Clustering by Pattern Similarity in Large Data Sets

Haixun Wang Wei Wang Jiong Yang Philip S. Yu

IBM T. J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
{haixun, ww1, jiyang, psyu}@us.ibm.com

ABSTRACT

Clustering is the process of grouping a set of objects into classes of *similar* objects. Although definitions of similarity vary from one clustering model to another, in most of these models the concept of similarity is based on distances, e.g., Euclidean distance or cosine distance. In other words, similar objects are required to have close values on at least a set of dimensions. In this paper, we explore a more general type of similarity. Under the pCluster model we proposed, two objects are similar if they exhibit a coherent pattern on a subset of dimensions. For instance, in DNA microarray analysis, the expression levels of two genes may rise and fall synchronously in response to a set of environmental stimuli. Although the magnitude of their expression levels may not be close, the patterns they exhibit can be very much alike. Discovery of such clusters of genes is essential in revealing significant connections in gene regulatory networks. E-commerce applications, such as collaborative filtering, can also benefit from the new model, which captures not only the closeness of values of certain leading indicators but also the closeness of (purchasing, browsing, etc.) patterns exhibited by the customers. Our paper introduces an effective algorithm to detect such clusters, and we perform tests on several real and synthetic data sets to show its effectiveness.

1. INTRODUCTION

Clustering has been extensively studied in many areas, including statistics [14], machine learning [13, 10], pattern recognition [11], and image processing. Recent efforts in data mining have focused on methods for efficient and effective cluster analysis [9, 16, 21] in large databases. Much active research has been devoted to areas such as the scalability of clustering methods and the techniques for high-dimensional clustering.

Clustering in high dimensional spaces is often problematic as theoretical results [5] questioned the meaning of closest matching in high dimensional spaces. Recent research work [2, 3, 4, 6, 12] has focused on discovering clusters embedded in the subspaces of a high dimensional data set. This problem is known as subspace clustering. In this paper, we explore a more general type of subspace clustering which uses pattern similarity to measure the distance between two objects.

1.1 Goal

Most clustering models, including those used in subspace clustering, define similarity among different objects by distances over either all or only a subset of the dimensions. Some well-known distance functions include Euclidean distance, Manhattan distance, and cosine distance. However, distance functions are not always adequate in capturing correlations among the objects. In fact, strong correlations may still exist among a set of objects even if they are far apart from each other as measured by the distance functions. This is demonstrated with an example in Figure 1 and Figure 2.

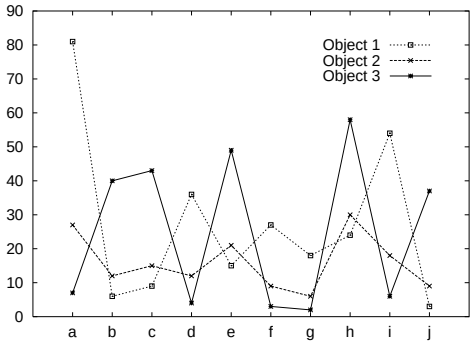
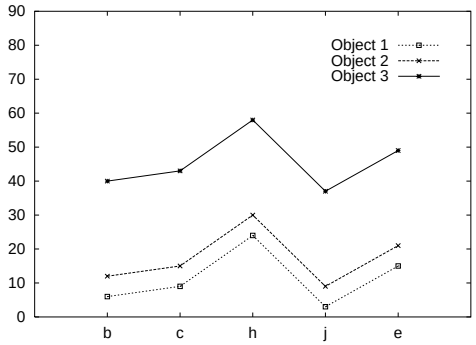


Figure 1: Raw data: 3 objects and 10 columns

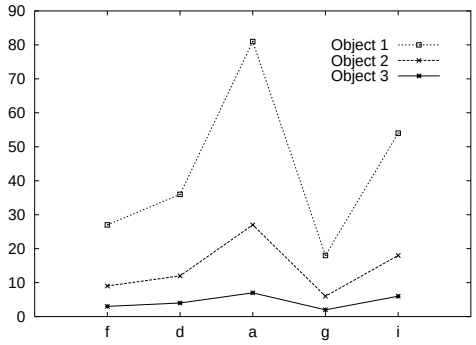
Figure 1 shows a data set with 3 objects and 10 columns (attributes). No patterns among the 3 objects are visibly explicit. However, if we pick a subset of the attributes, $\{b, c, h, j, e\}$, and plot the values of the 3 objects on these attributes in Figure 2(a), it is easy to see that they manifest similar patterns. However, these objects may not be considered in a cluster by any traditional (subspace) clustering model because the distance between any two of them is not close.

The same set of objects can form different patterns on different sets of attributes. In Figure 2(b), we show another pattern in subspace $\{f, d, a, g, i\}$. This time, the three curves

do not have a shifting relationship. Instead, values of object 2 are roughly three times larger than those of object 3, and values of object 1 are roughly three times larger than those of object 2. If we think of columns f, d, a, g, i as different environmental stimuli or conditions, the pattern shows that the 3 objects respond to these conditions coherently, although object 1 is more responsive or more sensitive to the stimuli than the other two. The goal of this paper is to discover such shifting or scaling patterns from raw data sets such as Figure 1.



(a) objects in Figure 1 form a *Shifting Pattern* in subspace $\{b, c, h, j, e\}$



(b) objects in Figure 1 form a *Scaling Pattern* in subspace $\{f, d, a, g, i\}$

Figure 2: Objects form patterns on a set of columns.

1.2 Applications

Discovery of clusters in data sets based on pattern similarity is of great importance because of its potential for actionable insights.

- DNA micro-array analysis: Micro-array is one of the latest breakthroughs in experimental molecular biology. It provides a powerful tool by which the expression patterns of thousands of genes can be monitored simultaneously and is already producing huge amount of valuable data. Analysis of such data is becoming one of the major bottlenecks in the utilization of the technology. The gene expression data are organized as matrices — tables where rows represent genes, columns represent various samples such as tissues or experimental conditions, and numbers in each cell characterize the expression level of the particular gene in the particular sample. Investigations show that more often than not, several genes contribute to a disease, which motivates researchers to identify a subset of genes whose expression levels rise and fall coherently under a subset of conditions, that is, they exhibit fluctuation of a similar shape when conditions change. Discovery of such clusters of genes is essential in revealing the significant connections in gene regulatory networks [8].

- E-commerce: Recommendation systems and target marketing are important applications in the E-commerce area. In these applications, sets of customers/clients with similar behavior need to be identified so that we can predict customers' interest and make proper recommendations. Let's consider the following example. Three viewers rate four movies of a particular type (action, romance, etc.) as (1, 2, 3, 6), (2, 3, 4, 7), and (4, 5, 6, 9), where 1 is the lowest and 10 is the highest score. Although the rates given by each individual are not close, these three viewers have coherent opinions on the four movies. In the future, if the 1st viewer and the 3rd viewer rate a new movie of that category as 7 and 9 respectively, then we have certain confidence that the 2nd viewer will probably like the movie too, since they have similar tastes in that type of movies.

The above applications focus on finding cluster of objects that have coherent behaviors rather than objects that are physically close to each other.

1.3 Challenges

There are two major challenges. First, identifying subspace clusters in high-dimensional data sets is a difficult task because of the curse of dimensionality. Real life data sets in DNA array analysis or collaborative filtering can have hundreds of attributes. Second, a new similarity model is needed as traditional distance functions can not capture the pattern similarity among the objects. For instance, objects in Figure 2(a) and (b) are not close if measured by distance functions such as Euclidean, Manhattan, or Cosine.

Also, the task is not to be confused with pattern discovery in time series data, such as trending analysis in stock closing prices. In time series analysis, patterns occur during a continuous time period. Here, mining is not restricted by any fixed ordering among the columns of the data set. Patterns on an arbitrary subset of the columns are usually deeply buried in the data when the entire set of the attributes are present, as exemplified in Figure 1 and 2.

Similar reasoning reveals why models treating the entire set of attributes as a whole do not work. The $PearsonR$ model [18] studies the coherence among a set of objects. $PearsonR$ defines the correlation between two objects o_1 and o_2 as:

$$\frac{\sum(o_1 - \bar{o_1})(o_2 - \bar{o_2})}{\sqrt{\sum(o_1 - \bar{o_1})^2 \times \sum(o_2 - \bar{o_2})^2}}$$

where $\bar{o_1}$ and $\bar{o_2}$ are the mean of all attribute values in o_1 and o_2, respectively. From this formula, we can see that the $Pearson\ R$ correlation measures the correlation between two objects with respect to all attribute values. A large

positive value indicates a strong positive correlation while a large negative value indicates a strong negative correlation. However, some strong coherence may only exist on a subset of dimensions. For example, in collaborative filtering, six movies are ranked by viewers. The first three are action movies and the next three are family movies. Two viewers rank the movies as $(8, 7, 9, 2, 2, 3)$ and $(2, 1, 3, 8, 8, 9)$. The viewers' ranking can be grouped into two clusters, the first three movies in one cluster and the rest in another. It is clear that the two viewers have consistent bias within each cluster. However, the *Pearson R* correlation of the two viewers is small because globally no explicit pattern exists.

1.4 Our Contributions

Our objective is to cluster objects that exhibit similar patterns on a subset of dimensions. Traditional subspace clustering is a special case in our task, in the sense that objects in a subspace cluster have exactly the same behavior, that there is no coherence need to be related by shifting or scaling. In other words, these objects are physically close – their similarity can be measured by functions such as the Euclidean distance, the Cosine distance, and etc.

Our contributions include:

- We propose a new clustering model, namely the pCluster[1], to capture not only the closeness of objects but also the similarity of the patterns exhibited by the objects.
- The pCluster model is a generalization of subspace clustering. However, it finds a much broader range of applications, including DNA array analysis and collaborative filtering, where pattern similarities among a set of objects carry significant meanings.
- We propose an efficient depth-first algorithm to mine pClusters. Compared with the bicluster approach [7, 20], our method mines multiple clusters simultaneously, detects overlapping clusters, and is resilient to outliers. Our method is deterministic in that it discovers all qualified clusters, while the bicluster approach is a random algorithm that provides only an approximate answer.

1.5 Paper Layout

The rest of the paper is structured as follows. In Section 2, we review some related work, including the bicluster model. Our pCluster model is presented in Section 3. Section 4 explains our clustering algorithm in detail. Experimental results are shown in Section 5 and we conclude the paper in Section 6.

2. RELATED WORK

Clustering in high dimensional spaces is often problematic as theoretical results [5] questioned the meaning of closest matching in high dimensional spaces. Recent research work [4, 2, 3, 12, 6, 15] has focused on discovering clusters embedded in the subspaces of a high dimensional data set. This problem is known as subspace clustering.

[1]pCluster stands for Pattern Cluster

A well known clustering algorithm capable of finding clusters in subspaces is CLIQUE [4]. CLIQUE is a density and grid based clustering method. It discretizes the data space into non-overlapping rectangular cells by partitioning each dimension to a fixed number of bins of equal length. A bin is dense if the fraction of total data points contained in the bin is greater than a threshold. The algorithm finds dense cells in lower dimensional spaces and merge them to form clusters in higher dimensional spaces. Aggarwal et al [2, 3] used an effective techinque for the creation of clusters for very high dimensional data. The PROCLUS [2] and the ORCLUS [3] algorithm find projected clusters based on representative cluster centers in a set of cluster dimensions. Another interesting approach, Fascicles [12], finds subsets of data that share similar values in a subset of dimensions.

The above algorithms find clustered objects if the objects share similar values in a subset of dimensions. In other words, the similarity among the objects is measured by distance functions, such as Euclidean. However, this model captures neither the shifting pattern in Figure 2(a) nor the scaling pattern in Figure 2(b), since objects therein do not share similar values in the subspace where they manifest patterns.

One way to discover the shifting pattern in Figure 2(a) using the above algorithms is through data transformation. Given N attributes, $a_1, ..., a_N$, we define a derived attribute, $A_{ij} = a_i - a_j$, for every pair of attributes a_i and a_j. Thus, our problem is equivalent to mining subspace clusters on the objects with the derived set of attributes. However, the converted data set will have $N(N-1)/2$ dimensions and it becomes intractable even for a small N because of the curse of dimensionality.

Cheng et al introduced the bicluster concept [7] as a measure of the coherence of the genes and conditions in a sub matrix of a DNA array. Let X be the set of genes and Y the set of conditions. Let $I \subset X$ and $J \subset Y$ be subsets of genes and conditions. The pair (I, J) specifies a sub matrix A_{IJ} with the following *mean squared residue score*:

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (d_{ij} - d_{iJ} - d_{Ij} + d_{IJ})^2 \quad (1)$$

where

$$d_{iJ} = \frac{1}{|J|} \sum_{j \in J} d_{ij}, \;\; d_{Ij} = \frac{1}{|I|} \sum_{i \in I} d_{ij}, \;\; d_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} d_{ij}$$

are the row and column means and the means in the submatrix A_{IJ}. A submatrix A_{IJ} is called a δ-bicluster if $H(I, J) \leq \delta$ for some $\delta > 0$. A random algorithm is designed to find such clusters in a DNA array.

Yang et al [20] proposed a move-based algorithm to find biclusters more efficiently. It starts from a random set of seeds (initial clusters) and iteratively improves the clustering quality. It avoids the cluster overlapping problem as multiple clusters are found simultaneously. However, it still has the outlier problem and it requires the number of clusters as an input parameter.

We noticed several limitations of this pioneering work:

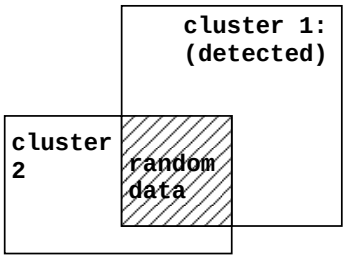


Figure 3: Replacing entries in the shaded area with random values obstructs the discovery of the 2nd cluster.

1. The mean squared residue used in [7, 20] is an averaged measurement of the coherence for a set of objects. A much undesirable property of Formula 1 is that *a submatrix of a δ-bicluster is not necessarily a δ-bicluster*. This creates difficulties in designing efficient algorithms. Furthermore, many δ-biclusters found in a given data set may differ only in one or two outliers they contain. For instance, the bicluster shown in Figure 4(a) contains an obvious outlier but it still has a fairly small mean squared residue (4.238). The only way to get rid of such outliers is to reduce the δ threshold, but that will exclude many biclusters which do exhibit coherent patterns, e.g., the one shown in Figure 4(b) with residue 5.722.

2. The algorithms presented in [7] detects a bicluster in a greedy manner. To find other biclusters after the first one is identified, it mines on a new matrix derived by replacing entries in the discovered bicluster with random data. However, clusters are not necessarily disjoint, as shown in Figure 3. The random data will obstruct the discovery of the 2nd cluster.

3. THE pCluster MODEL

This section describes the pCluster model for mining clusters of objects that exhibit coherent patterns on a set of attributes.

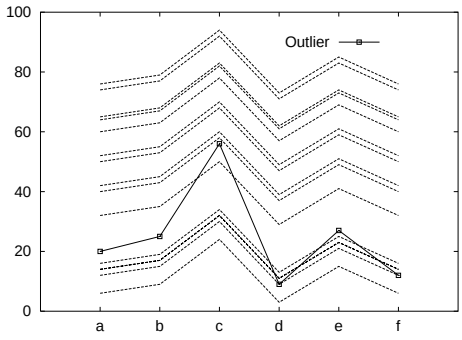
3.1 Notations

$\mathcal{D}$	A set of objects
$\mathcal{A}$	Attributes of objects in $\mathcal{D}$
$(\mathcal{O}, \mathcal{T})$	A submatrix of the data set, where $\mathcal{O} \subseteq \mathcal{D}$, $\mathcal{T} \subseteq \mathcal{A}$
$x, y, ...$	Objects in $\mathcal{D}$
$a, b, ...$	Attributes of $\mathcal{A}$
d_{xa}	Value of object x on attribute a
δ	User-specified clustering threshold
nc	User-specified minimum # of columns of a pCluster
nr	User-specified minimum # of rows of a pCluster
$\mathcal{T}_{xy}$	A maximum dimension set for objects x and y
$\mathcal{O}_{ab}$	A maximum dimension set for columns a and b

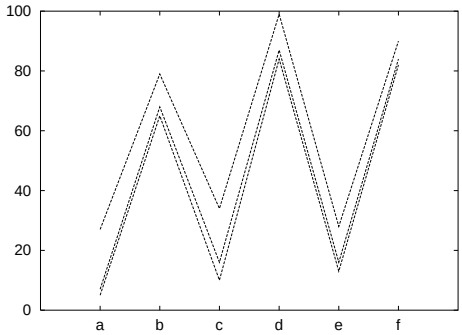
3.2 Definitions and Problem Statement

Let $\mathcal{D}$ be a set of objects, where each object is defined by a set of attributes $\mathcal{A}$. We are interested in objects that exhibit a coherent pattern on a subset of attributes of $\mathcal{A}$.

DEFINITION 1. *Let $\mathcal{O}$ be a subset of objects in the database*



(a) Dataset A: residue 4.238
without the outlier: residue 0



(b) Dataset B: residue 5.722

Figure 4: The mean squared residue can not exclude outliers in a bicluster

($\mathcal{O} \subseteq \mathcal{D}$), and let $\mathcal{T}$ be a subset of attributes ($\mathcal{T} \subseteq \mathcal{A}$). Pair $(\mathcal{O}, \mathcal{T})$ specifies a submatrix. Given $x, y \in \mathcal{O}$, and $a, b \in \mathcal{T}$, we define the pScore of the 2×2 matrix as:

$$pScore\left(\begin{bmatrix} d_{xa} & d_{xb} \\ d_{ya} & d_{yb} \end{bmatrix}\right) = |(d_{xa} - d_{xb}) - (d_{ya} - d_{yb})| \quad (2)$$

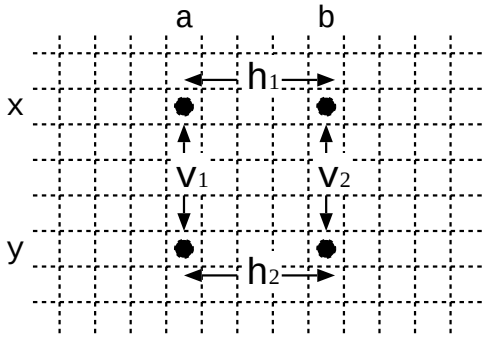
Pair $(\mathcal{O}, \mathcal{T})$ forms a δ-pCluster if for any 2×2 submatrix X in $(\mathcal{O}, \mathcal{T})$, we have $pScore(X) \leq \delta$ for some $\delta \geq 0$.

Intuitively, $pScore(X) \leq \delta$ means that the change of values on the two attributes between the two objects in X is confined by δ, a user-specified threshold. If such confines apply to every pair of objects in $\mathcal{O}$ and every pair of attributes in $\mathcal{T}$ then we have found a δ-pCluster.

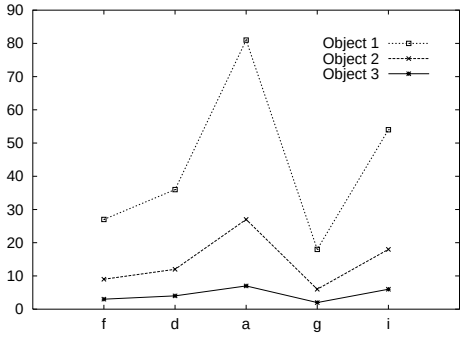
In the bicluster model, a submatrix of a δ-bicluster is not necessarily a δ-bicluster. However, based on the definition of pScore, the pCluster model has the following property:

PROPERTY 1. *Compact Property*
Let $(\mathcal{O}, \mathcal{T})$ be a δ-pCluster. Any of its submatrix, $(\mathcal{O}', \mathcal{T}')$, where $\mathcal{O}' \subseteq \mathcal{O}$, $\mathcal{T}' \subseteq \mathcal{T}$, is also a δ-pCluster.

Note that the definition of pCluster is symmetric: as shown in Figure 5(a), the difference can be measured both horizontally or vertically, as the inequality in Definition 1 is



(a) Definition is symmetric:
$|h_1 - h_2| \leq \delta$ is equivalent to $|v_1 - v_2| \leq \delta$



(b) Objects 1, 2, 3 form a pCluster
after we take the logarithm of the data

Figure 5: pCluster Definition

equivalent to:

$$\begin{aligned} & |(d_{xa} - d_{xb}) - (d_{ya} - d_{yb})| \\ = \; & |(d_{xa} - d_{ya}) - (d_{xb} - d_{yb})| \\ = \; & pScore\left(\begin{bmatrix} d_{xa} & d_{ya} \\ d_{xb} & d_{yb} \end{bmatrix}\right) \end{aligned} \tag{3}$$

When only 2 objects and 2 attributes are considered, the definition of pCluster conforms with that of the bicluster model [7]. According to Formula 1, and assuming $I = \{x, y\}, J = \{a, b\}$, the mean squared residue of a 2×2 matrix $X = \begin{bmatrix} d_{xa} & d_{xb} \\ d_{ya} & d_{yb} \end{bmatrix}$ is:

$$\begin{aligned} H(I,J) &= \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} (d_{ij} - d_{Ij} - d_{iJ} + d_{IJ})^2 \\ &= \frac{((d_{xa} - d_{xb}) - (d_{ya} - d_{yb}))^2}{4} \\ &= (pScore(X)/2)^2 \end{aligned} \tag{4}$$

Thus, for a 2-object/2-attribute matrix, a δ-bicluster is a $(\frac{\delta}{2})^2$-pCluster. However, since a pCluster requires that every 2 objects and every 2 attributes conform with the inequality, it models clusters that are more homogeneous. Let's review the problem of bicluster in Figure 4. The mean squared residue of data set A is 4.238, less than that of data set B, 5.722. Under the pCluster model, the maximum pScore between the outlier and another object in A is 26, while the maximum pScore found in data set B is only 14. Thus, any δ between 14 and 26 will eliminate the outlier in A without obstructing the discovery of the pCluster in B.

In order to model the cluster in Figure 5(b), where there is a scaling factor among the objects, it seems we need to introduce a new inequality:

$$\frac{d_{xa}/d_{ya}}{d_{xb}/d_{yb}} \leq \delta' \tag{5}$$

However, this is not necessary because Formula 2 can be regarded as a logarithmic form of Formula 5. The same pCluster model can be applied to the dataset after we convert the values therein to the logarithmic form. As a matter of fact, in DNA micro-array, each array entry d_{ij}, representing the expression level of gene i in sample j, is derived in the following manner:

$$d_{ij} = log\left(\frac{Red\ Intensity}{Green\ Intensity}\right) \tag{6}$$

where `Red Intensity` is the intensity of gene i, the gene of interest, and `Green Intensity` is the intensity of a reference (control) gene. Thus, the pCluster model can be used to monitor the changes in gene expression and to cluster genes that respond to certain environmental changes in a coherent manner.

conditions

genes	CH1I	CH1B	CH1D	CH2I	CH2B
CTFC3	4392	284	4108	280	228
VPS8	401	281	120	275	298
EFB1	318	280	37	277	215
SSA1	401	292	109	580	238
FUN14	2857	285	2576	271	226
SP07	228	290	48	285	224
MDM10	538	272	266	277	236
CYS3	322	288	41	278	219
DEP1	312	272	40	273	232
NTG1	329	296	33	274	228

(a) gene expression data

conditions

genes	CH1I	CH1B	CH1D	CH2I	CH2B
CTFC3					
VPS8	401		120		298
EFB1	318		37		215
SSA1					
FUN14					
SP07					
MDM10					
CYS3	322		41		219
DEP1					
NTG1					

(b) a pCluster

Figure 6: A pCluster of Yeast Genes

Figure 6(a) shows a micro-array matrix with ten genes (one for each rows) under five experiment conditions (one for each column). This example is a portion of micro-array data that can be found in [19]. A pCluster ({VPS8, EFB1, CYS3}, {CH1I, CH1D, CH2B}) is embedded in the micro-array. Apparently, their similarity can not be revealed by Euclidean distance or Cosine distance.

Objects form a cluster when a certain level of density is reached. The volume of a pCluster is defined by the size of $\mathcal{O}$ and the size of $\mathcal{T}$. The task is thus to find all those pClusters beyond a user-specified volume:

Problem Statement

Given: i) δ, a cluster threshold, ii) nc, a minimal number of columns, and iii) nr, a minimal number of rows, the task of mining is to find all pairs $(\mathcal{O}, \mathcal{T})$ such that $(\mathcal{O}, \mathcal{T})$ is a δ-pCluster according to Definition 1, and $|\mathcal{O}| \geq nr$, $|\mathcal{T}| \geq nc$.

4. THE ALGORITHM

Unlike the bicluster algorithm [7], our approach simultaneously detects multiple clusters that satisfy the user-specified δ threshold. Furthermore, since our algorithm is deterministic, we will not miss any qualified pCluster, while random algorithms for the bicluster approach [7, 20] provide only an approximate answer.

The biggest challenge of our task is in subspace clustering. Objects can form cluster in any subset of the data columns, and the number of data columns in real life applications, such as DNA array analysis and collaborative filtering, are usually in the hundreds or even thousands.

The second challenge is that we want to build a "depth-first" [1] clustering algorithm. Unlike many subspace clustering algorithms [4, 2] that find clusters in lower dimensions first and then merge them to derive clusters in higher dimensions, our approach first generate clusters in the highest dimensions, and then find low dimensional clusters not already covered by the high dimensional clusters. This approach is beneficial because clusters that span a large number of columns are usually more of interest, and it avoids generating clusters which are part of other clusters. It is also more efficient because the combination of low dimensional clusters to form high dimensional ones are usually very expensive.

Note that if the objects cluster on columns $\mathcal{T}$, then they also cluster on any subset of $\mathcal{T}$. In our approach, we are interested in objects clustered on column set $\mathcal{T}$ only if there does not exist $\mathcal{T}' \supset \mathcal{T}$, such that the objects also cluster on $\mathcal{T}'$. To facilitate further discussion, we define the concept of *Maximum Dimension Set (MDS)*.

DEFINITION 2. *Assuming $c = (\mathcal{O}, \mathcal{T})$ is a δ-pCluster. Column set $\mathcal{T}$ is a Maximum Dimension Set (MDS) of c if there does not exist $\mathcal{T}' \supset \mathcal{T}$ such that $(\mathcal{O}, \mathcal{T}')$ is also a δ-pCluster.*

Apparently, objects can form pClusters on multiple MDSs. Our algorithm is depth-first, meaning we only generate pClusters that cluster on MDSs.

4.1 Pairwise Clustering

Given a set of objects $\mathcal{O}$ and a set of columns $\mathcal{A}$, it is not trivial to find all the *Maximum Dimension Sets* for $\mathcal{O}$, since $\mathcal{O}$ can cluster on any subset of $\mathcal{A}$.

Below, we study a special case where $\mathcal{O}$ contains only two objects. Given objects x and y, and a column set $\mathcal{T}$, we define $S(x, y, \mathcal{T})$ as:

$$S(x, y, \mathcal{T}) = \{d_{xa} - d_{ya} | a \in \mathcal{T}\}$$

Based on the definition of δ-cluster, we can make the following observation:

Pairwise Clustering Principle
Given objects x and y, and a dimension set $\mathcal{T}$, x and y form a δ-pCluster on $\mathcal{T}$ iff the difference between the largest and smallest value in $S(x, y, \mathcal{T})$ is below δ.

PROOF. Given objects x and y, we define function $f(a, b)$ on any two dimensions a,$b \in \mathcal{T}$ as:

$$f(a, b) = |(d_{xa} - d_{ya}) - (d_{xb} - d_{yb})|$$

According to the definition of δ-pCluster, objects x and y cluster on $\mathcal{T}$ if $\forall a, b \in \mathcal{T}, f(a, b) \leq \delta$. In other words, $(\{x, y\}, \mathcal{T})$ is a pCluster if the following is true:

$$\max_{a,b \in \mathcal{T}} f(a, b) \leq \delta$$

□

According to the Pairwise Clustering Principle, we do not have to compute $f(a, b)$ for every two dimensions a, b in $\mathcal{T}$, as long as we know the largest and smallest values in $S(x, y, \mathcal{T})$. We use $\vec{S}(x, y, \mathcal{T})$ to denote a sorted sequence of values in $S(x, y, \mathcal{T})$:

$$\vec{S}(x, y, \mathcal{T}) = s_1, ..., s_k$$

$$s_i \in S(x, y, \mathcal{T}) \quad and \quad s_i \leq s_j \ where \ i < j$$

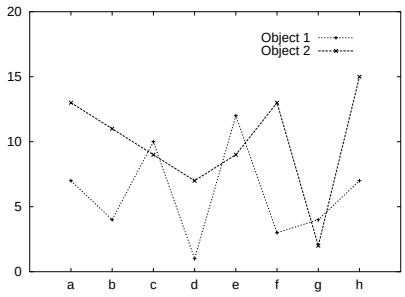
Thus, x and y forms a δ-pCluster on $\mathcal{T}$ if $s_k - s_1 \leq \delta$. Given a set a attributes, $\mathcal{A}$, it is also not difficult to find the maximum dimension sets for object x and y.

LEMMA 2. *MDS Principle*
Given a set of dimensions $\mathcal{A}$, $\mathcal{T}_s \subseteq \mathcal{A}$ is a maximum dimension set of x and y iff:
i) $\vec{S}(x, y, \mathcal{T}_s) = s_i \cdots s_j$ is a (contiguous) subsequence of $\vec{S}(x, y, \mathcal{T}) = s_1 \cdots s_i \cdots s_j \cdots s_k$, and
ii) $s_j - s_i \leq \delta$, whereas $s_{j+1} - s_i > \delta$ and $s_j - s_{i-1} > \delta$.

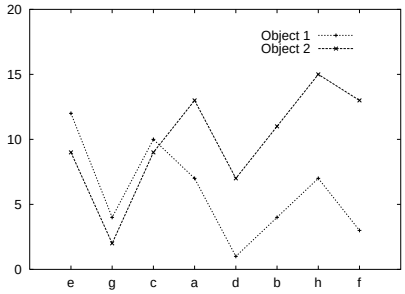
PROOF. Given $\vec{S}(x, y, \mathcal{T}_s) = s_i \cdots s_j$ and $s_j - s_i \leq \delta$, according to the pairwise clustering principle, $\mathcal{T}_s$ is a δ-pCluster. Furthermore, $\forall a \in \mathcal{T} - \mathcal{T}_s$, we have $d_{xa} - d_{ya} \geq s_{j+1}$ or $d_{xa} - d_{ya} \leq s_{i-1}$, otherwise $a \in \mathcal{T}_s$ since $\vec{S}(x, y, \mathcal{T}_s) = s_i \cdots s_j$. If $d_{xa} - d_{ya} \geq s_{j+1}$, from $s_{j+1} - s_i > \delta$ we get $(d_{xa} - d_{ya}) - s_i > \delta$, thus $\{a\} \cup \mathcal{T}_s$ is not a δ-pCluster. On the other hand, if $d_{xa} - d_{ya} \leq s_{i-1}$, from $s_j - s_{i-1} > \delta$ we get $s_j - (d_{xa} - d_{ya}) > \delta$, thus $\{a\} \cup \mathcal{T}_s$ is not a δ-pCluster either. Since $\mathcal{T}_s$ can not be enlarged, $\mathcal{T}_s$ is an MDS. □

Based on the MDS principle, we can find the MDSs for objects x and y in the following manner: we start with both the left-end and the right-end placed on the first element of the sorted sequence, and we move the right-end rightward one position at a time. For every move, we compute the difference of the values at the two ends, until the difference is greater than δ. At that time, the elements between the two ends form a maximum dimension set. To find the next maximum dimension set, we move the left-end rightward one position, and repeat the above process. It stops when the right-end reaches the last element of the sorted sequence.

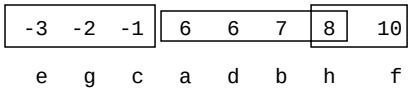
Figure 7 gives an example of the above process. We want to find the maximum dimension sets for two objects whose values on 8 dimensions are shown in Figure 7(a). The patterns are hidden until we sort the dimensions by the difference of x and y on each dimension. The sorted sequence



(a) Raw Data



(b) Sort by dimension discrepancy



(c) Cluster on sorted differences ($\delta = 2$)

Figure 7: Finding MDS for two objects

$S = -3, -2, -1, 6, 6, 7, 8, 10$ is shown in Figure 7(c). Assuming $\delta = 2$, we start at the left end of S. We move rightward until we stop at the first 6, since $6-(-3) > 2$. The columns between the left end and 6, $\{e, g, c\}$, is an MDS. We move the left end to -2 and repeat the process until we find all 3 maximum dimension sets for x and y: $\{e, g, c\}$, $\{a, d, b, h\}$, and $\{h, f\}$. Note that maximum dimension sets might overlap.

A formal description of the above process is given in Algorithm 1. We use the following procedure to find MDSs for objects x and y:

$$pairCluster(x, y, \mathcal{A}, nc)$$

where nc is the (user-specified) minimal number of columns in a pCluster. According to the definition of the pCluster model, the columns and the rows of the data matrix carry the same significance. Thus, the same method can be used to find MDSs for each column pair, a and b:

$$pairCluster(a, b, \mathcal{O}, nr)$$

The above procedure returns a set of MDSs for column a and b, except that here the maximum dimension set is made up of objects instead of columns. As an example, we study the data set shown in Figure 8(a). We find 2 object-pair MDSs and 4 column-pair MDSs.

4.2 MDS Pruning

The number of pairwise maximum dimension sets depends on the clustering threshold δ and the user-specified minimum number of columns, nc. However, if $nr > 2$, then only some of the pairwise MDSs are *valid*, i.e., they actually occur in δ-pClusters with a size larger than $nr \times nc$. In this section, we describe how the MDSs can be pruned to eliminate invalid pairwise clusters.

Input: x, y: two objects, $\mathcal{T}$: set of columns, nc: minimal number of columns, δ: cluster threshold
Output: All δ-pClusters with more than nc columns

$s \leftarrow d_x - d_y$; /* i.e., $s_i \leftarrow d_{xi} - d_{yi}$ for each i in $\mathcal{T}$ */
sort array s;
$start \leftarrow 0$; $end \leftarrow 1$;
$new \leftarrow$ `TRUE`; /* a boolean variable, if TRUE, indicates an untested column in $[start, end]$ */
repeat
  $v \leftarrow s_{end} - s_{start}$;
  if $|v| \leq \delta$ **then**
    /* expands δ-pCluster to include one more columns */
    $end \leftarrow end + 1$;
    $new \leftarrow$ `TRUE`;
  else
    Return δ-pCluster if $end - start \geq nc$ and $new =$ `TRUE`;
    $start \leftarrow start + 1$;
    $new \leftarrow$ `FALSE`;
until $end \geq |\mathcal{T}|$;
Return δ-pCluster if $end - start \geq nc$ and $new =$ `TRUE`;

Algorithm 1: Find two-object pClusters: pairCluster$(x, y, \mathcal{T}, nc)$

Given a clustering threshold δ, and minimum cluster size $nr \times nc$, we use $\mathcal{T}_{xy}$ to denote an MDS for objects x and y, and $\mathcal{O}_{ab}$ to denote an MDS for columns a and b. We prove the following lemma:

LEMMA 3. *MDS Pruning Principle*
Let $\mathcal{T}_{xy}$ be an MDS for objects x, y, and $a \in \mathcal{T}_{xy}$. For any $\mathcal{O}$ and $\mathcal{T}$, a necessary condition of $(\{x, y\} \cup \mathcal{O}, \{a\} \cup \mathcal{T})$ being a δ-pCluster is $\forall b \in \mathcal{T}$, $\exists \mathcal{O}_{ab} \supseteq \{x, y\}$.

PROOF. Assume $(\{x, y\} \cup \mathcal{O}, \{a\} \cup \mathcal{T})$ is a δ-pCluster. Since a submatrix of a δ-pCluster is also a δ-pCluster, we know $\forall b \in \mathcal{T}$, $(\{x, y\} \cup \mathcal{O}, \{a, b\})$ is a δ-pCluster. According to the definition of MDS, there exists at least one MDS $\mathcal{O}_{ab} \supseteq \{x, y\} \cup \mathcal{O} \supseteq \{x, y\}$. Thus, there are at least $|\mathcal{T}|$ such MDSs. □

Since we are only interested in δ-pClusters $(\{x, y\} \cup \mathcal{O}, \{a\} \cup \mathcal{T})$ with size $\geq nr \times nc$, the minimum number of such column MDSs is $nc - 1$. Thus, the pruning criterion can be stated as follows:

For any dimension a in a MDS $\mathcal{T}_{xy}$, we count the number of $\mathcal{O}_{ab}$ that contain $\{x, y\}$. If the number of such $\mathcal{O}_{ab}$ is less than $nc - 1$, we remove a from $\mathcal{T}_{xy}$. Furthermore, if the removal of a makes $|\mathcal{T}_{xy}| < nc$, we remove $\mathcal{T}_{xy}$ as well.

	c_0	c_1	c_2
o_0	1	4	2
o_1	2	5	5
o_2	3	6	5
o_3	4	200	7
o_4	300	7	6

(a) A 5×3 Data Matrix

$(o_0, o_2) \rightarrow \{c_0, c_1, c_2\}$
$(o_1, o_2) \rightarrow \{c_0, c_1, c_2\}$

(b) MDS for object pairs

$(c_0, c_1) \rightarrow \{o_0, o_1, o_2\}$
$(c_0, c_2) \rightarrow \{o_1, o_2, o_3\}$
$(c_1, c_2) \rightarrow \{o_1, o_2, o_4\}$
$(c_1, c_2) \rightarrow \{o_0, o_2, o_4\}$

(c) MDS for column pairs

Figure 8: *Maximum Dimension Sets* **for Column- and Object-pairs ($\delta = 1$, $nc = 3$, and $nr = 3$)**

$(c_0, c_1) \rightarrow \{o_0, o_1, o_2\}$
$(c_0, c_2) \rightarrow \{o_1, o_2, o_3\}$
$(c_1, c_2) \rightarrow \{o_1, o_2, o_4\}$
$(c_1, c_2) \rightarrow \{o_0, o_2, o_4\}$

(a) Generating MDS_c from data.

$(o_0, o_2) \rightarrow \{c_0, c_1, c_2\}$ $\times$
$(o_1, o_2) \rightarrow \{c_0, c_1, c_2\}$

(b) Generating MDS_o from data, using MDS_c in (a) for pruning

$(c_0, c_1) \rightarrow \{o_0, o_1, o_2\}$ $\times$
$(c_0, c_2) \rightarrow \{o_1, o_2, o_3\}$ $\times$
$(c_1, c_2) \rightarrow \{o_1, o_2, o_4\}$ $\times$
$(c_1, c_2) \rightarrow \{o_0, o_2, o_4\}$ $\times$

(c) Pruning MDS_c in (a) using MDS_o in (b)

Figure 9: Generating and Pruning MDS iteratively ($\delta = 1$, $nc = 3$, and $nr = 3$)

Apparently, the same reasoning applies to pruning column-pair MDSs. Indeed, we can prune the column-pair MDSs and object-pair MDSs by turns. We first generate column-pair MDSs from the data set. Next, when we generate object-pair MDSs, we use column-pair MDSs for pruning. Then, we prune column-pair MDSs using the pruned object-pair MDSs. This procedure can go on until no more MDSs can be eliminated.

We continue with our example in Figure 8. First, we choose to generate column-pair MDSs, and they are shown in Figure 9(a). Second, we generate object-pair MDSs. MDS $(o_0, o_2) \rightarrow \{c_0, c_1, c_2\}$ is to be eliminated because the column-pair MDS of (c_0, c_2) does not contain o_0. Third, we review the column-pair MDSs based on the remaining object-pair MDSs, and we find each of them is to be eliminated. Thus, the original data set in Figure 8(a) does not contain any 3×3 pCluster.

4.3 The Main Algorithm

Algorithm 2 outlines the main routine of the mining process. It can be summarized in three steps. In the first step, we scan the dataset to find column-pair MDSs for every column-pair, and object-pair MDSs for every object-pair. This step is realized by calling procedure `pairCluster()` in Algorithm 1. Our implementation uses bitmaps to manage column sets and object sets in MDSs. We store these MDSs on disk.

In the second step, we prune object-pair MDSs and column-pair MDSs by turns until no pruning can be made. The pruning process is described in Section 4.2.

In the third step, we insert the remaining object-pair MDSs into a prefix tree. Each node of the prefix tree uniquely represents a maximum dimension set. Let's assume there exists a total order among the columns in $\mathcal{A}$, for instance, $a \prec b \prec c \prec \cdots$. An example of a prefix tree is shown in Figure 10. To insert a 2-object pCluster $(\mathcal{O}, \mathcal{T})$ into the prefix tree, we first sort the columns in $\mathcal{T}$ into its ordered form, $\mathcal{T}'$. Then we descend from the root of the tree following the path of the columns in $\mathcal{T}'$ and add the two objects in $\mathcal{O}$ to the node we reached in the end. For instance, if objects $\{x, y\}$ clusters on columns $\mathcal{T} = \{f, c, a\}$, then x and y are inserted into the tree by following the path acf. Since any qualified MDS has more than nc dimensions, no objects will be inserted into nodes whose depth is less than nc.

After all the objects are inserted, each node of the tree can be regarded as a candidate cluster, $(\mathcal{O}, \mathcal{T})$. The next step is to prune objects in $\mathcal{O}$ so that $(\mathcal{O}, \mathcal{T})$ becomes a pCluster. This is realized by calling the *pairCluster*() routine again. The fact that objects in $\mathcal{O}$ cluster on column set $\mathcal{T}$ also means that they cluster on $\mathcal{T}'$, where $\mathcal{T}'$ is any subset of $\mathcal{T}$. Let's assume $\mathcal{T}'$ is represented by another node, m. To prune objects in node m we have to consider the objects in $\mathcal{O}$ as well. Based on this observation, we start with the leaf nodes. Let n be a leaf node, and it represents a candidate cluster $(\mathcal{O}, \mathcal{T})$. Apparently, there does not exist any node in the prefix tree whose column set contains $\mathcal{T}$, so we can safely prune the objects in $\mathcal{O}$ to find all the pClusters. After it is done, we add the objects in $\mathcal{O}$ to nodes whose column set $\mathcal{T}' \subset \mathcal{T}$ and $|\mathcal{T}'| = |\mathcal{T}| - 1$.

Based on the above discussion, we perform a post-order traversal of the prefix tree. For each node, we detect the pClusters contained within. Then we distribute the objects in the node to other nodes which represent a sub column set of the current node. We repeat this process until no nodes of depth $\geq nc$ are left.

Algorithm Complexity

The initial generation of MDSs in Algorithm 2 has time complexity $O(M^2 N log N + N^2 M log M)$, where M is the number

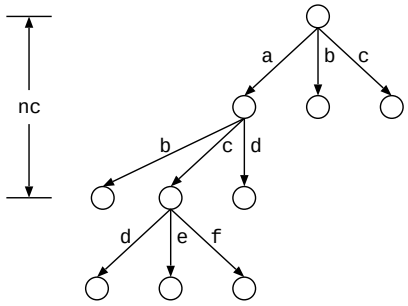


Figure 10: A Prefix Tree. Each node represents a candidate cluster, $(\mathcal{O}, \mathcal{T})$, where $\mathcal{O}$ is the set of objects inserted into this node, and $\mathcal{T}$ is the set of columns (represented by the arcs leading from the root to this node).

Input: $\mathcal{D}$: data set, δ: pCluster threshold
nc: minimal number of columns, nr: minimal number of rows
Output: All pClusters with size $\geq nr \times nc$

for *each* $a, b \in \mathcal{A}, a \neq b$ **do**
  find column-pair MDSs: $pairCluster(a, b, \mathcal{D}, nr)$;
for *each* $x, y \in \mathcal{D}, x \neq y$ **do**
  find object-pair MDSs: $pairCluster(x, y, \mathcal{A}, nc)$;
repeat
  for *each object-pair pCluster* $(\{x, y\}, T))$ **do**
    use column-pair MDSs to prune columns in T;
    eliminate MDS $(\{x, y\}, T)$ if $|T| < nc$;
  for *each column-pair pCluster* $(\{a, b\}, O))$ **do**
    use object-pair MDSs to prune objects in O;
    eliminate MDS $(\{a, b\}, O)$ if $|O| < nr$;
until *no pruning takes place*;
insert all object-pair MDSs $(\{x, y\}, T)$ into the prefix tree: $insertTree(x, y, T)$;
make a post-order traversal of the tree;
for *each node n encountered in the post-order traversal* **do**
  $\mathcal{O}$:= objects in node n;
  $\mathcal{T}$:= columns represented by node n;
  for *each* $a, b \in \mathcal{T}$ **do**
    find column-pair MDSs: $\mathcal{C} = pairCluster(a, b, \mathcal{O}, nr)$;
    remove from $\mathcal{O}$ those objects not contained in any MDS $c \in \mathcal{C}$;
  Output $(\mathcal{O}, \mathcal{T})$;
  Add objects in n to nodes which has one less column than n;

Algorithm 2: Main Algorithm for Mining pClusters: pCluster()

of columns and N is the number of objects. The worst case for pruning is $O(kM^2N^2)$, although on average it is much less, since the average size of a column-pair MDS (number of objects in a MDS) is usually much smaller than M. The worst case of prefix-tree depth-first clustering complexity is exponential with regard to the number of columns. However, since most invalid MDSs are eliminated in the pruning phase, the complexity of this final step is greatly reduced.

5. EXPERIMENTS

We experimented our pCluster algorithm with both synthetic and real life data sets. The algorithm is implemented on a Linux machine with a 700 MHz CPU and 256 MB main memory. Traditional subspace clustering algorithms can not find clusters based on pattern similarity. We implemented an alternative algorithm that first transforms the matrix by creating a new column A_{ij} for every two columns a_i and a_j, provided $i > j$. The value of the new column A_{ij} is derived by $A_{ij} = a_i - a_j$. Thus, the new data set will have $N(N-1)/2$ columns, where N is the number of columns in the original data set. Then, we apply a subspace clustering algorithm on the transformed matrix, and discover subspace clusters from the data. There are several subspace clustering algorithms to choose from and we used CLIQUE [4] in our experiments.

5.1 Data Sets

We experiment our pCluster algorithm with synthetic data and two real life data sets: one is the MovieLens data set and the other is a DNA microarray of gene expression of a certain type of yeast under various conditions.

Synthetic Data

We generate synthetic data sets in matrix forms. Intially, the matrix is filled with random values ranged from 0–500, and then we embed a fixed number of pClusters in the raw data. Besides the size of the matrix, the data generator takes several other parameters: nr, the average number of rows of the embedded pClusters; nc, the average number of columns; and k, the number of pClusters embeded in the matrix. To make the generator algorithm easy to implement, and without loss of generality, we embed *perfect* pClusters in the matrix, i.e., each embedded pCluster satisfies a cluster threshold $\delta = 0$. We investigate both the correctness and the performance of our pCluster algorithm using the synthetic data.

Gene Expression Data

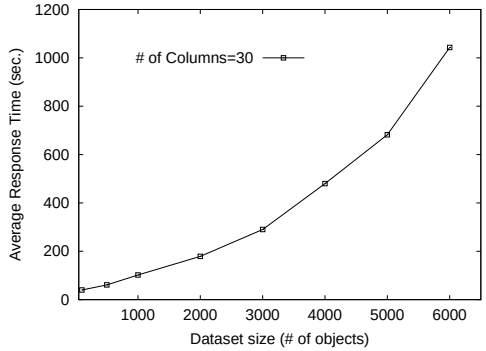
Gene expression data are being generated by DNA chips and other microarray techniques. The yeast microarray contains expression levels of 2,884 genes under 17 conditions [19]. The data set is presented as a matrix. Each row corresponds to a gene and each column represents a condition under which the gene is developed. Each entry represents the relative abundance of the mRNA of a gene under a specific condition. The entry value, derived by scaling and logarithm from the original relative abundance, is in the range of 0 and 600. Biologists are interested in the finding of a subset of genes showing strikingly similar up-regulation and down-regulation under a subset of conditions [7].

MovieLens Data Set

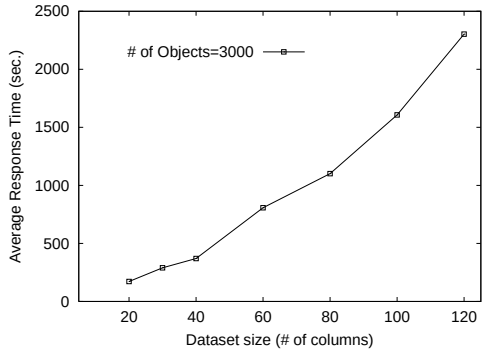
The MovieLens data set [17] was made available by the GroupLens Research Project at the University of Minnesota. The data set contains 100,000 ratings, 943 users and 1682 movies. Each user has rated at lease 20 movies. A user is considered as an object while a movie is regarded as an attribute. In the data set, many entries are empty since a user only rated less than 10% movies on average.

5.2 Performance Analysis

We evaluate the performance of the pCluster algorithm as we increase the number of objects and columns in the dataset. The results presented in Figure 11 are average response times obtained from a set of 10 synthetic data. As we know, the columns and the rows of the matrix carry the same significance in the pCluster model, which is symmetrically defined in Formula 2. Although the algorithm is not entirely symmetric as it chooses to generate MDSs for column-pairs first, the curves in Figure 11 demonstrate similar superlinear patterns.



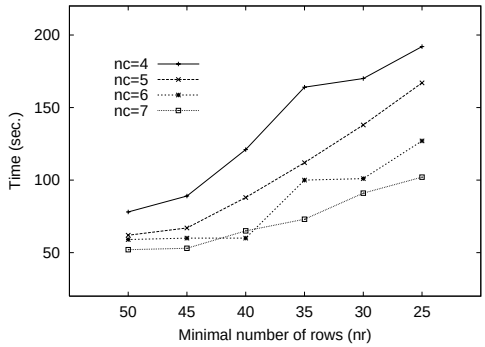
(a) Response time varying # of rows in data sets



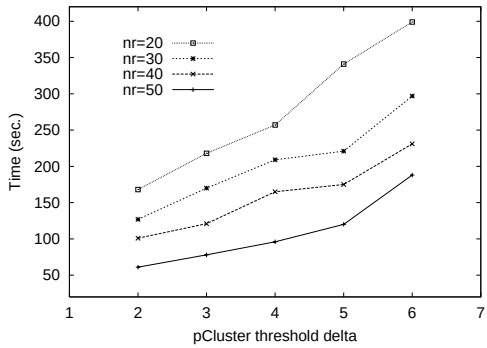
(b) Response time varying # of columns in data sets

Figure 11: Performance Study: response time vs. data size.

Data sets used for Figure 11(a) are generated with number of columns fixed at 30. There is a total of 30 embedded pClusters in the data. The minimal number of columns of the embedded pCluster is 6, and the minimal number of rows is set to $0.01N$, where N is the number of rows of the synthetic data. The mining algorithm is invoked with $\delta = 3$, $nc = 5$, and $nr = 0.01N$. Data sets used in Figure 11(b) are generated in the same manner, except that the number of rows is fixed at 3,000, and each embedded pClusters has at least $0.02C$ columns, where C is the number of columns of the data set. The mining algorithm is invoked with $\delta = 3$, $nc = 0.02C$, and $nr = 30$.



(a) Response time v. min # of rows (with δ fixed at 3)



(b) Response time v. mining threshold δ (with nc fixed at 4)

Figure 12: Sensitiveness to Mining Parameters: δ, nc, and nr

Next we study the impact of the mining parameters (δ, nc, and nr) on the response time. The results are shown in Figure 12. The synthetic data sets in use have 3,000 objects, 30 columns, and the volume of each of the 30 embedded pClusters is 30×5 on average.

We now evaluate the pruning method used in the algorithm. The data sets in use are $3,000 \times 100$ in size, each of the 30 embeded pClusters having a minimal volume 30×15. The statistics collected in Table 1 and Table 2 show that the pruning method reduces the number of MDS dramatically. In Table 2, the time spent in the first round is roughly the same, because the data set has fixed size. The time spent in round 2 and afterwards decreases as the number of valid MDS goes down.

The pruning process is essential in the pCluster algorithm. This is demonstrated by Figure 13(a), where the data set in use is the same as in Figure 11(a). Without pruning, the clustering algorithm can not go beyond $3,000$ objects, as the number of the MDSs become too large to put into a prefix tree. It also shows that roughly 5 rounds of pruning are enough to get rid of most of the invalid MDSs.

Finally, we compare the pCluster algorithm with an alternative approach based on the subspace clustering algorithm CLIQUE [4]. The data set has $3,000$ objects and the subspace algorithm does not scale when the number of columns

Parameters			# of MDSs for object pairs after pruning					# of found clusters
δ	nc	nr	1st round	2nd round	3rd round	4th round	5th round	
3	16	50	8,184	2,132	102	0	0	0
3	14	40	46,360	19,142	10,241	5,710	3,123	1
4	14	35	78,183	41,435	26,625	16,710	9,104	8
4	12	30	415,031	119,342	62,231	39,101	21,176	30
5	12	30	726,145	345.432	182,127	110,452	77,352	30

Table 1: Pruning MDS for object pairs (first 5 rounds)

Parameters			Time spent in the first 5 rounds of pruning					Total time (including all rounds)
δ	nc	nr	1st round	2nd round	3rd round	4th round	5th round	
3	16	50	73	16	10	0	0	99
3	14	40	73	18	14	9	3	117
4	14	35	75	28	18	10	4	161
4	12	30	76	49	27	19	9	222
5	12	30	77	63	39	23	10	253

Table 2: Time (sec.) spent in each round of pruning

goes beyond 100.

We apply the pCluster algorithm on the yeast gene microarray [19]. We present some performance statistics in Figure 14. It shows that a majority of maximum dimension sets are eliminated after the 1st and 2nd round. The overall running time is around 200-300 seconds, depending on the user parameters. Our algorithm has performance advantage over the bicluster algorithm [7], as it takes roughly 300-400 seconds for the bicluster algorithm to find a single cluster. We also discovered some interesting pClusters in the MovieLens dataset. For example, there is a cluster whose attributes consists of two types of movies, family movies (e.g., First Wives Club, Adam Family Values, etc.) and the action movies (e.g., , Golden Eye, Rumble in the Bronx, etc.). Also the rating given by the viewers in this cluster is quite different, however, they share a common phenomenon: the rating of the action movies is about 2 points higher than that of the family movies. This cluster can be discovered in the pCluster model. For example, two viewers rate four movies as (3, 3, 4, 5) and (1, 1, 2, 3). Although the absolute distance between the two rankings are large, i.e., 4, but the pCluster model groups them together because they are coherent.

6. CONCLUSION AND FUTURE WORK

Recently, there has been considerable amount of research in subspace clustering. Most of the approaches define similarity among objects based on their distances (measured by distance functions, e.g. Euclidean) in some subspace. In this paper, we proposed a new model called pCluster to capture the similarity of the patterns exhibited by a cluster of objects in a subset of dimensions. Traditional subspace clustering, which focuses on value similarity instead of pattern similarity, is a special case of our generalized model. We devised a depth-first algorithm that can efficiently and effectively discover all the pClusters with a size larger than a user-specified threshold.

The pCluster model finds a wide range of applications including management of scientific data, such as the DNA microarray, and ecommerce applications, such as collaborative filitering. In these datasets, although the distance among the objects may not be close in any subspace, they can still manifest shifting or scaling patterns, which are not captured by tradition (subspace) clustering algorithms. We have demonsrtated that these patterns are often of great interest in DNA microarray analysis, collaborative filtering, and other applications.

As for future work, we believe the concept of similarity in pattern distance spaces has opened the door to quite a few research topics. For instance, currently, the similarity model used in data retrieval and nearest neighbor search is based on value similarity. By extending the model to reflect pattern similarity will benefit a lot of applications.

7. REFERENCES

[1] R. C. Agarwal, C. C. Aggarwal, and V. Parsad. Depth first generation of long patterns. In *SIGKDD*, 2000.

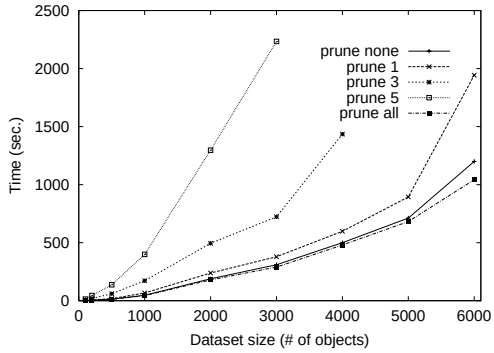
[2] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD*, 1999.

[3] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD*, pages 70–81, 2000.

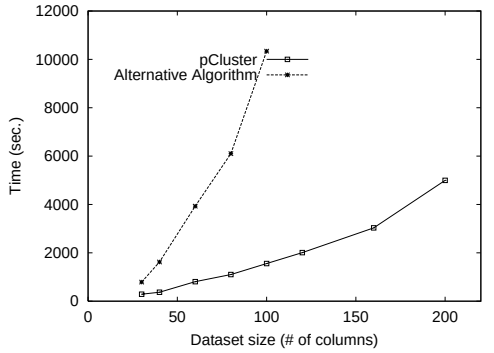
[4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Authomatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, 1998.

[5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful. In *Proc. of the Int. Conf. Database Theories*, pages 217–235, 1999.

[6] C. H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *SIGKDD*, pages 84–93, 1999.

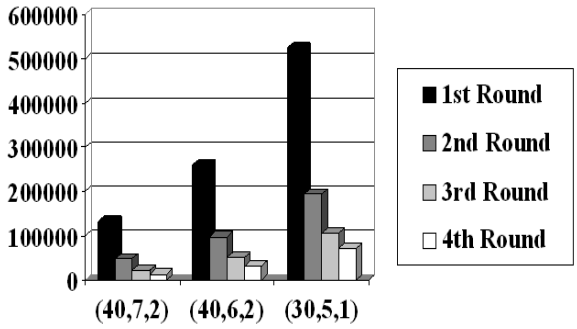


(a) Rounds of pruning and response time



(b) Subspace clustering vs. pCluster

Figure 13: Performance study on rounds of pruning and comparison with subspace clustering



(a) Number of MDSs for gene pairs

Parameters			Pruning Time (sec.)				Total time
nr	*nc*	δ	1st	2nd	3rd	4th	(all rounds)
40	7	2	67	23	10	7	197
40	6	2	67	32	23	17	257
30	5	1	69	39	28	17	312

(b) Pruning time

Figure 14: Performance statistics in mining the Yeast gene microarray

large data sets. Technical Report 9906-010, Northwestern University, 1999.

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

[8] P. D'haeseleer, S. Liang, and R. Somogyi. Gene expression analysis and genetic network modeling. In *Pacific Symposium on Biocomputing*, 1999.

[9] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-bsed algorithm for discovering clusters in large spatial databases with noise. In *SIGKDD*, pages 226–231, 1996.

[10] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. In *Machine Learning*, 1987.

[11] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[12] H. V. Jagadish, J. Madar, and R. Ng. Semantic compression and pattern extraction with fascicles. In *VLDB*, pages 186–196, 1999.

[13] R. S. Michalski and R. E. Stepp. Learning from observation: conceptual clustering. In *Machine Learning: An Artificial Intelligence Approach*, pages 331–363, 1983.

[14] F. Murtagh. A survey of recent hierarchical clustering algorithms. In *The Computer Journal*, 1983.

[15] H. Nagesh, S. Goil, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very

[16] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *VLDB*, 1994.

[17] J. Riedl and J. Konstan. Movielens dataset. In *http://www.cs.umn.edu/Research/GroupLens*.

[18] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating 'word of mouth'. In *Proceeding of ACM CHI*, pages 210–217, 1995.

[19] S. Tavazoie, J. Hughes, M. Campbell, R. Cho, and G. Church. Yeast micro data set. In *http://arep.med.harvard.edu/biclustering/yeast.matrix*, 2000.

[20] J. Yang, W. Wang, H. Wang, and P. S. Yu. δ-clusters: Capturing subspace correlation in a large data set. In *ICDE*, pages 517–528, 2002.

[21] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *SIGMOD*, pages 103–114, 1996.