Paul Martins

Tree-based methods

Notes from ISLR book

Contents

1. Decision Trees

Trees split the predictors space into R_j boxes that are found via recursive binary splitting, a greedy approach to find the best split at each step (not looking at further potential split).

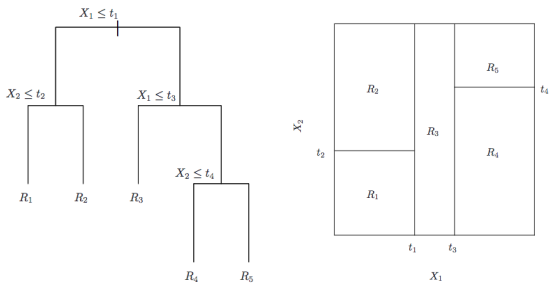

Figure 1. Two representations of a regression tree

1.1 Tree building

The choices of predictor X_j and cutpoint s to perform the split are dictated by the largest decrease in the RSS for regression.

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{1}$$

with $\hat{y}_{R_j}$ the mean response value in box R_j.

Classification boxes use the smallest Gini index G (eq. 2) or entropy D (eq. 3) for classification

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{2}$$

with $\hat{p}_{mk}$ the proportion of training observations of class k in the m box.

$$D = \sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk}) \tag{3}$$

The smaller G and D, the more pure the node (ie. all $\hat{p}_{mk}$ close to 0 or 1). A large tree T_0 can be developed until a criterion is reached (ex: no box with more than 5 obs), but this model will surely overfit the training data.

1.2 Tree pruning

To avoid overfit, we select a subtree T via weakest link pruning (or cost-complexity). Each α correspond to a subtree $T \subset T_0$ with $|T|$ terminal nodes that minimises

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \tag{4}$$

where R_m is the box corresponding to the mth terminal node.

The α term is a penalty to pay for having many terminal nodes ($\alpha = 0$ means $T = T_0$) and can be found by cross-validation (using RSS, Gini index, entropy or classification error rate).

α	0	$\rightarrow$	∞
Fit	overfitting	good	underfitting
$\lvert T \rvert$ (tree size)		$\searrow$	
Training error		$\nearrow$	
Test error		$\searrow\nearrow$	

Trees often suffer from high variance: a small change in the data can have a big impact on the tree shape.

2. Ensemble methods

Recall that averaging a set of observations reduces the variance, we can improve the tree prediction accuracy with this principle.

2.1 Bagging

Using B bootstrapped samples of the training data, we can build B different deep decision trees $\hat{f}^*$ (not pruned) and average over them.

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x) \tag{5}$$

Usually $B = 100$ is sufficient.

The bagged trees used bootstrapped samples which correspond to $\sim 2/3$ of the total training data. Hence we obtain $B/3$ out of bag (OOB) predictions for the ith obs. Averaging (regression) or taking a majority vote (classification) on these OOB prediction gives a single OOB prediction for the n obs. The MSE or classification error can then be estimated.

Although we lost the interpretability of the single decision tree, we gain a measure of variable importance by measuring the total decrease of RSS or Gini index for split performed on a given predictor as in Figure 2 (Right). Moreover, increasing B will not lead to overfitting !

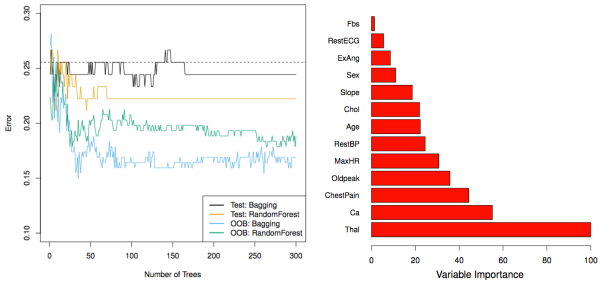

Figure 2. Left: Test errors comparison. Right: bagging variable importance

2.2 Random Forests

To further reduce the variance and improve prediction accuracy, random forest decorrelates trees (so that there is no average on correlated obs.). At each split, we only consider a fraction m/p of the predictors selected randomly. Often we select $m = \sqrt{p}$ (bagging corresponds to $m = p$).

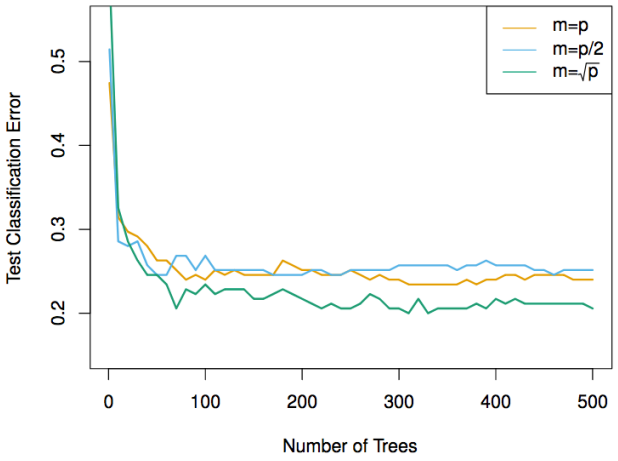

Figure 3. Left: RF test error for different m

2.3 Boosting

Boosted decision trees (BDT) are grown sequentially and fitted on residuals (no bootstrap samples). For each iteration, $\hat{f}$ is improved where it does not perform well.

---

1 Set $\hat{f}(x) = 0$ adn $r_i = y_i$ for all i in training set

2 For $b = 1, \ldots, B$ do

   a Fit a tree $\hat{f}^b$ with d splits ($|T| = d + 1$) to the training data (X, r).

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

   c Update the residuals $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$

4 Output the boosted model $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$

- B is the number of trees. Unlike bagging and RF, BDT can overfit when B is big. We can us CV to determine it
- λ is the learning rate, usually around 0.005. If too small, B needs to be bigger for good performance
- d is the number of splits in each tree, or interaction depth. $d = 1$ is like fitting an additive model.

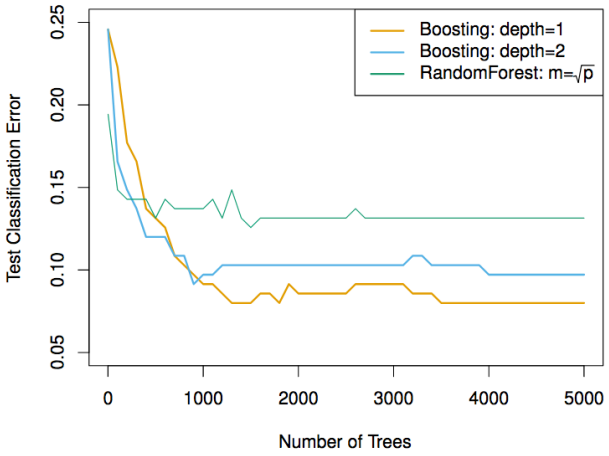

Figure 4. BDT and RF test error comparison