Discovering Correlated Subspace Clusters in 3D Continuous-Valued Data

Kelvin Sim
Institute for Infocomm Research
*A*STAR, Singapore*
Email: shsim@i2r.a-star.edu.sg

Zeyar Aung
Masdar Institute of Science and Technology
Abu Dhabi, United Arab Emirates
Email: zaung@masdar.ac.ae

Vivekanand Gopalkrishnan
Nanyang Technological University
Singapore
Email: asvivek@ntu.edu.sg

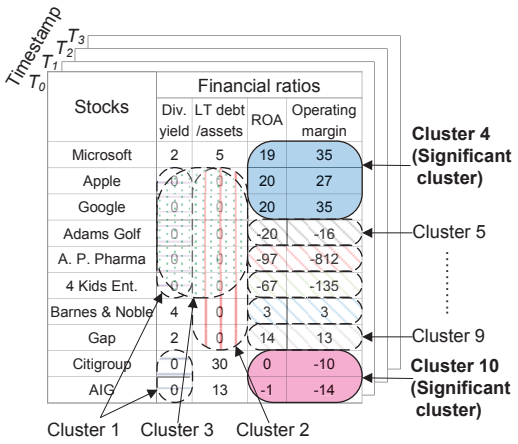
***Abstract*—Subspace clusters represent useful information in high-dimensional data. However, mining significant subspace clusters in continuous-valued 3D data such as *stock-financial ratio-year* data, or *gene-sample-time* data, is difficult. Firstly, typical metrics either find subspaces with very few objects, or they find too many insignificant subspaces – those which exist by chance. Besides, typical 3D subspace clustering approaches abound with parameters, which are usually set under biased assumptions, making the mining process a 'guessing game'. We address these concerns by proposing an information theoretic measure, which allows us to identify 3D subspace clusters that stand out from the data. We also develop a highly effective, efficient and parameter-robust algorithm, which is a hybrid of information theoretical and statistical techniques, to mine these clusters. From extensive experimentations, we show that our approach can discover significant 3D subspace clusters embedded in 110 synthetic datasets of varying conditions. We also perform a case study on real-world stock datasets, which shows that our clusters can generate higher profits compared to those mined by other approaches.**

***Keywords*-3D subspace clustering, financial data mining, information theory.**

I. Introduction

Three-dimensional (3D) data, in the general form of *object-attribute-time/location* has become increasingly popular in data analysis. Many real-world applications, such as microarray analysis based on *gene-sample-time* or *gene-sample-region* data, and stock analysis based on *stock-financial ratio-year* data, basically cluster the continuous 3D data to perform their task. However, because these data are essentially high dimensional, traditional clustering approaches operating on the full data space become ineffective. Zhao and Zaki [1] attempted to tackle this problem by clustering subspaces in the 3D data, so that objects are grouped based upon their similarity in some subset of attributes and time. In such formulations, a 3D subspace cluster can be considered as a cuboid spanned by a group of objects, a group of attributes and a group of timestamps. This cuboid is inherently axis-parallel, which is important for the user to easily interpret and understand the cluster.

In order to be useful, a 3D subspace cluster generally requires a substantial number of objects which have similar values in a subset of attributes and timestamps. Consider the *stock-financial ratio-year* 3D continuous dataset as shown in Fig. 1(a). The figure represents 3D subspace clusters of



(a)

Metric	Clusters obtained
Support	1,2,3,4,10
Lift [2]	5,...,9
Non-uniform distributed [3]	1,2,4,10
Correlation information [4]	4,10

(b)

Figure 1. (a) Each rounded rectangle represents a 3D subspace cluster. Solid rectangles represent significant clusters, as their values have high occurrences and their occurrences purely constitute the cluster. (b) 3D subspace clusters considered to be significant by different metrics.

different characteristics by rounded rectangles. Intuitively, clusters 4 and 10 are the most significant among them; technology stocks Microsoft, Apple, Google are in cluster 4, and financial stocks Citigroup and AIG are in cluster 10. On the other hand, clusters 1,2,3 contain stocks from different industries, while clusters 5 to 9 contain a single stock each. Clusters 1 and 2 are based on values that most of the stocks in the dataset have, which is obvious and useless information. Cluster 3 is induced by the clusters 1 and 2, thus its existence is by chance. Both, clusters 4 and 10, have high occurrences of similar values on multiple financial ratios, and all occurrences of the similar values constitute the cluster.

We denote such *significant* axis-parallel 3D subspace clusters as **C***orrelated* 3D **S**ubspace **C**lusters (CSCs), where the values in each cluster have the *correlation* characteristics:

- they have high co-occurrences

1550-4786/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDM.2010.19

- their co-occurrences are not by chance

Having defined the desired clusters, we are now faced with an open question: *"How to measure the goodness of a CSC?"* Almost all subspace clustering algorithms use the *support* metric, which requires the values in the cluster to have high occurrences together, but they do not consider the second characteristic. Brin et al. [2] proposed the *lift* metric, which measures the second characteristic, but is biased towards values with low occurrences. Moise and Sander [3] proposed that a cluster is significant when the occurrences of values in it do not follow a uniform distribution. This statistical approach handles the above characteristics, but it can only handle data with uniform distribution.

We propose using a metric *correlation information*, which also embodies the above characteristics, but is not dependent on the data distribution. Correlation information quantifies the correlation between a pair of subspace clusters [4], thus it can be naturally extended to quantify the correlation within a 3D subspace cluster. Figure 1(b) shows the 3D subspace clusters obtained using different metrics. As we can see, several metrics can be used to find the desired CSCs, but some of them generate spurious results.

In order to use correlation information on 3D continuous-valued data, we must calculate the probability density function (pdf) and probabilities of high dimensional continuous values, which is extremely challenging. We use kernel density estimation to estimate the pdf, as it is non-parametric and can converge to the true pdf [5]. To calculate the probabilities of high dimensional continuous values, we use the *normalized pdf* method [6], which is faster than performing Monte Carlo integration of pdf [7] and more accurate than estimating the pdf as the probability [8].

Now that an appropriate metric for CSCs has been chosen, the remaining task is to design a subspace clustering algorithm that mines the clusters based upon this metric. Generally, most clustering algorithms require the user to set the parameters of their metrics, and then these algorithms return clusters that satisfy the parameters. Hence, it is the user who determines the results, based upon his/her biased assumptions. Besides, these algorithms typically abound with parameters, thereby increasing the burden on the user. For example, due to the complex nature of 3D continuous-valued data, the pioneering work in 3D subspace clustering [1] requires a total of 7 parameter settings.

Recent work by Keogh et al. [9], promoting the concept of *parameter-free* or *parameter-light* data mining, declares:

"*A parameter-free algorithm prevents us from imposing our prejudices and presumptions on the problem at hand, and lets the data itself speak to us*".

Following this spirit, we should be interested in 3D subspace clusters that are intrinsically prominent in the data, in other words, the mining process should discover clusters that 'stand out' in the data, without requiring fine tuning of parameters.

This brings us to our second question: *"How to efficiently mine CSCs with high correlation information, and with minimum user interference?"* The ideal solution is to exhaustively mine all CSCs, and output those whose correlation information are high. But this approach is computationally infeasible, as even the simpler 2D subspace clustering is a NP-hard problem [10]. We propose a more pragmatic and efficient approach, by using pairs of values whose correlation information are significantly high as building blocks for CSCs. This approach has two main advantages: the CSCs are guaranteed to have high correlation information, and the search space for CSCs is tremendously reduced.

Next, we need to determine when a correlation information is considered to be significantly high. Explicitly setting a threshold is meaningless, as the user would not know the correct setting. We propose using the notion of rarity to measure significance: a high correlation information is significant when its occurrence is extremely rare, i.e., when its probability (denoted as *p-value*) is less than a threshold α. We show that our default setting of α works well in practice, and is insensitive to the input data.

Contributions In summary, we address the problem of mining CSCs, which are *significant* axis-parallel 3D subspace clusters, and make the following contributions:

- We present a novel information theoretic measure to quantify the correlation of CSCs (c.f., Section III).
- We develop a highly effective and efficient algorithm, which uses a hybrid of information theoretical and statistical methods to mine CSCs (c.f., Section IV).
- We empirically show the superiority of our approach over the state-of-the-art using a wide range of experiments (c.f., Section V). Our approach is better at finding significant 3D subspace clusters embedded in 110 synthetic datasets of varying conditions. We also provide a case study on real stock market datasets, which shows that CSCs generate higher profits than other clusters.

II. Related Work

Most of the subspace clustering algorithms find subspace clusters that fulfill certain distance or similarity-based functions [10]–[12], so that the members in each cluster display a certain degree of homogeneity. The user has to set parameter thresholds on these functions, but the optimal thresholds are generally unknown. Likewise in density-based subspace clustering [13], a global density threshold is required. Wang et al. [14] proposed k-subspace clustering, which replaces the thresholds requirement with specifying k number of subspace clusters needed, but the optimal k is unknown. The parameter-light algorithm of Moise and Sander [3] mines subspace clusters that are statistically significant but it only handle uniformly-distributed data. Sim et al. [4] proposed mining top-k multi-attribute co-clusters (MACs) from 2D data, which are highly correlated pairs of subspace clusters.

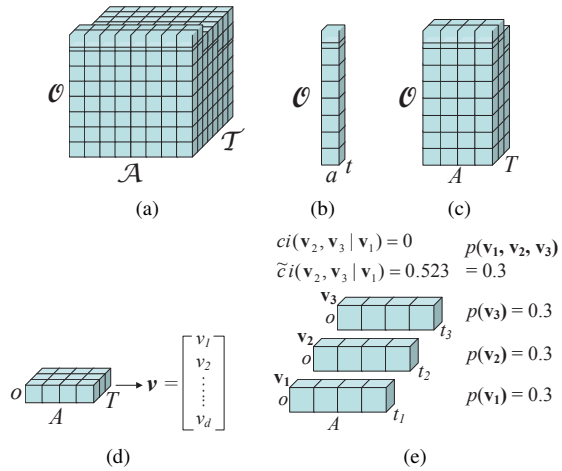


Figure 2. (a) Continuous-valued cuboid $\mathcal{D} = \mathcal{O} \times \mathcal{A} \times \mathcal{T}$. (b) A sub-cuboid: domain of attribute a at time t, $D(a,t) = \mathcal{O} \times \{a\} \times \{t\}$. (c) A sub-cuboid: domain of a set of attributes A at a set of timestamps T, $D(A,T) = \mathcal{O} \times A \times T$. (d) A slice of $D(A,T)$, $S = \{o\} \times A \times T$, mapped to a vector $\mathbf{v}$. (e) Example of $\widetilde{ci}(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1}) \geq ci(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1})$ (c.f., Section III-D).

In this paper, we extend its concept of using correlation information to mine CSCs. Note that the aforementioned methods are only applicable for 2D data, and not suitable for solving our problem. Furthermore, it is non-trivial to extend them to handle 3D subspace clusters. Jiang et al. [15] attempted this by transforming the 3D data in a 2D data and then mine 3D subspace clusters from the transformed data; but they consider the temporal/spatial dimension in full space, which means there is no concept of subspace in the temporal/spatial dimension.

Axis-parallel 3D subspace clusters are extensions of the 2D subspace clusters with time/location as the third dimension. Tricluster [1] is the pioneer work on 3D subspace clusters. Similar to 2D subspace clusters, triclusters fulfill certain similarity-based functions and thresholds have to be set on these functions. Xu et al. [16] also proposed a 3D subspace clustering model, S^2D^3 cluster, which also requires thresholds to be set on its parameters, but S^2D^3 clusters are not axis-parallel. Triclusters and S^2D^3 clusters also suffer from the parameter settings problem that their 2D peers have. Moreover, their problem is compounded by the 3D nature of the data; they have 7 and 5 parameters to set respectively. Our model does not suffer from the parameter setting problem, as it only has a single parameter, and the data is insensitive to this parameter. Ji et al. [17] and Cerf et al. [18] also proposed 3D subspace clusters which satisfy certain parameters' thresholds, but they can only handle simple binary data and cannot be extended to handle continuous-valued data. Even if we transform the data by discretization, selecting the appropriate discretization method is a hard problem, and the number of attributes may increase exponentially.

Other clustering techniques which handle high dimensional data have been proposed, such as co-clustering [19] and tensor clustering [20]. However, they partition the data into clusters, which is different from subspace clustering.

Parameter-free algorithms have been developed for traditional clustering [9], graph partitioning [21] and clusters refinement [22], but due to the complexity of mining 3D subspace clusters in 3D continuous-valued data, no parameter-free or parameter-light algorithm has yet been developed for this problem.

Sun et al. [23] proposed a global approach of projecting a data of M order (e.g. the data in cuboid is of 3^{rd} order) into a tensor of the same order, but of a much smaller magnitude. Its objective is to get a summary of the data, which is different from the task of subspace clustering. Jakulin and Bratko [24] proposed using mutual information to analyze if attributes 'interact' with each other, i.e. how correlated they are. Thus, theirs is a global approach of finding correlation between attributes, whereas ours is a local approach of finding correlations within a 3D subspace cluster.

III. Correlation Information

A. Preliminaries

Our data is a continuous-valued **cuboid** $\mathcal{D} = \mathcal{O} \times \mathcal{A} \times \mathcal{T}$, with objects $\mathcal{O}$, attributes $\mathcal{A}$ and timestamps $\mathcal{T}$ as its dimensions (Fig. 2(a)). We denote the value of object o on attribute a, and at time t as x_{oat}. Let $O \subseteq \mathcal{O}$, $A \subseteq \mathcal{A}$ and $T \subseteq \mathcal{T}$. We define **sub-cuboid** $C = O \times A \times T$ as a subset of cuboid $\mathcal{D}$.

The **domain** of attribute a at time t is a sub-cuboid denoted as $D(a,t) = \mathcal{O} \times \{a\} \times \{t\}$. The domain of a set of attributes A at a set of timestamps T is a sub-cuboid denoted as $D(A,T) = \mathcal{O} \times A \times T$. Examples of $D(a,t)$ and $D(A,T)$ are shown in Fig. 2(b) and Fig. 2(c) respectively. We define a **slice** as a sub-cuboid $S = \{o\} \times A \times T \in D(A,T)$ (Fig. 2(d)). We can also map a slice S to a column vector $\mathbf{v}$ as follows:

Definition 1: (Mapping of slice $S = \{o\} \times A \times T$ to column vector $\mathbf{v}$) Let slice S be represented as a partially ordered set $\{x_{oat} | a \in A, t \in T\}$ with cardinality d, and let $\mathbf{v} = (v_1, \ldots, v_d)^{\mathrm{T}}$ be a column vector of d values. We map S to $\mathbf{v}$ using function $\beta : S \to \mathbf{v} = x_{oat} \mapsto v_i (1 \leq i \leq d)$.

B. Estimation of Probability Density Function (pdf) using Kernel Density Estimation

In order to calculate correlation information, we need to first calculate the pdf and the probability of the values and vectors of the data.

Let us assume domain $D(a,t)$ as a continuous random variable with values x_{oat} and $f(x_{oat})$ as the pdf of domain $D(a,t)$. We use kernel density estimation to estimate pdf $f(x_{oat})$. Firstly, kernel density estimation is non-parametric, so there are no rigid assumptions on the distribution of the data. Secondly, amongst the non-parametric pdf estimators,

it gives the smoothest pdf estimate on continuous-valued data. Furthermore, Parzen [5] showed that it converges to the true pdf if the bandwidth of the kernel is properly selected.

Definition 2: (pdf estimate of domain $D(a,t)$)

$$\hat{f}(x_{oat}) = \frac{1}{|\mathcal{O}|h\sqrt{2\pi}} \sum_{x_{o'at} \in D(a,t)} exp\left(-\frac{1}{2}\left(\frac{x_{oat} - x_{o'at}}{h}\right)^2\right) \tag{1}$$

We use the Gaussian kernel but it is acceptable to use other kernels, as the difference in the pdf estimate by different kernels is negligible [25]. h is the bandwidth which determines the width of the kernel and we adopt Silverman's bandwidth formula [25] and set $h = 0.09m|\mathcal{O}|^{-1/5}$, where $m = min\{\sigma_{D(a,t)}, iqr_{D(a,t)}/1.34\}$. $\sigma_{D(a,t)}$ is the sample standard deviation of $D(a,t)$ and $iqr_{D(a,t)}$ is the interquartile range of $D(a,t)$. We set our bandwidth to be an order of magnitude smaller than Silverman's formula to increase the details of the pdf, so that regions of different densities in the pdf are more distinguishable. There is danger of undersmoothing the data, but in Section V, we show that it works well in practice.

Definition 3: (pdf estimate of domain $D(A,T)$)

$$\hat{f}(\mathbf{v}) = \frac{\frac{1}{|\mathcal{O}|h^d(2\pi)^{d/2}det(\mathcal{S})^{1/2}}}{\sum_{\mathbf{v}' \in D(A,T)} exp\left(-\frac{(\mathbf{v}-\mathbf{v}')^T\mathcal{S}^{-1}(\mathbf{v}-\mathbf{v}')}{2h^2}\right)} \tag{2}$$

$\mathcal{S}$ is the sample covariance matrix of the vector $\mathbf{v}$ of the domain $D(A,T)$, and $det(\mathcal{S})$ is the determinant of $\mathcal{S}$. We also use the Gaussian kernel in calculating $\hat{f}(\mathbf{v})$. We adopt Silverman's multivariate Gaussian kernel bandwidth formula [25] and set the bandwidth $h = 0.01\left(\frac{4}{d+2}\right)^{1/(d+4)}|\mathcal{O}|^{-1/(d+4)}$, which is also an order of magnitude smaller than Silverman's formula.

C. Estimation of Probability using Normalized pdf

Kwak and Choi [8] assumed the pdf estimate as the probability of the value or vector, which is a very poor approximation because it is possible that pdf is greater than one. A more accurate approximation is by integrating the pdf to obtain the area surrounding the value or hyperrectangle containing the vector, and the result is the probability[1]. The Newton-Cotes formulas [26] can be used to calculate areas, but they cannot be used to calculate hyperrectangles. Monte Carlo integration can be used on vectors, but the number of random vectors needed to calculate the hyperrectangle is exponentially large [7].

We propose using *normalized pdf*, which is more accurate than assuming pdf as probability and more efficient than Monte Carlo integration. *Normalized pdf* approximates the probability by calculating the pdf of the value or vector over the total pdf of all values or vectors in the domain. As such, the method intuitively follows the concept of probability [6].

Definition 4: (Probability of value $x_{oat} \in D(a,t)$)

$$\hat{p}(x_{oat}) = \frac{\hat{f}(x_{oat})}{\sum_{x_{o'at} \in D(a,t)} \hat{f}(x_{o'at})} \tag{3}$$

Definition 5: (Probability of vector $\mathbf{v} \in D(A,T)$)

$$\hat{p}(\mathbf{v}) = \frac{\hat{f}(\mathbf{v})}{\sum_{\mathbf{v}' \in D(A,T)} \hat{f}(\mathbf{v}')} \tag{4}$$

D. Correlation Information

Correlation information is derived from the generalization of mutual information [27], and is defined as follows.

Definition 6: (Correlation information between two vectors $\mathbf{v_2}$ and $\mathbf{v_3}$) Let $\mathbf{v_2} = (v_1, \ldots, v_d)^{\mathrm{T}} \in D(A, \{t_2\})$ and $\mathbf{v_3} = (w_1, \ldots, w_n)^{\mathrm{T}} \in D(A, \{t_3\})$.

$$ci(\mathbf{v_2}, \mathbf{v_3}) = \sum_{i=1}^{d}\sum_{j=1}^{n} \hat{p}(v_{1\ldots i}, w_{1\ldots j}) \log \frac{\hat{p}(v_i, w_j | v_{1\ldots i-1}, w_{1\ldots j-1})}{\hat{p}(v_i | v_{1\ldots i-1}, w_{1\ldots j-1})\hat{p}(w_j | v_{1\ldots i-1}, w_{1\ldots j-1})} \tag{5}$$

For brevity, we denote a sequence of values $v_1, \ldots, v_d$ as $v_{1\ldots d}$. Note that $\hat{p}(x|y)$ can be calculated by simply expanding $\hat{p}(x|y) = \frac{\hat{p}(x,y)}{\hat{p}(y)}$.

Equation 5 only considers the correlation information between a pair of vectors, and we need to extend it to calculate the correlation information of vectors between different timestamps. Let $\mathbf{v_1} = (u_1, \ldots, u_m)^{\mathrm{T}} \in D(A, \{t_1\})$. To consider the correlation information between $\mathbf{v_2}$ and $\mathbf{v_3}$, given prior knowledge of $\mathbf{v_1}$, we can extend Eq. 5 as:

$$\begin{aligned} ci(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1}) &= \sum_{i=1}^{d}\sum_{j=1}^{n} \hat{p}(v_{1\ldots i}, w_{1\ldots j}, u_{1\ldots m}) \\ &\log \frac{\hat{p}(v_i, w_j | v_{1\ldots i-1}, w_{1\ldots j-1}, u_{1\ldots m})}{\hat{p}(v_i | v_{1\ldots i-1}, w_{1\ldots j-1}, u_{1\ldots m})\hat{p}(w_j | v_{1\ldots i-1}, w_{1\ldots j-1}, u_{1\ldots m})} \\ &= \sum_{i=1}^{d}\sum_{j=1}^{n} \hat{p}(v_{1\ldots i}, w_{1\ldots j}, u_{1\ldots m}) \\ &\log \frac{\hat{p}(v_{1\ldots i}, w_{1\ldots j}, u_{1\ldots m})\hat{p}(u_{1\ldots m})}{\hat{p}(v_{1\ldots i}, w_{1\ldots j-1}, u_{1\ldots m})\hat{p}(v_{1\ldots i-1}, w_{1\ldots j}, u_{1\ldots m})} \end{aligned} \tag{6}$$

However, Eq. 6 is not exactly suitable for our problem. The original Eq. 5 measures correlation between a pair of vectors. Hence, using Eq. 6 will penalize sequences of vectors that are correlated across time. To remedy this problem, we amend Eq. 6 to get the following equation which promotes correlation across time.

Definition 7: (Adjusted correlation information between vectors $\mathbf{v_2}$ and $\mathbf{v_3}$, given $\mathbf{v_1}$)

$$\tilde{ci}(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1}) = \sum_{i=1}^{d}\sum_{j=1}^{n} \hat{p}(v_{1\ldots i}, w_{1\ldots j}, u_{1\ldots m}) \log \frac{p(\hat{v}_{1\ldots i}, w_{1\ldots j}, u_{1\ldots m})}{\hat{p}(v_{1\ldots i}, w_{1\ldots j-1}, u_{1\ldots m})\hat{p}(v_{1\ldots i-1}, w_{1\ldots j}, u_{1\ldots m})} \tag{7}$$

[1]The integration of a value or vector is zero, hence we integrate an area surrounding the value or hyperrectangle containing the vector.

Proposition 1: $\tilde{ci}(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1}) \geq ci(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1})$

Proof: Let a represents $\log \frac{\hat{p}(v_{1...i}, w_{1...j}, u_{1...m})}{\hat{p}(v_{1...i}, w_{1...j-1}, u_{1...m})\hat{p}(v_{1...i-1}, w_{1...j}, u_{1...m})}$ and b represents $\hat{p}(v_{1...i}, w_{1...j}, u_{1...m})$. For any two real numbers x and y, we know that $\log(x) < \log(y)$ if $0 < x < y$. Since $\hat{p}(u_{1...m}) \leq 1$, $\forall i \in \{1, \ldots, d\}$ and $\forall j \in \{1, \ldots, n\}, a.b \geq a.b.\hat{p}(u_{1...m})$. ■

According to Proposition 1, the prior knowledge of $\mathbf{v_1}$ penalizes $ci(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1})$ even if $\mathbf{v_1}$ is highly correlated to $\mathbf{v_2}$ and $\mathbf{v_3}$, due to the term $\hat{p}(u_{1...m})$. This penalty does not occur in $\tilde{ci}(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1})$ as this term is removed. We use Fig. 2(e) to illustrate our point, which shows 3 slices $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$, and $\hat{p}(\mathbf{v_1}) = \hat{p}(\mathbf{v_2}) = \hat{p}(\mathbf{v_3}) = \hat{p}(\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}) = 0.3$. Based on the pairwise correlation concept of Eq. 6, $ci(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1}) = 0$ because by having the prior information of $\mathbf{v_1}$, we know that the correlation between $\mathbf{v_2}$ and $\mathbf{v_3}$ is the same as the correlation between $\mathbf{v_1}$ and $\mathbf{v_2}$, so no new information is gained. However, based on the correlation across time concept of Eq. 7, $\tilde{ci}(\mathbf{v_2}, \mathbf{v_3}|\mathbf{v_1}) = 0.523$ because by having the prior information of $\mathbf{v_1}$, we know that $\mathbf{v_1}$, $\mathbf{v_2}$ and $\mathbf{v_3}$ are correlated, and this is new information gained.

E. Correlated 3D Subspace Cluster (CSC)

We denote $D_O(A, \{t\}) = O \times A \times \{t\}$ as the domain of the set of attributes A at time t, projected on the set of objects O.

Definition 8 (Correlated 3D subspace cluster (CSC)): Sub-cuboid $C = O \times A \times T$ is a CSC if the adjusted correlation information of C,

$$\tilde{ci}(C) = \sum_{i \in T} \sum_{\mathbf{v_1} \in D_O(A,\{1\}) \ldots \mathbf{v_i} \in D_O(A,\{i\})} \tilde{ci}(\mathbf{v_i}, \mathbf{v_{i-1}}|\mathbf{v_{1,\ldots,i-2}}) \tag{8}$$

is high.

Intuitively, a sub-cuboid C is a CSC if (1) for each time frame $O \times A \times \{t\}$ of C, its values are correlated and (2) for each pair of contiguous time frames $O \times A \times \{t\}$ and $O \times A \times \{t+1\}$ of C, they are correlated, given prior time frames. We empirically show that high $\tilde{ci}(C)$ leads to significant 3D subspace clusters. We embedded a significant 3D subspace cluster having 15 objects, 3 attributes and 5 timestamps in a synthetic 3D continuous-valued dataset of 100 objects, 10 attributes and 10 timestamps, and exhaustively mined all 3D subspace clusters and calculated their $\tilde{ci}(C)$. Figure 3(a) shows the results of using the mined clusters to recover the embedded cluster. The results are measured by significance, which is defined in Section V-A. We can see that mined clusters with high $\tilde{ci}(C)$ leads to exact discovery of the embedded cluster.

Details of determining how high $\tilde{ci}(C)$ is considered to be significant is explained in the next section. We only mine *maximal* CSCs to remove redundancies in the CSCs. A CSC $C = O \times A \times T$ is maximal when there does not exist another CSC $C' = O' \times A' \times T'$ such that $O \subseteq O'$, $A \subseteq A'$ and $T \subseteq T'$.

IV. Algorithm for Mining CSCs (MIC)

We present the algorithm MIC to mine CSCs from a continuous-valued cuboid $\mathcal{D}$, with its framework shown in Fig. 3(b). MIC consists of two parts:

1) *Generating seeds.* From $\mathcal{D}$, we obtain pairs of values (x_{oat}, x_{oat+1}) that have significantly high correlation information, which we denote as **seeds**.
2) *Mining CSCs.* The seeds are used as building blocks for CSCs.

A. Generating seeds

For each pair of values (x_{oat}, x_{oat+1}) given $o \in \mathcal{O}, a \in \mathcal{A}, t \in \mathcal{T}$, we calculate the adjusted correlation information $\tilde{ci}(x_{oat}, x_{oat+1})$. Although adjusted correlation information is for vectors, we can simply represent a pair of values as a pair of vectors (each vector contains a value), and use the same formula. For conciseness, we denote $\tilde{ci}(x_{oat}, x_{oat+1})$ as ci. Let the set of positive adjusted correlation information of pairs of values be denoted as $CI = \{ci | ci > 0, o \in \mathcal{O}, a \in \mathcal{A}, t \in \mathcal{T}\}$.

We propose that the significance of a seed is determined on the rarity of its correlation information, which can be calculated using statistics. Let us assume that we have the null hypothesis "A sample ci is equal to the mean of CI", and let the probability of having ci be *p-value*(ci), assuming the null hypothesis holds. Very low *p-value*(ci) means that it is very rare to have such seed with high correlation information, and we deem a pair of values to be a seed if its ci is statistically significant, i.e., *p-value*$(ci) \leq \alpha$, where α is a probability threshold. Therefore, the set of seeds is

$$seeds = \{(x_{oat}, x_{oat+1}) | \textit{p-value}(ci) \leq \alpha, o \in \mathcal{O}, a \in \mathcal{A}, t \in \mathcal{T}\}$$

We set a default setting of $\alpha = 1.0E-4$, which we shown in our experiments to work well in practice. However, the user can also set his preferred α.

We now explain how we derive *p-value*(ci). We first need to model the probability distribution of CI. Given that (1) values in CI are continuous, (2) they are positive and (3) probability distribution of CI is unknown and dependent on data $\mathcal{D}$, either gamma or Weibull distribution is a suitable candidate. Both offers the flexibility of modeling any continuous and positive distribution, as the scale and shape of the distribution can be adjusted by their two parameters [28]. Hence, using either one will result in obtaining the same quality of clusters, but we adopt the gamma distribution for computation efficiency's sake, as the convergences of the maximum likelihood estimation (MLE) of its parameters are much faster [29].

Let CI be gamma-distributed with parameters, shape k and scale θ, $CI \sim \Gamma(k, \theta)$. The pdf of the gamma distribution is $f(ci; k, \theta) = \frac{ci^{k-1}}{\Gamma(k)\theta^k} exp(-\frac{ci}{\theta})$, where $ci \in CI$ and $\Gamma(k) = \int_0^\infty t^{k-1}e^{-t}dt$ is the gamma function.

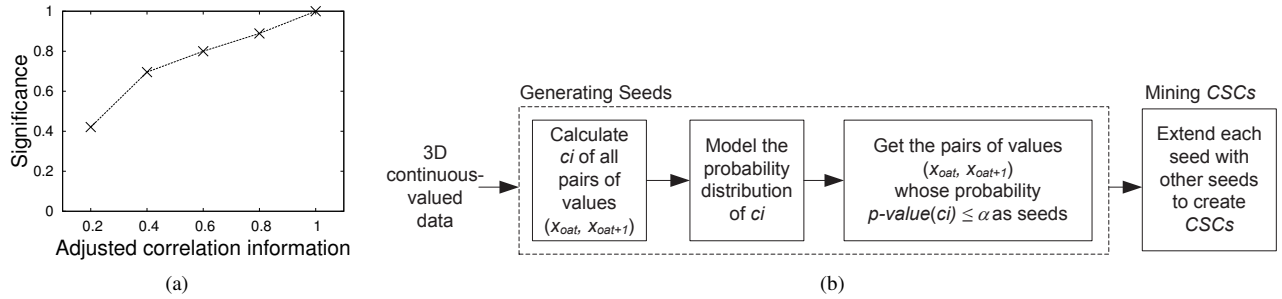


Figure 3. (a) Significance of the mined 3D subspace clusters with varying adjusted correlation information in recovering the embedded cluster. The x-axis is normalized from 0 to 1, (b) Framework of Algorithm MIC.

Algorithm 1 $growSeeds$

Input:
$seeds$ (building blocks for CSCs)
Output:
Maximal $CSCs$
Description:
1: prune $seeds$;
2: **for all** seed in $seeds$ **do**
3: initialize $seed$ as a cluster $C = O \times A \times T$;
4: **while** C can still be extended **do**
5: $ci_{att} \leftarrow \max_{a' \in \mathcal{A}/A} \tilde{ci}(extend(C, a'))$;
6: $t' \leftarrow$ the next time where $extend(C, t')$ is valid;
7: $ci_{time} \leftarrow \tilde{ci}(extend(C, t'))$;
8: **if** $ci_{att} > ci_{time}$ **then**
9: $C \leftarrow extend(C, a')$;
10: **else**
11: $C \leftarrow extend(C, t')$;
12: **end if**
13: **end while**
14: add pruned seeds to C; output C as a CSC;
15: **end for**

After obtaining the estimated parameters $\tilde{k}, \tilde{\theta}$ using MLE [30], we use gamma distribution to model CI. We then proceed to calculate *p-value*(ci) using the cumulative distribution function (cdf) of the gamma distribution

$$\textit{p-value}(ci) = \frac{1}{\Gamma(\tilde{k})\tilde{\theta}^{\tilde{k}}} \int_0^{ci} t^{\tilde{k}-1} e^{-t/\tilde{\theta}} dt \qquad (9)$$

which can be efficiently calculated using the Newton-Raphson method [26].

B. Mining CSCs

Algorithm 1 presents the function $growSeeds$, which uses the generated seeds as building blocks for CSCs. The general idea is that each seed is considered as an initial cluster, and we greedily 'grow' this initial cluster by extending it with other seeds. This growth is guided by maximizing the correlation information of the cluster. Figure 4 shows the workflow of function $growSeeds$. Although we use greedy-based approach, we show in Section V that the quality of our clusters is high across different experiments.

As we are dealing with continuous-valued data, it is possible to have seeds whose values are highly similar and are concentrated in certain value ranges. Hence, we remove these duplicates to improve the algorithm's efficiency. Let there be two values x and z in $\mathcal{D}$ such that $x \leq z$. They are denoted as *neighbors* if there does not exist another value y in $\mathcal{D}$ such that $x \leq y \leq z$. Let there be two seeds (x_{oat}, x_{oat+1}) and $(x_{o'at}, x_{o'at+1})$. They are considered as duplicates if and only if (1) x_{oat} and $x_{o'at}$ are neighbors and (2) x_{oat+1} and $x_{o'at+1}$ are neighbors. The seed with the lower correlation information will be pruned, but the pruned seeds are kept for later usage.

In line 3 of Algorithm 1, a seed (x_{oat}, x_{oat+1}) is initialized as a cluster $C = \{o\} \times \{a\} \times \{t, t+1\}$. Cluster C is iteratively extended either by attribute a or time t until no more extensions are valid. The choice is dependent on which extension gives the higher correlation information. The illustration of the extension is given in the middle diagram of Fig. 4.

In function $extend(C, a')$, cluster C is extended by a set of seeds whose values belong to attribute $a' \in \mathcal{A}/A$. These seeds are selected from $\{(x_{oa't}, x_{oa't+1}) | o \in \mathcal{O}, a' \in \mathcal{A}/A, t, t+1 \in T\}$, which we denote as *candidates*. An extension by a' is valid if, for each pair of timestamps $t, t+1 \in T$, C can be extended with the candidate that gives the largest increase in $\tilde{ci}(C)$. The extension by attribute a' is invalid if there is no increase of $\tilde{ci}(C)$ or there are no candidates in any pair of timestamps $t, t+1 \in T$. A new cluster is created when the extension is valid.

Similarly, in function $extend(C, t')$, cluster C is extended by a set of seeds whose values belong to time t', with the seeds selected from the candidates $\{(x_{oat'}, x_{oat'+1}) |\ o \in \mathcal{O}, a \in A\}$. An extension by t' is valid if, for each attribute $a \in A$, cluster C can be extended with the candidate that gives the largest increase in $\tilde{ci}(C)$. The extension by time t' is invalid if there is no increase of $\tilde{ci}(C)$ or there are no candidates of any attribute $a \in A$. A new cluster is created

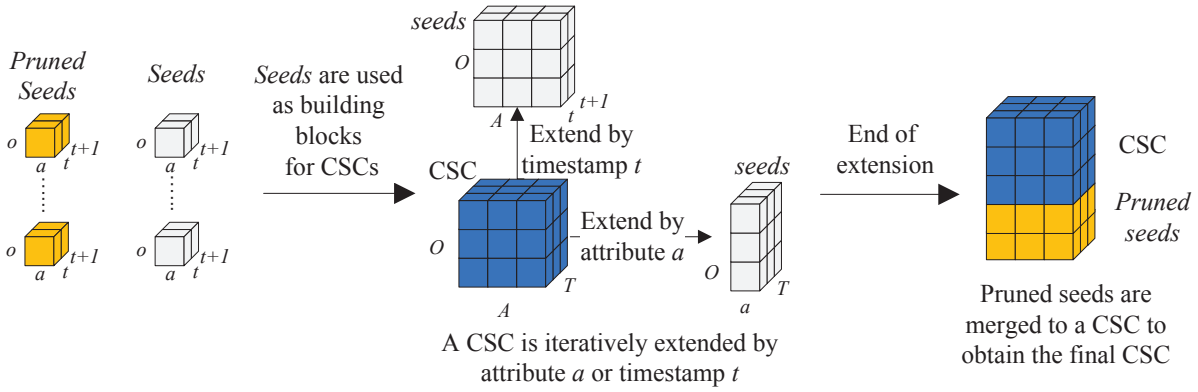


Figure 4. Framework of function $growSeeds$. Each seed is initialized as a CSC and is iteratively extended.

when the extension is valid. We select $t' \leftarrow t_{|T|-1}$ as the time for the extension and check if the extension is valid. If not, the time for extension will be iteratively incremented until there is a valid extension.

Note that there is no need to directly extend the set of objects of cluster C as the extension of the set of objects is induced during the extension by attribute or time. After cluster C is fully extended, we add seeds pruned earlier (Algo 1 line 1) to C to further extend it. Pruned seeds can be added to C if they are neighbors of the seeds used to build C. After all CSCs are mined, we do a simple post-processing to output the maximal CSCs.

C. Time Complexity of MIC

We discuss the worst case time complexity of MIC. Let n be the number of values in the cuboid $\mathcal{D}$. The generating seeds phase has a time complexity of $O(n^2)$ as we calculate the pairwise correlation information between values. Let s be the number of seeds generated. At the worst case, each seed is extended $s-1$ times, and the upper bound of the number of values involved in each extension is n. Hence the worst case time complexity of the *growSeed* is $O(ns^2)$. In total, the worst case complexity of MIC is $O(n^2+ns^2)$. Since the size of the data is not within our control, limiting the number of seeds can improve the efficiency of MIC, which is achievable by decreasing α.

V. Experiments & Analyses

We conducted three main experiments to validate our approach. First, we embedded significant 3D subspace clusters of various characteristics in a large number of synthetic datasets and checked if different algorithms are able to mine them. Second, we evaluated the efficiency and scalability of the algorithm MIC. Third, we performed a financial data mining case study by mining 3D subspace clusters from real-world *stock-financial ratio-year* datasets and investigated the usefulness of the different types of 3D subspace clusters.

All programs were coded in C++, and the codes for competing techniques were kindly provided by their respective authors. The experiments were performed in a Windows 7 environment, using an Intel Core 2 Quad 3.0 Ghz CPU with 8Gb RAM. As TRICLUSTER could only be run on Unix, we evaluated it using AMD Opteron, a powerful server with 1024 2.5 Ghz CPUs and 4096 Gb RAM.

A. Quality of 3D Subspace Clusters

We investigated the quality of the 3D subspace clusters mined by various algorithms. We created synthetic 3D continuous-valued datasets $\mathcal{D} = \mathcal{O} \times \mathcal{A} \times \mathcal{T}$, each having 1000 objects, 10 attributes and 10 timeframes, with values ranging from -1 to 1. In each dataset $\mathcal{D}$, we embedded 10 random 3D subspace clusters, each having 10–20 objects being similar in 2–4 attributes, across 4–6 timeframes. In each time of the cluster, we set a maximum difference (denoted as *diff*) between its objects' values on each of its attributes. This ensures that the cluster is homogeneous, and we varied *diff* from 0–0.1. In order to have a thorough and fair experiment, we created 10 datasets for each *diff* setting, resulting in a grand total of 110 synthetic datasets.

Let $\mathcal{C}^*$ be the set of embedded 3D subspace clusters, and $C^* = O^* \times A^* \times T^*$ be one such embedded cluster. Similarly, let $\mathcal{C}$ be the set of mined 3D subspace clusters, and $C = O \times A \times T$ be one such mined cluster. We use the following quality metrics to measure the closeness of $\mathcal{C}$ to $\mathcal{C}^*$ [31].

- *Recoverability*: $re(C^*) = \arg\max_{C \in \mathcal{C}} \sum_{t \in T^*} \frac{r(S^*)}{|S^*|}$, where $r(S^*) = max\{|S^* \cap S|$ such that $S^* = O^* \times A^* \times \{t\} \subset C^*, S = O \times A \times \{t\} \subset C\}$. *Recoverability* measures the ability of C to recover C^*.
- *Spuriousness*: $sp(C) = \arg\min_{C^* \in \mathcal{C}^*} \sum_{t \in T} \frac{s(S)}{|S|}$, where $s(S) = |S| - max\{|S^* \cap S|$ such that $S^* = O^* \times A^* \times \{t\} \subset C^*, S = O \times A \times \{t\} \subset C\}$. *Spuriousness* measures how spurious C is.
- *Significance* $= \frac{2Re(1-Sp)}{Re+(1-Sp)}$, where $Re = \sum_{C^* \in \mathcal{C}^*} re(C^*)$ and $Sp = \sum_{C \in \mathcal{C}} sp(C)$. *Significance* is a measure to find the best trade-off between *recoverability* and *spuriousness*. The higher the *Significance*, the more similar are the mined clusters to

the embedded clusters. In an ideal case, *Recoverability* is 1 and *Spuriousness* is 0.

We compared our algorithm MIC with parameter-laden 3D and 2D subspace clustering algorithms, TRICLUSTER [1] and MaxnCluster [12] (denoted as MNC in the graphs), and a parameter-light 2D subspace clustering algorithm STATPC [3]. For the 2D algorithms, we mine subspace clusters in each time frame, intersect all combinations of the subspace clusters to form 3D subspace clusters, and output 3D subspace clusters that are maximal.

The parameter-laden algorithms have two main types of parameters: minimum size of the clusters, and similarity functions. Varying all parameter settings (TRICLUSTER has 7 parameters, MaxnCluster has 4 parameters) for the experiments is practically impossible. Hence, we gave them an unfair advantage by letting them have the prior knowledge of the size of the embedded clusters and we just varied their similarity functions. For TRICLUSTER and MaxnCluster, we varied their similarity function parameters ϵ and δ from 0.05 to 0.15 respectively, and kept their other parameters at default settings. For STATPC, we used its default settings. For MIC, we set *p-value* at 1.0E-4 and 1.0E-5.

Figure 5 presents the average recoverability, spuriousness and significance of the algorithms on the 110 datasets across varying *diff*. We can see that MIC has the highest significance and recoverability across varying *diff*. Even with one order of magnitude difference in *p-value*, the results are still highly similar, which shows that MIC is parameter-insensitive, and fine tuning of *p-value* is not required. Although MIC does not have the prior knowledge of the actual size of the embedded clusters while MaxnCluster and TRICLUSTER have this advantageous knowledge, MIC is still able to outperform them. For MaxnCluster, its best results are depended on δ; for embedded clusters with higher *diff*, $\delta = 0.1$ gives the best results, and for embedded clusters with lower *diff*, $\delta = 0.05$ gives the best results. This shows that unless the user knows exactly what he wants or what are the actual clusters in the data (if this is the case, then there is no need for clustering), setting the right parameters to obtain the correct result is a guessing game for parameter-laden algorithms. The results of TRICLUSTER are not shown as it has scalability issues; it could not complete execution even after 24 hours on the powerful server. For STATPC, it could not find most of the embedded clusters.

B. Efficiency and Scalability Analysis

We analyzed the efficiency of MIC in two main areas: the size of the dataset $\mathcal{D}$ and the parameter *p-value*. Unless specifically stated, the default dataset $\mathcal{D}$ is a synthetic 3D continuous-valued data which contains 1000 objects, 10 attributes and 10 timestamps.

1) Size of the dataset $\mathcal{D}$: We investigated the scalability of MIC with respect to the number of objects, attributes and timestamps of dataset $\mathcal{D}$. We set the number of seeds to be 100 in this experiment (we will discuss how the efficiency of MIC is affected by the number of seeds in the next paragraph). Figure 6(a) presents the running times for varying number of attributes and timestamps in dataset $\mathcal{D}$, and Figure 6(b) presents the running time for varying number of objects in dataset $\mathcal{D}$. Although MIC uses computationally intensive techniques such as kernel density estimation and correlation information, it is still scalable when the attributes and timestamps are in tens and when the objects are in thousands. Hence MIC is more suitable for medium-sized data, such as the financial ratios data and the microarray (gene expression) data. In fact, MIC can complete the experiments described in Section V-A and V-C, whereas TRICLUSTER, the only other axis-parallel 3D subspace clustering algorithm, fails to do so.

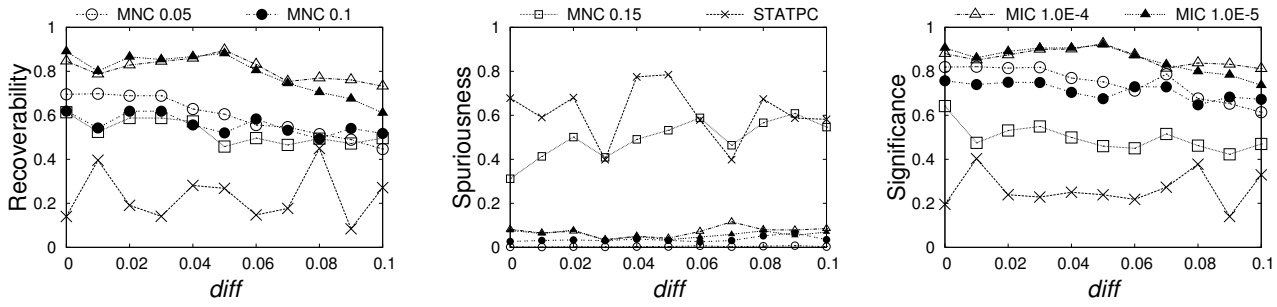
2) Parameter α: The *p-value* threshold parameter α determines the number of seeds generated in MIC – lower α results in lesser number of seeds. Figure 6(c) presents the running time for varying α. We can see that it is important to keep α small for a faster running time. Besides, keeping α small is also necessary for yielding CSCs with high correlation information.

C. Case Study on Stock Market Data

Fundamental investors analyze the *stock-financial ratio-time* data to pick stocks, as they believe that financial ratios are indicators of future stock price movements [32]. This relation between stock price and financial ratios can be studied by finding groups of high-performance stocks with similar financial ratios across years, which can be represented as 3D subspace clusters. In this experiment, we investigated the effectiveness of using significant 3D subspace clusters in stocks selection. We mined 3D subspace clusters from a training data of stocks having constant high growth in their prices, and in the testing data, we bought stocks that contain the financial ratios' values of these 3D subspace clusters. Then the price returns of these stocks are used to gauge the effectiveness of the 3D subspace clusters.

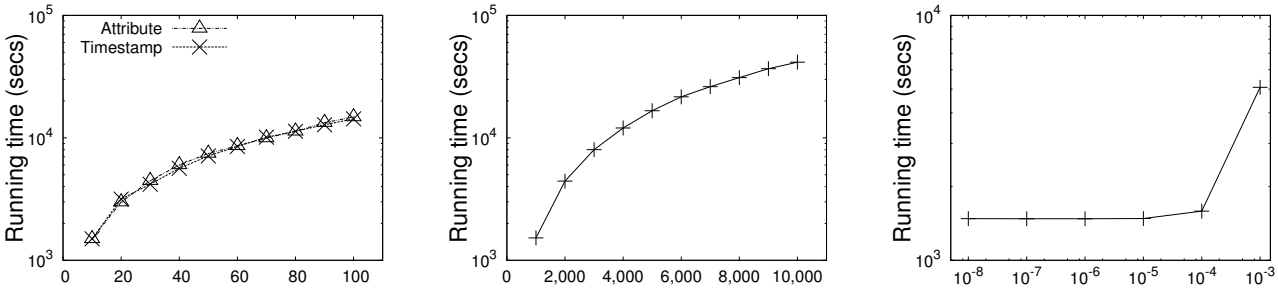
We downloaded financial figures of North American stocks from year 1980 to 2000, from Compustat [33]. We converted these financial figures into 30 financial ratios, based on the ratios' formula from Investopedia [34]. Stocks whose prices are less than USD$5 were removed from the data, as they are manipulative in nature and their financial figures are less transparent [35]. We used stocks with compound annual growth rate (CAGR) of at least 10% from year 1980 to 1989 as the training data. For the testing data, we used all stocks from year 1990 to 2000. Thus, the training data contains 231 stocks and the testing data contains 8406 stocks.

In the training phase, we set *p-value* to 10^{-4} and 10^{-5} for MIC. For STATPC, we used its default settings. For the parameter-laden algorithms, it is impossible to try all combinations of their parameters. For both TRICLUSTER



(a) (b) (c)

Figure 5. Quality of the 3D subspace clusters mined by different algorithms across 110 synthetic datasets. Each synthetic dataset is embedded with clusters having varying homogeneity. Although the parameter-laden algorithms are informed of the size of the embedded clusters prior to mining, MIC still outperforms them. The results of TRICLUSTER are not shown as it could not complete execution even after 24 hours.



(a) Varying number of attributes and timestamps (b) Varying number of objects (c) Varying α

Figure 6. The running time of MIC across different settings.

and MaxnCluster, we fixed the minimum attributes and timestamps to 2 and 4 respectively, varied minimum stocks from 50–100 with increments of 10, and varied the similarity setting from 0–0.15 in increments of 0.05. So, we had 24 settings for each algorithm.

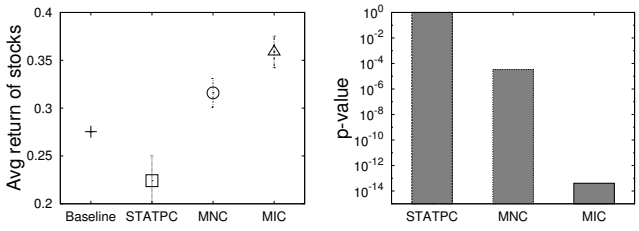
In the testing data from year 1990 to 2000, we bought a stock if more than one 3D subspace cluster covered it. Let us assume that a 3D subspace cluster has a set of years T. We deem that a 3D subspace cluster covers a stock if there exists a set of years T' for the stock, such that there is a one-to-one correspondence between T and T': $\forall k \in \{1, \ldots, n\}$, the financial ratios' values of the stock in year $j_k \in T'$ are in the financial ratios' ranges of the 3D subspace cluster in year $i_k \in T$. We bought the stock on the last year of T' and to evaluate the stock, we used the selling method proposed by Graham [36]. In this method, the stock is sold after two years, or if its price appreciates to 50% within two years. If we had bought all stocks in the testing data, we would have obtained an average return of 27.5%, which is the baseline of this experiment.

Figure 7(a) shows the average returns with 95% confidence intervals of the stocks, based on the different types of 3D subspace clusters, along with the baseline. Unfortunately, TRICLUSTER could not complete execution, even after 24 hours under any of the parameter settings. We can see that the average return of stocks by MIC are the highest, and even its 95% confidence interval is substantially higher than the baseline. For STATPC, its average return of stocks is much lower than the baseline. For MaxnCluster, its average return of stocks is above the baseline, but is lower than MIC.

Let us assume that we have the following null hypothesis *"The average return of a stock using Graham's selling method is 27.5%"*. Based on this null hypothesis, we calculate the *p-values* of the results shown in Fig. 7(a), and present them in Fig. 7(b). As the *p-values* of MIC and MaxnCluster are extremely small, we can conclude that their results are statistically significant. On the other hand, the result of STATPC is statistically insignificant. In summary, MIC is able to select stocks whose average return is statistically higher than the baseline and stocks selected by other algorithms.

VI. Conclusion

We proposed mining CSCs, which are 3D subspace clusters that are intrinsically prominent or significant in 3D continuous-valued data. They are clusters that stand out in the data, and not manifestations of the bias and prejudices of the user. We developed an algorithm MIC, which uses a hybrid of information theory and statistical techniques to mine CSCs. In certain situations where user's



(a) (b)

Figure 7. (a) Average returns with 95% confidence intervals of the stocks bought using different 3D subspace clusters, compared with the baseline, (b) p-values of the average returns.

presumption is needed on the CSCs mining, we allow the user to set a parameter which controls the number of CSCs to be mined. Its default setting is also shown to work well in practice. In our experiments, we showed that CSCs are significant 3D subspace clusters in a wide range of synthetic datasets and from real-world *stock-financial ratio-year* datasets, higher profits can be generated using CSCs, compared to 3D subspace clusters mined by other algorithms. We also showed that MIC is scalable to medium-sized data, which are common in financial and biological domains.

REFERENCES

[1] L. Zhao and M. J. Zaki, "TRICLUSTER: An effective algorithm for mining coherent clusters in 3D microarray data," in *SIGMOD*, 2005, pp. 694–705.

[2] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," in *SIGMOD*, 1997, pp. 255–264.

[3] G. Moise and J. Sander, "Finding non-redundant, statistically significant regions in high dimensional data: A novel approach to projected and subspace clustering," in *KDD*, 2008, pp. 533–541.

[4] K. Sim, V. Gopalkrishnan, H. N. Chua, and S.-K. Ng, "MACs: Multi-attribute co-clusters with high correlation information," in *ECML/PKDD (2)*, 2009, pp. 398–413.

[5] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Stat.*, vol. 33, no. 3, pp. 1065–1076, 1962.

[6] R. R. Yager, "On the instantiation of possibility distributions," *Fuzzy Sets and Systems*, vol. 128, no. 2, pp. 261–266, 2002.

[7] W. J. Morokoff, R. E. Caflisch, and O. Numbers, "Quasi-Monte Carlo integration," *J. Comput. Phy.*, vol. 122, pp. 218–230, 1995.

[8] N. Kwak and C.-H. Choi, "Input feature selection by mutual information based on Parzen window," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 12, pp. 1667–1671, 2002.

[9] E. J. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Towards parameter-free data mining," in *KDD*, 2004, pp. 206–215.

[10] Y. Cheng and G. M. Church, "Biclustering of expression data," in *ISMB*, 2000, pp. 93–103.

[11] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *SIGMOD*, 1998, pp. 94–105.

[12] G. Liu, K. Sim, J. Li, and L. Wong, "Efficient mining of distance-based subspace clusters," *Stat. Anal. Data Min.*, vol. 2, no. 5-6, pp. 427–444, 2009.

[13] P. Kröger, H.-P. Kriegel, and K. Kailing, "Density-connected subspace clustering for high-dimensional data," in *SDM*, 2004, pp. 246–257.

[14] D. Wang, C. H. Q. Ding, and T. Li, "K-subspace clustering," in *ECML/PKDD (2)*, 2009, pp. 506–521.

[15] D. Jiang, J. Pei, M. Ramanathan, C. Tang, and A. Zhang, "Mining coherent gene clusters from gene-sample-time microarray data," in *KDD*, 2004, pp. 430–439.

[16] X. Xu, Y. Lu, K.-L. Tan, and A. K. H. Tung, "Finding time-lagged 3D clusters," in *ICDE*, 2009, pp. 445–456.

[17] L. Ji, K.-L. Tan, and A. K. H. Tung, "Mining frequent closed cubes in 3D datasets," in *VLDB*, 2006, pp. 811–822.

[18] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut, "Data-Peeler: Constraint-based closed pattern mining in n-ary relations," in *SDM*, 2008, pp. 37–48.

[19] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *KDD*, 2003, pp. 89–98.

[20] H. Huang, C. Ding, D. Luo, and T. Li, "Simultaneous tensor subspace selection and clustering: The equivalence of high order SVD and k-means clustering," in *KDD*, 2008, pp. 327–335.

[21] D. Chakrabarti, "AutoPart: Parameter-free graph partitioning and outlier detection," in *PKDD*, 2004, pp. 112–124.

[22] C. Böhm, C. Faloutsos, J.-Y. Pan, and C. Plant, "Robust information-theoretic clustering," in *KDD*, 2006.

[23] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: Dynamic tensor analysis," in *KDD*, 2006, pp. 374–383.

[24] A. Jakulin and I. Bratko, "Analyzing attribute dependencies," in *PKDD*, 2003, pp. 229–240.

[25] B. W. Silverman, *Density Estimation for Statistics and Data Analysis (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*, 1st ed. Chapman and Hall/CRC, 1986.

[26] R. L. Burden and J. D. Faires, *Numerical analysis*, 9th ed. Cengage Learning, 2010.

[27] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 1991.

[28] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions, Vol. 1 (Wiley Series in Probability and Statistics)*, 2nd ed. Wiley-Interscience, 1994.

[29] L. J. Bain and M. Englehardt, *Statistical Analysis of Reliability and Life-testing Models*, 2nd ed. CRC-Press, 1991.

[30] T. P. Minka, "Estimating a Gamma distribution," 2002.

[31] R. Gupta, G. Fang, B. Field, M. Steinbach, and V. Kumar, "Quantitative evaluation of approximate frequent pattern mining algorithms," in *KDD*, 2008, pp. 301–309.

[32] B. Graham, *The Intelligent Investor: A Book of Practical Counsel*. Harper Collins Publishers, 1986.

[33] "Compustat," http://www.compustat.com [Last accessed 2009].

[34] "Investopedia," http://www.investopedia.com/university/ratios/ [Last accessed 2009].

[35] U.S. Securities and Exchange Commission, "Microcap stock: A guide for investors," http://www.sec.gov/investor/pubs/microcapstock.htm, 2009.

[36] H. R. Oppenheimer, "A test of Ben Graham's stock selection criteria," *Finan. Anal. J.*, vol. 40, no. 5, pp. 68–74, 1984.