A Projection Pursuit Forest Algorithm for Supervised Classification

Natalia da Silva

Instituto de Estadística (IESTA), Universidad de la República
and
Dianne Cook
Department of Econometrics and Business Statistics, Monash University
and Eun-Kyung Lee
Department of Statistics, Ewha Womans University

September 8, 2019

Abstract

This paper presents a new ensemble learning method for classification problems called projection pursuit random forest (PPF). PPF uses the `PPtree` algorithm introduced in Lee et al. (2013). In PPF, trees are constructed by splitting on linear combinations of randomly chosen variables. Projection pursuit is used to choose a projection of the variables that best separates the classes. Utilizing linear combinations of variables to separate classes takes the correlation between variables into account which allows PPF to outperform a traditional random forest when separations between groups occurs in combinations of variables.

The method presented here can be used in multi-class problems and is implemented into an R (R Core Team, 2019) package, `PPforest`, which is available on CRAN.

Keywords: data mining, ensemble model, statistical computing, exploratory data analysis, high dimensional data

1 Introduction

There are two main features from a random forest algorithm (Breiman, 2001) that are broadly applicable for building ensemble classifiers from any basic method: bootstrap aggregation (bagging) and random predictor selection. Bagging stabilizes the variance (Breiman, 1996a,b) and random predictor selection reduces correlation (Amit and Geman, 1997; Ho, 1998) between trees in the forest (ensemble components).

This paper presents the projection pursuit random forest (PPF), a new ensemble learning method for classification problems, built from trees utilizing combinations of predictors. PPF builds a forest from many projection pursuit trees (PPtree) (Lee et al., 2013), which is available in the R package `PPtreeViz` (Lee, 2018). Projection pursuit (Friedman and Tukey, 1974) is used to find the linear combination of variables that best separates groups, and many different rules to make the actual split are provided.

Trees that use linear combinations of predictors in a split are known in the literature as oblique trees (Kim and Loh, 2001; Brodley and Utgoff, 1995; Tan and Dowe, 2004; Truong, 2009; Lee et al., 2013). All these algorithms use different approaches for finding linear combinations of predictors upon which to make a split. Some of the methods used for selecting the linear combination include random coefficient generation, linear discriminant analysis, and linear support vector machines. Theoretically, these could also be used as a base underlying PPF.

In the machine learning literature numerous work has been conducted on algorithms for building forests from oblique trees (Tan and Dowe, 2006; Menze et al., 2011; Do et al., 2010). The performance is reported to be better than random forests. A limitation of building on these approaches is the lack of readily available software.

In a PPF, for each split, a random sample of predictors is selected, an optimal linear combination for separating the classes is computed by using projection pursuit. The algorithm is targeted for problems where classes can be separated by linear combinations of predictors, which define separating hyperplanes that are oblique to the axes rather than orthogonal to them. This may sound like a support vector machine (Cortes and Vapnik, 1995), but the tree construction makes it different – the separation in linear combinations

is examined for subspaces of the high-dimensional space. PPF also accommodates imbalanced classes by using stratified bootstrap samples, and variable importance measures are computed using the coefficients of the projections. PPF can be used for multi-class problems and is implemented into an R package, called `PPforest`.

PPF can outperform RF, and an application is included to demonstrate this. However, it has the same interpretability and diagnostics as RF, which helps analysts to better understand the data.

This paper is organized as follows. Section 2 explains PPtree algorithm underlying PPF. Section 3 describes the PPF algorithm; diagnostics, including how to compute variable importance and the implementation details. Section 4 evaluates the algorithm performance on benchmark machine learning data in comparison with other tree methods. Section 5 explains the diagnostics in comparison to those from RF, and discusses parameter selection. An application illustrating the power of PPF over other tree methods is provided in Section 6. Section 7 discusses possible extensions and future directions.

2 Background on the projection pursuit tree

The projection pursuit algorithm searches for low dimensional projections that optimize a function which measures some aspect of interest, for PPtree, this is class separation. Friedman and Tukey (1974) coined the term "projection pursuit", but the ideas existed earlier than this, and are discussed in Kruskal (1969). Indexes for finding projections that separate classes were developed in Lee et al. (2005) (LDA) and Lee and Cook (2010) (PDA). These form the basis of PPtree, and are explained below.

Let $\mathbf{x}_{gi}$ be a p-dimensional data vector, representing the i^{th} observation of the g^{th} class, with $g = 1, \ldots, G$, G is the number of classes, $i = 1, \ldots, n_g$, and n_g is the number of observations in class g. The LDA index is defined as follows:

$$I_{LDA}(A) = \begin{cases} 1 - \frac{|A^T W A|}{|A^T (W+B) A|} & \text{for } |A^T(W+B)A| \neq 0 \\ 0 & \text{for } |A^T(W+B)A| = 0 \end{cases} \quad (1)$$

where $p \times k$ matrix $A = [a_1, a_2, ..., a_k]$ defines an orthonormal projection from p-dimensions

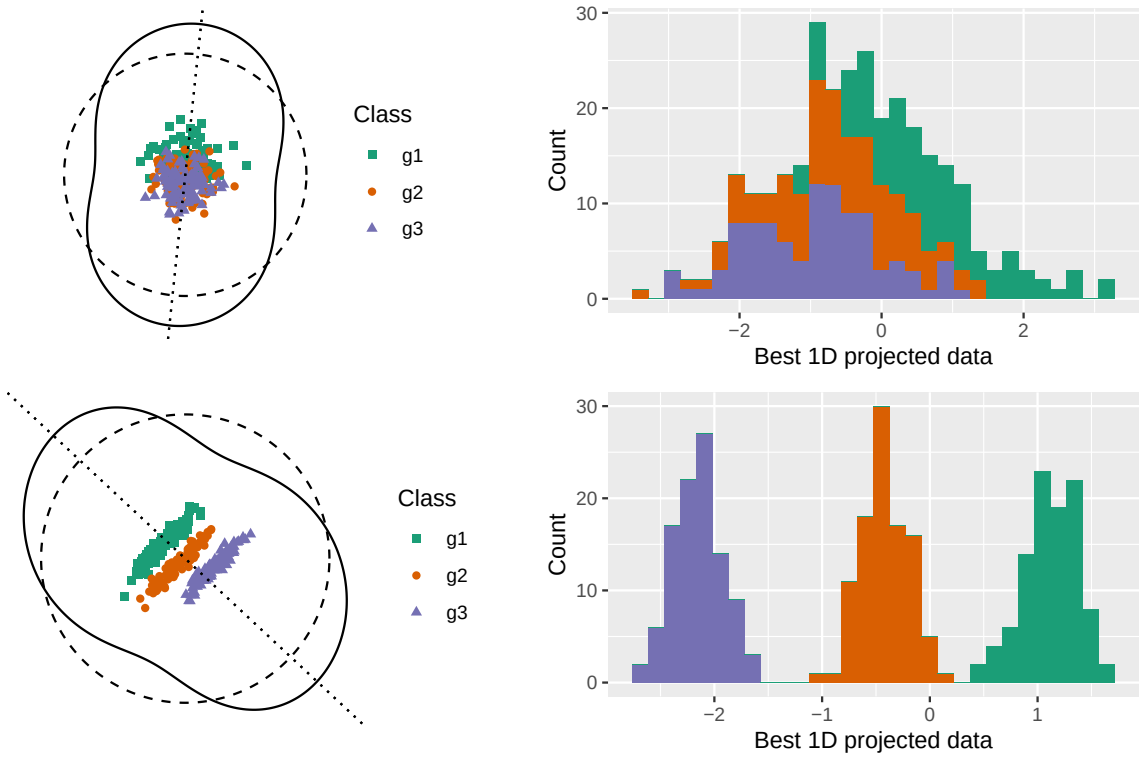

Figure 1: Behavior of the LDA index on 1D projections shown using a Huber plot for simulated data, two different sets of 2D. The solid line indicates index value for each 1D projection of the 2D data. The dashed circle indicates the median index value across all projections, and the dotted line corresponds to the projection producing the maximum index value. The histogram shows the projected data corresponding the maximum index value.

onto a k-dimensional subspace, $B = \sum_{g=1}^{G} n_g(\bar{\mathbf{x}}_{g.} - \bar{\mathbf{x}}_{..})(\bar{\mathbf{x}}_{g.} - \bar{\mathbf{x}}_{..})^T$ is the between-group sum of squares, and $W = \sum_{g=1}^{G} \sum_{i=1}^{n_g}(\mathbf{x}_{gi} - \bar{\mathbf{x}}_{g.})(\mathbf{x}_{gi} - \bar{\mathbf{x}}_{.g})^T$ is the within-group sum of squares. If the LDA index value is high, there is a large difference between classes.

Figure 1 examines the behavior of the LDA index on two 2D simulated data sets, using Huber's plot (Huber, 1990) available in the `PPtreeViz` package (Lee, 2018). This plot shows the projection pursuit index values in all possible directions in a 2D space. These indices are calculated using the projections for $(cos\theta, sin\theta)$, $\theta = 1°, \ldots, 180°$. For each projection, the index value is computed on the projected data, and displayed as the solid line. The circle (dashed line) is a guideline, and corresponds to the median of all index values, with the solid line. The projection corresponding to the maximum index value is indicated as a dotted line, and the projected data is shown in the histogram.

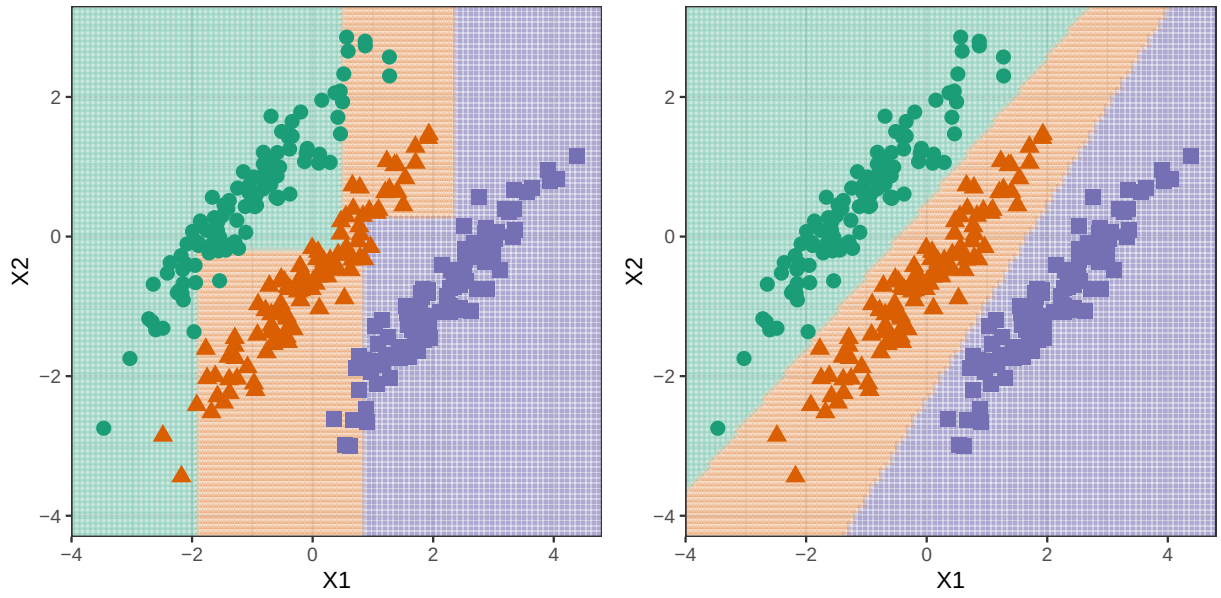

Figure 2: Comparison of decision boundaries for the `rpart` (left) and PPtree (right) algorithms on 2D simulated data. The partitions generated by PPtree algorithm are oblique to the axis, incorporating the association between the two variables.

The PDA index is useful when $n \leq p$ and when the variables are highly correlated. In these situations the maximum likelihood variance-covariance matrix estimator will be close to singular, affecting the inverse calculation. The PDA index adjusts the variance-covariance matrix calculation, as follows:

$$I_{PDA}(A, \lambda) = 1 - \frac{|A^T W_{PDA} A|}{|A^T (W_{PDA} + B) A|} \tag{2}$$

where notation as for the LDA index, with the addition of $\lambda \in [0, 1)$ is a shrinkage parameter, and a different within group sum of squares, $W_{PDA}(\lambda) = \text{diag}(W) + (1 - \lambda)\text{offdiag}(W)$.

The PPtree algorithm uses a multi-step approach to fit a multi-class model by finding linear combinations to split on. Figure 2 compares the boundaries that would result from a classification tree fitted using the rpart algorithm (Therneau et al., 2010) and the PPtree algorithm.

Figure 3 illustrates the PPtree algorithm for three classes, and the algorithm steps are

detailed below. Let $d_n = \{(\mathbf{x_i}, y_i)\}_{i=1}^n$ be the data set where $\mathbf{x_i}$ is a p-dimensional vector of explanatory variables and $y_i \in \mathscr{G}$ ($\mathscr{G} = \{1, 2, \ldots G\}$) represents class information with $i = 1, \ldots n$.

1. Optimize a projection pursuit index to find an optimal one-dimensional projection, α^*, for separating all classes in the current data yielding projected data $z = \alpha^* x$.

2. On the projected data, z, redefine the problem into a two class problem using the distances among the means of classes, and assign a new label, either g_1^* or g_2^* to each observation, generating a new class variable y_i^*. Note that this requires joining two or more groups into one, so that the new groups g_1^* and g_2^* possibly contain more than one of the original classes.

3. Find an optimal one-dimensional projection α^{**}, using $\{(\mathbf{x_i}, y_i^*)\}_{i=1}^n$ to separate the two class problem g_1^* and g_2^*. The best separation of g_1^* and g_2^* is determined in this step providing the decision rule for the node,

   > if $\alpha^{**T} M_1 < c$ then assign g_1^* to the left node else assign g_2^* to the right node,

   where M_1 is the mean of g_1^*.

4. For each group, all the previous steps are repeated until g_1^* and g_2^* have only one class from the original classes. The depth of PPtree is at most the number of classes.

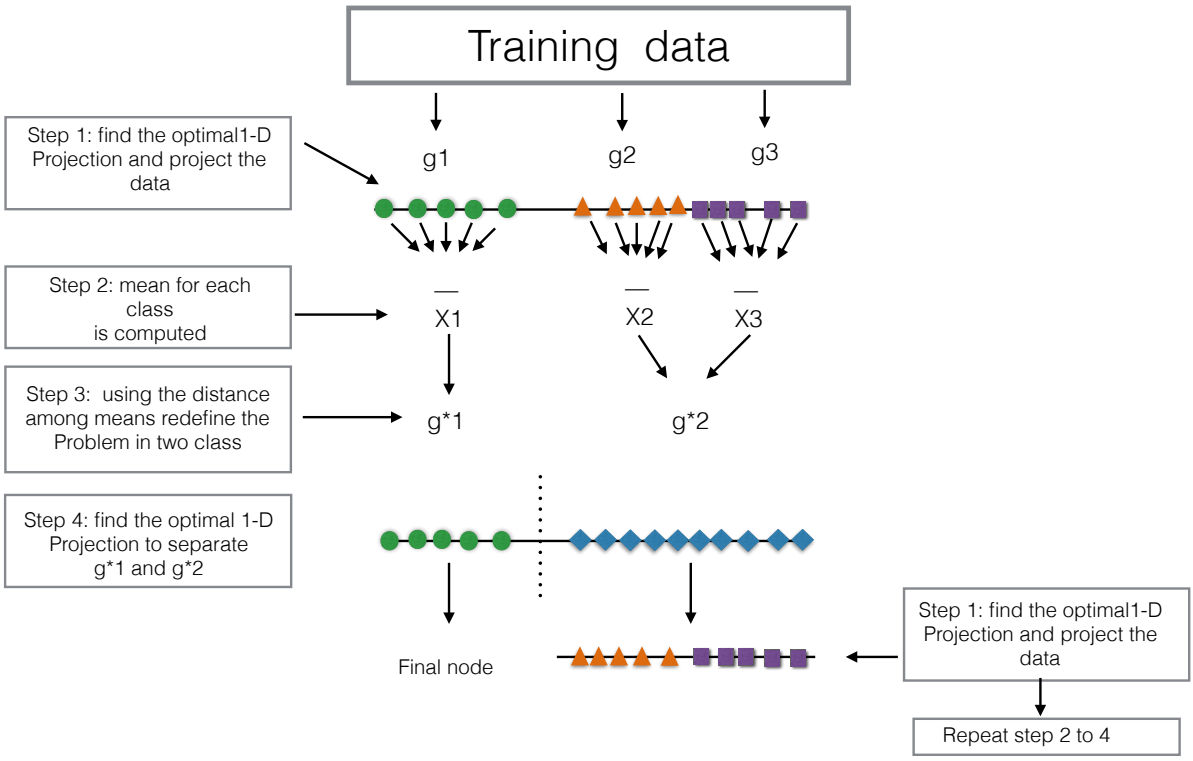

Figure 3: Illustration of the PPtree algorithm for $g = 3$ classes. It is a dual pass algorithm for multiclass problems, for each split. It first finds the best separation and combines classes into two super-groups. It then searches again for the best separation between these two super-groups and splits on this. It proceeds sequentially on the subsets in the nodes, but only $g - 1$ splits are allowed.

3 Projection pursuit random forest

This section provides the definition of PPF for classification, an explanation of the algorithm, implementation details, and a description of how the diagnostics are defined and computed.

3.1 Definition

This definition follows closely the original random forest notation in Breiman (2001), a more recent overview paper by Biau et al. (2008), and is important to provide here for

consistency in the PPF explanation. Let the random vector of predictor variables $\mathbf{X} \in \Re^p$ and the output random variable $Y \in \mathscr{G}$, where $\mathscr{G}$ is a finite set such that $\mathscr{G} = \{1, 2, \ldots, G\}$. The training sample is defined as $D_n = \{(\mathbf{X_1}, Y_1), \ldots (\mathbf{X}_n, Y_n)\}$ of i.i.d $\Re^p \times \mathscr{G}$ random variables ($p \geq 2$). The objective is to build a classifier which predicts y from $\mathbf{x}$ using D_n given an ensemble of classifiers h.

A projection pursuit classification random forest can be defined as a collection of randomized classification trees $\{h_n(\mathbf{x}, \Theta_m, D_n), m \geq 1\}$ where $\{\Theta_m\}$ are i.i.d. random vectors. Θ_m includes the two sources of randomness in the tree (random variable selection and random bootstrap sample), then Θ_m has information about which variables were selected in each partition and which cases were selected in the bootstrap sample.

For each tree, h_n, a unique vote is collected based on the most popular class for the selected predictor variables. Equation 3 defines the PPF estimator based on combining the trees.

$$\begin{aligned} f_n(\mathbf{X}, D_n) &= \arg\max_{g \in \mathscr{G}} \{E_\Theta(I[h_n(\mathbf{X}, \Theta, D_n) = g])\} \\ &= \arg\max_{g \in \mathscr{G}} P_\Theta(h_n(\mathbf{X}, \Theta, D_n) = g) \end{aligned} \tag{3}$$

E_Θ is the expectation wrt Θ, conditionally on $\mathbf{X}$ and D_n. In practice, the PPF estimator is evaluated by generating B random trees and take the average of the individual outcomes. This procedure is justified in a similar way to the original random forest defined by Breiman (2001), and is based on the Law of the Large Numbers (Athreya and Lahiri, 2006).

Equation 4 describes the prediction of a new observation $\mathbf{x_0}$.

$$\hat{f}_n(\mathbf{x_0}) = \arg\max_{g \in \mathscr{G}} \sum_{k=1}^{B} I[h_n(\mathbf{x_0}, \Theta_{bk}) = g] \tag{4}$$

3.2 Algorithm

1. Let $n = \sum_{i=1}^{G} n_i$ the total number of cases in the training set $d_n = \{\mathbf{x_i}, y_i\}_{i=1}^{n}$. B stratified bootstrap samples from d_n are taken. Then for each class, independently

and uniformly re-sample cases from d_{ng} (training data set for group g) with size n_g to create a stratified bootstrap data set $\{bk = b_{k1}, b_{k2}, \ldots b_{kg}\}$.

2. Use a bootstrap sample bk to grow a PPtree ($h_n(\mathbf{x}, \Theta_{bk})$) to the largest extent possible without pruning. (Note that the depth of the PPtree is at most $G - 1$, where G is the number of classes).

   (a) Start with all the cases in b_k in the root node.

   (b) A simple random sample of m predictor variables from the set of all the predictor variables M is drawn, where $m << M$.

   (c) Find the optimal one-dimensional projection α^* to separate all the classes in b_k.

   (d) If more than two class, then reduce the number of classes to two by comparing means, and assign new labels, g_1^* and g_2^* to each case (called the new response y_i^* in b_k).

   (e) Find the optimal one-dimensional projection, α^{**}, using the bootstrap data set with the relabeled response, y^*, to separate g_1^* and g_2^*. The linear combination is computed by optimizing a projection pursuit index to get a projection of the variables that best separates the classes using the m random selected variables. Two index options are available LDA or PDA.

   (f) Compute the decision boundary c. Eight different rules to define the cutoff value of each node can be used. All the rules are defined in Lee (2018).

   (g) Keep α^{**} and c.

   (h) Separate the data into two groups using the new labels g_1^* and g_2^*.

   (i) Repeat from (b) to (h) if g_1^* or g_2^* have more than two original classes.

3. Repeat 2 for $k = 1, \ldots B$.

4. The output is the ensemble of PPtrees, $\{h_n^{bk}\}_{k=1}^B$.

Split values on the projected data can be computed by one of eight methods, which use the group means, or medians, sample size and variance or IQR weighting

Figure 4 has a diagram illustrating the PPforest algorithm.

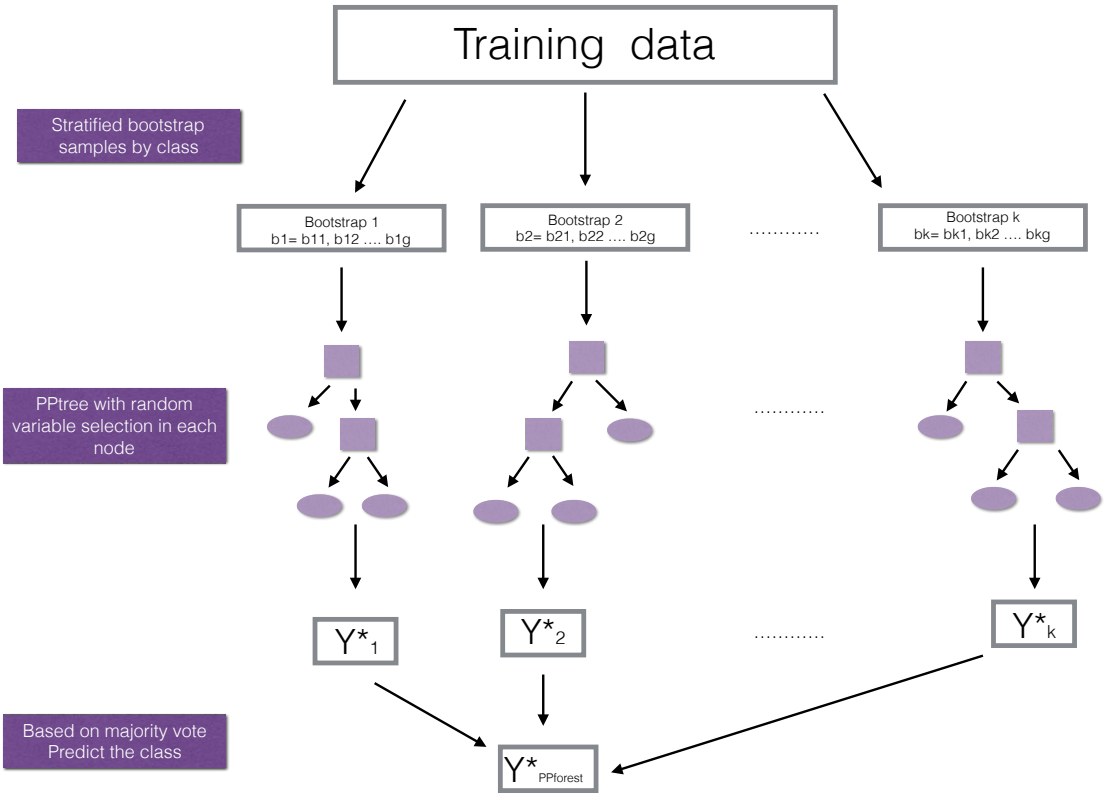

Figure 4: Illustration of the PPforest algorithm. It is effectively the same as a random forest algorithm except that the PPtree classifier is used on each bootstrap sample.

3.3 Implementation

The R package `PPforest` (da Silva et al., 2018), provides an implementation of PPF. (The development version is available at `https://github.com/natydasilva/PPforest`.) The initial code for PPforest was developed entirely in R. It was subsequently profiled using `profvis` (Chang and Luraschi, 2016), and two code optimization strategies were employed: translate main functions into `Rcpp` (Eddelbuettel et al., 2011) and parallelization using `plyr`. The `microbenchmark` package was used to compare the speed before and after optimization, and also for the comparison in Figure 5 which shows the performance for a range of sample size (n), dimension (p), groups (g), number of trees (m) and proportion of variables considered at each node. The proportion has the largest effect on speed.

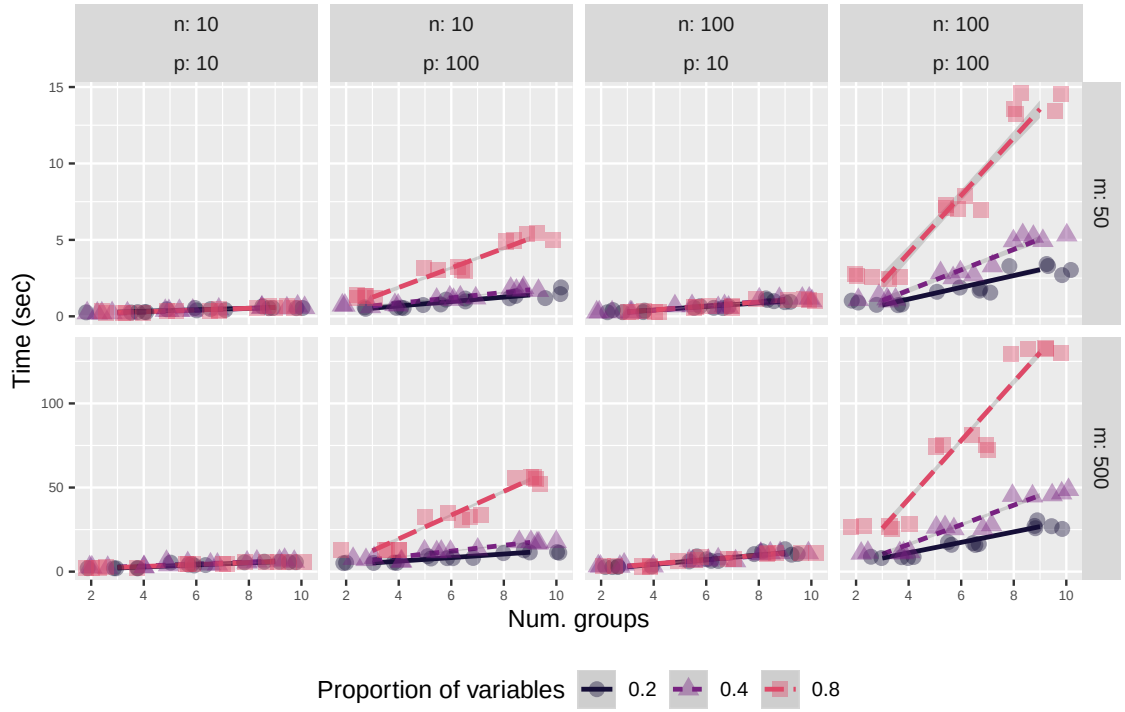

Figure 5: Computational performance for different sample size (n), number of variables (p), number of groups (g), number of trees (m) and proportion of variables considered for each node. Computational speed suffers most when the proportion of variables is high.

3.4 PPF diagnostics

The process of bagging and combining results from multiple trees produces numerous diagnostics which can provide a lot of insight into the class structure in high dimensions. Because ensemble methods are composed of many models fitted to subsets of the data, many statistics can be calculated to be analyzed as a separate data set. This provides the ability to understand how the model is working. The diagnostics of interest are the error rate, variable importance measure, vote matrix, and proximity matrix. These diagnostics are used to assess model complexity, individual model contributions, variable importance and dimension reduction, and uncertainty in prediction associated with individual observations. Most of the diagnostics work as in RF, with the exception being the variable

importance measure, which is specific to PPforest.

3.4.1 Model error

Using the out-of-bag (oob) cases from bagged trees in the forest construction allows ongoing estimates of the generalization error for an ensemble of trees, described in Breiman (2001). Given a training data set d_n, B bootstrap samples from d_n are taken. For each bootstrap sample ($b = 1, 2, \ldots B$), a `PPtree` classifier $h_n(\mathbf{x}, \Theta_b)$ is constructed, and a majority vote is used to get the PPF predictor. The oob cases are used to get the error rate estimates. For each $\{\mathbf{x_i}, y_i\}$ in d_n, the votes are aggregated only for the classifiers $h_n(\mathbf{x}, \Theta_b)$ that do not contain $\{\mathbf{x_i}, y_i\}$. Hence, PPF is called the out-of-bag classifier, and the error rate for this classifier (out-of-bag error rate) is the estimate of the generalized error. The out-of-bag error rate is a measure for each model that is combined in the ensemble and is used to provide the overall error of the ensemble.

3.4.2 Variable importance

PPF calculates variable importance in two ways: (1) permuted importance using accuracy, and (2) importance based on projection coefficients on standardized variables. The permuted variable importance is comparable to the measure defined in the classical random forest algorithm. It is computed using the oob cases for the tree k $(B^{(k)})$ for each X_j predictor variable. Then the permuted importance of the variable X_j in the tree k can be defined as:

$$IMP^{(k)}(X_j) = \frac{\sum_{i \in B^{(k)}} I(y_i = \hat{y}_i^{(k)}) - I(y_i = \hat{y}_{i,P_j}^{(k)})}{|B^{(k)}|}$$

where $\hat{y}_i^{(k)}$is the predicted class for the observation i in the tree k, and $y_{i,P_j}^{(k)}$ is the predicted class for the observation i in the tree k after permuting the values for variable X_j. The global permuted importance measure is the average importance over all the trees in the forest.

This measure is based on comparing the accuracy of classifying oob observations using the true class with permuted (nonsense) class.

For the second importance measure, the coefficients of each projection are examined. The magnitude of these values indicates importance if the variables have been standardized. The variable importance for a single tree is computed by a weighted sum of the absolute values of the coefficients across node, then the weights take the number of classes in each node into account(cl_{nd}) (Lee et al., 2013) . The importance of the variable X_j in the PPtree k can be defined as:

$$IMP^{(k)}_{pptree}(X_j) = \sum_{nd=1}^{nn} \frac{|\alpha^{(k)}_{nd}|}{cl_{nd}}$$

where $\alpha^{(k)}_{nd}$ is the projected coefficient for node ns and variable k and nn the total number of node partitions in the tree k.

The global variable importance in a PPforest then can be defined in different ways. The most intuitive are the average variable importance from each PPtree across all the trees in the forest.

$$IMP_{ppforest1}(X_j) = \frac{\sum_{k=1}^{K} IMP^{(k)}_{pptree}(X_j)}{K}$$

Alternatively, a global importance measure is defined for the forest as a weighted mean of the absolute value of the projection coefficients across all nodes in every tree. The weights are based on the projection pursuit indexes in each node (Ix_{nd}), and 1-(OOB-error of each tree)(acc_k).

$$IMP_{ppforest2}(X_j) = \frac{\sum_{k=1}^{K} acc_k \sum_{nd=1}^{nn} \frac{Ix_{nd}|\alpha^{(k)}_{nd}|}{nn}}{K}$$

3.4.3 Observational level diagnostics

The vote matrix is an uncertainty measure for each observation, across models, is the proportion of times that a case is predicted to be in each class. If a case is always predicted to be the one class, there is no uncertainty about its group, and if this matches the true class then it is correctly labeled. Cases that are predicted to be multiple classes, indicate difficult-to-classify observations. These cases may be important in that they might indicate special attention is needed in some neighborhoods of the data space, or more simply, could

be errors in measurements in the data.

In a tree, each pair of observations can be in the same terminal node or not. Tallying this up across all trees in a forest gives the proximity matrix, an $n \times n$ matrix of the proportion of trees that the pair shares a terminal node. A proximity matrix can be considered to be a similarity matrix. This is typically used for follow-up cluster analysis to assess the strength of the class structure, and whether there are additional unlabeled clusters.

4 Performance comparison

This section presents simulation results and a benchmark data study to examine the predictive performance of PPF in comparison to other methods. In the benchmark data study, PPF is compared with PPtree, CART and RF. The simulation results are designed to compare PPF with RF on data with linear projections defining class differences.

The performance of PPF is compared with the classification methods, PPtree, CART and RF using 10 benchmark data sets taken from the UCI Machine Learning archive (Lichman, 2013). Table 1 presents summary information about the benchmark data, number of groups, cases, and predictors for each data set. The *imbalance* between groups is measured by the range of group size proportions and *correlation* is the average of all pairwise correlation coefficients among predictor variables.

Table 1: Overview of benchmark data: number of cases, predictors, groups, imbalance and correlation. Imbalance indicates relative class sizes (0=balanced classes), and higher correlation indicates the potential for separations occurring in combinations of variables.

Data	Cases	Predictors	Groups	Imbalance	Correlation
crab	200	5	4	0.00	0.95
lymphoma	80	50	3	0.41	0.75
NCI60	61	30	8	0.07	0.56
parkinson	195	22	2	0.51	0.50
fishcatch	159	6	7	0.31	0.46
leukemia	72	40	3	0.40	0.44
olive	572	8	9	0.32	0.35
wine	178	13	3	0.13	0.30
image	2310	18	7	0.00	0.28
glass	214	9	6	0.31	0.23

For each benchmark data set, 2/3 of the observations are randomly chosen and used for training while the remaining 1/3 are used as test data for computing predictive error. This procedure is repeated 200 times and the mean error rate is reported in Table 2. In PPF, the number of variables selected in each node partition is a tuning parameter, the proportion of variables selected at each partition. Three different values were used (0.6, 0.9 and the RF default). The test error reported for PPF is the best from these.

The results show that PPF has a better performance in the test data set than the other methods for the crab, fishcatch, leukemia, lymphoma, olive and wine data, while the RF test error is smaller for glass, image, NCI60 and parkinson data. When there is a big improvement over RF, PPtree also performs well, and the reason is that the difference between classes is in combinations of variables so using linear combinations provides better performance. PPF has the advantage, over PPtree, of providing additional diagnostics for a problem, so even though performance is similar, the diagnostics obtained are beneficial. Overall, PPF performs comparably to these other tree-based methods, and provides consistently low error rates.

Table 2: Comparison of CART, PPtree, RF and PPF results with various data sets. The mean of training and test error rates from 200 re-samples is shown. (Order of rows is same as in Table 1.) PPF performs favorably compared to the other methods.

	TRAINING				TEST			
Data	CART	PPtree	RF	PPforest	CART	PPtree	RF	PPforest
crab	0.277	0.044	0.244	0.046	0.453	0.057	0.238	0.057
lymphoma	0.052	0.000	0.100	0.000	0.155	0.069	0.081	0.053
NCI60	0.503	0.000	0.458	0.019	0.676	0.423	0.376	0.388
parkinson	0.081	0.175	0.107	0.112	0.159	0.229	0.101	0.171
fishcatch	0.118	0.000	0.193	0.000	0.184	0.012	0.191	0.011
leukemia	0.037	0.000	0.033	0.000	0.146	0.049	0.032	0.030
olive	0.072	0.048	0.053	0.037	0.119	0.068	0.052	0.048
wine	0.050	0.001	0.019	0.001	0.127	0.021	0.021	0.018
image	0.069	0.067	0.024	0.079	0.082	0.073	0.024	0.083
glass	0.237	0.331	0.240	0.306	0.330	0.403	0.224	0.390

Figures 6 displays the performance comparison graphically. Each line connects the errors for one data set. Even though RF outperforms PPF on almost half the data (Table 2) PPF tends to have consistently low error.

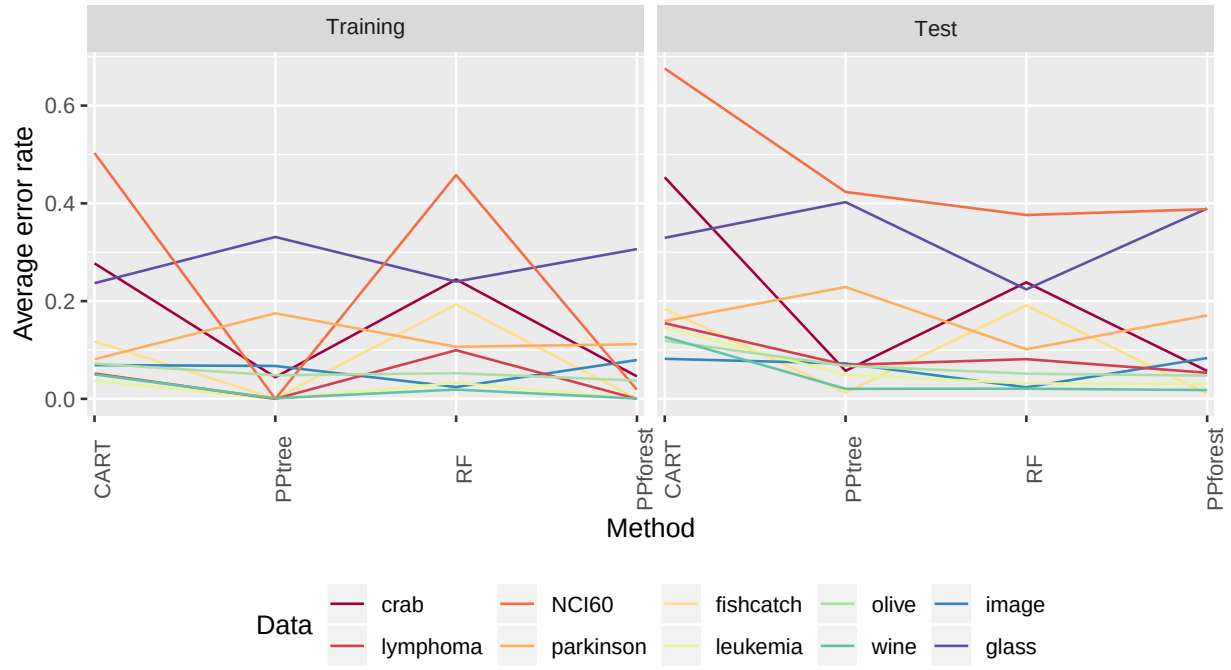

Figure 6: Benchmark data results shown graphically. PPF performs consistently well across most of the data sets.

5 Diagnostics comparison

The diagnostics computed by PPF (Section 3) and RF are compared for the lymphoma data, which helps to understand why and how PPF outperforms RF with this data.

5.1 Variable importance

Figure 7 illustrates how the variable importance differs, using the lymphoma data. PPF outperformed RF for this data. There are three groups, and it is a high-dimension, low sample size data set. With PPF, the PDA index is used, and the 60% of variables are available at each node. The number of trees used is the same as the RF default. Only the top ten most important variables are shown. There are some common on both lists and a some differences. Showing just the first two variables from each list is sufficient to illustrate the different type of boundaries induced by the classifiers. The two ways of computing importance in PPF do produce a different hierarchy of variables. With the global average importance, Gene35 and Gene50 are the top two, and these distinguish the

small group FL best. With the global importance, Gene35 and Gene44 are featured, and together these find a big gap between DLBCL and the other two groups. PPF is utilizing the association between variables to classify groups, as would be expected.

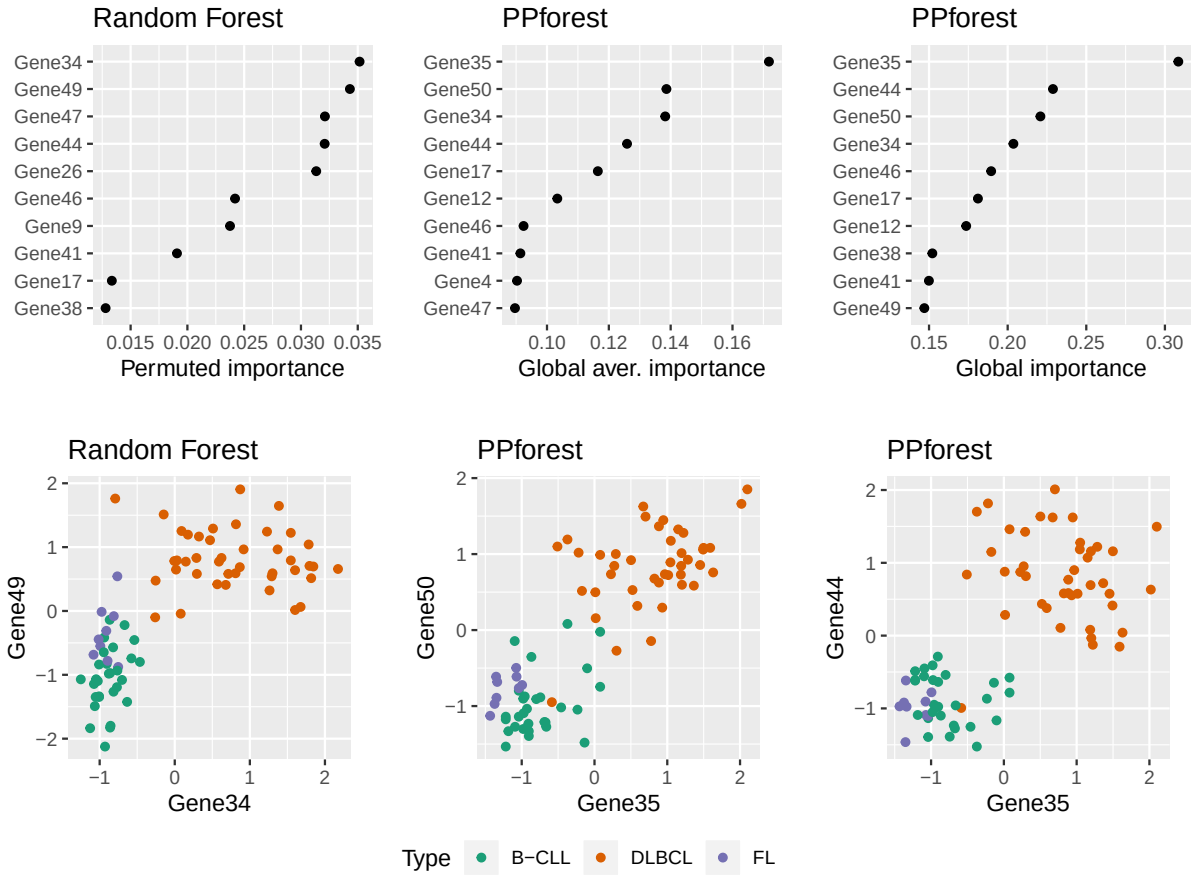

Figure 7: Comparison of importance measures for the lymphoma data, where PPF outperformed RF. Top row shows the top 10 variables by each method, with two ways of calculating with PPF. Bottom row shows the top two variables from each, which illustrates the difference between methods. PPF is detecting differences between groups when there is association between variables. Using the global average importance (middle), Gene35 and Gene50 better distinguish group FL. Using the global importance, Gene35 and Gene44 find a big gap between group DLBCL and the other two.

5.2 Vote matrix

Figure 8 shows the vote matrices returned by PPF and RF for three classes of the lymphoma data. It is represented in two ways: as a ternary plot and as a side-by-side jittered dotplot.

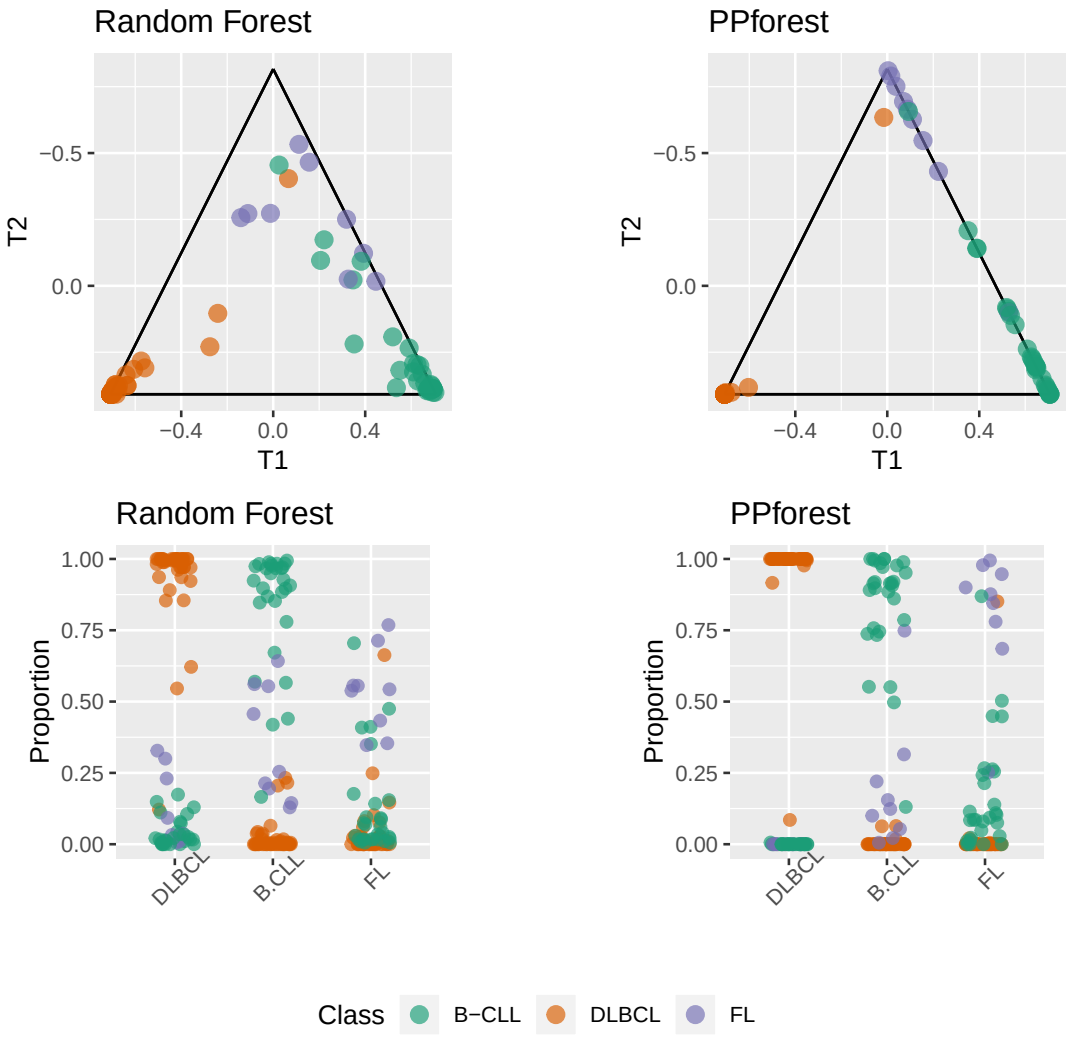

Figure 8: Comparison of the out-of-bag vote matrix for the three groups of the lymphoma data, returned by RF and PPF: (top) ternary plot, (bottom) side-by-side jittered dotplots. This illustrates the difference between methods. PPF votes more decisively for most cases, than RF. Especially this is true for the DLCBL class, where all but one are almost always predicted to the true class.

The vote matrix has three columns corresponding to the proportion of times the case was predicted to be class B-CLL, DLBCL or FL, and thus is constrained to lie in a 2D triangle in 3D space. A ternary diagram is created using a helmert transformation of the vote matrix to capture the 2D subspace. The way to read it is: points near the vertex are clearly predicted to be one class, points along an edge are confused between two classes, and points in the middle are confused between the three classes. PPF provides more distinct classification of observations than RF, because the points are more concentrated in the vertices, and along one edge.

The side-by-side jittered dotplot is an alternative representation that readily can be used for any number of classes. The proportion each case is classified to a group is displayed vertically along a horizontal axis representing the categorical class variable. Points are jittered a little horizontally to better see the distribution of proportions, and colour represents the true class. Points concentrated at the top part indicate cases that are clearly grouped into a class, and if the colour matches the true class then these are correct classifications. The message is similar to the ternary diagram: DLBCL is much more clearly distinguished by PPF, and FL is actually distinguishable from B-CLL by PPF but confused by RF.

5.3 Proximity

Figure 9 shows multidimensional scaling plots of the proximity matrix produced by PPF and RF classification of the lymphoma data. PPF provides the cleaner proximities. This means that more frequently observations from the same class reside in the same terminal node of the trees making up the PPF, than those of RF.

5.4 Parameter selection

The primary parameters for PPF are mostly the same as those for RF: number of trees, and number of variables used in each node partition, with the addition of λ when PDA is used as the index.

Figure 10 (left) shows the effect of proportion of variables for the benchmark data

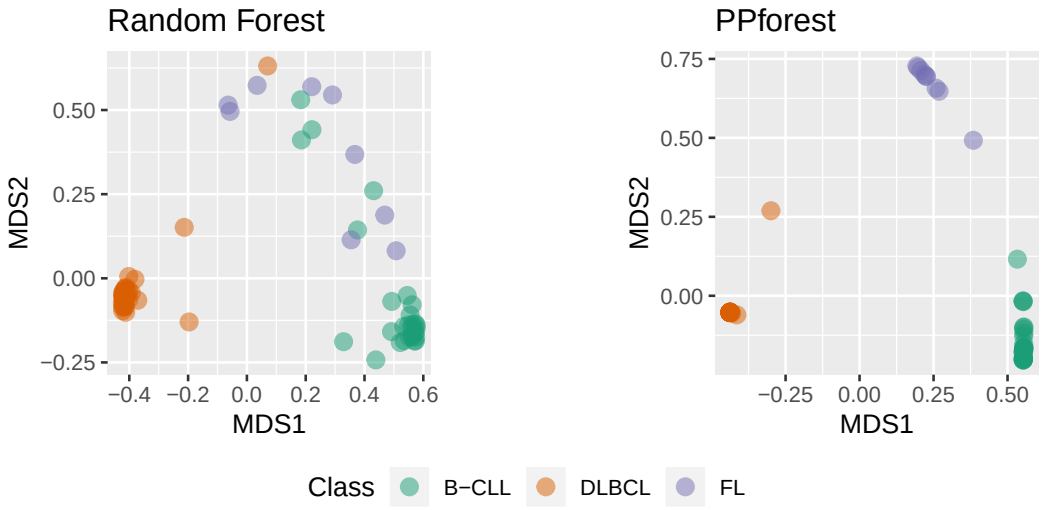

Figure 9: Examining similarity between cases, using pairwise plots of multidimensional scaling on the proximity matrix from RF and PPF fits of the lymphoma data. It can be seen that most cases are grouped closely with their class in PPF while in RF FL and B-CLL are mixed.

comparison. The average error over 200 training/test splits is shown. For several data sets error is lower when the more variables are used. Several converge to low error rate when half the variables are included. In some cases, leukemia and NCI60, the error increases when more variables are included. These two examples are where the cases variables ratio is smaller among the benchmark data set. This suggest that in small n big p problem increase the number of variables will be not the best strategy. In Section 6 we will show an example to illustrate this point.

The right plot compares the number of trees needed to optimize the OOB error for both PPF and RF on the lymphoma data. Both need around 100 trees to produce best performance.

6 Application: RNA-seq gene expression

We applied the proposed method to analyze an RNA-seq gene expression data set. The data are high-dimensional, with the number of observations, n is very small relative to the number of features, p. Traditional classification methods like linear discriminant analysis

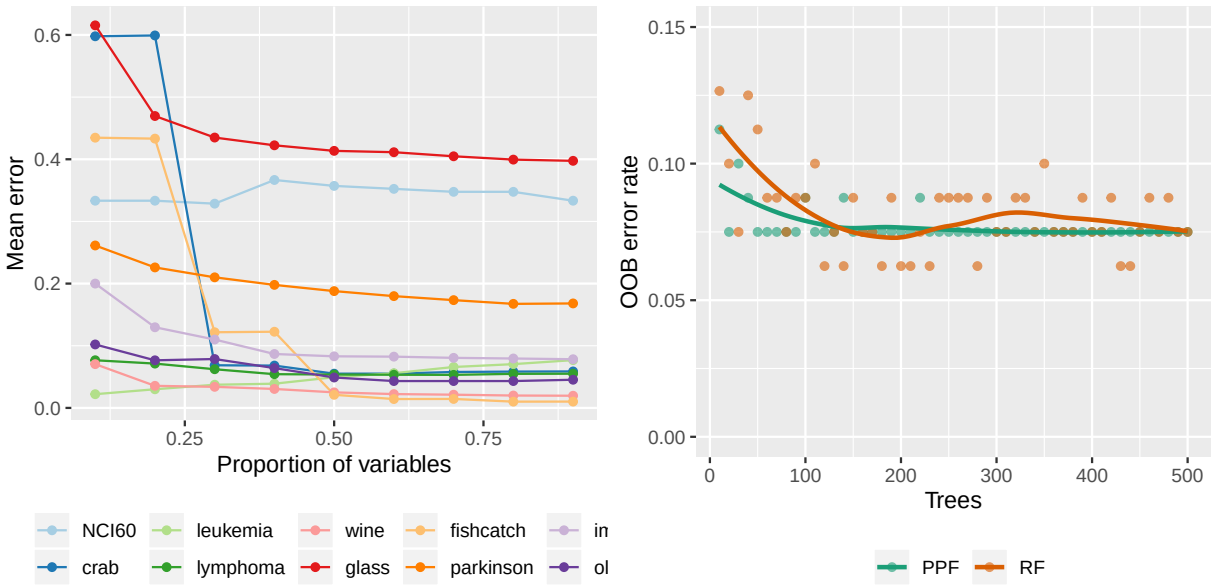

Figure 10: Illustrating model tuning using error rate reduction. The average error rate plotted against proportion of variables in all the benchmark data is shown at left. The error rate tends to be better with more variables, but it does vary substantially by data set. OOB error is plotted against number of trees (right) on the lymphoma data for both PPF and RF. PPF has the consistently lower error, but both would indicate about 100 trees is sufficient to get the best results.

or logistic regression cannot be applied directly in this context. PPforest can be applied for small n big p problem using PDA index.

There are four genotypes of maize, two inbred genotypes (B73 and Mo17) and its reciprocal hybrids (B73xMo17 and Mo17xB7), with just four replicates of each. The details of the experiment are described in Paschold et al. (2012). A subset of 1078 genes are flagged as important for understanding the so-called gene heterosis, in at least one of the two hybrids. In plant breeding, heterosis appears when the hybrid is deferentially expressed with respect to its parents.

We set up the classification problem as in Datta and Nettleton (2014), where the response variable is the genotype (four classes, B73, Mo17, B73xMo17 and Mo17xB7) and the feature variables are the expression of the 1078 genes. The goal is to identify important genes to distinguish among genotypes. The subset is formed with genes that are already found to be relevant, for distinguishing between inbred genotypes and its reciprocal hy-

brids, because choosing genes with smallest p-values can be misleading (Cook et al., 2007). To compare the predictive performance among different methods we will use leave-one out cross validation since we have a limited amount of data (16 observations).

Table 3: Comparison of CART, PPF, PPtree, and RF results with maize RNA-seq gene exression data set. The mean of training and test error rates from leave-one-out cross validation is shown. PPF.03 performs favorably compared to the other methods.

Method	Training	Test
CART	0.733	1.000
PPF.4	0.000	0.062
PPF.03	0.000	0.000
PPtree	0.000	0.062
RF	0.000	0.250

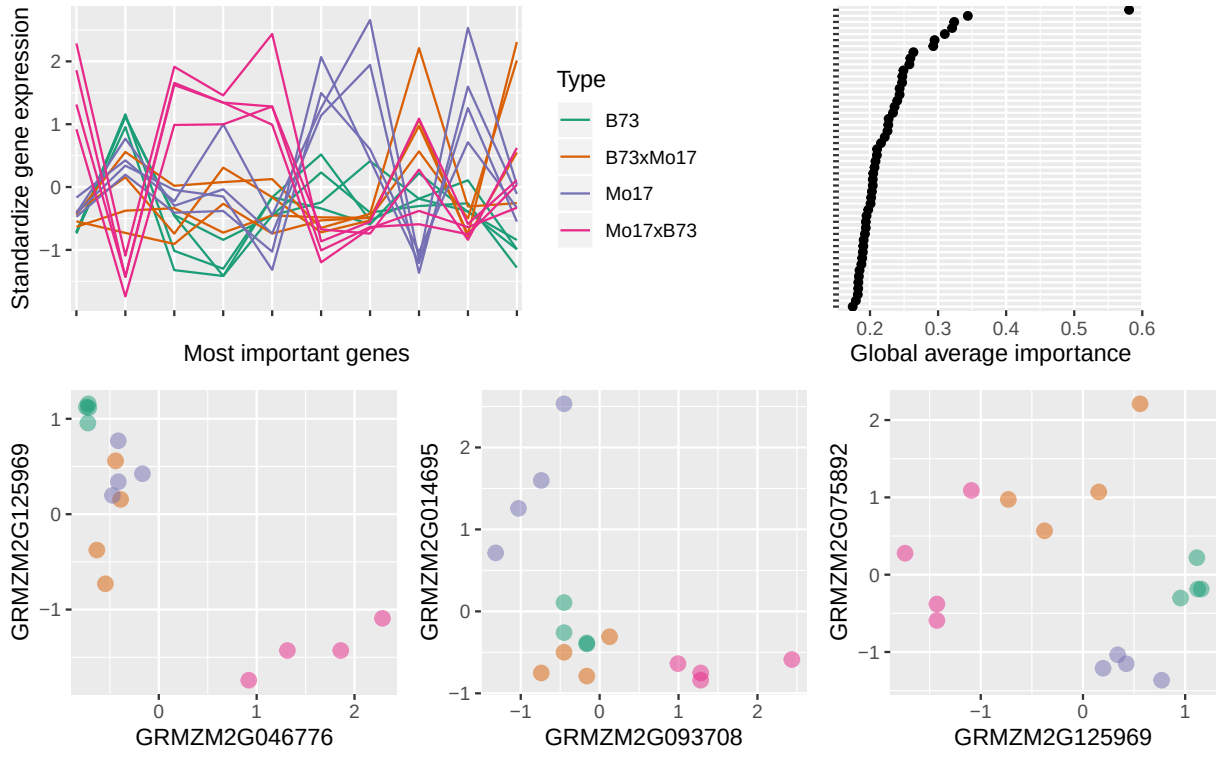

Figure 11: Overview of the PPF results for the maize RNA-seq data. The parallel coordinate plot (top left) shows the expression values for 10 most important genes. A dotplot shows the global average importance for the top 200 genes (top right). The three scatterplots show pairs of important variables (bottom row), indicating clear separation between classes.

Table 3 presents mean of training and test error rates from leave-one-out cross valida-

tion comparing CART, PPF, PPtree and RF. For PPF two cases are presented based on different proportions of variables considered for each node partition: PPF.4 use 40% of the variables while PPF.03 use 3% of them. PPF.03, with 32 variables in each node partition, is comparable with default for RF. While PPF with 40% of variables and PPtree results are the same in test error (6.2%), PPF performance is better when a smaller proportion of variables (3%) is used (0% of test error). RF presents a test error of 25% using the same number of variables than PPF.03.

Figure 11 shows the most important genes analysed based on global average importance measure for the best model (PPF.03). The importance measure is clearly bigger for one gene (GRMZM2G046776) and there is a breaking point in the importance measure for the first seven genes. The parallel coordinate plot shows the expression values for the top 10 variables, ordered from left to right. In the left scatterplot, of first two important variables, Mo17xB73 can be clearly separated from the other genotypes. The other scatterplots show various separations between the groups.

With 1078 variables and only 16 observations, one might be suspicious about the results. To check this, the analysis was re-computed several times with permuted class labels, where the class label is randomly re-assigned to a different sample. This removes any real class structure, while keeping the multivariate distribution of the expression values fixed. The results showed no important variables, and little separation between classes. The results from PPF are substantially better.

7 Discussion

This article has presented a new ensemble method (PPF) for classification problems, that is built on an oblique tree classifier (PPtree). PPF effectively uses linear combinations of variables, incorporating the correlation between variables to build better boundaries between classes. It provides a range of diagnostics for variable importance, confusion of observations between groups and proximity of observations. PPF provides consistently low error rates, and retains the interpretability afforded by tree-based algorithms.

The benchmark data study showed that PPF's predictive performance is as good, or

better than, CART and PPtree, and can outperform RF for some problems. PPF performs better than RF when the classes are separated by a linear combination of variables and when the correlation between variables increases. This can be seen by comparing the variable importance diagnostics of PPF and RF, showing that different variables are combined to create the classification model.

There are several directions where the work could be extended. The two projection pursuit indexes, LDA and PDA, can be readily supplemented by other indices, or by adapting a linear SVM. For example, using a regression index could extend the approach to a continuous response. Another direction is to adapt the PPtree algorithm to allow more than $g - 1$ splits. This PPtree constraint protects the single tree model from overfitting. With PPF, there is some protection against this from bagging, and we expect it would enable deeper non-linear boundaries to be constructed. Lastly, because the accuracy of each tree is collected, automatic pruning of poor performing trees is a possibility.

8 Acknowledgements

This paper was written with the R packages knitr (Xie (2015)), ggplot2 (Wickham (July 2009)) and dplyr (Wickham et al. (2015)), and the files to reproduce the paper and results is available at `https://github.com/natydasilva/PPforestpaper`.

The authors are grateful for the helpful reviews provided by the editor, associate editor and two anonymous reviewers.

References

Amit, Y. and Geman, D. (1997), "Shape quantization and recognition with randomized trees," *Neural computation*, 9, 1545–1588.

Athreya, K. B. and Lahiri, S. N. (2006), *Measure theory and probability theory*, Springer Science & Business Media.

Biau, G., Devroye, L., and Lugosi, G. (2008), "Consistency of Random Forests and Other

Averaging Classifiers," *J. Mach. Learn. Res.*, 9, 2015–2033, URL http://dl.acm.org/citation.cfm?id=1390681.1442799.

Breiman, L. (1996a), "Bagging predictors," *Machine learning*, 24, 123–140.

— (1996b), "Heuristics of instability and stabilization in model selection," *The annals of statistics*, 24, 2350–2383.

— (2001), "Random forests," *Machine learning*, 45, 5–32.

Brodley, C. E. and Utgoff, P. E. (1995), "Multivariate decision trees," *Machine learning*, 19, 45–77.

Chang, W. and Luraschi, J. (2016), "profvis: Interactive Visualizations for Profiling R Code," Version 0.3.2.

Cook, D., Hofmann, H., Lee, E.-K., Yang, H., Nikolau, B., and Wurtele, E. (2007), "Exploring gene expression data, using plots," *Journal of Data Science*, 5, 151.

Cortes, C. and Vapnik, V. (1995), "Support-vector networks," *Machine Learning*, 20, 273–297, URL https://doi.org/10.1007/BF00994018.

da Silva, N., Lee, E.-K., and Cook, D. (2018), *PPforest: Projection Pursuit Classification Forest*, URL https://CRAN.R-project.org/package=PPforest. R package version 0.1.1.

Datta, S. and Nettleton, D. (2014), *Statistical analysis of next generation sequencing data*, volume 15, Springer.

Do, T.-N., Lenca, P., Lallich, S., and Pham, N.-K. (2010), "Classifying very-high-dimensional data with random forests of oblique decision trees," in *Advances in knowledge discovery and management*, Springer, 39–55.

Eddelbuettel, D., François, R., Allaire, J., Chambers, J., Bates, D., and Ushey, K. (2011), "Rcpp: Seamless R and C++ integration," *Journal of Statistical Software*, 40, 1–18.

Friedman, J. H. and Tukey, J. W. (1974), "A projection pursuit algorithm for exploratory data analysis," *IEEE Transactions on computers*, 100, 881–890.

Ho, T. K. (1998), "The random subspace method for constructing decision forests," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20, 832–844.

Huber, P. (1990), "Data analysis and projection pursuit," in *Technical report PJH-90-1*, Dept. of Mathematics, Massachusetts Institute of Technology Cambridge, MA.

Kim, H. and Loh, W.-Y. (2001), "Classification trees with unbiased multiway splits," *Journal of the American Statistical Association*, 96.

Kruskal, J. B. (1969), "Toward a practical method which helps uncover the structure of a set of multivariate observations by finding the linear transformation which optimizes a new 'index of condensation'," in *Statistical Computation*, Academic Press, New York.

Lee, E.-K. (2018), "PPtreeViz: An R Package for Visualizing Projection Pursuit Classification Trees," *Journal of Statistical Software*, 83, 1–30.

Lee, E.-K. and Cook, D. (2010), "A projection pursuit index for large p small n data," *Statistics and Computing*, 20, 381–392.

Lee, E.-K., Cook, D., Klinke, S., and Lumley, T. (2005), "Projection pursuit for exploratory supervised classification," *Journal of Computational and Graphical Statistics*, 14.

Lee, Y. D., Cook, D., Park, J.-w., and Lee, E.-K. (2013), "PPtree: Projection pursuit classification tree," *Electronic Journal of Statistics*, 7, 1369–1386.

Lichman, M. (2013), "UCI Machine Learning Repository," URL `http://archive.ics.uci.edu/ml`.

Menze, B. H., Kelm, B. M., Splitthoff, D. N., Koethe, U., and Hamprecht, F. A. (2011), "On oblique random forests," in *Machine Learning and Knowledge Discovery in Databases*, Springer, 453–469.

Paschold, A., Jia, Y., Marcon, C., Lund, S., Larson, N. B., Yeh, C.-T., Ossowski, S., Lanz, C., Nettleton, D., Schnable, P. S., et al. (2012), "Complementation contributes to transcriptome complexity in maize (Zea mays L.) hybrids relative to their inbred parents," *Genome research*, 22, 2445–2454.

R Core Team (2019), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, URL `https://www.R-project.org/`.

Tan, P. J. and Dowe, D. L. (2004), "MML inference of oblique decision trees," in *AI 2004: Advances in Artificial Intelligence*, Springer, 1082–1088.

— (2006), "Decision forests with oblique decision trees," in *MICAI 2006: Advances in Artificial Intelligence*, Springer, 593–603.

Therneau, T. M., Atkinson, B., and Ripley, M. B. (2010), "The rpart package," .

Truong, A. (2009), "Fast growing and interpretable oblique trees via probabilistic models," *Univ. of Oxford, A thesis submitted for the degree of Doctor of Philosophy, Trinity term*, 1–119.

Wickham, H. (July 2009), *ggplot2: Elegant graphics for data analysis*, useR, Springer.

Wickham, H., Francois, R., and Rstudio (2015), "`dplyr`: A Grammar of data manipulation," `http://cran.r-project.org/web/packages/dplyr/index.html`. Maintained by Wickham, H.

Xie, Y. (2015), *Dynamic documents with R and knitr*, Boca Raton, Florida: Chapman and Hall/CRC, 2nd edition, URL `http://yihui.name/knitr/`. ISBN 978-1498716963.