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, moreso 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 othogonal trees [**?**], Breiman also characterized Forest-RC, which used linear combinations of coordinates rather than indivudal 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:
  Data: $\mathcal{D}_n = (X_i, y_i) \in (\mathbb{R}^p \times \mathcal{Y})$ for $i \in [n]$
  Tree rules: $n_{tree}$, stopping criteria, pruning rules, rule for sampling data points per tree, etc.
  Distributions on $s \times d$ matrices: $A \sim f_A(\mathcal{D}_n)$, for all $s \in [n]$
  Preprocessing rules
Output: decision trees, predictions, out of bag errors, etc.
  Preprocess according to rule
  for each tree do
      Subsample data to obtain $(\bar{X}, \bar{y})$, the set of data points to be used in this tree
      for each leaf node in tree do
          Let $\widetilde{X} = A^{\mathsf{T}}\bar{X} \in \mathbb{R}^{d \times s}$, where $A \sim f_A(\mathcal{D}_n)$
          Find the coordinate $k^*$ in $\widetilde{X}$ with the “best” split
          Split $X$ according to whether $X(k) > t^*(k^*)$
          Assign each child node as a leaf or terminal node according to convergence criteria
      end for
  end for
  Prune trees according to rule
```

amongst other minor differences. Denote linear threshold forests that use the above scheme for sampling A matrices RerFs.

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).

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 Simulated Data

We constructed three synthetic datasets (Trunk, parity, and multimodal) to compare classification performance (Fig 1*jovo says: use fig labels to refer to figs*) and training time (Fig 2) of RerF, RerF(δ), and RerF(δ+r) 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 $\mu_2 = (-1, -\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{3}}, ..., -\frac{1}{\sqrt{p}})$. Parity is a binary classification problem in which each of the two classes is distributed as a mixture of 2^{p-1} multivariate gaussians with covariance matrices $\boldsymbol{\Sigma} = \frac{1}{32}\boldsymbol{I}$. The means of class one are the subset of $[0, 1]^p$ for which the number of zeros in a mean is even. The means of class two are the subset of $[0, 1]^p$ for which the number of zeros in a mean is odd. 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. The bottom panels of Figure 1 depict scatter plots of the data for $p = 2$. The top panels depict average misclassification rate over ten trials estimated using the out of bag error. For the trunk and multimodal simulations, 100 points were sampled and p varied from two to 1000 dimensions. For the parity simulation, 100 points were sampled and p varied from two to ten. The appropriate number of trees trained for RF and all variants of RerF for trunk, parity, and multimodal were empirically determined to be 1500, 1000, and 500 respectively (Fig A1). Bayes error is also included for reference. In binary classification problems in which each class consists of a single multivariate gaussian with equal covariance matrices, such as the Trunk simulation, Bayes error can be computed analytically (Bickel and Levina 2004). In the Parity and Multimodal simulations, Bayes error was estimated by averaging the misclassification rate of the Bayes optimal classifier on 1000 points over ten trials.

Panel A of Figure 1 shows that RerF(δ) and RerF(δ+r) outperform RF across all numbers of dimensions. This can be attributed to projection onto the difference in means. All variants outperform RF up to approximately 250 dimensions. Above this, misclassification rate of RerF is comparable to RF. In Panel B, we observe that all variants of RerF outperform RF for all numbers of dimensions. This can be understood by observing that for p dimensions, all splits in RF up to a tree depth of $p - 1$ will result in daughter nodes having chance posterior probabilities of being in either class. Therefore, any splits up to this depth that are not aligned with the coordinate axes can only be better. Multimodal???

In Figure 2, we observe a slight increase training time of all RerF variants compared to RF. Despite this increase, training time for all classifiers scales similarly with the number of dimensions. The apparent increase in training time for the RerF variants is largely due to sampling of the random projection matrices.

4.2 Effects of Transformations and Outliers on Classifier Performance

RF does especially well in classification problems in which the optimal decision boundaries are aligned or nearly aligned with the coordinate axes. Rotations in such situations can lead to a decrease in classification performance. Naturally, we wanted to examine the effects of various transformations on classification performance of RerF. These effects were examined using the Trunk and parity simulations as previously described. The transformations we applied were random rotations, scaling, and general affine transformations. Uniformly random rotation matrices were generated by first performing SVD on a p-dimensional matrix in which each element is sampled from a multivariate standard normal distribution. The rotation matrix was taken as the right singular vectors of this SVD. If the determinant of this matrix was equal to -1, the first two columns were permuted to render the determinant equal to $+1$. Random scaling was performed by applying to each dimension 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 to Trunk and parity simulations 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. The classifiers evaluated were RF, RF(s), RF(s+δ). Additionally, Fisherfaces was evaluated as a reference. The misclassification rate for Fisherfaces was estimated using leave-one-out cross validation.

The top panels of Figure 4 illustrate the effects of the transformations and outliers on the Trunk simulation. The bottom panels show these effects on the parity simulation. In the Trunk simulation, rotation results in noticeable degradation in classification performance of RF when the number of dimensions is greater than approximately 100. On the other hand, both RerF(δ) and RerF(δ+r) are unaffected by rotations. RerF(δ) exhibits an increase in misclassification rate when scaling is applied. This is expected because ambient dimensions with a relatively large scale will dominate the variance of new dimensions constructed from a linear combination of the ambient ones. For this same reason, Fisherfaces is also affected by scaling. Since RerF(δ+r) maps all dimensions to the same scale, it is invariant to scaling and is also the only classifier to exhibit invariance to affine transformations. We also observe that all classifiers are only slightly affected by the addition of outliers to the Trunk simulation. As shown in panel E, rotations in the Parity simulation actually improve classification performance of RF. As mentioned previously, all splits in RF up to a tree depth of $p-1$ in the untransformed parity simulation result in chance probabilities of being in either class in the daughter nodes. Therefore, rotating can only improve performance. As in the Trunk simulation, performance of RerF is hurt when scaling is applied in the parity simulation.

4.3 Real Data

In addition to the simulations, RF, RerF, RerF(δ), and RerF(δ+r) were evaluated on 121 datasets as described in []. 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. Misclassification rates and training times were averaged over all 121 datasets (Fig 4).

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(δ), 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.

References

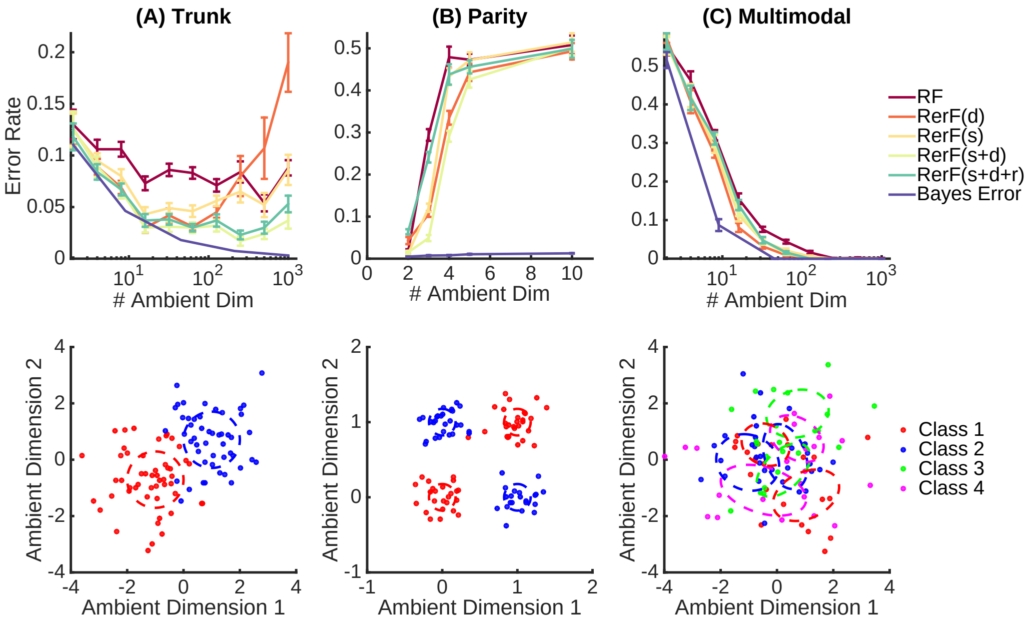

Figure 1: Classification performance comparing Random Forest (RF) to several variants of Randomer Forest (RerF), and Bayes optimal performance, on three distinct simulation settings: (A) Trunk, (B) Parity, and (C) Multimodal (see Methods for details). For all settings, the top panel depicts misclassification rate vs. the number of ambient (coordinate) dimensions, and the bottom panel shows a 2D scatter plot of the first 2 coordinates (dashed circles denote the standard deviation level set). Note that in all settings, for all number of dimensions, RerF outperforms RF, even Trunk and Parity, which were designed specifically for RF because the discriminant boundary naturally lies along the coordinate basis.

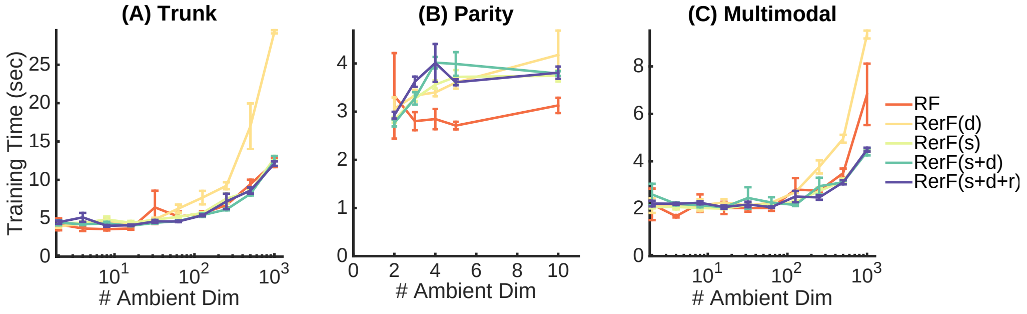

Figure 2: Classifier training time comparing RF to several variants of RerF, same setting as the top row of Figure 1. The only difference is that the y-axis here labels training time (in seconds). Although RerF requires slightly more time than RF (largely due to random sampling of projection matrices), they scale similarly.

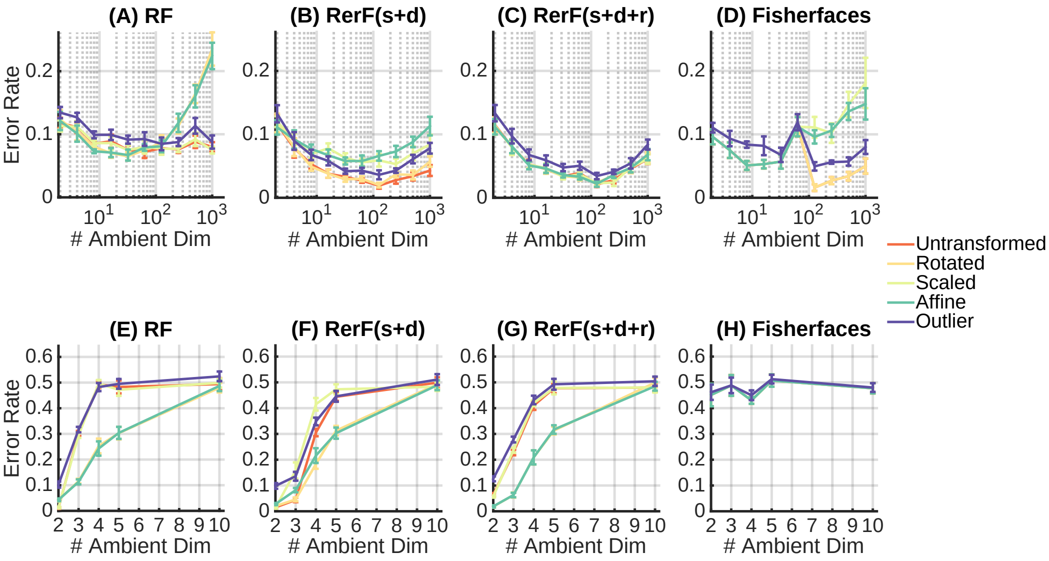

Figure 3: The effect of various transformations applied to the Trunk (panels A-D) and Parity (panels E-H) simulations (see Methods for details) on classification performance of (A,E) RF, (B,F) RerF (s+d), (C,G) RerF (s+d+r), and (D,H) Fisherfaces. Specifically, we consider, rotations, scalings, and affine 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.

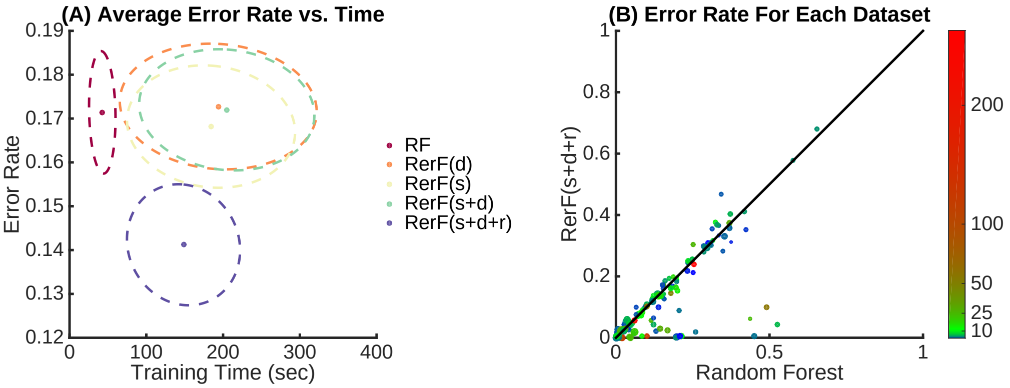

Figure 4: (A) Classification error and training time of Random Forest and Randomer Forests variants for all 121 datasets from the XXX benchmark comparison paper [**?**]. (A) Average (dots) and 0.1 standard deviation level sets (dashed lines) for each method. (B) Classification error of Randomer Forest (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.

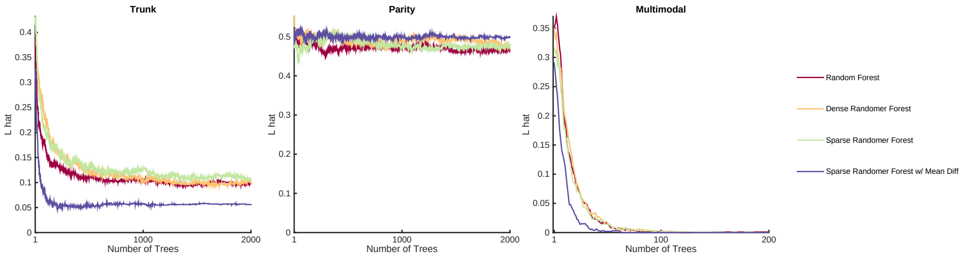

Figure A1: The number of trees until a stable misclassification rate is achieved for RF and the RerF variants in the Trunk, parity, and multimodal simulations. Simulation settings are the same as in Fig1. The number of ambient dimensions for panels A, B, and C were chosen to be 1000, 10, and 1000 respectively.

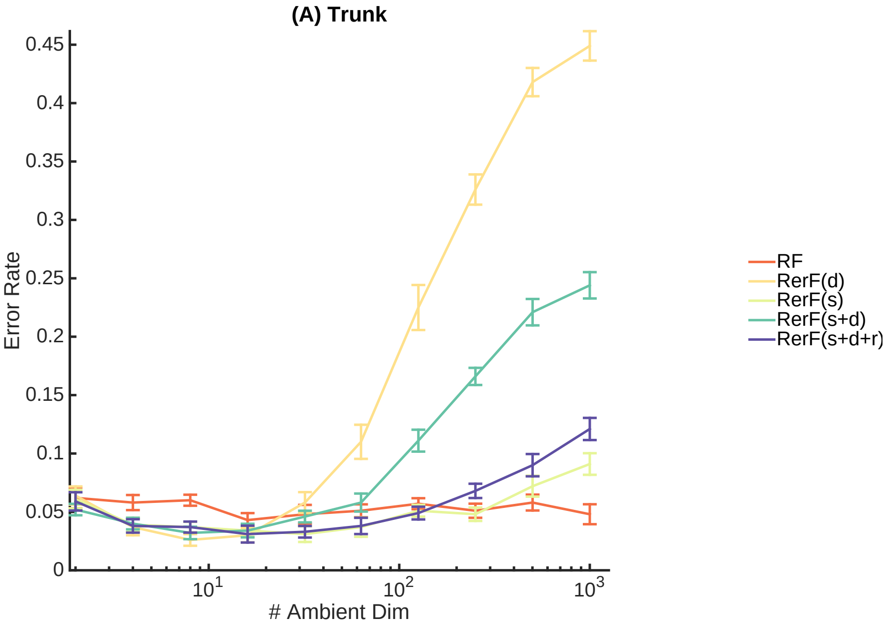

Figure A2: Classification performance comparing Random Forest (RF) to several variants of Randomer Forest (RerF) on the modified Trunk simulation setting. Here, the ith diagonal of the covariance matrices of both classes is equal to the inverse of the difference in means of the ith dimension between 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.

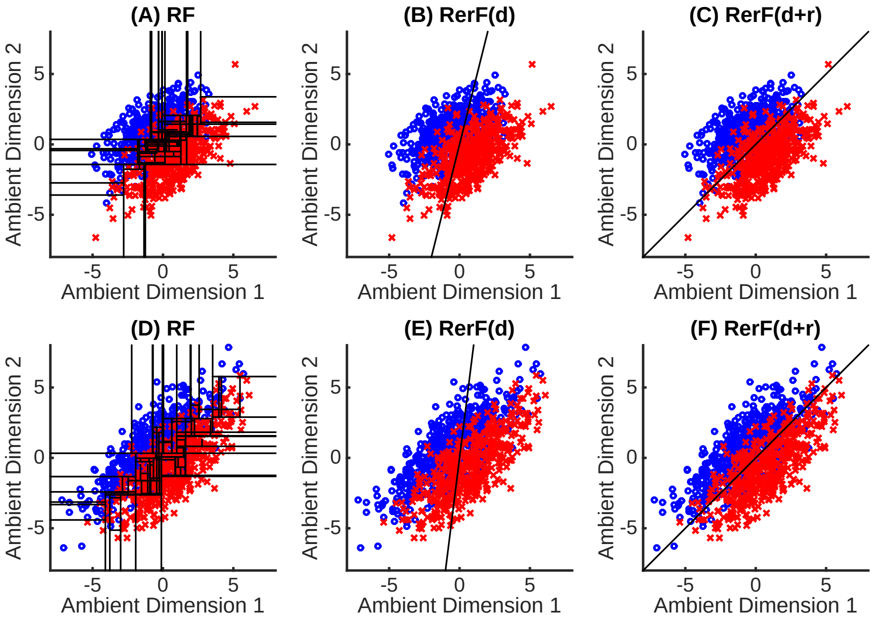

Figure A3: Scatter plot of parallel cigars in two dimensions. (A) Decision boundaries from a single tree of RF. (B) The decision boundary obtained by projecting the input data onto the difference in class-conditional means. (C) The same as (B), but first passing the input to ranks before computing the difference in means.