IMLI: An Incremental Framework for MaxSAT-Based Learning of Interpretable Classification Rules

Bishwamittra Ghosh and **Kuldeep S. Meel**
School of Computing
National University of Singapore

Abstract

The wide adoption of machine learning in the critical domains such as medical diagnosis, law, education had propelled the need for interpretable techniques due to the need for end user to understand the reasoning behind decisions due to learning systems. The computational intractability of interpretable learning led practitioners to design heuristic techniques, which fail to provide sound handles to tradeoff accuracy and interpretability.

Motivated by the success of MaxSAT solvers over the past decade, recently MaxSAT-based approach, called MLIC, was proposed that seeks to reduce the problem of learning interpretable rules expressed in Conjunctive Normal Form (CNF) to a MaxSAT query. While MLIC was shown to achieve accuracy similar to that of other state of the art *black-box* classifiers while generating small interpretable CNF formulas, the runtime performance of MLIC is significantly lagging and renders approach unusable in practice. In this context, authors raised the question: *Is it possible to achieve the best of both worlds, i.e., a sound framework for interpretable learning that can take advantage of MaxSAT solvers while scaling to real-world instances?*

In this paper, we take a step towards answering the above question in affirmation. We propose an incremental approach to MaxSAT based framework that achieves scalable runtime performance via partition-based training methodology. Extensive experiments on benchmarks arising from UCI repository demonstrate that IMLI achieves up to three orders of magnitude runtime improvement without loss of accuracy and interpretability.

1 Introduction

The recent advances in the machine learning techniques have led autonomous decision making systems be adopted in wide range of domains to perform data-driven decision making. As such the domains range from movie recommendations, ad predictions to legal, medical, and judicial. The diversity of domains mandate different criteria for the machine learning techniques. For domains such as movie recommendations and ad predictions, accuracy is usually the primary objective but for safety critical domains (Otte 2013) such as medical and legal, interpretability, privacy, and fairness (Barocas, Hardt, and Narayanan 2017) are of paramount importance.

It has been long observed that the interpretable techniques are typically trusted and adopted by decision makers as interpretability provides them understanding of reasoning behind a tool's decision making (Ribeiro, Singh, and Guestrin 2016). At this point, it is important to acknowledge that formalizing interpretability is a major challenge (Doshi-Velez and Kim 2017) and we do not claim to have final word on this. In this context, it is worth noting that for several domains such as medical domain, which was the motivation for our investigation, decision rules with small number of rules tend to be most interpretable (Letham et al. 2015).

Since the problem of rule learning is known to be in NP-hard, the earliest efforts focused on heuristic approaches that sought to combine heuristically chosen optimization functions with greedy algorithmic techniques. Recently, there has been surge of effort to achieve balance between accuracy and rule size via principled objective functions and usage of combinatorial optimization techniques such as linear programming (LP) relaxations, sub-modular optimization, or Bayesian methods (Bertsimas, Chang, and Rudin 2012; Marchand and Shawe-Taylor 2002; Malioutov and Varshney 2013)(Boros et al. 2000; Wang et al. 2015). Motivated by the success of MaxSAT solving over the past decade, Malioutov and Meel proposed a MaxSAT-based approach, called MLIC (Maliotov and Meel 2018), that provides a precise control of accuracy vs. interpretability. The said approach was shown to provide interpretable Boolean formulas without significant loss of accuracy compared to the state of the art classifiers. MLIC, however, has poor scalability in terms of training time and times out for most instances beyond hundreds of samples. In this context, we ask: *Can we design a MaxSAT-based framework to efficiently construct interpretable rules without loss of accuracy and scaling to large real-world instances?*

The primary contribution of this paper is an affirmative answer to the above question. We first investigate the reason for poor scalability of MLIC and attribute it to large size (i.e., number of clauses) of MaxSAT queries constructed by MLIC. In particular, for training data of n samples over m boolean features, MLIC constructs a formula of size $\mathcal{O}(n \cdot m \cdot k)$ to construct a $k-$clause Boolean formula. We empirically observe that the performance of MaxSAT

solvers has worse than quadratic degradation in runtime with increase in the size of query. This leads us to propose a novel incremental framework, called IMLI, for learning interpretable rules using MaxSAT. In contrast to MLIC, IMLI makes p queries to MaxSAT solvers with each query of the size $\mathcal{O}(\frac{n}{p} \cdot m \cdot k)$. IMLI relies on first partitioning the data into p partitions and then incrementally learning rules on the p partitions in a linear order such that rule learned for the i-th partition not only uses the current partition but regularizes itself with respect to the rules learned from the first $i-1$ partitions. We conduct a comprehensive experimental study over the large set of benchmarks and show that IMLI significantly improves upon the runtime performance of MLIC by achieving speedup of up to three orders of magnitude. Furthermore, the rules learned by IMLI are significantly small and easy to interpret compared to that of the state of the art classifiers such as RIPPER.

Similar to Malioutov and Meel (2018), we hope that IMLI will excite researchers in machine learning and CP/SAT (Constraint Programming/Satisfiability) communities to consider this topic further: in designing new MaxSAT-based formulations and in turn designing the MaxSAT solvers tuned for interpretable machine learning.

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 variables appearing in F. A literal 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 *true*. 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 $F := C_1 \wedge C_2 \cdots \wedge C_k$, where each clause C_i is represented as disjunction of literals. We use $|C_i|$ to denote the number of literals in C_i. For two vectors $\mathbf{u}$ and $\mathbf{v}$ over propositional variables or constants (0, 1, $true$, $false$ etc.), we define $\mathbf{u} \vee \mathbf{v} = \bigvee_i (u_i \wedge v_i)$, 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 $\wedge$ between a variable and a constant follows the standard interpretation, i.e., $0 \wedge b = 0$ and $1 \wedge b = b$.

We consider a standard binary classification, where we are given a collection of training samples $\{\mathbf{X}_i, y_i\}$ where 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 return a class y, i.e., $y = \mathcal{R}(\mathbf{x})$. 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 define two rules $\mathcal{R}_1$ and $\mathcal{R}_2$ to be equivalent if $\forall i, \mathcal{R}_1(\mathbf{X}_i) = \mathcal{R}_2(\mathbf{X}_i)$. In this work, we restrict $\mathbf{x}$ and y to be Boolean (we discuss in Sect. 4.2 that such a restriction can be achieved without loss of generality) and focus on classifiers that can be expressed compactly in CNF. We use $clause(\mathcal{R}, i)$ to denote the i-th clause of $\mathcal{R}$. Furthermore, we use $|\mathcal{R}|$ to denote the rule-size of classifier $\mathcal{R}$ that is the sum of the count of literals in all the clauses, i.e., $|\mathcal{R}| = \Sigma_i |clause(\mathcal{R}, i)|$.

In this work, we focus on the weighted variant of CNF wherein a weight function is defined over clauses. For a clause C_i and weight function $W(\cdot)$, we use $W(C_i)$ to denote the weight of clause C_i. We say that a clause C_i is hard if $W(C_i) = \infty$, otherwise C_i is called a soft clause. To avoid notational clutter, we overload $W(\cdot)$ to denote the weight of an assignment or clause, depending on the context. We define weight of an assignment σ as the sum of weight of clauses that σ does not satisfy. Formally, $W(\sigma) = \Sigma_{i|\sigma \not\models C_i} W(C_i)$.

Given F and weight function $W(\cdot)$, the problem of MaxSAT is to find an assignment σ^* that has the minimum weight, i.e., $\sigma^* = \mathsf{MaxSAT}(F, W)$ if $\forall \sigma \neq \sigma^*, W(\sigma*) \leq W(\sigma)$. Our formulation will have positive clause weights, hence MaxSAT corresponds to satisfying as many clauses as possible, and picking the strongest clauses among the unsatisfied ones. Borrowing terminology of community focused on developing MaxSAT solvers, we are solving a partial weighted MaxSAT instance wherein we mark all the clauses with ∞ weight as hard and clauses with other positive value less than ∞ weight as soft and ask for a solution that optimizes the partial weighted MaxSAT formula. The knowledge of inner working of MaxSAT solvers and encoding of our representation into weighted MaxSAT is not required for this paper.

3 Problem Formulation

Given a training set $\{\mathbf{X}, \mathbf{y}\}$, our goal is to find an interpretable rule that is as accurate as possible. As noted earlier, there are several notions of interpretability. We follow the notion employed in Malioutov and Meel (Maliotov and Meel 2018), which focuses on the construction of rules involving few clauses each with few literals [1].

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 $\mathcal{R}$ which is the sparse (most interpretable) one.

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

A classifier rule, however, can not classify all samples correctly. Hence we choose a classifier that makes less prediction error. $\mathcal{E}_{\mathcal{R}}$ is 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.

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

λ is the regularization parameter balancing the trade-off between classifier complexity (opposite to interpretability in our model) and prediction accuracy. Higher value of λ guarantees less prediction error while sacrificing the sparsity of $\mathcal{R}$ by adding more literals in $\mathcal{R}$, and vice versa.

[1] An advantage of Malioutov and Meel's formulation is a formal notion of interpretability, which is amenable to formal analysis. We do not wish to claim that Malioutov and Meel's notion is the only formal definition of interpretability.

4 IMLI: MaxSAT-Based Incremental Learning Framework of Interpretable Rules

In this section, we present the primary contribution of this paper, IMLI, which is a MaxSAT-based incremental learning framework for interpretable classification rules. The core technical idea behind IMLI is to divide the training data into a fixed number p of partitions and employ MaxSAT based learning framework for each partition such that the MaxSAT query constructed for partition i is based on the training data for partition i and the rule learned until partition $i-1$. To this end, we use the notation $(\mathbf{X}^i, \mathbf{y}^i)$ to refer to the training data for the i-th partition. We assume that $\forall i, |\mathbf{X}^i| = |\mathbf{X}^{i-1}|$.

The rest of the section is organized as follows: we first describe the construction of MaxSAT query for the i-th partition in Sect. 4.1 to learn CNF rules, and then discuss the discretization techniques for real-world datasets in Sect. 4.2. The incrementality of IMLI gives rise to the challenge of having redundant literals in the learned rules; we address such redundancy in Sect. 4.3 and finally we discuss, in Sect. 4.4, how our framework for learning CNF rules can be easily extended to learn DNF rules as well.

4.1 Construction of MaxSAT Query

We now discuss the construction of a MaxSAT query, denoted by Q_i, for the i-th partition ($i \in [1, p]$). To construct the MaxSAT query for the i-th partition, we assume an access to the rule learned from the $(i-1)$-th partition (where $\mathcal{R}_0$ is an empty formula).

The construction of Q_i takes in four parameters: (i) k, the desired number of clauses in CNF rule, (ii) λ, the regularization parameter, (iii) a matrix $\mathbf{X}^i \in \{0,1\}^{n \times m}$ describing the binary value of m features for each of n samples with $\mathbf{X}^i_q$ being a binary valued vector for the q-th sample corresponding to feature vector $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$, (iv) a label vector $\mathbf{y}^i \in \{0,1\}^n$ containing a class label y^i_q for the sample $\mathbf{X}^i_q$. Consequently, IMLI constructs a MaxSAT query for the i-th partition and invokes an off-the-shelf MaxSAT solver to compute the underlying rule $\mathcal{R}_i$.

IMLI considers two types of propositional variables: (i) feature variables and (ii) noise variables. For the i-th partition, IMLI formulates a classifier rule $\mathcal{R}_i$ based on following intuition. Recall, a k-clause CNF rule $\mathcal{R}_i = \bigwedge_{l=1}^{k} C_l$ is represented as the conjunction of k clauses where clause C_l is the disjunction of feature variables. A sample $\mathbf{X}^i_q$ satisfies C_l if $\mathbf{X}^i_q$ has at least one similar feature whose representative variable is present in C_l. If $\mathbf{X}^i_q$ satisfies $\forall l, C_l$, then $\mathcal{R}_i(\mathbf{X}^i_q) = 1$ otherwise $\mathcal{R}_i(\mathbf{X}^i_q) = 0$. Since feature x_j can be present or not present in each of k clauses, IMLI considers k boolean variables, each denoted by b^l_j ($l \in [1, k]$) for feature x_j to denote its participation in each of k clauses. A sample $\mathbf{X}^i_q$, however, can be misclassified by $\mathcal{R}_i$ i.e., $\mathcal{R}_i(\mathbf{X}^i_q) \oplus y^i_q = 1$. IMLI introduces a noise variable η_q corresponding to sample $\mathbf{X}^i_q$ so that the assignment of η_q can be interpreted whether $\mathbf{X}^i_q$ is misclassified by $\mathcal{R}_i$ or not. Hence the key idea of IMLI for learning the i-th partition is to define a MaxSAT query over $k \times m + n$ propositional variables, denoted by $\{b^1_1, b^1_2, \ldots, b^1_m, \ldots, b^k_m, \eta_1, \ldots, \eta_n\}$. The MaxSAT query of IMLI consists of the following three sets of constraints:

1. Since our objective is to find sparser rules, the default objective of IMLI would be to add a constraint to falsify as many b^l_j as possible. As noted earlier, rule $\mathcal{R}_{i-1}$ from the $(i-1)$-th partition plays an important role in the construction of MaxSAT constraints of the i-th partition. Therefore, if $x_j \in clause(R_{i-1}, l)$, IMLI would deviate from its default behavior by adding a constraint to keep the corresponding literal $true$ in the optimal assignment. The weight corresponding to this clause is 1. We formalize our discussion as follows:

$$V^l_j := \begin{cases} b^l_j & \text{if } x_j \in clause(R_{i-1}, l) \\ \neg b^l_j & \text{otherwise} \end{cases}; \quad W(V^l_j) = 1$$

2. We use noise variables to handle mis-classifications and therefore, IMLI tries to falsify as many noise variables as possible. Since regularization parameter λ is proportionate to accuracy (higher λ ensures higher accuracy), IMLI puts λ weight to each following soft clause.

$$N_q := (\neg \eta_q); \qquad W(N_q) = \lambda$$

3. Let $\mathbf{B}_l = \{b^l_j \mid j \in [1, m]\}$. Here we provide the third set of constraints of IMLI.

$$D_q := (\neg \eta_q \rightarrow (y_q \leftrightarrow \bigwedge_{l=1}^{k} (\mathbf{X}_q \vee \mathbf{B}_l))); W(D_q) = \infty$$

Every hard clause D_q can be interpreted as follows. If η_q is assigned to *false* ($\neg \eta_q = true$) then $y^i_q = \mathcal{R}_i(\mathbf{X}^i_q) = \bigwedge_{l=1}^{k} (\mathbf{X}_q \vee \mathbf{B}_l)$. The operator "$\vee$" is defined in Sect. 2.

Finally, the set of constraints Q_i for the i-th partition constructed by IMLI is defined as follows:

$$Q_i := \bigwedge_{j=1,l=1}^{j=m,l=k} V^l_j \wedge \bigwedge_{q=1}^{n} N_q \wedge \bigwedge_{q=1}^{n} D_q$$

Next, we extract $\mathcal{R}_i$ from the solution of Q_i as follows.

Construction 1. *Let* $\sigma^* = \mathsf{MaxSAT}(Q_i, W)$*, then* $x_j \in clause(\mathcal{R}_i, l)$ *iff* $\sigma^*(b^l_j) = 1$.

In the rest of the manuscript, we will use $\mathcal{R}$ to denote $\mathcal{R}_p$.

4.2 Beyond Binary Features

We have considered that the feature value of a training sample is binary. Real-world datasets, however, contain categorical, real-valued or numerical features. We use the standard discretization technique to convert categorical and continuous (real or integer value) features to boolean features. We use *one hot encoding* to convert categorical features to binary features by introducing a boolean vector with the cardinality equal to the number of distinct categories of individual categorical features. Furthermore, we can discretize the continuous-valued features into binary features by comparing the feature value to a collection of thresholds within

range and introducing a boolean feature vector with cardinality proportional to the number of considered thresholds. Specifically, for a continuous feature x_c we consider a number of thresholds $\{\tau_1, \ldots, \tau_t\}$ where $\tau_i < \tau_{i+1}$ and define two separate Boolean features $I[x_c \geq \tau_i]$ and $I[x_c < \tau_i]$ for each τ_i. We present the following definitions based on the discretization of continuous features.

Definition 2. *$tval(b) : b \rightarrow \tau$ is a function over boolean variables corresponding to discretized binary features (from a continuous feature) and outputs the compared threshold value.*

Definition 3. *$op(b) : b \rightarrow \{\geq, <\}$ is a function over boolean variables corresponding to discretized binary features (from a continuous feature) and outputs the comparison operator between continuous feature value and $tval(b)$.*

Definition 4. *$siblings(b_i, b_j) : (b_i, b_j) \rightarrow \{true, false\}$ is a function over pair of boolean variables b_i, b_j and outputs $true$ if the boolean features corresponding to b_i, b_j are constructed by discretizing the same continuous feature and $op(b_i) = op(b_j)$.*

Example 4.1. *Consider a continuous feature x_c with range $(0, 100)$ and three thresholds $\{25, 50, 75\}$ associated with this feature.* IMLI *introduces 6 new boolean features $\{x_1 : I[x_c \geq 25], x_2 : I[x_c \geq 50], x_3 : I[x_c \geq 75], x_4 : I[x_c < 25], x_5 : I[x_c < 50], x_6 : I[x_c < 75]\}$. Following this discretization technique, the binary feature vector of a sample with feature value $x_c = 37.5$ is 100011, because among the 6 introduced boolean features $x_1 : I[37.5 \geq 25] = 1$, $x_5 : I[37.5 < 50] = 1$, and $x_6 : I[37.5 < 75] = 1$.*

Example 4.2. *In Example 4.1, b_i is a boolena variable corresponding to feature x_i. Now $tval(b_1) = 25$, $op(b_1) =$ " $\geq$ ", $siblings(b_1, b_2) = true$, and $siblings(b_1, b_4) = false$.*

4.3 Redundancy Removal

Given the incremental procedure of learning $\mathcal{R}$ where the constraints for the i-th partition are influenced from the rule learned until the $(i-1)$-th partition, one key challenge is to address potential redundancy in the learned rules. In particular, we observe that redundancy manifests itself in binary features corresponding to continuous-valued features as the $(i-1)$-th partition might suggest inclusion of feature $I[x_c < \tau_u]$ while the i-th partition also suggests inclusion of feature $I[x_c < \tau_v]$ where $\tau_u \neq \tau_v$. To this end, we present Algorithm 1 to remove redundant literals.

Algorithm 1 Remove Redundancy

procedure REMOVEREDUNDANTLITERALS($\mathcal{R}$)
  for each clause C_l of $\mathcal{R}$ **do**
    for each pair $\langle b_i^l, b_j^l \rangle$ where $\sigma(b_i^l) = \sigma(b_j^l) = 1$, $siblings(b_i^l, b_j^l) = true$, and $tval(b_i^l) < tval(b_j^l)$ **do**
      if $op(b_i^l) = op(b_j^l) =$ " $\geq$ " **then**
        $\mathcal{R}' = \mathcal{R}[\sigma(b_j^l) \mapsto 0]$ ▷ b_j^l is redundant
      else
        $\mathcal{R}' = \mathcal{R}[\sigma(b_i^l) \mapsto 0]$
  return $\mathcal{R}'$

Lemma 5. *$|\mathcal{R}'| \leq |\mathcal{R}|$ and $\mathcal{R}'$ is equivalent to $\mathcal{R}$.*

Proof. For lack of space, proof is removed. □

4.4 Learning DNF Rules

Primarily we focus on learning rule $\mathcal{R}$ which is in CNF form. We can also apply incremental technique for learning DNF rules. Suppose, we want to learn a rule $y = S(x)$ where $S(x)$ is expressible in DNF. We show that $y = S(x) \leftrightarrow \neg(y = \neg S(x))$. Here $\neg S(x)$ is in CNF. Therefore, to learn DNF rule $S(x)$, we simply call IMLI with ¬y as input for all p batches, learn CNF rule, and finally negate the learned rule. Hence Algorithm 1 can be directly applied.

5 Experiment

We have implemented a prototype implementation in Python to evaluate the performance of IMLI[2]. The experiment has been conducted on high performance computer cluster, where each node consists of E5-2690 v3 CPU with 24 cores, 96GB of RAM, and 130,000 CPU hours. We have conducted an extensive set of experiments on publicly available benchmarks from UCI repository (Dheeru and Karra Taniskidou 2017) to answer the following questions.

1. How do the training time and accuracy of IMLI compare to that of state of the art classifiers including both interpretable and non-interpretable ones?
2. How do accuracy, rule size, and training time of IMLI vary with regularization parameter λ and the number of partitions p?
3. How interpretable are the rules generated by IMLI?

In summary, the experimental results demonstrate that IMLI can scale to large datasets involving tens of thousands of samples with hundreds of binary features. In contrast to MLIC, IMLI achieves up to three orders of magnitude improvement in training time without loss of accuracy and interpretability. IMLI generates rules which are not only interpretable but also accurate compared to other classifiers, which often produce non-interpretable models for the sake of accuracy.

5.1 Experiment Methodology:

To measure the performance gain over MLIC, we measure the accuracy and training time of IMLI vis-a-vis MLIC. We also perform comparisons with another state of the art classifier RIPPER and other (mostly) non-interpretable classifiers such as random forest (RF), support vector classifier (SVC), Nearest Neighbors classifier (NN), l_1-penalized Logistic Regression (LR).

The number of parameter values is comparable (10) for each technique. For RF and RIPPER, we use control based on the cutoff of the number of examples in the leaf node. For SVC, NN, and LR we discretize the regularization parameter on a logarithmic grid. For both IMLI and MLIC, we have two choices of $\lambda \in \{5, 10\}$, three choices of $k \in \{1, 2, 3\}$, and two choices of the type of rule as $\{\text{CNF}, \text{DNF}\}$. For IMLI

[2] https://github.com/meelgroup/mlic

we vary the number of partitions p for each dataset such that each partition has at least eight samples and at most 512 samples. For all classifiers, we set the training time cutoff to be 1000 seconds.

We perform an assessment of test accuracy on a holdout set and mean validation accuracy on a 10-fold cross-validation set (holdout set 10%, validation set 9%, training set 81%). We compute test accuracy and mean validation accuracy across the ten folds for each choice of the parameter for each technique, and report test accuracy, mean validation accuracy, and mean training time for a choice of the parameter which incurs the best test accuracy. To remove the bias of a particular holdout set we perform ten repetitions with different holdout sets and present the mean statistics.

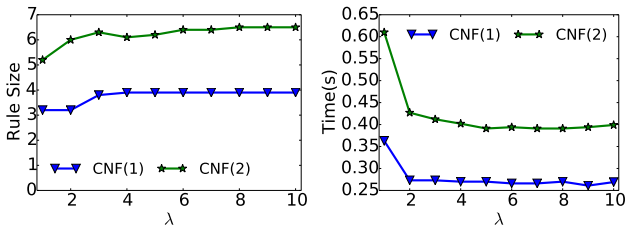
For MLIC and IMLI, we experimented with different MaxSAT solvers and finally chose MaxHS (Davies and Bacchus 2011) for MLIC since MaxSAT queries generated by MLIC timeout for all the solvers and MaxHS is the only solver to return the best found answer so far. In contrast, queries constructed by IMLI are easier and the best runtime performance is obtained by using Open-WBO solver (Martins, Manquinho, and Lynce 2014).

5.2 Results

Comparison Among Different Classifiers: Table 1 presents the comparison of IMLI vis-a-vis typical interpretable and non-interpretable classifiers. The first three columns list the name, size (number of samples), and the number of binary features (discretized) for each dataset. The next seven columns present test accuracy, validation accuracy, and training time of the classifiers.

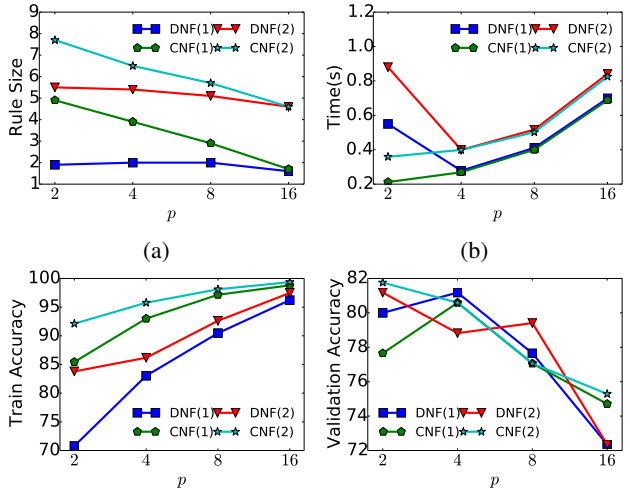
In Table 1 we observe that MLIC and RIPPER have slightly higher accuracy than IMLI. Specifically considering all datasets MLIC (resp. RIPPER) has on average 1.12% (resp. 0.12%) higher test accuracy and 3.09% (resp. 2.29%) higher validation accuracy than that of IMLI. In contrast, IMLI takes up to three order of magnitude less training time compared to MLIC and upto one order of magnitude less time compared to RIPPER. Interestingly, IMLI is competitive to black-box classifiers, e.g. SVC and NN for large datasets. In this context, we think IMLI achieves a sweet spot in achieving significant runtime improvement in training without losing accuracy.

At this point, one may wonder as to whether minor loss in accuracy also leads to loss of interpretability. To this end, we illustrate a detailed comparison among the generated rules of IMLI, RIPPER, and MLIC in Table 2. We observe that rule size of IMLI is significantly smaller than that of RIPPER and MLIC. In particular, note that IMLI can generate rules with size less than eight for all the datasets (exception in Adult dataset where IMLI still has the most sparse rule), thereby demonstrating the sparsity of generated rules. In contrast, MLIC and RIPPER generate rules of significantly larger size than IMLI. As indicated earlier, sparsity is only one of several possible approaches to quantify interpretability. Therefore, we also decided to observe the generated rules and interestingly, the generated rules seem very intuitive.



(a) $k \in \{1, 2\}, p = 4$ (b) $k \in \{1, 2\}, p = 4$

Figure 1: Effect of regularization parameter λ on rule size and training time. The number within parenthesis denotes the number of clause k for the respective rule.



(a) (b) (c) (d)

Figure 2: Effect of the number of partitions p on rule size, training time, train accuracy, and validation accuracy ($k \in \{1, 2\}, \lambda = 10$). The number within parenthesis denotes the number of clause k for the respective rule.

Varying Regularization Parameter λ: In Figure 1 we present the result for varying λ. Our experiment result finds a similar observation in all the datasets, and here we present result for Parkinsons dataset.

Recall that size of a rule is the total number of literals appearing in $\mathcal{R}$. As we increase the value of λ, rule size (Figure 1a) and the time taken to solve the MaxSAT query (Figure 1b) decreases. When $\lambda = 1$, all the soft clauses have equal weight. However, when λ is higher, soft clause N_q is put a higher weight than V_j^l, which turns out in finding the solution of the query requiring less time because of the priority among soft clauses. Therefore, the generated rule becomes sparser. We find a similar trend for DNF rules too. In empirical study we find that as we increase λ, training accuracy increases gradually but validation accuracy and test accuracy do not follow a monotonic behavior in the partition-based learning.

Varying The Number of Training Partitions p: Figure 2 presents the effect on rule size, training time, train accuracy, and validation accuracy as we vary p. For all the datasets we find a similar observation and here present the result for Parkinsons dataset for ease of exposition.

In Figure 2a we observe that the size of the rule decreases

Dataset	Size	Features	LR	NN	RF	SVC	RIPPER	MLIC	IMLI
Parkinsons	195	392	97.5 [69.76] (0.22)	90 [82.68] (0.14)	97.5 [80.98] (1.7)	87.5 [83.5] (0.18)	85.00 [76.74] (2.92)	97.5 [82.35] (114.75)	95 [79.41] (0.37)
Ionosphere	351	540	93.06 [90.64] (0.32)	86.11 [87.18] (0.26)	95.83 [92.77] (1.72)	95.83 [90.65] (0.26)	93.06 [85.73] (3.3)	93.06 [91.94] (917.13)	91.67 [85.48] (0.5)
WDBC	569	540	98.28 [96.77] (0.33)	95.69 [97.27] (0.45)	96.55 [96.68] (1.8)	96.55 [97.16] (0.27)	92.24 [93.54] (3.53)	93.97 [95.1] Timeout	89.66 [91.18] (0.78)
Blood	748	64	80 [75.92] (0.2)	76.67 [76.14] (0.2)	76 [76.22] (1.68)	76 [76.22] (0.18)	76 [76.22] (2.23)	75.33 [77.61] (5.96)	76 [76.12] (0.24)
PIMA	768	134	75.32 [74.75] (0.3)	77.92 [73.23] (0.32)	76.62 [75.54] (1.99)	75.32 [76.63] (0.37)	75.32 [74.36] (2.58)	75.97 [71.74] Timeout	73.38 [68.12] (0.74)
Tom's HW	28179	844	96.98 [97.12] (2.24)	94.11 [93.91] (910.36)	97.11 [97.35] (27.11)	96.83 [97.1] (354.15)	96.75 [97.12] (37.81)	96.61 [96.55] Timeout	96.86 [96.23] (23.67)
Adult	32561	262	84.58 [84.99] (5.8)	83.46 [83.62] (640.81)	84.31 [84.68] (36.64)	84.39 [84.69] (918.26)	83.72 [83.49] (37.66)	79.72 [79.53] Timeout	80.84 [77.43] (25.07)
Credit-default	30000	334	80.81 [82.16] (6.87)	79.61 [80.2] (872.97)	80.87 [82.13] (37.72)	80.69 [82.1] (847.93)	80.97 [82.07] (20.37)	80.72 [82.14] Timeout	79.41 [75.81] (32.58)
Twitter	49999	1050	95.67 [96.32] (3.99)	Timeout	95.16 [96.46] (67.83)	Timeout	95.56 [96.16] (98.21)	94.78 [95.69] Timeout	94.69 [95.08] (59.67)

Table 1: Comparisons of classification accuracy with 10-fold cross validation for different classifiers. For every cell in the last seven columns the top value represents the test accuracy (%) on unseen data, the middle value surrounded by square bracket represents average validation accuracy (%) of 10-fold, and the bottom value surrounded by parenthesis represents the average training time in seconds.

Dataset	RIPPER	MLIC	IMLI
Parkinsons	2.6	2	8
Ionosphere	9.6	13	5
WDBC	7.6	14.5	2
Blood	1	3	3.5
Adult	107.55	44.5	28
PIMA	8.25	16	3.5
Tom's HW	30.33	2	2.5
Twitter	21.6	20.5	6
Credit	14.25	6	3

Table 2: Size of the rule of interpretable classifiers.

as p increases. This observation can be attributed to the decrease in the number of training samples per partition with the increase in the number of partitions and consequently, smaller rules suffice. In Figure 2b IMLI empirically shows that the training time at first decreases significantly and then increases slowly with the increase in p. This observation can be attributed to the combined effect of the number of queries and the size of queries. Initially, we achieve a significant reduction in the size of query while the number of queries eventually dominate the runtime.

In Figure 2c we observe that IMLI tends to make less training error as p goes higher because IMLI learns on fewer samples with fixed λ value. Moreover, we observe that CNF rules have higher train accuracy than DNF rules, and 2-clause rules have higher train accuracy than 1-clause rules for both CNF and DNF rules.

In Figure 2d we notice a decrease in validation accuracy as we allow more partitions because learning on fewer samples results in a rule that has less predictive power on validation set. For small p, IMLI, however, ensures higher validation accuracy if the rule has more clauses. Moreover, effect of the number of partitions on test accuracy computed on unseen data does not follow any specific pattern.

In summary, we observe that the number of partitions gives a sound handle to the end user to tradeoff the training time, validation accuracy, and interpretability of the rules.

6 Conclusion

In this paper, we present IMLI: an incremental framework for MaxSAT-based learning of interpretable classification rules. Extensive experiments on UCI datasets demonstrate that IMLI achieves up to three orders of magnitude improvement in training time with only a minor loss of accuracy. We think IMLI highlights the promise of MaxSAT-based approach and opens up several interesting directions of future research at the intersection of AI and SAT/SMT community. In particular, it would be an interesting direction of future research if the MaxSAT solvers can be designed to take advantage of incrementality of IMLI.

References

Barocas, S.; Hardt, M.; and Narayanan, A. 2017. Fairness and machine learning. *NIPS Tutorial*.

Bertsimas, D.; Chang, A.; and Rudin, C. 2012. An integer optimization approach to associative classification. In *Proc. of NIPS*.

Boros, E.; Hammer, P.; Ibaraki, T.; Kogan, A.; Mayoraz, E.; and Muchnik, I. 2000. An implementation of logical analysis of data. *Proc. of TKDE*.

Davies, J., and Bacchus, F. 2011. Solving maxsat by solving a sequence of simpler sat instances. In *Proc. of CP*.

Dheeru, D., and Karra Taniskidou, E. 2017. UCI machine learning repository.

Doshi-Velez, F., and Kim, B. 2017. Towards a rigorous science of interpretable machine learning.

Letham, B.; Rudin, C.; McCormick, T. H.; Madigan, D.; et al. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*.

Maliotov, D., and Meel, K. S. 2018. Mlic: A maxsat-based framework for learning interpretable classification rules. In *Proc. of CP*.

Malioutov, D. M., and Varshney, K. R. 2013. Exact rule learning via boolean compressed sensing. In *Proc. of ICML*.

Marchand, M., and Shawe-Taylor, J. 2002. The set covering machine. *Journal of Machine Learning Research* (Dec).

Martins, R.; Manquinho, V.; and Lynce, I. 2014. Open-wbo: A modular maxsat solver. In *Proc. of SAT*.

Otte, C. 2013. Safe and interpretable machine learning: a methodological review. In *Computational intelligence in intelligent data analysis*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proc. of KDD*.

Wang, T.; Rudin, C.; Doshi-Velez, F.; Liu, Y.; Klampfl, E.; and MacNeille, P. 2015. Or's of and's for interpretable classification, with application to context-aware recommender systems.