Boosting

Xiang Zhou

School of Data Science,
Department of Mathematics
City University of Hong Kong

2019

Boosting

- Iteratively learning weak classifiers
- Final result is the weighted sum of the results of each weak classifiers
- Many different boosting algorithms: adaboost (adaptive boosting) by Freund and Schapire (1996) is the first. [1]
- Examples of other boosting algorithms:
  - LogitBoost (Friedman, Hastie and Tibshirani, AOS, 2000)
  - L2Boost (Buhlmann and Yu, JASA, 2002)
  - **Gradient Boosting**(= Gradient Descent + Boosting) is an extension over boosting method.
- Focus on binary classification, and may be extended to multiclass case

[1] In 2004, Yoav Freund and Rob Schapire won the 2003 Godel prize in Theoretical Computer Science and the ACM's Paris Kanellakis Award for their AdaBoost in 2004, for theoretical accomplishments that have had a significant and demonstrable effect on the practice of computing.

Adaboost.M1 algorithm

Training data: (x_i, y_i); $i = 1, \dots, n$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$

1. Initialize $w_{1,i} = 1/n$; $i = 1, \dots, n$
2. For $m = 1$ to M:
   a. Fit a weak classifier[2] $h_m(x) : \mathbb{R}^p \to \{-1, 1\}$ to the training data with weights $w_{m,i}$ [3]
   b. Compute weighted misclassification error:

$$\epsilon_m = \sum_{i=1}^{n} w_{m,i} I(y_i \neq h_m(x_i)) \tag{1}$$

   c. Compute $\alpha_m = \frac{1}{2}\log((1-\epsilon_m)/\epsilon_m)$
   d. Update $w_{m+1,i} = w_{m,i}\exp(-y_i\alpha_m h_m(x_i))/Z_m$, where $Z_m = \sum_i w_{m,i}\exp(-y_i\alpha_m h_m(x_i))$ is for normalization.
3. Output $f(x) = \frac{\sum_m \alpha_m h_m(x)}{\sum_m \alpha_m}$ and $G(x) = \mathrm{sign}(f(x))$

[2] also called base learner. h_m can take real values ("confidence-rated prediction" by Shapire and Singer) and the weak learner is then $\mathrm{sign}(h_m)$;

[3] This means the loss function is weighted: $\min_{h_m} \sum_{i=1}^{n} w_{m,i}\ell(x_i, h_m(x_i))$

Weak classifier and weights for sample population

- The key parameters in Adaboost.M1 are the weights $\{w_i\}$ which are adaptively updated in each iteration:

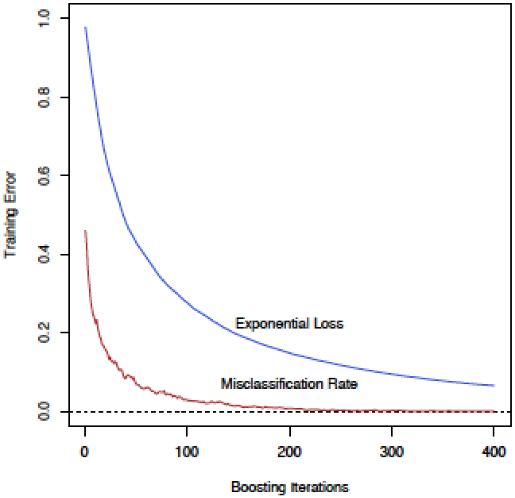
The weight w_i increases weights on misclassified points and decreases weights on correctly classified points.

and $\{\alpha_m\}$ summing up all the contributions from each h_m:

The weight α_m should be larger for the weak classifiers h_m with better performance (i.e. with the smaller ϵ_m).

- But there are still infinite number of choices of these two sets of weight parameters. Why Adaboost takes this form?
- Adaboost is almost assumption free except that $\alpha_m > 0$ or equivalently $\epsilon_m < 1/2$; i.e., the weak classifier needs to be better than "random guessing" for any weights $w_{m,i}$.
- The weak classifiers are often set as shallow decision tree.

A simulated example



The recursive idea in boosting

$$\min_f L(f) = \sum_{i=1}^{n} \ell(y_i, f(x_i))$$

The final classifier takes the additive form

$$f_M(x) = \sum_{m=1}^{M} \alpha_m h_m(x).$$

Consider the forward stagewise additive modeling [4]:

- Given $h_0(x), \ldots, h_{m-1}(x)$, how to find optimal $h_m(x)$ and α_m at stage m?

Then we can update f_m as $f_{m-1} + \alpha_m h_m(x)$

[4] instead of considering all terms h_m and α_m, $1 \le m \le M$, simultaneously.

- Compute the update as

$$\boxed{(\alpha_m, h_m) = \underset{\alpha \in \mathbb{R}^+, h \in \mathcal{H}}{\operatorname{argmin}} L(f_{m-1} + \alpha h)} \approx \sum_i \ell(y_i, f_m(x_i) + \alpha h(x_i))$$

(Regularization can be added for h by restricting $\mathcal{H}$)

- Update $f_m(x) = f_{m-1}(x) + \alpha_m h_m(x)$, and then iterate.

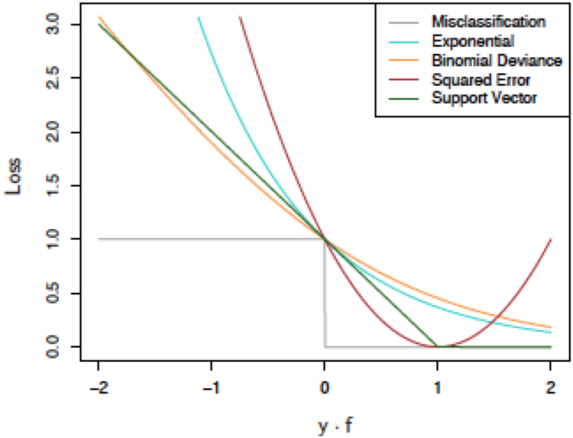
Loss functions

Recall Table 12.1 in [ESL]. The loss function $\ell(y, f)$ usually takes the product form: $\ell(y, f) = \ell(yf)$ for the binary classification.

- 0-1 loss: $\ell(y, f) = I(y \neq \mathrm{sign}(f)) = \text{heaviside}(-yf)$
    - Ideal but hard to work with (non-differentiable, discontinuous)
- Binomial deviance: $\ell(y, f) = \log(1 + \exp(-yf))$
    - Fisher consistent; logistic regression
- squared error: $(y - f)^2 = (1 - yf)^2$
    - Not desirable as it penalizes correct classification
- hinge loss: $(1 - yf)_+$
    - Fisher consistent; support vector machine
- exponential loss: $\exp(-yf)$
    - Fisher consistent; large margin separation; Adaboost

The exponential function is very useful in our stagewise addition setting since we are using the *additive* model now.

loss functions



Note: The 0-1 loss $\ell_{01}(y, f) = \text{heavisde}(-yf)$ is bounded from above by the other loss $\ell(y, f)$.

AdaBoost: the magic of exponential loss for additive model

1. The exponential loss function $\ell(y, f(x)) = \exp(-yf(x))$ is *Fisher consistent*:
$$f^* = \operatorname*{argmin}_f E(\exp(-Yf(X))) = \frac{1}{2}\log\frac{Pr(Y=1|X)}{1-Pr(Y=1|X)},$$
and $\mathrm{sign}(f^*(x)) = \mathrm{sign}(Pr(Y=1|X=x) - 1/2)$ is the Bayes rule.
2. The stagewise minimization for the additive model
$$\begin{aligned}\min_{\alpha,h}&\sum_{i=1}^{n}\exp(-y_i(f_{m-1}(x_i)+\alpha h(x_i)))\\ &=\min_{\alpha,h}\sum_{i=1}^{n}w_i^{(m)}\exp(-\alpha y_i h(x_i))\end{aligned} \tag{2}$$
where $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$ equals $w_i^{(m-1)}\exp(-y_i\alpha_{m-1}h_{m-1}(x_i))$, as in the Step 2.d of AdaBoost.

- The exponential loss function is Fisher consistence, so for any $\alpha > 0$, the weak learner h_m is consistent with the Bayes classifier: $h_m = \operatorname{argmin}_{h \in \mathcal{H}} \sum_i w_i^{(m)} I(y_i \neq h(x_i))$. [5]
- Then for given h_m, (2) becomes the minimization for α:

$$\min_\alpha \sum_{i=1}^n w_i^{(m)} \exp(-\alpha y_i h_m(x_i)) = \min_\alpha \left((1 - \epsilon_m) e^{-\alpha} + \epsilon_m e^{\alpha}\right)$$

  where the weighted misclassification rate ϵ_m is defined in (1). The minimizer is $\alpha_m = \frac{1}{2} \log((1 - \epsilon_m)/\epsilon_m)$, as in Step 2.c of AdaBoost.
- **The weak learner assumption**:

$$\alpha_m > 0 \iff \epsilon_m < 1/2$$

[5] In practice, h_m is not exactly this 0-1 loss minimizer but a minimizer of the empirical risk for some Fisher consistent loss function.

- LogitBoost: apply the same idea to the log-loss function used in logistic regression, but with more complicated and less elegant computations.

Boosting

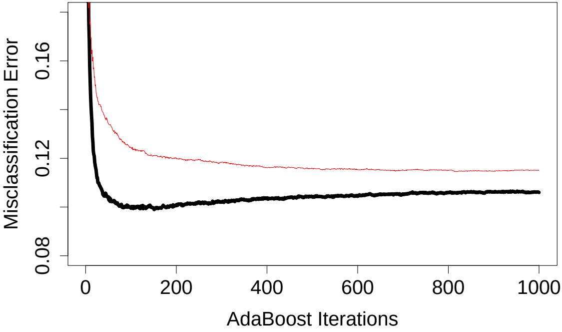
$$(\alpha_m, h_m) = \underset{\alpha \in \mathbb{R}^+, h \in \mathcal{H}}{\operatorname{argmin}} L(f_{m-1} + \alpha h)$$

1. This is similar to the idea of coordinate descent for $\min_{\{\alpha_m, h_m : m=1,2,\ldots\}} L(\sum_m \alpha_m f_m)$.
2. This stagewise addition has some implicit effect of regularization, since the search direction for f is restricted to one component each time.
3. Gradient boosting[6] : h is set as the gradient of L at f_{m-1}, or its projection in some restricted space $\mathcal{H}$.
4. h does not have to be gradient: it works as long as adding h_m can decrease the loss function further.

[6] Friedman, J. H. Greedy function approximation: a gradient boosting machine. Ann. Statist., 29(5):11891232, 2001.

Does boosting overfit?

In practice, the test error does not rise much as the number of iterations m increases. Boosting algorithms are relatively immune to overfitting but it overfit slightly after a long time of iterations.



Red line: Adaboost with 8-node trees; black line: Adaboost with stumps (one-node trees)

Regularization

- Early stopping (Zhang and Yu, 2005): stop the boosting algorithm at a medium number of iterations
- Shrinkage (Lugosi and Vayatis, 2003):

$$f_m(x) = f_{m-1}(x) + \nu \cdot \alpha_m h_m(x),$$

  where $0 < \nu < 1$ is a shrinkage factor
- These two approaches operate in a similar fashion by controlling $\sum_m \alpha_m$: small value of M and small value of ν result in small $\sum_m \alpha_m$ and large training error, and vice versa

Analyzing the training error

The most basic theoretical property of AdaBoost concerns its ability to reduce the training error.

1. The 0-1 loss is bounded by exponential loss: with $f = \sum_m \alpha_m h_m$

$$\frac{1}{n}\sum_{i=1}^{n} \ell_{01}(y_i, f(x_i)) \quad \leq \quad \frac{1}{n}\sum_{i=1}^{n} \exp\left(-y_i \sum_{m=1}^{M} \alpha_m h_m(x_i)\right)$$

2. By $w_{m+1,i} = w_{m,i} \exp(-y_i \alpha_m h_m(x_i))/Z_m$ and $w_{1,i} \equiv 1/n$,

$$\begin{aligned}
\tfrac{1}{n}\textstyle\sum_{i=1}^{n} \exp\left(-y_i \sum_{m=1}^{M} \alpha_m h_m(x_i)\right) &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} \prod_{m=1}^{M} \left(Z_m \frac{w_{m+1,i}}{w_{m,i}}\right) \\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} \left(\prod_{m=1}^{M} Z_m\right) \frac{w_{M+1,i}}{w_{1,i}} \\
&= \left(\textstyle\prod_{m=1}^{M} Z_m\right) \sum_{i=1}^{n} w_{M+1,i} = \prod_{m=1}^{M} Z_m
\end{aligned}$$

Theorem

The training error of the final classifier of the adaboost decays exponentially in the iteration number M:

$$\frac{1}{n}\sum_{i=1}^{n}\ell_{01}(y_i, f(x_i)) \leq \prod_{m=1}^{M} Z_m = \prod_{m=1}^{M}\sqrt{1-4\gamma_m^2}$$
$$\leq \exp(-2\sum_{m=1}^{M}\gamma_m^2) \leq \exp(-2\gamma^2 M)$$

where $\gamma_m = 0.5 - \epsilon_m$ *and* $\gamma = \min_m \gamma_m$.

The second line is a trivial consequence of the fact $\sqrt{1-t} \leq e^{-t/2}, t \in [0,1]$.

Proof

The last equality in the first line follows from

$$w_{m+1,i} = \frac{w_{m,i}\exp(-y_i\alpha_m h_m(x_i))}{Z_m}.$$

Note that $\sum_{i=1}^{n} w_{M+1,i} = 1$, $\alpha_m = \frac{1}{2}\log((1-\epsilon_m)/\epsilon_m)$, and

$$\begin{aligned} Z_m &= \sum_{i=1}^{n} w_{m,i}\exp(-y_i\alpha_m h_m(x_i)) \\ &= \sum_{\{i:y_i=h_m(x_i)\}} w_{m,i}e^{-\alpha_m} + \sum_{\{i:y_i\neq h_m(x_i)\}} w_{m,i}e^{\alpha_m} \\ &= (1-\epsilon_m)e^{-\alpha_m} + \epsilon_m e^{\alpha_m} = 2\sqrt{\epsilon_m(1-\epsilon_m)}. \end{aligned}$$

The desired upper bound is obtained after plugging in γ_m and Z_m.

The following exercise is a generalization of the above theorem.

Exercise

For any $\theta > 0$, show that

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i f(x_i) \leq \theta) \leq \left(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}}\right)^M,$$

where the expression inside parenthesis < 1 when $\theta < \gamma$.

Generalization Error

R. E. Schapire, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. Ann. Stat., 1998.

The generalization error is bounded by

$$\Pr_{\mathcal{T}}(Yf(X) \leq \theta) + O\left(\sqrt{\frac{d}{n\theta^2}}\right)$$

for any positive θ, where d is the VC dimension of the hypothesis space $\mathcal{H}$ and n is the number of training samples. $\Pr_{\mathcal{T}}$ is the prob. w.r.t. the training data.

1. The first term decays exponentially in M by the above exercise. The second term is independent of the iteration number M.
2. $yf(x) = y\sum_m \alpha_m h_m(x)$ [7] is called the **margin** of example (x, y): the value of the margin ($\in [-1, 1]$) can be interpreted as a measure of confidence in the prediction.

[7] where $\sum_m \alpha_m = 1$ is normalized and h_m is $+1/-1$ response

Example: L2Boost

$$L(f) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

- Initialize $f_0(x) = \bar{y}$
- Compute $g_m(x_i) = y_i - f_{m-1}(x_i) = r_i$
- Compute $(\alpha_m, h_m) = \operatorname*{argmin}_{\alpha, h \in \mathcal{H}} \sum_{i=1}^{n}(r_i - \alpha h(x_i))^2$.
    - One can interpret this as to find the closest point, $\alpha_m h_m(x)$, in $\mathcal{H}$ to the gradient r_i: the projection of the gradient onto the space $\mathcal{H}$.
    - or to interpret this as iteratively fitting the residuals $\{(x_i, r_i)\}$.
- Update $f_m(x) = f_{m-1}(x) + \alpha_m h_m(x)$, and iterate

Example: LogitBoost

Loss function of logistic regression with binary classification where $Y = \pm 1$ (softplus function):

$$L(f) = \sum_{i=1}^{n} \log(1 + \exp(-2y_i f(x_i)))$$

- Initialize $f_0(x) = \frac{1}{2} \log \frac{1+\bar{y}}{1-\bar{y}}$

Exercise

Assume $f(x) \equiv f_0$ be a constant the solve the problem $\min_{f_0} L(f_0) = \sum_{i=1}^{n} \log(1 + \exp(-2y_i f_0)$

- Compute the gradient w.r.t. $f(x_i)$ [8] at f_{m-1}: $g_m(x_i) = -2y_i/(1 + \exp(2y_i f_{m-1}(x_i))) =: -\tilde{y}_i$
- Compute $h_m = \mathrm{argmax}_{h \in \mathcal{H}} \ \sum_{i=1}^{n} \tilde{y}_i h(x_i)$
- Approximate α_m by a Newton-Raphson step
- Update $f_m(x) = f_{m-1}(x) + \alpha_m h_m(x)$, and iterate

[8] Exercise!

Multiclass boosting

Training sample $(x_i, y_i)_{i=1}^n$ with $x_i \in \mathcal{R}^p$ and $y_i \in \{1, \ldots, K\}$.
Denote $\mathbf{f} = (f_1, \ldots, f_K)^T$ and $G(x) = \mathrm{argmax}_k f_k(x)$.

- Bayes rule:
  $G^*(x) = \mathrm{argmax}_k Pr(Y = k|X = x) = \mathrm{argmax}_k f_k(x)$
- Generalization error: $GE(\mathbf{f}) = Pr(Y \neq G(X))$
- Sum-to-zero constraint: $\sum_k f_k(x) = 0$ for all x

Multiclass loss function

Various multiclass loss functions $L(y, \mathbf{f})$ are proposed:

- $\sum_{k \neq y} \exp(f_k(x) - f_y(x))$ (Schapire and Singer, 1999)
- $\sum_{k \neq y} f_k^q(x)$ (Lozano and Abe, 2008)
- $\exp(-\frac{f_y(x)}{K-1})$ (Zhu et al., 2009)
- $\sum_{k \neq y} \exp(f_k(x))$ (Wang, 2011)

Further Readings

Boosting is argued to be one of the most successful practical tools arising from theoretic machine learning. One successful applications include the human face recognition, the spam filters. There are a few great theories on adaboost we did not cover here.

Explaining AdaBoost by Robert E. Schapire

An overview of Boosting Approach

Book: Boosting: Foundations and Algorithms by by Robert E. Schapire and Yoav Freund (2012)