Classification Rules in Relaxed Logical Form

Bishwamittra Ghosh[1] and **Dmitry Malioutov**[2] and **Kuldeep S. Meel**[1]

Abstract. ML algorithms that produce rule-based predictions in Conjunctive Normal form (CNF) or in Disjunctive Normal form (DNF) are arguably some of the most interpretable ones. Although CNF/DNF rules are considered interpretable in practice, propositional logic has other very interpretable representations which are more expressive. In this paper, we generalize CNF/DNF rules and introduce *relaxed*-CNF rules, which is motivated by the popular usage of checklists in the medical domain. We consider relaxed definitions of standard OR/AND operators which allow exceptions in the construction of a clause and also in the selection of clauses in a rule. While the combinatorial structure of relaxed-CNF rules offers exponential succinctness, the naive learning techniques are computationally expensive.

The primary contribution of this paper is to propose a novel incremental mini-batch learning procedure, called CRR, that employs advances in the combinatorial solvers and efficiently learns relaxed-CNF rules. Our experimental analysis demonstrates that CRR can generate relaxed-CNF rules which are more accurate compared to CNF rules and sparser compared to decision lists.

1 Introduction

The widespread adoption of prediction systems in various safety-critical domains such as medical diagnosis, law, education, and many others has led to the increased importance of presenting the decision functions in interpretable representations [24, 12, 26, 27, 19]. To enable safe, robust and trustworthy integration of such systems, the end users require them to support interpretability, privacy, and fairness in decision-making [6, 32, 28, 30]. In this context, rule-based representations are particularly effective for presenting the decision functions to people specifically in the medical domain [13, 16, 29, 31]. A recent body of work has studied sparsity-inducing objectives for classification rules in CNF (Conjunctive Normal Form) or DNF (Disjunctive Normal Form) and demonstrated that they often achieve high interpretability (defined in terms of rule-sparsity) with minimal sacrifice in classification accuracy [16, 13, 10]. Although CNF/DNF rules are considered interpretable, they are less expressive compared to the Boolean cardinality constraints in propositional logic, wherein a Boolean cardinality constraint allows one to express numerical bounds on Boolean variables [23]. In this work, we introduce relaxed-CNF, a classification rule in propositional logic that takes the benefits of both worlds: represents the decision boundary in an interpretable manner similar to CNF/DNF and is more expressive for allowing Boolean cardinality constraints in its representation.

We find the motivation of designing relaxed-CNF rules from checklists, where a checklist is a list of things one needs to check, e.g., a list of items to take on a travel trip. There are several practical applications of checklists in interpretable decision making, for example, CHADS2 score in medicine, which is a clinical prediction rule for estimating the risk of stroke. Another example is a psychometric test, known as MyersBriggs Type Indicator (MBTI) [2], which indicates differing psychological preferences in how people perceive the world around them and make decisions. An influential study on the importance of checklists [9] finds that highly complex and specialized problems can be handled smoothly by the development and consistent usage of checklists. In this paper, we aim to learn classification rules in the form of a checklist, which we call relaxed-CNF rules.

We now provide an introduction to relaxed-CNF. The simplest logical rules are single level-rules: ORs or ANDs of a subset of *literals*, where a literal can be assigned 1 (true) or 0 (false). An OR operator requires 1 out of N literals to be assigned to 1, while an AND operator requires all N out of N literals to be assigned to 1. A *clause* is a collection of literals connected by OR/AND. A CNF formula is a conjunction (AND) of clauses where each clause is a disjunction (OR) of literals, while a DNF formula is a disjunction of clauses where each clause is a conjunction of literals. Therefore, CNF and DNF formulas can be viewed as two-level rules. In this paper, we consider a richer set of logical formulas that capture the structure of checklists. We consider *hard*-OR clauses, where at least $M > 1$ out of N literals need to 1, and we similarly define *soft*-AND clauses which allow some of the literals (at most $N - M$) to be 0. To be more precise, the definitions of hard-OR and soft-AND overlap. We use the term hard-OR when $M \leq N/2$ and soft-AND otherwise. To extend the standard definition of CNF (which is ANDs of ORs), we define *relaxed*-CNF to denote soft-ANDs of hard-ORs. Similarly, *relaxed*-DNF is hard-ORs of soft-ANDs. Since hard-OR and soft-AND are differentiated based on the value of M and N, relaxed-CNF and relaxed-DNF have a similar structural representation. In early work, Craven and Shavlik [4] considered single level M-of-N rules to explain black-box neural-network classifiers. Recently, Emad et al. [7] have developed a semi-quantitative group testing approach for learning sparse single level M-of-N rules, which are quite restrictive in their ability to fit the data. In this work, we study a much richer family of two-level relaxed-CNF rules.

Relaxed-CNF rules are more flexible than pure CNF rules and can accurately fit more complex classification boundaries. For example, relaxed-CNF clauses allow compact encoding of the majority function, which would require exponentially many clauses in CNF, showing the exponential gap in the succinctness of the two representations. In addition, relaxed-CNF and CNF rules have the same functional form where a user has to compute the sum of true literals/clauses and then compare the sum to different thresholds. From the computational perspective, the structural flexibility of relaxed-CNF rules compared to CNF/DNF rules makes it harder to learn, which poses the following question, "*can we design a combinatorial*

[1] School of Computing, National University of Singapore
[2] T.J. Watson IBM Research center

framework to efficiently learn relaxed-CNF rules?"

The primary contribution of this paper is an affirmative answer to the above question by proposing an efficient combinatorial framework for learning relaxed-CNF rules which we call CRR (**C**lassification **R**ules in **R**elaxed form). In CRR, we construct an objective function to maximize both prediction accuracy and rule-sparsity and design a MILP (mixed-integer linear programming) formulation for learning the optimal relaxed-CNF rules. To learn a k-clause relaxed-CNF rule (say $\mathcal{R}$) using the direct (naive) MILP formulation, the size of the MILP query expressed as the number of constraints is $\mathcal{O}(n \cdot k)$, where the number of features/attributes in the dataset is fixed and n is the number of training samples. Consequently, the naive MILP formulation fails to handle large datasets. To address scalability concerns, we propose an efficient mini-batch training methodology based incremental approach for learning $\mathcal{R}$. The proposed incremental approach significantly enhances the scalability of CRR.

Through a comprehensive experimental evaluation over datasets from the UCI and Kaggle repository, we observe that CRR with relaxed-CNF rules achieves an improved trade-off between accuracy and rule sparsity and scales to large datasets. More significantly, CRR can generate relaxed-CNF rules which have higher accuracy than CNF rules generated by [10]. Also, compared to decision lists generated by [3], relaxed-CNF rules are sparser in large datasets.

Example 1.1. As an example, we illustrate a relaxed-CNF rule learned for Breast Cancer Wisconsin Diagnostic Dataset (WDBC) from UCI repository. The dataset is used to predict whether the tumor is malignant or benign based on the characteristics of a tumor cell. We can learn the following rule for detecting malignant cancer.

Tumor is diagnosed as malignant if,
$[\{(\text{ smoothness } \geq 0.089) + (\text{standard error of area } \geq 53.78) +$
$(\text{largest radius } \geq 18.225)\} \geq 2] +$
$[\{(98.76 \leq \text{ perimeter } < 114.8) + (\text{largest smoothness } \geq 0.136)$
$+ (105.95 \leq \text{ largest perimeter } < 117.45)\} \geq 2] \geq 1$

The generated rule has two clauses presented within the square brackets. Each clause consists of three literals (presented within the parentheses) and a comparator with a *threshold on literals* (shown outside the curly brace). For example, "smoothness $\geq$ 0.089" is a literal/predicate and is assigned 1 (or true) if a tumor cell satisfies this characteristic and 0 otherwise. Inside a clause, we connect the literals using "+" to sum the number of true literals and then compare the sum with the threshold, which is 2 in both clauses in the above example. A clause is assigned 1 (or true) if the sum of true literals satisfies the threshold. Similarly, each clause is connected using "+" to sum the number of true clauses in the rule. Finally, the rule is satisfied, i.e, a tumor is diagnosed as malignant if the number of true clauses is at least equal to the *threshold on clauses*, which is 1 in this example. Since each clause has three literals, we measure the size of the rule as $3 + 3 = 6$. In this example, the decision function is represented as a two-level checklist, where one would check items (i.e., literals/predicates) inside a checklist in the first level and then check among the checklists (i.e, clauses) in the second level.

The rest of the paper is organized as follows. We discuss related works in Section 2, introduce the notation and preliminaries in Section 3 and formulate the problem in Section 4. We describe the main contribution of this paper, CRR, in Section 5 and extend to an incremental mini-batch learning in Section 5.3. We then describe the experimental results in Section 6 and conclude in Section 7.

2 Related Works

In the growing field of interpretable machine learning, several rule-based systems such as decision lists [22] and classification rules [3, 5] have been proposed over the years. More recently, a Bayesian framework has been adopted to learn rule sets [31], rule lists [14], and falling rule lists [29]. Building on the connection between rule learning and Boolean compression sensing, a rule-based classification system was proposed to trade-off classification accuracy and representation size [17]. Su et al. [25] proposed an extension that allows two-level Boolean rules. In addition to designing classifiers that produce forms amenable to end-users, there is a large body of work to describe the working of opaque models (model-agnostic interpretability) [13, 21, 15]. While our work can also be applied to interpret black-box classifiers, we do not pursue this direction in this paper and leave for future work.

In recent work, Ghosh and Meel [10] propose an incremental approach for efficiently learning CNF classification rules using a MaxSAT-based framework. They adopt a sequential partition-based training methodology in incremental learning. Although the said approach achieves the runtime improvement, the generated rules highly depend on the order/position of samples in the training dataset. In contrast, the incremental framework proposed in this work does not suffer from strong dependencies on the order of samples in the training dataset.

3 Preliminaries

We use capital boldface letters such as $\mathbf{X}$ to denote matrices, while lower boldface letters $\mathbf{y}$ are reserved for vectors/sets. For a matrix $\mathbf{X}$, $\mathbf{X}_i$ represents the i-th row of $\mathbf{X}$ while for a vector/set $\mathbf{y}$, y_i represents the i-th element of $\mathbf{y}$.

Let F be a Boolean formula and $\mathbf{b} = \{b_1, b_2, \ldots, b_m\}$ be the set of Boolean propositional variables appearing in F. A literal v_i is a variable (b_i) or its complement ($\neg b_i$). A *satisfying assignment* or a *witness* σ of F is an assignment of variables in $\mathbf{b}$ that makes F evaluate to 1 and is denoted as $\sigma \models F$. If σ is an assignment of variables and $b_i \in \mathbf{b}$, we use $\sigma(b_i)$ to denote the value assigned to b_i in σ. F is in Conjunctive Normal Form (CNF) if $F := \bigwedge_{i=1}^{k} C_i$, where each clause $C_i := \bigvee_j v_j$ is represented as a disjunction of literals. Let $\mathbb{1}[true] = 1$ and $\mathbb{1}[false] = 0$. Motivated by checklists, we propose relaxed-CNF where in addition to F, we have two more parameters η_c and η_l. We say that (F, η_c, η_l) is in relaxed-CNF and σ is its witness if $\sigma \models (F, \eta_c, \eta_l)$ whenever $\sum_{i=1}^{k} \mathbb{1}[\sigma \models (C_i, \eta_l)] \geq \eta_c$ where $\sigma \models (C_i, \eta_l)$ iff $\sum_{v \in C_i} \mathbb{1}[\sigma \models v] \geq \eta_l$. Informally, σ satisfies a clause (C_i, η_l) if it at least η_l literals in C_i are set to true by σ and σ satisfies (F, η_c, η_l) if at least η_c of $\{(C_i, \eta_l)\}$ are true. In Example 1.1, $\eta_c = 1$ and $\eta_l = 2$.

Theorem 1. *[1] Let (C, η) be a relaxed-CNF clause where C has m literals and $\eta \in \{1, \ldots, m\}$ is the threshold on literals. An equivalent compact encoding of (C, η) into a CNF formula $F = \bigwedge_i C_i$ requires $\binom{m}{m-\eta+1}$ clauses where each clause is distinct and has $m - \eta + 1$ literals of (C, η). Therefore, the total number of literals in F is $(m - \eta + 1)\binom{m}{m-\eta+1} = m$ when $\eta = 1$, otherwise $(m - \eta + 1)\binom{m}{m-\eta+1} > m$.*

Between two vectors $\mathbf{u}$ and $\mathbf{v}$ over Boolean variables/constants (e.g., 0, 1), we use $\mathbf{u} \circ \mathbf{v}$ to denote the inner product, i.e., $\mathbf{u} \circ \mathbf{v} =$

$\sum_i (u_i \cdot v_i)$, where u_i and v_i are a variable/constant at the i-th index of $\mathbf{u}$ and $\mathbf{v}$ respectively. In this context, the operator "$\cdot$" between a variable and a constant follows the standard interpretation, i.e., $0 \cdot b = 0$ and $1 \cdot b = b$.

We consider a standard binary classification problem, where we are given a collection of training samples $\{(\mathbf{X}_i, y_i)\}$. Each vector $\mathbf{X}_i \in \mathcal{X}$ contains the valuation of the features $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ for sample i, and $y_i \in \{0, 1\}$ is the binary label for sample i. A classifier $\mathcal{R}$ is a mapping that takes in a feature vector $\mathbf{x}$ and returns a class $\hat{y} \in \{0, 1\}$, i.e. $\hat{y} = \mathcal{R}(\mathbf{x})$. The goal is not only to design $\mathcal{R}$ to approximate the training set, but also to generalize to unseen samples arising from the same distribution. We focus on classifiers that can be expressed as relaxed-CNF. We use $\mathsf{clause}(\mathcal{R}, i)$ to denote the i-th clause of $\mathcal{R}$ and $|\mathsf{clause}(\mathcal{R}, i)|$ to denote the size of $\mathsf{clause}(\mathcal{R}, i)$, which is measured as the number of literals in the i-th clause. Furthermore, we use $|\mathcal{R}|$ to denote the rule-size of classifier $\mathcal{R}$, which is defined as the number of literals in all the clauses, i.e., $|\mathcal{R}| = \Sigma_i |\mathsf{clause}(\mathcal{R}, i)|$. In Example 1.1, $|\mathcal{R}| = |\mathsf{clause}(\mathcal{R}, 1)| + |\mathsf{clause}(\mathcal{R}, 2)| = 3 + 3 = 6$.

In this work, we consider the learning problem as an optimization problem, wherein we reduce the construction of $\mathcal{R}$ to computing assignments of appropriately designed variables. An optimization problem consisting of Boolean variables can be solved using Integer Linear Programming (ILP) where each variable takes value either 0 or 1 [3]. Given an objective function and a set of constraints comprising of variables with range $\{0, 1\}$, ILP finds an optimal assignment of variables which minimizes the objective function.

4 Problem Formulation

We now present the formal definition of the problem. Given (i) an instance space $(\mathbf{X}, \mathbf{y})$, where $\mathbf{X} \in \{0, 1\}^{n \times m}$ is the feature matrix with m binary features and n samples[4], and $\mathbf{y} \in \{0, 1\}^n$ is the label vector, (ii) an positive integer k indicating the number of clauses in the desired rule, (iii) an integer threshold on literals $\eta_l \in \{0, \ldots, m\}$ indicating the minimum number of literals required to be assigned 1 to satisfy a clause, (iv) an integer threshold on clauses $\eta_c \in \{0, \ldots, k\}$ indicating the minimum number of clauses required to be satisfied to satisfy a formula, and (v) the data-fidelity parameter $\lambda \in [0, 1]$, we learn a classification rule $\mathcal{R}$, which is expressed as a k clause relaxed-CNF formula.

Our goal is to find rules that balance two goals: being accurate but also sparse to avoid over-fitting. To this end, we seek to minimize the total number of literals in all clauses, which motivates us to find $\mathcal{R}$ with minimum $|\mathcal{R}|$. In particular, suppose $\mathcal{R}$ classifies all samples correctly, i.e., $\forall i, y_i = \mathcal{R}(\mathbf{X}_i)$. Among all the rules that classify all samples correctly, we choose the sparsest such $\mathcal{R}$.

$$\min_{\mathcal{R}} |\mathcal{R}| \text{ such that } \forall i, y_i = \mathcal{R}(\mathbf{X}_i)$$

In practical classification tasks, perfect classification is very unusual. Hence, we need to balance rule-sparsity with prediction error. Let $\mathcal{E}_\mathcal{R}$ be the set of samples which are misclassified by $\mathcal{R}$, i.e., $\mathcal{E}_\mathcal{R} = \{\mathbf{X}_i | y_i \neq \mathcal{R}(\mathbf{X}_i)\}$. Hence we aim to find $\mathcal{R}$ as follows[5]:

$$\min_{\mathcal{R}} \frac{\lambda}{n} |\mathcal{E}_\mathcal{R}| + \frac{1 - \lambda}{k \cdot m} |\mathcal{R}| \text{ such that } \forall \mathbf{X}_i \in \mathcal{E}_\mathcal{R}, y_i \neq \mathcal{R}(\mathbf{X}_i).$$

Each term in the objective function is normalized in $[0, 1]$. The data-fidelity parameter $\lambda \in [0, 1]$ serves to balance the trade-off between prediction accuracy and sparsity. Higher values of λ produces lower prediction errors by sacrificing the sparsity of $\mathcal{R}$, and vice versa. It can be viewed as an inverse of the regularization parameter.

5 CRR: Classification Rules in Relaxed Logical Form

In this section, we describe the main contribution of our work, CRR, a framework for learning relaxed-CNF rules. CRR converts the learning problem into an ILP-based formulation, learns the optimal assignment of variables and constructs rule $\mathcal{R}$ based on the assignment. We organize the rest of this section as follows. We discuss the decision variables in Section 5.1, the constraints and the linear programming relaxation in Section 5.2, the incremental learning in Section 5.3, feature discretization in Section 5.4, and adaptation of CRR for learning other classification rules in Section 5.5.

5.1 Description of Variables

CRR considers two types of *decision* variables: (i) feature variables and (ii) noise (classification error) variables. Since feature x_j can be present or not present in each of k clauses, CRR considers k variables, each denoted by b_j^i corresponding to feature x_j to denote its participation in the i-th clause, i.e, $b_j^i = \mathbb{1}[j\text{- th feature is selected in } i\text{-th clause}]$. The q-th sample in the training dataset, however, can be misclassified by $\mathcal{R}$. Therefore, CRR introduces a noise variable $\xi_q \in \{0, 1\}$ corresponding to the q-th sample, so that the assignment of ξ_q can be interpreted whether $\mathbf{X}_q$ is misclassified by $\mathcal{R}$ or not, i.e., $\xi_q = \mathbb{1}[q\text{-th sample is misclassified}]$. Hence, the key idea of CRR for learning $\mathcal{R}$ is to define an ILP query over $k \cdot m + n$ decision variables, denoted by $\{b_1^1, b_2^1, \ldots, b_m^1, \ldots, b_m^k, \xi_1, \ldots, \xi_n\}$. In this context, we define $\mathbf{b}_i = \{b_j^i \mid j = 1, \ldots, m\}$ as a *vector of feature variables* corresponding to the i-th clause.

5.2 Construction of ILP Query

We first discuss the objective function of the ILP query Q for learning a k-clause relaxed-CNF rule $\mathcal{R}$. The objective function takes care of both the rule-sparsity and the prediction accuracy of $\mathcal{R}$. Since CRR prefers a sparser rule with as few literals as possible, we construct the objective function by preferring b_j^i to be 0. Moreover, to encourage $\mathcal{R}$ to predict the training samples accurately, we penalize the number of variables ξ_q that are different from 0. In this context, we utilize the parameter λ to trade off between sparsity and accuracy. Therefore, the objective function of the ILP query Q is to minimize the normalized sum of all noise variables ξ_q weighed by data-fidelity parameter λ and feature variables b_j^i weighed by $1 - \lambda$ (Eq. 1a).

We formulate the constraints of the ILP query Q as follows. Initially, we define the range of the decision variables and add constraints accordingly (Eq. 1b and 1c). For each sample, at first, we add constraints to mimic the behavior of hard-OR of literals in a clause, and then we add constraints to apply soft-AND of clauses in a formula.

[3] The problem can be viewed as either ILP or MaxSAT, but we obtained better performance from ILP solvers.

[4] $\mathbf{X}$ contains both the features and their complements as columns as in [17].

[5] In our formulation, it is straightforward to add class-conditional weights (e.g. to penalize false-alarms more than mis-detects), and to allow instance weights (per sample).

We first consider the case when the q-th sample has positive class label (Eq. 1d). $\mathbf{X}_q \circ \mathbf{b}_i \geq \eta_l$ resemblances the hard-OR operation of literals in a clause. We introduce k *auxiliary* 0-1 variables $\{\xi_{q,1}, \ldots, \xi_{q,k}\}$ to check whether at least η_c clauses are satisfied, i.e, $\xi_{q,i} = \mathbb{1}[i\text{-th clause is dissatisfied for } q\text{-th sample}]$, which let us impose the operation of soft-AND over clauses. We then add a constraint to make sure that at most $k - \eta_c$ clauses are allowed to be dissatisfied, otherwise the noise variable ξ_q is assigned to 1, i.e., the q-th sample is detected as noise.

A negative labeled sample has to dissatisfy more than $k - \eta_c$ clauses in $\mathcal{R}$ so that the sample is predicted as 0, which is equivalent to satisfying more than $k - \eta_c$ constraints $\mathbf{X}_q \circ \mathbf{b}_i < \eta_l$. As mentioned earlier, we introduce 0-1 variable $\xi_{q,i}$ to specify if the constraint $\mathbf{X}_q \circ \mathbf{b}_i < \eta_l$ is dissatisfied or not and restrict the count of dissatisfied clauses $\sum_{i=1}^{k} \xi_{q,i}$ to be less than η_c.

In the following, we present the ILP query Q.

$$\min \frac{\lambda}{n} \sum_{q=1}^{n} \xi_q + \frac{1-\lambda}{k \cdot m} \sum_{i=1}^{k} \sum_{j=1}^{m} b_j^i \tag{1a}$$

such that,

$$b_j^i \in \{0,1\}, i = 1, \ldots, k, j = 1, \ldots, m \tag{1b}$$

$$\xi_q \in \{0,1\}, q = 1, \ldots, n \tag{1c}$$

$$\text{if } \forall q \in \{1, \ldots, n\}, \text{ if } y_q = 1, \tag{1d}$$

$$\mathbf{X}_q \circ \mathbf{b}_i + m\xi_{q,i} \geq \eta_l, i = 1, \ldots, k \tag{1e}$$

$$k\xi_q + k - \eta_c \geq \sum_{i=1}^{k} \xi_{q,i}$$

$$\xi_{q,i} \in \{0,1\}, i = 1, \ldots, k$$

$$\text{if } \forall q \in \{1, \ldots, n\}, y_q = 0, \tag{1f}$$

$$\mathbf{X}_q \circ \mathbf{b}_i < \eta_l + m\xi_{q,i}, i = 1, \ldots, k \tag{1g}$$

$$k\xi_q + \eta_c > \sum_{i=1}^{k} \xi_{q,i}$$

$$\xi_{q,i} \in \{0,1\}, i = 1, \ldots, k$$

In the following, we show the complexity of the ILP query in terms of the number of variables and constraints.

Proposition 2. *Given a training dataset with n samples and m binary features, the ILP query Q for learning a binary classification rule in relaxed-CNF has $k \cdot m + n$ decision variables, $k \cdot n$ auxiliary variables, and $k \cdot m + n \cdot (k + 3)$ integer constraints.*

An ILP solver can take query Q as input and returns the optimal assignment $\sigma^*$of the variables. We extract relaxed-CNF rule $\mathcal{R}$ from the solution as follows.

Construction 1. *Let $\sigma^* = \mathsf{ILP}(Q)$, then $x_j \in \mathsf{clause}(\mathcal{R}, i)$ iff $\sigma^*(b_j^i) = 1$.*

Learning thresholds η_l and η_c: Given the training dataset $(\mathbf{X}, \mathbf{y})$ and data-fidelity parameter λ, one would learn the optimum value of the thresholds η_c and η_l of the desired rule $\mathcal{R}$ by specifying their range as constraints in the ILP query Q in Eq. 1. More precisely, we need to add two integer constraints $\eta_c \in \{0, \ldots, k\}$ and $\eta_l \in \{0, \ldots, m\}$ in the above query and then learn their values from the solution.

A more generalized version to CNF rules would be to learn different thresholds on literals for different clauses, i.e., $\eta_{l,i}$ for the i-th clause. This facilitates to capture the complex decision boundaries, while still producing rule-based decisions. In our ILP formulation, it is straight-forward to consider such generalization where we put constraints $\eta_{l,i} \in \{0, \ldots, m\}, i = 1, \ldots, k$ and replace η_l with $\eta_{l,i}$ in Eq. 1e and 1g.

Relaxing the ILP problem: Since the ILP query Q has binary integer constraints, the problem is computationally expensive. A common approach to solve such a problem efficiently is to relax the integer constraints, solve the LP-relaxed (linear programming relaxation) problem, and then round the non-integer variables to get an integer solution as in [17]. In our case, we cannot relax all integer constraints because it may violate the structure of relaxed-CNF rules. Specifically, η_c and η_l (or $\eta_{l,i}$) must be integers in the construction of relaxed-CNF rules, and $\xi_{q,i}$ needs to be an integer to precisely calculate the noise variable ξ_q. However, we can relax feature variable b_j^i and noise variable ξ_q and solve the corresponding MILP (mixed integer linear programming) problem. To construct the MILP problem, we replace Eq. 1b with $0 \leq b_j^i \leq 1$ and Eq. 1c with $0 \leq \xi_q \leq 1$, and the rest of the constraints in Q remain the same. If non-zero b_j^i is found in the solution, we set it to 1 and then construct the rule according to Construction 1.

5.3 Incremental Mini-batch Learning

In Section 5.2, we have presented an ILP formulation for learning relaxed-CNF rules and then discuss the relaxation to the integer constraints in the MILP formulation to make the approach computationally tractable. However, all integer constraints in the formulation can not be relaxed as discussed in Section 5.2, which necessitates an improved learning technique to achieve scalability. We now describe a mini-batch learning approach to CRR, which learns relaxed-CNF rule $\mathcal{R}$ incrementally from a set of mini-batches while solving a modified MILP query for each mini-batch.

In incremental mini-batch learning, the learning process repeats for a fixed number of iterations. In each iteration, we randomly select an equal number of samples from the full training set and generate a mini-batch with less number of samples. We then construct a MILP query on the current mini-batch with a modified objective function. This objective function simultaneously penalizes the prediction errors on the current mini-batch and the change in the rules learned in consecutive iterations. In the following, we discuss the modified objective function and the MILP formulation.

Let $(\mathbf{X}, \mathbf{y})$ be the training dataset with n samples and m binary features and τ be the number of iterations in the learning process. In the p-th iteration, we randomly sample a mini-batch $(\mathbf{X}_p, \mathbf{y}_p)$ with equal number n_p of samples and $n_p \leq n, p = 1, \ldots, \tau$. Note that all mini-batches have the same binary features $\mathbf{x} = \{x_1, \ldots, x_m\}$ as in the training set and thus share the same feature variables b_j^i in the MILP query. Therefore, we devise an objective function that prefers to keep the assignment of b_j^i learned in the $(p-1)$-th iteration while minimizing the prediction error on the current mini-batch $(\mathbf{X}_p, \mathbf{y}_p)$. To this end, each b_j^i is assigned 0 initially in the learning process. This technique enables us to update the learned rule in terms of the update in assignment of feature variables b_j^i over mini-batches.

Let Q_p be the MILP query for the p-th mini-batch for learning the k-clause relaxed-CNF rule $\mathcal{R}_p$. Thus $\mathcal{R}_0$ is an empty rule. We consider an indicator function $I(\cdot) : b_j^i \rightarrow \{1, -1\}$, which takes a feature variable b_j^i as input and outputs -1 if b_j^i is assigned 1 in the solution of Q_{p-1} (i.e., feature x_j is selected in the i-th clause of $\mathcal{R}_{p-1}$), otherwise outputs 1.

$$I(b_j^i) = \begin{cases} -1 & \text{if } x_j \in \mathsf{clause}(\mathcal{R}_{p-1}, i) \\ 1 & \text{otherwise} \end{cases}$$

We now discuss the modified objective function in the following.

$$\min \frac{\lambda}{n} \sum_{q=1}^{n_p} \xi_q + \frac{1-\lambda}{k \cdot m} \sum_{i=1}^{k} \sum_{j=1}^{m} b_j^i \cdot I(b_j^i) \tag{2}$$

In the objective function, the first term penalizes the prediction error of samples in the current mini-batch and the second term penalizes when b_j^i is assigned differently than its previous assignment. Note that the total prediction error is normalized by dividing by n, which is the size of the full training dataset. This normalization is useful as it assists in updating $\mathcal{R}_p$ while also considering the relative size of the mini-batch. Intuitively, if the size n_p of the mini-batch is close to the size n of the training set, more priority is given to the prediction accuracy and vice versa. The constraints in the query Q_p are similar to the constraints in Eq. 1. Finally, the rule $\mathcal{R}$ is equivalent to $\mathcal{R}_\tau$ which is learned for the last mini-batch.

Proposition 3. *At each iteration, the MILP query in the incremental mini-batch learning approach has $k \cdot m + n_p \cdot (k+3)$ constraints, where n_p is the size of the mini-batch.*

Learning thresholds η_l and η_c: In the incremental learning, the objective function in Eq 2 does not impose any constraint on the thresholds. In fact, the thresholds are learned in each iteration by solving the corresponding MILP query and we consider their final values in the last iteration.

5.4 Learning with Non-binary Features

Since our problem formulation requires input instances to have binary features, datasets with categorical and continuous features require a preprocessing stage. Initially, for all continuous features, we apply entropy-based discretization [8] to infer the most appropriate number of categories/intervals by recursively splitting the domain of each continuous feature to minimize the class-entropy of the given dataset[6]. For example, let $x_c \in [a, b]$ be a continuous feature, and entropy-based discretization splits the domain $[a, b]$ into three intervals with two split points $\{a', b'\}$, where $a < a' < b' < b$. Therefore, the result intervals are $x_c < a'$, $a' \leq x_c < b'$, and $x_c \geq b'$.

After applying entropy-based discretization on continuous features, the dataset contains only categorical features, which can be converted to binary features using one-hot encoding as in [10, 13]. In this encoding, a Boolean vector is introduced with cardinality equal to the number of distinct categories. Let a categorical feature have three categories 'red','green', and 'yellow'. In one-hot encoding, samples with category-value 'red','green', and 'yellow' would be converted into binary features while taking values 100, 010, and 001 respectively.

5.5 Learning Rules in Other Logical Forms

CRR learns classification rules in relaxed-CNF form and we can leverage this framework for learning classification rules in other logical forms, for example, CNF and DNF. To learn a CNF rule one can set $\eta_l = 1$ and $\eta_c = k$. Moreover, to learn a DNF rule, one would first complement the class label of all samples, then learn a CNF rule by setting the parameters as described and finally negate the learned rule.

6 Experiments

We implement a prototype of the incremental version of CRR[7] based on the Python API for CPLEX and conduct an extensive empirical analysis to understand the behavior of CRR on real-world instances. The objective of our experimental evaluation is to answer the following questions:

1. How do the accuracy and training time for CRR behave vis-a-vis state of the art classifiers on large datasets arising in ML problems in practice?
2. Can CRR generate sparse rules compared to that of other rule-based models?
3. How do the training time, accuracy and rule size vary with model hyper-parameters?

In summary, relaxed-CNF rules generated by CRR achieves higher accuracy and concise representation than CNF rules in most of the datasets. Moreover, relaxed-CNF rules are shown to be sparser than decision lists with competitive accuracy in large datasets. Finally, we show how to control the trade-off between rule-sparsity and accuracy using the hyper-parameter λ; and between accuracy and training time using the hyper-parameter k and size n_p of each mini-batch. In the following, we give a detailed description of experiments.

6.1 Experiment Methodology

We perform experiments on a high-performance computer cluster, where each node consists of E5-2690 v3 CPU with 24 cores, 96 GB of RAM. Each experiment is run on four cores of a node with 16 GB memory. We compare the performance of CRR with state-of-the-art classifiers, e.g. IMLI [10], RIPPER [3], BRS [31], random forest (RF), support vector classifier (SVC), nearest neighbors classifiers (k-NN), and l_1 penalized logistic regression (LR). Among them, IMLI, BRS, and RIPPER are rule-based classifiers. In particular, IMLI generates classification rules in CNF using a MaxSAT-based formulation and we use Open-WBO [18] as the MaxSAT solver for IMLI. We compare with propositional rule learning algorithm RIPPER, which is implemented in WEKA [11] and generates classification rules in the form of decision lists. BRS is a Bayesian framework for generating rule sets expressed as DNF. For other classifiers, we use the Scikit-learn module of Python [20]. For all classifiers, we set the training cut off time to 1500 seconds.

We consider a comparable number of hyper-parameter choices for each classifier. Specifically for CRR, we choose data-fidelity parameter $\lambda \in \{0.5, 0.67, 0.84, 0.99\}$, number of clauses $k \in \{1, 2, 3\}$, relative size of mini-batch $\frac{n_p}{n} \in \{0.25, 0.50, 0.75\}$, and number of iterations $\tau \in \{2, 4, 8, 16\}$. We learn the value of η_c and η_l from the dataset as described in Eq. 5.2. In CPLEX, we set the maximum solving time to 1000 seconds ($\frac{1000}{\tau}$ seconds for each iteration). We present the current best solution of CPLEX when the solver times out while finding the optimal solution.

We control the cutoff of the number of examples in the leaf node in the case of RF and RIPPER. For SVC, k-NN, and LR we discretize the regularization parameter on a logarithmic grid. For BRS, we vary

[6] A simple quantile-based discretization also works, but it requires an extra parameter (i.e., the number of quarantines).

[7] The source code along with benchmarks is available in the supplementary material

Dataset	Size	Features	RIPPER	BRS	IMLI	CRR
Heart	303	31	78.69	72.13	72.13	77.69
			7	19	13	4.5
			5.27s	25.07s	1.8s	122.5s
Ionosphere	351	144	88.65	91.43	89.29	91.43
			8	4	8.5	20
			5.87s	75.53s	2.09s	5.59s
WDBC	569	88	95.22	95.65	93.91	94.69
			7.5	12	7	34.5
			5.7s	630.23s	1.38s	316.32s
Magic	19020	79	84.04	74.15	71.97	81.31
			115	3	24	31
			15.86s	56.46s	141.2s	1012.6s
Tom's HW	28179	910	97.4		95.88	97.34
			36	—	30	4
			42.73s		92.65s	1071.58s
Credit	30000	110	81.68		81.42	82.04
			38.5	—	10	32
			14.52s		17.66s	1021.35s
Adult	32561	144	84.31		82.08	84.86
			94	—	23	18
			27.61s		11.91s	1016.36s
Twitter	49999	1511	95.74		94.24	95.16
			179.5	—	57	12
			170.87s		238.29s	1144.66s
Weather-AUS	107696	141	84.57		82.83	83.34
			195	—	7	2
			121.02s		366.12s	1115.27s
Skin	245057	119	98.32		98.92	95.08
			725	—	201	29
			1313.19s		103.8s	825.6s

Table 1: Comparisons of test accuracy, rule-size and training time among different rule-based classifiers. Every cell in the last eight columns contains the test accuracy in percentage (top value), rule size (middle value), and training time in seconds (bottom value). In the experiments, CRR shows higher test accuracy than IMLI and generates sparser rules than RIPPER.

max clause-length $\in \{3, 4, 5\}$, support $\in \{5, 10, 15\}$, and two other parameters $s \in \{100, 1000, 10000\}$ and $\rho \in \{0.9, 0.95, 0.99\}$. For IMLI, we consider $\lambda \in \{1, 5, 10\}$ and $k \in \{1, 2, 3\}$ and vary the number of batches τ such that each batch has at least 32 samples and at most 512 samples.

6.2 Results

6.2.1 *Performance Evaluation of* CRR *with Rule-based Classifiers:*

We conduct an assessment of performance using five fold nested cross-validation as in [5] and report the median of test accuracy, rule-size and training time of all rule-based classifiers in Table 1. Specifically, we show the dataset, the number of samples and the number of discretized features in the first three columns in Table 1. Inside each cell of column four to 11, we present the test accuracy (top value), rule-size (middle value) and training time (bottom value) of each classifier for each dataset.

We first compare relaxed-CNF rules generated by CRR with CNF rules generated by IMLI. In Table 1, relaxed-CNF rules exhibit higher prediction accuracy than CNF rules in the majority of the datasets, showing the effectiveness of using a more expressive representation of classification rules in capturing the decision boundary. In addition, the generated relaxed-CNF rules are comparatively smaller than CNF rules in terms of rule-size in most of the datasets. Therefore, relaxed-CNF rules improve upon CNF rules in terms of both prediction accuracy and rule size in the majority of the datasets. In this context, CRR provides a trade-off between accuracy and rule-size depending on the choice of hyper-parameters and the experimental results are discussed later in Section 6.2.3. We then compare relaxed-CNF rules with DNF rules generated by BRS and find that relaxed-CNF rules outperform DNF rules w.r.t. prediction accuracy in several datasets. At this point, BRS fails to scale for larger datasets as shown in Table 1. We finally compare relaxed-CNF rules with decision lists generated by RIPPER. In the experiments, relaxed-CNF rules achieve comparable prediction accuracy with decision lists in most of the datasets. In contrast, RIPPER generates very large decision lists compared to relaxed-CNF rules in the majority of the datasets, more precisely in large datasets. To summarize the performance of CRR among different rule-based classifiers, CRR can generate smaller relaxed-CNF rules with better accuracy in most of the cases with a couple of exceptions.

Moving focus on the training time, the non-incremental version of CRR times out on larger instances in the experiments, potentially producing sub-optimal rules with reduced accuracy, thereby highlighting the need for the incremental approach. On the other hand, the incremental version of CRR can handle most of the datasets within the allotted amount of time. In Table 1, CRR takes a comparatively longer time to generate relaxed-CNF rules in comparison with other rule-based classifiers, e.g., RIPPER, and IMLI because of the flexible combinatorial structure of relaxed-CNF rules. However, the testing time of CRR is insignificant (< 0.01 seconds) and thus can be deployed in practice.

Dataset	LR	SVC	RF	k-NN	CRR
Heart	84.29	83.33	81.97	78.69	77.69
Ionosphere	94.29	91.43	92.96	91.43	91.43
WDBC	98.26	96.46	96.9	95.61	94.69
Magic	85.15	84.45	85.3	77.9	81.31
Tom's HW	97.62	97.66	97.52	94.59	97.34
Credit	82.04	82.12	81.97	80.5	82.04
Adult	87.24	86.82	86.84	84.68	84.86
Twitter	96.28	96.34	96.37	—	95.16
Weather-AUS	85.71	—	85.63	—	83.34
Skin	97.21	—	99.81	—	95.08

Table 2: Comparisons of test accuracy among CRR and non-rule-based classifiers. In the experiments, CRR achieves competitive prediction accuracy in spite of being a rule-based classifier.

6.2.2 *Performance Evaluation of* CRR *with Non-rule-based Classifiers:*

We have compared the test accuracy of CRR with non-rule-based classifiers: LR, SVC, RF and k-NN and report the results in Table 2. In the experiments, we find that CRR in spite of being a rule-based classifier is able to achieve competitive prediction accuracy with non-rule-based classifiers. In this context, SVC and k-NN sometimes can not complete training within the allotted time specifically in the large datasets, while CRR can still generate relaxed-CNF rules with competitive accuracy. Therefore, CRR shows the promise of applying rule-based classifiers in practice with an added benefit of interpretability along with competitive accuracy.

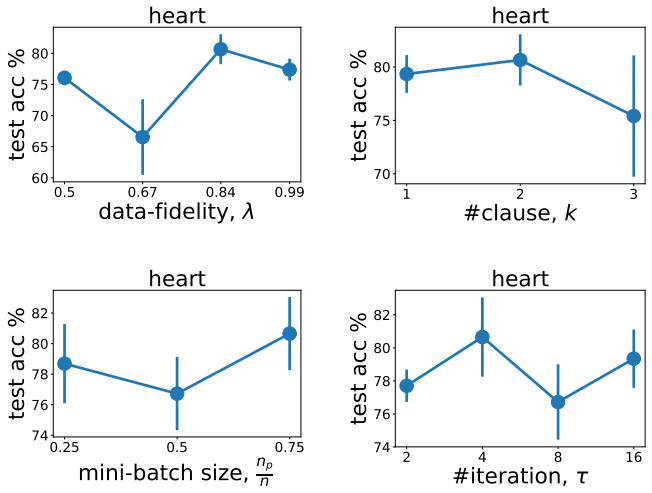

Figure 1: Effect of model hyper-parameters on test accuracy for Heart disease dataset.

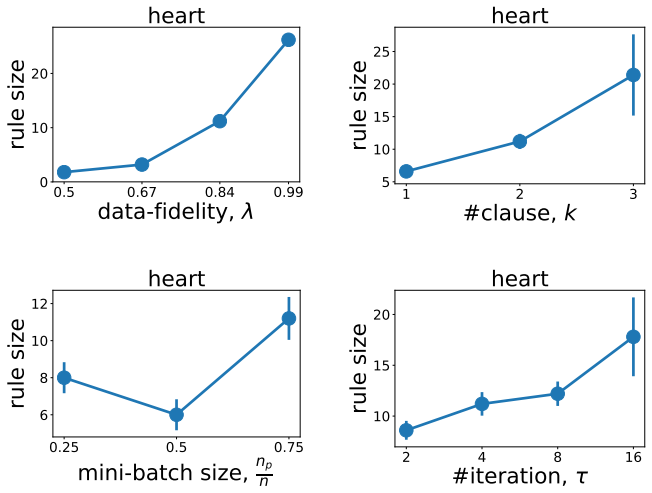

Figure 2: Effect of model hyper-parameters on rule-size for Heart disease dataset.

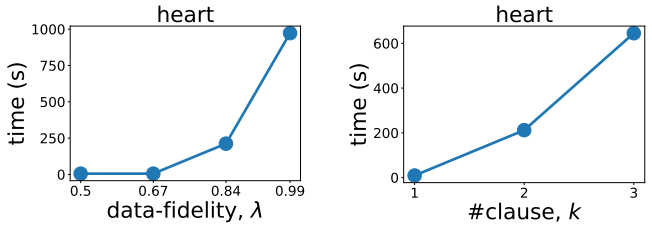

Figure 3: Effect of model hyper-parameters on training time for Heart disease dataset.

6.2.3 Varying Model Parameters:

In Figure 1, 2, and 3, we demonstrate the effect of varying the hyper-parameters of CRR[8]. To understand the effect of a single hyper-parameter, we fix the values of other hyper-parameters to a default choice where the default choice results in the most accurate rule.

Varying data-fidelity parameter (λ): As we increase data-fidelity parameter λ in the objective function in Eq. 1a and Eq. 2, more priority is given on the prediction accuracy than the sparsity of the rules. We similarly observe an increase in accuracy and also an increase in the size of the rules when λ increases in most of the datasets. This suggests that improved interpretability can often come at a minor cost in accuracy. In addition, we find an increase in training time for most of the datasets, which indicates that the MILP query usually takes a longer time to find the solution when more priority is given on the prediction accuracy.

Varying the number of clauses (k): As we increase k, CRR allows the generated rules to capture the variance in the given dataset more effectively, which results in higher accuracy in most of the datasets. The rule-size also increases as we learn more clauses. In addition, the training time increases, which can be reasoned by the fact that the number of constraints in the MILP formulation is linear with k. Thereby, the number of clauses k provides a control over the accuracy of the generated rule and training time.

Relative size of the mini-batch: We have varied the relative size of the mini-batch $\frac{n_p}{n}$ to observe its effect on the accuracy and the size of the rules. In most datasets, the accuracy increases when more samples are considered in the mini-batch costing higher training time. Moreover, the size of the generated rule increases as $\frac{n_p}{n}$ increases, which can be supported by the increase in the variance of the samples in the mini-batch.

Varying the number of iterations (τ): As we allow more iterations in the learning process, we find an increase in accuracy in most datasets. The training time also increases with τ because CRR is required to solve in total τ queries. We also observe an increase in rule-size in most datasets. The reason is that the objective function in the incremental mini-batch approach in Eq. 2 does not put a restriction on the size of the rules, rather on the change of rules in consecutive iterations. In addition, the learned values of the thresholds η_c and η_l in one iteration are not carried to the MILP query in the next iteration, which may cause an increase of rule-size.

[8] Additional experimental results can be found in `https://github.com/hoisoserious/irr`.

7 Conclusion

In this paper, we proposed an efficient combinatorial framework called CRR, for learning relaxed-CNF classification rules, which are more expressive than CNF/DNF rules. CRR uses a novel integration of mini-batch learning procedure with the MILP framework to learn sparse relaxed-CNF rules. Our experimental results demonstrate that CRR is able to learn relaxed-CNF rules with higher accuracy and concise representation than CNF rules. Moreover, the generated rules are sparser than decision lists in large datasets. The ability of CRR to handle large datasets has the motivation of our current investigations into conduction of human study in the medical domain: a lengthly multi-year procedure requiring approvals from multiple ethics boards.

REFERENCES

[1] Belaid Benhamou, Lakhdar Sais, and Pierre Siegel, 'Two proof procedures for a cardinality based language in propositional calculus', in *Proc. of STACS*, pp. 71–82. Springer, (1994).

[2] Katharine C Briggs, *Myers-Briggs type indicator*, Consulting Psychologists Press Palo Alto, CA, 1976.

[3] William W Cohen, 'Fast effective rule induction', in *Machine Learning Proceedings 1995*, 115–123, Elsevier, (1995).

[4] Mark Craven and Jude W Shavlik, 'Extracting tree-structured representations of trained networks', in *Proc. of NIPS*, pp. 24–30, (1996).

[5] Sanjeeb Dash, Oktay Gunluk, and Dennis Wei, 'Boolean decision rules via column generation', in *Proc. of NIPS*, 4655–4665, (2018).

[6] Julia Dressel and Hany Farid, 'The accuracy, fairness, and limits of predicting recidivism', *Science advances*, **4**(1), eaao5580, (2018).

[7] Amin Emad, Kush R Varshney, and Dmitry M Malioutov, 'A semi-quantitative group testing approach for learning interpretable clinical prediction rules', in *Proc. Signal Process. Adapt. Sparse Struct. Repr. Workshop, Cambridge, UK*, (2015).

[8] Usama Fayyad and Keki Irani, 'Multi-interval discretization of continuous-valued attributes for classification learning', (1993).

[9] Atul Gawande, *Checklist manifesto, the (HB)*, Penguin Books India, 2010.

[10] Bishwamittra Ghosh and Kuldeep S. Meel, 'IMLI: An incremental framework for MaxSAT-based learning of interpretable classification rules', in *Proc. of AIES*, (2019).

[11] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten, 'The WEKA data mining software: an update', *ACM SIGKDD explorations newsletter*, **11**(1), 10–18, (2009).

[12] Igor Kononenko, 'Machine learning for medical diagnosis: history, state of the art and perspective', *Proc. of AIME*, **23**(1), 89–109, (2001).

[13] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec, 'Faithful and customizable explanations of black box models', in *Proc. of AIES*, (2019).

[14] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al., 'Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model', *Proc. of AOAS*, **9**(3), 1350–1371, (2015).

[15] Scott M Lundberg and Su-In Lee, 'A unified approach to interpreting model predictions', in *Proc. of NIPS*, pp. 4765–4774, (2017).

[16] Dmitry Malioutov and Kuldeep S Meel, 'MLIC: A MaxSAT-based framework for learning interpretable classification rules', in *Proc. of CP*, pp. 312–327. Springer, (2018).

[17] Dmitry Malioutov and Kush Varshney, 'Exact rule learning via boolean compressed sensing', in *Proc. of ICML*, pp. 765–773, (2013).

[18] Ruben Martins, Vasco Manquinho, and Inês Lynce, 'Open-WBO: A modular MaxSAT solver', in *Proc. of SAT*, pp. 438–445. Springer, (2014).

[19] Martin Možina, Jure Žabkar, Trevor Bench-Capon, and Ivan Bratko, 'Argument based machine learning applied to law', *Artificial Intelligence and Law*, **13**(1), 53–73, (2005).

[20] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., 'Scikit-learn: Machine learning in python', *Proc. of JMLR*, **12**(Oct), 2825–2830, (2011).

[21] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, 'Why should i trust you?: Explaining the predictions of any classifier', in *Proc. of KDD*, pp. 1135–1144. ACM, (2016).

[22] Ronald L Rivest, 'Learning decision lists', *Machine learning*, **2**(3), 229–246, (1987).

[23] Carsten Sinz, 'Towards an optimal cnf encoding of boolean cardinality constraints', in *International conference on principles and practice of constraint programming*, pp. 827–831. Springer, (2005).

[24] Panigrahi Srikanth, Ch Anusha, and Dharmaiah Devarapalli, 'A computational intelligence technique for effective medical diagnosis using decision tree algorithm', *i-Manager's Journal on Computer Science*, **3**(1), 21, (2015).

[25] Guolong Su, Dennis Wei, Kush R Varshney, and Dmitry M Malioutov, 'Interpretable two-level boolean rule learning for classification', *arXiv preprint arXiv:1511.07361*, (2015).

[26] Harry Surden, 'Machine learning and law', *Wash. L. Rev.*, **89**, 87, (2014).

[27] Nikolaj Tollenaar and PGM Van der Heijden, 'Which method predicts recidivism best?: a comparison of statistical, machine learning and data mining predictive models', *Proc. of RSS*, **176**(2), 565–584, (2013).

[28] Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa, 'Making machine learning models interpretable.', in *ESANN*, volume 12, pp. 163–172. Citeseer, (2012).

[29] Fulton Wang and Cynthia Rudin, 'Falling rule lists', in *Proc. of AISTATS*, pp. 1013–1022, (2015).

[30] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille, 'Or's of and's for interpretable classification, with application to context-aware recommender systems', *arXiv preprint arXiv:1504.07614*, (2015).

[31] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille, 'A bayesian framework for learning rule sets for interpretable classification', *Proc. of JMLR*, **18**(1), 2357–2393, (2017).

[32] Jiaming Zeng, Berk Ustun, and Cynthia Rudin, 'Interpretable classification models for recidivism prediction', *Proc. of RSS*, **180**(3), 689–722, (2017).