# ROFLMAO: Robust Oblique Forests with Linear MAtrix Operations[*]

Tyler M. Tomita[†]
ttomita@jhu.edu

Mauro Maggioni[‡]
mauro@math.jhu.edu

Joshua T. Vogelstein[†]
jovo@jhu.edu

## Abstract

Random Forest (RF) remains one of the most widely used general purpose classification methods. Two recent large-scale empirical studies demonstrated it to be the best overall classification method among a variety of methods evaluated. One of its main limitations, however, is that it is restricted to only axis-aligned recursive partitions of the feature space. Consequently, RF is particularly sensitive to the orientation of the data. Several studies have proposed "oblique" decision forest methods to address this limitation. However, these methods either have a time and space complexity significantly greater than RF, are sensitive to unit and scale, or empirically do not perform as well as RF on real data. One promising oblique method that was proposed alongside the canonical RF method, called Forest-RC (F-RC), has not received as much attention by the community. Despite it being just as old as RF, virtually no studies exist investigating its theoretical or empirical performance. In this work, we demonstrate that F-RC empirically outperforms RF and another recently proposed oblique method called Random Rotation Random Forest, while approximately maintaining the same computational complexity. Furthermore, a variant of F-RC which rank transforms the data prior to learning is especially invariant to affine transformations and robust to data corruption. Open source code is available.

## 1 Introduction

Data science is becoming increasingly important as our ability to collect and process data continues to increase. Supervised learning— the art and science of using labeled training data to make predictions about some target data—is of special interest to many applications, ranging from science to government to industry. Classification, a special case of supervised learning, in which the target data is categorical, is one of the most fundamental learning problems, and has been the subject of much study. A simple Pubmed search for the term "classifier" reveals nearly 10,000 publications. One of the most popular and best performing classifiers is random forests (RFs) [1, 3]. Two recent benchmark papers assess the performance of many different classification algorithms on many different datasets [9, 4], and both concluded the same thing: RFs are the best classifier.

RF typically refers to what Breiman called Forest-RI—an ensemble of axis-parallel, or orthogonal decision trees [11, 14]. In these types of trees, the feature space is recursively split along directions parallel to the axes of the feature space. Thus, in cases in which the classes seem inseparable along any single dimension, RF may be suboptimal. To address this drawback, several variations of "oblique" decision forests have been proposed, including efforts to learn good projections [11, 16], using principal components analysis to find the directions of maximal variance [12, 15], or directly learning good discriminant directions [14]. Another recently proposed method, called Random Rotation RF (RR-RF), uniformly randomly rotates the data for every decision tree in the ensemble prior to inducing the tree [2]. While all of these recent approaches deal with rotational invariance, they fail to address several important issues that RF can natively handle:

1. **Scale Invariance** By linearly combining variables when generating candidate oblique splits, unit and scale invariance is lost.

2. **Efficiency** As our ability to collect and process data continues to increase, we are encountering increasingly massive datasets. Therefore, time- and space-efficient methods are becoming more and more important.

3. **Data Corruption** - RF is known to be relatively robust to corrupted data. This property has not been investigated in any of the oblique methods, but it is likely that at least some of these methods will be more sensitive to data corruption.

4. **Sparsity** - In real world data, the proportion of features that are irrelevant is often large, especially for high dimensional data, giving rise to optimal decision boundaries that are sparse. In such cases, oblique methods that use a subset of features for making oblique splits would be expected to perform better than "dense" methods.

[*] Supported by Defense Advanced Research Projects Agency (DARPA) SIMPLEX program through SPAWAR contract N66001-15-C-4041 and DARPA GRAPHS N66001-14-1-4028

[†] Department of Biomedical Engineering, Institute for Computational Medicine, Center for Imaging Science, Johns Hopkins University

[‡] Department of Mathematics, Johns Hopkins University

In addition to proposing RF, Breiman also proposed and characterized Forest-RC (F-RC), which used linear combinations of features rather than individual features to split along. F-RC addresses the issue of sparsity listed above by controlling the number of features that are linearly combined via a user-specified parameter. Additionally, in theory, the time- and space-efficiency of F-RC can be very similar to that of RF under the right conditions. Breiman found in his studies on synthetic datasets that F-RC "compares more favorably ... than Forest-RI," yet it was never investigated further in the literature. Furthermore, to our knowledge, no openly available software implementation of F-RC exists. We believe F-RC has not garnered the attention it deserves.

In this study, we sought to identify an oblique method that does not suffer from the pathologies enumerated above. We hypothesized that a sparse variant of F-RC that rank transforms the data prior to inducing the forest (called Frank) would have the aforementioned desiderata and would exhibit superior performance and robustness over other decision forest methods across a wide range of settings. To this end, we explore the relative performance of several decision forest methods under different conditions using synthetic datasets as well as the large suite of benchmark datasets used in Fernandez-Delgado et al. (2014) [9] (hereafter referred to as FD14). We demonstrate that Frank is the only RF method that has all of the desired properties, both in simulations and real data. We provide an open source MATLAB implementation to facilitate further investigations and generalizations of these methods.

## 2 Methodology

### 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 [17]. 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 Benchmark Datasets

In addition to the simulations, all classification methods were evaluated on 114 of the 121 datasets as described in FD14 [9]. 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.

### 2.3 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 $\{10^2, 10^3, 10^4, 10^5\}$.

### 2.4 Classification Algorithms

The algorithms examined in this study were RF, F-RC, and 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, 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 [2]. 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.

### 2.5 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 $x_{(1)}, \dots, x_{(n)}$ 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.

### 2.6 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 [9] 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.

### 2.7 Classifier Background

Let $\mathcal{D}^n = \{(X_i, Y_i) : i \in [n]\}$ be a given dataset, where $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{Y} = \{c_1, \dots, c_K\}$. A classifier is a function $g : \mathcal{X} \mapsto \mathcal{Y}$ learned from $\mathcal{D}^n$ 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 $g(\cdot\,|\mathcal{D}^n)$ such that $P(g(X\,|\mathcal{D}^n) \neq Y)$ is minimized. The optimal classification rule is the Bayes classifier $g^*(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, P(Y = y|X = x)$ [8].

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 $\hat{P}(Y|X\,;\mathcal{D}^n)$ and use those as surrogates, leading to the plug-in estimate of the Bayes classifier $\hat{g}(x\,;\mathcal{D}^n) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, \hat{P}(Y = y|X = x\,;\mathcal{D}^n)$. 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

$\bar{g}(\cdot\,|\mathcal{D}^n)$ is an ensemble of $T$ randomized decision trees, where each tree $g_t(\cdot\,|\mathcal{D}_t)$ is trained on a bootstrap sample of the data $\mathcal{D}_t$. The output of $\bar{g}$ is the majority vote. That is, $\bar{g}(x\,|\mathcal{D}^n) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{t=1}^{T} \mathbb{I}(g_t(x,\,|\mathcal{D}^t) = y) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \frac{1}{T} \sum_{t=1}^{T} I(g_t(x,\,|\mathcal{D}^t) = y)$, where $\mathbb{I}(g_t(x,\,|\mathcal{D}^t) = y)$ 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 $\bar{g}(x\,|\mathcal{D}^n) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \hat{P}(Y = y|X = x\,;\mathcal{D}^n)$.

## 3 Results

### 3.1 Comparison of Classification Methods on Simulated 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 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.

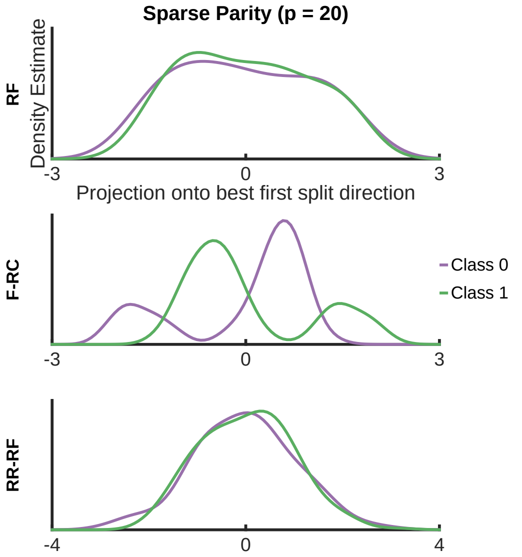

Figure 1: Distribution of the sparse parity data when projected onto the best split direction found by RF (top), F-RC (center), and RR-RF (bottom). For RF, all twenty dimensions were evaluated in the search for the best split. For F-RC, 8000 random linear combinations of three variables were evaluated. 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.

Figure 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 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 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 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 of trees were empirically

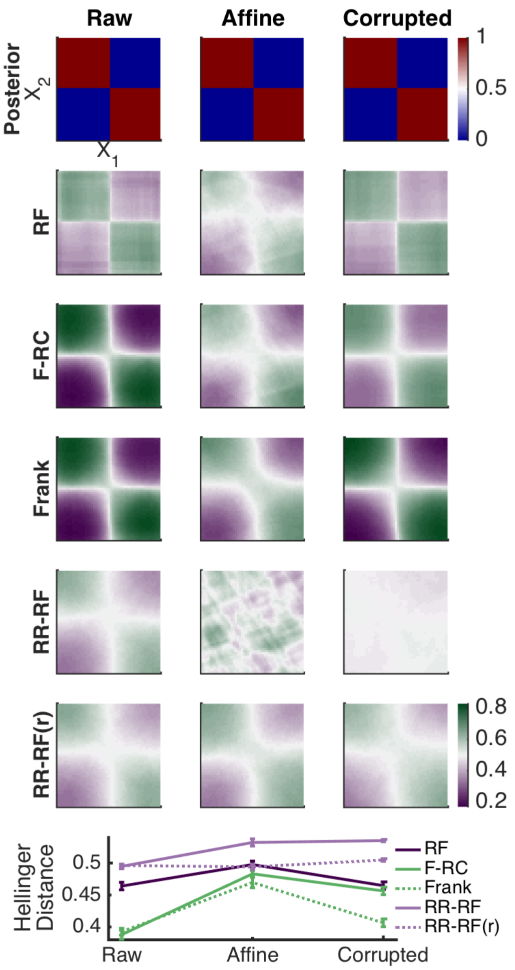

Figure 2: Posteriors and classifier estimates of posteriors for the sparse parity problem. Also plotted below are Hellinger distances between estimates and the true posteriors. Oblique forests are especially sensitive to relative scale of predictor variables and data corruption. Rank transforming the data simultaneously robustifies oblique methods against incommensurable features and data corruption.

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 $p = 500$.

### 3.2 Theoretical Space and Time Complexity

For a RF, let $T$ denote the number of trees, $n$ the number of data points, and $d$ the number of features sampled at each node. Sorting the data for each candidate split at each node requires $\mathcal{O}(n_k \log n_k)$ time, where $n_k$ is the number of data points at the $k^{th}$ split node. Let's assume the worst-case scenario, which happens if the proportions of data points in each class are completely balanced and if

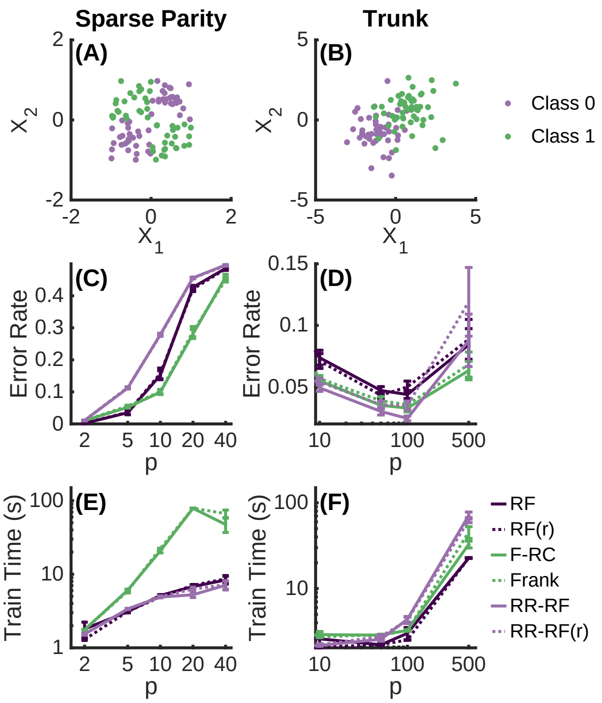

Figure 3: Sparse parity (A,C,E) and Trunk (B,D,F) simulations (see section 2.1 for details).

every split results in one child node having a single data point and the other child node having $n_k - 1$ data points. If we assume each tree grows until terminal nodes have only $\mathcal{O}(1)$ data points, then there are $\mathcal{O}(n)$ nodes. In this case, the average value of $n_k$ is $n/2$. The complexity of constructing the RF in this case, disregarding cross-validation or other randomization techniques for preventing overfitting, is $\mathcal{O}(Tdn^2 \log n)$. Storing RF requires $\mathcal{O}(Tn)$ space because each node can be represented by the index of the nonzero coordinate.

F-RC can have a similar time and space complexity, depending on the values of the parameters. Specifically, assume that F-RC also has $T$ trees, and $n$ data points per tree, and no pruning, so $\mathcal{O}(n)$ nodes. F-RC defines $d$ new features at each split node, each feature being a linear combination of $L$ original features. The matrix multiplication for this requires $\mathcal{O}(Ldn)$ time because it is a sparse matrix multiplication. F-RC also takes another $\mathcal{O}(n_k \log n_k)$ time to find the best split for each of the $d$ features. Thus, in the worst-case scenario, F-RC requires $\mathcal{O}(Tdn^2(L + \log n))$ time to train. When $L$ is small, then the complexity approximates that of RF. As mentioned previously, unlike RF, F-RC can take values of $d$ greater than $p$. Therefore, when the optimal value of $d$ happens to be greater than $p$, F-RC will have a longer training time than RF. This explains why the F-RC variants take longer than the RF variants in the sparse parity experiments (Figure 3E) but not in the Trunk experiments (Figure 3F). To store the resulting F-RC requires $\mathcal{O}(TLn)$ space, because each node can be represented by the indices of the coordinates that received a non-zero element, and the number of such indices is $L$.

These complexities are in stark contrast to other oblique methods. RR-RFs, in particular, require a QR decomposition having a time complexity of $\mathcal{O}(p^3)$ in order to generate a random rotation matrix for each tree. Rotating the data matrix prior to inducing each tree additionally requires $\mathcal{O}(np^2)$. Therefore, RR-RF in the worst-case scenario requires $\mathcal{O}(T(p^3 + np^2 + dn^2 \log n))$ time. The $p^3$ term can explain the trend for the RR-RF variants in Figure 3F. In addition to storing all of the trees, RR-RF has to store a rotation matrix for each tree, resulting in an overall space complexity of $\mathcal{O}(T(n + p^2))$.

### 3.3 Effects of Transformations

Figure 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 2.5 for details). Figure S1 (supplement available here) 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.

### 3.4 Benchmark Data

Next we compared performance of the RF, F-RC, and RR-RF variants on 114 benchmark datasets (refer to section 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

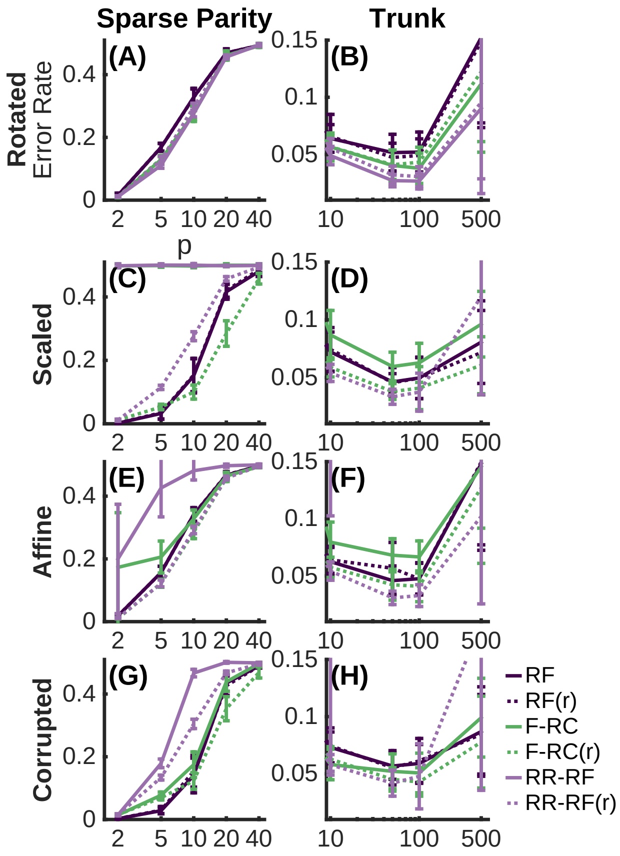

Figure 4: The effects of different transformations applied to the sparse parity (left column) and Trunk (right column) simulations on classification performance (see section 2 for details). Specifically, we consider rotations, scalings, affine transformations, and corruptions.

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 5. 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

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

Table 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)

normalized relative error was exactly 0. Table 1 summarizes which methods perform significantly better or worse than RF across the 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 S2 shows that unlike percentile normalization and z-scoring, rank transformations helps against data corruption. Overall, FRANK is the best performing method.

## 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 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 Hastie et al. [10]. 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,

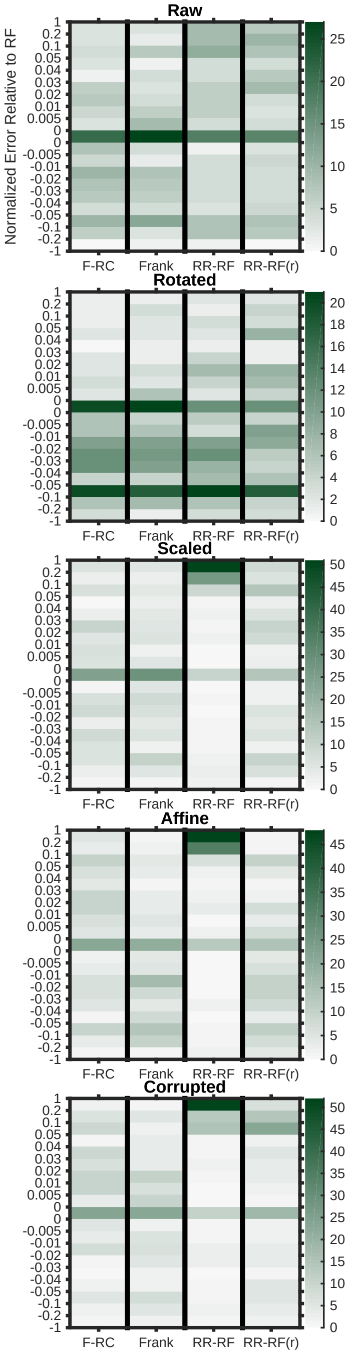

Figure 5: Histograms of error relative to RF normalized by chance probability of error on 114 benchmark datasets with various transformations applied. A negative normalized relative error indicates better performance than RF, while a positive value indicates worse performance. Bin edges are indicated by ticks on the y-axis. Color indicates the absolute frequency in a particular bin.

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 [13, 7, 5, 6]. 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.

## 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 our work is two-fold. First, it augments the large-scale empirical study conducted by FD14 [9]. 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.

To conclude, our hope is that the intuition provided in this work will help data scientists choose the most suitable classification method for their tasks at hand, and that it will help steer the development of stronger, efficient, and more robust decision forest methods. Open source code is available at https://github.com/ttomita/RandomerForest.

## 6 References

[1] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997.

[2] Rico Blaser and Piotr Fryzlewicz. Random rotation ensembles. *Journal of Machine Learning Research*, 17(4):1–26, 2016.

[3] L. Breiman. Random forests. *Machine Learning*, 4(1):5–32, October 2001.

[4] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[5] William J Conover and Ronald L Iman. Rank transformations as a bridge between parametric and nonparametric statistics. *The American Statistician*, 35(3):124–129, 1981.

[6] William J Conover and Ronald L Iman. Analysis of covariance using the rank transformation. *Biometrics*, pages 715–724, 1982.

[7] WJ Conover and Ronald L Iman. The rank transformation as a method of discrimination with some examples. *Communications in Statistics-Theory and Methods*, 9(5):465–487, 1980.

[8] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

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

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[11] D. Heath, S. Kasif, and S. Salzberg. Induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(2):1–32, 1993.

[12] T. K. Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, Aug 1998.

[13] Ronald L Iman and William J Conover. The use of the rank transform in regression. *Technometrics*, 21(4):499–509, 1979.

[14] B. H. Menze, B.M Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht. On oblique random forests. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6912 of *Lecture Notes in Computer Science*, pages 453–469. Springer Berlin Heidelberg, 2011.

[15] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1619–1630, 2006.

[16] P. J. Tan and D. L. Dowe. Mml inference of oblique decision trees. In *AI 2004: Advances in Artificial Intelligence*, pages 1082–1088. Springer, 2005.

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