Tree-Based Methods

Chapter 8

1. 8.1 The basics of decision trees.

2. 8.2 Bagging, random forests and boosting

About this chapter

- Decisions trees: splitting each variable sequentially, creating rectugular regions.
- Making fitting/prediction locally at each region.
- It is intuitive and easy to implement, may have good interpreation.
- Generally of lower prediction accuracy.
- Bagging, random forests and boosting ... make fitting/prediction based on a number of trees.
- Bagging and Boosting are general methodologies, not just limited to trees.

Regression trees

- Trees can be applied to both regression and classifcation.
- CART refers to classification and regression trees.
- We first consider regression trees through an example of predicting Baseball players' salaries.

The Hitters data

- Response/outputs: Salary.
- Covarites/Inputs:
  Years (the number of years that he has played in the major leagues)
  Hits (the number of hits that he made in the previous year).
- preparing data: remove the observations with missing data and log-transformed the Salary (preventing heavy right-skewness)

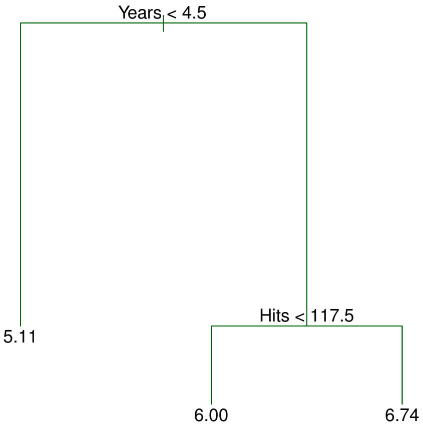

Figure: 8.1. Next page

Figure 8.1. For the Hitters data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to Years$<$ 4.5, and the right-hand branch corresponds to Years$\geq$ 4.5. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.

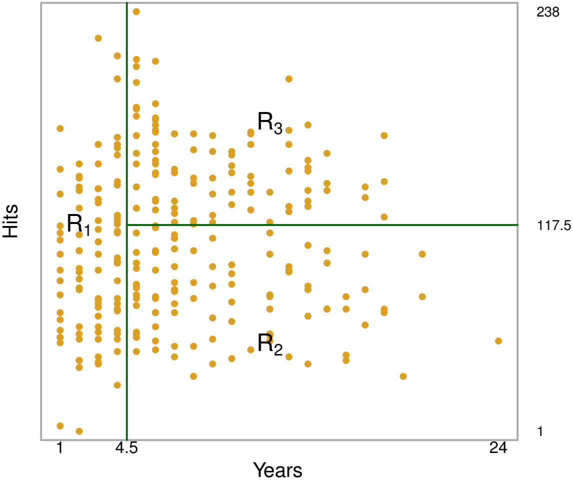

Figure: 8.2. The three-region partition for the Hitters data set from the regression tree illustrated in Figure 8.1.

Estimation/prediction

- On Regions R_1, R_2, R_3, the mean-log-salary is 5.107, and 6.74.
- Our prediction for any players in R_1, R_2 and R_3 are, respectively $1000 \times e^{5.107} = \$165,174$, $1,000 \times e^{5.999} = \$402,834$, and $1,000 \times e^{6.740} = \$845,346$.

Estimation/prediction

- Trees involve a series of splittings of the data, each time by one variable.
- The series of actions taken place sequentially creates a tree-like results.
- As in Figure 8.1, the terminal nodes are the three indexed by the numbers, which represent the regions R_1, R_2 and R_3. These regions constitute he final partiation of the data.
- Terminal nodes are also called leaves.
- Each *internal node* represents a splitting,
- In Figure 8.1, the two internal nodes are indexed by $Y < 4.5$ and $Hits < 117.5$.
- The lines connecting nodes are called branches.
- Trees are typically drawn upside down.

Two step towards prediction

- Run the splitting according to input values sequentially, and obtain final partition of the data in regions $R_1, ..., R_J$.
- For any new observation with covariates in region R_k, we predict its response by the average of the reponses of the data points in region R_k.

How to split

- Suppose we wish to partition a region R. In other words, we wish to separate the data in region R into two parts, day R_1 and R_2, according to one input values.
- What would be the optimal or efficient split in some sense?
- Only two flexibility in the split: 1. Choice of the input variable to split, 2. the cutpoint of the split of that chose input.
- Imagine that this is the final split of R: R_1 and R_2 would be leaves.

  And we would use the mean response of data in R_1 and R_2 to predict the response of any new/old observations.

  We wish our choice of R_1 and R_2 would be optimal in the sense of achieving miminum prediction error on the training data in region R.

Recursive binary splitting

- A greedy algorithm (geedy means it is optimal at the current step): For $j = 1, ..., p$ and all real value s, let $R_1(j,s) = \{i \in R : X_j < s\}$ and $R_2(j,s) = \{i \in R : X_j \geq s\}$. And let $\hat{y}_1$ and $\hat{y}_2$ be the mean response of all observations in $R_1(j,s)$ and $R_2(j,s)$, respectively. Consider the following prediction error:

  $$\mathrm{RSS}_{new} = \sum_{i \in R_1(j,s)} (y_i - \hat{y}_1)^2 + \sum_{i \in R_2(j,s)} (y_i - \hat{y}_2)^2$$

  Choose the split which has the smallest prediction error. This split is the optimal one, denoted as R_1 and R_2.
- Continue the split till the final partition.

8.1 The basics of decision trees.

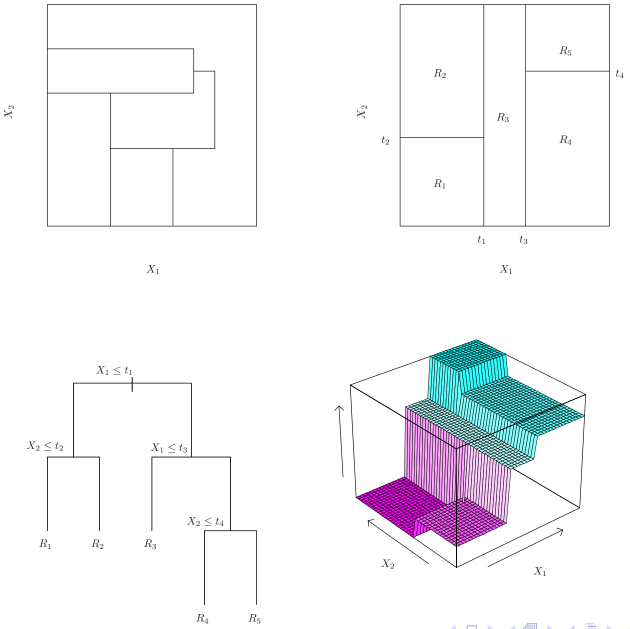

Figure 8.3. Top Left: A partition of two-dimensional feature space that could not result from recursive binary splitting. Top Right: The output of recursive binary splitting on a two-dimensional example. Bottom Left: A tree corresponding to the partition in the top right panel. Bottom Right: A perspective plot of the prediction surface corresponding to that tree.

When to stop split

- The problem of when to stop.
- If too many steps of splitting: many leaves, too complex model, small bias but large variance, may overfit.
- If too few steps of splitting: few leaves, too simple model, large bias but small variance, may underfit.

One natural idea

- When splitting R into R_1 and R_2, consider the RSS before the split

  $$\text{RSS}_{old} = \sum_{i \in R} (y_i - \hat{y})^2$$

  where $\hat{y}$ is the average of the response of data in R. With the optimal split, the reduction of RSS is

  $$\text{RSS}_{old} - \text{RSS}_{new}$$

- We can pre-choose a threshold, h, and decide the worthiness of the split.
- If the reduction is smaller than h, we do not do it, and stop right there; then R is one terminal node (a leave).
- If the reduction is greater than h, we make the split, and continue with next step.

One natural idea

- The idea is seemingly reasonable, but is too near-sighted.
- Only look at the effect of the current split.
- It is possible that even if the current split is not effective, the future splits could be effective and, maybe, very effective.

Tree pruning

- Grow a very large tree.
- Prune the true back to obtain a subtree.
- Objective: find the subtree that has the best test error.
- Cannot use cross-validation to examine the test errors for all possible subtrees, since there are just too many.
- Even if we can, this would probably be overfitting, since model space is too large.

Cost complexity pruning

- Let T_0 be the original (large) tree. Let T be any subtree. Use $|T_0|$ and $|T|$ to denote their numbers of teminal nodes, which represent complexity.
- Consider "Loss + Penalty":

$$\sum_{m=1}^{T} \sum_{i \in R_m} (y_i - \hat{y}_m)^2 + \alpha |T|$$

  where R_m are the terminal nodes of the subtree T, and the mean response of R_m is $\hat{y}_m$; α is tuning parameter.
- Denote the minimized subtree as T_α.
- If $\alpha = 0$, no penalty the optimal tree is the original T_0.
- If $\alpha = \infty$, the tree has no split at all. The predictor is just $\bar{y}$.
- The larger the α, the more penalty for model complexity.

Cost complexity pruning

- Just like Lasso, there exists efficient computation algorithm to compute the entire sequence of T_α for all α.
- Use cross-validation to find the best α to minimize the test error.

The algorithm

- 1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
- 2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

The algorithm

- 3. Use K-fold cross-validation to determine best α. That is, divide the training observations into K folds. For each $k = 1, ..., K$
  (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.
  (b) Evaluate the mean squared prediction error on the data in the left-out k-th fold, as a function of α.
  (c) Average the results for each value of α, and pick α to minimize the average error.
- 4. Return the subtree from Step 2 that corresponds to the chosen value of α.

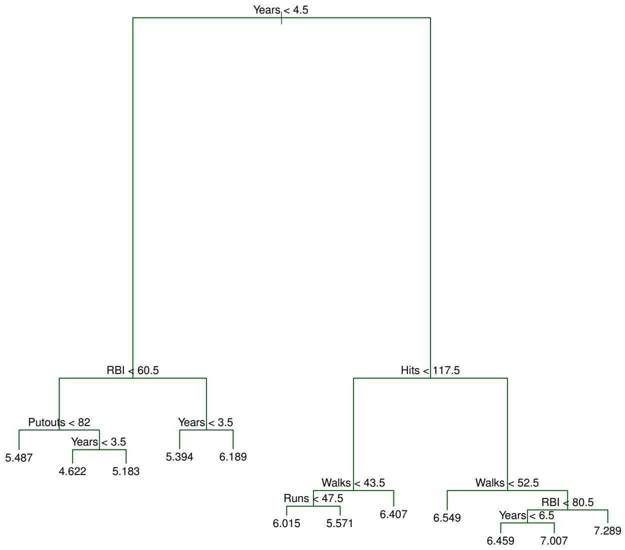

Figure: 8.4. Regression tree analysis for the Hitters data. The unpruned tree that results from top-down greedy splitting on the training data is shown.

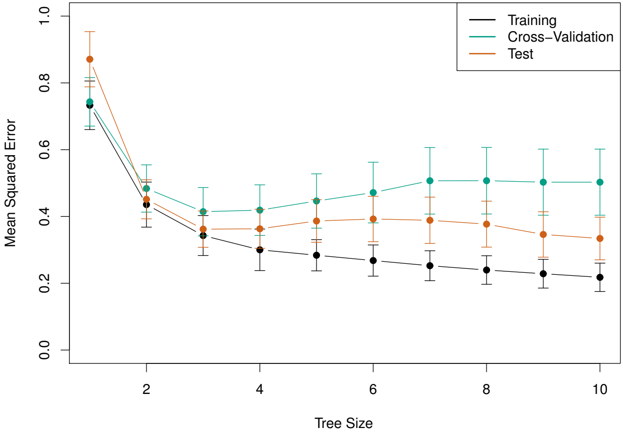

Figure: 8.5. Regression tree analysis for the Hitters data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.

Classification trees

- Regression has numerical responses; and classification has qualitative responses.
- Recall that for regression trees, we chose to obtain the greatest reduction of RSS.
  RSS is using sum of squares to measure the error.
- For classification trees, one can follow the same line of procedure as that of regression trees, but using error measurements that are more appropriate for classification.

Classification error rates

- For a region R, let $\hat{p}_k$ be the percentage of observations in this region that belong to class k.
- We assign any new observation in region R as from the class with largest $\hat{p}_k$, which is the so-called *most commonly occuring class* in training data.

The impurity measure

- The classification error rate (for this region R) is

$$E = 1 - \mathrm{max}_k \hat{p}_k.$$

- The Gini index is

$$G = \sum_{k=1}^{K} \hat{p}_k (1 - \hat{p}_k)$$

- The cross-entropy is

$$D = -\sum_{k=1}^{K} \hat{p}_k \log(\hat{p}_k)$$

.

- If R is nearly pure, most of the observations are from one class, then the Gini-index and cross-entropy would take smaller values than classfication error rate.
- Gini-index and cross-entropy are more sentive to node purity.
- To evaluate the quality of a particluar split, the Gini-index and cross-entropy are more popularly used as error measurement crietria than classification error rate.
- Any of these three approaches might be used when pruning the tree.
- The classification error rate is preferable if prediction accuracy of the final pruned tree is the goal.

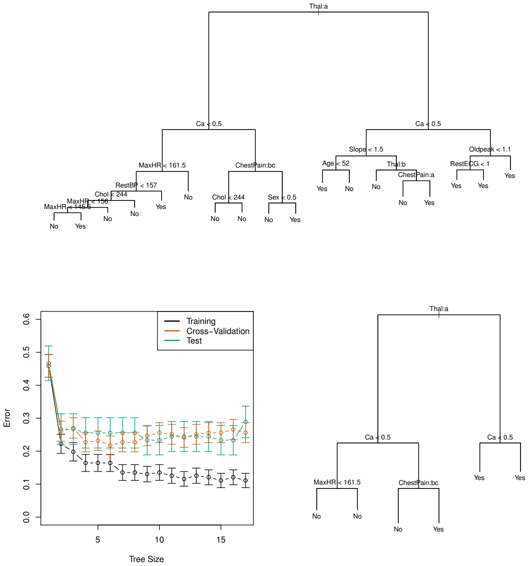
Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Slope < 1.5
Oldpeak < 1.1
RestBP < 157
Chol < 244
MaxHR < 156
MaxHR < 145.5
No
Yes
No
No
Yes
No
Chol < 244
Sex < 0.5
No
No
No
Yes
Age < 52
Thal:b
ChestPain:a
Yes
No
No
No
Yes
RestECG < 1
Yes
Yes
Yes
Training
Cross−Validation
Test
Error
0.0
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
Tree Size
Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
No
No
No
Yes
Yes
Yes

Figure 8.6. Heart data. Top: The unpruned tree. Bottom Left: Cross-validation error, training, and test error, for different sizes of the pruned tree. Bottom Right: The pruned tree corresponding to the minimal cross-validation error.

Trees vs. Linear models

- For regression model:

$$Y = f(X) + \epsilon$$

- Linear model assumes

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

- Regression trees assume

$$f(X) = \sum_{j=1}^{M} c_m 1(X \in R_m)$$

  where $R_1, ..., R_M$ are rectagular partitions of the input space.

- If the underlying realation is close to linear, linear model is better. Otherwise, regression trees are generally better. (Useless comments)

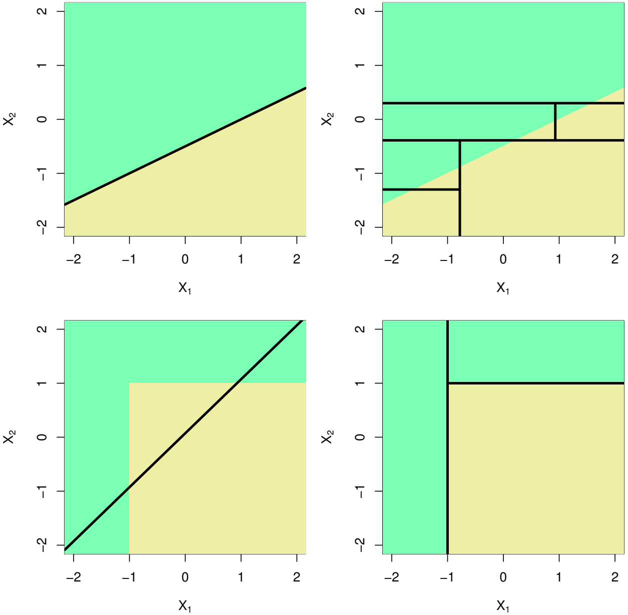
2
1
0
−1
−2
X_2
−2
−1
0
1
2
X_1

Figure 8.7. Top Row: A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right). Bottom Row: Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).

Advantages of Trees

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

Disadvantages of Trees

- Trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.
- Trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.
- However, by aggregating many decision trees, using methods like bagging, random forests, and boosting, the predictive performance of trees can be substantially improved. We introduce these concepts in the next section.

Bagging (Boostrap Aggregating)

- A general purpose procedure to reduce variance of a learning method.
- A model averaging technique.
- Decision tree is generally a high variance method. (Apply the method based on different data based on same sampling scheme would lead to very different result.)
- Average of iid random variables would have a reduced variance σ^2/n

The procedure.

- Model
$$y_i = f(x_i) + \epsilon_i, \quad i = 1, ..., n.$$
- Suppose a statistical learning method gives $\hat{f}(\cdot)$ based on the training data $(y_i, x_i), i =, 1..., n$.
- For example,
  1. Linear model: $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}^T x_i$
  2. KNN: $\hat{f}(x) = \sum_{j=1}^{J} \bar{y}_{\tilde{R}_j}$ with least distance to K-cluster partition.
  3. Decision tree: $\hat{f}(x) = \sum_{j=1}^{J} \bar{y}_{R_j}$ with rectangular partition.
  4. ...

The procedure of Bagging

- Data $(y_i, x_i), i = 1, ..., n$; and a learning method $\hat{f}$
- Draw a boostrap sample from the data, and compute a $\hat{f}_1^*$ based on this set of bootstrap sample.
- Draw another boostrap sample from the data, and compute a $\hat{f}_2^*$ based on this set of bootstrap sample.
- ....
- Repeat M times, obtain $\hat{f}_1^*, ..., \hat{f}_M^*$.
- Produce the learning method with bagging as

$$\frac{1}{M}\sum_{j=1}^{M}\hat{f}_j^*$$

The Bagging

- Bagging is general-purpose.
- It works best for high variance low bias learning methods.
- This is the case for decision trees, particularly deep trees.
- Also the case for large p.
- If the response is qualitative, we can take the majority vote (not averaging) of the predicted class based on all learning methods based on boostrap samples.

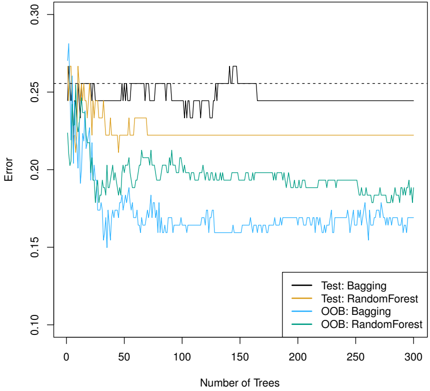

Figure: 8.8. Bagging and random forest results for the Heart data. The test error (black and orange) is shown as a function of B, the number of bootstrapped training sets used. Random forests were applied with $m = \sqrt{p}$. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower

Out-of-Bag (OOB) error estimation

- Estimation of test error for the bagged model.
- For each bootstrap sample, observation i is bootstrap sampled with probabilty $(1-1/n)^n \approx 1/e$.
- For each bootstrap sample, the number of observations not taken into this bootstrap sample is $n(1-1/n)^n \approx n/e$. These are referred to as out-of-bag (OOB) observations.
- For totally B bootstrap samples, about B/e times, the bootstrap sample does not contain observation i.
- The trees based on these bootstrap sample can be used to predict the response of observation i. Tatoally about B/e predictions.
- We average these predictions (for regression) or take majority vote (for classification) to produce the Bagged prediction for observation i, denote it as $\hat{f}^*(x_i)$.

Out-of-Bag (OOB) error estimation

- The OOB MSE is
$$\sum_{i=1}^{n}(y_i - \hat{f}^*(x_i))^2$$
- The OOB classification error is
$$\sum_{i=1}^{n} I(y_i \notin \hat{f}^*(x_i))$$
- 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.

Variable importance measures

- Bagging improves prediction accuracy at the expense of interpretability.
- An overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Gini index (for bagging classification trees).
- 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.
- 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.

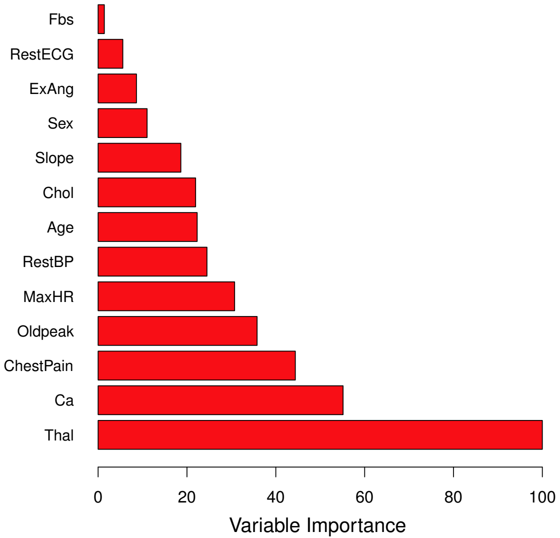

Figure: 8.9. A variable importance plot for the Heart data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.

Random forest

- Same as bagging decision trees, except ...
- 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
- Typically $m \approx \sqrt{p}$.

Random forest

- Every step, the split is constrained on a small number m and randomly selected inputs.
- Avoid all trees are too similar to each other.
- Too similar trees are too highly correlated, average highly correlated trees cannot achieve large amount of variance reduction.
- Extreme case: If all trees are the same, average of them is still the same one.
- Averaging uncorrelated or low-correlated trees can achieve large amount of variance reduction.
- Random forest produces less correlated trees.
- Random forest reduces to bagging if $m = p$.

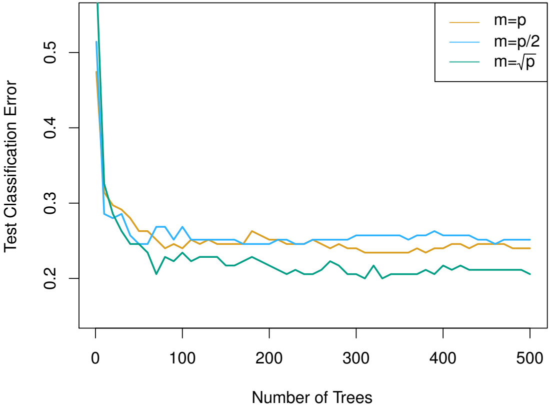

Figure: 8.10. Results from random forests for the 15-class gene expression data set with $p = 500$ predictors. The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree 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

- General purpose for improving learning methods by combining many weaker learners in attempt to produce a strong learner.
- Like bagging, boosting involves combining a large number of weaker learners.
- The weaker learners are created sequentially. (no boostrap involved).
- Bagging create large variance and possibly over-fit boostrap learners and try to reduce their variance by averaging.
- Boosting create weak learners sequentially and slowly (to avoid over-fit).

Boosting

- Suppose we have model

$$y_i = f(x_i) + \epsilon_i$$

  and a learning method to produce $\hat{f}$ based on $(y_i, x_i), i = 1, .., n$.
- Start with an initial predictor $\hat{f} = 0$. Let $r_i = y_i$.
- Start loop:
  1. Fit the data $(x_i, r_i), i = 1, .., n$, to produce $\hat{g}$.
  2. Update $\hat{f}$ by $\hat{f} + \lambda\hat{g}$.
  3. Update r_i by $r_i - \lambda\hat{g}(x_i)$.
- Continue the loop ... till a stop.
- Output $\hat{f}$
- Note that the output $\hat{f}$ is the sum of $\lambda\hat{g}$ at each step.

Algorithm for tree boosting

- 1. Set $f(x) = 0$ and $r_i = y_i$ for all i in the training set.
- 2. For $b = 1, 2, ..., B$, repeat:
  1. Fit a tree with d splits ($d + 1$ terminal nodes) to the training data (x_i, r_i).
  2. Update $\hat{f}$ by adding in a shrunken version of the new tree:

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

  3. Update the residuals,

     $$r_i \leftarrow r_i - \lambda \hat{f}_b(x_i) = y_i - \hat{f}(x_i).$$

- 3. Output the boosted model $\hat{f}$. In fact,

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

Tuning parameters for boosting trees

- The number of trees B. Large B leads to overfit. (not a tuning parameter for bagging)
- The learning rate λ.
- The number d in splits in each tree (the size of each tree). Often $d = 1$ works well, in which case each tree is a *stump*, consisting of a single split

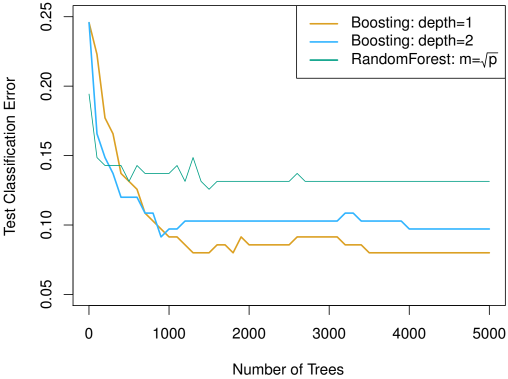

Figure: 8.11. Results from performing boosting and random forests on the 15-class gene expression data set in order to predict cancer versus normal. The test error is displayed as a function of the number of trees. For the two boosted models, $\lambda = 0.01$. Depth-1 trees slightly outperform depth-2 trees, and both 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%.

Exercises

*Run the R-Lab codes in Section *.3 of ISLR*
Exercises 1-4 and 7-8 of Section 8.4 of ISLR

End of Chapter 8.