STAT 5630, Fall 2019

Boosting

Xiwei Tang, Ph.D. <xt4yj@virginia.edu>

University of Virginia
October 24, 2019

AdaBoost

AdaBoost

- Boosting produce a sequence of learners:

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

- How to train each $f_t(x)$? At the t-th iteration, given perviously estimated $f_1, \ldots, f_{t-1}$, we estimate a new function $h(x)$ to minimize the loss:

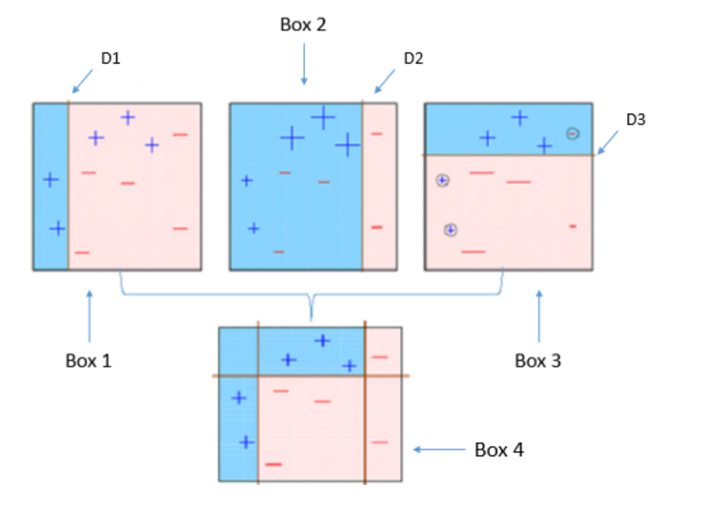
$$\min_h \sum_{i=1}^{n} L\Big(y_i, \sum_{k=1}^{t-1} f_k(x_i) + h(x_i)\Big)$$

- Instead of using the entire $h(x)$, we only use a small "fraction" of it, and add $\alpha_t h(x)$ to the current model. Then proceed to the next iteration.

AdaBoost

- Boosting is an additive model
- Boosting is also different from **random forests**, another additive model. In random forests, each tree is generated independently, so they can't borrow information from each other.
- AdaBoost is a special case of this framework with Exponential loss for classification, which is formulated by Yoav Freund and Robert Schapire (won the 2003 Godel Prize)
- For classification we use labels $y_i \in \{-1, 1\}$.

AdaBoost



$$D_4 = \text{sign}\big(0.42 \cdot D_1 + 0.65 \cdot D_2 + 0.92 \cdot D_3\big)$$

AdaBoost: intuition

- At the initial step, we treat all subject with equal weight
- Learn a classifier $f_t(x)$ and inspect which subjects got mis-classified.
- Put more weights on the mis-classified subjects
- Add $\alpha_t f_t(x)$ to the existing model and train the next iteration using the updated weights

AdaBoost: algorithm

1. Initiate weights $w_i^{(1)} = 1/n$, $i = 1, 2, \ldots, n$.
2. For $t = 1$ to T, repeat [a] - [d]
   (a) Fit a classifier $f_t(x) \in \{-1, 1\}$ to the weighted training data, with individual weights $w_i^{(t)}$.
   (b) Compute
   $$\epsilon_t = \sum_i w_i^{(t)} \mathbf{1}\{y_i \neq f_t(x_i)\}.$$
   (c) Compute $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$.
   (d) Update weights
   $$w_i^{(t+1)} = \frac{w_i^{(t)}}{Z_t} \exp[-\alpha_t y_i f_t(x_i)],$$
   where Z_t is a normalization factor to keep $w_i^{(t+1)}$ a distribution.
3. The final model: $F_T(x) = \sum_{t=1}^T \alpha_t f_t(x)$; Output the classification rule: sign$(F_T(x))$

AdaBoost

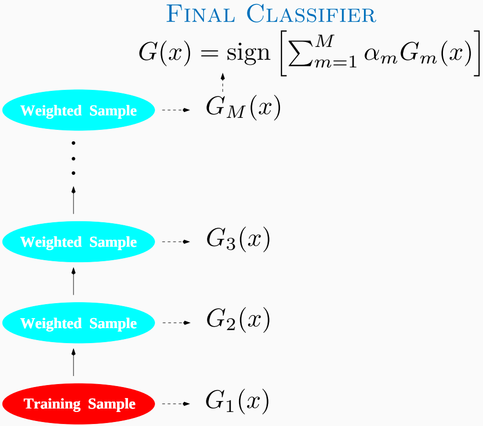


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Some facts

- The weights of the data points are multiplied by $\exp[-\alpha_t y_i f_t(x_i)]$

$$\exp[-\alpha_t y_i f_t(x_i)] = \begin{cases} \exp[-\alpha_t] < 1 & \text{if } y_i = f_t(x_i) \\ \exp[\alpha_t] > 1 & \text{if } y_i \neq f_t(x_i) \end{cases}$$

- The weights of correctly classified points are reduced, and the weights of incorrectly classified points are increased. Hence the incorrectly classified points receive more attention in the next iteration.

Some facts

- The weights α_t are always positive as long as the weak learner is better than random guessing

$$\epsilon_t < \frac{1}{2} \Longrightarrow \alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} > 0$$

Some facts

- The weights α_t are always positive as long as the weak learner is better than random guessing

$$\epsilon_t < \frac{1}{2} \Longrightarrow \alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} > 0$$

- Note: If the weak learner is worse than random guessing $\epsilon_t > \frac{1}{2}$, α_t will be negative, meaning we should revert the learner. Hence we can simply use $-f_t(x)$, and α_t is still positive.

Some facts

- The weights α_t are always positive as long as the weak learner is better than random guessing

$$\epsilon_t < \frac{1}{2} \Longrightarrow \alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t} > 0$$

- Note: If the weak learner is worse than random guessing $\epsilon_t > \frac{1}{2}$, α_t will be negative, meaning we should revert the learner. Hence we can simply use $-f_t(x)$, and α_t is still positive.
- The smaller the classification error ϵ_t, the larger the α_t is, meaning the weak learner has more impact on the final aggregated classifier.

Some facts

- The weights can be recursively computed:

$$
\begin{aligned}
w_i^{(t+1)} &= \frac{1}{Z_t} w_i^{(t)} \exp[-\alpha_t y_i f_t(x_i)] \\
&= \frac{1}{Z_1 \cdots Z_t} w_i^{(1)} \prod_{k=1}^{t} \exp[-\alpha_k y_i f_k(x_i)] \\
&= \frac{1}{Z_1 \cdots Z_t} \frac{1}{n} \prod_{k=1}^{t} \exp[-\alpha_k y_i f_k(x_i)] \\
&= \frac{1}{Z_1 \cdots Z_t} \frac{1}{n} \exp\Big[- y_i \sum_{k=1}^{t} \alpha_k f_k(x_i) \Big]
\end{aligned}
$$

- Note: $\sum_{k=1}^{t} \alpha_k f_k(x_i)$ is the just the grand model at the t-th iteration, we can rewrite it as $F_t(x_i)$.

Some facts

- Since $w_i^{(t+1)}$ always sums up to 1, we have

$$1 = \sum_i^n w_i^{(t+1)} = \frac{1}{Z_1 \cdots Z_t} \frac{1}{n} \sum_{i=1}^n \exp\left[- y_i F_t(x_i) \right]$$

$$\Longrightarrow \quad Z_1 \cdots Z_t = \frac{1}{n} \sum_{i=1}^n \exp[-y_i F_t(x_i)]$$

- Recall that the Z_i's are just the normalizing constants used in each iteration.
- The right hand side bounds above the training error
- Training error is: $\frac{1}{n} \sum_{i=1}^n \mathbf{1}\{y_i \neq \text{sign}(F_t(x_i))\}$

Convergence

- Claim: AdaBoost minimizes an upper bound on the classification error.
- To see this, we just need to analyze each Z_t separately.
- Recall that Z_t is used to normalize the weights, we have

$$Z_t = \sum_i^n w_i^{(t)} \exp[-\alpha_t y_i f_t(x_i)]$$

- Two different cases: $y_i f_t(x_i) = 1$ (the weak learner is correct); $y_i f_t(x_i) = -1$ (not correct). Hence, we have

$$Z_t = \sum_{y_i = f_t(x_i)} w_i^{(t)} \exp[-\alpha_t] + \sum_{y_i \neq f_t(x_i)} w_i^{(t)} \exp[\alpha_t]$$

- Since this term has nothing to do with other iterations, we just need to minimize Z_t such that the overall training loss can be reduced.

Convergence

- By our definition $\epsilon_t = \sum_i w_i^{(t)} \mathbf{1}\{y_i \neq f_t(x_i)\}$ is the proportion of weights for mis-classified samples
- And noticing that α_t is a constant for all subjects,

$$
\begin{aligned}
Z_t &= \sum_{y_i = f_t(x_i)} w_i^{(t)} \exp[-\alpha_t] + \sum_{y_i \neq f_t(x_i)} w_i^{(t)} \exp[\alpha_t] \\
&= (1 - \epsilon_t) \exp[-\alpha_t] + \epsilon_t \exp[\alpha_t]
\end{aligned}
$$

- Minimize this? take the derivative of α_t and set to 0.
- We have $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$ (best α_t for reducing the loss)
- Plug that back into Z_t, we have $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$

Convergence

- Change a variable $\gamma_t = \frac{1}{2} - \epsilon_t$, $\gamma_t \in (0, \frac{1}{2}]$.
- $\gamma_t > 0$ means that our weak learner is improving from random guessing.
- Then the minimum of Z_t becomes

$$
\begin{aligned}
Z_t &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \\
&= \sqrt{1-4\gamma_t^2} \\
&\leq \exp[-2\gamma_t^2]
\end{aligned}
$$

Convergence

- Let's go back to the $0/1$ training error for our final model with T weak learners:

$$
\begin{aligned}
\text{Err} &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i \neq \text{sign}(F_t(x_i))\} \\
&\leq \frac{1}{n}\sum_{i=1}^{n} \exp[-y_i F_t(x_i)] \\
&= Z_1 \cdots Z_t \\
&\leq \exp\Big[-2\sum_{t=1}^{T} \gamma_t^2\Big]
\end{aligned}
$$

- Hence the training error of AdaBoost decreases the upper bound exponentially
- A weak classifier with small error rate (large γ_t) will lead to faster descent

Remarks

- The Adaboost algorithm outputs a classifier $F_T(x)$ with small testing error? No. We need to tune T. Careful! — You can easily overfit.
- The training error of $F_T(x)$ decreases wrt T? No.
  - After each iteration, Adaboost decreases a particular upper-bound of the 0/1 training error. So in a long run, the training error is going to zero, but not necessarily monotonically.
- We can use a classifier that is worse than random guessing? Yes. The reverse of that classier will be used ($\alpha_t < 0$)
- In practice, a classification tree model is used as the weak learner.

Remarks

- If we consider just the exponential loss $E(e^{-yF(x)})$ (this is the upper bound of Adaboost error), then the optimal minimizer should be

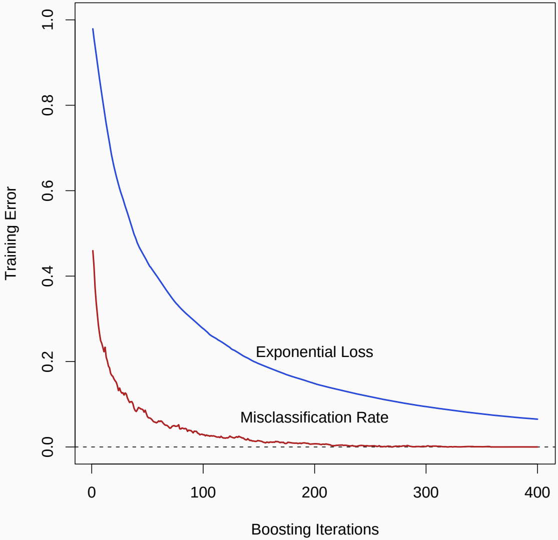
$$F(x) = \frac{1}{2} \log \frac{\mathsf{P}(y = 1|x)}{\mathsf{P}(y = -1|x)}$$

- This leads to the estimated probability:

$$\mathsf{P}(y = 1|x) = \frac{e^{2F(x)}}{1 + e^{2F(x)}}$$

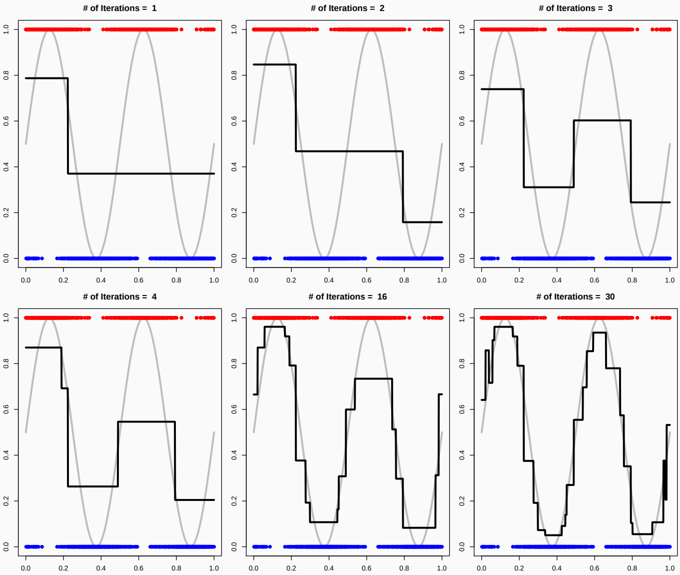
which is just the logistic model with a factor of 2.

AdaBoost



Training Error
0.0
0.2
0.4
0.6
0.8
1.0
Exponential Loss
Misclassification Rate
0
100
200
300
400
Boosting Iterations

A Simple Example in Regression



Implementation

- Use R package gbm: function gbm
- Tuning parameters:
  - Specify distribution = “adaboost”
  - n.trees controls the number of iterations T
  - shrinkage: further set a shrinkage factor on each $f_t(x)$. The default is 0.01. The original AdaBoost uses 1, however, can be less stable. A small value of this will require a large number of trees.
  - bag.fraction: each $f_t(x)$ uses a bootstrapped sample. If set to < 1, two different runs will produce slightly different models
  - cv.folds: number of cross validations
- Other parameters to consider: interaction.depth $= 1$ means stumps (additive model), > 1 allows interations
- Other resources: XGBoost (gradient boosting)