Boosting

Wei Li

Syracuse University

Spring 2021

Boosting

Forward stagewise modeling

Exponential Loss for classification

Why exponential loss?

Robust loss functions for classification

Robust loss functions for regression

Boosting

Boosting

Boosting: like a committee of weak learners, evolving over time, with members cast a weighted vote.

$\tau > 0$ arbitrarily small

- strong learners: for large dataset, the classifier can arbitrarily accurately learn the target function with probability $1 - \tau$
- weak learners: for large dataset, the classifier can barely learn the target function with probability $\frac{1}{2} + \tau$.

Can we construct a strong learner from weak learners and how?

Motivation of boosting: combines the outputs of many weak classifiers to produce a powerful “committee”.

- Similar to bagging and other committee-based approaches
- Originally designed for classification problems, but can also be extended to regression problem.

Practical experience suggests that boosted trees have highly competitive performance as random forests.

Consider the two-class problem $Y \in \{-1, 1\}$:

The general paradigm of boosting:

- Sequentially apply the weak classification algorithm to repeatedly modified versions of the data (re-weighted data)
- produces a sequence of weak classifiers

$$G_m(x), \quad m = 1, 2, \ldots, M$$

- The predictions from G_m 's are then combined through a weighted majority vote to produce the final prediction

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

Here $\alpha_1, \ldots, \alpha_M \geq 0$ are computed by the boosting algorithm.

AdaBoost.M1 or discrete AdaBoost

AdaBoost algorithm (Freund and Schapire, 1997):

1. Initially the observation weights $w_i = 1/n, i = 1, \ldots, n$.
2. For m = 1 to M
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i. For example, $G_m = \arg min_G \sum_{i=1}^{n} w_i I(y_i \neq G(x_i))$
   (b) Compute the *weighted error rate*

$$\mathrm{err}_m = \frac{\sum_{i=1}^{n} w_i I\left(y_i \neq G_m\left(x_i\right)\right)}{\sum_{i=1}^{n} w_i}$$

   (c) Compute the *importance* of G_m as

$$\alpha_m = \log\left(\frac{1 - \mathrm{err}_m}{\mathrm{err}_m}\right)$$

   (d) Update $w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot I\left(y_i \neq G_m\left(x_i\right)\right)\right], i = 1, \ldots, n$
3. Output $G(x) = \mathrm{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$.

- The weights α_m 's weigh the contribution of each G_m.
- The higher (lower) err $_m$, the smaller (larger) α_m.
- The w_i increases for x_i misclassified by G_m. The more important classifier G_m is (thus smaller α_i), the smaller increase of the weight in this w_i.

Forward stagewise modeling

Forward stagewise modeling

The key is that the Adaboost is equivalent to fitting an additive model using the exponential loss function

More generally, basis function expansions take the form

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

where β_m 's are the expansion coefficients and $b(x;\gamma) \in \mathbb{R}$ are simple functions of the input x parameterized by γ.

Examples of additive models:

- In single-hidden-layer neural networks with continuous output, $b(x;\gamma) = \sigma\left(\gamma_0 + \ \gamma_1^T x\right)$, where $\sigma(t) = 1/\left(1 + e^{-t}\right)$ is the sigmoid function, and γ parameterizes a linear combination of the input variables.
- In signal processing, wavelets are a popular choice with γ parameterizing the location and scale shifts of a mother wavelet.
- 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.

The model $\hat{f}$ is obtained by minimizing a loss (say squared error or a likelihood-based loss) averaged over the training data

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

Typically the computational cost is very high. But it is feasible to rapidly solve the sub-problem of fitting just a single basis function.

Forward stagewise modeling

Forward stagewise modeling approximate the solution to the above optimization problem by sequentially adding new basis functions to the expansion without adjusting the parameters and coef. of those that have been added.

- At iteration m, one solves for the optimal basis function $b\left(x, \hat{\gamma}_m\right)$ and corresponding coefficient $\hat{\beta}_m$, which then is added to the current expansion $f_{m-1}(x)$. Previously added terms are *not* modified.
- This process is repeated.

Forward Stagewise Additive Modeling (Algorithm 10.2):

- Set $f_0(x) = 0$
- for $m = 1, \ldots, M$
  - Compute

$$\left(\hat{\beta}_m, \hat{\gamma}_m\right) = \underset{\beta_m, \gamma_m}{\arg\min} \sum_{i=1}^{n} L\left(y_i, \sum_{m=1}^{M} f_{m-1}\left(x_i\right) + \beta_m b\left(x_i; \gamma_m\right)\right)$$

  - Set

$$f_m(x) = f_{m-1}(x) + \hat{\beta}_m b\left(x; \hat{\gamma}_m\right)$$

Squared-error loss example

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

At the m th step, given the current fit $f_{m-1}(x)$, we solve

$$\begin{array}{ll} \min_{\beta,\gamma} & \sum_i L\left(y_i, f_{m-1}\left(x_i\right) + \beta b(x, \gamma)\right) \Longleftrightarrow \\ \min_{\beta,\gamma} & \sum_i \left[y_i - f_{m-1}\left(x_i\right) - \beta b\left(x_i; \gamma\right)\right]^2 = \sum_i \left[r_{im} - \beta b\left(x_i, \gamma\right)\right]^2 \end{array}$$

where r_{im} is the residual of he current model on the i-th observation. The term $\hat{\beta}_m b\left(x; \hat{\gamma}_m\right)$ is the best fit to the current residual.

The updated fit

$$f_m(x) = f_{m-1}(x) + \hat{\beta}_m b\left(x; \hat{\gamma}_m\right)$$

Exponential Loss for classification

Exponential Loss for classification

For classification $Y \in \{-1, 1\}$:

Boosting fits an additive expansion in a set of elementary basis functions:

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

where the basis functions are the weak classifiers $G_m(x) \in \{-1, 1\}$.

AdaBoost is equivalent to forward stagewise additive modeling using the exponential loss function

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

using individual classifiers $G_m(x) \in \{-1, 1\}$ as basis functions

One can view AdaBoost.M1 as a method that approximates minimizing

$$\arg \min_{\beta_1, G_1, \ldots \beta_M, G_M} \sum_{i=1}^{n} \exp \left(-y_i \sum_{m=1}^{M} \beta_m G_m \left(x_i \right) \right)$$

via a forward-stagewise additive modeling approach.

Why exponential loss?

Why exponential loss?

Let $Y \in \{-1, 1\}$. The (population) minimizer of the exponential loss function is

$$
\begin{aligned}
f^*(x) &= \arg\min_f E_{Y|x}\left[e^{-Yf(x)}\right] \\
&= \frac{1}{2}\log\frac{\Pr(Y=1 \mid x)}{\Pr(Y=-1 \mid x)}
\end{aligned}
$$

which is equal to one half of the log-odds.

binomial deviance

Let

$$p(x) = \Pr(Y = 1 \mid x) = \frac{e^{f(x)}}{e^{-f(x)} + e^{f(x)}} = \frac{1}{1 + e^{-2f(x)}}$$

and define $Y' = (Y+1)/2 \in \{0, 1\}$. Then the binomial log-likelihood loss function (for Y') is

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

or equivalently the deviance (-log likelihood) is

$$-l(Y, f(x)) = \log\left(1 + e^{-2Y f(x)}\right)$$

Note that

$$\min_f \mathrm{E}_{Y|x}[-l(Y, f(x))] = \min_f \mathrm{E}_{Y|x}\left[e^{-Y f(x)}\right]$$

K-Class classification with exponential loss

For a K -class classification problem, class label $Y \in \{1, \ldots, K\}, K \geq 3$.

Consider the coding $Y' = (Y'_1, \ldots, Y'_K)^T$ with

$$Y'_k = \begin{cases} 1, & \text{if } Y = k \\ -\frac{1}{K-1}, & \text{otherwise} \end{cases}$$

Let $f = (f_1, \ldots, f_K)^T$ with $\sum_{k=1}^{K} f_k = 0$, and define

$$L(Y, f) = \exp\left(-\frac{1}{K} Y'^T f\right)$$

Then the minimizers f^* satisfies

$$\Pr(Y = k \mid x) = \frac{e^{\frac{f_k^*(x)}{K-1}}}{\sum_{l=1}^{K} e^{\frac{f_l^*(x)}{K-1}}}$$

Robust loss functions for classification

Robust loss functions for classification

Suppose $Y \in \{-1, 1\}$. The Margin of $f(x)$ is defined as $yf(x)$. It plays a similar role to the residuals $y - f(x)$ in regression. Consider the classification rule is $G(x) = \text{sign}[f(x)]$. Then

- Observations with positive margin $y_i f(x_i) > 0$ are correctly classified
- Observations with negative margin $y_i f(x_i) < 0$ are incorrectly classified
- The decision boundary is $f(x) = 0$

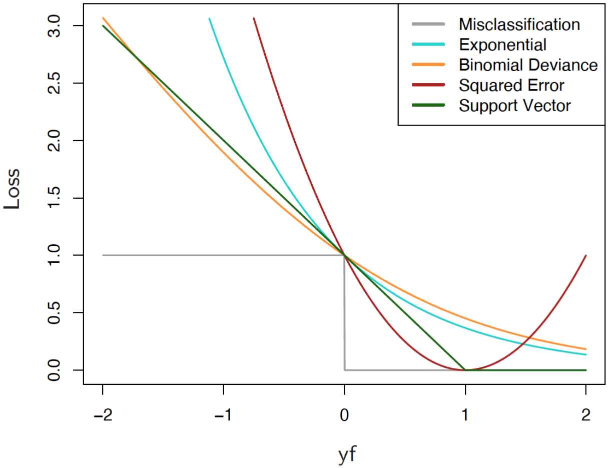
The goal of a classification algorithm is to produce positive margins as frequently as possible. Any loss function should penalize negative margins *more heavily* than positive margins.

various loss functions: Monotone decreasing loss functions of the margin yf:

- Misclassification loss: $I(\text{sign}(f(x)) \neq y)$
- Exponential loss: $\exp(-yf)$ (not robust against mislabeled samples)
- Binomial deviance: $\log\{1 + \exp(-2yf)\}$ (more robust against influential points)
- SVM loss: $[1 - yf]_+$

Other loss functions (not monotonic decreasing in yf):

- Squared error: $(y - f)^2 = (1 - yf)^2$ (penalize large positive margins heavily)



ESL Fig. 10.4. Loss functions for two-class classification. Each function has been scaled so that it passes through the point (0,1).

Loss Function	$L[y, f(x)]$	Minimizing Function
Binomial Deviance	$\log[1 + e^{-yf(x)}]$	$f(x) = \log \dfrac{\Pr(Y = +1\|x)}{\Pr(Y = -1\|x)}$
SVM Hinge Loss	$[1 - yf(x)]_+$	$f(x) = \text{sign}[\Pr(Y = +1\|x) - \frac{1}{2}]$
Squared Error	$[y - f(x)]^2 = [1 - yf(x)]^2$	$f(x) = 2\Pr(Y = +1\|x) - 1$
"Huberised" Square Hinge Loss	$-4yf(x)$, $yf(x) < -1$ $[1 - yf(x)]_+^2$ otherwise	$f(x) = 2\Pr(Y = +1\|x) - 1$

ESL Table 12.1. Loss functions for two-class classification.

K-Class classification

- Class label $Y \in \{1, \ldots, K\}, K \geq 3$.
- The classifier $h : R^d \longrightarrow \{1, \ldots, K\}$.

Using the 0-1 loss, the Bayes rule h^* is

$$h^*(x) = \arg \max_{k=1,\ldots,K} P(Y = k \mid x)$$

i.e., assign x to the most probable class using $P(Y \mid x)$.

One can model $p_k(x) := P(Y = k|x)$ by logistic model:

$$p_k(x) := P(Y = k|x) = \frac{e^{f_k(x)}}{\sum_{l=1}^K e^{f_l(z)}}$$

For identification, set $\sum_{k=1}^K f_k(x) = 0$, thus
$f_k(x) = \log p_k(x) - \frac{1}{K} \sum_{l=1}^K \log p_l(x)$.

The K-class multinomial deviance is $-\sum_{k=1}^{K} 1(y=k)\log p_k$ (i.e., -2 negative log-likelihood),

$$L(y,p(x)) = -\sum_{k=1}^{K} I\,(y=k)\log p_k(x)$$
$$= -\sum_{k=1}^{K} I\,(y=k)\,f_k(x) + \log\left(\sum_{\ell=1}^{K} e^{f_\ell(x)}\right)$$

Note: the multinomial deviance (negative loglikelihood) loss is used for gradient boosted trees.

Robust loss functions for regression

Robust loss functions for regression

- Squared error loss

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

  Population optimum for this loss function: $f(x) = E[Y \mid x]$
- Absolute loss

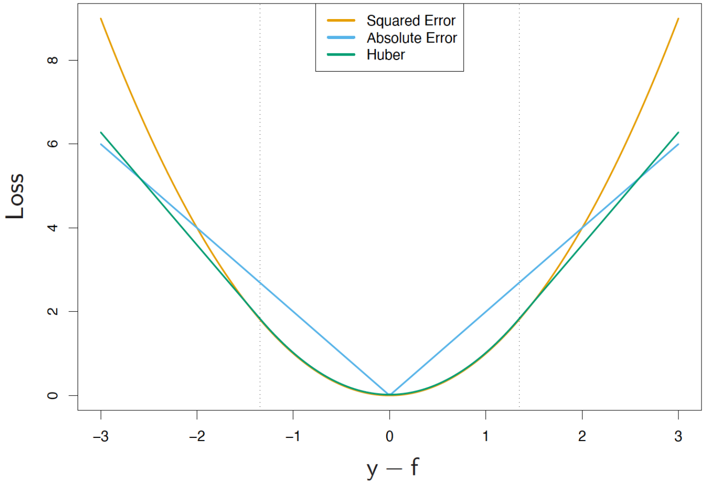
$$L(y, f(x)) = |y - f(x)|$$

  Population optimum for this loss function: $f(x) = \text{median}(Y \mid x)$

- Huber loss

$$L(y, f(x)) = \begin{cases} (y - f(x))^2 & \text{for } |y - f(x)| \le \delta \\ 2\delta|y - f(x)| - \delta^2 & \text{otherwise} \end{cases}$$

Such loss provides strong resistance to gross outliers while being nearly as efficient as least squares for Gaussian errors. It combines the good properties of squared-error loss near zero and absolute error loss when $|y - f|$ is large.



ESL Fig 10.5.