☛ Instead of using the original tree, we use the classifier obtained by aggregating the B bootstrap trees.

☛ This classifier can work much better than the original tree because it has reduced variance.

☛ This aggregated classifier is no longer a tree (the decision rule is obtained from B trees but we can't express this as a tree).

☛ Example (contd): do bagging with $B = 10, ..., 200$ to see the effect of B on performance. Apply to 2000 simulated test data and compute classification error (test error).

Example (contd). Taken from Hastie at al., 2017, page 285:

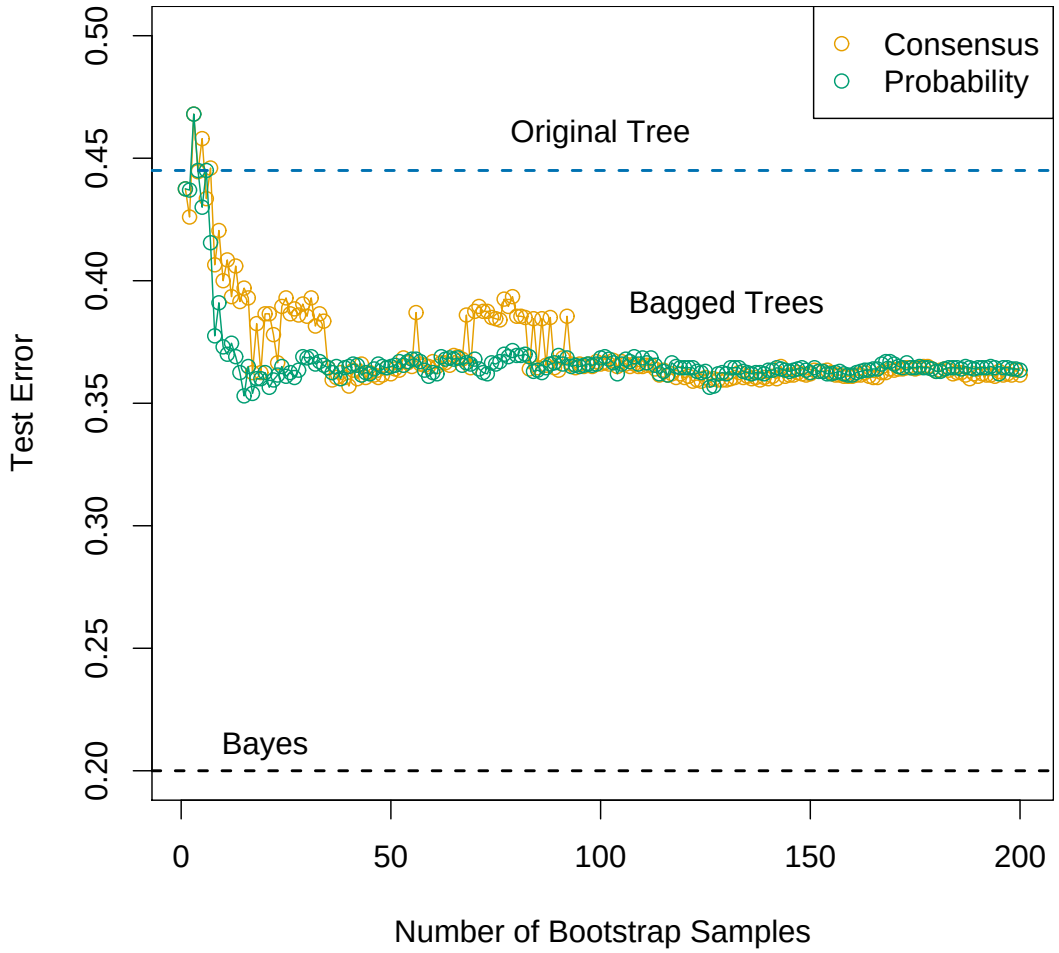

FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

☛ A problem with bagging is that the trees are not independent.

☛ This reduces the variance reduction of averaging/aggregating.

☛ To understand this intuitively, note that the average of B ident. distr. variables of variance σ^2 and with positive pairwise correlation ρ, has variance

$$\rho\sigma^2 + (1-\rho)\sigma^2/B.$$

☛ Thus no matter how large B is, the first term does not shrink and prevents the variance from being small.

☛ Random forest is a modified technique which reduces the correlation between the trees. When growing a bootstrap tree, before each split, it selects $m < p$ of the X_j's at random as candidates for splitting (instead of considering all X_j's as candidates).

☛ Default value of m is $\sqrt{p}$ and default size of each terminal node is 1 (grow the largest possible tree) but it is better to choose m adaptively (i.e. from the data) as it is a tuning parameter.

☛ The number B of trees is not a tuning parameter. As B increases the RF becomes more and more stable, but as long as B is large enough the benefit of taking B even larger is small. Just take B large (for example what is computationally feasible).

Algorithm. Taken from Hastie at al., 2017, page 588:

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 $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

Spam data. Taken from Hastie at al., 2017, page 589: improves bagging dramatically. Need enough trees but don't need as many as possible.

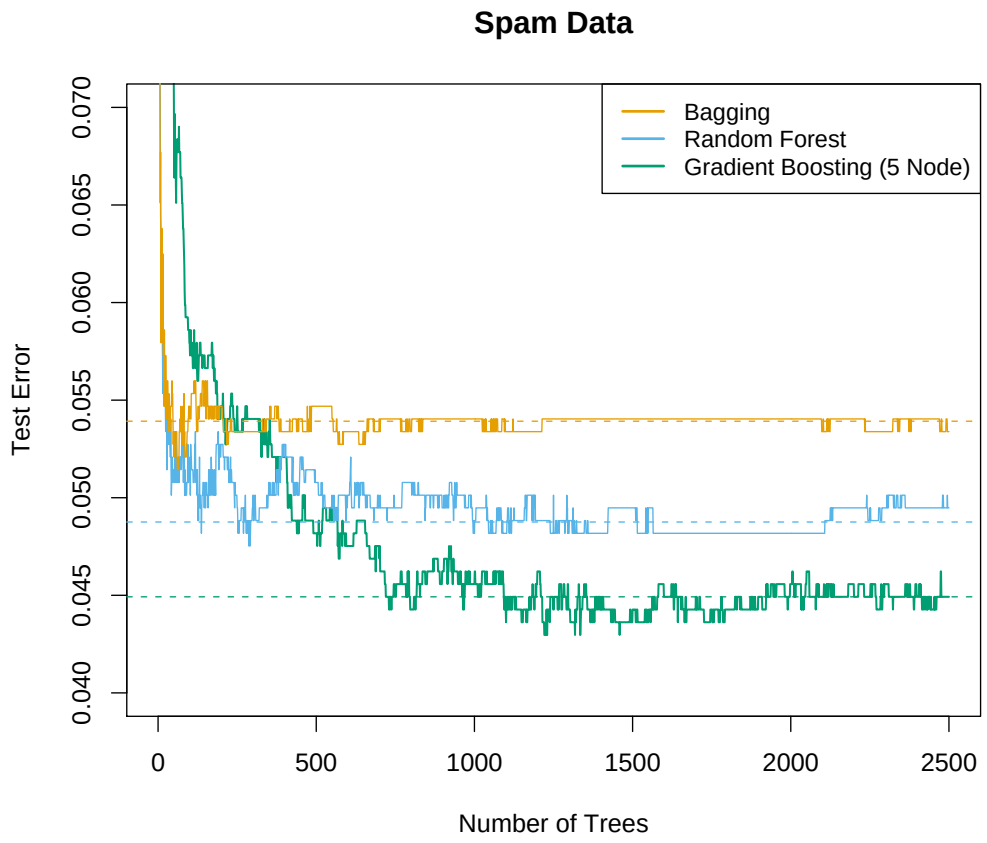

FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting,* 5*-node trees were used, and the number of trees were chosen by* 10*-fold cross-validation (*2500 *trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of* 1536*).*

8.3.1 OUT OF BAG DATA

☛ For random forests we do not need to use cross-validation to select m. Instead we can use out of bag (OOB) sampling.

☛ OOB: for each $(\mathbf{X}_i, G_i)$ in the training data, compute the random forest classifier, using only the bootstrap trees for which $(\mathbf{X}_i, G_i)$ does not belong to the corresponding bootstrap sample. Denote the result by $\hat{G}_i$.

☛ Then instead of using CV, we can choose m that minimises the OOB misclassification error

$$n^{-1} \sum_{i=1}^{n} I\{\hat{G}_i \neq G_i\}$$

(it does some sort of CV by itself and is faster to compute).

Spam data. Taken from Hastie at al., 2017, page 592: OOB error is not too far from true classification error.

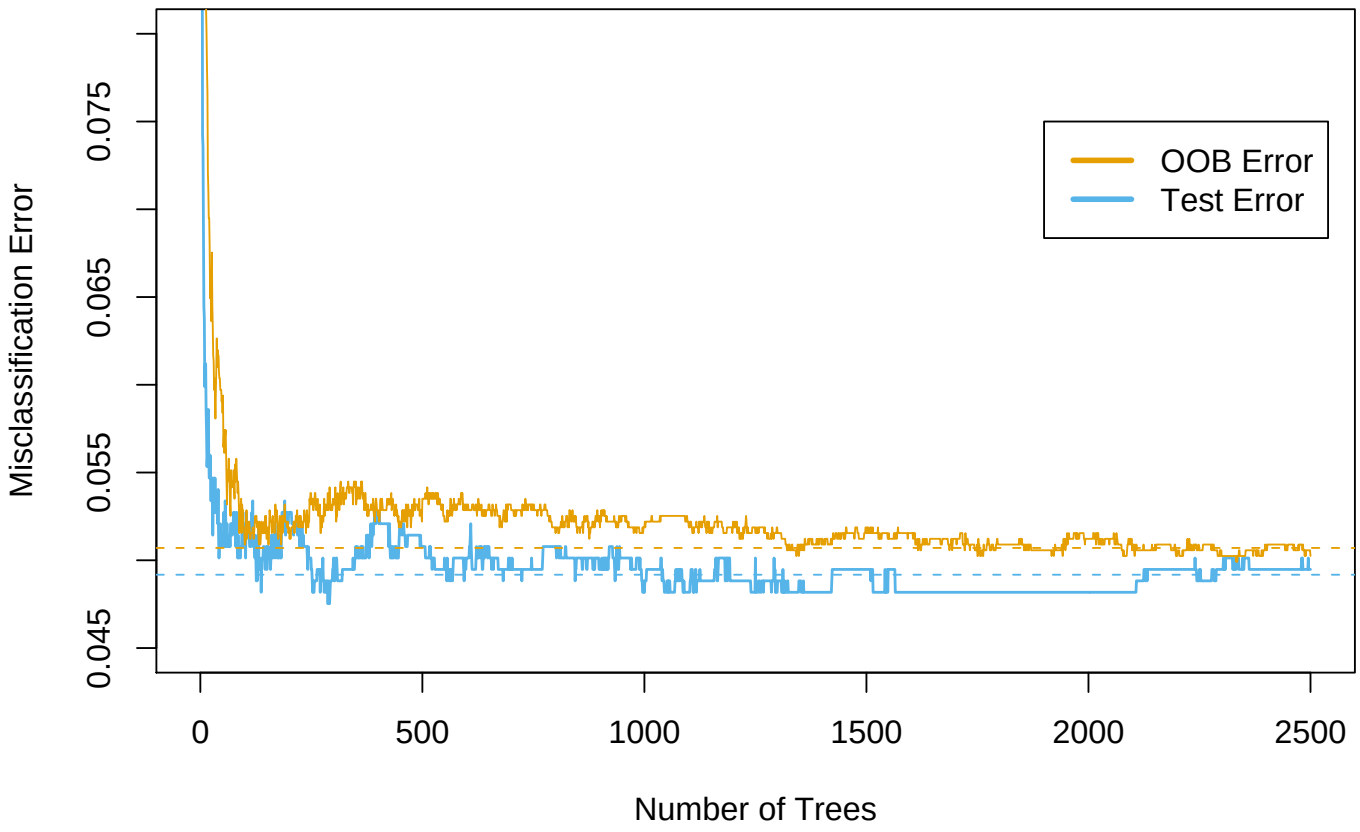

FIGURE 15.4. OOB *error computed on the* `spam` *training data, compared to the test error computed on the test set.*

8.3.2 VARIABLE IMPORTANCE

☛ Not all variables have the same importance for classification (some do not help distinguishing individuals from different classes). We can compute a measure of the importance of each variable X_j.

☛To construct a tree, we use consecutive binary splits of regions in two smaller regions. We have seen that if we already have L regions, to split one of them in two to obtain $L+1$ regions $R_1, \ldots, R_{L+1}$, we consider splitting all L regions in two and find the split that produces the smallest value of the risk

$$\sum_{\ell=1}^{L+1} N_\ell Q_\ell,$$

where N_ℓ is the number of training data in region R_ℓ and Q_ℓ is either the Gini index or the cross-entropy/deviance.

☛ Say at the tth internal node we have split a region in two according to the variable $X_{v(t)}$. For ex, if tth node is split according to $\{X_2 < 5.1\}$ and $\{X_2 \geq 5.1\}$, then $v(t) = 2$.

☛ Let $\widehat{\text{impr}}_t$ denote the decrease in $\sum_\ell N_\ell Q_\ell$ obtained by introducing this split, i.e. the difference between the value of the risk before and after the split. (When introducing a split, this risk can only decrease).

☛ For a single tree T, one way to measure the importance of X_j is via

$$\mathcal{I}_j(T) = \sum_t \widehat{\text{impr}}_t \cdot I\{v(t) = j\}\,,$$

where the sum is over all internal nodes of the tree.

☛ In other words, each time X_j is used for a split in T, we compute by how much this split reduces the risk. If it reduces the risk a lot, chances are this is an important variable: it makes a big difference to split according to that variable.

☛ In random forests we have B trees $T_1, \ldots, T_B$. To compute the importance of X_j, we compute $\mathcal{I}_j(T_b)$ for $b = 1, \ldots, B$ and take

$$\frac{1}{B}\sum_{b=1}^{B} \mathcal{I}_j(T_b).$$

☛ Other measure of importance based on OOB: for $b = 1, \ldots, B$, denote classifier computed from T_b by $\hat{G}_b$. Apply $\hat{G}_b$ to bth OOB sample $\mathcal{S}_b$ (data not in bth bootstrap sample). Compute prediction accuracy:

$$Acc_b = \sum_{i:(\mathbf{X}_i, G_i)\in\mathcal{S}_b} 1\{\hat{G}_b(\mathbf{X}_i) = G_i\}/\#\{i : (\mathbf{X}_i, G_i) \in \mathcal{S}_b\}\,.$$

☛ Monitor what happens if we change the values of the X_{ij}'s for the $\mathbf{X}_i$'s in $\mathcal{S}_b$ (all other components remain fixed). If X_j is important for classification then changing the value of the X_{ij}'s should worsen classification performance.

☛ How to do that? Randomly permute (among them) the X_{ij}'s of the $\mathbf{X}_i$'s in $\mathcal{S}_b$. Compute decrease in Acc_b resulting from these random permutations. Measure the importance of X_j by averaging these decreases over the B trees.