Chapter 8

Bagging and Boosting

8.1 Introduction

Bootstrap **agg**regat**ing** (bagging) and boosting are useful techniques to improve the predictive performance of tree models. Boosting may also be useful in connection with many other models, e.g. for additive models with high-dimensional predictors; whereas bagging is most prominent for improving tree algorithms.

8.2 Bagging

Consider a base procedure, e.g. a tree algorithm such as CART, which yields a function estimate

$$\hat{g}(\cdot) : \mathbb{R}^p \to \mathbb{R}$$

(or $\hat{g}(\cdot)$ takes values in $[0, 1]$ for classification).

8.2.1 The bagging algorithm

Bagging works as follows.

1. Generate a bootstrap sample

   $$(X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*)$$

   and compute the bootstrapped estimator $\hat{g}^*(\cdot)$.

2. Repeat step 1 B times, yielding

   $$\hat{g}^{*1}(\cdot), \ldots, \hat{g}^{*B}(\cdot).$$

3. Aggregate the bootstrap estimates

   $$\hat{g}_{Bag}(\cdot) = B^{-1} \sum_{i=1}^{B} \hat{g}^{*i}(\cdot).$$

The bagging algorithm is nothing else than an approximation

$$\hat{g}_{Bag}(\cdot) \approx \mathbb{E}^*[\hat{g}^*(\cdot)]$$

which can be made arbitrarily good by increasing B. The novel point is that we should use now $\mathbb{E}^*[\hat{g} * (\cdot)]$ as a new estimator.

A trivial identity hints at some properties of bagging: write (the theoretical version of bagging with $B = \infty$)

$$\begin{aligned}\hat{g}_{Bag}(\cdot) &= \hat{g}(\cdot) + (\mathbb{E}^*[\hat{g}^*(\cdot)] - \hat{g}(\cdot)) \\ &= \hat{g}(\cdot) + \text{ bootstrap bias estimate.}\end{aligned}$$

Instead of subtracting the bootstrap bias estimate, we are adding it! What we can hope for is a variance reduction at the price of a higher bias. This turns out to be true if $\hat{g}(\cdot)$ is a tree-based estimator.

8.2.2 Bagging for trees

It can be shown that for tree-based estimators $\hat{g}(\cdot)$,

$$\mathrm{Var}(\hat{g}_{Bag}(x)) \overset{\text{asymp.}}{<} \mathrm{Var}(\hat{g}(x)),$$

for very many x. Thus, bagging is a variance reduction technique. The reason for this is that a bagged tree turns out to be a product of probit functions $\Phi(d - \cdot)$ instead of indicator functions $\mathbf{1}_{[\cdot \leq d]}$. This causes a variance reduction at the price of some bias. For example,

$$\mathrm{Var}(\mathbf{1}_{[X \leq d]}) = \mathbb{P}[X \leq d](1 - \mathbb{P}[X \leq d]).$$

If $X \sim \mathcal{N}(0,1)$ and $d = 0$, the latter quantity equals 1/4. On the other hand,

$$\mathrm{Var}(\Phi(-X)) = \mathrm{Var}(U) = 1/12, \quad U \sim \text{ Unif.}([0,1]),$$

which reduces the variance by the factor 3!

We should use large trees for bagging, because the variance reduction due to bagging asks for a large tree to balance the bias-variance trade-off.

8.2.3 Subagging

Subagging (**sub**sample **agg**regat**ing**) is a version of bagging: instead of drawing a bootstrap sample in step 1 of the bagging algorithm, we draw

$$(X_1^*, Y_1^*), \ldots, (X_m^*, Y_m^*) \text{ without replacement}$$

for some $m < n$. In some simple cases, it can be shown that $m = [n/2]$ is equivalent to bagging. Thus, subagging with $m = [n/2]$ can be viewed as a computationally cheaper version of bagging.

We consider a dataset about ozone concentration with $p = 8$ predictor variables (different from the previous ozone dataset). The performance of (su-)bagging for trees and MARS are shown in Figure 8.1. We see that bagging improves a regression tree substantially while it does not improve MARS at all (for this example).

The main drawback of bagging is the loss of interpretation in terms of a tree. It is by no means simple to interpret a linear combination of trees.

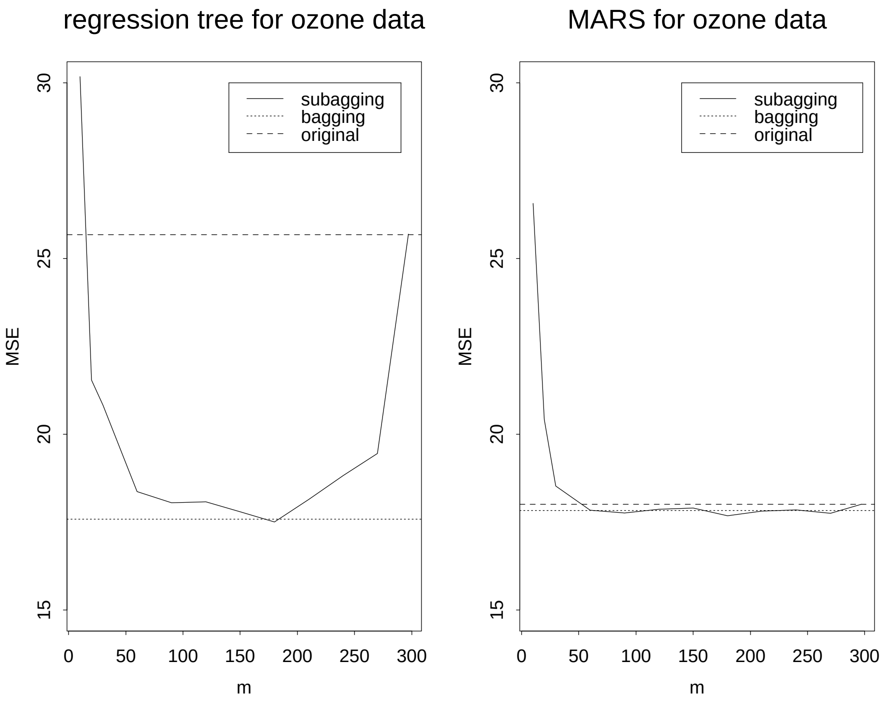

Figure 8.1: Mean squared error performance for a large regression tree and MARS and their (su-)bagged versions for an ozone data (different from the previous one).

8.3 Boosting

Boosting is a very different method to generate multiple predictions (function estimates) and combine them linearly. As with bagging, we have a base procedure yielding function estimates $\hat{g}(\cdot)$ (e.g. a tree algorithm).

8.3.1 L_2Boosting

The so-called L_2Boosting method (for regression) works as follows.

1. Fit a first function estimate from the data $\{(X_i, Y_i);\ i = 1, \ldots, n\}$ yielding a first function estimate $\hat{g}_1 \cdot)$.
   Compute residuals
   $$U_i = Y_i - \hat{g}_1(X_i)\ (i = 1, \ldots, n).$$
   Denote by $\hat{f}_1(\cdot) = \nu \hat{g}_1(\cdot)\ (0 < \nu \leq 1)$.

2. For $m = 2, 3, \ldots, M$ do:
   Fit the residuals
   $$(X_i, U_i) \to \hat{g}_m(\cdot)$$
   and set
   $$\hat{f}_m(\cdot) = \hat{f}_{m-1}(\cdot) + \nu \hat{g}_m(\cdot).$$
   Compute the current residuals
   $$U_i = Y_i - \hat{f}_m(X_i)\ (i = 1, \ldots, n).$$

The shrinkage parameter ν can be chosen to be small, e.g. $\nu = 0.1$. The stopping parameter M is a tuning parameter of boosting. For ν small we typically have to choose M large.

Boosting is a bias reduction technique, in contrast to bagging. Boosting typically improves the performance of a single tree model. A reason for this is that we often cannot construct trees which are sufficiently large due to thinning out of observations in the terminal nodes. Boosting is then a device to come up with a more complex solution by taking linear combination of trees. In presence of high-dimensional predictors, boosting is also very useful as a regularization technique for additive or interaction modeling.