Generalized Linear Splitting Rules in Decision Forests

by

Tyler M. Tomita

A dissertation submitted to The Johns Hopkins University
in conformity with the requirements for the degree of
Doctor of Philosophy

Baltimore, Maryland
January, 2018

Abstract

Random forests (RFs) is one of the most widely employed machine learning algorithms for general classification tasks due to its speed, ease-of-use, and excellent empirical performance. Recent large-scale comparisons of classification algorithms have concluded that RFs outperform many other classifiers on a variety of datasets. However, the trees in a RF are constructed via a series of recursive axis-aligned splits, rendering the learning procedure sensitive to the orientation of the data. Several studies have proposed "oblique" decision forest methods to address this limitation, which search for good splits that aren't constrained to be axis-aligned. In this work, we explore how properties of the split selection procedure relate to empirical and theoretical performance. We then establish a generalized decision forest framework called Randomer Forests (RerFs), which encompasses RFs and many previously proposed decision forest algorithms as particular instantiations. With this framework in mind, we propose a default instantiation and provide theoretical and experimental evidence motivating its use. Additionally, we demonstrate how our framework can exploit prior domain knowledge to boost performance. Last, we use RerF to identify important biomarkers for ovarian cancer classification and learn

a classifier with high sensitivity and specificity.

Thesis Committee

Primary Readers

Joshua T. Vogelstein (Primary Advisor)
Assistant Professor
Department of Biomedical Engineering
Johns Hopkins University

Randal Burns
Professor
Department of Computer Science
Johns Hopkins University

Carey E. Priebe
Professor
Department of Applied Math and Statistics
Johns Hopkins University

Acknowledgments

First and foremost, I'd like to thank my advisor Dr. Joshua T. Vogelstein. Without Joshua, none of this would have been possible. He has always supported me and made sure that my research was in line with my zone of genius. I have learned a great deal from him about what it means to be a great researcher and mentor.

I would also like to thank my thesis committee members. Dr. Randal Burns for helping to keep the wheels turning and for invaluable discussions about algorithm complexity, speed, and scalability. Dr. Carey E. Priebe for inspiring discussions about statistical theory and pattern recognition. The first lecture I sat in from him may have been the first time I got genuinely excited over statistics.

Furthermore, I would like to thank my peers at Johns Hopkins whom I have worked closely with for their diverse expertise, thought-provoking discussions, and generous feedback on presentations, papers, etc. I would especially like to thank James Browne, who has been integral to the development of RerFs. With all of the great work he has done on the RerF CRAN package, I am confident that RerFs will soon be adopted by many data scientists and researchers world-wide.

Finally, I could never have gotten this far without my friends and family. I thank my parents and sister for all of their love and support. Thanks to Steven Chow for all the fun times in Baltimore and beyond that have kept me sane during the long and winding journey towards my doctorate. Thanks to my dear friends Chris Botros, David Kam, Walter Su, Ashwini Bhat, and Lisa Beppu for their unconditional love, camaraderie, and the unforgettable experiences we have shared over the years.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Classification

The classification problem is briefly described as follows. Let $(X, Y) \sim f_{XY}$, where $X \in \mathbb{R}^p$ is a random real-valued input vector (often called a feature vector), $Y \in \mathcal{Y} = \{c_1, ..., c_K\}$ is an associated categorical response or class label, and f_{XY} is their joint probability distribution, which is generally unknown. In this work, we will use the lower case counterparts (x, y) to denote a particular realization of the random variable pair (X, Y). Given a training set $D_n = \{(X_i, Y_i)\}_1^n \in \mathcal{D}_n \subseteq \mathbb{R}^p \times \mathcal{Y}$, the goal is to learn a classifier $h(\cdot|D_n) : \mathbb{R}^p \to \mathcal{Y}$ that correctly predicts the unobserved class label Y associated with an observed X. Specifically, we would like to minimize $P(h(X|D_n) \neq Y)$, the probability of misclassification.

1.2 Classification Trees

A classification tree (more generally a decision tree) is a data structure representing a series of recursive binary partitions of the training data

into disjoint subspaces. The nodes in a tree are split into two child nodes by maximizing some notion of information gain, which typically reflects the reduction in class impurity of the resulting partitions. A common measure of information gain in decision trees is the decrease in Gini impurity. For a set of observations $\mathcal{S}$, the Gini impurity is defined as: $I(\mathcal{S}) = \sum_{k=1}^{K} |\mathcal{S}| f_k(1 - f_k)$, where $f_k = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \mathbb{I}[y_i = k]$. Let $\theta = (j, \tau)$, where j is an index selecting a dimension and τ is a splitting threshold. Furthermore, let $\mathcal{S}^L(\theta) = \{i : x_i^{(j)} \leq \tau, \forall i \in \mathcal{S}\}$ and $\mathcal{S}^R(\theta) = \{i : x_i^{(j)} > \tau, \forall i \in \mathcal{S}\}$ be the subsets of $\mathcal{S}$ to the left and right of the splitting threshold, respectively. A split is made on a "best" $\theta^* = (j^*, \tau^*)$ via the following optimization:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, I(\mathcal{S}) - I(\mathcal{S}^L(\theta)) - I(\mathcal{S}^R(\theta)). \tag{1.1}$$

In the canonical classification tree algorithm, optimization is made via exhaustive search over θ. Nodes are recursively split until a stopping criteria is reached. Most commonly, the recursion stops when either a maximum tree depth is reached, a minimum number of observations in a node is reached, or a node is completely pure with respect to class label. The result of the tree induction algorithm is a set of split nodes and leaf nodes. The leaf nodes are disjoint partitions of the input space $\mathbb{R}^p$, and each one is associated with a local prediction function. Let l_m be the mth leaf node of a tree, and let $\mathcal{S}(l_m) = \{i : x_i \in l_m \forall i \in [n]\}$ be the subset of the

training data contained in l_m. The local leaf prediction is:

$$h(l_m) = \underset{c_k \in \mathcal{Y}}{\operatorname{argmax}} \sum_{i \in \mathcal{S}(l_m)} \mathbb{I}[y_i = c_k] \tag{1.2}$$

A tree predicts the class label for a new observation x by moving the observation down the tree according to the split functions associated with each split node until a terminal leaf node is reached. Let $m(x)$ be the index of the leaf node that x falls into. Then the tree prediction is $h(l_{m(x)})$.

1.3 Ensemble Learning

In the 1990s, research in ensemble learning, sometimes called multiple classifier systems, began gaining traction. In ensemble learning, predictions are made by aggregating the predictions of multiple different classifiers. Different classifiers can be learned either by using inherently different learning algorithms or by randomly perturbing a single base algorithm. Dietterich lists three reasons motivating the use of ensemble methods [1].

One reason is statistical. Each classifier can be thought of as a hypothesis about the nature of the relationship between the input X and its class label Y. In general, learning algorithms seek a hypothesis that fits the training data well, assuming that such a hypothesis will generalize to unseen examples. Unfortunately, when the number of training samples is sufficiently small, many hypotheses may fit the training data equally well, while not all of these will generalize well to new samples. For example, there are many polynomials of degree two that can fit two points. By

averaging over many hypotheses, ensembles mitigate the risk of learning a hypothesis with poor generalization. A more formal presentation of the statistical motivation for ensemble decision making is Condorcet's Jury Theorem [2], which is restated below.

Theorem 1. *Suppose each of n members of a jury votes on whether a defendant on trial is guilty or not guilty. Let $Y \in \{0, 1\}$ be the true state of the defendant, where 0 indicates not guilty and 1 indicates guilty. Suppose each juror has a probability p of voting in agreement with Y, and assume the votes are independent from one another. Let h_i denote the vote made by the ith juror and $\bar{h}_n = \underset{y \in \{0,1\}}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbb{I}[h_i = y]$ denote the majority vote. If $p > 0.5$, then*

$$\lim_{n \to \infty} P(\bar{h}_n = Y) = 1.$$

Similarly, if $p < 0.5$, then

$$\lim_{n \to \infty} P(\bar{h}_n = Y) = 0.$$

The proof of this theorem is a straightforward computation of the limiting probability of achieving $n/2 + 1$ successes (the definition of majority vote) in n independent Bernoulli trials each having probability $p > 0.5$ of success. In terms of classification, Condorcet's jury theorem says that if multiple independent classifiers all have a probability greater than chance (i.e. random guessing) of classifying correctly, then the probability of the majority vote classifier being correct approaches one as the number of

classifiers approaches infinity. The convergence of the probability of success of the ensemble is dependent on two key conditions: independence of the individual classifiers and the probability of success of each classifier being better than chance (random guessing). Therefore, ensemble learning methods attempt to learn multiple classifiers that are individually strong, yet diverse.

A second motivation for ensemble methods is representational. There are two sources of error in classification. One source arises from choosing the wrong hypothesis, even when the true hypothesis is contained within the hypothesis class. This is related to the statistical reason mentioned above. The second source of error is approximation error, which occurs when the hypothesis class does not contain the true hypothesis [3]. In other words, the learning procedure cannot learn a classifier representative of the true hypothesis. Averaging over hypotheses may expand the hypothesis space and reduce the approximation error.

A third reason motivating the use of ensemble methods is computational. Methods such as decision trees and those that use gradient descent iteratively perform greedy local optimizations. Therefore, there is no guarantee that a globally optimal solution will be found if it exists. By running an algorithm multiple times from different starting points, the learning procedure may have a better chance of finding a solution close to the globally optimal one.

Once a set of classifiers has been constructed, a class label prediction for a new x is typicall made via majority vote of the individual classifier

predictions. Let $\widehat{y}^{(t)}$ be the prediction made by the t^{th} classifier. Then the prediction of the ensemble is :

$$\widehat{y} = \underset{c_k \in \mathcal{Y}}{\operatorname{argmax}} \sum_{t=1}^{T} \mathbb{I}[\widehat{y}^{(t)} = c_k] \tag{1.3}$$

1.4 Random Forests

Much of the focus in development of ensemble methods has been centered around the use of decision trees as base learners. In this work both "tree ensembles" and "decision forests" will be used interchangeably. The popularity of trees in ensemble learning can be attributed to (1) the observation that decision trees are relatively sensitive to the training data and (2) the amenability of tree construction to randomization. Therefore, it is easy to generate a diverse set of classifiers using trees. Breiman took advantage of the first reason, demonstrating that aggregation of trees constructed from bootstrap samples of the training data, dubbed "bagging", was more robust than a single tree classifier [4]. Next, Ho [5] and Amit and Geman [6] independently demonstrated the effectiveness of ensembles of trees constructed on random feature subspaces. Subsequently, in 2001 Brieman introduced Random Forests (RFs), which combined both bagging and the random subspace method. In his version of the random subspace method, optimization of the splitting dimension j at each node is restricted to a random subset of the total p dimensions, rather than over all dimensions. He showed that aggregation of trees constructed with both bagging and random subspaces more often than not leads to generalization performance

much better than that of a single deterministic classification tree [7]. He also proved one of the few theoretical results in ensemble learning, which gives an upper bound on the generalization error of a classifier that depends on the strength and correlation of the individual classifiers. This theorem is restated for completeness.

Theorem 2. *An upper bound for the generalization error L of an ensemble classifier is given by*

$$L \leq \bar{\rho}(1 - s^2)/s^2 \tag{1.4}$$

Here $\bar{\rho}$ is a measure of the correlation of the base classifiers and s is a measure of their average strength (classification performance). While this bound is loose, it suggests that the strength and diversity of the base classifiers are key factors influencing the success of the ensemble.

Due to this result, numerous methods have been proposed for learning trees that are stronger and/or more diverse than those capable of being learned by Breiman's RF algorithm. One prominent feature of RFs that limits both of these is its restriction of splitting hyperplanes to be orthogonal to the coordinate axes of the feature space. Therefore, one of the largest efforts for improving upon RFs has been in relaxing this restriction. The resulting forests are sometimes referred to as "oblique" decision forests, since the splits can be oblique to the coordinate axes. Mathematically, directions along which splits are made are linear combinations of the original p inputs.

Various algorithms have been proposed for constructing oblique forests.

Breiman proposed the Forest-RC algorithm, which constructs d univariate projections, each projection a linear combination of L randomly chosen dimensions [7]. The weights of each projection are independently sampled uniformly over the interval $[-1, 1]$. Heath et al. samples a randomly oriented hyperplane at each split node, then iteratively perturbs the orientation of the hyperplane until a good split is obtained [8]. Rodriguez et al. attempts to find discriminative split directions via PCA [9]. Menze et al. performs supervised learning of linear discriminative models at each node [10]. Blaser et al. uniformly randomly rotates the data prior to inducing each tree. Trees are then learned via the typical axis-aligned procedure on the rotated data [11]. Rainforth and Wood learn discriminative split directions via canonical correlation analysis [12].

1.5 Overview

In this work, we explore the behavior of the axis-aligned RF and oblique decision forests in the context of classification. In Chapter 2, we seek to answer the question, "When presented with a wide variety of synthetic and real-world classification problems, are any of the oblique forest methods actually more robust than RFs? And if so, why?" In Chapter 3, we present a general framework called Randomer Forests (RerFs) which encompasses RFs and all of the oblique forest methods mentioned above. Using insights from Chapter 2, we propose a default instantiation of RerFs. Finally, in Chapter 4, we demonstrate how RerFs can be used to identify salient biomarkers for ovarian cancer classification.

Chapter 2

Robust Decision Forests

2.1 Introduction

Two recent benchmark papers assess the performance of many different classification algorithms on many different datasets [13, 14], and both concluded the same thing: on average, RFs are the best classifier. Due to the large number and wide variety of datasets used in these comparisons, the results suggest that RFs are exceptionally robust to different distributions and representations of the data. Unfortunately, [13] only included one oblique forest in its comparisons, and [14] did not include any. Since one of the primary motivations for developing oblique forest algorithms is to alleviate the sensitivity that the axis-aligned RF algorithm has to orientation of the data, we might wonder whether oblique methods are more robust in practice. In this chapter, we demonstrate that a sparse variant of Breiman's Forest-RC (F-RC) algorithm that rank transforms the data prior to inducing the forest (which we call FRANK) exhibits superior performance and robustness over other decision forest methods across a

wide range of settings. We offer insights into why we observe this.

2.2 Methodology

2.2.1 Simulated Datasets

Many classification problems arise in which both the signal (i.e. the information regarding class membership) is mostly contained in a small subset of dimensions and the optimal split directions are not axis-aligned. We constructed two synthetic datasets with both of these properties (to varying degrees) in order to compare classification performance and training time of different decision forest methods:

Sparse parity is a variation of the noisy parity problem. The noisy parity problem is a multivariate generalization of the noisy XOR problem and is one of the hardest constructed binary classification problems. In the noisy parity problem, a given sample has p features, each of which being uniformly distributed on $(-1, 1)$. Let $s = \sum_{j=1}^{p} \mathbb{I}(x^{(j)}) > 0$, where $\mathbb{I}(x^{(j)} > 0)$ is the indicator that the jth feature of a sample point x has a value greater than zero. A sample's class label is equal to the parity of s. Sparse parity is an adaption of this problem in which the sample's class label is equal to the parity of $s^* = \sum_{j=1}^{p^*} \mathbb{I}(x^{(j)} > 0)$, where $p^* < p$. In other words, this is a variant of the noisy parity problem in which only the first p^* features carry information about the class label.

Trunk is a well-known binary classification problem in which each class is distributed as a p-dimensional multivariate Gaussian with identity covariance matrices [15]. The means of the two classes are $\mu_1 =$

$(1, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}, ..., \frac{1}{\sqrt{p}})$ and $\mu_2 = -\mu_1$. The signal-to-noise ratio of the jth dimension asymptotically decreases to zero with increasing j.

2.2.2 Benchmark Datasets

In addition to the synthetic classification problems, all algorithms were evaluated on 114 of the 121 datasets as described in Fernandez-Delgado et al. [13]. The seven remaining datasets were not used because their high dimensionality and large number of data points rendered the rotation-based classifiers costly in both time and space.

2.2.3 Transformations

We evaluated classifier sensitivity to various transformations of the data. To do so, we consider several different modifications to the data: rotation, scale, affine, and corruption. To rotate the data, we generate rotation matrices uniformly at random and apply them to the data. To scale, we applied a scaling factor sampled from a uniform distribution on the interval $[10^{-5}, 10^5]$ to each dimension. Affine transformations were performed by applying a rotation followed by a scaling. Data was corrupted by randomly selecting 20% of the entries in the data matrix and multiplying each of those entries by a factor uniformly sampled from the set $\{10^2, 10^3, 10^4, 10^5\}$.

2.2.4 Classification Algorithms

The algorithms examined in this study were RF, F-RC, and Blaser et al.'s Random Rotation Random Forest (RR-RF). In order to motivate the choice

of these three algorithms, we will first provide a brief overview of each.

A RF is an ensemble of randomized decision trees constructed using a CART-like procedure [16], where each tree is trained on a bootstrap sample of the data. Each of the trees represents a series of greedy binary recursive partitions of the feature space, where the objective is to create partitions that have minimal class impurity. At each split node of each tree, d features are randomly sampled without replacement from the set of p features. For each of these randomly sampled features, a candidate split is identified by finding the threshold value of that feature that splits the data into the least impure partitions. The best split is chosen from the set of d candidate splits and the partition is generated.

The sole difference between RF and F-RC lies in the sampling of candidate splits. Rather than randomly sampling features at each split node, F-RC defines new features by taking random linear combinations of the original features. A new feature is defined by first specifying a number L of the original features to be combined. L features are randomly selected and added together with coefficients randomly sampled with uniform probability on the interval [-1,1]. d such linear combinations are created, and the best split is found over them.

RR-RF is an oblique decision forest method that adopts random rotations of the feature space prior to inducing each tree in order to further increase the diversity of trees [11]. For each tree, a uniformly random rotation matrix is generated via a QR decomposition on a $p \times p$ matrix

of independently identically distributed samples from a univariate standard normal distribution. The feature space is rotated and the tree is constructed in the same way as RF.

As mentioned previously, the method adopted by RF for constructing trees restricts the splits to be axis-aligned. By uniformly randomly rotating the feature space each time a tree is constructed, RR-RF can construct oblique splits. However, random rotations of the feature space imply that in general, splits will not be sparse (i.e. oblique splits are linear combinations of all features rather than a subset of features). F-RC, on the other hand, controls the sparsity of oblique splits via the parameter L. Therefore, we conjecture that RR-RF will perform increasingly poorly as the ratio of the number of irrelevant features to the number of relevant features becomes larger, while RF and F-RC with a small value for L will be relatively more robust to the increasing presence of irrelevant features. Furthermore, we suspect that linearly combining features will lead to higher sensitivity to data corruption. Therefore we also conjecture that F-RC and RR-RF will be more susceptible to data corruption than is RF, with RR-RF being the most sensitive.

2.2.5 Feature Scaling

Oblique forests are sensitive to scale. In all experiments, we tried three different methods for commensurating features:

1. **Rank Transformation** Let $\{x_i\}_{i=1}^n$ be a set of n real-valued observations and $x_{(1)}, \ldots, x_{(n)}$ be the corresponding order statistics. Suppose

the ith observation x_i corresponds to the jth order statistic $x_{(j)}$. Then the rank transformation of x_i is $r_i = j$. In other words, if x_i is the jth largest value then its corresponding rank is equal to j. In the case of ties, all tied observations are assigned the average of what the ranks would have been had they not been tied. For instance, if two observations x_i and x_k are both the jth largest value, had they not been tied then one would have been the jth largest value and the other the $(j+1)th$ largest value. Therefore $r_i = r_k = j + \frac{1}{2}$. Out-of-sample observations are assigned a rank via linear interpolation if its value falls between two in-sample points. If it happens to be less than all in-sample point then it is assigned a rank of zero, and if it happens to be greater than all n in-sample points then it is assigned a rank of $n+1$.

2. **Percentiles** Observations are scaled to [0,1] by linearly translating each feature so that the smallest value is zero, and then linearly scaling each feature so its largest value is one. Out-of-sample observations are scaled using the in-sample minimum and maximum.

3. **Z-score** For each feature, we subtract the sample mean and divide by the sample standard deviation. Out-of-sample observations are centered and scaled using the in-sample mean and standard deviation.

While all of these methods scale features to proportion, rank-based methods are exceptionally robust to noise and corruption. We suspect

this will be true when applied to the oblique decision forest classifiers as well. In all that follows, the suffixes "(r)", "(n)", and "(z)" after an algorithm name denotes that the algorithm uses the rank transformation, percentile normalization, or z-score, respectively. The one exception is F-RC with rank transformation, which we call FRANK. Note that RF intrinsically operates on the ranks of each feature. Therefore, RF(r) should have identical behavior to RF in all settings.

2.2.6 Training, Parameter Selection, and Testing

In all experiments, data was split into separate training and test sets. For the benchmark datasets, training and test partitions were provided (see [13] for details). In experiments pertaining to the simulated data, every experiment was repeated ten times using different randomly generated training and testing sets. The results reported for the simulated data are the average over the ten trials. Classification algorithms were trained using a range of parameter values. The best model for each algorithm was chosen according to minimum out-of-bag-error on the training set, and predictions were made on the test set.

2.2.7 Classifier Background

Let $D^n = \{(X_i, Y_i) : i \in [n]\}$ be a given dataset, where $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{Y} = \{c_1, \ldots, c_K\}$. A classifier is a function $h : \mathcal{X} \mapsto \mathcal{Y}$ learned from D^n that predicts the class label Y in a new sample pair (X, Y) when only X is observed. Typically, the goal in classification is to learn a classifier $h(\cdot\,|D^n)$

such that $P(h(X\,|D^n) \neq Y)$ is minimized. The optimal classification rule is the Bayes classifier $h^*(x) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}}\, P(Y = y|X = x)$ [3].

In general, the joint distribution of (X, Y) is unknown so that the true class posterior distribution $P(Y|X)$ is unknown as well. However, we can attempt to find good estimates $\hat{P}(Y|X\,; D^n)$ and use those as surrogates, leading to the plug-in estimate of the Bayes classifier $\hat{h}(x\,; D^n) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}}\, \hat{P}(Y = y|X = x\,; D^n)$. Any ensemble classifier that outputs the majority vote of the ensemble, has a natural interpretation as a plug-in estimate of the Bayes classifier. To see this, note that a decision forest classifier $\bar{h}(\cdot\,|D^n)$ is an ensemble of T randomized decision trees, where each tree classifier $h_t(\cdot\,|D^t)$ is trained on a bootstrap sample of the data D^t. The output of $\bar{h}$ is the majority vote. That is, $\bar{h}(x\,|D^n) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}} \sum_{t=1}^{T} \mathbb{I}(h_t(x,\,|D^t) = y) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}} \frac{1}{T} \sum_{t=1}^{T} I(h_t(x,\,|D^t) = y)$, where $\mathbb{I}(h_t(x,\,|D^t) = y)$ is the indicator that the t^{th} tree predicts the class label to be y. The summation in the right hand side of the last equality is the fraction of trees that predict $Y = y$ when $X = x$, which can naturally be interpreted as an estimate of the probability that $Y = y$ given $X = x$. Interpreted in this way, we have $\bar{h}(x\,|D^n) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}}\, \hat{P}(Y = y|X = x\,; D^n)$.

2.3 Results

2.3.1 Comparison of Classification Methods on Synthetic Data

One property that nearly all of the recent oblique forest methods have is that the candidate splits sampled at each node are dense linear combinations of all of the features. This property is undesirable when the information regarding class membership is contained in a small subset of features, which is often the case. We illustrate this phenomenon using the sparse parity dataset with 20 dimensions in Figure 2.1. The top, center, and bottom panels show kernel density estimates of the data projected onto the best first split direction found by a tree in RF, F-RC, and RR-RF, respectively. In this example, RF searched over all 20 dimensions to find the best split. F-RC searched over 8000 random linear combinations of three variables. RR-RF rotated the data and searched over all 20 rotated dimensions. Since every single dimension is uninformative, RF has a zero probability of finding an informative first split direction. RR-RF, which rotates the data via a dense matrix multiplication, has a very small probability (virtually zero in this example) of finding an informative split direction. F-RC, on the other hand, can find an informative split with high probability. This motivates the use of a sparse oblique forest method such as F-RC.

Figure 2.2 depicts both the true class posterior distribution $P(Y = 1|X)$ (top three panels) and estimates of the posteriors for RF, F-RC, FRANK, RR-RF, and RR-RF(r) in three different representations of the sparse parity

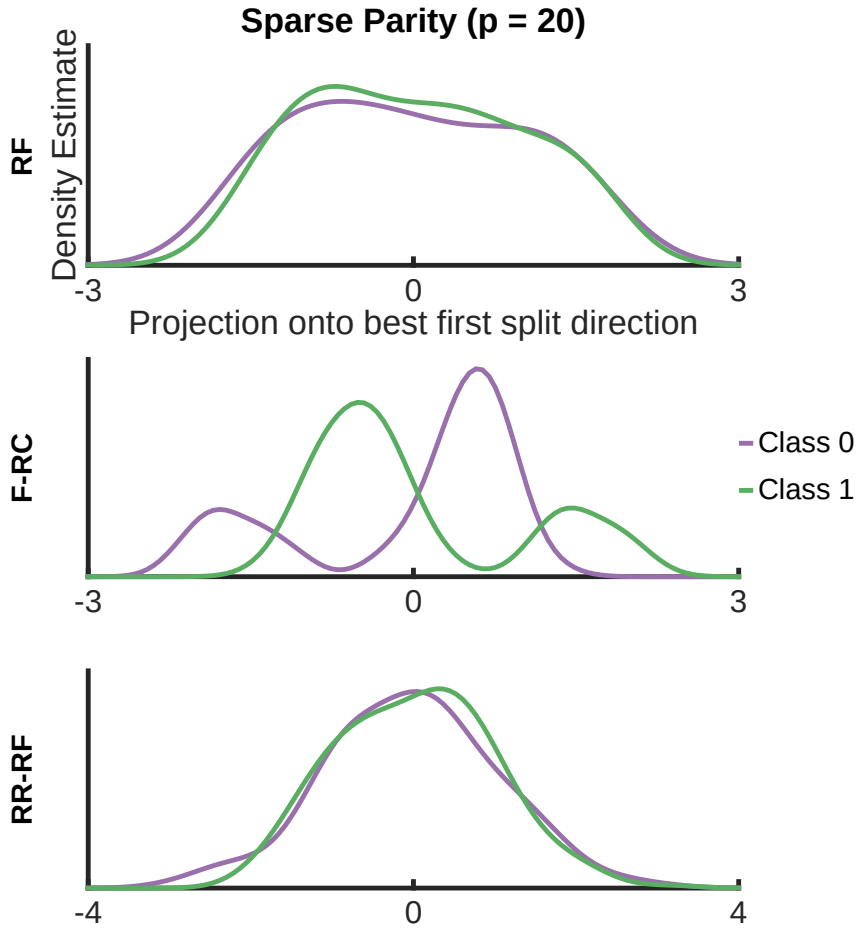

Figure 2.1: Distribution of the sparse parity data when projected onto the best split direction found by RF (top), F-RC (center), and RR-RF (bottom). For RF, all twenty dimensions were evaluated in the search for the best split. For F-RC, 8000 random linear combinations of three variables were evaluated. For RR-RF, the data was randomly rotated and all 20 rotated dimensions were evaluated. Only F-RC has a high probability of finding a good first split.

simulation. The left column is the native sparse parity simulation, the middle is affine-transformed sparse parity, and the right column is corrupted sparse parity. The plot on the very bottom shows the mean pointwise Hellinger distance between the estimates of the posterior probabilities and the true posterior probabilities for each classifier for each of the three

sparse parity representations. The Hellinger distance ranges from zero to one, with a value of zero indicating a perfect estimate of the posterior distribution. For these simulations, $p = 10$, $p^* = 3$, and $n_{train} = 1000$, where p is the total number of dimensions, p^* is the number of relevant dimensions, and n_{train} is the number of training points. All of the posterior maps in Figure 2.2 are shown for the $X_1 - X_2$ plane with $X_3 = -0.5$ and $X_4, \ldots, X_{10} = 0$.

Comparing all algorithms on the raw sparse parity problem shows that F-RC and FRANK give estimates of the posteriors closest to the true posteriors. RF produces poorer estimates because oblique splits allow the feature space to be partitioned more effectively. RR-RF produces poor estimates because the signal is contained in only three of the ten dimensions, so that rotating the data obscures the signal. All methods except for RR-RF(r) are affected by affine transformations. However, FRANK is slightly less affected than F-RC. The right most panels show that both F-RC and RR-RF are vulnerable to data corruption, while FRANK, RR-RF(r), and RF are robust to it. Across all three sparse parity representations, FRANK performs the best.

The top panels of Figure 2.3 show two-dimensional scatter plots from each of the two example simulations (using the first two dimensions). The middle panels show the misclassification rate against the number of dimensions p. The bottom panels show training time against p for all classifiers. The number of trees used for each method in the sparse parity and Trunk simulations were 500 and 1000, respectively. These numbers

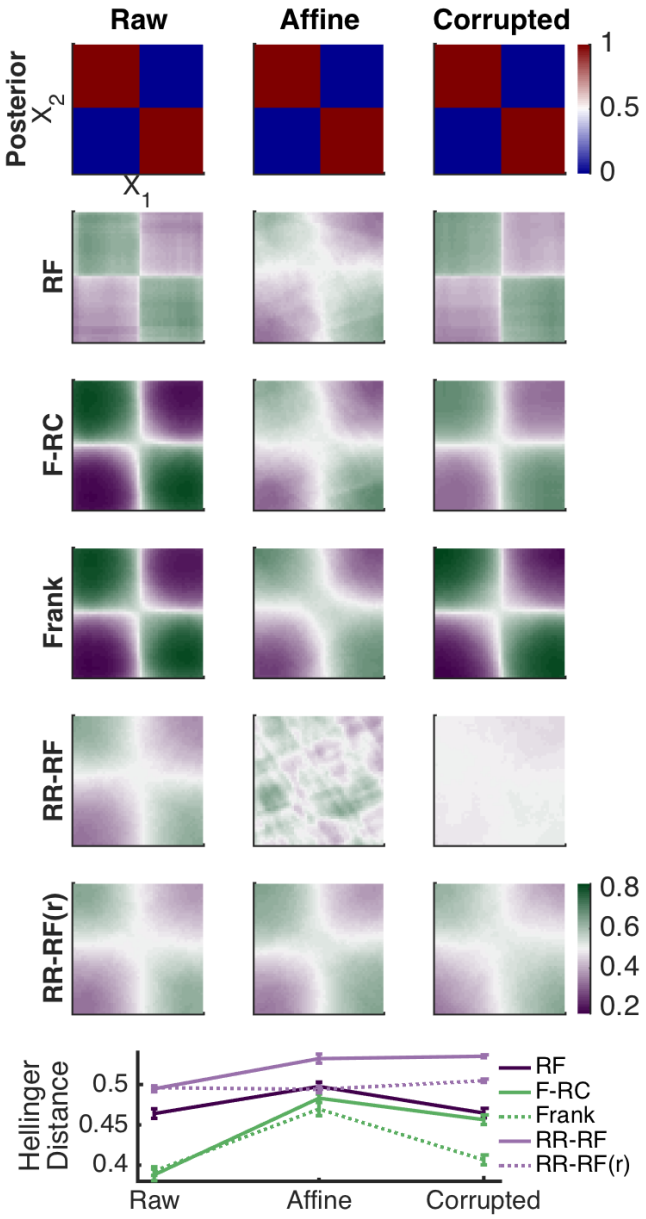

Figure 2.2: Posteriors and classifier estimates of posteriors for the sparse parity problem. Also plotted below are Hellinger distances between estimates and the true posteriors. Oblique forests are especially sensitive to relative scale of predictor variables and data corruption. Rank transforming the data simultaneously robustifies oblique methods against incommensurable features and data corruption.

of trees were empirically determined to be sufficient for convergence of out of bag error for all methods. In all methods, trees were unpruned, and nodes were leaf nodes if they had less than 10 data points. The split criteria was minimum Gini impurity. The only parameter tuned was d, the number of candidate split directions evaluated at each split node. When $p \leq 5$, each classifier was trained for $d = 1, \ldots, p$. When $p > 5$, each classifier was trained for $d = p^{1/4}$, $p^{1/2}$, $p^{3/4}$, and p. Additionally, F-RC and FRANK were trained for $d = p^{3/2}$ and p^2. Note that for RF and RR-RF, d is restricted to be no greater than p by definition. For F-RC, the parameter L, which denotes the number of predictor variables to linearly combine when generating new features, was fixed to two. The reported training time for each algorithm is that corresponding to the classifier using the best value of d. For sparse parity, $n_{train} = 1000$, $n_{test} = 10000$, and classifiers were evaluated for $p = 2$, 5, 10, 20, and 40. The relevant number of features $p^* = min(p, 3)$. For Trunk, $n_{train} = 100$, $n_{test} = 10000$ and classifiers were evaluated for $p = 10$, 50, 100, and 500.

In panel C, both F-RC and FRANK perform as well as or better than the RF and RR-RF variants for all values of p. RF, F-RC, and FRANK perform comparably when $p \leq 5$, but F-RC and FRANK perform better for larger p. As conjectured, the RR-RF variants perform the worst when $p \geq 5$ because they have a difficult time finding good discriminant directions when the signal is contained in a small subset features (the line for RR-RF(r) is directly on top of that for RR-RF). The ability of F-RC and FRANK to perform well compared to the others can be attributed to: 1) the ability to

generate oblique splits and 2) the sparsity imposed on said splits. In panel D, F-RC and FRANK outperform RF and RF(r) for all values of p. This is because linear combinations of a few features can yield a higher signal-to-noise ratio than any single feature. RR-RF exhibits superior performance up to $p = 100$. RR-RF is able to perform better than F-RC and FRANK in these cases because a larger number of features are linearly combined to yield an even higher signal-to-noise ratio. When $p = 500$, classification performances of RR-RF and RR-RF(r) significantly degrade. This can be explained by the fact that when p is large enough, many features contain little information. When this occurs, random rotations of the feature space often result in new rotated features that are less informative. Panel E indicates that training time of F-RC and FRANK can be significantly larger than RF and RR-RF variants on certain problems. The reason for the trend seen in panel E is that the optimal value of d is p^2 for F-RC variants for all values of p in the sparse parity problem. On the other hand, the RF and RR-RF variants cannot have a value of d greater than p. As will be detailed in the next section, the theoretical time complexity of all algorithms is proportional to d. Had d been restricted to be at most p for the F-RC variants, the training times would be comparable to those of the RF and RR-RF variants (not shown). As panel F indicates, training times of RR-RF and RR-RF(r) increase the most quickly with increasing p because of the expensive QR decomposition required for each random rotation. F-RC, on the other hand, is just as fast as RF when $p \leq 100$ and only slightly slower than RF when $p = 500$.

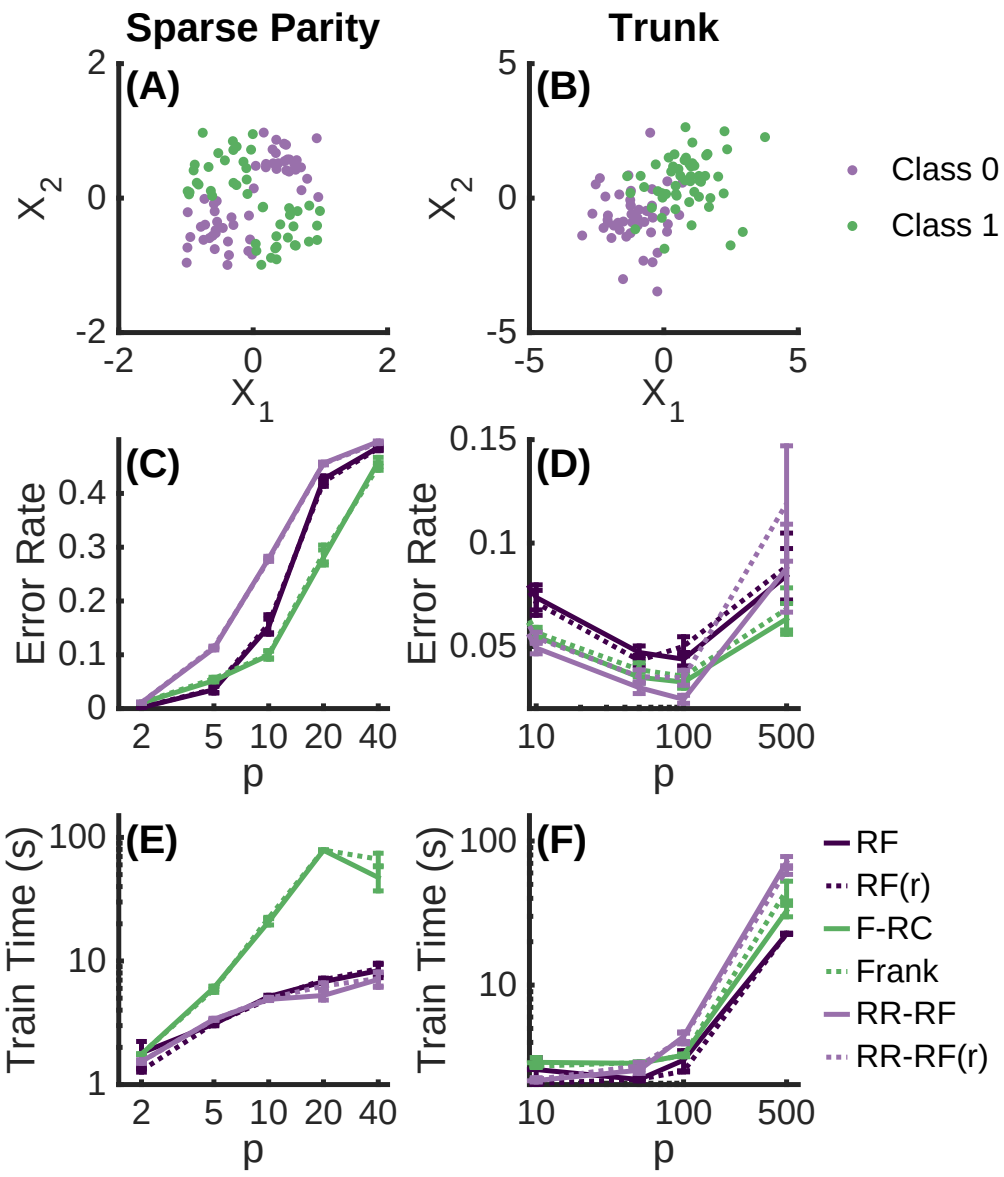

Figure 2.3: Sparse parity (A,C,E) and Trunk (B,D,F) simulations (see section 2.2.1 for details).

2.3.2 Effects of Transformations

Figure 2.4 shows the effect of various transformations applied to the sparse parity (left panels) and Trunk (right panels) problems on classification performance of RF, F-RC, RR-RF, and their rank variants. RF and RF(r) perform slightly worse than the oblique methods on rotated sparse parity. Performance of F-RC and RR-RF are severely degraded when random scaling is applied to sparse parity, and therefore also when affine-transformations are applied. However, the performances of FRANK and RR-RF(r) are unaffected by scale and affine transformations. Performance of RF on sparse parity is minimally affected by data corruption, F-RC is slightly affected, and RR-RF is very affected. FRANK and RR-RF(r), on the other hand, are not noticeably affected by corruption.

Similar trends hold for the Trunk simulations. Note than in panels D, F, and H the error plots of RR-RF are not seen because they are above the limits of the y-axes. The improved performances of FRANK and RR-RF(r) compared to their non-rank variants on the affine and corrupted simulations indicate that rank transformations robustify oblique methods to both the affects of affine transformations and corruption. We also explored whether other feature scaling methods had a similar robustifying effect on F-RC and RR-RF (see section 2.2.5 for details). Figure 2.5 suggests that both normalization and z-scoring have a similar robustifying effect as rank transformations on scaling and affine transformations but do not help with data corruption.

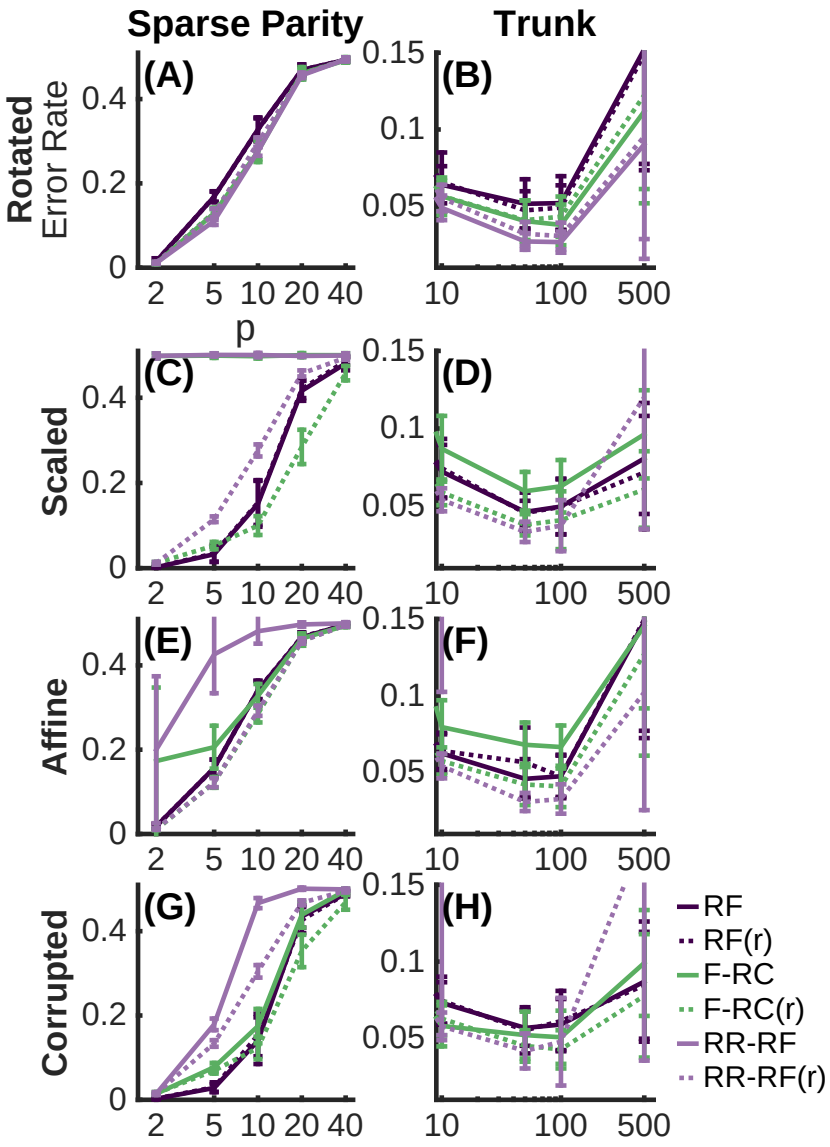

Figure 2.4: The effects of different transformations applied to the sparse parity (left column) and Trunk (right column) simulations on classification performance (see section 2.2 for details). Specifically, we consider rotations, scalings, affine transformations, and corruptions.

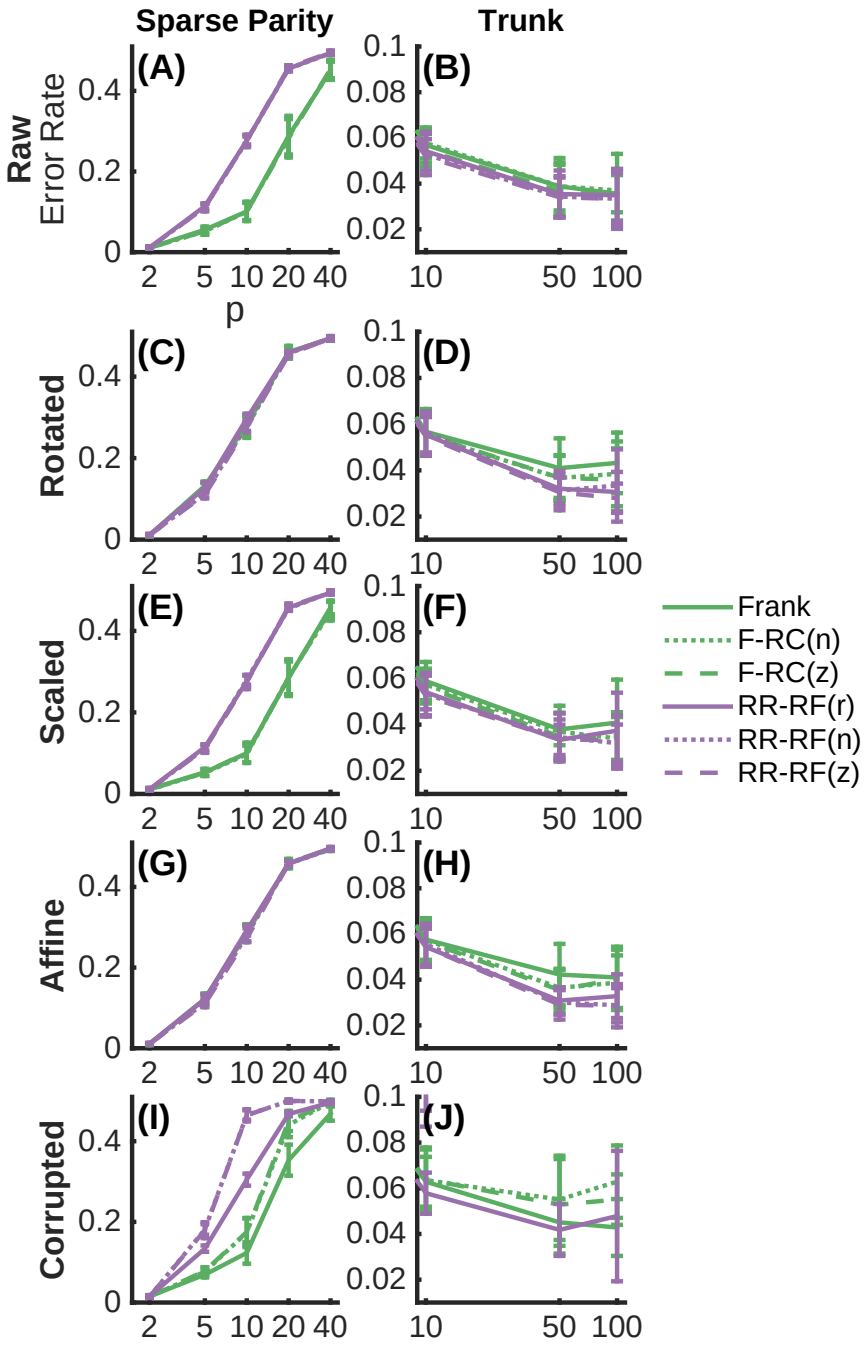

Figure 2.5: Comparison of feature scaling methods on the simulated datasets.

2.3.3 Benchmark Data

Next we compared performance of the RF, F-RC, and RR-RF variants on 114 benchmark datasets (refer to section 2.2.2 for details). As in the previous section, transformations were applied to the datasets to observe their effects on performance of the six classification methods. We used the same values of d as in the simulations. The number of trees used in each algorithm was 1000 for datasets having at most 1000 data points and 500 for datasets having greater than 1000 data points. These numbers of trees were empirically seen to be sufficient for convergence of out-of-bag error. Convergence plots for each benchmark dataset are available here.

For each benchmark dataset, for each algorithm, error was subtracted by that of RF and normalized by the chance probability of error. Therefore, a negative value indicates that an algorithm had a lower error rate than RF. Chance probability of error for a particular dataset is defined as the probability of error if the most populous class was always predicted for a randomly sampled data point. These normalized relative errors were then binned and the counts in each bin were computed. Histograms showing the counts in each bin are shown in Figure 2.6. The y-axis represents the bins. Color indicates how many times the normalized relative error of an algorithm fell into a particular bin. For instance, the figure shows that FRANK has a normalized error 0.05 to 0.10 less than that of RF on approximately 15 datasets. The "0 to 0" bin indicates the number of times the normalized relative error was exactly 0. Table 2.1 summarizes which methods perform significantly better or worse than RF across the

	Raw	Rotated	Scaled	Affine	Corrupted
RF(r)					
F − RC	+	+		−	−
Frank	+	+		+	
RR − RF	−	+	−	−	−
RR − RF(r)	−	+	−	+	−

Table 2.1: Summary of performance relative to RF on each of the benchmark settings. (+) indicates method performed significantly better than RF ($p < 0.05$) and (-) indicates method performed significantly worse than RF ($p < 0.05$)

different transformation settings according to one-sided Wilcoxon signed-rank tests at a significance level of 0.05. Both F-RC variants perform better than RF on the raw datasets, while the RR-RF variants perform worse. As expected, all oblique methods perform better than RF on the rotated datasets. Both RR-RF variants perform worse than RF on the scaled datasets, with RR-RF(r) performing slightly better than RR-RF. F-RC and RR-RF perform worse than RF on the affine datasets, while FRANK and RR-RF(r) perform better. All oblique methods perform worse than RF on the corrupted datasets with the exception of FRANK. Figure 2.7 shows that unlike percentile normalization and z-scoring, rank transformations helps against data corruption. Overall, FRANK is the most robust method.

2.4 Discussion

Incorporating random rotations into RF in the way that RR-RF does is clearly a double-edged sword. While RR-RF is the only method of the three that is completely invariant to the orientation of the data, the sparse parity simulations demonstrate that it performs poorly when the signal is

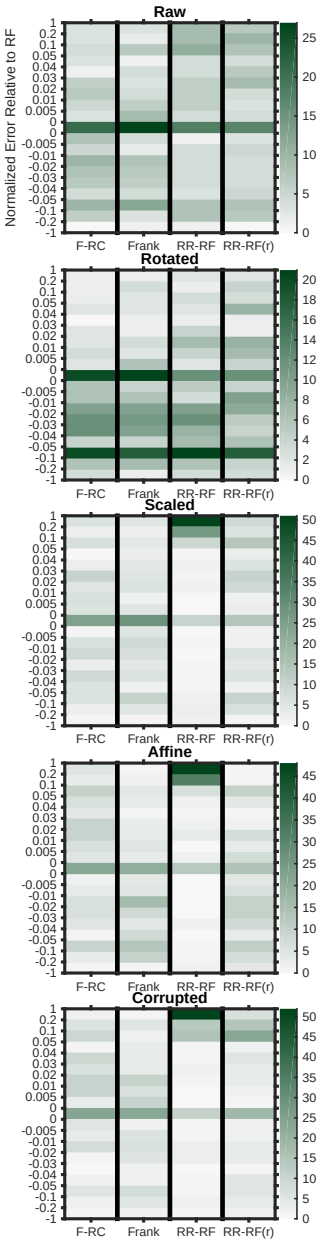

Figure 2.6: Histograms of error relative to RF normalized by chance probability of error on 114 benchmark datasets with various transformations applied. A negative normalized relative error indicates better performance than RF, while a positive value indicates worse performance. Bin edges are indicated by ticks on the y-axis. Color indicates the absolute frequency in a particular bin.

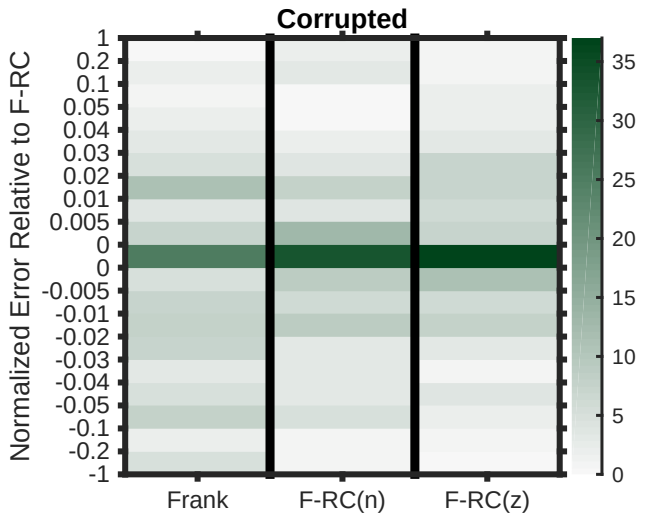

Figure 2.7: Histograms of error relative to F-RC normalized by chance probability of error on the corrupted benchmark datasets. FRANKperforms better than F-RC more often than does F-RC(n) or F-RC(z), suggesting that rank transformations are better than perentile normalization or z-scoring at mitigating the effects of data corruption.

concentrated in a small subset of the features. The simulations indicate that F-RC and RF do not suffer from this problem. These results may likely be explained by the "Bet on sparsity" principal proposed by Friedman, Hastie, and Tibshirani [17]. This principle states that no method - dense or sparse - will perform well in high dimensions when the truth is dense. On the other hand, when the truth is sparse, then sparse methods will perform well while dense methods will still perform poorly. RF is the most sparse because each split is made on a single feature. The version of F-RC in this study is relatively sparse because splits are made on linear combinations of just two features. RR-RF, on the other hand, is dense because the features in the rotated feature space are each linear combinations of all of the features in the original space. RR-RF indeed performs the best up

to 100 dimensions on the Trunk problem, which has information in all of the dimensions. However, its performance is only marginally better than F-RC and is in fact worse when $p = 500$. We suspect that the dense nature of RR-RF is also the reason that it is the most sensitive to scale, affine transformations, and data corruptions. Overall, our results suggest that the cost of using random rotations tends to outweigh the benefits.

There are many ways to deal with sensitivity to scale. Most often, incommensurate features are either normalized to $[0, 1]$ or transformed to z-scores rather than rank transforming. Statistical procedures based on ranks have shown to possess exceptional robustness to noise in several different contexts [18–21]. Here we have shown that rank transformations deal with incommensurability equally as well as other scaling methods, and has the additional benefit of robustifying oblique methods to data corruption.

Since the simulated datasets have known distributions, we can be certain in our characterizations of their underlying data structures. While this is not the case with the benchmark datasets, our experiments allow us to conjecture about the nature of these datasets. First, the fact that F-RC and FRANK tend to outperform RF and RF(r) on the raw benchmark datasets suggests that the intrinsic discriminant boundaries between classes are often not axis-aligned. Second, the fact that RR-RF and RR-RF(r) perform substantially worse than the other methods on the raw benchmark datasets suggests that the information regarding class membership is often concentrated in a small subset of features.

2.5 Conclusion

In this work, we explored classification performance of several decision forest methods, and in the process, identified an oblique forest method that is robust to the representation of the data and has relatively low computational complexity. While no single method dominates on every classification problem, we observe that FRANK exhibits superior robustness to the representation of the data and dominates more often than any of the other methods evaluated. The RR-RF variants, on the other hand, tend to lose more often than the other methods due to the dense nature of splits.

The contribution of this chapter is two-fold. First, it augments the large-scale empirical study conducted by FD14 [13]. They concluded RF to be the overall best classification method. On these same datasets, we demonstrate that both F-RC and FRANK tend to have statistically significantly better classification performance than RF. Second, we provide a novel analysis in which we investigated the conditions under which various decision forest methods dominate. As this study has demonstrated, different algorithms vary in their sensitivity to the representation of the data. In the real world, it is often the case that the optimal representation is not the one given. Therefore, a procedure that is robust to the representation is desirable. While our studies indicate FRANK to be the most robust of the procedures analyzed, there is much room for improvement. For instance, the sparsity of splits in FRANK, defined by the parameter L, is fixed across all nodes and trees. With complex data structures, it might be the case that randomizing the sparsity across nodes may yield better performance. Additionally,

FRANK naively samples the variables to combine as well as the weighting of each variable. The search space using this approach can be vast. There may exist more efficient ways to narrow down or guide the search.

Chapter 3

Randomer Forests

3.1 Introduction

In this chapter we develop novel procedures for learning decision forests. To do so, we first state a generalized forest framework, Randomer Forests (RerFs), which includes RFs and all oblique methods as particular instantiations. The generalization stems from the fact that RFs and oblique forests all evaluate a set of randomly oriented univariate projections at each split node, the only difference between the methods being the distribution from which the projections are sampled. This framework provides a lens that enables us to propose our own instantiation. We demonstrate that our method is robust over a wide range of datasets. Furthermore, we investigate why and when a particular algorithm within this framework dominates, and highlight which aspects of the sampling distribution of split projections are important to consider. We then show how a sampling distribution can be chosen to exploit prior domain knowledge. Lastly, we show that our proposed method maintains a time complexity similar to that of RFs, and

offer a fast parallelized R implementation available on the Comprehensive R Archive Network (CRAN) (https://cran.r-project.org/web/packages/rerf/).

3.2 Methods

3.2.1 Randomer Forests (RerF) Algorithm

We propose a general decision forest framework called Randomer Forest (RerF), which encompasses any forest algorithm that recursively partitions the data via arbitrarily oriented hyperplanes. RFs as well as all of the previously mentioned oblique methods are particular instantiations RerFs. The key idea of RerFs is that at each split node of the tree, we have a set of predictor data points, $\bar{X} = \{X_s\}_{s \in \mathcal{S}_i^l} \in \mathbb{R}^{p \times S_i^l}$, where $S_i^l = |\mathcal{S}_i^l|$ is the cardinality of the set of predictor data points at the i^{th} node of the l^{th} tree. We sample a matrix $A \sim f_A$, where $A \in \mathbb{R}^{p \times d}$, possibly in a data dependent fashion, which we use to randomly project the predictor matrix $\bar{X}$, yielding $\widetilde{X} = A^\mathsf{T} X \in \mathbb{R}^{d \times S_i^l}$, where d is the dimensionality of the projected space. See Algorithm 1 for details. Table 3.1 summarizes the particular form of f_A adopted by various decision forest algorithms.

While the best f_A is dataset dependent, it is unreasonable and/or undesirable to try more than a handful of different instantiations. Therefore, for general purpose classification we advocate for a default f_A that addresses the following issues:

1. While RF empirically performs well in many settings, it is quite

Algorithm	f_A	Ref
RF	Let $\{j_k\}_{k=1}^{d}$ be a set of indices obtained by sampling without replacement from $\{1,\ldots,p\}$. Let $\mathbf{e}_i$ be the ith column of the $p \times p$ identity matrix. Then $A = [\mathbf{e}_{j_1}\ \mathbf{e}_{j_2}\ \cdots\ \mathbf{e}_{j_d}]$.	[7]
F-RC	Let a_{ij} denote the element corresponding to the ith row and jth column of A. For each $j \in \{1,\ldots,d\}$, let S_j^L be a set of L indices obtained by sampling without replacement from $\{1,\ldots,p\}$. Then $a_{ij} \overset{iid}{\sim} U(-1,1)\ \forall i \in S_j^L$, and $a_{ij} = 0\ \forall i \notin S_j^L$.	[7]
RR-RF	Let R be a $p \times p$ uniformly random rotation matrix. Then $A = RA_{RF}$, where A_{RF} is a random matrix sampled from the f_A defined for RF above.	[11]
Rot-For	Let $X \in \mathbb{R}^{n \times p}$ be the input data matrix at a split node. Let $S_j\ \forall j \in \{1,\ldots,K\}$ be uniformly random disjoint subsets of the column indices $\{1,\ldots,p\}$, and let each $I'_j\ \forall j \in \{1,\ldots,K\}$ be a copy of the identity matrix such that the columns indexed by S_j are zeroed out. Then $A = [PCA(XI'_1)\ PCA(XI'_2)\ \cdots PCA(XI'_K)]$, where $PCA(\cdot)$ returns the matrix of principal components having nonzero eigenvalues.	[9]
O-RF	Let $X \in \mathbb{R}^{n \times p}$ be the input data matrix at a split node and $\mathbf{y} \in \{0,1\}^{n \times 1}$ be corresponding class labels. Let I' be a copy of the $p \times p$ identity matrix with L columns – chosen at random – zeroed out. Then $A = RIDGE(XI', \mathbf{y})$, where $RIDGE(\cdot)$ returns the vector projection found by ridge logistic regression.	[10]

Table 3.1: A summary of the random projection matrix distribution f_A adopted by previously proposed decision forest algorithms. Note that this list is not exhaustive. We use the notation $[A_1\ A_2\ A_3]$ to denote a matrix defined by the column-wise concatenation of the matrices (or column vectors) A_1, A_2, and A_3

restrictive in that candidate splits evaluated at each node are constrained to be axis-aligned. Often, linear interactions of features are more informative than individual features, in which case oblique splits would be desired.

2. Robustness to irrelevant dimensions. In the previous chapter, we

Pseudocode 1 Pseudocode for learning a Randomer Forest decision tree on a dataset.

Input: (1) $\mathcal{D}_n$: training data (2) d: dimensionality of the projected space, (3) $f_{\mathbf{A}}$: distribution of the random projection matrix, (4) Θ: set of split eligibility criteria

Output: A RerF decision tree T

function $T = \text{GROWTREE}(\mathbf{X}, \mathbf{y}, f_{\mathbf{A}}, \Theta)$
  $c = 1$ ▷ c is the current node index
  $M = 1$ ▷ M is the number of nodes currently existing
  $S^{(1)} = \text{bootstrap}(\{1, ..., n\})$ ▷ $S^{(c)}$ is the indices of the observations at node c
  while $c < M + 1$ **do** ▷ visit each of the existing nodes
    $(\mathbf{X}', \mathbf{y}') = (\mathbf{x}_i, y_i)_{i \in S^{(c)}}$ ▷ data at the current node
    for $k = 1, \ldots, K$ **do** $n_k^{(c)} = \sum_{i \in S^{(c)}} I[y_i = k]$ **end for** ▷ class counts
    if Θ satisfied **then** ▷ do we split this node?
      $\mathbf{A} \sim f_{\mathbf{A}}$ ▷ sample random matrix
      $\widetilde{\mathbf{X}} = \mathbf{A}^T \mathbf{X}' = (\widetilde{\mathbf{x}}_i)_{i \in S^{(c)}}$ ▷ random projection into new feature space
      $(j^*, \tau^{*(c)}) = \text{findbestsplit}(\widetilde{\mathbf{X}}, \mathbf{y}')$ ▷ Algorithm 2 (see supplementary material)
      $\mathbf{a}^{*(c)} = \mathbf{a}_{j^*}$ ▷ sparse split projection of current node
      $S^{(M+1)} = \{i : \mathbf{a}^{*(c)} \cdot \widetilde{\mathbf{x}}_i \leq \tau^{*(c)} \quad \forall i \in S^{(c)}\}$ ▷ assign to left child node
      $S^{(M+2)} = \{i : \mathbf{a}^{*(c)} \cdot \widetilde{\mathbf{x}}_i > \tau^{*(c)} \quad \forall i \in S^{(c)}\}$ ▷ assign to right child node
      $\kappa^{(c)} = \{M + 1, M + 2\}$ ▷ node indices of children of current node
      $M = M + 2$ ▷ update the number of nodes that exist
    else
      $(\mathbf{a}^{*(c)}, \tau^{*(c)}, \kappa^{*(c)}) = \text{NULL}$
    end if
    $c = c + 1$ ▷ move to next node
  end while
  return $(S^{(1)}, \{\mathbf{a}^{*(c)}, \tau^{*(c)}, \kappa^{(c)}, \{n_k^{(c)}\}_{k \in \mathcal{Y}}\}_{c=1}^{m-1})$
end function

demonstrated that F-RC empirically performed much better than RR-RF. The main difference between the two is that F-RC samples

directions defined by *sparse* linear combinations of inputs, whereas in RR-RF split directions are *dense* linear combinations. Thus, it seems that proper control of the sparsity of the random matrix A is necessary when irrelevant dimensions are prevalent.

3. Often times models need to be interpretable in addition to being accurate. While RF models can be complicated, suitable measures have been proposed to assess the relative contribution (importance) of each feature. This becomes prohibitive to compute for oblique forests if the space of possible split projections is not sufficiently constrained.

4. Existing oblique decision forest algorithms involve expensive computations to identify and select splits, rendering them less space and time efficient than RF.

Figure 3.1 offers geometric intuition of how RerF addresses the first two issues above. A synthetic classification problem was constructed in which two classes lie in parallel hyperplanes in 20 dimensions. Only the first two dimensions are informative of the class label. Furthermore, neither one of the first two dimensions are individually informative. Specifically, class 0 is uniformly distributed on the noisy hyperplane $X_1 + X_2 = -\epsilon$, where $\epsilon \sim N(0.1, 0.01)$ is a small amount of independent Gaussian noise. Each of $X_1, X_3, \ldots, X_{20} \overset{iid}{\sim} U(-0.5, 0.5)$, and X_2 is distributed according to the (noisy) hyperplane constraint. The distribution of class 1 is the same as that for class 0, except that the hyperplane is defined as $X_1 + X_2 = +\epsilon$. The best projections at the root node found by RF, RerF, and RR-RF were compared. For RerF we set λ to $1/20$. RF cannot find a good

projection because no single dimension is informative. RR-RF samples projections uniformly over the 20-dimensional hypersphere, and thus has an infinitesimal probability of sampling a projection sufficiently close to the optimal projection $(1, 1, 0, \ldots, 0)$. RerF is the only one that can find a good projection. Indeed, if the example was instead constructed so that all 20 dimensions were used to define the parallel hyperplanes (i.e. all 20 are informative but marginally uninformative), then RerF would struggle because of the sparsity constraint on the distribution of A. However, RR-RF, which does not have this constraint, would still struggle because any projection it samples is very likely to be nearly orthogonal to the optimal projection (see A for a more thorough explanation). This further supports the adoption of our default f_A for RerF.

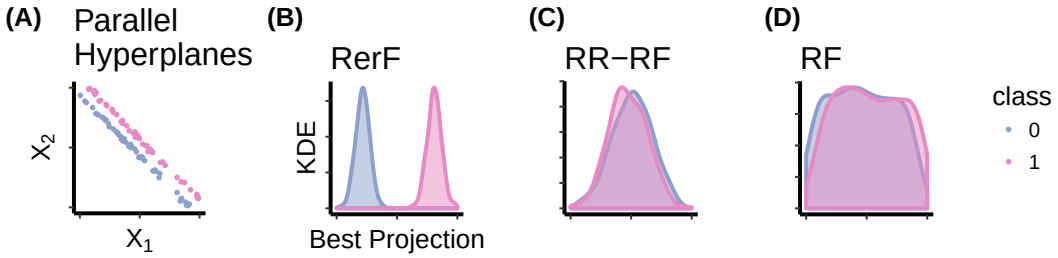

Figure 3.1: Comparison of the best projections found by RerF, RR-RF, and RF on the 20-dimensional parallel hyperplanes synthetic dataset.

3.2.2 Synthetic Datasets

In the sections that follow, we perform a variety of experiments on three carefully constructed synthetic classification problems. These constructions are chosen in order to highlight various properties of different algorithms and gain insight into their behavior. The three problems are as

follows:

Sparse Parity is a multivariate generalization of the noisy XOR problem. It is a p-dimensional two-class problem in which the class label is 0 if the number of dimensions having positive values amongst the first $p^* < p$ dimensions is even and 1 otherwise. Thus, only the first p^* dimensions carry information about the class label, and no individual dimension contains any information. Specifically, let $X = (X_1, \ldots, X_p)$ be a p-dimensional feature vector, where each $X_1, \ldots, X_p \overset{iid}{\sim} U(-1, 1)$. Furthermore, let $S = \sum_{j=1}^{p^*} \mathbb{I}(X_j > 0)$, where $p^* < p$ and $\mathbb{I}(X_j > 0)$ is the indicator that the jth feature of a sample point x has a value greater than zero. A sample's class label Y is equal to the parity of S. That is, $Y = odd(S)$, where odd returns 1 if its argument is odd and 0 otherwise. The Bayes optimal decision boundary for this problem is a union of hyperplanes aligned along the first three dimensions. For the experiments presented in the following sections, $p^* = 3$ and $p = 20$. Figure 3.2 (panels A and B) show cross-sections of the first two dimensions taken at two different locations along the third dimension.

Orthant is a multi-class problem in which the class label is determined by the orthant that a datapoint resides in. A key characteristic of this problem is that the individual dimensions are strongly and equally informative. An orthant in $\mathbb{R}^p$ is a generalization of a quadrant in $\mathbb{R}^2$. In other words, it is a subset of $\mathbb{R}^p$ defined by constraining each of the p coordinates to be positive or negative. For instance, in $\mathbb{R}^2$, there are four such subsets: (X_1, X_2) can either be in 1) $\mathbb{R}^+ \times \mathbb{R}^+$, 2) $\mathbb{R}^- \times \mathbb{R}^+$, 3) $\mathbb{R}^- \times \mathbb{R}^-$, or 4) $\mathbb{R}^+ \times \mathbb{R}^-$.

Note that the number of orthants in p dimensions is 2^p. Specifically for our experiments, we sample each $X_1, \ldots, X_p \overset{iid}{\sim} U(-1,1)$. Associate a unique integer index from 1 to 2^p with each orthant, and let $O(X)$ be the index of the orthant that X belongs to. The class label is $Y = O(X)$. The Bayes optimal decision boundary in this setting is a union of hyperplanes aligned along each of the p dimensions. We set $p = 6$ in the following experiments. Figure 3.2 (panels D and E) show cross-sections of the first two dimensions taken at two different locations along the third dimension.

Trunk is a balanced two-class problem in which each class is distributed as a p-dimensional multivariate Gaussian with identity covariance matrices [15]. Every dimension is informative, but each subsequent dimension is less informative than the last. The means of class 1 and 0 are $\mu_1 = (1, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}, ..., \frac{1}{\sqrt{p}})$ and $\mu_0 = -\mu_1$, respectively. The Bayes optimal decision boundary is the hyperplane $(\mu_1 - \mu_0) \cdot X = 0$. We set $p = 10$ in the following experiments.

3.2.3 Benchmark Datasets

In addition to the synthetic datasets, comparisons of RF, RerF, and F-RC was made on 105 benchmark datasets from the UCI machine learning repository. These datasets are most of the datasets used in Fernandez-Delgado et al. [13]. However, rather than using the preprocessed datasets provided by them, we independently preprocessed them. The motivation for our preprocessing steps was to try to minimize the noise concomitant with real-world data. Details of the preprocessing steps are described as

follows:

1. **Removal of nonsensical features**. Some features, such as unique sample identifiers, were removed because they have no relevance to the classification problem.

2. **Imputation of missing values**. The R randomForest package was used to impute missing values. This method was chosen because it is nonparametric and is one of the few imputation methods that can natively impute missing categorical entries.

3. **One-hot-encoding categorical features**. Most classifiers cannot handle categorical data natively. In one-hot encoding, sometimes called one-of-K encoding, a categorical feature that can assume K possible categories is expanded into K binary features, where each binary feature corresponds to presence or absence of a category. If the ith category is observed for a sample, then the ith binary feature is assigned a value of one, while the remaining $K-1$ features are assigned a value of zero.

4. **Integer encoding of ordinal features**. Categorical features having order to them, such as "cold", "luke-warm", and "hot", were numerically encoded to respect this ordering with integers starting from 1.

5. **Standardization of the format**. Lastly, all datasets were stored as CSV files, with rows representing observations and columns representing features. The class labels were placed as the last column.

6. **Five-fold paritioning**. Each dataset was randomly divided into five partitions for five-fold cross validation. Partitions preserved the relative class frequencies as much as possible.

3.3 Results

3.3.1 Illustrative Synthetic Data Experiments

We compared error rates of RF, RerF, and F-RC on the sparse parity and orthant problems over a range of training set sizes n_{train}. Error rates were estimated by taking a random sample of size n_{train}, training the classifiers, and computing the fraction misclassified in a test set of size 10000. This was repeated ten times for each value of n_{train}. The reported error rate is the mean over the ten repeated experiments. The number of trees used for each algorithm was 1000. This number of trees was empirically determined to be sufficient for convergence of out-of-bag error for all methods. In all methods, trees were unpruned and fully grown (i.e. nodes were split until pure). The split objective was to maximize the reduction in Gini impurity. Two hyperparameters were tuned via minimization of out-of-bag error. The first parameter tuned was d, the number of candidate split directions evaluated at each split node. Each algorithm was trained for $d = p^{1/4}$, $p^{1/2}$, $p^{3/4}$, and p. Additionally, RerF and F-RC were trained for $d = p^2$. Note that for RF d is restricted to be no greater than p by definition. The second hyperparameter tuned was λ, the average sparsity of univariate projections sampled at each split node. Note, for RF λ is fixed to $1/p$ by definition, since the univariate projections are constrained to be along one

of the coordinate axes of the data.

Figure 3.2 (panels C and F) indicate that RerF performs as well as or better than the other algorithms on both the sparse parity and orthant problems, respectively. RF performs relatively poorly on the sparse parity problem. Although the optimal decision boundary is a union of axis-aligned hyperplanes, each dimension is completely non-informative on its own. Since axis-aligned partitions are chosen one-at-a-time in a greedy fashion, the trees in RF struggle to learn the correct partitioning. On the other hand, oblique splits are informative, which substantially helps the generalization ability of RerF and F-RC. While F-RC performs well on the sparse parity problem, it performs much worse than RF and RerF on the orthant problem. We highlight that on the orthant problem, in which RF is is designed to do exceptionally well on, RerF performs just as well.

A key difference between the default distribution of RerF and F-RC is that F-RC requires specification of a hyperparameter that fixes the sparsity of the sampled univariate projections. RerF on the other hand, requires specification of a sparsity on the entire random matrix A, and hence, only an *average* sparsity on the univariate projections. In other words, RerF induces a distribution on the sparsity of univariate projections, whereas F-RC does not. An implication of this is that if the Bayes optimal decision boundary is locally sparse, misspecification of the hyperparameter controlling the sparsity of A may be more detrimental to F-RC than RerF. Therefore, we examined the sensitivity of RerF and F-RC to the sparsity hyperparameter on the synthetic datasets previously described. Let λ be

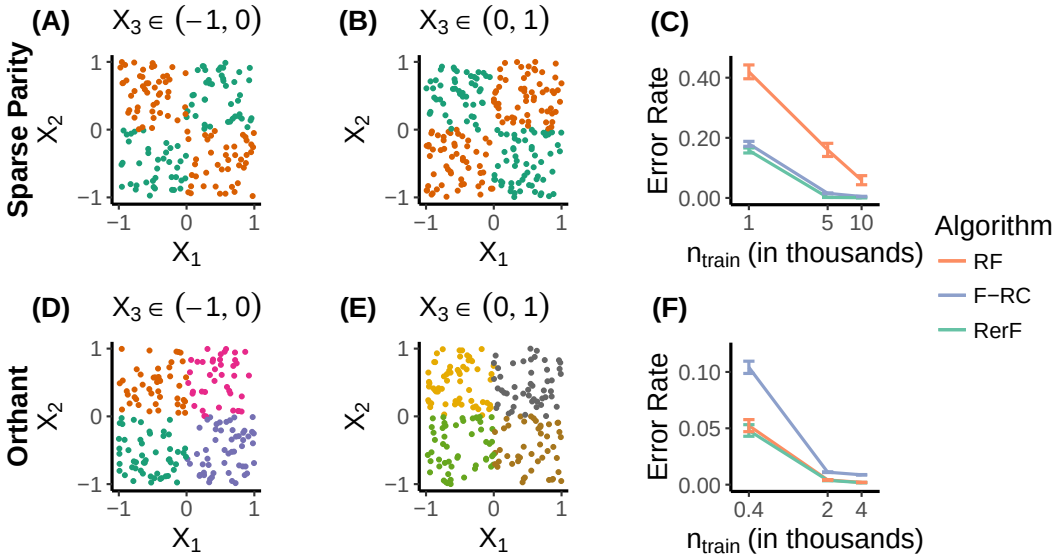

Figure 3.2: Classification performance on the 20-dimensional sparse parity and 6-dimensional orthant problems for various numbers of training samples. F-RC has been known to perform much better than RF on the sparse parity problem [22]. The orthant problem is designed for RF to perform well because the optimal splits are axis-aligned. **(A)** A cross-section of the first two dimensions of sparse parity when $X_3 \in (-1, 0)$. Each of $X_1, \ldots, X_{20} \overset{iid}{\sim} U(-1, 1)$. Only the first three dimensions are informative w.r.t. class label. **(B)** The same as (A), except that the cross-section is taken over $X_3 \in (0, 1)$. **(C)** Error rate plotted against the number of training samples for sparse parity. Error rate is the average over ten repeated experiments. Error bars indicate the standard error of the mean. **(D)** A cross-section of the first two dimensions of orthant when $X_3 \in (-1, 0)$. Each of $X_1, \ldots, X_6 \overset{iid}{\sim} U(-1, 1)$. All dimensions are required to determine the class label, since each orthant corresponds to a different class. **(E)** The same as (D), except that the cross-section is taken over $X_3 \in (0, 1)$. **(F)** The same as (C), except for orthant. RerF exhibits superior performance on both problems, and is therefore more robust than RF and F-RC to the distribution of the data.

the density (fraction of nonzeros) of A. For each of $\lambda \in \{\frac{1}{p}, \ldots, \frac{5}{p}\}$, the best model for each algorithm was selected with respect to the hyperparameter d based on minimum out-of-bag error. Error rate on a test set was computed for each of the five models for the two algorithms. Sensitivity of each algorithm is defined as the sample standard deviation of the set of error

rates corresponding to different values of λ. The sensitivities of error rates of RerF and F-RC to λ are shown in panels (A) and (B) of Figure 3.3 for the sparse parity (n_{train} = 5000) and orthant ($n_{train} = 400$) settings. In both settings, RerF is far more robust to the choice in λ than is F-RC.

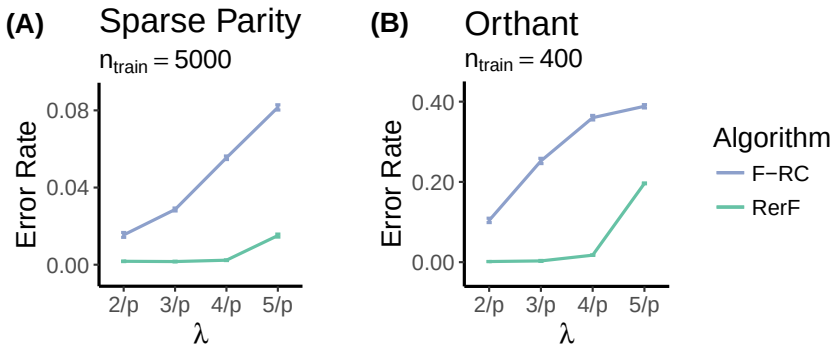

Figure 3.3: Sensitivity of error rate to the hyperparameter λ, which controls the average sparsity of projections. **(A)** Error rate as a function of λ on sparse parity ($p = 20$). **(B)** The same as (A) except on orthant ($p = 6$). In both cases, RerF is far less sensitive to different values of λ than is F-RC.

3.3.2 RerF Performance on Benchmark Datasets

Training procedures and hyperparameter selection were the same as those described for the synthetic datasets. For each benchmark dataset, error rates were estimated via five-fold cross-validation. Error rates were then normalized by the chance probability of error. Pairwise comparisons were made for each pair of algorithms. For each pair, for each dataset the normalized error rate of one algorithm was subtracted by that of the other. Figure 3.4 shows histograms of the pairwise differences in errors over the benchmark datasets. The left column shows comparisons for all 105 datasets, the middle column shows comparisons for the 65 numeric

datasets, and the right column shows comparisons for the 40 categorical datasets. Categorical datasets are defined as those datasets having at least one categorical (non-ordinal) feature. Over all datasets, both RerF and F-RC tend to outperform RF, as indicated by the skew of the histograms to the left. Wilcoxon signed-rank tests produce p-values < 0.01 for both of these comparisons. A close examination of the top center and right panels suggests that RerF performs disproportionately better on the numeric datasets. This phenomenon is also evident in the comparison of F-RC with RF. We note that the datasets having categorical features had to undergo more processing due to one-of-K encoding of the categorical features. Also, the categorical datasets tended to have relatively more missing data than the other datasets. It is possible that this heavier processing of the categorical datasets introduces additional noise. A Wilcoxon signed-rank test suggests no significant difference in performance between RerF and F-RC. Cross-validation errors for each dataset are tabulated in Table 3.2. Bold indicates the lowest error for each dataset.

	5-fold CV Error Rate		
Dataset	RF	RerF	F-RC
abalone	0.759 ± 0.006	0.758 ± 0.007	$\mathbf{0.751 \pm 0.009}$
acute_inflammation_task_1	0 ± 0	0 ± 0	0 ± 0
acute_inflammation_task_2	0 ± 0	0 ± 0	0 ± 0
adult	0.139 ± 0.002	$\mathbf{0.138 \pm 0.002}$	0.139 ± 0.002
annealing	$\mathbf{0.467 \pm 0.05}$	0.482 ± 0.066	0.628 ± 0.004
arrhythmia	$\mathbf{0.29 \pm 0.016}$	0.296 ± 0.017	0.535 ± 0.027
audiology_std	0.388 ± 0.025	0.375 ± 0.023	$\mathbf{0.295 \pm 0.049}$
balance_scale	0.118 ± 0.018	$\mathbf{0.034 \pm 0.006}$	0.053 ± 0.007
balloons	0.4 ± 0.113	0.4 ± 0.138	$\mathbf{0.267 \pm 0.113}$
bank	0.078 ± 0.001	$\mathbf{0.071 \pm 0}$	0.074 ± 0.001
blood	$\mathbf{0.213 \pm 0.005}$	0.214 ± 0.009	0.235 ± 0.006

breast_cancer	**0.262 ± 0.006**	0.266 ± 0.007	0.283 ± 0.009
breast_cancer-wisconsin	0.029 ± 0.01	0.027 ± 0.01	**0.027 ± 0.012**
breast_cancer-wisconsin-diag	0.04 ± 0.008	**0.026 ± 0.007**	**0.026 ± 0.007**
breast_cancer-wisconsin-prog	**0.207 ± 0.019**	**0.207 ± 0.019**	0.212 ± 0.024
car	0.02 ± 0.002	0.012 ± 0.001	**0.008 ± 0.002**
cardiotocography_task_1	0.129 ± 0.003	0.125 ± 0.003	**0.122 ± 0.005**
cardiotocography_task_2	0.056 ± 0.005	**0.053 ± 0.004**	0.054 ± 0.006
chess_krvk	**0.508 ± 0.002**	0.512 ± 0.001	0.508 ± 0.002
chess_krvkp	0.005 ± 0.002	**0.004 ± 0.001**	0.005 ± 0.001
congressional_voting	0.032 ± 0.012	0.032 ± 0.009	**0.028 ± 0.009**
conn_bench-sonar-mines-rocks	0.144 ± 0.026	0.149 ± 0.028	**0.134 ± 0.025**
conn_bench-vowel-deterding	0.042 ± 0.005	0.032 ± 0.005	**0.029 ± 0.005**
contrac	0.449 ± 0.013	0.463 ± 0.016	**0.462 ± 0.011**
credit_approval	**0.122 ± 0.008**	0.13 ± 0.01	0.13 ± 0.01
dermatology	0.022 ± 0.009	**0.019 ± 0.01**	0.027 ± 0.01
ecoli	0.134 ± 0.018	0.143 ± 0.016	**0.131 ± 0.016**
flags	0.37 ± 0.021	0.371 ± 0.01	**0.365 ± 0.018**
glass	0.218 ± 0.042	**0.21 ± 0.04**	0.228 ± 0.038
haberman_survival	0.281 ± 0.02	**0.268 ± 0.02**	0.317 ± 0.026
hayes_roth	**0.191 ± 0.037**	0.198 ± 0.044	0.206 ± 0.035
heart_cleveland	0.446 ± 0.01	0.423 ± 0.021	**0.403 ± 0.014**
heart_hungarian	0.092 ± 0.014	**0.085 ± 0.02**	0.092 ± 0.017
heart_switzerland	**0.617 ± 0.04**	0.634 ± 0.024	0.658 ± 0.038
heart_va	**0.645 ± 0.052**	0.66 ± 0.043	0.675 ± 0.048
hepatitis	**0.116 ± 0.024**	0.129 ± 0.014	**0.116 ± 0.024**
hill_valley	0.449 ± 0.012	**0 ± 0**	0.002 ± 0.002
hill_valley-noise	0.508 ± 0.02	0.048 ± 0.01	**0.036 ± 0.011**
horse_colic	0.157 ± 0.016	**0.154 ± 0.014**	**0.154 ± 0.014**
ilpd_indian-liver	0.287 ± 0.017	**0.261 ± 0.021**	0.278 ± 0.017
image_segmentation	**0.071 ± 0.024**	0.09 ± 0.016	0.081 ± 0.022
ionosphere	0.068 ± 0.01	0.068 ± 0.008	**0.063 ± 0.013**
iris	0.053 ± 0.017	0.06 ± 0.016	**0.047 ± 0.017**
led_display	**0.278 ± 0.009**	0.285 ± 0.013	0.279 ± 0.009
lenses	0.29 ± 0.046	0.21 ± 0.064	**0.17 ± 0.077**
letter	0.034 ± 0.001	**0.03 ± 0.001**	0.031 ± 0.001
libras	0.178 ± 0.018	**0.128 ± 0.014**	0.133 ± 0.02
low_res-spect	0.426 ± 0.021	0.356 ± 0.022	**0.349 ± 0.025**
lung_cancer	**0.533 ± 0.076**	**0.533 ± 0.066**	0.567 ± 0.047

magic	0.119 ± 0.003	0.111 ± 0.002	$\mathbf{0.108 \pm 0.002}$
mammographic	0.169 ± 0.009	$\mathbf{0.166 \pm 0.012}$	0.184 ± 0.01
molec_biol-promoter	0.303 ± 0.026	0.302 ± 0.013	$\mathbf{0.293 \pm 0.024}$
molec_biol-splice	0.033 ± 0.004	0.034 ± 0.004	$\mathbf{0.032 \pm 0.003}$
monks_1	0.322 ± 0.039	$\mathbf{0.25 \pm 0.047}$	0.266 ± 0.057
monks_2	0.32 ± 0.024	$\mathbf{0.26 \pm 0.032}$	0.272 ± 0.023
monks_3	0.098 ± 0.016	$\mathbf{0.09 \pm 0.015}$	0.106 ± 0.009
mushroom	0 ± 0	0 ± 0	0 ± 0
musk_1	0.118 ± 0.016	$\mathbf{0.103 \pm 0.013}$	0.109 ± 0.016
musk_2	0.022 ± 0.003	0.022 ± 0.003	0.022 ± 0.003
nursery	0.009 ± 0.001	0.001 ± 0.001	$\mathbf{0 \pm 0}$
optical	0.023 ± 0.002	0.024 ± 0.003	$\mathbf{0.022 \pm 0.002}$
ozone	0.056 ± 0.002	$\mathbf{0.055 \pm 0.001}$	$\mathbf{0.055 \pm 0.002}$
page_blocks	0.026 ± 0.001	$\mathbf{0.025 \pm 0.002}$	0.026 ± 0.002
parkinsons	0.062 ± 0.006	0.072 ± 0.013	$\mathbf{0.062 \pm 0.019}$
pendigits	0.008 ± 0.001	$\mathbf{0.005 \pm 0.001}$	$\mathbf{0.005 \pm 0.001}$
pima	0.232 ± 0.028	$\mathbf{0.213 \pm 0.028}$	0.221 ± 0.025
pittsburgh_bridges-MATERIAL	0.142 ± 0.021	$\mathbf{0.123 \pm 0.012}$	0.132 ± 0.017
pittsburgh_bridges-REL-L	$\mathbf{0.29 \pm 0.048}$	0.298 ± 0.076	0.299 ± 0.057
pittsburgh_bridges-SPAN	0.403 ± 0.051	0.369 ± 0.056	$\mathbf{0.337 \pm 0.043}$
pittsburgh_bridges-T-OR-D	$\mathbf{0.128 \pm 0.03}$	0.138 ± 0.02	$\mathbf{0.128 \pm 0.03}$
pittsburgh_bridges-TYPE	0.369 ± 0.03	$\mathbf{0.35 \pm 0.022}$	0.388 ± 0.032
planning	0.296 ± 0.026	$\mathbf{0.28 \pm 0.019}$	0.297 ± 0.009
post_operative	0.353 ± 0.057	$\mathbf{0.321 \pm 0.026}$	0.367 ± 0.04
ringnorm	0.038 ± 0.003	0.02 ± 0.001	$\mathbf{0.019 \pm 0.002}$
seeds	0.071 ± 0.025	$\mathbf{0.057 \pm 0.012}$	0.076 ± 0.03
semeion	0.056 ± 0.005	0.058 ± 0.005	$\mathbf{0.056 \pm 0.007}$
soybean	0.108 ± 0.02	$\mathbf{0.098 \pm 0.006}$	0.107 ± 0.008
spambase	0.05 ± 0.004	$\mathbf{0.044 \pm 0.004}$	0.045 ± 0.004
spect	$\mathbf{0.312 \pm 0.071}$	0.325 ± 0.067	0.325 ± 0.067
spectf	0.238 ± 0.023	0.262 ± 0.046	$\mathbf{0.225 \pm 0.032}$
statlog_australian-credit	0.123 ± 0.006	$\mathbf{0.122 \pm 0.013}$	0.133 ± 0.01
statlog_german-credit	$\mathbf{0.234 \pm 0.009}$	0.236 ± 0.013	0.245 ± 0.018
statlog_heart	$\mathbf{0.163 \pm 0.014}$	0.174 ± 0.014	0.178 ± 0.009
statlog_image	$\mathbf{0.021 \pm 0.004}$	0.024 ± 0.003	0.023 ± 0.002
statlog_landsat	0.088 ± 0.005	$\mathbf{0.087 \pm 0.005}$	0.087 ± 0.004
statlog_shuttle	0 ± 0	0 ± 0	0 ± 0
statlog_vehicle	0.238 ± 0.012	$\mathbf{0.194 \pm 0.006}$	0.197 ± 0.007

steel_plates	**0.203 ± 0.004**	0.207 ± 0.008	0.211 ± 0.008
synthetic_control	0.018 ± 0.007	**0.015 ± 0.007**	0.017 ± 0.006
teaching	0.496 ± 0.038	**0.404 ± 0.034**	**0.404 ± 0.034**
thyroid	0.002 ± 0.001	0.002 ± 0.001	**0.002 ± 0.001**
tic_tac-toe	**0.205 ± 0.009**	0.215 ± 0.014	0.214 ± 0.014
titanic	0.217 ± 0.004	**0.213 ± 0.003**	0.213 ± 0.003
trains	0.3 ± 0.2	**0.2 ± 0.2**	0.3 ± 0.2
twonorm	0.026 ± 0.002	0.022 ± 0.002	**0.021 ± 0.002**
vertebral_column_task_1	0.165 ± 0.011	**0.158 ± 0.009**	0.161 ± 0.011
vertebral_column_task_2	0.177 ± 0.017	0.174 ± 0.014	**0.165 ± 0.013**
wall_following	**0.004 ± 0.001**	0.005 ± 0.001	0.008 ± 0.002
waveform	0.145 ± 0.004	0.134 ± 0.003	**0.134 ± 0.003**
waveform_noise	0.143 ± 0.004	0.135 ± 0.003	**0.131 ± 0.003**
wine	0.028 ± 0.022	**0.017 ± 0.011**	0.028 ± 0.018
wine_quality-red	0.31 ± 0.018	0.306 ± 0.015	**0.305 ± 0.016**
wine_quality-white	0.313 ± 0.007	0.311 ± 0.012	**0.31 ± 0.011**
yeast	0.377 ± 0.017	**0.369 ± 0.019**	0.375 ± 0.014
zoo	**0.06 ± 0.029**	0.06 ± 0.037	0.07 ± 0.037

Table 3.2: Five-fold cross-validation error rates for each of the UCI datasets. For each dataset, bold indicates lowest error rate among the algorithms.

3.3.3 Strength and Correlation of Trees

One of the most important and well-known results in ensemble learning theory for classification states that the generalization error of an ensemble learning procedure is bounded above by the quantity $\frac{\bar{\rho}(1-s^2)}{s^2}$, where $\bar{\rho}$ is a particular measure of the correlation of the base learners and s is a particular measure of the strength of the base learners [7]. In both RerF and F-RC, the set of possible splits that can be sampled is far larger in size than that for RF, which may lead to more diverse trees. Simultaneously, the ability to sample a more diverse set of splits may increase the likelihood

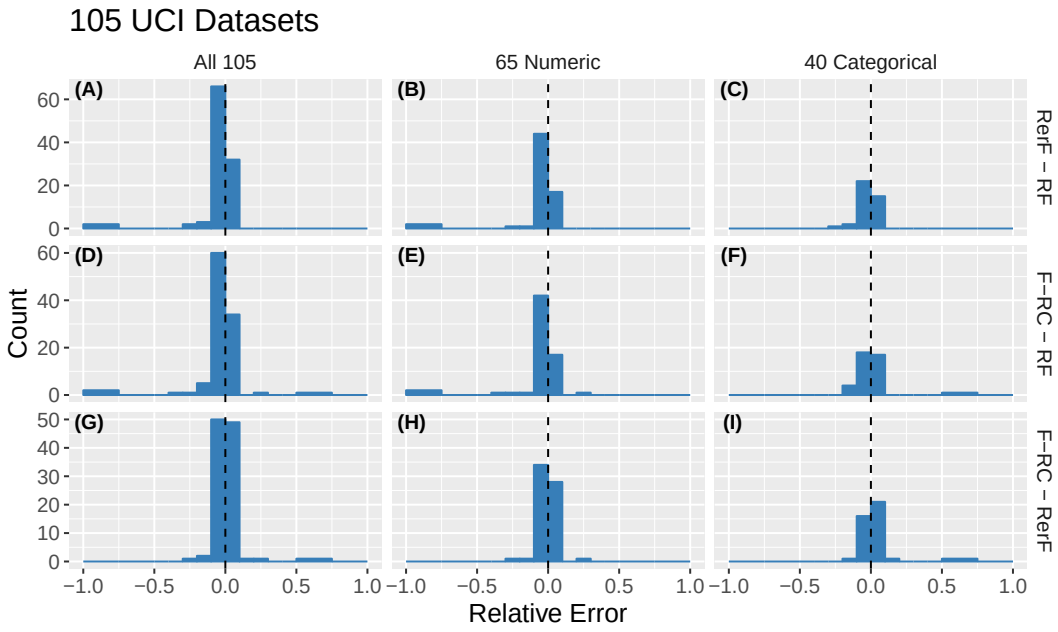

Figure 3.4: Histograms of the pairwise relative errors of RF, RerF, and F-RC, normalized by chance on benchmark datasets from the UCI machine learning repository. Each row corresponds to a comparison of a particular pair of algorithms. For instance, in the top row, the relative error is defined as error of RerF subtracted by that of RF. The x-axis represents the relative error discretized into bins. The y-axis represents the counts (number of datasets) in each of the bins. The left, center, and right columns are the histograms for all 105 UCI datasets, the 65 numeric datasets, and the 40 categorical datasets, respectively. Both RerF and F-RC tend to outperform RF across all 105 datasets. However, RerF performs better than RF more frequently than does F-RC. When directly compared, RerF and F-RC show no significant difference in classification performance.

of finding good splits and therefore boost the strength of the trees. In order to investigate the effects of the various sampling distributions on f_A, we estimated these quantities for all of the experiments previously described. Scatter plots of tree strength vs tree correlation are shown in Figure 3.5 for sparse parity ($n_{train} = 1000$), orthant ($n_{train} = 400$), Trunk ($n_{train} = 10$), and Trunk ($n_{train} = 100$) (panels A-D, respectively). In all four settings, we note that RerF classifies as well as or better than RF and F-RC.

On the sparse parity dataset Figure 3.5 (panel A), RerF and F-RC produce significantly stronger trees than does RF, at the expense of an increase in correlation amongst the trees. Noting that both RerF and F-RC are much more accurate than RF in this setting, any performance degradation due to the increase in correlation relative to RF is outweighed by the increased strength. RerF produces slightly less correlated trees than does F-RC, which may explain why RerF has a slightly lower error rate than does F-RC on this setting.

On the orthant dataset Figure 3.5 (panel B), F-RC produces trees of roughly the same strength as those in RF, but significantly more correlated. This may explain why F-RC has substantially worse prediction accuracy than does RF. RerF also produces trees more correlated than those in RF, but to a lesser extent than F-RC. Furthermore, the trees in RerF are stronger than those in RF. Observing that RerF has roughly the same error rate as RF does, it seems that any contribution of greater tree strength in RerF is canceled by a contribution of greater tree correlation.

On the Trunk problem with $p = 10$ and $n_{train} = 10$ Figure 3.5 (panel C), RerF and F-RC produces trees that are comparable in strength to those in RF but less correlated. However, increasing n_{train} to 100 Figure 3.5 (panel D), the trees in RerF and F-RC become both stronger and more correlated. In both cases, RerF and F-RC have better classification performance than RF.

The results shown in panels (C) and (D) suggest something that may be indicative of a general phenomenon. Namely, for smaller training set

sizes, tree correlation may be a more critical factor than tree strength because their simply is not enough data to induce strong trees, and thus, the only way to improve performance is through increasing the diversity of trees. Likewise, when the training set is sufficiently large enough, tree correlation matters less because their is enough data to induce strong trees. Since RerF has the ability to produce both stronger and more diverse trees, it is better adaptive to both regimes than is RF. We point out that in all four settings, RerF never produces more correlated trees than does F-RC, and sometimes produces less correlated trees. A possible explanation for this is that the splits made by RerF are linear combinations of a random number of dimensions, whereas in F-RC the splits are linear combinations of a fixed number of dimensions. Thus, in some sense, there is more randomness in the default f_A adopted by RerF than in that adopted by F-RC.

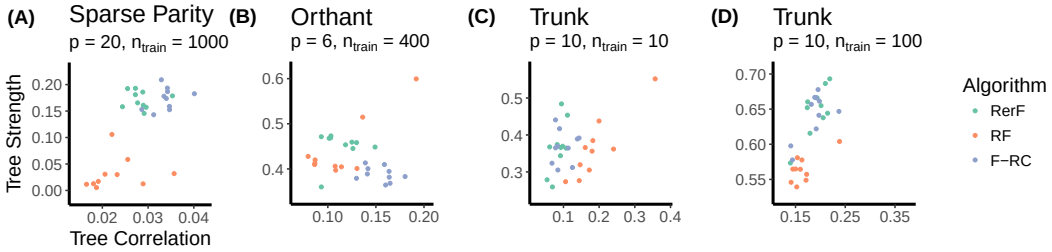

Figure 3.5: Comparison of tree strength and correlation of RerF, RF, and F-RC on four of the synthetic datasets: **(A)** sparse parity with $p = 10, n_{train} = 1000$, **(B)** orthant with $p = 6, n_{train} = 400$, **(C)** Trunk with $p = 10, n_{train} = 10$, and **(D)** Trunk with $p = 10, n_{train} = 100$. For a particular algorithm, there are ten dots, each corresponding to one of ten trials. Note in all settings, RerF beats RF and/or F-RC. However, the mechanism by which it does varies across the different settings. In sparse parity RerF wins because the trees are substantially stronger, even though the correlation increases. In Trunk for small sample size, it's purely because of less correlated trees. However, when sample size increases 10-fold, it wins purely because of stronger trees. This suggests that the degree of randomization may be more influential for smaller sample size, whereas tree strength may be more influential when sample size is sufficiently large.

3.3.4 Understanding the Bias and Variance of RerF

The crux of all learning tasks is to try to optimize the trade-off between bias and variance. As a first step in understanding how the choice in f_A effects the balance between bias and variance, we estimated bias, variance, and error rate of the various algorithms on the sparse parity problem. Universally agreed upon definitions of bias and variance for 0-1 loss do not exist, and several such definitions have been proposed for each. Here we adopt the framework for defining bias and variance for 0-1 loss proposed in James (2003) [23]. Under this framework, bias and variance for 0-1 loss have similar interpretations to those for mean squared error. That is, bias is a measure of the distance between the expected output of a classifier and the true output, and variance is a measure of the average deviation of a classifier output around its expected output. Unfortunately, these definitions (along with the term for Bayes error) do not provide an additive decomposition for the expected 0-1 loss. Therefore, James (2003) provides two additional statistics that do provide an additive decomposition. In this decomposition, the so-called "systematic effect" measures the contribution of bias to the error rate, while the "variance effect" measures the contribution of variance to the error rate. For completeness, we restate these definitions below.

Suppose we observe a training set $D_n = (X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^p \times \{1, \ldots, K\}$, where X is a random p-dimensional input vector and Y is a class label associated with X. Let f_{XY} denote the joint distribution of X and Y, which induces a conditional probability distribution $P_{Y|X}$. Suppose we use

a learning procedure to train a classifier on D_n, which is to be used in order to predict the unobserved value of Y associated with a newly observed X. Denote this classifier as $h(\cdot|D_n)$. Let $h^*(X) = \underset{k}{\operatorname{argmax}} P_{Y|X}(Y = k|X)$ be the Bayes optimal classifier, and let $\bar{h}(X) = \underset{k}{\operatorname{argmax}} P_{D_n}(h(X|D_n) = k)$ be the most common prediction (mode) with respect to the distribution of D_n. The latter is referred to as the "systematic" prediction in James (2003). Furthermore, let $P^*(X) = P_{Y|X}(Y = h^*(X)|X)$ and $\bar{P}(X) = P_{D_n}(h(X|D_n) = \bar{h}(X))$ The bias, variance, systematic effect (SE), and variance effect (VE) are defined as

$$Bias = P_X(\bar{h}(X) = h^*(X)), \tag{3.1}$$

$$Var = 1 - E_X[\bar{P}(X)], \tag{3.2}$$

$$SE = E_X[P^*(X) - P_{Y|X}(Y = \bar{h}(X)|X)], \tag{3.3}$$

$$VE = E_X[P_{Y|X}(Y = \bar{h}(X)|X) - \sum_k P_{Y|X}(Y = k|X)P_{D_n}(h(X|D_n) = k)]. \tag{3.4}$$

Figure 3.6 compares estimates of bias, variance, variance effect, and error rate for RerF, RF, and F-RC as a function of number of training samples. Note that since Bayes error is zero in this setting, systematic effect is the same as bias. The four metrics were estimated from 100 repeated experiments for each value of n_{train}. In panel (A) of Figure 3.6, RerF has lower bias than both RF and F-RC for all training set sizes. All algorithms converge to approximately zero bias after about 3000 samples.

Panel (B) shows that RF has substantially more variance than do RerF and F-RC, and RerF has slightly less variance than F-RC at 3000 samples. The trend in panel (C) is similar to that in panel (B), which is not too surprising since VE measures the direct contribution of the variance to the error rate. Interestingly, although RF has noticeably more variance at 500 samples than do RerF and F-RC, it has slightly lower VE. In Figure 3.6 (panel D), the error rate is shown for reference, which is the sum of bias and VE. Overall, these results suggest that RerF wins primarily through lower bias/SE for fewer training samples, while it wins mainly via lower variance/VE for greater training samples.

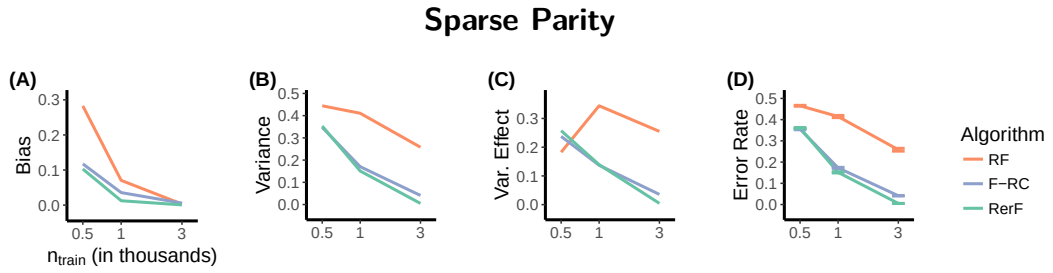

Figure 3.6: Bias **(A)**, variance **(B)**, variance effect **(C)**, and error rate **(D)** of RerF, RF, and F-RC on sparse parity as a function of the number of training samples. Error rate is the sum of systematic effect and variance effect, which roughly measure the contributions of bias and variance to the error rate, respectively. Note that in this example, bias and systematic effect are exactly equal because Bayes error is zero (refer to [23]). For smaller training sets, RerF wins primarily through lower bias/systematic effect, while for larger training sets it wins primarily through lower variance effect.

3.3.5 Consistency of Randomer Forests

Suppose $(X,Y),(X_1,Y_1),\ldots,(X_n,Y_n) \in [0,1]^p \times \{1,\ldots,K\}$ are *iid* pairs of random variables, where (X,Y) is the testing pair and the remaining are the training. A classifier $h_n(x,D_n)$ is a function of all the training pairs

$D_n = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ and the testing data x, which estimates the unknown label $y \in \{1, \ldots, K\}$. For brevity, we always denote the classifier by $h_n(x)$, and the probability of error is defined by $L(h_n) = P(h_n(X) \neq Y)$. The classifier that minimizes the probability of error is called the Bayes classifier, whose error rate is optimal and denoted by L^*. The sequence of classifiers $h_n(x)$ is consistent for a certain distribution f_{xy} if and only if $L(h_n) \to L^*$ as $n \to \infty$, and universally consistent if and only if consistent against all possible distributions f_{xy}. For theoretical purposes, we define a data-agnostic version of RerF.

> **Definition 1.** The data-agnostic RerF is defined as the original RerF whose partition is random and independent of the class labels, i.e., the split algorithm 2 shall be replaced by any random split mechanism that is independent from the class labels, and the projection matrix A is also sampled independently from the class labels.

Theorem 3. *Denote the number of partitions of the random forest as t_n. Then data-agnostic RerF is universally consistent for classification when $t_n \to \infty$ and $\frac{t_n}{n} \to 0$ as $n \to \infty$.*

The universal consistency of data-agnostic RerF essentially follows from Stone's theorem for local averaging estimates [24, 25]. It essentially states that the RerF algorithm is as consistent as the RF method previously proposed. The proof is in Appendix B .

3.3.6 RerF Provides Feature Importance

For many data scientists and researchers, understanding the observed data is just as critical as finding an algorithm with excellent predictive performance (and often the two are coupled). One of the reasons for RF's popularity is its ability to learn good predictive models that simultaneously lend themselves to extraction of suitable feature importance measures. One such measure is the Gini importance [7]. For a particular feature, it is defined as the sum of the reduction in Gini impurity over all splits of all trees made on that feature. With this metric, features that are used in splits often, and when used, result in much purer nodes, will have large importance scores. Unfortunately, with RF, features that are not very informative on there own but are informative when linearly combined with other features may not be given large importance scores. Since splits in RerF are linear combinations of the original features, such features will not be missed. For RerF, we compute Gini importance for each unique projection (linear combination of canonical features).

Gini importance was computed for both RF and RerF on the Trunk problem with $n_{train} = 1000$. Figure 3.7 (panels A and B) depict the linear weights of the observed features that define each of the top ten split node projections. Projections are sorted from highest Gini importance to lowest Gini importance. Note that the top ten projections in RerF are all linear combinations of dimensions, whereas in RF the projections can only be along single dimensions. RF fails to sort some of the individual features according to their "true" informativeness, where true informativeness is

measured by the Bayes error of a distribution along a projection. For example, it ranks a projection along the X_9 dimension as third most important. However, X_9 is actually the ninth most informative projection when only considering axis-aligned projections. The linear combinations in RerF tend to include the first few dimensions, which contain most of the "true" signal. The best possible projection that RerF could sample is the vector of all ones. However, since $\lambda = 5/10$ for this experiment, the probability of sampling such a dense projection having the appropriate coefficients is almost negligible. Figure 3.7 (panel C) shows the normalized Gini importance of the top ten projections for each algorithm. RerF finds many important projections, whereas for RF, only one projection seems to be important. Figure 3.7 (panel D) shows the Bayes error rate of the top ten projections for each algorithm. The results show that RerF finds highly discriminative split node projections compared to RF.

3.3.7 Time and Space Complexity of RerF

3.3.7.1 Theory

The time complexity of an algorithm characterizes how the theoretical processing time for a given input relies on both the hyper-parameters of the algorithm and the characteristics of the input. Let T be the number of trees, n the number of training samples, and d the number of features sampled at each split node. The average case time complexity of RF is $\mathcal{O}(Tdn(\log n)^2)$ [26]. The $dn \log n$ accounts for the sorting of d features at each node. The additional $\log n$ accounts for both the reduction in

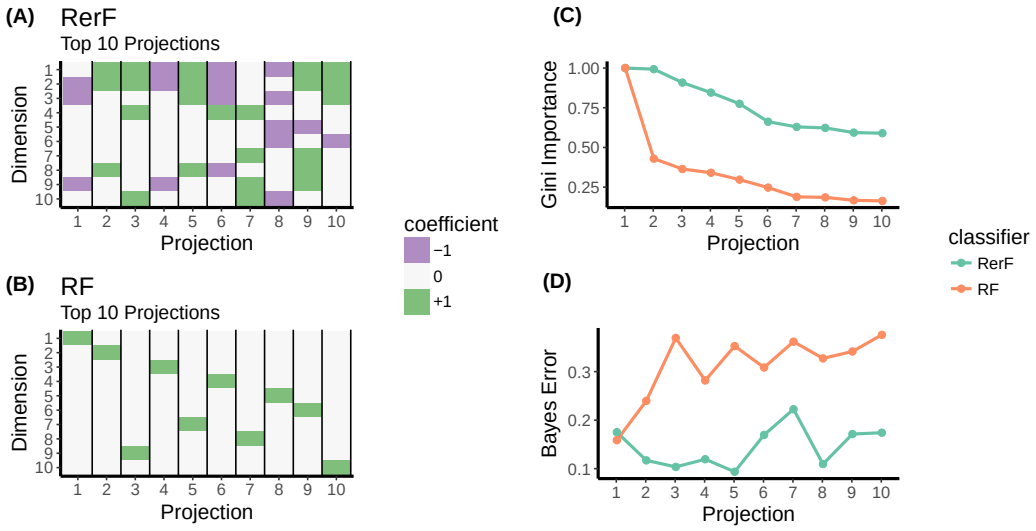

Figure 3.7: The ten projections with the highest Gini importance found by RF and RerF on the Trunk problem with $p = 10, n_{train} = 1000$. **(A)** Visual representation of the top 10 projections identified by RerF. The x-axis indicates the projection. The y-axis indicates the index of the ten canonical dimensions. The colors in the heat map indicate the linear coefficients of each canonical dimension that define each of the projections. **(B)** The same as (A), except for RF. **(C)** Comparison of the Gini importances of the 10 best projections found by each algorithm. **(D)** Comparison of the Bayes error of the 10 best projections found by each algorithm (see B.2 for details). The top 10 projections used in RerF all have substantially lower Bayes error than those used in RF.

node size at lower levels of the tree and the average number of nodes produced. RF's polynomial complexity shows that a good implementation will scale nicely with large input sizes, making it a suitable algorithm to process big data. RerF's average case time complexity is similar to RF's, the only difference being the addition of a term representing a sparse matrix multiplication which is required in each node. This makes RerF's complexity $\mathcal{O}(T * n \log n(d \log n + s))$, where s is the number of nonzeros in the sparse projection matrix. We generally let s be on the order of d, giving a complexity of $\mathcal{O}(Tdn \log^2 n)$, which is the same as for RF. Of note, in RF d is constrained to be no greater than p, the dimensionality of the data.

RerF, on the other hand, does not have this restriction on d. Therefore, if d is selected to be greater than p, RerF may take longer to train. However, $d > p$ often results in substantial predictive performance.

3.3.7.2 Empirical Speed

Comparison of training times on the sparse parity problem is shown in Figure 3.8 (panel A). Training times reported are those corresponding to the best hyperparameter settings for each algorithm. F-RC is the slowest, RF is the fastest, and RerF is in between. While not shown, we note that a similar trend holds for the orthant problem. 3.8 (panel B) shows that when the hyperparameter d of RerF and F-RC is the same as that for RF, training times are comparable. However, training time continues to increase as d exceeds p for RerF and F-RC, which largely accounts for the trend seen in panel (A). 3.8 (panel C) indicates that this additional training time comes with the benefit of substantially improved accuracy. We point out that we could restrict d to be no greater than p for RerF in this setting and it would still perform noticeably better than RF at no additional cost in training time.

Since allowing d to be greater than p can result in substantially greater training times, we wondered if we could restrict d to be no greater than p without hurting performance. Therefore we repeated the experiments for the synthetic datasets, this time not allowing d to be greater than p in RerF and F-RC. Thus, the computational budgets for these algorithms are restricted to be (approximately) the same as that for RF. Error rates are

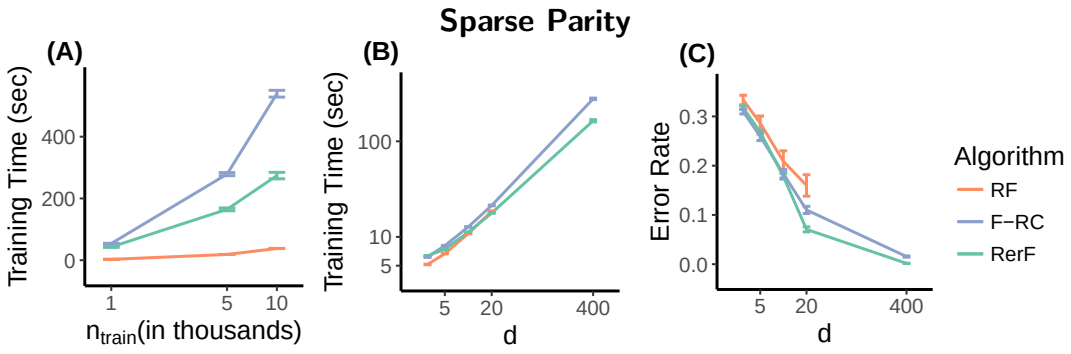

Figure 3.8: Comparison of training times of RF, RerF, and F-RC on the sparse parity dataset. **(A)** Dependency of training time (y-axis) on the number of training samples (x-axis) for the sparse parity problem. 40 cores were used. Reported times correspond to the best set of hyperparameters. **(B)** Dependency of training time (y-axis) on the number of projections sampled at each split node (x-axis) for the sparse parity problem with 5000 training samples. **(C)** Dependency of error rate (y-axis) on the number of projections sampled at each split node (x-axis) for the sparse parity problem with 5000 training samples. RerF and F-RC can sample many more than p projections, unlike RF. As seen in panels (B) and (C), doing so dramatically improves classification performance at the expense of larger training times. However, comparing error rates and training times at $d = 20$, RerF can classify substantially better than RF even with no additional cost in training time.

shown in Figure 3.9. Overall, we still see that RerF is the best performing algorithm.

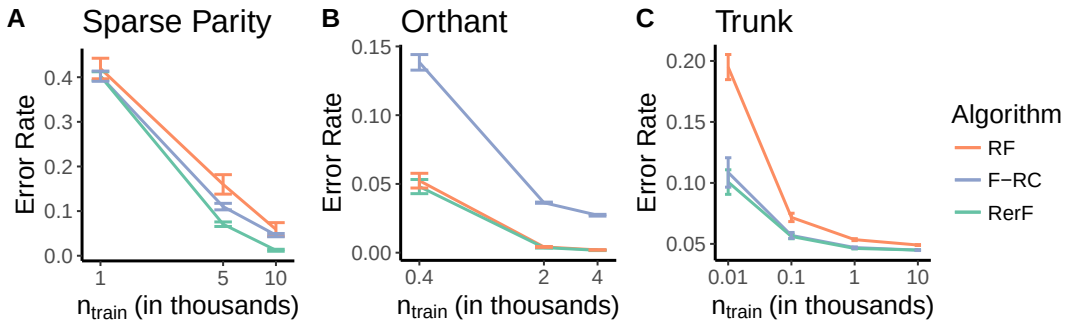

Figure 3.9: The same as Figure 3.2, except that the number of projections sampled by RerF and F-RC were restricted to be no greater than p

3.3.7.3 RerF Implementation Scalability

We offer an openly available multi-core R implementation of RerF on CRAN [27]. We compared speed of training and strong scaling of our implementation to those of the R Ranger [28] and XGBoost [29] packages, which are currently two of the fastest decision tree ensemble software packages available. Ranger offers a fast multicore version of RF that has been extensively optimized for runtime performance. XGBoost offers a fast multicore version of gradient boosted trees. Strong scaling is the relative increase in speed of using multiple cores over that of using a single core. In the ideal case, the use of N cores would produce a factor N speedup. Comparisons were made using three openly available large datasets. For our comparisons, hyperparameters were chosen for each implementation so as to make the comparisons as fair as possible. For all implementations, trees were grown to full depth, 100 trees were constructed, and $d = \sqrt{p}$ features sampled at each node. For RerF, the default f_A was used with $\lambda = 1/p$.

MNIST The MNIST dataset [30] has 60,000 training observations and 784 (28x28) features. In Figure 3.10 (panel A), for a small number of cores, RerF is faster than XGBoost but slower than Ranger. However, when 48 cores are used, RerF is just as fast as Ranger and still faster than XGBoost. Figure 3.10 (panel D) shows that RerF has the best scaling with number of cores.

Higgs The Higgs dataset [31] has 250,000 training observations and 31

features. Figure 3.10 (panel B) shows that when 48 cores are used, RerF is just as fast as ranger and faster than XGBoost. Figure 3.10 (panel E) again shows that RerF utilizes additional cores more effectively than the other implementations.

p53 The p53 dataset [32] has 31,159 training observations and 5,409 features. Figure 3.10 (panel C) shows a similar trend as for MNIST. Figure 3.10 (panel F) indicates that RerF has strong scaling in between that of Ranger and XGBoost. For this dataset, utilizing additional resources with RerF does not provide as much benefit due to the classification task being too easy – the trees are shallow, causing the overhead cost of multithreading to outweigh the speed increase due to parallelism.

3.3.8 Structured RerF

In this section, we highlight that the matrix distribution f_A can be chosen in order to exploit domain knowledge. In computer vision tasks, convolutional neural networks (CNNs) typically exhibit exceptional performance partially because they exploit the spatial relationship of pixels within images. In a similar vein, we propose an f_A which enables the learning procedure to exploit spatial structure. We will refer to this particular instantiation of RerF as Structured RerF (S-RerF). At each split node, S-RerF randomly samples d patches of spatially contiguous pixels. For each patch, a new feature is constructed by taking a randomly weighted linear combination of the pixels within the patch. The split is made by

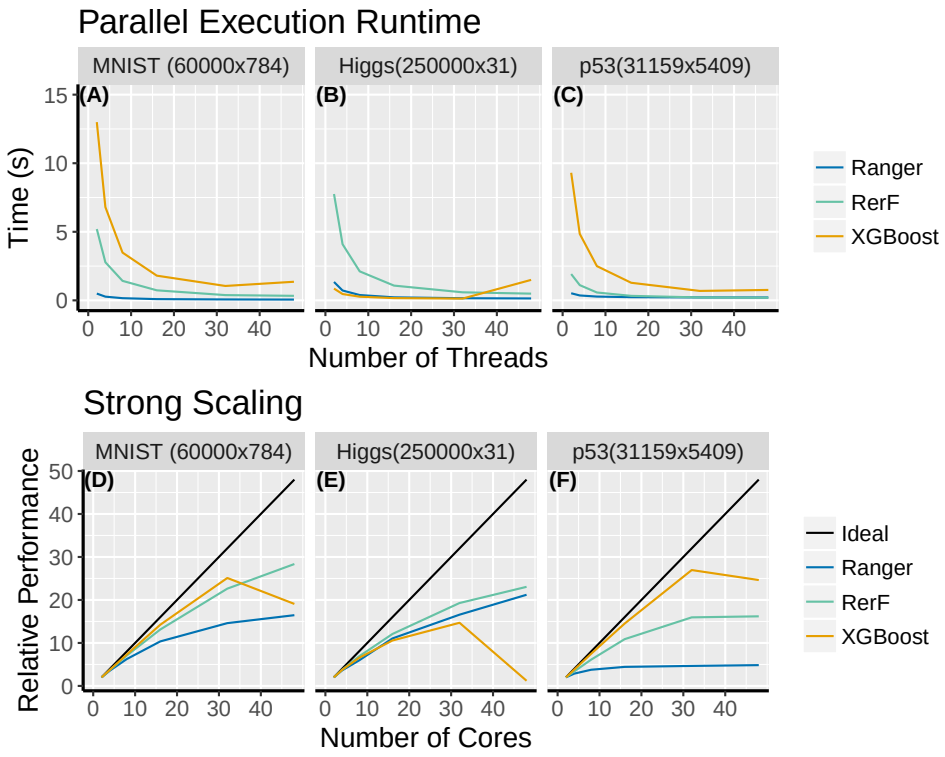

Figure 3.10: *Top:* The per tree training time for three large real world datasets. Training was performed using matching parameters where possible and default parameters otherwise. RerF's performance is comparable to the highly optimized XGBoost and Ranger and even outperforms XGBoost on two of the datasets. This is achieved despite RerF both having to rotate data at each node and being highly customizable due to it almost entirely being implemented in R. *Bottom:* Strong scaling is the time needed to train a forest with one core divided by the time needed to train a forest with multiple cores. This is a measurement of a systems ability to efficiently utilize additional resources. RerF is able to scale well over the entire range of tested cores whereas XGBoost has sharp drops in scalability where it is unable to use additional threads due to characteristics of the given datasets. The p53 dataset, despite having a large number of dimensions, is easily classifiable, which leads to short trees. The p53 strong scaling plot shows that when trees are short the overhead of multithreading prevents RerF from efficiently using the additional resources.

optimizing the split criteria over this set of d constructed features. The key idea here is that by constructing new features from spatially contiguous pixels, the features can represent low-level objects like simple edges and shapes. These low-level objects can then be used to construct meaningful hierarchical decision rules using a dictionary of patches (rather than

pixels).

To test this idea, we constructed a toy image classification problem. One class of images contains randomly sized and spaced horizontal lines. The other class contains randomly sized and spaced vertical lines. Figure 3.11 (panel A) shows three example images from each class. Figure 3.11 (panel B) shows performance of RF, the default RerF, S-RerF, and a control RerF. The control RerF samples contiguous patches as in S-RerF, but subsequently randomly reassigns the sampled pixel locations so as to abolish the exploitation of spatial structure. S-RerF demonstrates a remarkable improvement in classification performance over all other algorithms, achieving near-perfect accuracy with just ten training samples.

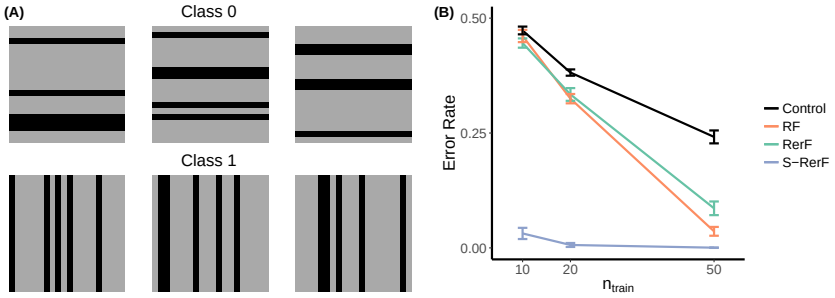

Figure 3.11: Exploiting spatial structure in image classification. **(A)** Simulated images: Class 0 are horizontal bars; Class 1 are vertical bars. **(B)**: classification performance of Structure RerF (S-RerF), RerF, RF, and Control RerF (using the same projection sparsity as that in Structure RerF), demonstrates that S-RerF achieves a dramatic empirical improvement in efficiency.

3.4 Conclusion

RerFs provides a general framework under which all orthogonal and oblique tree ensemble methods fall. Our open-source implementation allows users to easily specify any instantiation they desire. Nonetheless,

we propose a default instantiation and motivate its use. Experiments on synthetic problems show that it learns particularly well on datasets for which individual dimensions have little or no information but are jointly informative, and when there are a large number of irrelevant dimensions, which we hypothesize are two properties that generic datasets often possess. The superior performance of RerFs on a suite of 105 benchmark datasets support this hypothesis. We offer experimental evidence highlighting three properties of the random projection matrix that can substantially affect classification performance, namely 1) its average sparsity, 2) the distribution of column sparsity (in particular, fixed vs. varying), and 3) the width (number of projections sampled). We also highlight that we can exploit prior domain knowledge via selection of an appropriate random matrix distribution. Additionally, we show that the training time of our method can be on par with some of the fastest RF implementations currently available. Lastly, we demonstrate that unlike other oblique methods, we can tractably adopt the Gini feature importance measure, which is a fast and useful approach for identifying informative linear combinations of features.

The RerF framework opens up a myriad of interesting questions. On the theoretical side, the work of Biau et al. (2016) can be extended to the RerF setting [25, 33]. In this work, we proved consistency of a simplified version of RerF that doesn't use the data to choose splits. However, it is very likely that the theorems in Biau et al. (2016) for the original supervised RF algorithm can be immediately extended to the default RerF with some

minor modifications – their proofs rely on clever adaptations of classical consistency results for data-independent partitioning classifiers, which are agnostic to whether the partitions are hyper-rectangular or not. Moreover, we hope that theoretical investigations will yield more insight into which distributions f_A will be optimal under different distributional settings, both asymptotically and under finite sample assumptions. For instance, Biau et al. construct a distribution for which RF with fixed depth is guaranteed to have a probability of error of at least 1/6. Although the optimal decision boundary is a union of axis-aligned splits, the greedy nature in which splits are selected (rather than global optimization) prevents it from learning the appropriate rules, regardless of the amount of training data. The default RerF should be able to achieve a probability of error lower than 1/6 on this problem, for similar reasons that it is able to perform better than RF on the sparse parity problem (the distributions of the two problems are quite similar). The idea that oblique methods are consistent on a wider class of problems seems likely to be true, given that they are more flexible in some sense. Additionally, it would be interested to see what theoretical guarantees hold when the random matrix distribution depends on the data. In other words, when a supervised procedure is used to identify (hopefully) strong discriminant directions, are there different/additional conditions needed to guarantee consistency? Since such procedures may substantially reduce the diversity of trees, it seems plausible that the data subsampling and/or depth conditions required for consistency in Biau et al. (2016) may need to be changed. Indeed, the work of Rainforth and Wood (2015) [12], which utilizes canonical correlation analysis to explicitly

compute directions along which to split, suggests that in order to achieve strong empirical performance, a novel sampling procedure which they call projection bootstrapping is often necessary.

Another avenue is to further explore Structured RerF, in computer vision as well as in other domains. In this work we only present one sampling distribution for exploiting spatial structure for computer vision. Furthermore, the distribution we use is still rather simplistic and naive. However, the point was simply to illustrate how domain knowledge can be used to bias the sampling distribution in order to achieve better performance. As a final statement, we vouch that the RerF implementation developed in this work is a competitive alternative to existing tree ensemble implementations, and can in fact realize any previously proposed tree ensemble methods, possibly with some minor modifications. Open source code is available: https://github.com/neurodata/R-RerF.

Chapter 4

Identifying Predictive Markers for Ovarian Cancer Classification

4.1 Introduction

More than 22,000 women are estimated to receive an ovarian cancer diagnosis for the year 2018, and approximately 14,000 will die from ovarian cancer [34]. If caught and treated early, the five-year survival rate is above 90%. Unfortunately, due to a lack of effective screening methods, only 15% of cases are found early enough [35]. When discovered at late-stage, survival rates are poor. Thus, there is a need for an effective early screening method for ovarian cancer. In this chapter, we briefly review a highly sensitive technique developed by Wang et al. [36] for measuring blood plasma levels of peptides known to be associated with cancer. We then demonstrate how RerFs can be used to identify the most salient peptides for ovarian cancer and to learn a highly accurate model for classifying a sample as ovarian cancer or normal.

4.2 Methods

4.2.1 Quantifying Peptide Abundance in Plasma Using SAFE-SRM

Wang et al. [36] develops a high-throughput, robust, and reproducible method for validating candidate peptide biomarkers, demonstrating remarkable effectiveness when applied to distinguishing normal patients from those having ovarian cancer. The method, called sequential analysis of fractionated eluates by selected reaction monitoring (SAFE-SRM), enables highly sensitive and robust quantification of low-abundance peptides present in blood plasma samples.

SAFE-SRM was used to measure the abundance of 318 candidate peptides contained in blood plasma samples from patients known to have ovarian cancer and those without it. Various supervised learning techniques were used to identify interpretable and predictive classifiers on a training set consisting of 27 non-cancerous samples and 7 cancerous samples (refer to 4.2.2 below). The class-imbalance in the training set was intentional for the purpose of minimizing the false positive rate (FPR). For cancer screening and diagnosis, it is common to prioritize FPR over false negative rate (FNR) because cancer therapies can be harsh and/or invasive. FNR and FPR were reported for various classifiers on a test set consisting of 10 non-cancerous samples and 22 cancerous samples.

4.2.2 Identification of Salient Peptides

In order to identify the most predictive peptides, a stepwise forward selection logistic regression was employed. First, a logistic regression model was fit to the training set. Leave-one-out cross-validation was used to estimate predictive performance of each model. The peptide yielding the lowest cross-validated misclassification rate on the training set was selected for inclusion in the model. If more than one peptide achieved the lowest misclassification rate, ties were broken by selecting the peptide that produced the greatest model likelihood. This process of selecting a peptide biomarker to be added to the model was repeated until no further decrease in cross-validated misclassification rate could be achieved by addition of a peptide. To find a subset of peptides from the same protein that could achieve perfect classification, the same stepwise forward selection procedure was applied for each potential biomarker protein. After identifying the best classifiers, predictive performance of models fit to different combinations of the peptide biomarkers was compared on the test set.

For comparison, we also utilized the Gini importance metric of RF and RerF for identifying important peptides. Both algorithms were trained with varying numbers d of projections sampled at each split node. The best model for each algorithm was chosen via out-of-bag error.

4.3 Results

Figure 4.1 shows the logistic regression mean squared errors on the training data for each individual peptide in order of increasing error. The peptide VSFELFADK from the PPIA gene was identified as being substantially more predictive than any other peptide. Next, we determined whether including any additional peptides from PPIA in the logistic regression model could further improve predictive performance. The peptide FEDENFILK further reduced the mean squared error when combined with VSFELFADK. On a test set consisting of 22 ovarian samples and 10 normal samples, the logistic regression classifier using these two peptide abundances has a 0% false positive rate (FPR) and 15.6% false negative rate (FNR).

Figure 4.2 shows the ten projections with largest Gini importance for each algorithm. Both RF and RerF identify VSFELFADK as the most predictive peptide, which agrees with the sequential forward selection logistic regression approach. Both RF and RerF have a 0% FPR. However, they both have FNRs significantly greater than that of the logistic regression model. This is not surprising, since no dimensionality reduction was performed prior to learning the forest classifiers, unlike for the logistic regression classifier. Thus, the curse of dimensionality hurts the generalization ability of the forest classifiers. However, the Gini importance metric offers a straightforward approach for reducing the dimensionality. Both RF and RerF were run again using only the two peptides with highest Gini importance. The second most important peptides found by RF and

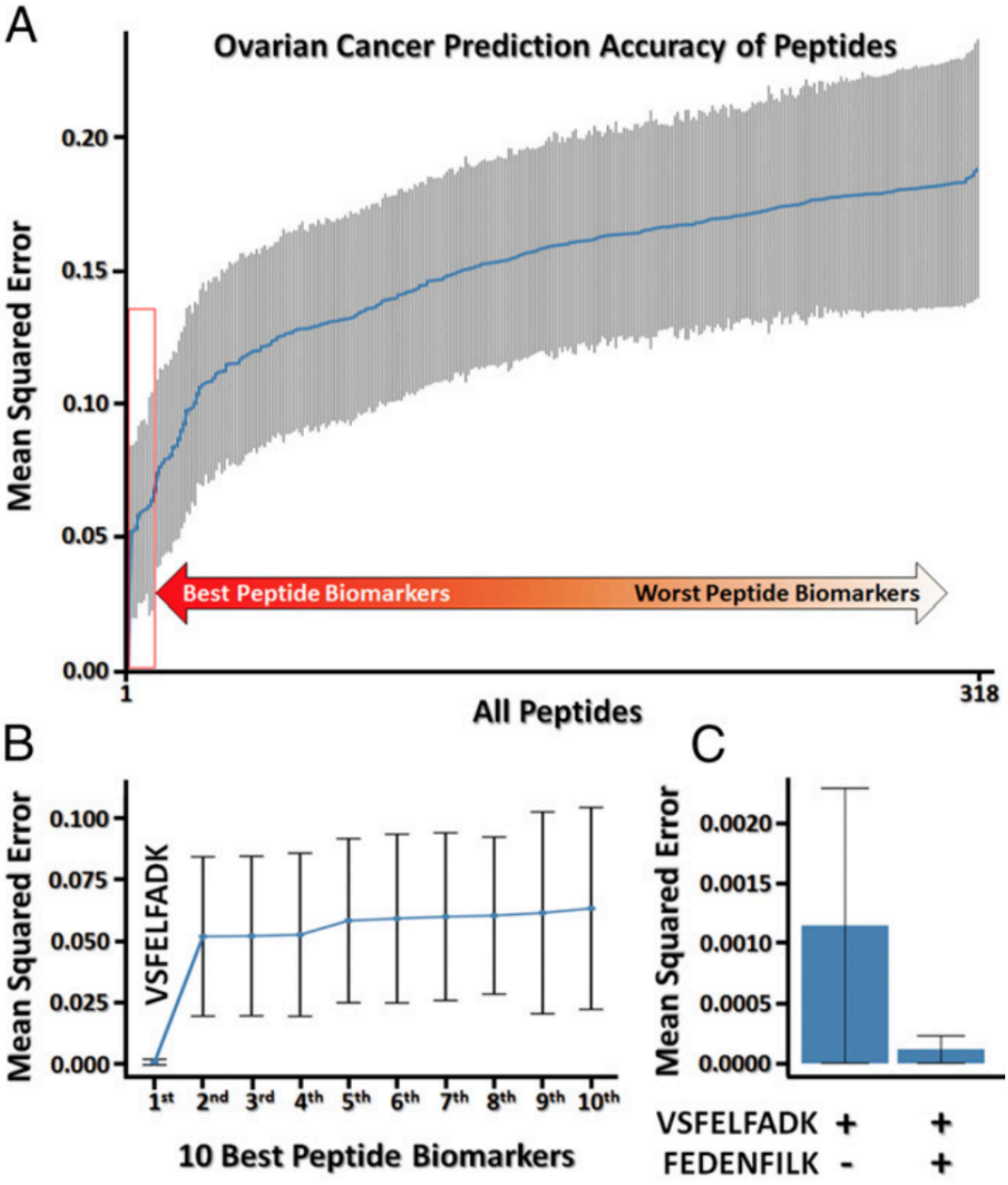

Figure 4.1: **(A)** Mean squared errors (MSEs) on the training set of LR models using each of the 318 peptides. Peptides are sorted from most predictive (lowest MSE) to least predictive. **(B)** A zoomed in view of the first ten peptides from (A). The peptide VSFELFADK from the PPIA gene is the strongest predictor. **(C)** Ovarian cancer prediction performance was further improved by additionally incorporating FEDENFILK into the LR model. Adapted from Wang et al. [36]

RerF were VVLAYEPVWAIGTGK and ICLDLQAPLYK, respectively. By reducing the dimensionality via Gini importance-based feature selection, RerF achieves identical classification performance to that of the logistic regression model. The performance of RF improves substantially, but still has a slightly higher FNR than the logistic regression model. The worse performance of RF compared to RerF when feature selection is performed prior to model fitting suggests that RerF may be better at identifying the predictive peptides. The error rates for LR, RF, and RerF with and without dimensionality reduction are compared in Figure 4.3. Notice that when using the full 318-dimensional data rather than performing feature selection to reduce the dimensionality, the LR classifier performs significantly worse than the RF and RerF classifiers, even having an unacceptable FPR (as deemed by the cancer experts) of about 3%. This suggests that RF and RerF are less affected by the curse of dimensionality than is LR.

4.4 Conclusion

In this chapter, we identified peptide biomarkers with high sensitivity and perfect specificity for predicting ovarian cancer. We compared a sequential forward selection logistic regression approach for learning a classifier with decision forest Gini importance-based feature selection approaches. The most predictive models were the LR model with the peptides VSFELFADK and FEDENFILK and the RerF model with VSFELFADK and ICLDLQAPLYK, both classifiers having 15.6% FNR and 0% FPR. The

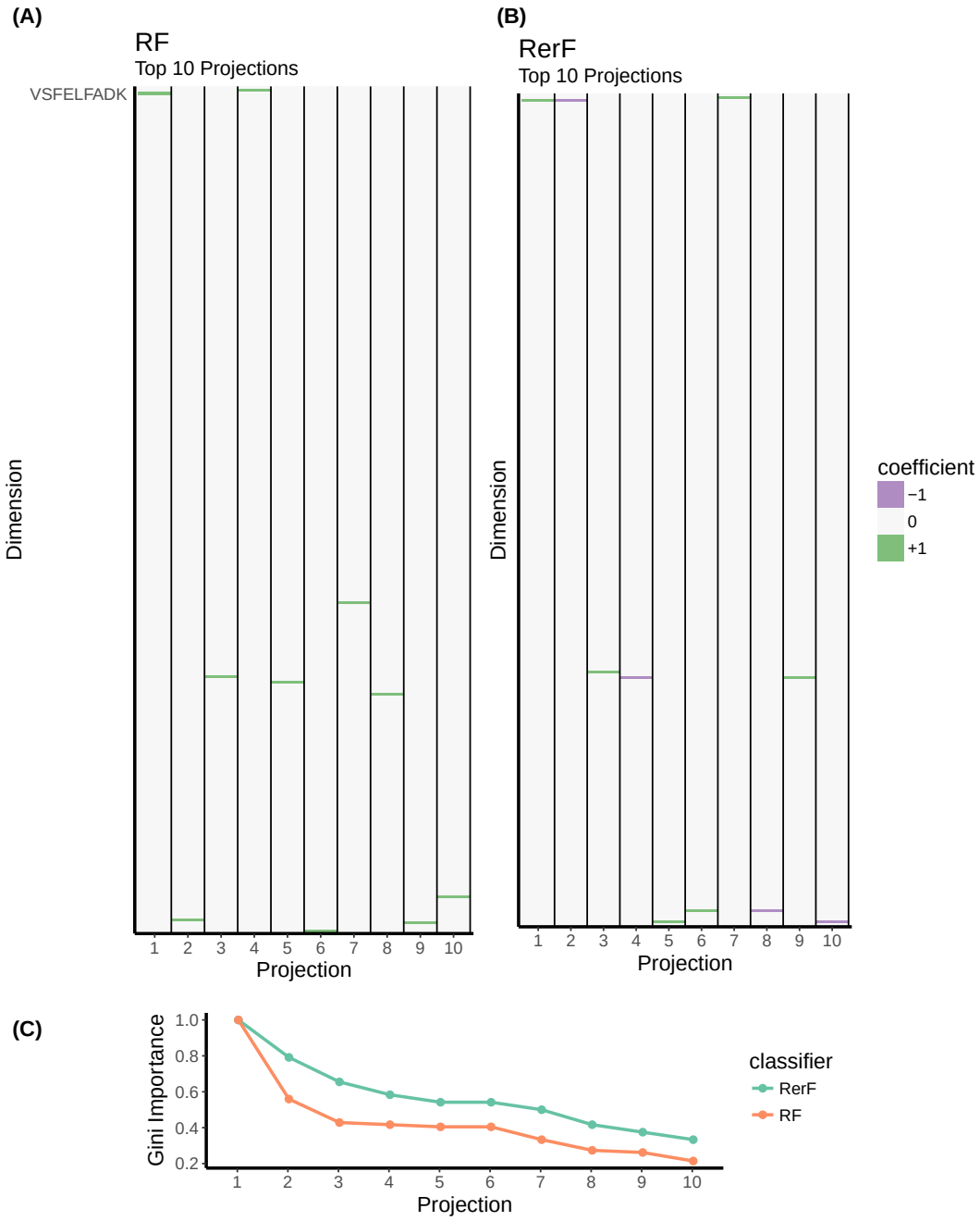

Figure 4.2: Top 10 projections. **(A)** Projections found by RF with highest Gini importance, **(B)**: Same as (A), but for RerF. **(C)**: Gini importances corresponding to the projections shown in (A) and (B).

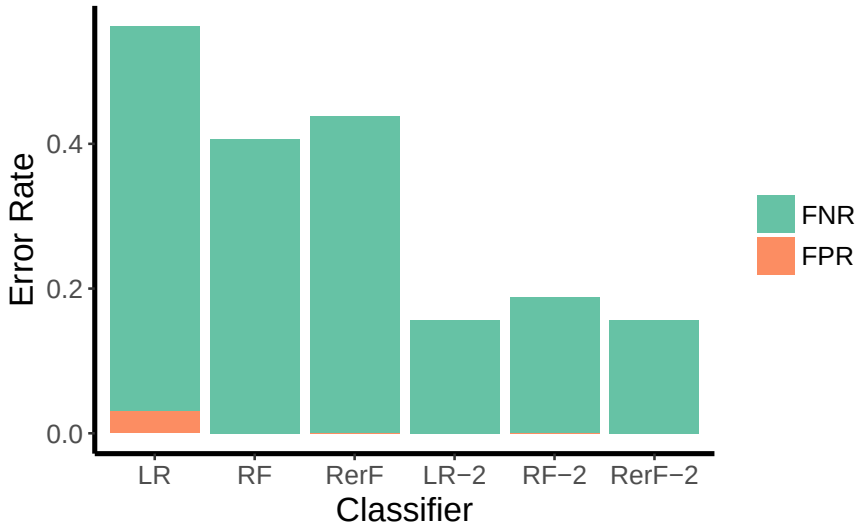

Figure 4.3: FPR and FNR of LR, RF, and RerF classifiers on the ovarian cancer data set. The "-2" suffix denotes dimensionality was reduced down to two dimensions prior to training the classifier.

worse performance of RF, even when Gini importance-based feature selection is performed, suggests that it does not identify salient peptides as accurately as RerF. Last, we show that when no feature selection is performed to reduce the dimensionality, the decision forest procedures are less affected by the curse of dimensionality. Therefore, RerFs demonstrates great promise as a tool for identifying salient features in large p small n classification problems.

Appendix A

Random Vectors in High Dimensions

Here we provide additional geometric motivation for the adoption of sparse random matrix distributions in the RerF framework. Let v^* be a hypothetical locally optimal split projection in p dimensions at a generic split node. We sample a random matrix A according to a specified distribution f_A and measure the angle of the closest univariate projection in A. If we repeat this many times, we can estimate the probability of sampling a projection whose direction comes within some angle Θ of v^*. We performed this experiment using both the f_A adopted by the default implementation of RerF and that adopted by RR-RF, which is simply a rotation matrix. For RerF, two values of d, which specifies the number of columns in A, were tested. Note for RR-RF that since A is a rotation matrix, it will always be $p \times p$. Two cases were tested for v^*. In one case, v^* has a single nonzero. That is, the split is sparse. In another case, v^* is all ones (dense). We repeated this experiment 10000 times for various values of p. Figure A.1 shows that when v^* is sparse, RerF has a high probability of sampling a

projection close in angle to v^* for all values of p, while RR-RF has a low probability of sampling a close projection. On the other hand, when v^* is dense and p is large, both RerF and RR-RF have a very low probability of sampling a projection close in angle to v^*. These results are in line with established theory on high-dimensional random vectors, which says that as the number of dimensions increases, the probability that two independent and isotropic random vectors are nearly orthogonal tends to one [37].

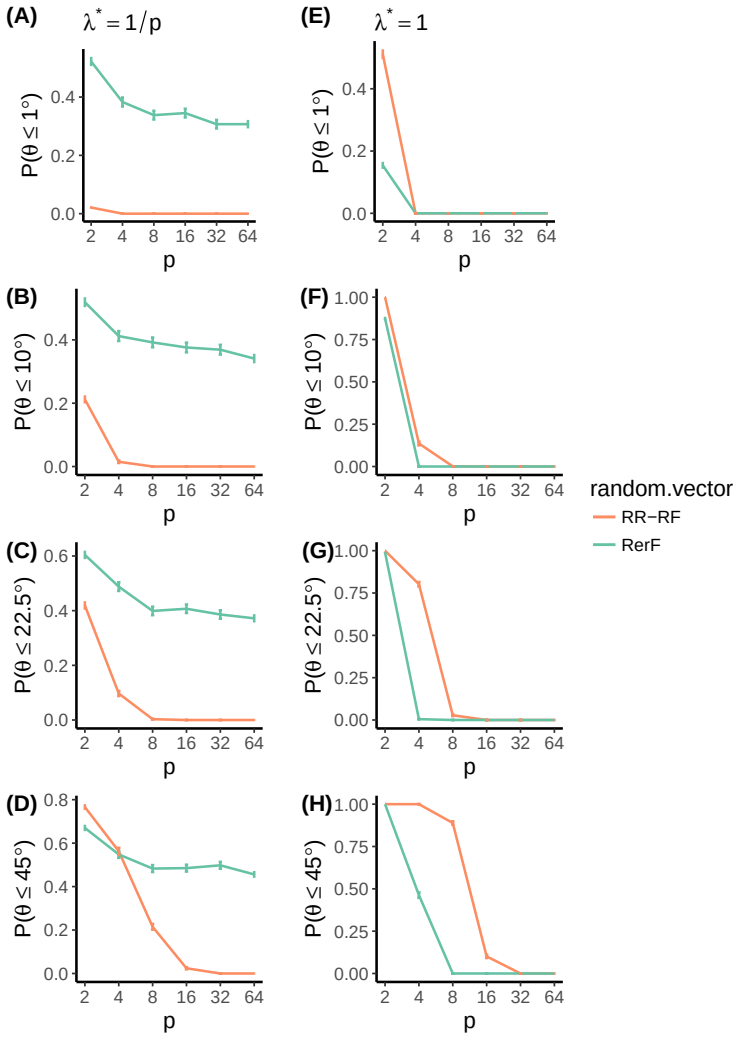

Figure A.1: The probability that RerF and RR-RF sample a projection within an angle θ of some hypothetical optimal node projection v^* in p dimensions when the density (number of nonzeros) λ^* of v^* is minimal ($\lambda^* = 1/p$) and when it is maximal ($\lambda^* = 1$) for varying values of θ and p. When the optimal projection is sparse (A - D), RerF has a reasonable probability of sampling projections close to it for all values of p. The probability of RR-RF sampling a close projection quickly degrades with increasing p. When the optimal projection is dense (E - H), both RerF and RR-RF have a low probability of sampling a close projection for $p \geq 16$. Therefore, when the number of dimensions is large, it may be safer to assume v^* is sparse and use a sampling distribution such as that adopted by RerF rather than the one adopted by RR-RF.

Appendix B

Statistical Theory

B.1 Proof of Theorem 3

Proof. This theorem essentially follows from Theorem 3.1 in Biau, Devroye, and Lugosi [25] by incorporating the random projection matrix: by Theorem 6.1 in Devroye, Györfi, and Lugosi [3], data-agnostic RerF (or any partition algorithm that is independent of the class label) is consistent if $diam(B_n(X)) \to 0$ and $N_n(X) \to \infty$, where $B_n(X)$ is the random partition (the child node of RerF) that contains X, and

$$N_n(X) = \sum_{i=1}^{n} \mathbf{I}(X_i \in B_n(X))$$

is the number of data points in the same partition as X, i.e., the number of training data in the same child node as X. Following the same step in [25], any random partition algorithm satisfies

$$Prob(N_n(X) < t) \leq (t-1)t_n/(n+1) \to 0$$

for any fixed $t > 0$ when $t_n/(n+1) \to 0$. Thus $N_n(X) \to \infty$, and it remains to show that the diameter of $B_n(X)$ converges to 0 in probability.

As $t_n \to \infty$, the number of partitions for each dimension of $B_n(X)$ increases to ∞, and since the partitions of data-agnostic RerF are randomly chosen for each dimension up-to a random projection, the size of each dimension of $B_n(X)$ is guaranteed to converge to 0 in probability. Therefore classification consistency holds for data-agnostic RerF. □

B.2 Bayes Error of Trunk's Problem Along a Univariate Projection

Suppose the prototypical pair $(X, Y) \in \mathbb{R}^p \times \{c_1, \ldots, c_K\}$ has joint distribution f_{XY}. Let L^* be defined as in Section 3.3.5, and $a \in \mathbb{R}^p$ be a projection vector. Then the projection $X' = \langle X, a \rangle \in \mathbb{R}$ of X onto a induces a joint distribution $f_{X'Y}$. The Bayes error with respect to $f_{X'Y}$ is denoted by L'^*.

In Trunk's problem, the task is to discriminate between two p-dimensional normal populations $N(\mu_0, \Sigma)$ and $N(\mu_1, \Sigma)$, where an observation comes from each population with equal probability and μ_0, μ_1 and Σ are described as in Section 3.2.2. The Bayes error for this problem is

$$\begin{aligned} L^* &= 1 - \Phi(\frac{1}{2}(\Delta^T \Sigma^{-1} \Delta)^{1/2}) \\ &= 1 - \Phi(\frac{1}{2}||\Delta||_2) \\ &= 1 - \Phi(||\mu_1||_2) \end{aligned}$$

where Φ is the standard normal cumulative distribution function and $\Delta =$

$\mu_1 - \mu_0$ [38]. Now suppose we have an arbitrarily oriented projection vector a. Without loss of generality, let $||a||_2 = 1$ (only direction of the projection will effect the Bayes error). Using the fact that for Trunk's problem, any vector projection of X is a sum of (scaled) independent random variables, it is straightforward to show that the Bayes error of the projection X' is

$$L'^* = 1 - \Phi(|\langle \mu_1, a \rangle|).$$

Appendix C

Pseudocode

Pseudocode 2 Pseudocode for finding the best split of the data. This function is called by growtree (Alg 1) at every split node. For each of the p dimensions in $\mathbf{X} \in \mathbb{R}^{p \times n}$, a binary split is assessed at each location between adjacent observations. The dimension j^* and split value τ^* in j^* that best split the data are selected. The notion of "best" means maximizing some choice in scoring function. In classification, the scoring function is typically the reduction in Gini impurity or entropy. The increment function called within this function updates the counts in the left and right partitions as the split is incrementally moved to the right.

Input: (1) $(\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{p \times n} \times \mathcal{Y}^n$, where $\mathcal{Y} = \{1, \ldots, K\}$
Output: (1) dimension j^*, (2) split value τ^*

function $(j^*, \tau^*) =$ FINDBESTSPLIT$(\mathbf{X}, \mathbf{y})$
  for $j = 1, \ldots, p$ **do**
    Let $\mathbf{x}^{(j)} = (x_1^{(j)}, \ldots, x_n^{(j)})$ be the jth row of $\mathbf{X}$.
    $\{m_i^j\}_{i \in [n]} = \text{sort}(\mathbf{x}^{(j)})$ ▷ m_i^j is the index of the i^{th} smallest value in $\mathbf{x}^{(j)}$
    $t = 0$ ▷ initialize split to the left of all observations
    $n' = 0$ ▷ number of observations left of the current split
    $n'' = n$ ▷ number of observations right of the current split
    for $k = 1, \ldots, K$ **do**
      $n_k = \sum_{i=1}^{n} I[y_i = k]$ ▷ total number of observations in class k
      $n'_k = 0$ ▷ number of observations in class k left of the current split
      $n''_k = n_k$ ▷ number of observations in class k right of the current split
    end for
    for $t = 1, \ldots, n-1$ **do** ▷ assess split location, moving right one at a time
      $(\{(n'_k, n''_k)\}, n', n'', y_{m_t^j}) = \text{increment}(\{(n'_k, n''_k)\}, n', n'', y_{m_t^j})$
      $Q^{(j,t)} = \text{score}(\{(n'_k, n''_k)\}, n', n'')$ ▷ measure of split quality
    end for
  end for
  $(j^*, t^*) = \underset{j,t}{\text{argmax}}\, Q^{(j,t)}$
  for $i = 0, 1$ **do** $c_i = m_{t^*+i}^{j^*}$ **end for**
  $\tau^* = \frac{1}{2}(x_{c_0}^{(j^*)} + x_{c_1}^{(j^*)})$ ▷ compute the actual split location from the index j^*
  return (j^*, τ^*)
end function

Pseudocode 3 Pseudocode for predicting the class label associated with an input $\mathbf{x} \in \mathbb{R}^p$ using a RerF decision tree constructed using the function growtree (Alg 1) . A decision tree is a sequence of M nodes, each node being defined by a split projection $\mathbf{a}$, a split value τ, the indices of left and right child nodes $\kappa = \{\kappa_L, \kappa_R\}$, and the class counts $\{n_k\}_{k \in \mathcal{Y}}$

Input: (1) $\mathbf{x} \in \mathbb{R}^p$, (2) $T = \{(\mathbf{a}^{(m)}, \tau^{(m)}, \kappa^{(m)}, \{n_k^{(m)}\})\}_{m=1}^{M}$
Output: (1) Predicted class label $\widehat{y}$

function $\widehat{y} =$ PREDICT$(T, \mathbf{x})$
  $m = 1$
  while $\kappa^{(m)} \neq$ NULL **do** ▷ move down the tree until a leaf node is reached
    $\widetilde{x} = \mathbf{a}^{(m)} \cdot \mathbf{x}$
    if $\widetilde{x} \leq \tau^{(m)}$ **then**
      $m = \kappa_L^{(m)}$
    else
      $m = \kappa_R^{(m)}$
    end if
  end while
  return $\underset{k \in \mathcal{Y}}{\operatorname{argmax}}\, n_k^{(m)}$ ▷ prediction is the most populous class at the leaf node
end function

References

[1] Thomas G Dietterich et al. “Ensemble methods in machine learning”. In: *Multiple classifier systems* 1857 (2000), pp. 1–15.

[2] Robi Polikar. “Ensemble learning”. In: *Ensemble machine learning*. Springer, 2012, pp. 1–34.

[3] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*. Vol. 31. Springer Science & Business Media, 2013.

[4] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[5] T. K. Ho. “The random subspace method for constructing decision forests”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20.8 (1998), pp. 832–844. ISSN: 0162-8828. DOI: 10.1109/34.709601.

[6] Yali Amit and Donald Geman. “Shape quantization and recognition with randomized trees”. In: *Neural computation* 9.7 (1997), pp. 1545–1588.

[7] L. Breiman. “Random Forests”. In: *Machine Learning* 4.1 (2001), pp. 5–32.

[8] D. Heath, S. Kasif, and S. Salzberg. “Induction of Oblique Decision Trees”. In: *Journal of Artificial Intelligence Research* 2.2 (1993), pp. 1–32.

[9] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. “Rotation forest: A new classifier ensemble method”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.10 (2006), pp. 1619–1630.

[10] B. H. Menze, B.M Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht. “On Oblique Random Forests”. English. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis. Vol. 6912. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 453–469. ISBN: 978-3-642-23782-9. DOI: 10.1007/978-3-642-23783-6_29. URL: http://dx.doi.org/10.1007/978-3-642-23783-6_29.

[11] Rico Blaser and Piotr Fryzlewicz. “Random Rotation Ensembles”. In: *Journal of Machine Learning Research* 17.4 (2016), pp. 1–26.

[12] Tom Rainforth and Frank Wood. “Canonical correlation forests”. In: *arXiv preprint arXiv:1507.05444* (2015).

[13] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim. “Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?” In: *Journal of Machine Learning Research* 15.1 (2014), pp. 3133–3181.

[14] R. Caruana, N. Karampatziakis, and A. Yessenalina. “An Empirical Evaluation of Supervised Learning in High Dimensions”. In: *Proceedings of the 25th International Conference on Machine Learning* (2008).

[15] G. V. Trunk. “A problem of dimensionality: A simple example”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 3 (1979), pp. 306–307.

[16] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001.

[18] Ronald L Iman and William J Conover. “The use of the rank transform in regression”. In: *Technometrics* 21.4 (1979), pp. 499–509.

[19] WJ Conover and Ronald L Iman. “The rank transformation as a method of discrimination with some examples”. In: *Communications in Statistics-Theory and Methods* 9.5 (1980), pp. 465–487.

[20] William J Conover and Ronald L Iman. “Rank transformations as a bridge between parametric and nonparametric statistics”. In: *The American Statistician* 35.3 (1981), pp. 124–129.

[21] William J Conover and Ronald L Iman. “Analysis of covariance using the rank transformation”. In: *Biometrics* (1982), pp. 715–724.

[22] Tyler M Tomita, Mauro Maggioni, and Joshua T Vogelstein. “ROFLMAO: Robust Oblique Forests with Linear MAtrix Operations”. In: *SIAM Data Mining*. 2017.

[23] Gareth M James. “Variance and bias for general loss functions”. In: *Machine Learning* 51.2 (2003), pp. 115–135.

[24] Charles J Stone. “Consistent nonparametric regression”. In: *The annals of statistics* (1977), pp. 595–620.

[25] G. Biau, L. Devroye, and G. Lugosi. “Consistency of random forests and other averaging classifiers”. In: *The Journal of Machine Learning Research* 9 (2008), pp. 2015–2033.

[26] Gilles Louppe. “Understanding random forests: From theory to practice”. In: *arXiv preprint arXiv:1407.7502* (2014).

[27] *rerf: Randomer Forest*. https://cran.r-project.org/web/packages/rerf/.

[28] *ranger: A fast Implementation of Random Forests*. https://cran.r-project.org/web/packages/ranger/.

[29] *xgboost: Extreme Gradient Boosting*. https://cran.r-project.org/web/packages/xgboost/.

[30] *The MNIST Database of Handwritten Digits*. http://yann.lecun.com/exdb/mnist/.

[31] *Kaggle: Higgs Boson Machine Learning Challenge*. https://www.kaggle.com/c/higgs-boson.

[32] *UCI Machine Learning Repository: p53 Dataset*. https://archive.ics.uci.edu/ml/dataset

[33] Gérard Biau, Erwan Scornet, and Johannes Welbl. “Neural Random Forests”. In: *arXiv preprint arXiv:1604.07143* (2016).

[34] *American Cancer Society: Ovarian Cancer*. https://www.cancer.org/cancer/ovarian-cancer/.

[35] Lynn AG Ries, D Harkins, M Krapcho, Angela Mariotto, Barry A Miller, Eric J Feuer, Limin X Clegg, MP Eisner, Marie-Josèphe Horner, Nadia Howlader, et al. “SEER cancer statistics review, 1975-2003”. In: (2006).

[36] Qing Wang, Ming Zhang, Tyler Tomita, Joshua T Vogelstein, Shibin Zhou, Nickolas Papadopoulos, Kenneth W Kinzler, and Bert Vogelstein. “Selected reaction monitoring approach for validating peptide biomarkers”. In: *Proceedings of the National Academy of Sciences* (2017), p. 201712731.

[37] Roman Vershynin. *High Dimensional Probability: An Introduction with Applications in Data Science*. 2017. URL: http://www-personal.umich.edu/~romanv/papers/HDP-book/HDP-book.pdf.

[38] P. J. Bickel and E. Levina. “Some theory for Fisher’s linear discriminant function, ‘naive Bayes’, and some alternatives when there are many more variables than observations”. In: *Bernoulli* 10.6 (2004), pp. 989–1010. DOI: 10.3150/bj/1106314847. URL: http://dx.doi.org/10.3150/bj/1106314847.

Tyler M. Tomita

17 E Centre St, Apt 7
Baltimore, MD 21202
Phone: (925) 596-1688

Email: ttomita@jhu.edu
Web: https://github.com/ttomita

Education

2011–2018 Johns Hopkins University, Baltimore, MD
Ph.D. in Biomedical Engineering
Thesis Title: Generalized Linear Splitting Rules in Decision Tree Ensembles
Advisor: Joshua T. Vogelstein

2005–2010 University of California, Davis
B.S. in Biomedical Engineering and Biological Systems Engineering, *cum laude*

Research Experience

2011–present Department of Biomedical Engineering, Johns Hopkins University
Doctoral Research
Research Advisor: Joshua T. Vogelstein

- Developed a novel machine learning algorithm called Randomer Forest for general-purpose classification.
- Demonstrated superior accuracy of Randomer Forest to that of existing state-of-the-art classification algorithms.
- Investigated the statistical and computational properties of the algorithm.
- Created a memory-efficient parallelized R implementation and package openly available on the Comprehensive R Archive Network (CRAN).

Teaching Assistant

2014 EN.580.321 Statistical Mechanics and Thermodynamics, Johns Hopkins University
2013 EN.580.321 Statistical Mechanics and Thermodynamics, Johns Hopkins University

Manuscript Preprints

Tomita, T.M., Browne, J., Maggioni, M., Vogelstein, J.T. Randomer Forests. URL: https://arxiv.org/pdf/1506.03410.pdf

Peer-Reviewed Publications

[1] **Tomita, T.M.**, Maggioni, M., Vogelstein, J.T. (2017). ROFLMAO: Robust Oblique Forests with Linear Matrix Operations. Proceedings of the 2017 SIAM International Conference on Data Mining, 498–506. URL: http://epubs.siam.org/doi/pdf/10.1137/1.9781611974973.56

[2] Wang, Q., Zhang, M., **Tomita, T.**, Vogelstein, J.T., Zhou, S., Papadopoulos, N., Kinzler, K.W., Vogelstein, B. (2017) A Selected Reaction Monitoring Approach for Validating Candidate Biomarkers. *Forthcoming in* Proceedings of the National Academy of Sciences.

[3] Masica, D.L., Dal Molin, M., Wolfgang, C.L., **Tomita, T.**, Ostovaneh, M.R., Blackford, A., Moran, R.A., Law, J.K., Barkley, T., Goggins, M. Irene Canto, M. (2016). A novel approach for selecting combination clinical markers of pathology applied to a large retrospective cohort of surgically resected pancreatic cysts. Journal of the American Medical Informatics Association, 24(1), 145-152.

[4] Springer, S., Wang, Y., Dal Molin, M., Masica, D.L., Jiao, Y., Kinde, I., Blackford, A., Raman, S.P., Wolfgang, C.L., **Tomita, T.**, Niknafs, N. (2015). A combination of molecular markers and clinical features improve the classification of pancreatic cysts. Gastroenterology, 149(6), 1501-1510.

[5] Sumida, G.M., **Tomita, T.M.**, Shih, W., Yamada, S. (2011). Myosin II activity dependent and independent vinculin recruitment to the sites of E-cadherin-mediated cell-cell adhesion. BMC cell biology, 12(1), 48.

Contributed Presentations

Tomita, T.M. (2017, April). ROFLMAO: Robust Oblique Forests with Linear Matrix Operations. 2017 SIAM Interational Conference on Data Mining, Houston, TX, USA.

Posters

Tomita, T.M. (2017, April). ROFLMAO: Robust Oblique Forests with Linear Matrix Operations. 2017 SIAM Interational Conference on Data Mining, Houston, TX, USA.

Honors and Awards

2009 Tau Beta Pi Engineering Honor Society inductee

2008 Robert Roy Owen Scholarship recipient

Software Proficiencies

Python, R, MATLAB, C++, Linux, git