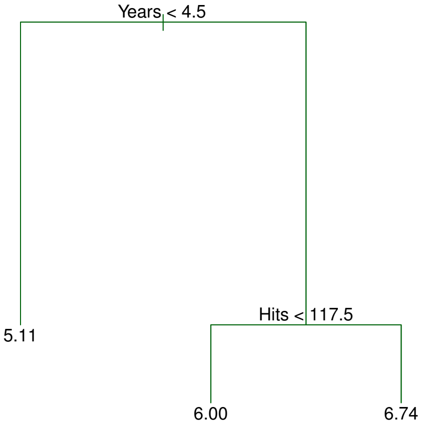
STA 5820
Chapter 8
Tree-based methods

Kazuhiko Shinki

Wayne State University

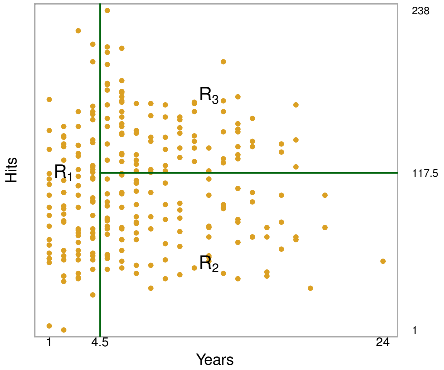
8.1 Basics of Decision Trees

The tree-based method is a framework for regression and classification by simple partitioning of variables.



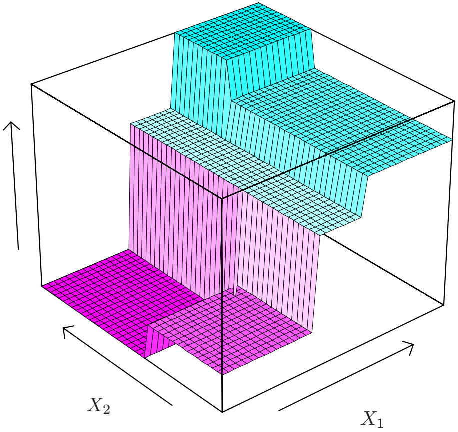
8.1.1 Regression Trees

The regression tree splits the predictor space with a single predictor at each step, and fits a regression (typically just a constant) in each region. The splits are made to minimize the sum of squared errors. Stop splitting the predictors when all terminal nodes (split regions) have less than a certain number of observations.



8.1.1 Regression Trees

Figure 8-3: Image of regression tree with two variables.



8.1.1 Regression Trees

How to determine the size of tree?

The number of splits ***T*** in a regression tree is determined so as to balance the size of tree ***T*** and the total error size.

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

For each value of α, minimize the objective function above to determine the optimal subtree. A subtree is obtained by removing some of the splits in the original tree (obtained in the previous slide). Creating a subtree is called pruning.

How to determine the parameter α?

α is a hyper-parameter. It is usually determined by cross-validation. (α is denoted as ***k*** in the `tree` library.)

8.1.1 Regression Trees

A summary of building a regression tree:

1. Make a large tree based on the method in the slide No.3.
2. For each value of α, prune the original tree to minimize the cost function in the slides No. 5.
3. Use cross validation to find the optimal α. Adopt the subtree obtained from the optimal α.

8.1.2 Classification trees

Classification trees

The tree methods are applied for classification problems in the same way as regression trees, with different error functions to be minimized. Two of the most popular functions for binary classification problems are as follows.

- Deviance (entropy): This is $-2 \times$ log of maximized likelihood. Suppose a terminal node i has N_i observations, of which n_i observations have a label '1' and the others have a label '0'. Let $p_i = n_i/N_i$. Then,

$$-2\sum_i \{N_i p_i \log p_i + N_i(1-p_i)\log(1-p_i)\}$$

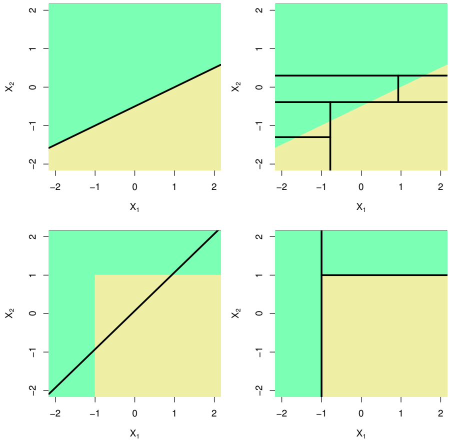
- Error rate: The proportion of observations classified incorrectly.

A general case for K categories is in 8.1.2 of the textbook.

8.1.3 Linear models vs tree-based methods

Linear models vs tree-based methods

The relative performance of linear models and trees depends on the data.



8.1.4 Advantages of disadvantages of trees

Advantages

- Results are intuitive and displayed graphically.
- No need to create dummy variables for categorical predictors.

Disdvantages

- Prediction performance is generally not high (performance will improve with the methods in 8.2).
- Not robust. A small change in the data dramatically changes the results.
- Methods often suffer from overfitting.

8.2 Bagging, Random Forests, Boosting

Methods to improve trees

It is well known that tree-based methods have a high variance. You can imagine that the split is largely determined by chance, so it is unstable.

The following methods improve the performance of tree-based methods.

- Bagging: Randomize the training set to make a tree, repeat the procedure, and take the average.
- Random forest: Similar to bagging, but each randomized sample icontains fewer randomly-chosen predictors.
- Boosting: Gradually determine the tree.

8.2.1 Bagging

Algorithm

Suppose that the training set has n observations.

1. Repeat the following procedure for $b = 1, \cdots, B$.
   1. Bootstrap n observations (with replacement) from the training set.
   2. Fit a regression (or classification) tree. Let $\hat{f}_b(x)$ be the fitted function.
2. The fitted curve is defined as:

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

There are hyper-parameters such as B (`ntree` in R), the maximum number of nodes for each tree (`maxnodes` in R), the minimum size of the terminal node (`nodsize` in R).

8.2.1 Bagging

Illustration of algorithm

8.2.1 Bagging

Out-of-bag (OOB) error estimation

In each tree created in bagging, unused observations are called out-of-bag (OOB) observations.

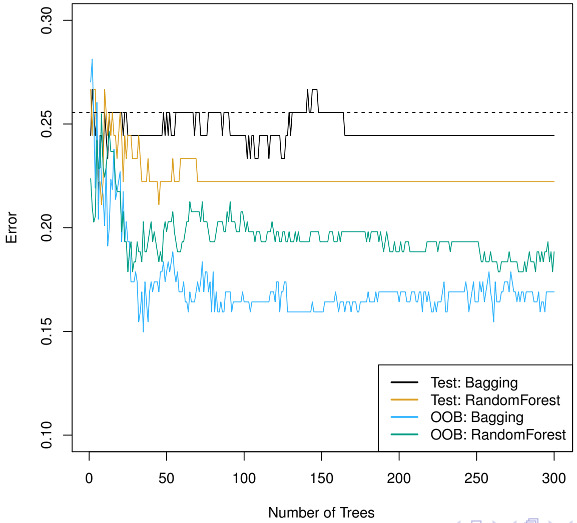
For example, if there are 6 observations (say, 1, 2, 3, 4, 5, 6) in the training set and a bootstrapped sample is $\mathbf{(5, 1, 3, 2, 3, 5)}$, then the OOB observations are $\mathbf{\{4, 6\}}$. Typically, one third of training set are OOB observations.

In the OOB error estimation, for each given observation $\boldsymbol{y_i}$, all trees in which $\boldsymbol{y_i}$ is not included are used to estimate $\boldsymbol{\hat{y}_i}$. The mean of OOB error size is the OOB error size. An OOB error size is a good measure of test performance, similar to the cross validated error.

Any error measure (such as the mean square error for regression and the error rate for classification) can be used in the OOB error estimation, similar to cross validation.

8.2.1 Bagging

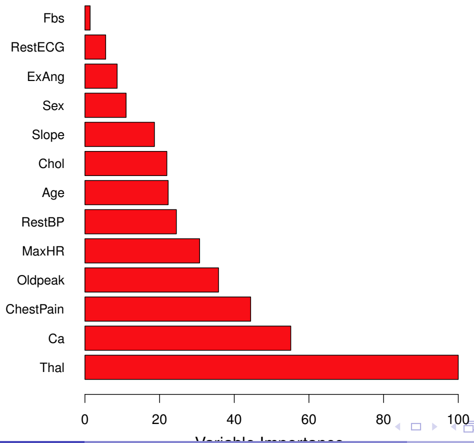
OOB error and test error



8.2.2 Random forests

Variable importance measures

The importance of each predictor in bagging is represented by how much the variable decreases an error measure such as the sum (or mean) of squared errors for regression.



8.2.2 Random forests

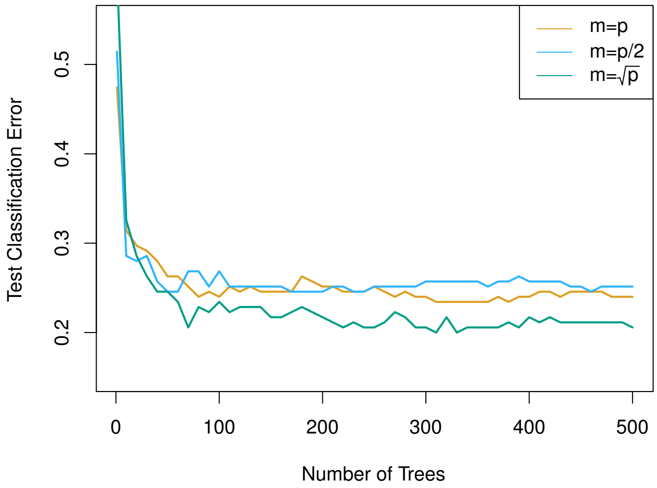
Random forests are the same as bagging except for the number of predictors allowed for each tree.

In random forests, each tree is allowed to use randomly selected m predictors out of all p predictors. Often $m = p/3$ or $m = \sqrt{p}$ is used. If $m = p$, random forests are the same as the bagging.

The idea behind random forests is to diversify the trees generated.

8.2.2 Random forests

Example: Test errors of random forests for different *m*'s



8.3 Boosting

Boosting is a method of slowly fitting a statistical model. That is, the fitted values are changed by small amounts at a time, but the model is fitted many times to obtain the final predicted values.

Boosting is a general approach that can be applied to many statistical learning methods for regression and classification.

8.3 Boosting

Algorithm

1. Set $\hat{f}(x) = 0$ for all x and $r_i = y_i$ for all $i (1 \le i \le n)$.
2. For $b = 1, ..., B$, repeat
    1. Fit a tree $\hat{f}_b$ with d splits to the training data (X, r).
    2. Update $\hat{f}$ by adding a fraction of the new tree:

       $$\hat{f}_{new}(x) := \hat{f}_{old}(x) + \lambda \hat{f}_b(x). \tag{1}$$

    3. Update the residuals:

       $$r_i := r_i - \lambda \hat{f}_b(x). \tag{2}$$

The final fitted values are

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}_b(x). \tag{3}$$

8.3 Boosting

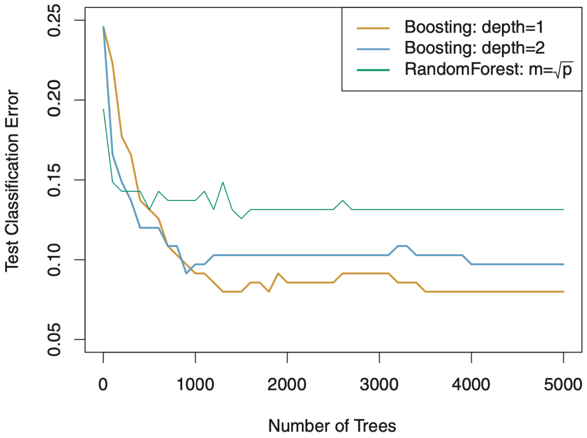
There are three hyperparameters.

- The number of trees B: A large B may overfit the model. Determine by cross-validation.
- The shrinkage parameter λ: Use a small positive number such as 0.01 or 0.001. The right choice depends on the problem. A small λ often implies a large B.
- The number of splits d in each tree. A small number. Often, $d = 1$.

8.3 Boosting

An image of boosting for trees

8.3 Boosting



Boosting test error rates by d ($d = 1$ for orange, $d = 2$ for blue) and B (on horizontal axis) for the binary classification problem in gene expression data set. λ is set to be **0.01**.