Robust Oblique Forests

Anonymous Author(s)
Affiliation
Address
email

Abstract

Random Forest (RF) remains one of the most widely used general purpose classification methods, due to its tendency to perform well in a variety of settings. 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, the ways in which these methods address this issue compromise many of the nice properties that RF possesses. In particular, unlike RF, these methods either don't deal with incommensurate predictors, aren't well-adapted to problems in which the number of irrelevant features are overwhelming, have a time and space complexity significantly greater than RF, or require additional hyperparameters to be tuned, rendering training of the classifier more difficult. In this work, we establish a generalized forest building scheme, random projection forests. Random forests and many other currently existing decision forest algorithms can be viewed as special cases of this scheme. With this scheme in mind, we propose some special cases which we call randomer forests (RerFs). We demonstrate that RerF emprically outperforms RF and other oblique methods on simulations and on benchmark data, and that one variant of RerF is especially robust to affine transformations of the data. We also show that RerF scales comparably to RF in terms of time and space complexity. We conclude that RerF is a competitive alternative to RF and existing oblique forest methods. Open source code will be available.

1 Introduction

Data science is becoming increasingly important as our ability to collect and process data continues to increase. Supervised learning—the act of using predictors to make a prediction 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) [2]. Two recent benchmark papers assess the performance of many different classification algorithms on many different datasets [5, 3], and both concluded the same thing: RFs are the best classifier.

RF typically refers to Breiman's Forest-RI, which uses axis-parallel [6], or orthogonal trees [10]. That is, 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, Breiman also proposed and characterized Forest-RC (F-RC), which used linear combinations of coordinates rather than individual coordinates, to split along. Others have studied several different variants of "oblique" decision forests, including efforts to learn good projections [6, 13], using principal components analysis to find the directions of maximal variance

Submitted to 29th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

[7, 11], or directly learning good discriminant directions [10]. 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 [1]. While all of these recent approaches deal with rotational invariance, they fail to address several important issues:

1. By linearly combining variables when generating candidate oblique splits, unit and scale invariance is lost.
2. 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, we say that the signal is *compressive*. To our knowledge, all of the proposed oblique decision forests, with the exception of Forest-RC if tuned appropriately, do not impose a constraint on the sparsity of recursive partitions of the feature space. Therefore, such methods may be suboptimal in cases in which the signal is compressive.
3. 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. With the exception of F-RC, all oblique methods require expensive computation and/or storage.

There is therefore a gap that calls for the construction of a scalable oblique decision forest classifier that maintains scale invariance and performs well in the presence of an overwhelming number of irrelevant features.

To address this, we first state a generalized forest building scheme, random projection forests, which includes all of the above algorithms as special cases. This enables us to formally state our objective, and provides a lens that enables us to propose a few novel special cases, which we refer to as "*Randomer Forests*" (RerFs), for reasons that will become clear in the sequel. We show empirically that our methods are robust to affine transformations and outperform both RF and RR-RF on both synthetic datasets and a suite of benchmark datasets. Additionally we both theoretically and empirically demonstrate that our methods have the same time and space complexity as random forests. To conclude, we propose RerFs as a competitive alternative to RF and other oblique decision forest methods. Open source code will be made available.

2 Random Projection Forests

Let $\mathbf{X} \in \mathbb{R}^p$ be a vector of p random predictor variables and $Y \in \mathcal{Y} = \{\widetilde{y}_1, \ldots, \widetilde{y}_C\}$ be a categorical random variable. Suppose $\mathbf{X}$ and Y are jointly distributed according to some unknown distribution and we observe a training set $\mathcal{D}^n = \{(\mathbf{x}_i, y_i) : i \in [n]\}$. A classification forest $\bar{g}(\mathbf{x}|\mathcal{D}^n)$ is an ensemble of L decision trees, where each tree $g^l(\mathbf{x}|\mathcal{D}^l)$ is trained on a subset of the data $\mathcal{D}^l \subset \mathcal{D}^n$. Random projection forests are a special case of classification forests that subsume all of the strategies mentioned above (see Pseudocode 1). The key idea of all of them is that at each node of each tree, we have a set of predictor data points, $\underline{X}^{(k)} = \{\mathbf{x}_s\}_{s \in \mathcal{S}_k^l} \in \mathbb{R}^{p \times S_k^l}$, where $S_k^l = |\mathcal{S}_k^l|$ is the cardinality of the set of predictor data points at the k^{th} node of the l^{th} tree. We sample a matrix $A \sim f_A(\mathcal{D}^l)$, where $A \in \mathbb{R}^{p \times d}$, possibly in a data dependent fashion, which we use to project the predictor matrix $\underline{X}^{(k)}$ onto a lower dimensional subspace, yielding $\widetilde{X}^{(k)} = A^\mathsf{T} \underline{X}^{(k)} \in \mathbb{R}^{d \times S_{ij}^l}$, where $d \leq p$ is the dimensionality of the subspace. Breiman's original RF algorithm can be characterized as a special case of random projection forests. In particular, in RF one constructs A such that for each of the d columns, we sample a coordinate (without replacement), and put a 1 in that coordinate, and zeros elsewhere. Breiman's F-RC constructs A by sampling a fixed number of coordinates (without replacement) for each of the d columns, and puts a value uniformly sampled from [-1,1] in each of those coordinates. Blaser's RR-RF defines A by first uniformly randomly rotating the feature space prior to fitting each tree, then subsampling variables at each node as RF does.

The goal of this work is to find a random projection forest that shares RF's scalability and ability to perform well when many irrelevant predictors are present, yet is able to find decision boundaries that are less biased by geometrical constraints (i.e. not constrained to be axis-aligned), by changing the distribution f_A. Furthermore, we desire the random projection forest to be robust to incommensurability of the predictor variables. The result we call randomer forests (RerFs).

Algorithm 1 Psuedocode for Random Projection Forests

Input: data: $\mathcal{D}^n = (\mathbf{x}_i, y_i) \in (\mathbb{R}^p \times \mathcal{Y})$ for $i \in [n]$, tree rules (stopping criteria, rules for sampling data points per tree, etc.), distributions on $d \times p$ matrices: $A \sim f_A(\mathcal{D}^l)$, preprocessing rules
Output: decision trees, predictions, out of bag errors, etc.

Preprocess according to rule
for each tree **do**
  Subsample data to obtain $\underline{X}$ and $\underline{y}$
  for each leaf node k in tree **do**
    Let $\widetilde{X}^{(k)} = A^{\mathsf{T}}\underline{X}^{(k)} \in \mathbb{R}^{d \times s}$, where $A \sim f_A(\mathcal{D}^l)$
    Find the "best" split coordinate $i^* \in [d]$ in $\widetilde{X}^{(k)}$ and "best" split value $t^*(i^*)$ for this coordinate
    Split $\widetilde{X}^{(k)}$ according to whether $\widetilde{X}^{(k)}(i*) > t^*(i^*)$
    Assign each child node as a leaf or terminal node according to stopping criteria
  end for
end for
Prune trees according to rule

3 Randomer Forests

Our choice in f_A is based on the following three ideas:

1. As mentioned previously it's frequently the case that axis-aligned splits are suboptimal, and oblique splits may be desired. Therefore, we desire the construction of A to be such that columns of A are allowed to have more than a single nonzero.

2. RF often does well when the signal is compressive, which may be attributed, in part, to the extreme sparsity constraint on the columns of A. Therefore, we desire A to be sparse.

3. Except for F-RC, existing oblique decision forest algorithms involve expensive computations to identify and select splits, rendering them less space and time efficient than RF or F-RC. An oblique decision forest having a space and time complexity comparable to RF or F-RC is desirable. This is further motivation to have A be sparse, since sparse matrix multiplication can be much faster and require less storage.

To this end, we employ very **sparse random projections** [8]. Rather than sampling d non-zero elements of A and enforcing that each column gets a single non-zero number (without replacement) as RF does, we relax these constraints and select d non-zero numbers from $\{-1, +1\}$ with equal probabilities and distribute uniformly at random in A.

We note that this construction bares a resemblance to Forest-RC, yet there are several key differences. In Forest-RC, the number of nonzeros in each column of A is fixed to be the same across A and for every split node, and its optimal value has to be found through parameter selection. Our construction circumvents selection of this parameter by randomizing the number of nonzeros in each column of A. Furthermore, our algorithm allows A to have columns of varying sparsity, which may promote more diversity in the ensemble. To our knowledge, this property does not exist in any of the proposed random projection forest algorithms. Additionally, rather than the nonzeros being uniform random numbers in the interval $[-1, 1]$, nonzeros in our proposed construction are either exactly -1 or 1, thereby improving computational efficiency. Lastly, we note that our construction of A preserves distances between points with high probability [8] and has ties to matrix sketching [9].

When combining predictor variables in this way, the decision forest is sensitive to the incommensurability of predictor variables. In fact, all previously proposed oblique methods that we are aware of also have this issue. We note that RFs, on the other hand, possess the property that they are invariant to monotonic transformations applied to each predictor variable. They are effectively operating on the order statistics, rather than actual magnitudes, of predictors.Therefore, we can adopt the same policy and **rank transform** prior to inducing the forest. We call RerFs that do this RerF(r).

4 Experimental Results

4.1 Simulations

Many classification problems arise in which the signal is compressive and the optimal split directions are not axis-aligned. We constructed two synthetic datasets with both of these properties to compare classification performance and training time of RF, RerF, RerF(r), RR-RF, and F-RC:

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 a mean whose elements are Bernoulli samples with probability 1/2, and then Gaussian noise is independently added to each dimension with the same variance. A sample's class label is equal to the parity of its mean. Sparse parity is an adaption of this problem in which the sample's class label is equal to the parity of only the first p^* elements of the mean, rendering the remaining $p - p^*$ dimensions as noise.

Trunk is a well-known binary classification in which each class is distributed as a p-dimensional multivariate Gaussian with identity covariance matrices [14]. The means of the two classes are $\mu_1 = (1, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}, ..., \frac{1}{\sqrt{p}})$ and $\mu_2 = -\mu_1$. It follows from this that the signal-to-noise ratio of the ith dimension asymptotically decreases to zero with increasing i.

RR-RF is an oblique decision forest method that has been recently proposed in [1]. Uniformly random rotations of the feature space imply that in general, splits will not be sparse. 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 RerF will be relatively more robust to the increasing presence of irrelevant features. Furthermore, we suspect that RR-RF will be especially sensitive to the relative scales of the predictor variables, since it is linearly combining a large number of features.

Figure 1 depicts both the true class posterior probabilities and estimates of the posteriors for RF, RerF, and RR-RF in two different representations of the sparse parity simulation. The left column is the native sparse parity simulation and the right column is the sparse parity simulation with dimensions randomly scaled by factors sampled uniformly from $[10^{-5}, 10^5]$. For this simulation, $p = 15$, $p^* = 3$, and $n = 1000$, where p is the total number of dimensions, p^* is the number of relevant dimensions, and n is the number of sampled data points. The number of trees used for all three algorithms was 500. Various values of d, the number of columns in the random projection matrix A, were tried and the best for each algorithm was selected.

Comparing panels C, E, G, and I with panel A shows that RerF gives estimates of the posteriors closest to the true posteriors, followed by RerF(r). RF produces poor estimates because the linearly combinations of predictors are more informative than single predictors. RR-RF produces poor estimates because the proportion of irrelevant predictor variables is large, so that rotating the data is likely to obscure the signal. Comparing panels C and D show that RF isn't affected by scale. Comparing panel E with F and I with J show that both RerF and RR-RF are sensitive to scale. Panels G and H demonstrate that RerF(r), on the other hand, maintains ability to produce reasonable estimates of the class posteriors.

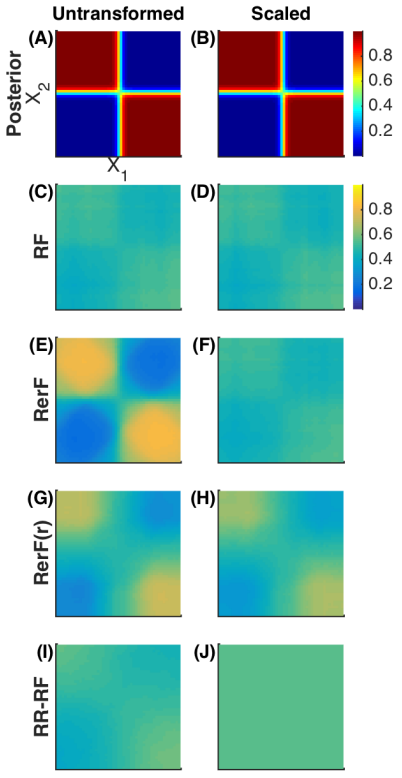

Figure 1: Posteriors and classifier estimates of posteriors for the sparse parity problem. Oblique forests are sensitive to relative scale of predictor variables, unlike RF. RerF(r), which rank transforms the data, is more robust to scale.

The left panels of Figure 2 show two-dimensional scatter plots from each of the two example simulations (using the first two coordinate dimensions). The middle panels show the error rate relative to RF against the number of observed dimensions p. Relative error was computed as the difference between the error of either RerF, RerF(r), F-RC, and RR-RF and that of RF. The error of RF relative to itself is shown for reference. The right 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. In all methods, trees were pruned and the minimum number of data points at a node in order to be considered for splitting was 10. The split criteria used was Gini impurity. The only hyperparameter tuned was d, the number of candidate split directions evaluated at each split node. When $p \leq 5$, each classifier was trained for all $d \in [p]$. When $p > 5$, each classifier was trained for all $d \in \{1, p^{1/4}, p^{1/2}, p^{3/4}, p\}$ Additionally, RerF, RerF(r), and F-RC were trained for $d \in p^{3/2}, p^2$. Note that for RF and RR-RF, d is restricted to be no greater than p by definition. For F-RC, the hyperparameter K, which denotes the number of predictor variables to linearly combine when forming candidate splits, was fixed to two. Errors for each classifier were selected as the lowest achieved from the different values of d tried. Training times for each classifier are the average given by all values of d tried. For sparse parity, n was fixed at 1000 and classifiers were evaluated for $p \in \{2, 5, 10, 25, 50, 100\}$. The relevant number of features $p*$ was fixed at a value of 3. For Trunk, n was fixed at 100 and RF and RerF were evaluated for $p \in \{2, 10, 50, 100, 500, 1000\}$. RR-RF was not evaluated for $p = 1000$ due to computational burden. Relative error and training times plotted are the average of 25 trials, and error bars represent the standard error of the mean.

In panel B, both RerF and F-RC perform as well as or better than both RF and RR-RF for all values of p. RerF(r) performs better than RF when $p = 10$ and performs about the same otherwise. RR-RF performs as well as or better than RF except for when $p = 25$. As conjectured, RR-RF performs better than RF when p is small because oblique splits provide an advantage over axis-aligned splits in the sparse parity problem. As p increases and the ratio p^*/p decreases, RerF begins to outperform RR-RF. Ultimately, when this ratio is small enough, RR-RF performs even worse than RF. RerF and F-RC's ability to perform relatively well can be attributed to the sparsity of oblique splits. In panel E, RerF, RerF(r), and F-RC outperform RF 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 RerF, RerF(r), and F-RC in these cases because a larger number of features can be linearly combined to yield an even higher signal-to-noise ratio. When $p = 500$, classification performance of RR-RF significantly degrades and exhibits the highest error rate. This can be explained by the fact that when p is large enough, RR-RF often samples linear combinations of many features each having a low signal-to-noise ratio. Such projections will yield a lower signal-to-noise ratio than any single feature. In panels C and F, training times are comparable when p is small. As panel F indicates, training time of RR-RF is significantly longer than the others when $p = 500$.

4.2 Theoretical Space and Time Complexity

For a RF, assume there are L trees. If there are m data points per tree, and each tree grows until terminal nodes have only $\mathcal{O}(1)$ data points with p coordinates in them, there are $\mathcal{O}(m)$ nodes. Then the complexity of constructing the random forest, disregarding cross-validation or other randomization techniques for preventing overfitting, is $\mathcal{O}(Lm^2 p \log m)$. In practice the trees are shallower and stop much earlier than when nodes have $\mathcal{O}(1)$ points, so "in practice" the complexity often appears to be $\mathcal{O}(Lmp \log m)$. Storing RF requires $\mathcal{O}(Lm \log m)$ because each node can be represented by the index of the nonzero coordinate. The only additional space constraint is storing which indices are positive, and which are negative, which is merely another constant.

RerF has a very similar time and space complexity, unlike many of the other oblique random forest variants. Specifically, assume that RerF also has L trees, and m data points per tree, and no pruning, so $\mathcal{O}(m)$ nodes. Let m_k be the number of data points at node k of the tree. Like RF, RerF requires $\mathcal{O}(p)$ time to sample p non-zero numbers, and $\mathcal{O}(pm_k)$ time to obtain the new matrix, $\widetilde{X}$, because it is a sparse matrix multiplication, in node k with $\mathcal{O}(m_k)$ points. RerF also takes another $\mathcal{O}(p/m \log(p/m)) = \mathcal{O}(1)$ time to find the best dimension. Thus, in total, in the case of well-balanced trees, RerF also requires only $\mathcal{O}(Lpm^2 \log m)$ time to train. To store the resulting RerF, like RF, requires $\mathcal{O}(Lm \log m)$, because each node can be represented by the indices of the coordinates that received a non-zero element, and the expected number of such indices is $\mathcal{O}(1)$. Note

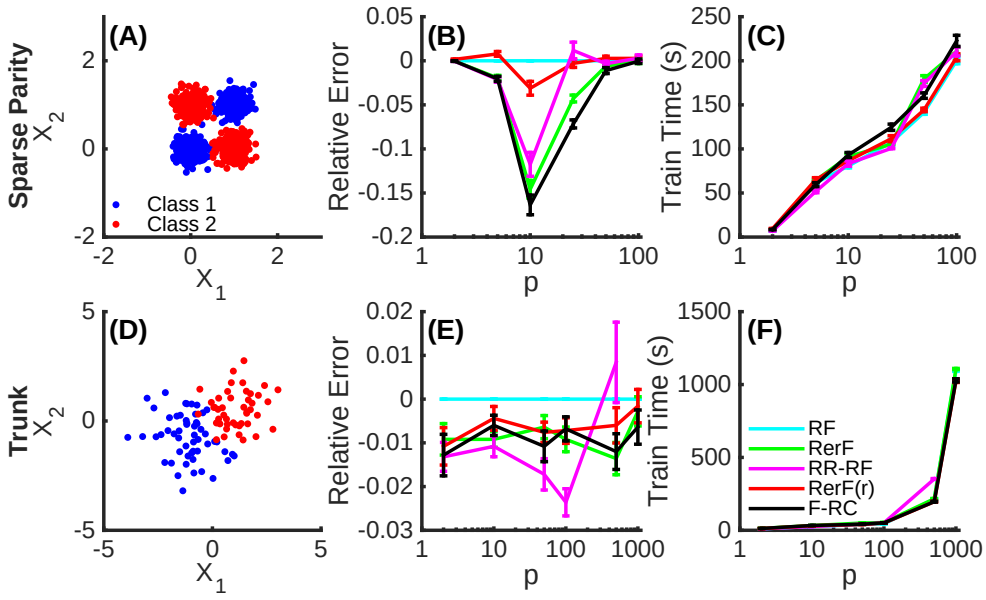

Figure 2: Sparse parity (A-C) and Trunk (D-F) simulations (see section 4.1 for details). (A) and (D): Scatterplots of sampled points in the first two dimensions. (B) and (E): Error rates of various random projection forest classifiers relative to RF across different values of p. (C) and (F): Same as (B) and (E) except absolute training time is plotted on the y-axis instead. All oblique methods do better than RF when the number of irrelevant features is sufficiently small, due to their ability to generate oblique parititions. However, when the number of irrelevant features becomes large enough, performance of RR-RF rapidly degrades. Training times show that RerF, RerF(r), and F-RC scales comparably with RF, while RR-RF scales poorly with large p (note that in panels E and F, RR-RF is only plotted up to $p = 500$ due to computational constraints. *Note*: the color-coding here is adopted in figures 3 and 4 that follow.

that these numbers 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 random rotation matrix for each tree. Rotating the data matrix prior to inducing each tree additionally requires $\mathcal{O}(mp^2)$. Therefore, RR-RF becomes very expensive to compute when p is large. This can explain the trend seen in Figure 2(F). In addition to storing all of the decision trees, RR-RF has to store a rotation matrix for each tree, which requires $\mathcal{O}(p^2)$.

4.3 Effects of Transformations and Outliers

We next want to compare the robustness classifier robustness to various data transformations across a variety of simulation settings. To do so, we consider several different modifications to the simulation settings described in the previous section: rotation, scale, affine, and outliers. To rotate the data, we simply generate rotation matrices uniformly 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 scaling as just described. Additionally, we examined the effects of introducing outliers. Outliers were introduced by sampling points from the distributions as previously described but instead using covariance matrices scaled up by a factor of four. Empirically, an addition of 20 points from these outlier models to the original 100 points was found to produce a noticeable but not overwhelming effect on classifier performance.

Figure 3 shows the effect of these transformations and outliers on the sparse parity (panels A-E) and Trunk (panels F-J) problems. Comparing panel B with A shows that all methods except for RR-RF and RF are affected by rotations, by noting that their error rates are noticeably shifted up for $p \geq 25$.

Comparing panel G with F shows that when $p = 500$ all classifiers except for RR-RF incur a loss in performance when the data is rotated. Comparing panel C with A and panel H with F show that F-RC and RR-RF suffer an increase in error rate when the data is scaled (error rate of RR-RF in panel H exceeds the y-axis limits), RerF is slightly hurt, and RerF(r) and RF are unaffected. In panels D and I, all classifiers are sensitive to affine transformations. However, RerF is more robust than F-RC and RR-RF, and RerF(r) is even more robust. Panels E and J show that all methods are slightly affected by the introduction of outliers.

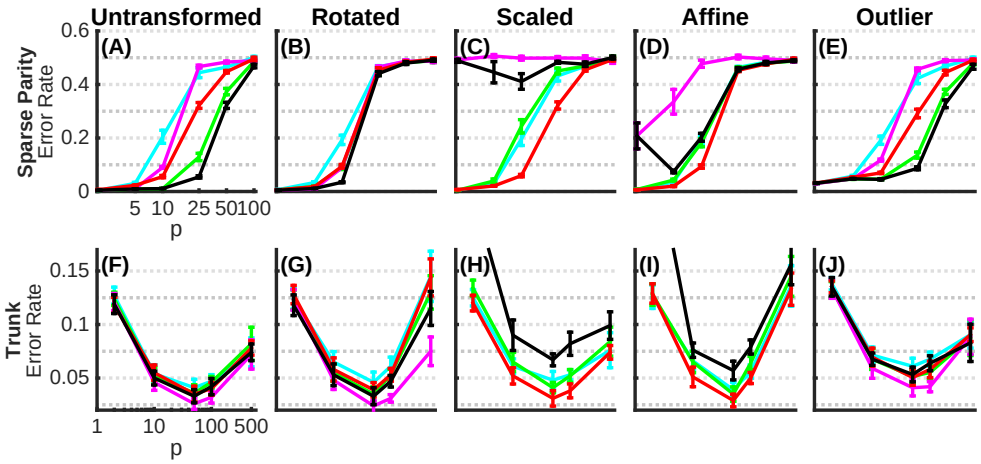

Figure 3: The effects of different transformations applied to the sparse parity (A-E) and Trunk (F-J) simulations on classification performance (see section 4.3 for details). Specifically, we consider rotations, scalings, affine transformations, as well as the addition of outliers. RR-RF and F-RC are severely affected by random scaling of the predictor variables, and therefore, also affected by affine transformations. RerF is slightly affected by scale, while RerF(r) is unaffected. When a general affine transformation is applied, RerF(r) is the most robust.

4.4 Benchmark Data

In addition to the simulations, RF, RerF, F-RC, and RR-RF were evaluated on 113 of the 121 datasets as described in [5]. The eight remaining datasets were not used because their high dimensionality and large number of data points rendered the classifiers both time and space costly, particularly for RR-RF. As in the previous section, transformations, with the exception of outliers, were applied to the datasets to observe their affects on performance of the three algorithms. Classifiers were trained on the entire training sets provided. For each data set, error rates were again estimated by out of bag error. 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. When $p \leq 5$, each algorithm was trained for all $d \in [p]$. When $p > 5$, each algorithm was trained for all $d \in \{1, p^{1/4}, p^{1/2}, p^{3/4}, p\}$. Error rates for each algorithm were selected as the minimum given by the five. For each dataset, relative performance ratios for each algorithm were computed by dividing the error rate of each algorithm by the minimum error rate of the three. The empirical cumulative distribution functions of relative performance ratios for each algorithm were computed and plotted in Figure 4. Such plots are called performance profiles [4]. Performance profiles are useful in visualizing how frequently a particular algorithm wins, and when it loses, how frequently it loses by a certain amount. For instance, a value of 0.9 on the y-axis and a value of 2.0 on the x-axis means that the error of that classifier was at most twice the error of the best performing classifier on a given dataset in 90% of all benchmark datasets.

In panel A of Figure 4, F-RC, RerF, and RerF(r) outperform RF, while RR-RF exhibits inferior performance. Panel B shows that when the benchmark datasets are rotated, RR-RF and F-RC perform the best, followed by RerF and RerF(r), and lastly RF. Panel C shows that RerF(r) performs the best when the predictors are randomly scaled, while RF, RerF, and F-RC are approximately the same.

Lastly, panel D shows that RerF(r) performs the best when affine transformations are applied, while F-RC and RR-RF perform very poorly.

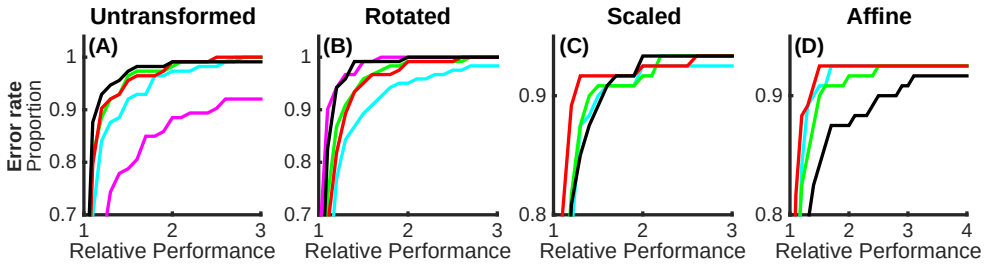

Figure 4: Classification performance profiles on benchmarks with various transformations applied (see section 4.4 for details). In terms of classification accuracy, RerF performs as well as or better than RF in all settings except for when affine transformations are applied to the data. RerF, RerF(r), and F-RC outperform RR-RF for all settings except for when the data is rotated. RR-RF performs terribly when scale or affine transformations are applied, and F-RC also performs poorly when affine transformations are applied. RerF(r) dominates all others in the face of affine transformations.

5 Conclusion and Future Work

We have proposed novel methods for constructing decision forests, which we call RerFs. We view these methods as special cases of a more general random projection forest, which include Breiman's original Forest-RI and Forest-RC, as well as previously proposed oblique decision forests. Our proposed method bares some similarity to Forest-RC, but is different in important ways. The choice of $f_A(\mathcal{D}_l)$ we propose produces a decision forest that is more computationally efficient. Additionally, it can be viewed as a generalization of both RF and F-RC by noting that we don't restrict the number of nonzeros per column to be fixed across A. Instead, we have columns of varying sparsity. This imparts more diversity into the decision forest and can enhance performance. We have demonstrated in simulations that RerFs are especially well-suited for classification problems in which axis-parallel splits are suboptimal, and at the same time, have a large number of irrelevant features relative to relevant ones. This could explain RerF's excellent empirical performance on a suite of 113 benchmark datasets, as real data often has the properties just described. Moreover, RerF preserves the time and space complexity of Forest-RI, and is more robust to affine transformations than are other oblique methods, especially the variant RerF(r).

The simplicity of RerFs and the nice properties of very sparse random projections [8] suggest that they will be amenable to theoretical investigation, extending the work of Scornet et al. (2015) to the RerF setting [12]. Moreover, we hope that theoretical investigations will yield more insight into which distributions $f_A(\mathcal{D}_l)$ will be optimal under different distributional settings, both asymptotically and under finite sample assumptions. Even under the currently proposed $f_A(\mathcal{D}_l)$, our implementation has room for improvement. Although it has the same space and time complexity as RF, we will determine explicit constants, and improve our implementation accordingly. Indeed, our current implementation is a proof-of-concept MATLAB implementation. We will utilize recent GPU and semi-external memory methods to significantly speed up RerF [15]. As with all decision forests, multiclass classification is but one exploitation task they can be used for; therefore, we will also extend this work to enable regression, density estimation, clustering, and hashing. We will provide open source code to enable more rigorous and extended analyses by the machine learning community.

6 References and Notes

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

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

[3] 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.

[4] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2008.

[5] 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.

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

[7] 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.

[8] P. Li, T. J. Hastie, and K. W. Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296. ACM, 2006.

[9] E. Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM, 2013.

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

[11] 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.

[12] E. Scornet, G. Biau, J.-P. Vert, et al. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015.

[13] 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.

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

[15] D. Zheng, D. Mhembere, R. Burns, J. T. Vogelstein, C. E. Priebe, and A. S. Szalay. Flashgraph: Processing billion-node graphs on an array of commodity ssds. In *13th USENIX Conference on File and Storage Technologies (FAST 15)*. USENIX Association, 2015.