

Statistics 452: Statistical Learning and Prediction

Chapter 8, Part 3: Boosting

Brad McNeney

2018-11-02

Introduction to Boosting

- ▶ Reference: Hastie, Tibshirani and Friedman (2001). The Elements of Statistical Learning (hereafter ESL).
- ▶ Motivation for boosting: Combine many “weak” classifiers to produce a powerful “committee”.
 - ▶ Similar in this respect to bagging, but otherwise fundamentally different.
- ▶ A weak classifier is one that does little better than guessing.
 - ▶ On its own a weak classifier is not useful, but if applied *sequentially*, it can produce a powerful classifier.

Example Boosting Algorithm: AdaBoost.M1

- ▶ Due to Freund and Schapire (1997).
- ▶ Suppose two outcome classes $Y = -1$ or 1 and a “base” classifier that produces a prediction.
 - ▶ Need not be a decision tree classifier at this point.
- ▶ Sequentially apply the classifier to modified versions of the data (more on next slide), leading to a sequence of weak classifiers $G_m(x)$; $m = 1, \dots, M$ which are weighted to give final predictions.

AdaBoost Weighting

- ▶ Combine predictions with a weighted majority vote

$$G(x) = \text{sign} \left(\sum_{m=1}^M \alpha_m G_m(x) \right),$$

which classifies as 1 if weighted sum > 0 and -1 otherwise.

- ▶ The classifier weights α_m are computed by the algorithm to give higher weight to more accurate classifiers.
- ▶ Modify the data at each boosting step by applying observation weights w_1, \dots, w_n .
 - ▶ Initially all weights are equal.
 - ▶ At step m , observations that were misclassified at step $m - 1$ are up-weighted.
 - ▶ As we go, observations that are difficult to classify receive more and more weight, forcing the weak classifier to focus on them.

AdaBoost algorithm

- ▶ Algorithm 10.1 of ESL (page 339).

Algorithm 10.1 *AdaBoost.M1*.

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \dots, N$.
 2. For $m = 1$ to M :
 - (a) Fit a classifier $G_m(x)$ to the training data using weights w_i .
 - (b) Compute
$$\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
 - (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
 - (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \dots, N$.
 3. Output $G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$.
-

Schematic

Elements of Statistical Learning (2nd Ed.) ©Hastie, Tibshirani & Friedman 2009 Chap 10

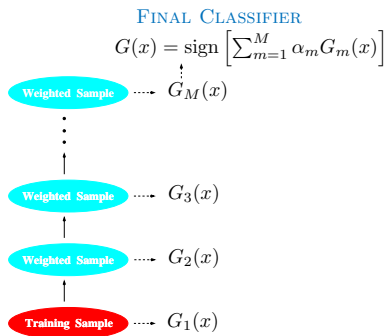


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

AdaBoost as an Additive Model

- ▶ Let $b(x; \gamma)$ be the base classifier for parameters γ .
 - ▶ Let γ_m denote the values at step m , so that $G_m(x) = b(x; \gamma_m)$ is the classifier at step m . This is a basis function.
- ▶ The classifier weights are the coefficients of the basis functions.
- ▶ The additive model is

$$f(x; \alpha, \gamma) = \sum_{m=1}^M \alpha_m b(x; \gamma_m)$$

- ▶ We would like to find the coefficients $\alpha = (\alpha_1, \dots, \alpha_M)$ and $\gamma = (\gamma_1, \dots, \gamma_M)$ that minimize a loss function,

$$\sum_{i=1}^n L(y_i, f(x_i; \alpha)).$$

- ▶ We are used to the squared-error loss $L(y, f(x)) = (y - f(x))^2$ and misclassification $L(y, f(x)) = I(y \neq f(x))$; it turns out (ESL, Section 10.4) that AdaBoost uses exponential loss function $L(y, f(x)) = \exp(-yf(x))$.

Forward Stagewise Additive Modelling

- ▶ Finding the best values of α, γ is a difficult problem.
- ▶ Approximate the solution by a greedy algorithm that sequentially adds the best new basis function, without adjusting the coefficients of those previously added.
 1. Initialize $f_0(x) = 0$.
 2. For $m = 1 : M$
 - 2.1 Find the α_m and γ_m that minimize
$$\sum_{i=1}^n L(y_i, f_{m-1}(x_i) + \alpha b(x_i; \gamma))$$
 - 2.2 Set $f_m(x) = f_{m-1}(x) + \alpha_m b(x; \gamma_m)$
 3. Return $\hat{f}(x) = f_M(x)$.

Boosting Decision Trees

- ▶ The parameters of a decision tree are the disjoint regions (obtained by recursive partitioning) and the values assigned to each region.
- ▶ Let $T(x; \gamma)$ be a tree.
- ▶ The boosted tree model is a sum

$$f_M(x) = \sum_{m=1}^M T(x; \gamma_m)$$

(**no weighting**), where the trees at step m are fit according to the forward stagewise algorithm.

- ▶ At step m we find the γ_m that minimizes

$$\sum_{i=1}^n L(y_i, f_{m-1}(x_i) + T(x_i; \gamma)) \quad (1)$$

and take $f_m(x) = f_{m-1}(x) + T(x; \gamma_m)$.

Boosting Regression Trees

- ▶ If a regression tree and the loss is squared-error loss,

$$\begin{aligned} L(y_i, f_{m-1}(x_i) + T(x_i; \gamma)) &= (y_i - f_{m-1}(x_i) - T(x_i; \gamma))^2 \\ &= (r_i^{(m-1)} - T(x_i; \gamma))^2, \end{aligned}$$

where $r_i^{(m-1)}$ is the i th residual from step $m - 1$.

- ▶ Solve (1) by fitting a tree to the residuals (Our text, Alg. 8.2).
- ▶ Note: As a basis function, $T(x; \gamma)$ could, in general, depend on all predictors, which would make the boosted model not additive in the sense of Chapter 7.
 - ▶ When the trees have only two leaves (i.e., one split on one variable), the boosted model is additive in the sense of Chapter 7.

Gradient Boosting

- ▶ With loss functions other than squared-error and exponential, the solution to (1) is more challenging.
- ▶ A general, but approximate algorithm based on ideas from optimization is called gradient boosting.
 - ▶ A description is beyond the scope of this course.
 - ▶ We use the implementation in the `gbm` package.
 - ▶ A more modern implementation of boosting trees is in the package `xgboost` (eXtreme Gradient Boosting). More flexible, computations highly optimized and parallelized, but same basic idea.

Choosing the Depth of the Trees

- ▶ Set the tree depth to be the same for all trees.
- ▶ Could consider the depth as a tuning parameter and choose it by cross-validation.
- ▶ Text and software suggest $d = 1$ is often fine.
 - ▶ Software calls d the interaction depth. For $d > 1$ each tree depends on more than one variable and would represent an “interaction”.

Shrinkage

- ▶ Large M will lead to overfitting.
- ▶ Can select M as a tuning parameter, but experience has shown that it is better to take a large M and shrink the contributions of each tree by a factor λ ; that is, take
$$f_m(x) = f_{m-1}(x) + \lambda T(x; \gamma_m).$$

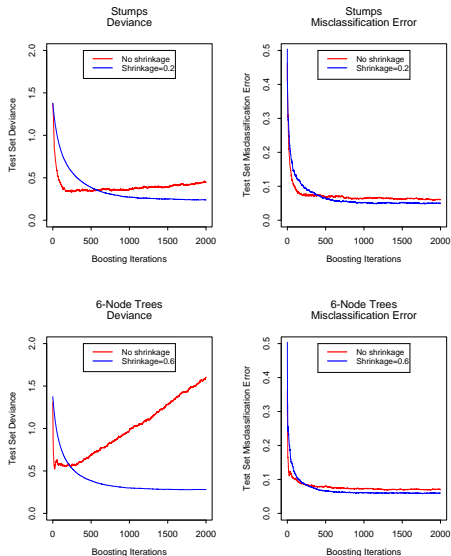
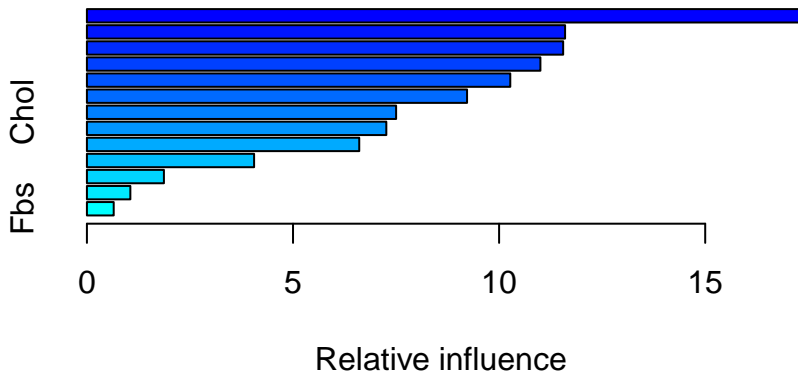


FIGURE 10.11. Test error curves for simulated example (10.2) of Figure 10.9, using gradient boosting (MART). The models were trained using binomial deviance, either stumps or six terminal-node trees, and with or without shrinkage. The left panels report test

Example: Heart Data

- ▶ Recall that the best tree fit the to Heart data had test-set misclassification rate about 27%,
- ▶ Random forest had a test-set misclassification of about 17%.

```
library(gbm)
hboost <- gbm(I(AHD=="Yes") ~ ., data=Heart[train,],
              n.trees=5000,distributio="bernoulli") # default shrinkage = 0.1
summary(hboost)
```



```
##          var    rel.inf
## ChestPain ChestPain 17.3498534
## Age       Age      11.6008572
## MaxHR     MaxHR    11.5538027
## Thal      Thal     11.0018459
## Ca        Ca       10.2708258
## RestBP    RestBP   9.2227245
```



```
boo.hpred <- predict(hboost,newdata=Heart[-train,],  
                    n.trees=5000,type="response")  
boo.hpred <- (boo.hpred>0.5)  
tt <- table(boo.hpred,Heart[-train,]$AHD)  
tt
```

```
##  
## boo.hpred No Yes  
##      FALSE 39  16  
##      TRUE  13  31
```

```
sum(tt[row(tt) != col(tt)]) / sum(tt)
```

```
## [1] 0.2929293
```

Change Shrinkage

```
hboost <- gbm(I(AHD=="Yes") ~ ., data=Heart[train,],  
              n.trees=5000,distribution="bernoulli",shrinkage=.2)  
boo.hpred <- predict(hboost,newdata=Heart[-train,],  
                     n.trees=5000,type="response")  
boo.hpred <- (boo.hpred>0.5)  
tt <- table(boo.hpred,Heart[-train,]$AHD)  
tt
```

```
##  
## boo.hpred No Yes  
##      FALSE 41  17  
##      TRUE  11  30
```

```
sum(tt[row(tt) != col(tt)]) / sum(tt)
```

```
## [1] 0.2828283
```