Improving Trees

Thomas Lonon

Financial Engineering/Financial Analytics
Stevens Institute of Technology

August 23, 2018

Bagging (Revisited)

The bagging estimate is defined as

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

for regression and

$$\hat{f}_{\text{bag}}(x) = \max_{k} \sum_{b=1}^{B} \mathbb{I}_{\{\hat{f}^{*b}(x)=k\}}$$

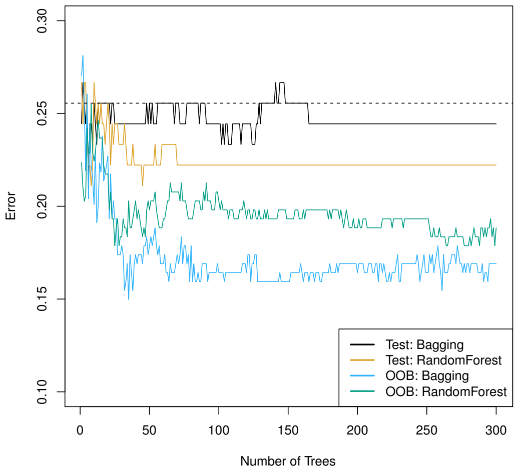
for classification where $k \in K$ are the possible classes.

Out-of-Bag Error

1. Predict response for the i^{th} observation using the trees that did not utilize this observation in their construction
2. Take the average of these responses (or majority vote) to get a single OOB prediction for each of the n observations
3. Use these to calculate the OOB MSE or the OOB Classification error

Random Forests

1. Create a bootstrapped sample of the observations
2. Build a tree, but at each split, only consider $m \leq p$ predictors
3. Use the trees built in this way to calculate your prediction like in bagging.


Error
0.10
0.15
0.20
0.25
0.30
Number of Trees
0
50
100
150
200
250
300
Test: Bagging
Test: RandomForest
OOB: Bagging
OOB: RandomForest

AdaBoost.M1

Consider a two-class problem with output variable $Y \in \{-1, 1\}$.

For a given vector of predictor variables X and a classifier $G(X)$ taking one of the two values, we have the error rate of the predictor as:

$$\overline{err} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{\{y_i \neq G(x_i)\}}$$

For a sequence of weak classifiers $G_m(x), m = 1, \ldots, M$, these are combined using a weighted majority to produce a final prediction:

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

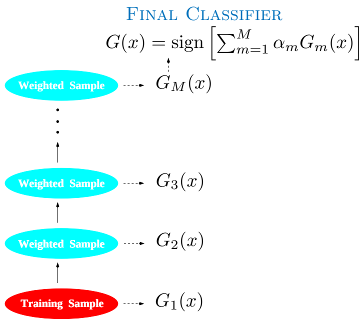


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Algorithm 10.1: AdaBoost.M1

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

[2]

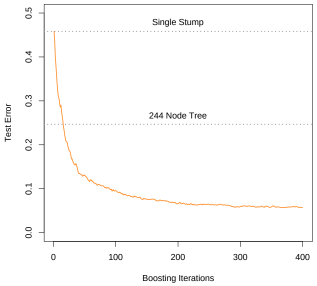


FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

General Idea

More generally, basis functions expansions take the form:

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

where $\beta_m, m = 1, 2, \ldots, M$ are the expansion coefficients, and $b(x; \gamma) \in \mathbb{R}$ are usually simple functions of the multivariate argument x, characterized by parameters γ.[2]

Possible Applications

- In single-hidden-layer neural networks, $b(x; \gamma) = \sigma(\gamma_0 + \gamma_1^T x)$, where $\sigma(t) = 1/(1 + e^{-t})$ is the sigmoid function, and γ parameterizes a linear combination of the input variables
- Multivariate adaptive regression splines uses truncated-power spline basis functions where γ parameterizes the variables and values for the knots.
- For trees, γ parameterizes the split variables and split points at the internal nodes, and the predictions at the terminal nodes

[2]

Fitting

Typically these models are fit by minimizing a loss function averaged over the training data:

$$\min_{\{\beta_m,\gamma_m\}_1^M} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i;\gamma_m)\right)$$

Or a simple alternative being fitting a single basis function:

$$\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, \beta b(x_i;\gamma))$$

[2]

Algorithm 10.2: Forward Stagewise Additive Modeling

```
1. Initialize $f_0(x) = 0$
2. For $m = 1$ to $M$
    2.1 Compute
        $$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$$
    2.2 Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$
```

[2]

Squared Error Loss

$$L(y, f(x)) = (y - f(x))^2$$

we have:

$$\begin{aligned} L(y_i; f_{m-1}(x_i) + \beta b(x_i; \gamma)) &= (y_i - f_{m-1}(x_i) - \beta b(x_i; \gamma))^2 \\ &= (r_{im} - \beta b(x_i; \gamma))^2 \end{aligned}$$

where $r_{im} = y_i - f_{m-1}(x_i)$ is the residual of the current model on the i^{th} observation

Exponential Loss

$$L(y, f(x)) = e^{-yf(x)}$$

Using this results in AdaBoost.M1 being equivalent to forward stagewise additive modeling. (as will be shown)

We use the basis functions as the individual classifiers, and so using the exponential loss function, we have to solve:

$$(\beta_m, G_m) = \arg\min_{\beta, G} \sum_{i=1}^{N} e^{-y_i(f_{m-1}(x_i)+\beta G(x_i))}$$

for the classifier G_m and the coefficient β_m

We can express this as:

$$(\beta_m, G_m) = \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} e^{-\beta y_i G(x_i)}$$

with $w_i^{(m)} = e^{-y_i f_{m-1}(x_i)}$.

This solution can be obtained in two steps. First for $\beta > 0$ we have:

$$G_m = \arg\min_{G} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}_{\{y_i \neq G(x_i)\}}$$

Our criterion in (β_m, G_m) becomes:

$$e^{-\beta} \sum_{y_i = G(x_i)} w_i^{(m)} + e^{\beta} \sum_{y_i \neq G(x_i)} w_i^{(m)}$$

which can be written as:

$$(e^{\beta} - e^{-\beta}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}_{\{y_i \neq G(x_i)\}} + e^{-\beta} \sum_{i=1}^{N} w_i^{(m)}$$

If we plug this G_m into the earlier equation, we have:

$$\beta_m = \frac{1}{2}\log\frac{1-\text{err}_m}{\text{err}_m}$$

where

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)}\mathbb{I}_{\{y_i \neq G_m(x)\}}}{\sum_{i=1}^{N} w_i^{(m)}}$$

The approximation is then updated

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x)$$

Boosting Trees

As a reminder, once we have partitioned a space into regions $R_j, j = 1, \ldots, J$, we then assign a constant γ_j to each region such that our predictive rule is:

$$x \in R_j \Rightarrow f(x) = \gamma_j$$

So our tree can be formally expressed as:

$$T(x; \Theta) = \sum_{j=1}^{J} \gamma_j \mathbb{I}_{\{x \in R_j\}}$$

with $\Theta = \{R_j, \gamma_j\}_1^J$

To find the parameters for the optimal fitting tree, we minimize the empirical risk:

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i; \gamma_j)$$

Suboptimal Problems

Finding γ_j given R_j: Given the R_j, estimating γ_j is typically trivial and often $\hat{\gamma} = \bar{y}_j$

Finding R_j: The difficult part. Typical strategy is to use a greedy, top-down recursive algorithm to find the R_j

To optimize the R_j we sometimes need to look at the criterion

$$\widetilde{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} \widetilde{L}(y_i, T(x_i, \Theta))$$

The boosted tree model is a sum of such trees

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

induced in a forward stagewise manner (Algorithm 10.2)[2]

At each step we must solve:

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i, \Theta_m))$$

for $\Theta_m = \{R_{jm}, \gamma_{jm}\}_1^{J_m}$ given current model $f_{m-1}(x)$.

Given regions R_{jm} finding the optimal constants γ_{jm} is straightforward:

$$\hat{\gamma}_{jm} = \arg\min_{\gamma_{jm}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma_{jm})$$

Finding the regions is difficult, but in special cases it can be simplified.

For squared-error loss, the solution is the regression tree that predicts the current residuals $y_i - f_{m-1}(x_i)$ and so is the approach outlined in Algorithm 8.2.

For two-class classification and exponential loss, the stagewise approach gives rise to the AdaBoost method for boosting classification trees in Algorithm 10.1.

Algorithm 8.2: Boosting for Regression Trees

```
1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all $i$ in the training set
2. For $b = 1, \ldots, B$ repeat:
    2.1 Fit a tree $\hat{f}^b$ with $d$ splits ($d + 1$ terminal nodes) to the
        training data $(X, r)$
    2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:
            $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$
    2.3 Update the residuals,
            $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$
3. Output the boosted model
            $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$
```

[1]

[1] Trevor Hastie Gareth James, Daniela Witten and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Number v. 6. Springer, 2013.

[2] Robert Tibshirani Trevor Hastie and Jerome Friedman. *The Elements of Stastical Learning: Data Mining, Inference, and Prediction*. Number v.2 in Springer Series in Statistics. Springer, 2009.