Mining Top-k Covering Rule Groups for Gene Expression Data*

Gao Cong †
School of Informatics
University of Edinburgh
gao.cong@ed.ac.uk

Kian-Lee Tan, Anthony K.H.Tung ‡, Xin Xu
Dept. of Computer Science,
National University of Singapore
{tankl, atung, xuxin}@comp.nus.edu.sg

ABSTRACT

In this paper, we propose a novel algorithm to discover the top-k covering rule groups for each row of gene expression profiles. Several experiments on real bioinformatics datasets show that the new top-k covering rule mining algorithm is orders of magnitude faster than previous association rule mining algorithms.

Furthermore, we propose a new classification method RCBT. RCBT classifier is constructed from the top-k covering rule groups. The rule groups generated for building RCBT are bounded in number. This is in contrast to existing rule-based classification methods like CBA [19] which despite generating excessive number of redundant rules, is still unable to cover some training data with the discovered rules. Experiments show that the RCBT classifier can match or outperform other state-of-the-art classifiers on several benchmark gene expression datasets. In addition, the top-k covering rule groups themselves provide insights into the mechanisms responsible for diseases directly.

1. INTRODUCTION

Microarray technology makes it possible to measure the expression levels of tens of thousands of genes in cell simultaneously and has been widely used in post-genome cancer research studies. Meanwhile, mass spectrometry technology is also increasingly being used in cancer research by measuring the mass/charge ratios of molecular proteins in tumor tissues. Both technologies typically generate only tens or hundreds of very high-dimensional data. The generated high-dimensional datasets naturally require powerful computational analysis tools to extract the most significant and reliable rules, which reveal the important correlation between gene expression patterns and disease outcomes and translate the complex raw data into relevant and clinically useful diagnostic knowledge. In this paper, we focus on gene expression profiles while our proposed techniques are also applicable to data generated by mass spectrometry technology.

*Dedicated to the late Prof. Hongjun Lu, our mentor, colleague and friend who will forever be remembered.

†Work done while at the National University of Singapore

‡Contact Author

SIGMOD 2005 June 14-16, 2005, Baltimore, Maryland, USA.

Considering the above requirements, we define a rule as a set of items, or specifically a set of conjunctive gene expression level intervals (*antecedent*) with a single class label (*consequent*). The *general* form of a *rule* is: $gene_1[a_1, b_1]$, ..., $gene_n[a_n, b_n] \rightarrow class$, where $gene_i$ is the name of the gene and $[a_i, b_i]$ is its expression interval. For example, $X95735\_at[-\infty, 994] \rightarrow ALL$ is one rule discovered from the gene expression profiles of ALL/AML tissues.

Recent studies have shown that such association rules themselves are very useful in the analysis of gene expression data. Due to their relative simplicity, they can be easily interpreted by biologists, providing great help in the search for gene predictors (especially those still unknown to biologists) of the data categories (classes). Moreover, it is shown in [6, 9, 18] that classifiers built from association rules are rather accurate in identifying cancerous cell. This is exciting because classification is one of the most important applications of microarray technology. Unfortunately, two challenges remain.

First, it has been shown in [6, 7] that huge number of rules will be discovered from the high-dimensional gene expression dataset even with rather high minimum support and confidence thresholds. This makes it difficult for the biologists to filter out rules that can encapsulate very useful diagnostic and prognostic knowledge discovered from raw datasets. Although recent row-wise enumeration algorithms like FARMER [6] can greatly reduce the number of rules by clustering similar rules into rule groups, it is still common to find tens of thousands and even hundreds of thousands of rule groups from gene expression dataset, which are rather hard to interpret.

Second, the high dimensionality together with the huge number of rules results in extremely long mining process. Rule mining algorithms using column enumeration (combinations of columns are tested systematically to search for rules), such as CHARM [31] and CLOSET+ [30], are usually unsuitable for gene expression datasets because searching in the huge column enumeration space results in extremely long running time. Although FARMER efficiently clusters rules into rule groups and adopts an anti-monotone confidence pruning with a delicate row ordering strategy, it is still very slow when the number of rule groups is huge.

These two challenges greatly limit the application of rules to analyze gene expression data. It will be ideal to discover only a small set of the most significant rules instead of generating a huge number of rules. To address this basic problem, we propose to discover the **most significant top-*k* covering rule groups (TopkRGS) for each row of a gene expression dataset**. We will illustrate this with an example.

EXAMPLE 1.1. **TopkRGS**
For the running example shown in Figure 1(a), given $minsup = 2$, *the top-1 covering rule group for rows* r_1 *and* r_2 *is* $\{abc \rightarrow C\}$ *with confidence 100%, the top-1 covering rule group for row* r_3 *is*

i	r_i	class
1	a, b, c, d, e	C
2	a, b, c, o, p	C
3	c, d, e, f, g	C
4	c, d, e, f, g	¬C
5	e, f, g, h, o	¬C

(a) Example Table

i_j	$\mathcal{R}(i_j)$	
	C	$\neg C$
a	1, 2	
b	1, 2	
c	1, 2, 3	4
d	1, 3	4
e	1, 3	4, 5
f	3	4, 5
g	3	4, 5
h		5
o	2	5
p	2	

(b) $TT|_{\emptyset}$ (or TT)

i_j	$\mathcal{R}(i_j)$	
	C	$\neg C$
a	2	
b	2	
c	2, 3	4
d	3	4
e	3	4, 5

(c) $TT|_{\{1\}}$

i_j	$\mathcal{R}(i_j)$	
	C	$\neg C$
c		4
d		4
e		4, 5

(d) $TT|_{\{1,3\}}$

Figure 1: Running Example

$\{cde \rightarrow C\}$ with confidence 66.7%, and the top-1 covering rule group for rows r_4 and r_5 is $\{fge \rightarrow \neg C\}$ with confidence 66.7%. The support values of the above top-1 covering rule groups are all 2, which is equal to $minsup$.

While formal definition will be given later, we summarize the task of finding top-k covering rule groups as essentially doing the following:

- Define an interestingness criterion that ranks the rule groups in certain order.
- Based on the ranking, for **each** row r in the dataset, find the k highest ranked rule groups of the same class as r such that the antecedent of the k rule groups are all found in r (i.e. r is covered by these k rule groups).

The top-k covering rule groups are beneficial in several ways, as listed below:

- TopkRGS can provide a more complete description for each row. This is unlike previous proposal of interestingness measure like confidence which may fail to discover any interesting rules to cover some of the rows if the mining threshold is set too high. Correspondingly, information in those rows that are not covered will not be captured in the set of rules found. This may result in the loss of important knowledge since gene expression datasets have small number of rows;
- Finding TopkRGS helps us to discover the complete set of useful rules for building a classifier while avoiding the excessive computation adopted by algorithms like the popular CBA classifier [19]. These algorithms first discover a large number of redundant rules from gene expression data most of which will be pruned in the later rule selection phase. We will prove later that the set of top-1 covering rule group for each row contains the complete set of rules required to build the CBA classifier while avoiding the generation of huge redundant rules;
- We do not require users to specify the minimum confidence threshold. Instead only the minimum support threshold and the number of top covering rule groups, k, are required. Such an improvement is useful since it is not easy for users to set an appropriate confidence threshold (we do not claim that specifying minimum support is easy here) while the choice of k is semantically clear. In fact, the ability to control k allows us to balance between two extremes. While rule induction algorithms like decision tree typically induce only 1 rule from each row and thus could miss interesting rules, association rule mining algorithms are criticized for finding too many redundant rules covering the same rows. Allowing users to specify k gives them control over the number of rules to be generated.
- The number of discovered top-k covering rule groups is bounded by the product of *k* and the number of gene expression data, which is usually quite small.

In addition to building CBA classifier with top-1 covering rule groups of each row, we try to address an open problem of CBA. When the generated CBA classifier does not cover a test data, CBA uses a default class to classify test data. Such case happens quite often for CBA on gene expression data. In fact, discussion with biologists has revealed that they are usually reluctant to believe in the classification made by selecting a default class which is done without giving any deciding factors. In this paper, we try to refine the classification of such test data by building standby classifiers. We also improve CBA by aggregating the discriminating powers of a subset of rules.

In this paper, we develop an efficient algorithm to mine the top-k covering rule groups for each row of the gene expression data. We build CBA classifier from the set of top-1 covering rule groups and develop a new classifier called RCBT. We identify our main contributions as follows.

First, we propose the concept of top-*k* covering rule groups for each row of a gene expression dataset, which is useful as discussed above.

Second, we design an efficient algorithm to discover the top-*k* covering rule groups for each row. Extensive experiments on real-life gene expression datasets show that our algorithm can be several order of magnitudes better than FARMER, CLOSET+ and CHARM which uses diff-sets.

Third, we propose a new classification technique using top-k covering rule groups. Experiments on several gene expression datasets show that our classifier outperforms or is competitive with CBA [19], IRG classifier [6], SVM [15], and C4.5 family algorithms [27] (single tree, bagging and boosting) on real-life datasets. Furthermore, we show that our method does provide knowledge of biological significance.

The rest of this paper is organized as follows: in the next section, we will introduce the concept of top-k covering rule groups for each row and its application in building CBA classifier. The proposed algorithm will be presented in Sections 3 and 4. Section 5 will present our classification methods. To illustrate the performance of our proposed algorithm and our classifier, experimental results will be given in Section 6. Section 7 reviews some related work. We will conclude our discussion in Section 8.

2. PRELIMINARY

TopkRGS and RCBT work on discretized gene expression data.

Dataset: the gene expression dataset (or table) D consists of a set of rows, R=$\{r_1, ..., r_n\}$. Let I=$\{i_1, i_2, ..., i_m\}$ be the complete set of items of D (each item represents some interval of gene expression level), and $C = \{C_1, C_2, ..., C_k\}$ be the complete set of class labels of D, then each row $r_i \in R$ consists of one or more items from I and a class label from C.

As an example, Figure 1(a) shows a dataset with 5 rows, r_1,...,r_5, the first three of which are labelled C while the other two are labelled $\neg C$. To simplify the notation, we use the *row id set* to represent a set of rows and the *item id set* to represent a set of items. For instance, "134" denotes the row set $\{r_1, r_3, r_4\}$, and "cde" denotes the itemset $\{c, d, e\}$.

As a mapping between rows and items, given a set of items $I' \subseteq I$, we define the **Item Support Set**, denoted $\mathcal{R}(I') \subseteq R$, as the largest set of rows that contain I'. Likewise, given a set of rows $R' \subseteq R$, we define **Row Support Set**, denoted $\mathcal{I}(R') \subseteq I$, as the largest set of items common among the rows in R'.

EXAMPLE 2.1. $\mathcal{R}(I')$ **and** $\mathcal{I}(R')$
Consider again the table in Figure 1(a). Let I' be the itemset $\{c, d, e\}$, then $\mathcal{R}(I') = \{r_1, r_3, r_4\}$. Let R' be the row set $\{r_1, r_3\}$, then $\mathcal{I}(R')$=$\{c, d, e\}$ since this is the largest itemset that appears in both r_1 and r_3.

Based on our definition of item support set and row support set, we can redefine the association rule.

Association Rule: an **association rule** γ, or just **rule** for short, from dataset D takes the form of $A \rightarrow C$, where $A \subseteq I$ is the antecedent and C is the consequent (here, it is a class label). The **support** of γ is defined as the $|\mathcal{R}(A \cup C)|$, and its **confidence** is $|\mathcal{R}(A \cup C)|/|\mathcal{R}(A)|$. We denote the antecedent of γ as $\gamma.A$, the consequent as $\gamma.C$, the support as $\gamma.sup$, and the confidence as $\gamma.conf$.

As discussed in the introduction, in real biological applications, biologists are often interested in rules with a specified consequent C, which usually indicates the cancer outcomes or cancer status.

2.1 Top-k Covering Rule Groups (TopkRGS)

The rule group is a concept which helps reduce the number of rules discovered by identifying rules that come from the same set of rows and clustering them conceptually into rule groups.

DEFINITION 2.1. **Rule Group**
Let D be the dataset with itemset I and C be the specified class label. $G = \{A_i \rightarrow C | A_i \subseteq I\}$ is a rule group with antecedent support set R and consequent C, iff (1) $\forall A_i \rightarrow C \in G$, $\mathcal{R}(A_i) = R$, and (2) $\forall \mathcal{R}(A_i) = R$, $A_i \rightarrow C \in G$. Rule $\gamma_u \in G$ (γ_u: $A_u \rightarrow C$) is an **upper bound** *of G iff there exists no $\gamma' \in G$ (γ':$A' \rightarrow C$) such that $A' \supset A_u$. Rule $\gamma_l \in G$ (γ_l: $A_l \rightarrow C$) is a* **lower bound** *of G iff there exists no $\gamma' \in G$ (γ': $A' \rightarrow C$) such that $A' \subset A_l$.*

LEMMA 2.1. *Given a rule group G with the consequent C and the antecedent support set R, it has a unique upper bound γ (γ: $A \rightarrow C$).*

Based on lemma 2.1, we use upper bound rule γ_u to refer to a rule group G in the rest of this paper.

EXAMPLE 2.2. **Rule Group**
Given the table in Figure 1(a). $\mathcal{R}(\{a\}) = \mathcal{R}(\{b\}) = \mathcal{R}(\{ab\}) = \mathcal{R}(\{ac\}) = \mathcal{R}(\{bc\}) = \mathcal{R}(\{abc\}) = \{r_1, r_2\}$. They make up a rule group $\{a \rightarrow C, b \rightarrow C, ..., abc \rightarrow C\}$ of consequent C, with the upper bound $abc \rightarrow C$ and the lower bounds $a \rightarrow C$ and $b \rightarrow C$.

It is obvious that all rules in the same rule group have the same support and confidence since they are essentially derived from the same subset of rows. Based on the upper bound and all the lower bounds of a rule group, it is easy to identify the remaining members. Besides, we evaluate the significance of rule groups consistently with the individual rule ranking criterion.

DEFINITION 2.2. **Significant**
Rule group γ_1 is more ***significant*** *than γ_2 if $(\gamma_1.conf > \gamma_2.conf) \vee (\gamma_1.sup > \gamma_2.sup \wedge \gamma_1.conf = \gamma_2.conf)$.*

The top-k covering rule groups, as defined below, encapsulate the most significant information of the dataset while enabling users to control the amount of information in a significance-top-down manner.

DEFINITION 2.3. **Top-k covering Rule Groups (TopkRGS)**
Given the database D and a user-specified minimum support $minsup$, the top-k covering rule groups for a row r_i is the set of rule groups $\{\gamma_{r_ij}\}$ ($1 < j < k$), where $\gamma_{r_ij}.sup \geq minsup$, $\gamma_{r_ij}.A \subset r_i$ and there exists no rule group γ' $\gamma' \notin \{\gamma_{r_ij}\}$ such that γ' is more significant than γ_{r_ij}. For brevity, we will use the abbreviation ***TopkRGS*** *to refer to top-k covering rule groups for each row.*

2.2 Usefulness of TopkRGS in Classification

In this subsection, we prove that the set of top-1 covering rule groups for each row contain the set of rules required to build CBA classifier. The basic idea of CBA classification method can be summarized as the following steps:

Step 1: Generate the complete set of class association rules CR for each class that satisfy the user-specified minimum support and minimum confidence.

Step 2: Sort the set of generated rules CR according to the relations "$\prec$". Given two rules, r_i and r_j, $r_i \prec r_j$ if and only if one of the following three conditions is satisfied (1) $r_i.conf > r_j.conf$; (2) $r_i.conf = r_j.conf \wedge r_i.sup > r_j.sup$; or (3) $r_i.conf = r_j.conf \wedge r_i.sup = r_j.sup$ and r_i is discovered before r_j. Because CBA discovers rules in breadth-first manner, this implies that CBA will select the shortest one when several rules have the same support and confidence.

Step 3: Select rules from sorted rule set CR. For each rule r in CR, if it can correctly classify some training data in D, we put it into classifier C', remove those training data covered by r and continue to test the rules after r in CR. Meanwhile, we select the majority class in the remaining data as the default class and compute the errors made by current C' and default class. This process continues until there are no rules or no training data left.

Step 4: Locate the rule r in C' that results in the least errors and discard those rules in C' after r to get the final classifier C.

As can be seen, in CBA, the rule generation scheme using fixed support and confidence thresholds at Step 1 and the rule selection scheme based on coverage test at Step 3 are simply NOT compatible with each other. Because of the extremely high dimensionality of gene expression data, even when the confidence threshold is set as high as 95%, CBA cannot finish running at Step 1 in several days. It is even more ridiculous that most of the time spent is used to generate redundant rules which will eventually be pruned away at Step 3. The following lemma proves that the rules selected by CBA for classification are actually a subset of rules of TopkRGS with $k = 1$.

LEMMA 2.2. *Given a minimum support. Let Ψ be the set of discovered top-1 covering rule groups for each training data, Ψ_s be the set of shortest lower bounds of Ψ, and C' be the set of rules selected at Step 3 of CBA method. We get $C' \subseteq \Psi_s$.*
Proof: *For each rule $r \in C'$, it must correctly classify some training data. Because of the sorting at step 2 of CBA method, r must be the top-1 covering rule of a training data if it correctly classifies the training data. This means that r must be in Ψ_s. We get the proof.*

Note that mining top-1 covering rule group does not require a *minimum confidence* threshold while CBA algorithm needs one when generating rules at Step 1. Setting too high a confidence threshold will result in some rows not being covered by the discovered rule while lowering the confidence threshold will result in substantial increase in running time. This is unlike our approach which will still find the most significant top-1 covering rule for each training data without specifying the appropriate confidence threshold in advance.

2.3 Problem Statement

The first problem that we address is to efficiently discover the set of top-k covering rule groups for each row (TopkRGS) of gene expression data given a user specified minimum support $minsup$.

In addition to applying the top-1 covering rule groups to build CBA classifier, we also propose a refined classification method based on TopkRGS (RCBT). RCBT improves CBA method in two aspects:

- RCBT reduces the chance that a test data is classified based on a default class;
- RCBT uses a subset of rules to make a collective decision.

3. EFFICIENT DISCOVERY OF TOPKRGS

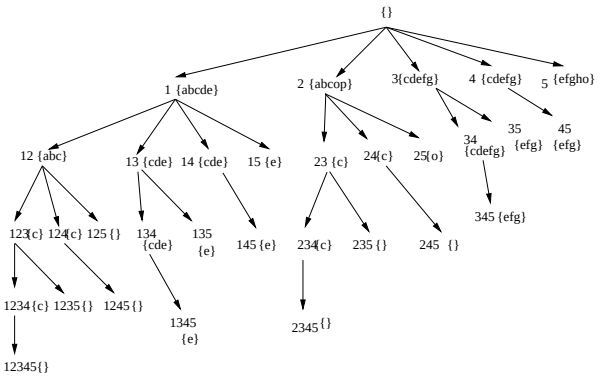


Figure 2: Row Enumeration Tree.

We first give a general review of how **row enumeration** takes place using the **(projected) transposed table** first proposed in [6] before proceeding to our TopkRGS discovery strategies. Implementation details will then be discussed.

Figure 1(b) is a transposed version TT of the table in Figure 1(a). In TT, the items become the row ids while the row ids become the items. The rows in the transposed tables are referred as *tuples* to distinguish from the so-called *rows* in the original table. Let X be a subset of rows. Given the transposed table TT, a **X-projected transposed table**, denoted as $TT|_X$, is a subset of tuples from TT such that: 1) For each tuple t in TT which contains all the row ids in X, there exists a corresponding tuple t' in $TT|_X$. 2) t' contains all rows in t with row ids larger than any row in X. As an example, the $\{13\}$-projected transposed table, $TT|_{13}$, is shown in Figure 1(d).

A complete **row enumeration tree** will then be built as shown in Figure 2. Each node X of the enumeration tree corresponds to a combination of rows R' and is labelled with $\mathcal{I}(R')$ that is the antecedent of the upper bound of a rule group identified at this node. For example, node "12" corresponds to the row combination $\{r_1, r_2\}$ and "abc" indicates that the maximal itemset shared by r_1 and r_2 is $\mathcal{I}(\{r_1, r_2\}) = \{a, b, c\}$. An upper bound $abc \rightarrow C$ can be discovered at node "12". The correctness is proven by the following lemma in [6].

LEMMA 3.1. *Let X be a subset of rows from the original table, then $\mathcal{I}(X) \rightarrow C$ must be the upper bound of the rule group G whose antecedent support set is $\mathcal{R}(\mathcal{I}(X))$ and consequent is C.*

By imposing a **class dominant order** order $\mathcal{ORD}$ on the set of rows, FARMER [6] performs a systematic search by enumerating the combinations of rows based on the order $\mathcal{ORD}$. For example, let "$1 \prec 2 \prec 3 \prec 4 \prec 5$" be the $\mathcal{ORD}$ order, then the depth-first order of search in Figure 2 will be {"1", "12", "123", "1234", "12345", "1235",...,"45", "5"} in absence of any optimization strategies. Ordering the rows in class dominant order is essential for FARMER to apply its confidence and support pruning efficiently. Class dominant order is also essential for efficient pruning based on the top-k dynamic minimum confidence, as we will discuss later.

DEFINITION 3.1. **Class Dominant Order**
A **class dominant order** *$\mathcal{ORD}$ of the rows in the dataset is an order in which all rows of class C are ordered before all row of class $\neg C$.*

Given the row enumeration strategies introduced above, a naive method of deriving the top-k covering rule groups is to first obtain the complete set of upper bound rules in the dataset by running the row-wise algorithm FARMER [6] with a low minimum confidence threshold and then picking the top-k covering rule groups for each row in the dataset. Obviously, this is not efficient. Instead, our algorithm will maintain a list of top-k covering rule groups for each row during the depth-first search and keep track of the k-th highest confidence of rule group at each enumeration node dynamically. The dynamic minimum confidence will be used to prune the search space. That is, whenever we discover that the rule groups to be discovered in the subtree rooted at the current node X will not contribute to the top-k covering rule groups of any row, we immediately prune the search down node X. The reasoning of our pruning strategies is based on the following lemma.

LEMMA 3.2. *Given a row enumeration tree T, a minimum support threshold $minsup$, and an $\mathcal{ORD}$ order based on specified class label C, suppose at the current node X, $\mathcal{R}(\mathcal{I}(X)) = X$, X_p and X_n represent the set of rows in X with consequent C and $\neg C$ respectively, and R_p and R_n are the set of rows ordered after rows in X with consequent C and $\neg C$ respectively in the transposed table of node X, $TT|_X$. Then, we can conclude that the maximal set of rows that the rule groups to be identified in the subtree rooted at node X can cover is $X_p \cup R_p$.*
Proof: *As $\mathcal{R}(\mathcal{I}(X)) = X$, the maximal antecedent support set of the rule groups to be identified at the subtree rooted at node X is $(X \cup R_p \cup R_n)$. In addition, as the rule groups are labelled C, the maximal set of rows covered by these rule groups is $(X_p \cup R_p)$.*

Combined with Definition 2.2, we compute $minconf$ and sup, the cutting points of the TopkRGS thresholds for the rows in $(X_p \cup R_p)$, where $minconf$ is the minimum confidence value of the discovered TopkRGS of all the rows in $X_p \cup R_p$, assuming the top-k covering rule groups of each row r_i is ranked in significance such that $\gamma_{r_i 1} \prec \gamma_{r_i 2} \prec ... \prec \gamma_{r_i k}$,

$$minconf = \min_{r_i \in (X_p \cup R_p)} \{\gamma_{r_i k}.conf\}, \quad (1)$$

and sup is the support value of the corresponding covering rule group with confidence $minconf$,

$$sup = \gamma_{r_c k}.sup, \ where \ \gamma_{r_c k}.conf = minconf. \quad (2)$$

According to the definition of the top-k covering rule groups (Definition 2.3), we can further obtain Lemma 3.3 below.

LEMMA 3.3. *Given the current node X, $minconf$ and sup computed according to Equations 1 and 2, if the rule group identified inside the subtree rooted at node X is less significant (according to Definition 2.2) than $\gamma_{r_c k}$ ($\gamma_{r_c k}.conf = minconf$ and $\gamma_{r_c k}.sup = sup$), then the rule group cannot become a rule group in the top-k covering rule group list of any row.*

Naturally our **top-k pruning** will proceed in the following way:

- If the upper bound of the confidence value of the rule groups to be identified in the subtree rooted at node X is below $minconf$ which is dynamically calculated at node X, then prune the search down node X;
- If the upper bound of the confidence value of the rule groups to be identified in the subtree rooted at node X is equal to $minconf$ which is dynamically calculated at node X and the upper bound of the support value of the rule groups to be identified in the subtree rooted at node X is smaller than sup, then prune the search space down node X.

The reasoning of our TopkRGS discovery is clearly that the rule groups to be discovered down node X will not contribute to the TopkRGS of any row. The top-k pruning strategy introduced above can be perfectly integrated with the backward pruning, loose and tight upper bound pruning of confidence or support values of FARMER, which further speeds up our mining process. The following is an example.

EXAMPLE 3.1. **Discovery of Top-1 Covering Rule Groups**
For the running example in Figure 1(a) where $k = 1$, specified class is C, and $minsup = 2$, when the depth-first traversal comes to node $\{1, 2\}$, the top-1 covering rule group for both r_1 and r_2 is dynamically updated to $abc \rightarrow C$ (conf:100%, sup:2). At node $\{1, 3\}$, when $X_p = \{1, 3\}$ and $R_p = \emptyset$, as the identified top-1 covering rule group for r_1 has confidence 100% while no top-1 covering rule group of r_3 has been discovered yet, we get $minconf = 0$ and $sup = 0$. Since the rule group $cde \rightarrow C$ identified at node $\{1, 3\}$ has confidence 66.7% and support 2, which is above the $minconf$ and $minsup$ thresholds, it is output to update the top-1 covering rule group of r_3. The estimated upper bound of the confidence values of the sub-level nodes down node $\{1, 2\}$ and $\{1, 3\}$ are all below the corresponding $minconf$ and are simply pruned. The consequent search down node $\{2\}$ and $\{3\}$ is pruned using the backward pruning because of the rule groups down these nodes are identified already in previous enumerations.

4. ALGORITHM

Our algorithm performs a depth-first traversal of the row enumeration tree, where each node X will be associated with X-projected transposed table. As an example, when visiting node 1 in the enumeration tree, the 1-projected transposed table will be formed as shown in Figure 1(c). Also, it is important to note that the projected transposed table at a node can in fact be computed from the projected transposed table of its parent node. To compute the 13-projected transposed table as shown in Figure 1(d), we can simply scan $TT|_1$ and extract those tuples in $TT|_1$ that contain r_3. Since the enumeration order is such that parent node will always be visited before the child node, this gives rise naturally to a recursive algorithm where each parent node will call its children passing the relevant projected transposed table to the children nodes.

Formally, the algorithm is shown in Figure 3. There are four input parameters of the algorithm, the original dataset D, class label C, the minimum support $minsup$ and k. The algorithm will scan through the dataset D to count the frequency of each item and remove infrequent items from each row in D. D will then be transformed into the corresponding transposed table. At the same time, the top-k covering rule groups for each row r_i with consequent C denoted as ζ_{r_i}=[$\gamma_{r_i 1}$, ..., $\gamma_{r_i k}$] will be initialized. Then the procedure Depthfirst() is called to perform the depth-first traversal of row enumeration tree.

The procedure Depthfirst() takes in six parameters at node X: $TT'|_X$, X_p, X_n, R_p, R_n, and $minsup$. $TT'|_X$ is the X-projected transposed table at node X. X_p and X_n represent the the set of rows in X with consequent C and $\neg C$ respectively. R_p is the set of candidate enumeration rows with consequent C that appear in $TT'|_X$ and R_n is the set of enumeration candidate rows with $\neg C$ appearing in $TT'|_X$. Among the steps in Depthfirst(), only steps 10, 12 and 14 are necessary if no pruning strategies are adopted. Step 10 scans the projected table $TT'|_X$ and computes $freq(r_i)$, the frequency of occurrence of each row r_i in $TT'|_X$. Based on $freq(r_i)$, rows that occur in all tuples (i.e. $freq(u) = \mathcal{I}(X)$) of $TT'|_X$ are found. These rows will appear in all descendant nodes of X and are thus added directly into X. Correspondingly, X_p and X_n are updated based on the consequent of these rows and they are removed either from R_p or R_n at step 12. Step 14 moves on into the next level enumerations in the search tree by selecting each row r_i that is either in R_p or R_n, creating a new $\{X \cup \{r_i\}\}$-projected transposed table and then passing the updated information to another call of MineTopkRGS.

Note that Step 14 implicitly does some pruning since it is possible that there is no row available for further enumeration, i.e. $R_p \cup R_n = \emptyset$. It can be observed from the enumeration tree that there exist some combinations of rows, X, such that $\mathcal{I}(X) = \emptyset$.

4.1 Pruning Strategies

In MineTopkRGS, top-k pruning is the main pruning strategy, and other pruning techniques first introduced in [6] are the supplementary pruning that we have seamlessly combined with our top-k pruning.

We first briefly introduce how to estimate the support upper bounds at an enumeration node X. At Step 9, it is obvious that the support of any rule groups enumerated along X cannot be more than $|X_p| + |R_p|$. The maximal number of rows with consequent C, denoted as m_p($m_p \leq R_p$), among all the branches under node X can be obtained at Step 10. As a result, we can get a tighter support upper bound at Step 11, i.e. $|X_p| + m_p$.

The estimation of confidence upper bounds is a bit complicated. For a rule γ discovered in the subtree rooted at X, its confidence is computed as $|\mathcal{R}(\gamma.A \cup C)|/(|\mathcal{R}(\gamma.A \cup C)| + |\mathcal{R}(\gamma.A \cup \neg C)|)$. This

Algorithm MineTopkRGS (D, C, $minsup$, k)

1. Scan database D to find the set of frequent items F and remove the infrequent items in each row r_i of D;
2. Let D_p be the set of rows in D with consequent C and D_n be the set of rows in D without consequent C;
3. Convert table D into transposed table $TT|_\emptyset$;
4. Initiate a list of k dummy rule groups with both confidence and support values of 0, ζ_{r_i}=[$\gamma_{r_i 1}$, ..., $\gamma_{r_i k}$], for each row r_i in D_p;
5. Call Depthfirst ($TT|_\emptyset$, $\emptyset$, $\emptyset$, D_p, D_n, $minsup$);
6. Return ζ_{r_i} for $\forall r_i \in D_p$.

Procedure: Depthfirst($TT'|_X$, X_p, X_n, R_p, R_n, $minsup$)

7. **Backward Pruning: If** there is a row r' that appears in every tuple w.r.t $\mathcal{I}(X)$ and does not belong to X, **Then** return.
8. **Threshold Updating:** Check the *k*th covering rule group $\gamma_{r_i k}$ for each row $r_i \in X_p \cup R_p$ to find the lowest confidence $minconf$ and the corresponding support sup.
9. **Threshold Pruning: If** prunable with the loose upper bounds of support or confidence, **Then** return.
10. Scan $TT'|_X$ and count the frequency, $freq(r_i)$, for each row, $r_i \in R_p \cup R_n$.
    Let $Y_p \subset R_p$ be the set of rows such that $freq(u) = |\mathcal{I}(X)|, u \in R_p$ and $Y_n \subset R_n$ be the set of rows such that $freq(u) = |\mathcal{I}(X)|, u \in R_n$;
    $X_p = X_p \cup Y_p$, $X_n = X_n \cup Y_n$ and $X = X_p \cup X_n$;
11. **Threshold Pruning: If** prunable with the tight upper bounds of support or confidence, **Then** return.
12. $R_p = R_p - Y_p,\ R_n = R_n - Y_n$.
13. $c = |X_p|/(|X_p| + |X_n|)$; //*compute confidence*
    If $((|X_p| \geq minsup) \wedge (c > minconf)) \vee ((c = minconf) \wedge (|X_p| > sup))$ **Then**
    For each $r_i \in X_p$ **Do**
    If $\exists \gamma_{r_i j} \in \zeta_{ri}, j \leq k$ such that
    $(\gamma_{r_i j}.conf < c)$ or
    $((\gamma_{r_i j}.conf = c) \wedge (\gamma_{r_i j}.sup < |X_p|))$,
    Then update ζ_{ri} with $\mathcal{I}(X) \rightarrow C$;
14. **For** each $r_i \in R_p \cup R_n$ **Do**
    If $r_i \in R_p$ **Then** $R_p = R_p - \{r_i\}$, $X_p = X_p \cup \{r_i\}$;
    If $r_i \in R_n$ **Then** $R_n = R_n - \{r_i\}$, $X_n = X_n \cup \{r_i\}$;
    $Depthfirst(TT'|_{X \cup r_i}, X_p, X_n, R_p, R_n, minsup)$;

Figure 3: Algorithm MineTopkRGS

expression can be simplified as $x/(x+y)$, where $x = |\mathcal{R}(\gamma.A \cup C)|$ and $y = |\mathcal{R}(\gamma.A \cup \neg C)|$. This value is maximized with the largest x and the smallest y. The smallest y is $|R_n|$ at node X and the largest x can be $|R_p|$ or m_p as we just discussed. Therefore, we can get a loose confidence upper bound $|R_p|/(|R_p| + |R_n|)$ at Step 9 and a tight confidence upper bound $m_p/(m_p + |R_n|)$ at Step 11.

4.1.1 Top-k Pruning

Step 8 is a very important step in our algorithm. In this step, the $minconf$ threshold is dynamically set for enumeration down X, which makes it possible to use the confidence threshold to prune the search space at steps 9 and 11. The $minconf$ threshold is obtained according to Equation 1. Steps 9 and 11 perform pruning by utilizing the user-specified minimum support threshold, $minsup$ and the dynamic minimum confidence threshold, $minconf$ (generated dynamically at step 8). If the estimated upper bound of either measure at X is below either $minsup$ or $minconf$, we stop searching down node X. At Step 9, we will perform pruning using the two loose upper bounds of support and confidence that can be calculated without scanning $TT'|_X$. At Step 11, we compute the tight upper bounds of support and confidence after scanning $TT'|_X$.

The corresponding support sup information is also recorded for computation at Step 13. Note that $sup \geq minsup$. Whenever a new rule group $\mathcal{I}(X) \rightarrow C$ is discovered at node X, a check is made to see whether the new rule is more significant than one or more rule groups in the list of top-k covering rule group for some rows in X_p, the top-k covering rule groups of such rows will be updated dynamically. This is done at Step 13.

Two additional optimization methods are utilized in our top-k pruning.

- First, because we can easily know the confidence of the rules whose antecedent is a single item at Step 1 of algorithm MineTopkRGS, we use these confidence values to initiate the confidence and support values of the list of TopkRGS at Step 4 instead of initiating them with zero. Such an optimization may cause a problem. That is, if a single item is a lower bound of an upper bound rule, the result set will not include the upper bound rule because they have the same support and confidence. We need to update the single item with the upper bound rule by adapting step 13 of algorithm MineTopkRGS. Another technical detail here is that we need ensure that any two single items to be used to initiate the top-k rule groups for one row cannot be the lower bounds of the same upper bound rules.

- Second, we dynamically increase the user-specified $minsup$ threshold if we find that all TopkRGS have 100% confidence and the lowest support value of the k rule groups is larger than the user-specified one.

MineTopkRGS outputs the most significant information for each row, as well as dramatically improves the efficiency and reduces the memory usage, compared to FARMER.

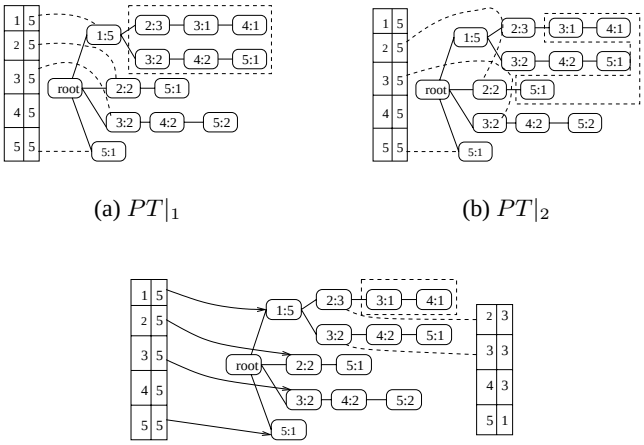
4.1.2 Backward Pruning

Step 7 implements the *backward pruning* first introduced in [6]. If there exists a row r' that appears in each prefix path w.r.t the set of nodes contributing to $\mathcal{I}(X)$ and does not belong to row set X, the rule groups $\mathcal{I}(X) \rightarrow C$ and all rule groups below X must have already been discovered below some enumeration node containing r' as proved in [6]. The principle is the same but our integration with the prefix tree makes TopkRGS more efficient. For example, at node $\{2\}$ in Figure 4 (b), we just need to do a back scan along the corresponding pointer list of node $\{2\}$ and can quickly find that there exists no such r'.

In addition, in $\mathcal{ORD}$, the rows from the same class are sorted in the ascending order of the number of frequent items contained in each row. This will improve the efficiency of algorithm MineTopkRGS.

4.2 Implementation

Next, we will illustrate how to represent (projected) transposed tables with prefix trees. The transposed table in Figure 1(b) is represented with the prefix tree shown in Figure 4 (a) (corresponding to the root node). The left head table in the figure records the list of rows in the transposed table and their frequencies. At each node of the prefix tree, we record row id and the count of the row in a prefix path (separated by ":" in Figure 4 (a)). Additional information recorded at each node but not shown in the figure is the set of items represented at the node, such as items a, b, c, d and e at node "1:5". Such information will help to determine quickly the rule group w.r.t. a projected transposed table.



(a) $PT|_1$ (b) $PT|_2$

(c) $PT|_{12}$

Figure 4: Projected Prefix Trees

EXAMPLE 4.1. *Projected Prefix Tree*
The part of nodes enclosed by dotted line in Figure 4(a) is the 1-projected prefix tree, $PT|_1$. Note that there are pointers linking the child nodes of the root with the corresponding rows in the head table. By following the pointer starting from row 1 of the header table, we can get the $PT|_1$. After $PT|_1$ has been mined recursively, the child paths of the node with label 1 will be assigned to other rows of the header table after row 1 (i.e. rows 2, 3, 4 and 5) and we get the 2-projected prefix tree, $PT|_2$. In Figure 4(b), the part enclosed by dotted line is $PT|_2$. By following the pointer from row 2 in the header table, we can get $PT|_2$.

5. USEFULNESS IN CLASSIFICATION

In this section, we will first explain how a CBA classifier can be built from the set of discovered top-1 covering rule groups for each row, then describe our proposed classification method, Refined Classification Based on TopkRGS (RCBT).

5.1 Building CBA Classifier

In order to build CBA classifier, we need to discover one of the shortest lower bounds for each top-1 covering rule group. [6] proposed a method to discover all lower bounds of a rule group. However, in entropy-based discretized gene expression datasets, a rule group may contain tens of thousands of lower bounds and discovering all these lower bounds is not only unnecessary but also computationally expensive. Since we do not aim to discover all the lower bounds, we give a straightforward but effective method to search only a given number of lower bounds for classification purpose.

LEMMA 5.1. *Rule γ' is a* lower bound rule *of* rule group G *with* upper bound rule γ *iff (1) $\gamma'.A \subseteq \gamma.A$, (2)$|\mathcal{R}(\gamma'.A)| = |\mathcal{R}(\gamma.A)|$ and (3) there is no other rule member γ'' of G such that $\gamma'.A \supset \gamma''.A$.*

With Lemma 5.1, we derive the algorithm FindLB() in Figure 5. It takes in four parameters: training data D, the upper bound rule γ, the set of rows covered by γ (denoted as $rowset$ and can be recorded when generating γ in algorithm MineTopKRGS), and the number of required shortest lower bounds nl (nl=1 for CBA classifier). At Step 1, we first rank genes based on their discriminant ability in classification measured by entropy score [3], and then rank the items in an upper bound rule based on the rankings of their corresponding genes (one gene may be discretized into several intervals, each represented by an item). In this way, we discover the shortest lower bound rules that contain items from the most discriminant genes to build CBA classifier. At step 2, for a candidate lower bound combination c_{lb}, we first test the condition (3) in Lemma 5.1; if condition (3) is satisfied, we continue to test condition (2), which is satisfied only if there does not exist a row $r \in D \wedge r \notin rowset$ that c_{lb} is contained in r. If both (2) and (3) are satisfied, c_{lb} is a lower bound. This process continues until we get the nl lower bound rules.

Algorithm FindLB(D, γ, $rowset$, nl)

1. Rank the items in $\gamma.A$ according to the descending order of the entropy scores of the corresponding genes;
2. Perform a breadth-first search in the search space formed by the list of items $\gamma.A$ until we get nl lower bound rules;

Figure 5: Algorithm FindLB

Both dataset D and candidate lower bound combinations are represented with bitmap to speed up the containment test. The discovered lower bounds usually contain 1-5 items while the upper bounds usually contain hundreds of items in the data we tested. We use one heuristic rule to speed-up the algorithm FindLB. Consider two upper bound rules, γ_1 and γ_2. Let $A' = \gamma_1.A \cap \gamma_2.A$. The lower bound rules of γ_2 will contain at least one item in $\gamma_2.A - A'$ if $\gamma_2.A - A' \neq \emptyset$, and the lower bound rules of γ_1 will contain at least one item in $\gamma_1.A - A'$ if $\gamma_1.A - A' \neq \emptyset$. We can prune the search space when we find that it will not generate a lower bound.

With the set of lower bound rules, we can build CBA classifier using the method presented in Section 2.2. Note that a minimum confidence threshold can be imposed on the set of lower bounds to filter out rules that do not satisfy the threshold to be consistent with CBA method in [19]. However, in this case, for some training data, all the rules cover them may have confidences beneath the specified confidence threshold and will be pruned off totally. So some information will be lost.

5.2 RCBT

As discussed earlier, RCBT tries to reduce the chance of classifying test data based on the default class by building stand-by classifiers to classify test data that cannot be handled by the main classifier. Moreover, RCBT uses a subset of lower bound rules to make a collective decision instead of selecting only one shortest lower bound rule for classifier building like CBA. The subset of lower bound rules are selected based on the discriminant ability of genes.

Building Classifier: RCBT has two input parameters, k, the number of covering rule groups for each row and nl, the number of lower bound rules to be used.

Let RG_j denote the set of rules groups that appear as a top-j rule group in at least one of the training data. We will thus have k sets of rule groups RG_1,...,RG_k. These k sets of rule groups are used to build k classifiers CL_1,...,CL_k with CL_j being built from RG_j. We call CL_1 the main classifier and CL_2,...,CL_k backup classifiers. For each rule group in RG_j, RCBT will find the nl shortest lower bound rules by calling algorithm FindLB(). The union of the lower bound rules will be sorted and pruned (as in Step 3 of Section 2.2) to form CL_j.

Besides both main and backup classifiers, we set a default class

like in CBA. This default class is set as the majority class of the remaining training data after step 4 in Section 2.2.

Prediction: Given a test data t, we will go through CL_1 to CL_k in that order to see if t can be handled by any of these classifiers. The first classifier that has matching rules for t will determine its class. If the test data cannot be handled by any of the classifiers, then the default class will be used for the prediction.

Instead of predicting a test data with the class of the first matching rule like in CBA classifier, RCBT tries to match all rules with an individual classifier (the main classifier or individual standby classifiers) and makes a decision by aggregating voting scores. We design a new voting score for a rule γ^{c_i} by considering both confidence and support as follows:

$S(\gamma^{c_i}) = \gamma^{c_i}.conf * \gamma^{c_i}.sup/d_{c_i}$.

where d_{c_i} is the number of training data of the class $\gamma.C$, i.e. c_i. Note that $0 \leq S(\gamma^{c_i}) \leq 1$. By summing up the scores of all rules in each class c_i, we get a score $S^{c_i}_{norm}$ for normalization purpose. Given a test data t, we suppose that t satisfies the following m_i rules of class c_i: $\gamma(t)^{c_i}_1, \gamma(t)^{c_i}_2, ...\gamma(t)^{c_i}_{m_i}$,. The classification score of class c_i for the test data t is calculated as:

$Score(t)^{c_i} = (\sum_{i=1}^{m_i} S(\gamma(t)^{c_i}_i))/S^{c_i}_{norm}$.

We make a prediction for t with the highest classification score.

6. EXPERIMENTAL STUDIES

In this section, we will look at both the efficiency of our algorithm in discovering TopkRGS and the usefulness of the discovered TopkRGS in terms of both CBA classifier and our proposed RCBT classifier. All our experiments were performed on a PC with a Pentium IV 2.4 Ghz CPU, 1GB RAM and a 80GB hard disk. Algorithms were coded in Standard C.

Datasets: We use 4 popular gene expression datasets for experimental studies. The 4 datasets are the clinical data on ALL-AML leukemia (ALL) [1], lung cancer (LC)[2], ovarian cancer(OC) [3], and prostate cancer (PC) [4]. In such datasets, the rows represent clinical samples while the columns represent the activity levels of genes/proteins in the samples. There are two categories of samples in these datasets.

We adopt the entropy-minimized partition [5] to discretize gene expression datasets. The entropy discretization algorithm also performs feature selection as part of its process. Table 1 shows the characteristics of the four discretized datasets: the number of original genes, the number of genes after discretization, the two class labels (class 1 and class 0), and the number of rows for training and test data. All experiments presented here use the class 1 as the consequent; we have found that using the other consequent consistently yields qualitatively similar results.

6.1 Efficiency

Algorithms: In term of efficiency, we compare algorithm MineTopkRGS with FARMER, CLOSET+ and CHARM (which uses diff-sets). But CLOSET+ is usually unable to run to completion within reasonable time (for several hours without results) and CHARM will report errors after using up memory on the entropy discretized datasets. Therefore, we only report the runtime of MineTopkRGS and FARMER in discovering the upper bounds of discovered rule groups. The reported time here includes the I/O time. We should point out that MineTopkRGS discovers different kinds of rules from all these existing methods.

Figure 6 (a-d) shows the effect of varying minimum support threshold $minsup$. The graphs plot the runtime for the two algorithms at various settings of minimum support. Note that the y-axes in Figure 6 are in logarithmic scale. We run algorithm MineTopkRGS by setting the parameter k at 1 and 100 respectively on all the datasets. For FARMER algorithm, we run it by setting minimum confidence $minconf$ at 0.9 and 0 (which disables the pruning with confidence threshold) on datasets ALL, and LC. Due to the relatively large number of rows in the other two datasets, FARMER is slow even when we set $minconf$ at 0.9 and 0.95 respectively. For dataset PC, the runtime curve of FARMER at $minconf$ =0.9 is at the upper right corner. We do not show the runtime of FARMER on dataset OC because it cannot finish in several hours even at $minconf$ =0.95. To further show the effect of prefix tree structure on the runtime and thus the improvement of top-k prunning alone on the runtime, we also implemented FARMER with prefix tree structure and the runtime curve is labelled as "FARMER+prefix". Note that the minimum supports shown in Figure 6 are absolute values. We usually vary minimum support from 95% to 60% when measured with a relative value. We begin with a high minimum support in order to allow FARMER to finish in reasonable time.

Figure 6 (a-d) shows that MineTopkRGS is usually 2 to 3 orders of magnitude faster than FARMER. Especially at low minimum support, MineTopkRGS outperforms both FARMER+Prefix and FARMER substantially. This is because FARMER discovers a large number of rule groups at lower minimum support while the number of rule groups discovered by MineTopkRGS is bounded. This also explains why MineTopkRGS is not sensitive to the change of minimum support threshold as shown in Figure 6. Besides, Figure 6 (a-d) demonstrates that the combination of row enumeration and the prefix tree technique speeds up the mining process successfully, by which, FARMER+prefix can improve the efficiency of FARMER by about one order of magnitude.

Figure 6 (e) shows the effect of varying k on runtime. We observe similar tendencies on all datasets and report results on datasets ALL and PC only. It is quite reasonable that MineTopkRGS is monotonously increasing with k.

The impressive performance of MineTopkRGS can be contributed to four main factors. First, TopkRGS bounds the number of discovered rules. Second, the row enumeration strategy fits the problem of mining TopkRGS very well. Third, the prefix tree structure speeds up frequency computation. Fourth, the dynamically generated minimum confidence helps in pruning search space although MineTopkRGS does not require users to specify minimum confidence.

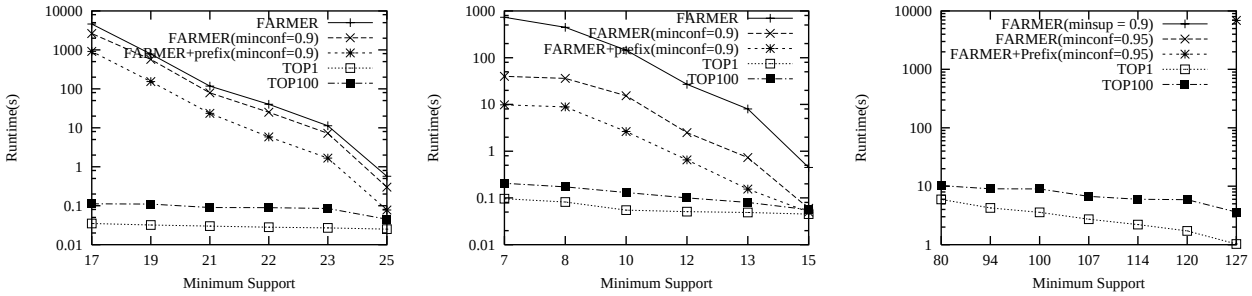
6.2 Classification Accuracy

In term of classification accuracy, we compare the performance of RCBT classifier with CBA, IRG classifier, the C4.5 family algorithms (single tree, bagging and boosting), and the support vector machine (SVM). For the C4.5 family algorithms, we use the open-source software Weka version 3.2. We use SVM^{light} 5.0 for the SVM algorithm. To keep the comparisons fair, SVM and the C4.5 family algorithms are run using the same genes selected by entropy discretization, but with the original real values of the gene expression levels. Besides, we report the best accuracy of SVM when varying between the linear and polynomial kernel functions. The open-source-code CBA usually cannot finish after running several days. We set the minimum support at 0.7 of the number of instances of the specified class to generate top-1 covering rule group of each row to build CBA classifier. The same minimum support is set for

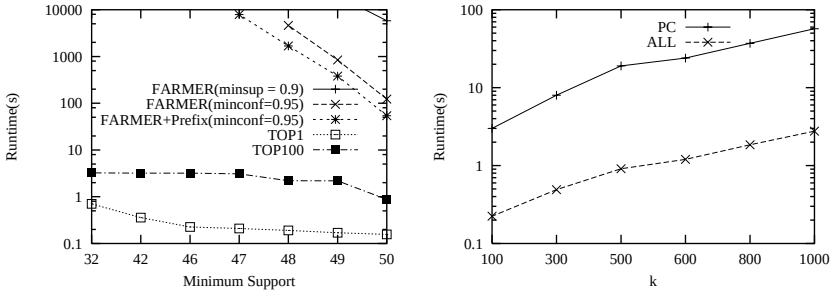
[1] http://www-genome.wi.mit.edu/cgi-bin/cancer
[2] http://www.chestsurg.org
[3] http://clinicalproteomics.steem.com/
[4] http://www-genome.wi.mit.edu/mpr/prostate
[5] the code is available at http://www.sgi.com/tech/mlc/

Dataset	# Original Genes	# Genes after Discretization	Class 1	Class 0	# Training	# Test
ALL/AML (ALL)	7129	866	ALL	AML	38 (27 : 11)	34
Lung Cancer (LC)	12533	2173	MPM	ADCA	32 (16 : 16)	149
Ovarian Cancer (OC)	15154	5769	tumor	normal	210 (133 : 77)	43
Prostate Cancer (PC)	12600	1554	tumor	normal	102 (52 : 50)	34

Table 1: Gene Expression Datasets



(a) ALL-AML leukemia (b) Lung Cancer (c) Ovarian Cancer



(d) Prostate Cancer (e) Varying k

Figure 6: Comparisons of Runtime on Gene Expression Datasets

IRG classifier and RCBT. We set minimum confidence 0.8 for IRG Classier (the same threshold is applied to CBA but we find all top-1 covering rule groups satisfy the threshold in our experiments). We set parameters $k = 10$ (TopkRGS) and $nl = 20$ (the number of lower bound rules) for RCBT.

Because the test data of all the benchmark datasets are not biased, the classification accuracy on the independent test data is used to evaluate these classification methods. Table 2 lists the classification results on the five datasets.

We first look at the last row of Table 2 to have a rough idea of these classifiers on gene expression datasets by comparing their average accuracy on four datasets. We see that the RCBT classifier has the highest average accuracy. Note that the result of IRG classifier on OC is not available since FARMER cannot finish in one day on OC and the average is computed on the other three data.

Comparison with SVM: RCBT outperforms the SVM significantly on dataset PC. SVM achieves the best results on dataset ALL although RCBT is still comparable to SVM on ALL. However, the complexity together with the distance model of SVM is much more complicated than our RCBT classifier and it is hard to derive understandable explanation of any diagnostic decision made by SVM. No doubt, these problems limit the practical use of SVM in biological discovery and clinical practice. In contrast, the RCBT classifier is very intuitive and easy to understand.

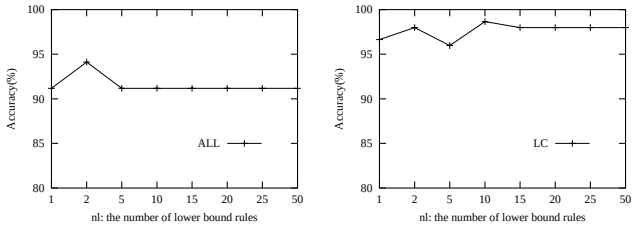
Comparison with C4.5 family algorithms: RCBT usually outperforms the C4.5 family algorithms. The C4.5 family algorithms fail on the PC data while RCBT classifier still performs well. This is because C4.5 always considers the top-ranked genes first when generating the rules to construct the decision trees, and it misses the globally significant rules on the PC data containing lower-ranked genes, as discovered by RCBT.

Comparison with CBA, IRG Classifier and RCBT: RCBT performs better than both CBA and IRG Classifier. Compared with CBA, RCBT classifies much fewer test data using default class. CBA classifies 5 test data (2 errors) on OC and 16 test data (5 errors) on PC using default class while RCBT classifies 1 test data (0 error) on OC, and 1 test data (0 error) on PC using default class. There is no test data classified using default class on ALL and LC for both CBA and RCBT.

For SVM and C4.5, we also try to use only the top 10, 20, 30,

Dataset	RCBT	CBA	IRG Classifier	C4.5 family			SVM
				single tree	bagging	boosting	
AML/ALL (ALL)	91.18%	91.18%	64.71%	91.18%	91.18%	91.18%	97.06%
Lung Cancer(LC)	97.99%	81.88%	89.93%	81.88%	96.64%	81.88%	96.64%
Ovarian Cancer(OC)	97.67%	93.02%	-	97.67%	97.67%	97.67%	97.67%
Prostate Cancer(PC)	97.06%	82.35%	88.24%	26.47%	26.47%	26.47%	79.41%
Average Accuracy	95.98%	87.11%	80.96%	74.3%	77.99%	74.3%	92.70%

Table 2: Classification Results



(a) ALL-AML leukemia (b) Lung Cancer

Figure 7: Effect of Varying *nl* on Classification Accuracy

or 40 entropy-ranked genes when building the classifier. In both cases, the performances of SVM and C4.5 often become worse. There are two main reasons that contribute to the performance of RCBT classifier. The first is that we build a series of standby classifiers besides the main classifier. The second is that we use a subset of lower bound rules in building classifier. Next, we analyze the effect of both factors in detail and explain how we can set the parameters for RCBT.

Usefulness of Standby Classifiers in RCBT: In our experiments, we set $k = 10$ for TopKRGS to build RCBT classifiers, which means that we build 9 standby classifiers besides a main classifier for each dataset. We find that the standby classifiers classify 2 test data of OC (no error) and 2 test data of PC (no error). On datasets ALL and LC, the main classifier makes all decision. This shows the usefulness of standby classifiers. We would like to stress that these standby classifiers not only improve the classification accuracy but also make the results more convincing to biologists since most test data are not classified by default class.

We also find that only the first 4 standby classifiers are used to classify some test data on all the four datasets. Therefore, RCBT is quite insensitive to the value of k as long as k is set to a sufficiently large value.

Sensitivity Analysis of nl for RCBT: We set $nl = 20$ to build RCBT classifier. Figure 7 shows the effect of varying nl on the classification accuracy on datasets ALL and LC. Both curves are quite plain especially when $nl > 15$ (changing nl does not affect accuracy). We observe similar trend on other datasets and only report results on ALL and LC. Again, as long as nl is set to a reasonable large value, RCBT will not be affected by it.

We also study the effect of varying minimum support thresholds from 0.6 to 0.8 on accuracy and find that the performance of both CBA and RCBT are not affected for all datasets.

In summary, the discovered TopkRGS are shown to be useful for classification for both CBA and RCBT. RCBT is both accurate and easy to understand. The parameters for RCBT are also easy for tuning. Besides, experimental results show that some important genes used in RCBT are really responsible for the cancer pathogenesis.

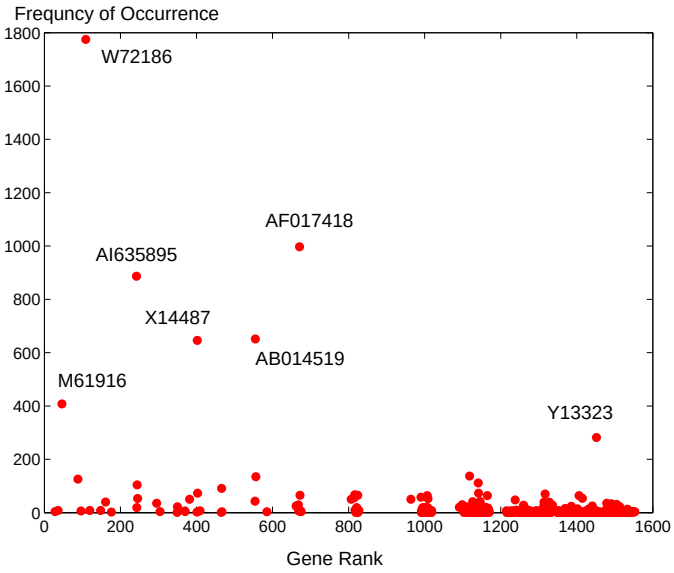


Figure 8: Chi-square based Gene Ranks and the Frequencies of Occurrence of the 415 Genes which Form the Top-1 Covering Rules of RCBT on the Prostate Cancer Data. Genes whose Frequencies of Occurrence are Higher than 200 are Labelled.

Biological Meaning: As the lower bound rules RCBT selected from the Prostate Cancer data contain genes of lower-ranks, it is interesting to have a further study of the relationship between gene ranks and usefulness in the lower bound rules. We assume that the more important genes are more likely to be used in the globally significant rules. Figure 8 illustrates the chi-square based gene ranks and the frequencies of occurrence of 415 genes (which are involved in forming the top-1 rule groups) in the shortest lower bound rules of top-1 rule groups. As can be seen, most of the genes that occur frequently in the rules are those that are ranked high in the chi-square based ranking (most are ranked 700^{th} and above).

This includes six genes which occur more than 200 times in the discovered lower bound rules of the Prostate Cancer data: M61916 (408 times), W72186 (1775 times), AI635895 (887 times), X14487 (646 times), AB014519 (651 times), and AF017418 (997 times). Among the lower ranked gene, only gene Y13323 occurs for a large number of times (282).

This indicates that the genes of lower ranks generally serve as a certain supplementary information provider for the genes of higher ranks. The large proportion of lower-ranked genes also suggests their necessity for globally significant rules. Based on the experi-

ment, we suspect that the 7 most active genes, M61916, W72186, AI635895, X14487, AB014519, AF017418, and Y13323, are most likely to be correlated with the disease outcomes. Interestingly, gene AF017418 of rank 671 corresponds to MRG1 which has been reported to be useful in detecting glycosphingolipid antigen present in normal epithelium and superficial bladder tumor in patients with blood group A or AB, but absent in the invasive type of bladder (essentially prostate) tumor [17]. Also stated in [2, 10, 16, 5], MRG1 may function as a coactivator through its recruitment of p300/CBP in prostate cancer cell lines and stimulate glycoprotein hormone α-subunit gene expression. Gene AB014519 is related to Rock2 under certain cancer pathway known as the Wnt/planar cell polarity pathway [6]. X14487 is also a cancer-related gene for acidic (type I) cytokeratin. As reported in [21], X14487 shows consistently different expression levels in OSCC tissues and is one of the potential biomarkers for lymph node metastasis.

7. RELATED WORKS

Association rule mining has attracted considerable interest since a rule provides a concise and intuitive description of knowledge. It has already been applied to analyze biological data, such as [7, 8, 26]. Association rule can relate gene expressions to their cellular environments or categories, thus they can be used to build accurate classifiers on gene expression datasets as in [9, 18]. Moreover, it can discover the relationship between different genes, so that we can infer the function of an individual gene based on its relationship with others [7] and build the gene network.

Many association rule mining algorithms have been proposed to find the complete set of association rules satisfying user-specified constraints by discovering frequent (closed) patterns as the key step, such as [1, 11, 12, 22, 23, 24, 30, 31]. The basic approach of most existing algorithms is column enumeration in which combinations of columns are tested systematically to search for association rules. Such an approach is usually unsuitable for gene expression datasets since the maximal enumeration space can be as large as 2^i, where i is the number of columns and is in the range of tens of thousands for gene expression data. These high-dimensional bioinformatics datasets render most of the existing algorithms impractical. On the other hand, the number of rows in such datasets is typically very small and the maximum row enumeration space 2^m (m is the number of rows) is relatively small.

CARPENTER [23] is **the first row enumeration** algorithm which uses this special property of gene expression dataset to find closed patterns. However, it is still unable to work well when large numbers of patterns are generated in the case of entropy-based discretized gene expression data. Following CARPENTER, various groups have adopted the row enumeration [7] approach to find patterns in microarray datasets. In [29], the transposition of the gene expression table is proposed to facilitate efficient discovery of patterns in microarray datasets. This is essentially the same as the concept of a transposed table in CARPENTER. Likewise, in [25, 13, 14], the row-wise enumeration approach of CARPENTER is adopted and their pruning strategies essentially follow CARPENTER as well.

There are also many proposals about mining interesting rules with various interestingness measures. Some of them do a post-processing to remove those uninteresting rules, such as [20]. Such methods cannot work on gene expression data since it is usually too computationally expensive to mine the huge association rules from gene expression data. Other works [4, 28] try to mine interesting rules directly. The proposed algorithm in [4] adopts column enumeration method and usually cannot work on gene expression data as shown in the experiments of [6]. FARMER [6] is designed to mine interesting rule groups from gene expression data by row enumeration. But it is still very time-consuming on some entropy based discretized gene expression datasets. Although we also adopt the row enumeration strategy, our algorithm is different from FARMER: (1) we discover different kinds of rule groups; (2) we use top-k pruning; (3) we use a compact prefix-tree to improve efficiency while FARMER adopts in-memory pointer.

Our work is also related to those work on gene expression data classification. Traditional statistical and machine learning methods of classifying gene expression data usually select top-ranked genes (ranked according to measures such as gain ratio, chi-square and etc.) to alleviate the computational problems of high-dimensional data. However, such methods have two main problems. First, it is difficult to determine how many top-ranked genes to be used for classification model; Second, as observed in [18] and our experiments, low-ranked genes are often contained in significant rules that are sometimes necessary for perfect classification accuracy. Our proposed methods do not rely on the feature selection to reduce the number of dimensions for computational efficiency.

Our work is closely related to the classification methods [6, 9, 19] based on association rules. These algorithms first try to mine all rules satisfying minimum support and minimum confidence thresholds, and then sort and prune the discovered rules to get the classification rules. The high-dimensional gene expression data renders these algorithms impractical because of the huge number of discovered rules. Our proposed RCBT method addresses an open problem of these methods, that is to refine the classification of the test data when it matches no rules of the main classifier. This problem is especially severe since biologist may not accept the default class results for important cancer diagnosis. RCBT also improves CBA by aggregating the discriminating powers of a subset of rules selected by considering the gene discriminant ability. The IRG classifier [6] is similar to CBA except that it uses upper bound rules.

8. DISCUSSIONS AND CONCLUSIONS

Although it is true that current gene expression datasets have small number of rows, we may extend TopkRGS to other large datasets that are characteristic of both long columns and a large number of rows by utilizing column-wise mining first, then switching to row-wise enumeration in later levels to mine top-k covering rules in the partition formed by column-wise mining, and finally aggregating the top-k covering rules in all partitions.

This method could also help MineTopkRGS to deal with those datasets too large to fit in memory, as it is well known that some column-wise mining algorithms have linear scalability with dataset size. Another method for MineTopkRGS to deal with the memory limitation problem is to utilize the database projection (disk-based) techniques as suggested in [11].

In this paper, we proposed the concept of top-k covering rule groups for each row of gene expression data and an algorithm called MineTopkRGS to find the TopkRGS. Experiments showed that MineTopkRGS outperforms existing algorithms like CHARM, CLOSET+ and FARMER by a large order of magnitude on gene expression datasets.

This paper also show that the set of top-1 covering rule group for each row makes it feasible to build CBA classifier. Moreover, a

[6] http://www.csl.sony.co.jp/person/tetsuya/Pathway/Cancer-related/cancer-related.html, http://www.csl.sony.co.jp/person/tetsuya/Pathway/Cancer-related/Wnt/Wnt-planar

[7] Some groups refer to row enumeration as sample enumeration since the rows in the gene expression datasets are essentially tissue samples of patients.

classification method RCBT is proposed based on TopkRGS discovered by MineTopkRGS. Both kinds of classification demonstrated the usefulness of discovered TopkRGS. Our experiments showed RCBT has the highest average accuracy compared with CBA, IRG classifier, SVM and C4.5 family. Moreover, RCBT classifier is more understandable for biologists than SVM because rules themselves are intuitive.

Acknowledgment: We like to thank the anonymous ICDE'2005 reviewers for their comments which have help to improve this paper substantially.

9. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487–499, Sept. 1994.

[2] T. R. Anderson and T. A. Slotkin. Maturation of the adrenal medulla–iv. effects of morphine. *Biochem Pharmacol*, August 1975.

[3] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. MIT Press, 1998.

[4] R. J. Bayardo and R. Agrawal. Mining the most intersting rules. In *Proc. of ACM SIGKDD*, 1999.

[5] K. S. Bose and R. H. Sarma. Delineation of the intimate details of the backbone conformation of pyridine nucleotide coenzymes in aqueous solution. *Biochem Biophys Res Commun*, October 1975.

[6] G. Cong, A. K. H. Tung, X. Xu, F. Pan, and J. Yang. Farmer: Finding interesting rule groups in microarray datasets. In *23rd ACM International Conference on Management of Data*, 2004.

[7] C. Creighton and S. Hanash. Mining gene expression databases for association rules. *Bioinformatics*, 19, 2003.

[8] S. Doddi, A. Marathe, S. Ravi, and D. Torney. Discovery of association rules in medical data. *Med. Inform. Internet. Med.*, 26:25–33, 2001.

[9] G. Dong, X. Zhang, L. Wong, and J. Li. Caep: Classification by aggregating emerging patterns. *Discovery Science*, 1999.

[10] D. J. Glenn and R. A. Maurer. Mrg1 binds to the lim domain of lhx2 and may function as a coactivator to stimulate glycoprotein hormone α-subunit gene expression. *J Biol Chem*, 274, December 1999.

[11] J. Han and J. Pei. Mining frequent patterns by pattern growth: methodology and implications. *KDD Exploration*, 2, 2000.

[12] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, 2000.

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

[14] D. Jiang, J. Pei, and A. Zhang. A general approach to mining quality pattern-based clusters from gene expression data. In *DASFAA 2005. To Appear*.

[15] T. Joachims. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning*, 1999. http://svmlight.joachims.org/.

[16] M. Kasai, J. Guerrero-Santoro, R. Friedman, E. S. Leman, R. H. Getzenberg, and D. B. DeFranco. The group 3 lim domain protein paxillin potentiates androgen receptor transactivation in prostate cancer cell lines. *Cancer Research*, 63:4927–4935, August 2003.

[17] S. Kurimoto, N. Moriyama, K. Takata, S. A. Nozaw, Y. Aso, and H. Hirano. Detection of a glycosphingolipid antigen in bladder cancer cells with monoclonal antibody mrg-1. *Histochem J.*, 1995.

[18] J. Li and L. Wong. Identifying good diagnostic genes or genes groups from gene expression data by using the concept of emerging patterns. *Bioinformatics*, 18:725–734, 2002.

[19] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, 1998.

[20] B. Liu, W. Hsu, and Y. Ma. Pruning and summarizing the discovered associations. In *ACM KDD*, 1999.

[21] M. Nagata, H. Fujita, H. Ida, H. Hoshina, T. Inoue, Y. Seki, M. Ohnishi, T. Ohyama, S. Shingaki, M. Kaji, T. Saku, and R. Takagi. Identification of potential biomarkers of lymph node metastasis in oral squamous cell carcinoma by cdna microarray analysis. *International Journal of Cancer*, 106:683–689, June 2003.

[22] R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, 1998.

[23] F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. J. Zaki. Carpenter: Finding closed patterns in long biological datasets. In *Proc. 2003 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*, 2003.

[24] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. 7th Int. Conf. Database Theory (ICDT'99)*, Jan. 1999.

[25] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *ICDM*, pages 259–266, 2003.

[26] J. L. Pfaltz and C. M. Taylor. Closed set mining of biological data. *Workshop on Data Mining in Bioinformatics*, pages 43–48, 2002.

[27] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proc. 1996 Nat. Conf. Artificial Intelligence (AAAI'96)*, volume 1, pages 725–730, Portland, OR, Aug. 1996.

[28] R. Rastogi and K. Shim. Mining optimized association rules with categorical and numeric attributes. In *Int. Conf. on Data Engineering*, 1998.

[29] F. Rioult, J.-F. Boulicaut, B. Crémilleux, and J. Besson. Using transposition for pattern discovery from microarray data. In *DMKD*, pages 73–79, 2003.

[30] J. Wang, J. Han, and J. Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proc. 2003 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*, 2003.

[31] M. Zaki and C. Hsiao. Charm: An efficient algorithm for closed association rule mining. In *Proc. of SDM 2002*, 2002.