Bagged projection methods for supervised classification in big data

Natalia da Silva
Iowa State University

December 5, 2015

Contents

Chapter 1

Introduction

Classification problems are important for a wide variety of applications, genetics, forensics, meteorology, among others. For example, using genetic information we can classify patients with certain diseases and identify which are the key gens to classify each disease. Firearm identification is a relevant problem in forensics, then using information from the interior of the barrel, parts of the action and ammunition components it is possible to identify the firearm from a scene. In meteorology we can predict if will rain based on atmospheric conditions. There are a lot of different approaches to get class predictions but not all the methods work well for all kind of data structure. A new classification method will be presented in this project with the objective to improve the predictive performance when the variables are highly correlated and the separation between classes occur on combinations of variables.

1.1 Scope

This thesis research develops a new algorithm based on bagged trees for classification problems. Classification algorithms, like classification and regression trees (CART), are unstable because the model can vary substantially from one sample to another. Using bootstrap aggregated trees provides a more reliable classifier, and one that better predicts new data. In Breiman (2001) two random forest methods were presented, one with trees calculated using axis-parallel partitions and other using trees with oblique partitions at random orientations. The second approach was not as successful as the first, but interest in this approach has peaked in recent years. The method presented here is closer to the second original idea, and it is called projection pursuit classification random forest.

The trees used in our ensemble, find the best split using linear combination of variables instead of only one variable for each split. The main difference with the second random forest approach is that the oblique partitions are not selected at random, the linear combination is computed by optimizing a projection pursuit index, to get 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, and can outperform the basic forest when separations between groups occur on combinations of variables.

To analyze the performance of the new approach simulation study and real data were tested. Comparison with other classification method were done for simulated and real data. This new method is implemented in a R package that is available in github.

1.2 Proposed research

Table 1.1: Project structure description

Chapter 1	Overview and scope.
Chapter 2	Literature review of tree and forest classifiers.
Chapter 3	Draft paper describing new projection pursuit random forest method. The methodology is based on the previous work of Lee et. al. (2013). The method is implemented in a new R package, called PPforest, available on github. The algorithm incorporates methods to determine OOB error, variable importance and proximity measures.
Chapter 4	Initial ideas for visualization related to a PPforest object are described, that help to understand class structure in the data, and to diagnose the forest fit. This work needs further development.
Chapter 5	In our implementation of the algorithm, thus far, several modifications are suggested which would enable it to be more useful in practice issues. These are described here and will be developed more completely during the latter part of the thesis research. Forests are relatively efficient for work with large p types of big data, because they effectively reduce the dimensionality. Once projections are used, the flexibility to cope with correlation between variables is improved by the speed of the computation is reduced. There are two types of weights which are useful for classification: weighted classes, to account for unbalanced data and cost functions, and weighted classes which will enable tackling data containing survey weights, and also better enable boosting. The PPtree algorithm upon which the PPforest is built handles multiclass problems in just one way. Some simple additional ways may make it more effective in a broader set of problems. It may also be possible to build projection trees using a boosted algorithm to obtain multiple linear projections to build nonlinear boundaries. The original forest can be used with a continuous response, and for unsupervised classification, which should also be achieved with a PPforest by extending the PPtree algorithm to handle continuous responses, or incorporating a different projection pursuit function for unsupervised learning.
Chapter 6	Time table for completion.

Chapter 2

Literature review

Supervised and unsupervised learning are two important methods in statistical learning. The main objective of supervised learning is to predict the value of a response variable Y for a given set of predictor variables $X^T = (X_1, \ldots X_p)$. On the other hand, in unsupervised learning there is not information about the response variable Y and the idea is to make inference about the density using only information from the predictor variables. When the objective is to predict a categorical variable then the supervised learning method is called classification problem while it is a regression problem when a quantitative outcome is predicted.

Classification problems can be address in different ways; we can use linear methods like linear regression, discriminant analysis, separating hyperplanes, etc. Additional approaches maybe based on kernel-smoothing methods, like kernel density estimation, mixture models for classification, etc. This project is focused on classification problems using bagged trees methods. In the next subsections a literature review in trees an random forest is presented.

2.1 Classification and regression trees

Classification and regression trees are two supervised learning methods that have been used for a long time to solve a wide variety of problems. Classification trees are used when the objective is to predict a qualitative variable while is called regression when a quantitative outcome is predicted. These two techniques are not new, the first regression tree algorithm, was published in 1963 (Morgan and Sonquist (1963)). Automatic Interaction Detection (AID) is the name of this regression tree algorithm.

After AID many other tree algorithms were developed across the years, CART Breiman et al. (1984), CHAID Kass (1980), C4.5 Quinlan (1993), FACT Loh and Vanichsetakul (1988), QUEST Loh and Shih (1997), CRUISE Kim and Loh (2001), GUIDE Loh (2009), CTREE Hothorn et al. (2006) and many more. One key point differentiate some of these methods is the node spliting. Some of the methods use kernel density, nearest neighbor or linear splits on subset of variables in the node partition.

Decision trees can be grouped based on the number of predictor variables used in each node partition. Trees that use one variable at a node partition produce axis-parallel splits. While trees that test multiple feature variables at every node, can produce oblique splits and are characterize to be smaller than axis-parallel ones.

One of the main attractive of classification trees is the simplicity to get the predictions. The most extended trees use binary partitioning with axis-parallel splits, like CART. These kinds of trees uses only one variable in each split and then define hyperplanes that are orthogonal to the axis. In this thesis we will work with classification trees which define hyperplanes that are oblique to the axis. More specifically, projection pursuit classification tress (PPtree) algorithm will be used and these trees will be the base learner for a random forest approach.

Tree model

A tree can be seen as a set of decisions rules that define recursive partitions of the feature space. The expected values for the response variable Y can be defined as follows:

$$E(Y/X = x) = \sum_{s=1}^{S} c_s I_{R_s}(x) \tag{2.1}$$

Where R_s represents a partition in the feature space such that $R_s \in R$, with $R = \cup_{i=1}^{S} R_i$ and the intersection of two partitions in R are exclusive, $R_s \cap R_i = \emptyset$. If $x \in R_s$ then the predicted value for Y is c_s.

$$I_{R_s}(x) = \begin{cases} 1 & \text{if} \quad x \in R_s \\ 0 & \text{if} \quad x \notin R_s \end{cases}$$

c_s is computed differently if the problem is a classification or a regression problem.

For classification problems $Y_i \in \{1, 2 \ldots K\}$ denotes the class of each observation. Here the expected value for Y when $X_i \in R_s$ is the most frequent class in the partition R_s, i.e.

$$c_s = \arg\max_k \left\{ \frac{\#(Y = k)}{\#R_s} \right\} \quad X_i \in R_s$$

For regression problem the predicted value is :

$$c_s = \frac{1}{\#R_s} \sum_{i/X_i \in R_s} Y_i$$

Finally for a given data set $\{Y_i, X_{1i,} X_{2i} \ldots X_{pi}\}_{i=1}^n$ the predicted values for the response variable Y can be defined as $\hat{f}(x) = \sum_{s=1}^S \hat{c}_s I_{\hat{R}_s}(x)$ One thing that distinguish a single decision tree algorithm is the way that the regions, R_s, are estimated.

2.2 Axis-parallel trees, CART

For axis parallel trees CART will be describe. Classification and regression tress (CART) Breiman et al. (1984) is an important algorithm because was the first decision tree described with analytically rigor.

Given a training data of the form $\Theta = (X, Y)$, where Y is the response variables and $X^T = (X_1, \ldots X_p)$ the predictor variables. The main objective in CART is to predict the values of the response using the information form X. The response and the feature variable can be quantitative or categorical variables. In a classification problem $Y \in \{1, 2, \ldots K\}$ and the objective is to classify subjects in some of the K classes using information from the feature variables. If the response $Y \in \mathbb{R}$ is quantitative variable then CART has the same objective than a linear model, is to predict the numerical value of Y.

The CART decision tree produces binary recursive partitioning procedure by considering axis-parallel splits. This method split the feature space in rectangles using only one feature variable in each node split. Growing a tree consists in beginning from a root node and split the data in two children subnodes. The main idea of node splitting is to get each children as pure as possible based on some impurity measure. Each children node is splited again and this process stop when every distinct observation is in the training set has its own rectangle.

Figure 2.1 shows an example of classification tree with three classes and two feature variables base on simulated data. Data are simulated from three bivariate normal distribution with the following variance-covarance structure with $\rho_1 = 0.2$, $\rho_2 = -0.35$ and $\rho_3 = 0.25$: $\Sigma_i = \begin{pmatrix} 1 & \rho_i \\ \rho_i & 1 \end{pmatrix}$, and different mean $\mu_1 = \begin{pmatrix} -5 \\ -0.8 \end{pmatrix}$, $\mu_2 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$ and $\mu_3 = \begin{pmatrix} 0 \\ -4 \end{pmatrix}$.

In the tree diagram we can see the values where the node was splited and the order of the different partitions. In the first split X_1 was used, and if and observation has $X_1 < -2.54$ follow the left branch and otherwise follows the right branch. In the second partition X_2 was used and a similar procedure was applied. In this example six terminal nodes were defined and the regions associated to them were:

$R_1 = \{X_1 < -2.54\}$, $R_2 = \{X_1 \geq -2.54, X_2 < -2.865\}$, $R_3 = \{-2.54 \leq X_1 < 1.685, -2.865 \leq X_2 < -2.4\}$, $R_4 = \{X_1 \geq 1.685, -2.865 \leq X_2 < -2.4\}$, $R_5 = \{X_1 \geq -2.54, X_2 \geq -2.455\}$ and $R_6 = \{X_1 \geq 1.465, X_2 \geq -2.455\}$

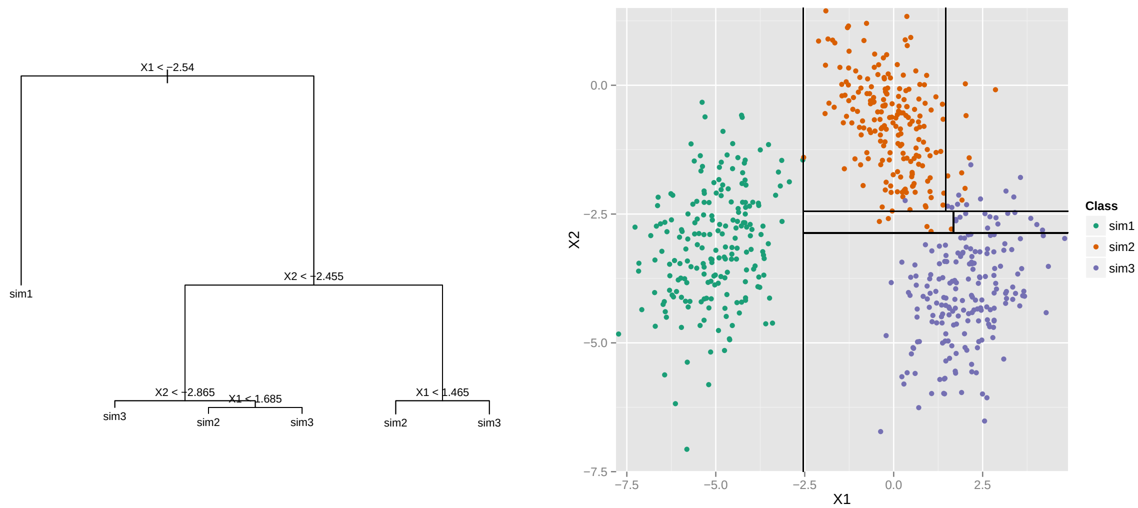

Figure 2.1: Classification tree model example with simulated data. In the left panel a decision tree with six terminal nodes and 5 splits are shown. Right panel shows a scatterplot with the simulated data and the partition of $\mathbb{R}^2$ into six regions corresponding to the six terminal nodes

Tree construction

The construction of the optimal tree needs two basic steps. First, a "maximal tree" is grown using the training data $\Theta = (X, Y)$. A set of partitions are used and in a simple way these partitions can be thought as a set of questions with binary response. $x \in Q$? where Q is a subset of the sample and the partition is created based on one variable. If X_i is a continuous variables the test will have the form $X_i < c$ vs $X_i \geq c$ for some constant $c \in \mathbb{R}$, in case X_i is categorical the test rule will be define as $X_i \in H$ vs $X_i \notin H$ for some subset $H \subset \{h_1 \dots h_{|X_i|}\}$ of the factors of X_i. For these two options in each node the test rule is checked and if the rule is true $(X_i < c \;\; or \;\; X_i \in H)$ the brunch follows to the left and if the condition is false the brunch follows the right side.

All the partitions are evaluated and the best partition is selected based on some impurity measure of the node. The total number of possible splits when the predictor variable is categorical with K categories is $2^{K-1} - 1$ while if X is continuous or ordinal with L different values, $L - 1$ splits on X can be defined. After the best partition is selected, the initial data set is divided in two subsets and within each subset the same procedure is repeated. The second step consists in pruning the maximal tree to get the optimal tree. Instead of using a stopping rule the tree grows as large as possible and then the tree is pruned back to the root based on the lowest cross-validation estimation error which defines the place where the tree is pruned. Basically the next split to be pruned is the node which has the smaller contribution in the overall tree performance.

Impurity measures

For every node t a set of decision rules are defined and the best rule s is selected using a node impurity measure $I(t)$. This impurity measure of a node is associate with the heterogeneity of the dependent variable in this node. The way in which the heterogeneity is measure depends if the tree is a classification tree or a regression tree. In the first case we have to take into account the characteristics of the qualitative variable while in the regression type the heterogeneity is given by the distribution of the continuous variable. For each rule s is defined $\phi(s,t) = I(t) - I(t_r) - I(t_l)$ which represents the impurity reduction when a rule s is used to divided t. Finally the selected rule is the rule which maximizes $\phi(s,t)$, this is $s^* = \arg\max_s\{\phi(s,t)\}$. The optimization is done considering all the variables, s^* is the best partition from all possible partitions.

Advantages & disadvantages

Some of the advantages we can mention about CART are the variable selection can be done automatically and the importance measure is given in a natural way. Quantitative and categorical variables can be used as dependent and independent variables. Are invariant to monotone transformations of the quantitative variables. Works with missing data. It is easy to interpret and fast to implement. One disadvantage is that these trees are unstable, also the there are some works that shows the induction of bias in the variable selection. Finally these trees make the separation only using one variable in each node partition then doesn't work well when the data can be separable with linear combinations.

2.3 Oblique splits trees, PPtree

One of the limitations of trees like CART is that the nodes can only separate the data with hyperplanes orthogonal to the feature axis. Oblique trees uses discriminant functions in each node with more than one variable, then the defined hyperplanes are oblique to the axes (polygonal partitioning of the feature space). Oblique trees tend to be more interpretable than axis-parallel trees. Two kinds of oblique trees can be defined based if they use all the feature variables or only some of them, then full or concise oblique trees can be defined.

To describe oblique split trees I will focus on projection pursuit classification tree (PPtree) Lee et al. (2013), these trees are of interest in this oblique trees review because they are the basic learning of the bagging classification method purposed in this work. PPtree optimize a projection pursuit index to find low-dimensional projections to separate classes.

PPtree method is defined for classification problems where the response variable is categorical and the method is define to use quantitative feature variables.

One important characteristic of PPtree is that treats the data always as a two-class system, when the classes are more than two the algorithm uses a two step projection pursuits optimization in every node split. In the first step optimize a projection pursuit index to redefine the problem in a two class problem and the second step is to find an optimal one dimensional projection to separate the two class problem. Base on this process to grow the tree, the depth of PPtree is at most the number of classes. PPtree uses binary partitioning test, if X_i is a continuous variables the test will have the form $\sum_{i=1}^{p} \alpha_i X_i < c$ vs $\sum_{i=1}^{p} \alpha_i X_i \geq c$ for some constant $c \in \mathbb{R}$ and the coefficients $\alpha_i \in \mathbb{R}$. In each node the test rule is checked and if the rule is true ($\sum_{i=1}^{p} \alpha_i X_i < c$) the brunch follows to the left and if the condition is false the brunch follows the right side.

Figure 2.2 shows an example of classification projection pursuit tree with three classes and two feature variables base on the same simulated data described before.

In this simple example the two feature variables are linearly combined to do the partitions en each split. In the first split if an observation has $0.91X_1 + 0.40X_2 < -2.90$ follow the

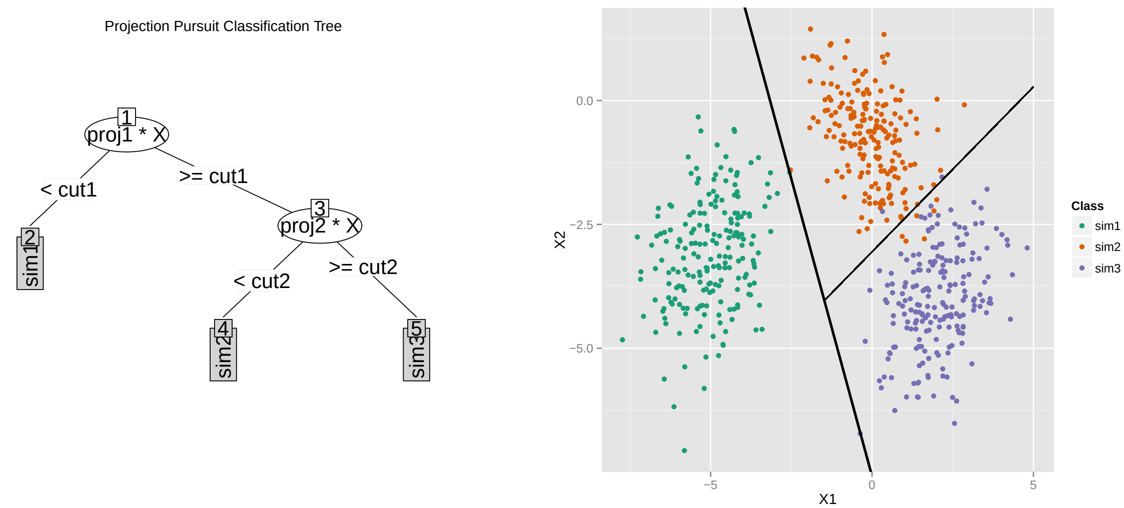

Figure 2.2: Projection pursuit classification tree model example with simulated data. In the left panel a decision tree with tree terminal nodes and 2 splits is shown. Right panel shows a scatterplot with the simulated data and the partition of $\mathbb{R}^2$ into six regions corresponding to the six terminal nodes

left branch and if no, follow the right branch. In the second partition if an observation has $0.46X_1 - 0.88X_2 < 2.61$ follows the left branch and if no, follows the right branch. In this example three terminal nodes were defined and the regions associated to them were:

$R_1 = \{0.91X_1 + 0.40X_2 < -2.90\}$, $R_2 = \{0.91X_1 + 0.40X_2 \geq -2.90, 0.46X_1 - 0.88X_2 < 2.61\}$, and $R_3 = \{0.91X_1 + 0.40X_2 \geq -2.90, 0.46X_1 - 0.88X_2 \geq 2.61\}$

Tree construction

Using the training data $\Theta = (X, Y)$ a full (maximal) tree is grown. As in CART we can think in the partitions as set of questions with binary response. $x \in Q$? where Q is a subset of the sample and the partition is created based on one a linear combination of variables. In the PPtree construction the data are treat always as a two-class system, when the classes are more than two the algorithm uses a two step projection pursuits optimization in every node split. In the first step, an optimal one-dimension projection α^* is found, for separating all classes in the current data. All the data are projected in α^* and comparing means the classes are reduced to two classes. A new variable y_i^* is defined by assigning a new label "$G1$" or "$G2$" to each observation. The new groups "G1" and "G2" can contain more than one original classes. Then a second projection pursuit optimization is done using these new group labels, $G1$ and $G2$, to finding the optimal one dimension projection, α, using (X_i, y_i^*). This step is to find the best separation between "G1" and "G2", if $\sum_{i=1}^{p} \alpha_i M1 < c$ then assign "G1" to the left node else assign "G2" to the right node, where $M1$ is the mean of "G1". This procedure is repeated in each node until there is one class in each node from the original classes.

Impurity measure

Let $(p_1, p_2 \ldots p_K)$ the probabilities for each class in the training data. Let $Z_i = \alpha^T X_i$ where α is a p-dimensional projection in a 1-dimensional space. The projected data were examined in two ordered groups $Z_{(1)} \ldots Z_{(i)}$ and $Z_{(i+1)} \ldots Z_{(n)}$, and the class probabilities for each group were defined for a given i like $(p_{L_i,1} \ldots p_{L_i,g})$ and $(p_{R_i,1} \ldots p_{R_i,g})$ where and p_{L_i} and p_{R_i} be the proportion of each group. To measure the impurity of each group the class probability measure was used and IM_{L_i} (impurity measure for left group) and IM_{R_i} (impurity measure for right) can be calculated. To find the best split i^* for the projected data $\{Z_{(1)} \ldots Z_{(i^*)}\}$ and $\{Z_{(i^*+1)} \ldots Z_{(n)}\}$ we have to minimized the weighted mean of the impurity measures, $IM_i = p_{L_i} IM_{L_i} + p_{R_i} IM_{R_i}$ and get IM_i^*. Then in the tree construction in each node all the partitions are evaluated and the best partition is selected based on the impurity measure IM_i (Lee et al., 2005).

Advantages & disadvantages

PPtree has a simpler structure than other tree methods like CART. The number of classes will be the same as the number of final nodes, so the depth of the tree is at most $K - 1$ where K is the number of classes. PPtree does not need to be pruned as CART does. In PPtree the correlation between the original variables is took into account in the tree construction. At each node, the PPtree separates two classes using a linear combination and *if a linear boundary exists* in the data, PPtree produces a tree without misclassification. Another interesting characteristic, is this method can be used for variable selection since each projection coefficient of each node represents the importance variable to separate classes in each node.

2.4 Random forest

A random forest is an ensemble learning method, built on bagged trees developed by Breiman (2001). There are two main concept used in random forest, bootstrap aggregation (Breiman (1996) and Breiman et al. (1996)) and random feature selection (Amit and Geman (1997) and Ho (1998)) to individual classification or regression trees for prediction. Bootstrap samples from training test and random feature selection in each split are the two ways in which random forest incorporate randomness in the model. The most used random forest implementation uses CART or C4.5 trees as individual learners, these trees generate partitions that use only one feature variable generating boundaries with box structure.

Let $\Theta = (X, Y)$ the training set of size N, B bootstrap samples with size N are extracted. For each bootstrap sample a tree is grown using random variable selection in each node and the trees are not pruned. In this process as in bagging the variance is reduced due to the aggregation and the bias because the trees are fully grown. Additionally to bagging the trees correlation in random forest is reduced because the random feature selection in each split. The number of selected variables to use in each node split should be much smaller than the total number of variables. In the case of classification they recommend $\sqrt(m)$ in classification problems and $\frac{m}{3}$ in regression problems. In each node the best split based on the selected variables is done. Final predictions are obtained by aggregating the results from the trees, if the problem is classification the final result is based on majority vote while if the problem is regression the prediction is base on average over the trees. A formal definition of random forest from Breiman (2001) is:

Definition: *A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{h(x, \Theta_k), k = 1, \ldots\}$ *where* $\{\Theta_k\}$ *are independent, identically distributed random vectors* *and each tree cast a unit vote for the most popular class at input x.*

Where Θ_k is a random vector where each Θ_k is independent from $\Theta_1, \ldots, \Theta_{k-1}$ but with the same distribution. $h(x, \Theta_k)$ is a single decision tree. The random forest error rate depends on the correlation between the trees in the forest and the strength of each individual tree in the forest (Breiman, 2001). The expected values for the response variable Y in random forest can be defined as follows:

For classification

$$E(Y/X = x) = \arg\max_k \sum_{b=1}^{B} I[E_b(Y/X = x) = k] \tag{2.2}$$

For regression

$$E(Y/X = x) = \frac{1}{B} \sum_{b=1}^{B} E_b(Y/X = x) \tag{2.3}$$

Chapter 3

Projection pursuit classification random forest

Draft of a paper to be submitted to JCGS

273

Projection pursuit classification random forest

October 21, 2015

Abstract

A random forest is an ensemble learning method, built on bagged trees. The bagging provides power for classification because it yields information about variable importance, predictive error and proximity of observations. This research adapts the random forest to utilize combinations of variables in the tree construction, which we call the projection pursuit classification random forest (PPforest). In a random forest each split is based on a single variable, chosen from a subset of predictors. In the PPforest, each split is based on a linear combination of randomly chosen variables. The linear combination is computed by optimizing a projection pursuit index, to get a projection of the variables that best separates the classes. The PPforest uses the PPtree algorithm Lee et al. (2013), which fits a single tree to the data. Utilizing linear combinations of variables to separate classes takes the correlation between variables into account, and can outperform the basic forest when separations between groups occur on combinations of variables. Two projection pursuit indexes, LDA and PDA, are used for PPforest. The methods are implemented into an R package, called PPforest, which is available on `https://github.com/natydasilva/PPforest`.

1 Introduction

The most common random forest implementations uses univariate decision trees like CART or C4.5. These kinds of trees uses only one variable in each split and then define hyperplanes that are orthogonal to the axis. Sometimes we have data where the class can be separated by linear combinations and in these cases use a classifier which define hyperplanes that are oblique to the axis maybe do a better job.

This paper describes a random forest to utilize combinations of variables in the tree construction, which we call the projection pursuit classification random forest (PPforest). For each split a random sample of variables is selected and a linear combination is computed by optimizing a projection pursuit index, to get a projection of the variables that best separates the classes.

Trees that use linear combinations of predictors in a split are known in the literature as oblique trees. There are different approaches in the literature about these kind of trees Kim and Loh (2001), Brodley and Utgoff (1995), Tan and Dowe (2005), Truong (2009) and Lee et al. (2013). All these trees look for linear combinations of predictors to use in a split and then the main difference between all these approaches is the way in which the best partition is selected in each node. Some of the oblique trees implementations use penalized least square, L2 regularization or linear support vector machines to find the optimal combination.

To illustrate the idea of oblique trees we will use an example based on simulated data. Data are simulated from three bivariate normal distributions with the same variance-covarance structure: $\Sigma = \begin{pmatrix} 1 & 0.95 \\ 0.95 & 1 \end{pmatrix}$, and different mean $\mu_1 = \begin{pmatrix} 1 \\ 0.6 \end{pmatrix}$, $\mu_2 = \begin{pmatrix} 0 \\ -0.6 \end{pmatrix}$ and $\mu_3 = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$. Each group has 100 samples, Figure 1 shows a scatterplot of the simulated data. We can observe that the data are strongly correlated and the classes can be separated by linear combinations.

The decision boundaries of a classifier are a representation of the model in the data space and can be very insightful to see how the model responds to the data. Figure 2 shows the

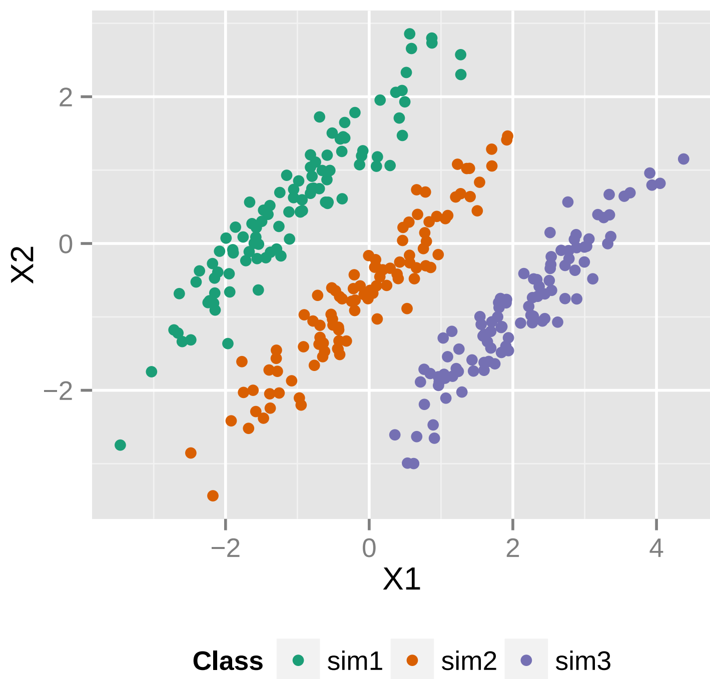

Figure 1: Scatterplot of simulated data for a three class problem

276

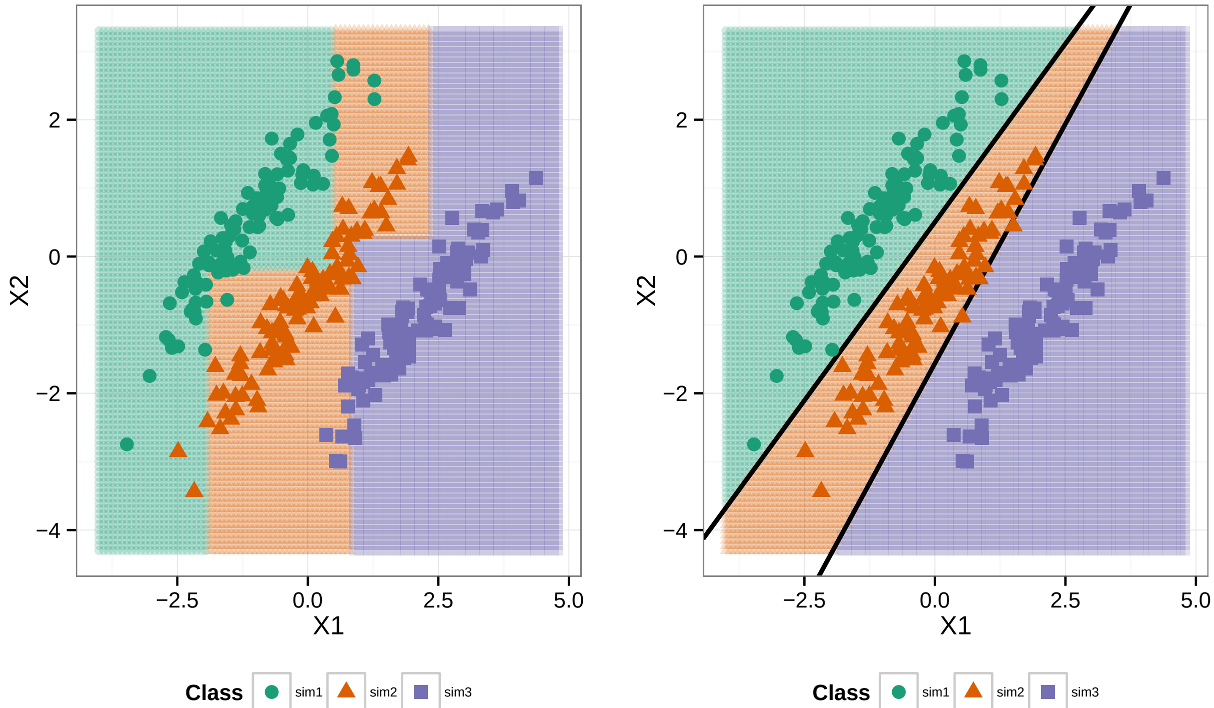

Figure 2: Decision boundaries for rpart and PPtree from simulated three class problem

decision boundaries for `rpart` and `PPtree` using the simulated data. In the left plot we can observe the decision boundaries defined by `rpart` and in the right side the decision boundaries defined by `PPtree.`We can see that `rpart` needs to do a lot of work to approach the data shape while the boundaries defined by `PPtree` follows the structure of our data.

It is important to mention that most of the oblique trees methods presented in the bibliography are not available to use or the code is not open source like Kim and Loh (2001).

Some random forest approaches that use oblique trees as base learners were found in the literature review Tan and Dowe (2006), Menze et al. (2011). There is a package in R obliqueRF that implements and oblique random forest but it is only for a two class problem.

The classification method presented here PPforest uses the PPtree algorithm implemented in R, which fits a single tree to the data. Utilizing linear combinations of variables to separate classes takes the correlation between variables into account, and can outperform the basic forest when separations between groups occur on combinations of variables. Two projection pursuit indexes, LDA and PDA, are used for PPforest.

2 Background

2.1 Random forest

Random forest is an ensemble learning method, built on bagged trees developed by Breiman (2001). There are two main concept used in random forest, bootstrap aggregation (Breiman (1996) and Breiman et al. (1996)) and random feature selection (Amit and Geman (1997) and Ho (1998)) to individual classification or regression trees for prediction.

Let $\Theta = (X, Y)$ the training set of size N, B bootstrap samples with size N are extracted. For each bootstrap sample a tree is grown using random variable selection in each node and the trees are not pruned. In this process as in bagging the variance is reduced due to the aggregation and the bias because the trees are fully grown. Because the random feature selection in each split, the trees correlation in random forest is reduced.

The number of selected variables to use in each node split (m) should be much smaller than the total number of variables (M). In the case of classification they recommend $\sqrt{(m)}$ in classification problems and $\frac{m}{3}$ in regression problems. In each node the best split based on the selected variables is done. Final predictions are obtained by aggregating the results from the trees, if the problem is classification the final result is based on majority vote while if the problem is regression the prediction is base on average over the trees.

A formal definition of random forest from Breiman (2001) is:

Definition: *A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{h(x, \Theta_k), k = 1, \ldots\}$ *where* Θ_k *are independent, identically distributed random vectors and each tree cast a unit vote for the most popular class at input* x*.*

Where Θ_k is a random vector where each Θ_k is independent from $\Theta_1, \ldots, \Theta_{k-1}$ but with the same distribution. $h(x, \Theta_k)$ is a single decision tree.

Algorithm description:

1. Let N the number of cases in the training set $\Theta = (X, Y)$, B bootstrap samples from the training set are taking (samples of size N with replacement)

2. For each bootstrap sample a tree is grown to the largest extent possible $h(x, \Theta_k)$. No pruning.

3. Let M the number of input variables, a number of $m << M$ variables are selected at random at each node and the best split based on these m variables is used to split the node.

Final predictions are obtained by aggregating the results from the trees, if the problem is classification the final result is based on majority vote while if the problem is regression the prediction is base on average over the trees.

The random forest error rate depends on the correlation between the trees in the forest and the strength of each individual tree in the forest (Breiman, 2001).

2.2 Projection Pursuit classification tree

Friedman and Tukey (1973) is the first work which use the term "projection pursuit" they present this method useful for multivariate data analysis. Projection pursuit is presented as an algorithm for dimensional reduction that provides revealing information. Projection pursuit algorithm search low dimensional projections optimizing a continuous index that measure if the projection is useful in some sense. There are a few projection pursuits indices that incorporate class or group information in the calculation. Lee et al. (2005) proposed a index derived form linear discriminant analysis that is useful in exploratory supervised classification. They define a index that incorporate information between group variation relative to within group variation, this approach can be useful to explore cluster information and also to build a classifier for prediction. Let X_{ij} a p-dimensional data where for the j-th observation of the i-th class, $i = 1, \ldots, g$, g is the number of classes, $j = 1, \ldots, n_i$, and n_i is the number of observations in class i.

The projection pursuit index based on LDA can be defined as follows:

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

Where B is between-group sums of squares $(\sum_{i=1}^{g} n_i(\bar{X}_{i.} - \bar{X}_{..})(\bar{X}_{i.} - \bar{X}_{..})^T)$ and W is the within-group sums of squares $(\sum_{i=1}^{g} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_{i.})(X_{ij} - \bar{X}_{i.})^T)$. If LDA index is small then we expect to observe small differences between classes and if the LDA index is high then we expect to observe large differences between classes.

Most of the projection pursuits indexes do not work well in the case of large p small n cases (large number of variables vs small number of observations). Lee and Cook (2010) proposed a new projection pursuit index that overcomes the large p small n problem for exploratory classification. The main idea used in the index construction is that when $n \leq p$ or the variables are highly correlated the maximum likelihood variance-covariance matrix estimator will be close to be singular and then this will affect the inverse calculation. Then a different way to compute the variance-covariance matrix is used in the new index.

PDA index is an extension of LDA index were the main objective is to find projections that contain class separations when there are a small number of observations and a large number of variables.

$$I_{PDA}(A, \lambda) = 1 - \frac{|A^T(1-\lambda)W^s + n\lambda I_p A|}{|A^T(1-\lambda)(B^s + W^s) + n\lambda I_p A|} \tag{1}$$

Where B^s is the between-class sums of squares of the standardized data and W^s is the within-class sums of squares of the standardized data. A is an orthonormal projection onto a $k-$dimensional space and $\lambda \in [0, 1)$ is a predetermined parameter.

Lee et al. (2013) describes the projection pursuit classification tree algorithm implemented in PPtree package.

Construct of the PPtree from Lee et al. (2013) is as follows: ....“Let (X_i, y_i) the data set, X_i is a p-dimensional vector of explanatory variables and $y_i \in 1, 2, \ldots G$ represents class information with $i = 1, \ldots n$.

1. Optimal one-dimension projection α^* if found, for separating all classes in the current data.
2. Reduce the number of classes to two, by comparing means, and assign a new label “$G1$” or “$G2$”(y_i^*) to each observation.
3. Re-do projection pursuit these new group labels , $G1$ and $G2$, finding the optimal one dimension projection, α, using (X_i, y_i^*)
4. Calculate the decision boundary c.
5. Keep α and c
6. Separate data into two groups using new group labels, “G1” and “G2”
7. For “G1” group,
   - If there is only one class among the original classes $1, 2, \ldots G$ stop expanding the PPtree.
   - Else repeat step 1 through step 6 with the se classes.
8. For “G2” group
   - If there is only one class among the original classes $1, 2, \ldots G$, stop expanding the PPtree.
   - Else, repeat step 1 through step 6 with the se classes.

”....

281 3 Projection pursuit random forest

In this paper we combine projection pursuit classification trees and random forest ideas. As we described before random forest is an ensemble learning method, built on bagged trees. The trees used in the most common random forest implementations are CART or C4.5 trees. These kinds of trees uses only one variable in each split and then define hyperplanes that are orthogonal to the axis. Sometimes we have data where the class can be separated by linear combinations and in this cases use a classifier which define hyperplanes that are oblique to the axis maybe do a better job.

We adapts the random forest using PPtree algorithm instead of CART, the idea is to utilize combinations of variables in the tree construction, which we call the projection pursuit classification random forest (PPforest). In a random forest each split is based on a single variable, chosen from a subset of predictors. In the PPforest, each split is based on a linear combination of randomly chosen variables.

Algorithm description

1. Let N the number of cases in the training set $\Theta = (X, Y)$, B bootstrap samples from the training set are taking (samples of size N with replacement)

2. For each bootstrap sample a `PPtree`is grown to the largest extent possible $h(x, \Theta_k)$. No pruning. This tree is grown using step 3 modification.

3. Let M the number of input variables, a number of $m << M$ variables are selected at random at each node and the best split based on a linear combination of these randomly chosen variables. The linear combination is computed by optimizing a projection pursuit index, to get a projection of the variables that best separates the classes.

The PPforest uses the PPtree algorithm, which fits a single tree to the data. Utilizing linear combinations of variables to separate classes takes the correlation between variables into account, and can outperform the basic forest when separations between groups occur on combinations of variables. Two projection pursuit indexes, LDA and PDA, are used for PPforest. The methods are implemented into an R package, called PPforest, which is available on github.

3.1 Importance

In PPtree the projection coefficients used to obtain the dimension reduction at each node can be used to determine the variable importance (variables are standardized). Then before construct each tree in the forest we need to standardized the data so the coefficients can be used to interpret the contribution of the variables. Since in PPforest there is more than one tree, a global importance measure can be define. This global measure should be take into account the importance for each tree, the OOB-error of each tree and the node where the variable was used. Then we define the global importance measure for a PPfores object as a weighted mean of the absolute value of the projection coefficients across all nodes in every tree. The weights are the projection pursuit indexes in each node, and 1-(the out of bag error of each tree).

4 Performance comparison

This section compares the performance of PPforest classifier with other classification methods, PPtree, CART, randomForest and PPforest with results for 9 benchmark data sets. For each data set we divide the sample in 2/3 of the observations in training and 1/3 in test. We use the 4 different methods based on training data and compute training and test error. The same procedure is repeated 200 times and the mean of the error rate for each method is computed.

The data we will use will be:

1. Australian crab: contains 200 observations from two species (blue and orange) and for each specie (50 in each one) there are 50 males and 50 females. Class variable has 4 classes with the combinations of specie and sex (BM, BF, OM and OF). There are 5

continuous feature variables. FL is the size of the frontal lobe, RW is rear width, CL is carapace length, CW width and BD body depth.

2. Leukemia: contains 72 observations and 40 feature variables. Class variables have 3 classes with 38 cases of B-cell ALL, 25 cases of AML and 9 cases of T-cell ALL. The feature variables are 41 Gene expressions with 3571 values.

3. Lympohma: contains 80 observations and 50 feature variables. Class variable has 3 classes with 29 cases of B-cell ALL (B-CLL), 42 cases of diffuse large B-cell lymphoma (DLBCL) and 9 cases of follicular lymphoma (FL).

4. NCI60: contains 61 observations and 30 feature variables. Class variable has 8 different tissue types, 9 cases of breast, 5 cases of central nervous system (CNS), 7 cases pf colon, 8 cases of leukemia, 8 cases of melanoma, 9 cases of non-small-cell lung carcinoma (NSCLC), 6 cases of ovarian and 9 cases of renal. There are 6830 genes.

5. Wine: contain 178 observations and 13 feature variables. Class variable has 3 classes that are 3 different wine grown cultivares in Italy.

6. Glass: contain 214 observations and 9 feature variables. Class variable has 6 classes.

7. Fishcatch: contains 169 observations and 6 feature variables. Class variables has 7 classes, with 35 cases of Bream, 11 cases of Parkki, 56 cases of Perch 17 cases of Pike, 20 cases of Roach, 14 cases of Smelt and 6 cases of Whitewish. The feature variables are, weight, height, width and 3 different measures of fish length.

8. Image: contains 2310 observations and 18 feature variables. The 2310 observations are instances from 7 outdoor images: brickface, cement, foliage, grass, path, sky, and window.

9. Parkinson: contains 195 observations and 22 feature variables. Class variable has 2 classes; there are 48 cases of healthy people and 147 cases with Parkinson. The feature variables are biomedical voice measures.

Table 1: Comparison of PPtree, CART, random forest and PPforest results with various data sets. The mean of training and test error rates from 200 re-samples is shown

	TRAINING				TEST			
	PPtree	Cart	RF	PPforest	PPtree	Cart	RF	PPforest
Crab	0.0430	0.2709	0.2105	0.0451	0.0616	0.4517	0.3134	0.0448
Leukemia	0.0060	0.0380	0.0625	0.0000	0.0523	0.1525	0.0000	0.0000
Lymphoma	0.0281	0.0525	0.0943	0.0000	0.0746	0.1656	0.0370	0.0370
NCI60	0.0641	0.4572	0.4500	0.0000	0.4845	0.7505	0.3333	0.1905
Wine	0.0019	0.0498	0.0336	0.0084	0.0200	0.1210	0.0000	0.0000
Glass	0.3140	0.2376	0.2517	0.2657	0.4229	0.3370	0.1831	0.3239
Fishcatch	0.0001	0.1438	0.1981	0.0000	0.0164	0.2341	0.2642	0.0189
Image	0.0666	0.0692	0.0227	0.0610	0.0722	0.0834	0.0286	0.0727
Parkinsons	0.1222	0.0813	0.1077	0.1231	0.1761	0.1592	0.0923	0.1846

In Table 1 the mean of training and test error rates from 200 re-samples is shown for the different supervise classification methods used. For the PPforest method for each data set 9 classifiers were fitted changing the size.p parameter, size.p is the proportion of variables selected in each partition (size.p from 0.1 to .9 with 0.1 jumps). We have included the results with smaller test error. The variable size proportion in each case were; crab .9, leukemia .9, lymphoma from .5 to .9 we got the same test error result, NCI60 0.2 and 0.3, wine same test error result from .7 to .9, glass .8, fishcatch same test error form .7 to .9, image .5 and Parkinson same test error result from .7 to .9.

The results presented here show that PPforest procedure get a better performance than the other methods presented here (crab, NCI60) while in other cases the test error is the same than using random forest procedure (Leukemia, Lymphoma and Wine).

5 Simulation study (Still more to do here)

For many problems, particularly large p, random forests may be adequate. It is when the differences between classes is on linear combinations of variables that the PPforest shold perform better. This simulation study is designed to examine just where this divergence occurs: at what correlation does the PPforest provide lower error than a random forest.

The data were simulated from tree four-variate normal distribution, each of them corresponds to one class, with the same variance structure:

$$\Sigma_s = \begin{pmatrix} 1 & \rho_s & \rho_s & \rho_s \\ \rho_s & 1 & \rho_s & \rho_s \\ \rho_s & \rho_s & 1 & \rho_s \\ \rho_s & \rho_s & \rho_s & 1 \end{pmatrix} \tag{2}$$

A scenario is then determined by a set of mean vectors $\{\mu_1, \mu_2, \mu_3\}$ and a correlation value,ρ_s. In total, 28 different scenarios were defined. Table 2 shows the mean values in each scenario and the values for ρ_s were $(.7, 0.75, 0.8, .85, .9, .95)$

Table 2: Simulation mean scenarios

Mean scenario	μ_1	μ_2	μ_3
1	$(0,0,0,1)^T$	$(0,0,0,3)^T$	$(0,0,0,5)^T$
2	$(0,0,0,1)^T$	$(0,0,0,5)^T$	$(0,0,0,7)^T$
3	$U(-1,1)$	$U(-1,1)$	$U(-1,1)$
4	$(0,0,0,0)^T$	$(1,1,1,1)^T$	$(2,2,2,2)^T$

Figure 3 shows the mean 00B error from 50 forest for each scenario and each correlation value. We can observe than in scenarios 2, 3 and 4 the performance of PPforest is better for all the correlation values but in the first scenario PPforest is better only for bigger values of correlation.

6 Optimality? (Still to do)

It may be possible to show that the PP forest is optimal under some conditions for a two group problem, theoretically.

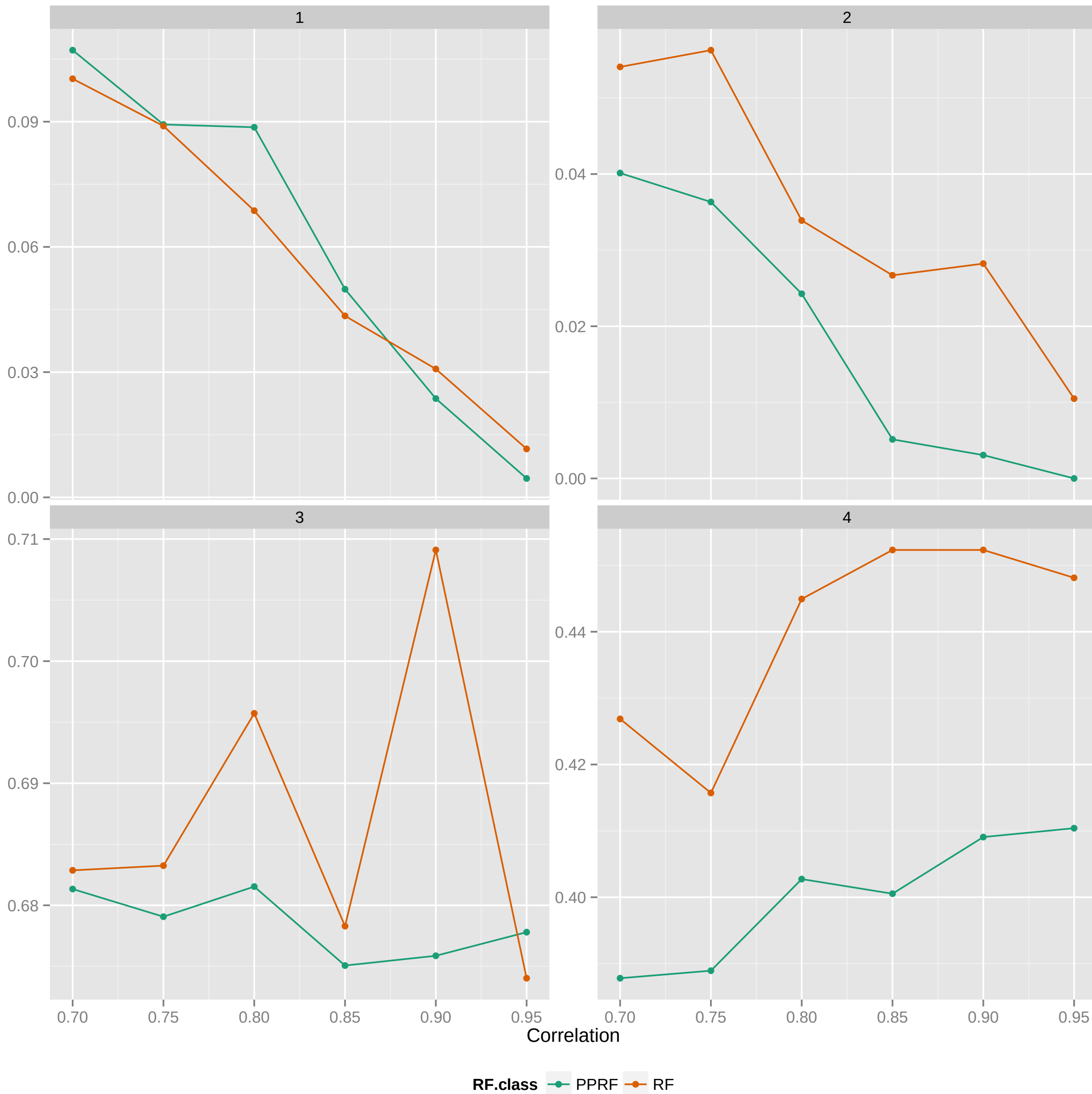

Figure 3: Mean OOB error from 50 simulated forest for each scenario by different correlation levels

7 Remarks

Utilizing linear combinations of variables to separate classes, PPforest takes the correlation between variables into account, and can outperform the basic forest when separations between groups occurs on combinations of variables. Two projection pursuit indexes, LDA and PDA, are used for PPforest. In the cases presented in this paper, we have observed that PPforest performance is better in some cases than CART, PPtree and random forest classification methods. Additionally the benchmark data shows some of the situations where our method does a better job than the random forest. This give us margin to continue working in this direction and improve our classifier to get a better performance in the cases is not working well.

Chapter 4

Visualization of a PPforest object

Visualization is critical to help obtain an understanding of the class structure in the data and how the model fits it. Wickham et al. (2015) describe a conceptual framework and examples for visualizing models, with some focus on classification problems. In this chapter the visualization methods provided with the PPtreeViz and some examples from ggRandomForests packages are described, and ideas for extending these to the new package PPforest are proposed.

Because the PPforest is composed of many tree fits on subsets of the data, a lot of statistics can be calculated to analyze as a separate data set, and better understand how the model is working. Some of the diagnostics of interest are: variable importance, OOB error rate, vote matrix and proximity matrix.

The crab data, consisting of 200 observations and 4 classes, is used to illustrate the visual methods. Feature variables are:

1. FL, the size of the frontal lobe length, in mm
2. RW, rear width, in mm
3. CL, length of mid-line of the carapace, in mm
4. CW, maximum width of carapace, in mm

5. BD, depth of the body; for females, measured after displacement of the abdomen, in mm

Importance variable

The PPtree algorithm, organizes a multiclass problem by first separating classes in two groups, and using projection pursuit to find the best separation of these two groups. The variable importance for the group separation can be measured by the projection coefficients. Based on these coefficients we can examine how the classes are separated and which variables are more relevant for the separation. Figure 4.1 shows the visualization tool from `PPtreeViz` package to describe the variable importance in a PPtree object. A bar chart represents the projection coefficients, with larger values indicating more important variables. In this example, FL and CW are the most important variables to get a separation between species of crab.

The projection coefficients are still the primary indication of variable importance for a PPforest object, however with multiple tree fits a simple bar chart will not suffice. As a first approach a parallel coordinate plot is used. Parallel coordinate plots were introduced by Inselberg (1985) and it is a very useful tool to see multiple dimensions. A vertical axis is used for each variable, drawn in parallel to each other, and each observation is represented by a point on each axis, finally each point is connected by a piece-wise line. For representing the projection coefficients of multiple projections in a parallel coordinate plot, the variables form the vertical axes, as usual, and each line connects values of the coefficients obtained for a node in one tree.

Figure 4.2 shows the parallel coordinate plot with importance variable information a PPforest fit on the crabs data, where two variables were randomly selected. Each node is displayed in a separate plot. In this plot each line represent a single tree.

A lot of overlap is presented in this plot which indicates a lot of variables substitute for each other. If we focus on node 1, we can observe that FL has always a positive contribution while BD most of the time negative contribution. Based on this plot these are the most important

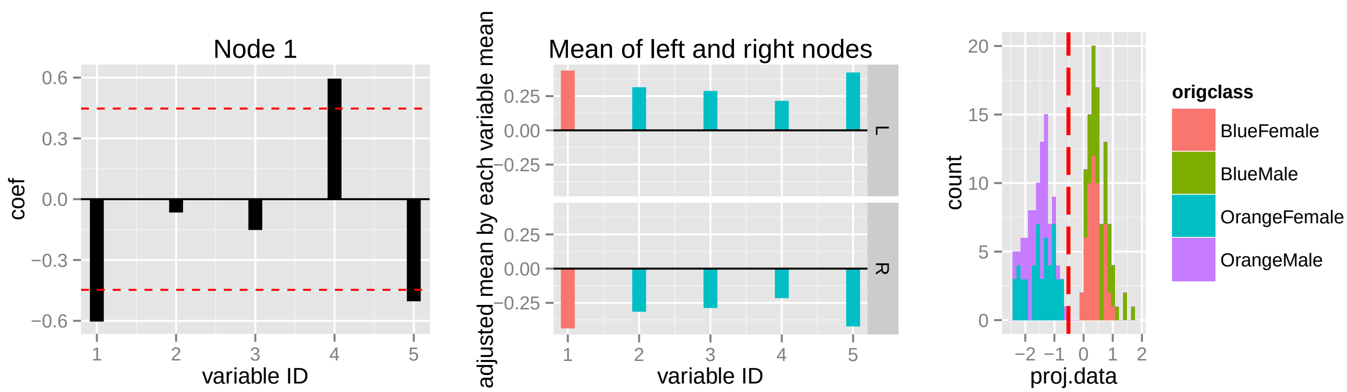

Figure 4.1: Importance measure visualization using PPtreeViz . The left panel shows a bar chart with the projected coefficients from the node 1. Variable 1 and 4 have important roles to separate blue from orange crab, but they work in different direction. The middle plot "Mean of left and right node", where "L" means the left group (orange male and orange female in the node 1) and "R" means the right group (blue male and blue female). We can see that orange male and orange female have large value of variable 1 and variable 4 while the other two classes have small value of variable 1 and 4. The right panel shows an histogram with the projected data, the dashed vertical red line represents the cutoff values for the selected rule.

variables for class separation. Other variables like CL and CW present contribution in both directions. But this is a small example were all the variables are quite important for the class separation.

Maybe this plot is a good tool if we have some variables that are really important and others that are not important at all. To explore this Figure 4.3 shows the same data with five additional noise variables. The difference in pattern is dramatic. The first five variables (CL, CW, BD, FL, RW) have a different pattern to the last five (X1-X5) with the last five being mostly 0, and only occasionally having a much larger coefficient in a few trees. It is clear to see that all five crab variables are important to some extent in the classification.

Figure 4.4 shows the parallel coordinate plot of importance variable for each node and each tree in the case of three variables were selected in the node partition. In this plot we can observe more variability, in all the cases the contribution of the variables are in both directions, and it is not clear which variable is more important.

Since we are changing the number of variables used in each plot maybe we need to redefine a measure of importance by pairs of variables or for three variables for the previous example. Figure 4.5 and 4.6 show a boxplot for the importance variable measure but considering two and three variables. The importance here is the sum of the absolute value of the projection coefficients. Figure 4.5 shows a small variability in the importance measure for all the 10 pair of variables. With 500 trees in the forest we expect 50 observation for each of the 10 pair of variables. The most important pairs are CL.CW and FL.BD follow by FL.RW and RW.BD. Figure 4.6 shows us the case where we 3 variables were selected at random in each node split. In this boxplot we can observe more variability than in the previous one. The most important group of variables are FL.CL.BD and FL.CW.BD.

To get a simpler importance variable measure from a PPforest object we can define a global measure to take into account the importance for each tree and the level in which the variable was used in each tree. The global importance measure for PPforest object take into account the OOB-error of each tree and the node where the variable was used. Then the importance is a weighted mean of the absolute value of the projection coefficients across all nodes in every tree. The weights are the projection pursuit indexes in each node, and 1-(OOB-error of

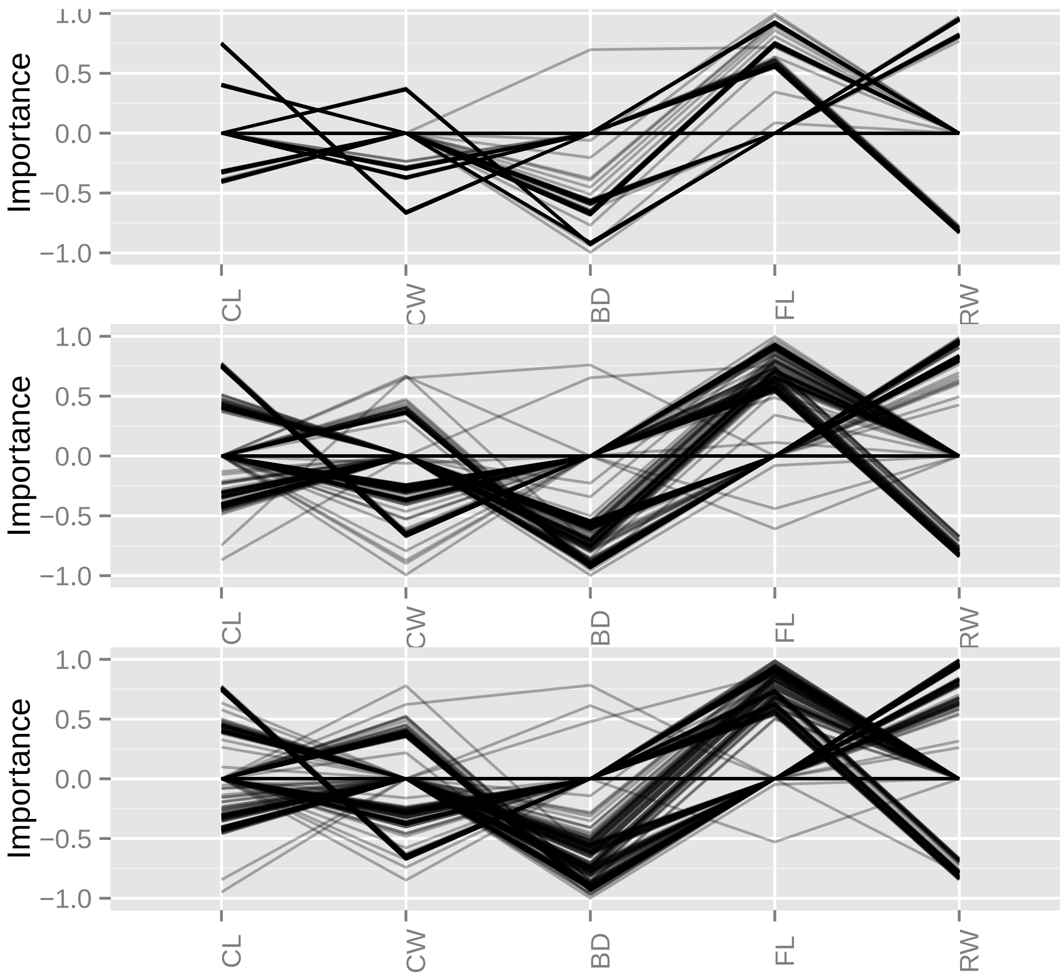

Figure 4.2: Parallel coordinate plot of importance variable for each node (top=node 1, middle=node 2, bottom=node 3) of each tree, using 2 variables randomly selected in each node. Its really messy! But there are some common threads. FL factors highly at all nodes, and clearly has a large coefficient regardless of what other variable it is paired with. Something similar could be said for BD. RW factors in always at ± whenever it is selected. CL and CW factor more in smaller amounts.

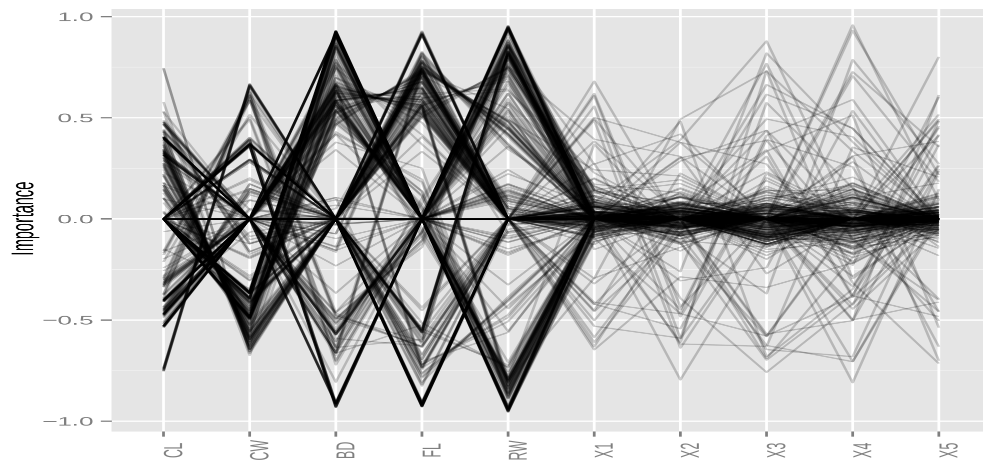

Figure 4.3: Parallel coordinate plot of importance variable for node 1 and each tree, in the presence of 5 additional noise variables. It is clear that the crabs and noise variables have different patterns.

each tree). Figure 4.7 shows a dot plot with the global importance measure for the PPforest. Based on this we can say that RW and BD are the most important variables to separate the classes in our PPforest example.

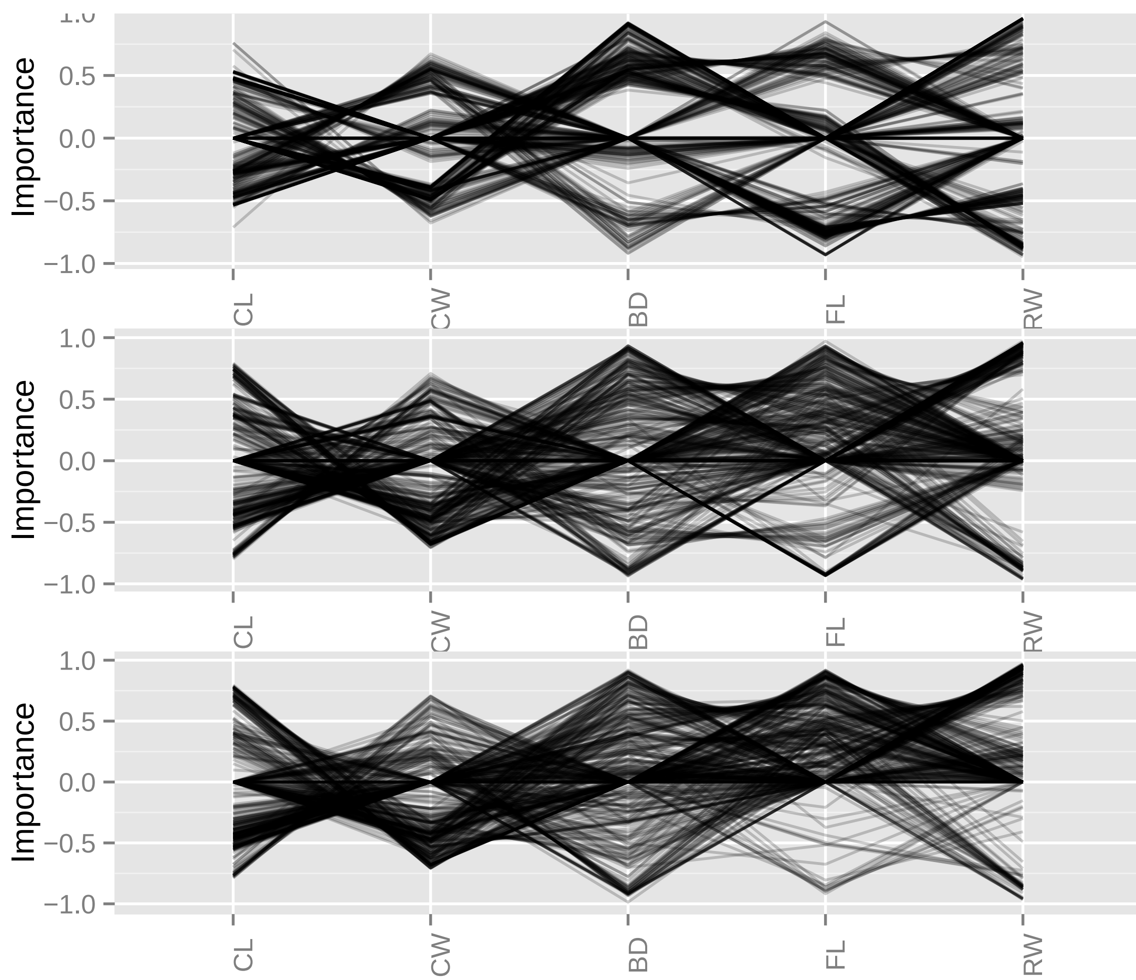

Figure 4.4: Parallel coordinate plot of importance variable for each node (top=node 1, middle=node 2, bottom=node3) of each tree, using 3 variables randomly selected in each node. Its messier still! But the resolution is clearer at node 1, and nodes 2 and 3 clearly have a strong contribution from RW when it is included in the three, and CL. The other three variables clearly have strong contributions much of the time, but have commonly much smaller coefficients also.

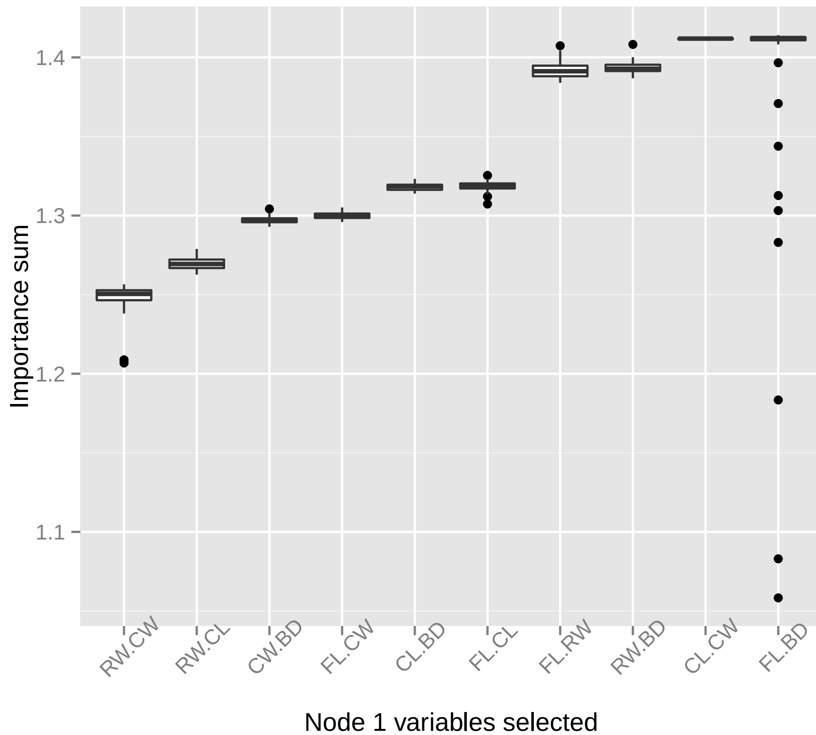

Figure 4.5: Boxplot with importance measure for a group of two variables only with the first node information. The importance measure is the sum of the absolute value of the projected coefficient for the two random selected variables in node 1. We can observe than some pairs are more important than others, FL.BD, CL.CW, RW.BD and FL.RW. The variability is small in all the cases and in FL.BD we can identify some outliers.

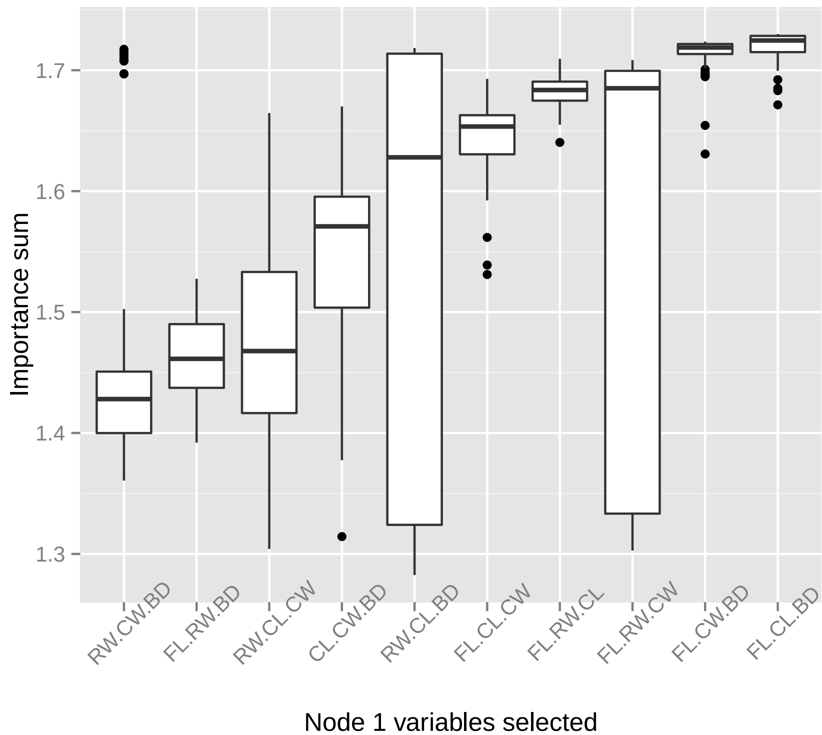

Figure 4.6: Boxplot with importance measure for a group of three variables only with first node information. The importance measure is the sum of the absolute value of the projected coefficient for the three random selected variables in node 1. We can observe for some groups a big variability in the importance measure, RW.CL.BD and FL.RW.CW. Base on this plot we can observe that the most relevant group of variables are FL.CL.BD and FL.CW.BD.

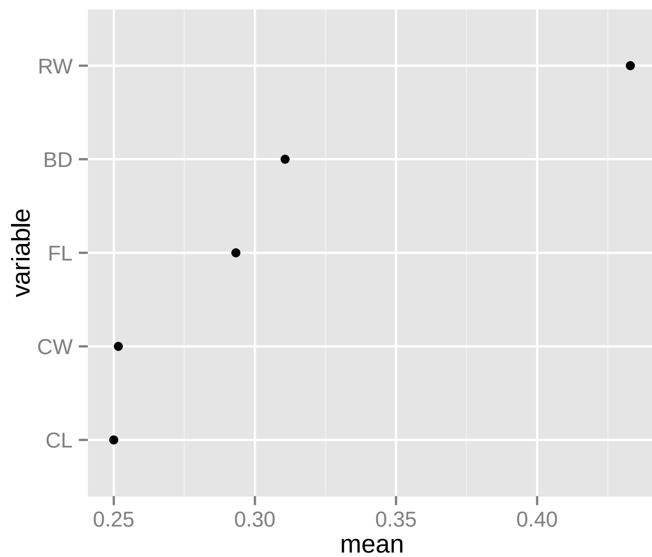

Figure 4.7: Dot plot with the global importance variable for a PPforest. This global measure take into account the OOB-error for each tree and the projection pursuit index. Based on this measure, the most important variable is RW followed by BD .

OOB error rate

In PPforest like in random forest is not needed to separate the data in test and training because we can get an unbiased estimate of the test error internally from the forest. These error are called out of bag errors, as we describe in the literature review each tree is grown using different bootstrap samples from the original data then part of the data are not used in the tree construction (one-third). With these "out-of-bag" data in each tree we get the classification of them. At the end we consider i to be the class with more votes every time case n was out-of-bag. The proportion of times that case n is wrong classified (i different to true class of case n) averaged over all cases is the OOB error estimate. Out of bag error rate is the class of errors.

We would like to visualize if there is any reduction in the OOB error when we increase the number of trees in the forest. We would like to know if there is any reduction in the OOB-error and if this error change for the different classes.

Figure 4.8 shows the cumulative out of bag error for each class. In this case we can observe

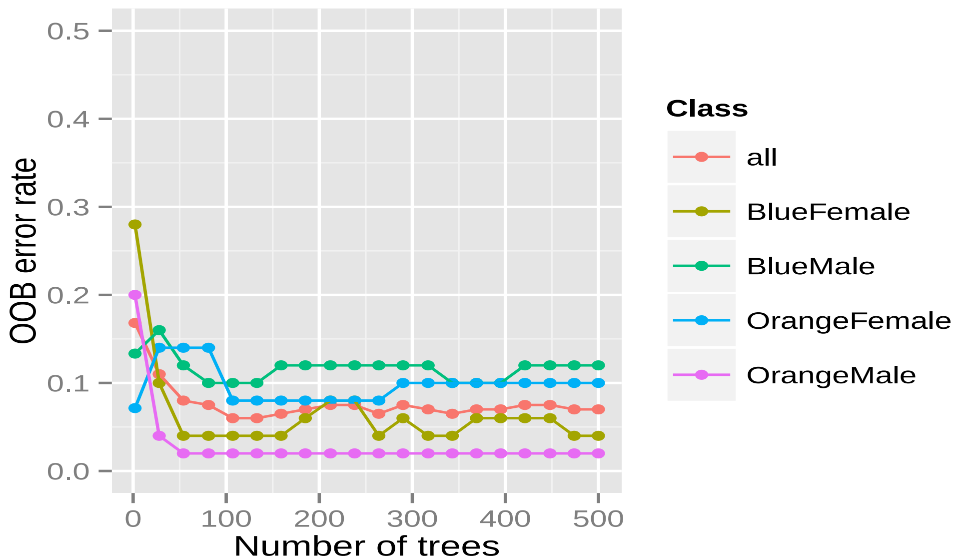

Figure 4.8: Cumulative OOB-error visualization . In this plot a line with cumulative OOB-error for each class and all the classes is computed. We observe a reduction of the cumulative OOB-error for all the classes until 150 trees. In this case will be enough for the error reduction no more than 200 trees in the forest.

that the reduction after 200 trees is very small. Then in this case with less than 200 trees will be enough to get the OOB error reduction we expect.

Proximity matrix

With a decision tree we can compute for every pair of observations the proximity matrix. This is a nxn matrix where if two cases k_i and k_j are in the same terminal node increase their proximity by one, at the end normalize the proximities by dividing by the number of trees. To visualize the proximity matrix we use a heat map plot and an a scatter plot with information from multidimensional scaling method.

Figure 4.9 in left panel shows the heat map for the proximity matrix, we can observe that strong color indicates that the observations are more similar. We can see a block structure in the diagonal explained by the order of the data. The data are order by classes then this block diagonal structure means that the same class were classified most of the time in the correct class. Additionally we can observe in the top left of the left panel a block structure that means that blue males are similar to the blue females and then these two groups are difficult to classify in their correct class.

A second proximity matrix visualization is based on multidimensional scaling (MDS). MDS is a statistical method used to study the similarity between observations, we can use it to see if there are some clear cluster structure in our data. Multidimensional scaling transform the data set in a set of points where the distance between points are approximately equal to the similarities. The distance we see in the plot is similar to how close the observations are. In Figure 4.9 (right panel) we can observe that blue males and blue female are closer than orange male and orange females and these are the points that present a bigger classification error.

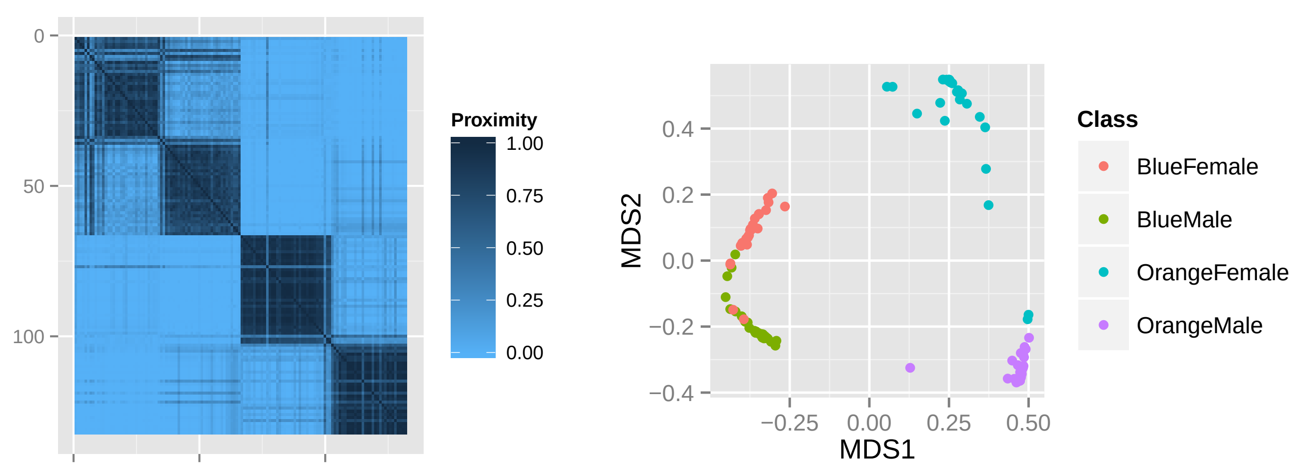

Figure 4.9: Left panel shows a heat-map with the proximity matrix. Stronger color indicate that the observations are more similar. We can observe a block diagonal structure with stronger color explained because the data are ordered by class. This indicates that the same class were classified most of the time in the correct class. Also in the top left panel we see a strong blue color block that indicate that blue males are similar to blue females and these two classes are difficult to classify. In the right panel we can see a MDS plot. In this plot we can see that blue male and blue female are closer than the other classes .

Vote matrix

In a vote matrix the information that we have is the proportion of time a observation was classified in each class. Then in this matrix we have one row for each input data point and one column for each class, giving the fraction of (OOB) votes from the PPforest. We have two possible approaches to visualize the vote matrix information, using a ternary plot or a side by side jittered dot plot.

A ternary plot is a triangular diagram that shows the proportion of three variables that sum to a constant and is done using barycentric coordinates. One advantage of ternary plot is can show the proportion of tree variables in two dimension and are a good way to visualize compositional data. Ternary plot are good when we have tree classes but when we have more than three classes the ternary plot need to be generalized to more dimensions.

In this case since we have four classes in crab data example the ternary plot generalization will be a tetrahedron and we can use the tour package to see this. Figure 4.10 shows the tetrahedron structure with our vote matrix information from different rotations. In this tetrahedron we wish to see each color group as close as possible to the tetrahedron corners, the orange dots in green corner are errors while the orange dots in the violet corner too.

Finally Figure 4.11 shows a side by side plot with vote matrix information. We can see here that points close to 1 and with correct color are clearly that class while points close to 1 but with a wrong color are wrong classified by our model.

For future work we plan to include visualization in big data context where the data sometimes is split in chunks and the final model is base in some kind of model averaging where it is important to visualize the intermediate models results to understand the final model averaging for example.

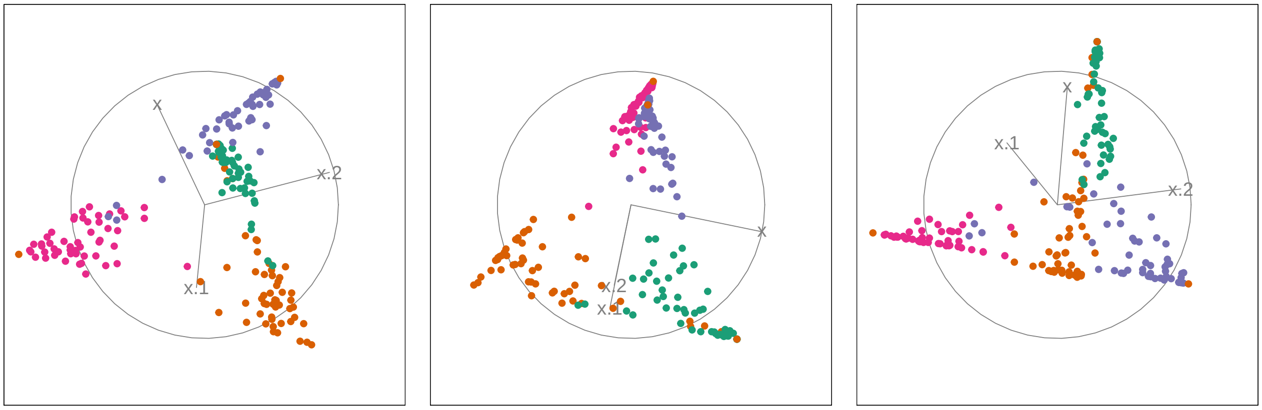

Figure 4.10: Vote matrix images from tour package . This plot is a generalization of a ternary plot with four classes, then we got a tetrahedron with the vote matrix information in different rotations from the tour package. We observe that our model is doing a good job to determine the classes since the colors are concentrated in the corners and there are not a lot of mixed colors in each corner.

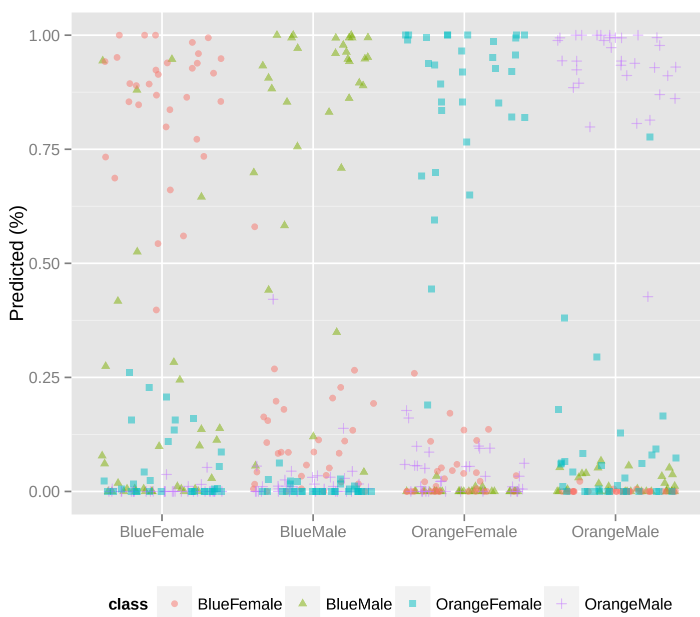

Figure 4.11: Vote matrix side by side plot . We see that blue male and blue female some times are wrong classified, some blue males are classified as blue females because are close to one but in the wrong class. Also we can see one orange female classified as orange male in the last column.

Chapter 5

Practical issues in classifier building

In this section some ideas about future work will be presented, the focus will be practical issues in classification problems. PPforest method have shown better performance than some other method based on real data and simulated data but some issues already appears. An improvement in the speed of the algorithm is needed to work with big data problems. Weights can be consider in the tree construction to deal with practical problems like unbalance data and survey data PPtree algorithm modifications can be done to get more separable classes.

5.1 How to incorporate weights in a PPtree and PP-forest

There are two basic practical problems that can be addressed using weights in the tree construction. One is unbalance data and the other is survey data.

Unbalance data, sometimes when we work with multiple class problems with very different sample size negative effects can be seen in the classification performance. There are different approaches to deal with unbalance data in a multi-class problem, a common approach is re-balance the data artificially oversampling or under sampling some classes. Also with this class unbalance problem it is important to consider some performance measure that take into account this characteristic of the data.

Survey data, Two interesting questions arrive when we work with survey data; how we can deal with the non-response? and how we can incorporate in our model the information from the sample design? Survey data came from subjects that have been selected using complex sample designs and we have to incorporate this information in our model. When our survey data presents a large amount of non-response, sometimes is recommended to do an non-response adjustment to reduce the bias before analyze the data. When we analyze survey data then we need to use the survey weights and the variances of the survey estimates need to be computed also taking into account the complex sample design.

For future work here the idea is to discuss possible benefits of using projection pursuit trees and projection pursuit random forests to analyze survey data. We want to study ways to incorporate survey weights. Sensitivity of the response to different ways to consider the sample design in our model can be analyzed.

5.2 Classification using projection pursuit with big data

Working with big data involve a lot of challenges and one of them is how to deal with the small n large p problem. For the small n large p problem Lee and Cook (2010) proposes a new projection pursuit index that overcomes this problem for exploratory classification. This index is available in the PPtree package, but utilizing it requires cross-validation selection of the control parameters. To work with big data this needs to be computed more efficiently.

5.3 PPtree algorithm modification

PPtree construction can be modified to get better separation of classes in some specific problems. For example some times the boundaries of our model are defined very close to some of the classes maybe we can instead of work with all the classes to find the best separation we can work with two classes at a time and evaluate the performance for different alternatives.

5.4 Projection pursuit functions

Utilizing different projection pursuit functions may provide extensions of the PPtree and PPforest to tackle other data problems: where there is a continuous rather than categorical response variable, when there is no response variable and the purpose is to group observations to construct a class variable, and the presence of categorical predictors.

Chapter 6

Plans and timeline

Table 6.1: Completed work

Product	Description	Data
Talk	Presented the PPforest package at useR! 2015 conference in Alborg, Denmark	July 2015
Poster	Poster presented in JSM 2015, Seattle, USA	August 2015
R package	PPforest in github	October 2015
Paper	Projection pursuit classification random forest	December 2015

Table 6.2: Schedule for completion

Product	Description	Data
Paper	Document new visualization methods for projection forests, that incorporate some interactive graphics	March 2016
R package	Optimize speed of PPforest, upload in CRAN, and submit a condensed version of the vignette to JSS or R Journal	April 2016
R package	Explore modifications to the projection pursuit forest, perhaps by modifying the underlying tree algorithm, to address the practical issues such as weights	April 2016
Application	The use of the new algorithm for nutrition or forensics data will be examined	May 2016
Paper	Document the modifications to address practical issues	December 2016
Thesis defense	Planned thesis defense	May 2017

Bibliography

Amit, Yali, and Donald Geman. 1997. Shape quantization and recognition with randomized trees. *Neural computation* 9 (7): 1545–1588.

Breiman, Leo. 1996. Bagging predictors. *Machine learning* 24 (2): 123–140.

Breiman, Leo. 2001. Random forests. *Machine learning* 45 (1): 5–32.

Breiman, Leo, et al.. 1996. Heuristics of instability and stabilization in model selection. *The annals of statistics* 24 (6): 2350–2383.

Breiman, Leo, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 1984. Classification and regression trees. wadsworth. *Belmont, CA*.

Brodley, Carla E, and Paul E Utgoff. 1995. Multivariate decision trees. *Machine learning* 19 (1): 45–77.

Friedman, Jerome H, and John W Tukey. 1973. A projection pursuit algorithm for exploratory data analysis.

Ho, Tin Kam. 1998. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20 (8): 832–844.

Hothorn, Torsten, Kurt Hornik, and Achim Zeileis. 2006. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics* 15 (3): 651–674.

Kass, Gordon V. 1980. An exploratory technique for investigating large quantities of categorical data. *Applied statistics*.

Kim, Hyunjoong, and Wei-Yin Loh. 2001. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association* 96 (454).

Lee, Eun-Kyung, and Dianne Cook. 2010. A projection pursuit index for large p small n data. *Statistics and Computing* 20 (3): 381–392.

Lee, Eun-Kyung, Dianne Cook, Sigbert Klinke, and Thomas Lumley. 2005. Projection pursuit for exploratory supervised classification. *Journal of Computational and Graphical Statistics* 14 (4).

Lee, Yoon Dong, Dianne Cook, Ji-won Park, Eun-Kyung Lee, et al.. 2013. PPtree: Projection pursuit classification tree. *Electronic Journal of Statistics* 7: 1369–1386.

Loh, Wei-Yin. 2009. Improving the precision of classification trees. *The Annals of Applied Statistics.*

Loh, Wei-Yin, and Yu-Shan Shih. 1997. Split selection methods for classification trees. *Statistica sinica* 7 (4): 815–840.

Loh, Wei-Yin, and Nunta Vanichsetakul. 1988. Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association* 83 (403): 715–725.

Menze, Bjoern H, B Michael Kelm, Daniel N Splitthoff, Ullrich Koethe, and Fred A Hamprecht. 2011. On oblique random forests. In *Machine learning and knowledge discovery in databases*, 453–469. Springer.

Morgan, James N, and John A Sonquist. 1963. Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association* 58 (302): 415–434.

Quinlan, J. R. 1993. C4.5: Programs for machine learning.

Tan, Peter J, and David L Dowe. 2005. Mml inference of oblique decision trees. In *Ai 2004: Advances in artificial intelligence*, 1082–1088. Springer.

Tan, Peter J, and David L Dowe. 2006. Decision forests with oblique decision trees. In *Micai 2006: Advances in artificial intelligence*, 593–603. Springer.

Truong, Alfred. 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.*

Wickham, Hadley, Dianne Cook, and Heike Hofmann. 2015. Visualizing statistical models: Removing the blindfold. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 8 (4): 203–225.