Truly Unordered Probabilistic Rule Sets for Multi-class Classification

Lincen Yang[0000−0003−1936−2784] and Matthijs van Leeuwen[0000−0002−0510−3549]

LIACS, Leiden University, Leiden, The Netherlands
l.yang, m.van.leeuwen@liacs.leidenuniv.nl

Abstract. Rule set learning has long been studied and has recently been frequently revisited due to the need for interpretable models. Still, existing methods have several shortcomings: 1) most recent methods require a binary feature matrix as input, while learning rules directly from numeric variables is understudied; 2) existing methods impose orders among rules, either explicitly or implicitly, which harms interpretability; and 3) currently no method exists for learning probabilistic rule sets for multi-class target variables (there is only one for probabilistic rule lists). We propose TURS, for Truly Unordered Rule Sets, which addresses these shortcomings. We first formalize the problem of learning truly unordered rule sets. To resolve conflicts caused by overlapping rules, i.e., instances covered by multiple rules, we propose a novel approach that exploits the probabilistic properties of our rule sets. We next develop a two-phase heuristic algorithm that learns rule sets by carefully growing rules. An important innovation is that we use a surrogate score to take the global potential of the rule set into account when learning a local rule.
Finally, we empirically demonstrate that, compared to non-probabilistic and (explicitly or implicitly) ordered state-of-the-art methods, our method learns rule sets that not only have better interpretability but also better predictive performance.

1 Introduction

When using predictive models in sensitive real-world scenarios, such as in health care, analysts seek for intelligible and reliable explanations for predictions. Classification rules have considerable advantages here, as they are directly readable by humans. While rules all seem alike, however, some are more interpretable than others. The reason lies in the subtle differences of how rules form a model. Specifically, rules can form an unordered *rule set*, or an explicitly ordered *rule list*; further, they can be categorized as probabilistic or non-probabilistic.

In practice, probabilistic rules should be preferred because they provide information about the uncertainty of the predicted outcomes, and thus are useful when a human is responsible to make the final decision, as the expected "utility" can be calculated. Meanwhile, unordered rule sets should also be preferred, as they have better properties regarding interpretability than ordered rule lists. While no agreement has been reached on the precise definition of interpretability of machine learning models [16, 14], we specifically treat interpretability with

domain experts in mind. From this perspective, a model's interpretability intuitively depends on two aspects: the degree of difficulty for a human to comprehend the model itself, and to understand a single prediction. Unordered probabilistic rule sets are favorable with respect to both aspects, for the following reasons. First, comprehending ordered rule lists requires comprehending not only each individual rule, but also the relationship among the rules, while comprehending unordered rule sets requires only the former. Second, the explanation for a single prediction of an ordered rule list must contain the rule that the instance satisfies, together with all of its preceding rules, which becomes incomprehensible when the number of preceding rules is large.

Further, crucially, existing methods for rule set learning claim to learn unordered rule sets, but most of them are not truly unordered. The problem is caused by *overlap*, i.e., a single instance satisfying multiple rules. Ad-hoc schemes are widely used to resolve prediction conflicts caused by overlaps, typically by ranking the involved rules with certain criteria and always selecting the highest ranked rule [24, 12] (e.g., the most accurate one). This, however, imposes implicit orders among rules, making them entangled instead of truly unordered.

This can badly harm interpretability: to explain a single prediction for an instance, it is now insufficient to only provide the rules the instance satisfies, because other higher-ranked rules that the instance does *not* satisfy are also part of the explanation. For instance, imagine a patient is predicted to have *Flu* because they have *Fever*. If the model also contains the higher-ranked rule *"Blood in stool* $\rightarrow$ *Dysentery"*, the explanation should include the fact that *"Blood in stool"* is not true, because otherwise the prediction would change to *Dysentery*. If the model contains many rules, it becomes impractical to have to go over all higher-ranked rules for each prediction.

Learning truly unordered probabilistic rule sets is a very challenging problem though. Classical rule set learning methods usually adopt a separate-and-conquer strategy, often sequential covering: they iteratively find the next rule and remove instances satisfying this rule. This includes 1) binary classifiers that learn rules only for the "positive" class [8], and 2) its extension to multi-class targets by the one-versus-rest paradigm, i.e., learning rules for each class one by one [4, 2]. Importantly, by iteratively removing instances the *probabilistic predictive conflicts* caused by overlaps, i.e., rules having different probability estimates for the target, are ignored. Recently proposed rule learning methods go beyond separate-and-conquer by leveraging discrete optimization techniques [24, 21, 22, 12, 5], but this comes at the cost of requiring a binary feature matrix as input. Moreover, these methods are neither probabilistic nor truly unordered, as they still use ad-hoc schemes to resolve predictive conflicts caused by overlaps.

Approach and contributions. To tackle these challenges and learn truly unordered probabilistic rules, we first formalize rule sets as probabilistic models. We adopt a probabilistic model selection approach for rule set learning, for which we design a criterion based on the minimum description length (MDL) principle [10]. Second, we propose a novel surrogate score based on decision trees that we use to evaluate the potential of incomplete rule sets. Third, we are the first to

design a rule learning algorithm that deals with probabilistic conflicts caused by overlaps already during the rule learning process. We point out that rules that have been added to the rule set may become obstacles for new rules, and hence carefully design a two-phase heuristic algorithm, for which we adopt diverse beam search [19]. Last, we benchmark our method, named TURS, for Truly Unordered Rule Sets, against a wide range of methods. We show that the rule sets learned by TURS, apart from being probabilistic and truly unordered, have better predictive performance than existing rule list and rule set methods.

2 Related Work

Rule lists. Rules in a rule list are connected by IF-THEN-ELSE statements. Existing methods include CBA [13], ordered CN2 [3], PART [6], and the recently proposed CLASSY [17] and Bayesian rule list [23]. We argue that rule lists are more difficult to interpret than rule sets because of their explicit orders.

One-versus-rest learning. This category focuses on only learning rules for a single class label, i.e., the "positive" class, which is already sufficient for binary classification [21, 5, 22]. For multi-class classification, two approaches exist. The first, taken by RIPPER [4] and C4.5 [18], is to learn each class in a certain order. After all rules for a single class have been learned, all covered instances are removed (or those with this class label). The resulting model is essentially an ordered list of rule sets, and hence is more difficult to interpret than rule set.

The second approach does not impose an order among the classes; instead, it learns a set of rules for each class against all other classes. The most well-known are unordered-CN2 and FURIA [2, 11]. FURIA avoids dealing with conflicts of overlaps by using all rules for predicting unseen instances; as a result, it cannot provide a single rule to explain its prediction. Unordered-CN2, on the other hand, handles overlaps by "combining" all overlapping rules into a "hypothetical" rule, which sums up all instances in all overlapping rules and hence ignoring probabilistic conflicts for constructing rules. In Section 6, we show that our method learns smaller rule sets with better predictive performance than unordered-CN2.

Multi-class rule sets. Very few methods exist for directly learning rules for multi-class targets, which is algorithmically more challenging than the one-versus-rest paradigm, as the separate-and-conquer strategy is not applicable. To the best of our knowledge, the only existing methods are IDS [12] and DRS [24]. Both are neither probabilistic nor truly unordered. To handle conflicts of overlaps, IDS follows the rule with the highest F1-score, and DRS uses the most accurate rule.

Last, different but related approaches include 1) decision tree based methods such as CART [1], which produce rules that are forced to share many "attributes" and hence are longer than necessary, as we will empirically demonstrate in Section 6, and 2) a Bayesian rule mining [9] method, which adopts naive bayes with the mined rules for prediction, and hence does not produce a rule set model in the end. The 'lazy learning' approach for rule-based models can also avoid

the conflicts of overlaps [20], but no global rule set model describing the whole dataset is constructed in this case.

3 Rule Sets as Probabilistic Models

We first formalize individual rules as *local* probabilistic models, and then define rule sets as *global* probabilistic models. The key challenge lies in how to define $P(Y = y|X = x)$ for an instance (x, y) that is covered by multiple rules.

3.1 Probabilistic Rules

Denote the input random variables by $X = (X_1, \ldots, X_d)$, where each X_i is a one-dimensional random variable representing one dimension of X, and denote the categorical target variable by $Y \in \mathscr{Y}$. Further, denote the dataset from which the rule set can be induced as $D = \{(x_i, y_i)\}_{i\in[n]}$, or (x^n, y^n) for short. Each (x_i, y_i) is an instance. Then, a probabilistic rule S is written as

$$(X_1 \in R_1 \wedge X_2 \in R_2 \wedge \ldots) \rightarrow P_S(Y), \tag{1}$$

where each $X_i \in R_i$ is called a *literal* of the *condition* of the rule. Specifically, each R_i is an interval (for a quantitative variable) or a set of categorical levels (for a categorical variable).

A probabilistic rule of this form describes a subset S of the full sample space of X, such that for any $x \in S$, the conditional distribution $P(Y|X = x)$ is approximated by the probability distribution of Y conditioned on the event $\{X \in S\}$, denoted as $P(Y|X \in S)$. Since in classification Y is a discrete variable, we can parametrize $P(Y|X \in S)$ by a parameter vector $\boldsymbol{\beta}$, in which the jth element β_j represents $P(Y = j|X \in S)$, for all $j \in \mathscr{Y}$. We therefore denote $P(Y|X \in S)$ as $P_{S,\boldsymbol{\beta}}(Y)$, or $P_S(Y)$ for short. To estimate $\boldsymbol{\beta}$ from data, we adopt the maximum likelihood estimator, denoted as $P_{S,\hat{\boldsymbol{\beta}}}(Y)$, or $\hat{P}_S(Y)$ for short.

Further, if an instance (x, y) satisfies the condition of rule S, we say that (x, y) is *covered* by S. Reversely, the *cover* of S denotes the instances it covers. When clear from the context, we use S to both represent the rule itself and/or its cover, and define the number of covered instances $|S|$ as its *coverage*.

3.2 Truly Unordered Rule Sets as Probabilistic Models

While a rule set is simply a set of rules, the challenge lies in how to define rule sets as probabilistic models while keeping the rules truly unordered. That is, how do we define $P(Y|X = x)$ given a rule set M, i.e., a model, and its parameters? We first explain how to do this for a single instance of the training data, using a simplified setting where at most two rules cover the instance. We then discuss—potentially unseen—test instances and extend to more than two rules covering an instance. Finally, we define a rule set as a probabilistic model.

Class probabilities for a single training instance. Given a rule set M with K individual rules, denoted $\{S_i\}_{i\in[K]}$, any instance (x, y) falls into one of

four cases: 1) exactly one rule covers x; 2) at least two rules cover x and no rule's cover is the subset of another rule's cover (*multiple non-nested*); 3) at least two rules cover x and one rule's cover is the subset of another rule's cover (*multiple nested*); and 4) no rule in M covers x.

To simplify the notation, we here consider at most two rules covering an instance—we later describe how we can trivially extend to more than two rules.

Covered by one rule. When exactly one rule $S \in M$ covers x, we use $P_S(Y)$ to "approximate" the conditional probability $P(Y|X = x)$. To estimate $P_S(Y)$ from data, we adopt the maximum likelihood (ML) estimator $\hat{P}_S(Y)$, i.e.,

$$\hat{P}_S(Y = j) = \frac{|\{(x, y) : x \in S, y = j\}|}{|S|}, \forall j \in \mathscr{Y}. \tag{2}$$

Note that we do not exclude instances in S that are also covered by other rules (i.e., in overlaps) for estimating $P_S(Y)$. Hence, the probability estimation for each rule is independent of other rules; as a result, each rule is *self-standing*, which forms the foundation of a truly unordered rule set.

Covered by two non-nested rules. Next, we consider the case when x is covered by S_i and S_j, and neither $S_i \subseteq S_j$ nor $S_j \subseteq S_i$, i.e., the rules are non-nested.

When an instance is covered by two non-nested, partially overlapping rules, we interpret this as probabilistic *uncertainty*: we cannot tell whether the instance belongs to one rule or the other, and therefore approximate its conditional probability by the *union* of the two rules. That is, in this case we approximate $P(Y|X = x)$ by $P(Y|X \in S_i \cup S_j)$, and we estimate this with its ML estimator $\hat{P}(Y|X \in S_i \cup S_j)$, using all instances in $S_i \cup S_j$.

This approach is particularly useful when the estimator of $P(Y|X \in S_i \cap S_j)$, i.e., conditioned on the event $\{X \in S_i \cap S_j\}$, is indistinguishable from $\hat{P}(Y|X \in S_i)$ and $\hat{P}(Y|X \in S_j)$. Intuitively, this can be caused by two reasons: 1) $S_i \cap S_j$ consists of very few instances, so the variance of the estimator for $P(Y|X \in S_i \cap S_j)$ is large; 2) $P(Y|X \in S_i \cap S_j)$ is just very similar to $P(Y|X \in S_i)$ and $P(Y|X \in S_i)$, which makes it undesirable to create a separate rule for $S_i \cap S_j$. Our model selection approach, explained in Section 4, will ensure that a rule set with non-nested rules has high goodness-of-fit only if this 'uncertainty' is indeed the case.

Covered by two nested rules. When x is covered by both S_i and S_j, and S_i is a subset of S_j, i.e., $x \in S_i \subseteq S_j$, the rules are nested[1]. In this case, we approximate $P(Y|X = x)$ by $P(Y|X \in S_i)$ and interpret S_i as an *exception* of S_j. Having such nested rules to model such exceptions is intuitively desirable, as it allows to have general rules covering large parts of the data while being able to model smaller, deviating parts. In order to preserve the self-standing property of individual rules, for $x \in S_j \setminus S_i$ we still use $P(Y|X \in S_j)$ rather than $P(Y|X \in S_j \setminus S_i)$. Although this might seem counter-intuitive at first glance, using $P(Y|X \in S_j \setminus S_i)$ would implicitly impose an order between S_j and S_i,

[1] Note that "nestedness" is based on the rules' covers rather than on their conditions. For instance, if S_i is $X_1 <= 1$ and S_j is $X_2 <= 1$, S_i and S_j could still be nested.

or—equivalently—implicitly change S_j to another rule that only covers instances in $S_j \wedge \neg S_i$.

Not covered by any rule. When no rule in M covers x, we say that x belongs to the so-called "else rule" that is part of every rule set and equivalent to $x \notin \bigcup_i S_i$. Thus, we approximate $P(Y|X = x)$ by $P(Y|X \notin \bigcup_i S_i)$. We denote the else rule by S_0 and write $S_0 \in M$ for the else rule in M. Observe that the else rule is the only rule in every rule set that depends on the other rules and is therefore not self-standing; however, it will also have no overlap with other rules by definition.

Predicting for a new instance. When an unseen instance x' comes in, we predict $P(Y|X = x')$ depending on which of the aforementioned four cases it satisfies. An important question is whether we always need access to the training data, i.e., whether the probability estimates we obtain from the training data points are sufficient for predicting $P(Y|X = x')$. Specifically, if x' is covered by non-nested S_i and S_j, $P(Y|X = x')$ is predicted as $\hat{P}(Y|X \in S_i \cup S_j)$. However, if there are no training data points covered both by S_i and S_j, then we would not obtain $\hat{P}(Y|X \in S_i \cup S_j)$ in the training phase. Nevertheless, in this case we have $|S_i \cup S_j| = |S_i| + |S_j|$, and hence

$$\hat{P}(Y|X \in S_i \cup S_j) = \frac{|S_i|\hat{P}(Y|X \in S_i) + |S_j|\hat{P}(Y|X \in S_j)}{|S_i| + |S_j|}. \tag{3}$$

Thus, if x' is covered by one rule, two nested rules, or no rule in M, the corresponding probability estimates are already obtained during training. Thus, we conclude that access to the training data is not necessary for prediction.

Extension to overlaps of multiple rules. Whenever an instance x is covered by multiple rules, denoted $J = \{S_i, S_j, S_k, ...\}$, three cases may happen. The first case is all rules in J are nested. Without loss of generality, assume that $S_i \subseteq S_j \subseteq S_k \subseteq ...$; then, following the rationale for case of two nested rules, $P(Y|X = x)$ should be approximated by $P_{S_i}(Y)$. Therefore, when x is covered by multiple nested rules, only the "smallest" rule matters and we can act as if x is only covered by that single rule.

The second case is that all rules in J are non-nested with each other. Following the solution for modeling two non-nested rules, we use $P(Y|X \in \bigcup_{S \in J} S)$.

The third case is a mix of the previous two cases, i.e., rules in J are partially nested. In this case, we iteratively go over all $S \in J$: if there exists an $S' \in J$ satisfying $S' \subseteq S$ we remove S from J, and continue iterating until no nested overlap in J remains. If one single rule is left, we act as if x is covered by that single rule; otherwise, we follow the paradigm of modeling the non-nested overlaps with the rules left in J.

Probabilistic rule sets. We can now build upon the previous to define rule sets as probabilistic models. Formally, the probabilistic model corresponding to a rule set M is a family of probability distributions, denoted $P_{M,\theta}(Y|X)$ and parametrized by θ. Specifically, θ is a parameter vector representing all necessary probabilities of Y conditioned on events $\{X \in G\}$, where G is either a single rule or the union of multiple rules. θ is estimated from data by estimating each $P(Y|X \in G)$ by its maximum likelihood estimator. The resulting estimated

vector is denoted as $\hat{\theta}$ and contains $\hat{P}(Y|X \in G)$ for all $G \in \mathscr{G}$, where $\mathscr{G}$ consists of all individual rules and the unions of overlapping rules in M.

Finally, we assume the dataset $D = (x^n, y^n)$ to be i.i.d. Specifically, let us define $(x, y) \vdash G$ for the following two cases: 1) when G is a single rule (including the else rule), then $(x, y) \vdash G \iff x \in G$; and 2) when G is a union of multiple rules, e.g., $G = \bigcup S_i$, then $(x, y) \vdash G \iff x \in \bigcap S_i$. We then have

$$P_{M,\theta}(y^n|x^n) = \prod_{G \in \mathscr{G}} \prod_{(x,y) \vdash G} P(Y = y|X \in G). \tag{4}$$

4 Rule Set Learning as Probabilistic Model Selection

Exploiting the formulation of rule sets as probabilistic models, we define the task of learning a rule set as a probabilistic model selection problem. Specifically, we use the minimum description length (MDL) principle for model selection.

4.1 Normalized Maximum Likelihood Distributions for Rule Sets

The MDL principle is one of the best off-the-shelf model selection methods and has been widely used in machine learning and data mining [10]. Although rooted in information theory, it has been recently shown that MDL-based model selection can be regarded as an extension of Bayesian model selection [10].

The core idea of MDL-based model selection is to assign a single probability distribution to the data given a rule set M, the so-called *universal distribution* denoted by $P_M(Y^n|X^n = x^n)$. Informally, $P_M(Y^n|X^n = x^n)$ should be a representative of the rule set model—as a family of probability distributions—$\{P_{M,\theta}(y^n|x^n)\}_\theta$. The theoretically optimal "representative" is defined to be the one that has minimax regret, i.e.,

$$\arg\min_{P_M} \max_{z^n \in \mathscr{Y}^n} -\log_2 P_M(Y^n = z^n|X^n = x^n) - \left(-\log_2 P_{\hat{\theta}(x^n,z^n)}(Y^n = z^n|X^n = x^n)\right). \tag{5}$$

We write the parameter estimator as $\hat{\theta}(x^n, z^n)$ to emphasize that it depends on the values of the target variables Y^n. The unique solution to P_M of Equation 5 is the so-called normalized maximum likelihood (NML) distribution:

$$P_M^{NML}(Y^n = y^n|X^n = x^n) = \frac{P_{M,\hat{\theta}(x^n,y^n)}(Y^n = y^n|X^n = x^n)}{\sum_{z^n \in \mathscr{Y}^n} P_{M,\hat{\theta}(x^n,z^n)}(Y^n = z^n|X^n = x^n)}. \tag{6}$$

That is, we "normalize" the distribution $P_{M,\hat{\theta}}(.)$ to make it a proper probability distribution, which requires the sum of all possible values of Y^n to be 1. Hence, we have $\sum_{z^n \in \mathscr{Y}^n} P_M^{NML}(Y^n = z^n|X^n = x^n) = 1$ [10].

4.2 Approximating the NML Distribution

A crucial difficulty in using the NML distribution in practice is the computation of the normalizing term $\sum_{z^n} P_{\hat{\theta}(x^n,z^n)}(Y^n = z^n|X^n = x^n)$. Efficient algorithms

almost only exist for exponential family models [10], hence we approximate the term by the product of the normalizing terms for the individual rules.

NML distribution for a single rule. For an individual rule $S \in M$, we write all instances covered by S as (x^S, y^S), in which y^S can be regarded as a realization of the random vector $Y^S = (Y, ..., Y)$, and Y^S takes values in $\mathscr{Y}^{|S|}$, the $|S|$-ary Cartesian power of $\mathscr{Y}$. Then, the NML distribution for $P_S(Y)$ equals

$$P_S^{NML}(Y^S = y^S | X^S = x^S) = \frac{\hat{P}_S(Y^S = y^S | X^S = x^S)}{\sum_{z^S \in \mathscr{Y}^S} \hat{P}_S(Y^S = z^S | X^S = x^S)}. \tag{7}$$

Note that $\hat{P}_S$ depends on the values of z^S. As $\hat{P}_S(Y)$ is a categorical distribution, the normalizing term can be written as $\mathcal{R}(|S|, |\mathscr{Y}|)$, a function of $|S|$—the rule's coverage—and $|\mathscr{Y}|$—the number of unique values that Y can take [15]:

$$\mathcal{R}(|S|, |\mathscr{Y}|) = \sum_{z^S \in \mathscr{Y}^S} \hat{P}_S(Y^S = z^S | X^S = x^S), \tag{8}$$

which can be efficiently calculated in sub-linear time [15].

The approximate NML distribution. We propose to approximate the normalizing term of P_M^{NML} as the product of the normalizing terms of P_S^{NML} for all $S \in M$, and propose the approximate-NML distribution as our model selection criterion:

$$P_M^{apprNML}(Y^n = y^n | X^n = x^n) = \frac{P_{M,\hat{\theta}(x^n, y^n)}(Y^n = y^n | X^n = x^n)}{\prod_{S \in M} \mathcal{R}(|S|, |\mathscr{Y}|)}. \tag{9}$$

Note that the sum over all $S \in M$ *does* include the "else rule" S_0. Finally, we can formally define the optimal rule set M^* as

$$M^* = \arg\max_M P_M^{apprNML}(Y^n = y^n | X^n = x^n). \tag{10}$$

The rationale of using the approximate-NML distribution is as follows. First, it is equal to the NML distribution for a rule set without any overlap, as follows.

Proposition 1. *Given a rule set M in which for any $S_i, S_j \in M$, $S_i \cap S_j = \emptyset$, then $P_M^{NML}(Y^n = y^n | X^n = x^n) = P_M^{apprNML}(Y^n = y^n | X^n = x^n)$.*

Second, when overlaps exist in M, approximate-NML puts a small extra penalty on overlaps, which is desirable to trade-off overlap with goodness-of-fit: when we sum over all instances in each rule $S \in M$, the instances in overlaps are "repeatedly counted". Third, approximate-NML behaves like the Bayesian information criterion (BIC) asymptotically, which follows from the next proposition.

Proposition 2. *Assume M contains K rules in total, including the else rule, and we have n instances. Then $\log\left(\prod_{S \in M} \mathcal{R}(|S|, |\mathscr{Y}|)\right) = \frac{K(|\mathscr{Y}|-1)}{2} \log n + \mathcal{O}(1)$, where $\mathcal{O}(1)$ is bounded by a constant w.r.t. to n.*

We defer the proofs of the two propositions to the Supplementary Material.

5 Learning Truly Unordered Rule Sets from Data

As our MDL-based model selection criterion unfortunately does not enable efficient search for the optimal model, we resort to heuristics. We first address the challenge of evaluating incomplete rule sets, after which we explain how to grow individual rules in two phases and implement this with beam search. Finally, we show how everything comes together to iteratively learn rule sets from data.

5.1 Evaluating Incomplete Rule Sets with a Surrogate Score

When iteratively searching for the next "best" rule, defining "best" is far from trivial: rule coverage and precision are contradicting factors and typical scores therefore combine those two factors in some—more or less—arbitrary way.

This issue is further aggravated by the iterative rule learning process, in which the intermediate rule set is evaluated as an *incomplete rule set* in each step. Evaluating incomplete rule sets is a challenging task [7], mainly because any good score needs to simultaneously consider two aspects: 1) how well do all the rules currently in the rule set describe the already covered instances; and 2) what is the "potential" for the uncovered instances, in the sense that how well can those uncovered instances be described by rules that might be added later?

Without knowing the rules that will be added later, we cannot compute the NML-based criterion for the complete rule set. Yet, we should take into account the potential of the uncovered instances. We propose to approximate the latter using a *surrogate score*, which we obtain by fitting a decision tree on the uncovered instances and using the leafs of the resulting tree as a surrogate for "future" rules. Formally, we define the tree-based surrogate score as

$$L_{\mathcal{T}}(M) = P_{M \oplus \mathcal{T}}^{apprNML}(Y^n = y^n | X^n = x^n), \tag{11}$$

where $M \oplus \mathcal{T}$ denotes the surrogate rule set obtained by converting the branches of $\mathcal{T}$ to rules and appending those to M (parameters are estimated as usual).

Although the branches of the decision tree learned from the currently uncovered instances may be different from the rules that will later be added to the rule set, using the tree-based surrogate score will make it easier to gradually grow good rule sets. We use decision trees because they are quick to learn and use, and the correspondence of branches to rules makes using them straightforward. We will empirically study the effects of the surrogate score on the predictive performance of rule sets in Section 6.

5.2 Two-phase Rule Growth

To avoid having to traverse all possible rules when searching for the rule to add to an incomplete rule set, we resort to a common heuristic: we start with an empty rule and gradually refine it by adding literals—also referred to as *growing* a rule [8]. In contrast to existing methods, we propose a two-phase method.

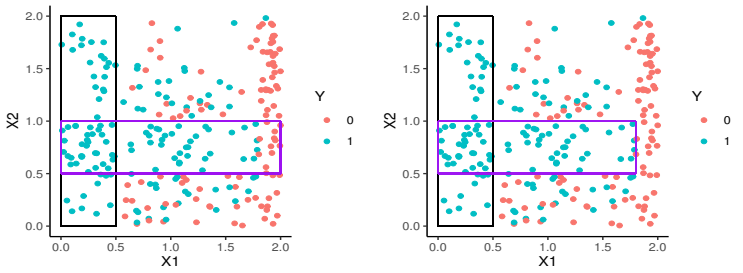

Fig. 1. (Left) Simulated data with two overlapping rules: $S_1 : X_1 < 0.5$ (outlined in black) and $S_2 : 0.5 < X_2 < 1$ (purple). (Right) S_2 has grown to $0.5 < X_2 < 1 \wedge X_1 < 1.8$, which changes $P(Y|X \in S_2)$ and resolves the problematic overlap.

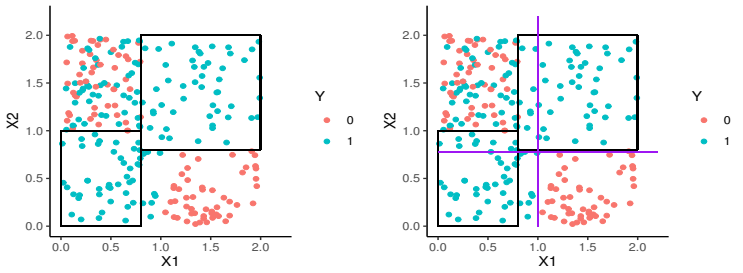

Fig. 2. (Left) Simulated data with a rule set containing two rules (black outlines). (Right) Growing a rule to describe the bottom-right instances will create conflicts with existing rules. I.e., adding either $X_1 > 1$ (vertical purple line) or $X_2 < 0.8$ (horizontal purple line) would create a huge overlap that deteriorates the surrogate score (Eq. 11).

Motivation. A rule can only improve the surrogate score—and thus be added to the rule set—if it achieves two goals: 1) it should improve the likelihood of currently uncovered instances (penalized by the approximate-NML normalizing term); and 2) it should not deteriorate the goodness-of-fit of the rule set by creating "bad" overlaps. These goals can be conflicting though, for two reasons.

First, it is not necessarily bad to have overlaps between a rule being grown and the current rule set, because the rule and its probability estimates for the target variable may still change. For example, consider the left plot of Figure 1. If the current rule set consists of S_1 (indicated in black), then adding S_2 (in purple) would be problematic: this would strongly deteriorate the likelihood of the instances covered by both rules. However, as we further grow S_2, as shown in the right plot, we get $P(Y|S_1) = P(Y|S_2)$ and the problem is solved.

Second, rules already in the rule set may become obstacles to growing a new rule. For example, consider the data and rule set with two rules (in black) in Figure 2. If we want to grow a rule that covers the bottom-right instances, the existing rules form a blockade: the right plot shows how adding either $X_1 > 1$

Algorithm 1: Find Next Rule Ignoring Overlaps

Input: rule set M, data (x^n, y^n)
Output: A beam that contains the w best rules

RULE $\leftarrow \emptyset$; Beam $\leftarrow$ [RULE] `// Initialize the empty rule and beam`
BeamList $\leftarrow$ Beam `// Record all the beams in the beam search`
while length($Beam$) $\neq 0$ **do**
  candidates $\leftarrow$ [ ] `// initialized to store all possible refinements`
  for $RULE \in Beam$ **do**
    Rs $\leftarrow$ [Append L to RULE for L $\in$ all possible literals]
    candidates.extend(Rs)
  Beam $\leftarrow$ the w rules in candidates that have 1) the highest positive $g_{unc}()$, and 2) coverage diversity $> \alpha$ `// w is the beam width`
  if length($Beam$) $\neq 0$ **then**
    BeamList.extend(Beam) `// extend the BeamList as an array`
for $Rule \in BeamList$ **do**
  Beam $\leftarrow$ w rules in BeamList with best $L_{\mathcal{T}}(M \oplus S_{unc})$
return Beam

or $X_2 < 0.8$ to the empty rule (in purple) would create a large overlap with the existing rules, with significantly different probability estimates.

Therefore, instead of navigating towards the two goals simultaneously, we propose to grow the next rule in two phases: 1) grow the rule as if the instances covered by the (incomplete) rule set are excluded; 2) further grow the rule to eliminate potentially "bad" overlaps, to further optimize the tree-based score.

Method. Given a rule S, define S_{unc} as its uncovered "counterpart", which covers all instances in S not covered by M, i.e., $S_{unc} = S \setminus \cup\{S_i \in M\}$. Then, given M, the search for the next best rule that optimizes the surrogate tree-based score is divided into two phases. First, we aim to find the m rules for which the uncovered counterparts have the highest surrogate scores, defined as

$$L_{\mathcal{T}}(M \oplus S_{unc}) = P^{apprNML}_{M \oplus S_{unc} \oplus \mathcal{T}}(Y^n = y^n | X^n = x^n), \tag{12}$$

where $M \oplus S_{unc} \oplus \mathcal{T}$ denotes M appended with S_{unc} and all branches of $\mathcal{T}$. Here, m is a user-specified hyperparameter that controls the number of candidate rules that are selected for further refinement in the second phase. In the second phase, we further grow each of these m rules to search for the best one rule that optimizes

$$L_{\mathcal{T}}(M \oplus S) = P^{apprNML}_{M \oplus S \oplus \mathcal{T}}(Y^n = y^n | X^n = x^n). \tag{13}$$

Given a rule S and its counterpart S_{unc}, the score of S_{unc} is an upper-bound on the score of S: if S can be further refined to cover exactly what S_{unc} covers, we

can obtain $L_{\mathcal{T}}(M \oplus S_{unc}) = L_{\mathcal{T}}(M \oplus S_{unc})$. This is often not possible in practice though, and we therefore generate m candidates in the first phase (instead of 1).

5.3 Beam Search for Two-phase Rule Growth

In both phases we aim for growing a rule that optimizes the tree-based score (Equation 11); the difference is that we ignore the already covered instances in the first phase. To avoid growing rules too greedily, i.e., adding literals that quickly reduce the coverage of the rule, we use a heuristic that is based on the NML distribution of a single rule and motivated by Foil's information gain [4].

Phase 1: rule growth ignoring covered instances. We propose the NML-gain to optimize $L_T(M \oplus S_{unc})$: given two rules S and Q, where we obtain S by adding one literal to Q, we define the NML-gain as $g_{unc}(S, Q)$:

$$g_{unc}(S,Q) = \left(\frac{P^{NML}_{S_{unc}}(y^{S_{unc}}|x^{S_{unc}})}{|S_{unc}|} - \frac{P^{NML}_{Q_{unc}}(y^{Q_{unc}}|x^{Q_{unc}})}{|Q_{unc}|} \right) |S_{unc}| \tag{14}$$

$$= \left(\frac{\hat{P}_{S_{unc}}(y^{S_{unc}}|x^{S_{unc}})}{\mathcal{R}(|S_{unc}|, |\mathscr{Y}|)\ \ |S_{unc}|} - \frac{\hat{P}_{Q_{unc}}(y^{Q_{unc}}|x^{Q_{unc}})}{\mathcal{R}(|Q_{unc}|, |\mathscr{Y}|)\ \ |Q_{unc}|} \right) |S_{unc}|, \tag{15}$$

which we use as the navigation heuristic.

The advantage of having a tree-based score to evaluate rules, besides the navigation heuristic (local score), is that we can adopt beam search, as outlined in Algorithm 1. We start by initializing 1) the rule as an *empty rule* (a rule without any condition), 2) the Beam containing that empty rule, and 3) the BeamRecord to record the rules in the beam search process (Line 1-2). Then, for each rule in the beam, we generate refined candidate rules by adding one literal to it (Ln 5-7). Among all candidates, we select at most w rules with the highest NML-based gain g_{unc}, satisfying two constraints: 1) $g_{unc} > 0$; and 2) for each pair of these (at most) w rules, e.g., S and Q, their "coverage diversity" $\frac{|S_{unc} \cap Q_{unc}|}{|S_{unc} \cup Q_{unc}|} > \alpha$, where α is a user-specified parameter that controls the diversity of the beam search [19]. We update the Beam with these (at most) w rules (Ln 8-10). We repeat the process until we can no longer grow any rule with positive g_{unc} based on all rules in Beam (Ln 3). Last, among the record of all Beams we obtained during the process, we return the best w rules with the highest tree-based score $L(S_{unc} \cup M)$ (Ln 11-13).

Phase 2: rule growth including covered instances. We now optimize $L(M \oplus S)$ and select a rule based on the candidates obtained in the previous step. We first define a navigation heuristic: given two rules S and Q, where S is obtained by adding one literal to Q, we define the NML-gain $g(S, Q)$ as

$$g(S,Q) = \left(\frac{\hat{P}_{S}(y^{S_{unc}}|x^{S_{unc}})}{\mathcal{R}(|S_{unc}|, |\mathscr{Y}|)\ \ |S_{unc}|} - \frac{\hat{P}_{Q}(y^{Q_{unc}}|x^{Q_{unc}})}{\mathcal{R}(|Q_{unc}|, |\mathscr{Y}|)\ \ |Q_{unc}|} \right) |S_{unc}|. \tag{16}$$

Note that the difference between $g(S, Q)$ and $g_{unc}(S, Q)$ is that they use a different maximum likelihood estimator: $\hat{P}_Q$ is the ML estimator based on all instances in Q, while $\hat{P}_{Q_{unc}}$ is based on all instances in Q_{unc}.

Algorithm 2: Find Rule Set

Input: training data (x^n, y^n)
Output: rule set M

1 $M \leftarrow \emptyset$; $M\_record \leftarrow [M]$
2 scores $\leftarrow [P_M^{apprNML}(y^n|x^n)]$ `// Record` $P_M^{apprNML}$ `while growing`
3 **while** *True* **do**
4 $S^* \leftarrow \text{FindNextRule}(M, (x^n, y^n))$ `// find the next best rule` S^*
5 **if** $S^* = \emptyset$ *or* $L_{\mathcal{T}}(M \oplus S) = P_{M \oplus S^*}^{apprNML}(y^n|x^n)$ **then**
6 **Break**
7 **else**
8 $M \leftarrow M \oplus S^*$; $M\_record.append(M)$ `// update and record` M
9 scores.append$(P_M^{apprNML}(y^n|x^n))$
10 **return** the rule set with the maximum score in M_record

The algorithm is almost identical to Algorithm 1, with four small modifications: 1) the navigation heuristic is replaced by $g(S, Q)$; 2) $L_{\mathcal{T}}(M \oplus S)$ is used to select the best rule from the BeamRecord instead of $L_{\mathcal{T}}(M \oplus S_{unc})$; and 3) the coverage diversity is based on the rules itself instead of the counterparts; 4) only the best rule is returned.

5.4 Iterative search for the rule set

Algorithm 2 outlines the proposed rule set learner. We start with an empty rule set (Ln 1-2), then iteratively add the next best rule (Ln 3–9) until the stopping criterion is met (Ln 5–6). That is, it stops when 1) the surrogate score equals the 'real' model selection criterion (i.e., the model's NML distribution), or 2) no more rules with positive NML-gain can be found. We record the 'real' criterion when adding each rule to the set, and pick the one maximizing it (Ln 10).

6 Experiments

We demonstrate that Turs learns rule sets with competitive predictive performance, and that using the surrogate score substantially improves the AUC scores. Further, we demonstrate that Turs achieves model complexities comparable to other rule set methods for multi-class targets.

We here discuss the most important parts of the experiment setup; for completeness, additional information can be found in the Supplementary Material[2].

Decision trees for surrogate score. We use CART [1] to learn the trees for the surrogate score. For efficiency and robustness, we do not use any post-pruning for the decision trees but only set a minimum sample size for the leafs.

[2] The source code is available at https://github.com/ylincen/TURS

Beam width and coverage diversity. We set the coverage diversity $\alpha = 0.05$, and beam width $w = 5$. With the coverage diversity as a constraint, we found that $w \in \{5, 10, 20\}$ all give similar results. Due to the limited space, we leave a formal sensitivity analysis of α as future work.

Benchmark datasets and competitor algorithms. We test on 13 UCI benchmark datasets (shown in Table 1), and compare against the following methods: 1) unordered CN2 [2], the one-versus-rest rule sets method without implicit order among rules; 2) DRS [24], a representative multi-class rule set learning method; 3) BRS [21], the Bayesian rule set method for binary classification; 4) RIPPER [4], the widely used one-versus-rest method with orders among class labels; 5) CLASSY [17], the probabilistic rule list methods using MDL-based model selection; and 6) CART [1], the well-known decision tree method, *with* post-pruning by cross-validation.

Table 1. ROC-AUC scores, averaged over 10 cross-validated folds. The rank (smaller means better) is further averaged over all datasets. Among the four *rule set* methods, TURS is substantially better on 7 out 13 datasets (AUC scores in bold).

data	TURS	CN2	DRS	BRS	CLASSY	RIPPER	CART	TURS %overlap
anuran	0.998	1.000	0.858	—	0.983	0.999	0.996	0.395
avila	0.968	0.978	0.530	—	0.954	0.997	0.988	0.286
backnote	**0.991**	0.969	0.945	0.957	0.987	0.979	0.984	0.297
car	**0.978**	0.633	0.924	—	0.945	0.980	0.971	0.063
chess	**0.995**	0.536	0.823	0.945	0.991	0.995	0.994	0.264
contracept	**0.667**	0.597	0.544	—	0.630	0.626	0.600	0.074
diabetes	**0.766**	0.677	0.628	0.683	0.761	0.735	0.661	0.155
ionosphere	0.914	0.912	0.663	0.837	0.909	0.901	0.845	0.310
iris	0.964	0.985	0.935	—	0.960	0.973	0.965	0.018
magic	**0.886**	0.590	0.695	0.794	0.895	0.818	0.800	0.500
tic-tac-toe	0.972	0.826	0.971	0.976	0.983	0.954	0.847	0.231
waveform	**0.902**	0.775	0.588	—	0.833	0.884	0.803	0.528
wine	0.954	0.962	0.810	—	0.961	0.945	0.932	0.031
Avg Rank	2.231	4.077	5.846	5.462	3.154	3.000	4.231	/

6.1 Results

Predictive performance. We report the ROC-AUC scores in Table 1. For multi-class classification, we report the weighted one-versus-rest AUC scores, as was also used for evaluating the recently proposed CLASSY method [17].

Compared to non-probabilistic rule set methods—i.e., CN2, DRS, and BRS (only for binary targets)—TURS is much better in terms of the mean rank of its AUC scores. Specifically, it performs substantially better on about half of the datasets (shown in bold). Besides, it is ranked better than rule list methods, which produce explicitly ordered rules that may be difficult for domain experts to comprehend and digest in practice. Next, CART attains AUCs generally inferior

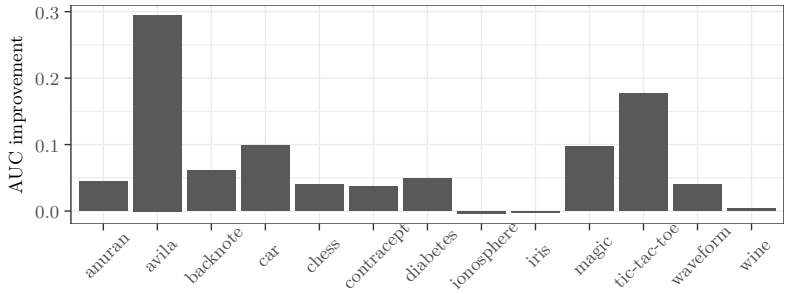

Fig. 3. Improvement in AUC by enabling the surrogate score for Turs.

to Turs, although it helps Turs to get a higher AUC as part of the surrogate score.

Last, we report the percentage of instances covered by more than one rule for Turs in Table 1, and we show that overlaps are common in the rule sets obtained for different datasets. This empirically confirms that our way of formalizing rule sets as probabilistic models, i.e., treating overlaps as uncertainty and exception, can indeed lead to improved predictive performance, as the overlapping rules are a non-negligible part of the model learned from data and hence indeed play a role.

Effects of the surrogate score. Figure 3 shows the difference in AUC obtained by our method with and without using the surrogate score (i.e., without surrogate score means replacing it with the final model selection criterion). We conclude that the surrogate score has a substantial effect on learning better rule sets, except for three "simple" datasets, of which the sample sizes and the number of variables are small, as shown in Table 2 (Left).

Model complexity. Finally, we compare the 'model complexity' of the rule sets for all methods. As this is hard to quantify in a unified manner, as a proxy we report the *total number of literals in all rules in a rule set*, averaged over 10-fold cross-validation (the same as used for the results reported in Table 1).

We show that among all rule set methods (TURS, CN2, DRS, BRS), Turs has better average ranks than both CN2 and DRS. Although BRS learns very small rule sets, it is only applicable to binary targets and its low model complexity also brings worse AUC scores than Turs. Further, although rule list methods (CLASSY, RIPPER) generally have fewer literals than rule sets methods, this does not make rule lists easy to interpret, as every rule depends on all previous rules. Last, we empirically confirm that tree-based method CART produces much larger rule sets.

7 Conclusion

We formalized the problem of learning truly unordered probabilistic rule sets as a model selection task. We also proposed a novel, tree-based surrogate score for

Table 2. Left: The sample sizes and number of features of datasets. Right: total number of literals, i.e., average rule lengths × number of rules in the set, averaged over 10-fold cross-validation. The rank is averaged over all datasets, for rule sets methods only.

#instances	#features	data	TURS	CN2	DRS	BRS	CLASSY	RIPPER	CART
1372	5	backnote	42	41	55	22	22	16	94
1473	10	contracept	75	275	73	—	14	14	6241
768	9	diabetes	55	152	131	10	10	6	827
150	5	iris	7	9	23	—	3	3	9
958	10	tic-tac-toe	86	90	108	24	27	60	816
178	14	wine	10	6	134	—	6	5	15
1728	7	car	211	163	325	—	92	111	718
7195	24	anuran	74	37	407	—	49	7	96
3196	37	chess	299	316	482	21	37	44	355
351	35	ionosphere	50	30	261	14	6	5	101
5000	22	waveform	707	802	60	—	139	115	3928
20867	11	avila	890	1296	179	—	988	574	8145
19020	11	magic	1321	2238	48	23	256	69	22566
		Avg Rank	2.15	2.46	2.77	1.00	—	—	—

evaluating incomplete rule sets. Building upon this, we developed a two-phase heuristic algorithm that learns rule set models that were empirically shown to be accurate in comparison to competing methods.

For future work, we will study the practical use of our method with a case study in the health care domain. This involves investigating how well our method scales to larger datasets. Furthermore, a user study will be performed to investigate whether, and in what degree, the domain experts find the truly unordered property of rule sets obtained by our method helps them comprehend the rules better in practice, in comparison to rule lists/sets with explicit or implicit orders.

Acknowledgements. We are grateful for the very inspiring feedback from the anonymous reviewers. This work is part of the research programme 'Human-Guided Data Science by Interactive Model Selection' with project number 612.001.804, which is (partly) financed by the Dutch Research Council (NWO).

References

1. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and regression trees. CRC press (1984)
2. Clark, P., Boswell, R.: Rule induction with cn2: Some recent improvements. In: European Working Session on Learning. pp. 151–163. Springer (1991)
3. Clark, P., Niblett, T.: The cn2 induction algorithm. Machine learning **3**(4), 261–283 (1989)
4. Cohen, W.W.: Fast effective rule induction. In: Machine learning proceedings 1995, pp. 115–123. Elsevier (1995)
5. Dash, S., Gunluk, O., Wei, D.: Boolean decision rules via column generation. Advances in Neural Information Processing Systems **31**, 4655–4665 (2018)

6. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization (1998)
7. Fürnkranz, J., Flach, P.A.: Roc 'n'rule learning—towards a better understanding of covering algorithms. Machine learning **58**(1), 39–77 (2005)
8. Fürnkranz, J., Gamberger, D., Lavrač, N.: Foundations of rule learning. Springer Science & Business Media (2012)
9. Gay, D., Boullé, M.: A bayesian approach for classification rule mining in quantitative databases. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 243–259. Springer (2012)
10. Grünwald, P., Roos, T.: Minimum description length revisited. International journal of mathematics for industry **11**(01), 1930001 (2019)
11. Hühn, J., Hüllermeier, E.: Furia: an algorithm for unordered fuzzy rule induction. Data Mining and Knowledge Discovery **19**(3), 293–319 (2009)
12. Lakkaraju, H., Bach, S.H., Leskovec, J.: Interpretable decision sets: A joint framework for description and prediction. In: Proceedings of the 22nd ACM SIGKDD. pp. 1675–1684 (2016)
13. Liu, B., Hsu, W., Ma, Y., et al.: Integrating classification and association rule mining. In: KDD. vol. 98, pp. 80–86 (1998)
14. Molnar, C.: Interpretable machine learning. Lulu. com (2020)
15. Mononen, T., Myllymäki, P.: Computing the multinomial stochastic complexity in sub-linear time. In: PGM08. pp. 209–216 (2008)
16. Murdoch, W.J., Singh, C., Kumbier, K., Abbasi-Asl, R., Yu, B.: Interpretable machine learning: definitions, methods, and applications. arXiv preprint arXiv:1901.04592 (2019)
17. Proença, H.M., van Leeuwen, M.: Interpretable multiclass classification by mdl-based rule lists. Information Sciences **512**, 1372–1393 (2020)
18. Quinlan, J.R.: C4. 5: programs for machine learning. Elsevier (2014)
19. Van Leeuwen, M., Knobbe, A.: Diverse subgroup set discovery. Data Mining and Knowledge Discovery **25**(2), 208–242 (2012)
20. Veloso, A., Meira, W., Zaki, M.J.: Lazy associative classification. In: Sixth International Conference on Data Mining (ICDM'06). pp. 645–654. IEEE (2006)
21. Wang, T., Rudin, C., Doshi-Velez, F., Liu, Y., Klampfl, E., MacNeille, P.: A bayesian framework for learning rule sets for interpretable classification. The Journal of Machine Learning Research **18**(1) (2017)
22. Yang, F., He, K., Yang, L., Du, H., Yang, J., Yang, B., Sun, L.: Learning interpretable decision rule sets: A submodular optimization approach. Advances in Neural Information Processing Systems **34** (2021)
23. Yang, H., Rudin, C., Seltzer, M.: Scalable bayesian rule lists. In: International Conference on Machine Learning. pp. 3921–3930. PMLR (2017)
24. Zhang, G., Gionis, A.: Diverse rule sets. In: Proceedings of the 26th ACM SIGKDD. pp. 1532–1541 (2020)