Bagging, Random Forests and Boosting

Bagging

- The decision trees 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

- A low variance procedure 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

- Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method. It is particularly useful in the context of decision trees

- We generate B samples with replacement from the (single) training data set
- Then train our (regression) method on the b bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally 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.

- While bagging can improve predictions for many regression methods, it is particularly useful for decision 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

- Each individual tree has high variance. A small change in the data can result in a very different series of splits, making interpretation somewhat precarious. The major reason for this instability is the hierarchical nature of the process: the effect of an error in the top split is propagated down to all of the splits below… But trees have low bias

- Averaging these B trees reduces the variance. It's similar to the way in which a set of n independent observations $Z_1, Z_2, \ldots, Z_n$, each with variance σ^2, has a mean $\bar{Z}$ with smaller variance σ^2/n

- Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure

- How can bagging be extended to a classification problem where Y is qualitative (e.g. $0/1$ variable)?

- The simplest approach is as follows. 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 (i.e. majority) 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, without the need to perform cross-validation (CV) or the validation set approach

- The key to bagging is that 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 is 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 will yield around $B/3$ ($B * (1/e) \approx B * 38\%$) predictions for the ith observation.

- The explanation for the 38% factor above is that for a fixed bootstrap sample any observation from the original sample is drawn with equal probability $1/N$.

  Then the probability that this drawn observation will not be equal to some fixed observation i from the original sample is equal to $1-\frac{1}{N}$ and the probability that the whole bootstrap sample will not contain observation i is $\left(1-\frac{1}{N}\right)^N$:

$$\Pr\{obs\ i \notin bootstrap\ \mathrm{b}\} = \left(1-\frac{1}{N}\right)^N \approx e^{-1} = 0.378$$

- In order to obtain a single prediction for the ith observation, we can average these predicted responses (if regression is the goal) or can take a majority vote (if classification is the goal)

- This leads to a single OOB prediction for the ith observation. An OOB prediction can be obtained in this way for each of the n observations, from which
  - the overall OOB MSE (for a regression problem) or
  - classification error (for a classification problem)

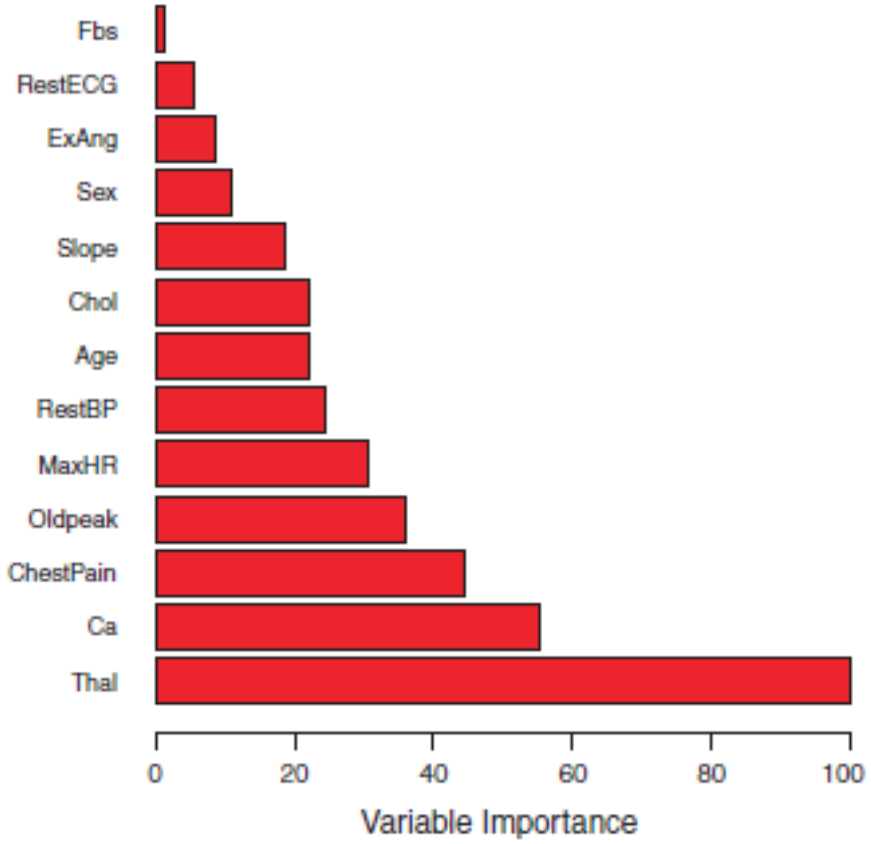
  can be computed. The resulting OOB error is a valid estimate of the test error for the bagged model

- It can be shown that with B sufficiently large, OOB error is virtually equivalent to leave-one-out cross-validation error. This is important for large datasets (why?)

Variable Importance Measures

- Bagging typically results in improved accuracy over prediction using a single tree
- It can be difficult though to interpret the model resulting from bagging. It is no longer clear which variables are most important to the procedure.
- Bagging improves prediction accuracy at the expense of interpretability
- One can still obtain an overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Gini index (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. A large value indicates an important predictor



Similarly, in the context of bagging classification trees, we can add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.

See a graphical presentation on left – here "Thal" is the most important predictor, the importance of the rest is relative to that of "Thal"

Random Forests

- Random forests provide an improvement over bagged trees by way of a random small tweak that de-correlates the trees

- 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 sample 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 sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$

- Why we would want to “block/drop” certain variables?

- Suppose that there is 1 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

- Consequently, all of the bagged trees will look quite similar to each other and the predictions from the bagged trees will be highly correlated. Averaging many highly (*positively*) correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities (why?). In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting

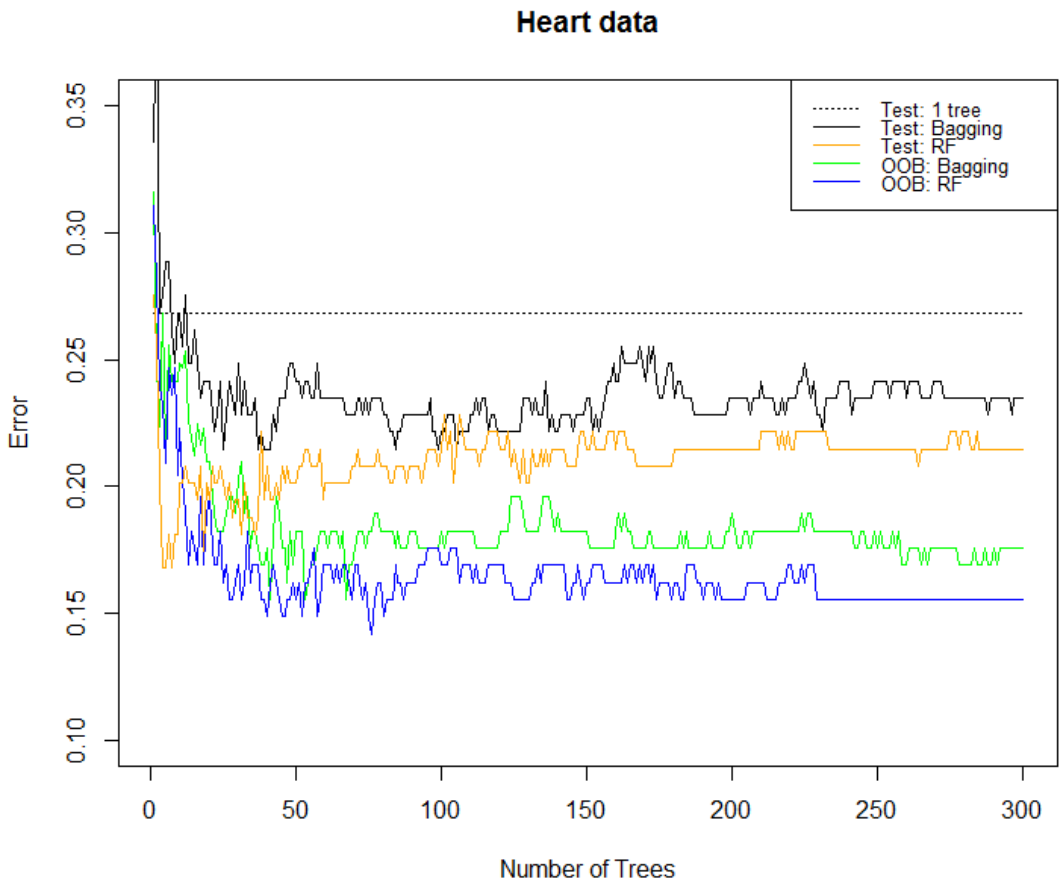
- Random forests overcome this problem by forcing each split to consider only a subset of the predictors

- Therefore, 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 de-correlating 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 1. On `Heart` data (see next slide), random forests using $m = \sqrt{p}$ leads to a reduction in both test error and OOB error over bagging



The test error (**black** and orange) is shown as a function of B, the number of bootstrapped training sets used.

The dashed line indicates the test error resulting from a single classification tree (unpruned).

The green and blue traces show the OOB error

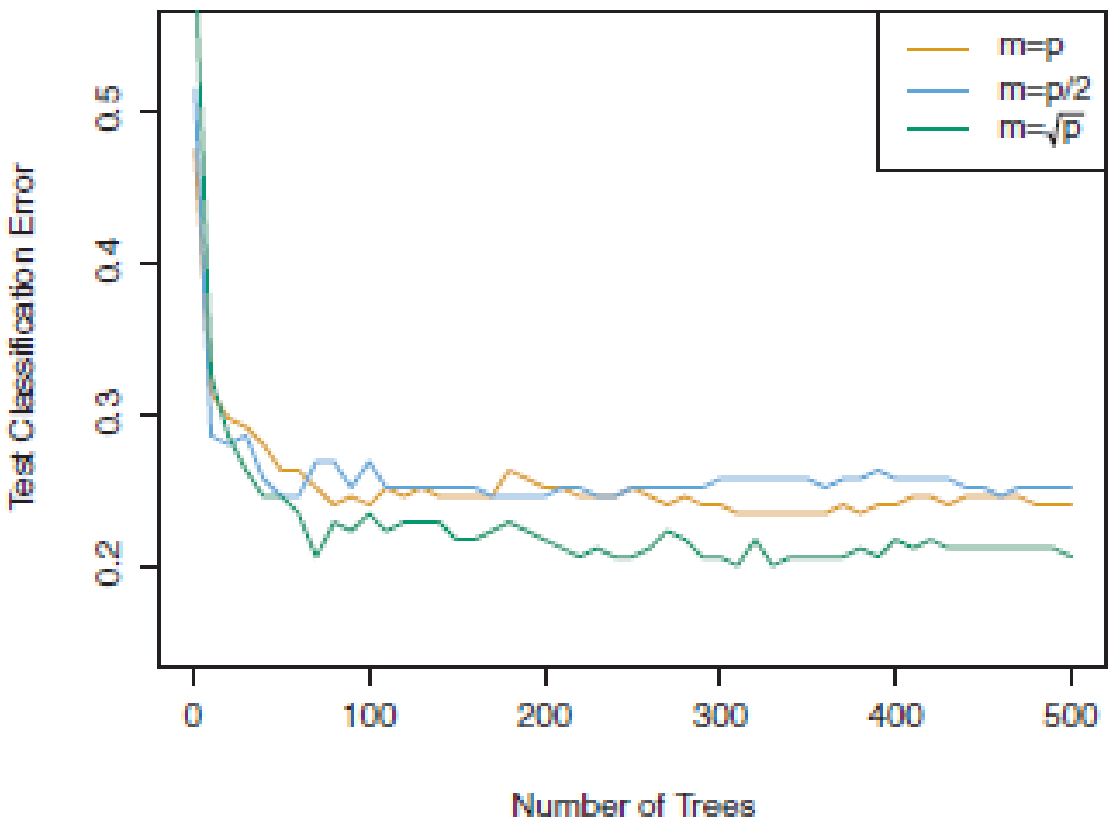
`Heart` data set: These data contain a binary outcome HD for 303 patients who presented with chest pain. An outcome value of Yes indicates the presence of heart disease based on an angiographic test, while No means no heart disease.

There are 13 predictors including `Age`, `Sex`, `Chol` (a cholesterol level), and other heart and lung function measurements.

Example 2. High-dimensional biological data set consisting of expression measurements of $4{,}718$ genes measured on tissue samples from $n = 349$ patients. There are around $20{,}000$ genes in humans, and individual genes have different expression, in particular cells, tissues etc.

In this data set, each of the patient samples has a qualitative label with 15 different levels: either normal or 1 of 14 different types of cancer.

Predict cancer type based on the 500 genes that have the largest variance in the training set.



- Randomly divide the observations into a training and a test set, and apply random forests to the training set for three different values of the number of splitting variables m.
- The error rate of a single tree is 45.7%, and the null rate is 75.4%. Using 400 trees is sufficient to give good performance, and the choice $m = \sqrt{p}$ gave a small improvement in test error over bagging ($m = p$) in this example.
- As with bagging, random forests will not overfit if we increase B, so in practice we use a value of B sufficiently large for the error rate to have settled down.

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. We'll consider boosting only in the context of regression trees

- Bagging involves creating multiple copies of the original training data set using the bootstrap and then

  - Fitting a separate decision tree to each copy
  - Combining all of the trees in order to create a single predictive model

  Each tree is built on a data set, independent of the other trees

- In boosting trees are grown sequentially: each tree is fit on a modified version of the original data set. Like bagging, boosting involves combining a large number of decision trees, $\hat{f}^1, \ldots, \hat{f}^B$

Boosting for regression trees:

Step1: Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

Step2: 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), r being the vector of residuals

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

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

(c) Update the residuals

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

Step3: Output the boosted model

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

- Idea: 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 improve the residuals

- In general, statistical learning approaches that learn slowly tend to perform well ("slow and steady wins the race")

- Note that in boosting, unlike in bagging, the construction of each tree depends strongly on the trees that have already been grown

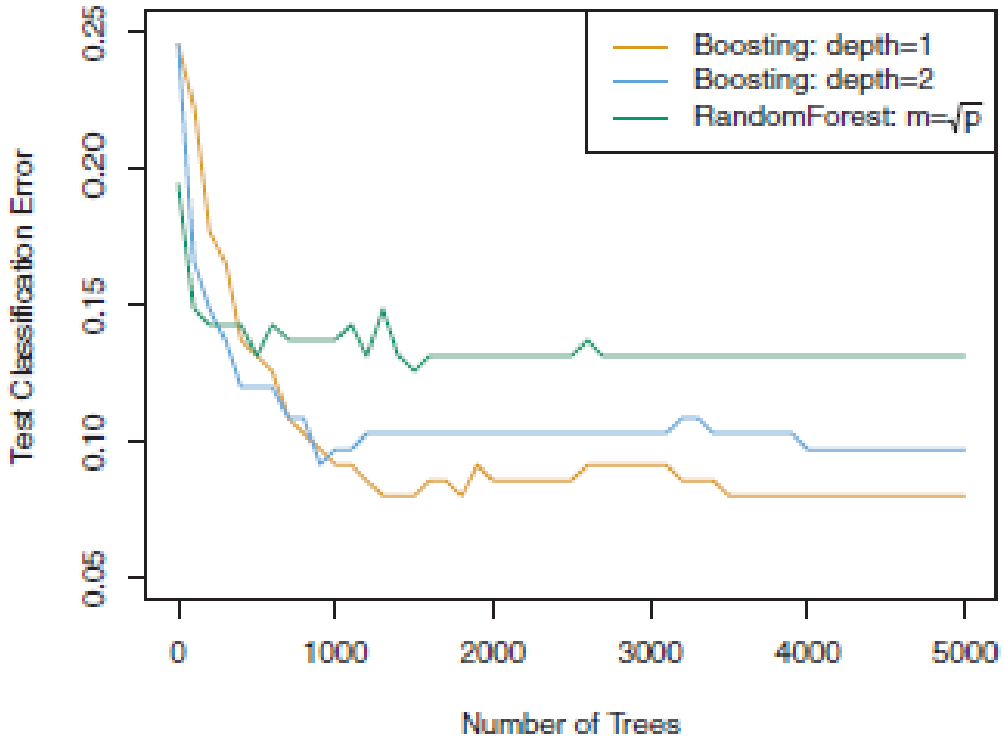
Boosting has 3 tuning parameters:

1. 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. Cross-validation is used to select B.

2. 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 in order to achieve good performance.

3. The number d of splits 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. In this case, the boosted stump ensemble is fitting an additive model, since each term involves only a single variable. 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 2 (cont'd) 15-class cancer gene expression data set



Simple stumps with an interaction depth of 1 perform well if enough of them are included. This model outperforms the depth-2 model, and both outperform a random forest.

Difference b/w boosting and random forests: in boosting, because the growth of a particular tree takes into account the other trees that have already been grown, smaller trees are typically sufficient. Using smaller trees can aid in interpretability as well.

Ensemble Models

- Basic idea is to combine different classifiers
  - build different "experts" and let them vote
  - Using more than one "head" is better than one

- Bagging, Random Forests and Boosting are some examples of Ensemble models

- Advantage
  - often improves predictive performance (can work with "weak" learners)

- Disadvantage
  - produces output that is more difficult to interpret

Reading:
ISLR: 8.2, 8.3.3-8.3.4
ESLII: 10.1, 15.1-15.3