New York University: Machine Learning in Economics

Lecture 7: GAMs, Trees and Random Forests

Dr. George Lentzas

Decision Trees

The Basics

- Tree-based methods for regression and classification involve segmenting the predictor space using simple splitting rules into a number of regions and using the average or most popular response for prediction in regression and classification respectively.
- A crucial advantage of tree-based methods is their simplicity and interpretability.
- Their main drawback is that they are not very effective for prediction. Their prediction accuracy can however be greatly improved using methods such as *bagging*, *random forests* and *boosting*.
- All these approaches involve fitting a large number of trees that are then combined in some "optimal" way.

Decision Trees

Regression Trees

- Lets begins with a simple example from the "Hitters" data. We try to predict a baseball player's Salary based on two predictors: the Years played and the number of Hits that he made the previous year.
- The figure below shows a simple decision tree. It consists of a series of splitting rules; the first split assigns observations with $Years < 4.5$ to the left branch. The predicted salary is then the mean response value for such players.
- The rest of the players are further split according to the number of hits into two groups, under and over 117.5 hits.
- The regions R_1, R_2 and R_3 are the "terminal nodes" ("leaves"), the points of predictor splits are the "internal nodes" and the segments that connect the nodes are the "branches".

Decision Trees

Regression Trees

Figure: A simple Regression Tree.

Decision Trees

Regression Trees

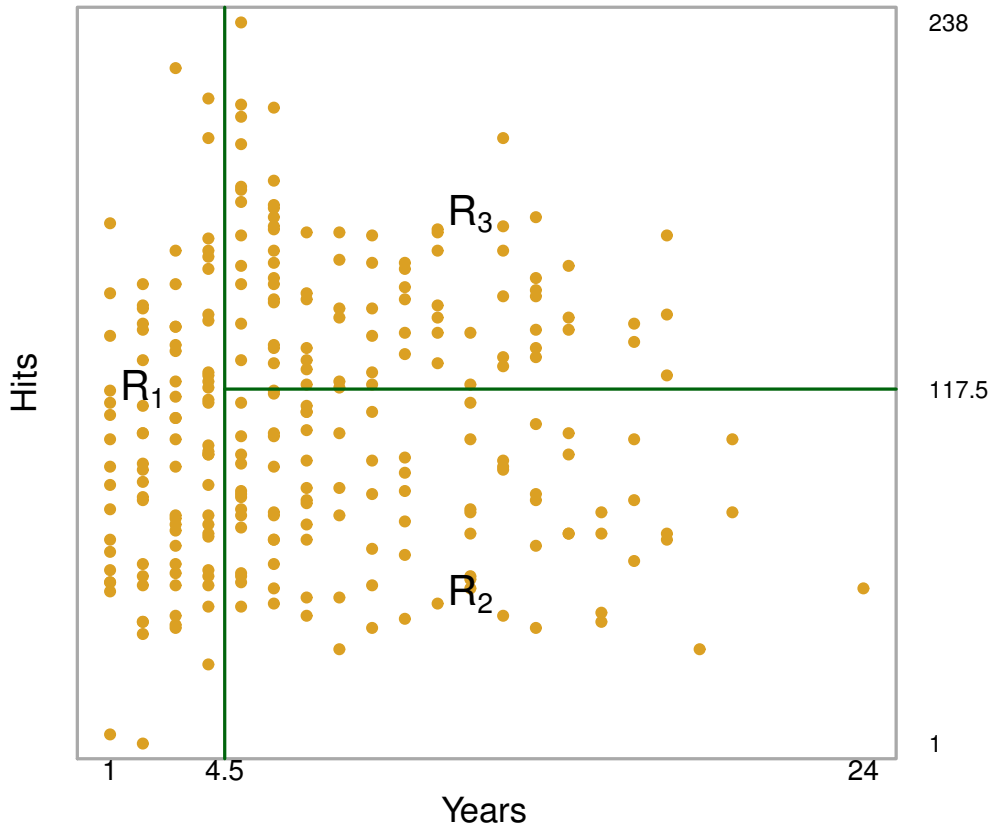

Figure: A simple Regression Tree.

Decision Trees

Regression Trees

- This nice graphical representation also has a natural interpretation.
- The most important factor in determining Salary is Years played; players with less experience earn a lower salary. Among the less experienced players the number of hits does not seem to play an important role. But among experienced players the number of hits plays an important role in determining salary and players with more hits earn a higher salary.
- So we want to divide the predictor space (i.e. the set of possible values for $X_1, X_2, \ldots, X_p$) into J distinct regions $R_1, R_2, \ldots, R_J$. How can we do that?
- In theory these regions can have any shape! But in practice we choose to divide the space into "rectangles" and try to find the boxes $R_1, \ldots, R_J$ that minimize the RSS

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2$$

where $\hat{y}_{R_j}$ is the mean response of the training observations within the j box.

Decision Trees

Regression Trees

- It is computationally infeasible to consider every possible partition. So we take a top-down, greedy approach known as "recursive binary splitting".
- We first select the predictor X_j and the cut-point s such that splitting the predictor space into two associated regions leads to the greatest reduction is RSS.
- So for any j and s we define:

$$R_1(j,s) = \{X|X_j < s\},\ R_2(j,s) = \{X|X_j > s\}$$

and seek j, s that minimize

$$\sum_{i:x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{i:x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2$$

where $\hat{y}_{R_1}$ is the mean response for the training observations in $R_1(j,s)$ and similarly for $\hat{y}_{R_2}$.

Decision Trees

Regression Trees

- Then we repeat the same process, looking for the best predictor and cut-point pair to further split the data in order to minimize RSS within each of these regions. At each step we are making a binary split at the best predictor, cu- point pair.
- We continue this process until some stoping criterion is reached, usually a maximum number of observations for each region. Once the regions have been created, we predict the response of a test observation by the mean of the training observations of the region in which the test observation belongs to.
- The figure below shows a non trivial tree that results from the Hitters data.

Decision Trees

Regression Trees

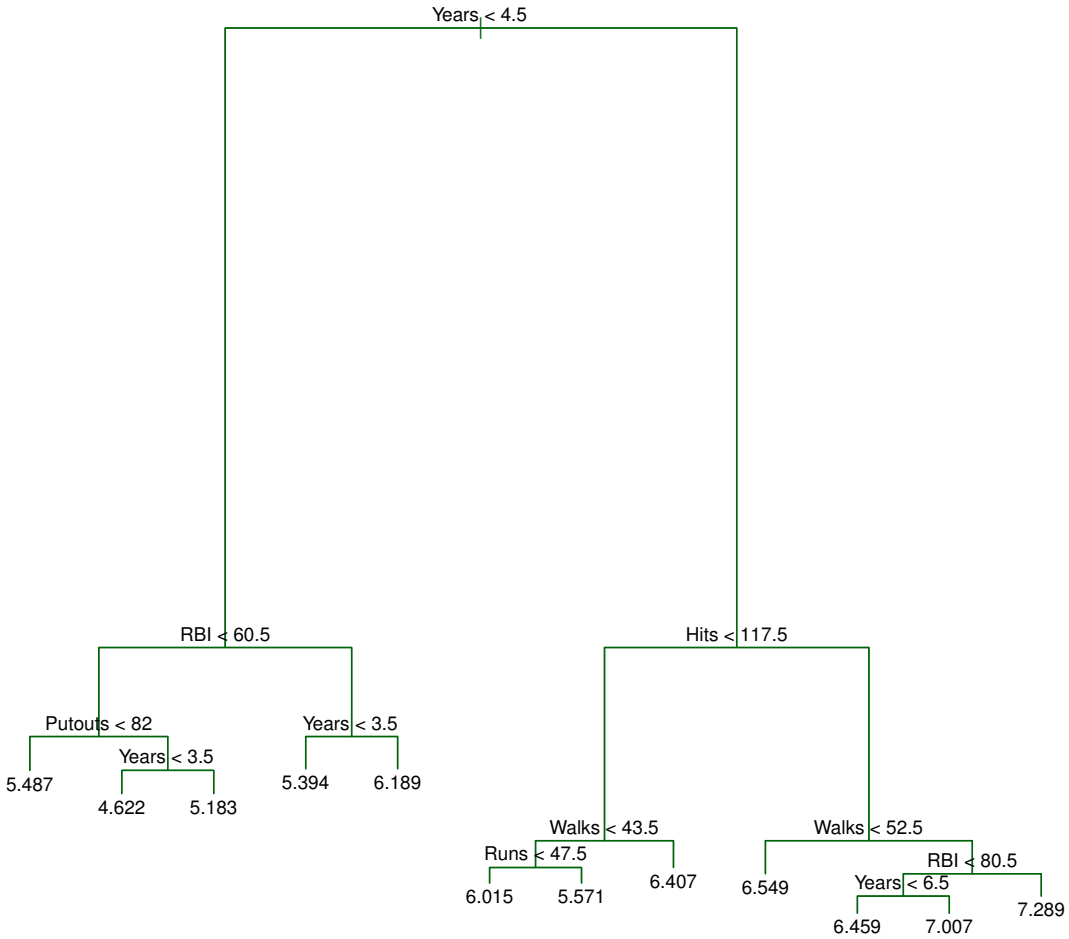

Figure: A Bigger Regression Tree.

Decision Trees

Tree Pruning

- The above process is likely to overfit the data and cause poor out of sample performance. To deal with this we engage in a strategy called "pruning".
- We grow a very large tree T_0 and then prune it to obtain a "best" subtree that is one that leads to the lowest (or low) test error which we estimate using cross-validation.
- Since the number of possible subtrees is very large instead of considering every possible subtree we instead look at a sequence of trees indexed by a non-negative tuning parameter. Specifically, for each value of the tuning parameter α we have a subtree T such that

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible. Here $|T|$ is the number of terminal nodes of the tree T.

Tree Pruning

Complexity Pruning

- This is called cost complexity pruning and the parameter α is typically estimated using cross validation. It controls the trade-off between the subtree's complexity and the fit to the training set.
- Note that although the cross validation error is computed as a function of α it is common to display it as a function of the number of terminal nodes T.
- The graph below shows the CV error as a function of the tree size (inversely proportional to the tuning parameter). The minimum CV error occurs at a pruned tree of size three.

Tree Pruning

Complexity Pruning

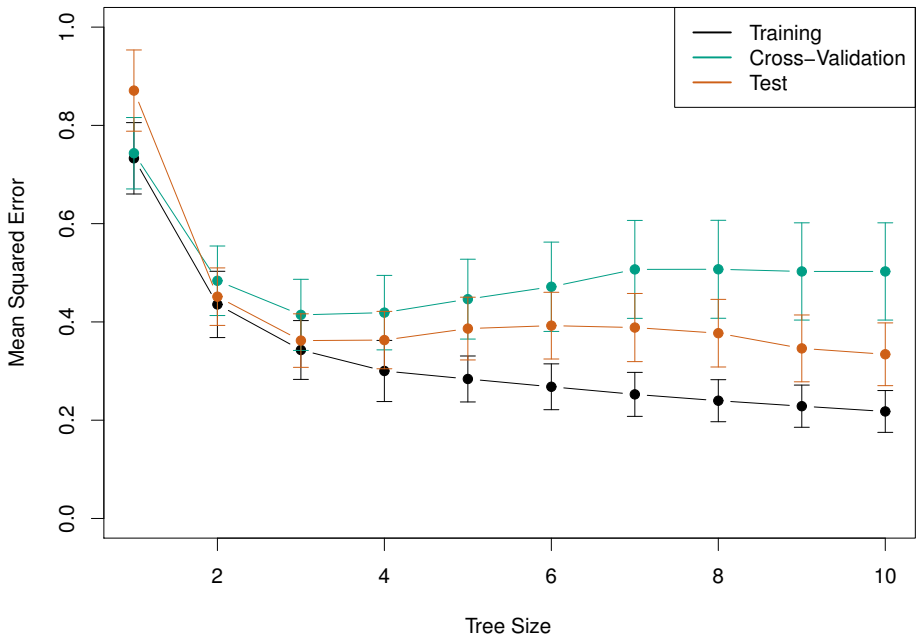

Figure: Tree Pruning by CV

Classification Trees

Classification

- So far we have been looking at regression trees; a classification tree works in a very similar fashion, with the crucial difference being that we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.
- Analogously to RSS we can look at the classification rate (i.e. the fraction of the training observations in the region that do not belong to the most common class) as the criterion for making binary splits.
- However, out turns out that the classification error is not a sufficiently sensitive metric for tree-growing and hence two other measures are commonly used. As a simple example consider a two class problem with 400 observations in each class $(400, 400)$and compare two splits one creating nodes $(3.00, 100)$ and $(100, 300)$ while the other created nodes $(200, 400)$ and $(200, 0)$. Both splits produce a misclassification error of 25% but the second split is probably preferable as it produces a "pure" node.

Classification Trees

loss functions for classification probs

Gini and Cross Entropy

- The *Gini Index* given by

$$G = \sum_{k=1}^{K} \hat{p}_{mk} \left(1 - \hat{p}_{mk}\right)$$

think of the variance of a bernoulli random var

where $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class and is a measure of total variance across the K classes (and of node purity).

- An alternative is cross entropy defined by

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log\left(\hat{p}_{mk}\right).$$

Classification Trees

Gini and Cross Entropy

- Either the Gini Index or cross entropy can be used to evaluate the quality of a split.
- However, the classification error is typically preferred for pruning since we are typically interested in classification accuracy!
- In the figure below we see at the top the unpruned tree, the CV error on bottom left and the resulting pruned tree in bottom right.
- Notice the counter-intuitive result that some of the splits yield two terminal nodes with the same predicted value. This is because we use a splitting criterion that focuses on node purity. ???

Trees vs Linear Models

Trees vs. Linear Models

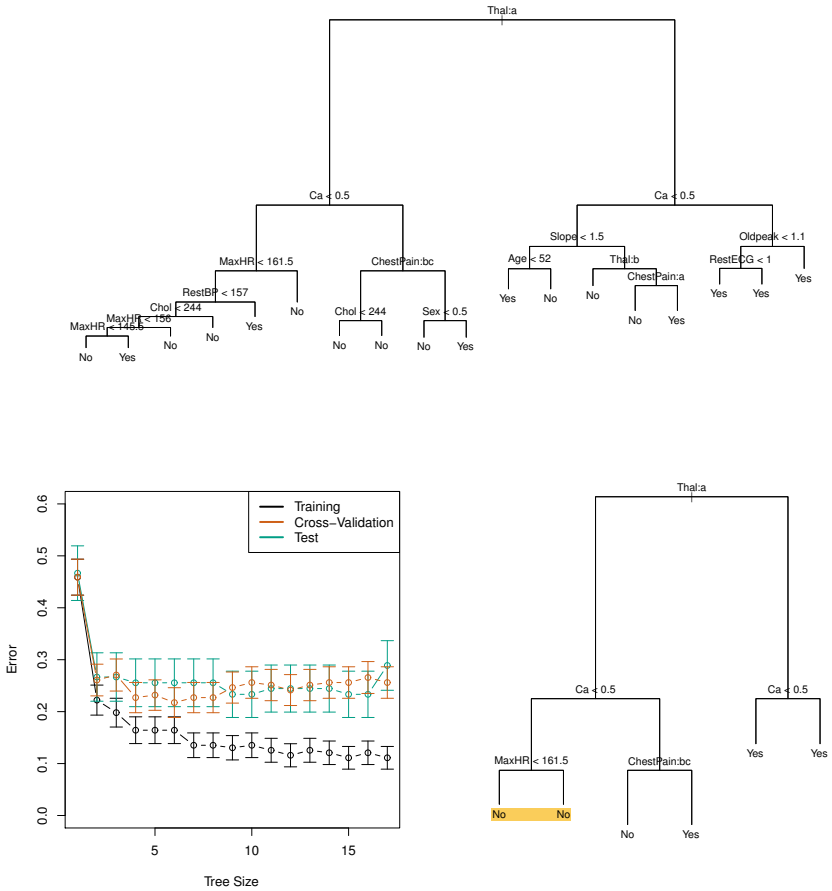

Figure: Classification Tree Example

Further Issues

Trees vs Linear Models

- How do decision trees compare with regression trees?
- In theory the answer depends on the linearity of the true decision boundary.
- In practice other considerations (like interpretability and visualization) might come into play. Trees are easy to explain and present graphically. They are also intuitive as they mirror human decision making.
- For predictive purposes trees do not perform as well as other methods, mostly because of high variance. In addition they lack smoothness in the prediction space which may degrade performance in a regression setting.

Further Issues

Decision Boundary Example

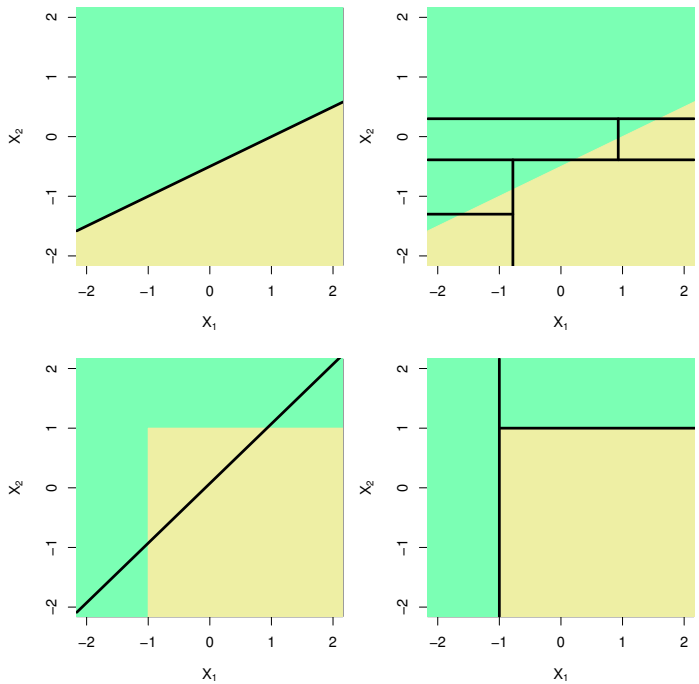

Figure: Linear vs.Non-Linear Decision Boundary

Further Issues

Categorical Predictors

- When splitting a predictor having q possible unordered values the computational burden becomes prohibitive for large q. The solution is to order the predictor classes according to the proportion falling in outcome class 1 and split the predictor as if it was an ordered predictor.
- Keep in mind that trees tend to favor categorical predictors with many levels; large qs can lead to over-fitting and such variables should be avoided.

Other Tree Building Algorithms

- The discussion so far covers the so-called CART methodology for building a tree. There are other popular methodologies such as ID3 and its later versions such as C5.0.

Bagging

Bootstrapping and Bagging

- Simple decision trees suffer from high variance: the same method applied to different data sets can give pretty different results.
- Bagging is a general purpose procedure for reducing the variance of a method by averaging.
- The general idea is to generate bootstrapped samples, fit the predictor at each sample and average all the predictors to obtain a new predictor with lower variance.

Bagging

Bootstrapping and Bagging

- To apply bagging to regression trees we construct B regression trees using B bootstrapped training sets, constructed by taking repeated samples with replacement from the training data set and average the predictions.
- These trees are grown deep, with each tree having a high variance; however, the averaging reduces the variance, keeping the low bias property of the deep trees intact.
- The number of trees B is not critical here as a high B will not lead to over-fitting. Typically $B = 100$ or larger is used.
- With bagging we do not actually need to do CV. To each bagged tree correspond some training set observations that are not used (the "out of the bag" observations).

Bagging

Bootstrapping and Bagging

- We can predict the response for the ith observation using all the trees for which it is an out of the bag observation and averaging (or majority voting) the predicted responses. The resulting out of the bag error is a valid estimate of the test error rate for the bagged model and for large B nearly equivalent to the leave-one-out cross validation error rate.
- The downside of bagging s that we loose some of the (graphical) interpretability of the model, although we can still obtain an overall summary of the importance of the predictors by averaging the effect of each predictor in reducing RSS (or Gini Index) over the B trees. We can see an example in the graph below.

Trees vs Linear Models

Decision Boundary Example

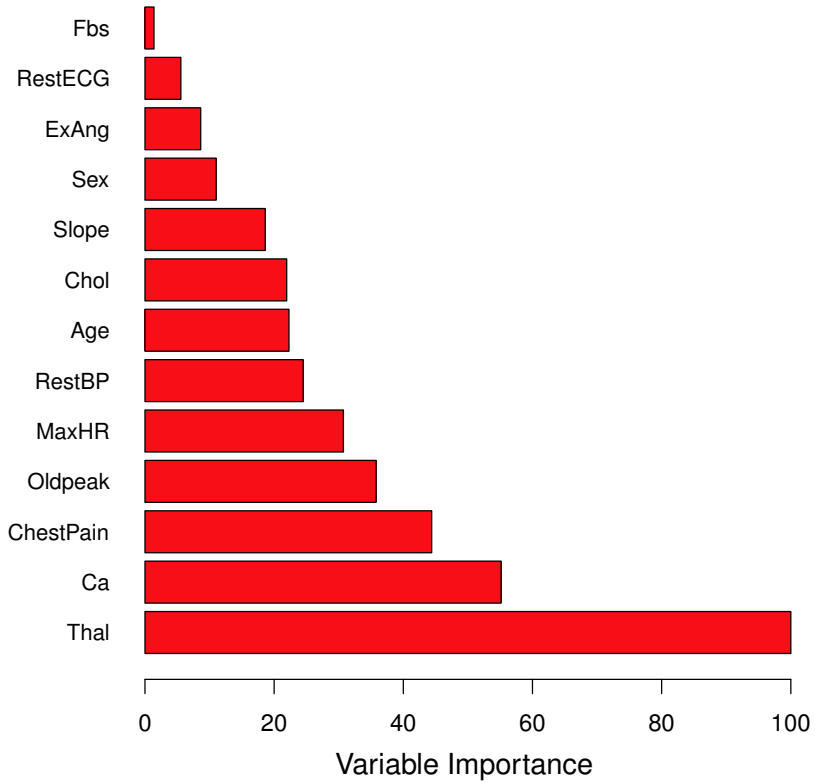

Figure: Variable importance

Random Forests & Boosting

Random Forests

- Random forests build on the idea of bagging and improve it by de-correlating the bagged trees. As with bagging, we build a number of decision trees on bootstrapped training data.
- The motivation here is that if we have one strong predictor in the data set, most or all of the bagged trees will look similar, having that predictor as the top split. As averaging highly correlated quantities does not lead to a large reduction in variance, bagging will not be very effective in this case.
- Random forests overcome this problem by forcing each split to consider only a randomly chosen subset of predictors, in a way de-correlating the trees.

this is the defining characteristic of Random Forests

Random Forests & Boosting

Random Forests

- The key variable here is the number of predictors m allowed to be considered at each split. Typically this is just set at

$$m = \sqrt{p}.$$

- This means that in a random forest, at each split the model is typically not even allowed to consider a majority of the predictors.
- This is particularly useful in the case of a large number of correlated predictors.
- As with bagging, random forests will not overfit the data if we increase the number of bootstrapped samples B
- The example below illustrates this; note that the case $m = p$ is bagging.

Random Forests & Boosting

Random Forests & Boosting Example

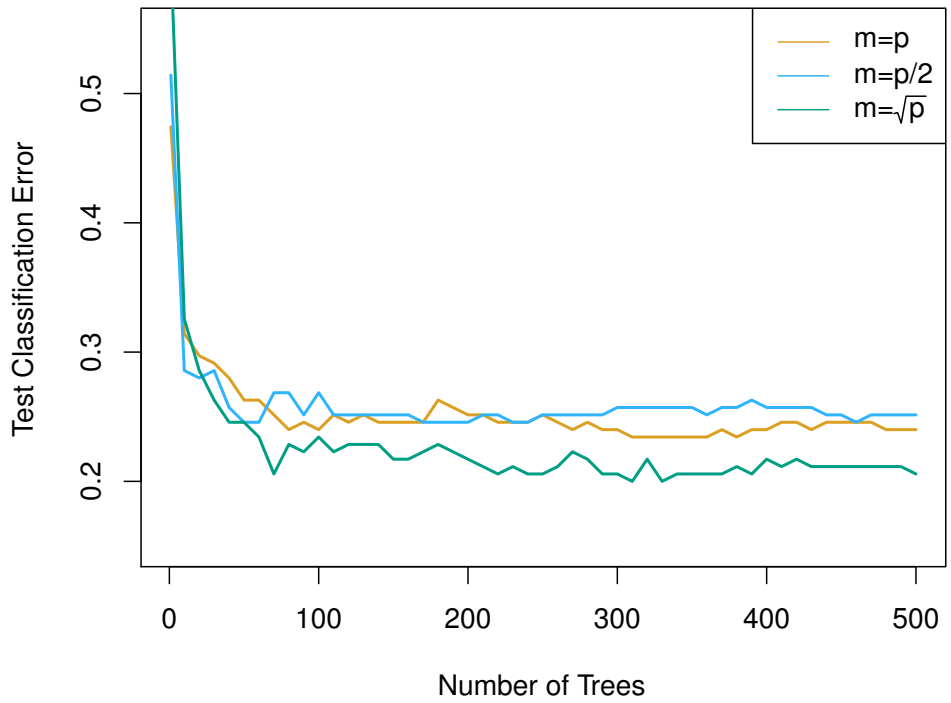

Figure: Random Forest vs Bagging

Boosting

The Algorithm

- Boosting is another general approach for improving prediction performance that can be applied to many statistical methods.
- Here we will focus on the application of boosting to regression trees, but these ideas generalize to classification trees and beyond.
- Boosting works by sequentially fitting trees to the residuals from the previous tree. Unlike bagging there is no bootstrapping of the sample here; instead each trees is fit to a modified version of the data.

Boosting

The Algorithm

- We start with residuals $r_i = y_i$ and $\hat{f}(x) = 0$ for all i in the training set and for b in $1, \ldots, B$ we repeat:
  1. Set the training data to be (X, r).
  2. Fit a tree f^b with d splits to (X, r).
  3. Update $\hat{f} = \hat{f} + \lambda \hat{f}^b$.
  4. Update the residuals $r_i = r_i - \lambda \hat{f}^b$.
  5. Output the "boosted" model $\hat{f} = \sum_{b=1}^{B} \lambda \hat{f}^b$

4. r_(i+1) = r_i - lambda*f^b

Boosting

idea: modify a quick learner to become better

The intuition

- The general idea is to fit a tree to the residuals from the previous model; i.e. we fit a tree using the residuals rather than the outcome Y as the response. Then this new tree is added into the fitted function in order to update the residuals.
- By fitting trees to the residuals we aim to improve the model performance in areas where it does not do well.
- Each of these trees are rather small, with d terminal nodes (hence they are "weak" learners).
- Here we have three tuning parameters, the number of trees B, the shrinkage parameter λ (controls the speed of the process) and the number of splits d.

Boosting

Tuning Parameters

- We typically use cross validation to set B (unlike bagging or random forests, boosting can overfit B). because we are estimating the trees sequentially: more trees = more "learning"
- The shrinkage parameter λ controls the rate at which the model learns and is usually set at a small arbitrary level (typically 0.01 or 0.001
- The number of splits d controls the complexity of each fitted tree. Typically this is also is set to a small number, say 1 or 2.
- The next graph applies two boosted models with $d = 1, 2$ and $\lambda = 0.01$ and compares it with a random forest. We can see that the model does particularly well (better than the random forest) but learns slower.

Comparison

Random Forests vs Boosting

- Random forests work by growing many deep trees to bootstrapped and split variable randomized versions of the data and averaging them. The motivation is to perform variance reduction for low-bias/high-variance models.
- Boosting works by recursively growing shallow trees to residuals and building an additive models from the sum of trees. The motivation is to preform bias reduction in high-bias/low-variance models. Random forests are an out of the box solution; not much tuning int he form of cross-validation of other parameter selection is needed. Recall the B is not a tuning parameter for random forests.
- Boosted models require careful tuning of thee parameters. This generally requires both extra work and extra data to perform cross-validation. The upside is that they generally tend to out-perform random forests in predictive ability.

Comparison

- The following graphs show results from bagging, random forests and boosting.