An Optimization Approach to Learning Falling Rule Lists

Chaofan Chen
Duke University

Cynthia Rudin
Duke University

Abstract

A falling rule list is a probabilistic decision list for binary classification, consisting of a series of **if-then** rules with antecedents in the **if** clauses and probabilities of the desired outcome ("1") in the **then** clauses. Just as in a regular decision list, the order of rules in a falling rule list is important – each example is classified by the first rule whose antecedent it satisfies. Unlike a regular decision list, a falling rule list requires the probabilities of the desired outcome ("1") to be monotonically decreasing down the list. We propose an optimization approach to learning falling rule lists and "softly" falling rule lists, along with Monte-Carlo search algorithms that use bounds on the optimal solution to prune the search space.

1 INTRODUCTION

In many real-life scenarios, we want to learn a predictive model that allows us to easily identify the most significant conditions that are predictive of a certain outcome. For example, in health care, doctors often want to know the conditions that signify a high risk of stroke, so that patients with such conditions can be prioritized in receiving treatment. A falling rule list, whose form was first proposed by Wang and Rudin (2015), is a type of model that serves this purpose.

Table 1 shows a falling rule list we learned from the bank-full dataset, which was used by Moro et al. (2011) in their study of applying data mining techniques to direct marketing. As we can see, a falling rule list is a probabilistic decision list for binary classification, consisting of a series of **if-then** rules with antecedents in the **if** clauses and probabilities of the desired outcome ("1") in the **then** clauses, where the probabilities of the desired outcome ("1") are monotonically decreasing down the list (hence the name "falling" rule list). The falling rule list in Table 1 has identified clients for whom the previous marketing campaign was successful ("poutcome=success"), and who have no credit in default ("default=no"), as individuals who are most likely to subscribe to a term deposit in the current marketing campaign. Their probability of subscribing is 0.65. Of the remaining clients, those who are next most likely to sign up for a term deposit are older people (aged between 60 and 100) with no credit in default. Their probability of subscribing is 0.28. The two rightmost columns in Table 1, labeled + and −, show the number of positive training examples (i.e. clients who subscribe to a term deposit in the current campaign) and of negative training examples, respectively, that satisfy the antecedent in each rule of the falling rule list.

Falling rule lists can provide valuable insight into data – if we know how to construct them well. In this paper, we propose an optimization approach to learning falling rule lists and "softly" falling rule lists, along with Monte-Carlo search algorithms that use bounds on the optimal solution to prune the search space. The falling rule list shown in Table 1 was produced using Algorithm FRL, which we shall introduce later.

Our work lives within several well-established fields, but is the first work we know of to use an optimization approach to handling monotonicity constraints in rule-based models. It relates closely to associative classification (e.g. the RIPPERk algorithm (Cohen, 1995) and the CBA algorithm (Liu et al., 1998); see Thabtah (2007) for a comprehensive review) and inductive logic programming (Muggleton and De Raedt, 1994). The proposed algorithms are competitors for decision tree methods like CART (Breiman et al., 1984), ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993),

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

Table 1: Falling Rule List for bank-full Dataset

	antecedent		prob.	+	−
IF	poutcome=success AND default=no	THEN success prob. is	0.65	978	531
ELSE IF	$60 \leq$ age < 100 AND default=no	THEN success prob. is	0.28	434	1113
ELSE IF	$17 \leq$ age < 30 AND housing=no	THEN success prob. is	0.25	504	1539
ELSE IF	previous ≥ 2 AND housing=no	THEN success prob. is	0.23	242	794
ELSE IF	campaign=1 AND housing=no	THEN success prob. is	0.14	658	4092
ELSE IF	previous ≥ 2 AND education=tertiary	THEN success prob. is	0.13	108	707
ELSE		success prob. is	0.07	2365	31146

and C5.0 (Quinlan, 2004), and decision list learning (Rivest, 1987). Almost all methods from this class build decision trees from the top down using greedy splitting criteria. Greedy splitting criteria do not lend naturally to constrained models like falling rule lists. There are some works on decision trees with monotonicity constraints (e.g. Altendorf et al., 2005; Ben-David, 1995; Feelders and Pardoel, 2003), but they focus mostly on enforcing the monotonic relationship between certain attributes and ordinal class labels. In addition, our work also relates to those that underline the importance of the interpretability of models (Freitas, 2014; Huysmans et al., 2011; Kodratoff, 1994; Martens and Baesens, 2010).

Wang and Rudin (2015) proposed the form of a falling rule list, and a Bayesian approach to learning falling rule lists (extending the ideas of Letham et al. (2015) and Yang et al. (2017)). The Bayesian approach offers some advantages: e.g. a full posterior over rule lists allows model averaging. However, the optimization perspective has an important computational advantage: the search space is made substantially smaller by the tight bounds presented here. The concept of softly falling rule lists is novel to this paper and has not been done in the Bayesian setting.

2 PROBLEM FORMULATION

We first formalize the notion of an antecedent, of a rule list, of a falling rule list, and of a prefix.

Definition 2.1. An *antecedent* a on an input domain $\mathcal{X}$ is a Boolean function that outputs true or false. Given an input $\mathbf{x} \in \mathcal{X}$, we say that $\mathbf{x}$ satisfies the antecedent a if $a(\mathbf{x})$ evaluates to true. For example, (poutcome=success AND default=no) in Table 1 is an antecedent.

Definition 2.2. A *rule list* $d : \mathcal{X} \to [0, 1]$ on an input domain $\mathcal{X}$ is a probabilistic decision list of the following form: "if $\mathbf{x}$ satisfies $a_0^{(d)}$, then $\Pr(y = 1|\mathbf{x}) = \hat{\alpha}_0^{(d)}$; else if $\mathbf{x}$ satisfies $a_1^{(d)}$, then $\Pr(y = 1|\mathbf{x}) = \hat{\alpha}_1^{(d)}$; ...; else if $\mathbf{x}$ satisfies $a_{|d|-1}^{(d)}$, then $\Pr(y = 1|\mathbf{x}) = \hat{\alpha}_{|d|-1}^{(d)}$; else $\Pr(y = 1|\mathbf{x}) = \hat{\alpha}_{|d|}^{(d)}$" where $a_j^{(d)}$ is the j-th antecedent in d, $j \in \{0, 1, ..., |d| - 1\}$, and $|d|$ denotes the size of the rule list, which is defined as the number of rules, excluding the final else clause, in the rule list. We can denote the rule list d as follows:

$$d = \{(a_0^{(d)}, \hat{\alpha}_0^{(d)}), (a_1^{(d)}, \hat{\alpha}_1^{(d)}), ..., (a_{|d|-1}^{(d)}, \hat{\alpha}_{|d|-1}^{(d)}), \hat{\alpha}_{|d|}^{(d)}\}. \tag{1}$$

The rule list d of Equation (1) is a *falling rule list* if the following inequalities hold:

$$\hat{\alpha}_0^{(d)} \geq \hat{\alpha}_1^{(d)} \geq ... \geq \hat{\alpha}_{|d|-1}^{(d)} \geq \hat{\alpha}_{|d|}^{(d)}. \tag{2}$$

For convenience, we sometimes refer to the final else clause in d as the $|d|$-th antecedent $a_{|d|}^{(d)}$ in d, which is satisfied by all $\mathbf{x} \in \mathcal{X}$. We denote the space of all possible rule lists on $\mathcal{X}$ by $\mathcal{D}(\mathcal{X})$.

Definition 2.3. A *prefix* e on an input domain $\mathcal{X}$ is a rule list without the final else clause. We can denote the prefix e as follows:

$$e = \{(a_0^{(e)}, \hat{\alpha}_0^{(e)}), (a_1^{(e)}, \hat{\alpha}_1^{(e)}), ..., (a_{|e|-1}^{(e)}, \hat{\alpha}_{|e|-1}^{(e)})\}. \tag{3}$$

where $a_j^{(e)}$ is the j-th antecedent in e, $j \in \{0, 1, ..., |e| - 1\}$, and $|e|$ denotes the size of the prefix, which is defined as the number of rules in the prefix.

Definition 2.4. Given the rule list d of Equation (1) (or the prefix e of Equation (3)), we say that an input $\mathbf{x} \in \mathcal{X}$ is *captured* by the j-th antecedent in d (or e) if $\mathbf{x}$ satisfies $a_j^{(d)}$ (or $a_j^{(e)}$, respectively), and for all $k \in \{0, 1, ..., |d|\}$ (or $k \in \{0, 1, ..., |e| - 1\}$, respectively) such that $\mathbf{x}$ satisfies $a_k^{(d)}$ (or $a_k^{(e)}$, respectively), $j \leq k$ holds – in other words, $a_j^{(d)}$ (or $a_j^{(e)}$, respectively) is the first antecedent that $\mathbf{x}$ satisfies. We define the function capt by $\text{capt}(\mathbf{x}, d) = j$ (or $\text{capt}(\mathbf{x}, e) = j$) if $\mathbf{x}$ is captured by the j-th antecedent in d (or e). Moreover, given the prefix e of Equation (3), we say that an input $\mathbf{x} \in \mathcal{X}$ is captured by the prefix e if $\mathbf{x}$ is captured by some antecedent in e, and we define $\text{capt}(\mathbf{x}, e) = |e|$ if $\mathbf{x}$ is not captured by the prefix e.

Let $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be the training data, with $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \{1, -1\}$ for each $i \in \{1, 2, ..., n\}$. We now define the empirical positive proportion of an antecedent, and introduce the notion of a rule list (or a prefix) that is compatible with D.

Definition 2.5. Given the training data D and the rule list d of Equation (1) (or the prefix e of Equation (3)), we denote by $n^+_{j,d,D}$, $n^-_{j,d,D}$, $n_{j,d,D}$ (or $n^+_{j,e,D}$, $n^-_{j,e,D}$, $n_{j,e,D}$), the number of positive, negative, and all training inputs captured by the j-th antecedent in d (or e), respectively, and define the *empirical positive proportion* of the j-th antecedent in d (or e), denoted by $\alpha_j^{(d,D)}$ (or $\alpha_j^{(e,D)}$), as:

$$\alpha_j^{(d,D)} = n^+_{j,d,D}/n_{j,d,D} \text{ (or } \alpha_j^{(e,D)} = n^+_{j,e,D}/n_{j,e,D}).$$

Moreover, given the training data D and the prefix e of Equation (3), we denote by $\tilde{n}^+_{e,D}$, $\tilde{n}^-_{e,D}$, $\tilde{n}_{e,D}$, the number of positive, negative, and all training inputs that are not captured by the prefix e, and define the *empirical positive proportion after the prefix e*, denoted by $\tilde{\alpha}_{e,D}$, as $\tilde{\alpha}_{e,D} = \tilde{n}^+_{e,D}/\tilde{n}_{e,D}$.

Definition 2.6. Given the training data D and the rule list d of Equation (1) (or the prefix e of Equation (3)), we say that the rule list d (or the prefix e) is *compatible* with D if for all $j \in \{0, 1, ..., |d|\}$ (or $j \in \{0, 1, ..., |e| - 1\}$, respectively), the equation $\hat{\alpha}_j^{(d)} = \alpha_j^{(d,D)}$ ($\hat{\alpha}_j^{(e)} = \alpha_j^{(e,D)}$, respectively) holds. We denote the space of all possible rule lists on $\mathcal{X}$ that are compatible with the training data D by $\mathcal{D}(\mathcal{X}, D)$.

To formulate the problem of learning falling rule lists from data as an optimization program, we first observe that, given a threshold τ, the rule list d of Equation (1) can be viewed as a classifier $\tilde{d}_\tau : \mathcal{X} \to \{1, -1\}$ that predicts 1 for an input $\mathbf{x} \in \mathcal{X}$ only if the inequality $\hat{\alpha}^{(d)}_{\text{capt}(\mathbf{x},d)} > \tau$ holds. Hence, we can define the empirical risk of misclassification by the rule list d on the training data D as that by the classifier $\tilde{d}_\tau$. More formally, we have the following definition.

Definition 2.7. Given the training data D, the rule list d of Equation (1), a threshold τ, and the weight w for the positive class, the *empirical risk of misclassification by the rule list d* on the training data D with threshold τ and with weight w for the positive class, denoted by $R(d, D, \tau, w)$, is:

$$R(d, D, \tau, w) = \frac{1}{n}\left(w \sum_{i:y_i=1} \mathbb{1}[\hat{\alpha}^{(d)}_{\text{capt}(\mathbf{x}_i,d)} \leq \tau] + \sum_{i:y_i=-1} \mathbb{1}[\hat{\alpha}^{(d)}_{\text{capt}(\mathbf{x}_i,d)} > \tau]\right). \quad (4)$$

If d is compatible with D, we can replace $\hat{\alpha}^{(d)}_{\text{capt}(\mathbf{x}_i,d)}$ in Equation (4) with $\alpha^{(d,D)}_{\text{capt}(\mathbf{x}_i,d)}$. We define the *empirical risk of misclassification by the prefix e* on the training data D with threshold τ and with weight w for the positive class, denoted by $R(e, D, \tau, w)$, analogously:

$$R(e, D, \tau, w) = \frac{1}{n}\left(w \sum_{\substack{i:y_i=1\wedge \\ \text{capt}(\mathbf{x}_i,e)\neq|e|}} \mathbb{1}[\hat{\alpha}^{(e)}_{\text{capt}(\mathbf{x}_i,e)} \leq \tau] + \sum_{\substack{i:y_i=-1\wedge \\ \text{capt}(\mathbf{x}_i,e)\neq|e|}} \mathbb{1}[\hat{\alpha}^{(e)}_{\text{capt}(\mathbf{x}_i,e)} > \tau]\right). \quad (5)$$

If e is compatible with D, we can replace $\hat{\alpha}^{(e)}_{\text{capt}(\mathbf{x}_i,e)}$ in Equation (5) with $\alpha^{(e,D)}_{\text{capt}(\mathbf{x}_i,e)}$. Note that for any rule list d that begins with a given prefix e, $R(e, D, \tau, w)$ is the contribution by the prefix e to $R(d, D, \tau, w)$.

We can formulate the problem of learning falling rule lists as a minimization program of the empirical risk of misclassification, given by Equation (4), with a regularization term $C|d|$ that penalizes each rule in d with a cost of C to limit the number of rules, subject to the monotonicity constraint (2). For now, we focus on the problem of learning falling rule lists that are compatible with the training data D.

Let $L(d, D, \tau, w, C) = R(d, D, \tau, w) + C|d|$ and $L(e, D, \tau, w, C) = R(e, D, \tau, w) + C|e|$ be the regularized empirical risk of misclassification by the rule list d and by the prefix e, respectively, on the training data D. The former defines the objective of the minimization program, and the latter gives the contribution by the prefix e to $L(d, D, \tau, w, C)$ for any rule list d that begins with e. The following theorem provides a motivation for setting the threshold τ to $1/(1 + w)$ in the minimization program – the empirical risk of misclassification by a given rule list d is minimized when τ is set in this way.

Theorem 2.8. *Given the training data D, a rule list d that is compatible with D, and the weight w for the positive class, we have $R(d, D, 1/(1 + w), w) \leq R(d, D, \tau, w)$ for all $\tau \geq 0$.*

For reasons of computational tractability and model interpretability, we further restrict our attention to learning compatible falling rule lists whose antecedents must come from a pre-determined set of antecedents $A = \{A_l\}_{l=1}^m$. We now present the optimization program for learning falling rule lists, which forms the basis of the rest of this paper.

Program 2.9 (Learning compatible falling rule lists)**.**

$$\min_{d \in \mathcal{D}(\mathcal{X}, D)} L(d, D, 1/(1+w), w, C) \text{ subject to}$$

$$\alpha_0^{(d,D)} \geq \alpha_1^{(d,D)} \geq ... \geq \alpha_{|d|-1}^{(d,D)} \geq \alpha_{|d|}^{(d,D)}, \qquad (6)$$

$$a_j^{(d)} \in A, \text{ for all } j \in \{0, 1, ..., |d|-1\}. \qquad (7)$$

The constraint (6) is exactly the monotonicity constraint (2) for the falling rule lists that are compatible with D. The constraint (7) limits the choice of antecedents. An instance of Program 2.9 is defined by the tuple (D, A, w, C).

3 ALGORITHM

In this section, we outline a Monte-Carlo search algorithm, Algorithm FRL, based on Program 2.9, for learning compatible falling rule lists from data. Given an instance (D, A, w, C) of Program 2.9, the algorithm constructs a compatible falling rule list d in each iteration, while keeping track of the falling rule list d^* that has the smallest objective value $L_{\text{best}} = L(d^*, D, \tau, w, C)$ among all the falling rule lists that the algorithm has constructed so far. At the end of T iterations, the algorithm outputs the falling rule list that has the smallest objective value out of the T lists it has constructed.

In the process of constructing a falling rule list d, the algorithm chooses the antecedents successively, and uses various properties of Program 2.9, presented in Section 4, to prune the search space. In particular, when the algorithm is choosing the p-th antecedent in d, it considers only those antecedents $A_l \in A$ satisfying the following conditions: (1) the inclusion of A_l as the p-th antecedent in d gives rise to a rule $(a_p^{(d)}, \alpha_p^{(d,D)})$ that respects the monotonicity constraint $\alpha_p^{(d,D)} \leq \alpha_{p-1}^{(d,D)}$ and the necessary condition for optimality $\alpha_p^{(d,D)} > 1/(1+w)$ (Corollary 4.5), and (2) the inclusion of A_l as the p-th antecedent in d gives rise to a prefix e' such that e' is feasible for Program 2.9 under the training data D (Proposition 4.2), and the best possible objective value $L^*(e', D, w, C)$ achievable by any falling rule list that begins with e' and is compatible with D (Theorem 4.6) is less than the current best objective value L_{best}. The algorithm terminates the construction of d if Inequality (9) in Theorem 4.6 holds. The details of the algorithm can be found in the supplementary material.

4 PREFIX BOUND

The goal of this section is to find a lower bound on the objective value of any compatible falling rule list that begins with a given compatible prefix, which we call a *prefix bound*, and to prove the various results used in the algorithm. To derive this prefix bound, we first introduce the concept of a feasible prefix, with which it is possible to construct a compatible falling rule list from data.

Definition 4.1. Given the training data D and the set of antecedents A, a prefix e is feasible for Program 2.9 under the training data D and the set of antecedents A if e is compatible with D, and there exists a falling rule list d such that d is compatible with D, the antecedents of d come from A, and d begins with e.

The following proposition gives necessary and sufficient conditions for a prefix e to be feasible.

Proposition 4.2. *Given the training data D, the set of antecedents A, and a prefix e that is compatible with D and satisfies $a_j^{(e)} \in A$ for all $j \in \{0, 1, ..., |e|-1\}$ and $\alpha_{k-1}^{(e,D)} \geq \alpha_k^{(e,D)}$ for all $k \in \{1, 2, ..., |e|-1\}$, the following statements are equivalent: (1) e is feasible for Program 2.9 under D and A; (2) $\tilde{\alpha}_{e,D} \leq \alpha_{|e|-1}^{(e,D)}$ holds; (3) $\tilde{n}_{e,D}^- \geq ((1/\alpha_{|e|-1}^{(e,D)}) - 1)\tilde{n}_{e,D}^+$ holds.*

We now introduce the concept of a hypothetical rule list, whose antecedents do not need to come from the pre-determined set of antecedents A.

Definition 4.3. Given a pre-determined set of antecedents A, a hypothetical rule list with respect to A is a rule list that contains an antecedent that is not in A.

We need the following lemma to prove the necessary condition for optimality (Corollary 4.5), and to derive a prefix bound (Theorem 4.6).

Lemma 4.4. *Suppose that we are given an instance (D, A, w, C) of Program 2.9, a prefix e that is feasible for Program 2.9 under D and A, and a (possibly hypothetical) falling rule list d that begins with e and is compatible with D. Then there exists a falling rule list d', possibly hypothetical with respect to A, such that d' begins with e, has at most one more rule (excluding the final else clause) following e, is compatible with D, and satisfies*

$$L(d', D, 1/(1+w), w, C) \leq L(d, D, 1/(1+w), w, C).$$

As a special case, if either $\alpha_j^{(d,D)} > 1/(1+w)$ holds for all $j \in \{|e|, |e|+1, ..., |d|\}$, or $\alpha_j^{(d,D)} \leq 1/(1+w)$

holds for all $j \in \{|e|, |e|+1, ..., |d|\}$, then the falling rule list $\bar{e} = \{e, \tilde{\alpha}_{e,D}\}$ (i.e. the falling rule list in which the final else clause follows immediately the prefix e, and the probability estimate of the final else clause is $\tilde{\alpha}_{e,D}$) is compatible with D and satisfies $L(\bar{e}, D, 1/(1+w), w, C) \le L(d, D, 1/(1+w), w, C)$.

A consequence of the above lemma is that an optimal solution for a given instance (D, A, w, C) of Program 2.9 should not have any antecedent whose empirical positive proportion falls below $1/(1+w)$.

Corollary 4.5. *If d^* is an optimal solution for a given instance (D, A, w, C) of Program 2.9, then we must have $\alpha_j^{(d^*,D)} > 1/(1+w)$ for all $j \in \{0, 1, ..., |d^*|-1\}$.*

Another implication of Lemma 4.4 is that the objective value of any compatible falling rule list that begins with a given prefix e cannot be less than a lower bound on the objective value of any compatible falling rule list that begins with the same prefix e, and has at most one more rule (excluding the final else clause) following e. This leads to the following theorem.

Theorem 4.6. *Suppose that we are given an instance (D, A, w, C) of Program 2.9 and a prefix e that is feasible for Program 2.9 under D and A. Then any falling rule list d that begins with e and is compatible with D satisfies*

$$L(d, D, 1/(1+w), w, C) \ge L^*(e, D, w, C),$$

where

$$L^*(e, D, w, C) = L(e, D, 1/(1+w), w, C) + \min\left(\frac{1}{n}\left(\frac{1}{\alpha_{|e|-1}^{(e,D)}} - 1\right)\tilde{n}_{e,D}^{+} + C, \frac{w}{n}\tilde{n}_{e,D}^{+}, \frac{1}{n}\tilde{n}_{e,D}^{-}\right) \tag{8}$$

is a lower bound on the objective value of any compatible falling rule list that begins with e, under the instance (D, A, w, C) of Program 2.9. We call $L^*(e, D, w, C)$ the prefix bound for e. Further, if

$$C \ge \min\left(\frac{w}{n}\tilde{n}_{e,D}^{+}, \frac{1}{n}\tilde{n}_{e,D}^{-}\right) - \frac{1}{n}\left(\frac{1}{\alpha_{|e|-1}^{(e,D)}} - 1\right)\tilde{n}_{e,D}^{+} \tag{9}$$

holds, then the compatible falling rule list $\bar{e} = \{e, \tilde{\alpha}_{e,D}\}$, where the prefix e is followed directly by the final else clause, satisfies $L(\bar{e}, D, 1/(1+w), w, C) = L^*(e, D, w, C)$.

The results presented in this section are used in Algorithm FRL to prune the search space. The proofs can be found in the supplementary material.

5 SOFTLY FALLING RULE LISTS

Program 2.9 and Algorithm FRL have some limitations. Let us consider a toy example, where we have a training set D of 19 instances, with 14 positive and 5 negative instances. Suppose that we have an antecedent A_1 that is satisfied by 8 positive and 3 negative training instances. If A_1 were to be the first rule of a falling rule list d that is compatible with D, we would obtain a prefix $e = \{(A_1, 8/11)\}$. However, the empirical positive proportion after the prefix e is $\tilde{\alpha}_{e,D} = 6/8 > 8/11$. This violates (2) in Proposition 4.2, so e is not a feasible prefix for Program 2.9 under the training data D. In fact, if every antecedent in A is satisfied by 8 positive and 3 negative instances in the training set D, then the only possible compatible falling rule list we can learn using Algorithm FRL is the trivial falling rule list, which has only the final else clause. At the same time, if we consider the rule list $d = \{(A_1, 8/11), 6/8\}$, which is compatible with the given toy dataset D but is not a falling rule list, we may notice that the two probability estimates in d are quite close to each other – it is very likely that the difference between them is due to sampling variability in the dataset itself.

The two limitations of Program 2.9 and Algorithm FRL – the potential non-existence of a feasible non-trivial solution and the rigidness of using empirical positive proportions as probability estimates – motivate us to formulate a new optimization program for learning "softly" falling rule lists, where we remove the monotonicity constraint and instead introduce a penalty term in the objective function that penalizes violations of the monotonicity constraint (6) in Program 2.9. More formally, define a softly falling rule list as a rule list of Equation (1) with $\hat{\alpha}_j^{(d)} = \min_{k \le j} \alpha_k^{(d,D)}$. Note that any rule list d that is compatible with the given training data D can be turned into a softly falling rule list by setting $\hat{\alpha}_j^{(d)} = \min_{k \le j} \alpha_k^{(d,D)}$. Hence, we can learn a softly falling rule list by first learning a compatible rule list with the "softly falling objective" (denoted by $\tilde{L}$ below), and then transforming the rule list into a softly falling rule list. Let

$$\begin{aligned}&\tilde{L}(d, D, \tau, w, C, C_1)\\ &= L(d, D, \tau, w, C) + C_1 \sum_{j=0}^{|d|} \lfloor \alpha_j^{(d,D)} - \min_{k<j} \alpha_k^{(d,D)} \rfloor_+\end{aligned}$$

and $\tilde{L}(e, D, \tau, w, C, C_1)$

$$= L(e, D, \tau, w, C) + C_1 \sum_{j=0}^{|e|-1} \lfloor \alpha_j^{(e,D)} - \min_{k<j} \alpha_k^{(e,D)} \rfloor_+$$

be the regularized empirical risk of misclassification by a rule list d and by a prefix e, respectively, with a penalty term that penalizes violations of monotonicity in the empirical positive proportions of the antecedents in d and in e, respectively. We call $\tilde{L}$ the softly falling objective function, set the threshold $\tau = 1/(1+w)$ as before, and obtain the following optimization program:

Program 5.1 (Learning compatible rule lists with the softly falling objective)**.**

$$\min_{d \in \mathcal{D}(\mathcal{X}, D)} \tilde{L}(d, D, 1/(1+w), w, C, C_1)$$
$$\text{subject to} \quad a_j^{(d)} \in A, \ \text{for all } j \in \{0, 1, ..., |d|-1\}.$$

An instance of Program 5.1 is defined by the tuple (D, A, w, C, C_1). Similarly, we have a Monte-Carlo search algorithm, Algorithm softFRL, based on Program 5.1, for learning softly falling rule lists from data. Given an instance (D, A, w, C, C_1) of Program 5.1, this algorithm searches through the space of rule lists that are compatible with D and finds a compatible rule list whose antecedents come from A, and whose objective value is the smallest among all the rule lists that the algorithm explores. It then turns this compatible rule list into a softly falling rule list. In the search phase, the algorithm uses the following prefix bound (Theorem 5.2) to prune the search space of compatible rule lists. The details of Algorithm softFRL and the proof of Theorem 5.2 can be found in the supplementary material.

Theorem 5.2. *Suppose that we are given an instance (D, A, w, C, C_1) of Program 5.1 and a prefix e that is compatible with D. Then any rule list d that begins with e and is compatible with D satisfies*

$$\tilde{L}(d, D, 1/(1+w), w, C, C_1) \geq \tilde{L}^*(e, D, w, C, C_1),$$

where

$$\begin{aligned}
&\tilde{L}^*(e, D, w, C, C_1) = \tilde{L}(e, D, 1/(1+w), w, C, C_1) \\
&+ \min\Bigg(\frac{1}{n}\left(\frac{1}{\alpha_{\min}^{(e,D)}} - 1\right)\tilde{n}_{e,D}^{+} + C \\
&\quad + C_1\lfloor \tilde{\alpha}_{e,D} - \alpha_{\min}^{(e,D)} \rfloor_+ + \frac{w}{n}\tilde{n}_{e,D}^{+}\mathbb{1}[\tilde{\alpha}_{e,D} \geq \alpha_{\min}^{(e,D)}], \\
&\quad \inf_{\beta:\zeta<\beta\leq 1} g(\beta), \frac{w}{n}\tilde{n}_{e,D}^{+} + C_1\lfloor \tilde{\alpha}_{e,D} - \alpha_{\min}^{(e,D)} \rfloor_+, \\
&\quad \frac{1}{n}\tilde{n}_{e,D}^{-} + C_1\lfloor \tilde{\alpha}_{e,D} - \alpha_{\min}^{(e,D)} \rfloor_+ \Bigg)
\end{aligned} \tag{10}$$

is a lower bound on the objective value of any compatible rule list that begins with e, under the instance (D, A, w, C, C_1) of Program 5.1. In Equation (10), $\alpha_{\min}^{(e,D)}$, ζ, and g are defined by

$$\alpha_{\min}^{(e,D)} = \min_{k<|e|} \alpha_k^{(e,D)},$$

$$\zeta = \max(\alpha_{\min}^{(e,D)}, \tilde{\alpha}_{e,D}, 1/(1+w)),$$
$$g(\beta) = \frac{1}{n}\left(\frac{1}{\beta} - 1\right)\tilde{n}_{e,D}^{+} + C + C_1(\beta - \alpha_{\min}^{(e,D)}).$$

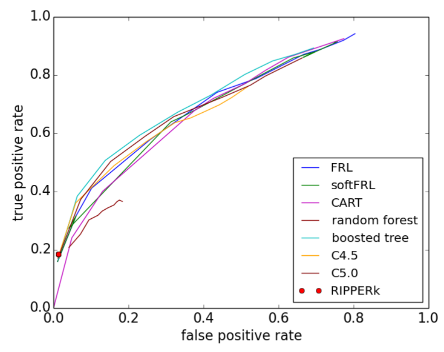
Note that $\inf_{\beta:\zeta<\beta\leq 1} g(\beta)$ can be computed analytically: $\inf_{\beta:\zeta<\beta\leq 1} g(\beta) = g(\beta^*)$ if $\beta^* = \sqrt{\tilde{n}_{e,D}^{+}/(C_1 n)}$ satisfies $\zeta < \beta^* \leq 1$, and $\inf_{\beta:\zeta<\beta\leq 1} g(\beta) = \min(g(\zeta), g(1))$ otherwise.

6 EXPERIMENTS

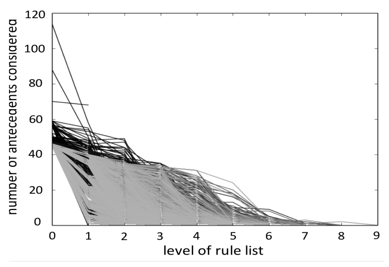
In this section, we demonstrate our algorithms for learning falling rule lists using a real-world application – learning the conditions that are predictive of the success of a bank marketing effort, from previous bank marketing campaign data. We used the public bank-full dataset (Moro et al., 2011), which contains 45211 observations, with 12 predictor variables that were discretized. We used the frequent pattern growth (FP-growth) algorithm (Han and Pei, 2000) to generate the set of antecedents A from the dataset. For reasons of model interpretability and generalizability, we included in A the antecedents that have at most 2 predicates, and have at least 10% support within the data that are labeled positive or within the data that are labeled negative. Besides the FP-growth algorithm, there is a vast literature on rule mining algorithms (e.g. Agrawal and Srikant, 1994; Han et al., 2000; Landwehr et al., 2005), and any of these can be used to produce antecedents for our algorithms.

The bank-full dataset is imbalanced – there are only 5289 positive instances out of 45211 observations. A trivial model that always predicts the negative outcome for a bank marketing campaign will achieve close to 90% accuracy on this dataset, but it will not be useful for the bank to understand what makes a marketing campaign successful. Moreover, when predicting if a future campaign will be successful in finding a client, the bank cares more about "getting the positive right" than about "getting the negative right" – a false negative means a substantial loss in revenue, while a false positive incurs little more than some phone calls.

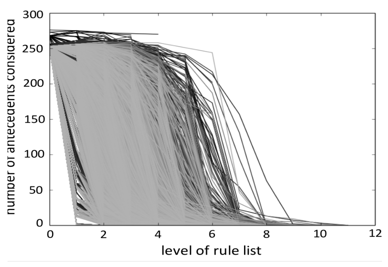
We compared our algorithms with other classification algorithms in a cost-sensitive setting, where a false negative and a false positive have different



(a) ROC curves on the test set using different w values for one training-test split

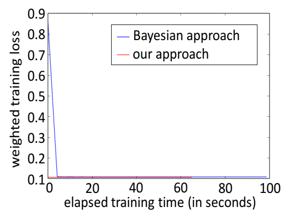


(b) Number of antecedents considered by Algorithm FRL

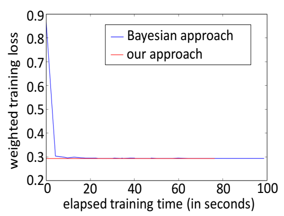


(c) Number of antecedents considered by Algorithm softFRL

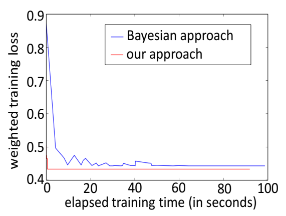
Figure 1: Experimental Results



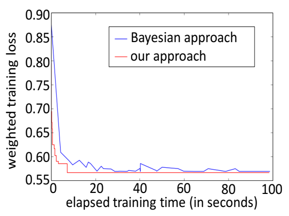
(a) positive weight $w = 1$



(b) positive weight $w = 3$



(c) positive weight $w = 5$



(d) positive weight $w = 7$

Figure 2: Plots of Weighted Training Loss over Real Runtime for Bayesian Approach and Algorithm FRL

costs of misclassification. We generated five random splits into a training and a test set, where 80% of the observations in the original bank-full dataset were placed into the training set. For each training-test split, and for each positive class weight $w \in \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\}$, we learned from the training set: (1) a falling rule list d, which is treated as a classifier $\tilde{d}_{1/(1+w)}$, using Algorithm FRL with $C = 0.000001$ (which is small enough so that no training accuracy will be sacrificed for sparsity), (2) a softly falling rule list d', which is treated as a classifier $\tilde{d}'_{1/(1+w)}$, using Algorithm softFRL with $C = 0.000001$ and $C_1 = 0.5$, (3) three decision trees using cost-sensitive CART (Breiman et al., 1984), cost-sensitive C4.5 (Quinlan, 1993), and cost-sensitive C5.0 (Quinlan, 2004), respectively, (6) a random forest (Breiman, 2001) of decision trees trained with cost-sensitive CART, (7) a boosted tree classifier using AdaBoost (Freund and Schapire, 1996) on trees trained with cost-sensitive CART, and (8) a decision list using RIPPERk (Cohen, 1995), and we computed the true positive rate and the false positive rate on the test set for each classifier. For each split and for each algorithm, we plotted a receiver operating characteristic (ROC) curve on the test set using different values of w. Figure 1a shows the ROC curves for one of the training-test splits. The ROC curves for the other training-test splits can be found in the supplementary material. Note that since RIPPERk is not a cost-sensitive algorithm, its ROC curve based on different w values has only a single point. As we can see, the curves in Figure 1a lie close to each other. This demonstrates the effectiveness of our algorithms in producing falling rule lists that, when used as classifiers, are comparable with classifiers produced by other widely used classification algorithms, in a cost-sensitive setting. This is possibly surprising since our models are much more constrained than other classification methods.

We also plotted the number of antecedents considered by Algorithm FRL and Algorithm softFRL in the process of constructing a rule list at each iteration (Figures 1b and 1c), when we applied the two algorithms to the entire dataset. Each curve in either plot corresponds to a rule list constructed in an iteration of the appropriate algorithm. The intensity of the curve is inversely proportional to the iteration number – the larger the iteration number, the lighter the curve is. The number of antecedents considered by Algorithm FRL stays below 60 in all but a few early iterations (despite a choice of 276 antecedents available), and the number considered by either algorithm generally decreases drastically in each iteration after three or four antecedents have been chosen. The curves generally become lighter as we move vertically down the plots, indicating that as we find better rule lists, there are less antecedents to consider at each level. Algorithm softFRL needs to consider more antecedents in general since the search space is less constrained. All of these demonstrate that the prefix bounds we have derived for our algorithms are effective in excluding a large portion of the search space of rule lists. The supplementary material contains more rule lists created using our algorithms with different parameter values.

Since this paper was directly inspired by Wang and Rudin (2015), who proposed a Bayesian approach to learning falling rule lists, we conducted a set of experiments comparing their work to ours. We trained falling rule lists on the entire bank-full dataset using both the Bayesian approach and our optimization approach, and plotted the weighted training loss over real runtime for each positive class weight $w \in \{1, 3, 5, 7\}$ with the threshold set to $1/(1 + w)$ (By Theorem 2.8, this is the threshold with the least weighted training loss for any given rule list). Since we want to focus our experiments on the efficiency of searching the model space, the runtimes recorded do not include the time for mining the antecedents. Note that the Bayesian approach is not cost-sensitive, and does not optimize the weighted training loss directly. However, in many real-life applications such as predicting the success of a future marketing campaign, it is desirable to minimize the expected weighted loss. Therefore, it is reasonable to compare the two approaches using the weighted training loss to demonstrate the advantages of our optimization approach. We compared the Bayesian approach only with Algorithm FRL, because both methods strictly enforce the monotonicity constraint on the positive proportions of the training data that are classified into each rule. Softly falling rule lists do not strictly enforce the monotonicity constraint, and are therefore not used for comparison. Figure 2 shows the plots of the weighted training loss over real runtime. Due to the random nature of both approaches, the experiments were repeated several times – more plots of the weighted training loss over real runtime for different trials of the same experiment, along with falling rule lists created using both approaches, can be found in the supplementary material. As shown in Figure 2, our optimization approach tends to find a falling rule list with a smaller weighted training loss faster than the Bayesian approach. This is not too surprising because in our approach, the search space is made substantially smaller by the tight bounds presented here, whereas in the original Bayesian approach, there are no tight bounds on optimal solutions to restrict the search space – even if we constructed bounds for the original Bayesian approach, they would involve loose approximations to gamma functions.

7 CONCLUSION

We have proposed an optimization approach to learning falling rule lists and softly falling rule lists, along with Monte-Carlo search algorithms that use bounds on the optimal solution to prune the search space. A recent work by Angelino et al. (2017) on (non-falling) rule lists showed that it is possible to exhaustively optimize an objective over rule lists, indicating that the space of lists is not as large as one might think. Our search space is a dramatically constrained version of their search space, allowing us to reasonably believe that it can be searched exhaustively. Unfortunately, almost none of the logic of Angelino et al. (2017) can be used here. Indeed, introducing the falling constraint or the monotonicity penalty changes the nature of the problem, and the bounds in our work are entirely different. The algorithm of Angelino et al. (2017) is not cost-sensitive, which led in this work to another level of complexity for the bounds.

Falling rule lists are optimized for ease-of-use – users only need to check a small number of conditions to determine whether an observation is in a high risk or high probability subgroup. As pointed out by Wang and Rudin (2015), the monotonicity in probabilities in falling rule lists allows doctors to identify the most at-risk patients easily. Typical decision tree methods (CART, C4.5, C5.0) do not have the added interpretability that comes from the falling constraint in falling rule lists: one may have to check many conditions in a decision tree to determine whether an observation is in a high risk or high probability subgroup – even if the decision tree has a small depth, it is possible that high risk subgroups are in different parts of the tree, so that one still has to check many conditions in order to find high risk subgroups. In this sense, falling rule lists and softly falling rule lists are as sparse as we need them to be, and they can provide valuable insight into data.

Supplementary Material and Code: The supplementary material and code are available at `https://github.com/cfchen-duke/FRLOptimization`.

Acknowledgments

This work was sponsored in part by MIT Lincoln Laboratory.

References

Agrawal, R. and Srikant, R. (1994). Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 487–499.

Altendorf, E. E., Restificar, A. C., and Dietterich, T. G. (2005). Learning from Sparse Data by Exploiting Monotonicity Constraints. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 18–26. AUAI Press.

Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., and Rudin, C. (2017). Learning Certifiably Optimal Rule Lists. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 35–44, New York, NY, USA. ACM.

Ben-David, A. (1995). Monotonicity Maintenance in Information-Theoretic Machine Learning Algorithms. *Machine Learning*, 19:29–43.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). Classification and Regression Trees.

Cohen, W. W. (1995). Fast Effective Rule Induction. In *Proceedings of the twelfth international conference on machine learning*, pages 115–123.

Feelders, A. and Pardoel, M. (2003). Pruning for Monotone Classification Trees. *Advances in Intelligent Data Analysis V, IDA 2003, Lecture Notes in Computer Science*, 2810:1–12.

Freitas, A. A. (2014). Comprehensible Classification Models – a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10.

Freund, Y. and Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, volume 96, pages 148–156.

Han, J. and Pei, J. (2000). Mining Frequent Patterns by Pattern-Growth: Methodology and Implications. *ACM SIGKDD explorations newsletter*, 2(2):14–20.

Han, J., Pei, J., and Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation. *ACM SIGMOD Record*, 29(2):1–12.

Huysmans, J., Dejaeger, K., Mues, C., Vanthienen, J., and Baesens, B. (2011). An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154.

Kodratoff, Y. (1994). The comprehensibility manifesto. *KDD Nuggets*, 94(9).

Landwehr, N., Kersting, K., and De Raedt, L. (2005). nFOIL: Integrating Naïve Bayes and FOIL. In *Proceedings of the twentieth national conference on artificial intelligence (AAAI-05)*, pages 795–800.

Letham, B., Rudin, C., McCormick, T. H., and Madigan, D. (2015). Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371.

Liu, B., Hsu, W., and Y., M. (1998). Integrating Classification and Association Rule Mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 80–86.

Martens, D. and Baesens, B. (2010). Building Acceptable Classification Models. *Data Mining*, pages 53–74.

Moro, S., Laureano, R., and Cortez, P. (2011). Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In *Proceedings of European Simulation and Modelling Conference (ESM'2011)*, pages 117–121. Eurosis.

Muggleton, S. and De Raedt, L. (1994). Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19:629–679.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, 1(1):81–106.

Quinlan, J. R. (1993). C4.5: programs for machine learning. 1.

Quinlan, R. (2004). Data Mining Tools See5 and C5.0.

Rivest, R. L. (1987). Learning Decision Lists. *Machine Learning*, 2(3):229–246.

Thabtah, F. (2007). A review of associative classification mining. *The Knowledge Engineering Review*, 22(01):37–65.

Wang, F. and Rudin, C. (2015). Falling Rule Lists. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Yang, H., Rudin, C., and Seltzer, M. (2017). Scalable Bayesian Rule Lists. In *International Conference on Machine Learning*, pages 3921–3930.