OPERATIONS RESEARCH CENTER

Working Paper

ORC: Ordered Rules for Classification
A Discrete Optimization Approach to Associative Classification

by

Dimitris Bertsimas
Allison Chang
Cynthia Rudin

OR 386-11 **October 2011**

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Submitted to the Annals of Statistics

ORDERED RULES FOR CLASSIFICATION: A DISCRETE OPTIMIZATION APPROACH TO ASSOCIATIVE CLASSIFICATION*

By Dimitris Bertsimas, Allison Chang, and Cynthia Rudin

Massachusetts Institute of Technology

We aim to design classifiers that have the interpretability of association rules yet match the predictive power of top machine learning algorithms for classification. We propose a novel mixed integer optimization (MIO) approach called Ordered Rules for Classification (ORC) for this task. Our method has two parts. The first part mines a particular frontier of solutions in the space of rules, and we show that this frontier contains the best rules according to a variety of interestingness measures. The second part learns an optimal ranking for the rules to build a decision list classifier that is simple and insightful. We report empirical evidence using several different datasets to demonstrate the performance of this method.

1. Introduction. Our goal is to develop classification models that are on par in terms of accuracy with the top classification algorithms, yet are interpretable, or easily understood by humans. This work thus addresses a dichotomy in the current state-of-the-art for classification: On the one hand, there are algorithms such as support vector machines (SVM) [Vapnik, 1995] that are highly accurate but not interpretable; for instance, trying to explain a support vector kernel to a medical doctor is not likely to persuade him to use an SVM-based diagnostic system. On the other hand, there are algorithms such as decision trees [Breiman et al., 1984, Quinlan, 1993] that are interpretable, but not specifically optimized to achieve the highest accuracy. For applications in which the user needs an accurate model as well as an understanding of how it makes predictions, we develop a new classification model that is both intuitive and optimized for accuracy.

Our models are designed to be interpretable from multiple perspectives. First, the models are designed to be *convincing*: for each prediction, the algorithm also provides the reasons for why this particular prediction was made, highlighting exactly which data were used to make it. To achieve this, we use "association rules" to build "decision lists," that is, ordered sets of rules.

*Supported by NSF Grant IIS-1053407.

AMS 2000 subject classifications: Primary 68T05; secondary 90C11

Keywords and phrases: Association rules, associative classification, integer optimization

The second way our models are interpretable involves their size: these models are designed to be *concise*. Psychologists have long studied human ability to process data, and have shown that humans can simultaneously process only a handful of cognitive entities, and are able to estimate relatedness of only a few variables at a time [e.g. Miller, 1956, Jennings et al., 1982]. Thus conciseness contributes to interpretability, and our formulations include two types of regularization towards concise models. The first encourages rules to have small left-hand-sides, so that the reasons given for each prediction are sparse. The second encourages the decision list to be shorter. That is, the regularization essentially pulls the default rule (the rule that applies if none of the rules above it apply) higher in the list. We aim to construct a convincing and concise model that limits the reasoning required by humans to understand and believe its predictions. These models allow predictions to more easily be communicated in words, rather than in equations.

The principal methodology we use in this work is mixed integer optimization (MIO), which helps our classification algorithm achieve high accuracy. Rule learning problems suffer from combinatorial explosion, in terms of both searching through a dataset for rules and managing a massive collection of potentially interesting rules. A dataset with even a modest number of features can contain thousands of rules, thus making it difficult to find useful ones. Moreover, for a set of L rules, there are $L!$ ways to order them into a decision list. On the other hand, MIO solvers are designed *precisely* to handle combinatorial problems, and the application of MIO to rule learning problems is reasonable given the discrete nature of rules. However, designing an MIO formulation is a nontrivial task because the ability to solve an MIO problem depends critically on the strength of the formulation, which is related to the geometry of the feasible set of solutions. This is considerably more challenging than linear optimization, which has a similar form but without integrality constraints on the variables. We develop MIO formulations for both the problem of mining rules and the problem of learning to rank them. Our experiments show predictive accuracy on a variety of datasets at about the same level as the current top algorithms, as well as advantages in interpretability.

In Section 2, we discuss related work. In Section 3, we state our notation and derive MIO formulations for association rule mining. In Section 4, we present a learning algorithm, also an MIO formulation, that uses the generated rules to build a classifier. In Section 5, we show results on classification accuracy, and in Section 6, we demonstrate the interpretability of our classifiers. In Section 7, we discuss the application of our methodology to large-scale data. We conclude in Section 8.

2. Related Work. Association rule mining was introduced by Agrawal et al. [1993] to aid market-basket analysis, the purpose of which was to discover sets of items, or *itemsets*, that were often purchased together, such as the well-known (though probably fictitious) correlation between sales of beer and diapers [Büchter and Wirth, 1998]. To help increase store profit and customer satisfaction, these easy-to-understand patterns could be used to guide the management of store layout, customer segmentation, and items for sale. Consider the rule $\{i, j\} \Rightarrow k$, where $s\%$ of customers purchased items i, j, and k, and $c\%$ of customers who purchased items i and j also purchased item k. In this case, $\{i, j\}$ is the *body* of the rule, k is the *head*, s is the *support*, and c is the *confidence*. In general, the most challenging part of rule mining is to first generate all itemsets with support exceeding a specified threshold, called *frequent itemsets*. Frequent itemsets have a downward closure property, that is, any subset of a frequent itemset must also be frequent. Even so, the problem of counting the number of maximal frequent itemsets, or itemsets that are not subsets of other frequent itemsets, is #P-complete, suggesting that the problem of enumerating all frequent itemsets can in general be hard [Yang, 2004]. Since the introduction of the Apriori method by Agrawal and Srikant [1994], researchers have proposed many algorithms for frequent pattern mining that apply heuristic techniques to traverse the search space, which grows exponentially with the number of items in the dataset [Han et al., 2007, Hipp et al., 2000, Goethals, 2003].

Frequent itemset generation often leads to an overwhelming number of rules, making it difficult to distinguish the most useful rules. To make sense of such an enormous collection of rules, users typically rank them by a measure of "interestingness," which can be defined in many different ways. There is a large body of literature on interestingness measures, such as lift, conviction, Laplace, and gain [review articles include those of Tan and Kumar, 2000, McGarry, 2005, Geng and Hamilton, 2006]. The existence of so many interestingness measures introduces another problem of how to select an interestingness measure for a particular task. Bayardo and Agrawal [1999] showed that if the head of the rule is fixed, then a number of interestingness metrics are optimized by rules that lie along the upper support-confidence border, where a rule on this border has the highest confidence among rules with equal or higher support. They proposed an algorithm to mine only this border, which indeed produces a reduced set of rules. In this paper, we extend the idea of an optimal border to general rules, not just the case of rules with fixed heads, and we use MIO to find the border.

Association rules were originally designed for data exploration, and later *associative classification* developed as a framework to use the rules for clas-

sification, with algorithms such as CBA, CMAR, and CPAR [Liu et al., 1998, Li et al., 2001, Yin and Han, 2003, Simon et al., 2011]. Reviews of the different approaches are given by Thabtah [2007], Rückert [2008], and Vanhoof and Depaire [2010]. Methods to build a classifier using a sorted set of association rules fall into two categories: those that predict based on an ensemble of rules, and those that predict based on a single rule in a ranked list of rules. The first category uses more information by classifying based on a sort of majority vote of rules, but typically has two disadvantages: first, it ignores the dependency between rules, so even two rules that are almost exactly the same have two separate votes instead of one; and second, the model loses some interpretability by combining rules together. Boosted decision trees share a related problem–they no longer have the interpretability of single decision trees. Examples of rule ensemble classifiers are in Friedman and Popescu [2008] and Meinshausen [2010]. These models are similar to the Logical Analysis of Data (LAD) model [Boros et al., 2000], though the LAD model uses only rules that have confidence equal to one, so that even rules with confidence 0.99 are discarded, which could lead to overfitting. The second category of sorted-rule-based classification algorithms produces decision lists [see Rivest, 1987, Klivans and Servedio, 2006, Sokolova et al., 2003, Anthony, 2005, Long and Servedio, 2007, Marchand and Sokolova, 2005, Rudin et al., 2011]. These classifiers are simple to understand and use the highest ranked rules for prediction. However, if the list is not properly ordered, it may not yield an accurate classifier. Decision lists are commonly created by ordering rules according to an interestingness measure. In this work, we *learn* the ordering of rules from data instead. Learning the rule list has the potential to be substantially more accurate in terms of misclassification error than ranking rules by an arbitrary choice of interestingness measure. As far as we know, there are no other mathematical programming approaches to creating decision lists in the literature.

3. Mining Optimal Association Rules. In this section, we describe an MIO method to generate the rules that form the building blocks for the classifier. First, we derive constraints that characterize the full set of possible rules for a dataset. Then, we present an MIO algorithm to find a set of general rules. Finally, we address the special case of mining rules for binary classification, for which the rules have a particular form.

3.1. *Interestingness and the Frontier.* We use the following standard notation: Let $\mathcal{I} = \{1, \ldots, d\}$ be a set of items, and let $\mathcal{D}$ be a dataset in which each observation is some subset of items in $\mathcal{I}$. Let there be n observations in $\mathcal{D}$. An association rule has the form $X \Rightarrow Y$, where $X, Y \subseteq \mathcal{I}$ and $X \cap Y = \emptyset$.

We want to formulate a set of constraints that define the space $\mathcal{P}$ of possible rules. In what follows, the t_i are data, while b, h, x_i, y_i, and z_i are variables. The binary vector $t_i \in \{0,1\}^d$ represents observation i:

$$t_{ij} = \mathbf{1}_{[\text{observation } i \text{ includes item } j]}, \qquad 1 \le i \le n, \quad 1 \le j \le d.$$

The binary vectors $b, h \in \{0,1\}^d$ represent the body and head respectively of a given rule $X \Rightarrow Y$. That is, for $j = 1, \ldots, d$,

$$b_j = \mathbf{1}_{[j \in X]} \quad \text{and} \quad h_j = \mathbf{1}_{[j \in Y]}.$$

We also use variables x_i, y_i, and z_i, for $i = 1, \ldots, n$, to represent

$$x_i = \mathbf{1}_{[\text{observation } i \text{ includes } X]}, \quad y_i = \mathbf{1}_{[\text{observation } i \text{ includes } Y]}, \quad \text{and}$$
$$z_i = \mathbf{1}_{[\text{observation } i \text{ includes } X \text{ and } Y]}.$$

$\mathcal{P}$ is constrained by (1) through (10). Note that e_d is the d-vector of ones. Each constraint is explained below.

$$b_j + h_j \le 1, \quad \forall j, \tag{1}$$
$$x_i \le 1 + (t_{ij} - 1)b_j, \quad \forall i, j, \tag{2}$$
$$x_i \ge 1 + (t_i - e_d)^T b, \quad \forall i, \tag{3}$$
$$y_i \le 1 + (t_{ij} - 1)h_j, \quad \forall i, j, \tag{4}$$
$$y_i \ge 1 + (t_i - e_d)^T h, \quad \forall i, \tag{5}$$
$$z_i \le x_i, \quad \forall i, \tag{6}$$
$$z_i \le y_i, \quad \forall i, \tag{7}$$
$$z_i \ge x_i + y_i - 1, \quad \forall i, \tag{8}$$
$$b_j, h_j \in \{0,1\}, \quad \forall j, \tag{9}$$
$$0 \le x_i, y_i, z_i \le 1, \quad \forall i. \tag{10}$$

Since an item cannot be in both the body and head of a rule ($X \cap Y = \emptyset$), b and h must satisfy (1). To understand (2), consider the two cases $b_j = 0$ and $b_j = 1$. If $b_j = 0$, then the constraint is just $x_i \le 1$, so the constraint has no effect. If $b_j = 1$, then the constraint is $x_i \le t_{ij}$. That is, if $b_j = 1$ (item j is in X) but $t_{ij} = 0$ (item j is not in observation i), then $x_i = 0$. This set of constraints implies that $x_i = 0$ if observation i does not include X. We need (3) to say that $x_i = 1$ if observation i includes X. Note that $t_i^T b$ is the number of items in the intersection of observation i and X, and $e_d^T b$ is the number of items in X. Constraint (3) is valid because

$$t_i^T b = \sum_{j=1}^{d} t_{ij} b_j \le \sum_{j=1}^{d} b_j = e_d^T b,$$

TABLE 1
The body X of the rule is in observation i since (2) and (3) are satisfied.

	j				
	1	2	3	4	5
t_i (1 if item j in observation i, 0 otherwise)	1	0	1	1	0
b (1 if item j in body of rule, 0 otherwise)	1	0	0	1	0

where equality holds if and only if observation i includes X and otherwise $t_i^T b \leq e_d^T b - 1$. Table 1 helps to clarify (2) and (3). Constraints (4) and (5) capture the y_i in the same way that (2) and (3) capture the x_i. The z_i are 1 if and only if $x_i = y_i = 1$, which is captured by (6) through (8). Constraints (9) and (10) specify that b and h are restricted to be binary, while the values of x, y, and z are restricted only to be between 0 and 1.

Each point in $\mathcal{P}$ corresponds to a rule $X \Rightarrow Y$, where $X = \{j : b_j = 1\}$ and $Y = \{j : h_j = 1\}$. There are $2d$ binary variables and $3n$ continuous variables. Computationally, it is favorable to reduce the number of integer variables, and here we explain why x, y, and z are not also restricted to be integral. There are two cases when deciding whether X is in observation i. If it is, then (3) implies $x_i \geq 1$, so $x_i = 1$. If it is not, then there exists j such that $t_{ij} = 0$ and $b_j = 1$, so (2) implies $x_i \leq 0$, or $x_i = 0$. Thus, in either case, x_i is forced to be an integer, regardless of whether we specify it as an integer variable. The argument is similar for y_i. For z_i, there are two cases when deciding whether X and Y are both in observation i. If they are, then $x_i = y_i = 1$, so (8) implies $z_i \geq 1$, or $z_i = 1$. If they are not, then either (6) or (7) implies $z_i \leq 0$, or $z_i = 0$. Thus, z_i is also always integral.

$\mathcal{P}$ grows exponentially in the number of items $d = |\mathcal{I}|$. It includes the full set of association rules, which is many more than we usually need or wish to collect. In order to generate only the potentially interesting rules, we judge each rule according to three of its fundamental properties, namely

$$s_X = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad s_Y = \frac{1}{n}\sum_{i=1}^{n} y_i, \quad \text{and} \quad s = \frac{1}{n}\sum_{i=1}^{n} z_i,$$

called *coverage*, *prevalence*, and *support* respectively. When we refer to these measures for a particular rule r, we use the notation $s_X(r)$, $s_Y(r)$, and $s(r)$; we omit the parenthetical "(r)" when referring to them in general. We now define a partial order $\leq_p$ over the set of possible rules to rank them in order of interestingness. Given two rules r and r^*, we have $r \leq_p r^*$, or r^* is at least as interesting as r if and only if:

$$s_X(r) \geq s_X(r^*), \quad s_Y(r) \geq s_Y(r^*), \quad \text{and} \quad s(r) \leq s(r^*).$$

Moreover, $r =_p r^*$ if and only if $s_X(r) = s_X(r^*)$, $s_Y(r) = s_Y(r^*)$, and $s(r) = s(r^*)$. In words, "$r \leq_p r^*$" means the coverage and prevalence of r^* are no greater than that of r, but the support of r^* is at least that of r. Let F^* be the set of rules that are not dominated by any other rules, that is,

$$\mathcal{F}^* = \{r : \text{ There does not exist any } \bar{r} \text{ such that } r <_p \bar{r}.\}.$$

The rules $r \in \mathcal{F}^*$ fall along a three dimensional frontier in s_X, s_Y, and s.

For intuition on why this frontier reasonably captures interestingness, consider the interestingness measure of *confidence*, which is the empirical probability of Y given X. Refer to the data in Tables 2 and 3. Suppose we have 20 observations, and we wish to compare the interestingness of two rules: **a** {chips}⇒{guacamole} and **b** {cookies}⇒{milk}. In Case 1 in the table, the two rules have equal coverage $s_X = \frac{8}{20}$, but the support s is higher for **a** ($\frac{7}{20}$ versus $\frac{5}{20}$), so **a** has higher confidence (**a** is more interesting). In Case 2, the rules have equal support $s = \frac{5}{20}$, but the coverage s_X is lower for **b** ($\frac{8}{20}$ versus $\frac{10}{20}$), so **b** has higher confidence (**b** is more interesting). This example shows that higher support and lower coverage increase the confidence of a rule; for other measures, lower prevalence also often increases the interestingness.

TABLE 2
Number of observations containing certain items (assume 20 observations total in both Case 1 and Case 2).

	Case 1	**Case 2**
{chips}	8	10
{cookies}	8	8
{chips, guacamole}	7	5
{cookies, milk}	5	5

TABLE 3
Support, coverage, and confidence of rules ***a*** *{chips}⇒{guacamole} and* ***b*** *{cookies}⇒{milk}.*

		Case 1	**Case 2**
a	supp	7/20	5/20
	cov	8/20	10/20
	conf	**7/8**	**5/10**
b	supp	5/20	5/20
	cov	8/20	8/20
	conf	**5/8**	**5/8**

The point is that many measures in addition to confidence, including those in Table 4, increase with decreasing s_X (holding s_Y and s constant), decreasing s_Y (holding s_X and s constant), and increasing s (holding s_X and s_Y constant). Thus, the rules that optimize each of these measures is in the frontier $\mathcal{F}^*$, and this is the set of rules that we focus on generating.

3.2. *MIO Algorithm for General Association Rule Mining.* We can find each rule on the frontier $\mathcal{F}^*$ corresponding to $\leq_p$ by putting upper bounds on both s_X and s_Y, and then maximizing s. We vary the bounds over all

TABLE 4
Interestingness measures. The notation $P(A)$ means the fraction, or empirical probability, of observations containing set A.

Measure	**Definition**	
Confidence/Precision	$P(Y\|X)$	$\frac{s}{s_X}$
Recall	$P(X\|Y)$	$\frac{s}{s_Y}$
Accuracy	$P(X \cup Y) + P(X^c \cup Y^c)$	$1 - s_X - s_Y + 2s$
Lift/Interest	$\frac{P(X \cup Y)}{P(X)P(Y)}$	$\frac{s}{s_X s_Y}$
Conviction	$\frac{P(X)P(Y^c)}{P(X \cup Y^c)}$	$\frac{1 - s_Y}{1 - s/s_X}$
Laplace Correction	$\frac{nP(X \cup Y) + 1}{nP(X) + k}$, k is # of classes	$\frac{ns + 1}{ns_X + k}$
Piatetsky-Shapiro	$P(X \cup Y) - P(X)P(Y)$	$s - s_X s_Y$

possible values to produce the entire frontier. In particular, Formulation (11) maximizes the "scaled support" $(n \cdot s)$ for a certain choice $\bar{s}_X$ and $\bar{s}_Y$, which denote the user-specified upper bounds on the "scaled coverage" $(n \cdot s_X)$ and "scaled prevalence" $(n \cdot s_Y)$ respectively.

(11)
$$\begin{aligned} \max_{b,h,x,y,z} \quad & \sum_{i=1}^{n} z_i - R_{\text{gen\_xy}} \left(\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i \right) - R_{\text{gen\_bh}} \left(\sum_{j=1}^{d} b_j + \sum_{j=1}^{d} h_j \right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i \leq \bar{s}_X, \\ & \sum_{i=1}^{n} y_i \leq \bar{s}_Y, \\ & (b, h, x, y, z) \in \mathcal{P}. \end{aligned}$$

The first term in the objective is the scaled support. The second set of terms $\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i$ correspond to the coverage s_X and prevalence s_Y; if there are multiple rules with optimal support, we want those with smaller coverage and prevalence since otherwise we would be generating rules not on the frontier. The third set of terms $\sum_{j=1}^{d} b_j + \sum_{j=1}^{d} h_j$ are for regularization, and correspond to the sparsity of the rule; if there are multiple rules that maximize s and have equal s_X and s_Y, we want those with smaller bodies and heads, that is, more zeros in b and h. The parameters $R_{\text{gen\_xy}}$ and $R_{\text{gen\_bh}}$ control the weight of these terms in the objective, where the former ensures that we properly trace out the frontier, and the latter could potentially trade

off sparsity for closeness to the frontier.

Solving (11) once for each possible pair $(\bar{s}_X, \bar{s}_Y)$ does not yield the entire frontier since there may be multiple optimal rules at each point on the frontier. To find other optima, we add constraints making each solution found so far infeasible, so that they cannot be found again when we re-solve. Specifically, for each pair $(\bar{s}_X, \bar{s}_Y)$, we iteratively solve the formulation as follows: Let (h^*, b^*) be the first optimum we find for (11). In each iteration, we add the constraint

$$\sum_{j:b_j^*=0} b_j + \sum_{j:b_j^*=1} (1 - b_j) + \sum_{j:h_j^*=0} h_j + \sum_{j:h_j^*=1} (1 - h_j) \geq 1 \tag{12}$$

to the formulation. This constraint says that in either the vector b or the vector h, at least one of the components must be different from in the previous solution; that is, at least one of the zeros must be one or one of the ones must be zero. The previous solution $b_j = b_j^*$ and $h_j = h_j^*$ is infeasible since it would yield $0 \geq 1$ in (12). After adding this constraint, we solve again. If the optimal value of $\bar{s} = \sum_{i=1}^{n} z_i$ decreases, then we exit the loop. Otherwise, we have a new optimum, so we repeat the step above to generate another constraint and re-solve.

3.3. *MIO Algorithm for Associative Classification.* As our main goal is to use association rules to construct a decision list for binary classification, we show in this section how to use MIO to mine rules for this purpose. In this case, the rules are of a specific form, either $X \Rightarrow 1$ or $X \Rightarrow -1$. That is, we prespecify the heads Y of the rules to be a class attribute, 1 or -1. Our rule generation algorithm mines two separate frontiers of rules, one frontier for each class.

Suppose we want rules on the frontier for a fixed class $y \in \{-1, 1\}$. Let $S = \{i : \text{ observation } i \text{ has class label } y\}$. Then $s = \frac{1}{n} \sum_{i \in S} x_i$. Since $s_Y = |S|$ is equal for all rules of interest, we simplify the partial order (2) so that given two rules r and r^*, we have $r \leq_p r^*$ if and only if:

$$s_X(r) \geq s_X(r^*) \quad \text{and} \quad s(r) \leq s(r^*).$$

Also, $r =_p r^*$ if and only if $s_X(r) = s_X(r^*)$ and $s(r) = s(r^*)$. Each rule on the corresponding two dimensional frontier in s_X and s can be found by upper bounding s_X and maximizing s. Since Y is fixed, we do not need the h, y, or z variables from (11). Formulation (13) finds a rule with maximum s

for a given upper bound $\bar{s}_X$ on $n \cdot s_X$.

$$
\begin{aligned}
(13) \qquad \max_{b,x} \quad & \sum_{i \in S} x_i - R_{\text{gen\_x}} \sum_{i=1}^{n} x_i - R_{\text{gen\_b}} \sum_{j=1}^{d} b_j \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_i \leq \bar{s}_X, \\
& x_i \leq 1 + (t_{ij} - 1) b_j, \quad \forall i, j, \\
& x_i \geq 1 + (t_i - e_d)^T b, \quad \forall i, \\
& b_j \in \{0, 1\}, \quad \forall j, \\
& 0 \leq x_i \leq 1, \quad \forall i.
\end{aligned}
$$

The first term in the objective corresponds to support, and the others correspond to coverage and sparsity, similar to the terms in (11). Solving (13) once for each value of $\bar{s}_X$ does not yield the entire frontier since there may be multiple optima. Analogous to the general case, we solve the formulation iteratively: Start by setting $\bar{s}_X = n$ since the largest possible value of the scaled coverage is n. Let b^* be the first optimum. Add the "infeasibility constraint"

$$
(14) \qquad \sum_{j : b_j^* = 0} b_j + \sum_{j : b_j^* = 1} (1 - b_j) \geq 1
$$

to the formulation, and solve again. If we find another optimum, then we repeat the step above to generate another constraint and re-solve. If the optimal value of $\bar{s} = \sum_{i \in S} x_i$ decreases, then we set the upper bound on $\bar{s}_X$ to a smaller value and iterate again. Note that we can set this new value to be the minimum of $\sum_{i=1}^{n} x_i$ and $\bar{s}_X - 1$ (previous bound minus one); we know that no rule on the remainder of the frontier has scaled coverage greater than $\sum_{i=1}^{n} x_i$, so using this as the bound provides a tighter constraint than using $\bar{s}_X - 1$ whenever $\sum_{i=1}^{n} x_i < \bar{s}_X - 1$.

Thus our rule generation algorithm, called "RuleGen," generates the frontier, one rule at a time, from largest to smallest coverage. The details are shown in Figure 1. RuleGen allows optional minimum coverage thresholds $\texttt{mincov}_{-1}$ and $\texttt{mincov}_1$ to be imposed on each of the classes of rules. Also, `iter_lim` limits the number of times we iterate the procedure above for a fixed value of s_X with adding (14) between iterates. To find all rules on the frontiers, set $\texttt{mincov}_{-1} = \texttt{mincov}_1 = 0$ and $\texttt{iter\_lim} = \infty$.

To illustrate the steps of the algorithm, Figure 2 shows the following fictitious example:

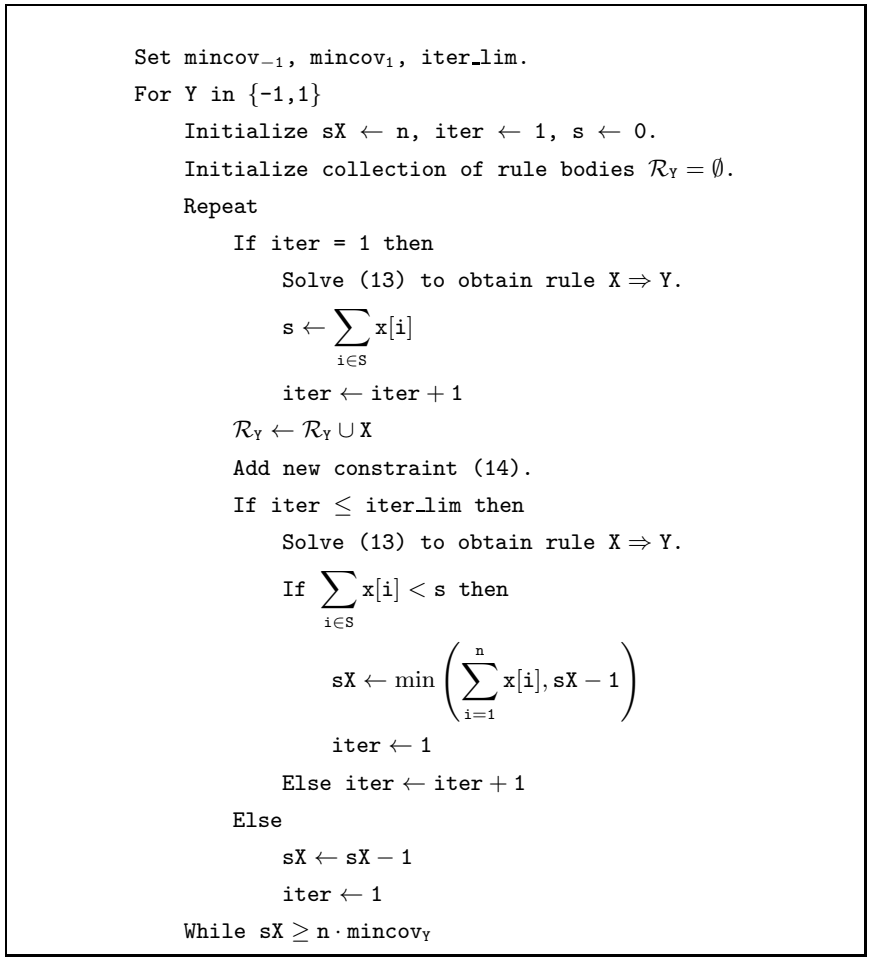

FIG 1. *RuleGen algorithm. (Note* $\texttt{sX}=\bar{s}_X$ *and* $\texttt{s}=\bar{s}$*.)*

a. Suppose we are constructing the frontier for data with $n = 100$. Initialize $\texttt{sX}$ to n and solve (13). Assume the first solution has $\sum_{i\in S} x_i = 67$. Then the algorithm adds the first rule to $\mathcal{R}_\texttt{Y}$ and sets $\texttt{s}$ to 67. It adds the infeasibility constraint (14) to (13) and re-solves. Assume the new rule still has $\sum_{i\in S} x_i = 67$, so the algorithm adds this rule to $\mathcal{R}_\texttt{Y}$, then adds another infeasibility constraint to (13) and re-solves.
b. Assume the new rule has $\sum_{i\in S} x_i = 65$ and $\sum_{i=1}^{n} x_i = 83$ (corresponding to the support and coverage respectively). Since $\sum_{i\in S} x_i$ decreased, the algorithm sets $\texttt{sX}$ to $\min\left(\sum_{i=1}^{n} x_i, \texttt{sX} - 1\right) = \min(83, 99) = 83$ before re-solving to obtain the next rule and adding it to $\mathcal{R}_\texttt{Y}$.
c. This process continues until the minimum coverage threshold is reached.

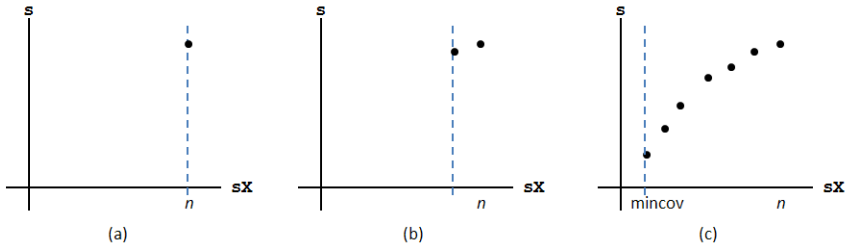

FIG 2. *Illustrative example to demonstrate the steps in the RuleGen algorithm.*

4. Building a Classifier. Suppose we have generated L rules, where each rule ℓ is of the form $X_\ell \Rightarrow -1$ or $X_\ell \Rightarrow 1$. Our task is now to rank them to build a decision list for classification. Given a new observation, the decision list classifies it according to the highest ranked rule ℓ such that X_ℓ is in the observation, or the highest rule that "applies." In this section, we derive an empirical risk minimization algorithm using MIO that yields an optimal ranking of rules. That is, the ordering returned by our algorithm maximizes the (regularized) classification accuracy on a training sample.

We always include in the set of rules to be ranked two "null rules:" $\emptyset \Rightarrow -1$, which predicts class -1 for any observation, and $\emptyset \Rightarrow 1$, which predicts class 1 for any observation. In the final ranking, the higher of the null rules corresponds effectively to the bottom of the ranked list of rules; all observations that reach this rule are classified by it, thus the class it predicts is the default class. We include both null rules in the set of rules because we do not know which of them would serve as the better default, that is, which would help the decision list to achieve the highest possible classification accuracy; our algorithm learns which null rule to rank higher.

We use the following parameters:

$$p_{i\ell} = \begin{cases} 1 & \text{if rule } \ell \text{ applies to observation } i \text{ and predicts its class correctly,} \\ -1 & \text{if rule } \ell \text{ applies to observation } i \text{ but predicts its class incorrectly,} \\ 0 & \text{if rule } \ell \text{ does not apply to observation } i, \end{cases}$$

$v_{i\ell} = \mathbf{1}_{[\text{rule } \ell \text{ applies to observation } i]} = |p_{i\ell}|,$

$R_{\text{rank}} =$ regularization parameter, trades off accuracy with conciseness,

and decision variables:

$r_\ell =$ rank of rule ℓ,

$r_* =$ rank of higher null rule,

$u_{i\ell} = \mathbf{1}_{[\text{rule } \ell \text{ is the rule that predicts the class of observation } i]}.$

The r_ℓ variables store the ranks of the rules; r_* is the rank of the default rule, which we want to be high for conciseness. The $u_{i\ell}$ variables help capture the mechanism of the decision list, enforcing that only the highest applicable rule predicts the class of an observation: for observation i, $u_{i\ell} = 0$ for all except one rule, which is the one, among those that apply, with the highest rank r_ℓ. The formulation to build the optimal classifier is:

$$\max_{r,r_*,g,u,s,\alpha,\beta} \quad \sum_{i=1}^{n}\sum_{\ell=1}^{L} p_{i\ell}u_{i\ell} + R_{\text{rank}}r_* \tag{15}$$

$$\text{s.t.} \quad \sum_{\ell=1}^{L} u_{i\ell} = 1, \quad \forall i, \tag{16}$$

$$g_i \geq v_{i\ell}r_\ell, \quad \forall i,\ell, \tag{17}$$

$$g_i \leq v_{i\ell}r_\ell + L(1-u_{i\ell}), \quad \forall i,\ell, \tag{18}$$

$$u_{i\ell} \geq 1 - g_i + v_{i\ell}r_\ell, \quad \forall i,\ell, \tag{19}$$

$$u_{i\ell} \leq v_{i\ell}, \quad \forall i,\ell, \tag{20}$$

$$r_\ell = \sum_{k=1}^{L} k s_{\ell k}, \quad \forall \ell, \tag{21}$$

$$\sum_{k=1}^{L} s_{\ell k} = 1, \quad \forall \ell, \tag{22}$$

$$\sum_{\ell=1}^{L} s_{\ell k} = 1, \quad \forall k, \tag{23}$$

$$r_* \geq r_A, \tag{24}$$

$$r_* \geq r_B, \tag{25}$$

$$r_* - r_A \leq (L-1)\alpha, \tag{26}$$

$$r_A - r_* \leq (L-1)\alpha, \tag{27}$$

$$r_* - r_B \leq (L-1)\beta, \tag{28}$$

$$r_B - r_* \leq (L-1)\beta, \tag{29}$$

$$\alpha + \beta = 1, \tag{30}$$

$$u_{i\ell} \leq 1 - \frac{r_* - r_\ell}{L-1}, \quad \forall i,\ell, \tag{31}$$

$$\alpha, u_{i\ell}, s_{\ell k} \in \{0,1\}, \quad \forall i,\ell,k,$$

$$0 \leq \beta \leq 1,$$

$$r_\ell \in \{1,2,\ldots,L\}, \quad \forall \ell.$$

TABLE 5

Observation t_i is represented by $\{1\ 0\ 1\ 1\ 0\}$, and its class is -1. The highest rule that applies is the one ranked 8th ($r_\ell = 8$) since $\{1\ 0\ 1\ 0\ 0\}\subset\{1\ 0\ 1\ 1\ 0\}$ (the rules ranked 10th and 9th do not apply). Thus $u_{i\ell} = 1$ for this rule. This rule has $p_{i\ell} = 1$ since the rule applies to t_i and correctly predicts -1, so the contribution of observation i to the accuracy part of the objective in (15) is $\sum_{\ell=1}^{L} p_{i\ell}u_{i\ell} = 1$.

Observation t_i: **$\{1\ 0\ 1\ 1\ 0\}$, class=-1**			
Ranked rules	$p_{i\ell}$	r_ℓ	$u_{i\ell}$
$\{0\ 1\ 0\ 0\ 1\} \Rightarrow -1$	0	10	0
$\{0\ 1\ 1\ 0\ 0\} \Rightarrow 1$	0	9	0
$\{1\ 0\ 1\ 0\ 0\} \Rightarrow -1$	**1**	**8**	**1**
$\{1\ 0\ 0\ 0\ 1\} \Rightarrow -1$	0	7	0
$\{0\ 0\ 0\ 0\ 0\} \Rightarrow 1$	1	6	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\{0\ 0\ 1\ 1\ 0\} \Rightarrow -1$	-1	1	0

The first term in the objective corresponds to classification accuracy. Given an ordering of rules, the quantity $c_i = \sum_{\ell=1}^{L} p_{i\ell}u_{i\ell}$ equals 1 if the resulting decision list correctly predicts the class of observation i and -1 otherwise. Thus, the number of correct classifications is

$$\sum_{i=1}^{n}\left(\frac{c_i+1}{2}\right) = \frac{1}{2}\left(n + \sum_{i=1}^{n} c_i\right).$$

So to maximize classification accuracy, it suffices to maximize

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n}\sum_{\ell=1}^{L} p_{i\ell}u_{i\ell}.$$

Table 5 shows an example of the parameters ($p_{i\ell}$) and variables (r_ℓ, $u_{i\ell}$) for a particular ranking of rules and observation to be classified.

Constraint (16) enforces that for each i, only one of the $u_{i\ell}$ variables equals one while the rest are zero. To capture the definition of the $u_{i\ell}$, we also use auxiliary variables g_i, which represent the highest rank of the rules such that X_ℓ is in observation i. Through (17) and (18), there is only one ℓ such that $u_{i\ell} = 1$ is feasible, namely the ℓ corresponding to the highest value of $v_{i\ell}r_\ell$. Constraints (19) and (20) help improve the linear relaxation and thus are intended to speed up computation. We assign the integral ranks r_ℓ using (21) through (23), which imply $s_{\ell k} = 1$ if rule ℓ is assigned to rank k. The matching between ranks and rules is one-to-one.

We add regularization in order to favor a shorter overall list of rules. That is, our regularizer pulls the rank of one of the null rules as high as

possible. If r_A is the rank of $\emptyset \Rightarrow -1$ and r_B is the rank of $\emptyset \Rightarrow 1$, then we add r_* to the objective function, where r_* is the maximum of r_A and r_B. The regularization coefficient of r_* in the objective is R_{rank}. We capture r_* using (24) through (30): Either $\alpha = 1$ and $\beta = 0$ or $\beta = 1$ and $\alpha = 0$. If $\alpha = 1$, then $r_* = r_B$. If $\beta = 1$, then $r_* = r_A$. Since we are maximizing, r_* equals the higher of r_A and r_B. Note that if α is binary, then β need not be binary because the constraint $\alpha + \beta = 1$ forces integral values for β. If the rank r_ℓ of rule ℓ is below r_*, then $u_{i\ell} = 0$ for all i, so (31) is also valid, and we include it to help speed up computation.

The Ordered Rules for Classification (ORC) algorithm consists of generating rules using the method shown in Figure 1, computing the $p_{i\ell}$ and $v_{i\ell}$, and then solving (15). The rule generation step could also be replaced by a different method, such as Apriori [Agrawal and Srikant, 1994]. We use our integer optimization approach in the experiments. Note that (13) and (15) are not restricted to binary classification problems; both formulations can be directly applied in the multi-class setting.

5. Computational Results. We used a number of publicly available datasets to demonstrate the performance of our approach. Eight are from the UCI Machine Learning Repository: Breast Cancer Wisconsin (Original), Car Evaluation, Haberman's Survival, Mammographic Mass, MONK's Problem 2, SPECT Heart, Tic-Tac-Toe, and Congressional Voting Records [Asuncion and Newman, 2007]. Crime1 and Crime2 are derived from a study funded by the US Department of Justice [Courtney and Cusick, 2010]. Titanic is from a report on the sinking of the "Titanic" [British Board of Trade, 1990]. For each dataset, we divided the data evenly into three folds and used each fold in turn as a test set, training each time with the other two folds. The training and test accuracy were averaged over these three folds. We compared the ORC algorithm with six other classification methods—logistic regression [see Hastie et al., 2001, Dreiseitl and Ohno-Machado, 2002], Support Vector Machines (SVM) [Vapnik, 1995, Burges, 1998], Classification and Regression Trees (CART) [Breiman et al., 1984], C4.5 [Quinlan, 1993] (J48 implementation), Random Forests [Breiman, 2001], and AdaBoost [Freund and Schapire, 1995]—all run using R 2.15.0. We used the radial basis kernel and regularization parameter $C = 1$ for SVM, and decision trees as base classifiers for AdaBoost. The ORC algorithm was implemented using ILOG AMPL 11.210 with the Gurobi solver.[1]

[1]For B.Cancer, Mammo, MONK2, and TicTacToe, we used Gurobi 3.0.0 on a computer with two Intel quad core Xeon E5440 2.83GHz processors and 32GB of RAM. For the other datasets, we used Gurobi 4.5.2 on a computer with an Intel quad core Xeon E5687 3.60GHz processor and 48GB of RAM.

Here we explain how we chose the parameter settings for the ORC experiments; these parameters were the same for all datasets. In generating rules with (13), we wanted to ensure that $R_{\text{gen\_x}}$ was small enough that the solver would never choose to decrease the scaled support $\sum_{i\in S} x_i$ just to decrease the scaled coverage $\sum_{i=1}^{n} x_i$. That is, $R_{\text{gen\_x}}$ should be such that we would not sacrifice maximizing s for lower s_X; this required only that this parameter be a small positive constant, so we chose $R_{\text{gen\_x}} = \frac{0.1}{n}$. Similarly, we did not want to sacrifice maximizing s or lowering s_X for greater sparsity, so we chose $R_{\text{gen\_b}} = \frac{0.1}{nd}$. In order to not sacrifice classification accuracy for a shorter decision list in ranking the rules with (15), we chose $R_{\text{rank}} = \frac{1}{L}$. We also used a minimum coverage threshold of 0.05, and iterated up to five times at each setting of $\bar{s}_X$ ($\texttt{mincov}_{-1} = \texttt{mincov}_1 = 0.05$, $\texttt{iter\_lim} = 5$); these choices were based on preliminary experiments to determine parameters that would yield a reasonable number of rules.

Table 6 shows the average training and test classification accuracy for each dataset; corresponding standard deviations are in the appendix. Bold indicates the highest average in the row. Table 7 shows the dataset sizes as well as average number of rules generated by RuleGen and average runtimes for our algorithms (±one standard deviation); runtimes for the other methods were too small to be a significant factor in assessment. Time_1 is the total time for generating all rules; Time_2 is the time when the final solution was found, either before solving to optimality or before being terminated after a specified time limit. We generally terminated the solver before (15) solved to provable optimality. The appendix includes more detail about the experiments. Table 8 shows a pairwise comparison of the ORC algorithm to the other algorithms; for each of the other methods, the table contains a count of the number of datasets for which the method is more accurate, equally accurate, or less accurate than ORC. These results show that in terms of accuracy, the ORC algorithm is on par with top classification methods.

6. Interpretability. Interpretability is subjective, but in this section, we aim to demonstrate that the ORC classifier performs well in terms of being easy to understand. Classifiers generated by CART and C4.5 are interpretable because of their decision tree structure. Other methods are not as easily interpreted. For example, the logistic regression model is

$$p = \frac{1}{1 + e^{-\beta_0 + \beta^T t}},$$

where p is the probability that the class of observation t is 1. The SVM model is a hyperplane that maximizes the margin between the hyperplane and the closest point to it from both classes; by using kernels, we can raise

TABLE 6
Classification accuracy (averaged over three folds).

		LR	SVM	CART	C4.5	RF	ADA	ORC
B.Cancer	train	0.9780	0.9846	0.9561	0.9671	0.9876	0.9693	0.9766
	test	0.9502	**0.9619**	0.9488	0.9590	0.9575	0.9605	0.9532
CarEval	train	0.9580	0.9821	0.9659	0.9907	0.9997	0.9959	0.9598
	test	0.9485	0.9728	0.9618	0.9815	0.9826	**0.9890**	0.9508
Crime1	train	0.8427	0.8439	0.8380	0.8932	0.9918	0.8885	0.8897
	test	0.7394	0.7394	0.7488	0.7465	0.7629	0.7723	**0.7817**
Crime2	train	0.6812	0.7477	0.6858	0.7409	0.8211	0.7156	0.7133
	test	**0.6722**	0.6354	0.6171	0.5941	0.6239	0.6630	0.6699
Haberman	train	0.7712	0.7876	0.7680	0.7745	0.7892	0.7712	0.7680
	test	**0.7582**	0.7386	0.7418	0.7386	0.7386	0.7320	**0.7582**
Mammo	train	0.8482	0.8687	0.8422	0.8596	0.8837	0.8560	0.8536
	test	0.8374	0.8217	0.8301	0.8301	0.8289	**0.8422**	0.8337
MONK2	train	0.6470	0.6736	0.7500	0.9317	0.9907	0.7940	0.8299
	test	0.6019	0.6713	0.6690	**0.8866**	0.6528	0.6389	0.7338
SPECT	train	0.8783	0.8633	0.8390	0.8801	0.9363	0.8839	0.8970
	test	0.7978	**0.8464**	0.7828	0.7940	0.8090	0.8052	0.7753
TicTacToe	train	0.9833	0.9494	0.9348	0.9796	1.0000	0.9917	1.0000
	test	0.9823	0.9165	0.8873	0.9259	0.9781	0.9770	**1.0000**
Titanic	train	0.7783	0.7906	0.7862	0.7906	0.7906	0.7862	0.7906
	test	0.7783	0.7847	0.7846	**0.7906**	0.7833	0.7797	**0.7906**
Votes	train	0.9816	0.9747	0.9598	0.9724	0.9954	0.9701	0.9747
	test	**0.9586**	0.9563	0.9540	**0.9586**	**0.9586**	**0.9586**	0.9563

TABLE 7
Number of observations (n), number of items (d), average number of rules generated, average time to generate all rules ($Time_1$), average time to rank rules ($Time_2$).

Dataset	n	d	**#Rules**	**Time**$_1$ (sec)	**Time**$_2$ (sec)
B.Cancer	683	27	198.3 ± 16.2	616.3 ± 57.8	12959.3 ± 1341.9
CarEval	1728	21	58.0	706.3 ± 177.3	7335.3 ± 2083.7
Crime1	426	41	100.7 ± 15.3	496.0 ± 88.6	12364.0 ± 7100.6
Crime2	436	16	27.3 ± 2.9	59.3 ± 30.4	2546.0 ± 3450.6
Haberman	306	10	15.3 ± 0.6	14.7 ± 4.0	6.3 ± 2.3
Mammo	830	25	58.3 ± 1.2	670.7 ± 34.5	3753.3 ± 3229.5
MONK2	432	17	45.3 ± 4.0	124.0 ± 11.5	5314.3 ± 2873.9
SPECT	267	22	145.3 ± 7.2	71.7 ± 9.1	8862.0 ± 2292.2
TicTacToe	958	27	53.3 ± 3.1	1241.3 ± 38.1	4031.3 ± 3233.0
Titanic	2201	8	24.0 ± 1.0	92.0 ± 15.1	1491.0 ± 1088.0
Votes	435	16	266.0 ± 34.8	108.3 ± 5.0	21505.7 ± 1237.2

the dimension of the model and achieve high accuracy, but not interpretability. Though there is work devoted to interpreting SVMs, the result is usually a smaller set of nonlinear features, still within a linear combination [Sonnenburg et al., 2005]. AdaBoost combines weak classifiers—decision trees in our experiments—by minimizing an exponential loss function; thus, even though the base classifiers may be interpretable, the final model is not necessarily

TABLE 8
Number of datasets for which each algorithm exceeded, tied, or fell below ORC in classification accuracy.

	>ORC	**=ORC**	**<ORC**
LR	4	1	6
SVM	3	1	7
CART	2	0	9
C45	5	1	5
RF	4	0	7
ADA	5	0	6

TABLE 9
Average number of leaves for CART and C4.5, average length of decision list for ORC.

	CART	**C4.5**	**ORC**
B.Cancer	5	5.3	12.7
CarEval	14	27.7	12.7
Crime1	11.3	16.3	21
Crime2	8.3	16.3	9.3
SPECT	6.3	7.7	18
Haber	4.7	3.7	4.3
Mammo	4	8.7	12.3
MONK2	14.7	29.3	19.7
TicTacToe	21	34.7	9
Titanic	3.7	5	3.3
Votes	3	4.7	7.3
Average	8.7	14.5	11.8
Standard Deviation	5.8	11.3	5.9

as interpretable. Random Forests also combines trees.

Recall that we want our models to be concise because simpler models are easier to understand. Though we cannot exactly compare conciseness among different types of models, one reasonable measure for decision trees and lists is the number of rules. Table 9 shows the number of rules in the models generated by CART, C4.5, and ORC for each dataset, averaged over the three folds. For CART and C4.5, the number of rules is the number of leaves; for ORC, it is the number of rules above and including the default. In general, the ORC decision lists are larger models than those produced by CART, but can be significantly smaller than those produced by C4.5. The standard deviations in the bottom row of Table 9 also indicate that ORC produces a more *consistently* small model compared to C4.5.

In the remainder of this section, we show the decision lists for a few of the datasets from Section 5 as examples of the interpretability of the decision lists produced by ORC.

TABLE 10
Rules for predicting whether a patient survived at least five years (+5) or not (<5).

	$\bar{s}$	$\bar{s}_X$	**Rule**		
1	80	93	Number of nodes $= 0$	$\Rightarrow$	5+
2	53	83	$1 \leq$ Number of nodes ≤ 9	$\Rightarrow$	5+
3	18	55	$40 \leq$ Age ≤ 49	$\Rightarrow$	<5
4	20	71	$50 \leq$ Age ≤ 59	$\Rightarrow$	<5
5	147	204	Default	$\Rightarrow$	5+

6.1. *Haberman's Survival.* In this dataset, each "observation" represents a patient who underwent surgery for breast cancer. The goal is to predict whether the patient survived at least five years (5+) or not (< 5). Table 10 shows the ORC classifier from training on Folds 1 and 2. We can easily state in words how this classifier makes predictions:

1. If the patient has no more than nine positive axillary nodes, then she is classified as 5+.
2. If she has more than nine nodes and is in her 40s or 50s, then she is classified as <5.
3. Otherwise, she is classified as 5+.

6.2. *Crime.* The Crime1 and Crime2 datasets were derived from a study of crime among youth as they transition to adulthood. There were three waves of interviews—the first when the youth were between 17 and 18 years of age, the second when they were between 19 and 20, and the third when they turned 21. Table 11 shows some of the binary variables from the first two waves. Using the data, we can design a number of prediction problems. The two problems corresponding to Crime1 and Crime2 are:

- **Crime1:** Based on the 41 binary variables from Waves 1 and 2, predict whether or not a youth is arrested between the Wave 2 and Wave 3 interviews. There were 426 observations after removing those with missing values.
- **Crime2:** Based on the 16 variables in the top half of Table 11 that describe the background of the youth, predict whether a youth reports a violent offense at any of the Wave 1, 2, or 3 interviews. There were 432 observations after removing those with missing values.

As an example of the kind of interpretable result we can obtain from the Crime2 data, Table 12 shows the decision list from training on Folds 1 and 3.

TABLE 11
Variable descriptions.

Variable	Description
Female	Respondent is female
Male	Respondent is male
Hispanic	Respondent is hispanic
White	Respondent is white
Black	Respondent is black
OtherRace	Respondent is mixed or other race
AlcoholOrSubstanceAbuse	Alcohol or substance abuse diagnosis
MentalHealthDiagnosis	Mental health diagnosis
TeenageParent	Respondent is a teenage parent
SexAbuseVictim	Victim of sex abuse
InFosterCare_W1	In foster care at Wave 1
InKinshipCare_W1	In kinship care at Wave 1
InGroupCare_W1	In group care at Wave 1
IndependentOtherCare_W1	Independent living or other care at Wave 1
NoMomOrStepmom	No mom or stepmom
NoDadOrStepdad	No dad or stepdad
PropertyDamage_W1	Deliberately damaged property at Wave 1
StoleOver50_W1	Stole something worth >$50 at Wave 1
SoldMarijuanaOrDrugs_W1	Sold marijuana or other drugs at Wave 1
BadlyInjuredSomeone_W1	Badly injured someone at Wave 1
UsedWeapon_W1	Used or threatened to use a weapon at Wave 1
ViolentOffense_W1	Violent offense at Wave 1
NonviolentOffense_W1	Nonviolent offense at Wave 1
ViolentOffense_W2	Violent offense at Wave 2
NonviolentOffense_W2	Nonviolent offense at Wave 2
ArrestedBetweenW1andW2	Arrested since Wave 1 at Wave 2
InSchool_W1	In school at Wave 1
Employed_W1	Employed at Wave 1
InSchool_W2	In school at Wave 2
Employed_W2	Employed at Wave 2

TABLE 12
Rules for predicting whether there is a violent offense in any wave (1) or not (−1).

	s	s_X	**Rule**		
1	29	41	InGroupCare_W1	⇒	1
2	13	19	Female, SexAbuseVictim, IndependentOtherCare_W1	⇒	−1
3	13	16	AlcoholOrSubstanceAbuse, MentalHealthDiagnosis	⇒	1
4	14	26	Female, Black, InFosterCare_W1	⇒	−1
5	103	154	Black	⇒	1
6	32	53	Female, White	⇒	−1
7	41	58	AlcoholOrSubstanceAbuse	⇒	1
8	119	291	Default	⇒	−1

6.3. *Titanic.* Each row of this dataset represents one of the 2201 passengers aboard the Titanic, a passenger liner that sank in 1912 after striking an iceberg. The features of the dataset are: social class (first, second, third, crew), age (adult, child), and gender (male or female). We want to predict whether or not each passenger survived. Table 13 shows the decision list from training on Folds 2 and 3. This result makes sense in light of the "women and children first" policy and the fact that little effort was made to help the third-class passengers.

TABLE 13
Rules for predicting whether a passenger survived the Titanic sinking (1) or not (−1).

	s	s_X	**Rule**		
1	341	462	Third Class	⇒	−1
2	888	1108	Adult, Male	⇒	−1
3	477	1467	Default	⇒	1

6.4. *Tic-Tac-Toe.* Our final example is the Tic-Tac-Toe dataset. Each data point represents a board configuration at the end of a Tic-Tac-Toe game where player x played first, and the classification problem is to identify whether player x won. This is an easy task for a human, who needs only to determine if there are three x's in a row. There are nine features in the original data, each representing a square on a Tic-Tac-Toe board. The possible values for each feature are: x, o, or b (player x, player o, or blank).

This example demonstrates that for certain datasets, the ORC algorithm may have a substantial advantage by optimizing both accuracy and conciseness. Figure 3 shows the CART classifier from training on Folds 1 and 2. The notation o.5 means 'o' in box 5, x.7 means 'x' in box 7, etc. Figure 4 shows the C4.5 classifier. The ORC classifier, shown in Figure 5, decides the class of a board the same way a typical human would: if the board has three x's in a row, which can occur in eight different configurations, then player x wins; otherwise, player x does not win. It achieves perfect accuracy in training and testing; the accuracies of CART and C4.5 are about 0.94 and 0.99 respectively for training and 0.88 and 0.93 respectively for testing. The ORC classifier is much more concise than either of those produced by CART or C4.5. It has only nine rules, versus 21 for CART and 36 for C4.5.

7. Large-Scale Models. MIO is computationally intensive, as illustrated in Table 7. Nevertheless, even for large datasets, our MIO approach has the potential to order rules into an accurate decision list. The runtimes in the Time_2 column of Table 7 could in fact be substantially shorter if we

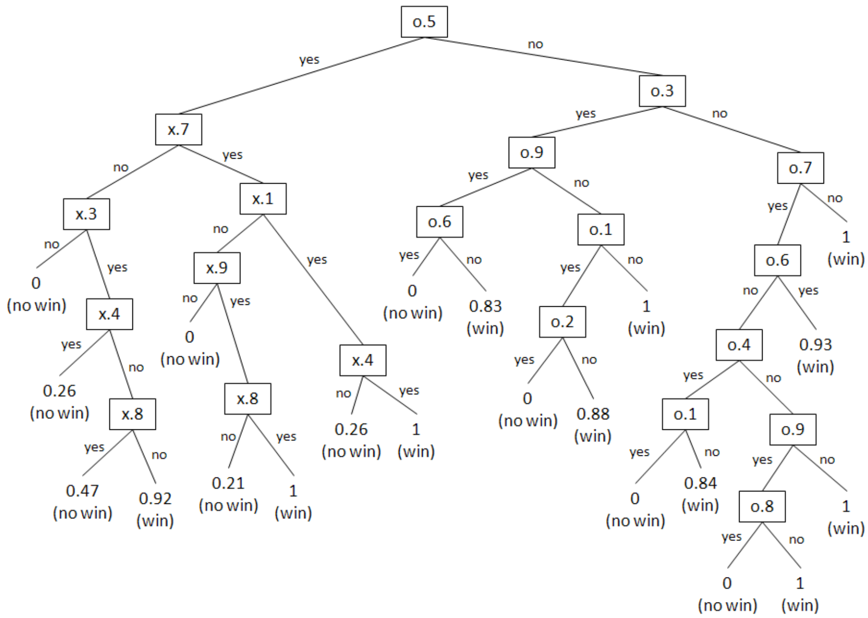

FIG 3. *CART Classifier for Tic-Tac-Toe dataset (predicted class in parentheses).*

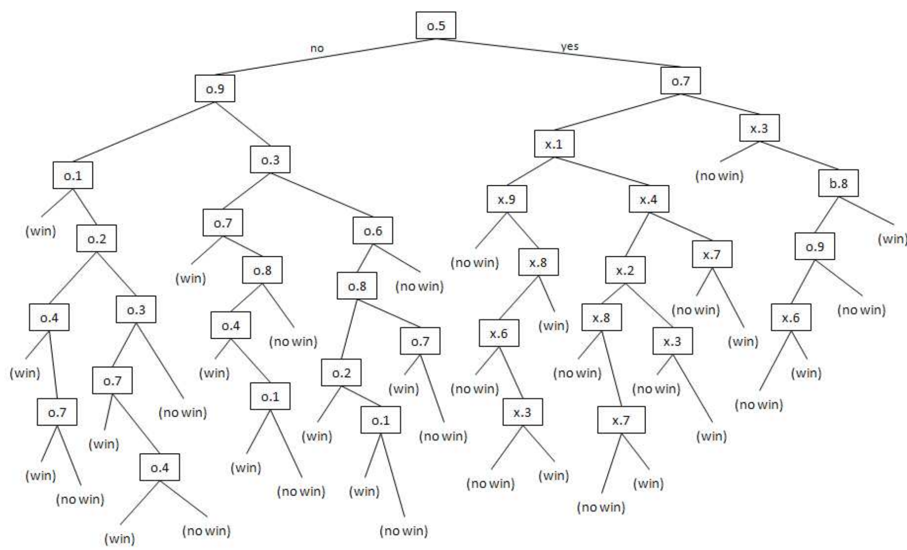

FIG 4. *C4.5 Classifier for Tic-Tac-Toe dataset (predicted class in parentheses), left branch always means 'no' and right branch always means 'yes.'*

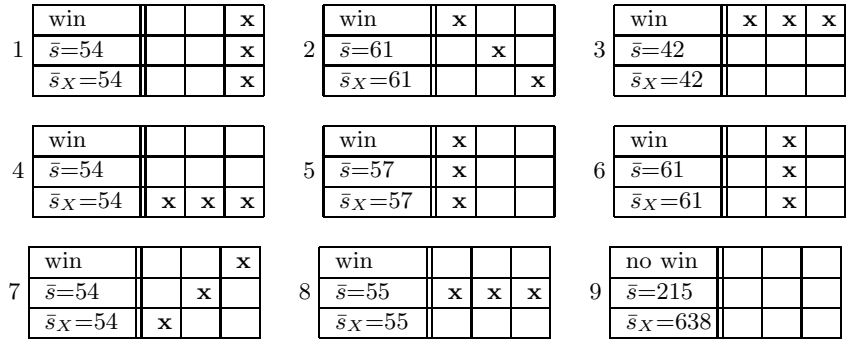

FIG 5. *Rules for predicting whether player x wins a Tic-Tac-Toe game.*

were seeking just a good solution rather than an optimal solution. The MIO solver typically finds a good solution quickly, but takes longer to improve upon it and to finally prove optimality. As it was not our goal to obtain a solution in the shortest time possible, we allowed the solver to search for the best solution it could find in a reasonable amount of time. For larger datasets than those shown in Section 5, we can run (15) just until we obtain a good solution, or let it run for a longer period of time to obtain better solutions. As the speed of computers continues to advance exponentially, and also as cloud computing becomes more accessible, we expect our MIO approach to produce high quality solutions with decreasing computing effort.

In this section, we show results on an additional dataset from the UCI Repository, Wine Quality, which has $n = 4898$ observations and $d = 44$ items, yielding a data table with $nd = 215,512$ entries. The largest dataset in Section 5 was CarEval, which had $nd = 36,288$. The large size of the Wine dataset caused long runtimes for (13), which we did want to solve to optimality to obtain rules on the frontier. Thus we used Apriori to generate the rules instead. We ran Apriori using R, and specified minimum support thresholds of 0.006 and 0.1 for the positive and negative class rules respectively to generate a reasonable number of rules. Otherwise, our experimental setup was the same as for the other datasets; we divided the data into three folds, and to rank the rules, we used (15) with $R_{\text{rank}} = \frac{1}{L}$. We let (15) run for approximately 20 hours. Figure 6 shows the accuracy of the solutions found by the solver over time. (Train12 refers to training on Folds 1 and 2, Train13 refers to training on Folds 1 and 3, and Train23 refers to training on Folds 2 and 3.) The figure illustrates how a good solution is often found relatively quickly but then improves only slowly. Averaged over three folds, we generated 71.3 rules, and the time to find the final solution of (15) before termination was 57258.3 seconds. Table 14 shows the accuracy of the

algorithms. ORC achieves about the same accuracy as SVM and AdaBoost, though Random Forests achieved the highest accuracy for this dataset.

In terms of interpretability, ORC produces a much more concise model than C4.5. Averaged over three folds, the C4.5 trees have 184.7 leaves, whereas the ORC lists have 25.3 rules in addition to higher test accuracy than C4.5. (CART was even more concise, with an average of 4.7 leaves, but it lost accuracy.) This is another example of the consistency of the ORC algorithm in producing classification models that have interpretability advantages and compete well against the best methods in accuracy.

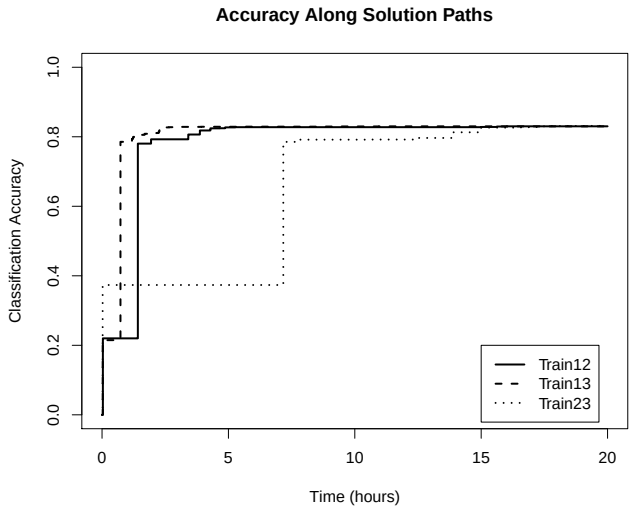

FIG 6. *Accuracy of solutions found by ORC algorithm over time for Wine data.*

TABLE 14
Classification accuracy for Wine dataset (averaged over three folds).

		LR	SVM	CART	C4.5	RF	ADA	ORC
Wine	train	0.8040	0.8460	0.7850	0.9129	0.9918	0.8432	0.8304
	test	0.7987	0.8105	0.7842	0.8085	**0.8608**	0.8105	0.8103

8. Conclusion. In this work, we developed algorithms for producing interpretable, yet accurate, classifiers. The classifiers we build are decision lists, which use association rules as building blocks. Both of the challenges addressed in this work, namely the task of mining interesting rules, and the task of ordering them, have always been hampered by "combinatorial explosion." Even with a modest number of items in the dataset, there may

be an enormous number of possible rules, and even with a modest number of rules, there are an enormous number of ways to order them. On the other hand, MIO methods are naturally suited to handle such problems; they not only encode the combinatorial structure of rule mining and rule ordering problems, but also are able to capture the new forms of regularization introduced in this work, that is, favoring more compact rules and shorter lists. Our computational experiments show that ORC competes well in terms of accuracy against the top classification algorithms on a variety of datasets. In our paper, we used only one setting of the parameters for all of the experiments to show that even an "untuned" version of our algorithm performs well; however, by varying these parameters, it may be possible to achieve still better predictive performance. Since our paper is among the first to use MIO methods for machine learning, and in particular to create decision lists using optimization-based (non-heuristic) approaches, it opens the door for further research on how to use optimization-based approaches for rule mining, creating interpretable classifiers, and handling new forms of regularization.

APPENDIX A: DETAILS FROM COMPUTATIONAL EXPERIMENTS

As explained in Section 3, each observation in our data is represented by a binary vector. Six of the datasets used in our experiments had only categorical variables: CarEval, MONK2, SPECT, TicTacToe, Titanic, and Votes. Thus it was straightforward to transform them into binary features. Here we describe how we transformed the other datasets:

1. **Breast Cancer Wisconsin (Original).** The dataset has 699 rows. There are 683 remaining observations after removing rows with missing values. There are nine original features, each taking integer values between 1 and 10. We used categorical variables to capture whether each feature is between 1 and 4, 5 and 7, or 8 and 10.
2. **Crime1 and Crime2.** The derivation of these datasets is described in Section 6.
3. **Haberman's Survival.** The dataset has three features: age, year of operation, and number of positive axillary nodes. We removed the second feature since it did not seem to be predictive of survival and thus would not contribute to interpretability. We split the age feature into five bins by decades: 39 and under, 40 to 49, 50 to 59, 60 to 69, and 70 and over. We also split the nodes feature into five bins: none, 1 to 9, 10 to 19, 20 to 29, and at least 30.
4. **Mammographic Mass.** The dataset has 961 rows, each representing a patient. There are 830 remaining observations after removing rows

with missing values. The only feature that is not categorical is patient age, which we split into seven bins: 29 and under, 30 to 39, 40 to 49, 50 to 59, 60 to 69, 70 to 79, and 80 and over.

5. **Wine Quality.** We used the data for white wine. All features were continuous, so we binned them by quartiles. A wine was in class 1 if its quality score between 7 and 10, and class -1 if its quality score was between 1 and 6.

Table 15 shows the standard deviations that correspond to the averages in Table 6. Table 16 shows the following results for each dataset (Train12 refers to training on Folds 1 and 2, Train13 refers to training on Folds 1 and 3, and Train23 refers to training on Folds 2 and 3): $\mathbf{L_{-1}}$ and $\mathbf{L_1}$ are the numbers of rules generated by RuleGen for class -1 and class 1 respectively. Time_1 is the total time for generating all $\mathbf{L_{-1}} + \mathbf{L_1}$ rules; Time_2 is the time when the final solution was found, either before solving to optimality or before being terminated after a specified amount of time. Table 16 also shows the time limit we used for each of the different datasets. For some datasets, the time limit was significant longer than Time_2, illustrating how it can take a long time for the solver to reach provable optimality even though the solution appears to have converged. Note that in running (15) for the TicTacToe dataset, Train12 solved to optimality in 1271 seconds; Train12 and Train23 had optimality gaps of about 0.02% and 0.01% respectively when the final solutions were found.

Table 17 shows the average classification accuracy for three methods, two of which are SVM and ORC from Table 6. The other is a tuned version of SVM, where we varied the C parameter and chose the one with the best average test performance in hindsight. The overall performance of the untuned ORC algorithm is still on par with that of the tuned SVM algorithm.

REFERENCES

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases*, pages 487–499, 1994.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

Martin Anthony. Decision lists. Technical report, CDAM Research Report LSE-CDAM-2005-23, 2005.

A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/ mlearn/MLRepository.html`.

Roberto J. Bayardo and Rakesh Agrawal. Mining the most interesting rules. In *Proceedings*

TABLE 15
Standard deviation of classification accuracy.

		LR	SVM	CART	C4.5	RF	ADA	ORC
B.Cancer	train	0.0114	0.0022	0.0110	0.0137	0.0064	0.0116	0.0108
	test	0.0417	0.0142	0.0091	0.0167	0.0198	0.0274	0.0091
CarEval	train	0.0027	0.0018	0.0035	0.0018	0.0005	0.0018	0.0093
	test	0.0027	0.0066	0.0046	0.0044	0.0076	0.0044	0.0036
Crime1	train	0.0108	0.0318	0.0070	0.0073	0.0020	0.0054	0.0073
	test	0.0244	0.0141	0.0147	0.0070	0.0267	0.0041	0.0186
Crime2	train	0.0217	0.0169	0.0484	0.0273	0.0066	0.0219	0.0258
	test	0.0559	0.0369	0.0366	0.0402	0.0376	0.0438	0.0700
Haberman	train	0.0247	0.0221	0.0221	0.0321	0.0225	0.0172	0.0242
	test	0.0442	0.0204	0.0453	0.0484	0.0283	0.0204	0.0442
Mammo	train	0.0136	0.0088	0.0076	0.0036	0.0020	0.0089	0.0165
	test	0.0249	0.0245	0.0217	0.0097	0.0115	0.0240	0.0202
MONK2	train	0.0256	0.0035	0.0284	0.0361	0.0053	0.0231	0.0217
	test	0.0526	0.0145	0.0729	0.0743	0.0208	0.0139	0.0356
SPECT	train	0.0399	0.0366	0.0227	0.0117	0.0032	0.0141	0.0471
	test	0.0297	0.0619	0.0425	0.0507	0.0195	0.0234	0.0389
TicTac	train	0.0080	0.0133	0.0047	0.0072	0.0000	0.0048	0.0000
	test	0.0148	0.0262	0.0061	0.0066	0.0156	0.0095	0.0000
Titanic	train	0.0037	0.0054	0.0055	0.0054	0.0054	0.0087	0.0054
	test	0.0074	0.0143	0.0106	0.0108	0.0120	0.0101	0.0108
Votes	train	0.0190	0.0020	0.0105	0.0103	0.0020	0.0121	0.0072
	test	0.0276	0.0080	0.0159	0.0138	0.0000	0.0000	0.0080
Wine	train	0.0037	0.0019	0.0047	0.0107	0.0013	0.0009	0.0007
	test	0.0142	0.0050	0.0055	0.0029	0.0053	0.0120	0.0035

of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 145–154, 1999.

Endre Boros, Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan, Eddy Mayoraz, and Ilya Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):292–306, 2000.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

British Board of Trade. Report on the loss of the 'Titanic' (s.s.). In *British Board of Trade Inquiry Report (reprint)*. Gloucester, UK: Allan Sutton Publishing, 1990.

Oliver Büchter and Rüdiger Wirth. Discovery of association rules over ordinal data: A new and faster algorithm and its application to basket analysis. 1394:36–47, 1998.

Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

Mark E. Courtney and Gretchen Ruth Cusick. Crime during the transition to adulthood: How youth fare as they leave out-of-home care in Illinois, Iowa, and Wisconsin, 2002-2007. Technical report, Ann Arbor, MI: Inter-university Consortium for Political and Social Research [distributor], 2010.

Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35:352–359, 2002.

Yoav Freund and Robert Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Computational Learning Theory*, 904:23–37, 1995.

TABLE 16
Number of rules generated for the negative (L_{-1}) and positive (L_1) classes, average time in seconds to generate all rules ($Time_1$), average time in seconds to rank rules ($Time_2$), time limit on ranking rules.

		$\mathbf{L_{-1}}$	$\mathbf{L_1}$	$\mathbf{Time_1}$	$\mathbf{Time_2}$	**Limit on Time$_2$**
B.Cancer	Train12	123	66	551	12802	14400 (4 hours)
	Train13	136	53	637	11703	
	Train23	135	82	661	14373	
CarEval	Train12	45	13	598	8241	21600 (6 hours)
	Train13	45	13	610	4952	
	Train23	48	10	911	8813	
Crime1	Train12	60	58	582	15481	18000 (5 hours)
	Train13	70	25	501	17373	
	Train23	43	46	405	4238	
Crime2	Train12	17	12	94	6489	7200 (2 hours)
	Train13	15	14	47	1071	
	Train23	13	11	37	78	
Haber	Train12	6	10	19	9	600 (10 minutes)
	Train13	4	11	11	5	
	Train23	5	10	14	5	
Mammo	Train12	33	26	637	1340	10800 (3 hours)
	Train13	29	28	706	2498	
	Train23	29	30	669	7422	
MONK2	Train12	28	21	135	2786	10800 (3 hours)
	Train13	30	16	125	8440	
	Train23	27	14	112	4717	
SPECT	Train12	10	127	73	9611	10800 (3 hours)
	Train13	6	144	62	10686	
	Train23	10	139	80	6289	
TicTacToe	Train12	11	43	1278	1232	10800 (3 hours)
	Train13	14	36	1202	3292	
	Train23	12	44	1244	7570	
Titanic	Train12	13	10	75	295	7200 (2 hours)
	Train13	15	10	104	1756	
	Train23	15	9	97	2422	
Votes	Train12	110	116	103	22899	25200 (7 hours)
	Train13	137	146	109	20536	
	Train23	141	148	113	21082	
Wine	Train12	34	41	–	58978	74000 (∼20 hours)
	Train13	35	34	–	39483	
	Train23	35	35	–	73314	

Jerome H. Friedman and Bogdan E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: a survey. *ACM Computing Surveys*, 38, September 2006.

Bart Goethals. Survey on frequent pattern mining. Technical report, Helsinki Institute for Information Technology, 2003.

Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15:55–86, 2007.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, New York, 2001.

Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association

TABLE 17
Results of tuned SVM (highest in row highlighted in bold).

Dataset		**SVM** (default $C = 1$)	**SVM** (hindsight)	C	**ORC**
B.Cancer	train	0.9846 ± 0.0022	0.9868 ± 0.0044	1.2	0.9766 ± 0.0108
	test	0.9619 ± 0.0142	$\mathbf{0.9634 \pm 0.0203}$		0.9532 ± 0.0091
CarEval	train	0.9821 ± 0.0018	1	7.2	0.9598 ± 0.0093
	test	0.9728 ± 0.0066	$\mathbf{0.9988 \pm 0.0010}$		0.9508 ± 0.0036
Crime1	train	0.8439 ± 0.0318	0.9648 ± 0.0035	4.6	0.8897 ± 0.0073
	test	0.7394 ± 0.0141	0.7512 ± 0.0361		$\mathbf{0.7817 \pm 0.0186}$
Crime2	train	0.7477 ± 0.0169	0.6996 ± 0.0216	0.4	0.7133 ± 0.0258
	test	0.6354 ± 0.0369	0.6445 ± 0.0391		$\mathbf{0.6699 \pm 0.0700}$
Haberman	train	0.7876 ± 0.0221	0.7761 ± 0.0279	0.4	0.7680 ± 0.0242
	test	0.7386 ± 0.0204	$\mathbf{0.7582 \pm 0.0442}$		$\mathbf{0.7582 \pm 0.0442}$
Mammo	train	0.8687 ± 0.0088	0.8608 ± 0.0095	0.6	0.8536 ± 0.0165
	test	0.8217 ± 0.0245	0.8313 ± 0.0197		$\mathbf{0.8337 \pm 0.0202}$
MONK2	train	0.6736 ± 0.0035	0.6736 ± 0.0035	1	0.8299 ± 0.0217
	test	0.6713 ± 0.0145	0.6713 ± 0.0145		$\mathbf{0.7338 \pm 0.0356}$
SPECT	train	0.8633 ± 0.0366	0.8745 ± 0.0319	1.2	0.8970 ± 0.0471
	test	0.8464 ± 0.0619	$\mathbf{0.8502 \pm 0.0566}$		0.7753 ± 0.0389
TicTacToe	train	0.9494 ± 0.0133	0.9901 ± 0.0009	6	1
	test	0.9165 ± 0.0262	0.9844 ± 0.0143		**1**
Titanic	train	0.7906 ± 0.0054	0.7906 ± 0.0054	1	0.7906 ± 0.0054
	test	0.7847 ± 0.0143	0.7847 ± 0.0143		$\mathbf{0.7906 \pm 0.0108}$
Votes	train	0.9747 ± 0.0020	0.9793 ± 0.0034	1.4	0.9747 ± 0.0072
	test	0.9563 ± 0.0080	$\mathbf{0.9586 \pm 0.0069}$		0.9563 ± 0.0080
Wine	train	0.8460 ± 0.0019	0.9117 ± 0.0079	4.8	0.8304 ± 0.0007
	test	0.8105 ± 0.0050	$\mathbf{0.8348 \pm 0.0035}$		0.8103 ± 0.0035

rule mining - a general survey and comparison. *SIGKDD Explorations*, 2:58–64, June 2000.

Dennis L. Jennings, Teresa M. Amabile, and Lee Ross. Informal covariation assessments: Data-based versus theory-based judgements. In Daniel Kahneman, Paul Slovic, and Amos Tversky, editors, *Judgment Under Uncertainty: Heuristics and Biases,*, pages 211–230. Cambridge Press, Cambridge, MA, 1982.

Adam R. Klivans and Rocco A. Servedio. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, 7:587–602, 2006.

Wenmin Li, Jiawei Han, and Jian Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. *IEEE International Conference on Data Mining*, pages 369–376, 2001.

Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 80–96, 1998.

Philip M. Long and Rocco A. Servedio. Attribute-efficient learning of decision lists and linear threshold functions under unconcentrated distributions. In *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 921–928, 2007.

Mario Marchand and Marina Sokolova. Learning with decision lists of data-dependent features. *Journal of Machine Learning Research*, 2005.

Ken McGarry. A survey of interestingness measures for knowledge discovery. *The Knowledge Engineering Review*, 20:39–61, 2005.

Nicolai Meinshausen. Node harvest. *The Annals of Applied Statistics*, 4(4):2049–2072, 2010.

George A. Miller. The magical number seven, plus or minus two: Some limits to our capacity for processing information. *The Psychological Review*, 63(2):81–97, 1956.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

Ulrich Rückert. *A Statistical Approach to Rule Learning*. PhD thesis, Technischen Universität München, 2008.

Cynthia Rudin, Benjamin Letham, Ansaf Salleb-Aouissi, Eugen Kogan, and David Madigan. Sequential event prediction with association rules. In *Proceedings of the 24th Annual Conference on Learning Theory (COLT)*, 2011.

György J. Simon, Vipin Kumar, and Peter W. Li. A simple statistical model and association rule filtering for classification. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 823–831, 2011.

Marina Sokolova, Mario Marchand, Nathalie Japkowicz, and John Shawe-Taylor. The decision list machine. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 921–928, 2003.

Sören Sonnenburg, Gunnar Rätsch, and Christin Schäfer. Learning interpretable svms for biological sequence classification. 3500(995):389–407, 2005.

Pang-Ning Tan and Vipin Kumar. Interestingness measures for association patterns: a perspective. Technical report, Department of Computer Science, University of Minnesota, 2000.

Fadi Thabtah. A review of associative classification mining. *The Knowledge Engineering Review*, 22:37–65, March 2007.

Koen Vanhoof and Benoît Depaire. Structure of association rule classifiers: a review. In *Proceedings of the International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pages 9–12, 2010.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 344–353, 2004.

Xiaoxin Yin and Jiawei Han. CPAR: Classification based on predictive association rules. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 331–335, 2003.

Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139
E-mail: dbertsim@mit.edu
aachang@mit.edu
rudin@mit.edu