Statistics 452: Statistical Learning and Prediction

Chapter 8, Part 3: Boosting

Brad McNeney

2017-11-06

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 $G_m(x)$; $m = 1, \ldots, M$ 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 classifis as 1 if the weighted sum is > 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 $w_1, \ldots, w_n$.
  - 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.
- Full details in Algorithm 10.1 of ESL (page 301).

Schematic

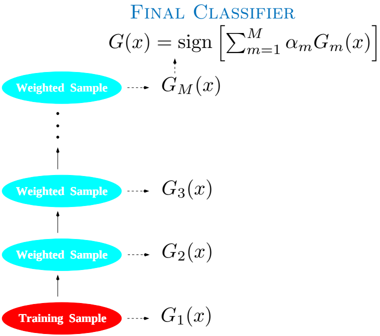


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

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 $L(y, f(x)) = (y - f(x))^2$, but others are possible.

Forward Stagewise Additive Modelling

- 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 $\hat{f}(x) = f_M(x)$.

Example Forward Stagewise Additive Model

- One can show (ESL Section 10.4) that AdaBoost is forward stagewise additive modelling with the exponential loss function $L(y, f(x)) = \exp(-yf(x))$.

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

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

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

Boosting Regression Trees

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

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

  where $r_i^{(m-1)}$ is the ith residual from step $m-1$.
- Solve (1) by fitting a tree 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.

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

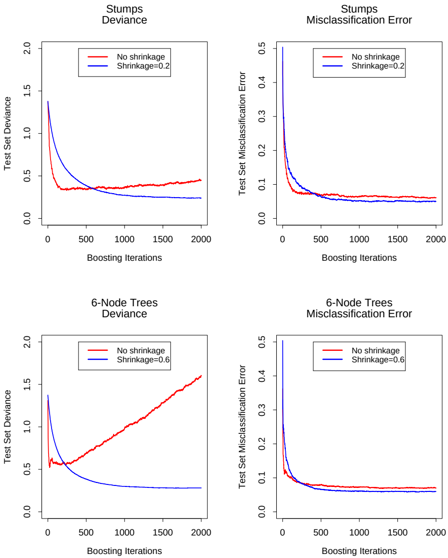


FIGURE 10.11. *Test error curves for simulated example (10.2) of Figure 10.9, using gradient boosting (MART). The models were trained using binomial deviance, either stumps or six terminal-node trees, and with or without shrinkage. The left panels report test*

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%.

```
## [1] 297  14
```

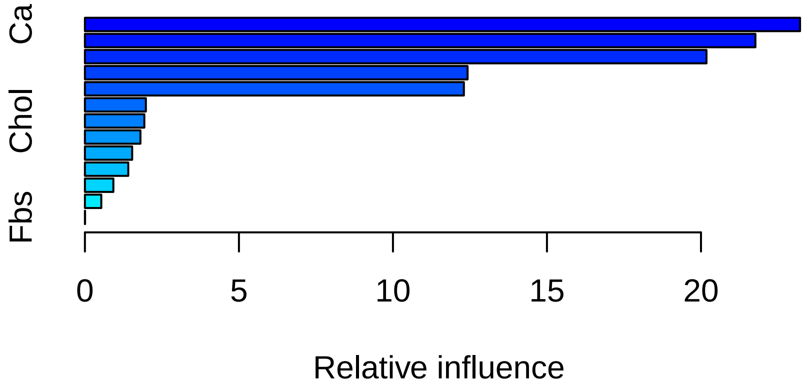
```
## [1] -1.0 -0.2  0.2  0.4  0.8  1.4
```

```
## [1] -0.2  0.4  0.4  0.8  0.8  0.8
```

```
## [1] -1.0 -0.2  0.2  0.2  0.2  0.2
```

```
## [1] -0.2 -0.2 -0.2  0.2  0.4  0.8
```

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




```
##                 var      rel.inf
## Ca               Ca 23.215816764
## Thal           Thal 21.766668524
## ChestPain ChestPain 20.178691076
## Oldpeak     Oldpeak 12.423059066
## MaxHR         MaxHR 12.302340643
```

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

```
##
## boo.hpred No Yes
##     FALSE 50  12
##     TRUE   4  33
```

```r
16/nrow(Heart[-train,]) # Lowest so far.
```

```
## [1] 0.1616162
```

Change Shrinkage

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

```
##
## boo.hpred No Yes
##     FALSE 47  13
##     TRUE   7  32
```

```r
16/nrow(Heart[-train,]) # Worse
```

```
## [1] 0.1616162
```