Randomer Forests

Tyler M. Tomita[*]
Department of Biomedical Engineering
Johns Hopkins University
Baltimore, MD
ttomita@jhu.edu

Mauro Maggioni
Department of Mathematics
Duke University
Durham, NC
mauro@math.duke.edu

Joshua T. Vogelstein
Department of Biomedical Engineering
Johns Hopkins University
Baltimore, MD
jovo@jhu.edu

Abstract

jovo says: please make this submittable asap. this means include bib, put figures in the right place, anonymize, etc.

1 Introduction

Data science is becoming increasingly important as our ability of a society 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 that has been the subject of much study. A simple pubmed search for the term "classifier" reveals nearly 9,000 publications, and a similar arxiv search reports that the number of hits is >1000. Of all the classifiers, random forests (RFs) [**?**], is generally considered to be best, with good reason. Several recent benchmark papers assess the performance of many different classifiers on many different datasets [**?**, **?**], and both concluded the same thing: random forest are the best classifier.

The reasons for the popularity and utility of random forests are many. Below, we list some of the primary reasons, in order of approximate importance (as assessed subjectively and qualitatively by us):

- **empirical performance**: doing well in a wide variety of contexts
- **scale invariant**: this means that different predictor variables can have totally different units, millimeters and kilometers and milligrams and kilograms, for example, and RF does not care.
- **robust to outliers**: certain data points could be outliers, either because some or all of the values of their feature vector are far from the others
- **computational efficiency**: it is reasonably efficient, more so than an exhaustive search to find the globally optimal tree, which is NP-hard [**?**]
- **storage efficiency**: when training on lots of data, storage of the classifier can become problematic
- **interpretability**: it is reasonably interpretable, as each variable can be assigned an importance score (such as the "gini" score)

[*]Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

Although the benefits of RF are many, as with everything, there is room to improve. The main drawback of using RFs, in our opinion, is its sensitivity to rotations. Because when people refer to RFs they often mean Breiman's Forest-IC, which uses axis-parallel [**?**], or orthogonal trees [**?**], Breiman also characterized Forest-RC, which used linear combinations of coordinates rather than individual coordinates, to split along. Others have studied several different variants "oblique" random forests, including efforts to learn good projections [**?**, **?**], or using principal components analysis to find the directions of maximal variance [**?**, **?**], or directly learn good discriminant directions [**?**]. While each of these approaches deal with rotational invariance, they all also lose at least one of the above benefits of RF, namely scale invariance. The scale invariance property of RF is one of its most important and special properties, distinguishing it from most other classifiers, and making it appropriate in a wider variety of contexts. Moreover, all of the above approaches become outlier sensitive, and lose computational and storage efficiency. Therefore, in the literature, there remains a gap to build a classifier that has excellent empirical performance, while maintaining scale invariance, as well as robustness and computational and storage efficiency.

To bridge this gap, we first state a generalized forest building scheme, linear threshold trees, 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" (or RerFs for short), for reasons that will become clear in the sequel. We both theoretically and empirically demonstrate that our methods have the same time and space complexity as random forests, while matching or exceeding their performance even on axis aligned data. Moreover, we demonstrate that a variant of RerF is approximately is both affine invariant and robust to outliers, two properties that are relatively easy to obtain independently, though difficult to obtain jointly (though see recent work on robust PCA, for example [**?**]). Finally, we demonstrate on a suite of benchmark datasets, that RerF outperforms RF in terms of both accuracy and interpretability. We conclude that RerF should be the new reference classifier.

2 Linear Threshold Forests

A classification forest, $\bar{g}_n(X; \mathcal{D}_n)$ is an ensemble of decision trees, each of which is trained on a (sub)set of the data, $\mathcal{D}_n = (X_i, y_i)$ for all $i \in [n]$, and $X \in \mathbb{R}^{p \times n}$. Linear threshold 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, given $\bar{X} \in \mathbb{R}^{p \times s}$—the set of predictors in the node—we sample a matrix $A \sim f_A(\mathcal{D}_n)$, where $A \in \mathbb{R}^{p \times d}$ possibly in a data dependent fashion, which we use to project the predictor matrix X onto a lower dimensional subspace, $\widetilde{X} = A^{\mathsf{T}} X \in \mathbb{R}^{d \times s}$, where $d \leq p$ is the dimensionality of the subspace, and $s \leq n$ is the number of samples at the given node. To wit, Breiman's original Forest-IC algorithm can be characterized as a special case of linear threshold forests. In particular, in Forest-IC 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. Similarly, Ho's rotation forests construct A from the top d principal components of the data $\bar{X}$ at a given node. Thus, the key difference in all these approaches is the choice of f_A.

The goal of this work is to find a linear threshold forest that has all of the desirable properties of random forests, as well as being approximately affine invariant, in part by changing the distribution f_A. The result we call randomer forests (or RerFs for short).

3 Randomer Forests

We propose three "tricks", each one designed to address one of the benefits/drawbacks of random forest. First, we address the rotation sensitivity. It is well known that principal components are rotationally invariant, and that random projections can be used to approximate principal components [**?**]. We use this trick to generate matrices A that are rotation invariant, but maintain the space and time complexity of RFs. Specifically, we take a lesson from random projections, particularly very sparse random projections [**?**]. Thus, rather than sampling d non-zero elements of A, enforcing that each columns gets a single non-zero number (without replacement), which is always one, we simply relax the constraints. More specifically, we simply drop d non-zero numbers in A, distributed uniformly at random. The result is equally sparse as RFs, but nearly rotationally invariant. Note that this aspect of RerFs is quite similar to Breiman's Forest-RC, although he used the interval $[-1, 1]$,

Procedure 1 Psuedocode for Linear Threshold Forests, which generalizes a wide range of previously proposed decision forests.

```
Input:
 1: Data: $\mathcal{D}_n = (X_i, y_i) \in (\mathbb{R}^p \times \mathcal{Y})$ for $i \in [n]$
 2: Tree rules: $n_{tree}$, stopping criteria, pruning rules, rule for sampling data points per tree, etc.
 3: Distributions on $s \times d$ matrices: $A \sim f_A(\mathcal{D}_n)$, for all $s \in [n]$
 4: Preprocessing rules
Output: decision trees, predictions, out of bag errors, etc.
 5: Preprocess according to rule
 6: for each tree do
 7:     Subsample data to obtain $(\bar{X}, \bar{y})$, the set of data points to be used in this tree
 8:     for each leaf node in tree do
 9:         Let $\widetilde{X} = A^{\mathsf{T}}\bar{X} \in \mathbb{R}^{d \times s}$, where $A \sim f_A(\mathcal{D}_n)$
10:         Find the coordinate $k^*$ in $\widetilde{X}$ with the "best" split
11:         Split $X$ according to whether $X(k) > t^*(k^*)$
12:         Assign each child node as a leaf or terminal node according to convergence criteria
13:     end for
14: end for
15: Prune trees according to rule
```

amongst other minor differences. Denote linear threshold forests that use the above scheme for sampling A matrices RerFs. Figure 1(A) and (D) depicts the difficulty of RF performing in even a very simple scenario, versus RerF 1(B), which does exceptionally well.

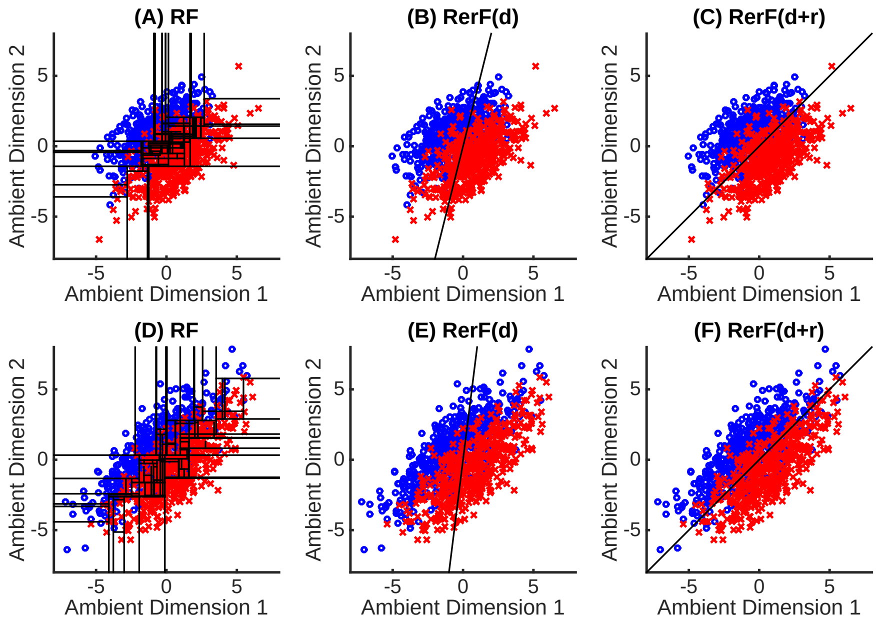

Figure 1: Scatter plot of parallel cigars in two dimensions. 1000 samples are obtained from multivariate Gaussians, where $\mu_j = [j, 0]$, for $j \in \{-1, 1\}$, and Σ is a diagonal matrix with diagonal elements $(1, k)$ for k is 4 (top row) or 8 (bottom row). The data are then rotated by 45 degrees. (A) Decision boundaries from a single tree of RF. (B) The decision boundary obtained by projecting the input data onto the difference in class-conditional means. (C) The same as (B), but first passing the input to ranks before computing the difference in means. (D), (E), and (F) are the same as (A), (B), and (C), respectively, for $k = 8$.

Unfortunately, by mixing dimensions of X, we have lost scale invariance. In fact, all previously proposed oblique random forests, that we are aware of, have lost scale invariance. We therefore note that random forests have a special property: they are invariant to monotonic transformations of the data applied to each coordinate in the ambient (observed) space. This is because they are effectively operating on the order statistics, rather than the actual magnitudes. In other words, if we convert, for each dimension, the values of the samples to their corresponding ranks, random forests yields the exact same result. Therefore, for our second trick, we adopt the same policy, and "pass to ranks" prior to doing anything else. Denote RerFs that pass to ranks RerF(r). Figure 1(E) demonstrates that RerF does not work well when the different dimensions are scaled quite differently from one another, but RerF(r) rectifies the situation, and does not harm the other case.

Finally, the above two tricks do not use the data to construct A at each node. Yet, there is evidence that doing so can significantly improve performance [**?**]. However, previously proposed approaches use relatively time and space intensive methods. We propose a quite simple strategy: compute the mean difference vector. In other words, given a two-class problem, let $\hat{d} = \hat{\mu}_0 - \hat{\mu}_1$, where $\hat{\mu}_c$ is the estimated class conditional mean. Under a spherically symmetric class conditional distribution assumption, $\hat{\delta}$ is the optimal projection vector. When there are $C > 2$ classes, the set of all pairwise distances between $\hat{\mu}_c$ and $\hat{\mu}_{c'}$ is of rank $C - 1$. Because RerFs are approximately rotationally invariant, we can simply compute all the class conditional means, and subtract each from one of them (we chose the $\hat{\mu}_c$ for which n_c, the number of samples in that class, is the largest). Computing this matrix is extremely fast, because it does not rely on costly singular value decompositions, matrix inversions, or iterative programming. Thus, it nicely balances using the data to find good vectors, but not using much computational space or time. Denote RerFs that include this matrix RerF(d), and if they pass to ranks first, RerF(r+d).

4 Experimental Evaluation

4.1 Axis Aligned Simulations

We constructed three synthetic datasets to compare classification performance (Figure 2) and training time (Figure 3) of RerF, RerF(d), and RerF(r+d) with that of random forest:

Trunk is a well-known binary classification (cite Trunk) in which each class is distributed as a p-dimensional multivariate Gaussian with identity covariance matrices. The means of the two classes are $\mu_1 = (1, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}, ..., \frac{1}{\sqrt{p}})$ and $\mu_2 = -\mu_1$. This is perhaps the simplest axis aligned problem, for which random forests should perform exceedingly well.

Parity is one of the hardest two-class classification problems, it is a multivariate generalization of the XOR 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. This is perhaps the hardest axis aligned problem, for which random forests should do well, whereas linear classifiers will perform at chance.

Multimodal is a four-class classification problem. Each of the four classes is distributed as an equal mixture of two multivariate Gaussians. The means are randomly sampled from a p-dimensional multivariate standard Gaussian and the covariance matrices are sampled from an inverse Wishart distribution with $10p$ degrees of freedom and a mean covariance matrix equal to the identity matrix.

For all comparisons, we used the same number of trees for each method, and the same number of non-zeros when generating A, to enable a fair comparison. The top panels of Figure 2 shows two-dimensional scatter plots from each of the three example simulations (using the first two dimensions). The bottom panels show the misclassification rate of RF vs RerF and its variants, as well as the Bayes optimal error, against the number of observed dimensions p. In all three cases, RerF(r+d) performs as well or better than RF, even though for the first two, the discriminant boundary is axis aligned. The reason that RerF outperforms RF even in the axis aligned discriminant boundary cases is that linear combinations of the ambient coordinates can yield new dimensions that have a higher signal to noise ratio than any of the original dimensions. Therefore, RerF can capitalize on that, and use those dimensions to obtain cleaner splits, and therefore better trees.

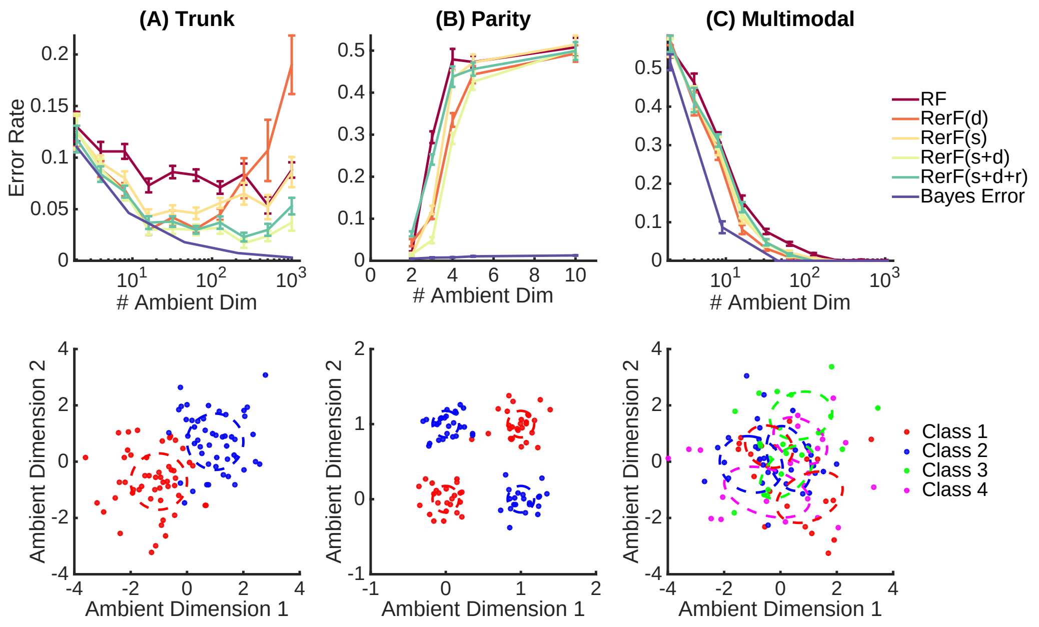

Figure 2: Classification performance comparing Random Forest (RF) to several variants of Randomer Forest (RerF), and Bayes optimal performance, on three distinct simulation settings: (A) Trunk, (B) Parity, and (C) Multimodal (see Methods for details). For all settings, the top panel shows a 2D scatter plot of the first 2 coordinates (dashed circles denote the standard deviation level set), and the bottom panel depicts misclassification rate vs. the number of ambient (coordinate) dimensions,. Note that in all settings, for all number of dimensions, RerF outperforms RF, even for Trunk and Parity, which were designed specifically for RF because the discriminant boundary naturally lies along the coordinate basis. In each case, $n = 100$ and the errorbars denote standard error over ten trials.

Figure 3 shows the corresponding running time for RF and the different RerF variants. Note that the RerF variants do not take significantly longer to train than RF.

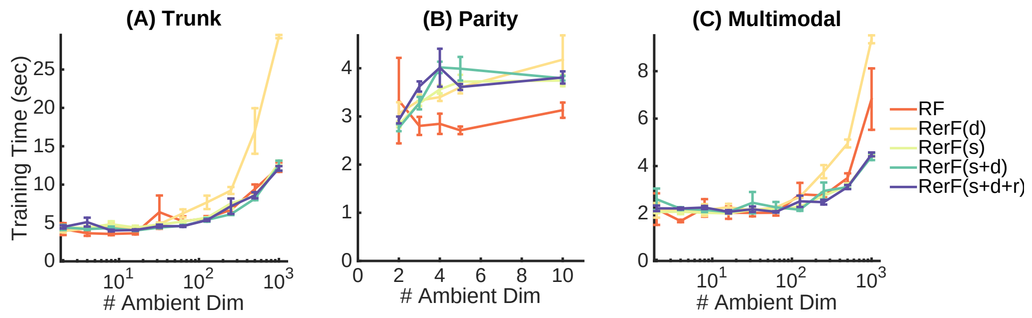

Figure 3: Classifier training time comparing RF to several variants of RerF, same setting as the top row of Figure 1. The only difference is that the y-axis here labels training time (in seconds). Although RerF requires slightly more time than RF (largely due to random sampling of projection matrices), they scale similarly.

4.2 Theoretical Space and Time Complexity

For a random forest, assume there are L trees. If there are s data points per tree, and each tree grows until terminal nodes have only a single data point in them, there are $\log(s)$ nodes. For each node, it takes $\mathcal{O}(d)$ time to sample d non-zero numbers per node, and $\mathcal{O}(ds)$ time to subsample the matrix to include only the d features of interest for each of the s samples. Although each node has a different number of samples, the root node dominates, so we can ignore all the others. In then takes $\mathcal{O}(d \log d)$ time to sort to find the best dimension to split on per node. Thus, in total, RF requires $\mathcal{O}(Ld(1 + sd \log d))$ time to train. To store the resulting RF, we simply require $\mathcal{O}(L \log(s))$ space because each node can be represented merely by the index of the coordinate that is used for splitting, plus its threshold.

Randomer forest 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 s data points per tree, and no pruning, so $\log s$ nodes. Like RF, it requires $\mathcal{O}(d)$ time to sample d non-zero numbers, and $\mathcal{O}(ds)$ time to obtain the new matrix, $\widetilde{X}$, because it is a sparse matrix multiplication. RerF also takes another $\mathcal{O}(\log d)$ time to find the best dimension. Thus, in total, RerF also requires only $\mathcal{O}(Ld(1+sd \log d))$ time to train. To store the resulting RerF also requires $\mathcal{O}(L \log(s+\epsilon))$, 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 slightly larger than one. The only additional space constraint is storing which indices are positive, and which are negative, which is merely another constant. RerF(r) requires an additional $\mathcal{O}(np \log p)$ time, to first sort the data for each sample. And RerF(d) requires an additional $\mathcal{O}(Cpn_c)$ time, assuming each class has n_c samples to compute the mean. Note that these numbers are in stark contrast to other oblique random forests. For example, rotation forests require running a singular value decomposition at each node.

4.3 Effects of Transformations and Outliers on Classifier Performance

Given that RerF do as well in misclassification rate as RF in multiple disparate examples, and have time and space complexity comparable, we next want to assess whether we have obtained the desired rotational invariance, while preserving these other properties. To do so, we consider several different modifications to the above described simulation settings: 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 [0,10]. Affine transformations were performed by applying a combination of rotations and scalings 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 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. For a baseline, we compare RF with RerF variants and Fisherfaces [**?**]. The misclassification rate for Fisherfaces was estimated using leave-one-out cross validation.

Figure 4 shows the effect of these transformations and outliers on the Trunk (top) and Parity (bottom) rows. Figure 4(A) shows that RF is sensitive to rotations, and therefore affine transformations, but robust to outliers. Figure 4(B) shows that RerF(d) is sensitive to scale, and therefore affine transformations, but not outliers. Figure 4(C) shows that RerF(d+r) is robust to affine transformations and outliers, as desired. Fisherfaces, which is essentially designed for this example, suffers from scale and therefore affine transformations, as well as outliers. The Parity example yields a similar result, except that rotating the data helps all the RF and RerF variants. This is because even $p - 1$ dimensional marginal distributions of the data in the ambient coordinates have no class conditional signal. Thus, rototating the data creates dimensions that have non-zero class conditional signal, which all of the RF and RerF variants can utilize. On the other hand, Fisherfaces, which is a fundamentally linear classifier, achieves chance levels no matter what.

4.4 Benchmark Data

In addition to the simulations, RF and RerF variants were evaluated on all 121 datasets as described in [1]. Classifiers were trained on the entire training sets provided. For each data set, misclassification rates were again estimated by out of bag error, and training time was measured as wall clock time. Figure 5 depicts misclassification rates and training time for the different algorithms. `something here.`

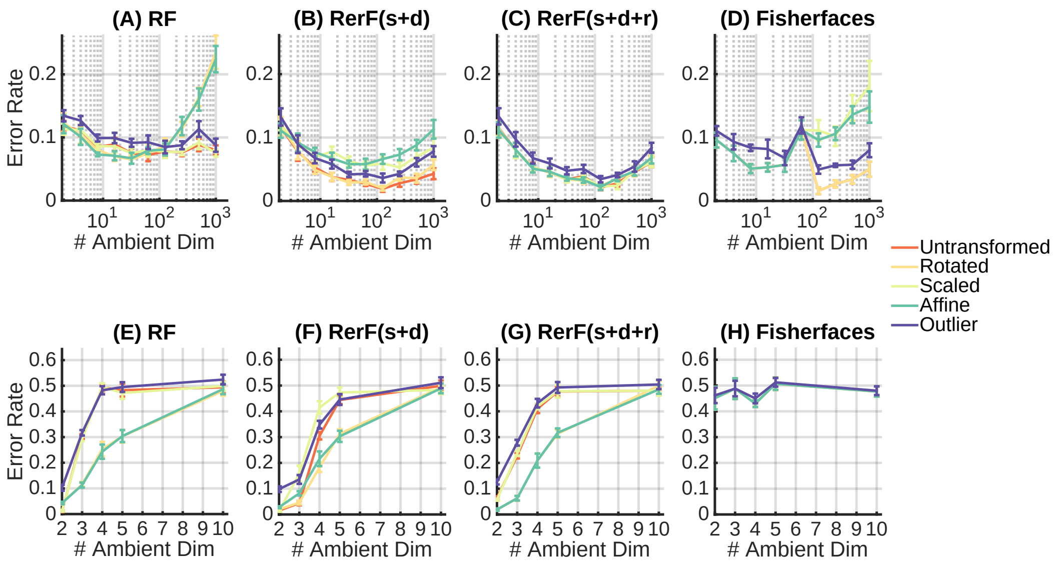

Figure 4: The effect of various transformations applied to the Trunk (panels A-D) and Parity (panels E-H) simulations (see Methods for details) on classification performance of (A,E) RF, (B,F) RerF (s+d), (C,G) RerF (s+d+r), and (D,H) Fisherfaces. Specifically, we consider, rotations, scalings, and affine transformations, as well as introducing outliers. Classification performance of RF is compromised by rotations and therefore affine transformations as well. RerF(s+d) is invariant to rotation, but not scaling and therefore not affine transformation. RerF(s+d+r) is invariant to to affine transformations. Like RerF (s+d), Fisherfaces is invariant to rotation but not scaling. Note that all variants are reasonably robust to outliers.

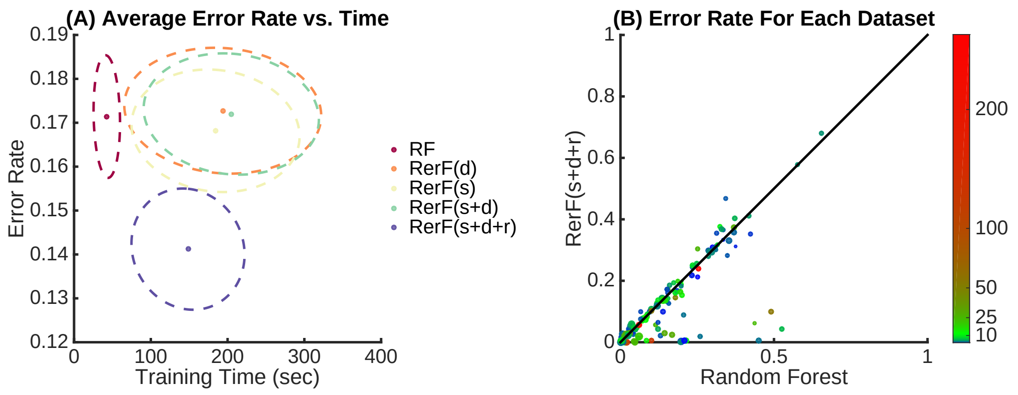

Figure 5: (A) Classification error and training time of Random Forest and Randomer Forests variants for all 121 datasets from the XXX benchmark comparison paper [**?**]. (A) Average (dots) and 0.1 standard deviation level sets (dashed lines) for each method. (B) Classification error of Randomer Forest (sparse+delta+robust) vs. that of Random Forest for each of the 121 datasets. The black line indicates equal classification error of the two algorithms. Color indicates dimensionality of the datasets and size of points indicates number of samples. Note that RerF almost always does as well, and often significantly better.

5 Conclusion

We have proposed a novel method for constructing ensemble classifiers. Like random forests, our method constructs an ensemble of randomized decision trees. However, by randomly projecting the data at each split node, partitions are not restricted to alignment with the coordinate axes. We have constructed datasets demonstrating settings in which RerF outperforms RF. Furthermore, one of the variants, RerF(d), exhibits robustness to affine transformations, a property lacking in RF.

Much work is still to be done with our proposed method. In this work, we only examined one method of constructing sparse random projection matrices. It is possible that other choices of construction will lead to improved performance in certain settings. Additionally, it will be useful to evaluate the sensitivity of our method to the use of different split criteria, pruning, and other parameters of the decision tree. It will also be of interest to establish consistency theorems for our method. While we only restricted our attention to classification thus far, our method can be generalized to other types of learning problems, such as regression, density estimation, etc.

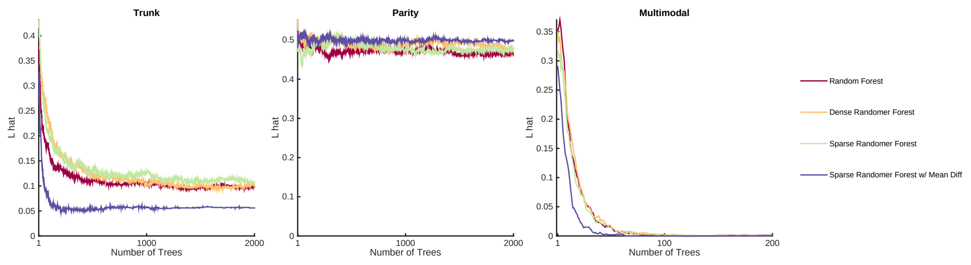

Figure A1: The number of trees until a stable misclassification rate is achieved for RF and the RerF variants in the Trunk, parity, and multimodal simulations. Simulation settings are the same as in Fig1. The number of ambient dimensions for panels A, B, and C were chosen to be 1000, 10, and 1000 respectively.

References

[1] M. Fern?andez-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*, vol. 15, no. 1, pp. 3133--3181, October 2014.

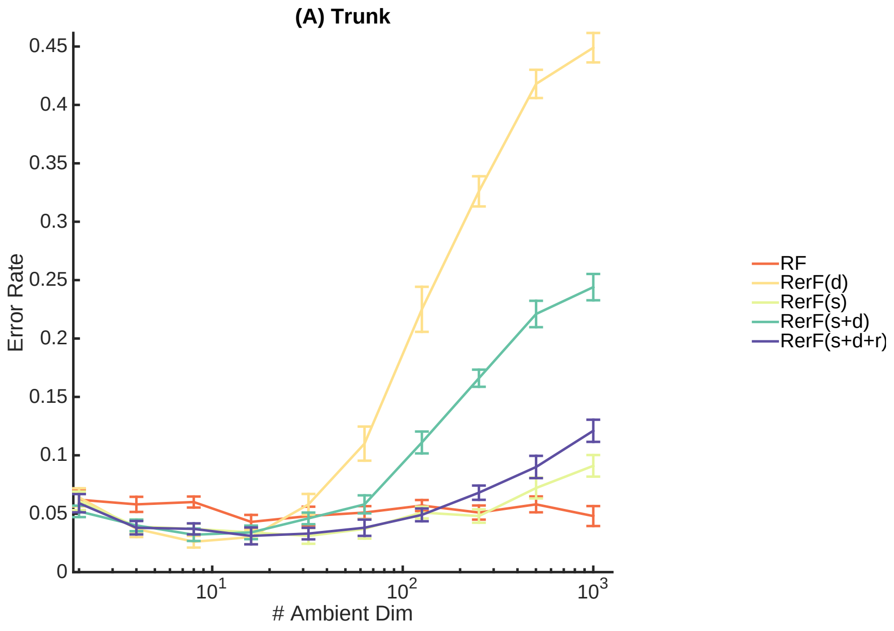

Figure A2: Classification performance comparing Random Forest (RF) to several variants of Randomer Forest (RerF) on the modified Trunk simulation setting. Here, the ith diagonal of the covariance matrices of both classes is equal to the inverse of the difference in means of the ith dimension between the two classes. For small values of p, all variants of RerF perform marginally better than RF. For large values of p, all variants of RerF perform worse than RF.