Jeremiah J. Nieves

CLASSIFICATION USING DECISION TREES

Decision trees are quite popular in part to their open and transparent processes and their intuitive interpretation. They are referred to as “trees” due to the beginning node, or “root” node, branching off into smaller nodes referred to as decision nodes, and ultimately terminal or “leaf” nodes. These trees result in “rules” which can be applied to other datasets in order to classify them or predict outcomes, if the decision tree is used as a regression. Decision trees can utilize multi-variate data that is both categorical and continuous. A generalized diagram of a decision tree is shown in Figure 1.

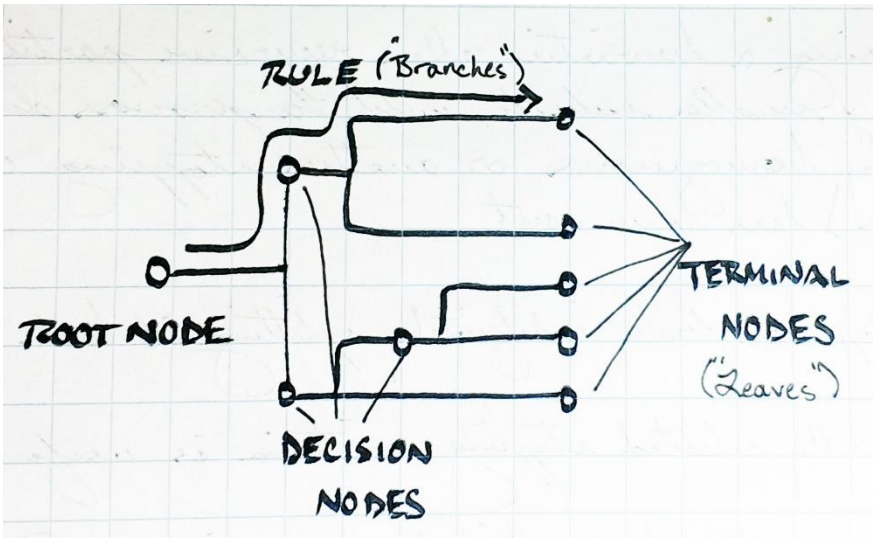

Figure 1. General schematic of a decision tree

How a Decision Tree Works

Decision trees are built using a heuristic process called *recursive partitioning* which involves splitting a dataset into smaller and smaller subsets until the process determines the subsets are sufficiently homogenous or some other *stopping criterion* has been met (e.g. number of terminal nodes). While there is nothing to stop you from indefinitely splitting (i.e. growing your decision trees) it leads to a loss of generalizability of the classifier or regression that has been built, i.e. the tree over-fits the data. This is where having carefully informed and selected selection criterion is useful in obtaining or “growing” the best tree.

So how does a decision tree split the data? To create a decision node, the tree chooses an attribute/variable of the data to split the data upon. How the decision tree chooses this attribute varies by the algorithm utilized by a given decision tree, but generally it is based upon splitting the data in a manner which results in two subsets that are as homogenous, i.e. composed of a single class as much as possible. This homogeneity is referred to as *purity* of the subset and can be measured in a variety of manners such as entropy.

Similar to organic trees, decision trees often require pruning for optimal "health." For decision trees there are two main types of pruning: pre-pruning and post-pruning. Pre-pruning involves stopping the model prior to it overfitting the training data it is being built upon. Pre-pruning often takes the form of specifying the maximum numbers of terminal nodes or specifying a homogeneity of subsets at which to stop growing the tree. Pre-pruning does run the risk of stopping the process prior to discovering all relevant data structures. Post-pruning involves growing the tree large in order to ensure discovery of all relevant data structures in the data. This "large" tree is then pruned or "cut back" to an appropriately identified level, via various statistical methods, in order to increase the generalizability of the tree and decrease its overfitting of the data. Post-pruning is considered more effective than pre-pruning.

IMPROVING DECISION TREE ACCURACY – ENSEMBLE METHODS

There are a few ways in which the accuracy of a decision tree can be improved; these methods are generally referred to as ensemble methods because they combine multiple decision trees that are considered "weak learners," i.e. they are only slightly correlated with the training data, in order to create a "strong learner." Ensemble methods utilize a training data set to build a number of models (e.g. decision trees) by using an *allocation function* to determine how much of the training data each model receives. These multiple models are then combined through a *combination function* which determines how best to resolve disagreements amongst the models' predictions (e.g. through voting, weighting, etc.). The output is an *ensemble model*. Ensemble methods differ primarily in their allocation functions and their combination functions as well as the method used to create the multiple models. The benefit of ensemble methods is that generalizability is increased, performance on extremely large or small datasets is improved, and the ability of the method to "understand" or model difficult learning tasks is more nuanced and effective. Additionally, the ensemble model produced is able to synthesize or predict data from very specific and distinct domains.

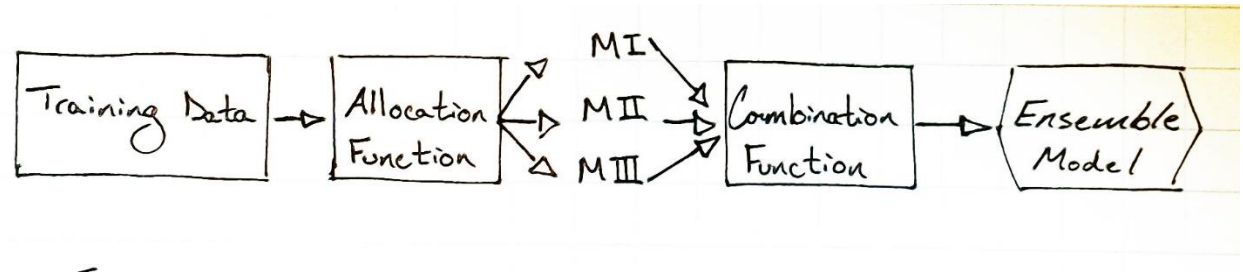

Figure 2. General schematic of ensemble methods

Bagging (Bootstrap Aggregating)

Bagging generates a set of models using a simple learning algorithm, but provides the chosen algorithm with a bevy of training datasets by bootstrap sampling the entire original

training dataset with replacement. This is contrary to traditional selection of training data sets in which at least 10 percent of the original training dataset is set aside for validation. The benefit is that you have more data from which to train each model, with each model having equal probability of accessing the entire training set, i.e. no training data is lost. This also increases the resilience of the resulting ensemble model to any noise in the data as most observations will be included in more than one of the trained models. Bagging performs well as long as the chosen learning algorithm is considered *unstable*, that is the model outcome will change significantly with only a small change in the input data. This is necessary to maintain the ensemble's diversity even though the constituent models of the ensemble have similar training data. Each learner is "combined" through the combination function in which each learner is given an equally weighted vote as to the correct classification or prediction. The downside of bagging is related to the increased number of training sets which give it its strength: increased number of training sets means an increased number of decision trees to grow which means increased computational resource costs.

Adaptive Boosting

Adaptive boosting is another ensemble method in which weak learners are grown adaptively and the training data is sourced similar to bagging in the sense that the entire training set is resampled. With each learner grown, the observations in the training dataset that were misclassified by the learner are weighted to increase the chance that they are used to train the next learner; this is the adaptive part of adaptive boosting and one of the ways in which it differs from bagging. This adaptive sampling of training datasets results in learners/models that are complimentary to each other's' weaknesses and strengths. Similar to bagging, adaptive boosting uses a combination function in which each learner/model is given a vote as to the correct classification or prediction, however, unlike bagging, the vote of each learner is weighted by their performance or "accuracy" on the training dataset. Therefore, the best or "strongest" learners have a larger influence on the final decision.

Random Forests

Random forests (RFs) combine the principles of bagging, ensembles of decision trees, and random section of the training data. RFs combine the versatility and power of those principles and its "signature" feature is the random selection of two-thirds of the total training data for a given model with the remaining third of the data used as the out-of-bag (OOB) data. Once a single tree upon which the "forest" is built, it is used to classify or regress the OOB data and the number of OOB elements that were misclassified are recorded. In growing a single tree, the RF takes a random subset of the variables to use in the decision of the split of every node (Brieman 2001). No pruning of trees is done with RF which is an added strength given that choice of pruning method can be time consuming and heavily influence the results of a tree (Pal and Mather 2003).

At the end of the RF process, the proportion of total errors to the total number of OOB observations is taken to calculate the OOB error which serves as an unbiased estimate of the

generalization error (Brieman 2001). On average, each data observation will be a part of the OOB 36 percent of the time (Liaw and Wiener 2002). RFs are robust to noise and overtraining, due in part to the fact that the generalization error converges with the number of trees grown due to the "Law of Strong Numbers" (Feller 1968; Brieman 1996; Briem et al. 2002; Pal and Mather 2003; Chan and Paelickx 2008). Additionally, the RF process provides a measure of the relative importance of each variable used in the RF. This is done by replacing a given variable with random data, effectively "removing" the variable from, the model, while holding all other variables constant and then measuring the decrease in the RF accuracy by one of several measures such as the OOB error, Gini Index, or the percent increase in the mean squared error (PerIncMSE) (Brieman 2001). Variables that do not decrease the accuracy with their removal are removed from the model, i.e. they are not selected for the final model output. The overall RF process is shown in Figure 3. Given that a RF is constructed from hundreds of individual decision trees, the transparency of how a RF classifies a single observation is not exactly transparent, as it would be with a single decision tree. However, the importance measures give an approximation as to the general influence of variables in the decisions.

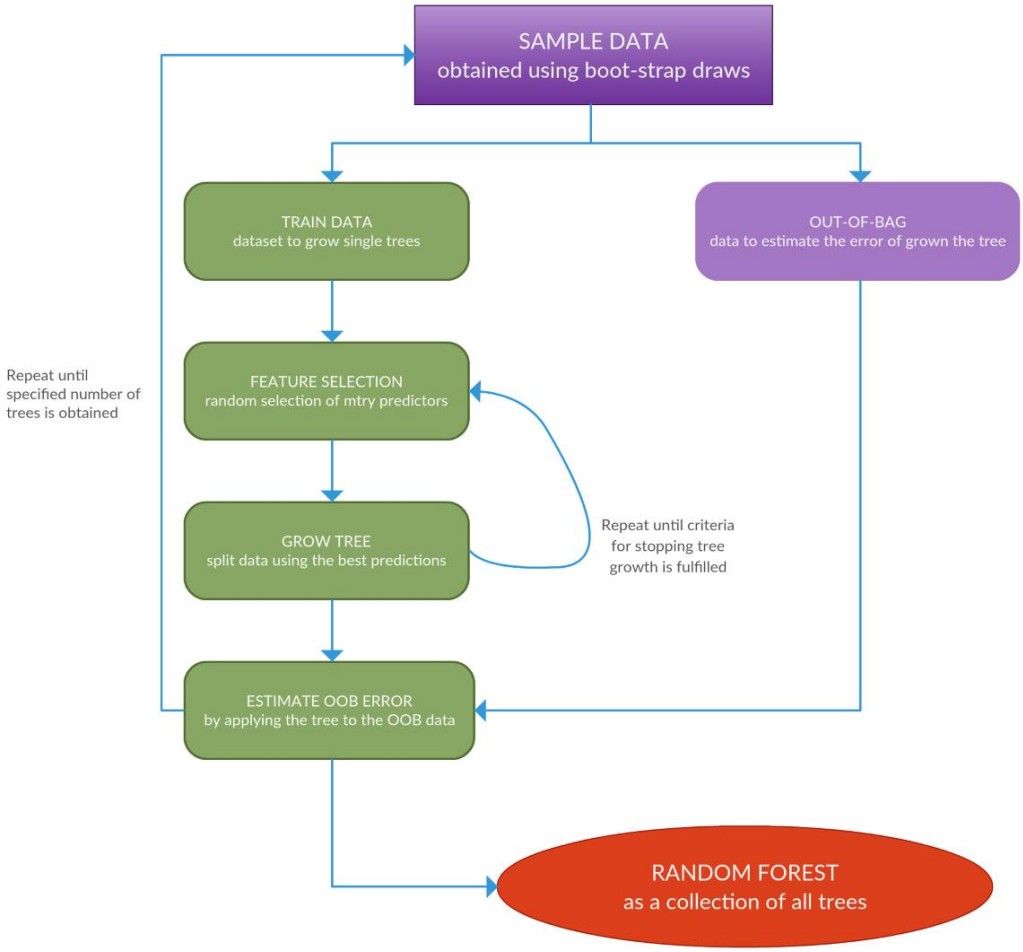

Figure 3. General process of building a random forest.

As related to user input and parameterization, RFs do not require much, which is an attractive feature. RFs only require the *k* number of trees to grow, typically 500 trees are a standard amount as the model does not over fit when run past the point of convergence (typically 100 trees), and the *m* number of variables used in each node split (Brieman 2001; Liaw and Wiener 2002). The number of variables *m* is important as it exhibits a positive relationship with the correlation of individual trees to each other and a negative relationship to the strength of an individual tree as a classifier (Brieman 2001). Forunately, the *randomForest* package in the R statistical package contains a function *tuneRF* which optimizes the selection of *m* based upon the OOB error rate (Liaw and Wiener 2002).

Summary of Random Forest

Strengths of RFs include:

* All-purpose model that performs well on most problems
* Handles noisy or missing data in addition to categorical and continuous data
* Selects only the most important variables

* Can handle extremely large datasets
* Can be used for either classification or regression
* Requires little to no tuning of parameters

Weaknesses of RFs include:
* Not as interpretable as a single decision tree since it is composed of hundreds of decision trees
* Can be computationally expensive

REFERENCES

Breiman, L. 1996. Bagging predictors. Machine Learning 24(2): 123–140.

---. 2001. Random Forests. *Machine Learning,* 45**,** 5-32.

Briem, G.J., Benediktsson, J.A., & J.R. Sveinsson. 2002. Multiple classifiers applied to multisource remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing* 40(10): 2291–2299.

Chan, J.C.-W., Paelinckx, D., 2008. Evaluation of Random Forest and Adaboost treebased ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery. Remote Sensing of Environment 112(6): 2999–3011.

Liaw, A. & Wiener, M. 2002. Classification and regression by randomForest. *R News* 2(3): 18-22.