

Mining Actionable Subspace Clusters in Sequential Data

Kelvin Sim*

Ardian Kristanto Poernomo[†]

Vivekanand Gopalkrishnan[‡]

Abstract

Extraction of knowledge from data and using it for decision making is vital in various real-world problems, particularly in the financial domain. We identify several financial problems, which require the mining of *actionable* subspaces defined by objects and attributes over a sequence of time. These subspaces are actionable in the sense that they have the ability to suggest profitable action for the decision-makers. We propose to mine *actionable subspace clusters* from sequential data, which are subspaces with high and correlated *utilities*. To efficiently mine them, we propose a framework MASC (Mining Actionable Subspace Clusters), which is a hybrid of numerical optimization, principal component analysis and frequent itemset mining. We conduct a wide range of experiments to demonstrate the actionability of the clusters and the robustness of our framework MASC. We show that our clustering results are not sensitive to the framework parameters and full recovery of embedded clusters in synthetic data is possible. In our case-study, we show that clusters with higher utilities correspond to higher actionability, and we are able to use our clusters to perform better than one of the most famous value investment strategies.

1 Introduction

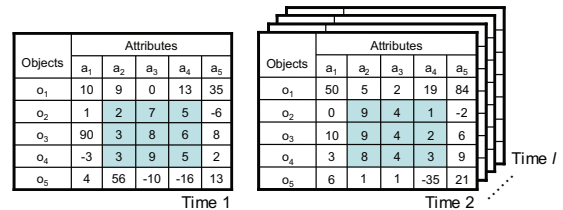
Clustering aims to find groups of similar objects and due to its usefulness, it is popular in a large variety of domains, such as astronomy, physics, geology, marketing, etc. Over the years, data gathering has become more effective and easier, resulting in many of these domains having high dimensional databases. As a consequence, the distance (difference) between *any* two objects becomes similar in high dimensional data, thus diluting the meaning of cluster [5]. One way to handle this issue is by clustering in subspaces of the dimension space, so that objects in a group need only be similar on some subset of attributes (subspace), instead of being similar across the entire set of attributes (full-space) [20].

Besides being high-dimensional, the databases in these

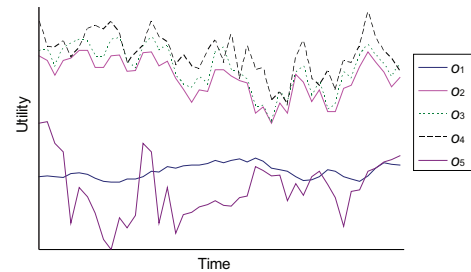
*Institute for Infocomm Research, A*STAR, Singapore. Email: shsim@i2r.a-star.edu.sg

[†]School of Computer Engineering, Nanyang Technological University, Singapore. Email: ardi0002@ntu.edu.sg

[‡]School of Computer Engineering, Nanyang Technological University, Singapore. Email: asvivek@ntu.edu.sg



(a) Objects o_2, o_3, o_4 and attributes a_2, a_3, a_4 form a 3D subspace cluster.



(b) Objects o_2, o_3, o_4 have high utility and they are correlated across time.

Figure 1: An example of an actionable subspace cluster.

domains also potentially change over time. In such sequential databases, finding subspace clusters per timestamp may produce a lot of spurious and arbitrary clusters, hence it is desirable to find clusters that persist in the database over some given period.

Moreover, the usefulness of these clusters, and in general of any mined patterns, lies in their ability to suggest concrete and useful actions. Such patterns are called *actionable* patterns [19], and they are normally associated with the amount of profit that their suggested actions bring [19, 30, 31].

In this paper, we identify real-world problems, particularly in the financial world, which motivates the need to infuse subspace clustering with actionability, for example:

Example 1 Value investors scrutinize fundamentals or financial ratios of companies, in belief that they are crucial indicators of their future stock price movements [7, 13, 14]. Although experts like Benjamin Graham [13] have suggested

desirable ranges of financial ratios, there is no concrete evidence to prove their accuracy, and so the goodness of financial ratios has remained subjective. Hence, finding good stocks via financial ratios remains the ‘holy grail’ for value investors. By grouping stocks based on their financial ratios, investors can study the association between financial ratios and high returns of stock.

Example 2 Financial analysts study financial ratios to forecast the profits of companies [22], or to predict the bankruptcy of companies [2]. Again, by grouping companies based on their financial ratios, analysts can study the association between financial ratios and profits of companies.

These two examples motivate the need to find actionable groups of stocks/companies that suggest high returns/profits, and to substantiate their actionability, these groups should be homogeneous and correlated across time. We model this problem as mining *actionable subspace clusters*, where the actions are determined by indicators such as stock price returns or profits of companies. We denote such indicators as the *utility* of the data.

Naturally, an ideal actionable subspace cluster should have the following properties:

1. its objects have high utility, so that the action suggested by the cluster is profitable or beneficial to the user.
2. its objects exhibit a significant degree of homogeneity, i.e., they are similar in some aspects across time.
3. the utilities of the objects are correlated to each other, so that these objects with homogeneous values and high utility do not occur together by chance.

In other words, we desire a cluster to be *actionable* (point 1) and *robust* (points 2 and 3). Figure 1 shows an example of an actionable subspace cluster.

1.1 Motivation for a New Approach While this problem is interesting and important, no existing clustering techniques are feasible to solve this problem. One possible way is to consider the timestamps as the third dimension, and then find 3D-clusters [34] in this space. To recall, 3D-clustering finds a set of objects whose data are similar in several attributes, and in several time frames. We note several limitations of this approach. First, it requires the objects to have similar values across time. This requirement is too strict, for example, the stock prices always change over time, and hence cannot be clustered by this approach. Second, this approach might find clusters that appear only in very few (and possibly far away) timestamps. Such clusters are likely to occur by chance, and hence cannot be trusted. Furthermore, this approach is very sensitive to the parameter (threshold) settings, which reduces its applicability.

Another way is to find all subspace clusters in each timestamp, and then build linkages of subspace clusters across timestamps. This approach suffers from scalability and quality issues. In one timestamp, there can be millions of potential subspace clusters, since there is no requirement of correlation over time. Moreover, it is vague on how to draw a linkage between clusters (among the potentially millions of clusters) to obtain good quality actionable subspace clusters.

To the best of our knowledge, this paper is the first that merges the concept of subspace clustering and actionability in sequential databases.

1.2 Proposed Method Many previous mining algorithms focus on the criteria of good clusters (patterns), and then set some thresholds to ensure the quality of mined patterns. Besides the need to set parameters, these approaches also suffer from sensitivity of the parameters.

Instead of setting thresholds, one alternative is to form an objective function to measure the goodness of clusters, and then find the patterns that maximize this objective function. This approach has several benefits. First, it is more robust to noise (random perturbation), since small changes should not drastically reduce the goodness of clusters. Second, optimization techniques have been studied quite extensively in the machine learning community, and there are efficient solutions for many classes of optimization problems. And finally, such approaches are usually less sensitive to the input parameters.

For our problem, we define an objective function for calculating the goodness (similarity, correlation, and utility) of a set of objects in an attribute, across all time frames. This objective function can be optimized efficiently and elegantly by transforming it into an *augmented Lagrange equation*. The solution of this optimization problem projects the original sequential (and actionable) database into a standard relational database, where the goodness of each object on an attribute is represented by a value. Having obtained this familiar representation, we can now choose from many existing algorithms to find subspace clusters. In this paper, we binarize the transformed data, and mine closed itemsets on the binary data to achieve our goal.

1.3 Contributions Summarizing our contributions, we:

- formalize the novel and useful problem of actionable subspace clustering in sequential databases, which cannot be solved with existing techniques.
- propose a highly effective and efficient algorithm for this problem, based on a hybrid of numerical optimization, principal component analysis and frequent itemset mining.
- empirically show the superiority of our techniques with an extensive range of experiments. In synthetic

datasets, we show that our approach is able to find the exact clusters under various conditions efficiently, where previous techniques fail. In a real stock market dataset, we show that the mined actionable subspace clusters generate higher profits than one of the most famous value investment strategies [27].

1.4 Organization The rest of the paper is organized as follows. Section 2 presents the related work, Section 3 presents the preliminaries and problem formulation. Section 4 presents the algorithm. Section 5 presents the experimentation results and Section 6 presents the conclusion.

2 Related Work

There is a wide range of subspace clustering works, and Kriegel et al. gave an excellent survey in [20]. In this section, we focus on subspace clustering works that are more related to the real-world problems that we described in Section 1. The problems that we posed requires axis-parallel subspace clustering, and there are several axis-parallel subspace clustering algorithms [10, 21, 23, 24, 34], each has its own definitions of how to define subspace clusters, where some homogeneity criteria are fulfilled.

In pattern-based subspace clustering [10, 23], the values in the subspace clusters satisfy some distance or similarity based functions, and these functions normally require some thresholds to set. However, setting the correct thresholds to obtain significant subspace clusters from real-world data is generally a guessing game, and these subspace clusters are usually sensitive to these thresholds, i.e., the thresholds determine the results. Similarly the density-based subspace clustering [21] also requires a global density threshold which is generally hard to set. In our work, we model the subspace clustering problem into a numerical optimization problem, which the sensitivity problem of thresholds is mitigated as the clustering results are not sensitive to the optimization parameters.

In STATPC [24], statistically significant subspace clusters which satisfy statistical significant thresholds are mined. Thus, the sensitivity problem of thresholds are removed as the clustering results are not sensitive to the statistical significant thresholds.

In [10, 21, 23, 24], their work focus on subspace clustering on two dimensional dataset, and thus is not suitable for subspace clustering on three dimensional dataset. Moreover, the utilities of the objects to be clustered are not considered to make the subspace clusters actionable.

There are subspace clustering algorithms [8, 17, 32, 34] which handles three dimensional (3D) dataset, but similar to the 2D subspace clustering algorithms, none of them incorporates the concept of actionability in their clustering. CubeMiner [17] and DATA-PEELER [8] only mine from 3D binary dataset, where the subspace clusters can be view as

cuboids containing value ‘1’. TRICLUSTER [34] and LagMiner [32] are able to mine 3D dataset containing quantitative data, but the clusters mined by LagMiner are not axis-parallel. TRICLUSTER can be used for the real-world problems that we posed, but similar to 2D distance and similarity based subspace clustering algorithms, its clustering results is sensitive to its threshold settings.

Actionable patterns [19, 30, 31] have been proposed, but they cannot be applied in the real-world problems identified by us. Firstly, the datasets of these real-world problems are sequential. Actionable patterns are developed to mine from a time frame of the data. Secondly, these datasets are quantitative data, and actionable patterns cater only to nominal data. Thirdly, it is not possible to extend actionable patterns to actionable subspaces on sequential data, as actionable subspaces may contain values that evolve across time.

A similar area related to actionable patterns is constrained clustering [3, 9, 33], where the clustering is changed into a semi-supervised process, due to the additional knowledge of object labels (which are used to improve the clustering result). However, constrained clustering algorithms focus on partitioning of objects into separate groups, which does not follow the principle of subspace clustering – objects can be in multiple groups due to them being relevant and significant in subspaces of the dataset. In addition, current constraints in constrained clustering are simple constraints indicating if an object should be clustered together with another object. In this paper, we can see that utilities dictate the clustering in a semi-supervised manner, and since utilities are quantitative, existing constrained clustering algorithms are not suitable for our problems.

3 Preliminaries and Problem Formulation

We present the basic definitions, and the problem formulation for finding actionable subspace clusters in a sequential database.

We deal with a sequential data \mathcal{D} , containing objects \mathcal{O} and attributes \mathcal{A} , across timestamps \mathcal{T} . Our objective is to discover a group of objects which are similar, correlated, and have high utility.

We first define the *similarity* between two objects. Let the value of object o on attribute a in time t be denoted as $v_a^t(o)$. We measure the *distance* between two objects o_1 and o_2 based on attribute a , $dist_a(o_1, o_2)$, as their Euclidean distance across all timestamps, which is formally given as:

$$(3.1) \quad dist_a(o_1, o_2) = \sqrt{\sum_{t \in \mathcal{T}} (v_a^t(o_1) - v_a^t(o_2))^2}$$

With this measure, two objects are considered close if their values are similar in most of the timestamps. Note that we do not measure the similarity or the trend across timestamps.

The similarity between two objects can be defined as a function inversely-proportional to the distance measure. Here, we measure the similarity between two objects o_1 and o_2 on attribute a , using the Gaussian function, which is given as:

$$(3.2) \quad s_a(o_1, o_2) = \exp\left(-\frac{\text{dist}_a(o_1, o_2)}{2\sigma_{o_1}^2}\right)$$

where σ_{o_1} is a parameter which controls the width of the Gaussian function, centered at object o_1 .

Note that our similarity function is not symmetric, i.e., $s_a(o_1, o_2) \neq s_a(o_2, o_1)$, since it is calculated based on the distribution of objects centered at the former object.

The width of the Gaussian function is estimated using k -nearest neighbors heuristic [25] as:

$$(3.3) \quad \sigma_o = \frac{1}{k} \sum_{n \in \text{Neigh}_a(o)} \text{dist}_a(o, n)$$

where $\text{Neigh}_a(o)$ is the set of k -nearest neighbors of object o on attribute a . The k -nearest neighbors is obtained by using Equation 3.2.

The k -nearest neighbors heuristic adapts the width of the Gaussian function accordingly to the distribution of the objects projected in the data space of attribute a , thus it is more robust than setting σ to a constant value. In our experiments, we set the default value of k as 10, and show that this default setting works well in practice. Moreover, the results are not sensitive to various values of k .

Next, we define the quality of a cluster. Let $u^t(o)$ be the *utility* of object o at time t . We assume that the utility of an object measures the *quality* of the object; the higher the utility, the higher the quality of the object. The utility of object o over time is denoted as $\text{util}(o)$. In this paper, we simply use $\text{util}(o)$ as the average utility, given as:

$$(3.4) \quad \text{util}(o) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} u^t(o).$$

Note that our framework can also be adapted to other utility function, such as compound annual growth rate (CAGR).

We also require all objects in a cluster to behave similarly across time. The correlation between two objects are measured using the statistical correlation measure, which is the proportion between their covariance and the individual standard deviation. The standard deviation of the utility of object o is calculated as:

$$(3.5) \quad \sigma(o) = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} (u^t(o) - \bar{u}(o))^2}$$

and the covariance between two objects o_1, o_2 is calculated as:

$$(3.6) \quad \text{cov}(o_1, o_2) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} (u^t(o_1) - \bar{u}(o_1))(u^t(o_2) - \bar{u}(o_2)),$$

where $\bar{u}(o)$ represents the average utility across all times, i.e., $\bar{u}(o) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} (u^t(o))$.

The correlation between o_1 and o_2 is then calculated as:

$$(3.7) \quad \rho(o_1, o_2) = \frac{\text{cov}(o_1, o_2)}{\sigma(o_1)\sigma(o_2)}.$$

Lastly, we define the structure of the clusters. We define a cluster as a matrix $(O \times A)$ where $O \subseteq \mathcal{O}$ and $A \subseteq \mathcal{A}$. As mentioned before, all objects in a cluster should be *similar*, have high utility, and have correlated utility. Having these requirements, the actionable subspace cluster can be defined as:

DEFINITION 3.1. [ACTIONABLE SUBSPACE CLUSTER]. A matrix $O \times A$ is an actionable subspace cluster if all objects in O are similar on all attributes in A , have high utility, and have correlated utility.

We do not set thresholds to explicitly define the goodness of the objects' similarity and correlation. Instead, our framework forms an objective function to measure their goodness, and find clusters that maximize this objective function. On defining the goodness of high utility, we shall explain it in details in Section 4.1.

To remove redundancies in the clusters, we only mine *maximal* actionable subspace clusters. An actionable cluster $(O \times A)$ is maximal if there is no other actionable subspace cluster $(O' \times A')$ such that $O \subseteq O'$ and $A \subseteq A'$. As we always mine clusters that are maximal, for brevity, we simply denote them as actionable subspace clusters.

Having all aforementioned notations, we can formally define the actionable subspace cluster mining problem as:

DEFINITION 3.2. [ACTIONABLE SUBSPACE CLUSTERS MINING PROBLEM]. Given a database D , we find all actionable subspace clusters $(O \times A)$.

4 MASC Algorithm

4.1 Framework Our framework is illustrated in Figure 2, which consists of two modules.

1. *Projection into Standard Relational Database.* The actionable and sequential database is projected into a standard relational database, based on a chosen cluster center c . Note that the projection is per centroid, i.e., we will have one relational database for each cluster center. In our experiment, we choose the centroids to be objects with utility higher than a parameter util_{\min} . In practice, the domain expert might also select the centroids based on their internal knowledge.

The projection is done by setting up an objective function that incorporates the utility, similarity, and correlation of objects. We show that this function can be modeled as an *augmented Lagrangian* equation,

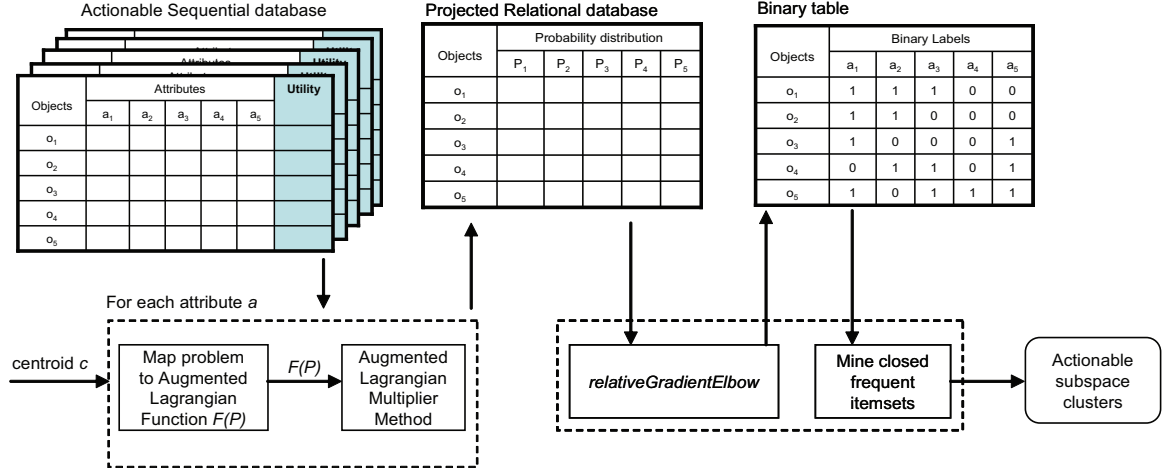


Figure 2: The MASC framework.

which can be optimized efficiently and elegantly using *Bound-Constrained Lagrangian Method (BLCM)* algorithm [26].

2. *Subspace Clustering.* After projecting into a standard relational database, we mine subspace clusters from that projected database. This can be solved with any subspace clustering algorithm for numerical database. However, since we need to run the algorithm per centroid, we opt for a highly efficient algorithm, which binarizes the data, and then mines frequent itemsets on that database [1]. Note that if efficiency is not of concern, we can use the more advanced techniques to obtain better clusters. Nonetheless, even with this simple method, we can find highly profitable objects/clusters, as shown in our empirical evaluation.

4.2 Projection into Standard Relational Database We project each attribute of the database independently. Within one attribute, we model the goodness of objects using probabilistic model. That is, the better objects should have higher probability to be selected in the subspace cluster. The resulting database is a relational database $\mathcal{D}' = \mathcal{O} \times \mathcal{A}$, where the content of cell $D'(o, a)$ is the probability of objects o to be part of the cluster, specified by attribute a .

As this module considers the goodness of objects based on a single attribute a , the notation a can be omitted throughout this section. Let p_o be the probability of object o to be part of the cluster, i.e. $p_o = D'(o, a)$ and let $P = (p_{o_1}, p_{o_2}, \dots, p_{o_{|\mathcal{O}|}})$.

We formulate this problem into the following objective function which we want to maximize.

$$(4.8) \quad f(P) = f^{util}(P) \cdot f^{corr}(P)$$

where

$$f^{util}(P) = \sum_{o \in \mathcal{O}} p_o s(c, o) util(o) \rho(c, o)$$

and

$$f^{corr}(P) = \sum_{o, o' \in \mathcal{O} \times \mathcal{O} | o \neq o'} p_o p_{o'} s(c, o) s(c, o') \rho(o, o'),$$

under constraint

$$\sum_{o \in \mathcal{O}} p_o = 1.$$

For a function f to be maximized, high probability should be assigned to objects which are similar to c , have high utilities, and their utilities are highly correlated to each other.

Let us analyze function f in detail. The first part, f^{util} , considers the utility of each object, and its similarity to the centroid. Function f^{util} will be maximized if the weights of the objects are adapted to maximize these three conditions. In other words, an object which is highly similar to c , has high utility and its utility is also correlated to utility of centroid c , will be assigned a high weight.

The second part, f^{corr} , measures the correlation between objects, and also the similarity to the centroid. Similarly, the score of this part will be maximized if the weights of the objects are adapted to maximize these two conditions. The second part of the function ensures that emphasis is placed on objects having correlated utilities and at the same time, they are similar with respect to centroid c .

Function $f(P)$ can be solved using the *augmented Lagrangian multiplier method*, where it is modeled as *augmented Lagrangian* equation as follows. First, we model the

Algorithm 1 *BCLM*

Input: $\delta, \lambda, \tau, \mu$ **Output:**The optimal probability distribution P **Description:**

```
1: initialize  $P^0, \lambda$ 
2:  $i \leftarrow 1$ 
3: while true do
4:    $P^i \leftarrow L\text{-BFGS}(P^{i-1}, \mu, \lambda)$ 
5:   if  $|P^i - P^{i-1}| < \delta$  then return  $P^i$ 
6:   if  $P^i$  is an improvement of  $P^{i-1}$  then
7:      $\tau \leftarrow \tau \cdot 0.1$  //strictly tighten  $\tau$ 
8:   else
9:      $\tau \leftarrow \tau \cdot 0.9$  //loosely tighten  $\tau$ 
10:   $\mu \leftarrow \mu \cdot 10$  //increase penalty violation
11:  update  $\lambda$ 
12:   $i \leftarrow i + 1$ 
```

constraint using another function $g(P)$ defined as:

$$g(P) = \sum_{o \in \mathcal{O}} p_o - 1 = 0.$$

The augmented Lagrangian function $F(P)$ is then given as:

$$F(P) = -f(P) - \lambda g(P) + \frac{\mu}{2} g(P)^2$$

In brief, the first term $-f(P)$ represents the function we want to minimize, and $g(P)$ represents the constraint. Another requirement for augmented Lagrange equation to be used is that both $f(P)$ and $g(P)$ must be smooth, which is clearly satisfied in our case. More details on optimizing augmented Lagrange equation can be referred to [26].

Algorithm 1 presents the augmented Lagrangian multiplier method, known as *Bound-Constrained Lagrangian Method (BCLM)* algorithm. The augmented Lagrangian multiplier method exploits the smoothness of both $f(P)$ and $g(P)$ and replace our constrained optimization problem by iterations of unconstrained optimization subproblems. In each iteration (Line 4), the probability distribution P is generated by using an unconstrained optimization algorithm which attempts to minimize $F(P)$. We use *L-BFGS* algorithm as the unconstrained optimization algorithm, which is proven to be efficient for problems with a large number of objects [26].

Algorithm *L-BFGS* uses P^{i-1} as the input and generates P^i when the Euclidean norm of the gradient of $F(P)$, $\|\nabla F(P)\|$, is not greater than the error allowance τ ; this means that P^i is an approximate optimal of $F(P)$. The gradient of $F(P)$ is expressed as $\nabla F(P) = \{\frac{\partial F(P)}{\partial p_o} | o \in \mathcal{O}\}$, and its complete expression is given in the Appendix.

BCLM algorithm requires four parameters, δ, λ, τ , and μ . Note that in most cases, the results are not sensitive to

Algorithm 2 *relativeGradientElbow*

Input:Probability distribution P **Output:**Set of objects with high probability distribution, S **Description:**

```
1: sort all objects  $o \in \mathcal{O}$  in descending order, based on  $p_o$ ;
2: calculate the relative gradient,  $\nabla_i \leftarrow \frac{p_{o_i} - p_{o_{i+1}}}{p_{o_i}}$  for all  $i \in [1, |\mathcal{O}| - 1]$ ;
3:  $i^* \leftarrow \arg \max(\nabla_i)$ ;
4:  $S \leftarrow \{o_i | i \leq i^*\}$ ;
5: return  $S$ ;
```

these parameters, and hence they can be set to their default value. Parameter δ specifies the closeness of the result to the optimal solution. Therefore, δ provides the usual trade-off between accuracy and efficiency, i.e., smaller δ implies longer computation time but better result. Parameter τ controls the tolerance level of violation to the constraint $g(P)$. Parameter μ specifies the severity of the penalty on $F(P)$, when the constraint is violated. And lastly, parameter λ is the Lagrange multiplier which is updated with $\lambda^i \leftarrow \lambda^i - \mu \cdot g(P^i)$ (Line 11). Details of λ is in [26].

The default parameters we use are as follows. We initialize the probability distribution P^1 by allocating equal probability to each object, that is $\frac{1}{|\mathcal{O}|}$. Parameters τ and μ are set to 1 and 10 respectively, as recommended in [26]. Parameter λ is set to 0.1 and parameter δ is set to 0.001. In our experiments, we show that the default settings work well in practice, and the results are not sensitive to various settings of the parameters.

4.3 Subspace Clustering Having the projected relational database, the remaining problem is to mine subspace clusters on this database. As our projection is per centroid, we need to run this algorithm once for every chosen centroid. Therefore, we choose a simple algorithm which is highly efficient. In a general level, our approach *binarizes* the database, and then mines closed frequent itemsets on the binary database.

4.3.1 Database Binarization As we project each attribute independently, the binarization is done per attribute as well. It is intuitive to select the discretized objects with high probability as one, and the ones with low probability as zero. The problem is to find the appropriate threshold. There are many possible ways to select the best objects, for example, we can simply select the top- k , or all objects with probability greater than a threshold. However, there are two subtle limitations of these approaches. First, these approaches need a user-defined parameter which is hard to set. Second, the difference between selected and non-selected objects can be

minimal, if they are very close to the threshold.

We relate this problem to the one of selecting the principal components for principal component analysis (PCA), and then propose *relativeGradientElbow* algorithm, which is based on *eigenvalue elbow* algorithm [18]. This algorithm is presented in Algorithm 2. In brief, this algorithm chooses the cutoff as the one with the steepest value drop. With this criteria, we ensure that the boundary between selected and non-selected objects is the largest possible.

4.3.2 Closed Frequent Itemset Mining A binary database can be treated as a transactional database, where each transaction represents objects, and each item represents attribute. A transaction t contains item i if its representative object o has value ‘1’ on attribute a .

Recall that the value ‘1’ represents object appearing in the cluster specified by individual attribute, the notion of maximal subspace clusters is equivalent to the closed-frequent-itemset (CFI) of the transactional database. An itemset is closed if it has no superset with the same support.

There are many algorithms for mining CFIs [4]. In our experiment, we used LCM algorithm [28] which is the state-of-the-art algorithm for this task.

5 Experimentation Results

We evaluated three main aspects of our approach using synthetic datasets: (1) cluster quality (including a comparison with TRICLUSTER [34], LCM-nCluster [23] and STATPC [24]), (2) parameter sensitivity, and (3) efficiency and scalability. A synthetic dataset \mathcal{D} contains 1000 objects, each with 10 attributes across 10 time frames. The attribute values of the objects range from 0 to 1, and their utilities range from -1 to 1. In each dataset, we embedded a set of 10 random subspace clusters, each with 3-10 objects being similar in 2-9 attributes. By default, we set the utility of each object ($util(o)$) and the correlation between each pair of objects o_1 and o_2 in each embedded cluster ($\rho(o_1, o_2)$) to be at least 0.5. To ensure the objects within a cluster are homogeneous, we also set the maximum difference between objects’ values on every attribute of the subspace, denoted as *diff*, to 0.1. These values hold for all experiments, unless explicitly changed.

We also performed an extensive case study on real stock market data to show the actionability of the resultant clusters, and compare them with the criteria provided by Graham. Our case study also shows the practical usage of cluster definitions for value investors.

All approaches were coded in C++, and code for competing approaches was kindly provided by their respective authors. The experiments were performed on a Windows Vista environment, using a Intel Core2 Dual 2.6 GHz CPU with 4GB RAM, except those involving algorithm TRICLUSTER, which were performed on a server with Linux

environment using 4-way Intel Xeon with 8GB RAM¹. We used the code by [6] for algorithm *L-BFGS*. Table 1 summarizes the parameter settings used in all our experiments.

5.1 Quality Evaluation In this section, we investigate the ability of different algorithms to mine actionable subspace clusters of different characteristics. While MASC and TRICLUSTER can be directly used, we need to extend *LCM-nCluster* and STATPC with a generic post-processing phase to obtain the actionable subspace cluster. More specifically, we mined subspace clusters in each time frame, and get all valid combinations of the subspace clusters to form actionable subspace clusters. That is, $(O \times A)$ is an actionable subspace cluster if and only if there exists a subspace cluster $(O' \times A')$ in each time frame, such that $O \subseteq O'$ and $A \subseteq A'$.

Let \mathcal{C}^* be the set of embedded actionable subspace clusters in a synthetic sequential dataset \mathcal{D} , and \mathcal{C} be the set of actionable subspace clusters mined from \mathcal{D} . Let an embedded actionable subspace cluster $C^* = O^* \times A^*$ and an actionable subspace cluster $C = O \times A$.

The following three measurements are used to measure how close the quality of \mathcal{C} is to \mathcal{C}^* , which are based on [15].

- *recoverability* $R = \sum_{C^* \in \mathcal{C}^*} \frac{r(C^*)}{|O^*| + |A^*|}$, where $r(C^*) = \max\{|O^* \cap O| + |A^* \cap A| \mid C^* = O^* \times A^*, C = O \times A, C \in \mathcal{C}\}$. *recoverability* measures the ability of \mathcal{C} to recover \mathcal{C}^* .
- *spuriousness* $S = \sum_{C \in \mathcal{C}} \frac{s(C)}{|O| + |A|}$, where $s(C) = |O| + |A| - \max\{|O^* \cap O| + |A^* \cap A| \mid C^* = O^* \times A^*, C = O \times A, C^* \in \mathcal{C}^*\}$. *spuriousness* measures how spurious \mathcal{C} is.
- *significance* $= \frac{2R(1-S)}{R+(1-S)}$. *significance* is a measure to find the best trade-off between the *recoverability* and *spuriousness* of \mathcal{C} . The higher the *significance*, the more similar is \mathcal{C} to the embedded clusters \mathcal{C}^* .

The other approaches do not consider utility in their clustering, and in order to have a fair comparison, we allowed the other approaches to use pre-processed \mathcal{D} only contains objects whose utility and correlation measures are at least those of the embedded clusters. For algorithm *LCM-nCluster*, we varied its parameter setting δ from 0.1 to 0.9. δ controls the differences allowed in the values of a subspace cluster. One surprising result is that algorithm TRICLUSTER did not produce any clusters in all our experimental settings. It either could not mine any 3D subspace clusters, or the mining could not be completed within 6 hours, which may be due to scalability issue on the dataset. For algorithm STATPC, we used its recommended statistical significance level settings.

¹Algorithm TRICLUSTER can only be run on Linux environment

Table 1: Settings for Algorithm *MASC*

Experiment	τ	μ	δ	λ	k	$util_{min}$
5.1.1	1	10	0.001	0.1	10	0.5
5.1.2					5-15	
5.1.3						
5.2.1	1	10	$10^{-7} - 10^{-1}$	$10^{-4} - 10^2$	10	0.5
5.2.2	$10^{-4} - 10^2$	$10^{-4} - 10^2$	0.001	0.1	2-50	
5.2.3	1	10				
5.3.1	1	10	0.001	0.1	10	0.5
5.3.2			$10^{-7} - 10^{-1}$			
5.4	1	10	0.001	0.1	10	0.3-1

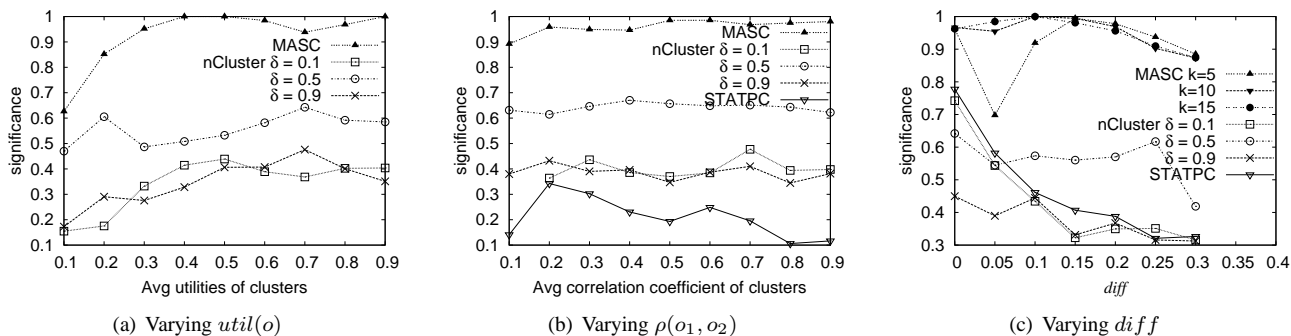


Figure 3: The significance of the actionable subspace clusters mined by different algorithms.

5.1.1 Varying utility $util(o)$ We varied the utility $util(o)$ of the embedded clusters and the results are presented in Figure 3(a). The *significance* of the clusters mined by MASC are close to 1, particularly when the utility is high (≥ 0.3). The *significance* drops when utility decreases, as we found the cluster to be contaminated by randomly generated objects. However, these random objects also have similar utility and values with the embedded clusters, so it makes sense to combine them with the cluster. On the other hand, the *significance* of the clusters mined by algorithm *LCM-nCluster* is quite low, and all clusters found by *STATPC* have zero significance. Hence, we do not present the results of *STATPC* in Figure 3(a).

5.1.2 Varying correlation coefficients $\rho(o_1, o_2)$ We varied the correlation coefficients $\rho(o_1, o_2)$ among objects in each embedded cluster and the results are presented in Figure 3(b). We can see that the *significance* of clusters mined by algorithm MASC is also much higher than those by the other approaches.

5.1.3 Varying $diff$ We varied $diff$ and the results are presented in Figure 3(c). The *significance* of clusters mined by MASC is still much higher than those mined by other approaches. When $diff = 0$, the *significance* of the clusters mined by *LCM-nCluster* reaches 0.78, but it drops dramatically as $diff$ increases. This shows that *LCM-nCluster* is only suitable to mine clusters that contain objects with exact

values, which is not common in real-world datasets. On the contrary, MASC can still maintain the quality of the clusters even as $diff$ increases.

5.2 Sensitivity Analysis. In this section, we evaluated how sensitive the parameters of algorithm MASC are in mining actionable subspace clusters. Similar to the previous section experimental setup, we embedded 10 actionable subspace clusters in a synthetic dataset \mathcal{D} for each experiment in this section.

5.2.1 Varying λ, δ We investigated the sensitivity of parameters λ, δ by fixing $\tau = 1, \mu = 10$ and varying λ, δ across a wide range. τ, μ are fixed at values which is recommended in [26]. Algorithm MASC is able to perfectly mine the embedded clusters in \mathcal{D} (*significance* = 1) in a wide range of parameter settings, as shown in Figure 4(a). The significance of the results drop only when extreme values are used on the parameters, that is when $\lambda \geq 10$ and $\delta \geq 0.1$. Thus, algorithm MASC is insensitive to the parameters λ, δ .

5.2.2 Varying τ, μ We then investigated the sensitivity of parameters τ, μ by fixing $\delta = 0.001, \mu = 0.1$ and varying τ, μ . The values of δ and μ are chosen at 0.001 and 0.1 respectively as the region of values around them give high significance for their results and this setting also gives a relatively faster running time, which we shall explain in Section 5.3. Algorithm MASC is still able to perfectly mine

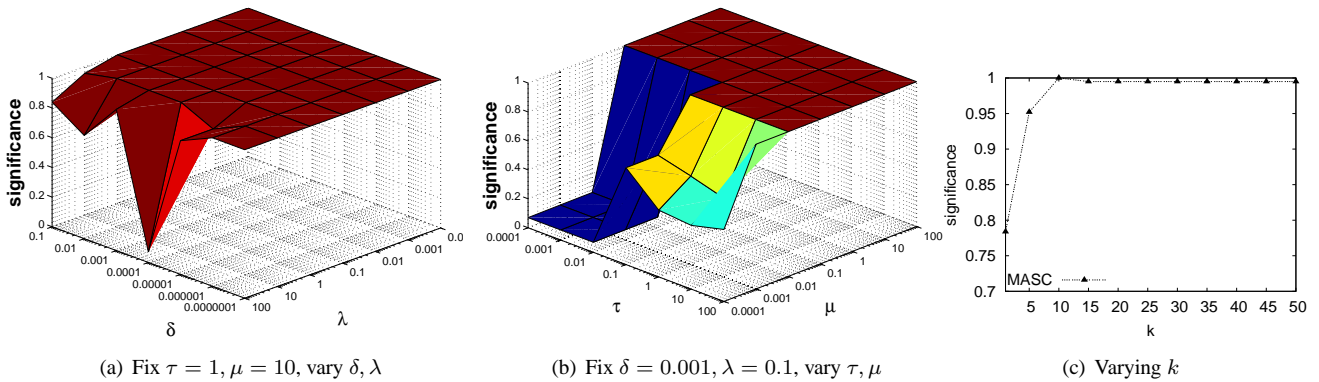


Figure 4: The significance of the actionable subspace clusters mined by algorithm MASC across varying parameter settings.

the embedded clusters in \mathcal{D} in a wide range of parameter settings, as shown in Figure 4(b). τ is insensitive to any range of values, and $\mu \geq 1$ is recommended to get results with high *significance*. Hence, our observations are in accordance with the author’s recommended settings of $\tau = 1, \mu = 10$ [26].

5.2.3 Varying k We checked the sensitivity of parameter k and used the default settings for the other parameters. Figure 4(c) presents the results, which shows that for all $k \geq 5$, the *significance* of the results is close to 1. Thus, we demonstrated the robustness of the k -nearest neighbors heuristic.

From these experiment results, we can see that problems in mining the embedded actionable subspace clusters only exist when we used extreme values for these parameters. Therefore, users can opt to use the default settings of these parameters.

5.3 Efficiency Analysis In this section, we evaluated the efficiency of algorithm MASC in mining actionable subspace clusters. We investigated two main areas that affect the running time of algorithm MASC, which are: (a) the size of dataset \mathcal{D} and (b) the parameter settings. On the size of \mathcal{D} , we argue that only the number of objects needs to be evaluated. The main computation costs of MASC lies in the computation of the optimal probability distribution P , and the number of objects directly affects this computation. The number of time frames does not affect the running time of MASC, as the objects’ values and utilities are averaged over time in MASC. Whereas for the number of attributes, it is a constant factor to the running time of computing P , as we compute P for every attribute.

5.3.1 Varying Number of Objects We varied the number of objects from 1,000 to 10,000 in a synthetic dataset, with

an attribute and a time frame. We then randomly chose 10 centroids from \mathcal{D} and averaged the running time of computing P of each centroid. Figure 5(a) presents the average running time used in computing P , which is a polynomial function of the number of objects. Furthermore, it takes less than a minute to compute P of size 10,000, which is feasible for real-world applications.

5.3.2 Varying δ, λ We investigated how the parameter settings affect the running time of MASC. For this experiment, we used the same experimental setup as Section 5.3.1. Figure 5(b) presents the running time of varying the tolerance level δ . As δ decreases, the convergence of solution is slower, which leads to the expected increase of running time.

Figure 5(c) presents the running time of varying λ , which shows that the running time increases as λ increases. The running time is fastest when $\lambda = 1$, which implies that 1 is a close estimate of the actual λ . At other settings, the running time is slower as MASC has to iterate more to achieve the correct λ . We set $\lambda = 1$ as the default, since it is also shown in Section 5.2.1 that the *significance* of the results is high at $\lambda = 1$. We do not show the running time of varying τ, μ since their default settings are recommended in [26].

5.4 Case Study: Using Actionable Subspace Clusters in the Stock Market In value investment, investors scrutinize the financial ratios of stocks to find stocks that potentially generate high profits. One of the most famous value investment strategy is formulated by the founder of value investment, Benjamin Graham, which was proven to be profitable [27]. This strategy consists of a buy phase and a sell phase. The buy phase consists of ten criteria on the financial ratios and price of stocks, with five ‘reward’ criteria and five ‘risk’ criteria. If a stock satisfies at least one of the ‘reward’ criteria and one of the ‘risk’ criteria, we buy this stock. This

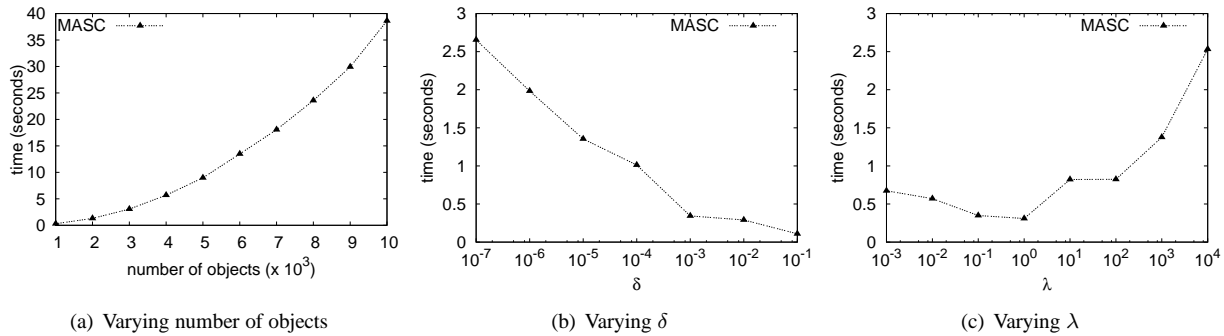


Figure 5: The running time of algorithm MASC across varying datasets and parameter settings.

set of ten criteria is presented in [27]. In the sell phase, a stock is sold if its price appreciates to 50% within two years. If not, then it is sold after two years.

5.4.1 Comparing Different Value Investment Strategies

The set of 10 criteria in the buy phase is formulated based on the Graham’s 40 years of research in the stock market. In this case-study, we attempt to replace this human perception-based buy phase with actionable subspace clustering-based buy phase. As described in Example 1, using actionable subspace clusters has two-fold effect. Firstly, stocks that are clustered have historical high returns (utilities), so they are potential good buys. Secondly, the subspace clusters of financial ratios can be studied to understand why they lead to stocks with high returns.

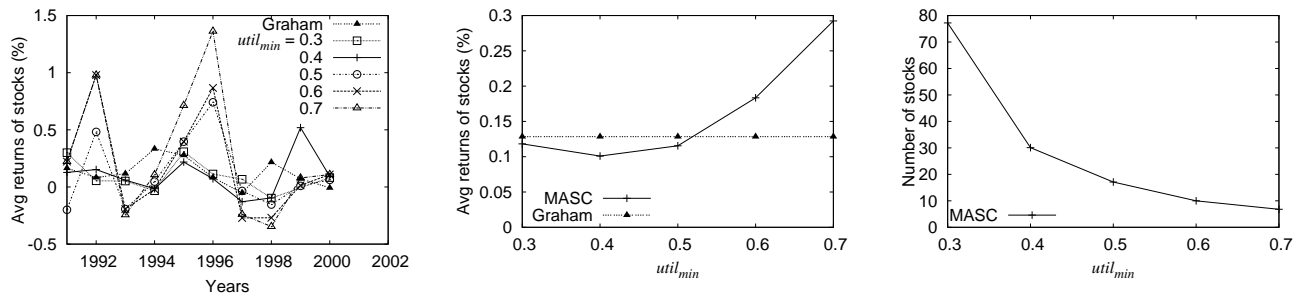
In this experiment, we downloaded financial figures and price data of all North American stocks from year 1980 to 2000. This 20 years of data was downloaded from Compustat [11]. We converted these financial figures into 30 financial ratios, based on the ratios’ formula from Investopedia [16] and Graham’s ten criteria. We removed penny stocks (whose prices are less than USD\$5) from the data as there is a high risk that these stocks are manipulative and their financial figures are less transparent [29]. Thus, we assume that the investor exercises due caution and only concentrates on big companies’ stocks. After pruning the penny stocks, we have 1762 to 2071 stocks for each year.

For Graham’s investment strategy, the buy phase was conducted over a ten year period starting from year 1991 to 2000. The buy phase starts on 1991 as Graham’s ten criteria requires the analysis of the past ten year window frame of the stocks’ historical financial ratios and price data. Figure 6(a) shows the average returns of the stocks bought using Graham’s ten criteria, from year 1991 to 2000. The average returns across the ten years is 12.84%, which is a substantial amount of profit.

To have a fair comparison, for each buy phase from year 1991 to 2000, we also used the ten year window frame of the stocks’ historical financial ratios and price data to mine

actionable subspace clusters. We assume utility $util(o)$ as the annual price return of the stock o and we use the average utility over time $util^{avg}(o)$ to measure the price return of stocks over time. From the set of actionable subspace clusters mined from each buy phase from year 1991 to 2000, we bought the stocks that are in them. Figure 6(a) shows the average returns of the stocks bought from year 1991 to 2000, and we varied $util_{min}$ from 0.3 to 1, and used the default settings for the parameters of the algorithm MASC. The results for settings $util_{min} > 0.7$ are not shown as no stocks could be clustered for certain years. The average returns across the ten years is shown in Figure 6(b). We can see that at certain $util_{min}$ settings, the average returns across the ten years are significantly much better than Graham’s investment strategy, especially when $util_{min} = 0.7$, the average return across the ten years is 29.23%. A possible explanation is Graham’s investment strategy is formulated in 1977 and his ten criteria are not adapted to the evolving dynamics of the stock market. On the other hand, actionable subspace cluster is able to capture this evolving dynamics between stock prices and financial ratios to cluster profitable stocks.

5.4.2 Effectiveness of Actionable In this experiment, we investigated if high utilities do correspond to the concept of actionable. From Figure 6(b), we can see that the average returns across the ten years follows in increasing trend as $util_{min}$ increases. As mentioned in the previous section, the results for settings $util_{min} > 0.7$ are not shown as no stocks could be clustered for certain years. Recall that $util_{min}$ is the minimum average return required by the stocks to be chosen as centroids for mining actionable subspace clusters, so the possibility of stocks having higher average returns being clustered is higher. Since Figure 6(b) shows that using actionable subspace clusters with higher utilities generates higher profits, we have shown that higher utilities correlates to higher actionability. Figure 6(c) shows that as $util_{min}$ increases, the average number of stocks in a cluster decreases. For an investor who wants to diversify his



(a) Average returns of stocks from 1991 to 2000 (b) Average returns of stocks for different utilities (c) Average number of stocks for different utilities

Figure 6: Case study on the stock market dataset.

portfolio, he can choose the appropriate $util_{min}$ to suit her desired number of stocks to buy.

5.4.3 Usefulness of the Actionable Subspace Clusters

Experiment 5.4.1 paints a simple scenario where the investors completely trust the algorithm MASC in stocks selection. For more sophisticated investors, they will take this clustering step as a pre-process to narrow down the number of stocks they need to analyze, and the actionable subspace clusters defined by stocks and financial ratios will serve as useful knowledge. The clusters serve two purposes to the sophisticated investor. Firstly, if the clusters contain the investor's 'favorite' financial ratios (financial ratios that he uses to pick stocks), then he may buy the stocks of this cluster, as this cluster substantiates the association between his favorite financial ratios and high returns. Secondly, the clusters may provide new, insightful knowledge to the investors.

We show in Table 2 some examples of actionable subspace clusters that are mined from Experiment 5.4.1, which illustrate the two purposes. The last column of the table shows the average returns of the stocks after selling them via Graham's sell phase.

For the first purpose, there are investors who believe that companies should not pay dividends and the funds should be retained and subsequently used or distributed in better ways [12]. Cluster 1 from Table 2 reflects this observation and they yield a 20.83% return for the investor.

For the second purpose, we look at the clusters which give the highest return from our experiments. In cluster 2, its cash flow to debt ratio increases from an average of 0.073 to 0.32 across the past 10 years, which implies this may be a crucial ratio to identify potential good stocks. Buying these two stocks give a return of 136%. In cluster 3, the operating profit margin increases over 10 years, while the effective tax rate is kept constant and the debt ratio is decreasing. This shows that increasing profit margins, maintaining the tax rate and keeping a healthy balance sheet are useful indicators as these stocks give a return of 78.46%.

6 Conclusion

We have proposed actionable subspace clusters, where (1) values of its objects exhibit significant degree of homogeneity in each timestamp, and evolution of the values across time is allowed, and (2) its objects have high and correlated utility across time. We proposed a framework MASC, which integrates several cross-domain techniques in a novel manner to mine actionable subspace clusters. We showed that the clustering results are not sensitive to parameter settings, which highlights the robustness of MASC. In the case-study on real world stock market data, we show that using actionable subspace clusters for investment yields a higher return than one of the most famous value investment strategies. Furthermore, we found that higher utility of the clusters correlates with higher actionability.

Acknowledgement

We would like to thank Zeyar Aung, Chuan-Sheng Foo and Suryani Lukman for their constructive suggestions. We would also like to thank the respective authors for providing us the source code of their algorithms.

References

- [1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pp. 207–216, 1993.
- [2] E. I. Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, 23(4):589–609, 1968.
- [3] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD*, pp. 59–68, 2004.
- [4] R. J. Bayardo, B. Goethals, and M. J. Zaki, editors. *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, volume 126 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
- [5] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *ICDT*, pp. 217–235, 1999.

Table 2: Actionable subspace clusters mined from Experiment 5.4.1

Cluster	Year bought	Stocks	Financial ratios	$util_{min}$	Avg return (%)
1	1994	BBAO, CPQ.2, DIGI.1, DV, HTCH	No Dividend Yield	0.3	20.83
2	1997	AB.H, DIGI.1	Increasing Cash Flow To Debt Ratio	0.7	136
3	1999	IPAR, MLI	Increasing Operating Profit Margin, Constant Effective Tax Rate, Decreasing Debt Ratio	0.3	78.46

- [6] S. Bochkonov and V. Bystritsky. ALGLIB 2.0.1 L-BFGS algorithm for multivariate optimization. <http://www.alglib.net/optimization/lbfgs.php> [Last accessed 2009].
- [7] J. Y. Campbell and R. J. Shiller. Valuation ratios and the long run stock market outlook: An update. In *Advances in Behavioral Finance II*. Princeton University Press, 2005.
- [8] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Data peeler: Constraint-based closed pattern mining in n-ary relations. In *SDM*, pp. 37–48, 2008.
- [9] H. Cheng, K. A. Hua, and K. Vu. Constrained locally weighted clustering. *PVLDB*, 1(1):90–101, 2008.
- [10] Y. Cheng and G. M. Church. Biclustering of expression data. In *ISMB*, pp. 93–103, 2000.
- [11] Compustat. <http://www.compustat.com> [Last accessed 2009].
- [12] M. Feldstein and J. Green. Why do companies pay dividends? *The American Economic Review*, 73(1):17–30, 1983.
- [13] B. Graham. *The Intelligent Investor: A Book of Practical Counsel*. Harper Collins Publishers, 1986.
- [14] B. Graham and D. Dodd. *Security Analysis*. McGraw-Hill Professional, 1934.
- [15] R. Gupta, G. Fang, B. Field, M. Steinbach, and V. Kumar. Quantitative evaluation of approximate frequent pattern mining algorithms. In *KDD*, pp. 301–309, 2008.
- [16] Investopedia. <http://www.investopedia.com/university/ratios/> [Last accessed 2009].
- [17] L. Ji, K.-L. Tan, and A. K. H. Tung. Mining frequent closed cubes in 3D datasets. In *VLDB*, pp. 811–822, 2006.
- [18] I. T. Jolliffe. *Principal Component Analysis*, pp. 115–118. Springer, 2002.
- [19] J. Kleinberg, C. Papadimitriou, and P. Raghavan. A microeconomic view of data mining. *Data Mining and Knowledge Discovery*, 2(4):311–324, 1998.
- [20] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*, 3(1):1–58, 2009.
- [21] P. Kröger, H.-P. Kriegel, and K. Kailing. Density-connected subspace clustering for high-dimensional data. In *SDM*, pp. 246–257, 2004.
- [22] J. W. Lewellen. Predicting returns with financial ratios. *Journal of Financial Economics*, 74:209–235, 2004.
- [23] G. Liu, J. Li, K. Sim, and L. Wong. Distance based subspace clustering with flexible dimension partitioning. In *ICDE*, pp. 1250–1254, 2007.
- [24] 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*, pp. 533–541, 2008.
- [25] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.
- [26] J. Nocedal and S. J. Wright. *Numerical Optimization*, pp. 497–528. Springer, 2006.
- [27] H. R. Oppenheimer. A test of ben graham’s stock selection criteria. *Financial Analysts Journal*, 40(5):68–74, 1984.
- [28] T. Uno, M. Kiyomi, and H. Arimura. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI*, 2004.
- [29] U.S. Securities and Exchange Commission. Microcap stock: A guide for investors. <http://www.sec.gov/investor/pubs/microcapstock.htm> [Last accessed 2009].
- [30] K. Wang, S. Zhou, and J. Han. Profit mining: From patterns to actions. In *EDBT*, pp. 70–87, 2002.
- [31] K. Wang, S. Zhou, Q. Yang, and J. M. S. Yeung. Mining customer value: From association rules to direct marketing. *Data Mining and Knowledge Discovery*, 11(1):57–79, 2005.
- [32] X. Xu, Y. Lu, K.-L. Tan, and A. K. H. Tung. Finding time-lagged 3D clusters. In *ICDE*, pp. 445–456, 2009.
- [33] K. Y. Yip, D. W. Cheung, and M. K. Ng. On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. In *ICDE*, pp. 329–340, 2005.
- [34] L. Zhao and M. J. Zaki. TRICLUSTER: an effective algorithm for mining coherent clusters in 3D microarray data. In *SIGMOD*, pp. 694–705, 2005.

Appendix

The complete expression of $\frac{\partial F(P)}{\partial p_{o_i}}$ on attribute a

$$\begin{aligned}
 \frac{\partial F(P)}{\partial p_{o_i}} = & -p_{o_i} s(o_i, c) \bar{u}_{o_i, c} \\
 & \cdot \sum_{o, o' \in \mathcal{O} \times \mathcal{O} | o \neq o'} p_o p_{o'} s(o, c) s(o', c) \rho_{o, o'} \\
 & - 2 \sum_{o \in \mathcal{O}} p_o s(o, c) \bar{u}_{o, c} \\
 & \cdot \sum_{o' \in \mathcal{O} | o \neq o'} p_o p_{o'} s(o, c) s(o', c) \rho_{o, o'} \\
 & - \lambda + \mu \left(\sum_{o \in \mathcal{O}} p_o - 1 \right)
 \end{aligned}$$