2016 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 13–16, 2016, SALERNO, ITALY

LEARNING SPARSE TWO-LEVEL BOOLEAN RULES

Guolong Su

Massachusetts Institute of Technology
Cambridge, MA, USA

Dennis Wei, Kush R. Varshney, Dmitry M. Malioutov

IBM Thomas J. Watson Research Center
Yorktown Heights, NY, USA

ABSTRACT

This paper develops a novel optimization framework for learning accurate and sparse two-level Boolean rules for classification, both in Conjunctive Normal Form (CNF, i.e. AND-of-ORs) and in Disjunctive Normal Form (DNF, i.e. OR-of-ANDs). In contrast to opaque models (e.g. neural networks), sparse two-level Boolean rules gain the crucial benefit of interpretability, which is necessary in a wide range of applications such as law and medicine and is attracting considerable attention in machine learning. This paper introduces two principled objective functions to trade off classification accuracy and sparsity, where 0-1 error and Hamming loss are used to characterize accuracy. We propose efficient procedures to optimize these objectives based on linear programming (LP) relaxation, block coordinate descent, and alternating minimization. We also describe a new approach to rounding any fractional values in the optimal solutions of LP relaxations. Experiments show that our new algorithms based on the Hamming loss objective provide excellent tradeoffs between accuracy and sparsity with improvements over state-of-the-art methods.

Index Terms— Sparse Boolean Classifier, Linear Programming Relaxation

1. INTRODUCTION

Rule-based classifiers have wide applications in various signal processing fields such as speech recognition [1], smart grid [2], and expert systems [3]. As an important subclass of such classifiers, a Boolean rule connects a subset of binary input features with the logical operators conjunction ("AND"), disjunction ("OR"), and negation ("NOT") to form the prediction.

This paper considers learning Boolean classifiers from a training dataset, where the classifiers have the form of two-level Boolean rules in CNF (AND-of-ORs) or DNF (OR-of-ANDs) [4]. In the lower level of a CNF (DNF), disjunctions (conjunctions) of binary features build *clauses* and in the upper level, the conjunction (disjunction) of the clauses forms the predictor. Such two level Boolean rules are a very expressive model. In fact, if the input features are binary and we include all negations, then two-level rules can represent any Boolean function of the input features [5]; moreover, continuous valued features may be incorporated by converting them to binary with threshold comparisons.

Inspired by the fruitful exploration of sparsity in topics such as Boolean compressive sensing and group testing [6], this work aims to build sparse Boolean rules. Sparsity in this context yields the benefit of *interpretability*, which is attracting considerable attention in machine learning and has substantial importance in a wide range of applications such as law and medicine [7, 8]. In these fields, predictions from classification models are generally presented to a human decision maker/agent who makes the final decision. Such a decision maker often needs an understanding of the reasons for the prediction before accepting the result; thus, high prediction accuracy without providing the reasons is not sufficient for the model to be trusted. For example, convincing reasons are legally required in fraud detection [8] to establish a claim of fraud.

Learning two-level Boolean rules that are both accurate and sparse is inherently combinatorial and NP-hard in some cases [9]. As we overview in Section 2, existing methods are mostly heuristic and/or greedy and there has been limited work on principled yet tractable approaches. This paper fills the gap by introducing a unified optimization framework for learning two-level Boolean rules that achieve good balance between accuracy and sparsity. We propose two formulations. The objective function in the first formulation is a weighted combination of the total number of classification errors and the sparsity, based on which we develop an LP relaxation approach. The second formulation replaces the 0-1 classification error with the Hamming distance from the current rule to the closest rule that correctly classifies a sample. Based on this second formulation, block coordinate descent and alternating minimization algorithms are developed. Experiments show that two-level rules can have considerably higher accuracy than one-level rules. The two algorithms based on the Hamming distance formulation obtain competitive tradeoffs between accuracy and sparsity with improvements over cutting edge approaches. In addition, this work tackles the issue of fractional optimal solutions to LP relaxations and introduces a new binarization method to convert LP solutions into binary values.

The remainder of this paper is organized as follows. Section 2 reviews existing work on two-level Boolean rule learning. After the problem formulations in Section 3, optimization approaches are introduced in Section 4 and evaluated in Section 5. Section 6 concludes this work.

2. REVIEW OF EXISTING WORK

The two-level Boolean rules in this work are examples of sparse decision rule lists [10], which have been extensively studied in machine learning and a number of strategies have been proposed [5]. The covering strategy [10, 11] sequentially constructs each clause in a two-level rule; in each step, it learns a new clause generally in a greedy manner, and then removes the newly covered samples or adjusts the weights on all samples for future steps. The bottom-up strategy [12] successively combines more specific clauses into more general clauses according to local criteria like pairwise similarity. A more flexible multi-phase strategy [13] is to first discover a large set of candidate *local patterns* (i.e. clauses), then heuristically select a subset of informative clauses, and finally construct a two-level rule by considering the selected clauses as new binary features. A fourth strategy is to convert trained decision trees into decision lists [14].

978-1-5090-0746-2/16/$31.00

Unlike our proposed approach, the above strategies lack a single, principled objective function to drive rule learning. Moreover, they employ heuristics that leave room for improvements on both accuracy and rule simplicity.

There has been some prior work on optimization-based formulations for rule learning. To our best knowledge, the most relevant prior work is [15], where an LP formulation using the Boolean compressed sensing framework is proposed to learn a clause, based on which two other algorithms are used for rule set learning, namely set covering and boosting. Although we apply this clause learning method as a component in some of our algorithms, there are significant differences between the current work and [15]. Compared with a single clause which represents only a subset of Boolean functions, two-level rules are significantly more expressive (i.e. they can represent all Boolean functions) and much more challenging to learn. In addition, the greedy style of the set covering method in [15] leaves room for improvement, and the additive combination of clauses in boosted classifiers reduces interpretability. Another work on DNF learning [16] provides a mixed integer program (MIP) formulation named OOA and a different heuristic formulation OOAx. The MIP in OOA is similar to our first formulation with the 0-1 error but without the relaxation to LP, which may thus result in quite high computational complexity. OOAx is similar to the multi-phase strategy [13] in using heuristic approaches to discover and select clauses before an optimization formulation is used to build a DNF with the selected clauses.

3. PROBLEM FORMULATION

We consider a standard binary supervised classification setting, where a training dataset has n labeled samples; the i^{th} sample has a binary label $y_i \in \{0,1\}$ and in total d binary features[1] $a_{i,j} \in \{0,1\}$ $(1 \le j \le d)$. The goal is to learn a classifier $\hat{y}(\cdot)$ in the Conjunctive Normal Form (AND-of-ORs) that can generalize well to unseen feature vectors sampled from the same distribution as the training dataset. In the lower level of the rule, we form each clause by the disjunction of a selected subset of input features; in the upper level, the final predictor is formed by the conjunction of all clauses. Denote the maximum number of clauses by R. Letting the binary decision variables $w_{j,r} \in \{0,1\}$ represent whether to include the j^{th} feature in the r^{th} clause, the output of the r^{th} clause for the i^{th} sample is

$$\hat{v}_{i,r} = \bigvee_{j=1}^{d} \left(a_{i,j} w_{j,r}\right), \text{ for } 1 \le i \le n,\ 1 \le r \le R. \tag{1}$$

Then, the predictor $\hat{y}_i$ satisfies

$$\hat{y}_i = \bigwedge_{r=1}^{R} \hat{v}_{i,r}, \text{ for } 1 \le i \le n. \tag{2}$$

To mitigate the need for careful specification of the model parameter R, a mechanism to "disable" a clause can be introduced to reduce the total number of actual clauses if the assigned R is too large. For a CNF rule, a clause can be regarded as disabled if its output is always 1. Thus, we can pad the input feature matrix with a trivial "always true" feature $a_{i,0} = 1$ for all samples, and also include the corresponding decision variables $w_{0,r}$ for all clauses; if $w_{0,r} = 1$, then the r^{th} clause has output 1 and is thus disabled in the CNF rule. The sparsity cost for $w_{0,r}$ can be lower than other variables or even zero. This mechanism can also be regarded as a part of the regularization, whereby the cost of activating a clause is weighed against the accuracy improvement that it brings.

In certain cases, DNF rules could be more natural than CNF. A CNF learning algorithm can learn DNF by De Morgan's laws:

$$y = \bigvee_{r=1}^{R} \left(\bigwedge_{j \in \mathcal{C}_r} x_j \right) \Leftrightarrow \overline{y} = \bigwedge_{r=1}^{R} \left(\bigvee_{j \in \mathcal{C}_r} \overline{x}_j \right)$$

where $\overline{y}$ and $\overline{x}_j$ mean the negation of binary variables y and x_j, respectively, and $\mathcal{C}_r$ is the index set of features selected in the r^{th} clause. To learn a DNF rule with a CNF learning algorithm, we can first negate both features and labels of all samples, then learn a CNF rule with the negated features and labels, and finally use the decision variables $w_{j,r}$ with the original features to construct a DNF rule. Since the formulations of CNF are slightly more concise, Sections 3 and 4 focus on CNF only.

Two formulations are introduced with different accuracy costs in Section 3.1 and 3.2, respectively.

3.1. Formulation with 0-1 Error

A natural choice for the accuracy term in the objective is the total number of misclassifications (i.e. 0-1 error for each sample). With the sparsity cost as the sum of the number of features used in each clause, a formulation is as below

$$\min_{w_{j,r}} \sum_{i=1}^{n} |\hat{y}_i - y_i| + \theta \cdot \sum_{r=1}^{R} \sum_{j=1}^{d} w_{j,r} \tag{3}$$

$$\text{s.t. } \hat{y}_i = \bigwedge_{r=1}^{R} \left(\bigvee_{j=1}^{d} \left(a_{i,j} w_{j,r}\right) \right), \text{ for } 1 \le i \le n, \tag{4}$$

$$w_{j,r} \in \{0,1\}, \text{ for } 1 \le j \le d,\ 1 \le r \le R.$$

3.2. Formulation with Minimal Hamming Distance

Instead of using the 0-1 error to measure accuracy, it may be desirable to have a more fine-grained measure such as the minimal Hamming distance. As an example, consider two CNF rules, both with two clauses, predicting the same sample with ground truth label $y_i = 1$. Suppose both clauses in the first rule predict 0, while only one clause in the second rule predicts 0 and the other predicts 1. Although both rules misclassify this sample after taking "AND" of their two clauses, the second rule is closer to correct than the first one. If we use an iterative algorithm to refine the learned rule, it might be beneficial for the accuracy cost term to favor the second rule in this example, which could push the solution towards being correct. An additional motivation for the Hamming loss is to avoid identical (and thus redundant) clauses by training each clause with a different subset of samples, as done in [10, 11].

In this second formulation, the accuracy cost for a single sample is the minimal Hamming distance from a given CNF rule to an *ideal* CNF rule, where the latter means a rule that correctly classifies this sample. The Hamming distance between two CNF rules is the total number of $w_{j,r}$ that are different in the two rules. An intuitive explanation of this minimal Hamming distance is the smallest number of modifications (i.e. negations) of the current rule $w_{j,r}$ that are needed to correct a misclassification on a sample, i.e. how far is the rule from being correct.

For mathematical formulation, we introduce *ideal clause outputs* $v_{i,r}$ with $1 \le i \le n$ and $1 \le r \le R$ to represent a CNF rule that

[1] We assume the negation of each feature is included as another input feature; if not, we can pad the input features with negations.

correctly classifies the i^{th} sample. The values of $v_{i,r}$ are always consistent with the ground truth labels, i.e. $y_i = \bigwedge_{r=1}^{R} v_{i,r}$ for all $1 \le i \le n$. We let $v_{i,r}$ have a ternary alphabet $\{0, 1, \text{DC}\}$, where $v_{i,r} = \text{DC}$ means that we "don't care" about the value of $v_{i,r}$. With this setup, if $y_i = 1$, then $v_{i,r} = 1$ for all $1 \le r \le R$; if $y_i = 0$, then $v_{i,r_0} = 0$ for at least one value of r_0, and we can have $v_{i,r} = \text{DC}$ for all $r \ne r_0$. In implementation, $v_{i,r} = \text{DC}$ implies the removal of the i^{th} sample in the training or updating for the r^{th} clause, which leads to a different training subset for each clause.

Denote η_i as the minimal Hamming distance from the current CNF rule $w_{j,r}$ to an ideal CNF rule for the i^{th} sample. We derive η_i for positive and negative samples, respectively. Since $y_i = 1$ implies $v_{i,r} = 1$ for all r, for each clause with output 0 in the current rule, at least one positive feature needs to be included to match $v_{i,r} = 1$. Thus, the minimal Hamming distance for a positive sample is the number of clauses with output 0:

$$\eta_i = \sum_{r=1}^{R} \max\left\{0, \left(1 - \sum_{j=1}^{d} a_{i,j} w_{j,r}\right)\right\}, \text{ for } y_i = 1.$$

For $y_i = 0$, we first consider for fixed r the minimal Hamming distance between the r^{th} clauses only of the current rule and an ideal rule where $v_{i,r} = 0$. We need to negate $w_{j,r}$ in the current rule for j with $w_{j,r} = a_{i,j} = 1$ to match $v_{i,r} = 0$, and thus the minimal Hamming distance of this clause is $\sum_{j=1}^{d} a_{i,j} w_{j,r}$. Then, since $v_{i,r} = 0$ needs to hold for at least one value of r while all other $v_{i,r}$ can be DC, the minimal Hamming distance of the CNF rule is given by the minimum over r, i.e. setting $v_{i,r_0} = 0$ with

$$r_0 = \underset{1 \le r \le R}{\arg\min} \left(\sum_{j=1}^{d} a_{i,j} w_{j,r}\right). \tag{5}$$

Combining all analysis above, the new formulation with the minimal Hamming distance cost is as below

$$\min_{w_{j,r}} \sum_{i=1}^{n} \eta_i + \theta \cdot \sum_{r=1}^{R} \sum_{j=1}^{d} w_{j,r} \tag{6}$$

$$\text{s.t. } \eta_i = \sum_{r=1}^{R} \max\left\{0, \left(1 - \sum_{j=1}^{d} a_{i,j} w_{j,r}\right)\right\}, \text{ for } y_i = 1,$$

$$\eta_i = \min_{1 \le r \le R} \left(\sum_{j=1}^{d} a_{i,j} w_{j,r}\right), \text{ for } y_i = 0, \tag{7}$$

$$w_{j,r} \in \{0, 1\}, \text{ for } 1 \le j \le d, \ 1 \le r \le R.$$

The binary variables $w_{j,r}$ can be relaxed to $0 \le w_{j,r} \le 1$. The minimum over r in (7) implies the non-convexity of such continuous relaxation with $R > 1$, making the exact solution challenging.

Letting $R = 1$ in formulation (6), it can be seen that we recover the formulation for one-level rule learning in [15].

To simplify description of algorithms later, we show a formulation (8) below, which is equivalent to (6) but involves both $v_{i,r}$ and $w_{j,r}$. Taking the minimization over $v_{i,r}$ in (8) with fixed $w_{j,r}$ eliminates the variables $v_{i,r}$, and (8) becomes identical to (6).

$$\min_{w_{j,r},\, v_{i,r}} \sum_{i=1}^{n} \sum_{r=1}^{R} \left[\mathbb{1}_{v_{i,r}=1} \cdot \max\left\{0, \left(1 - \sum_{j=1}^{d} a_{i,j} w_{j,r}\right)\right\} + \mathbb{1}_{v_{i,r}=0} \cdot \sum_{j=1}^{d} a_{i,j} w_{j,r}\right] + \theta \cdot \sum_{r=1}^{R} \sum_{j=1}^{d} w_{j,r} \tag{8}$$

$$\text{s.t.} \bigwedge_{r=1}^{R} v_{i,r} = y_i, \text{ for } 1 \le i \le n, \tag{9}$$

$$v_{i,r} \in \{0, 1, \text{DC}\}, \text{ for } 1 \le i \le n, \ 1 \le r \le R,$$

$$w_{j,r} \in \{0, 1\}, \text{ for } 1 \le j \le d, \ 1 \le r \le R.$$

4. OPTIMIZATION APPROACHES

This section discusses various optimization approaches to the two-level rule learning problem. Based on the formulation in Section 3.1, we develop an LP relaxation approach in Section 4.1. Based on the formulation in Section 3.2, we propose the block coordinate descent algorithm in Section 4.2 and the alternating minimization algorithm in Section 4.3 for the objective (8). All algorithms use LP relaxations and the dimensions of the resulting LPs are analyzed in Section 4.4. Section 4.5 considers the binarization problem for the case of non-binary solutions to LPs.

4.1. Two-level Linear Programming Relaxation

This approach considers the 0-1 error formulation (3) and applies the idea of replacing binary operations "AND" and "OR" with linear-algebraic operations. Since "AND" and "OR" are defined only on binary inputs, there are various interpolations of these functions to the continuous domain, and both convex and concave interpolations exist for both operators. The "OR" function has the following interpolations [17]

$$\bigvee_{j=1}^{d} x_j = \max_{1 \le j \le d} \{x_j\} = \min\left\{1, \sum_{j=1}^{d} x_j\right\},$$

where the first is convex and the second is concave, both of which are the respective tightest interpolations. The logical "AND" operator also has the tightest convex and concave interpolations as [17]

$$\bigwedge_{j=1}^{d} x_j = \max\left\{0, \left(\sum_{j=1}^{d} x_j\right) - (d-1)\right\} = \min_{1 \le j \le d} \{x_j\}.$$

Since the predictor $\hat{y}_i$ of the two-level rule in (4) is a composition of "AND" and "OR" operators, it is possible to properly interpolate it using both a convex function and a concave function by composing the individual interpolations of the two operators. From (1) and (2), a convex interpolation of $\hat{y}_i$ is

$$\hat{y}_i = \max\left\{0, \left(\sum_{r=1}^{R} \max_{1 \le j \le d} \{a_{i,j} w_{j,r}\}\right) - (R-1)\right\},$$

and a concave interpolation can be obtained similarly.

Denote the 0-1 error cost for the i^{th} sample as $\psi_i \triangleq |\hat{y}_i - y_i|$. Since the errors ψ_i in (3) should be minimized, if $y_i = 1$, then $\psi_i = 1 - \hat{y}_i$ and thus we need the concave interpolation for $\hat{y}_i$; if $y_i = 0$, then $\psi_i = \hat{y}_i$ and thus the convex interpolation is needed.

Finally, the formulation in (3) can be exactly converted into a mixed integer program:

$$\begin{aligned}\min_{w_{j,r},\psi_i,\beta_{i,r}} \quad & \sum_{i=1}^{n}\psi_i + \theta\cdot\sum_{r=1}^{R}\sum_{j=1}^{d} w_{j,r} \qquad (10)\\ \text{s.t. } & \psi_i \geq 0,\ \forall i,\\ & \psi_i \geq 1-\sum_{j=1}^{d} a_{i,j}w_{j,r},\ \text{for } y_i=1,\ \forall r,\\ & \psi_i \geq \left(\sum_{r=1}^{R}\beta_{i,r}\right)-(R-1),\ \text{for } y_i=0,\\ & \beta_{i,r} \geq a_{i,j}w_{j,r},\ \text{for } y_i=0,\ \forall j,\ \forall r,\\ & w_{j,r}\in\{0,1\},\ \forall j,\ \forall r.\end{aligned}$$

Relaxing the decision variables to $0 \leq w_{j,r} \leq 1$ leads to an LP.

Unfortunately, numerical results suggest that this LP relaxation is likely to have fractional values in the optimal solution $w_{j,r}$, possibly due to the gap between the convex and concave interpolations.

4.2. Block Coordinate Descent Algorithm

We now propose an algorithm which considers the decision variables in a single clause ($w_{j,r}$ with a fixed r) as a block of coordinates, and performs block coordinate descent to minimize the Hamming distance cost function in (8). Each iteration updates a single clause with all the other clauses fixed, using the one-level rule learning algorithm in [15]. We denote r_0 as the clause to be updated.

The optimization of (8) even with $(R-1)$ clauses fixed still involves a joint minimization over w_{j,r_0} and the ideal clause outputs $v_{i,r}$ for $y_i = 0$ ($v_{i,r} = 1$ for $y_i = 1$ and thus fixed), so the exact solution could still be challenging. To simplify, we fix the values of $v_{i,r}$ for $y_i = 0$ and $r \neq r_0$ to the actual clause outputs $\hat{v}_{i,r}$ in (1) with the current $w_{j,r}$ ($r \neq r_0$). Now we assign v_{i,r_0} for $y_i = 0$: if there exists $v_{i,r} = \hat{v}_{i,r} = 0$ with $r \neq r_0$, then this sample is guaranteed to be correctly classified and we can assign $v_{i,r_0} = \text{DC}$ to minimize the objective in (8); in contrast, if $\hat{v}_{i,r} = 1$ holds for all $r \neq r_0$, then the constraint (9) requires $v_{i,r_0} = 0$.

This derivation leads to the updating process as follows. To update the r_0^{th} clause, we remove all samples that have label $y_i = 0$ and are already predicted as 0 by at least one of the other $(R-1)$ clauses, and then update the r_0^{th} clause with the remaining samples using the one-level rule learning algorithm.

There are different choices of which clause to update in an iteration, such as cyclical or random updating. We can also try the update for each clause and then greedily choose the one with the minimum cost, as is used in our experiments.

The initialization of $w_{j,r}$ also has different choices. For example, one option is the set covering method, as is used in our experiments. Random or all-zero initialization can also be used.

4.3. Alternating Minimization Algorithm

This section proposes the alternating minimization algorithm that uses the Hamming distance formulation (8). It alternately minimizes with respect to the decision variables $w_{j,r}$ and the ideal clause outputs $v_{i,r}$. Each iteration has two steps: update $v_{i,r}$ with the current $w_{j,r}$, and update $w_{j,r}$ with the new $v_{i,r}$. The latter step is simpler and will be first discussed.

With fixed values of $v_{i,r}$, the minimization over $w_{j,r}$ is relatively straight-forward: the objective in (8) is separated into R terms, each of which depends only on a single clause $w_{j,r}$ with a fixed r. Thus, all clauses are decoupled in the minimization over $w_{j,r}$, and the problem becomes parallel learning of R one-level clauses. Explicitly, the update of the r^{th} clause removes samples with $v_{i,r} = \text{DC}$ and then uses the one-level rule learning algorithm.

The update over $v_{i,r}$ with fixed $w_{j,r}$ follows the discussion in Section 3.2: for positive samples $y_i = 1$, $v_{i,r} = 1$, and for the negative samples $y_i = 0$, $v_{i,r_0} = 0$ for r_0 defined in (5) and $v_{i,r} = \text{DC}$ for $r \neq r_0$. For negative samples with a "tie", i.e. non-unique r_0 in (5), tie breaking is achieved by a "clustering" approach. First, for each clause $1 \leq r_0 \leq R$, we compute its cluster center in the feature space by taking the average of $a_{i,j}$ (for each j) over samples i for which r_0 is minimal in (5) (including ties). Then, each sample with a tie is assigned to the clause with the closest cluster center in ℓ_1-norm among all minimal r_0 in (5).

Similar to the block coordinate descent algorithm, various options exist for initializing $w_{j,r}$ in this algorithm. The set covering approach is used in our experiments.

4.4. Complexity of LP Formulations

We now consider computational complexity of the proposed algorithms by characterizing the dimensions of the LP formulations. If we denote n_1 and n_0 as the numbers of samples with $y_i = 1$ and $y_i = 0$, respectively, then the two-level LP in (10) has $O(Rd + Rn_0 + n)$ variables and $O(Rn_1 + Rdn_0)$ constraints. The block coordinate descent and the alternating minimization algorithms are both iterative, and require solving R LPs per iteration. However, each single LP does not use all the training samples; if n_r denotes the number of samples used for updating a clause, then the LP to update that clause has $O(d+n_r)$ variables and $O(d+n_r)$ constraints. Thus, despite having to solve multiple LPs, the reduction in the dimensions of each single LP can still result in greater overall efficiency compared with the non-iterative two-level LP formulation.

4.5. Redundancy Aware Binarization

If the optimal solution to LP turns out to have fractional values, then we need to convert them into binary. Empirically, the straightforward binarization method of comparing each $w_{j,r}$ from LP with a specified threshold may result in *redundant* and unnecessarily complex rules.

The following improved binarization method considers two types of redundancies, each associated with a set of binary features that we call a redundancy set. Among the features in each redundancy set, no more than one feature will appear in any single clause of an optimal CNF rule[2].

The first type of redundancy set corresponds to *nested* features. If binary features a_{i,j_1} and a_{i,j_2} satisfy $a_{i,j_1} \leq a_{i,j_2}$ for all samples, then these two features cannot both appear in a single clause in the optimal CNF rule; otherwise, removing a_{i,j_1} from the clause keeps the same output and improves the sparsity, leading to a better rule. Among a nested set $a_{i,j_1} \leq a_{i,j_2} \leq \ldots \leq a_{i,j_P}$ ($\forall 1 \leq i \leq n$), at most one feature can be selected in a single clause.

The second type consists of over-complementary binary features a_{i,j_1} and a_{i,j_2} that satisfy $a_{i,j_1} \bigvee a_{i,j_2} = 1$ ($\forall i$). It applies when we use the mechanism to "disable" a clause. The optimal CNF rule cannot have both a_{i,j_1} and a_{i,j_2} in a single clause, otherwise disabling this clause keeps the output and improves sparsity.

[2]This statement holds for both formulations (3) and (8); for simplicity, we will focus on the formulation (3) for illustration.

Table 1. 10-fold Average Test Error Rates (unit: %). Standard Error of the Mean is Shown in Parentheses.

DATASET	TLP	BCD	AM	OCRL	SC	DLIST	C5.0	CART
ILPD	28.6(0.3)	**28.6(0.2)**	**28.6(0.2)**	**28.6(0.2)**	**28.6(0.2)**	36.5(1.4)	30.5(2.0)	32.8(1.3)
IONOS	8.3(1.2)	9.4(1.1)	11.4(1.1)	9.7(1.5)	10.5(1.3)	19.9(2.3)	**7.4(2.1)**	10.8(1.2)
LIVER	44.9(0.9)	37.1(3.2)	39.1(2.5)	45.8(2.2)	41.7(2.8)	45.2(2.6)	**36.5(2.4)**	37.1(2.5)
PARKIN	14.4(1.4)	**12.8(2.2)**	15.9(2.9)	16.4(2.1)	14.9(1.9)	25.1(3.3)	16.4(2.7)	13.9(2.9)
PIMA	26.8(1.8)	26.8(1.7)	**23.8(2.0)**	27.2(1.5)	27.9(1.5)	31.4(1.6)	24.9(1.7)	27.3(1.5)
SONAR	31.3(3.2)	29.8(3.0)	25.5(2.4)	34.6(2.7)	28.8(2.9)	38.5(3.6)	**25.0(4.2)**	31.7(3.5)
TRANS	23.8(0.8)	23.8(0.1)	23.8(0.1)	23.8(0.1)	23.8(0.1)	35.4(2.4)	**21.7(1.2)**	25.4(1.7)
WDBC	7.6(1.1)	**6.2(1.2)**	6.5(0.9)	9.3(2.0)	8.8(2.0)	9.7(0.8)	6.5(1.1)	8.4(1.0)

The new binarization approach takes the above types of redundancies into account. For illustration, suppose all binary features are obtained by thresholding continuous valued features. For a given clause and a single continuous valued feature, we may sweep over all non-redundant combinations of the binary features induced by this continuous feature and obtain the one with minimal cost. We can show that the total number of such non-redundant combinations is quadratic with the number of thresholds, which guarantees efficient sweeping. To avoid the combinatorial joint minimization for multiple continuous features, we first sort continuous features in decreasing order as determined by the sum of corresponding decision variables in the optimal solution to the LP relaxation. Then the decision variables corresponding to each continuous feature are sequentially binarized as described above.

5. NUMERICAL EVALUATION

5.1. Setup

This section evaluates the algorithms with UCI repository datasets [18]. To facilitate comparison with the most relevant prior work [15], we use the same 8 datasets as in that work: Indian liver patient dataset (ILPD), Ionosphere (Ionos), BUPA liver disorders (Liver), Parkinsons (Parkin), Pima Indian diabetes (Pima), connectionist bench sonar (Sonar), blood transfusion service center (Trans), and breast cancer Wisconsin diagnostic (WDBC). Each continuous valued feature is converted to binary using 10 quantile-based thresholds.

The goal is to learn a DNF rule (OR-of-ANDs) from each dataset. We use stratified 10-fold cross validation and then average the test and training error rates. All LPs are solved by IBM CPLEX version 12. The sparsity parameter is $\theta = A \times 10^B$ where we sweep $A = 1, 2, 5$ and $B = -4, -3, -2, -1, 0, 1$, for a total of 18 values. We use the redundancy aware binarization and the mechanism to "disable" a clause.

Algorithms in comparison and their abbreviations are: two-level LP relaxation (TLP), block coordinate descent (BCD), alternating minimization (AM), one-level conjunctive rule learning (OCRL, equivalent to setting $R = 1$ for BCD or AM) and set covering (SC), the last two from [15], decision list in IBM SPSS (DList), and decision trees (C5.0: C5.0 with rule set option in IBM SPSS, CART: classification and regression trees algorithm in Matlab's classregtree function). The maximum number of iterations in BCD and AM is set as 100. Without loss of generality, we set the maximum number of clauses $R = 5$ for TLP, BCD, AM, and SC.

We show the test error rates, the sparsity of the rules, and Pareto fronts indicating the tradeoff between accuracy and sparsity.

5.2. Accuracy and Rule Simplicity

In this section, we apply a second cross validation within the training partition to choose the optimal parameter θ among the 18 values that has the highest accuracy, and then evaluate its performance on the test partition. The mean test error rates and the standard error of the mean are listed in Table 1. Due to space constraints, we refer the reader to [15] for results from other classifiers that are not interpretable; the accuracy of our algorithms is generally quite competitive with them.

Table 2 provides the 10-fold average of the sparsity of the learned rules as a measure for interpretability. No features are counted if a clause is disabled; therefore, disabling all clauses makes the total number of features as 0. The best algorithm for each dataset is highlighted in bold in both tables.

Table 2. 10-fold Average Numbers of Features

DATASET	TLP	BCD	AM	SC	DLIST	C5.0
ILPD	4.8	**0.0**	**0.0**	**0.0**	5.4	45.5
IONOS	30.9	12.4	12.9	11.1	**7.7**	13.6
LIVER	6.5	9.3	7.7	5.2	**2.2**	46.6
PARKIN	9.0	8.2	12.6	3.2	**2.1**	16.6
PIMA	15.3	2.2	**2.0**	2.4	8.6	38.2
SONAR	27.8	14.2	23.6	9.0	**1.9**	27.3
TRANS	3.5	**0.0**	**0.0**	**0.0**	3.8	6.7
WDBC	21.8	13.6	11.9	8.7	**4.0**	15.8

Table 1 shows that two-level rules obtained by our algorithms (TLP, BCD, and AM) are more accurate than the one-level rules from OCRL for most datasets, which demonstrates the expressiveness of two-level rules.

Among optimization-based two-level rule learning approaches, BCD and AM generally have superior accuracy to TLP and SC (with the numbers of wins/ties/loses as: BCD vs. TLP 4/3/1, AM vs. TLP 4/2/2, BCD vs. SC 5/2/1, AM vs. SC 4/2/2). All these four approaches substantially outperform DList in terms of accuracy on all datasets. Compared with C5.0, BCD and AM obtain significantly more sparse rules (i.e. higher interpretability) with quite competitive accuracy. Compared with CART, BCD has higher or equal accuracy on all datasets, and AM is also superior overall. In addition, AM achieves the highest accuracy on the Pima dataset among the interpretable models in Table 1, and BCD obtains the highest accuracy on the Parkin and WDBC datasets.

5.3. Pareto Fronts

A DNF rule can be considered *dominated* by another rule if the former has higher error rate and uses more features than the latter. For a fixed algorithm, if we learn DNF rules with each of the 18

values of θ, then the accuracy-sparsity pairs of the DNF rules that are not dominated by any other DNF rule constitute the *Pareto front* for this algorithm, which shows the optimal tradeoff boundary in the accuracy-sparsity space achieved by varying the regularization parameter. The Pareto fronts of the BCD, AM, and SC algorithms are shown in Fig. 1, where we include the results for both using and not using the clause disabling mechanism for each algorithm. For ease of visualization, the dominated points are not shown and non-dominated points are connected by line segments. We use the Liver and Pima datasets as illustrating examples.

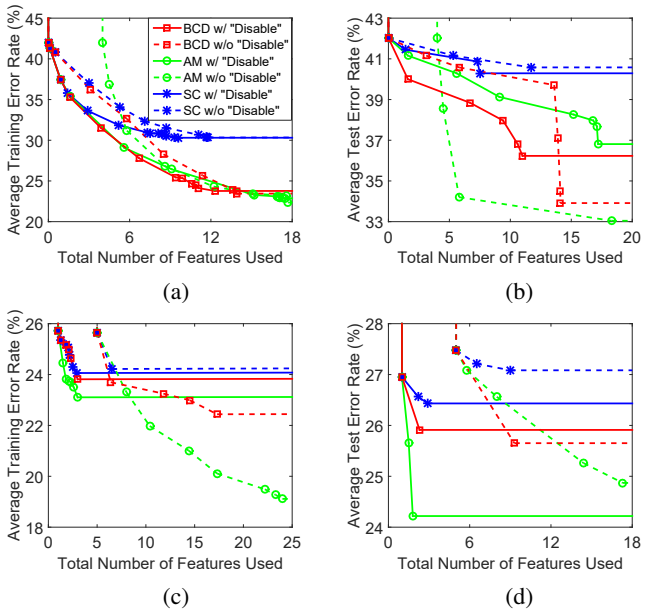

Fig. 1. Pareto Fronts: (a) Liver Training Error Rate, (b) Liver Test Error Rate, (c) Pima Training Error Rate, (d) Pima Test Error Rate. All Figures Use the Same Legends as in (a).

Comparing the Pareto fronts in Fig. 1, we can have the following observations. For the more difficult Liver dataset where $R = 5$ clauses are not too many, using or not using clause disabling (i.e. more or less regularization) accesses different parts of the accuracy-sparsity space as shown in Fig. 1 (b). In contrast, $R = 5$ is already unnecessarily large for the simpler Pima dataset, so clause disabling allows a much improved tradeoff (Pareto fronts to the lower left in Fig. 1 (d)). In addition, the Pareto fronts clearly show the superiority of BCD and AM to SC.

6. CONCLUSION

Motivated by the impact of sparsity and the need for interpretable classification models, this paper has provided two optimization-based formulations for two-level Boolean rule learning, the first based on 0-1 classification error and the second on Hamming distance. These complement the more heuristic strategies in the literature on two-level Boolean rules.

Numerical results show that two-level Boolean rules typically have considerably lower error rate than one-level rules. In addition, the proposed block coordinate descent and alternating minimization algorithms provide excellent tradeoffs between accuracy and sparsity with improvements over state-of-the-art approaches.

7. ACKNOWLEDGMENT

The authors thank for V. S. Iyengar, A. Mojsilović, K. N. Ramamurthy, and E. van den Berg for conversations and support.

8. REFERENCES

[1] D. J. Litman, M. A. Walker, and M. S. Kearns, "Automatic detection of poor speech recognition at the dialogue level," in *Proc. Annu. Meet. Assoc. Comput. Linguist.*, June 1999, pp. 309–316.

[2] R. Mitchell and R. Chen, "Behavior-rule based intrusion detection systems for safety critical smart grid applications," *IEEE Trans. Smart Grid*, vol. 4, no. 3, pp. 1254–1263, Sept. 2013.

[3] A. Abraham, "Rule-based expert systems," *Handbook Meas. Syst. Des.*, 2005.

[4] G. Su, D. Wei, K. R. Varshney, and D. M. Malioutov, "Interpretable two-level boolean rule learning for classification," in *Proc. ICML Workshop Human Interpret. Mach. Learn.*, June 2016, pp. 66–70.

[5] J. Fürnkranz, D. Gamberger, and N. Lavrač, *Foundations of rule learning*, Springer Science & Business Media, Nov. 2012.

[6] G. K. Atia and V. Saligrama, "Boolean compressed sensing and noisy group testing," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1880–1901, Mar. 2012.

[7] A. A. Freitas, "Comprehensible classification models – a position paper," *ACM SIGKDD Explor.*, vol. 15, no. 1, pp. 1–10, Mar. 2014.

[8] V. S. Iyengar, K. B. Hermiz, and R. Natarajan, "Computer-aided auditing of prescription drug claims," *Health Care Manag. Sci.*, vol. 17, no. 3, pp. 203–214, Sept. 2014.

[9] M. Kearns, M. Li, L. Pitt, and L. Valiant, "On the learnability of Boolean formulae," in *Proc. Annu. ACM Symp. on Theory of Comput.*, Jan. 1987, pp. 285–295.

[10] R. L. Rivest, "Learning decision lists," *Mach. Learn.*, vol. 2, no. 3, pp. 229–246, Nov. 1987.

[11] W. W. Cohen, "Fast effective rule induction," in *Proc. Int. Conf. Mach. Learn.*, July 1995, pp. 115–123.

[12] P. Domingos, "Unifying instance-based and rule-based induction," *Mach. Learn.*, vol. 24, no. 2, pp. 141–168, Aug. 1996.

[13] A. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz, "From local patterns to global models: The LeGo approach to data mining," in *Proc. ECML PKDD Workshop (LeGo-08)*, Sept. 2008, pp. 1–16.

[14] J. R. Quinlan, "Simplifying decision trees," *Int. J. Man-Mach. Studies*, vol. 27, no. 3, pp. 221–234, Sept. 1987.

[15] D. M. Malioutov and K. R. Varshney, "Exact rule learning via Boolean compressed sensing," in *Proc. Int. Conf. Mach. Learn.*, June 2013, pp. 765–773.

[16] T. Wang and C. Rudin, "Learning optimized Or's of And's," *arXiv preprint arXiv:1511.02210*, Nov. 2015.

[17] R. Hähnle, "Proof theory of many-valued logic–linear optimization–logic design: Connections and interactions," *Soft Comput.*, vol. 1, no. 3, pp. 107–119, Sept. 1997.

[18] M. Lichman, "UCI machine learning repository," http://archive.ics.uci.edu/ml, Univ. of Calif., Irvine, 2013.