Randomer Forests

Abstract

1. Introduction

2. Linear Threshold Forests

Let $\mathcal{D}^n = \{(X_i, y_i) : i \in [n]\}$ be a given dataset, where $X_i \in R^p$ 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, $\bar{X} = \{X_s\}_{s \in \mathcal{S}^l_{ij}} \in R^{p \times S^l_{ij}}$, 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 $A \in R^{p \times d}$, possibly in a data dependent fashion, which we use to project the predictor matrix $\bar{X}$ onto a lower dimensional subspace, yielding $\widetilde{X} = A^\mathsf{T} X \in R^{d \times S^l_{ij}}$, 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 posseses many of the properties that RF has, yet is able to find decision boundaries that are less biased by geometrical constraints, in part by changing the distribution f_A. The result we call randomer forests (or RerFs for short).

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

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

inputData: $\mathcal{D}^n = (X_i, y_i) \in (R^p \times \mathcal{Y})$ 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 each tree Subsample data to obtain $(\bar{X}, \bar{y})$, the set of data points to be used in this tree each leaf node in tree Let $\widetilde{X} = A^\mathsf{T} \bar{X} \in R^{d \times s}$, 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 Prune trees according to rule

3. Randomer Forests

4. Experimental Results

4.1. Simulations Involving Compressive Signals

We constructed two synthetic datasets to compare classification performance and training time of RF, RerF, and RotRF:

Sparse parity is a variation of the parity problem. The parity problem is a multivariate generalization of the XOR problem and is one of the hardest constructed binary classification problems. In parity, 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 the of the basic parity problem in which the sample's class label is equal to the parity of only the first p* elements of the mean, rendering the remaining p - p* dimensions as noise.

Trunk is a well-known binary classification in which each class is distributed as a p-dimensional multivariate Gaussian with identity covariance matrices (**?**). 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.

For all comparisons, we used the same number of trees for each method to enable a fair comparison. The only parameter tuned was mtry, the number of candidate split directions evaluated at each split node. The left panels of Figure **??** show two-dimensional scatter plots from each of the three 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 misclassifcation 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.

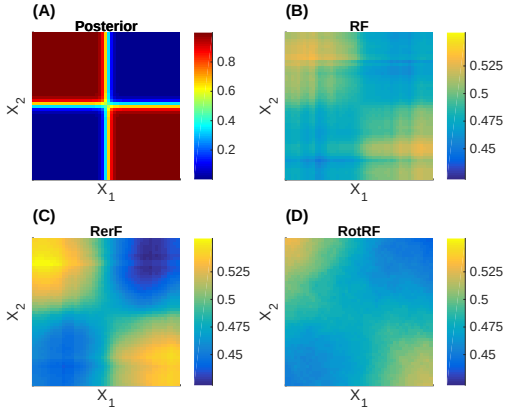

Figure 1. Class posteriors for the sparse parity problem in the first two dimensions. (A): True posteriors. (B) - (D): Posterior estimates for RF, RerF, and RotRF, respectively.

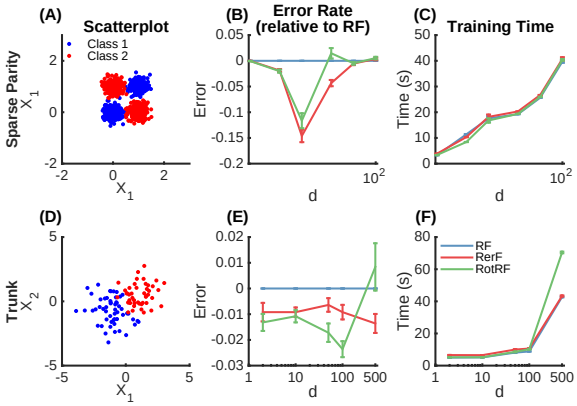

Figure 2. Sparse parity (A-C) and Trunk (D-F) simulations. (A) and (D): Scatterplots of sampled points for p = 2. (B) and (E): Error rates of RerF and RotRF relative to RF across different values of p. (C) and (F): Same as (B) and (E) except absolute training time is plotted on the y-axis instead.

4.2. Effects of Transformations and Outliers

4.3. Benchmark Data

4.4. Rank Transforming the Data and Fast Supervised Projections

5. Discussion

6. Conclusion

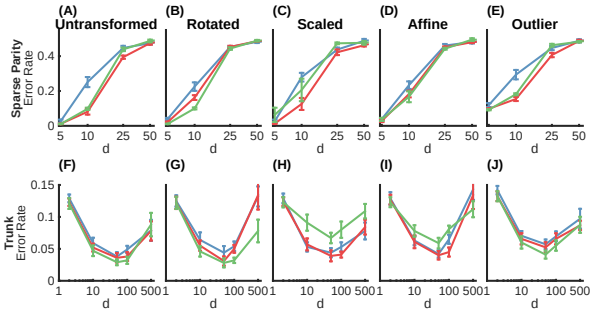

Figure 3. The effects of different transformations applied to the sparse parity (A-E) and Trunk (F-J) simulations on classification performance. Specifically, we consider rotations, scalings, affine transformations, as well as the addition of outliers.

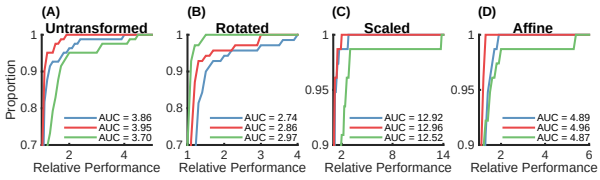

Figure 4. Performance profiles on benchmarks with various transformations applied. AUC denotes the area under the curve.