Boosting and Additive Models (part 1)

Matthew S. Shotwell, Ph.D.

Department of Biostatistics
Vanderbilt University School of Medicine
Nashville, TN, USA

March 27, 2020

Boosting

- combines many “weak” learners $\rightarrow$ powerful “committee”
- iteratively add “weak” learners by targeting regions of the input space where predictions were poor at previous iteration
- start with binary classification example: AdaBoost.M1

AdaBoost.M1

- AdaBoost.M1: popular boosted tree-based binary classifier
- binary output: $Y \in \{-1, 1\}$
- predictors: X
- classifier: $G(X)$ (returns -1 or 1)
- using zero-one loss:

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

- $\overline{\mathrm{err}}$ here is misclassificaiton rate

AdaBoost.M1

- a “weak” classifier has $\overline{\mathrm{err}}$ not much better than random guess
- boosting is to sequentially apply a weak classifier to repeatedly modified versions of the data, thereby producing a sequence of weak classifiers $G_m(x)$ for $m = 1, 2, \ldots, M$.

AdaBoost.M1

- the sequence of weak classifiers is combined using weighted majority vote:

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

- $G(x)$ returns -1 or 1
- weights α_m are selected as part of boosting algorithm; they upweight more accurate classifiers

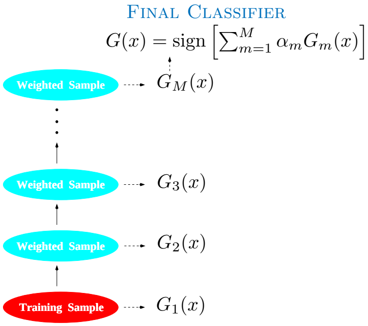

FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

AdaBoost.M1

- at each iteration, training data are weighted
- initially weights $w_1, \ldots, w_N = 1/N$
- weak learner is then applied to weighted training data
- at next iteration, misclassified observations get larger weights
- repeatedly misclassified obs get larger and larger weights

Algorithm 10.1 *AdaBoost.M1.*

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

Algorithm 10.1 *AdaBoost.M1.*

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

Algorithm 10.1 *AdaBoost.M1.*

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

Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
                                   fit weighted stump (e.g., using rpart)
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
                                   misclassification rate for weighted stump
        $$\mathrm{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
                                   model contributes more
                                   to committee if training
                                   error is small.
    (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.
```

Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:                              fit weighted stump (e.g., using rpart)
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute                                      misclassification rate for weighted stump
            $$\mathrm{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
                                                     model contributes more
                                                     to committee if training
    (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.   error is small.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
                                                     misclassified obs get
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.   bigger weight for next
                                                     iteration
```

AdaBoost.M1

- α_m is log odds of correct classification by $G_m(x)$
- err_m always $\geq$ 0.5, thus $a_m \geq 0$
- weight update:

$$w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq G_m(x_i))]$$

$$w_i \leftarrow \begin{cases} w_i \left(\frac{1-\mathrm{err}_m}{\mathrm{err}_m}\right) & \text{if } y_i \text{ misclassified} \\ w_i & \text{otherwise} \end{cases}$$

AdaBoost.M1

- unlike bagging, boosting is adaptive
- easy to overfit as M grows
- tuning parameters:
  - number of trees/iterations M
  - inherits tuning parameters of weak learner (e.g., tree depth)

AdaBoost.M1 example

- let features $X_1, \ldots, X_{10}$ be random normal variables
- let tartet Y be deterministic such that

$$y = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} X_j^2 > 10 \\ -1 & \text{otherwise} \end{cases}$$

- model is not additive in inputs
- high order interactions of inputs
- difficult classification problem
- use "stump" as weak learner (tree w/ 1 split)

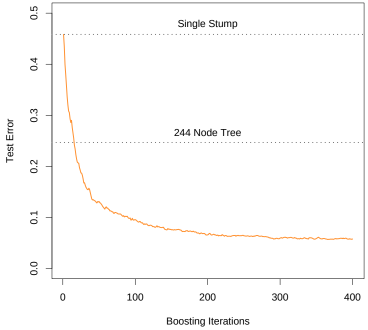

FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

Code example

`boosting-trees.R`