Bagging: motivation

- The decision trees suffer from **high variance**. Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method.
- **averaging a set of observations reduces variance.** Hence a natural way to reduce the variance and hence increase the prediction 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.
- we bootstrap to generate different training datasets.

Bagging: procedure

- Bootstrap training set to obtain the bth bootstrapped training set, $b = 1, ..., B$.
- Train a tree with the bth bootstrapped training set, **not pruned** so that the individual tree has high variance but low bias
- Let $\hat{f}^{*b}(x)$ be the prediction for the test case x. The bagging prediction is then

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

$$\hat{f}_{bag}(x) = argmax_k \sum_{b=1}^{B} I_{\{\hat{f}^{*b}(x)=k\}} \qquad \text{for classification.}$$

Test error estimation for bagging trees

- Cross-validation or vaidation set
- Out-of-Bag (OOB) error estimation
  - 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 will yield around B/3 predictions for the ith observation. 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.

- Out-of-Bag (OOB) error estimation
  - 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, since the response for each observation is predicted using only the trees that were not fit using that observation.
  - It can be shown that with B sufficiently large, OOB error is virtually equivalent to leave-one-out cross-validation error

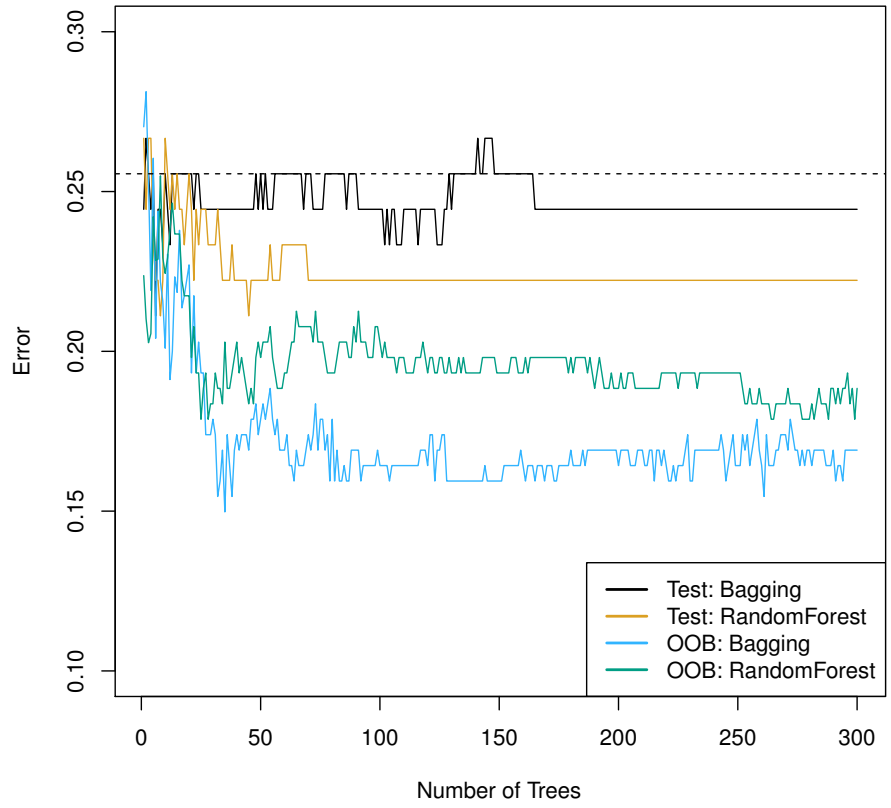

Figure: Bagging and random forest results for the Heart data. The test error (black and orange) is shown as a function of B, the number of bootstrapped training sets used. Random forests were applied with $m = \sqrt{p}$. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.

Variable importance measures

Bagging typically results in improved accuracy over prediction using a single tree at the expense of interpretability. 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. A large value indicates an important predictor.
- 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.

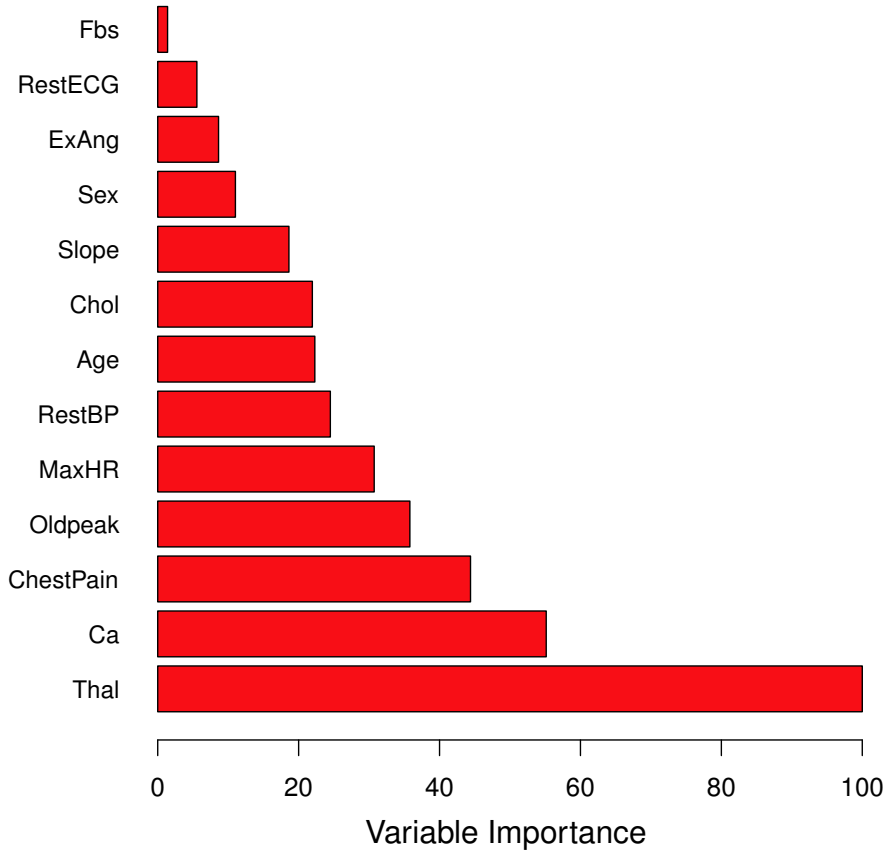

Figure: A variable importance plot for the Heart data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.

Random forests

- The bagged trees based on the bootstrapped samples often look quite similar to each other. They are therefore often **Highly correlated**.
- averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quanti- ties. In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.
- To **decorrelate** the trees on the bootstrapped samples, random forests 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.**

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

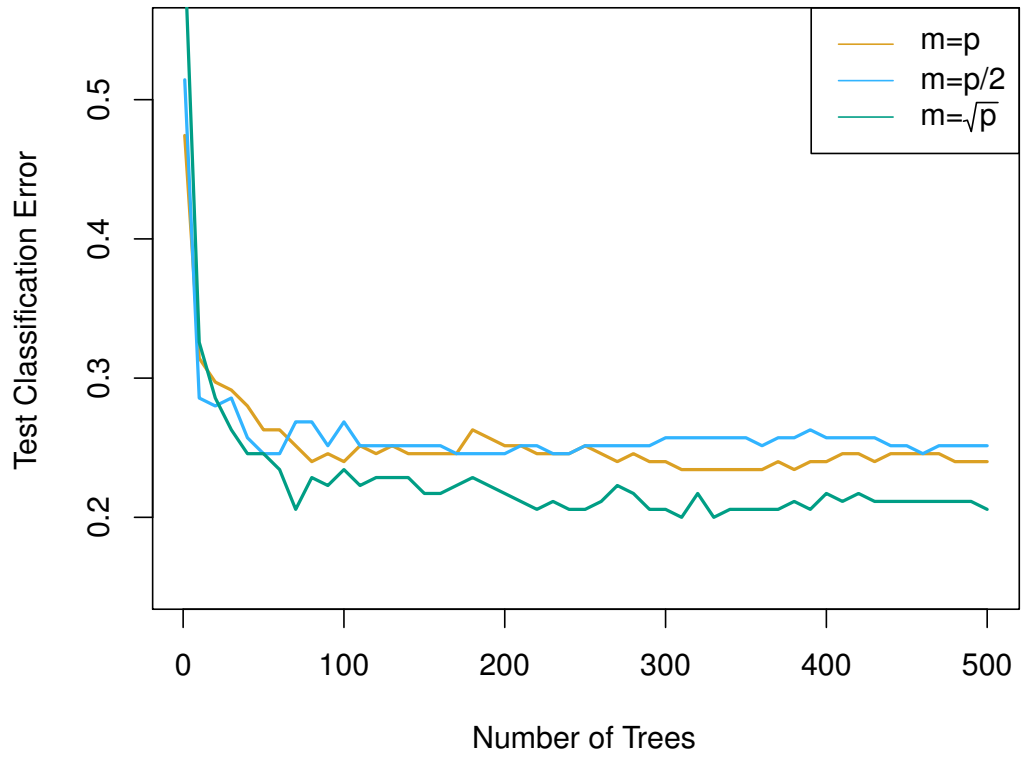

Figure: Results from random forests for the 15-class gene expression data set with $p = 500$ predictors. The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m ,the number of predictors available for splitting at each interior tree node. Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$). A single classification tree has an error rate of 45.7%

Choice of paramaters m and B

- m can be chosed by cross-validation
- For m, it is recommended that
  - For classification, the default value for m is $\sqrt{p}$ and the minimum node size is one.
  - For regression, the default value for m is $p/3$ and the minimum node size is five.
- 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: Adaboost (discrete boost) for classification

- Like bagging, boosting involves combining a large number of decision trees.
- Different from bagging, bossting 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 weighted original dataset: **The samples that are misclassified by the previouse tree receive more weights**
- The individual trees do not contribute to the final prediction equally: **Give larger weights to more accurate classifiers in the sequence.**

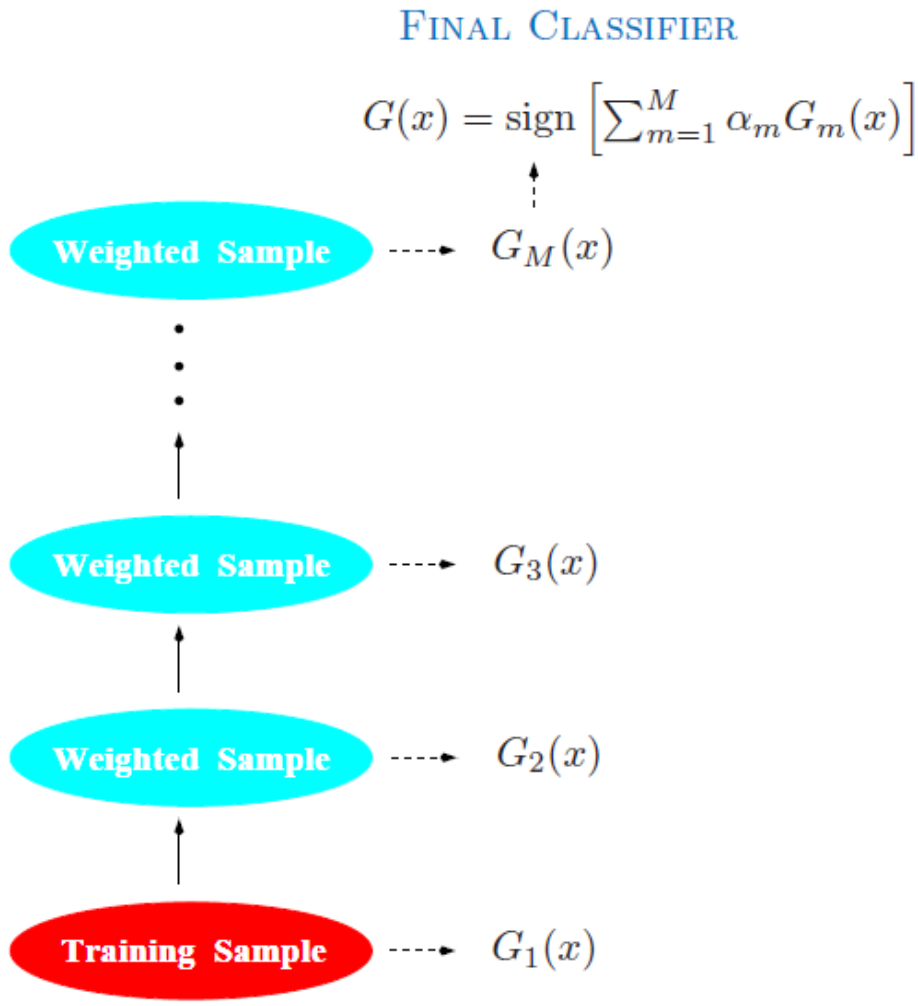

FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Figure: Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Boosting for regression trees

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,

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

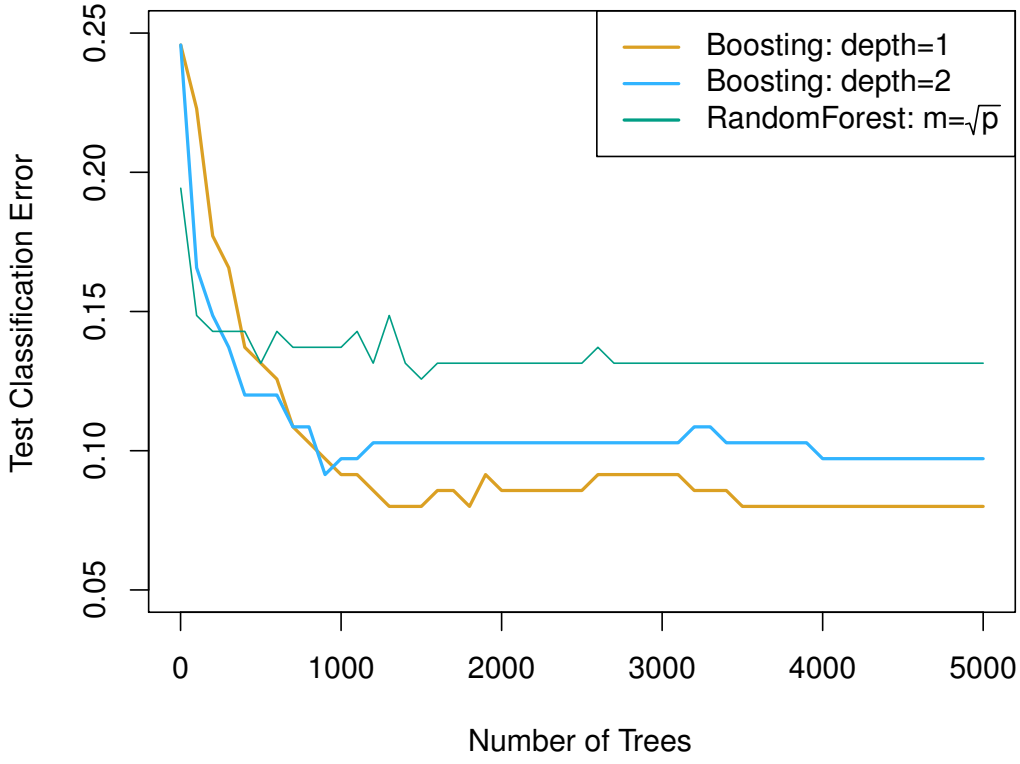

Figure: Results from performing boosting and random forests on the 15-class gene expression data set in order to predict cancer versus normal. The test error is displayed as a function of the number of trees. For the two boosted models, $\lambda = 0.01$. Depth-1 trees slightly outperform depth-2 trees, and both out- perform 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 %.

Choice of tuning parameters

- 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 common practice is to restrict all individual trees to have the same size.
  - Let d be the number of splits in each tree. When $d = 1$, 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**.
  - For regression, $d = 1$ often works well. For classiciation, experience indicates that $d \in [3, 7]$ works well in the context of boosting, with results being fairly insensitive to particular choices in this range. One can fine-tune the value for d by trying several different values and choosing the one that pro duces the lowest risk on a validation sample. However, this seldom provid es significant improvement over using $d \approx 5$.