Enhanced Random Forest Algorithms for Partially Monotone Ordinal Classification

Christopher Bartley, Wei Liu, Mark Reynolds
Computer Science and Software Engineering
University of Western Australia, Perth, Australia
christopher.bartley@research.uwa.edu.au,{wei.liu,mark.reynolds}@uwa.edu.au

Abstract

One of the factors hindering the use of classification models in decision making is that their predictions may contradict expectations. In domains such as finance and medicine, the ability to include knowledge of monotone (nondecreasing) relationships is sought after to increase accuracy and user satisfaction. As one of the most successful classifiers, attempts have been made to do so for Random Forest. Ideally a solution would (a) maximise accuracy; (b) have low complexity and scale well; (c) guarantee global monotonicity; and (d) cater for multi-class. This paper first reviews the state-of-the-art from both the literature and statistical libraries, and identifies opportunities for improvement. A new rule-based method is then proposed, with a *maximal accuracy* variant and a faster *approximate* variant. Simulated and real datasets are then used to perform the most comprehensive ordinal classification benchmarking in the monotone forest literature. The proposed approaches are shown to reduce the bias induced by monotonisation and thereby improve accuracy.

Introduction

This paper concerns *monotone* prior knowledge. A monotone (or *non-decreasing*) relationship between X and Y means that an increase in X should not lead to a decrease in Y. For example, a house with three bedrooms should not be cheaper than one with two bedrooms (all other factors constant). Monotonicity between input variable (feature) x_j and output variable y may thus be defined:

For an increase in x_j, response variable y should not decrease (all other variables held constant).

Monotonicity can be defined when the model output is ordered, including *ordinal classification*, which seeks to assign objects to two or more ordered classes (but where the *distances* between classes are irrelevant). Examples include credit ratings (AA, A, BB, ...), or cancer diagnosis (No/Yes).

This paper focuses on Random Forest (RF) style tree ensembles with independent large (unpruned) trees (Breiman 2001). RF is one of the most popular classifiers, achieving both high accuracy and ease of use. It's competitive accuracy was somewhat surprisingly reasserted in the comprehensive 2015 comparison by Fernandez et al. of 179 classifiers against 121 datasets (Fernandez-Delgado et al. 2014), where it was the highest ranked model family. We note that this comparison omitted deep learning approaches, to which monotonicity has recently been extended (You et al. 2017). However, there remains a significant use case for RF for smaller datasets, users with less expertise, where computational power is limited and where solution time is critical.

However, despite its popularity, for sensitive domains like finance and medicine where users want to know how the model works, RF's lack of *comprehensibility* is a problem. As a result the ability to define *monotone features* is popularly requested and implemented in many libraries such as `R`'s `GBM` and `Arborist` and `XGBoost`. This helps achieve *quality control* (ensuring a model behaves 'sensibly'), satisfaction of *user requirements* (e.g. known medical risk factors) and improved *comprehensibility*. Although still not fully comprehensible, features may now be categorised as increasing or decreasing, with magnitude estimated by variable importance measures. This latter is particularly compelling given the recently enacted GDPR legislation [1], which confers on *data subjects* the right to '*meaningful information about the logic involved*' in '*automated decision making*'.

This paper therefore pursues the twin goals of comprehensibility and accuracy. We first review the state-of-the-art in monotone random forest, with extensions where necessary to multi-class. This review for the first time combines both previous literature *and* approaches in popular statistical libraries that are absent from the literature.

A key limitation of some approaches is that monotonicity is increased *locally*, but not guaranteed *globally*. There is a big difference between being able to claim that an algorithm *usually* complies, and that it categorically *always* complies. The latter is surely preferable. Of the techniques that *do* achieve global monotonicity, we show their mechanisms can result in sub-optimal monotonisation (high loss/bias). Since RF primarily improves accuracy by *variance* reduction, this introduced *bias* cannot be corrected and accuracy suffers.

We thus propose a method with global monotonicity and minimal monotonisation loss. Examples and simulation demonstrate lower monotonisation loss, and experiments on 17 real datasets show a compelling increase in accuracy. The algorithm is available at `github`[2].

[1] https://eur-lex.europa.eu/eli/reg/2016/679/oj

[2] https://github.com/chriswbartley/monoensemble

Monotone Ordinal Classification and Random Forest

The classification task is to build a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ to predict the class $y \in \mathcal{Y}$ given input point $\mathbf{x} \in \mathcal{X}$, where $\mathcal{Y} = \{c_1, c_2, ..., c_C\}$. The classifier f is built using a training set of N samples $\{(\mathbf{x}_i, y_i)\}, i = 1..N, \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}$. The *ordinal* classification task adds a total order on the classes to be predicted $c_1 \prec c_2 \prec ... \prec c_C$. Unlike regression, the distance between classes is undefined.

Partially monotone ordinal classifiers allow the user to specify that one or more of the features in $\mathcal{X}$ has a *non-decreasing* (non-increasing) impact on the output class. For clarity we split the input space $\mathbb{R}^p$ into $\mathcal{X} \times \mathcal{Z}$, $\mathcal{X} \subseteq \mathbb{R}^{p_x}$,$\mathcal{Z} \subseteq \mathbb{R}^{p_z}$, where $\mathcal{X}$ contains the p_x monotone features, and $\mathcal{Z}$ the p_z non-monotone features ($p = p_x + p_z$). Thus:

Definition 1 ***Partial Dominance***
Given $(\boldsymbol{x}, \boldsymbol{z}), (\boldsymbol{x}', \boldsymbol{z}') \in \mathbb{R}^{p_x} \times \mathbb{R}^{p_z}$*, the partial order* $\preceq_{\mathcal{X}}$ *is:*

$$(\boldsymbol{x}, \boldsymbol{z}) \preceq_{\mathcal{X}} (\boldsymbol{x}', \boldsymbol{z}') \Leftrightarrow \boldsymbol{x} \preceq \boldsymbol{x}' \wedge \boldsymbol{z} = \boldsymbol{z}' \quad (1)$$

As a partial order, $\preceq_{\mathcal{X}}$ is transitive, anti-symmetric and reflexive. We can now define a partially monotone function:

Definition 2 ***Partially Monotone Function***
Function $F : \mathcal{X} \times \mathcal{Z} \rightarrow \mathcal{Y}$*,* $\mathcal{X} \subseteq \mathbb{R}^{p_x}$*,*$\mathcal{Z} \subseteq \mathbb{R}^{p_z}$ *is* monotone increasing *in the features of* $\mathcal{X}$ *(i.e.* $\{1..p_x\}$*) if:*

$$\boldsymbol{x} \preceq \boldsymbol{x}' \Rightarrow F(\boldsymbol{x}, \boldsymbol{z}) \leq F(\boldsymbol{x}', \boldsymbol{z}), \forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \boldsymbol{z} \in \mathcal{Z} \quad (2)$$

Random Forest (RF) The RF algorithm was proposed by Breiman in 2001 and emerges from the tree ensemble techniques developed in the late 1990s (Breiman 2001). Each component tree differs from the well-known CART algorithm due to: first, each tree is trained on a *bootstrap sample* of the training data ('bagging'); second, as each tree is inducted only a *random subset* of m_{try} of the features are considered for the best split. The greatly improved accuracy achieved by RF is due to the variance reducing impact of bagging on unbiased predictors (unpruned trees) that are decorrelated (due to m_{try}) (Hastie, Tibshirani, and J. 2009). As will become apparent, it is therefore crucial to minimise bias when monotonising the trees, because bagging (unlike boosting) cannot correct for it.

Existing Monotone RF Approaches

This section describes state-of-the-art monotone tree ensembles and critiques them as they apply to RF. To our knowledge this is the most comprehensive account in the literature, and includes approaches from statistical libraries that are not mentioned in the literature (split-wise constraints and the `XGBoost` approach) or exist independently within different branches (Isotonic Classification Trees vs Rule Reshaping).

First Order Stochastic Dominance RF (FSD-RF) The naïve solution to monotonising a tree is to constrain each branch to prohibit nonmonotone splits. For regression trees this corresponds to ensuring the means of proposed leaf nodes correspond to the leaf partial order (for splits on monotone features). This technique is implemented for example in the `GBM`[3] and `Arborist`[4] libraries for R. For binary classification the constraint is simply applied to $\Pr(y{=}{+}1)$ rather than the leaf mean.

We extend this to multi-class using First-order Stochastic Dominance (FSD). FSD is a partial order over probability distributions, and is expressed in terms of the cumulative probability distribution (cdf). For ordinal classes $c \in \{1..C\}$ the cdf $F_A(c^*) = Pr(c \leq c^*)$ has first order stochastic dominance over $F_B(c^*)$ if $F_A(c^*) \leq F_B(c^*) \quad \forall c^* \in \{1..C\}$. FSD-RF thus simply prohibits non-FSD branch splits.

The advantage of split-wise constraints is that it has virtually no computational overhead. The disadvantage is that monotonicity is not necessarily global. For example Figure 1 shows a tree that is *not* globally monotone despite each branch satisfying constraints on x_1 and x_2: when node t_6 splits on $x_2 \leq 3$, the created t_7 predicts $y = -1$ and is non-monotone in both x_1 and x_2.

Monotone Induction Decision Trees (MID-RF) González, Herrera, and Garcia (2015) propose a forest based on Ben-David's MID Trees (Ben-David 1995). MID modifies tree induction by choosing the split to minimise not just entropy E, but $T = E + RA$, where A is the 'order ambiguity', and $R \geq 0$ is a weighting factor. A penalises splits that result in nonmonotone branch pairs. MID-RF randomises $R \in \{1, 2, ..., R_{lim}\}, R_{lim} = 100$ for each tree and also prunes the ensemble to retain only the $x = 50\%$ most monotone trees. MID-RF has the advantage of natively allowing multi-class classification.

MID-RF does not achieve global monotonicity since the leaf-wise non-monotonicity penalty discourages but does not prohibit nonmonotone splits. The second limitation is complexity. Each branch split compares all possible splits against all other existing leaves to calculate A, resulting in $O(n_{tree} m_{try} p \tilde{N}^3)$ complexity, where n_{tree} is the number of trees, m_{try} is the number of random features considered, p is the number of features, and $\tilde{N}$ is the number of unique points in a bootstrap sample ($\approx 0.632N$).

Partially Monotone Random Forest (PM-RF) Bartley, Liu, and Reynolds (2016) monotonise RF by introducing training point weights $\mathbf{s} = \{s_i \mid i = 1..N\}$ and using convex optimisation to calculate them such that the ensemble complies with K discrete constraints $(\underset{\sim}{\mathbf{x}}_k, \tilde{\mathbf{x}}_k)$ designed to correct nonmonotonicities while minimising $\sum_{i=1}^{N}(s_i - \frac{1}{N})^2$. The disadvantages are similar to MID-RF: although constraint compliance is assured global monotonicity is not, and solving the quadratic program is approximately $O(N^3)$.

PM-RF is a binary classifier, and to extend it to multi-class we use the monotone ensembling by Kotlowski (Kotlowski 2008). Let $h_c(\mathbf{x}) = \mathbf{1}[y < c]$ be a binary classifier that distinguishes *class less than c* from *class greater than or equal to c*, where $\mathbf{1}[condition] = 1$ if $condition$ else 0. Then multi-class classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ ($\mathcal{Y} = \{1, 2, ..., C\}$) is given by:

[3] https://cran.r-project.org/web/packages/gbm/index.html
[4] past.rinfinance.com/agenda/2016/talk/MarkSeligman.pdf

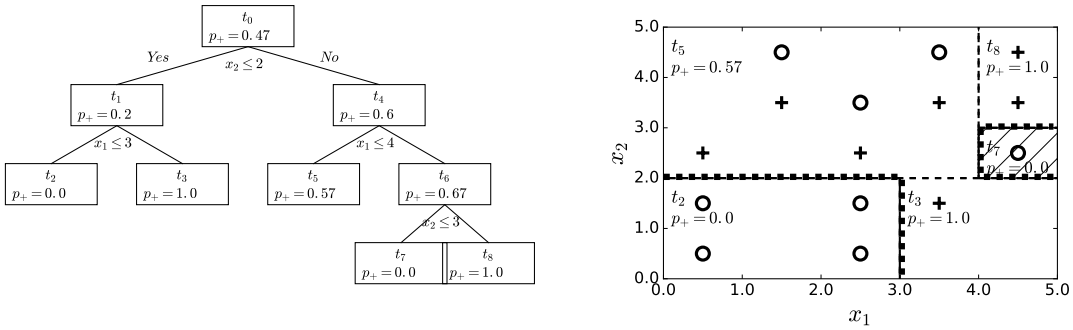

Figure 1: Split Constrained Tree Example (monotone in x_1, x_2). Left: Tree. Right: Leaf partitions shown dotted, class boundary bold dotted. Positive class $+$, negative $\circ$. p_+ denotes $\Pr(y{=}{+}1)$. Note nonmonotone t_7, despite splits respecting monotonicity.

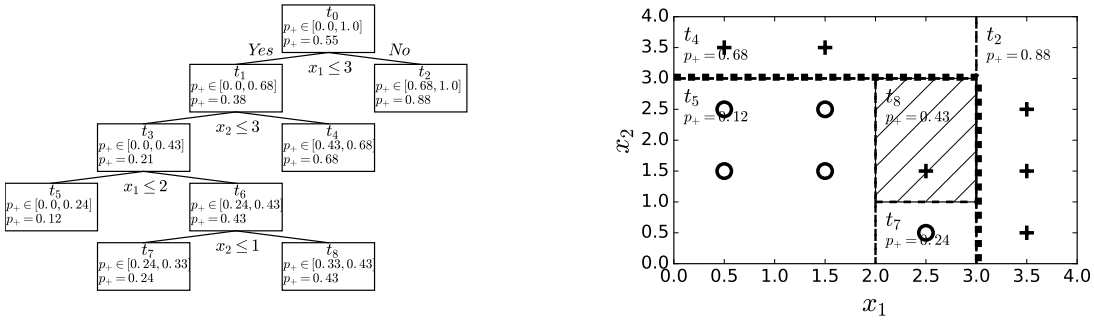

Figure 2: XGBoost Monotone Tree Example. Left: Tree. Right: Leaf partitions dotted, class boundary bold dotted. Positive class $+$, negative $\circ$. p_+ denotes $\Pr(y{=}{+}1)$. Note t_8 non-optimally predicts $y = -1$ ($p_+ \leq 0.5$) due to inherited constraints.

$$h(\mathbf{x}) = 1 + \sum_{c=2}^{C} h_c(\mathbf{x}) \tag{3}$$

If $h_c(\mathbf{x})$ are monotone, $h(\mathbf{x})$ are monotone with L1 loss bounded by the sum of the L1 loss of h_c (Kotlowski 2008).

XG-Boost Constrained Trees (XGM-RF) The popular `XGBoost` (Chen and Guestrin 2016) uses a novel approach that combines split constraints with *inherited leaf coefficient limits*. At each split, the mean of the left and right coefficients is passed down as an *upper* limit on future coefficients to the left leaf, and a *lower* limit to the right leaf (for binomial deviance these are the mean of $logit(\Pr(y{=}{+}1))$ rather than $\Pr(y{=}{+}1)$). These inherited constraints are respected when greedily finding optimal leaf splits and effectively guarantee global monotonicity. This elegant approach has virtually no computational overhead while achieving globally guaranteed monotonicity.

The disadvantage is that this can severely reduce the in-sample accuracy of large trees (i.e. it *biases* the tree away from the training data). Figure 2 shows how this occurs. The root node t_0 passes an upper limit to t_1 of $p_+ \leq 0.68$, which is reduced to $p_+ \leq 0.43$ when split to create t_3. Then from t_3 onwards it is impossible to predict $y{=}{+}1$ ($p_+ > 0.5$). Thus when t_8 (hatched) is created to extract the positive point, the lowest loss solution allowed is $p_+{=}0.43$, predicting $y{=}{-}1$. This is despite the fact t_8 could predict $p_+{=}0.68$ without violating monotonicity. In fact after depth ≈ 2 one branch will be constrained to predict $y{=}{+}1$, and one to predict $y{=}{-}1$.

In its original context (gradient boosting) this approach remains ideal because the trees are shallow (depth ≈ 3), *and* any bias introduced by one tree can be corrected by subsequent trees. In contrast, RF uses trees created from independent bootstrap samples (with no opportunity to correct introduced bias), and grows deep trees to near leaf purity (often having > 40 leaves and depth > 6). This approach thus introduces bias that appears likely to reduce RF accuracy.

To extend this method to multi-class we cannot use standard one-vs-rest (OVR) ensembling because the partial order is unclear (e.g. Class 2 vs Classes 1,3,4). Instead we use the Kotlowski ensembling (Kotlowski 2008), as outlined for PM-RF in (3).

Isotonic Classification Trees (ICT-RF) Recently Bonakdarpour et al. (2018) proposed a 'rule reshaping' based monotone RF. They define a partial order between the leaves of a tree, where leaf A dominates leaf B if it is *possible* for (a) points in leaf A to be greater than those in leaf B in the *monotone* features; *and* (b) for their *nonmonotone* features to be equal. Isotonic regression then provides a loss minimised leaf relabelling that respects

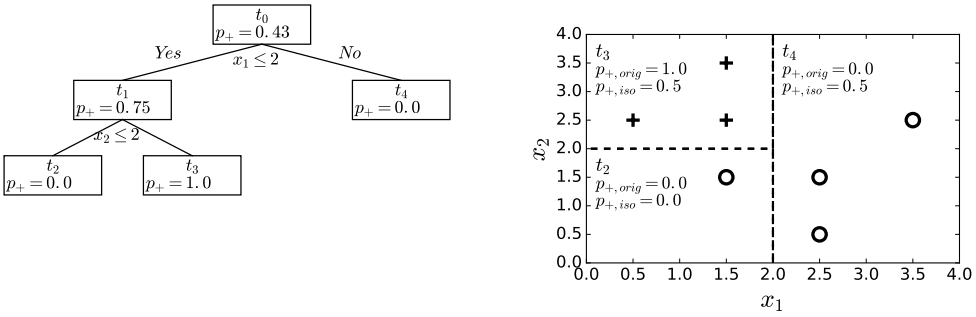

Figure 3: Isotonic Classification Tree Example. Left: Tree. Right: Leaf partitions dotted, class boundary bold dotted. Positive class +, negative ∘. p_+ denotes $\Pr(y{=}{+}1)$. Best possible solution to isotonic relabelling is $p_+ = 0.43$ for all leaves (i.e. $\hat{y}{=}{-}1$).

this leaf partial order. We note this is in effect the *Isotonic Classification Trees* (ICT) approach proposed by van de Kamp, Feelders, and Barile (2009), where it is also noted that the tree may be sequentially simplified to eliminate branches where both leaves predict the same class. Thus for ICT a relabelling/simplification cycle is repeated until all leaves are monotone with respect to each other. ICT also allows multi-class by using monotone ensembling based on FSD and selecting the lowest median to allow multi-class. Thus instead of the 'rule reshaping' approach we use ICT for each tree.

However, we observe this is likely to yield biased and sub-optimal monotonisation. Because the regression is limited to relabelling existing leaves, the monotonisation candidates are not always very good. This is illustrated in Figure 3. The original tree finds leaf purity with three leaves, with probabilities $p_{+,orig}$. The regression must then respect the leaf partial order $t_2 \preceq t_3 \preceq t_4$ and the optimal and only solution is $p_{+,iso} = 0.43$, universally predicting the majority class (-1) and misclassifying the *three* + points). The obvious solution (two nodes $x_2 \leq 2$ and $x_2 > 2$), which would only misclassify *one* point, is not possible by leaf relabelling because the regression was given an unfortunate leaf partition. Another disadvantage of ICT (and rule reshaping) is that the solution to isotonic regression is typically $O(N^3)$ for interior point solutions to the quadratic program.

A New Method: Monotone Rule RF (MR-RF)

We now propose an approach to address the weaknesses above. MID-RF, PM-RF, and FSD-RF are unable to guarantee monotonicity. XGM-RF and ICT-RF do guarantee global monotonicity, but both use mechanisms that are likely to lead to sub-optimal monotonisation.

We propose instead to convert the Random Forest to a rule ensemble in such a way as to ensure global monotonicity while reducing the introduced bias. First observe that binary RF can be considered as an additive rule ensemble (Bartley, Liu, and Reynolds 2016; Bonakdarpour et al. 2018). Splitting the input space into the monotone ($\mathbf{x}$) and non-monotone ($\mathbf{z}$), the classifier produced by tree ensembles can be written as a weighted sum over all leaves of all trees:

$$F(\mathbf{x}, \mathbf{z}) = sign\left(\sum_{t=1}^{T}\sum_{l=1}^{L_t} a_{t,l} f_{t,l}(\mathbf{x}, \mathbf{z})\right) \tag{4}$$

where:

$$f_{t,l}(\mathbf{x}, \mathbf{z}) = \mathbf{1}\big[(\mathbf{x}, \mathbf{z}) \in \text{ leaf } l \text{ of tree } t\big] \tag{5}$$

is the leaf membership indicator function

$$a_{t,l} = \frac{1}{N}\sum_{i=1}^{N} y_i \frac{b_{i,t}}{K_{t,l}} \text{ is the leaf prediction}$$

and $b_{i,t}$ is the number of occurrences of $\mathbf{x}_i$ in the bootstrap sample used for tree t and T, N, L_t are the number of trees, training points, and leaves in tree t respectively.

We first extend the typical instance based monotone rule ensemble (Kotlowski 2008) to partial monotonicity. $F(\mathbf{x}, \mathbf{z})$ will be partially monotone in the features of $\mathcal{X}$ if it fulfils:

Theorem 1 (Partially Monotone Ensemble Conditions) *Given a function $F : \mathcal{X} \times \mathcal{Z} \rightarrow \mathcal{Y}$ of the form $F(\boldsymbol{x}, \boldsymbol{z}) = a_0 + \sum_{m=1}^{M} a_m f_m(\boldsymbol{x}, \boldsymbol{z})$, F will satisfy (2) if each $f_m(\boldsymbol{x}, \boldsymbol{z})$ and a_m satisfy one of the below ($m = 1..M$):*

(a) $f_m(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{1}\big[\boldsymbol{x} \succeq \boldsymbol{x}_m \text{ and } \boldsymbol{z} \in \mathcal{Z}_m\big]$
$a_m \geq 0$

(b) $f_m(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{1}\big[\boldsymbol{x} \preceq \boldsymbol{x}_m \text{ and } \boldsymbol{z} \in \mathcal{Z}_m\big]$
$a_m \leq 0$

where:
$\boldsymbol{x}_m \in \mathcal{X}, \mathcal{Z}_m \subseteq \mathcal{Z}$
The proof is easily done and omitted.

The question then becomes how best to convert the RF leaf rules $f_{t,l}(\mathbf{x})$ into rules complying with Theorem 1. We start by rewriting (5) for leaf membership in terms of the logical conjunction of the branch node rules that lead to it:

$$f_{t,l}(\mathbf{x}, \mathbf{z}) = \mathbf{1}\big[R_{t,l}(\mathbf{x}) \wedge S_{t,l}(\mathbf{z})\big] \tag{6}$$

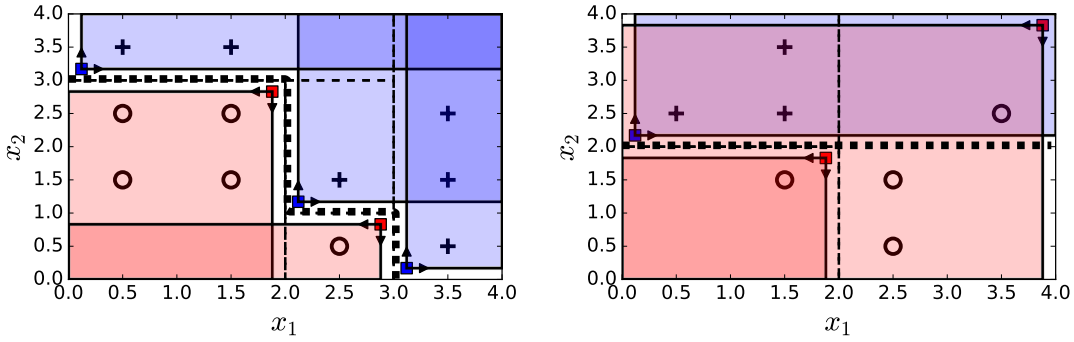

Figure 4: Example Monotone Rule solutions. Left panel: Solution to XGBoost Example (from Figure 2). Right panel: Solution to ICT-RF Example (from Figure 3). Class boundary bold dotted. In both cases a lower loss monotone solution is found.

where:

$$R_{t,l}(\mathbf{x}) = r_{t,l,1}(\mathbf{x}) \wedge ... \wedge r_{t,l,R_l}(\mathbf{x}) \quad (7)$$

$$S_{t,l}(\mathbf{z}) = s_{t,l,1}(\mathbf{z}) \wedge ... \wedge s_{t,l,S_l}(\mathbf{z}) \quad (8)$$

$$r_{t,l,k}(\mathbf{x}) \in \{x_i \leq v^{i-}_{t,l,k},\ x_i > v^{i+}_{t,l,k}\}$$

$$s_{t,l,p}(\mathbf{z}) \in \{z_j \leq q^{j-}_{t,l,k},\ z_j > q^{j+}_{t,l,k},\ z_j \in Q^{j}_{t,l,k},\ z_j \in \mathcal{Z}_j \setminus Q^{j}_{t,l,k}\}$$

In other words, each branch split leading to a leaf node is captured in a rule $r_{t,l,k}(\mathbf{x})$ for monotone features or $s_{t,l,p}(\mathbf{z})$ for non-monotone features. Non-monotone feature rules can take the form of thresholds ($z_j \leq q^{j-}_{t,l,k}$) for ordinal features, or set membership ($z_j \in Q^{j}_{t,l,k}$) for categorical features.

We observe that the Theorem 1 condition for *non-monotone* features $\mathbf{z}$ is satisfied by all leaves, because conditions of the form $S_{t,l}(\mathbf{z})$ simply describe a fixed subset of $\mathcal{Z}$. Thus for compliance with Theorem 1, leaves with $a_{t,l} \geq 0$ must be able to express $R_{t,l}(\mathbf{x})$ in the form $\mathbf{x} \succeq \mathbf{x}_{t,l}$ for some base point $\mathbf{x}_{t,l}$. Given it is trivial to show this also applies to strict inequalities $\succ$, it is possible if all conditions $r_{t,l,k}(\mathbf{x})$ use the $>$ inequality. Then we can rewrite (7) as:

$$R_{t,l}(\mathbf{x}) = \mathbf{x} \succ \mathbf{x}_{t,l} \quad (9)$$

$$\text{where: } \mathbf{x}_{t,l} = (x^1_{t,l}, ..., x^i_{t,l}, ..., x^{p_x}_{t,l})$$

$$x^i_{t,l} = \max\left[\{-\inf\} \cup \{v^{i+}_{t,l,k}\}_{k=1}^{R_l}\right]$$

In other words, element i of $\mathbf{x}_{t,l}$ is $-\inf$ if there are no branch nodes associated with feature i, or else the largest of the lower bounds $v^i_{t,l,k}$ in any branch node leading to leaf l.

For a given RF tree, leaves that can be written in this form (and analogously for $a_{t,l} \leq 0$ leaves) are therefore already monotone compliant under Theorem 1. To address the leaves that do *not* comply, we now present Monotone Rule RF (Algorithm 1), which takes a tree ensemble, and returns a monotone rule ensemble in log odds (logistic) form.

Monotonisation is achieved in two steps. Firstly the leaf rules are made compliant (lines 1-10): the sign of the leaf coefficient $a_{t,l}$ is taken as evidence of whether the rule should be positive or negative, and the rule is modified to eliminate upper limits (positive rules) or lower limits (negative rules).

This process is illustrated by comparing Figure 4 left panel with Figure 2 (XGM-RF example), and right panel with Figure 3 (ICT-RF example). In both cases MR-RF avoids the relevant pitfall, finding the superior monotone solution to t_8 for XGM-RF example (mislabelling 0 points rather than 1), and the lower loss two node solution for the ICT-RF example (mislabelling 1 instead of 3 points).

However, because we now have a set of *overlapping* rules we need to re-calculate the coefficients $\{\alpha_{t,l}\}$, with the constraint that the coefficient signs must match the rule direction (line 11). Below we present two options to achieve this.

Constrained Logistic Regression (MR-RF-log) The obvious solution is to calculate coefficients that minimise a loss function on the training data. Binomial deviance is an appropriate loss function for binary classification, resulting in the logistic regression problem with coefficient sign constraints:

$$\{\alpha_{t,l}\}_{l=0}^{L_t} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, F^*_t(\mathbf{x}_i, \mathbf{z}_i)\big) + \lambda\, ||\boldsymbol{\alpha}||_2^2 \quad (10)$$

$$\alpha_{t,l} \leq 0, \text{ if } a_{t,l} \leq 0$$

$$\alpha_{t,l} > 0, \text{ if } a_{t,l} > 0$$

where:

$$F^*_t(\mathbf{x}_i, \mathbf{z}_i)) = \alpha_{t,0} + \sum_{l=1}^{L_t} \alpha_{t,l} f^*_{t,l}(\mathbf{x}, \mathbf{z}) \quad (11)$$

$$L(y_i, F^*_t(\mathbf{x}_i, \mathbf{z}_i)) = log(1 + exp(-2 y_i F^*_t(\mathbf{x}_i, \mathbf{z}_i)))$$

It is well known regularisation is needed to stabilise logistic regression when the data is separable. We use L2 regularisation but L1 would also work as long as regularisation is minimal. This standard problem is solvable by interior point or gradient descent. We use `scipy`'s `fmin-tnc` solver.

Naïve Bayesian approximation (MR-RF-bay) The disadvantage of logistic regression is complexity. Here we derive an approximation equivalent to Naïve Bayesian (NB) classification. By making the 'naïve' assumption of conditional independence between the features, with the log odds

Algorithm 1 Monotone Rule Random Forest (MR-RF)

Input:

$X = \{(\mathbf{x}_1, \mathbf{z}_1)..(\mathbf{x}_N, \mathbf{z}_N)\}$ ▷ Training data, monotone in $\mathbf{x}$

$Y = \{y_1..y_N \mid y_i \in \{-1, +1\}\}$ ▷ Training labels

$F(\mathbf{x}, \mathbf{z}) = sign\left(\sum_{t=1}^{T} \sum_{l=1}^{L_t} a_{t,l} f_{t,l}(\mathbf{x}, \mathbf{z})\right)$

Output:

$F^*(\mathbf{x}, \mathbf{z}) = \frac{1}{T} \sum_{t=1}^{T} \left(\alpha_{t,0} + \sum_{l=1}^{L_t} \alpha_{t,l} f^*_{t,l}(\mathbf{x}, \mathbf{z})\right)$ ▷ Monotone Rule Ensemble

for each $t = 1..T$ **do**
  for each $l = 1..L_t$ **do**
    if $a_{t,l} > 0$ **then**
      $\mathbf{x}^*_{t,l} = (x^1_{t,l}, ..., x^i_{t,l}, ..., x^{p_x}_{t,l})$
      ▷ where $x^i_{t,l} = \max\left[\{-\inf\} \cup \{v^{i+}_{t,l,k}\}_{k=1}^{R_l}\right]$
      $f^*_{t,l}(\mathbf{x}, \mathbf{z}) = \mathbf{1}\left[\mathbf{x} \succ \mathbf{x}^*_{t,l} \wedge S_{t,l}(\mathbf{z})\right]$
    else
      $\mathbf{x}^*_{t,l} = (x^1_{t,l}, ..., x^i_{t,l}, ..., x^{p_x}_{t,l})$
      ▷ where $x^i_{t,l} = \min\left[\{+\inf\} \cup \{v^{i-}_{t,l,k}\}_{k=1}^{R_l}\right]$
      $f^*_{t,l}(\mathbf{x}, \mathbf{z}) = \mathbf{1}\left[\mathbf{x} \preceq \mathbf{x}^*_{t,l} \wedge S_{t,l}(\mathbf{z})\right]$
    end if
  end for
  $\{\alpha_{t,l}\}_{l=0}^{L_t}$= CalcCoefs$(X, Y, \{f^*_{t,l}(\mathbf{x}, \mathbf{z})\}_{l=1}^{L_t}, \{a_{t,l}\}_{l=1}^{L_t})$
end for
return $F^*(\mathbf{x}, \mathbf{z}) = \frac{1}{T} \sum_{t=1}^{T} \left(\alpha_{t,0} + \sum_{l=1}^{L_t} \alpha_{t,l} f^*_{t,l}(\mathbf{x}, \mathbf{z})\right)$

form for our rule 'features' $f^*_{t,l}(\mathbf{x}, \mathbf{z}) : \mathcal{X} \times \mathcal{Z} \to \{0, 1\}$:

$$logit(\Pr(y{=}{+}1 \mid \mathbf{x}, \mathbf{z})) = logit(\Pr(y{=}{+}1)) + \sum_{l=1}^{L_t} f^*_{t,l}(\mathbf{x}, \mathbf{z}) \; log\left(\frac{\Pr(f^*_{t,l}(\mathbf{x}, \mathbf{z}){=}1 \mid y{=}{+}1)}{\Pr(f^*_{t,l}(\mathbf{x}, \mathbf{z}){=}1 \mid y{=}{-}1)}\right) \quad (12)$$

Comparing (12) with (11), the unconstrained estimates are:

$$\hat{\alpha}_{t,0} = logit(\Pr(y{=}{+}1)) \quad (13)$$

$$\hat{\alpha}_{t,l} = log\left(\frac{\Pr(f^*_{t,l}(\mathbf{x}, \mathbf{z}){=}1 \mid y{=}{+}1)}{\Pr(f^*_{t,l}(\mathbf{x}, \mathbf{z}){=}1 \mid y{=}{-}1)}\right) \quad (14)$$

Thus we have closed form solutions, and (13) and (14) can be directly calculated by the plug-in estimates. The next step is to ensure sign constraints are respected:

$$\alpha_{t,0} = \hat{\alpha}_{t,0} \quad (15)$$

$$\alpha_{t,l} = \begin{cases} max(\hat{\alpha}_{t,l}, 0), & \text{if } a_{t,l} > 0 \\ min(\hat{\alpha}_{t,l}, 0), & \text{if } a_{t,l} \leq 0 \end{cases} \quad (16)$$

A final step can reduce bias further without compromising monotonicity or significant computation. After $\alpha_{t,l}$ are calculated, the *intercept is recalculated* to minimise empirical loss (we use the Newton method):

$$\alpha^*_{t,0} = \underset{\alpha_{t,0}}{\text{argmin}} \sum_{i=1}^{N} L\left(y_i, \alpha_{t,0} + \sum_{l=1}^{L_t} \alpha_{t,l} f^*_{t,l}(\mathbf{x}_i, \mathbf{z}_i)\right) \quad (17)$$

Table 1: Dataset Summary

	Dataset	Src	Rows	Feats	MT Feats	Classes
FINANCE	German Credit	UCI	1000	24	10	2
	Bankrupt	UCI	250	6	6	2
	Housing	UCI	506	13	3	4
MEDICAL	Pima Indians	UCI	394	8	8	2
	Cleveland Heart	UCI	299	18	12	2
	South African	KEEL	462	9	8	2
	Ljubljana Breast	UCI	277	13	5	2
	Haberman BC	UCI	306	3	3	2
	Wisconscin	UCI	683	9	9	2
EVALUATION	Car	UCI	390	6	4	2
	ERA	Ben-David	1000	4	4	9
	ESL	Ben-David	488	4	4	9
	LEV	Ben-David	1000	4	4	5
	SWD	Ben-David	1000	10	8	4
OTHER	CPU	UCI	209	6	6	4
	Auto Mileage	UCI	392	7	4	2
	Balance	UCI	625	4	4	3

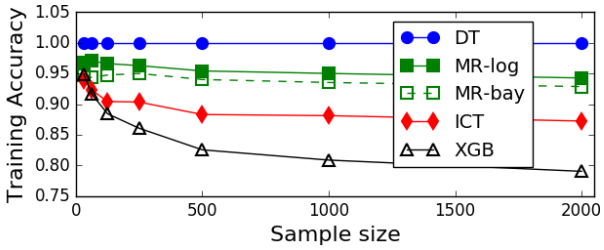

Figure 5: Training acc of globally MT trees (simulated data).

The NB assumption is false because the rules are conditionally *dependent*, so the predicted probabilities unreliable (pushed to 0 or 1). Nonetheless the surprisingly good performance of NB is well known (Rish 2001) and worth considering given the computational advantage.

So far we have created a monotone binary Random Forest. For multi-class ordinal applications we use Kotlowski monotone ensembling as described above for PM-RF.

Simulation

To assess the monotonisation loss associated with the globally monotone trees, a simulated dataset was created:

$$f(\mathbf{x}, \mathbf{z}){=}sign(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3)$$

where $a_1, a_2, a_3 \in [0.0, 1.0]$ were uniform random, a_0 was chosen to give 50:50 class balance, training data $x_1, x_2, x_3, z_1, z_2, z_3 \in [-1.0, 1.0]$ were uniform random, sample sizes between 32 and 2000 and 100 experiments. Noise was introduced by reversing 5% of the class labels. Trees were built to leaf purity with m_{try} of 2.

Figure 5 shows the resulting *in-sample* accuracy. The results are as expected: the standard tree (DT) has zero loss (leaf purity), XGB causes the largest loss, ICT is better, and the proposed MR induces the lowest loss (with logistic slightly better than Bayes). Below we test whether this translates into higher generalised accuracy in RF as hoped.

Table 2: Classifier Performance Measures

Acc Metrics	Kappa								F1 score								Mean Abs Error (MAE)							
	RF	Locally MT			Globally MT				RF	Locally MT			Globally MT				RF	Locally MT			Globally MT			
		MID	FSD	PM	XGM	ICT	MRbay	MRlog		MID	FSD	PM	XGM	ICT	MRbay	MRlog		MID	FSD	PM	XGM	ICT	MRbay	MRlog
Haber	0.141	0.206	0.254	0.237	0.181	0.230	0.307	0.253	0.566	0.589	0.619	0.608	0.567	0.599	0.652	0.620	0.307	0.259	0.254	0.254	0.254	0.246	0.254	0.255
LjubBC	0.234	0.239	0.208	0.260	0.312	0.230	0.265	0.275	0.611	0.602	0.586	0.616	0.644	0.593	0.623	0.633	0.288	0.267	0.281	0.260	0.243	0.262	0.265	0.276
German	0.381	0.353	0.405	0.399	0.307	0.329	0.406	0.423	0.687	0.668	0.700	0.697	0.643	0.655	0.701	0.710	0.237	0.238	0.232	0.234	0.252	0.246	0.234	0.230
SA Hrt	0.263	0.295	0.315	0.292	0.280	0.279	0.336	0.325	0.627	0.641	0.656	0.643	0.631	0.630	0.667	0.660	0.311	0.294	0.299	0.303	0.296	0.294	0.293	0.292
Pima	0.478	0.498	0.498	0.482	0.464	0.462	0.493	0.491	0.737	0.748	0.748	0.740	0.729	0.728	0.746	0.745	0.224	0.217	0.219	0.223	0.225	0.225	0.220	0.221
Cleve	0.613	0.616	0.622	0.617	0.615	0.622	0.638	0.631	0.806	0.807	0.810	0.808	0.806	0.810	0.818	0.815	0.192	0.191	0.187	0.190	0.190	0.187	0.179	0.183
Auto	0.798	0.800	0.809	0.800	0.781	0.794	0.808	0.813	0.899	0.900	0.904	0.900	0.890	0.897	0.904	0.906	0.078	0.077	0.074	0.077	0.084	0.079	0.074	0.073
Car	0.916	0.917	0.916	0.913	0.891	0.900	0.915	0.916	0.958	0.958	0.958	0.957	0.945	0.950	0.957	0.958	0.036	0.035	0.035	0.037	0.046	0.042	0.036	0.036
WBCd	0.936	0.940	0.929	0.937	0.928	0.935	0.924	0.930	0.968	0.970	0.965	0.969	0.964	0.967	0.962	0.965	0.029	0.027	0.032	0.029	0.033	0.030	0.035	0.032
Bnkrpt	0.988	0.988	0.988	0.989	0.985	0.988	0.988	0.990	0.994	0.994	0.994	0.994	0.993	0.994	0.994	0.995	0.006	0.006	0.006	0.006	0.007	0.006	0.006	0.005
CPU	0.756	0.769	0.766	0.764	0.744	0.747	0.764	0.772	0.706	0.723	0.719	0.722	0.702	0.703	0.720	0.726	0.306	0.289	0.293	0.290	0.312	0.311	0.292	0.284
Balance	0.816	0.862	0.851	0.796	0.825	0.872	0.832	0.823	0.602	0.619	0.615	0.650	0.742	0.765	0.690	0.681	0.184	0.138	0.148	0.202	0.173	0.127	0.166	0.176
Housing	0.757	0.759	0.764	0.772	0.759	0.755	0.766	0.774	0.723	0.723	0.728	0.737	0.725	0.719	0.731	0.740	0.306	0.305	0.298	0.283	0.297	0.303	0.289	0.280
ERA	0.362	0.378	0.386	0.376	0.369	0.379	0.395	0.375	0.284	0.289	0.287	0.259	0.243	0.242	0.219	0.249	1.378	1.329	1.287	1.241	1.222	1.210	1.245	1.231
ESL	0.754	0.766	0.769	0.771	0.738	0.778	0.779	0.779	0.458	0.475	0.475	0.477	0.449	0.485	0.490	0.489	0.377	0.349	0.349	0.349	0.377	0.333	0.334	0.339
LEV	0.561	0.565	0.547	0.571	0.497	0.547	0.579	0.577	0.502	0.483	0.475	0.489	0.429	0.470	0.521	0.495	0.438	0.420	0.426	0.414	0.451	0.423	0.411	0.408
SWD	0.428	0.449	0.458	0.462	0.426	0.435	0.480	0.465	0.485	0.463	0.458	0.468	0.449	0.429	0.498	0.475	0.487	0.457	0.451	0.436	0.446	0.438	0.428	0.435
MEAN	0.599	0.612	0.617	0.614	0.594	0.605	**0.628**	0.624	0.683	0.686	0.688	0.690	0.679	0.684	**0.700**	0.698	0.305	0.288	0.286	0.284	0.289	**0.280**	**0.280**	**0.280**
RANK MR-log	-	3.24	3.00	3.24	5.41	4.12	-	**2.00**	-	3.41	3.35	2.88	5.12	4.47	-	**1.76**	-	3.71	3.82	3.18	4.74	3.26	-	**2.29**
RANK MR-bay	-	3.18	2.88	3.12	5.41	4.12	**2.29**	-	-	3.29	3.18	2.71	5.00	4.35	**2.47**	-	-	3.65	3.71	2.94	4.79	3.21	**2.71**	-
RANK All	6.15	3.94	3.85	4.18	6.85	5.44	3.00	**2.59**	5.41	4.21	4.32	3.94	6.53	5.85	3.24	**2.50**	6.59	4.41	4.50	3.97	6.09	4.21	3.41	**2.82**

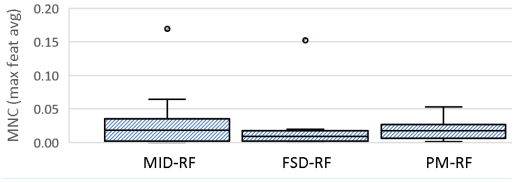

Figure 6: Classifier Nonmonotonicity for local approaches. MNC is proportion of test points with nonmonotone predictions, for least monotone feature (MNC_{max}).

Table 3: Adjusted P-values for Rank Differences (Hommel)

	MR-RF-log vs					MR-RF-bay vs				
	Locally MT			Global MT		Locally MT			Global MT	
	MID	FSD	PM	XGM	ICT	MID	FSD	PM	XGM	ICT
Kappa	0.108	0.119	0.108	0.000	0.004	0.338	0.359	0.359	0.000	0.018
F1	0.021	0.027	0.082	0.000	0.000	0.407	0.543	0.714	0.000	0.013
MAE	0.083	0.056	0.169	0.001	0.169	0.427	0.357	0.714	0.006	0.714

Datasets and Experiments

Datasets. The 17 datasets in Table 1 were used, from UCI (Lichman 2013) and KEEL (Alcala et al. 2010). Missing value rows were removed.

Experiment Design. 100 experiments were performed, with $\frac{2}{3}$ of data randomly selected for training and the remainder as test. Sub-sampling was stratified and the same partitions were used for all classifiers. Monotone features were identified from domain knowledge. 200 unpruned trees were used in each ensemble. The optimal m_{try} was estimated from minimum *out-of-bag* misclassification rate on a parameter sweep of $m_{try} \in \{1, 2, 3, 4, 5, 6, 8, 10, 12, 14\}$.

Measurement of Ordinal Classifier Accuracy. For ordinal classification on imbalanced data, we want to account for *error distance* (i.e. for class 4 a prediction of 1 is worse than 3) and *class imbalance* (so as not to reward random agreement). An ideal measure that accounts for both is *linear weighted Cohen's Kappa* (Ben-David 2008). We also report Mean Absolute Error (MAE) (which accounts for error distance but not random agreement) and F-measure (which accounts for class imbalance but not error distance).

Results and Discussion

Monotonicity Figure 6 shows that the local techniques generally have non-zero monotonicity noncompliance. Generally FSD-RF is best (median $MNC_{max} = 0.9\%$), followed by PM-RF (1.6%) and MID-RF (1.8%). The outlier for FSD-RF and MID-RF is the ERA dataset, where $MNC_{max} \geq 15\%$! PM-RF is better with MNC_{max}=5.2%. ERA is a highly non-monotone dataset. Thus local approaches can retain significant non-monotonicity.

Accuracy We first test our hypothesis that MR-RF can achieve lower loss monotonisation and higher accuracy. Examining Table 2, both MR-RF variants achieve the best rank and mean for all metrics. Statistically we follow Demsar (2006) and use non-parametric rank tests. The Friedman test for equal rank is rejected at $\alpha = 0.05$ ($F_{crit} = 2.33$) with χ^2_F of 32.4, 33.9, and 16.0 for MR-RF-log in Kappa, F-measure and MAE, and 29.7, 23.4 and 13.4 for MR-RF-bay. Proceeding with the Hommel test the adjusted p-values are in Table 3. Both are significantly better than the two globally monotone approaches ($p \ll 0.05$) except ICT-RF in MAE ($p = 0.169$ for MR-RF-log and $p = 0.714$ for MR-RF-bay). Compared to the *local* approaches MR-RF-log achieved close to a significant difference for all (p near 0.10) except PM-RF in MAE ($p = 0.169$), while MR-RF-bay is insignificantly different for all ($p > 0.3$). Thus MR-RF-log is the stronger of the two, but MR-RF-bay nevertheless performs very well. Altogether we have strong support for our thesis, albeit imperfect.

Table 2 also includes standard RF. It can be seen that MR-RF-log improved mean Kappa, F-score and MAE by 0.025, 0.015, and 0.025 respectively, and MR-RF-bay by 0.029,

Table 4: Classifier Solve Times. * MID-RF is pure python. ## XGM-RF estimated as $(C-1)\times$FSD-RF.

Solve t (secs)	MID-RF*	FSD-RF	PM-RF	XGM-RF##	ICT-RF	MR-RF-bay	MR-RF-log
Haber	54.7	**0.2**	0.6	**0.2**	2.4	0.6	2.7
LjubBC	42.7	**0.2**	0.7	**0.2**	2.9	0.7	5.6
German	429.1	**0.3**	1.4	**0.3**	8.6	1.4	15.2
SA Hrt	345.7	**0.2**	0.8	**0.2**	3.0	0.8	4.7
Pima	217.0	**0.2**	0.6	**0.2**	1.9	0.6	3.1
Cleve	72.9	**0.2**	0.7	**0.2**	2.3	0.7	3.6
Auto	97.0	**0.2**	0.5	**0.2**	1.1	0.5	2.0
Car	21.3	**0.2**	0.5	**0.2**	1.4	0.5	2.5
WBCd	22.2	**0.2**	0.6	**0.2**	1.4	0.6	2.2
Bankrupt	3.5	**0.2**	0.4	**0.2**	0.6	0.4	0.6
CPU	53.8	**0.2**	0.8	0.6	3.8	0.8	3.8
Balance	35.8	**0.2**	1.1	0.4	8.1	1.1	7.9
Housing	579.4	**0.2**	1.1	0.7	7.6	1.1	8.0
ERA	66.3	**0.2**	2.7	1.7	7.4	2.7	16.6
ESL	107.4	**0.2**	1.6	1.6	15.6	1.6	11.6
LEV	95.0	**0.2**	1.7	0.9	7.9	1.7	15.7
SWD	127.6	**0.2**	1.7	0.7	9.0	1.7	16.5
MEAN	139.5	**0.2**	1.0	0.5	5.0	1.0	7.2

0.017 and 0.025. These are notable improvements given the impressive performance of standard RF.

Complexity Computation times are in Table 4. Apart from XGM-RF and MID-RF, the techniques have comparable implementations based on `scikit-learn`'s DecisionTreeClassifier, with `cython` optimisations for array operations. XGM-RF and MID-RF were pure python and thus at a significant disadvantage. For XGM-RF it was possible to estimate comparable times by multiplying the FSD-RF time by $(C-1)$. For MID-RF no obvious correction was possible and times should be regarded lightly.

To solve the constrained logistic regression for MR-RF-log we use Newton Conjugate Gradient (`scipy`'s `fmin-tnc`) although many gradient or Newton methods can be used. Complexity can be difficult to estimate for gradient methods but worst case is $O(\tilde{N}^{2.5})$ (Shewchuk et al. 1994). This is solved $C-1$ times per tree, resulting in $O(n_{tree}C\tilde{N}^{2.5})$. MR-RF-bay is $O(n_{tree}Cp\tilde{N}^2)$, but in contrast is closed form, making it highly competitive.

Conclusions and Future Work

This paper compiled the state-of-the-art monotone tree ensemble approaches from both the literature and popular libraries, critiqued their mechanisms, and proposed a lower loss (bias) monotonisation method that works better with the variance reducing mechanism of Random Forest.

The proposed MR-RF-log increased experimental accuracy with globally guaranteed monotonicity and reasonable solution times. The approximate MR-RF-bay was slightly weaker, but still dominated the existing approaches and did so with a very fast closed form solution. The *local* methods had measureable nonmonotonicity, supporting the value of global monotonicity. Compared to standard RF, improvements in all metrics were seen, supporting the value of monotone knowledge purely for the sake of accuracy.

Future work includes developing a bespoke MR-RF-log solver, and automated monotone feature selection.

References

Alcala, J.; Fernandez, A.; Luengo, J.; Derrac, J.; Garcia, S.; Sanchez, L.; and Herrera, F. 2010. Keel data-mining software tool: Data set repository. *Journal of Multiple-Valued Logic and Soft Computing* 17(255-287):11.

Bartley, C.; Liu, W.; and Reynolds, M. 2016. A novel technique for integrating monotone domain knowledge into the random forest classifier. In *Fourteenth Australasian Data Mining Conference (AusDM 2016)*, volume 170 of *CRPIT*. Canberra, Australia: ACS.

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

Ben-David, A. 2008. Comparison of classification accuracy using cohen's weighted kappa. *Expert Systems with Applications* 34(2):825–832.

Bonakdarpour, M.; Chatterjee, S.; Barber, R. F.; and Lafferty, J. 2018. Prediction rule reshaping. *arXiv preprint arXiv:1805.06439*.

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

Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD Int'l Conf on Know. Disc. and Data Mining*, 785–794. ACM.

Demsar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7:1–30.

Fernandez-Delgado, M.; Cernadas, E.; Barro, S.; and Amorim, D. 2014. Do we need hundreds of classifiers to solve real world classification problems? *The Jnl of Machine Learning Research* 15(1):3133–3181.

González, S.; Herrera, F.; and Garcia, S. 2015. Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing* 33(4):367–388.

Hastie, T.; Tibshirani, R.; and J., F. 2009. *The Elements of Statistical Learning*. Springer.

Kotlowski, W. 2008. *Statistical approach to ordinal classification with monotonicity constraints*. Ph.D. Dissertation, Poznan University of Technology Institute of Computing Science.

Lichman, M. 2013. Uci machine learning repository.

Rish, I. 2001. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, 41–46. IBM.

Shewchuk, J. R., et al. 1994. An introduction to the conjugate gradient method without the agonizing pain.

van de Kamp, R.; Feelders, A.; and Barile, N. 2009. Isotonic classification trees. *Advances in Intelligent Data Analysis VIII* 405–416.

You, S.; Ding, D.; Canini, K.; Pfeifer, J.; and Gupta, M. 2017. Deep lattice networks and partial monotonic functions. In *Advances in Neural Information Processing Systems*, 2981–2989.