Falling Rule Lists

Fulton Wang
CSAIL and Sloan, MIT
fultonw@mit.edu

Cynthia Rudin
CSAIL and Sloan, MIT
rudin@mit.edu

Abstract

Falling rule lists are classification models consisting of an ordered list of if-then rules, where (i) the order of rules determines which example should be classified by each rule, and (ii) the estimated probability of success decreases monotonically down the list. These kinds of rule lists are inspired by healthcare applications where patients would be stratified into risk sets and the highest at-risk patients should be considered first. We provide a Bayesian framework for learning falling rule lists that does not rely on traditional greedy decision tree learning methods.

1 Introduction

In healthcare, patients and actions need to be prioritized based on risk. The most at-risk patients should be handled with the highest priority, patients in the second at-risk set should receive the second highest priority, and so on. This decision process is perfectly natural for a human decision-maker – for instance a physician – who might check the patient for symptoms of high severity diseases first, then check for symptoms of less serious diseases, etc.; however, the traditional paradigm of predictive modeling does not naturally contain this type of logic. If such clear logic were well-founded, a typical machine learning model would not usually be able to discover it: most machine learning methods produce highly complex models, and were not designed to provide an ability to reason about each prediction. This leaves a gap, where predictive models are not directly aligned with the decisions that need to be made from them.

The algorithm introduced in this work aims to resolve this problem, and could be directly useful for clinical practice. A *falling rule list* is an ordered list of if-then rules, where (i) the order of rules determines which example should be classified by each rule (falling rule lists are a type of decision list), and (ii) the estimated probability of success decreases monotonically down the list. Thus, a falling rule list directly contains the decision-making process, whereby the most at-risk observations are classified first, then the second set, and so on. A falling rule list might say, for instance, that patients with a history of heart disease are in the highest risk set with a 7% stroke risk, patients with high blood pressure (who are not in the highest risk set) are in the second highest risk set with a 4% stroke risk, and patients with neither conditions of these are in the lowest risk set with a 1% stroke risk.

Table 1 shows an example of one of the decision lists we constructed for the mammographic mass dataset (Elter *et al.* , 2007) as part of our experimental study. It took 35 seconds to construct this model on a laptop. The model states that if biopsy results show that the tumor has irregular shape, and the patient is over age 60, then the tumor is at the highest risk of being malignant (the risk is 85%). The next risk set is for the remaining tumors that have spiculated margins and are from patients above 45 years of age (the risk is 78%), and so on. The right column of Table 1 shows how many patients fit into each of the rules (so that its sum is the size of the dataset), and the risk probabilities were directly calibrated to the data.

Falling rule lists serve a dual purpose: they rank rules to form a predictive model, and stratify patients into decreasing risk sets. This saves work for a physician; sorting is an expensive mental operation, and this model does the sorting naturally. If one were to use a standard decision tree or decision list method instead, identifying the highest at-risk patients could be a much more involved calculation, and the number of conditions the most at-risk patients need to satisfy might be difficult for physicians to memorize.

Most of the models currently in use for medical decision making were designed by medical experts rather

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38.

	Conditions		Probability	Support
IF	IrregularShape AND Age $\geq$ 60	THEN malignancy risk is	85.22%	230
ELSE IF	SpiculatedMargin AND Age $\geq$ 45	THEN malignancy risk is	78.13%	64
ELSE IF	IllDefinedMargin AND Age $\geq$ 60	THEN malignancy risk is	69.23%	39
ELSE IF	IrregularShape	THEN malignancy risk is	63.40%	153
ELSE IF	LobularShape AND Density $\geq$ 2	THEN malignancy risk is	39.68%	63
ELSE IF	RoundShape AND Age $\geq$ 60	THEN malignancy risk is	26.09%	46
ELSE		THEN malignancy risk is	10.38%	366

Table 1: Falling rule list for mammographic mass dataset.

than by data-driven or algorithmic approaches. These manually-created risk assessment tools are used in possibly every hospital; e.g., the TIMI scores, $CHADS_2$ score, Apache scores, and the Ranson score, to name a few (Antman *et al.* , 2000; Morrow *et al.* , 2000; Gage *et al.* , 2001; Knaus *et al.* , 1981, 1985, 1991; Ranson *et al.* , 1974). These models can be computed without a calculator, making them very practical as decision aids. Of course, we aim for this level of interpretability in purely data-driven classifiers, with no manual feature selection or rounding coefficients.

Algorithms that discretize the input space have gained in popularity purely because they yield interpretable models. Decision trees (Breiman *et al.* , 1984; Quinlan, 1986, 1993), as well as decision lists (Rivest, 1987), organize a collection of simple rules into a larger logical structure, and are popular despite being greedy. Inductive logic programming (Muggleton & De Raedt, 1994) returns an unstructured set of conjunctive rules such that an example is classified as positive if it satisfies any of the rules in that set. An extremely simple way to induce a probabilistic model from the unordered set of rules given by an ILP method is to place them into a decision list (e.g., see Fawcett, 2008), ordering rules by empirical risk. This is also done in associative classification (e.g., see Thabtah, 2007). However, the resulting model cannot be expected to exhibit good predictive performance, as its constituent rules were chosen with a different objective.

Since it is possible that decision tree methods can produce results that are inconsistent with monotonicity properties of the data, there is a subfield dedicated to altering these greedy decision tree algorithms to obey monotonicity properties (Ben-David, 1995; Feelders & Pardoel, 2003; Altendorf *et al.* , 2005). Studies showed that in many cases, no accuracy is lost in enforcing monotonicity constraints, and that medical experts were more willing to use the models with the monotonicity constraints (Pazzani *et al.* , 2001).

Even with (what seem like) rather severe constraints on the hypothesis space such as monotonicity or sparsity in the number of leaves and nodes, it still seems that the set of accurate classifiers is often large enough so that it contains interpretable classifiers (see Holte, 1993). Because the monotonicity properties we enforce are much stronger than those of Ben-David (1995); Feelders & Pardoel (2003); Altendorf *et al.* (2005) (we are looking at monotonicity along the whole list rather than for individual features), we do find that accuracy is sometimes sacrificed, but not always, and generally not by much. On the other hand, it is possible that our method gains a level of practicality and interpretability that other methods simply cannot.

Interpretability is very context dependent (see Kodratoff, 1994; Pazzani, 2000; Freitas, 2014; Huysmans *et al.* , 2011; Allahyari & Lavesson, 2011; Martens & Baesens, 2010; Rüping, 2006; Verbeke *et al.* , 2011; Martens *et al.* , 2011), and no matter how one measures it in one domain, it can be different in the next domain. A falling rule list used in medical practice has the benefit that it can, in practice, be as sparse as desired. Since it automatically stratifies patients by risk in the order used for decision making, physicians can choose to look at as much of the list as they need to make a decision; the list is as sparse as one requires it to be. If physicians only care about the most high risk patients, they look only at the top few rules, and check whether the patient obeys any of the top clauses.

The algorithm we provide for falling rule lists aims to have the best of all worlds: accuracy, interpretability, and computation. The algorithm starts with a statistical assumption, which is that we can build an accurate model from pre-mined itemsets. This helps tremendously with computation, and restricts us to building models with only interpretable building blocks (see also Letham *et al.* , 2014; Wang *et al.* , 2014). Once the itemsets are discovered, a Bayesian modeling approach chooses a subset and permutation of the rules to form the decision list. The user determines the desired size of the rule list through a Bayesian prior. Our generative model is constructed so that the monotonicity property is fully enforced (no "soft" monotonicity).

The code for fitting falling rule lists is available online[1].

[1] `http://web.mit.edu/rudin/www/falling_rule_list`

2 Falling Rule Lists Model

We consider binary classification, where the goal is to learn a distribution $p(Y|x)$, where Y is binary. For example, Y might indicate the presence of a disease, and x would be a patient's features. We represent this conditional distribution as a decision list, which is an ordered list of IF...THEN... rules. We require a special structure to this decision list: that the probability of $Y = 1$ associated with each rule is *decreasing* as one moves down the decision list.

We use a Bayesian approach to characterize the posterior over falling rule lists given training data $D = \{(x_n, y_n)\}_{n=1,\dots,N}$ (of size N), $x_n \in X$, the patient feature space, hyperparameters H, and $y_n \in \{0, 1\}$. We represent a falling rule list with a set of parameters θ, specify the prior $p_\theta(\cdot; H)$ and likelihood $p_{\mathbf{Y}}(\{y_n\}|\theta; \{x_n\})$, and use simulated annealing and Monte Carlo sampling to approximate the MAP estimate and posterior over falling rule lists,

2.1 Parameters of Model

A falling rule list is parameterized by the following objects:

$$L \in \mathbb{Z}^+ \quad \text{(size of list)} \tag{1}$$

$$c_l(\cdot) \in B_X(\cdot), \text{ for } \quad l = 0, \dots, L-1 \quad \text{(IF clauses)} \tag{2}$$

$$r_l \in \mathbb{R}, \text{ for } \quad l = 0, \dots, L \quad \text{(risk scores)} \tag{3}$$

such that

$$r_{l+1} \leq r_l \text{ for } \quad l = 0, \dots, L-1 \quad \text{(monotonic)} \tag{4}$$

where $B_X(\cdot)$ denotes the space of boolean functions on patient feature space X. $B_X(\cdot)$ is the space of possible IF clauses; $c_l(x)$ will be 1 if x satisfies a given set of conditions. For this work, we will not assume L, the size of the decision list, to be known in advance. The value of r_l will be fed into the logistic function to produce a risk probability between 0 and 1. Thus, $c_0(\cdot)$ corresponds to the rule at the top of the list, determining the patients with the highest risk, and there are $L+1$ nodes and associated risk probabilities in the list: L associated with the $c_l(\cdot)$'s, plus one for *default* patients - those matching none of the L rules.

2.2 Likelihood

Given these parameters, the likelihood is as follows: Given L, let $Z(x; \{c_l(\cdot)\}_{l=0}^{L-1}) : X \to \{0, \dots, L\}$ be the mapping from feature x to the index of the length L rule list it "belongs" to (equals L for default patients):

$$Z(x; \{c_l(\cdot)\}_{l=0}^{L-1}) = \tag{5}$$

$$\begin{cases} L & \text{if } c_l(x) = 0 \text{ for } l = 0, \dots, L-1 \\ \min(l : c_l(x) = 1, \ l = 0, \dots, L-1) & \text{otherwise.} \end{cases}$$

Then, the likelihood is:

$$y_n | L, \{c_l(\cdot)\}_{l=0}^{L-1}, \{r_l\}_{l=0}^{L}; x_n \sim$$
$$\text{Bernoulli}(\text{logistic}(r_{z_n})), \text{ where} \tag{6}$$
$$z_n = Z(x_n; \{c_l(\cdot)\}_{l=0}^{L-1}). \tag{7}$$

2.3 Prior

Here, we describe the prior over the parameters L, $\{c_l\}_{l=0}^{L-1}, \{r_l\}_{l=0}^{L}$. We will provide a reparameterization that enforces monotonicity constraints, and finally give a generative model for the parameters.

As discussed earlier, to help with computation, we place positive prior probability of $\{c_l\}_{l=0}^{L-1}$ only over lists consisting of boolean clauses B returned by a frequent itemset mining algorithm, where for $c(\cdot) \in B$, we have $c(\cdot) : X \to \{0, 1\}$. For this particular work we used FPGrowth (Borgelt, 2005), whose input is a binary dataset where each x is a boolean vector, and whose output is a set of subsets of the features of the dataset. For example, x_2 might indicate the presence of diabetes, and x_{15} might indicate the presence of hypertension, and a boolean function returned by FPGrowth might return 1 for patients who have diabetes and hypertension. It does not matter which rule mining algorithm is chosen because they all perform breadth-first searches to return a set of clauses that have sufficient support. Here, B needs to be sufficiently large, so that the hypothesis space of considered models is not too small. B can be viewed as a hyperparameter, and the maximum length a decision list can have under our model is $|B|$, the number of rules in B.

2.3.1 Reparameterization

To ensure the monotonicity constraints that $r_l \geq r_{l-1}$ for $l = 1 \dots L$ in the posterior, we choose the scores r_l to, on a log scale, come from products of real numbers constrained to be greater than 1. That is, conditioned on L, we let

$$r_l = \log(v_l) \qquad \text{for } l = 0, \dots, L \tag{8}$$

$$v_l = K \Pi_{\hat{l}=l}^{L-1} \gamma_{\hat{l}} \qquad \text{for } l = 0, \dots, L-1 \tag{9}$$

$$v_L = K \tag{10}$$

and require that

$$\gamma_l \geq 1 \qquad \text{for } l = 0, \dots, L-1 \tag{11}$$

$$K \geq 0, \tag{12}$$

so that r_L, the risk score associated with the default rule, equals $\log K$. The prior we place over $\{\gamma_l\}_{l=0}^{L-1}$ and K will respect those constraints.

Thus, after reparameterizing, the parameters are

$$\theta = \{L, \{c_l(\cdot)\}_{l=0}^{L-1}, \{\gamma_l\}_{l=0}^{L-1}, K\}. \tag{13}$$

2.3.2 Prior Specifics

The prior over parameters $L, \{c_l(\cdot)\}_{l=0}^{L-1}, \{\gamma_l\}_{l=0}^{L-1}, K$ is generated through the following process:

1. Let hyperparameters $H = \{B, \lambda, \{\alpha_l\}_{l=0}^{|B|-1}, \{\beta_l\}_{l=0}^{|B|-1}, \alpha_K, \beta_K, \{w_l\}_{l=0}^{|B|-1}\}$ be given.
2. Initialize $\Theta \leftarrow \{\}$.
3. Draw $L \sim \text{Poisson}(\lambda)$.
4. For $l = 0, \ldots, L-1$ draw

$$c_l(\cdot) \sim p_{c(\cdot)}\left(\cdot|\Theta; B, \{w_l\}_{l=0}^{|B|-1}\right) \tag{14}$$

$$p_{c(\cdot)}\left(c(\cdot) = c_j(\cdot)|\Theta; B, \{w_l\}_{l=0}^{|B|-1}\right) \tag{15}$$

$$\propto w_j \text{ if } c_j(\cdot) \notin \Theta \text{ and 0 otherwise.} \tag{16}$$

$$\text{Update } \Theta \leftarrow \Theta \cup \{c_l(\cdot).\} \tag{17}$$

5. For $l = 0, \ldots, L-1$ draw $\gamma_l \sim \text{Gamma}_1(\alpha_l, \beta_l)$, which is a Gamma distribution truncated to have support only above 1.
6. Draw $K \sim \text{Gamma}(\alpha_K, \beta_K)$.

We now elaborate on our choice for each involved distribution. We let $L \sim \text{Poisson}(\lambda)$, where λ reflects the prior decision length desired by the user. We let $c_l(\cdot)$ be the result of a draw from a discrete distribution over the yet unchosen rules, $B \setminus \{c_l(\cdot)\}_{l=0}^{l-1}$, where the l-th rule is drawn with probability proportional to a user designed weight w_l. For example, a rule might be chosen with probability proportional to the number of clauses in it. This allows the user to express preferences over different types of clauses in the list. Given L, only $\{c(\cdot)_l\}_{l=1}^{L-1}$ are observed, though note this process specifies a joint distribution over all of $\{c(\cdot)_l\}_{l=1}^{|B|-1}$. Letting $\{\gamma_l\}_{l=0}^{L-1}$ to be independently distributed truncated gamma variables permits posterior Gibbs sampling while enforcing the monotonicity constraints and still permitting diversity over prior distributions. For example, one could encourage some of the γ's near the middle (of L) of the list to be large, in which case the risks would be widely spaced in the middle of the list (but this would force closer spacing at the top of the list where the risks concentrate near 1). Finally, K, which models the risk of patients not satisfying any rules, is Gamma distributed.

3 Fitting the Model

First we describe our approach to finding the decision list with the maximum a posteriori probability. Then we discuss our approach to perform Monte Carlo sampling from the posterior distribution over decision list parameters $\theta = \{L, c_{0,\ldots,L-1}(\cdot), K, \gamma_{0,\ldots,L-1}\}$ as described in Equation (13),

$$p_{\text{post}}(L, c_{0,\ldots,L-1}(\cdot), K, \gamma_{0,\ldots,L-1}|y_{1,\ldots,N}; \mathbf{x}_{1,\ldots,N}). \tag{18}$$

3.1 Obtaining the MAP decision list

We adopted a simulated annealing approach to find $\theta^* = \{L^*, c_{0,\ldots,L^*-1}(\cdot)^*, K^*, \gamma_{0,\ldots,L^*-1}\}$, where

$$L^*, c^*_{0,\ldots,L^*-1}(\cdot), K^*, \gamma^*_{0,\ldots,L^*-1} \in \text{argmax}_{L,c_{0,\ldots,L-1}(\cdot),K,\gamma_{0,\ldots,L-1}} \mathcal{L} \tag{19}$$

where $\mathcal{L}$ is shorthand for the unnormalized log of the posterior given in Equation (18). We note that the optimization problem in Equation (19) is equivalent to finding:

$$L^*, c_{0,\ldots,L^*-1}(\cdot)^* \in \text{argmax}_{L,\{c_l(\cdot)\}_{l=0}^{L-1}} \mathcal{L}(L, \{c_l(\cdot)\}_{l=0}^{L-1}, K^*, \gamma^*_{0,\ldots,L-1}) \tag{20}$$

where

$$K^*, \gamma^*_{0,\ldots,L-1} \in \text{argmax}_{K,\gamma_{0,\ldots,L-1}} \mathcal{L}(L, \{c_l(\cdot)\}_{l=0}^{L-1}, K, \gamma_{0,\ldots,L-1}). \tag{21}$$

Note that K^* and $\gamma^*_{0,\ldots,L-1}$ depend on $L, \{c_l(\cdot)\}_{l=0}^{L-1}$.

Furthermore, the solution to the subproblem of finding K^* and $\gamma^*_{0,\ldots,L-1}$ can be approximated closely using a simple procedure, as it involves maximizing the posterior probability of a decision list given the rules $\{c_l(\cdot)\}_{l=0}^{L-1}$. Optimizing Equation (20) lends itself better to simulated annealing than Equation (19); optimizing (20) involves optimization over a discrete space, namely the set and order of rules $c_l(\cdot)$. In this formulation, at each simulated annealing iteration, we need to evaluate the objective function for the current rule list $\{c_l(\cdot)\}_{l=0}^{L-1}$, which involves solving the continuous subproblem of finding the corresponding K^* and $\gamma^*_{0,\ldots,L-1}$.

Given a objective function $E(s)$ over discrete search space S, a function specifying the set of neighbors of a state $N(s)$, and a temperature schedule function over time steps, $T(t)$, a simulated annealing procedure is a discrete time, discrete state Markov Chain $\{s_t\}$ where at time t, given the current state s_t, the next state s_{t+1} is chosen by first randomly selecting a proposal $\grave{s}$ from the set $N(s)$, and setting $s_{t+1} = \grave{s}$ with probability $\min(1, \exp(-\frac{E(\grave{s})-E(s)}{T(t)}))$, and $s_{t+1} = s_t$ otherwise.

The search space of the optimization problem of Equation (20) is $\left\{L, \{c_l(\cdot)\}_{l=0}^{L-1}\right\}$, the set of ordered lists of rules drawing from the finite pre-mined set of rules B. Based on Equation (20), we let

$$E(\{c_l(\cdot)\}_{l=0}^{L-1}) = -\mathcal{L}(L, \{c_l(\cdot)\}_{l=0}^{L-1}, K^*, \gamma^*_{0,\ldots,L-1}). \tag{22}$$

We simultaneously define the set of neighbors and the process by which to randomly choose a neighbor through the following random procedure that alters the current rule list $\{c_l(\cdot)\}_{l=0}^{L-1}$ to produce a new rule list $\{\grave{c}_l(\cdot)\}_{l=0}^{\grave{L}-1}$ (The new list's length may be different):

Choose uniformly at random one of the following 4 operations to apply to the current rule list, $\{c_l(\cdot)\}_{l=0}^{L-1}$:

1. SWAP: Select $i \neq j$ uniformly from $0, \ldots, L-1$, and swap the rules at those 2 positions, letting $\grave{c}_i(\cdot) \leftarrow c_j(\cdot)$ and $\grave{c}_j(\cdot) \leftarrow c_i(\cdot)$.
2. REPLACE: Select i uniformly from $0, \ldots, L-1$, draw $c(\cdot)$ from the the distribution $p_{c(\cdot)}(\cdot|\Theta; B, \{w_l\}_{l=0}^{|B|-1})$ defined in Equation (16), where $\Theta = \{c_l(\cdot)\}_{l=0,\ldots,i-1,i+1,\ldots,L-1}$ and set $\grave{c}_i(\cdot) \leftarrow c(\cdot)$.
3. ADD: Choose one of the $L+1$ possible insertion points uniformly at random, draw a rule $c(\cdot)$ from $p_{c(\cdot)}(\cdot|\Theta; B, \{w_l\}_{l=0}^{|B|-1})$, where $\Theta = \{c_l(\cdot)\}_{l=0,\ldots,L-1}$, and insert it at the chosen insertion point, so that $\grave{L} \leftarrow L+1$.
4. REMOVE: Choose i uniformly at random from $0, \ldots, L-1$, and remove $c_i(\cdot)$ from the current rule list, so that $\grave{L} \leftarrow L-1$.

Note that this approach optimizes over the full set of rule lists from itemsets, and does not rely on greedy splitting. Even Bayesian tree methods that aim to traverse a wider search space use greedy splitting and local solutions, e.g. Chipman *et al.* (1998).

3.2 Obtaining the posterior

To perform posterior sampling, we use Gibbs sampling steps over $\{\gamma_l\}_{l=0}^{L-1}$ and K made possible by variable augmentation, and Metropolis-Hastings steps over L and $\{c_l(\cdot)\}$. We describe the variable augmentation step, the schedule of updates we employ, and finally the details of each individual update step.

Augmenting the model with two additional variables U_n, ζ_n for each $n = 1, \ldots, N$ preserves the marginal distribution over the original variables, and enables Gibbs sampling over K and each γ_l (see Dunson, 2004):

$$\zeta_n \sim \text{Exponential}(1) \qquad \text{for } n = 1, \ldots, N \tag{23}$$
$$U_n \sim \text{Poisson}(\zeta_n v_{z_n}) \qquad \text{for } n = 1, \ldots, N \tag{24}$$
$$Y_n = 1(U_n > 0) \qquad \text{for } n = 1, \ldots, N. \tag{25}$$

Marginalizing over ζ_n, we see that in this augmented model, $y_n \sim \text{Bernoulli}(\text{logistic}(r_{z_n}))$, as before:

$$p(Y_n = 1) = p(U_n > 0) \tag{26}$$
$$= 1 - \int_0^\infty p(U_n = 0|\zeta_n) p(\zeta_n) d\zeta_n \tag{27}$$
$$= 1 - \int_0^\infty \exp(-\zeta_n v_{z_n}) \exp(-\zeta_n) d\zeta_n \tag{28}$$
$$= 1 - (1 + v_{z_n})^{-1} \tag{29}$$
$$= \frac{\exp(r_{z_n})}{1 + \exp(r_{z_n})}. \tag{30}$$

3.2.1 Schedule of Updates

Given the augmented model, we cycle through the following steps in the following deterministic order. These will each be discussed in detail shortly. Regarding notation, we will use use θ_{aug} to refer to the parameters of the augmented model: $(L, \{c_l(\cdot)\}_{l=0}^{L-1}, K, \{\gamma_l\}_{l=0}^{L-1}, \{U_n\}_{n=1}^N, \{\zeta_n\}_{n=1}^N)$, so that Gibbs updates can be described more succinctly.

Step 1 (Gibbs steps for each γ_l):
Sample $\grave{\gamma}_l \sim p_{\gamma_l}(\cdot|(\theta_{\text{aug}} \setminus \gamma_l), \{y_n\}_{n=1}^N; \{x_n\}_{n=1}^N)$ for $l = 0, \ldots, L-1$.

Step 2 (Gibbs step for K):
Sample $\grave{K} \sim p_K(\cdot|(\theta_{\text{aug}} \setminus K), \{y_n\}_{n=1}^N; \{x_n\}_{n=1}^N)$.

Step 3 (Collapsed Metropolis Hastings Step):
Perform Metropolis-Hastings update over $(L, \{c_l(\cdot)\}_{l=0}^{L-1})$, under the original model from Equation (18). This can be viewed as a collapsed Metropolis-Hastings step, where the collapsed parameters are the augmenting variables $\{U_n\}_{n=1}^N, \{\zeta_n\}_{z=1}^N$.

Step 4 (Gibbs step for $(\{\zeta_n\}_{n=1}^N, \{U_n\}_{n=1}^N)$):
Jointly sample

$$(\{\grave{\zeta}_n\}_{n=1}^N, \{\grave{U}_n\}_{n=1}^N)$$
$$\sim p_{\{\zeta_n\}_n, \{U_n\}}(\cdot|\theta_{\text{aug}} \setminus (\{\zeta_n\}_n, \{U_n\}_n), \{y_n\}_n; \{x_n\}_n).$$

Mixing Gibbs and collapsed Metropolis-Hastings sampling steps requires special care to ensure the Markov chain is proper in that it possesses the desired stationary distribution. We refer to van Dyk & Park (2011) for details, but do note that after the collapsed Metropolis-Hastings step, first performing a Gibbs update of $\{\zeta_n\}_{n=1}^N$, and then a separate Gibbs update for $\{U_n\}_{n=1}^N$ (or in reverse) would not have been proper.

3.2.2 Update Details

We now elaborate on each step of the update schedule:

Step 1 In this augmented model, the full conditional distribution of each γ_l is Gamma distributed, so that it can be sampled from directly. Let, for $l = 0, \ldots, L-1$

$$\sigma_k^{(l)} = \begin{cases} K\Pi_{i=k, i\neq l}^{L-1} \gamma_i & \text{for } 0 \leq k \leq l \\ 0 & \text{for } l < k \leq L. \end{cases} \tag{31}$$

Then, it can be derived that

$$\gamma_l|(\theta_{\text{aug}} \setminus \gamma_l), \{y_n\}_{n=1}^N; \{x_n\}_{n=1}^N \tag{32}$$
$$\sim \text{Gamma}\left(\alpha_l + \sum_{n=1}^N 1[z_n \leq l] U_n, \beta_l + \sum_{n=1}^N \zeta_n \sigma_{z_n}^{(l)}\right),$$

where α_l, β_l govern the prior of γ_l and z_n as described in Equation (7) denotes the rule a datum belongs to.

Step 2 Similarily, let

$$o_k = \begin{cases} \Pi_{i=k}^{L-1}\gamma_i & \text{for } 0 \le k \le L-1 \\ 1 & \text{for } k = L. \end{cases} \quad (33)$$

Then

$$K|(\theta_{\text{aug}} \setminus K), \{y_n\}_{n=1}^N; \{x_n\}_{n=1}^N \quad (34)$$

$$\sim \text{Gamma}\left(\alpha_K + \sum_{n=1}^N U_n, \beta_K + \sum_{n=1}^N \zeta_n o_{z_n}\right). \quad (35)$$

Step 3 The reason for using a collapsed rather than regular Metropolis-Hastings step was to improve chain mixing. The Metropolis-Hastings proposal distributions over $(L, \{c_l(\cdot)\}_{l=0}^{L-1})$ are exactly as in the proposal distribution used to generate a successor state in the simulated annealing we used to find the MAP decision list. The only difference is that if the ADD operation is chosen and a rule $c(\cdot)$ inserted at position k in the rule list, then sample $\grave{\gamma}_k \sim \text{Gamma}(\alpha_k, \beta_k)$, the prior distribution of γ_k, and insert it at position k in $\{\gamma_l\}_{l=0}^{L-1}$. Thus, we simply provide the Metropolis-Hastings Q probabilities. For simplicity, we do so for the case when the weights $\{w_l\}_{l=0}^{|B|-1}$ associated with each $c(\cdot) \in B$ are equal.

$$Q(\{\grave{c}(\cdot)\}_{l=1}^{\grave{L}-1}; \{c_l(\cdot)\}_{l=1}^{L-1}) = \begin{cases} \frac{1}{(L+1)(|B|-L)} & \text{if ADD} \\ \frac{1}{(|B|-L)L} & \text{if REPLACE} \\ \frac{1}{L} & \text{if REMOVE} \\ \frac{2}{L(L-1)} & \text{if SWAP}. \end{cases}$$

Step 4 In the full conditional distribution of $(\{\zeta_n\}_{n=1}^N, \{U_n\}_{n=1}^N)$, the set of pairs of variables, $\{(U_n, \zeta_n)\}_{n=1}^N$ is mutually independent, due to conditioning. Therefore to sample from it, it is sufficient to independently sample (U_n, ζ_n) for $n = 1, \ldots, N$. It can be shown that the following sampling scheme samples from the full conditional distribution: For $n = 1, \ldots, N$, if $y_n = 0$, set $\grave{U}_n = 0$ and sample

$$\grave{\zeta}_n \sim \text{exponential}(1 + v_{z_n}). \quad (36)$$

Otherwise, sample

$$\grave{U}_n \sim 1 + \text{Geometric}\left(\frac{1}{1+v_{z_n}}\right), \text{ then} \quad (37)$$

$$\grave{\zeta}_n \sim \text{Gamma}\left(1 + \grave{U}_n, 1 + v_{z_n}\right). \quad (38)$$

4 Simulation Studies

We show that for simulated data generated by a known decision list, our simulated annealing procedure that searches for the MAP decision list, with high probability, recovers the true decision list.

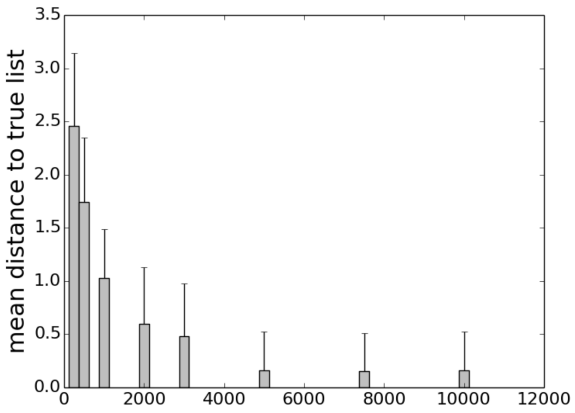

Figure 1: Mean distance to true list decreases with sample size.

Given observations with arbitrary features, and a collection of rules on those features, we can construct a binary matrix where the rows represent observations and the columns represent rules, and the entry is 1 if the rule applies to that observation and 0 otherwise. We need only simulate this binary matrix to represent the observations without losing generality. For our simulations, we generated independent binary rule sets with 100 rules by setting each feature value to 1 independently with probability 0.25.

We generated a random decision list of size 5 by selecting 5 rules at random, and setting the $\gamma_0, \ldots, \gamma_5$ so that the induced $p_0, \ldots, p_5$ were roughly evenly spaced on the unit interval: $(.84, .70, .54, .40, .25, .14)$. For each N, we performed the following procedure 100 times: generate the random rule matrix, random decision list, and assign it the aforementioned $\{\gamma_l\}$, obtain an independent dataset of size N by generating labels from this decision list, and then perform simulated annealing using the procedure described in Section 3.1 to obtain a point estimate of the decision list. We then calculate the edit distance of the decision list returned by simulated annealing to the true decision list. Figure 1 shows the average distance over those 100 replicates, for each N. We ran simulated annealing for 5000 steps in each replicate, and used a gradual cooling schedule.

5 Experiments

Our main experimental result is an application of Falling Rule Lists to predict 30 day hospital readmission from an ongoing collaboration with medical practitioners (details to appear in Cronin *et al.* , 2014).

Since we placed an extremely strong restriction on the characteristics of the predictive model (namely the monotonicity property, sparsity of rules, and sparsity of conditions per rule), we expect to lose predictive ac-

curacy over unrestricted methods. The interpretability benefit may or not be sufficient to compensate for this, but this is heavily application-dependent. We have found several cases where there is no loss in performance (with a substantial gain in interpretability) by using a falling rule list instead of, say, a support vector machine, consistent with the observations of Holte (1993) about very simple classifiers performing well.

Later in this section, we aim to quantify the loss in predictive power from Falling Rule Lists over other methods by using an out-of-sample predictive performance evaluation. Specifically, we compare to several baseline methods on standard publicly available datasets to quantify the possible loss in predictive performance.

5.1 Predicting Hospital Readmissions

We applied Falling Rule Lists to preliminary readmissions data being compiled through a collaboration with a major hospital in the U.S. (Cronin *et al.* , 2014), where the goal is to predict whether a patient will be readmitted to the hospital with 30 days, using data prior to their release. The dataset contains features and binary readmissions outcomes for approximately 8,000 patients who had no prior history of readmissions. The features are very detailed, and include aspects like "impaired mental status," "difficult behavior," "chronic pain," "feels unsafe" and over 30 other features that might be predictive of readmission. As we will see, luckily a physician may not be required to collect this amount of detailed information to assess whether a given patient is at high risk for readmission.

For these experiments and the experiments in the next section, no parameters were tuned in Falling Rule Lists (FRL), and the global hyperparameters were chosen as follows. We mined rules with a support of at least 5% and a cardinality of at most 2 conditions per rule. We assumed in the prior that conditioned on L, each rule had an equal chance of being in the rule list. We set the prior of $\{\gamma_l\}|L$ to have noninformative and independent distributions of gamma$(1, 0.1)$, and the prior expected length of the decision list, λ, to be 8. We performed simulated annealing search for 5000 steps with a constant temperature of 1 for simplicity.

We measured out-of-sample performance using the AUROC from 5-fold cross validation where the MAP decision list from training was used to predict on each test fold in turn. We compared with SVM (with Radial Basis Function kernels), ℓ_2 regularized logistic regression (Ridge regression, denoted LogReg), CART, and random forests (denoted RF), implemented in Python using the scikit-learn package. For SVM and logistic regression, hyperparameters were tuned with grid search in nested cross validation.

Method	Mean AUROC (STD)
FRL	.80 (.02)
NF_FRL	.75 (.02)
NF_GRD	.75 (.02)
RF	.79 (.03)
SVM	.62 (.06)
Logreg	.82 (.02)
Cart	.52 (.01)

Table 3: AUROC values for readmission data

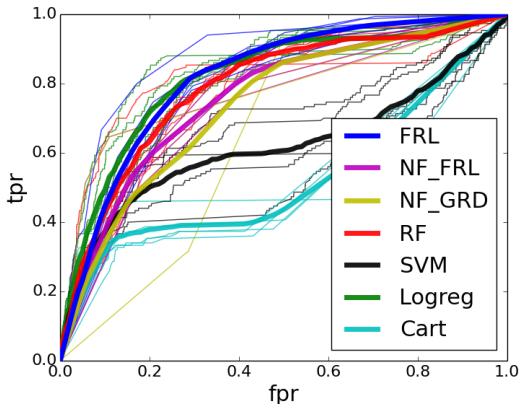

Figure 2: ROC curves for readmissions prediction.

As discussed, decision lists consisting of rules from an inductive logic programming method are not expected to exhibit strong performance. We tested nFoil (Landwehr *et al.* , 2005) with the default settings (max number of clauses set to 2) to obtain a set of rules. These rules were ordered in two different ways, to form two additional comparison methods: 1. by the empirical risk of each rule (denoted NF_GRD), and 2. by using the set of rules as the pre-mined rule set that FRL accepts as input (denoted NF_FRL). Note that the risk probabilities in rule lists returned by NF_GRD are not necessarily decreasing monotonically, and that not all of the nFoil rules are necessarily in the rule list returned by NF_FRL, since omission of a rule can increase the posterior.

The AUROC's for the different methods are in Table 3, indicating that there was no loss in accuracy for using Falling Rule Lists on this particular dataset. For all of the training folds, the decision lists had a length of either 6 or 7 – all very sparse.

Figure 2 shows ROC curves for all test folds for all methods. The mean ROC curves are bolded. For this particular dataset, SVM RBF and CART did not perform well. It is unclear why SVM did not perform well, as cross-validation was performed for SVM; usually SVM's perform well when cross-validated (though

	Conditions		Probability	Support
IF	BedSores AND Noshow	THEN readmissions risk is:	33.25%	770
ELSE IF	PoorPrognosis AND MaxCare	THEN readmissions risk is:	28.42%	278
ELSE IF	PoorCondition AND Noshow	THEN readmissions risk is:	24.63%	337
ELSE IF	BedSores	THEN readmissions risk is:	19.81%	308
ELSE IF	NegativeIdeation AND Noshow	THEN readmissions risk is:	18.21%	291
ELSE IF	MaxCare	THEN readmissions risk is:	13.84%	477
ELSE IF	Noshow	THEN readmissions risk is:	6.00%	1127
ELSE IF	MoodProblems	THEN readmissions risk is:	4.45%	1325
ELSE		Readmissions risk is:	0.88%	3031

Table 2: Falling rule list for patients with no multiple readmissions history.

it is definitely possible for them to have poor performance on some datasets – on the other hand, CART often performs poorly relative to other methods, in our experience). As expected, the nFoil-based methods exhibited worse performance than our faIling rule list method. FRL performed on par with the best method, despite its being limited to a very small number of features with the monotonic structure.

Table 2 shows a point estimate obtained from training Falling Rule Lists on the full dataset, which took 88 seconds. The "probability" column is the empirical probability of readmission for each rule; "support" indicates the number of patients classified by that rule.

The model indicates that patients with bed sores and who have skipped appointments are the most likely to be readmitted. The other conditions used in the model include "PoorPrognosis" meaning the patient is in need of palliative care services, "PoorCondition" meaning the patient exhibits frailty, signs of neglect or malnutrition, "NegativeIdeation" which means suicidal or violent thoughts, and "MaxCare" which means the patient needs maximum care (is non-ambulatory, confined to bed). This model lends itself naturally to decision-making, as one need only view the top of the list to obtain a characterization of high risk patients.

5.2 Performance on Public Datasets

We performed an empirical comparison on several UCI datasets (Bache & Lichman, 2013), using the above experimental setup. This allows us to quantify the loss in accuracy due to the restricted form of the model.

Table 4 displays the results. As discussed earlier, we observed that even with the severe restrictions on the model, performance levels are still on par with those of other methods, and not often substantially worse. This is likely due to the benefits of not using a greedy splitting method, restricting to the space of mined rules, careful formulation, and optimization.

Furthermore, FRL beats the nFoil based methods in performance on all the public datasets. Again, the reason is that the set of rules from nFoil was found using a different objective, not intended to predict well when placed into a decision list. Even with NF_FRL, where any subset of the nFoil rules could be selected and placed in any order, performance was poor, showing the nFoil rule set did not contain rules that were useful for a falling rule list. The set of nFoil rules was always much smaller than those from FPGrowth (see supplement), overly restricting the hypothesis space.

6 Conclusion

We present a new class of interpretable predictive models that could potentially have a major benefit in decision-making for some domains. As nicely stated by the director of the U.S. National Institute of Justice (Ridgeway, 2013), an interpretable model that is actually used is better than one that is more accurate that sits on a shelf. We envision that models produced by FRL can be used, for instance, by physicians in third world countries who require models printed on laminated cards. In high stakes decisions (like medical decisions), it is important we know whether to trust the model we are using to make decisions; models like FRL help tell us when (and when not) to trust.

Acknowledgment: We gratefully acknowledge funding from Wistron, Siemens, and NSF-CAREER IIS-1053407.

Method	Spam	Mamm	Breast	Cars
FRL	.91(.01)	.82(.02)	.95(.04)	.89(.08)
NF_FRL	.90(.03)	.67(.03)	.70(.11)	.60(.21)
NF_GRD	.91(.03)	.72(.04)	.82(.12)	.62(.20)
SVM	.97(.03)	.83(.01)	.99(.01)	.94(.08)
Logreg	.97(.03)	.85(.02)	.99(.01)	.92(.09)
CART	.88(.05)	.82(.02)	.93(.04)	.72(.17)
RF	.97(.03)	.83(.01)	.98(.01)	.92(.05)

Table 4: AUROC value comparisons over datasets

References

Allahyari, Hiva, & Lavesson, Niklas. 2011. User-oriented Assessment of Classification Model Understandability. *Pages 11–19 of: SCAI*.

Altendorf, Eric E, Restificar, Angelo C, & Dietterich, Thomas G. 2005. Learning from sparse data by exploiting monotonicity constraints. *Pages 18–26 of: Conf. Uncertainty in Artificial Intelligence*. AUAI Press.

Antman, Elliott M, Cohen, Marc, Bernink, Peter JLM, McCabe, Carolyn H, Horacek, Thomas, Papuchis, Gary, Mautner, Branco, Corbalan, Ramon, Radley, David, & Braunwald, Eugene. 2000. The TIMI risk score for unstable angina/non–ST elevation MI. *The Journal of the American Medical Association*, **284**(7), 835–842.

Bache, K., & Lichman, M. 2013. *UCI Machine Learning Repository*.

Ben-David, Arie. 1995. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, **19**(1), 29–43.

Borgelt, Christian. 2005. An Implementation of the FP-growth Algorithm. *Pages 1–5 of: Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*. ACM.

Breiman, Leo, Friedman, Jerome H., Olshen, Richard A., & Stone, Charles J. 1984. *Classification and Regression Trees*. Wadsworth.

Chipman, Hugh A, George, Edward I, & McCulloch, Robert E. 1998. Bayesian CART model search. *Journal of the American Statistical Association*, **93**(443), 935–948.

Cronin, Patrick, Ustun, Berk, Rudin, Cynthia, & Greenwald, Jeffrey. 2014. *Work In Progress*.

Dunson, David B. 2004. *Bayesian isotonic regression for discrete outcomes*. Tech. rept. Citeseer.

Elter, M, Schulz-Wendtland, R, & Wittenberg, T. 2007. The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process. *Medical physics*, **34**(11), 4164–4172.

Fawcett, Tom. 2008. PRIE: a system for generating rulelists to maximize ROC performance. *Data Mining and Knowledge Discovery*, **17**(2), 207–224.

Feelders, Ad, & Pardoel, Martijn. 2003. Pruning for monotone classification trees. *Pages 1–12 of: Advances in intelligent data analysis V*. Springer.

Freitas, Alex A. 2014. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations Newsletter*, **15**(1), 1–10.

Gage, Brian F, Waterman, Amy D, Shannon, William, Boechler, Michael, Rich, Michael W, & Radford, Martha J. 2001. Validation of clinical classification schemes for predicting stroke. *The Journal of the American Medical Association*, **285**(22), 2864–2870.

Holte, Robert C. 1993. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, **11**(1), 63–90.

Huysmans, Johan, Dejaeger, Karel, Mues, Christophe, Vanthienen, Jan, & Baesens, Bart. 2011. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, **51**(1), 141–154.

Knaus, William A, Zimmerman, Jack E, Wagner, Douglas P, Draper, Elizabeth A, & Lawrence, Diane E. 1981. APACHE-acute physiology and chronic health evaluation: a physiologically based classification system. *Critical Care Medicine*, **9**(8), 591–597.

Knaus, William A, Draper, Elizabeth A, Wagner, Douglas P, & Zimmerman, Jack E. 1985. APACHE II: a severity of disease classification system. *Critical Care Medicine*, **13**(10), 818–829.

Knaus, William A, Wagner, DP, Draper, EA, Zimmerman, JE, Bergner, Marilyn, Bastos, PG, Sirio, CA, Murphy, DJ, Lotring, T, & Damiano, A. 1991. The APACHE III prognostic system. Risk prediction of hospital mortality for critically ill hospitalized adults. *Chest Journal*, **100**(6), 1619–1636.

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

Landwehr, Niels, Kersting, Kristian, & De Raedt, Luc. 2005. nFOIL: Integrating naıve bayes and FOIL.

Letham, Benjamin, Rudin, Cynthia, McCormick, Tyler H., & Madigan, David. 2014. An interpretable model for stroke prediction using rules and Bayesian analysis. *In: Proceedings of 2014 KDD Workshop on Data Science for Social Good*.

Martens, David, & Baesens, Bart. 2010. Building acceptable classification models. *Pages 53–74 of: Data Mining*. Springer.

Martens, David, Vanthienen, Jan, Verbeke, Wouter, & Baesens, Bart. 2011. Performance of classification models from a user perspective. *Decision Support Systems*, **51**(4), 782–793.

Morrow, David A, Antman, Elliott M, Charlesworth, Andrew, Cairns, Richard, Murphy, Sabina A, de Lemos, James A, Giugliano, Robert P, McCabe, Carolyn H, & Braunwald, Eugene. 2000. TIMI risk score for ST-elevation myocardial infarction: a convenient, bedside, clinical score for risk assessment at presentation an intravenous nPA for treatment of infarcting myocardium early II trial substudy. *Circulation*, **102**(17), 2031–2037.

Muggleton, Stephen, & De Raedt, Luc. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, **19**, 629–679.

Pazzani, Michael J. 2000. Knowledge discovery from data? *Intelligent systems and their applications, IEEE*, **15**(2), 10–12.

Pazzani, MJ, Mani, S, & Shankle, WR. 2001. Acceptance of rules generated by machine learning among medical experts. *Methods of information in medicine*, **40**(5), 380–385.

Quinlan, J. Ross. 1986. Induction of decision trees. *Machine learning*, **1**(1), 81–106.

Quinlan, J. Ross. 1993. *C4. 5: programs for machine learning*. Vol. 1. Morgan kaufmann.

Ranson, JH, Rifkind, KM, Roses, DF, Fink, SD, Eng, K, Spencer, FC, *et al.* . 1974. Prognostic signs and the role of operative management in acute pancreatitis. *Surgery, gynecology & obstetrics*, **139**(1), 69.

Ridgeway, Greg. 2013. The Pitfalls of Prediction. *NIJ Journal,* National Institute of Justice, **271**, 34–40.

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

Rüping, Stefan. 2006. *Learning interpretable models*. Ph.D. thesis, Universität Dortmund.

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

van Dyk, David A, & Park, Taeyoung. 2011. Partially collapsed Gibbs sampling and path-adaptive Metropolis-Hastings in high-energy astrophysics. *Handbook of Markov Chain Monte Carlo*, 383–397.

Verbeke, Wouter, Martens, David, Mues, Christophe, & Baesens, Bart. 2011. Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert Systems with Applications*, **38**(3), 2354–2364.

Wang, Tong, Rudin, Cynthia, Doshi, Finale, Liu, Yimin, Klampfl, Erica, & MacNeille, Perry. 2014. *Bayesian Ors of Ands for Interpretable Classification with Application to Context Aware Recommender Systems*. Tech. rept.