Boosting - AdaBoost

Zhe Yang

Contents

Boosting

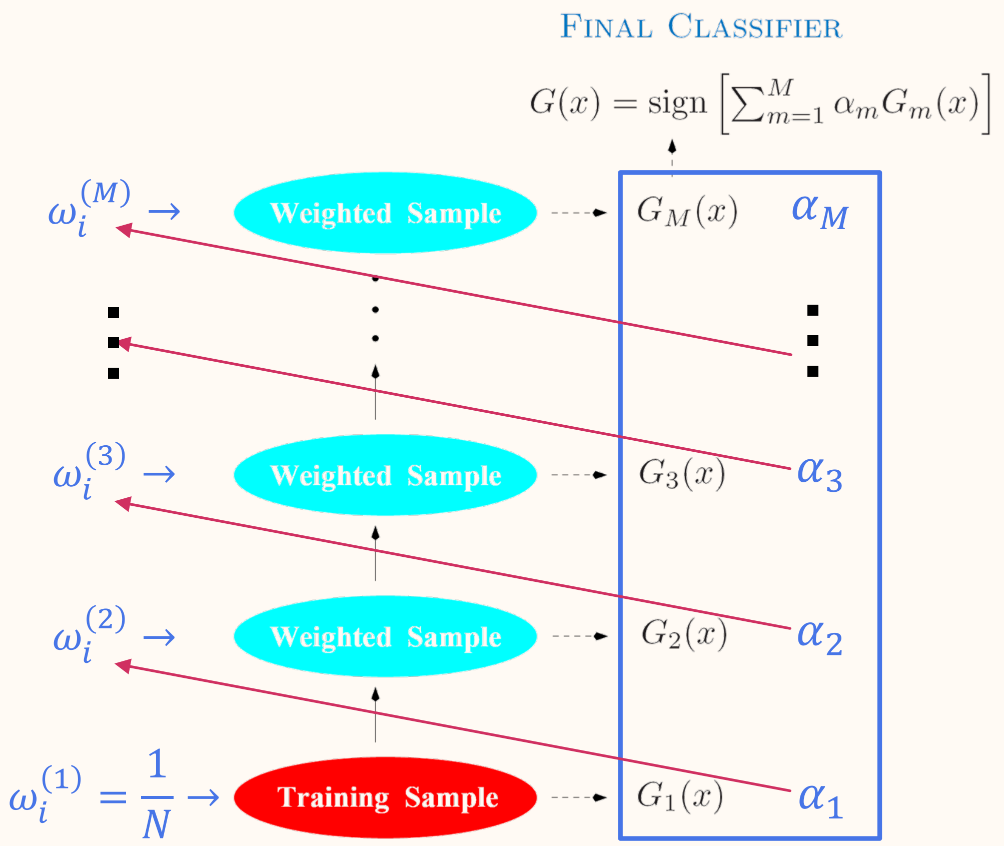
- Motivation: "Can a set of weak learners create a single strong learner?"
  - Kearns & Valiant (1988, 1989)
  - Weak learner: a ML algorithm with performance slightly better than random guessing

Yes !

- Brief history
  - 1990 – First simple boosting procedure (Schapire)
    - Showed that a weak learner could always improve its performance
    - Initial classifier + two classifiers trained on misidentified/disagreed data by previous classifier
  - 1993 – Idea of "Boost by majority" (Freund)
  - ---- Border between theoretical and practical ----
  - 1995 – AdaBoost (Freund & Schapire)
    - Significant improvement, combines many weighted weak learners, adjusted training weight

AdaBoost

Use binary classification as an example



Final Classifier
$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$
$\omega_i^{(M)} \rightarrow$
Weighted Sample
$G_M(x)$
α_M
$\omega_i^{(3)} \rightarrow$
Weighted Sample
$G_3(x)$
α_3
$\omega_i^{(2)} \rightarrow$
Weighted Sample
$G_2(x)$
α_2
$\omega_i^{(1)} = \frac{1}{N} \rightarrow$
Training Sample
$G_1(x)$
α_1

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

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
            $\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$.
    (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 $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

Points

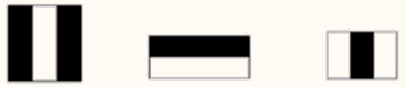
- It's a sequential process
  - Updated weights on training sample, focusing on mistakes made by previous classifiers
- It's a family of ensemble technique
  - Weighted combination of classifiers at different stage, good classifier's decision is more important

Details

- Base classifier G is not specified
  - Decision tree is an ideal choice

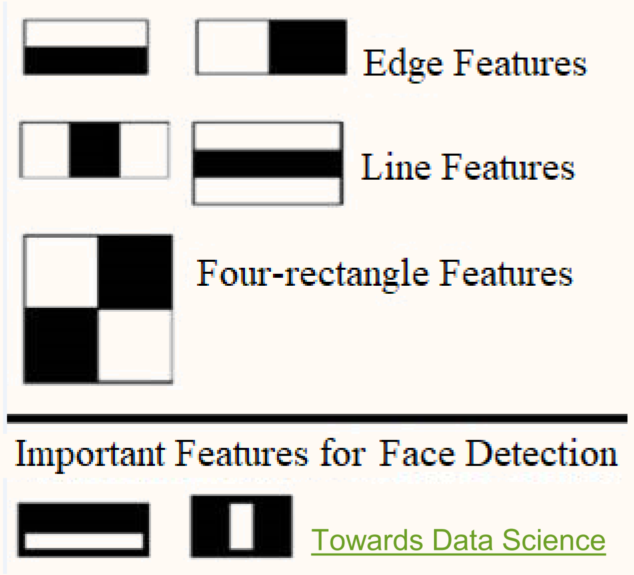
Why AdaBoost has big impact

- It's simple yet effective
  - The idea behind can be applied to different fields
    - Computer vision: Viola-Jones detector
      - Base classifiers are the haar-like feature with examples given on the right side
      - On average only 8 features need to be evaluated



$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

- Guaranteed by the theory to be valid
  - Train err decrease exponentially
  - "Never" overtrained in practice



AdaBoost: additive structure

- The structure of AdaBoost is indicated in the expression of final prediction

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

  - The basis functions are the individual classifiers $G_m(x) \in \{-1, 1\}$

- More generally, basis function expansions takes the form

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

  - $\beta_m, m = 1,2, \cdots, M$ are the expansion coefficients
  - $b(x; \gamma_m) \in \mathbb{R}$ are usually simple functions
    - Taking multivariate input x
    - Defined by parameter set γ

Use of additive expansions

- Additive expansions are at the heart of many of the learning techniques
  - Neural network with single hidden layer
    - $b(x;\gamma) = \sigma(\gamma^T x)$, σ is the activation function, γ are the linear combination weights for input x
  - Multivariate adaptive regression splines (MARS)
    - b is from truncated-power spline basis functions, γ is the choice of $x_{ij}, i = 1,2,\cdots,N, j = 1,2,\cdots,p$
  - Tree methods
    - b is simple tree, γ defines splits and prediction at leaf nodes

Forward stagewise additive modeling

- The additive models discussed in previous page are typically fit by minimizing a loss function averaged over the training data

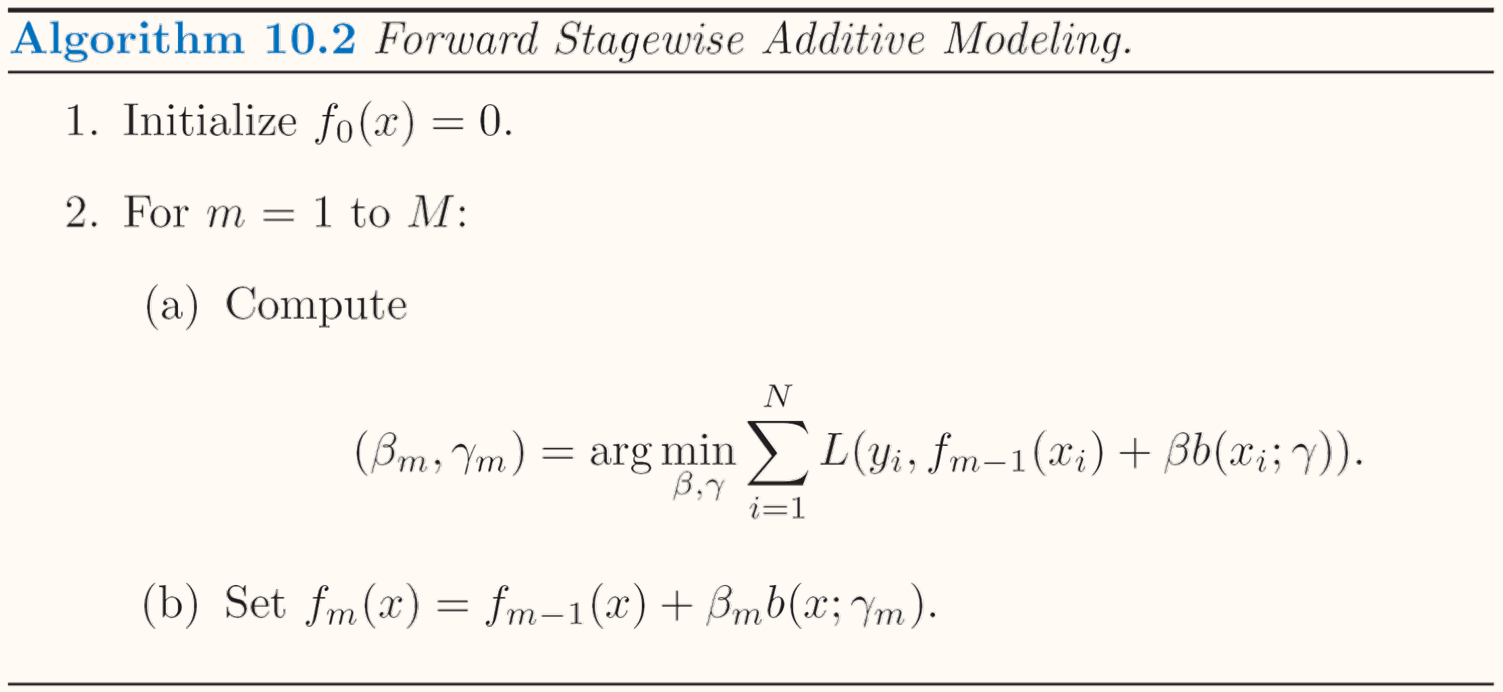
$$\min_{\{\beta_m,\gamma_m\}_i^M} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \beta_m b(x_i;\gamma_m))$$

  - The loss function L could be the squared-error or a likelihood-based loss function

- It requires computationally intensive numerical optimization techniques to optimize with all M basis functions considered simultaneously, a simple alternative often can be found when it is feasible to rapidly solve the subproblem of fitting just a single basis function

$$\min_{\{\beta,\gamma\}} \sum_{i=1}^{N} L(y_i, \beta b(x_i;\gamma))$$

Forward stagewise additive modeling (Algorithm)



```
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 $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.
```

At each stage m only update β_m and γ_m

- by fitting the partial residual $y - f_{m-1}(x)$
- without adjusting those have been already fixed in previous stage $1, 2, \cdots, m-1$
- we show next that AdaBoost is doing the same equivalently
    - even though we didn't use any loss function in the AdaBoost algorithm described earlier

AdaBoost as a FSAM

- AdaBoost is equivalent to forward stagewise additive modeling (FSAM) using the loss function

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

  - The reason to use exponential form discussed later
  - Note that we have $f(x) \in \{-1,1\} \to yf(x) = \begin{cases} 1, y = f(x) \\ -1, y \neq f(x) \end{cases}$

- At stage m, the optimization is done by solving

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

- Rewrite with $w_i^m = \exp(-y_i f_{m-1}(x_i))$

$$(\beta_m, G_m) = \arg\min_{\beta,G} \sum_{i=1}^{N} w_i^m \exp[-y_i \beta G(x_i)]$$

  - w_i^m is fixed in previous stages and can be regarded as a weight applied to observations
  - Note $G_m(x)$ also have $G_m(x) \in \{-1, 1\} \to yG_m(x) = \begin{cases} 1, y = G_m(x) \\ -1, y \neq G_m(x) \end{cases}$

Get (β_m, G_m)

- β_m is a positive constant even thought it's unknown yet, we consider it as fixed and solve the G_m

$$G_m = \arg\min_G \sum_{i=1}^{N} w_i^{(m)} \mathrm{I}(\mathrm{y_i} \neq G(x_i))$$

  - we only consider $y_i \neq G(x_i)$ situation here, because the $y_i = G(x_i)$ situation is the other side of the coin

$$I\big(y_i = G(x_i)\big) = 1 - I(y_i \neq G(x_i))$$

- With known G_m the loss function $L_m(\beta)$ will be

$$L_m(\beta) = \sum_{i=1}^{N} w_i^{(m)} \exp\big(-\beta y_i G_m(x_i)\big) = e^{-\beta} \sum_{y_i = G(x_i)} w_i^{(m)} + e^{\beta} \sum_{y_i \neq G(x_i)} w_i^{(m)}$$

- Get optimal β to minimize $L_m(\beta)$

$$\beta_m = \frac{1}{2} \ln\left(\frac{1 - err_m}{err_m}\right), err_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}$$

  - Note that β_m could be negative if $err_m < 0.5$, in which case it automatically reverse the polarity.
    - Move the "-1" in β_m to G_m: $\beta_m \rightarrow -\beta_m > 0$, $G_m(x) \rightarrow -G_m(x)$, $err_m \rightarrow 1 - err_m > 0.5$

Stage updates

- In each stage m, the prediction function is updated as

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x)$$

  - where β_m and G_m are estimated as illustrated in the previous page

- In next stage $m + 1$
  - the weight $w_i^{(m)}$ in loss function is updated to $w_i^{(m+1)}$ by

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{-\beta_m y_i G_m(x_i)}$$

  - then β_{m+1} and G_{m+1} are updated following the same procedures

AdaBoost as a Newton-like method

- Again, start from the exponential loss $L(y, f(x)) = \exp(-yf(x))$. Suppose we seek an improvement

$$f(x) + cg(x)$$

  - here c is a constant to be determined and $g(x) \in \{-1,1\}$ giving a binary prediction

- We can expand L at $g(x) = 0$

$$L(y, f(x) + cg(x)) = \exp(-yf(x)) \exp(-ycg(x))$$
$$\approx \exp(-yf(x)) \left[1 - cyg(x) + \frac{c^2}{2}\right] \overset{\text{def}}{=} L'$$

  - notice the derivation of $\frac{c^2}{2}$ use the fact that $y^2 = 1$ and $g(x)^2 = 1$
  - $\exp(-yf(x))$ is determined by previous process and can be view as weight w

Get $g(x)$

- Minimizing L' with c fixed is equivalent to maximizing $yg(x)$, the solution is

$$g(x) = \begin{cases} 1, & p(y=1) > p(y=-1) \\ -1, & otherwise \end{cases}$$

- Again, use the fact that $y^2 = 1$ and $g(x)^2 = 1$

$$yg(x) = -\frac{[y - g(x)]^2}{2} + 1$$

- Then minimizing L' is equivalent to minimizing the least-square loss

$$\frac{[y - g(x)]^2}{2} - 1$$

Get c

- With solved $g(x) \in \{-1,1\}$, we can get c again by optimizing L. The essential component is

$$\exp\big(-ycg(x)\big) = \sum_{y_i = g(x_i)} \mathrm{e}^{-c} + \sum_{y_i \neq g(x_i)} \mathrm{e}^{c} = n_T e^{-c} + n_F e^{c} \overset{\text{def}}{=} l(c)$$

  - where n_T is the number of correct classification and n_F is the number of incorrect classification

- Minimizing the equation above

$$\frac{\partial l(c)}{\partial c} = -n_T e^{-c} + n_F e^{c} = 0 \rightarrow c = \frac{1}{2}\ln\frac{n_T}{n_F} = \frac{1}{2}\ln\frac{1 - err}{err}$$

  - where $err = \frac{n_T}{n_T + n_F} = \frac{n_T}{n}$

Stage updates

- We get the exactly same form of coefficient and update function as what we got when we consider AdaBoost as FSAM
- And we can now construct a Newton-like method's update in each stage

$$f(x) \to f(x) + \frac{1}{2}\ln\frac{1-err}{err}g(x)$$

- In next round, the weight w is updated by

$$w \to w \cdot e^{-cyg(x)} = w \cdot e^{-c(1-2I(y \neq g(x)))}$$

What does exponential loss estimate?

- Investigating the population minimizer of $\exp(-yf(x))$

$$f^*(x) = \arg\min_{f(x)} E_{y|x}(e^{-yf(x)})$$

  - Get derivative $E_{y|x}(e^{-yf(x)})$ w.r.t $f(x)$ and set to zero

$$\frac{\partial E_{y|x}\left(e^{-yf(x)}\right)}{\partial f(x)} = E_{y|x}\left(-ye^{-yf(x)}\right) = E_{y=1|x}\left(-ye^{-yf(x)}\right) + E_{y=-1|x}\left(-ye^{-yf(x)}\right)$$

$$= -\Pr(y=1|x)\, e^{-f(x)} + \Pr(y=-1|x)\, e^{f(x)} = 0$$

  - Solve the equation

$$f^*(x) = \frac{1}{2}\ln\left(\frac{\Pr(y=1|x)}{\Pr(y=-1|x)}\right) \quad or \quad \Pr(y=1|x) = \frac{1}{1+e^{-2f^*(x)}}$$

- It is estimating the log-odds of $\Pr(y=1|x)$
  - This provides an inspect to justify that we use the sign of the model prediction as classification $\{-1,1\}$, where we have $G(x) = sign(\sum_{m=1}^{M} \alpha_m G_m(x))$.
    - if we have higher probability to predict $y=1$, $f^*(x)$ will be “+”
    - if we have higher probability to predict $y=-1$, $f^*(x)$ will be “-”,

Binomial deviance could be another candidate

- We now show that binomial deviance (cross-entry) leads to same solution at the population level. Using $y' = \frac{y+1}{2}$ to map $\{-1,1\}$ to $\{0,1\}$, we have the binomial deviance loss

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

- Get derivative $E_{y|x}(l\big(y, p(x)\big))$ w.r.t $p(x)$ and set to zero

$$\frac{\partial E_{y|x} l\big(y, p(x)\big)}{\partial p(x)} = E_{y=1|x}\left(\frac{1}{p(x)}\right) + E_{y=-1|x}\left(-\frac{1}{1-p(x)}\right)$$
$$= \frac{\Pr(y=1|x)}{p(x)} - \frac{\Pr(y=-1|x)}{1-p(x)} = 0$$

- Solve the equation we get

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

- Once $p(x)$ is determined, $f(x)$ is also determined via a logit relation

Continue

- Use $f(x)$ to replace $p(x)$

$$l(y, p(x)) = y' \log p(x) + (1 - y') \log(1 - p(x)) = \log \frac{e^{(y-1)f(x)}}{1 + e^{-2f(x)}}$$

  - Notice that $y \in \{-1,1\}$

$$l(y, p(x)) = \begin{cases} \log \frac{1}{1 + e^{-2f(x)}}, y = 1 \\ \log \frac{e^{-2f(x)}}{1 + e^{-2f(x)}}, y = -1 \end{cases} = -\log\left(1 + e^{-2yf(x)}\right)$$

  - Expand $e^{-yf(x)}$ and $-l(y, p(x))$ to Taylor series, we have

$$-l(y, p(x)) = \log(2) - yf(x) + \frac{y^2}{2} f^2(x) + \cdots$$

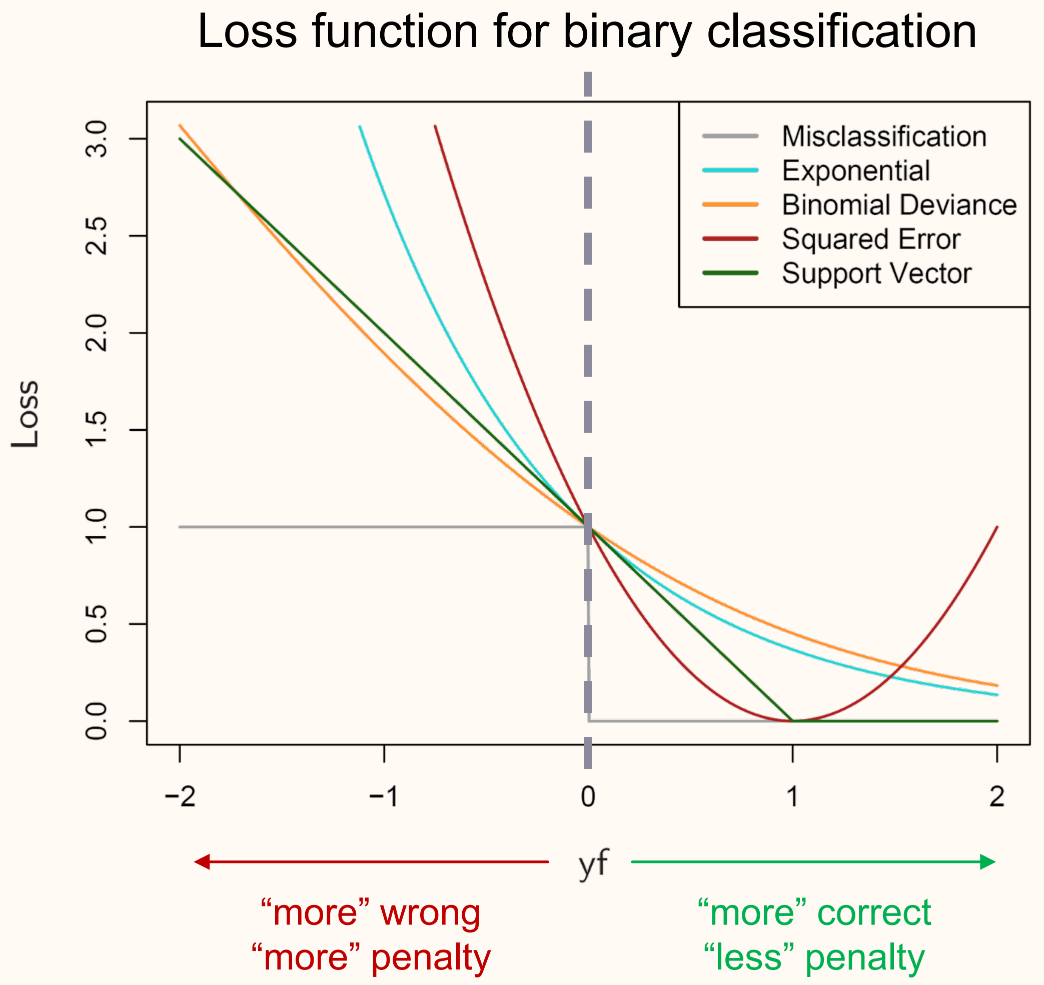
$$e^{-yf(x)} = 1 - yf(x) + \frac{y^2}{2} f^2(x) + \cdots$$

- Binomial deviance loss and exponential loss are equivalent to second order

$$-l(y, p(x)) \approx e^{-yf(x)} + \log(2) - 1$$

Compare different loss for classification

- In the right plot, "yf" serves as a margin which plays a role analogous to the residuals $y - f(x)$
- Investigate positive yf
  - Squared Error will penalize correct predictions
  - All other loss decrease monotonically
- Investigate negative yf
  - All the loss penalize incorrect predictions but with different degree
  - Exponential loss will penalize much more on incorrect predictions so it's less robust
    - If inputs have mislabeled entries, the performance of AdaBoost will degrade dramatically (empirical observation)



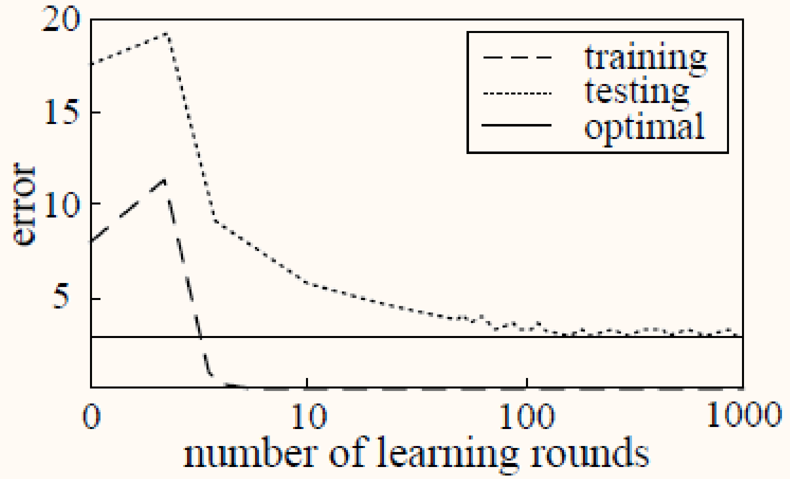
AdaBoost doesn’t overfit

- AdaBoost often doesn’t overfit in practice
- Freund & Schapire[JCSS97] proved that the generalization error of AdaBoostis bounded by:

$$\epsilon_{\mathfrak{D}} \leq \epsilon_D + \tilde{O}\left(\sqrt{\frac{dM}{n}}\right)$$

  - with probability at least $1-\delta$, where d represents the complexity of base learners,n is the number of training instances, M is the the number of learning rounds and $\tilde{O}$ hides the logarithmic and constant terms
  - It implies that AdaBoost will overfit if M is large
- Is there tighter bound to explain the observation in practice?

A typical performance plot of AdaBoost on real data

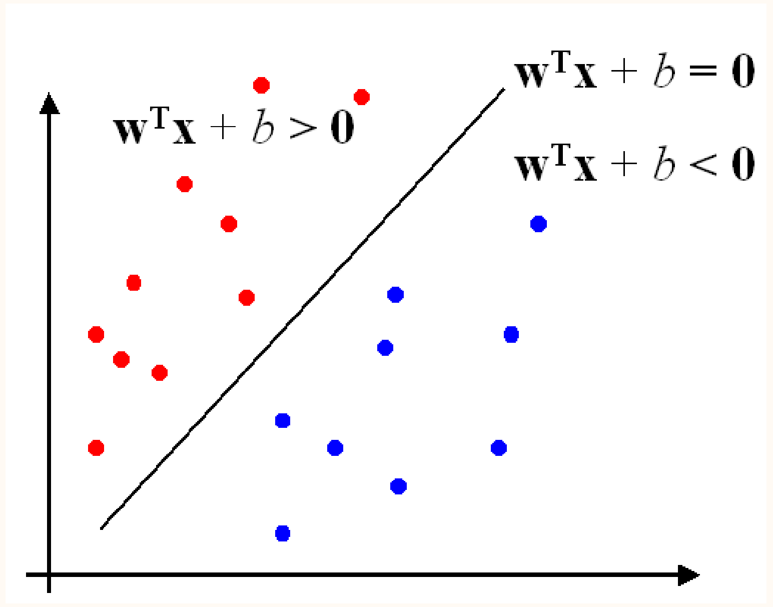


Seems contradict with the Occam’s Razor

Margin

- We have discussed the statistical view above. However, it did not explain why AdaBoost is resistant to overfitting according to observation in practice.
  - This can be explained by using margin
- Example
  - Binary classification can be viewed as the task of separating classes in a feature space
    - $f(x) = sign(w^T x + b)$
  - Margin of a single classifier f: $yf(x)$
  - Margin of the additive model F

$$yF(x) = \sum_{m=1}^{M} \alpha_m y f_m(x)$$


$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b > 0$
$\mathbf{w}^T\mathbf{x} + b < 0$

The bigger the margin,
the higher the predictive confidence

Margin explanation of AdaBoost

- Based on the concept of margin, Schapire et al. [1998] proved that, given any threshold $\theta > 0$ of margin over the training data D, with probability at least $1 - \delta$, the test error of the ensemble $\epsilon_{\mathrm{D}} = P_{x\sim\mathrm{D}}\big(y \neq F(x)\big)$ is bounded by

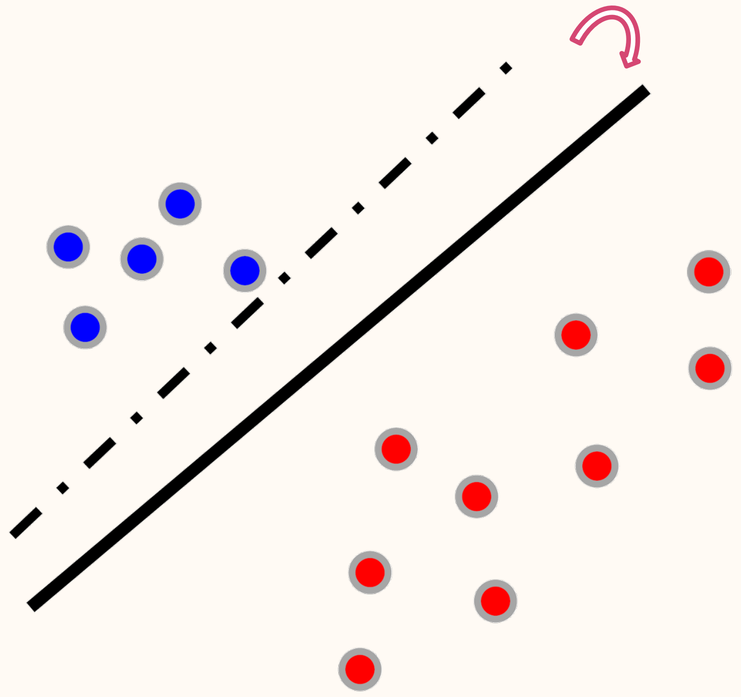
$$\epsilon_{\mathrm{D}} \leq P_{x\sim D}(yF(x) \leq \theta) + \tilde{O}\left(\sqrt{\frac{d}{n\theta^2} + \ln\frac{1}{\delta}}\right)$$

$$\leq 2^M \prod_{m=1}^{M} \sqrt{\epsilon_m^{1-\theta}(1-\epsilon_m)^{1+\theta}} + \tilde{O}\left(\sqrt{\frac{d}{n\theta^2} + \ln\frac{1}{\delta}}\right)$$

- This bound implies that, when other variables are fixed, the larger the margin over the training data, the smaller the test error

Margin explanation of AdaBoost (continued)

- Why AdaBoost tends to be resistant to overfitting?
- Margin theory: Because it is able to increase the ensemble margin even after the training error reaches zero



- ➢ This explanation is quite intuitive
- ➢ It receives good support in empirical study

The minimal margin bound

- Schapire et al.'s bound depends heavily on the smallest margin
  - If the smallest margin is large, $P_{x\sim D}(yF(x) \leq \theta)$ will be small
- Thus, by considering the minimum margin:

$$\varrho = \min_{x \in D} yF(x)$$

- Breiman [Neural Comp. 1999] proved a test error bound ($O\left(\frac{\log n}{n}\right)$), which is tighter than Schapire et al.'s bound

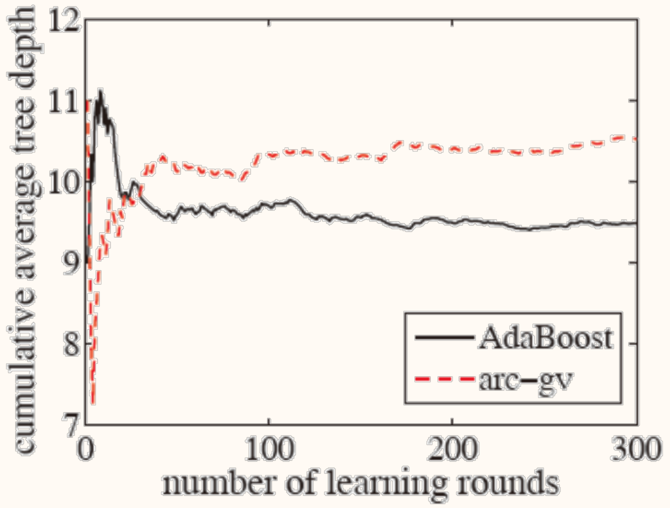
$$\epsilon_{\mathrm{D}} \leq R\left(\ln(2n) + \ln\left(\frac{1}{R}\right) + 1\right) + \frac{1}{n}\ln\frac{|\mathrm{H}|}{\delta}$$

The doubt about margin theory

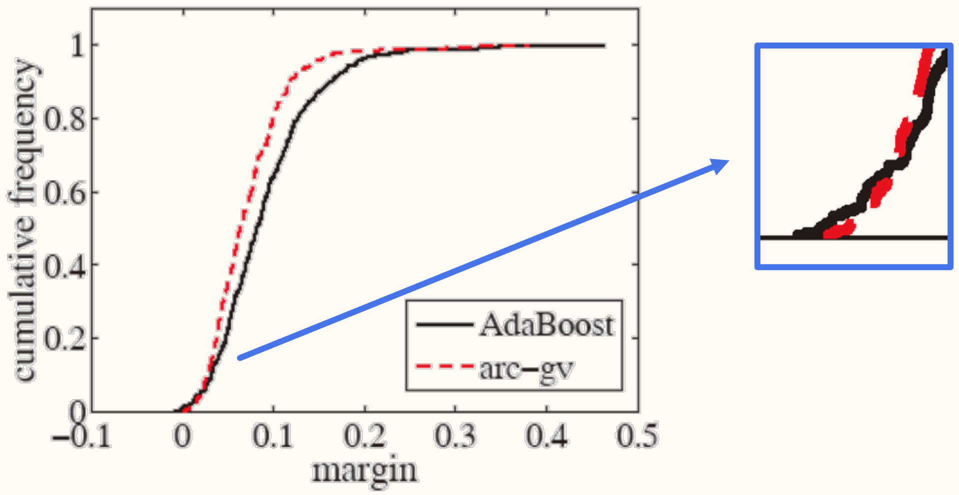
- Breiman [Neural Comp. 1999] designed a variant of AdaBoost, the arc-gv algorithm, which directly maximizes the minimum margin
  - the margin theory would appear to predict that arc-gv should perform better than AdaBoost
- However, experiments show that, comparing with AdaBoost:
  - arc-gv does produce uniformly larger minimum margin
  - the test error increases drastically in almost every case
- Thus, Breiman convincingly concluded that **the margin theory was in serious doubt**. This almost sentenced the margin theory to death

7 years later

- Reyzin& Schapire [ICML'06 best paper] found that, amazingly, Breiman had not controlled model complexity well in exps
  - Breiman controlled the model complexity by using decision trees with a fixed number of leaves
  - Reyzin & Schapire found that, the trees of arc-gv are generally "deeper" than the trees of AdaBoost
- Reyzin& Schapire repeated Brieman's exps using decision stumps with two leaves: arc-gv is with larger minimum margin, but worse margin distribution
  - R&S claimed that the minimum margin is not crucial, and the average or median margin is crucial



(a)



(b)

Supporting theory

- Equilibrium margin (Emargin) bound
  - Considered factors different from Schapire et al. and Breiman’s bounds
  - No intuition to optimize
- The kth margin bound
  - The minimum margin bound and Emargin bound are special cases of the kth margin bound, both are single-margin bound (not margin distribution bound)
- Finally
  - Random choice of sample S with size $n \geq 5$, every voting classifier $f \in C(H)$ satisfies

$$\epsilon_{\mathrm{D}} \leq \frac{2}{n} + \inf_{\theta \in (0,1]} \left[\Pr_S[yf(x) < \theta] + \underbrace{\frac{7\mu + 3\sqrt{3\mu}}{3\mu} + \sqrt{\frac{3\mu}{n} \Pr_S[yf(x) < \theta]}}_{O(\log n / n)} \right]$$

  - Where

$$\mu = \frac{8}{\theta^2} \ln n \ln(2|H|) + \ln \frac{2|H|}{\delta}.$$

    - Uniformly tighter than Breiman’s as well as Schapire et al.’ bounds
    - Considers the same factors as Schapire et al. and Breiman
  - thus, defends the margin theory against Breiman’sdoubt

Variance of margin also matters

Related to average margin

- We should pay attention to both average margin and margin variance.

$$\epsilon_{\mathcal{D}} \le \frac{1}{n^{50}} + \inf_{\theta\in(0,1]}\Big[\Pr_S[yf(x)<0] + \boxed{\underbrace{n^{-\frac{2}{1-\boxed{E_S^2[yf(x)]}+\frac{\theta}{9}}}}_{O(\log n/n)} + \frac{3\sqrt{\mu}}{n^{\frac{3}{2}}} + \frac{7\mu}{3n} + \sqrt{\frac{3\mu}{n}\boxed{\widehat{\mathfrak{T}}(\theta)}}}\Big]$$

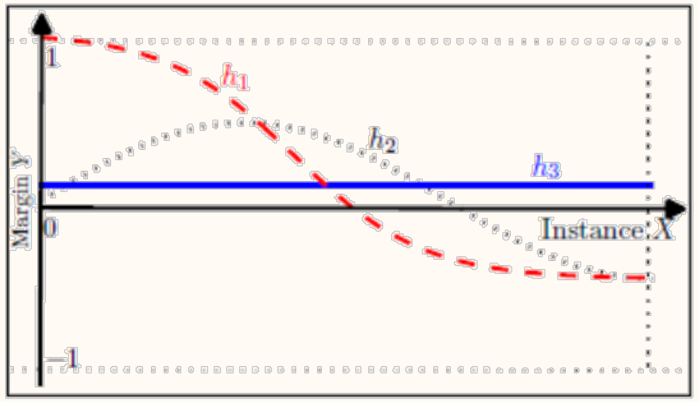
  - Where

$$\mu = 144\ln m\ln(2|H|)/\theta^2 + \ln(2|H|/\delta)$$

$$\widehat{\mathfrak{T}}(\theta) = \Pr_S[yf(x)<\theta]\Pr_S[yf(x)\ge\frac{2\theta}{3}]$$

Related to margin variance

Experimental results



- Margin variance really important

Figure from [Gao & Zhou, AIJ 2013]

Figure 1: Each curve represents a voting classifier. The X-axis and Y-axis denote example and margin, respectively, and uniform distribution is assumed on the example space. The voting classifiers h_1, h_2 and h_3 have the same average margin but with different generalization error rates: 1/2, 1/3 and 0.

[Shivaswamy& Jebara, NIPS 2011] tried to design new boosting algorithms by maximizing average margin and minimizing margin variance simultaneously, and the results are encouraging

Summary

- AdaBoost is an effective ensemble strategy to combines multiple weak models into a strong one
  - Additive structure + focusing on mistakes
  - It’s usually easy to find a bunch of weak models but hard to find one single strong model
  - The idea behind the algorithm is more important than the algorithm itself

- We can view the AdaBoost from different perspectives
  - It use a set up of forward stagewise additive modeling
  - It can be considered as a Newton method
  - The concept of margin explains why AdaBoost seems never overtrained

Homework

For a K-class classification problem, we can recode the class label c with a K-dimensional vector $\mathbf{y}$ with all entries equal to $-\frac{1}{K-1}$ except a 1 in position k if $c = k$, i.e.,

$$y_k = \begin{cases} 1, & \text{if } c = k, \\ -\frac{1}{K-1}, & \text{if } c \neq k. \end{cases}$$

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

$$L(\mathbf{y}, \mathbf{f}(\mathbf{x})) = \exp(-\frac{1}{K}\mathbf{y}^T \mathbf{f}(\mathbf{x})).$$

(a) Using Lagrange multipliers, derive the population minimizer $\mathbf{f}^*$ of $\mathbb{E}_{\mathbf{y}|\mathbf{x}}[L(\mathbf{y}, \mathbf{f}(\mathbf{x}))]$, subject to $\sum_{k=1}^K f_k(\mathbf{x}) = 0$, and relate these to the class probabilities.

(b) Derive a multiclass boosting algorithm using this loss function and verify that it covers the Adaboost algorithm as a special case ($K = 2$).

(c) Implement your derived algorithm, where you are allowed to call package of trees. Compare your implementation with the existing standard gradient boosting package on a multiclass classification problem. Make some discussion about what you observed. (Hint: there are two key differences: First, unlike adaboost which uses classification trees, gradient boosting uses regression trees. Second, gradient boosting has a shrinkage parameter (λ), a smaller learning rate ($\lambda = 0.01$) often dramatically improves the performance compared with $\lambda = 1$).

Backups

Reference

- ISL, ESL
- Boosting 25年 （2014 周志华） - Bilibili
- The Annals of Statistics 2000, Vol. 28, No. 2, 337-407