Boosting and Additive Models (part 2)

Matthew S. Shotwell, Ph.D.

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

March 30, 2020

Boosting fits an additive model

- additive model:

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m) \quad \text{where} \quad G(x) = \text{sign}[f(x)]$$

- for AdaBoost.M1, the notation was

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

- for each m

$$(\hat{\beta}_m, \hat{\gamma}_m)_1^M = \text{arg min}_{(\beta_m, \gamma_m)_1^M} \sum_{i=1}^{N} L(y_i, f(x_i))$$

- solving this is hard; use an algorithm to find approx solution (i.e., boosting)

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

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

for trees:

- $b(x, \gamma_m)$ is binary classification tree ($G(x) \in \{-1, 1\}$)
- γ_m is split information (R_m)
- β_m is the tree weight (α_m)

FSAM with squared-error loss

- using squared-error loss:

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

- r_{im} is the residual for observation i using model f_{m-1}
- step 2 of FSAM algorithm is a least-squares problem (easy!)

FSAM using exponential loss

- exponential loss:

$$L(y, f(x)) = \exp(-yf(x))$$

- if y and $f(x)$ have same sign, then $\exp(-yf(x)) \leq 1$ and vice versa
- $yf(x)$ is called the 'margin' in this context ($Y \in \{-1, 1\}$)
- the margin acts like a residual

FSAM using exponential loss

Consider binary classification ($Y \in \{-1, 1\}$) with exponential loss

- step 2.a. from FSAM algorithm:

$$
\begin{aligned}
(\beta_m, G_m) &= \arg\ \min_{(\beta,G)} \sum_{i=1}^{N} \exp[-y_i[f_{m-1}(x_i) + \beta G(x_i)]] \\
&= \arg\ \min_{(\beta,G)} w_{i(m-1)} \exp[-y_i \beta G(x_i)]
\end{aligned}
$$

where

$$
w_{i(m-1)} = \exp[-y_i f_{m-1}(x_i)]
$$

FSAM using exponential loss

- given G_m (see HTF Ex. 10.1):

$$\beta_m = \frac{1}{2} \log \left[\frac{1 - \mathrm{err}_m}{\mathrm{err}_m} \right]$$

- note: $w_{i(m-1)} = \exp[-y_i f_{m-1}(x_i)]$
- weight at next iteration is

$$\begin{aligned} w_{i(m)} &= \exp[-y_i(f_m(x_i)] \\ &= \exp[-y_i(f_{m-1}(x_i) + \beta_m G_m(x_i))] \\ &= w_{i(m-1)} \exp[-y_i \beta_m G_m(x_i)] \\ &\propto w_{i(m-1)} \exp[\alpha_m I(y_i \neq G_m(x_i))] \end{aligned}$$

- AdaBoost.M1 is equivalent to FSAM using exponential loss!

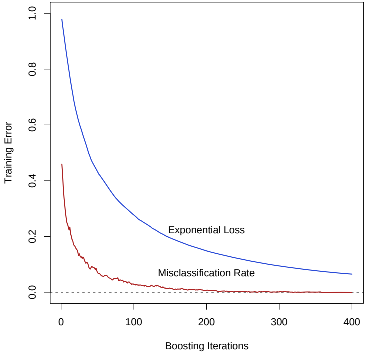

FIGURE 10.3. *Simulated data, boosting with stumps: misclassification error rate on the training set, and average exponential loss:* $(1/N)\sum_{i=1}^{N}\exp(-y_i f(x_i))$. *After about* 250 *iterations, the misclassification error is zero, while the exponential loss continues to decrease.*

Why exponential loss?

- AdaBoost.M1 and FSAM connection coincidental
- in binary classification problem ($Y \in \{-1, 1\}$), what estimator does exponential loss give (see Ex. 10.2)?

$$\begin{aligned}\hat{f}(x) &= \arg\min_{f(x)} E_{Y|X}[\exp(-Yf(X))] \\ &= \frac{1}{2}\log\frac{P(Y=1|X)}{P(Y=-1|X)}\end{aligned}$$

- sign of $\hat{f}(x)$ makes sense as classification rule
- exponential loss is like a smooth version of the zero-one loss
- when margin $yf(x)$ positive, small loss

Binomial deviance loss

- binomial log-likelihood where $Y' = (Y+1)/2 \in \{0,1\}$:

$$l(Y, p(x)) = Y' \log p(x) + (1 - Y') \log(1 - p(x))$$
$$l(Y, f(x)) = -\log(1 + \exp(-2Y f(x)))$$

  where $f(x)$ is one-half the log odds
- the binomial deviance loss:

$$L(Y, f(x)) = -l(Y, f(x)) = \log(1 + \exp(-2Y f(x)))$$

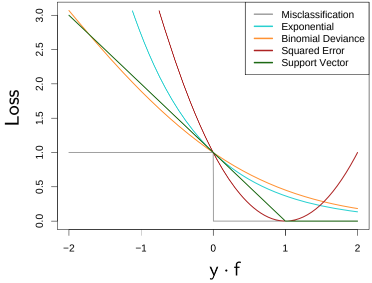

FIGURE 10.4. *Loss functions for two-class classification. The response is* $y = \pm 1$*; the prediction is* f*, with class prediction* $\text{sign}(f)$*. The losses are misclassification:* $I(\text{sign}(f) \neq y)$*; exponential:* $\exp(-yf)$*; binomial deviance:* $\log(1 + \exp(-2yf))$*; squared error:* $(y - f)^2$*; and support vector:* $(1 - yf)_+$ *(see Section 12.3). Each function has been scaled so that it passes through the point* $(0, 1)$*.*

Robustness: exponential vs. deviance

- deviance loss is less "severe" version of exponential loss
- for $yf(x) < 0$ exponential loss is... exponential, but deviance loss becomes linear:

$$\log(1 + \exp(-2Yf(x))) \approx \log(\exp(-2Yf(x))) = -2Yf(x)$$

- exponential loss allows big influence of observations with big negative margin, whereas deviance is less sensitive (i.e., robust)
- AdaBoost.M1 performance degrades when there are big outliers (i.e., with big margins)
- absolute error loss is robust (versus squared error loss) in regression problems for similar reason

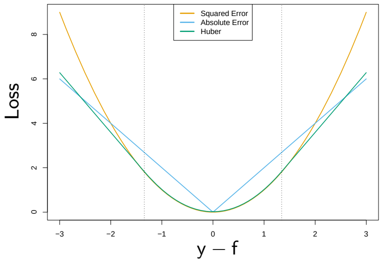

FIGURE 10.5. *A comparison of three loss functions for regression, plotted as a function of the margin $y-f$. The Huber loss function combines the good properties of squared-error loss near zero and absolute error loss when $|y-f|$ is large.*