Interpretable Classification Rules in Relaxed Logical Form

Bishwamittra Ghosh[1] , **Dmitry Malioutov**[*] and **Kuldeep S. Meel**[1]
[1]School of Computing, National University of Singapore

Abstract

Interpretability has become a central thread in ML research. As ML algorithms continue to permeate critical application domains such as medicine, legal, and transportation, it becomes increasingly important to allow human domain experts to understand and interact with ML solutions. ML algorithms that produce predictions in the form of rules are arguably some of the most interpretable ones, but their discrete combinatorial structure makes them computationally hard to learn. Here we generalize the widely popular CNF rules and introduce *relaxed*-CNF rules. These rules are much more flexible in terms of fitting data (have higher capacity) but about as interpretable to people as the traditional ones. 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. We first describe an exact ILP solution, which is computationally expensive. We then propose an incremental solution, which allows us to generate accurate interpretable relaxed-CNF rules with significantly improved runtime performance.

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 their output to people [Srikanth *et al.*, 2015; Kononenko, 2001; Surden, 2014; Tollenaar and Van der Heijden, 2013; Možina *et al.*, 2005]. To enable safe, robust and trustworthy integration of such systems, the end users require them to support interpretability, privacy, and fairness in decision-making [Dressel and Farid, 2018; Zeng *et al.*, 2017; Vellido *et al.*, 2012; Wang *et al.*, 2015]. In this context, rule-based representations are particularly effective for presenting decision functions to people [Lakkaraju *et al.*, 2019; Malioutov and Meel, 2018; Wang and Rudin, 2015; Wang *et al.*, 2017]. Although formalizing *interpretability* in the context of classification problems remains an open problem [Doshi-Velez and Kim, 2017], practitioners suggest smaller decision rules to be more interpretable, specifically in the medical domains [Letham *et al.*, 2015]. An influential study on the importance of checklists [Gawande, 2010] finds that highly complex and specialized problems can be handled smoothly by developing checklists and consistently using and improving them. An example of such a tool in medicine is the clinical prediction rule for estimating the risk of stroke, known as CHADS2 score [Gage *et al.*, 2001]. Another example is a psychometric test, known as Myers–Briggs Type Indicator (MBTI) [Briggs, 1976], which indicates differing psychological preferences in how people perceive the world around them and make decisions.

The earliest interpretable models include decision trees [Bessiere *et al.*, 2009; Quinlan, 2014], decision lists [Rivest, 1987], and classification rules [Cohen, 1995; Dash *et al.*, 2018]. More recently, a Bayesian framework has been adopted to learn rule sets [Wang *et al.*, 2017], rule lists [Letham *et al.*, 2015], and falling rule lists [Wang and Rudin, 2015]. Building on the connection between rule learning and Boolean compression sensing, a rule-based classification system was proposed to trade-off classification accuracy and interpretability [Malioutov and Varshney, 2013]. Su et al. 2015 proposed an extension that allows two-level Boolean rules. In addition to designing interpretable classifiers, there is a large body of work to find the interpretation of opaque models (model-agnostic interpretability) [Lakkaraju *et al.*, 2019; Ribeiro *et al.*, 2016; Lundberg and Lee, 2017]. While our work can also be applied to interpret black-box classifiers, we do not pursue this direction in this paper.

The simplest logical rules are single level-rules: ORs or ANDs of a subset of *literals* (please see Section 2 for a formal definition). An OR operator requires 1 out of N literals to be assigned to 1 (*true*), 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 (*Conjunctive Normal Form*) formula is a conjunction (AND) of clauses where each clause is a disjunction (OR) of literals, while a DNF (*Disjunctive Normal Form*) formula is a disjunction of clauses where each clause is a conjunction of literals. We refer to CNF and DNF formulas as two-level rules. A recent body of work has studied sparsity-inducing objectives

[*]Work performed while the author was at T.J. Watson IBM Research center.

for classification rules in CNF or DNF form and demonstrated that they often achieve high interpretability with minimal sacrifice in classification accuracy [Malioutov and Meel, 2018; Lakkaraju *et al.*, 2019; Ghosh and Meel, 2019]. In this paper, we consider a richer set of logical formulas that still allow high interpretability. We consider *hard*-OR clauses, where at least $M > 1$ out of N literals need to be active (*true*), and we similarly define *soft*-AND clauses which allow some of the literals (at most $N - M$) to be inactive (*false*)[1]. To extend the standard definition of CNF (ANDs of ORs), we define *relaxed*-CNF to denote soft-ANDs of hard-ORs. Similarly, *relaxed*-DNF is hard-ORs of soft-ANDs. In early work, Craven et al. 1996 considered M-of-N rules to explain black-box neural-network classifiers. Recently, Emad et al. 2015 has developed a semiquantitative 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.

Example 1.1. We present a toy example to illustrate a relaxed-CNF rule which we learn for Breast Cancer Wisconsin Diagnostic Data Set (WDBC) from UCI repository. The dataset is used to predict whether the cancer is benign or malignant based on the characteristics of a tumor cell. Our proposed framework can generate the following rule for detecting malignant cancer.

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

The generated rule has two clauses, each clause consisting of three literals, that are the characteristics of a tumor cell. In this rule, a clause is satisfied if two out of three characteristics match. Finally, the rule predicts malignant cancer if at least one out of two clauses is satisfied. Since each clause has three literals, we measure rule-size as $3 + 3 = 6$. We believe that this type of rule is intuitive and helpful to human experts in decision making.

Relaxed-CNF rules are more flexible than pure CNF rules and can accurately fit more complex classification boundaries. For example, relaxed-CNF clauses allow one to encode the majority function, which would require exponentially many clauses in CNF, showing the exponential gap in succinctness of the two representations. We argue that relaxed-CNF rules are about as interpretable as plain CNF rules, with higher expressibility. Specifically, relaxed-CNF and CNF rules have the same functional form where a user has to compute the sum of satisfied literals/clauses and then compare the sum to different thresholds. If he/she could evaluate a CNF rule with a given number of literals in clauses, then he/she should be able to evaluate a relaxed-CNF rule of the same form/size as well.

[1] An astute reader would notice that these definitions overlap. We use the term hard-OR when M is closer to 1, and use soft-AND when M is closer to N.

The primary contribution of this paper is a framework for learning relaxed-CNF rules which we call IRR (Interpretable Rules in Relaxed form). We propose an Integer Linear Programming (ILP)-based approach for learning optimal rules. To learn a k-clause relaxed CNF rule (say $\mathcal{R}$) with a direct (naive) ILP formulation, the size of ILP query expressed as the number of constraints is $\mathcal{O}(n \cdot k)$, where n is the number of samples in the training dataset. Such an approach would suffer from poor scalability for large datasets. To address this, we propose an incremental approach for learning $\mathcal{R}$ via a partition-based training methodology. Through a comprehensive experimental evaluation over datasets from the UCI and Kaggle repository, we observe that IRR with relaxed-CNF rules achieves an improved trade-off between accuracy and rule sparsity while still allowing run-time scalability owing to the incremental approach. We also find that smaller relaxed-CNF rules reach the same level of accuracy compared to plain CNF/DNF rules and decision lists.

2 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* of F is an assignment of variables in $\mathbf{b}$ that makes F evaluate to 1. 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 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 satisfied. In example 1.1, $\eta_c = 1$ and $\eta_l = 2$.

Between two vectors $\mathbf{u}$ and $\mathbf{v}$ over boolean variables or 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 denote a variable/constant at the i-th index of $\mathbf{u}$ and $\mathbf{v}$ respectively. In this context, note that the operation "$\cdot$" 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 our 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 $|\mathsf{clause}(\mathcal{R}, i)|$ denotes the size of $\mathsf{clause}(\mathcal{R}, i)$, which is measured as the number of literals in 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 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. [2] 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.

3 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, and $\mathbf{y} \in \{0,1\}^n$ is the label vector, (ii) a positive integer k indicating the number of clauses, (iii) a positive integer threshold η_l indicating the minimum number of literals required to be assigned 1 to satisfy a clause ($\eta_l \in \{1, \ldots, m\}$), (iv) a positive integer threshold η_c indicating the minimum number of clauses required to be satisfied to satisfy a formula ($\eta_c \in \{1, \ldots, k\}$), and (v) data fidelity parameter λ, 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: of being accurate but also interpretable. Various notions of interpretability have been proposed in the context of classification problems. A common proxy for interpretability in the context of decision rules is rule's sparsity. Namely, a rule involving fewer literals is more interpretable. In our problem setting, we 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 (most interpretable) 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 unlikely. Hence, we need to balance interpretability 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[3].

$$\min_{\mathcal{R}} |\mathcal{R}| + \lambda |\mathcal{E}_\mathcal{R}| \text{ such that } \forall \mathbf{X}_i \in \mathcal{E}_\mathcal{R}, y_i \neq \mathcal{R}(\mathbf{X}_i)$$

Here λ is the data-fidelity parameter, which serves to balance the trade-off between prediction accuracy and interpretability. 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.

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

[3] 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).

4 IRR: Interpretable Rules in Relaxed Form

In this section, we describe the main contribution of our work, IRR, an interpretable machine learning framework for learning relaxed-CNF rules. IRR 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 ILP variables in Section 4.1, the constraints in Section 4.2 and feature discretization in Section 4.3.

4.1 Description of Variables

IRR 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, IRR considers k variables, each denoted by b^i_j corresponding to feature x_j to denote its participation in the i-th clause. When assignment $\sigma(b^i_j) = 1$, x_j participates in the i-th clause of $\mathcal{R}$. The q-th sample, however, can be misclassified by $\mathcal{R}$. Therefore, IRR 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 $\mathcal{R}$ (when $\sigma(\xi_q) = 1$) or not. Hence the key idea of IRR for learning $\mathcal{R}$ is to define an ILP query over $k \times m + n$ decision variables, denoted by $\{b^1_1, b^1_2, \ldots, b^1_m, \ldots, b^k_m, \xi_1, \ldots, \xi_n\}$. In this context we define $\mathbf{B}_i = \{b^i_j \mid j \in \{1, \ldots, m\}$ as a *vector of feature variables* corresponding to the i-th clause.

4.2 Construction of ILP Query

At first, we 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 interpretability and the prediction accuracy of $\mathcal{R}$. Since IRR prefers a sparser rule with as few literals as possible, we construct the objective function by preferring b^i_j to be 0. Moreover, $\mathcal{R}$ should predict the training samples accurately, which penalizes the number of variables η_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 sum of all feature variables b^i_j and noise variables ξ_q weighed by data-fidelity parameter λ (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.

Let us consider that 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 *additional* variables $\xi_{q,1}, \ldots, \xi_{q,k}$ to check whether at least η_c clauses are satisfied, which let us impose the operation of soft-AND over clauses. Note that $\xi_{q,i}$ is assigned to 1 when the i-th clause is not satisfied. Therefore we add a constraint to make sure that at most $k - \eta_c$ clauses are allowed to be not satisfied,

otherwise the noise variable ξ_q is assigned to 1, i.e. the q-th sample is marked 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. Hence, if the q-th sample has negative label (Eq. 1e), more than $k-\eta_c$ constraints $\mathbf{X}_q \circ \mathbf{B}_i < \eta_l$ have to be satisfied. Therefore, for negative labeled samples, we restrict the count of dissatisfied clauses $\sum_{i=1}^{k} \xi_{q,i}$ to be less than η_c.

We present the ILP query Q as follows.

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

such that,

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

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

$$\text{if } \forall q \in \{1,\ldots,n\}, \text{ if } y_q = 1, \tag{1d}$$
$$\mathbf{X}_q \circ \mathbf{B}_i + m\xi_{q,i} \geq \eta_l, i = 1,\ldots,k$$
$$k\xi_q + k - \eta_c \geq \sum_{i=1}^{k} \xi_{q,i}$$
$$\xi_{q,i} \in \{0,1\}, i = 1,\ldots,k$$

$$\text{if } \forall q \in \{1,\ldots,n\}, y_q = 0, \tag{1e}$$
$$\mathbf{X}_q \circ \mathbf{B}_i < \eta_l + m\xi_{q,i}, i = 1,\ldots,k$$
$$k\xi_q + \eta_c > \sum_{i=1}^{k} \xi_{q,i}$$
$$\xi_{q,i} \in \{0,1\}, i = 1,\ldots,k$$

We can also learn $\eta_c \in \{1,\ldots,k\}$ and $\eta_l \in \{1,\ldots,m\}$ jointly with the original problem by putting their ranges as constraints in Q. These additional constraints make the ILP query more expensive. Since the number of clauses in $\mathcal{R}$ is often less than the total feature count (i.e., $k < m$) in practice, we additionally learn η_c in our experiments in Section 6.

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 $\sigma^*(b_j^i) = 1$.

Remark: IRR learns classification rules in relaxed-CNF form. To learn a CNF rule one can set $\eta_l = 1$ and $\eta_c = k$.

The following theorem states the succinctness of a clause in a relaxed-CNF rule [Benhamou *et al.*, 1994].

Theorem 2. Let (C, η) be a relaxed-CNF clause where C has m literals and η 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 $(m - \eta + 1)\binom{m}{m-\eta+1}$.

Due to lack of space, the proof is removed.

4.3 Learning on 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 [Fayyad and Irani, 1993] 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[4]. For example consider that $x_c \in [a, b]$ is 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 [Ghosh and Meel, 2019; Lakkaraju *et al.*, 2019]. In this encoding, a boolean vector is introduced with cardinality equal to the number of distinct categories. For example 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 A Fast Learning Approach

In this section, we describe an incremental approach to IRR, called inc-IRR, for learning a k-clause relaxed-CNF rule $\mathcal{R}$. We have described an ILP-based learning technique in Section 4 which generates an optimal rule. The number of constraints in the ILP query is linear with the number of training samples, which results in poor scalability for larger datasets. Hence, we propose an incremental learning technique via a partition-based training methodology [Ghosh and Meel, 2019]. In this context, we define $\mathcal{R}_p$ to be the rule learned for the p-th partition. The key idea of incremental learning is to divide the training dataset into a fixed number of partitions τ, learn rule for each partition in a sequential order such that $\mathcal{R}_p$ depends not only on the training samples in the p-th partition but also regularizes itself with the rule $\mathcal{R}_{p-1}$ learned for the $(p-1)$-th partition. To this end, we use notation $(\mathbf{X}^p, \mathbf{y}^p)$ to refer to the training data for the p-th partition. We assume that $\forall p \in \{1,\ldots,\tau\}, |\mathbf{X}^p| = n_\tau$. We propose the following modifications in the ILP-based formulation for incrementally learning relaxed-CNF rules.

5.1 ILP Query of Inc-IRR

In the incremental approach, we emphasize the assignment of the feature variables in the previous partition while constructing the ILP query for the current partition. This technique enables us to update the learned rule over partitions. Let us assume that we will construct query Q_p for the p-th partition of training data. We consider an indicator function $I(\cdot) : b_j^i \to \{1, -1\}$, which takes a feature variable b_j^i as input and outputs -1 if b_j^i is assigned 1 while solving Q_{p-1} for

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

the $(p-1)$-th partition (i.e., feature x_j is in the i-th clause of $\mathcal{R}_{p-1}$), otherwise outputs 1.

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

If a feature x_j is in $\mathcal{R}_{p-1}$, we try to keep x_j in $\mathcal{R}_p$ by modifying the objective function for Q_p as follows:

$$\min \sum_{i=1}^{k} \sum_{j=1}^{m} b_j^i \cdot I(b_j^i) + \lambda \sum_{q=1}^{n_\tau} \xi_q$$

Here we consider that the final rule $\mathcal{R}$ is equivalent to $\mathcal{R}_\tau$ which is learned for the last partition.

6 Experiments

We implemented a prototype of IRR (and inc-IRR)[5] based on the Python API for CPLEX[6] (version 12.8) and conducted an extensive rigorous empirical analysis to understand the behavior of IRR on real-world instances. The objective of our experimental evaluation was to answer the following questions:

1. Can inc-IRR handle and scale large datasets arising in ML problems in practice?
2. How does the test accuracy of inc-IRR compare vis-a-vis the state of the art classifiers?
3. Are the rules generated by inc-IRR interpretable to the end users?
4. How do the training time, accuracy and rule size vary with η_l, λ, k, and τ?

In summary, inc-IRR achieves comparable prediction accuracy vis-a-vis other state of the art classifiers while scaling to large instances. The impressive scalability of inc-IRR with negligible loss of accuracy illustrates the strength of the incremental approach. Furthermore, we observe that by using an explicit objective function in an optimization framework, we are able to generate rules with fewer terms in comparison to other interpretable classifiers. Finally, we perform an exhaustive analysis of the performance behavior of IRR and inc-IRR for various parameters such as η_l, λ, k, and τ.

6.1 Experiment Methodology

We performed 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 was run on four cores of a node with 16 GB memory. We compare the performance of both IRR and inc-IRR with other state of the art interpretable and non interpretable classifiers, e.g. IMLI [Ghosh and Meel, 2019], RIPPER [Cohen, 1995], BRS [Wang *et al.*, 2017], random forest (RF), support vector classifier (SVC), nearest neighbors classifiers (NN), and l_1 penalized logistic regression (LR). IMLI generates classification rules in CNF and we use Open-WBO [Martins *et al.*, 2014] as the MaxSAT solver for IMLI. We also used a classical (but still very competitive) propositional rule learning algorithm RIPPER, which is implemented in WEKA [Hall *et al.*, 2009]. BRS is a Bayesian framework for generating rule sets expressed as DNF. For other classifiers, we use Scikit-learn module of Python [Pedregosa *et al.*, 2011].

We consider a comparable number (10) of hyper-parameter choices for each classifier. Specifically, we control the cut-off of the number of examples in the leaf node in the case of RF and RIPPER. For SVC, NN, and LR we discretize the regularization parameter on a logarithmic grid. For BRS, we vary max clause-length ($\{3, 4, 5\}$), support ($\{5, 10, 15\}$), and two other parameters $s \in \{100, 1000, 10000\}$ and $\rho \in \{0.9, 0.95, 0.99\}$. For IMLI, IRR, and inc-IRR, we have considered three choices of $\lambda \in \{1, 5, 10\}$, three choices of $k \in \{1, 2, 3\}$. In addition, we consider three choices of $\eta_l \in \{1, 2, 3\}$ for both IRR and inc-IRR. We learn the value of η_c from the data for both IRR and inc-IRR. For inc-IRR and IMLI, we vary the number of partitions τ such that each partition has at least 32 samples and at most 512 samples. In CPLEX, we set MIP gap tolerance to 0.01 and the maximum solving time to 2000 seconds. We present the current best solution of CPLEX when the solver times out while finding the optimal solution. For all other classifiers, we set the training cut off time to 2000 seconds.

6.2 Results

Performance Evaluation Among Different Classifiers:

We perform an assessment of accuracy on ten fold cross validation set (90% training set, 10% test set). For each parameter choice, we compute the median accuracy over the cross-validation folds, and report the best parameter choice for each classifier. In Table 1, we present the best test accuracy and corresponding training time of different classifiers for different datasets[7]. The second and third column of the table shows the number of samples and features (after discretization) of a dataset respectively. From column four to 12, we present the test accuracy and training time (within parentheses) of each classifier for each dataset.

First, we observe that IRR times out on larger instances, potentially producing sub-optimal rules with reduced accuracy, thereby highlighting the need for the incremental approach. On the other hand, inc-IRR can handle most of the instances within the allotted amount of time. Observe that the complexity of the ILP formulation in Eq. 1 depends both on the dataset (for inc-IRR, it is the samples in a partition) and the parameters of the desired rule. Therefore, it is interesting to observe that inc-IRR may take higher training time for small datasets compared to large datasets. It is worth noting that the testing time is insignificant (less than 0.01s) for both IRR and inc-IRR.

At this point, one wonders whether we are able to generate succinct rules in comparison to other interpretable classifiers. To this end, we show the size of the most accurate rules by various interpretable classifiers in Table 2. The first column lists the benchmarks while column two to five lists the size of

[5] https://github.com/meelgroup/IRR

[6] www.cplex.com

[7] Generated relaxed-CNF rules and additional figures can be found at https://tinyurl.com/relaxed-CNF

Dataset	Size	Features	LR	NN	SVC	RF	RIPPER	BRS	IMLI	IRR	inc-IRR
Parkinsons	195	51	94.44 (2.0s)	94.87 (1.94s)	95.0 (1.88s)	92.5 (4.93s)	90.0 (5.28s)	95.0 (1.71s)	94.72 (3.09s)	94.87 (104.2s)	94.72 (4.66s)
Heart	303	31	86.21 (1.74s)	83.6 (1.5s)	85.48 (1.52s)	83.87 (4.53s)	81.59 (4.72s)	80.65 (2.76s)	80.65 (2.95s)	86.65 (2000s)	86.44 (2000s)
Ionosphere	351	144	95.63 (3.21s)	92.98 (1.64s)	92.73 (1.6s)	94.28 (4.62s)	92.81 (5.22s)	94.12 (224.42s)	91.3 (14.94s)	95.59 (1773.17s)	91.67 (2.08s)
WDBC	569	88	96.52 (3.2s)	96.49 (1.64s)	98.23 (1.56s)	96.49 (4.89s)	96.49 (5.05s)	97.35 (65.91s)	96.46 (2.58s)	97.34 (2000s)	96.49 (10.96s)
ILPD	583	14	71.19 (1.74s)	71.56 (1.55s)	71.19 (1.54s)	71.19 (4.46s)	72.41 (4.79s)	66.67 (2.75s)	71.31 (7.54s)	69.57 (5.44s)	74.14 (2.69s)
Pima	768	30	79.74 (2.54s)	79.22 (1.64s)	77.13 (1.63s)	78.57 (4.71s)	77.27 (4.92s)	77.92 (2.83s)	74.51 (20.95s)	78.57 (2000s)	77.27 (2000s)
Tic Tac Toe	958	27	98.44 (2.71s)	87.5 (1.6s)	98.44 (1.64s)	99.47 (5.28s)	98.44 (5.1s)	100.0 (1.62s)	82.72 (459.93s)	84.37 (2000s)	84.46 (2000s)
Titanic	1309	26	79.31 (3.1s)	77.1 (1.68s)	78.54 (1.69s)	79.01 (5.16s)	78.63 (5.06s)	77.78 (1.64s)	79.01 (3.86s)	81.22 (1669.87s)	78.63 (197.94s)
Compas	7210	19	67.75 (3.92s)	66.71 (8.09s)	66.02 (9.31s)	67.27 (20.86s)	67.34 (6.18s)	35.64 (2.34s)	66.37 (5.07s)	67.89 (162.13s)	65.81 (3.2s)
Tom's HW	28179	910	97.55 (6.0s)	—	97.6 (369.76s)	97.46 (82.44s)	97.6 (38.99s)	—	96.01 (85.04s)	97.34 (2000s)	96.52 (1956.6s)
Credit	30000	110	82.33 (7.76s)	80.69 (315.02s)	82.17 (386.28s)	82.12 (45.87s)	82.13 (16.61s)	—	81.75 (26.58s)	82.15 (2000s)	81.94 (27.7s)
Adult	32561	144	87.19 (7.07s)	84.72 (505.48s)	87.19 (511.57s)	86.98 (47.18s)	84.89 (52.38s)	—	83.63 (1999.41s)	85.23 (2000s)	83.14 (1959.2s)
Twitter	49999	1511	96.37 (10.28s)	—	—	96.48 (378.6s)	96.14 (162.02s)	—	94.57 (241.63s)	95.44 (2000s)	93.22 (1216.86s)

Table 1: Comparisons of test accuracy and training time for ten fold cross-validation for different classifiers. Every cell in the last nine columns contain the best test accuracy (%) and corresponding train time in seconds (within parentheses).

Dataset	RIPPER	BRS	IMLI	inc-IRR
Parkinsons	10.5	13.0	7.5	5.0
Heart	7.0	35.5	14.0	19.5
Ionosphere	11.0	4.0	8.5	6.0
WDBC	7.0	18.0	11.0	10.0
ILPD	5.0	3.0	5.0	2.0
Pima	8.0	8.0	15.0	21.5
Tic Tac Toe	25.0	24.0	11.5	12.0
Titanic	5.0	2.0	7.0	12.5
Compas	12.5	8.0	4.0	3.0
Tom's HW	16.5	—	32.0	5.5
Credit	33.0	—	9.0	3.0
Adult	106.0	—	35.5	13.0
Twitter	56.0	—	67.5	7.0

Table 2: Size of the rules generated by interpretable classifiers.

generated rules by RIPPER, BRS, IMLI, and inc-IRR. Note that inc-IRR is able to generate significantly smaller rules for larger datasets where RIPPER and IMLI tend to require tens of terms. For larger benchmarks, BRS fails to generate rule due to poor scalability. At this point, it is worth recalling that rules generated by inc-IRR are highly interpretable (Example 1.1) and can be explained to users in the form of checklists – a format widely adopted in several safety-critical domains.

Varying Model Parameters:

In Figure 1, we demonstrate the effect of varying the parameters of IRR and inc-IRR while generating relaxed-CNF rules. To understand the effect of a single parameter, we fix the values of other parameters to a default choice, wherein the default choice results in the most accurate rule.

Varying Threshold (η_l): As the threshold value on literals η_l increases, it constrains the underlying clause necessitating for more literals. The behavior is observed for all datasets, and here we show results for Titanic dataset. The added flexibility due to the addition of more literals allows the rule to generalize well, increasing the average cross-validation test accuracy. Similar to test accuracy, train accuracy also increases. Moreover, the ILP query becomes more expensive for a higher value of η_l and requires more training time.

Varying Data Fidelity Parameter (λ): As we increase λ (inverse to regularization parameter), we find that the average test and train accuracy increases while learning relaxed-CNF rules. The average size of the rules also increases with the increase of λ because of putting less weight on the sparsity of rules. This suggests that improved interpretability can often come at a minor cost in accuracy.

Varying The Number of Partitions (τ): As we make more partitions of the training data in our incremental approach, the number of training samples in each partition decreases, which incurs an over-fitting in train accuracy and

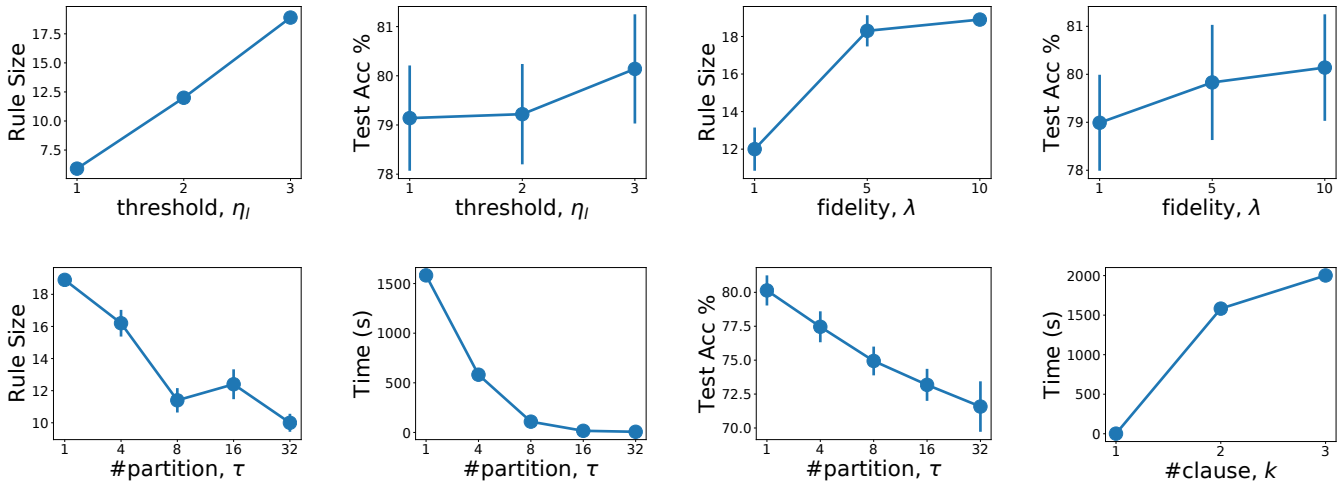

Figure 1: Effect of varying threshold η_l, data fidelity parameter λ, number of partitions τ, and number of clauses k while learning relaxed-CNF rules for Titanic dataset.

also a decrease in test accuracy. Moreover, the rules become shorter in our observation. The remarkable contribution of partitioning is a significant reduction in training time because of solving smaller size queries. Therefore, the choice of τ provides a trade-off between accuracy and training time.

Varying The Number of Clauses (k): As we increase k, IRR allows the generated rules to capture the variance in the given dataset more effectively, which results in higher average test accuracy. The rule-size also increases as we increase k. We also find an increase in train accuracy. Moreover, the training time also increases, since the number of constraints in the ILP formulation is linear with k.

7 Conclusion

In this paper, we propose relaxed-CNF rules, which allow increased flexibility to fit the data, while retaining high interpretability of CNF rules. Recent work has focused on closely related single-level M-of-N rules and in this work, we focus on multi-level rules, which can be viewed as extensions of checklists used in safety-critical domains. We demonstrate on experiments that relaxed-CNF rules offer succinctness in comparison to other state of the art interpretable classification techniques while still scaling to large instances.

Acknowledgements

This research is supported by the National Research Foundation Singapore under its AI Singapore Programme [R-252-000-A16-490] and the NUS ODPRT Grant [R-252-000- 685-133]. The computational work for this article was performed on resources of the National Supercomputing Centre, Singapore, https://www.nscc.sg/.

References

[Benhamou *et al.*, 1994] Belaid Benhamou, Lakhdar Sais, and Pierre Siegel. Two proof procedures for a cardinality based language in propositional calculus. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 71–82. Springer, 1994.

[Bessiere *et al.*, 2009] Christian Bessiere, Emmanuel Hebrard, and Barry O'Sullivan. Minimising decision tree size as combinatorial optimisation. In *International Conference on Principles and Practice of Constraint Programming*, pages 173–187. Springer, 2009.

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

[Cohen, 1995] William W Cohen. Fast effective rule induction. In *Machine Learning Proceedings 1995*, pages 115–123. Elsevier, 1995.

[Craven and Shavlik, 1996] Mark Craven and Jude W Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30, 1996.

[Dash *et al.*, 2018] Sanjeeb Dash, Oktay Gunluk, and Dennis Wei. Boolean decision rules via column generation. In *Advances in Neural Information Processing Systems*, pages 4655–4665. 2018.

[Doshi-Velez and Kim, 2017] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

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

[Emad *et al.*, 2015] Amin Emad, Kush R Varshney, and Dmitry M Malioutov. A semiquantitative group testing approach for learning interpretable clinical prediction rules. In *Proc. Signal Process. Adapt. Sparse Struct. Repr. Workshop, Cambridge, UK*, 2015.

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

[Gage *et al.*, 2001] Brian F Gage, Amy D Waterman, William Shannon, Michael Boechler, Michael W Rich, and Martha J Radford. Validation of clinical classification schemes for predicting stroke: results from the national registry of atrial fibrillation. *Jama*, 285(22):2864–2870, 2001.

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

[Ghosh and Meel, 2019] Bishwamittra Ghosh and Kuldeep S. Meel. Imli: An incremental framework for maxsat-based learning of interpretable classification rules. In *Proc. of AIES*, 2019.

[Hall *et al.*, 2009] 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.

[Kononenko, 2001] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

[Lakkaraju *et al.*, 2019] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Faithful and customizable explanations of black box models. In *Proc. of AIES*, 2019.

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

[Lundberg and Lee, 2017] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[Malioutov and Meel, 2018] Dmitry Malioutov and Kuldeep S Meel. Mlic: A maxsat-based framework for learning interpretable classification rules. In *International Conference on Principles and Practice of Constraint Programming*, pages 312–327. Springer, 2018.

[Malioutov and Varshney, 2013] Dmitry Malioutov and Kush Varshney. Exact rule learning via boolean compressed sensing. In *International Conference on Machine Learning*, pages 765–773, 2013.

[Martins *et al.*, 2014] Ruben Martins, Vasco Manquinho, and Inês Lynce. Open-wbo: A modular maxsat solver. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 438–445. Springer, 2014.

[Možina *et al.*, 2005] 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.

[Pedregosa *et al.*, 2011] 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. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[Quinlan, 2014] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

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

[Srikanth *et al.*, 2015] 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.

[Su *et al.*, 2015] 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.

[Surden, 2014] Harry Surden. Machine learning and law. *Wash. L. Rev.*, 89:87, 2014.

[Tollenaar and Van der Heijden, 2013] Nikolaj Tollenaar and PGM Van der Heijden. Which method predicts recidivism best?: a comparison of statistical, machine learning and data mining predictive models. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 176(2):565–584, 2013.

[Vellido *et al.*, 2012] Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa. Making machine learning models interpretable. In *ESANN*, volume 12, pages 163–172. Citeseer, 2012.

[Wang and Rudin, 2015] Fulton Wang and Cynthia Rudin. Falling rule lists. In *Artificial Intelligence and Statistics*, pages 1013–1022, 2015.

[Wang *et al.*, 2015] 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.

[Wang *et al.*, 2017] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

[Zeng *et al.*, 2017] Jiaming Zeng, Berk Ustun, and Cynthia Rudin. Interpretable classification models for recidivism prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 180(3):689–722, 2017.