

000  
001  
002  
003  
004  
005  
006  
007  
008  
009  
010  
011  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026  
027  
028  
029  
030  
031  
032  
033  
034  
035  
036  
037  
038  
039  
040  
041  
042  
043  
044  
045  
046  
047  
048  
049  
050  
051  
052  
053  
054

---

## Randomer Forests

---

### Abstract

Random forest (RF) has been shown to outperform many other classifiers on a variety of datasets, yet its restriction of recursive partitions of the feature space to be axis-aligned is suboptimal in many cases. Several studies have proposed “oblique” decision forest methods to address this limitation. However, these methods either 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 a special case which we call randomer forests (RerFs). We demonstrate that RerF empirically outperforms RF and another recently proposed oblique forest method in simulations and on benchmark data, and also show that it scales comparably to RF in terms of time and space complexity. Lastly, we propose two augmentations to RerF with the goal of making it more robust to affine transformations and outliers. We conclude that RerF is a competitive alternative to RF and other oblique forests. 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, and a similar arXiv search reports approximately 1000 hits. One of the most popular and best performing classifiers is random forests (RFs) (Breiman, 2001). Two recent benchmark papers assess the performance of many different classification algorithms on many different datasets (Fernandez-Delgado et al., 2014; Caruana et al., 2008), and both concluded the same thing: RFs are the best classifier.

RF typically refers to Breiman’s Forest-IC, which uses axis-parallel (Heath et al., 1993), or orthogonal trees (Menze et al., 2011). 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, 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 (Heath et al., 1993; Tan & Dowe, 2005), using principal components analysis to find the directions of maximal variance (Ho, 1998; Rodriguez et al., 2006), or directly learning good discriminant directions (Menze et al., 2011). Another recently proposed method, called random rotation RF, uniformly randomly rotates the data for every decision tree in the ensemble prior to inducing the tree (Blaser & Fryzlewicz, 2015). While all of these recent approaches deal with rotational invariance, they fail to address two important issues. First, 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. Second, 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 Forest-RC, all oblique methods require expensive computation and/or storage. There is therefore a gap that calls for the construction of a scalable

055  
056  
057  
058  
059  
060  
061  
062  
063  
064  
065  
066  
067  
068  
069  
070  
071  
072  
073  
074  
075  
076  
077  
078  
079  
080  
081  
082  
083  
084  
085  
086  
087  
088  
089  
090  
091  
092  
093  
094  
095  
096  
097  
098  
099  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109

110  
111  
112  
113  
114  
115  
116  
117  
oblique decision forest classifier that can perform well in  
the presence of an overwhelming number of irrelevant fea-  
tures. It is true that Forest-RC fits this description. How-  
ever, the number of features that are linearly combined to  
generate candidate splits is fixed across all nodes of every  
tree. Such a construction requires an additional hyperpa-  
rameter to be tuned, and also limits the diversity of trees.

118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
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 en-  
ables us to propose a few novel special cases, which we  
refer to as “*Randomer Forests*” (or RerFs for short), for rea-  
sons that will become clear in the sequel. We both theoreti-  
cally and empirically demonstrate that our methods have  
the same time and space complexity as random forests, and  
show on simulated datasets that it performs well in the face  
of many irrelevant features and when axis-aligned splits are  
suboptimal. Additionally, we demonstrate that our method  
empirically outperforms both RF and random rotation RF  
(RotRF), which is another oblique method (Blaser & Fry-  
zlewicz, 2015), on a suite of benchmark datasets. Lastly,  
we propose two augmentations to RerF for making the  
method more robust to affine transformations and outliers.  
To conclude, we propose our method as a competitive al-  
ternative to RF and other oblique decision forest methods.  
Open source code will be made available.

## 138 139 2. Random Projection Forests

140  
141 Let  $\mathcal{D}^n = \{(X_i, y_i) : i \in [n]\}$  be a given dataset, where  
142  $X_i \in \mathbb{R}^p$  and  $y_i \in \mathcal{Y} = \{\tilde{y}_1, \dots, \tilde{y}_C\}$ . A classifica-  
143 tion forest,  $\bar{g}(X; \mathcal{D}^n)$  is an ensemble of  $L$  decision trees,  
144 each tree  $g^l(X; \mathcal{D}^l)$  is trained on a (sub)set of the data,  
145  $\mathcal{D}^l \subset \mathcal{D}^n$ . Random Projection forests are a special case  
146 of classification forests that subsume all of the strategies  
147 mentioned above (see Pseudocode 1). The key idea of all  
148 of them is that at each node of the tree, we have a set of  
149 predictor data points,  $\bar{X} = \{X_s\}_{s \in \mathcal{S}_{ij}^l} \in \mathbb{R}^{p \times S_{ij}^l}$ , where  
150  $S_{ij}^l = |\mathcal{S}_{ij}^l|$  is the cardinality of the set of predictor data  
151 points at the  $(ij)^{th}$  node of the  $l^{th}$  tree. We sample a  
152 matrix  $A \sim f_A(\mathcal{D}^n)$ , where  $A \in \mathbb{R}^{p \times d}$ , possibly in a  
153 data dependent fashion, which we use to project the predi-  
154 tor matrix  $\bar{X}$  onto a lower dimensional subspace, yielding  
155  $\tilde{X} = A^\top \bar{X} \in \mathbb{R}^{d \times S_{ij}^l}$ , where  $d \leq p$  is the dimensionality  
156 of the subspace. To wit, Breiman’s original Forest-IC  
157 algorithm can be characterized as a special case of ran-  
158 dom projection forests. In particular, in Forest-IC one constructs  
159  $A$  such that for each of the  $d$  columns, we sample a coordi-  
160 nate (without replacement), and put a 1 in that coordinate,  
161 and zeros elsewhere. Breiman’s Forest-RC constructs  $A$  by  
162 sampling a fixed number of coordinates (without replace-  
163 ment) for each of the  $d$  columns, and puts a value uniformly  
164

sampled from [-1,1] in each of those coordinates. 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  $f_A$ .

165  
166  
167  
168  
169  
170  
171  
172  
**Algorithm 1** Psuedocode for Random Projection Forests,  
which generalizes a wide range of previously proposed de-  
cision forests.

173  
174  
175  
176  
**Input:** data:  $\mathcal{D}^n = (X_i, y_i) \in (\mathbb{R}^p \times \mathcal{Y})$  for  $i \in [n]$ , tree  
rules (stopping criteria, rules for sampling data points  
per tree, etc.), distributions on  $d \times p$  matrices:  $A \sim$   
 $f_A(\mathcal{D}^n)$ , preprocessing rules

177  
178  
**Output:** decision trees, predictions, out of bag errors,  
etc.

179  
Preprocess according to rule

180  
**for** each tree **do**

181  
182 Subsample data to obtain  $(\bar{X}, \bar{y})$ , the set of data points  
to be used in this tree

183  
**for** each leaf node in tree **do**

184  
Let  $\tilde{X} = A^\top \bar{X} \in \mathbb{R}^{d \times s}$ , where  $A \sim f_A(\mathcal{D}^n)$

185  
Find the “best” split coordinate  $k^*$  in  $\tilde{X}$  and “best”  
split value  $t^*(k^*)$  for this coordinate

186  
Split  $X$  according to whether  $X(k) > t^*(k^*)$

187  
Assign each child node as a leaf or terminal node  
according to stopping criteria

188  
**end for**

189  
**end for**

190  
Prune trees according to rule

191  
192  
The goal of this work is to find a random projection for-  
193 est that shares RF’s scalability and ability to perform well  
194 when many irrelevant predictors are present, yet is able to  
195 find decision boundaries that are less biased by geometri-  
196 cal constraints (i.e. not constrained to be axis-aligned), by  
197 changing the distribution  $f_A$ . The result we call randomer  
198 forests (or RerFs for short).

## 199 200 3. Randomer Forests

201  
Our choice in  $f_A$  is based on the following three ideas:

- 202  
203  
204  
205  
1. While RF empirically performs well in many settings,  
206 it is quite restrictive in that candidate splits evaluated  
207 at each node are constrained to be axis-aligned. It is  
208 sometimes the case that axis-aligned splits are subop-  
209 ptimal, and oblique splits may be desired.
- 210  
211  
2. It is often the case that the number of *relevant* fea-  
212 tures is substantially smaller than the number of total  
213 features. We say that the signal is *compressive*  
214 in such cases. RF often does well when the signal  
215 is compressive, which may be attributed, in part, to  
216 the extreme sparsity constraint on the columns of  $A$ .  
217 All of the oblique decision forest algorithms to date,  
218

219

220	with the exception of Forest-RC, do not impose a sparsity constraint on the candidate split directions, which may cause such algorithms to underperform when the signal is compressive.	275
221		276
222		277
223		278
224		279
225	3. Except for Breiman's Forest-RC, existing oblique decision forest algorithms involve expensive computations to identify and select splits, rendering them less space and time efficient than Forest-IC or Forest-RC. An oblique decision forest having a space and time complexity comparable to Forest-IC or Forest-RC is desirable.	280
226		281
227		282
228		283
229		284
230		285
231		286
232		287
233	To this end, we employ very <b>sparse random projections</b> (Li et al., 2006). 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 $A$ .	288
234		289
235		290
236		291
237		292
238		293
239		294
240	It is apparent that RerF bares a resemblance to Forest-RC, yet there is one key difference. 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. Lastly, we note that our construction of $A$ preserves distances between points with high probability (Li et al., 2006).	295
241		296
242		297
243		298
244		299
245		300
246		301
247		302
248		303
249		304
250		305
251		306
252		307
253		308
254	<b>4. Experimental Results</b>	309
255		310
256	<b>4.1. Simulations Involving Compressive Signals</b>	311
257	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, and RotRF:	312
258		313
259		314
260		315
261		316
262		317
263	<b>Sparse parity</b> 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	318
264		319
265		320
266		321
267		322
268		323
269		324
270		325
271		326
272		327
273		328
274		329
	$p - p*$ dimensions as noise.	
	<b>Trunk</b> is a well-known binary classification in which each class is distributed as a $p$ -dimensional multivariate Gaussian with identity covariance matrices (Trunk, 1979). The means of the two classes are $\mu_1 = (1, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}, \dots, \frac{1}{\sqrt{p}})$ and $\mu_2 = -\mu_1$ . It follows from this that the signal-to-noise ratio of the $i$ th dimension asymptotically decreases to zero with increasing $i$ .	
	RotRF is an oblique decision forest method that has been recently proposed in (Blaser & Fryzlewicz, 2015). RotRF works exactly like RF, except that the feature space is uniformly randomly rotated prior to inducing each decision tree. This induces oblique splits in an unstructured way, which the authors argue is favorable to structured methods such as in Rodriguez's rotation forest, because such structured methods reduce the diversity of the trees. Uniformly random rotations of the feature space imply that in general, splits will not be sparse. Therefore, we conjecture that RotRF 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.	
	Figure 1 depicts the true class posteriors in the first two dimensions of the sparse parity simulation in panel A and estimates of the posteriors for RF, RerF, and RotRF in panels B, C, and D, respectively. For this simulation, $p = 20$ , $p* = 3$ , 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 1000. Various values of $d$ , the number of columns in the random projection matrix $A$ , were tried and the best for each algorithm was selected. RerF gives estimates of the posteriors closest to the true posteriors.	
	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 misclassification rate relative to RF against the number of observed dimensions $p$ , for RerF and RotRF. Relative misclassification rate was computed as the difference between the misclassification rate of either RerF or RotRF and that of RF. The misclassification rate of RF relative to itself is shown for reference. The right panels show training time against the number of observed dimensions $p$ for all four classifiers. For all comparisons, we used the same number of trees for each method to enable a fair comparison. 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. Gini im-	

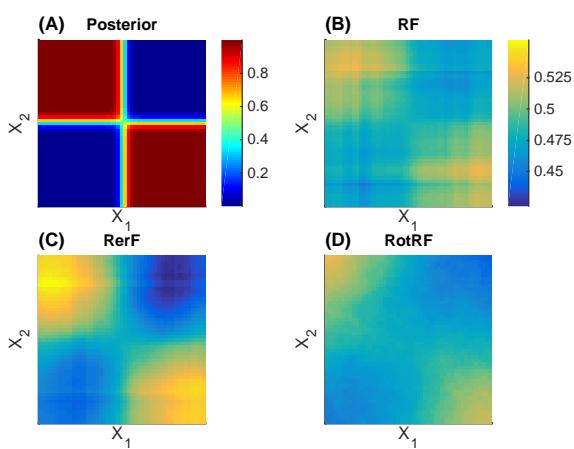


Figure 1. Class posteriors for the sparse parity problem in the first two dimensions (refer to section 4.1 for details). (A): True posteriors. (B) - (D): Posterior estimates for RF, RerF, and RotRF, respectively. Comparing panels B-D with A, it is evident that RerF produces better estimates of the posteriors than does RF or RotRF

purity was used as the split criteria. The only parameter tuned was  $d$ , the number of candidate split directions evaluated at each split node. When  $p \leq 5$ , each classifier was trained for all  $d \in [p]$ . When  $p > 5$ , each classifier was trained for all  $d \in \{1, p^{1/4}, p^{1/2}, p^{3/4}, p\}$ . Misclassification rates for each classifier were selected as the lowest achieved from the different values of  $d$  tried. Training times for each classifier were computed by averaging the training times 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  $p \in \{2, 10, 50, 100, 500, 1000\}$ . RotRF was not evaluated for  $p = 1000$  due to computational burden. Relative error and training times plotted are the average of 25 repeated trials, and error bars represent the standard error of the mean.

In panel B, RerF performs as well as or better than both RF and RotRF for all values of  $p$ . RotRF performs as well as or better than RF except for when  $p = 25$ . As conjectured, RotRF 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 RotRF. Ultimately, when this ratio is small enough, RotRF performs even worse than RF. RerF's ability to perform relatively well can be attributed to the sparsity of oblique splits. In panel E, RerF outperforms 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. RotRF outperforms both RF and RerF up to  $p = 100$ . RotRF is able to perform better than RerF 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 RotRF significantly degrades and becomes worse than both RF and RerF. This can be explained by the fact that when  $p$  is large enough, RotRF 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 RotRF is significantly longer than those of RF and RerF when  $p = 500$ .

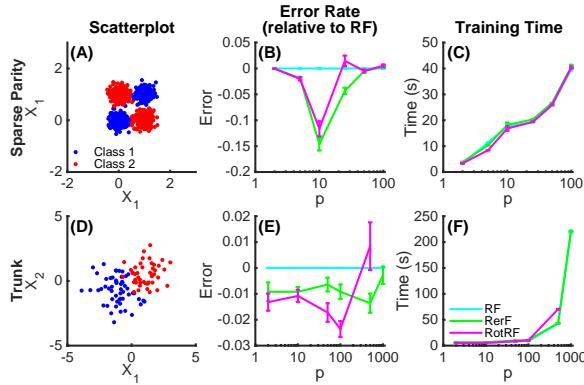


Figure 2. Sparse parity (A-C) and Trunk (D-F) simulations (see section 4.1 for details). (A) and (D): Scatterplots of sampled points in the first two dimensions. (B) and (E): Error rates of RerF and RotRF relative to RF across different values of  $p$ . (C) and (F): Same as (B) and (E) except absolute training time is plotted on the y-axis instead. The cyan, green, and magenta lines correspond to RF, RerF, and RotRF, respectively. This color coding is adopted in figures 3 and 4 that follow. Both RerF and RotRF do better than RF when the number of irrelevant features is sufficiently small, due to their ability to generate oblique partitions. However, when the number of irrelevant features becomes large enough, performance of RotRF rapidly degrades. Training times show that RerF scales identically with RF, while RotRF scales poorly with large  $p$  (note that in panels E and F, RotRF is only plotted up to  $p = 500$  due to computational constraints).

## 4.2. Theoretical Space and Time Complexity

For a RF, 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  $p$  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  $\mathcal{O}(Ln^2p \log n)$ . In practice the trees are shallower and

330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384

385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439

stop much earlier than when nodes have  $\mathcal{O}(1)$  points, so “in practice” the complexity often appears to be  $\mathcal{O}(Lnp \log n)$ . Storing RF requires  $\mathcal{O}(Ln \log n)$  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  $n$  data points per tree, and no pruning, so  $\mathcal{O}(n)$  nodes. Let  $n_k$  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  $\mathcal{O}(pn_k)$  time to obtain the new matrix,  $\tilde{X}$ , because it is a sparse matrix multiplication, in node  $k$  with  $\mathcal{O}(n_k)$  points. RerF also takes another  $\mathcal{O}(p/n \log(p/n)) = \mathcal{O}(1)$  time to find the best dimension. Thus, in total, in the case of well-balanced trees, RerF also requires only  $\mathcal{O}(Lpn^2 \log n)$  time to train. To store the resulting RerF, like RF, 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  $\mathcal{O}(1)$ . Note that these numbers are in stark contrast to other oblique methods. RotRFs, 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  $\mathcal{O}(np^2)$ . Therefore, RotRF 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, RotRF 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 of RF, RerF, and RotRF to various data transformations. 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  $[0,10]$  to each dimension. 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 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 out-

liers on the sparse parity (panels A-E) and Trunk (panels F-J) problems. Panels A-E show that RerF and RotRF outperform RF in both untransformed and transformed representations for Sparse Parity. This makes sense because linear combinations of variables can have some information about the class, whereas the original features are all uninformative on their own. For the trunk simulations, panel G shows that RotRF is more robust to rotations (as it should be), but is fragile in the face of affine transformations.

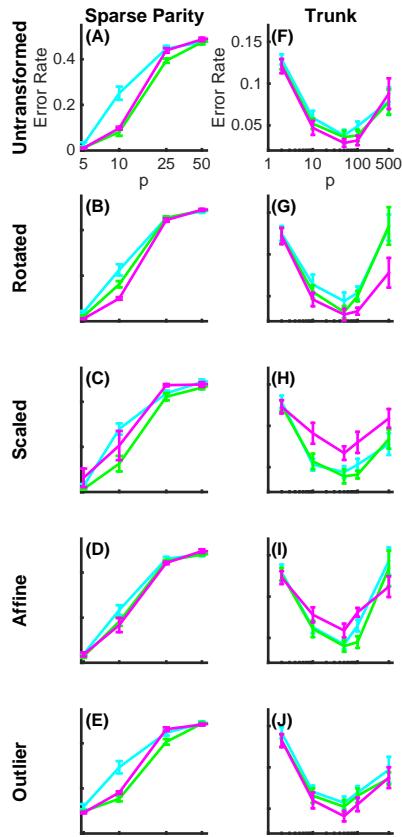


Figure 3. The effects of different transformations applied to the sparse parity (A-E) and Trunk (F-J) simulations on classification performance (see section 4.3 for details). Specifically, we consider rotations, scalings, affine transformations, as well as the addition of outliers. RerF and RotRF outperform RF on sparse parity. The trunk simulations demonstrate clearly that RotRF, while invariant to rotation, is sensitive to affine transformations.

### 4.4. Benchmark Data

In addition to the simulations, RF, RerF, and RotRF were evaluated on 115 of the 121 datasets as described in (Fernandez-Delgado et al., 2014). The six remaining datasets were not used because their high dimensionality

495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549

and large number of data points rendered the classifiers both time and space costly, particularly for RotRF. 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, misclassification 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. All algorithms were trained using five different values for  $d$ , and 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 panels A-D of Figure 4. Such plots are called performance profiles (Dolan & Moré, 2008). Performance profiles are useful in visualizing how frequently a particular algorithm wins, and when it loses, how frequently it loses by a certain amount. Areas under the curves (AUCs) were computed for each performance profile. This metric is valuable in that it doesn't necessarily indicate how frequently a particular algorithm is the best. Rather, it is an indicator of robustness. For instance, it is possible that an algorithm that wins on the majority of datasets can have a lower AUC than the other algorithms if, when it loses, it frequently loses by a large margin. On the original benchmarks (Figure 4 panel A), RerF achieves the largest AUC, followed by RF and lastly RotRF. The same trend is seen when scaling (panel C) and affine transformations (panel D) are applied. When the datasets are rotated, RotRF achieves the largest AUC, followed by RerF, and lastly RF.

Since the simulations in section 4.1 revealed that the speed of RF and RerF scaled comparably, we wanted to see if this observation held on the benchmark data. In the same fashion as the left panels of Figure 4, we plotted performance profiles for training times in the right panels. Across all transformations, we see that RF and RerF perform comparably in terms of speed. Panels A and C show that RotRF performs the worst on the original benchmark data and the randomly scaled benchmark data. It performs well, however, on the rotated and affine transformed data. One possible explanation for this is that when the data is rotated, RF and RerF have a more difficult time finding good split directions, which could result in deeper trees (and subsequently more time spent in training).

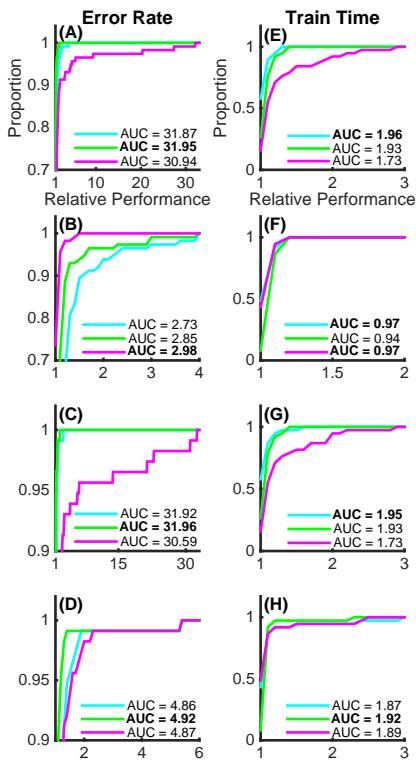


Figure 4. Classification (panels A-D) and speed (panels E-H) performance profiles on benchmarks with various transformations applied (see section 4.4 for details). AUC denotes the area under the curve. In terms of classification accuracy, RerF outperforms RF in all five data settings and only loses to RotRF when the data is rotated. RF and RerF have comparable training time performance, while RotRF tends to be slower.

#### 4.5. Rank Transforming the Data and Fast Supervised Projections

When dimensions of  $X$  are linearly combined, we lose *scale and unit 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. 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**”, or rank transform, prior to doing anything else. We call RerFs that pass to ranks RerF(r).

Additionally, there is evidence that *using the data to construct A at each node* can significantly improve performance (Heath et al., 1993). However, previously proposed

550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604

605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659

approaches for using the data require 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  $\tilde{X}$  by concatenating  $A^\top$  with  $\Delta = (\delta_{(2)} - \delta_{(1)}, \dots, \delta_{(C)} - \delta_{(1)})$ , where  $\delta_{(j)}$  is the  $\delta$  vector for the class with the  $j^{th}$  largest number of samples, to obtain  $\tilde{A}^\top$ . Then we compute  $\tilde{X} = \tilde{A}^\top \tilde{X}$ . 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. Furthermore, it also provides a set of dense projections to supplement the sparse projections of RerF. Denote RerFs that include this matrix RerF(d), and if they pass to ranks first, RerF(d+r).

Figure 5 demonstrates the utility of passing to ranks and using supervised projections on various transformations of the Trunk simulation. Panels A and B again show that RF and RerF are sensitive to rotation and therefore affine transformations. Panel C shows that RerF(d) achieves a lower error rate than RerF in the untransformed setting and is not sensitive to rotation. However, it is sensitive to scale and therefore affine transformations. Panel D shows that RerF(d+r) is more robust to affine transformations than the other methods, as hoped. Panel E again shows that RotRF is sensitive to scale and subsequently 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-IC and Forest-RC, as well as previously proposed oblique decision forests. 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 over 100 benchmark datasets, as real data often has the properties just described. Moreover, RerF, unlike other oblique decision forests, preserves the time and space complexity of Forest-IC, and is only marginally sen-

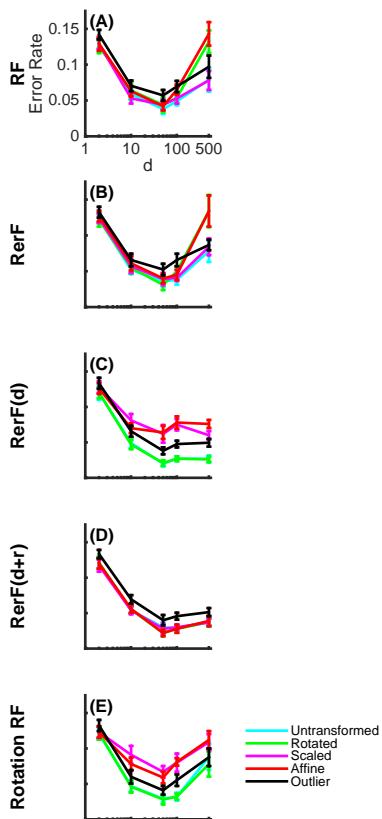


Figure 5. The utility of supplementing RerF with supervised projections and rank transforming the data prior to inducing trees (see section 4.5 for details). The effect of various transformations on classification performance of RF, RerF, RerF(d), RerF(d+r), and RotRF for the Trunk simulation are shown in panels A-E, respectively. Panel D shows that RerF(d+r), which supplements RerF with supervised projections, and passes the data to ranks, is more robust to affine transformations than the other methods.

sitive to scaling. Lastly, we propose two additional modifications. One involves supplementing sparse random projections with a particular form of fast supervised projections in the construction of candidate split directions. The other involves passing the data to ranks with the goal of making the classifier more robust to outliers.

Much work is still to be done with our proposed methods. The simplicity of RerFs suggest that they will be amenable to theoretical investigation, extending the work of Biau et al. (2008) to the RerF setting (Biau et al., 2008). 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 cur-

660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714

715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769

770	rently proposed $f_A(\mathcal{D}_n)$ , our implementation has room for	825
771	improvement. Although it has the same space and time	826
772	complexity as RF, we will determine explicit constants,	827
773	and improve our implementation accordingly. Indeed, our	828
774	current implementation is a proof-of-concept MATLAB	829
775	implementation. We will utilize recent GPU and semi-	830
776	external memory methods to significantly speed up RerF	831
777	(Zheng et al., 2015). As with all decision forests, multi-	832
778	class classification is but one exploitation task they can be	833
779	used for; therefore, we will also extend this work to enable	834
780	regression, density estimation, clustering, and hashing. We	835
781	will provide open source code to enable more rigorous and	836
782	extended analyses by the machine learning community.	837
783		838
784		839
785		840
786	<b>References</b>	841
787	Biau, G., Devroye, L., and Lugosi, G. Consistency of ran-	842
788	dom forests and other averaging classifiers. <i>The Journal</i>	843
789	of <i>Machine Learning Research</i> , 9:2015–2033, 2008.	844
790	Blaser, R. and Fryzlewicz, P. Random rotation ensem-	845
791	bles. 2015. URL <a href="http://stats.lse.ac.uk/fryzlewicz/rre/rre.pdf">http://stats.lse.ac.uk/fryzlewicz/rre/rre.pdf</a> .	846
792		847
793	Breiman, L. Random forests. <i>Machine Learning</i> , 4(1):5–	848
794	32, October 2001.	849
795		850
796	Caruana, R., Karampatziakis, N., and Yessenalina, A. An	851
797	empirical evaluation of supervised learning in high di-	852
798	mensions. <i>Proceedings of the 25th International Con-</i>	853
799	ference on Machine Learning, 2008.	854
800	Dolan, E. D. and Moré, J. J. Benchmarking optimization	855
801	software with performance profiles. <i>Mathematical Pro-</i>	856
802	gramming, 91:201–213, 2008.	857
803		858
804	Fernandez-Delgado, M., Cernadas, E., Barro, S., and	859
805	Amorim, D. Do we need hundreds of classifiers to solve	860
806	real world classification problems? <i>Journal of Machine</i>	861
807	Learning Research, 15(1):3133–3181, October 2014.	862
808		863
809	Heath, D., Kasif, S., and Salzberg, S. Induction of oblique	864
810	decision trees. <i>Journal of Artificial Intelligence Re-</i>	865
811	<i>search</i> , 2(2):1–32, 1993.	866
812	Ho, T. K. The random subspace method for constructing	867
813	decision forests. <i>Pattern Analysis and Machine Intelli-</i>	868
814	<i>gence, IEEE Transactions on</i> , 20(8):832–844, Aug 1998.	869
815	ISSN 0162-8828. doi: 10.1109/34.709601.	870
816		871
817	Li, P., Hastie, T. J., and Church, K. W. Very sparse random	872
818	projections. In <i>Proceedings of the 12th ACM SIGKDD</i>	873
819	<i>international conference on Knowledge discovery and</i>	874
820	<i>data mining</i> , pp. 287–296. ACM, 2006.	875
821		876
822	Menze, B. H., Kelm, B.M, Splitthoff, D. N., Koethe,	877
823	U., and Hamprecht, F. A. On oblique ran-	878
824	dom forests. In Gunopulos, Dimitrios, Hofmann,	879