Predictive Analytics (ISE529)

Tree-Based Methods (II)

Dr. Tao Ma
ma.tao@usc.edu

Tue/Thu, May 22 - July 1, 2025, Summer

USC Viterbi
School of Engineering
Daniel J. Epstein Department of Industrial and Systems Engineering

USC University of Southern California

Issue of Decision Tree

- The simple decision trees for regression or classification discussed before suffer from *high variance*.
- This means that if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different.
- In contrast, a procedure with low variance will yield similar results if applied repeatedly to distinct data sets; linear regression tends to have low variance, if the ratio of n to p is moderately large.
- To overcome the issue, we introduce ensemble methods.

Ensemble Method

- An ensemble method is an approach that combines many simple "building ensemble block" models in order to obtain a single and potentially very powerful model.

- These simple building block models are sometimes known as weak learners, since they may lead to mediocre predictions on their own.

- We will now discuss bagging, random forests, and boosting. These are ensemble methods for which the simple building block is a regression or a classification tree.

BAGGING

Bagging

- *Bootstrap aggregation*, or *bagging*, is a general-purpose procedure for reducing the variance of a statistical learning method.
- Recall that given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by

$$Var\left(\bar{Z}\right) = \frac{\sigma^2}{n}$$

- In other words, *averaging a set of observations reduces variance*.

Bootstrap aggregation

- Hence a natural way to reduce the variance and increase the test set accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions.
- Of course, we generally do not have access to multiple training sets. Instead, we can bootstrap, by taking repeated samples from the (single) training data set.

Bootstrap aggregation

- In this approach we generate B different bootstrapped training data sets. We then train our method on each bootstrapped training set to get B different decision trees.
- Calculate $\hat{f}^1(x), \hat{f}^2(x), \cdots, \hat{f}^B(x)$, the predictions at a point x, using B separate decision trees.
- Then average all the predictions to obtain a single low-variance statistical learning model, given by

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

This is called *bagging*.

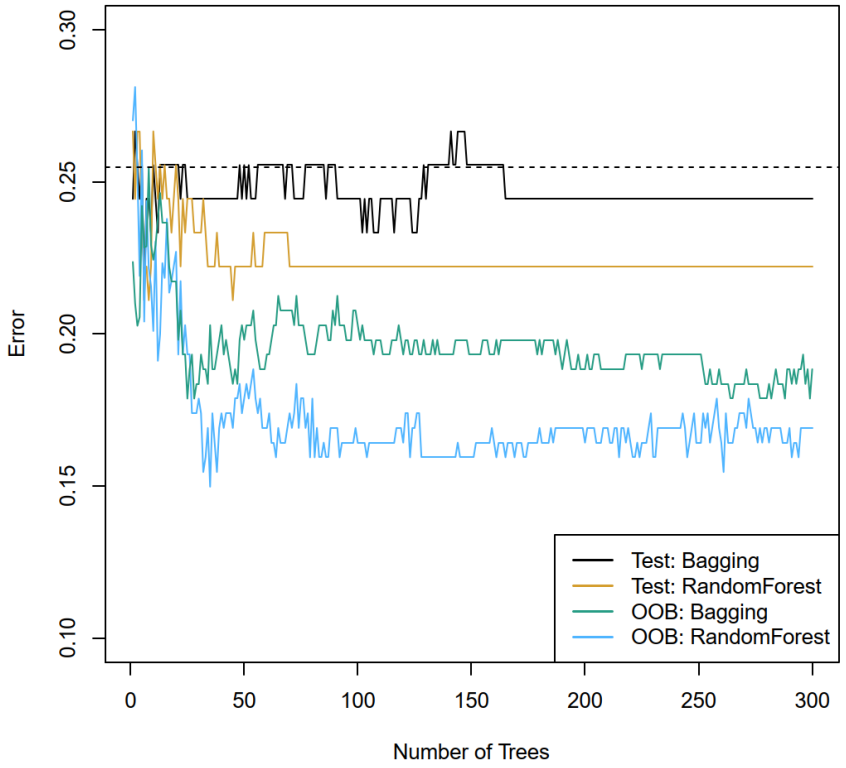
Regression vs. Classification

- For 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.
- For classification trees,
  - for each test observation, we record the class predicted by each of the B trees, and take a *majority vote*: the overall prediction is the most commonly occurring class among the B predictions.

Out-of-Bag Error Estimation

- There is a very straightforward way to estimate the test error of a bagged model.
- With bagging, trees are repeatedly fit to bootstrapped **subsets** of the observations. 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 *i*th observation using each of the trees in which the *i*th observation was OOB. This will yield around B/3 predictions for the *i*th observation, which we average.
- This estimate is equivalent to leave-one-out cross-validation error for bagging, if B is large.

Bagging the heart data

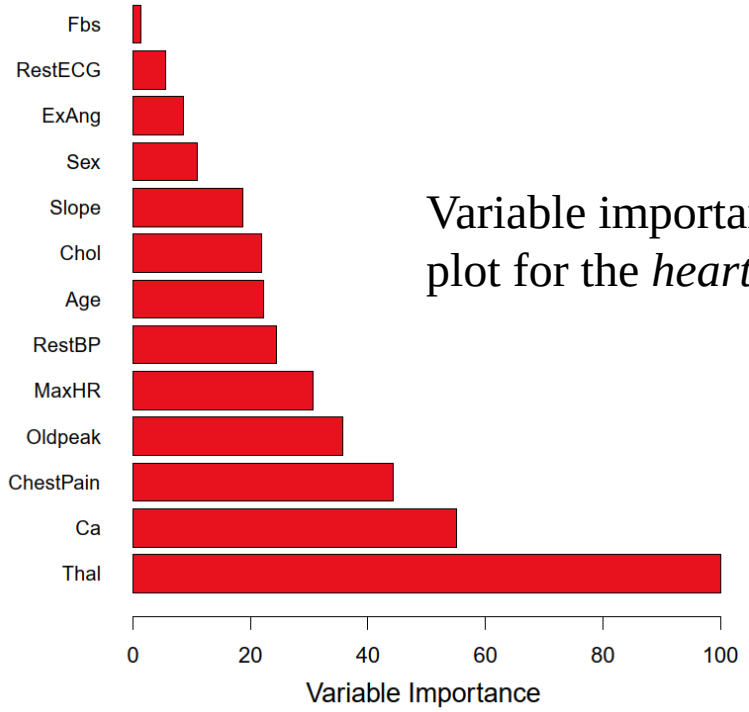


- The test error rate (black and orange) is shown as a function of *B*, the number of trees constructed using bootstrapped training data sets.
- The dashed line indicates the test error resulting from a **single** classification **tree**.
- The green and blue traces show the OOB error, which in this case is considerably lower

Using a very large value of *B* will not lead to overfitting. In practice, we use a value of *B* sufficiently large that the error has settled down. Using *B* = 100 is sufficient to achieve good performance in this example.

Variable Importance Measure

- For bagged *regression* trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all *B* trees. A large value indicates an important predictor.
- Similarly, for bagged *classification* trees, we add up the total amount that the Gini Index is decreased by splits over a given predictor, averaged over all *B* trees.



Variable importance plot for the *heart* data

RANDOM FORESTS

Issue with Bagging

- As in bagging, we build a number of decision trees on bootstrapped training samples. Each of splits considers all 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 trees will use this strong predictor in the top split (root node). Consequently, all of the bagged trees will look quite similar to each other.
- Hence the predictions from the bagged trees will be highly **correlated**. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. This means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.

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.
- Random Forests overcome this problem by forcing each split to consider only a subset of the predictors.
- But in Random Forests, 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).

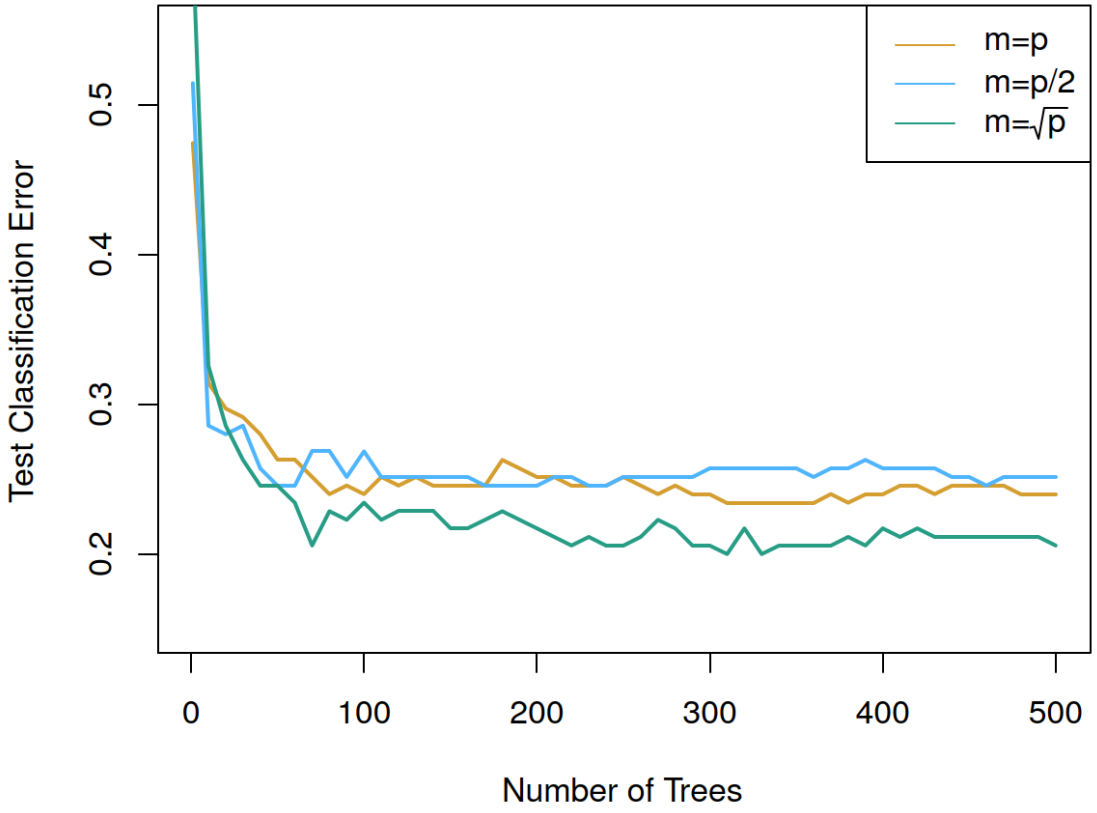
Random Forests

- On average (p − m)/p of the 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.
- The main difference between bagging and random forests is the choice of predictor subset size *m*. For instance, if a random forest is built using $m = p$, then this amounts simply to bagging.
- Using a small value of *m* in building a random forest will typically be helpful when we have a large number of correlated predictors.

Example: gene expression data

- We applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.
- There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.
- Each of the patient samples has a qualitative label with 15 different levels: either normal or one of 14 different types of cancer (response variable).
- We use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.
- We randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of *m*.

Example: gene expression data



- Results from random forests for the 15-class gene expression data set with $p = 500$ predictors.
- The test error is displayed as a function of the number of trees.

- Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node.
- Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$). A single classification tree has an error rate of 45.7%.

BOOSTING

Boosting

- 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.
- Notably, each tree is built on a bootstrap data set, independent of the other trees.
- 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 for Regression Trees

USC Viterbi School of Engineering

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,

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

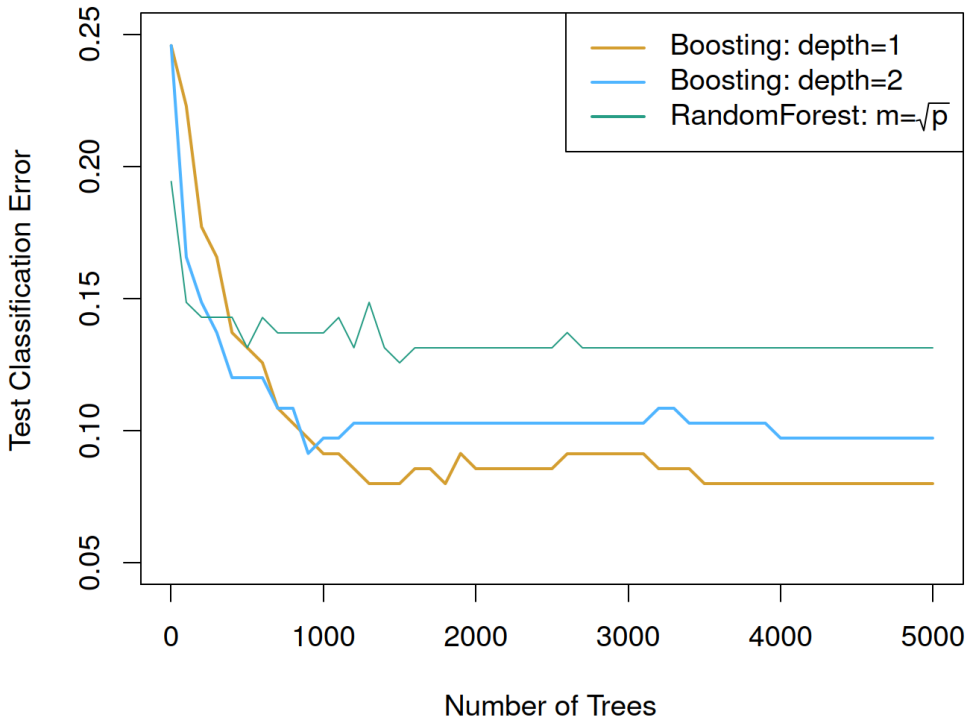
The idea behind this procedure

- As b = 1, we fit a decision tree to the response *Y*.
- Start from b = 2, we fit a decision tree to the residuals from the model. That is, we fit a tree using the current residuals, rather than the outcome Y , as the response. 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

- The number of trees B. Boosting can overfit if B is too large. 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, and the right choice can depend on the problem. Very small λ can require using a very large value of B to achieve good performance.
- 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.

Example: gene expression data (cont'd)



- Results from performing boosting and random forests on the 15-class gene expression data set 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.
- The test error rate for a single tree is 24%.

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 ensemble of 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.