Random Forest Variable Importance

To implement the various variable importance measures discussed in this chapter and in chapter 4, funcitons for creating trees, random forests, and their importance measures were created. The trees are fit following the standard two-part CART-like algorithm. The function chooses a variable to split on with linear correlation with respect to Y, but instead of looking for correlations above a certain threshold which is common, it chooses the variable with the highest correlation when compared to its peers. This alleviates the situation where a variable with a non-linear relationship would be passed over again and again. The splitting is done via minimization of the following function with respect to i:

$$RSS_{node}(i, X, Y) = RSS_{leaf}(Y|X < i) + RSS_{leaf}(Y|X \geq i)$$
$$RSS_{leaf} = \sum (y - \hat{y})^2$$

This function considers the regression case only, and only numeric predictors. Leaves are created when the resultant split would be unsatisfactory, i.e. at least one of these cases applies: one daughter node would have five members or less, the split on the chosen variable would not result in a decrease in RSS, or the data contained in the node is already suitable homogenous. This generates very large trees - a quality that is not an issue in random forests but may be problematic in a stand-alone setting.

Table 1: T, a home-grown tree grown on the first four columns and the first 140 rows of D2

var	n	dev	ypred	split.cutleft
W2	140	307025.374636735	60.9097319888865	-6.96100384656169
W2	135	83169.2792831922	43.0483530491652	-3.7891602649377
W4	120	50705.5649592679	31.9334396084353	3.2448237665193
leaf	9	6741.84547650294	78.1476720868562	0
W3	111	29377.3267469149	28.1863396777525	1.7396083892122
W3	22	6346.55869350374	51.2429853154712	2.20767162100899
leaf	15	4912.82259054594	60.1969690603788	0
leaf	7	1433.7361029578	32.0558772906692	0
W4	89	18301.6870406292	22.4869441268558	-0.836982585945597
leaf	56	11890.2428451579	24.9942412092431	0
W3	33	5898.67340773121	18.232136956744	0.0529476269400149
leaf	21	4741.70649830849	21.2119268553199	0
leaf	12	1156.96690942271	13.0175046342361	0
leaf	15	11683.3926459505	131.967660575004	0
leaf	5	90431.4904950311	543.166963361362	0

There are several ways to display a tree-model, but when it is displayed as a table it is read in the following way: each row corresponds to a node of the tree which contains a certain number, `n` observations. This number of observations, or rows in the data set is natuarally a subset of both the original data set and the subsets above the node on the tree. Here our predictions, `ypred`, are the mean of the Y values included in the node. If there is an optimal and allowable split, [1] then the chosen variable, `var`, and the RSS_{node}, `dev`, are recorded.[2] The value of the variable in question that acts as the split point is recorded as `split.cutleft`. If there is no split on the node in question, then `var` will be recorded as `<leaf>` and the `dev` value will be the value of RSS_{leaf} at this node.

The tree output is read roughly from top to bottom, with a coda in the middle. The first row corresponds

[1] Recall that we only allow splits to take place that split the data into two groups, each with more than five members.

[2] It's the convention to call the RSS_{node} the deviance at a node N, but, of course, this only makes sense when the node is a leaf.

to the first node, or the node that includes the entire dataset. The second row is the beginning of the right subtree or the right daughter of first node. This pattern continues, favoring the right daughter, until a leaf is reached. The left daughter of the first node is found after all of the splits off of the right daughter have finished but is easily identified as the row with a value of `n` that is exactly the difference between the `n` values of the first two rows. In the case where the right daughter contained many more observations of the original dataset, there may be a node within the right subtree that contains the same number of observations as the left daughter of the first node. In this case, the left daughter is simply the second row with this property. The pattern of following the right daughter until a leaf is reached continues with the left subtree.

Breiman et al Introduce Permuted Variable Importance (1984)

Variable Importance on a Single Tree

Breiman et al in *Classification and Regression Trees* (1984) propose a method for variable importance for individual trees that stems from their definition of $\tilde{s}$, a surrogate split. Surrogate splits help Brieman et al deal with several common problems: missing data, masking, and variable importance. They are defined using logic that resembles that behind random forests.

Before we discuss surrogate splits, lets cover an obvious definition of variable importance of a single tree. In the tree represented by table 1, define variable importance as the number of splits on each variable. This would allow us to answer the question: how useful (important) was variable X_i in constructing our model for Y? Just by counting the splits on that variable, we would arrive at the following ranking:

Table 2: The number of splits on each variable in the tree T

variable	appearances.in.tree
W1	0
W2	2
W3	3
W4	2

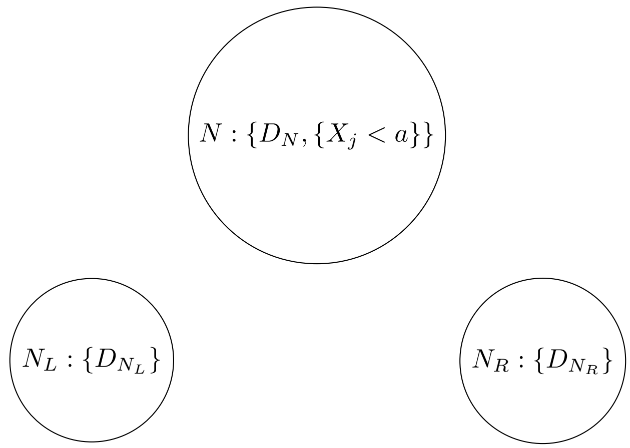
There are several downfalls to this method. One, trees are variable. If we were to resample this data and fit another tree, it's likely that this ranking would change. Two, in the case where two variables are close enough to each other that they could act as stand-ins for one another, these rankings are much less interesting. We are lucky in this case to know, without much of a doubt, that X_1 has a rich relationship with X_2 and the other predictors included in this model. (See chapter 2, section 1). This leads us to believe that while X_1 is left out of these rankings, it just as easily could have been included instead of X_2, or one of the other predictors. X_1 had bad luck by not being in this model and it wouldn't make sense to say that the X_1 is the least important predictor of Y when it is very nearly identical to X_2. But, it's possible that the tree algorithm would only pick one to be included in a model at a time. Is it possible, by only fitting one tree, that we can grasp this relationship?

Luckily this is one of the dilemmas solved by surrogate splits. To set the stage for surrogate splitting, imagine a CART tree, T, fit on some data set D according to the formula $Y \sim X$ where $X_i \in X, \ if i \in 1 : p$. Now say that we're only considering a single node, N, in T.The node N contains the subset of the rows in the original dataset D, D_N. D_N is determined by the previous nodes and splits in the tree.

$$N : \{D_N\}$$

On that node, we have the split on X_j where $X_j < a$. This gives us two daughter nodes to N, N_L and

N_R.



The datasets D_{N_L} and D_{N_R} are subsets of D_N and when combined, they equal D_N. They are determined by the rule: if a row of observations has a value of $X_j < a$ then it is a member of D_{N_L}, if the value of X_j in that row is greater than or equal to a then it belongs to D_{N_R}. X_j was chosen to split on in node N because the correlation between the subsets of X_j and Y in D_N was stronger than the correlations between Y and any of the other predictors in that subset of the original data. Imagine, however, that a split on X_i would lead to very similar [3] left and right daughter nodes, even though X_i and Y had a lower correlation than Y and X_j. This would be considered a surrogate split for our original split on N. Now define variable importance for a predictor X_j across the tree T as the decrease in RSS_{node} according to the split on X_j, whether *surrogate* or not. This allows X_j and X_i to share the importance measure, if both X_j and X_i would have provided a similar, valuable split on node N. In *Classification and Regression Trees*, Breiman et al, outline several potential problems with this method that they do not attempt to solve. First, that this is only one of a number of reasonable ways to define variable importance. Second, the variable importances for variables $X_1, .., X_p$ can be effected by outliers or random fluctuations within the data. (Ch 5.3). The second problem is mitigated when we move from single trees to a random forest, but the first is a problem with variable importance in general.

Variable Importance for a Random Forest

One way to define variable importance for a random forest follows directly from Breiman et al's definition for a single tree. Recall that each tree in a random forest is fit to a bootstrapped sample of the original observations. To estimate the test error, therefor, no cross validation is needed - each tree is simply tested against the test set of observations that were not in that tree's initial training set. Additionally, we may be interested in defining variable importance for a predictor X_j by considering the predictive capabilities of the other $p-1$ predictors. Recall: a random forest is a set of trees that are de-correlated with each other because at each split on each tree, less than half of the predictors are not even considered as possible candidates for splitting. To estimate the importance of X_j given the other variables X_{-j} and their relationship with Y, we can consider the "test" RSS of the set of trees that did not ever split on X_j. These values are then averaged over the subset forest that did not include X_j. A large value would imply that in trees that included X_j, the predictive capabilities were increased.

To formalize that idea, let's refer to the set of trees that did not consider X_j, $T^c_{x_j}$. Now, $T^c_{x_j} \subset R_f$, the random forest. The subset of the original data that will be tested on each tree, t, is $\bar{B}^t$. The dimensions of $\bar{B}^t$ are ν_t x p, where p is the number of predictors and $\nu \leq n$. The number of trees in $T^c_{x_j}$ is μ where $\mu \leq ntree$

Now, base variable importance is:

[3]This is intentionally vague. The level of similarity considered "similar enough" depends on the properties of the data set and there's no guarantee that suitable surrogate splits exist. **??**

$$VI_{\alpha}(X_j, R) = \sum_{t \in T^c_{x_j}} \frac{1}{\nu_t} RSS(t, \bar{B}_t)$$

However, this method poses some problems. Namely, while variable importance for random forests is more stable than for the variable importance values for CART, (this is because the model is less variable in general), it is lacking the traditional inferential capabilities of other regression models. In an effort to derive a p-value for variable importance values, Breiman 2001b, describes a *permuted variable importance* or VI_{β} that does not utilize $T^c_{x_j}$.

Algorithm 1 Permuted Variable Importance for Random Forests, VI_{β}

Fit a random forest, R_f on the dataset D estimating the model $Y \sim X_1, ..., X_p$.
for each $X_i \in X_1, ..., X_p$ **do**
  for each $T \in R$ **do**
    Calculate: $\Phi_o = \frac{1}{\nu_t} RSS(T, \bar{B}^t)$
    Permute X_i. Now find $\Phi^* = \frac{1}{\nu_t} RSS(T, \bar{B}^*_t)$
    The difference between these values, $\Phi^* - \Phi_o$, is the variable importance for X_j on t,
  end for
end for

In other words, we start with one tree in the random forest, T_u, and one variable, X_j, where $1 \le u \le ntree$ and $1 \le j \le p$. There are two subsets of the original data associated with T_u, one is the subset used to generate the tree B^t and the other is the rest of the original data set, notated as $\bar{B}^t$. We calculate the residual sum of squares for T_u on the new (to T_u) data, $\bar{B}^t$. Then we alter $\bar{B}^t$ by randomly shuffling X_j. This means that in one row of $\bar{B}^t$, the values are the same as they were was before, except the values for X_j may be interchanged with the values in other rows. Then RSS is calculated again and compared with the RSS before the shuffling took place. As each tree in the random forest is fit to a bootstrapped sample of the original dataset and splits on, potentially, on a fraction of the possible predictors, the tree-wise computation gives an estimate of the distribution of $VI_{\beta}(X_j)$.

Strobl et al Respond (2008)

Strobl et al respond to Breiman's method with one main argument: the null hypothesis implied by the permutation distribution utilized in permuted variable importance is that X_i is independent of Y **and** $X_j \notin X_1, ..., X_p$ so the null hypothesis will be rejected in the case where X_j is independent of Y but not some subset of the other predictors. As correlation among the predictors is very common in data sets that are used for random forests, this is a large problem for Breiman's method.To alleviate this difficulty, Strobl et al propose a permutation scheme under the null hypothesis that X_j given it's relationship with the other predictors is independent of Y.

Algorithm 2 Conditional Variable Importance for Random Forests, VI_{γ}

Fit a random forest, R on the dataset D fitting the model $Y \sim X_1, ..., X_p$.
for each $t \in R$ **do**
  Calculate: $\Psi_o = \frac{1}{\nu_t} RSS(t, \bar{B}^t)$
  for each $X_i \in X_1, ..., X_p$ **do**
    Select $Z \in X_1, ..., X_{i-1}, X_{i+1}, ..., X_p$ to condition on when permuting X_j
    Use the cutpoints on each variable in Z to create a grid on X_j
    Permute X_j with respect to this grid
    Now find $\Psi^* = \frac{1}{\nu_t} RSS(t, \bar{B}^*_t)$
    The difference between these values, $\Psi^* - \Psi_o$, is the variable importance for X_j on t,
  end for
end for

This method is fairly similar to permuted variable importance, but there are a few key differences. Given a tree T_u and a variable X_j, first we find the out of bag RSS, then we permute. In this case, however, our permutations or shuffling of X_j is not always done blindly. If X_j has no (or low) empirical correlation with each of its fellow predictors, then X_j is shuffled exactly like in permuted variable importance. If that is not the case, then we select the set, Z, of the predictors with the strongest empirical correlation [4] to X_j. Recall that our tree T_u contains many nodes, and each node contains a subset of the data along with a split that determines the subsets of the daughter nodes. We feed the out of bag sample for T_u into T_u and look at all the subsets of data in nodes that split on a predictor in Z. This time when we shuffle X_j it will only be in these subsets. The union of these subsets is called $\bar{B}_t^*$ and is used to calculate the permuted RSS.

[4] **??** recommends constructing the set Z from prior information about the data or as the set of predictors where each one has empirical correlation greater than or equal to .2 with X_j.