ELSE
11:       output (C_j, D); j = j + 1; add x_i to C_j; l = j − 1;
12:       WHILE l > 0 AND x_l ∈ N(x_i, D) DO
13:         add x_l to C_j; l = l − 1;
          END-WHILE
14:   ELSE // i.e., x_i is not dense
15:     let y be the last dense point in C_j;
16:     IF x_i ∈ N(y, D) THEN add x_i to C_j;
    END-FOR
17: IF C_j has not been output THEN output C_j;
```

Fig. 3. Finding 1-*d* subspace clusters.

in S_1 and S_2, respectively. Then, for each pair of clusters (C_1,U) and (C_2,V) where $U \subseteq S_1$ and $V \subseteq S_2$, we check the purity of $C_1 \cap C_2$ to identify the mutual clusters. The second step is straightforward. In this section, we discuss how to find all subspace clusters in one clustering space.

First, let us consider how to find 2-*d* clusters. According to Lemma 1, only the points in an intersection of two 1-*d* clusters on two attributes may form a 2-*d* cluster.

Example 3 (*Finding 2-d clusters*). As shown in Example 2, C_2 and C_4 are 1-*d* clusters on Y and X, respectively. We check whether some points in $C_2 \cap C_4$ form a 2-*d* cluster. Let $C = C_2 \cap C_4 = \{f,g,l,n,o,p,q,r\}$. We can start with an arbitrary point in the intersection. Suppose o is picked in this example. We first check whether o is dense. With the sorted lists on X and Y, this can be done easily. We only need to check whether $|N(o,X) \cap N(o,Y)| \geq \alpha$. Since $N(o,X) \cap N(o,Y) = \{p,q,l\}$, o is dense.

We assign o to a new cluster C_6 with signature subspace (X,Y). All points in $N(o,X) \cap N(o,Y)$ are also assigned to the cluster. Then, we check those points in $N(o,X) \cap N(o,Y)$ one by one. For each point, if it is dense, then the common points in its neighborhoods on X and Y are added to the cluster. For example, when point q is checked, it is dense. Moreover, n is in $N(q,X) \cap N(q,Y)$ and thus should be added to cluster C_6. This procedure continues until no more points can be added. Then, the cluster is output.

It can be verified that $C_6 = \{l,n,o,p,q,r\}$. Points f and g are not connected to o, and thus are not added into C_6. After C_6 is output, we pick one arbitrary point, say f, from the remainder, and conduct the search procedure as described above. f is not dense and is not in the neighborhood of any dense point. Thus, it is not in any 2-*d* cluster. Similarly, we can determine that g does not belong to any 2-*d* cluster. □

One major operation in finding clusters in a subspace is to compute the neighborhood and the density of a point. The sorted lists of points in single dimensions help due to the following result.

Lemma 3. *For a point x in a subspace $U = (D_{i_1}, \ldots, D_{i_l})$, $N(x,U) = \bigcap_{j=1}^{l} N(x,D_{i_j})$.* □

The Chebyshev distance defines the distance between two points in subspace U as the maximum distance between them in each dimension of U. Therefore, if a point is a neighbor of x in U, it must be x's neighbor in each dimension $D_{i_j} \in U$.

Generally, let (C_1,U) be an l-d subspace cluster and (C_2,D) be a 1-*d* cluster such that $D \notin U$ and $|C_1 \cap C_2| \geq \xi$. Then, we need to check whether some points in $C_1 \cap C_2$ can form a cluster in subspace $U \cup \{D\}$. The method is a straightforward extension of the method illustrated in Example 3. We start with an arbitrary point x in $C_1 \cap C_2$. If x is dense, then a new cluster is initialized. x and the points in $N(x,U \cup \{D\})$ are added to the new cluster. We extend the new cluster by including the neighborhoods of the dense points in the cluster, until the cluster cannot be enlarged anymore. Then, the cluster is output and those dense points in the cluster are removed. The procedure repeats until no new clusters can be formed.

When searching a cluster in one subspace, the above procedure resembles the framework in [9,10]. However, a critical difference is that we use Lemma 3 to compute the neighborhoods in high dimensional subspaces efficiently.

A multidimensional subspace cluster may be generated multiple times if the subspaces are not searched in a systematic way. For example, A cluster $(C,(D_1,D_2,D_3))$ may be generated by combining a 2-*d* cluster in subspace (D_1,D_2) and a 1-*d* cluster in D_3, or combining a 2-*d* cluster is subspace (D_2,D_3) and a 1-*d* cluster in D_1.

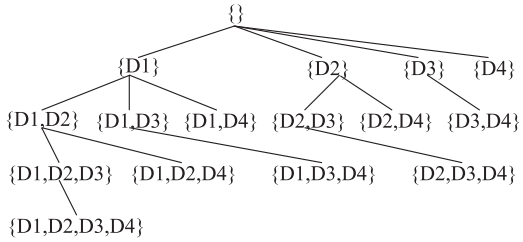

Fig. 4. A subspace enumeration tree of four attributes.

To avoid the redundancy, we can enumerate clusters by their signature subspaces using a set enumeration tree [22].

Fig. 4 shows a set enumeration tree of four attributes. Each node in the tree is a subspace. We conduct a depth-first search of the tree and find the subspace clusters in each subspace in the tree nodes. Clearly, a subspace cluster belongs to one and only one tree node which is the signature subspace.

In the search, apparently we can use the population ξ threshold to prune. If a cluster has less than ξ points, any subset of it cannot form a significant cluster and thus can be pruned.

3.4. Mutual pruning

In practice, not every density-based subspace cluster is mutual. In fact, many of them may not be mutual. Therefore, a major drawback of the straightforward, generate-and-test approach is that we have to find all subspace clusters in both clustering spaces before we can find the mutual ones. To overcome this, we need to let the clustering procedures in both clustering spaces interact to each other so that only those subspace clusters that may lead to mutual ones are computed. In this section, we present such an approach.

A mutual cluster must satisfy both the population requirement and the purity requirement. The central idea of our method is to use these two requirements to prune the search of subspace clusters in each clustering space.

The population requirement indicates that, for a mutual cluster (C,U,V) $(U \subseteq S_1, V \subseteq S_2)$, there are two subspace clusters (C_1,U) and (C_2,V) such that $|C| = |C_1 \cap C_2| \geq \xi$. For a subspace cluster (C_1,U) $(U \in S_1)$, if there does not exist a subspace cluster (C_2,V) $(V \in S_2)$ such that $|C_1 \cap C_2| \geq \xi$, then any subset of C_1 cannot lead to a mutual cluster. C_1 can be pruned.

To implement the pruning using the population requirement, we do not need to check (C_1,U) against all subspace clusters in S_2. Instead, we only need to check (C_1,U) against the 1-*d* subspace clusters in S_2 due to the following rule.

Lemma 4. *For a subspace cluster (C_1,U) $(U \subseteq S_1)$, if there does not exist a 1-d subspace cluster (C_2,D) such that $D \in S_2$ and $|C_1 \cap C_2| \geq \xi$, then any subset of C_1 cannot form a mutual cluster.*

Proof. We prove the lemma by contradiction. Let us assume that a subset $C_1' \subseteq C_1$ forms a mutual cluster, then there must be a subspace cluster (C_2',V) such that $V \subset S_2$ and $C_1' \subseteq C_2'$. According to Lemma 1, there exists a superset $C_2 \supseteq C_2'$ such that (C_2,D) is a 1-*d* cluster for $D \in V \subseteq S_2$. A contradiction. □

To use Lemma 4 in pruning, we first compute all 1-*d* clusters in S_1 and S_2, respectively. Then, for each cluster (C_1,U) in S_1, we collect the set $OC(C_1)$ of 1-*d* clusters in S_2 that have an intersection with C_1 of at least ξ points. If the set is empty, then C_1 should be pruned. Moreover, when combining (C_1,U) with another 1-*d* cluster (C_2,D) in S_1 to generate clusters in subspace $U \cup \{D\}$, $C_1 \cap C_2$ should be compared with the 1-*d* subspace clusters in $OC(C_1) \cap OC(C_2)$ only. The correctness can be proved easily. Simultaneously, we also use the 1-*d* subspace clusters in S_1 to prune the subspace clusters in S_2.

Now, let us consider how to use the purity requirement to help the pruning.

Lemma 5 (*Purity upper bound*). *Consider subspace clusters* (C_1,U_1) *and* (C_2,U_2) *such that* $U_1 \subset U_2 \subseteq S_1$ *and* $C_2 \subseteq C_1$. *Let* (C_3,D) $(D \in S_2)$ *be the 1-d subspace cluster in* S_2 *that maximizes* $|C_1 \cap C_3|$. *For any mutual cluster* (C,U',V') *such that* $C_2 \subseteq C \subseteq C_1$ *and* $U_1 \subseteq U' \subseteq U_2$, *we have* $\rho(C) \leq |C_1 \cap C_3|/|C_2|$. *Generally, let* (C_3,V) *be a subspace cluster in* S_2. *For any mutual cluster* (C,U',V') *such that* $C_2 \subseteq C \subseteq C_1$, $U_1 \subseteq U' \subseteq U_2$, *and* $V' \supset V$, *we have* $\rho(C) \leq |C_1 \cap C_3|/|C_2|$

Proof sketch. The lemma follows with Lemma 1. The upper bound of the purity is obtained when all the points in $C_1 \cap C_3$ are retained in the high-dimensional subspace clusters in both S_1 and S_2. □

Recall that we conduct a depth-first search of the subspace enumeration tree. Thus, in some situations, the clusters in subspace $U' \supset U$ may be already found. For example, in Fig. 4, clusters in subspace (D_1,D_2,D_3,D_4) are searched before those in subspace D_2. Therefore, we can use Lemma 5 to prune the clusters in some subspaces if the upper bound of their purity cannot pass the purity threshold ϱ.

The above pruning rules work well when many subspace clusters are not mutual. In such a situation, the above pruning rules can quickly prune the search of those non-mutual clusters using the population and purity requirements.

In addition to the pruning using the mutual subspace requirement, we can also reduce the redundancy in the clustering results. Consider two mutual clusters (C,U,V) and (C,U',V') such that $U' \subseteq U \subseteq S_1$ and $V' \subseteq V \subseteq S_2$. Let (C_1,U), (C'_1,U'), (C_2,V) and (C'_2,V') be the corresponding subspace clusters in S_1 and S_2, respectively. That is, $C = C_1 \cap C_2 = C'_1 \cap C'_2$. According to Lemma 1, $C'_1 \supseteq C_1$ and $C'_2 \supseteq C_2$. That is, $\rho(C,U,V) \geq \rho(C,U',V')$. Therefore, cluster (C,U,V) is purer. That means the U and V agree better than U' and V' on C as a mutual cluster. (C,U',V') is redundant given (C,U,V).

To reduce the redundant clusters, we have the following.

Lemma 6. *Let* (C,U,V) *be a mutual cluster. For a subspace cluster* (C_1,U') $(U' \subset U \subseteq S_1)$, *if for every 1-d cluster* (C_2,D) $(D \in V)$, $C_1 \cap C_2 = C$, *then C is the only subset in* C_1 *that can form a mutual cluster in subspaces* U'' *such that* $U' \subset U'' \subset U$, *and those clusters cannot have a purity higher than* (C,U,V). *Moreover, if* (C_1,U') *does not have an intersection of at least* ξ *points with any 1-d cluster on dimensions* $D \notin V$, *then any subset of* C_1 *cannot lead to any non-redundant mutual subspace cluster given* (C,U,V). □

To apply the rule, for each subspace cluster (C_1,U') in S_1, we maintain the set of 1-*d* clusters in S_1 that have an intersection with C of at least ξ points. Then, once the condition in Lemma 6 becomes true, the cluster can be pruned.

The above two pruning techniques and the redundancy removal technique can be integrated to the procedure of searching all subspace clusters described in Section 3.3. That is, we interleave the search of subspace clusters in the two clustering spaces. We start with computing all 1-*d* clusters in both clustering spaces. Mutual subspace clusters can be searched systematically using subspace enumeration trees. For each pair of 1-*d* clusters (C_1,A_1) and (C_2,B_2) where $|C_1 \cap C_2| \geq \xi$, $A_1 \in S_1$ and $A_2 \in S_2$, we explore the subspace clusters of subsets of C_1 in super-spaces of A_1 in S_1 using the subspace enumeration tree in S_1. Symmetrically, we explore the subspace clusters of subsets of C_2 in super-spaces of B_1 in S_2 using the subspace enumeration tree in S_2. If some nodes of paths of a tree are computed before, they are just reused. The pruning rules are applied to a clustering space using the information from the other clustering space.

4. A partitioning method

In this section, we present a partitioning, top-down method for mutual subspace clustering. The method is appropriate for the situations where major mutual subspace clusters exist. That is, most points in the data set belong to one mutual subspace cluster. We first analyze the goal of partitioning mutual subspace clustering. Then, we present the technical approach.

4.1. A partitioning model

Arguably, the partitioning model of clustering is the most classical and the most well accepted. Given a set of points O in space S and a parameter k of the number of clusters desired, the k-means clustering model is to find k centers $c_1,\ldots,c_k$ in S and partition O into k exclusive subsets $C_1,\ldots,C_k$ accordingly such that a point $o \in O$ is in cluster C_i if $dist(o,c_i) = \min_{1 \leq j \leq k}\{dist(o,c_j)\}$ and the partitioning minimizes $\sum_{i=1}^{k}\sum_{o \in C_i} dist(o,c_i)$, where $dist(x,y)$ is the distance between two points x and y. The intuition is to partition the points into k clusters such that the similarity within clusters is maximized.

Our partitioning mutual clustering model inherits the theme of partitioning clustering model. However, we need to address the concerns on multiple clustering spaces and signature subspaces of clusters.

Consider a set of points O in clustering spaces S_1 and S_2 where $S_1 \cap S_2 = \emptyset$. Our goal is to find k centers (c_1,U_1,V_1), …, (c_k,U_k,V_k), where $U_1,\ldots,U_k \subseteq S_1$ and $V_1,\ldots,V_k \subseteq S_2$ are the signature subspaces of the clusters in S_1 and S_2, respectively. The points in O are divided into exclusive subsets $C_1,\ldots,C_k$ accordingly. Ideally, a point $o \in O$ is assigned to cluster C_i if $dist_{U_i}(o,c_i) = \min_{1 \leq j \leq k}\{dist_{U_j}(o,c_j)\}$ and $dist_{V_i}(o,c_i) = \min_{1 \leq j \leq k}\{dist_{V_j}(o,c_j)\}$, where $dist_U(x,y)$ is the distance between two points x and y in space U. In the best case, the partitioning minimizes $\sum_{i=1}^{k}\sum_{o \in C_i} dist_{U_i}(o,c_i)$ and $\sum_{i=1}^{k}\sum_{o \in C_i} dist_{V_i}(o,c_i)$ simultaneously.

Generally, the ideal case may not happen on a large real data set. Due to the structural differences of the two clustering spaces, partitioning minimizing both distance sums may not exist at all on some data sets. As an extreme example, consider the four points in 2-*d* space: $a(0,10)$, $b(10,10)$, $c(0,0)$ and $d(10,0)$. If $k=2$, then $\{a,b\}$ and $\{c,d\}$ are the optimal partitioning in clustering space Y; and $\{a,c\}$ and $\{b,d\}$ are the optimal in clustering space X. It is impossible to partition the four points into two clusters minimizing the distance sums in both clustering spaces simultaneously. This counter example shows that the optimization goal may not always be feasible for mutual subspace clustering.

Moreover, even if the optimal partitioning exists, finding the exact optimal partitioning is computationally prohibitive on large data sets. The partitioning mutual subspace clustering optimization model can be viewed as a generalization of the k-means problem, which is known NP-hard.

Based on the above analysis, in the rest of this section, we will explore a heuristic method for the partitioning mutual subspace clustering.

4.2. Finding signature subspaces

A critical issue is to find the signature subspace for a cluster. Given a set of points C that are assigned to a cluster in space S, we want to find a subspace $U \subseteq S$ that manifests the similarity between points in C.

Again, let us assume that the attributes are normalized. Ideally, if we have a proper similarity measure $sim_U(x,y)$ which measures the similarity between points x and y in subspace U, then we can calculate $\sum_{x,y \in C} sim_U(x,y)$ straightforwardly for every non-empty

subspace $U \subseteq S$, and pick the subspace maximizing the sum of similarities as the signature subspace.

However, such a method is often impractical for two reasons. First, finding a proper similarity measure is often difficult. Many similarity or distance measures bias towards low-dimensional subspaces dramatically. Similarities in different subspaces are often incomparable directly. Second, if S has m dimensions, then we need to check 2^m-1 subspaces. When the dimensionality is high, enumerating all subspaces and calculating the sums of similarities in them are often very costly.

If U is the signature subspace of C which manifests the similarity among points in C, then the points must be similar in every attribute in U. Moreover, the points must be substantially dissimilar in attributes not in U. Technically, the *average pairwise distance* (*APD* for short), i.e., the average distance between two points in C, can be used to measure the compactness of the cluster. That is

$$APD(C,D) = \frac{\sum_{x,y \in C} dist_D(x,y)}{\frac{|C| \cdot (|C|-1)}{2}} = \frac{2\sum_{x,y \in C} dist_D(x,y)}{|C| \cdot (|C|-1)}.$$

Since the attributes are normalized, the average pairwise distance can be used as the measure of how well the points in C are clustered on an attribute. The attributes in the signature subspace of C should have a small average pairwise distance, while the attributes not in the signature subspace should have a large average pairwise distance.

We can sort all attributes in the average pairwise distance ascending order. The first attribute is the best to manifest the similarity. Now, the problem is how to select other attributes that manifest the similarity of the cluster together with the first one.

Hypothetically, the attributes are in two sets: the ones in the signature subspace and the ones not in the subspace. The ones in the signature subspace should share a similar average pairwise distance. We can apply the Chebyshev's inequality to confidently select the attributes in the signature subspace.

Let $D_1, \ldots, D_n$ be the attributes in the average pairwise distance increasing order. Suppose $D_1, \ldots, D_i$ $(1 \le i \le n)$ form the signature subspace. The expectation of average pairwise distance is $E(APD) = (1/i)\sum_{j=1}^{i} APD(C,D_j)$. Let σ be the standard deviation of $APD(C,D_1)$, …, $APD(C,D_i)$. Then, we require that for any attribute D_j $(1 \le j \le i)$, $|APD(C,D_j) - E(APD)| \le t \cdot \sigma$, where t is a small integer (typically between 3 to 5). $(1-(1/t^2))$ is the *confidence level* of the selection.

Algorithmically, we initialize the signature subspace U with D_1, the first attribute in the sorted list. Then, we add the attributes one by one in the average pairwise distance ascending order. For each attribute added, we check whether the confidence level is maintained. Once the confidence level is violated, then the attribute just added should be removed, and the selection procedure terminates. The pseudo-code of the procedure is given in Fig. 5.

4.3. The top-down algorithm

Almost all top-down partitioning clustering methods such as k-means for full space clustering and PROCLUS [1] for subspace clustering adopt an iterative greedy search. The central idea of our top-down mutual subspace clustering method is to interleave the iterative k-means clustering procedures in the clustering spaces.

We start with arbitrary k points $c_1, \ldots, c_k$ in the clustering space S_1 as the temporary centers of clusters $C_1, \ldots, C_k$, respectively. The k centers do not necessarily belong to O. We assign the points in O to the clusters according to their distances to the centers in space S_1: a point $o \in O$ is assigned to the cluster of the center closest to o. This is the first step of the classical k-means clustering procedure.

In order to find mutual subspace clusters, we use the information in the clustering space S_2 to refine the clusters. We need to find the signature subspaces in S_2 and also improve the cluster assignment.

For each cluster C_i, we find a subspace $V_i \subseteq S_2$ as the signature subspace of C_i in S_2 using the method discussed in Section 4.2, and calculate the center of C_i in V_i.

To improve the cluster assignment, for each point $o \in O$, we check $dist_{V_i}(o,c_i)$ for $1 \le i \le k$, and assign o to the cluster of the closest center in the signature subspace. This generates the refined clustering.

The clustering in S_2 is fed into the refinement using the information in S_1. That is, we compute the signature subspaces and the centers of the clusters in S_1, and adjust the cluster assignment. As the iteration continues, we use the information in the clustering spaces to form mutual subspace clusters.

A cluster can become stable if the signature subspaces in the clustering spaces agree with each other on the cluster assignment. That is, in the clustering spaces the centers attract the approximately same set of points to the cluster. On the other hand, for a temperate cluster, if the signature subspaces do not agree with each other, then the centers change substantially due to the substantial change of the cluster members.

When should the iteration terminate? If mutual subspace clusters exist, then the above iterative refinement can greedily approach the mutual subspace clusters. This is because the refinement in S_1 and S_2 iteratively reduces variance of clusters in S_1 and S_2. The exact termination (no points changing clusters) may require a large number of iterations. In practice, we define the *mis-assignment rate* as the portion of points in O that are assigned to a different cluster in each iteration. The clustering is stable if the signature subspaces of the clusters become stable and the mis-assignment rate is low. The iterative refinement stops if the signature subspaces in the clustering spaces do not change,

Input: a set of points O, a cluster $C \subseteq O$, and t;
Output: the signature subspace of C;
Method:
1: calculate the average pairwise distance for each attribute D;
2: sort all attributes in the average pairwise distance ascending order, let $D_1, \ldots, D_n$ be the sorted list;
3: let $U = \{D_1\}$;
4: FOR $i = 2$ to n DO
5: let $U = U \cup \{D_i\}$;
6: calculate $E(APD)$ and the standard deviation σ for $APD(D_1), \ldots, APD(D_i)$;
7: IF $E(APD) - APD(C, D_1) > t\sigma$ OR $APD(C, D_i) - E(APD) > t\sigma$ THEN RETURN$(U - \{D_i\})$;
END-FOR
8: RETURN U;

Fig. 5. Computing the signature subspace.

Input: a set of points O in clustering spaces S_1 and S_2, the number of clusters k, and parameters θ;
Output: a set of k mutual subspace clusters;
Method:
1: select arbitrary k centers $c_1, \ldots, c_k$ in S_1;
2: assign each point in O to a cluster of the closest center;
3: DO
4: FOR EACH cluster C_i DO
5: find the signature subspace in S_2 and the center;
END-DO
6: assign each point in O to a cluster of the closest center in the signature subspace in S_2;
7: FOR EACH cluster C_i DO
8: find the signature subspace in S_1 and the center;
END-DO
9: assign each point in O to a cluster of the closest center in the signature subspace in S_1;
10: IF cluster centers are stable THEN remove conflict points;
11: UNTIL the clustering is stable;
12: FOR EACH cluster C_i DO
13: output (C_i, U_i, V_i) where C_i is the set of points in C_i, U_i and V_i are the signature subspaces in S_1 and S_2, respectively
END-FOR

Fig. 6. The framework of the top-down algorithm.

Table 1
The mutual subspace clusters found in the housing data set.

Cluster id	Size	Purity (%)	Space 1: house information				Space 2: environment information			
			RM 6.28	Age 68.6	MEDV 22.5	Signature subspace	CRIM 3.6	DIS 3.8	LSTAT 12.7	Signature space
1	139	56	6.1	95.2	18.1	(RM, age)	7.8	2.09	17.1	(CRIM, DIS, LSTAT)
2	290	67	6.1	72.1	18.5	(RM, MEDV)	5.1	2.98	14.6	(CRIM, LSTAT)

and the mis-assignment rate is lower than θ% in two consecutive rounds of refinement, where θ is a user-specified parameter, and in each round both the signature subspaces in both clustering spaces S_1 and S_2 are refined.

On the other hand, as elaborated before, some points may not belong to any mutual clusters. Then, the iterative refinement may fall into an infinite loop, since the two clustering spaces do not agree with each other on those points. To detect the potential infinite loop, we compare the cluster assignments in the two clustering spaces in two consecutive rounds. For each mis-assigned point which is assigned to different clusters in different clustering spaces, if it is repeatedly assigned to the same cluster in the same clustering space, and the cluster centers are stable, then the point does not belong to a mutual cluster and should be removed. Such a point is called a *conflict point*. After such conflict points are removed, the centers and the cluster assignment become stable. Then, the mutual subspace clusters can be derived.

The top-down algorithm is summarized in Fig. 6.

5. Experimental results

In this section, we report a systematic empirical study using real data sets and synthetic data sets. All the experiments are conducted on a PC computer running the Microsoft Windows XP Professional Edition operating system, with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk. Our algorithms are implemented in Microsoft Visual C++ V6.0.

5.1. Results on real data sets

Two real data sets are used to illustrate the effectiveness of mutual subspace clustering on real applications.

The Boston Housing Database from the UCI Machine Learning Database Repository[2] contains 506 tuples about the house values in suburbs of Boston. There are 14 numerical attributes that can be partitioned into two groups. The first group contains the information about the housing regions, including *RM* (average number of rooms per dwelling), *Age* (proportion of owner-occupied units built prior to 1940) and *MEDV* (Median value of owner-occupied homes in $1000s). The other group describes the environment around the houses, such as the crime rate by town, the weighted distance to five Boston employment centers and so on. Thus, we have the two clustering spaces, *house information* and *environment information*. We want to find the clusters agreed by both clustering spaces.

We apply both the top-down and the bottom-up methods on the data set. The results are consistent. Two interesting clusters and their signature subspaces are listed in Table 1 as an example. For example, cluster 1 contains the older houses (on average 95% of the units are built before 1940) with fewer rooms in the *house information* clustering space. The signature subspace in the *environment* clustering space indicates that the criminal rate in cluster 1 is much higher than the average, while the distance to the employer centers is shorter and the percentage of the lower status population is much higher.

We also conduct the traditional subspace clustering on the data set and the clusters are not as explainable as the ones listed above. The detailed results are omitted due to the limit of space.

The second real data set is the NBA player career statistics.[3] We collect the profile of 316 NBA players and their career statistics. There are four numeric attributes about the profile: *Height*, *Weight*, *Age* and *Experience* (i.e. the number of years as a

[2] http://www.ics.uci.edu/~mlearn/MLRepository.html
[3] http://sports.yahoo.com/nba/players

player in the NBA league), while the *technical statistics* include *G* (number of games played), *Min.* (average number of minutes played per game), *3PT* (average number of 3-point throws per game), *FT* (average number of free throws made per game), *Rebound* (average number of rebounds per game), *Ast* (average number of assists per game), *TO* (average number of turn-overs per game), *Stl* (average number of steals per game), *Blk* (average numbers of blocks per game), *PF* (average number of personal fouls per game) and *PPG* (average points per game).

The *profile* and the *technical statistics* describe a player from two different angles. We want to find the groups of players having the similar performance in the game and also share similarity in their profiles. Thus, we use the *profile* and the *technical statistics* as two clustering spaces and apply the mutual subspace clustering.

We use the partitioning model introduced in Section 4 to compute the mutual subspace clusters. The parameter k controls the number of clusters we want. In the experiments, k varies from 2 to 5. We only list two clusters of $k=5$ in Table 2 due to the limit of space. We list the attributes in the signature subspaces of each cluster (below each attribute is the average value of that attribute in the whole data set). Clusters 1 and 2 are described using the average attribute value (mean) of the players in the cluster.

The average values of attributes *Height* and *Weight* in cluster 1 are smaller than the average values of the whole data set. Cluster 1 contains the relatively shorter players with less weight. Meanwhile, the signature space of cluster 1 in space *technical statistics* is (*rebound*, *block*) and the average values in those two attributes are lower than the average values in the whole data set. In other words, the players in cluster 1 performs relatively poor in *rebound* and *block*. The signature subspaces of cluster 1 in the two signature subspaces provide a good explanation of the cluster: it is easier for taller players to make *rebounds* and *blocks*. Similarly, we find that cluster 2 contains the relatively taller and heavier players, who perform worse than the average in *3 point throws* and *assists*. This also matches the knowledge that taller players often lead the interior-line attack and have less chances to make three point throws and assists.

The above results clearly show that mutual subspace clustering can detect meaningful mutual subspace clusters and their signature subspaces.

Can mutual subspace clusters be found by simply clustering in the union space? As a comparison, we combine the attributes from the *profile* space and the *technical statistics* space, and compute the traditional subspace clusters in the union space. Table 3 shows two best clusters. The signature subspaces of the clusters do not contain meaningful attributes from both clustering spaces. In other words, using the traditional subspace clustering may not find mutual clusters properly.

5.2. Results on synthetic data sets

We further test the effectiveness and the efficiency of the density based, bottom-up method and the partition based, top-down method, as well as the mutual pruning strategy.

Generally, a synthetic data set O contains the points with two set of attributes, S_1 and S_2. We generate two sets of clusters in S_1 and S_2, respectively. A pair of clusters $C \in S_1$ and $C' \in S_2$ are *fully mutual* if all the points in C are contained in C' and vice versa. Similarly, $C \in S_1$ and $C' \in S_2$ are *partially mutual* if most of the points in C also belongs to C' and vice versa.

By default, each data set has 10,000 points. We generate the synthetic data sets with different properties.

D_1: *All the clusters are fully mutual*: There is a one-to-one mapping between the clusters in S_1 and S_2 such that the two mapped clusters contain the same set of points. Though this may rarely happen in the real data sets, we can use this synthetic data set to test the best accuracy our methods can achieve.

D_2: *All the clusters are partially mutual*: In this data set, we only require the clusters to be *partially mutual*. That is, for each cluster C in S_1, more than half of the points (the proportion is specified by parameter ϱ) are contained in the same cluster in S_2 and vice versa. We want to test the robustness of our methods using this data set.

D_3: *Only a subset of the clusters are mutual*: In this data set, only a subset of the clusters are mutual. This better resembles the real case and we can test whether the mutual pruning strategy will take effect in this situation.

In order to generate a set of mutual subspace clusters in spaces S_1 and S_2, we take the following parameters as shown in Fig. 7.

Parameter	Explanation
d_1,d_2	The dimensionality of space S_1 and S_2
N_1,N_2	The number of subspace clusters in S_1, S_2
M	The number of mutual subspace clusters
ν	The percentage of outlier points
(μ_{U_1},σ_{U_1}) (μ_{U_2},σ_{U_2})	Mean and variance of the signature subspace dimensionality of the clusters in S_1 and S_2
(μ_C,σ_C)	Mean and variance of the cluster size
(μ_{V_1},σ_{V_1}) (μ_{V_2},σ_{V_2})	Mean and variance of the spread of the clusters in S_1 and S_2
ϱ	The proportion of the intersecting points in the mutual clusters

Fig. 7. The parameters of the synthetic data generator.

Table 2
The 2 mutual subspace clusters found in the NBA data set.

Cluster id	Size	Space 1: profile			Space 2: technical statistics				
		Height (cm) 200.8	Weight (lbs) 222.2	Signature subspace	3PT 0.5	Rebound 3.5	Ast 1.7	Blk 0.4	Signature subspace
1	111	198.2	214.3	(Height, weight)	0.65	2.55	2.32	0.2	(Rebound, Blk)
2	127	208.7	245.4	(Height, weight)	0.25	4.5	0.9	0.6	(3PT, Ast)

Table 3
The two subspace clusters found in the NBA data set.

Cluster id	Size	Height (cm) 200.8	Weight (lbs) 222.2	Min. 20	3PT 0.5	FT 1.5	Rebound 3.5	Ast 1.7	TO 1.2	Blk 0.4	PF 2	Signature subspace
1	80	200.7	222.4	31.3	0.7	2.9	5.5	3	2.1	0.66	2.7	(Min., TO, PF)
2	67	200.9	222	14.8	0.3	0.8	2.5	1	0.8	0.25	1.67	(FT, Blk)

We first define d_1 and d_2, the dimensionality of the two clustering spaces, and N_1 and N_2, the number of clusters. $M \leq \min(N_1,N_2)$ is the number of mutual subspace clusters in S_1 and S_2. For each pair of mutual clusters, ϱ specifies the proportion of points appearing in both clusters. The dimensionality of the signature subspaces in S_1 and S_2 follows the normal distribution. The mean and the variance is specified by $(\mu_{U_1}, \sigma_{U_1})$ and $(\mu_{U_2}, \sigma_{U_2})$, respectively. The attributes in the signature subspaces are randomly picked. In order to generate the points in a cluster, we first generate the center of the cluster by randomly picking a value in each attribute. The attribute values of each point in the signature subspace follow the normal distribution. The mean is the attribute values of the center and the variance is specified by $(\mu_{V_1}, \sigma_{V_1})$. The values in the rest attributes are randomly generated following the even distribution.

We test the bottom-up method and the top-down method on data set D_1, where all the 10 clusters are *fully mutual*. By default, the parameter k in the top-down method is set to 5, the purity threshold ϱ of the bottom-up method is set to 0.5 and the population threshold ξ is set to 1% of the data set size.

We first generate the synthetic data set with 10 dimensions in S_1 and S_2, respectively. There are 10 clusters (each contains 1000 points) in S_1 and S_2. The dimensionality of the signature subspaces of the clusters varies from 2 to 5. All the 10 clusters are *fully mutual*. We add a certain amount of noise points to the data set, ranging from 0 to 20% (i.e., from 0 to 2000 noise points). Figs. 8 and 9 show the accuracy of finding the clusters and the signature subspaces, respectively. The accuracy of a cluster is measured by $|C_{output} \cap C_{real}|/|C_{output} \cup C_{real}|$, where C_{output} is the cluster found by the algorithm and C_{real} is the real cluster matching C_{output} the best. The accuracy of the whole clustering is the average of accuracy of all clusters. Similarly, the accuracy of a signature subspace is measured by $|U_{output} \cap U_{real}|/|U_{output} \cup U_{real}|$, where U_{output} is the signature subspace found by the algorithm and U_{real} is the signature subspace of the corresponding real cluster. Again, the average of signature subspaces of all clusters is reported.

The accuracy of both methods decreases as the noise rate increases. The bottom-up method is always more accurate than the top-down method when the noise rate is 5% or higher. Moreover, the bottom-up method has a high accuracy in finding signature subspaces. This is because the bottom-up method is based on the density of points, and noise points with relatively low density do not affect the accuracy. On the other hand, the top-down method is based on the partitioning model, which can be easily affected by noise points. Those results show that the density-based model is more robust in finding clusters and signature subspaces against noise.

Then, we test the bottom-up method and the top-down method on the *partially mutual* data set D_2. There are 5 clusters with 2000 tuples in each cluster. For each cluster C in S_1 and its corresponding cluster C' in S_2, ϱ determines the proportion of points in $C \cap C'$. We vary ϱ from 60% to 100% and the accuracy of the clusters found are shown in Fig. 10.

The top-down method does not perform well when ϱ is low. The top-down method always tries to find the clusters agreed by both S_1 and S_2. If clusters in the two spaces are not consistent, the top-down method may encounter the conflicting situation described at the end of Section 4. For example, when $\varrho = 60\%$, the two spaces do not agree with each other in clustering and there are 53% of the conflicting points. The proportion of conflicting points is greater than $1-\varrho$ in this case. As ϱ increases, the accuracy of both methods improves. This results show that the density-based, bottom-up method is more robust to handle mutual clusters of various quality.

Last, we test the effectiveness of the mutual pruning in the bottom-up method. The algorithm *MUSC* is the bottom-up method enhanced by the mutual pruning introduced in Section 3.

Data set D_3 is used, where we vary the percentage of mutual clusters from 20% to 100%. Since the bottom-up method and the MUSC method adopt the same clustering parameter and output the same results, we only report the accuracy of the bottom-up method here.

Fig. 12 shows that the bottom-up method can correctly find the mutual clusters and their signature subspaces, no matter how many mutual clusters there are in the data set, but the top-down method

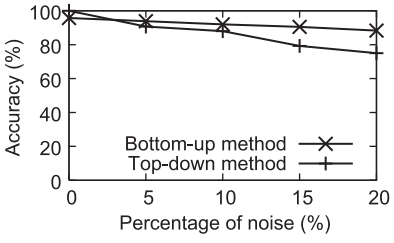

Fig. 8. Accuracy of finding clusters.

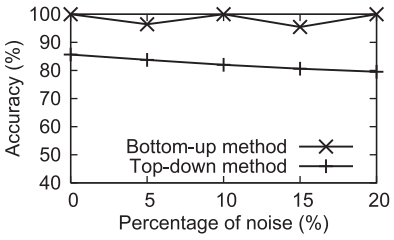

Fig. 9. Accuracy of finding signature subspaces.

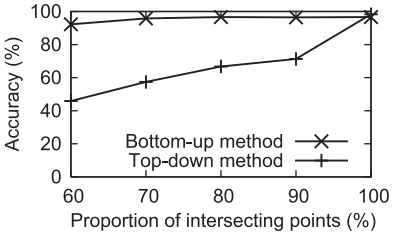

Fig. 10. Accuracy of finding the mutual clusters.

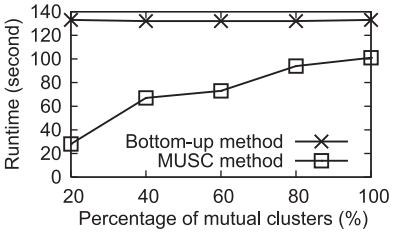

Fig. 11. The effectiveness of the mutual pruning.

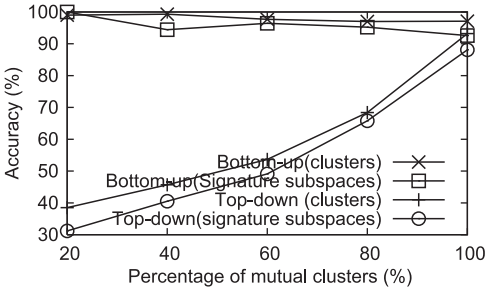

Fig. 12. Accuracy w.r.t. the proportion of intersections.

performs poorly since many clusters are not mutual. The runtime of the bottom-up and the MUSC method is shown in Fig. 11. The mutual pruning techniques improve the performance of MUSC substantially. When there are only 20% of the mutual clusters, MUSC runs nearly seven times faster than the bottom-up method. Most of the not mutual clusters can be pruned using the mutual pruning strategy. Even when all the clusters are mutual, MUSC still runs faster than the bottom-up method, since there are some clusters sharing the same intersections which can be detected by MUSC and pruned.

5.3. Scalability

We also evaluate the efficiency and the scalability of our methods with respect to database size and dimensionality.

We generate the synthetic data sets with the dimensionality varying from 10 to 30 and test the scalability of the top-down method, the bottom-up method and the MUSC method. There are 10,000 tuples in the data set and the dimensionality of the signatures subspaces is from 2 to 5. The results are shown in Fig. 13.

The runtime of the top-down method increases substantially as the number of attributes increases, because all the attributes are checked in each round to determine the signature subspaces. The runtime of the bottom-up method and the MUSC method does not increase greatly, since the clusters only exist in lower dimensional subspaces and thus most of the dimensions are pruned using the density threshold.

We also test the scalability of the methods with respect to the size of the data set. We set the dimensionality to 10, and increase the number of tuples from 20,000 to 100,000. The number of clusters is set to 10, five of which are *fully mutual* clusters. The results are shown in Fig. 14.

The runtime of the bottom-up and the MUSC method increases linearly as the cardinality of the data set goes up. But overall, MUSC runs faster than the bottom-up method, since it can prune the clusters with insufficient intersection size early. The runtime of the top-down method increases dramatically, since the average pairwise distance in each cluster in each attribute has to be calculated in order to determine the signature subspaces.

The above results show that the bottom-up method, the top-down method and the MUSC method are effective in finding the mutual subspace clusters and their signature subspaces in various situations. The top-down method works well when there are not many points and most of the clusters are mutual. The bottom-up method based on the density model is more robust compared to the top-down method, and works particularly well when many clusters are not mutual, and there are many noise points. Moreover, the mutual pruning strategy is effective in reducing the searching space and thus improving the efficiency and the scalability of the bottom-up method.

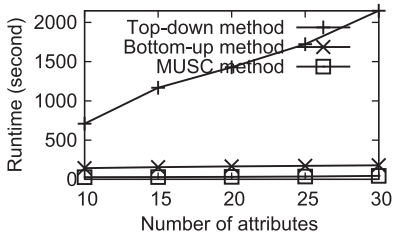

Fig. 13. Runtime vs. dimensionality.

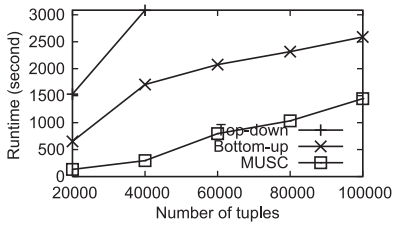

Fig. 14. Runtime vs. cardinality.

6. Conclusions

In this paper, we identified and studied a novel data mining problem: mining mutual subspace clusters from multiple sources. We developed two interesting models and the corresponding methods for mutual subspace clustering. We used both synthetic data sets and real data sets to examine the effectiveness and the efficiency of the methods.

In Section 4.1, we briefly mentioned one special case in which the optimal partition based clustering does not exist. As future work, we plan to systematically study the condition for optimal partition based clustering from multiple data sources. Specifically, we will try to answer the following questions: How to measure the similarity of two data sources in term of clustering consistency? How to derive the converge condition for partition based clustering method in multiple data sources? What is the quality bound of the greedy approximation algorithms?

Moreover, as jointly mining multiple data sources becomes an interesting and promising research direction, we will also investigate the applications of mutual subspace clustering in bioinformatics and health-informatics.

Acknowledgments

We sincerely thank the anonymous reviewers and the editors of the special issue for their constructive suggestions which help to improve the quality of the paper.

This research is supported in part by NSERC through an NSERC Discovery Grant and an NSERC Discovery Accelerator Supplements Grant, and by the GRAND NCE. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] C.C. Aggarwal, J.L. Wolf, P.S. Yu, C. Procopiuc, J.S. Park, Fast algorithms for projected clustering, in: Proceedings of 1999 ACM-SIGMOD International Conference on Management of Data (SIGMOD'99), Philadelphia, PA, June 1999, pp. 61–72.

[2] C.C. Aggarwal, P.S. Yu, Finding generalized projected clusters in high dimensional spaces, in: Proceedings of 2000 ACM-SIGMOD International Conference on Management of Data (SIGMOD'00), Dallas, TX, May 2000, pp. 70–81.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: Proceedings of 1998 ACM-SIGMOD International Conference on Management of Data (SIGMOD'98), Seattle, WA, June 1998, pp. 94–105.

[4] J.-W. Chang, D.-S. Jin, A new cell-based clustering method for large, high-dimensional data in data mining applications, in: SAC'02: Proceedings of the 2002 ACM symposium on Applied computing, ACM Press, New York, NY, USA, 2002, pp. 503–507.

[5] K. Chaudhuri, S.M. Kakade, K. Livescu, K. Sridharan, Multi-view clustering via canonical correlation analysis, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML'09, ACM, New York, NY, USA, 2009, pp. 129–136.

[6] C.H. Cheng, A.W.-C. Fu, Y. Zhang, Entropy-based subspace clustering for mining numerical data, in: Proceedings of 1999 International Conference on Knowledge Discovery and Data Mining (KDD'99), San Diego, CA, August 1999, pp. 84–93.

[7] S. Děroski, Multi-relational data mining: an introduction, SIGKDD Explor. Newsl. 5 (1) (2003) 1–16.

[8] M. Ester, R. Ge, B.J. Gao, Z. Hu, B. Ben-Moshe, Joint cluster analysis of attribute data and relationship data: the connected k-center problem, in: SDM, 2006.

[9] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases, in: Proceedings of 1996 International Conference on Knowledge Discovery and Data Mining (KDD'96), Portland, Oregon, August 1996, pp. 226–231.
[10] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, Density-connected sets and their application for trend detection in spatial databases, in: Proceedings of 1997 International Conference on Knowledge Discovery and Data Mining (KDD'97), Newport Beach, CA, August 1997, pp. 10–15.
[11] J.H. Friedman, J.J. Meulman, Clustering objects on subsets of attributes, J. R. Stat. Soc. Ser. B 66 (4) (2004) 815.
[12] C.-R. Lin, K.-H. Liu, M.-S. Chen, Dual clustering: integrating data clustering over optimization and constraint domains, IEEE Trans. Knowledge Data Eng. 17 (5) (2005) 628–637.
[13] B. Liu, Y. Xia, P.S. Yu, Clustering through decision tree construction, in: CIKM'00: Proceedings of the Ninth International Conference on Information and Knowledge Management, ACM Press, New York, NY, USA, 2000, pp. 20–29.
[14] B. Long, P.S. Yu, Z. Zhang, A general model for multiple view unsupervised learning, in: SDM, 2008, pp. 822–833.
[15] I. Muslea, S. Minton, C.A. Knoblock, Active learning with multiple views, J. Artif. Int. Res. 27 (October) (2006) 203–233.
[16] H. Nagesh, S. Goil, A. Choudhary, Mafia: efficient and scalable subspace clustering for very large data sets, Technical Report 9906-010, Northwestern University, June 1999.
[17] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in: Advances in Neural Information Processing Systems, MIT Press, 2001, pp. 849–856.
[18] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, SIGKDD Explor. Newsl. 6 (1) (2004) 90–105.
[19] J. Pei, D. Jiang, A. Zhang, On mining cross-graph quasi-cliques, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'05), Chicago, IL, USA, August 2005.
[20] J. Pei, X. Zhang, M. Cho, H. Wang, P.S. Yu, Maple: a fast algorithm for maximal pattern-based clustering, in: Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03), IEEE, Melbourne, FL, November 2003.
[21] C.M. Procopiuc, M. Jones, P.K. Agarwal, T.M. Murali, A Monte Carlo algorithm for fast projective clustering, in: SIGMOD'02: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, ACM Press, New York, NY, USA, 2002, pp. 418–427.
[22] R. Rymon, Search through systematic set enumeration, in: Proceedings of 1992 International Conference on Principle of Knowledge Representation and Reasoning (KR'92), Cambridge, MA, 1992, pp. 539–550.
[23] E. Segal, H. Wang, D. Koller, Discovering molecular pathways from protein interaction and gene expression data, Bioinformatics 19 (Suppl 1) (2003).
[24] H. Wang, W. Wang, J. Yang, P.S. Yu, Clustering by pattern similarity in large data sets, in: Proceedings of 2002 ACM-SIGMOD International Conference on Management of Data (SIGMOD'02), Madison, WI, June 2002.
[25] K.-G. Woo, J.-H. Lee, M.-H. Kim, Y.-J. Lee, Findit: a fast and intelligent subspace clustering algorithm using dimension voting, Inf. Software Technol. 46 (4) (2004) 255–271.
[26] T. Xia, D. Tao, T. Mei, Y. Zhang, Multiview spectral embedding, Trans. Syst. Man Cyber. Part B 40 (December) (2010) 1438–1446.
[27] B. Xie, Y. Mu, D. Tao, m-sne: Multiview stochastic neighbor embedding, in: ICONIP (1)'10, 2010, pp. 338–346.
[28] J. Yang, W. Wang, H. Wang, P.S. Yu, δ-cluster: capturing subspace correlation in a large data set, in: Proceedings of 2002 International Conference on Data Engineering (ICDE'02), San Fransisco, CA, April 2002.
[29] T. Zhou, D. Tao, Fast gradient clustering, in: NIPS 2009 Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity & Polyhedra (NIPS: DISCML), 2009, pp. 1–6.

Ming Hua is a Research Scientist at Facebook Inc., where she focuses on social network modeling and analysis. Ming has published in premier academic journals and conferences. She serves in the Program Committees of numerous International Conferences such as ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), IEEE International Conference on Data Mining (ICDM) and the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD). Ming holds a PhD degree in Computing Science from Simon Fraser University, Canada.

Jian Pei is a Professor at the School of Computing Science at Simon Fraser University, Canada. He is interested in developing effective and efficient data analysis techniques for novel data intensive applications, including various techniques of data mining, Web search, information retrieval, data warehousing, online analytical processing, and database systems, as well as their applications in social networks, health-informatics, business and bioinformatics. His research has been extensively supported in part by many government funding agencies and industry partners. He has published prolifically and served regularly for the leading academic journals and conferences in his fields. He is an associate editor of several premier journals in data mining and analytics. He received several prestigious awards.