Scalable Rule-Based Gene Expression Data Classification

Mark A. Iwen [1], Willis Lang [2], Jignesh M. Patel [3]

[1] *Department of Mathematics,* [2,3] *EECS Department*
University of Michigan, USA
[1]markiwen@umich.edu, [2]wlang@eecs.umich.edu, [3]jignesh@eecs.umich.edu

***Abstract*— Current state-of-the-art association rule-based classifiers for gene expression data operate in two phases: (i) Association rule mining from training data followed by (ii) Classification of query data using the mined rules. In the worst case, these methods require an exponential search over the subset space of the training data set's samples and/or genes during at least one of these two phases. Hence, existing association rule-based techniques are prohibitively computationally expensive on large gene expression datasets.**

Our main result is the development of a heuristic rule-based gene expression data classifier called Boolean Structure Table Classification (BSTC). BSTC is explicitly related to association rule-based methods, but is guaranteed to be polynomial space/time. Extensive cross validation studies on several real gene expression datasets demonstrate that BSTC retains the classification accuracy of current association rule-based methods while being orders of magnitude faster than the leading classifier RCBT on large datasets. As a result, BSTC is able to finish table generation and classification on large datasets for which current association rule-based methods become computationally infeasible.

BSTC also enjoys two other advantages over association rule-based classifiers: (i) BSTC is easy to use (requires no parameter tuning), and (ii) BSTC can easily handle datasets with any number of class types. Furthermore, in the process of developing BSTC we introduce a novel class of boolean association rules which have potential applications to other data mining problems.

I. Introduction

Microarray technology allows biologists to simultaneously measure the expression of thousands of genes in a single experiment. This technology provides a unique tool to examine how a cell's gene expression pattern changes under various conditions. Microarray methods could also play a critical role in personalized medicine as they could be used to determine the unique genetic susceptibility of an individual to disease.

See Table I for a sample microarray dataset shown using the common discretized relational representation. In this table, each sample row consists of (i) a list of discretized genes and (ii) a class label. A gene is present in a sample row if the sample expresses the gene. The absence of a gene in a row implies that the gene is not expressed in that sample. Thus, the sample/gene expression relationships for relational microarray data are essentially boolean.

Leading associated rule-based methods such as Top-k [1], FARMER [2], CLOSET+ [3], and CHARM [4] which have been applied to microarray datasets aim to correlate gene expression patterns with the classification labels. For these algorithms the discovered correlations take the form of **association rules** [5]. For an example association rule, consider the data shown in Table I. Note that only the Cancer samples s_1 and s_2 express both genes g_1 and g_3. Based on this observation we can create the following association rule: $g_1, g_3 \Rightarrow \text{Cancer}$. This rule implies that if a query sample express both g_1 and g_3 (i.e., if g_1 and g_3's associated genes are both expressed in their associated expression intervals), then the query sample is likely to be of type Cancer. Hence, we can use this rule to *classify* query samples of unknown type as Cancer if they express both g_1 and g_3. Note that there is nothing special about the class label Cancer. After noticing that only Healthy sample s_5 expresses both g_5 and g_6, we can also create the association rule $g_5, g_6 \Rightarrow \text{Healthy}$.

Sample	**Expressed Genes**				**Class Label**
s_1	g_1	g_2	g_3	g_5	Cancer
s_2	g_1	g_3	g_6		Cancer
s_3	g_2	g_4	g_6		Cancer
s_4	g_2	g_3	g_5		Healthy
s_5	g_3	g_4	g_5	g_6	Healthy

TABLE I
Running Example of Microarray Data

In this paper we focus on association rule-based classifiers (hereafter referred to simply as rule-based classifiers) for gene expression data. We focus on rule-based classifiers for two reasons: (i) rule-based classifiers have been demonstrated to be more accurate for gene expression analysis than other methods [1], [2], [6], [7] such as SVM [8] and tree-based C4.5 family algorithms [9], and (ii) as opposed to other classifiers such as SVM, rule-based classifiers can produce rules that can be used to understand the class characteristics. However, rule-based methods are not scalable due to their high association rule mining costs. Although these rule mining costs are "one-time costs" in the sense that rules must only be mined once per training set, larger training data sets are being generated at an ever increasing rate. It is very challenging for any exponential time method to keep up. Consequently, in this paper, we focus on extending accurate association rule-based classification methods to larger data sets.

This paper develops a scalable rule-based classifier called Boolean Structure Table Classification (BSTC) for microarray datasets. Given a labeled training set, such as the example in Table I, BSTC *efficiently* builds an *accurate* classifier. The emphasis on accuracy is easy to appreciate and comes from BSTC being related to association rule-based methods. Hence, BSTC supports its classifications with intuitive rules. The emphasis on efficiency is also critical since large gene expression datasets are computationally taxing for existing association rule-based algorithms and, as successful microarray techniques fuel the growth of gene expression datasets, these methods will quickly become infeasible. In contrast, BSTC's space and runtime costs are only polynomial. Hence, BSTC is scalable to large data sets on which current association rule-based methods are challenged computationally.

In an attempt to control runtime many current association rule methods [1], [2], [10] utilize support based rule pruning. Using a large enough support cutoff does allow rule mining to finish more quickly, but doesn't completely resolve the issue. If the user sets the support cutoff too small, then the rule mining step can take prohibitively long to complete. On the other hand, setting the support cutoff too high excludes the generation of important high-confidence lower-support rules [11]. In order to not miss too many important rules the user can't set the support cutoff too high. The end result is that in practice support cutoffs are difficult and time intensive to tune. In contrast, BSTC is fast and easy to use.

In addition, to the best of our knowledge all current association rule-based classifiers for gene expression data only handle datasets with two class labels. Although our example Table I data contains just two class labels, in practice microarray data can contain an arbitrary, though small, number of class types. Unlike previous association rule-based classifiers, BSTC easily generalizes to datasets with more than two class types.

To develop an accurate, scalable, multi-class, and easy to use rule-based classifier we carefully considered the underlying primitives that power association rule-based methods. These methods use **conjunctive association rules (CARs)**, where the rule antecedent is restricted to being a conjunction of terms. In contrast, we approach this problem by relaxing the types of rules to an important and larger subset of the more general class of boolean association rules (BARs). We develop a novel method for compactly storing these BARs in a simple data structure called a Boolean Structure Table (BST). BSTs can then be used for BAR generation and classification. BST classification (BSTC) collectively considers many simple BARs with 100% confidence in bulk. Because the rules are simple BSTC avoids extensive rule mining. Furthermore, considering rules in bulk keeps the computational cost low.

The main contributions of this paper are:

1) We propose a new polynomial time and space rule-based classifier for gene expression data analysis that is accurate, scalable, easy to use, and generalizable to multi-class classification.
2) We extensively evaluate our method against the current leading association rule-based method (RCBT [1]), and show that our method is orders of magnitude faster on large datasets while maintaining high classification accuracy.
3) We introduce a subclass of more general boolean association rules and relate them to existing CARs.

The remainder of this paper is organized as follows: First, in Section II we formalize the concept of BARs. Then in Section III, we define a concept called Boolean Structure Tables (BSTs) which are related to an important class of BARs. Section IV provides a polynomial time and parameter-free classifier based on BSTs. Section V presents an extensive empirical evaluation of our classifier. Finally, Section VI discusses related work, and Section VII briefly presents our conclusions and directions for future work.

II. Preliminaries

We work with the following type of data: We are given a finite set G of genes and N collections of subsets from G. These N collections are disjoint and represented as $C_1 = \{s_{1,1}, \ldots, s_{1,m_1}\}$, …, $C_N = \{s_{N,1}, \ldots, s_{N,m_N}\}$. Each C_i is called a **class type** or **class label**. Furthermore, we will refer to each set $s_{i,j} \subset G$ as a **sample** and every element $g \in G$ as a **gene**. We denote the total set of samples by $S = \cup_{i=1}^{N} C_i$. If $g \in s_{i,j}$ we will say that sample $s_{i,j}$ **expresses** gene g. Otherwise, if $g \in G$ and $g \notin s_{i,j}$ we will say that sample $s_{i,j}$ doesn't express gene g. Similarly, we say that sample s is of class type C_i if and only if C_i contains $s \subset G$. Consider the Table I data. Here we have samples $S = \{s_1, s_2, s_3, s_4, s_5\}$ and genes $G = \{g_1, g_2, g_3, g_4, g_5, g_6\}$. Furthermore, we have $N = 2$ classes: $C_1 = \mathsf{Cancer} = \{s_1, s_2, s_3\}$ and $C_2 = \mathsf{Healthy} = \{s_4, s_5\}$.

Given such relational training data, a **conjunctive association rule (CAR)** is any element of $2^G \text{x} \{1, \ldots, N\}$. A CAR $g_{j_1}, \ldots, g_{j_r} \Rightarrow n$ can be interpreted as follows: "If a query sample s contains all genes $g_{j_1}, \ldots, g_{j_r}$ then it should be grouped with class type C_n." Naturally, of the $2^{|G|} \cdot N$ possible association rules some are more useful than others. The following standard definitions were introduced in [5] to compare association rules:

Support: The support of a CAR $g_{j_1}, \ldots, g_{j_r} \Rightarrow n$, called $supp[g_{j_1}, \ldots, g_{j_r} \Rightarrow n]$, is:

$$|\{s_{n,j} \text{ s.t. } \{g_{j_1}, \ldots, g_{j_r}\} \subset s_{n,j}, 1 \leq j \leq m_n\}|.$$

Confidence: The confidence of a CAR $g_{j_1}, \ldots, g_{j_r} \Rightarrow n$ is:

$$\frac{supp[g_{j_1}, \ldots, g_{j_r} \Rightarrow n]}{|\{s_{i,j} \text{ s.t. } \{g_{j_1}, \ldots, g_{j_r}\} \subset s_{i,j} \forall i, j\}|}.$$

Consider the CAR $g_1, g_3 \Rightarrow \mathsf{Cancer}$ for our running example in Table I. We can see that the example CAR has a support of 2 since only two Cancer samples, $s1$ and $s2$, contain both g_1 and g_3. Furthermore, we can see that the example CAR has confidence 1 (or 100%) since no healthy samples contain both g_1 and g_3.

A. Boolean Association Rules

For any sample s and gene g_i, $1 \leq i \leq n = |G|$, let $s[g_i] \in \{0, 1\}$ represent whether or not sample s expresses gene g_i. Furthermore, define $s[-g_i]$ to be $-s[g_i]$ $\forall g_i \in G$. Now suppose that $B(x_1, \ldots, x_n)$ is a Boolean expression whose value depends on some subset of $\{x_1, \ldots, x_n\}$. We can evaluate B to true or false given a sample s by computing $B(s[g_1], \ldots, s[g_n])$. For example, consider the boolean expression:

$$\hat{B}(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1 \wedge x_3) \vee (x_2 \wedge x_4).$$

	s1	s2	s3
g1	●	●	
g2	(s4: g1)		(s4: -g3, -g5)
g3	(s4: g1) & (s5: -g4, -g6)	(s4: -g2, -g5) & (s5: -g4, -g5)	
g4			(s5: -g3, -g5)
g5	(s4: g1) & (s5: -g4, -g6)		
g6		(s5: -g4, -g5)	(s5: -g3, -g5)

Fig. 1. Example BST for the Cancer Class

Using Table I we can evaluate

$$\hat{B}(s_1[g_1], s_1[g_2], s_1[g_3], s_1[g_4], s_1[g_5], s_1[g_6]) \qquad (1)$$

to be $(1 \wedge 1) \vee (1 \wedge 0) = 1$. Note that $\hat{B}$ will only evaluate the Table I Cancer samples to $True$.

For a given class set C_i and boolean expression B we can create a **Boolean association rule (BAR)** of the form $B \Rightarrow C_i$. The interpretation of any such BAR, $B \Rightarrow C_i$, is "if $B(s[g_1], \ldots, s[g_n])$ evaluates to true for a given sample s, then s should belong to class C_i." From this point on we will work with the following generalized definitions of support and confidence:

Support: The support of any BAR $B \Rightarrow C_i$, represented as $supp(B \Rightarrow C_i)$, is:

$$\{\text{samples } s \in C_i \text{ s.t. } B(s[g_1], \ldots, s[g_n]) \text{ evaluates to true}\}.$$

The corresponding numerical support value of $B \Rightarrow C_i$ is denoted as $|supp(B \Rightarrow C_i)|$.

Confidence: The confidence of a BAR $B \Rightarrow C_i$ is

$$\frac{|supp(B \Rightarrow C_i)|}{|\{\text{samples } s \text{ s.t. } B(s[g_1], \ldots, s[g_n]) \text{ evaluates to true}\}|}$$

For CARs these definitions coincide with the CAR definitions of support and confidence found in [5], [12]. Hence, they are natural generalizations of the previous definitions (see section III-C).

Consider our example boolean expression $\hat{B}$ in terms of Table I. We can see that the BAR $\hat{B} \Rightarrow$ Cancer (shown in Eq. 1) has support 3 and confidence 1.

III. BSTs and BARs

The discussion in this section will focus on tables for each class C_i. These tables, called Boolean Structure Tables (BSTs), will form the basis for our classification method. In order to motivate the utility of BSTs for classification, we will present their close relationship to a special category of BARs which, in turn, will be related back to CARs. Through this discussion we will demonstrate that BSTs contain all the information of the high confidence CARs already known to be valuable for microarray data classification.

Algorithm 1 Create-BST: The BST Creation Algorithm

```
1: Input: Finite set of Genes $G$, set of samples $S$, Class $C_i$
2: Output: The BST Table for Class $C_i$.
3: for all $(c,h) \in C_i \times S - C_i$ do
4:    initialize a pointer $\leftarrow$ NULL
5: end for
6: for all $(g,c) \in G \times C_i$ s.t. $g \in c$ and $g \notin \cup_{h \in S - C_i} h$ do
7:    Set $BST(g,c) \leftarrow$ Black Dot
8: end for
9: for all $(g,c,h) \in G \times C_i \times S - C_i$ s.t. $g \in c$ and $g \in h$ do
10:   if pointer $(c,h) \neq$ NULL then
11:      push a copy of $(c,h) \rightarrow BST(g,c)$
12:   else
13:      $L = \{g \in G$ s.t. $g \in h$ & $g \notin c\}$
14:      if $L \neq \emptyset$ then
15:         $(c,h) \leftarrow L$'s address
16:      else
17:         $L = \{g \in G$ s.t. $g \notin h$ & $g \in c\}$
18:         $(c,h) \leftarrow L$'s address
19:      end if
20:   end if
21:   Push a copy of $(c,h) \rightarrow BST(g,c)$.
22: end for
```

A. Boolean Structure Tables

A **Boolean Structure Table (BST)** $T(i)$ is a two dimensional table, $T(i) = G \times C_i$, where each table entry refers to a maximum of $|S| - |C_i|$ lists of up to $|G|$ genes each. For every C_i the associated BST, $T(i)$, will require $O\left((|S| - |C_i|) \cdot |G| \cdot |C_i|\right)$ space and can be constructed with the same time complexity via Algorithm 1.

When the Algorithm 1 is run on the Table I example input and for class Cancer, the Boolean Structure Table shown in Figure 1 is produced. In Figure 1 a black dot at location (g,s) indicates that no healthy samples express gene g but some cancerous sample does. A cell (g,s) is left blank only if sample s didn't express gene g. If (g,s) contains a list of the form $(h : -g_1, \ldots, -g_n)$ it means that s may be distinguished from sample h by the non-expression of any one of genes g_1 through g_n. Similarly, if (g,s) contains a list of the form $(h : g_1, \ldots, g_n)$ it means that s may be distinguished from sample h by the expression of any one of genes g_1 through g_n. Such lists will hereafter be referred to as **exclusion lists**.

Note that there is no reason why the BST in Figure 1 was created for the Cancer class. We can just as easily build a BST for the Healthy class using the example shown in Table I. In general, if a relational gene expression dataset contains N classes, we can construct N different BSTs for the data set (one for each class).

1) Runtime Complexity for BST Creation: We can see that the total time to construct BSTs via Algorithm 1 for all of $C_1, \ldots, C_N$ is $O\left(\sum_{i=1}^{N}(|S| - |C_i|) \cdot |C_i| \cdot |G|\right)$. Given that the class sets C_i are all disjoint, we have $\sum_{i=1}^{N}(|S| - |C_i|) \cdot |C_i| \cdot |G| \leq \sum_{i=1}^{N} |S| \cdot |C_i| \cdot |G| = |S|^2 \cdot |G|$. Hence, BSTs can be constructed for all C_is in time $O(|S|^2 \cdot |G|)$.

Gene g_1: (g_1 expressed) $\Rightarrow$ Cancer.
Gene g_2: (g_2 expressed AND [EITHER (g_1 expressed) OR (either g_5 or g_3 not expressed)]) $\Rightarrow$ Cancer.
Gene g_3: (g_3 expressed AND [EITHER {(g_1 expressed) AND (either g_4 or g_6 not expressed)} OR { (either g_2 or g_5 not expressed) AND (either g_4 or g_5 not expressed)}]) $\Rightarrow$ Cancer.
Gene g_4: (g_4 expressed AND [either g_5 or g_3 not expressed]) $\Rightarrow$ Cancer.
Gene g_5: (g_5 expressed AND [g_1 expressed AND (either g_4 or g_6 not expressed)]) $\Rightarrow$ Cancer.
Gene g_6: (g_6 expressed AND [(either g_4 or g_5 not expressed) OR (either g_3 or g_5 not expressed)]) $\Rightarrow$ Cancer.

Fig. 2. Gene Row BARs with 100% Confidence Values.

B. BST Generable BARs

We view every BST cell, (g, c), as an atomic 100% confident BAR. For example, Figure 1's $(g3, s1)$-cell corresponds to the BAR

$g3$ expressed AND $g1$ expressed AND (either $g4$ or $g6$ not expressed) $\Rightarrow$ Cancer.

We refer to this rule as the Figure 1 BST's $(g3, s1)$-**cell rule**. Note that the cell rule is both (i) 100% confident, and (ii) supported by sample $s1$. Throughout the remainder of this section we will use such cell rules as atomic building blocks to construct more complicated BARs. Furthermore, in Section IV, we will directly employ BST cell rules to build a new classifier called BSTC.

1) Mining More Complicated BST BARs: Let $T(i)$ be a BST for sample type C_i. We can view each row of $T(i)$ as a 100% confident BAR by combining the row's cell rules. To see this, choose any $g_j \in G$ and consider the CAR $g_j \Rightarrow C_i$. This CAR can be augmented with exclusion list clauses from each of $T(i)$'s g_j-row cells. The result will be a BAR with 100% confidence which is logically equivalent to a disjunction of $T(i)$'s g_j-row cell rules. See Figure 2 for the gene row BARs which result from applying this idea to the BST in Figure 1.

For the remainder of this paper we will restrict our attention to BARs that may be generated by taking conjunctions of BST cell rule disjuncts. Henceforth we simply refer to these as BARs. It is very important to notice that all such BARs have a special form: Their antecedents consist of a CAR antecedent ANDed with a disjunction of BST exclusion list clause conjunctions. Consider the BAR for gene g_6 in Figure 2. Gene g_6's rule antecedent consists of a CAR antecedent, g_6, conjoined to a disjunction of the Figure 1 exclusion list clauses: (either g_4 or g_5 not expressed) and (either g_3 or g_5 not expressed). Progressive polynomial time algorithms for mining more complicated BARs via a BST can be found in an extended version of this paper [13].

C. BARs Relationships to CARs

Let $R \Rightarrow C_i$ be any 100% confident BST created BAR containing exclusion clauses for non-C_i samples $h_1, \dots h_m$. Removing all exclusion list clauses related to $\{\hat{h}_1, \dots, \hat{h}_p\} \subset \{h_1, \dots, h_m\}$, $p \leq m$, will create a new boolean association rule, $\hat{R} \Rightarrow C_i$, with support = $supp(R \Rightarrow C_i)$ and confidence $\geq \frac{|supp(R \Rightarrow C_i)|}{|(supp(R \Rightarrow C_i)|+p}$. Let's consider the g_3-row BAR from our running example:

(g_3 expressed AND [EITHER {(g_1 expressed) AND (either g_4 or g_6 not expressed)} OR { (either g_2 or g_5 not expressed) AND (either g_4 or g_5 not expressed)}]) $\Rightarrow$ Cancer.

It has 100% confidence and support $\{s1, s2\}$. Now, if we remove all exclusion list clauses related to sample row $s5$ we end up with the boolean association rule:

(g_3 expressed AND [EITHER (g_1 expressed) OR (either g_2 or g_5 not expressed)] $\Rightarrow$ Cancer.

This new rule has support $\{s1, s2\}$ and a confidence of $\frac{|\{s1,s2\}|}{|\{s1,s2,s5\}|} = \frac{2}{3}$. The preceding observation leads us to the following theorem:

Theorem 1: Let D be a relational data set containing s samples no two of which are the same (i.e. no two sample rows express the exact same set of genes). Then, there exists a pure conjunction B implying a class type C (i.e., a CAR) with confidence c and support $supp$ for $D \iff$ there exists a 100% confident BST generated BAR $\hat{B} \Rightarrow C$ for D that: (i) has $supp(\hat{B} \Rightarrow C) = supp$, and (ii) contains exclusion list clauses actively excluding $(\frac{1}{c} - 1)|supp|$ non-C samples.

Proof: See the extended version of this paper [13]. ■

Theorem 1 tells us how we can get CARs from BARs. Furthermore, it says 100% confident BARs with large support and a small number of excluded samples are equivalent to high support/confidence CARs. Hence, genes that show up in many high confidence, high support CARs will also be prevalent in many 100% confident BARS with high support and a low number excluded samples. Most importantly, we see that all high confidence CARs (which tend to be good classifiers) have closely related BAR counterparts.

IV. BST-Based Classification

In principal, 100% confident BST-generable BARs should be sufficient for classification because they contain at least as much information as all generable CARs do (see section III-C). Indeed, beyond what CARs with similar support are capable of, 100% confident BARs supply us with "unpolluted" ground truth. Thus, it's not too surprising that the class of BST-generable BARs we've looked at so far will be enough to enable highly accurate classification.

Let C_i be a class set of interest and $T(i)$ be the BST for class C_i constructed from the given training data. From section III-B we can see that all BST generable BARs for class C_i are created by combining $T(i)$ cell rules. Thus, we expect that by restricting our attention to the $O(|G| \cdot |C_i|)$ atomic $T(i)$ cell rules we will be, in some sense, still considering all $T(i)$ generable BARs for C_i. Our new scalable classifier, the

Algorithm 2 BST Cell rule quantized Evaluation (BSTCE)

```
1: Input: Class $C_i$, BST for the class $T(i)$, Samples $S$, Query sample $Q$
2: Output: Classification value
3: for all non-empty exclusion lists $e$ in $T(i)$'s cells do
4:     $V_e \leftarrow \frac{|\{\hat{g} \in e \text{ s.t. } Q[\hat{g}]=1\}|}{|e|}$
5: end for
6: for all $(g,s) \in \{g \in G \text{ s.t. } Q[g]=1\} \times C_i$ do
7:     if $T(i)(g,s)$ contains a ● then
8:         $T(i)[g][s] \leftarrow 1$
9:     else
10:        $T(i)[g][s] \leftarrow$ Min $\{V_e \text{ s.t. } e \text{ is in } T(i)(g,s)\}$
11:    end if
12: end for
13: for all non-blank sample columns $s \in T(i)$ do
14:     $V_s \leftarrow$ Mean of non-blank $T(i)[\star][s]$ values
15: end for
16: Return the Mean of step 16's $V_s$ values
```

Boolean Structure Table Classifier (BSTC), capitalizes on this line of thought by ignoring BAR generation and focusing exclusively on atomic BST cell rules.

A. BSTC Overview

Let Q be a test/query gene expression data sample and $T(i)$ be a BST for class set C_i. BSTC is a heuristic rule-based classifier motivated by standard Boolean formula arithmetization techniques [14] such as those employed in fuzzy satisfiability [15]. By using these ideas we can avoid the highly costly process of support/confidence based association rule mining. Instead of explicitly generating rules, BSTC decides (heuristically), for all C_i, how well Q collectively satisfies $T(i)$'s atomic cell rules. BSTC then classifies Q as the sample class whose BST has the highest expected atomic rule satisfaction level from Q.

Intuitively, we expect BSTC to be accurate because it approximates the results of CAR-based classification: Suppose that a high support/confidence CAR exists which classifies our query sample Q as class C_j. This will only happen if all the CAR's antecedent genes, AG, appear in both (i) Q and, (ii) most of the training samples in the CAR's consequent class C_j. Let $T(j)$ be the BST for class C_j. Because of (ii) most of $T(j)$'s sample columns must contain cell entries for all the AG genes. Furthermore, all $T(j)$'s AG cell entries will have few exclusion lists in common (by Theorem 1). Hence, $T(j)$'s expected atomic rule satisfaction level from Q (i.e., Q's classification value) should be heavily influenced (increased) by the AG rows and their few shared lists.

B. BST Cell Rule Satisfaction

As above, let Q be a test/query gene expression data sample and $T(i)$ be a BST for class set C_i. Algorithm 2, BSTCE, gives BSTC's method of calculating the level that Q satisfies a given atomic $T(i)$ cell rule. We next explain the rational behind BSTCE.

We know that each $T(i)$ (g,s)-cell exclusion list, L, corresponds to a disjunction in $T(i)$'s (g,s)-cell rule. Hence, if Q satisfies any one negation/inclusion in L, Q will satisfy L. However, if Q expresses most of its genes in common with L's associated non-C_i sample we assume it's probably not of type C_i (i.e., Q is weakly excluded). Hence, we use BSTCE's line 4 ratio to approximate the probability that L correctly excludes Q from being of L's associated sample's class.

In order for the (g,s)-cell rule to be satisfied, all of (g,s)'s exclusion lists must be satisfied (i.e., logical AND). If independence of each exclusion list's correct classification is assumed it's natural to multiply all of (g,s)'s list's probabilities. We don't assume independence and use a $\min$ instead (line 10). Finally, recall that all black dots in $T(i)$ correspond to genes expressed only in class C_i samples. If Q expresses a black dot gene it automatically satisfies all that gene's non-empty $T(i)$ cell rules. Hence, black dots are all assigned values of 1 in BSTCE's line 8.

Once we have used BSTCE lines 1-12 to calculate Q's classification values (i.e., $T(i)$'s atomic rule satisfaction levels from Q) for each relevant simple (g,s)-cell rule, we are nearly finished. We have all the values required to judge Q's similarity to $T(i)$ via an expectation calculation. For the sake of $T(i)$'s expectation calculation, all that is left to do is imagine choosing a relevant simple $T(i)$ rule at random and then using it to classify Q. To randomly select a (g,s) rule we first imagine selecting a non-empty $T(i)$ sample column uniformly at random and then picking a cell-rule from that column uniformly at random. The expected probability of correctly classifying Q with $T(i)$ via this method (which heuristically is proportional to $T(i)$'s expected satisfaction level from Q) is then calculated by averaging the approximate cell rule satisfaction levels down each non-empty sample column (line 14) and then averaging the resulting non-empty sample averages (line 16).

Algorithm 3 The BSTC Algorithm

```
1: Input: BSTs for all dataset classes $T(1),\ldots,T(N)$, Query sample $Q$
2: Output: Classification for query sample $Q$
3: for all $i \in \{1,\ldots,N\}$ do
4:     $CV(i) \leftarrow \text{BSTCE}(T(i),Q)$
5: end for
6: Return $min\{i|CV(i) = max\{CV(1),\ldots,CV(N)\}\}$
```

C. BSTC Algorithm

Suppose we are given relational training data D containing sample rows S split up into disjoint class sets $C_1,\ldots,C_N$. BSTC uses D to construct N BSTs, $T(1),\ldots,T(N)$. Next, let G be the union of the elements contained in each sample row of D (i.e. the gene set of D) and let Q be a query sample with expression information regarding G. BSTC will use the BSTCE algorithm to classify Q as being the C_i with smallest i such that BSTCE($T(i),Q$) = $max\{$BSTCE$(T(j),Q)|0 \leq j \leq N\}$. See Algorithm 3 for the BSTC algorithm.

Note that there is no reason why N must be 2. *BSTC easily generalizes to datasets containing more than two class labels.*

1) BSTC Runtime: As noted in section III-A.1 it takes time and space $O(|S|^2 \cdot |G|)$ to construct all the BSTs

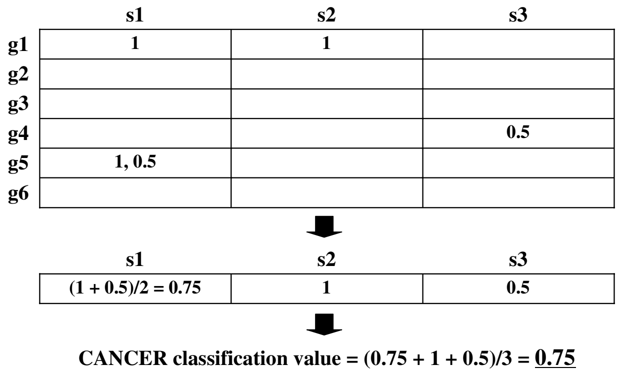

Fig. 3. BSTC cell rule Evaluation Example

$T(1), \ldots, T(N)$. Thus, BSTC requires time and space $O(|S|^2 \cdot |G|)$ to construct. Furthermore, during classification BSTC must calculate BSTCE($T(i), Q$) for $1 \leq i \leq N$. BSTCE (Algorithm 2) runs in $O\left((|S| - |C_i|) \cdot |G| \cdot |C_i|\right)$ time per query sample. Therefore the BSTC worst case evaluation time is also $O(|S|^2 \cdot |G|)$ per query sample. See Section VII for more on BSTC's per-query classification time.

2) Biological Meaning of BSTC Classification: Association rules mined from gene expression data provide an intuitive representation of biological knowledge (e.g., the expression of certain genes implies cancer). Hence, CAR-based classifiers have the desirable ability to justify each non-default consequent class query classification with the biologically meaningful CAR(s) the query satisfied. BSTC, being rule-based and related to CAR-classifiers, also has this property.

BSTC can support it's query classifications with BARs of any user specified complexity. Most simply, for any given query sample Q and $c \in (0, 1]$, BSTC can justify it's classification of Q as class C_i by reporting all $T(i)$ atomic cell rules with satisfaction levels $\geq c$. Note that returning this information requires no additional per-query classification time. Also note that section III-B.1 methods can be used to mine more complex highly satisfied BARs if desired.

D. BSTC Example

Consider our running example from Table I. In order to construct BSTC we must construct both $T(\textsf{Healthy})$ and $T(\textsf{Cancer})$ (shown in Figure 1). Once both BSTs have been constructed we can begin to classify query samples. Suppose, for example, we are given the query sample $Q = \{g1$ expressed, $g2$ not expressed, $g3$ not expressed, $g4$ expressed, $g5$ expressed, $g6$ not expressed$\}$. To classify this query we must first calculate BSTCE($T(\textsf{Cancer}), Q$) and BSTCE($T(\textsf{Healthy}), Q$).

The evaluation of BSTCE($T(\textsf{Cancer}), Q$) proceeds as follows: Since our query sample Q expresses gene g_5 we can see that we must, for example, determine the fraction of both of the (g_5, s_1)-cell's exclusion lists satisfied by Q. The (g_5, s_1)-cell's $(s_4 : g_1)$ exclusion list is totally satisfied since Q expresses g_1. Hence, it gets a value of 1. However, the $(s_5 : -g_4, -g_6)$ exclusion list is only half satisfied since, although Q doesn't express g_6, Q does expresses g_4. Thus, in total, we only consider half of the simple (g_5, s_1)-cell rule to be satisfied (i.e. the s_5 exclusion list is the weakest link). Continuing to use BSTC's approximation scheme for the expected probability of Q's correct Cancer classification via the Figure 1 BST we obtain Figure 3. Note that only Figure 3 gene rows corresponding to genes expressed in Q are non-empty.

If we now evaluate BST-EXPECT($T(\textsf{Healthy}), Q$) we obtain a final value of $\frac{3}{8}$. To finish, BSTC will compare Q's Cancer classification value of $\frac{3}{4}$ to Q's Healthy classification value of $\frac{3}{8}$ and conclude that Q is most probably Cancer. Hence, Q will be classified as Cancer.

Dataset	# Genes	Class 1 label	Class 0 label	# Class 1 samples	# Class 0 samples
ALL/AML (**ALL**)	7129	ALL	AML	47	25
Lung Cancer (**LC**)	12533	MPM	ADCA	31	150
Prostate Cancer (**PC**)	12600	tumor	normal	77	59
Ovarian Cancer (**OC**)	15154	tumor	normal	162	91

TABLE II
GENE EXPRESSION DATASETS

V. EXPERIMENTAL EVALUATION

All experiments reported here were carried out on a 3.6 GHz Xeon machine with 3GB of memory running Red Hat Linux Enterprise 4. For our empirical evaluation we use four standard real microarray datasets [16] Table II lists the dataset names, class labels, and the number of samples of each class. All discretization was done using the entropy-minimized partition [17] as in [1].

Executables for both RCBT and Top-k were provided by the authors of [1]. In all experiments, the Top-k rule generator was used to generate rule groups for RCBT. Unless otherwise noted we ran both Top-k and RCBT with the author suggested parameter values (i.e., support = 0.7, k = 10, nl = 20, 10 RCBT classifiers). Hence, while generating rules for RCBT we used Top-k with a minimum support value of 0.7 and found the 10 most confident covering rule groups (i.e. k = 10). Furthermore, during classification we used RCBT with the suggested 10 classifiers (1 primary and 9 standby). Finally, nl, the number of lower bound rules to use for classification per Top-k mined rule group, was set equal to 20. We coded BSTC in C++.

A. Preliminary Experiments

Each of Table II's four gene expression datasets comes with a clinically determined training set. The authors of [1] provided us with their discretizations of these four datasets. We ran BSTC on their discretizations and BSTC matched RCBT's reported mean accuracy (about 96%) outperforming CBA (87%), IRG (81%), Weka 3.2 (C4.5 family single tree (74%), bagging (78%), boosting(74%)), and SVM$^{\text{light}}$ 5.0 (93%) in reported mean performance [1].

To compare BSTC and RCBT with the most recent R e1071 package SVM implementation [18] and randomForest version 4.5 [19] we rediscretized the four datasets and reran

Dataset	# Class 1 Training Samples	# Class 0 Training Samples	Genes After Discr.	BSTC Accur.	RCBT Accur.	SVM Accur.	random-Forest Accuracy
ALL	27	11	866	82.35%	91.18%	91.18%	85.29%
LC	16	16	2173	100%	97.99%	93.29%	99.33%
PC	52	50	1554	100%	97.06%	73.53%	73.53%
OC	133	77	5769	100%	97.67%	100%	100 %
Avg. Accuracy				95.59%	95.98%	89.5%	89.54%

TABLE III
RESULTS USING GIVEN TRAINING DATA

BSTC/RCBT. To keep comparisons fair we ran SVM and randomForest on the same genes selected by our entropy discretization except with their original undiscretized gene expression values. SVM was run with its default radial kernel. We ran randomForest 10 times with its default 500 trees for ALL, LC, and OC and its accuracy was constant. For PC we had to increase randomForest's number of trees to 1000 before its accuracy stabilized over the 10 runs.

Table III contains the number of class 0/1 samples in the clinically determined training set, the number of genes selected by our entropy discretization, and our experimental results. As shown in this table, the overall average accuracies of BSTC and RCBT are again best at about 96% each. When compared against RCBT, SVM, and randomForest on the individual tests we can see that BSTC is alone in having 100% accuracy on the majority of datasets.

However, BSTC's performance on the preliminary AML/ALL dataset test is relatively poor. This is likely due to over fitting. Every error BSTC made mistook a class 0 (AML) test sample for a class 1 (ALL) test sample (i.e., all errors were made in this same direction). And, the ALL training data has both (i) about 2.5 times as many class 1 samples as class 0 samples, and (ii) a small number of total samples/genes. When the training set is more balanced and the number of samples/genes is larger we can expect that cancellation of errors will tend to neutralize/balance any over fitting effects in BSTC. And, BSTC is a method meant primarily for large training sets where CAR-mining is prohibitively expensive. As we will see below in Section V-B.1, BSTC's performance is much better for larger AML/ALL training set sizes.

B. Holdout Validation Studies

Holdout validation studies make comparisons less susceptible to the choice of a single training dataset and provide performance evaluations that are likely to better represent program behavior in practice. We next present results from a thorough holdout validation study completed using 100 different training/test sets from each of the ALL, LC, PC, and OC data sets. For these holdout validation tests we benchmark BSTC against Top-k/RCBT because (i) BSTC/RCBT perform best in our preliminary experiments, (ii) Top-k/RCBT is the fastest/most accurate CAR-based classifier for microarray data, and (iii) we are interested in BSTC's CAR-related vs Top-k/RCBT's CAR-based scalability.

For the holdout validation study we generated training sets of sizes 40%, 60%, and 80% of the total samples. Each training set was produced by randomly selecting samples from the original combined dataset. We then used the standard R dprep package's entropy minimized partition [17] to discretize the selected training samples. Finally, the remaining dataset samples were used for testing the two classifiers after rule/BST generation on the randomly selected training data. For each training set size we produced 25 independent tests. In addition to these training sets, we created an additional 25 1-x/0-y tests. To create these tests we chose training data by randomly selecting x class 1 samples and y class 0 samples to be used as training data. As before, the remaining samples were then used to test both classifiers. For each dataset the x and y values are chosen so that the resulting 25 classification tests have the exact same training/test data proportions as the single related dataset test reported in section V-A. For each training set size we plot our results using a boxplot.

Boxplot Interpretation: Each boxplot that we show in this section can be interpreted as follows: The median of the measurements is shown as a diamond, and a box with boundaries is drawn at the first and the third quartile. The range between these two quartiles is called the inter-quartile range (IQR). Vertical lines (a.k.a. "whiskers") are drawn from the box to indicate the minimum and the maximum value, unless outliers are present. If outliers are presents, the whiskers only extend to $1.5 \times IRQ$. The outliers that are near (i.e. within $3 \times IRQ$ are drawn as an empty circle, and further outliers are drawn using an asterisk.

1) ALL/AML (ALL) Experiment: Figure 4 shows the classification accuracy for the ALL/AML dataset. As can be seen in this figure, BSTC and RCBT have similar accuracy across the ALL/AML tests as a whole. BSTC outperforms RCBT in terms of median and mean accuracy on the 40% and 80% training set sizes while RCBT has better median/mean accuracy on the 1-27/0-11 training size tests. And, both classifiers have the same median on the 60% training set size. Over the 100 ALL/AML tests we see that BSTC has a mean accuracy of 92.13% while RCBT has a mean accuracy of 91.39% (they are very close).

It's noteworthy that BSTC is 100% accurate on the majority of 80% training size tests. However, BSTC appears to have slightly higher variance than RCBT on all but the 40% training tests. Considering all the results together both BSTC and RCBT have essentially equivalent classification accuracies on the ALL/AML dataset.

2) Lung Cancer (LC) Experiment: The results for the Lung Cancer dataset are reported in Figure 5. Here, again, both BSTC and RCBT have similar classification behavior. RCBT has higher mean and median accuracies on the 40% and 60% tests while BSTC outperforms RCBT on the 1-16/0-16 tests. Meanwhile, both classifier have the same median on the 80% training test. Over all 100 LC tests we find that BSTC has a mean accuracy of 96.32% while RCBT has a mean accuracy of 97.08% (again, they are very close).

As before, BSTC is alone in having 100% accuracy more

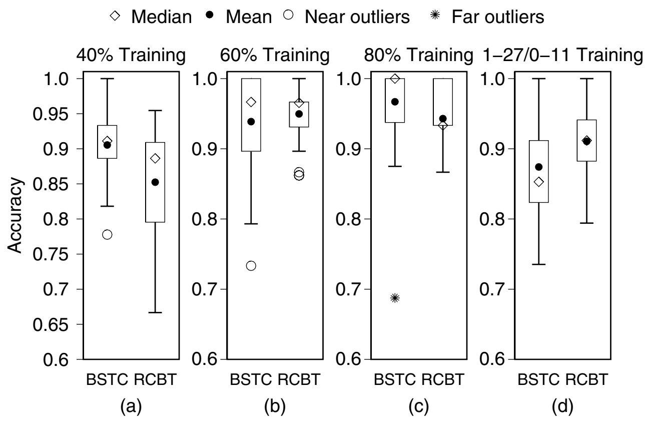

Fig. 4. ALL Holdout Validation Results

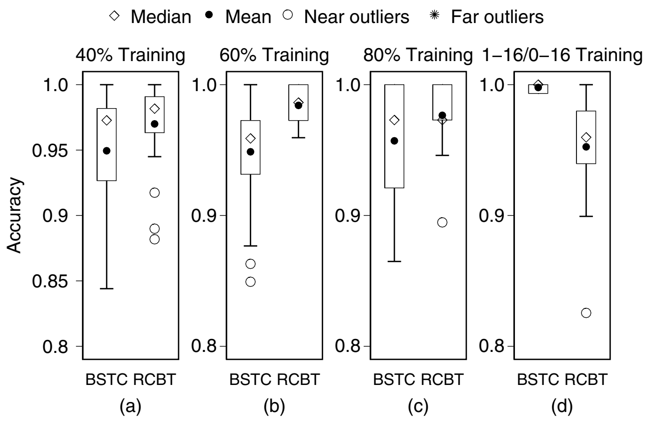

Fig. 5. LC Holdout Validation Results

then half the time for any training set size (see Figure 5 (d)). However, RCBT has smaller variance for 3 of the 4 training set sizes. Therefore, as for the ALL/AML data set, both BSTC and RCBT have about the same classification accuracy on LC.

3) Prostate Cancer (PC) Experiment: RCBT begins to run into a computational difficulties on PC's larger training set sizes. This is because before using a Top-k rule group for classification RCBT must first mine nl lower bound rules for the rule group. RCBT accomplishes rule group lower bound mining via a pruned breadth-first search on the subset space of the rule group's upper bound antecedent genes. This breadth-first search can be quite time consuming. In the case of the Prostate Cancer (PC) dataset all 100 classification tests (25 tests for each of the 4 training set sizes) generated at least one top-10 rule group upper bound with more than 400 antecedent genes. Due to the difficulties involved with a breadth-first search over the subset space of a several hundred element set, RCBT began suffering from long run times on many PC classification tests.

Table IV contains four average classification test run times (in seconds) for each PC training size. The 'BSTC' column run times reflect the average time required to build both class 0 and class 1 BSTs and then use them to classify all the test samples. Each 'Top-k' column run time is the average time required for Top-k to mine the top 10 covering rule groups (with minimum support 0.7) for each training set.

Table IV's 'RCBT' column gives average run times for RCBT using a time cutoff value of 2 hours for all the training sets. For each classification test, if RCBT was unable to complete the test in less than the cutoff time, it was terminated and it's run time was reported as the cutoff time. Hence, the 'RCBT' column gives lower bounds on RCBT's average run time per training set test. Finally, the '# RCBT DNF' column gives the number of tests RCBT was unable to finish in $<$ the cutoff time, over the number of tests for which Top-K finished mining rule group upper bounds.

Explanation for varying nl values: Run time cutoffs were necessary to mitigate excessive holdout validation CAR-mining times. Even with a cutoff of 2 hours these 100 PC experiments required about 11 days of computation time, with most experiments not finishing. For the 80% and 1-52/0-50 training set sizes RCBT with $nl = 20$ failed to finish lower bound rule mining for all 50 tests within 2 hours. Thus, RCBT's nl parameter was lowered from the default value of 20 to 2 in an attempt to improve its chances of completing tests. Not surprisingly, decreasing nl (i.e., mining fewer lower bound rules per Top-k rule group) decreases RCBT's runtime. However, RCBT was still unable to finish lower bound rule mining for any tests.

Classification Accuracy: Figure 6 contains accuracy results for BSTC on all four Prostate Cancer test sets. Prostate Cancer boxplots for RCBT weren't constructed for training set sizes that RCBT was unable to complete all 25 tests within the time cutoffs. In contrast, BSTC was able to complete each of the 100 PC classification tests in less than 6 seconds. Table V contains mean accuracies for the PC dataset with 40%, 60%, 80%, and 1-52/0-50 training. For each training set, the average accuracies were taken over the tests RCBT was able to complete within the cutoff time. Hence, the 40% row means were taken over all 25 results. Since RCBT was unable to complete any 80% or 1-52/0-50 training size tests we report these BSTC means over all 25 tests. RCBT has slightly better accuracy then BSTC on 40% training. For 60% training

Training	BSTC	Top-k	RCBT	# RCBT DNF
40%	2.13	0.09	418.81	0/25
60%	4.93	5.06	$\geq$ 7110.00	24/25
80%	5.78	120.63	$\geq$ 7200 †	25/25†
1-52/0-50	5.57	21.32	$\geq$ 7200 †	25/25†

TABLE IV
AVERAGE RUN TIMES FOR THE PC TESTS (IN SECONDS). † INDICATES nl WAS LOWERED TO 2.

Training	BSTC	RCBT
40%	75.08%	79.27%
60%	78.18%	85.45%
80%	84.98%	—
1-52/0-50	81.65%	—

TABLE V
MEAN ACCURACIES FOR THE PC TESTS THAT RCBT FINISHED.

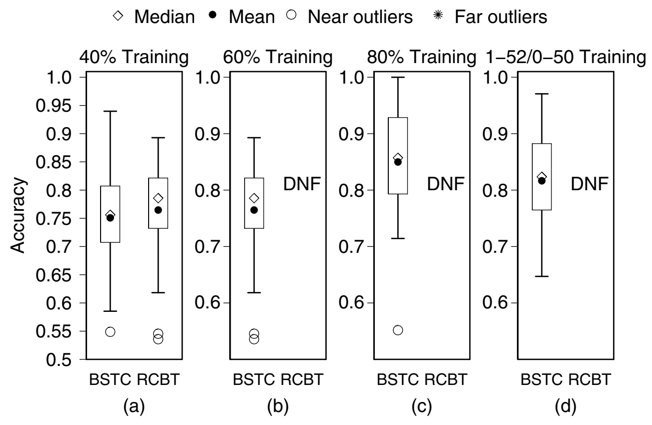

Fig. 6. PC Holdout Validation Results

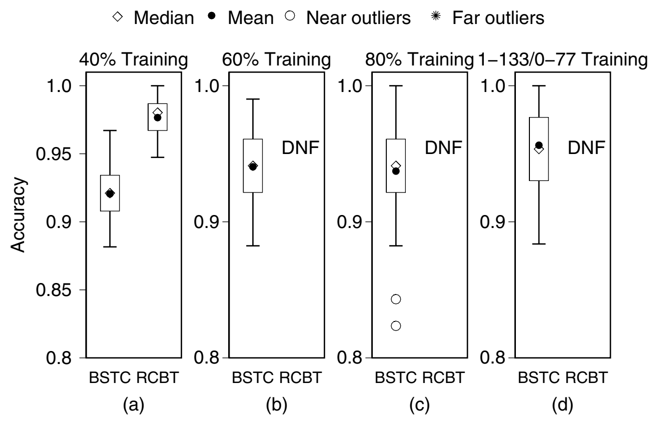

Fig. 7. OC Holdout Validation Results

RCBT outperforms BSTC on the single test it could finish by more then 7%, although it should be kept in mind that RCBT's results for the 24 unfinished tests could vary widely. Note that BSTC's (mean) accuracy increases monotonically with training set size as expected. At 60% training BSTC's accuracy behaves almost identically to RCBT's 40% training accuracy (see Figure 6).

4) Ovarian Cancer (OC) Experiment: For the Ovarian Cancer dataset, which is the largest dataset in this collection, the Top-k mining method that is used by RCBT also runs into long computational times. Although Top-k is an exceptionally fast CAR group upper bound miner, it still depends on performing a pruned exponential search over the training sample subset space. Thus, as the number of training samples increases Top-k quickly becomes computationally challenging to tune/use.

Table VI contains four average classification test run times (in seconds) for each Ovarian Cancer(OC) training size. As before, the second column run times each give the average time required to build both class 0/1 BSTs and then use them to classify all test's samples with BSTC. Note that BSTC was able to complete each OC classification test in about 1 minute. In contrast, RCBT again failed to complete processing most classification tests within 2 hours.

Table VI's third column gives the average times required for Top-k to mine the top 10 covering rule groups upper bounds for each training set test (with the same 2 hour cutoff procedure as used for PC testing). The fourth column gives the average run times of RCBT on the tests for which Top-k finished mining rules (also with a 2 hour cutoff). Finally, the '# RCBT DNF' column gives the number of tests that RCBT was unable to finish classifying in $<$ 2 hours each, over the number of tests for which Top-k finished. Because RCBT couldn't finish any 80% or 1-133/0-77 tests within 2 hours with $nl = 20$, we lowered nl to 2.

Classification Accuracy: Figure 7 contains boxplots for BSTC on all four OC classification test sets. Boxplots were not generated for RCBT with 60%, 80%, or 1-133/0-77 training since it was unable to finish all 25 tests for all these training set sizes in $<$ 2 hours each. Table VII lists the mean accuracies of BSTC and RCBT over the tests on which RCBT was able to produce results. Hence, Table VII's 40% row consists of averages over 25 results. Meanwhile Table VII's 60% row results are from 6 tests, 80% contains a single test's result, and 1-133/0-77 results from 3 tests. RCBT has better mean accuracy on the 40% training size, but the results are closer on the remaining sizes ($<$ 4% difference over RCBT's completed tests). Again, RCBT's accuracy could vary widely on its uncompleted tests.

CAR Mining Parameter Tuning and Scalability: We attempted to run Top-k to completion on the 3 OC 80% training and 2 OC 1-133/0-77 training tests. However it could not finish mining rules within the 2 hour cutoff. Top-k finished two of the three 80% training tests in 775 min 43.6 sec and 185 min 3.3 sec. However, the third test ran for over 16,000 min ($>$ 11 days) without finishing. Likewise, Top-k finished one of the two 1-133/0-77 tests in 126 min 45.2 sec but couldn't finish the other in 16,000 min ($>$ 11 days). After increasing Top-k's support cutoff from 0.7 to 0.9 it was able to finish the two unfinished 80% and 1-133/0-77 training tests in 5 min 13.8 sec and 35 min 36.9 sec, respectively. However, RCBT (with $nl = 2$) then wasn't able to finish lower bound rule mining for either of these two tests within 1,500 min. Clearly, CAR-mining and parameter tuning on large training sets is

Training	BSTC	Top-k	RCBT	# RCBT DNF
40%	30.89	0.6186	273.37	0/25
60%	61.28	41.21	$\geq$ 5554.37	19/25
80%	71.84	$\geq$ 1421.80	$\geq$ 7205.43 †	21/22
1-133/0-77	70.38	$\geq$ 1045.65	$\geq$ 6362.86 †	20/23

TABLE VI
AVERAGE RUN TIMES FOR THE OC TESTS (IN SECONDS). † INDICATES nl WAS LOWERED TO 2.

Training	BSTC	RCBT
40%	92.05%	97.66%
60%	95.75%	96.73%
80%	94.12%	98.04%
1-133/0-77	93.80%	96.12%

TABLE VII
MEAN ACCURACIES FOR THE OC TESTS THAT RCBT FINISHED.

computationally challenging. As training set sizes increase, it is likely that these difficulties will also increase.

VI. Related Work

While operating on a microarray dataset, current CAR [1], [2], [3], [4] and other pattern/rule [20], [21] mining algorithms perform a pruned and/or compacted exponential search over either the space of gene subsets or the space of sample subsets. Hence, they are generally quite computationally expensive for datasets containing many training samples (or genes as the case may be). BSTC is explicitly related to CAR-based classifiers, but requires no expensive CAR mining.

BSTC is also related to decision tree-based classifiers such as random forest [19] and C4.5 family [9] methods. It is possible to represent any consistent set of boolean association rules as a decision tree, and vice versa. However, it is generally unclear how the trees generated by current tree-based classifiers are related to high confidence/support CARs which are known to be particularly useful for microarray data[1], [2], [6], [7], [11]. BSTC is explicitly related to, and motivated by, CAR-based methods.

To the best of our knowledge there is no previous work on mining/classifying with BARs of the form we consider here. Perhaps the work closest to utilizing 100% BARs is the TOP-RULES [22] miner. TOP-RULES utilizes a data partitioning technique to compactly report item/gene subsets which are unique to each class set C_i. Hence, TOP-RULES discovers all 100% confident CARs in a dataset. However, the method must utilize an emerging pattern mining algorithm such as MBD-LLBORDER [23], and so generally isn't polynomial time. Also related to our BAR-based techniques are recent methods which mine gene expression training data for sets of fuzzy rules [24], [25]. Once obtained, fuzzy rules can be used for classification in a manner analogous to CARs. However, the resulting fuzzy classifiers don't appear to be as accurate as standard classification methods such as SVM [25].

VII. Conclusions and Future Work

To address the computational difficulties involved with preclassification CAR mining (see Tables IV and VI), we developed a novel method which considers a larger subset of CAR-related boolean association rules (BARs). These rules can be compactly captured in a Boolean Structure Table (BST), which can then be used to produce a BST classifier called BSTC. Comparison to the current leading CAR classifier, RCBT, on several benchmark microarray datasets shows that BSTC is competitive with RCBT's accuracy while avoiding the exponential costs incurred by CAR mining (see Section V-B). Hence, BSTC extends generalized CAR-based methods to larger datasets then previously practical. Furthermore, unlike other association rule-based classifiers, BSTC easily generalizes to multi-class gene expression datasets.

BSTC's worst case per-query classification time is worse then CAR-based methods, after all exponential time CAR mining is completed ($O(|S|^2 \cdot |G|)$ versus $O(|S| \cdot |G|)$). As future work we plan on investigating techniques to decrease this cost by carefully culling BST exclusion lists.

Acknowledgments

We thank Anthony K.H. Tung and Xin Xu for sending us their discretized microarray data files and Top-k/RCBT executables. This research was supported in part by NSF grant DMS-0510203, NIH grant 1-U54-DA021519-01A1, and by the Michigan Technology Tri-Corridor grant GR687. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] G. Cong, K. L. Tan, A. K. H. Tung, and X. Xu, "Mining top-k covering rule groups for gene expression data," *SIGMOD*, 2005.
[2] G. Cong, A. K. H. Tung, X. Xu, F. Pan, and J. Yang, "Farmer: Finding interesting rule groups in microarray datasets," *SIGMOD*, 2004.
[3] J. Wang, J. Han, and J. Pei, "Closet+: Searching for the best strategies for mining frequent closed itemsets," *KDD*, 2003.
[4] M. Zaki and C. Hsiao, "Charm: An efficient algorithm for closed association rule mining," *Proc. of the 2nd SIAM Int. Conf. on Data Mining (SDM)*, 2002.
[5] R. Agrawal, T. Imielinski, and A. Swami, "Mining associations between sets of items in large databases," *SIGMOD*, pp. 207–216, 1993.
[6] G. Dong, X. Zhang, L. Wong, and J. Li, "Caep: Classification by aggregating emerging patterns," *Proc. 2nd Int. Conf. Discovery Science (DS)*, 1999.
[7] J. Li and L. Wong, "Identifying good diagnostic genes or gene groups from gene expression data by using the concept of emerging patterns," *Bioinformatics*, vol. 18, pp. 725–734, 2002.
[8] C. Cortes and V. Vapnik, "Support-vector networks." *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
[9] J. R. Quinlan, "Bagging, boosting, and c4.5," *AAAI*, vol. 1, pp. 725–730, 1996.
[10] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," *KDD*, 1998.
[11] T. McIntosh and S. Chawla, "On discovery of maximal confident rules without support pruning in microarray data," *SIGKDD Workshop on Data Mining in Bioinformatics (BIOKDD)*, 2005.
[12] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *VLDB*, pp. 487–499, 1994.
[13] *Available at* http://www-personal.umich.edu/∼markiwen/.
[14] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.
[15] S. Sudarsky, "Fuzzy satisfiability," *Intl. Conf. on Industrial Fuzzy Control and Intelligent Systems (IFIS)*, 1993.
[16] *Available at* http://sdmc.i2r.a-star.edu.sg/rp/.
[17] "The dprep package," *http://cran.r-project.org/doc/packages/dprep.pdf.*
[18] C. Chang and C. Lin, "Libsvm: a library for support vector machines," 2007. [Online]. Available: www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf
[19] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[20] W. Li, J. Han, and J. Pei, "Cmar: Accurate and efficient classification based on multiple class-association rules," *ICDM*, 2001.
[21] F. Rioult, J. F. Boulicaut, B. Cremilleux, and J. Besson, "Using transposition for pattern discovery from microarray data," *DMKD*, pp. 73–79, 2003.
[22] J. Li, X. Zhang, G. Dong, K. Ramamohanarao, and Q. Sun, "Efficient mining of high confidence association rules without support thresholds," *Principles of Data Mining and Knowledge Discovery (PKDD)*, pp. 406 – 411, 1999.
[23] G. Dong and J. Li, "Efficient mining of emerging patterns: discovering trends and differences," *KDD*, pp. 43–52, 1999.
[24] S. Vinterbo, E. Kim, and L. Ohno-Machado, "Small, fuzzy and interpretable gene expression based classifiers," *Bioinformatics*, vol. 21, pp. 1964–1970, 2005.
[25] S.-Y. Ho, C.-H. Hsieh, H.-M. Chen, and H.-L. Huang, "Interpretable gene expression classifier with an accurate and compact fuzzy rule base for microarray data analysis," *Biosystems*, vol. 85, pp. 165–176, 2006.