MACs: Multi-Attribute Co-clusters with High Correlation Information

Kelvin Sim^{1,2}, Vivekanand Gopalkrishnan², Hon Nian Chua¹,
and See-Kiong Ng¹

¹ Institute for Infocomm Research, A*STAR, Singapore

² School of Computer Engineering, Nanyang Technological University, Singapore

Abstract. In many real-world applications that analyze correlations between two groups of diverse entities, each group of entities can be characterized by multiple attributes. As such, there is a need to co-cluster *multiple* attributes' values into pairs of highly correlated clusters. We denote this co-clustering problem as the *multi-attribute co-clustering* problem. In this paper, we introduce a generalization of the mutual information between two attributes into mutual information between two attribute sets. The generalized formula enables us to use *correlation information* to discover *multi-attribute co-clusters (MACs)*. We develop a novel algorithm MACminer to mine MACs with high correlation information from datasets. We demonstrate the mining efficiency of MACminer in datasets with multiple attributes, and show that MACs with high correlation information have higher classification and predictive power, as compared to MACs generated by alternative high-dimensional data clustering and pattern mining techniques.

1 Introduction

Co-clustering values of two attributes (also known as pairwise co-clustering) is a well-established research area with many successful applications, ranging from clustering words and documents [7], to clustering video shots and video features [17]. In pairwise co-clustering, the values of two attributes are partitioned into clusters such that pairwise pairings of these clusters formed co-clusters. More recently, star-structured co-clustering [8] was proposed to handle higher dimensional data. In essence, a star-structured co-cluster is a set of pairwise co-clusters, with the constraint that each pairwise co-cluster involves the center attribute and a non-center attribute. Figure 1(a) shows some examples of pairwise co-clusters.

Such pairwise co-cluster structures are not applicable in many real-world applications that involve co-clustering multiple attributes' values into pairs of clusters that correlate. We call such problems *multi-attribute co-clustering*. Figure 1(b) depicts the difference between the multi-attribute co-clusters (MACs) and the pairwise star-structured co-clusters (Figure 1(a)). Here are some real-world examples:

Fig. 1. (a) Pairwise co-clustering. The combination of clusters with horizontal and vertical lines form pairwise co-clusters. Combination of clusters with horizontal lines, vertical lines and dots form star-structured co-clusters. (b) Multi-attribute co-clustering. The pair of clusters with horizontal lines form a multi-attribute co-cluster, and so do the pair of clusters with vertical lines.

Table 1. Dataset of company management attributes and performance indicators

Company	Management		Performance		Correlation Information
	CEO Tenure	Management Team Size	ROE	D/E	
1	1	2	3	4	0.0322
2	1	2	3.1	4.1	0.0322
3	1	2	3.2	4.2	0.0322
4	1	2	3.3	4.3	0.0322
5	5	6	3	4	0.0322
6	5	6	3.1	4.1	0.0322
7	5	6	3.2	4.2	0.0322
8	5	6	3.3	4.3	0.0322
9	7	4	9	10	0.332
10	7.1	4	9	10.1	0.332

Example 1. In finance, a key research challenge is to investigate the correlation between management and the performance of the companies [6, 14]. In a study by Murray and Goyal [14], management attributes were shown to affect the performance of the companies. Table 1 shows an example dataset of companies. The first two attributes are management attributes: CEO Tenure and Management Team Size. The next two attributes reflect the performance of the companies, as measured by their efficiency indicator (ROE ratio) and debt indicator (D/E ratio). The problem of understanding how management attributes affect the company performance is a multi-attribute co-clustering problem. This concept is illustrated in Table 2, where each row represents a MAC mined from Table 1. We can see that each MAC contains two clusters of attributes' values that are highly correlated. In this example, the last MAC has the highest correlation information as all companies with CEO Tenure(7.1,7) and Management Team Size(4) will also have ROE(9) and D/E(10,10.1).

Table 2. Three multi-attribute co-clusters (MACs) obtained from the dataset in Table 1

MAC	CEO Tenure	Management Team Size	ROE	D/E	Correlation Information
1	1	2	(3,3,3)	(4,4,3)	0.129
2	5	6	(3,3,3)	(4,4,3)	0.129
3	(7,7,1)	4	9	(10,10,1)	0.464

Table 3. Example of protein-protein interactions (PPIs) data. Each transaction represents a pair of proteins that interact and the corresponding Gene Ontology (GO) functions that they are annotated with.

Protein-Protein Interactions	Gene Ontology (GO) functions							
	F_1^1	F_2^1	...	F_n^1	F_1^2	F_2^2	...	F_n^2
P1-P7	1	1	...	0	1	1	...	0
P3-P6	1	0	...	0	0	0	...	1
P2-P5	0	1	...	1	1	1	...	1
P4-P9	1	0	...	0	0	1	...	0
P1-P6	1	1	...	0	0	0	...	1

Example 2. In biology, proteins can be profiled using biologically relevant features such as functional annotations. A relevant problem is to discover *pattern pairs* in the biological profiles of proteins that may be associated with the proteins’ propensity to interact [11]. More specifically, given a set of known protein-protein interactions (PPIs), the objective is to find *correlated* pattern pairs from the biological profiles of interacting proteins. These pattern pairs can then be used to identify unknown interactions. Table 3 shows an example of PPI data. The first column of the table refers to protein-pairs that are known to be involved in an interaction. Each row of the table represents the Gene Ontology (GO) functions that the two interacting proteins are annotated with. $F_1^1, \dots, F_n^1$ and $F_1^2, \dots, F_n^2$ represent the two sets of GO functions associated with the first and second protein respectively. An entry of ‘1’ in the column F_i^x denotes that the x^{th} protein in the interaction is annotated with function i , and an entry of ‘0’ indicates otherwise. By treating each GO function as a binary attribute, we can mine MACs from the data, where a MAC consists of two clusters of GO functions that are correlated in the presence of PPI.

Conventional pairwise co-clustering algorithms typically use information theory to partition the values of two attributes into correlated clusters. The idea that two clusters are co-clustered if they are correlated, that is, attributes’ values in both clusters occur frequently together and not by chance, can be quantified by mutual information $I(X_i; Y_j) = \sum_{x_i, y_j} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}$, with X_i and Y_j as two attributes of a dataset. Intuitively, we can see that the first part of this formula measures how frequent values x_i, y_j occur together and the second part measures if their occurrence together is by chance.

In this work, we adopt the information theory principles of existing (pairwise) co-clustering works [7, 8, 17] to obtain our desired MACs. Existing co-clustering techniques are limited to co-clustering pairwise attributes due to

mutual information calculation being restricted to a pair of attributes. We overcome this limitation by generalizing the mutual information $I(X_i; Y_j)$ to mutual information $I(X_1, \dots, X_n; Y_1, \dots, Y_m)$ of two sets of attributes $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$, which is non-trivial.

For a MAC, the correlation between its two clusters of attributes' values can then be calculated by the contribution of these attributes' values to $I(X_1, \dots, X_n; Y_1, \dots, Y_m)$. We denote this contribution as the *correlation information* of the MAC — high correlation information means the pair of clusters forming the MAC is highly correlated. To validate our claim, we show that these MACs have good clustering quality, as well as high predictive and classification power in our experiments.

We develop an algorithm MACminer which mines k MACs with high correlation information, so that users will not be overwhelmed by a large amount of results. To overcome the curse of dimensionality problem faced when clustering data with multiple attributes, we adopt the subspace clustering approach [15]. We also remove redundant co-clusters, keep maximal co-clusters and merge highly overlapping co-clusters, to keep the results succinct but informative. As correlation information has no anti-monotone property [9], traversing the search space of the dataset to mine co-clusters can be computationally expensive. Hence, we introduce several heuristic techniques to prune the search space.

One may think that correlation information can be calculated by considering both sets of attributes as two meta-attributes and the pairwise mutual information is used instead. However, this will create an unnatural constraint that a value from each attribute must be in a MAC. It is possible that values from a subset of attributes form a MAC with high correlation and values from the excluded attributes are noise to the MAC. An alternative way to solve multi-attribute co-clustering is to use existing high-dimensional data clustering [12, 13] or pattern mining [9, 10] techniques and adapt them to mine MACs. However, each of these techniques has its own set of criteria to define its results, and these criteria are not catered to mine our MACs, which are pairs of clusters that are highly correlated. In Section 5, we show that our MACs have much higher predictive and classification power than MACs mined by existing high-dimensional data clustering or pattern mining techniques [9, 10, 12].

In summary, we address the multi-attribute co-clustering problem with the following contributions:

- we generalize the mutual information $I(X_i; Y_j)$ between two attributes X_i and Y_j , to the mutual information $I(X_1, \dots, X_n; Y_1, \dots, Y_m)$ of two *sets* of attributes $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$. (Section 3)
- we propose using *correlation information* to measure the correlation between the two clusters of attributes' values of a MAC. Correlation information is based on $I(X_1, \dots, X_n; Y_1, \dots, Y_m)$. (Section 3)
- we develop a novel algorithm MACminer, which efficiently mines k MACs with high correlation information. (Section 4)
- we conduct experiments to show the efficiency of MACminer and to show that MACs discovered by MACminer have good clustering quality, high predictive and classification power. (Section 5)

Table 4. Definitions of symbols

Symbol	Definition
X_i, Y_j	A left attribute and a right attribute
X^n	A set of left attributes $\{X_1, X_2, \dots, X_n\}$
Y^m	A set of right attributes $\{Y_1, Y_2, \dots, Y_m\}$
$x_i \in X_i$	A value from the left attribute X_i
$y_j \in Y_j$	A value from the right attribute Y_j
x_1^n	A tuple of values $(x_1, x_2, \dots, x_n)$ from the attributes $\{X_1, X_2, \dots, X_n\}$
$\tilde{x}_i \subseteq X_i$	A set of values from the attribute X_i
$\tilde{x}_1^n$	A tuple of sets of values $(\tilde{x}_1, \dots, \tilde{x}_n)$ from the attributes $\{X_1, X_2, \dots, X_n\}$
$\mathcal{C} = (\tilde{x}_a^b, \tilde{y}_c^d)$	A MAC (multi-attribute co-cluster)
$\mathcal{A} = \{X_a^b, Y_c^d\}$	An attribute set. $1 \leq a < b \leq n, 1 \leq c < d \leq m$
D	A dataset

2 Preliminaries

Let $\{X_1, X_2, \dots, X_n\}$ and $\{Y_1, Y_2, \dots, Y_m\}$ be two sets of attributes which we will refer to as the left and right attributes respectively. Let dataset D be a set of transactions. A transaction is a tuple $(x_1, \dots, x_n, y_1, \dots, y_m)$ of values from the attributes, with each $x_i \in X_i$ and each $y_j \in Y_j$, for $1 \leq i \leq n$ and $1 \leq j \leq m$. We denote $\tilde{x}_i$ as a set of values from the left attribute X_i , that is, $\tilde{x}_i \subseteq X_i$, and $\tilde{y}_j$ as a set of values from the right attribute Y_j , that is, $\tilde{y}_j \subseteq Y_j$. For brevity, we denote a set of attributes $\{X_1, \dots, X_n\}$ as X_1^n , a tuple of values $(x_1, \dots, x_n)$ as x_1^n , and a tuple of sets of values $(\tilde{x}_1, \dots, \tilde{x}_n)$ as $\tilde{x}_1^n$.

Definition 1 (MAC (multi-attribute co-cluster)). Assume that we have a tuple of sets of values $\tilde{x}_a^b = \{\tilde{x}_i \subseteq X_i : 1 \leq a \leq i \leq b \leq n\}$ from a set of left attributes X_a^b , and a tuple of sets of values $\tilde{y}_c^d = \{\tilde{y}_i \subseteq Y_i : 1 \leq c \leq i \leq d \leq m\}$ from a set of right attributes Y_c^d . We denote a MAC as $\mathcal{C} = (\tilde{x}_a^b, \tilde{y}_c^d)$.

Definition 2 (Attribute set of the MAC). Given a MAC $\mathcal{C} = (\tilde{x}_a^b, \tilde{y}_c^d)$, we denote $\mathcal{A} = \{X_a^b, Y_c^d\}$ as the attribute set of $\mathcal{C}$.

For example, the first row of Table 2 is a MAC $\mathcal{C} = ((1, 2), ((3, 3.3), (4, 4.3)))$. The attribute set of $\mathcal{C}$ is $\mathcal{A} = \{\{\text{CEO Tenure, Management Team Size}\}, \{\text{ROE, D/E}\}\}$.

Let us assume that the degree of correlation between $\tilde{x}_a^b$ and $\tilde{y}_c^d$ of a MAC is measured by some metric θ — the higher θ is, the higher is the degree of correlation. We are interested in mining the top- k MACs that have the highest θ . However, as mining the top- k MACs is a NP-complete problem [16], we propose to mine the approximate top- k MACs instead.

Problem Statement. Given a dataset D that can contain quantitative and categorical attributes, we propose to mine k MACs with high θ , sorted in descending order of their θ values.

3 Correlation Information

We propose to use correlation information as the metric θ to measure the correlation of MACs. An attribute X_i can be considered as a random variable with probability mass function $p(x_i) = \Pr\{X_i = x_i\} = \frac{occ(x_i)}{|D|}$, where $occ(x_i)$ is the number of times value x_i occurs in dataset D . The conditional probability of value x_i occurring in D , given the occurrence of value y_j in D , is $p(x_i|y_j) = \Pr\{X_i = x_i|Y_j = y_j\} = \frac{occ(x_i, y_j)}{occ(y_j)}$, where $occ(x_i, y_j)$ is the number of times values x_i, y_j occur together in the transactions of dataset D .

The *mutual information* $I(X_i; Y_j) = \sum_{x_i, y_j} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}$ is the reduction in the uncertainty of the right attribute Y_j due to the knowledge of the left attribute X_i . In other words, it shows the amount of information about Y_j that can be described by X_i .

We are interested to know specifically *which* value x_i of the attribute X_i is correlated to *which* value y_j of the attribute Y_j . We quantify this information as the *correlation information* between values x_i and y_j . The correlation information between a pair of values x_i and y_j is defined as $ci((x_i), (y_j))) = p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}$.

We are also interested in *group* phenomenon — how a group of values is correlated to another group of values. To find these kind of MACs, we must calculate the mutual information between their left attributes and right attributes, which is the mutual information of their attribute set.

For simplicity, let us consider a scenario where the attribute set contains only one left attribute and several right attributes, $\{\{X_i\}, Y_1^m\}$. To calculate the mutual information of this attribute set, we use the *chain rule for mutual information*, defined as

$$I(X_i; Y_1^m) = \sum_{j=1}^m I(X_i; Y_j|Y_1^{j-1})$$

The equation above shows that the chain rule for mutual information is a summation of *conditional mutual information*. The conditional mutual information of X_i and Y_j given Y_k is

$$I(X_i; Y_j|Y_k) = \sum_{x_i, y_j, y_k} p(x_i, y_j, y_k) \log \frac{p(x_i, y_j|y_k)}{p(x_i|y_k)p(y_j|y_k)}$$

We first make the assumption that the *generalized* conditional mutual information, which is the conditional mutual information of X_i and Y_j given X_1^{i-1}, Y_1^{j-1} , is

$$I(X_i; Y_j|X_1^{i-1}, Y_1^{j-1}) = \sum_{x_1^i, y_1^j} p(x_1^i, y_1^j) \log \frac{p(x_i, y_j|x_1^{i-1}, y_1^{j-1})}{p(x_i|x_1^{i-1}, y_1^{j-1})p(y_j|x_1^{i-1}, y_1^{j-1})}$$

We then use the generalized conditional mutual information to obtain the mutual information of an attribute set.

Definition 3 (Mutual information of an attribute set). *The mutual information of the attribute set $\mathcal{A} = \{X_1^n, Y_1^m\}$ is*

$$\begin{aligned} I(X_1^n; Y_1^m) &= \sum_{i=1}^n \sum_{j=1}^m I(X_i; Y_j | X_1^{i-1}, Y_1^{j-1}) \\ &= \sum_{i=1}^n \sum_{j=1}^m \sum_{x_1^i, y_1^j} p(x_1^i, y_1^j) \log \frac{p(x_i, y_j | x_1^{i-1}, y_1^{j-1})}{p(x_i | x_1^{i-1}, y_1^{j-1}) p(y_j | x_1^{i-1}, y_1^{j-1})} \end{aligned}$$

For the interested reader, the derivation for the equation above is given in [16]. The mutual information of an attribute set is the generalized chain rule for mutual information, thus it is not order dependent.

The correlation information of a MAC $\mathcal{C}$ is derived from the mutual information of the attribute set $\mathcal{A}$ of $\mathcal{C}$. Let the probability mass function of a set of values $\tilde{x}_i \subseteq X_i$ be $p(\tilde{x}_i) = \sum_{x_i \in \tilde{x}_i} Pr\{X_i = x_i\}$.

Definition 4 (Correlation information of a MAC). *The correlation information of $\mathcal{C}$ is*

$$ci(\mathcal{C}) = \sum_{i=a}^b \sum_{j=c}^d p(\tilde{x}_a^i, \tilde{y}_c^j) \log \frac{p(x_i, y_j | \tilde{x}_a^{i-1}, \tilde{y}_c^{j-1})}{p(x_i | \tilde{x}_a^{i-1}, \tilde{y}_c^{j-1}) p(y_j | \tilde{x}_a^{i-1}, \tilde{y}_c^{j-1})}$$

We can see that the correlation information of $\mathcal{C}$ is the total contributions of all subsets of $\tilde{x}_a^b$ and $\tilde{y}_c^d$ to the mutual information of attribute set $\mathcal{A}$. Since mutual information of attribute set $\mathcal{A}$ shows the amount of information about Y_c^d that can be described by X_a^b , correlation information shows the amount of information about $\tilde{y}_c^d$ that can be described by $\tilde{x}_a^b$. It is possible that contributions of subsets of $\tilde{x}_a^b$ and $\tilde{y}_c^d$ to the mutual information of $\mathcal{A}$ are negative, so correlation information is not biased towards large MACs.

We use Table 1 to show how correlation information of MACs is calculated. We focus on the first and third MACs $\mathcal{C} = ((1, 2), ([3, 3.3], [4, 4.3]))$ and $\mathcal{C}' = (([7, 7.1], 4), (9, [10, 10.1]))$. Their correlation information are $ci(\mathcal{C}) = p(1, [3, 3.3]) \log \frac{p(1, [3, 3.3])}{p(1)p([3, 3.3])} + p(1, [3, 3.3], [4, 4.3]) \log \frac{p(1, [4, 4.3] | [3, 3.3])}{p(1 | [3, 3.3]) p([4, 4.3] | [3, 3.3])} + p(1, 2, [3, 3.3]) \log \frac{p(2, [3, 3.3] | 1)}{p(2 | 1) p([3, 3.3] | 1)} + p(1, 2, [3, 3.3], [4, 4.3]) \log \frac{p(2, [4, 4.3] | 1, [3, 3.3])}{p(2 | 1, [3, 3.3]) p([4, 4.3] | 1, [3, 3.3])} = 0.129$ and $ci(\mathcal{C}') = 0.464$. An intuitive reason why $ci(\mathcal{C}')$ is high is because the occurrence of $([7, 7.1], 4)$ in the dataset always results in the occurrence of $(9, [10, 10.1])$. $ci(\mathcal{C})$ is low because $([3, 3.3], [4, 4.3])$ occurs not only together with $(1, 2)$, but also together with $(5, 6)$. Thus, $\mathcal{C}$ is less informative.

4 MACminer

We propose a novel algorithm, MACminer to mine k MACs with high correlation information, sorted in descending order of their correlation information. MACminer consists of three main parts:

1. Traversing the search space of D to obtain attribute sets
2. Obtaining MACs from each attribute set
3. Post-processing of the results by merging highly overlapping MACs

4.1 Traversing the Search Space

Given a dataset D with left attributes X_1^n and right attributes Y_1^m , MACminer obtains attribute sets $\mathcal{A}$ from the set enumeration tree of X_1^n and Y_1^m in depth-first order. Enumerating all $\mathcal{A}$ has a complexity of $O(2^{n+m})$, so pruning the search space is essential. Given that correlation information has no anti-monotone property, we have to implement heuristic pruning techniques to prune the search space.

Let us describe the pruning techniques by assuming that MACminer is in the search space of D that has attribute set $\mathcal{A} = \{X_a^b, Y_c^d\}$, and we are extending the attribute set with $X_k \in X_1^n - X_a^b$. These pruning techniques are also applicable when we extend the attribute set with $Y_k \in Y_1^m - Y_c^d$.

Pruning Technique 1: If $I(X_a^b \cup X_k; Y_c^d) = I(X_a^b; Y_c^d)$, we do not extend $\mathcal{A}$ with X_k . This is because no new information is gained between the left and right attributes of the attribute set when we add new attribute X_k to it.

Pruning Technique 2: If $I(X_a^b \cup X_k; Y_c^d) \leq \sum_{X_i \in X_a^b \cup X_k, Y_j \in Y_c^d} I(X_i; Y_j)$, we do not extend $\mathcal{A}$ with X_k . If the mutual information of the attribute set is less than the sum of pairwise mutual information of the attributes in the attribute set, then synergy between its left and right attributes does not exist.

Pruning Technique 3: If $\exists Y_j \in Y_c^d$ where $I(X_k; Y_j) > I(X_a^b \cup X_k; Y_c^d) - I(X_a^b; Y_c^d)$, we do not extend $\mathcal{A}$ with X_k . If the mutual information of X_k and Y_j is more than what X_k can contribute to the mutual information of the attribute set $\mathcal{A}$, then the attribute set $\{\{X_k\}, \{Y_j\}\}$ is more informative than $\mathcal{A}$ extended with X_k .

Pruning Technique 4: Let $f(i, j)$ be the highest mutual information that can be achieved, when the attribute set has i left attributes and j right attributes. If $I(X_a^b \cup X_k; Y_c^d) < f(|X_a^b \cup X_k|, |Y_c^d|)$, we do not extend $\mathcal{A}$ with X_k . So at a level of the set enumeration tree of X_1^n and Y_1^m , only the attribute set that has the highest mutual information at that level can be extended. We denote parameters l and r as the size of the left and right attribute sets the pruning technique is to be used.

4.2 Obtaining MACs from an Attribute Set

After obtaining an attribute set $\mathcal{A}$, MACminer takes the values of X_a^b and Y_c^d in each transaction t of D to be a MAC $\mathcal{C}$.

$\mathcal{C}$ is then used to update *list*, a list of MACs sorted in descending order of their correlation information. When updating the list, we also conduct the following two checks:

Non-redundant MACs. Given a MAC $\mathcal{C}$, we check if *list* contains a MAC $\mathcal{C}'$ such that $\mathcal{C}'$ is a proper superset of $\mathcal{C}$ and correlation information of $\mathcal{C}$ is equal to the correlation information of $\mathcal{C}'$. If so, we replace $\mathcal{C}'$ by $\mathcal{C}$ in *list*. Both $\mathcal{C}$ and $\mathcal{C}'$ giving the same amount of correlation information means that there are some redundant values in $\mathcal{C}'$ which do not increase the correlation information of $\mathcal{C}'$.

Maximal MACs. We also check in *list* if there exists any MAC $\mathcal{C}'$ such that $\mathcal{C}'$ is a proper subset of $\mathcal{C}$ and correlation information of $\mathcal{C}$ is larger than the

correlation information of $\mathcal{C}'$. If so, we replace $\mathcal{C}'$ with $\mathcal{C}$ in *list*, as we prefer maximal MACs that have higher correlation information than their subsets that have lower correlation information.

4.3 Merging MACs

After mining of MACs is completed, we can merge those that are highly overlapping. This is an effective way of handling noisy datasets—the noise may be the cause of the small differences in highly similar MACs. In addition, this step of merging MACs is particularly useful for MACs that contain categorical values. Since MACminer does not partition categorical values, a MAC that only contains categorical values becomes a pair of itemsets, with each itemset containing a value from each attribute of the attribute set of the MAC. For example, if the attributes are words and documents, then a MAC can contain only a word and a document.

MACminer merges two MACs if they satisfy the merging condition (Definition 5), which is adopted from [19]. MACminer iteratively merges two MACs that satisfy the merging condition. The merging stops when no more MACs satisfy the merging condition.

Definition 5 (Merging condition). *Given two MACs $(\tilde{x}_a^b, \tilde{y}_c^d)$ and $(\tilde{x}'_a^b, \tilde{y}'_c^d)$, if $\sum_{i=a}^b |\tilde{x}_i \cap \tilde{x}'_i| + \sum_{j=c}^d |\tilde{y}_j \cap \tilde{y}'_j| \geq \delta(\sum_{i=a}^b |\tilde{x}_i \cup \tilde{x}'_i| + \sum_{j=c}^d |\tilde{y}_j \cup \tilde{y}'_j|)$, then we merge them to become MAC $(\tilde{x}''_a^b, \tilde{y}''_c^d)$, where $\tilde{x}''_a^b = (\tilde{x}_a \cup \tilde{x}'_a, \dots, \tilde{x}_b \cup \tilde{x}'_b)$ and $\tilde{y}''_c^d = (\tilde{y}_c \cup \tilde{y}'_c, \dots, \tilde{y}_d \cup \tilde{y}'_d)$. δ is a parameter controlling the strictness of the merging.*

5 Experimentation Results

MACminer was coded in C++ and the algorithms [7, 9, 10, 12] that we used for comparison were kindly provided by the respective authors. Except for the QCoMine [10] and *Co-cluster* [7] algorithm, all of our experiments were conducted in a Win XP environment with a 3.6Ghz Pentium 4 CPU and 2GB memory. As QCoMine and *Co-cluster* can only be compiled in Linux, we ran them in Linux environment with a 4-way Intel Xeon based server and 8GB memory.

We used three real world in our experiments. The protein-protein interaction (PPI) dataset was downloaded from Gene Ontology [2] and BioGRID [5] databases, the 20 Newsgroup subsets dataset was downloaded from UCD Machine Learning Group [1], and the Insurance dataset was downloaded from UCI Machine Learning Repository [3].

PPI dataset. Proteins were profiled using their functional annotation from Gene Ontology (GO) [2]. GO annotations were organized as a Direct Acyclic Graph, with more specific terms at lower levels and more general terms at higher levels of the graph. To avoid overly general terms, we ignored terms in the top two

levels of the graph. Existing PPIs were obtained from the BioGRID [2] database. The data consists of a total of 38,555 physical interactions between proteins. We transformed both sets of data into a set of transactions, which contains 38,555 transactions with 3851 left and 3851 right attributes. Each transaction was labeled with two proteins that have interactions, and the attributes of the transaction are the two sets of GO functions that the two proteins are annotated with respectively. A toy example is shown in Table 3.

20 Newsgroup subset datasets. Each transaction of the datasets indicates a word, a document and the occurrence number of the word in the document. The left attribute is *words* and the right attribute is *documents*. Stop-word removal and stemming were applied on these datasets by the original authors. ob-2-1 is a dataset which contains two topics and has 80,009 transactions, while ob-8-1 is a dataset which contains eight topics and has 316,867 transactions. Both datasets have balanced clusters containing 500 documents each, and they were clustered according to their topics. In these datasets, the words and documents may belong to more than one topic. For example, the word “earth” belongs to topics *Space* and *Graphics*.

Insurance dataset. This dataset contains 5,822 customer transactions with 85 left attributes relating to the customer’s profile and 1 right (class) attribute indicating if the customer buys a caravan policy. This dataset contains 4 quantitative attributes, and they are discretized using a parameter-free, mutual information based technique. The interested reader may refer to [16] for the details of this discretization technique.

5.1 Performance of MACminer

As there is no existing work on mining MACs with high correlation information, we show the efficiency of MACminer by assessing the efficiency of its pruning techniques. We used the Insurance and PPI datasets, and we set $k = 100$ in this experiment. The newsgroup dataset is not used since there is no need to prune its search space of two attributes. We calculated how much times faster MACminer is with each pruning technique (denoted as speedup) and Figure 2 presents the results. Techniques 1, 2 are more effective on the PPI dataset than the Insurance dataset, while pruning techniques 3, 4 are very effective on both the PPI and Insurance datasets. In fact, for the Insurance dataset, MACminer could not complete mining after 24 hours without pruning technique 3. In general, the pruning techniques can improve the mining time of MACminer on different types of datasets.

5.2 Multi-Attribute Co-clustering: Predicting PPIs

We mined MACs from the PPI dataset, where each MAC is a pattern pairs of GO functions that are associated with proteins that interact. We then evaluated these MACs by using them to predict PPIs in the PPI dataset and the accuracy of the predictions was validated by a five-fold cross-validation on the PPI dataset. The proteins in the PPI dataset were randomly divided into five equal-sized groups.

Fig. 2. The speedup based on different pruning techniques (denoted as Tech)

Table 5. AUCs of the PPI predictions by different types of patterns

Pattern Type		AUC	
	minsup	avg num	
Closed itemsets	5000	24	60.31
	10000	127	59.76
	15000	434	54
	20000	2182	56.21
	25000	24631	56.5
MACs	k		
	100		74.46
	200		72.61
	300		72.24
	400		72
	500		72.3

Table 6. Performance results of the different classifiers

Classifier	tp	fp	Precision	Recall	F-measure
SVM	0	0	0	0	0
C4.5	0	238	0	0	0
Linear	17	221	0.227	0.071	0.109
k-nearest neighbor	29	209	0.155	0.122	0.136
Naive bayes	97	141	0.135	0.408	0.203
MACs, k=100	81	679	0.119	0.34	0.177

In each cross-validation run, the proteins in one fold were isolated for evaluation, while those from the remaining four folds were used for mining pattern pairs. The pattern pairs were then used to predict PPIs from the isolated fold. Given a pattern pair, if a protein from the isolated fold is annotated with the GO functions of one of the pattern and another protein from the isolated fold is annotated with the GO functions of the other pattern, then we predict these two proteins interact, and we assign a prediction score to this prediction, which is based on the correlation information of the pattern pair.

The predictions made for all five folds were evaluated against the known protein interactions, and we computed the Receiver Operating Characteristic (ROC) curve (a graphical plot of the sensitivity versus (1-specificity)) of the predictions results. We then calculated the area under the ROC curve (AUC). An AUC of 1 means the classifier is perfect, while an AUC of 0.5 means the classifier is similar to a random guesser.

Since there are no existing algorithms that perform the same task, we used the frequent closed itemset mining approach [9] with post-processing as our baseline. Frequent closed itemsets were mined from the PPI dataset, and in the post-processing, frequent closed itemsets that do not correspond to pattern pairs were pruned. We then used the confidence of these frequent closed itemsets to calculate their prediction score.

The AUCs of MACs and frequent closed itemsets are presented in Table 5. For the frequent closed itemsets, we set minimum support ms from 5000 to 25000, and avg num in Table 5 indicates the average number of pattern pairs mined from frequent closed itemsets across the 5 folds. For MACs, we varied the number of predictors k from 100 to 500, and we disabled pruning techniques 1-3 and set $l = k = 1$. From Table 5, we can see that our approach achieves an average AUC score from 72% to 74.46%, which indicates that the pattern pairs discovered by MACs are meaningful and relevant. On the other hand, the average AUC score achieved by the baseline method is significantly lower. A possible explanation of why frequent closed itemsets are poor predictors is that some functional annotations are more general than the others and may be annotated to many proteins. Hence high co-occurrences of such functions in interacting proteins may not be associated with biologically meaningful annotations patterns. Our method, on the other hand, takes into account the correlation between two sets of pattern, and thus is able to detect dependency relationships in pattern pairs.

5.3 Pairwise Co-clustering: Co-clustering Words and Documents

We performed pairwise co-clustering of words and documents in the 20 Newsgroup subsets using MACs and *Co-cluster* [7], which was developed specifically for this task. We assessed how accurate the co-clusters mined by MACs and *Co-cluster* are in clustering documents of the same topic together.

Since the 20 Newsgroup subsets used in [7] were unavailable, and to have an unbiased comparison, we used the 20 Newsgroup subsets that was prepared by a neutral party [1]. For the parameter settings of *Co-cluster* algorithm, we adjusted its two main parameters, the desired number of document clusters and the desired number of word clusters, and kept the rest as default settings. We set the number of document clusters to the number of topics for the dataset and we varied the number of word clusters from 8 to 128 (128 word clusters were shown to get good co-clustering results in [7]). Hence, the number of co-clusters mined varies from 16 to 1024. For MACminer, we merged co-clusters during mining to prevent them from being too specific, with $\delta = 0.3$. We mined 100 co-clusters without any pruning technique, since there are only two attributes, words and documents. Co-clusters mined by *Co-cluster* and MACminer that contain only one document were removed.

In the document cluster of each co-cluster, we checked the topic of each document and the *dominant topic* is the topic that the majority number of documents in the cluster belong to. So in a document cluster, documents belonging to the dominant topic c are deemed to be assigned correctly. We used the evaluation measure in [7], micro-averaged-precision $P(d)$, to measure the accuracy of clustering documents of the same topic together. $P(d) = \sum_c \alpha(c, d) / \sum_c (\alpha(c, d) + \beta(c, d))$, where $\alpha(c, d)$ is the number of documents correctly assigned to topic c , $\beta(c, d)$ is the number of documents incorrectly assigned to c and d is denoted as the set of documents in the dataset.

The micro-averaged precision of the co-clusters mined by MACminer and *Co-cluster* across different parameter settings was calculated and the results are

Fig. 3. Micro-averaged precision of the co-clusters mined by *Co-cluster* and MACminer

presented in Figure 3. The first five columns show the micro-averaged precision of the co-clusters mined by *Co-cluster* and the last column show the micro-averaged precision of the co-clusters mined by MACminer. In ob-2-1 dataset that contains two topics, Figure 3(a) shows that the micro-averaged precision of co-clusters mined by MACminer is much better than those by *Co-cluster*.

On the more complex ob-8-1 dataset that has 8 topics, the micro-averaged precision of the co-clusters mined by MACminer is again much better than those by *Co-cluster*, as shown in Figure 3(b). This demonstrates that MACminer is able to cluster documents of similar topics effectively, regardless of the number of topics. In these datasets, the words and documents can belong to more than one topic, so it is natural that the co-clusters of words and documents should overlap. Existing co-clustering frameworks do not allow overlapping co-clusters, thus they perform poorly in these datasets.

5.4 Multi-Attribute to One Attribute Co-clustering: Using MACs to Build a Simple Classifier

In this experiment, we used MACs as rules to build a simple classifier. The training dataset is the Insurance dataset and the testing dataset has the same attributes as the training dataset but it contains 4000 other customers. The objective is to predict the number of customers that will buy a caravan policy from the testing dataset. This dataset is challenging because only 238 of the customers bought the policy. 100 MACs were mined from the training dataset without pruning technique 4 and they were used as rules for our classifier. As these MACs were used for predicting customers buying, in each MAC, its left cluster contains values from the left attributes (customer’s profile attributes) and its right cluster contains the value ‘buy’ of the class attribute. If a customer from the testing dataset has attributes’ values that are in any of the 100 MACs, then our simple classifier predicts that this customer will buy.

For comparison, we tried using quantitative correlated patterns [10] and sub-space clusters [12] for comparison. We mined and post-processed them, with the requirement that each of them contains both left and right attributes’ values, so that they can be used as rules for a classifier. However, even at their lowest

parameter settings, QCoMine [10] mined 0 quantitative correlated patterns from the training dataset and LCM-nCluster [12] mined 121 subspace clusters from the training dataset but none contains the value ‘buy’ of the class attribute. Since we cannot use related works to build classifiers, we compared our simple classifier with the major classifiers C4.5, SVM, Linear, k-nearest neighbors and Naïve Bayes [18].

To measure the performance of the classifier, we used the standard precision, recall and F-measure, where precision $P = \frac{tp}{tp+fp}$, recall $R = \frac{tp}{tp+fn}$ and F-measure $F = \frac{2(P.R)}{P+R}$. In this experiment context, tp is the number of correct predictions in predicting that the customer will buy, fp is the number of wrong predictions, and fn is the number of customers who buy but they are not predicted by the classifier. Table 6 summarizes the results of the different classifiers. Our simple classifier using 100 MACs and Naïve Bayes are the top performers, based on their F-measure. Although sophisticated techniques were not used to build our classifier, it is still able to perform better than the other major classifiers. A possible explanation of why SVM and C4.5 performed poorly is because the data is highly skewed as only a small percentage of customers bought the policy. The performance of SVM and C4.5 performed well in predicting customers that do not buy, but this classification result is not meaningful.

6 Related Work

For co-clustering that involves only two attributes such as words and documents [7], each word/document value is assigned to a word/document cluster, and a pair of word cluster and document cluster form a co-cluster. Since a word/document value can only be assigned to a word/document cluster, no overlapping of co-clusters is allowed. Our proposed framework allows a word/document value to be in multiple word/document clusters, thereby allowing overlapping of co-clusters. Gao *et al.* [8] extend the traditional pairwise co-cluster into star structured co-cluster. In a star structured co-cluster, a cluster of values of the center attribute is in the center, and the clusters of values of other attributes are linked to it. To obtain this structure, pairwise co-clusterings are performed between values of the center attribute and values of each non-center attribute. It is therefore a series of pairwise co-clusters, with the constraint that the clustering result of the values of the center attribute are the same in each pairwise co-cluster. This is in contrast to MACs, where each cluster of a MAC contains values of multiple attributes.

Biclustering algorithms [13] have been developed for co-clustering genes and experimental conditions. In a bicluster, one cluster contains genes and the other cluster contains the expression values of different experimental conditions. This in essence, is clustering a group of attributes’ values, where the experimental conditions are the attributes. The clustering structure of subspace clustering [12] and correlation clustering [4] are also similar to biclustering. Thus, their clustering structures are different from MACs, which are a pair of clusters of attributes’ values. Biclustering [13] and subspace clustering [12] aim to cluster

attributes' values that satisfy certain homogeneity criteria, such as constancy, similarity or coherency (scaling or shifting), whereas correlation clustering aims to cluster attributes' values that exhibit linear dependency [4]. These are different from our clustering aim; we aim to find a cluster of attributes' values that is correlated to another cluster of attributes' values.

Ke *et al.* [10] mine quantitative correlated patterns based on mutual information. Quantitative correlated pattern is a set of attributes' values, with the requirement that the pairwise mutual information between the attributes exceed a threshold and the all-confidence of the pattern exceed another threshold. Hence, we are mining different things and our clustering aims are different.

7 Conclusions

In this paper, we have introduced the problem of *multi-attribute co-clustering* for data mining applications that involve co-clustering two highly correlated clusters of attributes' values (known as MAC). We proposed using correlation information to measure the correlation in a MAC. Correlation information is based on the mutual information of two sets of attributes, which we derived by generalizing the mutual information for two attributes. We developed an algorithm MACminer, which mines MACs that have high correlation information from datasets. MACminer adopts the subspace clustering approach to overcome the curse of dimensionality problem when clustering multiple attributes dataset. MACminer also used heuristic techniques to aggressively prune the search space of the dataset during mining. Our experimental results showed that MACs produced better clustering, classification and prediction results than other algorithms.

Acknowledgment

We would like to thank Zeyar Aung, Clifton Phua, Ardian Kristanto Poernomo and Ghim-Eng Yap for their constructive suggestions. We would also like to thank the authors for providing us the executables or source codes of their algorithms.

References

1. UCD Machine Learning Group (2008), <http://mlg.ucd.ie/content/view/22/>
2. Ashburner, M., et al.: Gene ontology: tool for the unification of biology. The gene ontology consortium. *Nature Genetics* 25(1), 25–29 (2000)
3. Asuncion, A., Newman, D.: UCI Machine Learning Repository (2007), <http://www.ics.uci.edu/~mllearn/MLRepository.html>
4. Böhm, C., Kailing, K., Kröger, P., Zimek, A.: Computing clusters of correlation connected objects. In: *SIGMOD*, pp. 455–466 (2004)
5. Breitkreutz, B.-J., Stark, C., Tyers, M.: The grid: The general repository for interaction datasets. *Genome Biology* 3, R23 (2002)

6. Denis, D.J., Denis, D.K.: Performance changes following top management dismissals. *Journal of Finance* 50(4), 1029–1057 (1995)
7. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: *KDD*, pp. 89–98 (2003)
8. Gao, B., Liu, T.-Y., Ma, W.-Y.: Star-structured high-order heterogeneous data co-clustering based on consistent information theory. In: *ICDM*, pp. 880–884 (2006)
9. Grahne, G., Zhu, J.: Efficiently using prefix-trees in mining frequent itemsets. In: *ICDM Workshop on FIMI* (2003)
10. Ke, Y., Cheng, J., Ng, W.: Mining quantitative correlated patterns using an information-theoretic approach. In: *KDD*, pp. 227–236 (2006)
11. Li, X., Tan, S.H., Ng, S.-K.: Improving domain based protein interaction prediction using biologically significant negative dataset. *International Journal of Data Mining and Bioinformatics* 1(2), 138–149 (2006)
12. Liu, G., Li, J., Sim, K., Wong, L.: Distance based subspace clustering with flexible dimension partitioning. In: *ICDE*, pp. 1250–1254 (2007)
13. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 1(1), 24–45 (2004)
14. Murray, F., Goyal, V.K.: Corporate leverage: how much do managers really matter? In: *Social Science Research Network* (2007)
15. Parsons, L., Haque, E., Liu, H.: Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter* 6(1), 90–105 (2004)
16. Sim, K., Gopalkrishnan, V., Chua, H.N., Ng, S.-K.: MACs: Multi-attribute co-clusters with high correlation information (2009), <http://www.ntu.edu.sg/home/asvivek/pubs/TR0902.pdf>
17. Wang, P., Cai, R., Yang, S.-Q.: Improving classification of video shots using information-theoretic co-clustering. In: *ISCAS* (2), pp. 964–967 (2005)
18. Witten, I.H., Frank, E.: *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco (2005)
19. Zhao, L., Zaki, M.J.: TRICLUSTER: an effective algorithm for mining coherent clusters in 3d microarray data. In: *SIGMOD 2005*, pp. 694–705 (2005)