Randomer Forests

Anonymous Author(s)
Affiliation
Address
email

Abstract

Random forests (RF) is a popular general purpose classifier that has been shown to outperform many other classifiers on a variety of datasets. The widespread use of random forests can be attributed to several factors, some of which include its excellent empirical performance, scale and unit invariance, robustness to outliers, time and space complexity, and interpretability. While RF has many desirable qualities, one drawback is its sensitivity to rotations and other operations that "mix" variables. In this work, we establish a generalized forest building scheme, linear threshold 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 a few special cases which we call randomer forests (RerFs). RerFs are linear threshold forest that exhibit all of the nice properties of RF, in addition to approximate affine invariance. In simulated datasets designed for RF to do well, we demonstrate that RerF outperforms RF. We also demonstrate that one particular variant of RerF is approximately affine invariant. Lastly, in an evaluation on 121 benchmark datasets, we observe that RerF outperforms RF. We therefore putatively propose that RerF be considered a replacement for RF as the general purpose classifier of choice.

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, and 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. One of the most popular and best performing classifiers is random forests (RFs) [1]. Several recent benchmark papers assess the performance of many different classifiers on many different datasets [2, 3], and both concluded the same thing: random forests are the best classifier.

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

- **empirical performance**: excellent prediction performance in a wide variety of contexts;
- **scale and unit invariance**: different predictor variables can have totally different units, e.g. millimeters and kilograms, for example, without affecting RF's;
- **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;
- **time complexity**: it is reasonably efficient, more so than an exhaustive search to find the globally optimal tree, which is NP-hard [4];

- **space complexity 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), albeit for large forests it can become hard to sort the variables in order of importance.

Although the benefits of RF are many, there is room for improvement. The main drawback of using RFs, in our opinion, is its sensitivity to rotations and other operations that "mix" variables. While RFs usually refers to Breiman's Forest-IC, which uses axis-parallel [4], or orthogonal trees [5], 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 [4, 6], or using principal components analysis to find the directions of maximal variance [7, 8], or directly learn good discriminant directions [5]. While these approaches deal with rotational invariance, they all also lose at least one of the above benefits of RF, namely scale and unit invariance. The scale and unit 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 oblique random forest approaches become outlier sensitive, and lose computational and storage efficiency. There is therefore a gap that calls for the construction of 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 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*" (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 classification problems that are inherently axis aligned. 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 and inexpensively (see recent work on robust PCA following Candes et al., 2009 [9]). 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

Let $\mathcal{D}^n = \{(X_i, y_i) : i \in [n]\}$ be a given dataset, where $X_i \in \mathbb{R}^p$ and $y_i \in \mathcal{Y} = \{\widetilde{y}_1, \ldots, \widetilde{y}_C\}$. A classification forest, $\bar{g}(X; \mathcal{D}^n)$ is an ensemble of L decision trees, each tree $g^l(X; \mathcal{D}^l)$ is trained on a (sub)set of the data, $\mathcal{D}^l \subset \mathcal{D}^n$). 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 of the tree, we have a set of predictor data points, $\bar{X} = \{X_s\}_{s \in \mathcal{S}^l_{ij}} \in \mathbb{R}^{p \times S^l_{ij}}$, where $S^l_{ij} = |\mathcal{S}^l_{ij}|$ is the cardinality of the set of predictor data points at the $(ij)^{th}$ node of the l^{th} tree. 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 $\bar{X}$ onto a lower dimensional subspace, yielding $\widetilde{X} = A^\mathsf{T} X \in \mathbb{R}^{d \times S^l_{ij}}$, where $d \leq p$ is the dimensionality of the subspace. To wit, Breiman's original Forest-IC algorithm can be characterized as a special case of linear threshold forests. In particular, in Forest-IC 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. 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).

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

3 Randomer Forests

Our construction has three key ingredients, 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, but can be expensive to compute and are biased towards directions of large variance rather than directions that may be useful for classification. Instead we use random projections: we generate matrices A that are distributed in a rotation invariant fashion, but maintain the space and time complexity of RFs, by employing very **sparse random projections** [10]. Rather than sampling d non-zero elements of A, enforcing that each column gets a single non-zero number (without replacement), which is always 1, we relax these constraints and select d non-zero numbers from $\{-1, +1\}$ with equal probabilities and distribute uniformly at random in A. 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 entries in the interval $[-1, 1]$, amongst other minor differences. We call linear threshold forests that use the above scheme for sampling A matrices "*RerFs*".

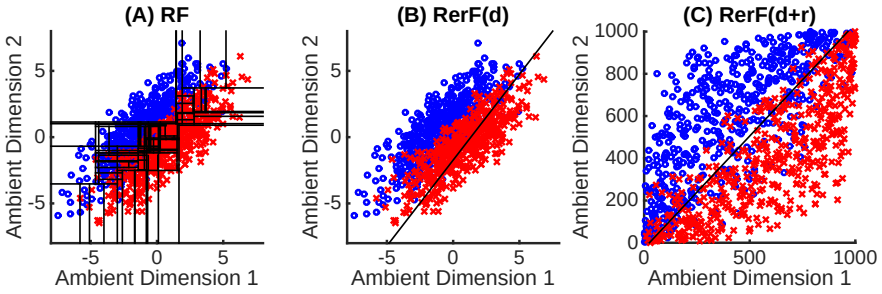

Figure 1: Scatter plot of parallel cigars in two dimensions. 1000 samples are obtained from multivariate Gaussians, where $\mu_j = [j, 0]$, for $j \in \{-1, 1\}$, and Σ is a diagonal matrix with diagonal elements $(1, 8)$. The data are then rotated by 45 degrees. (A) Decision boundaries from a single 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.

Unfortunately, by mixing dimensions of X, we lost *scale and unit invariance*. In fact, all previously proposed oblique random forests, that we are aware of, have lost scale and unit invariance. We therefore note that random forests have a special property: they are invariant to monotonic transfor-

mations of the data applied to each coordinate in the ambient (observed) space. They are effectively operating on the order statistics, rather than actual magnitudes of coordinates. In other words, if we convert for each dimension the values of the samples to their corresponding ranks, random forests yield the exact same result. Therefore, for our second trick, we adopt the same policy, and "**pass to ranks**" prior to doing anything else. We call RerFs that pass to ranks RerF(r).

Figure 1 shows the effect of these two tricks on an exceedingly simple problem: a two-class classification problem whose optimal discriminant boundary is linear. RF requires lots of nodes in each tree, even in this very simple scenario. RerF yields a much simpler decision boundary, but struggles because the two dimensions have different variances. Only RerF(r) finds a single split that performs approximately optimally.

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 [4]. However, previously proposed approaches use methods that are time- and space-intensive compared to standard random forests. We propose a simple strategy: "**compute the mean difference vectors**". In other words, given a two-class problem, let $\hat{\delta} = \hat{\mu}_0 - \hat{\mu}_1$, 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). Thus, we construct $\widetilde{X}$ by concatenating $A^{\mathsf{T}}\bar{X}$ with $\Delta = (\delta_{(2)} - \delta_{(1)}, \ldots, \delta_{(C)} - \delta_{(1)})$, where $\delta_{(j)}$ is the δ vector for the class with the j^{th} largest number of samples. Computing this matrix is extremely fast, because it does not rely on costly singular value decompositions, matrix inversions, or convex problems. 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(d+r). Note that if we relax the spherically symmetric assumption above, to diagonal covariance matrices, then it is more natural to use $\delta(k) = (\mu_0(k) - \mu_1(k))/\sigma(k)$, where $\sigma(k)$ is the standard deviation of the k^{th} dimension. However, after passing to ranks, each dimension has the same variance, so this additional step adds an unnecessary complication.

4 Experimental Evaluation

4.1 Axis Aligned Simulations

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

Trunk is a well-known binary classification in which each class is distributed as a p-dimensional multivariate Gaussian with identity covariance matrices [11]. 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.

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 left panels of Figure 2 show two-dimensional scatter plots from each of the three example simulations (using the first two ambient dimensions). The middle panels show the misclassification rate against the number of observed dimensions p, for RF, several RerF variants, and the Bayes optimal error. The right panels show training time against the number of observed dimensions p for all four classifiers.

In both cases, RerF(d+r) performs as well or better than RF, even though the discriminant boundary is naturally axis aligned. The top row shows that RerF outperforms RF even in the axis aligned Trunk example. This is because 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. Looking at the right panel, RerFs has comparable speed to RF. In the bottom row, we observe that all variants of RerF outperform RF for all numbers of dimensions in the parity problem. 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. Looking at the right panels, we observe a slight increase in training time of all RerF variants compared to RF in both simulation settings. 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.

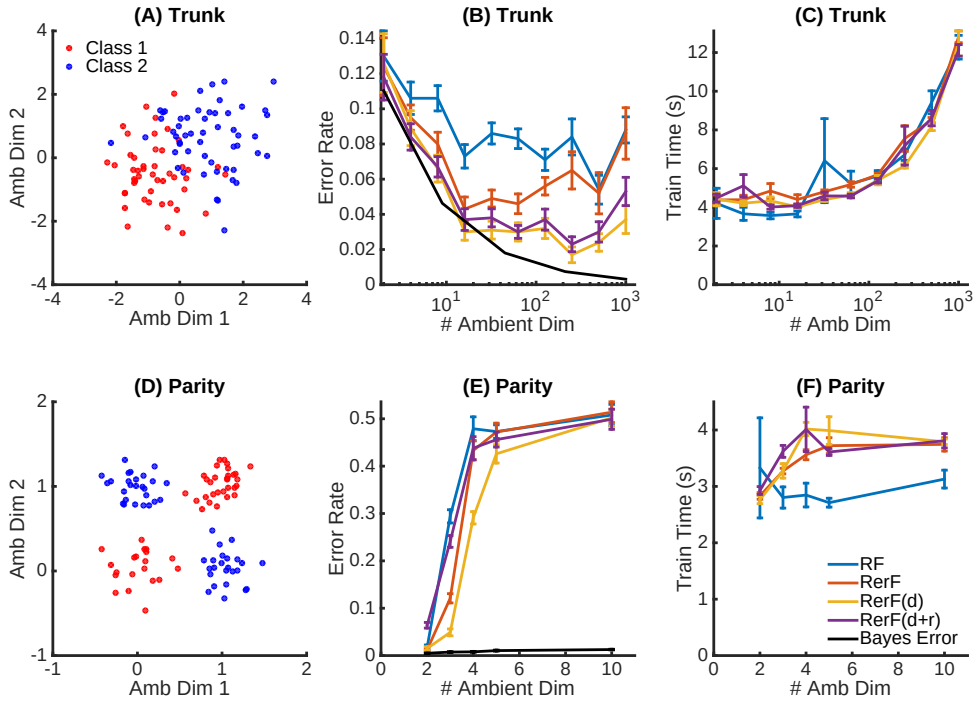

Figure 2: Classification performance comparing Random Forest (RF) to several variants of Randomer Forest (RerF), and Bayes optimal performance, on two distinct simulation settings: (A) Trunk and (B) Parity (see main text for details). For all settings, the left panels show a 2D scatter plot of the first 2 coordinates (dashed circles denote the standard deviation level set), the middle and right panels depict misclassification rate and training time against the number of ambient (coordinate) dimensions, respectively, RF to several variants of RerF, as well as the Bayes optimal performance. In each case, $n = 100$ and the errorbars denote standard error over ten trials. Note that in all settings, for all number of dimensions, RerF outperforms RF, even though these settings were designed specifically for RF because the discriminant boundary naturally lies along the coordinate basis. Training time of RerFs scales similarly to RF.

4.2 Theoretical Space and Time Complexity

For a random forest, assume there are L trees. If there are n data points per tree, and each tree grows until terminal nodes have only $\mathcal{O}(1)$ data points with d coordinates in them, there are $\mathcal{O}(n)$ nodes. Then the complexity of constructing the random forest, disregarding cross-validation or other randomization techniques for preventing overfitting, is $\mathcal{O}(Ln^2 d \log n)$. 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 $\mathcal{O}(Lnd \log n)$.

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 n data points per tree, and no pruning, so $\mathcal{O}(n)$ nodes. Like RF, RerF requires $\mathcal{O}(d)$ time to sample d non-zero numbers, and $\mathcal{O}(dn_k)$ time to obtain the new matrix, $\widetilde{X}$, because it is a sparse matrix multiplication, in node k with $\mathcal{O}(n_k)$ points. RerF also takes another $\mathcal{O}(d/n \log(d/n)) = \mathcal{O}(1)$ time to find the best dimension. Thus, in total, in the case of well-balanced trees, RerF also requires only $\mathcal{O}(Ldn^2 \log n)$ time to train. To store the resulting RerF requires $\mathcal{O}(Ln \log n)$, 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 $\mathcal{O}(1)$. The only additional space constraint is storing which indices are positive, and which are negative, which is merely another constant. The cost of initially sorting in RerF(r), and of computing the means of the classes in RerF(d) are negligible compared to the main cost above. 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

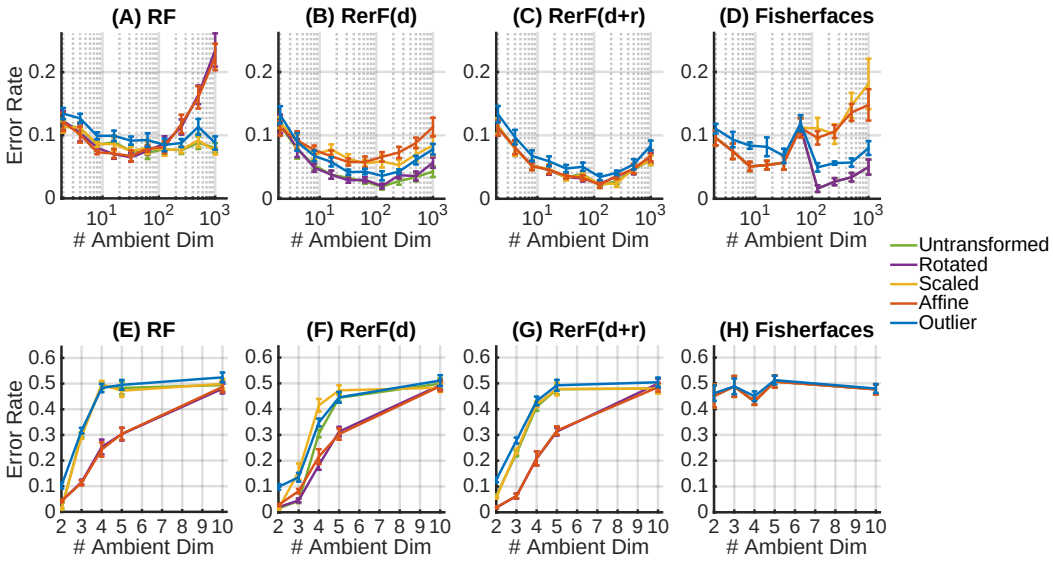

Figure 3: The effect of various transformations applied to the Trunk (panels A-D) and Parity (panels E-H) simulations (see Methods for details) on classification performance of (A,E) RF, (B,F) RerF (d), (C,G) RerF (d+r), and (D,H) Fisherfaces. Specifically, we consider, rotations, scalings, and affine transformations, as well as introducing outliers. Classification performance of RF is compromised by rotations and therefore affine transformations as well. RerF(d) is invariant to rotation, but not scaling and therefore not affine transformation. RerF(d+r) is invariant to to affine transformations. Like RerF (d), Fisherfaces is invariant to rotation but not scaling. Note that all variants are reasonably robust to outliers.

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 [12]. The misclassification rate for Fisherfaces was estimated using leave-one-out cross validation.

Figure 3 shows the effect of these transformations and outliers on the Trunk (top) and Parity (bottom) rows. Figure 3(A) shows that RF is sensitive to rotations, and therefore affine transformations, but robust to outliers. Figure 3(B) shows that RerF(d) is sensitive to scale, and therefore affine transformations, but not outliers. Figure 3(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 (for the same reason that RerF does better on the untransformed Parity setting). On the other hand, Fisherfaces, which is a fundamentally linear classifier, achieves chance levels no matter what on this example.

4.4 Benchmark Data

In addition to the simulations, RF and RerF variants were evaluated on all 121 datasets as described in [2]. 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. The number of trees used in each algorithm was 500. All algorithms were trained using three different values for the dimension s of the projected subspace, and error rates for each algorithm were selected as the minimum given by the three. Figure 4(A) depicts average misclassification rates and training times for each algorithm. The average misclassification rate of RerF is marginally lower than RF, while the other variants have marginally higher misclassification rates. Figure 4B depicts pairwise comparisons of individual misclassification rates between RF to RerF. The black line is the region where RF and RerF have equal performance. The Wilcoxon signed rank test indicates a larger misclassification rate for RF compared to RerF (p = .02). This suggests that RerF is significantly better than RF on this benchmark suite. Additionally, we were curious how the number of trees required to achieve a stable misclassification rate for each algorithm compared. Figure 4C depicts the average out of bag error of each algorithm as a function of the number of trees used in the ensemble. We see that all algorithms require a comparable number of trees in order to achieve a stable misclassification rate.

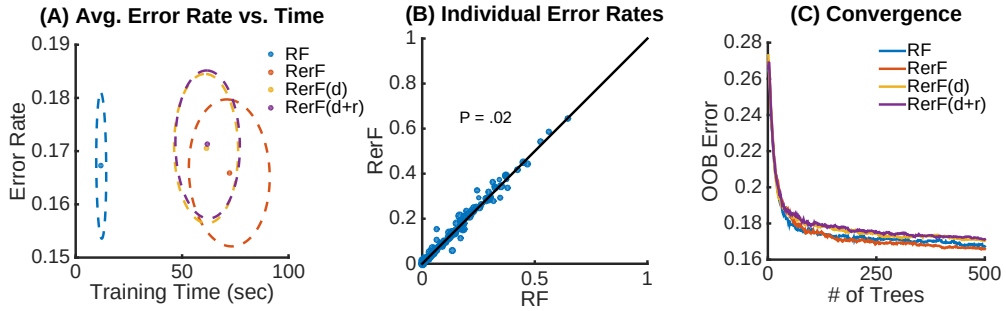

Figure 4: Comparison of RF versus RerF variants on the benchmark suite utilized in Delgado [2]. (A) Classification error and training time of RF and RerF variants for all 121 datasets. Mean (dots) and 0.1 standard deviation level sets (dashed lines) for the joint distribution of accuracy and time are shown for each method. (B) Classification error of RerF(d+r) vs. that of RF for each of the 121 datasets. The black line indicates equal classification error of the two algorithms. RerF is statistically significantly better than RF, with a p-value of 0.02 (Wilxon signed rank test). (C) Mean out of bag error for RF and RerF variants as a function of the number of trees used in the ensemble. All algorithms converge similarly.

5 Conclusion

We have proposed novel methods for constructing decision forests, which we call RerFs. We view these methods as special cases of a more general linear threshold forests, which include Breiman's original Forest-IC and Forest-RC, as well as previously proposed oblique decision forests. RerF,

unlike other oblique decision forests, preserve the time and space complexity of Forest-IC, as well as scale and unit invariance. Moreover, RerFs are demonstrated to perform as well or better than RF in a variety of both simulated and real data sets.

Much work is still to be done with our proposed methods. The simplicity of these ideas suggest that they will be amenable to theoretical investigation, extending the work of Biau et al. (2008) to the RerF setting [13]. Moreover, we hope that theoretical investigations will yield more insight into which distributions $f_A(\mathcal{D}_n)$ will be optimal under different distributional settings, both asymptotically and under finite sample assumptions. Even under the currently proposed $f_A(\mathcal{D}_n)$, 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 (which will be open sourced after blind revision) is a proof-of-concept MATLAB implementation. We will utilize recent GPU and semi-external memory methods to significantly speed up RerF [**?**]. Faster implementations will enable us to more easily investigate the contexts under which RerFs outperform RF, as we currently only know that we expect RerFs to perform better. 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. Even in the absense of all these improvements, we are cautiously optimistic that RerFs may prove to be a better default classifier than RF at this time.

References

[1] L. Breiman, "Random forests," *Machine Learning*, vol. 4, no. 1, pp. 5–32, October 2001.

[2] 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*, vol. 15, no. 1, pp. 3133–3181, October 2014.

[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] D. Heath, S. Kasif, and S. Salzberg, "Induction of oblique decision trees," *Journal of Artificial Intelligence Research*, vol. 2, no. 2, pp. 1–32, 1993.

[5] B. Menze, B. Kelm, D. Splitthoff, U. Koethe, and F. Hamprecht, "On oblique random forests," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, Eds. Springer Berlin Heidelberg, 2011, vol. 6912, pp. 453–469. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23783-6_29

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

[7] T. K. Ho, "The random subspace method for constructing decision forests," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 8, pp. 832–844, Aug 1998.

[8] 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*, vol. 28, no. 10, pp. 1619–1630, 2006.

[9] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *CoRR*, vol. abs/0912.3599, 2009. [Online]. Available: http://arxiv.org/abs/0912.3599

[10] 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*. ACM, 2006, pp. 287–296.

[11] G. Trunk, "A problem of dimensionality: A simple example," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 3, pp. 306–307, 1979.

[12] P. N. Belhumeur, J. P. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 711–720, 1997.

[13] G. Biau, L. Devroye, and G. Lugosi, "Consistency of random forests and other averaging classifiers," *The Journal of Machine Learning Research*, vol. 9, pp. 2015–2033, 2008.