8.2 Bagging, Random Forests, Boosting, and Bayesian Additive Regression Trees

- *Ensemble* methods are approaches that combine many simple “building block” models in order to obtain a single model.
- The building block models are sometimes called *weak learners*, since they may not perform well on their own.
- The methods in this section are ensemble methods that use trees as their building blocks.

8.2.1 Bagging

- The bootstrap can be used in a much wider variety of situations than we see in Chapter 5.
    - Useful in situations when it's difficult or impossible to directly compute a standard deviation.
    - Here, we present an entirely different way to use the bootstrap.

Bagging

- Decision trees suffer from high variance (they're not robust).
- We would prefer a procedure with lower variance.
- Bagging (bootstrap aggregation) is a general-purpose procedure for reducing the variance of a statistical learning method.
  - It's particularly useful here, for trees.

Bagging

- Recall: given a set of n indep. obs. $Z_1, \ldots, Z_n$, each with variance σ^2, the distribution of the mean $\bar{Z}$ has variance σ^2/n.
- That is, averaging a set of observations reduces variance.
- So we might think to reduce the variance and increase test set accuracy by taking many training sets, building a separate model for each, and then averaging the resulting predictions.
  - Typically, we only have one dataset and so can't utilize this.
  - But bootstrap allows us to generate repeated samples through resampling.

Bagging Regression Trees

- In this approach, we generate B different bootstrapped training data sets.
- We then train the method on the bth bootstrapped data set and generate the predicted value $\hat{f}^{*b}(x)$.
- Then we average all the predictions to obtain

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

This is called *bagging*.

Bagging Classification Trees

- Here, instead of averaging all the predictions:
  - Record the class predicted by each of the B trees.
  - Select $\hat{f}_{\text{bag}}(x)$ by majority vote; the overall prediction is the mode of the B predictions.

Bagging the Heart Data

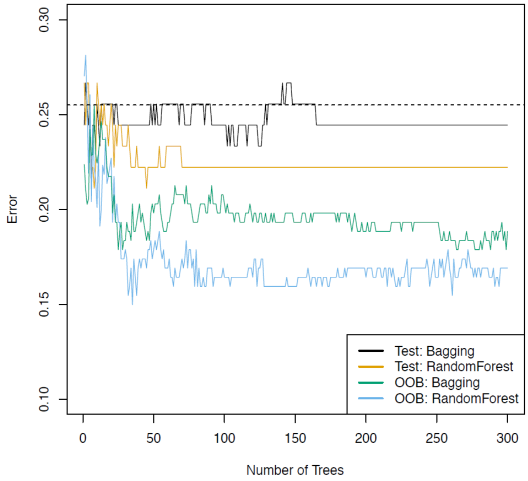


Figure Details

- The test error for bagging is shown in black.
- The dashed line is the test error for a single classification tree.
- We will get to the other methods shortly.

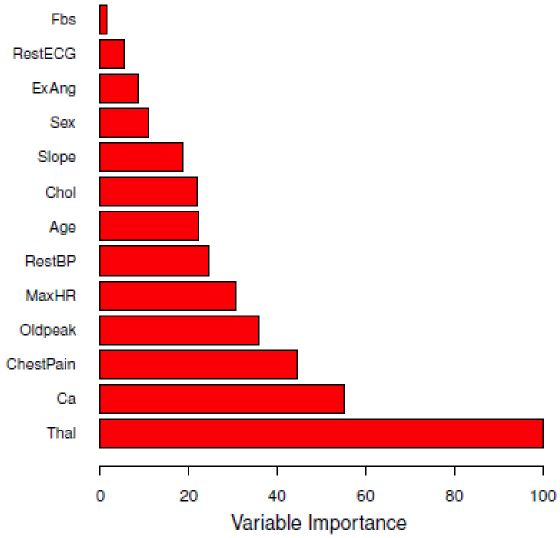
Out-of-Bag Error Estimation

- It turns out there is a straightforward way to estimate the test error of a bagged model, *without* having to use test/train splits or cross-validation.
- One can show that, on average, each bagged tree makes use of around 2/3 of the observations.
- The remaining 1/3 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 (around $B/3$ predictions, which we average).
- This estimate is essentially the LOO-CV error for bagging (for large B)

Variable Importance

- Bagging typically results in improved accuracy over a single tree, but it can be difficult to interpret models generated by bagging.
  - Thus, it is no longer clear which variables are most important to the procedure.
- We can obtain a summary of the importance of each predictor using RSS (or the Gini index).
  - We record the amount the RSS is decreased due to splits over a given predictor, averaged over all B trees.
  - This provides a measure of *variable importance*.

Heart Data


Fbs
RestECG
ExAng
Sex
Slope
Chol
Age
RestBP
MaxHR
Oldpeak
ChestPain
Ca
Thal
0
20
40
60
80
100
Variable Importance

Example Code

```r
library(randomForest)
set.seed(0)
X.heart <- Heart[,-14]
bag.hrt <- randomForest(x = X.heart, y = Heart$AHD,
                        mtry = 13, importance = TRUE)
```

```r
varImpPlot(bag.hrt)
```

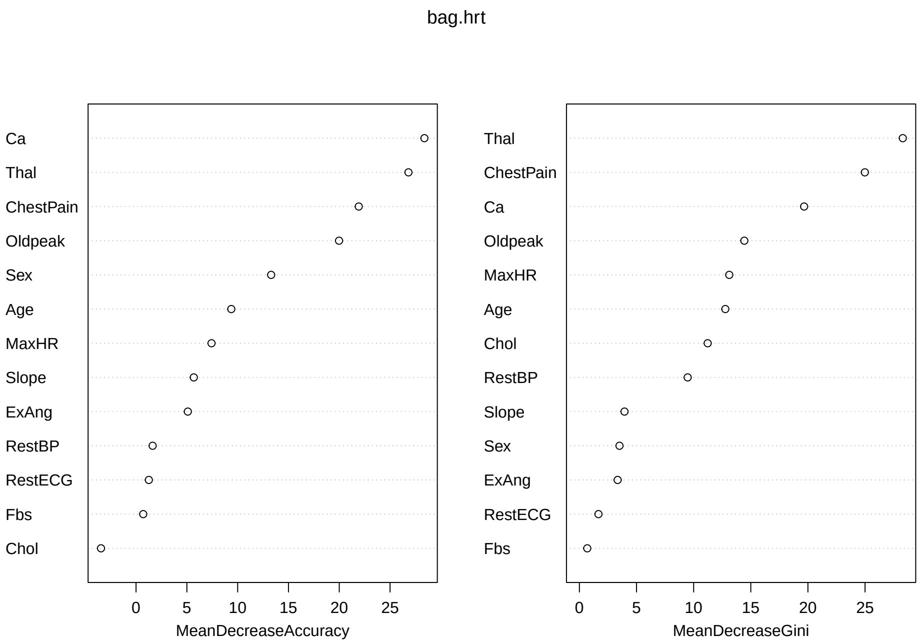
- Note that, unlike the textbook, I am not using a test/train split.

Error and Confusion Matrix

```
print(bag.hrt)
```

```
##
## Call:
##  randomForest(x = X.heart, y = Heart$AHD, mtry = 13, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##         OOB estimate of  error rate: 20.54%
## Confusion matrix:
##      No Yes class.error
## No  131  29   0.1812500
## Yes  32 105   0.2335766
```

- These values are based on OOB estimates.


bag.hrt
Ca
Thal
ChestPain
Oldpeak
Sex
Age
MaxHR
Slope
ExAng
RestBP
RestECG
Fbs
Chol
0
5
10
15
20
25
MeanDecreaseAccuracy
Thal
ChestPain
Ca
Oldpeak
MaxHR
Age
Chol
RestBP
Slope
Sex
ExAng
RestECG
Fbs
0
5
10
15
20
25
MeanDecreaseGini

Variable Importance Plots

- First plot: based on mean decrease of accuracy in predictions on the OOB samples when a given variable is permuted.
- Second plot: a measure of the total decrease in node impurity resulting from splits over that variable, averaged over all trees.

8.2.2 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

- We again build a number of decision trees on bootstrapped training samples.
- Unlike bagged trees, now each time a split 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 $m \approx \sqrt{p}$.

Random Forests

What does this do?

- At each split, a majority of the predictors are not available for consideration.
- If we had, for example, one very strong predictor, it would likely always be used for the initial split.
  - This prevents that.
  - This decorrelates the trees.
- Since tree building is top down and greedy, this also allows for some useful downstream splits that we might not see otherwise.

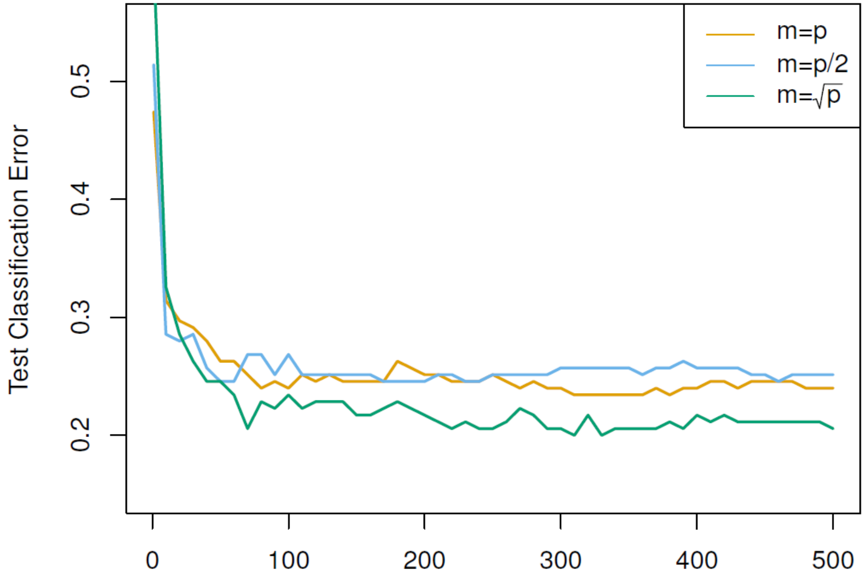
Random Forests

- Bagging is just a random forest with $m = p$.
- Making m small is a good way to deal with high dimensional, highly correlated data.

Example: Gene Expression Data

- Random forests applied to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.
- There are $\approx$ 20,000 genes in humans, and individual genes have different levels of 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.
- Use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.

Gene Expression Data



8.2.3 Boosting

- Boosting is another general approach that can be applied to many statistical learning methods for regression or classification.
  - We focus on boosting for decision trees.

Boosting vs Bagging

- Recall: 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 Algorithm for Regression Trees

1. Set $\hat{f}(x) = 0$ and $r_i = y$ for all i in the training set.
2. For $b = 0, 1, \ldots, B$, repeat:
   a. Fit a tree $\hat{f}^b$ with d splits to the training data (X, r).
   b. Update $\hat{f}$ by adding a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
   c. Update the residuals
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x)$$
3. Output the boosted model
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Boosting Idea

- Unlike fitting a single large decision tree to the data, the boosting approach instead learns slowly.
  - This reduces the chance of overfitting.
- 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.
- 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.

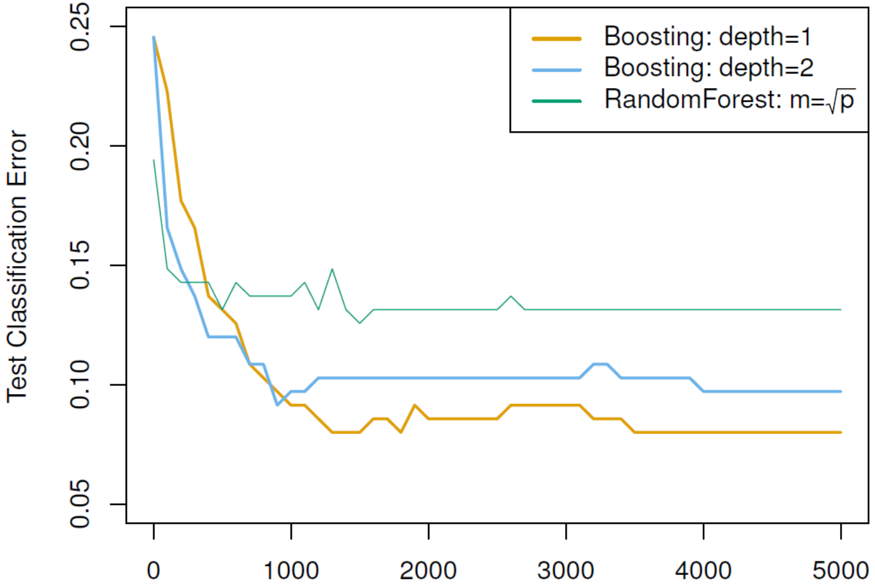
Boosting for Classification

- The idea is similar, but the technicalities are more complex.
- You can find the details in *Elements of Statistical Learning* (the textbook author's graduate-level textbook on this material)
- The R package `gbm` (gradient boosted models) handles a variety of regression and classifcation problems.

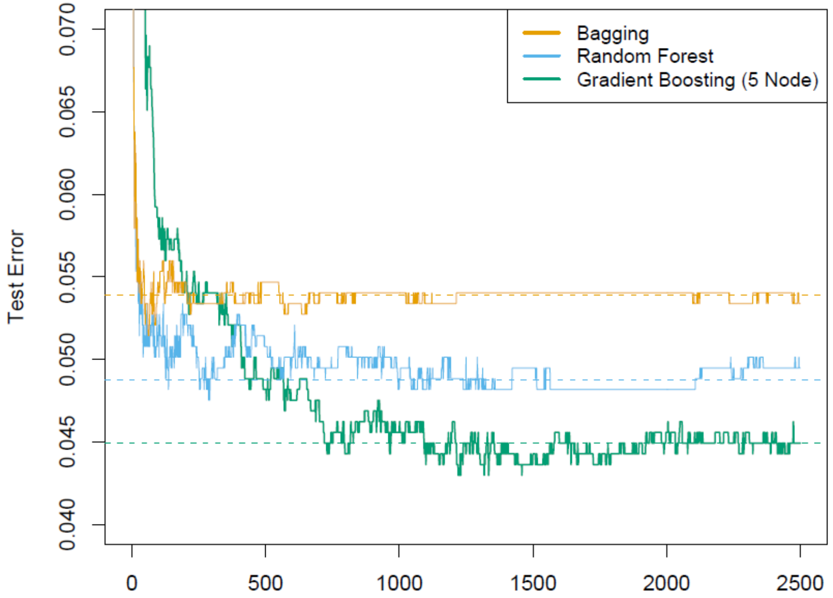
Tuning Parameters

1. The number of trees B
   - Unlike bagging and random forests, boosting can overfit if B is too large.
   - We will select B using cross-validation.
2. The shrinkage parameter, λ
   - Controls the rate at which boosting learns.
   - Typical values are 0.01 or 0.001.
   - Very small λ can require using a very large value of B in order to achieve good performance.
3. The number of splits in each tree, d.
   - 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.

Gene Expression Example, Continued



Another Classification Example


Bagging
Random Forest
Gradient Boosting (5 Node)
Test Error