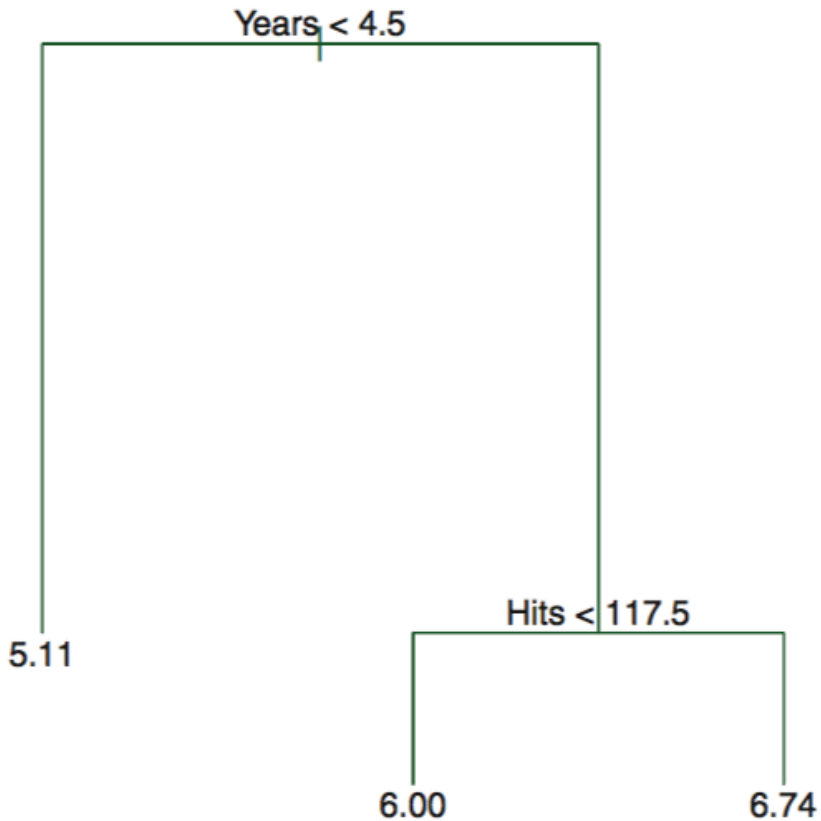
Lecture 19: Tree-Based Methods

Nayel Bettache

Department of Statistics and Data Science, Cornell University

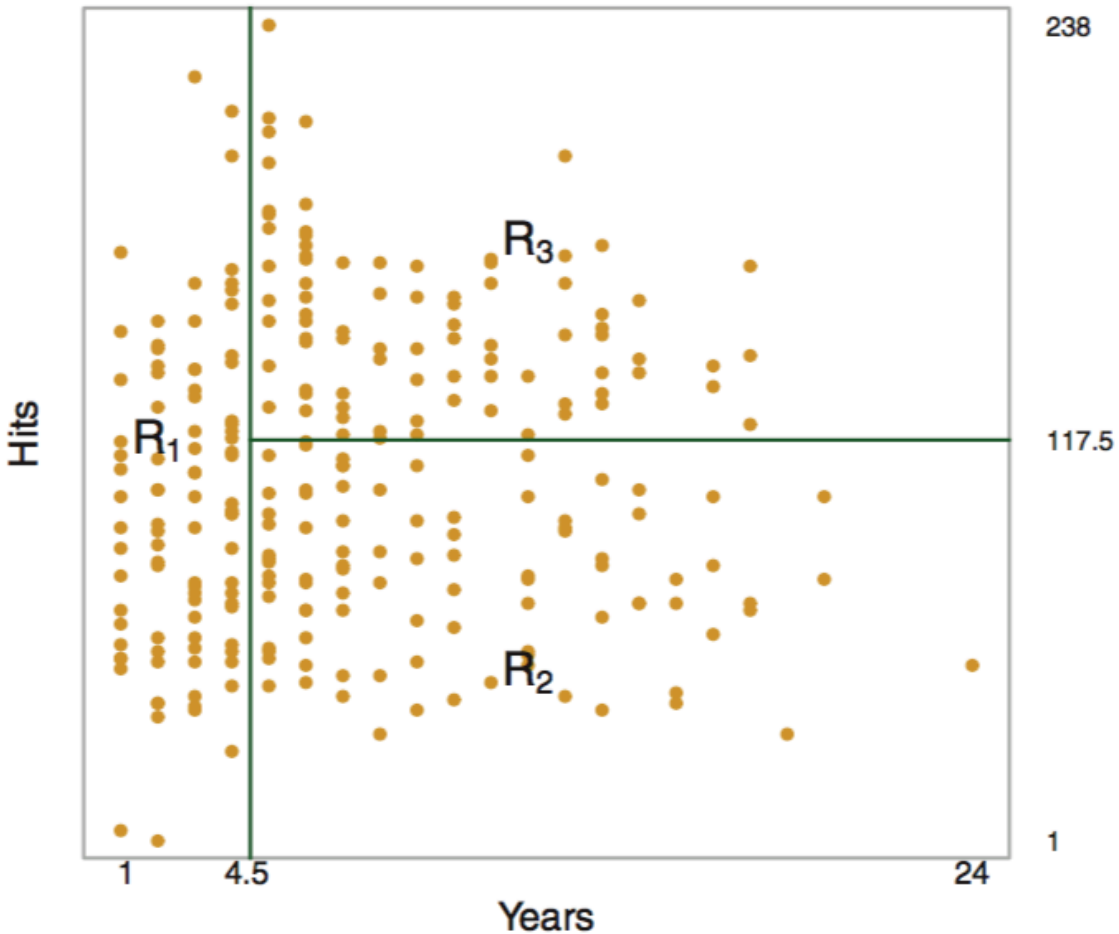
Review: What does Decision Tree Look Like?



At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. The number in each leaf (external node) is the mean of the response for the observations that fall there.

Review: The three-region partition for the Hitters data set

Overall, the tree stratifies or segments the players into three regions of predictor space: $R_1 = \{X | Years < 4.5\}$, $R_2 = \{X | Years >= 4.5, Hits < 117.5\}$, and $R3 = \{X | Years >= 4.5, Hits >= 117.5\}$.



Review: How to Build a Regression Tree?

- Step 1: We divide the predictor space–that is, the set of possible values for $X_1, X_2, ..., X_p$–into J distinct and non-overlapping boxes, $R_1, R_2, ..., R_J$. We use **recursive binary splitting**.
- Step 2: For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.
- The recursive binary splitting usually leads to a deep tree, which overfit the data. We use **tree pruning** idea to find the best subtree.

Trees versus Linear Models

- The linear model says

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j,$$

  whereas the regression tree says

$$f(X) = \sum_{m=1}^{M} c_m \cdot I(X \in R_m).$$

- If the relationship between the features and the response is well approximated by a linear model, then an approach such as linear regression will likely work well. If instead there is a highly non-linear and complex relationship between the features and the response, then decision trees perform better.

- The relative performances of tree-based and classical approaches can be assessed by estimating the test error, using either cross-validation or the validation set approach.

Advantages and Disadvantages of Trees

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.
- Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.
- Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the prediction.

However, by aggregating many trees (using bagging, random forests, and boosting), the predictive performance of trees can be substantially improved.

Bagging

- **Bootstrap aggregation**, or **bagging**, is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.
- Recall that given a set of n independent random variables $X_1, \ldots, X_n$, each with variance σ^2, the variance of the average of $X_1, \ldots, X_n$ is given by σ^2/n.
- In other words, averaging a set of random variables reduces variance.

Bagging

- Instead, we can bootstrap, by taking repeated samples from the (single) training data set.
- In this approach we generate B different bootstrapped training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point x. We then average all the predictions to obtain

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

This is called **bagging**.

Bagging Trees

- To apply bagging to regression trees, we simply construct B regression trees using B bootstrapped training sets, and average the resulting predictions. These trees are grown deep, and are not pruned. Hence each individual tree has high variance, but low bias. Averaging these B trees reduces the variance.
- Bagging can be also applied to classification trees. For a given test observation, we can record the class predicted by each of the B trees, and take a majority vote. The overall prediction is the most commonly occurring majority class among the B predictions.

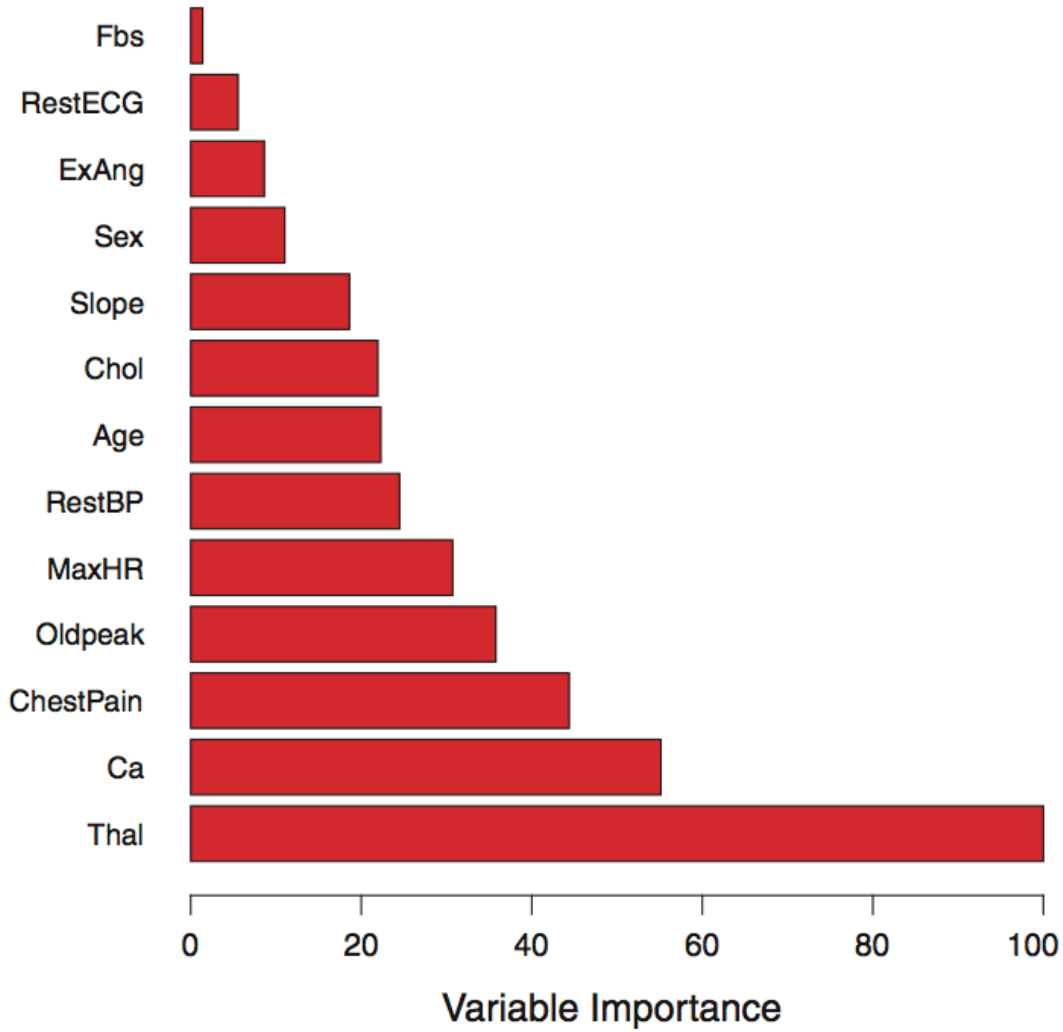
Out-of-Bag Error Estimation

- There is a simple way to estimate the test error of a bagged model, without the need to perform cross-validation or the validation set approach.
- Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. (One can show that on average, each bagged tree makes use of around two-thirds of the observations).
- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the **out-of-bag (OOB) observations**.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB.
- This leads to a single OOB prediction for the ith observation.
- We compute the averaged squared difference between the OOB prediction and the training data. This is the OOB MSE (for a regression problem).

Variable Importance Measures

- Bagging typically reduces variance (i.e. improve accuracy) over prediction using a single tree.
- However, it can be difficult to interpret the resulting model.
- One can obtain an overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Gini index (for bagging classification trees).
- In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. In bagging classification trees, we replace RSS by the Gini index.
- A large value of decreased RSS (or Gini index) indicates an important predictor.
- A graphical representation of variable importance is easy to draw.

Heart Data



Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.

Random Forests

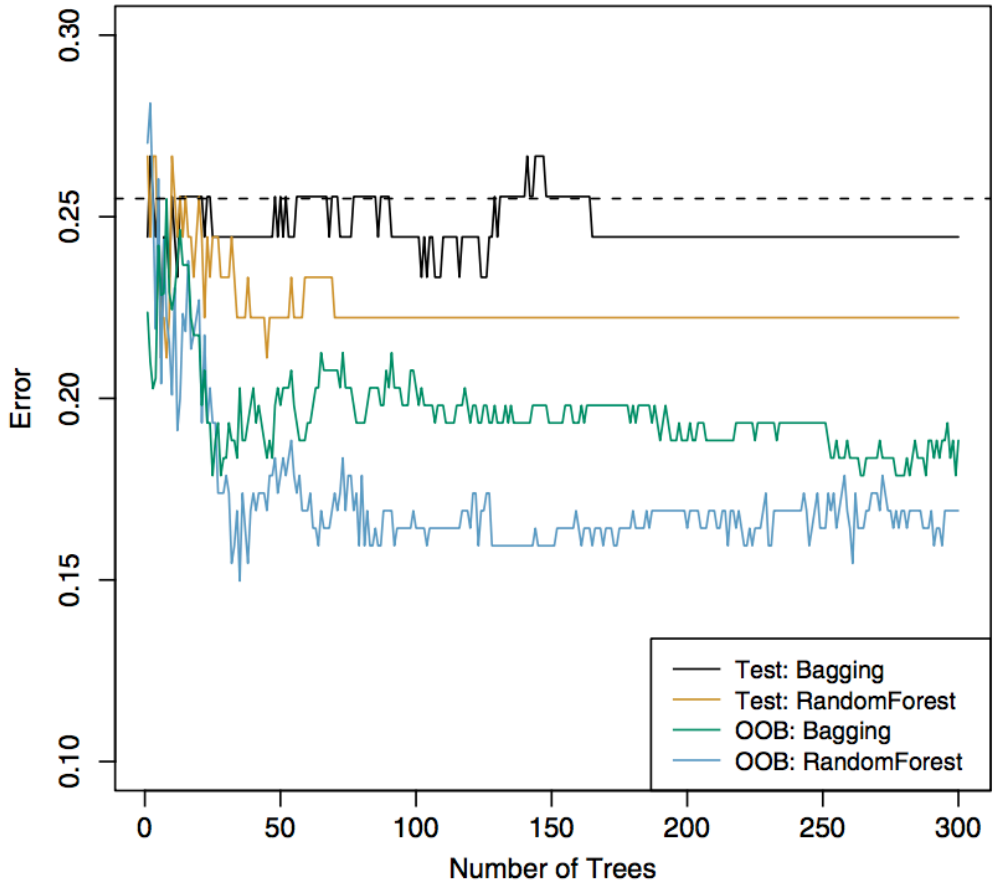
- **Random forests** provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, a random selection of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.
- A fresh selection of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ – that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data).

Why does Random Forest Work?

- In other words, in building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors.
- Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.
- Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split. All of the bagged trees will look quite similar and the resulting prediction will be highly correlated. Unfortunately, averaging many highly correlated quantities does not reduce variance.
- Random forests overcome this problem by forcing each split to consider only a subset of the predictors. Therefore, on average many splits will not even consider the strong predictor, and so other predictors will have more of a chance.
- We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable.

Heart Data

Random forest with $m = p$ is just bagging!



The test error (black and orange) is shown as a function of B, the number of bootstrapped training sets. Random forests were applied with $m = \sqrt{p}$. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which is considerably lower.

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. Here we restrict our discussion of boosting to the context of decision trees.
- Recall that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.
- Boosting works in a similar way, except that the trees are grown sequentially: each tree is grown using information from previously grown trees.
- Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set.

Boosting

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:

   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).

   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$

   (c) Update the residuals,

   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

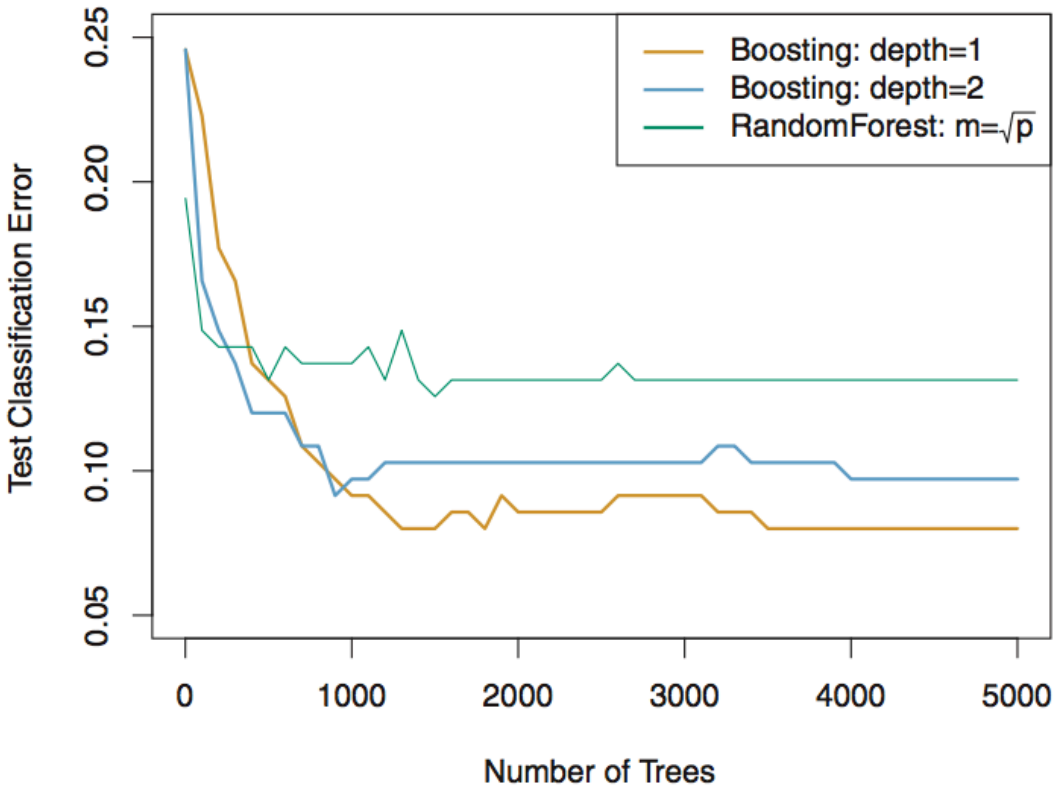
Boosting

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead **learns slowly**.
- Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals.
- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

Tuning parameters for Boosting

- The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
- The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001.
- The number of splits d in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split and resulting in an additive model. More generally d is the **interaction depth**, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

Gene Expression Data



Results from performing boosting and random forests on the 15-class gene expression data set in order to predict cancer versus normal. The test error is displayed as a function of the number of trees. For the two boosted models, $\lambda = 0.01$. Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant.

Summary

- Decision trees are simple and interpretable models for regression and classification
- However they are often not competitive with other methods in terms of prediction accuracy
- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting trees.
- The latter two methods – random forests and boosting – are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.

Some Final Conclusion Remarks

- Some recommended (more advanced) textbooks or reading materials
  - The Elements of Statistical Learning (`http://statweb.stanford.edu/~tibs/ElemStatLearn/`)
  - Statistical Learning from a Regression Perspective (`http://www.springer.com/la/book/9780387775005`)
  - Statistical Learning with Sparsity: The Lasso and Generalizations (`http://web.stanford.edu/~hastie/StatLearnSparsity/`)
- For unsupervised learning, you may want to take CS 4786/5786.