Boosting

Weikai Xu

School of Information Science and Technology
ShanghaiTech University

Mar 15th, 2020

Outline

Outline

A Formal Description of Boosting

1. given training set $(x_1, y_1), \cdots, (x_m, y_m)$
2. $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
3. for $t = 1, \cdots, T$:
    1. construct distribution D_t on $\{1, \cdots, m\}$
    2. find week classifier

    $$h_t : X \to \{-1, +1\}$$

    with small error ϵ_t on D_t:

    $$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

4. output final classifier H_{final}

AdaBoost

1. constructing D_t
   1. $D_1(i) = 1/m$
   2. given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$
$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(x_i))$$

Where Z_t = normalization constant, $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t}) > 0$

2. Final Classifier:

$$H_{final}(x) = sign(\sum_t \alpha_t h_t(x))$$

Analyzing the training error

1. Theorem
   1. write ϵ_t as $1/2 - \gamma_t$
   2. then

$$\begin{aligned}\text{training error}(H_{\text{final}}) &\leq \prod_t [2\sqrt{\epsilon_t(1-\epsilon_t)}] \\ &= \prod_t \sqrt{1-4\gamma_t^2} \\ &\leq \exp(-2\sum_t \gamma_t^2)\end{aligned}$$

2. so: if $\forall t : \gamma_t \geq \gamma > 0$, then training error$(H_{\text{final}}) \leq e^{-2\gamma^2 T}$
3. AdaBoost is adaptive:
   1. does not need to know γ or T a prior
   2. can exploit $\gamma \gg \gamma$

Proof

1. let $f(x) = \sum_t \alpha_t h_t(x) \Rightarrow H_{final}(x) = sign(f(x))$.
2. Step 1: unwrapping recurrence:

$$
\begin{aligned}
D_{final}(i) &= \frac{1}{m}\frac{\exp(-y_i \sum_t \alpha_t h_t(x_i))}{\prod_t Z_t} \\
&= \frac{1}{m}\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}
\end{aligned}
$$

Proof

1. Step 2: training error$(H_{final}) \leq \prod_t Z_t$
2. Proof:

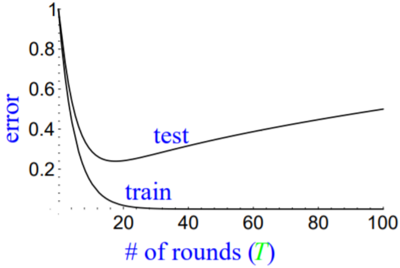
$$
\begin{aligned}
\text{training error}(H_{final}) &= \frac{1}{m}\sum_i \begin{cases} 1 & \text{if } y_i \neq H_{final}(x_i) \\ 0 & \text{else} \end{cases} \\
&= \frac{1}{m}\sum_i \begin{cases} 1 & \text{if } y_i f(x_i) \leq 0 \\ 0 & \text{else} \end{cases} \\
&\leq \frac{1}{m}\sum_i \exp(-y_i f(x_i)) \\
&= \sum_i D_{final}(i) \prod_t Z_t \\
&= \prod_t Z_t
\end{aligned}
$$

Proof

1. Step 3: $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$
2. Proof:

$$\begin{aligned} Z_t &= \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \\ &= \sum_{i: y_i \neq h_t(x_t)} D_t(i) e^{\alpha_t} + \sum_{i: y_i = h_t(x_t)} D_t(i) e^{-\alpha_t} \\ &= \epsilon_t e^{\alpha_t} + (1-\epsilon_t) e^{-\alpha_t} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \end{aligned}$$

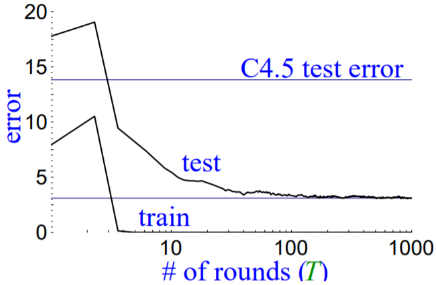
How Will Test Error Behave?(Guess)



expect:

1. training error to continue to drop (or reach zero)
2. test error to increase when Hfinal becomes "too complex"

Actual Typical Run



1. test error does not increase, even after 1000 rounds
2. test error continues to drop even after training error is zero!

A Better Story: The Margins Explanation

1. key idea:
   1. training error only measures whether classifications are right or wrong.
   2. should also consider confidence of classifications
2. recall: H_{final} is weighted majority vote of weak classifiers
3. measure confidence by margin = strength of the vote
   = (fraction voting correctly) - (fraction voting incorrectly)

Theoretical Evidence: Analyzing Boosting Using Margins

1. Theorem: large margins $\Rightarrow$ better bound on generalization error (independent of number of rounds)
   1. proof idea: if all margins are large, then can approximate final classifier by a much smaller classifier
2. Theorem: boosting tends to increase margins of training examples (given weak learning assumption)
   1. proof idea: similar to training error proof
3. so:
   although final classifier is getting larger,
   margins are likely to be increasing,
   so final classifier actually getting close to a simpler classifier,
   driving down the test error

Outline

Game Theory

1. game defined by matrix $M = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$
2. row player chooses row i
3. column player chooses column j (simultaneously)
4. row player's goal: minimize loss $M(i, j)$
5. usually allow randomized play: players choose distributions P and Q over rows and columns
6. learner's (expected) loss

$$= \sum_{i,j} P(i)M(i,j)Q(j)$$

$$= P^T MQ = M(P, Q)$$

The Minmax Theorem

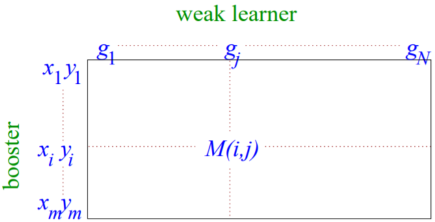
1. von Neumann's minmax theorem:

$$\begin{aligned}\min_P \max_Q M(P, Q) &= \max_Q \min_P M(P, Q) \\ &= v \\ &= \text{"value" of game M}\end{aligned}$$

2. in words:
   1. $v = $ min max means:
      1. row player has strategy P^* such that $\forall$ column strategy Q, loss $M(P^*, Q) \leq v$
   2. $v = $ max min means:
      1. column player has strategy Q^* such that $\forall$ row strategy P, loss $M(P, Q^*) \geq v$

The Boosting Game

1. let $\{g_1, \cdots, g_N\}$ = space of all weak classifiers
2. row player $\rightarrow$ sampler
3. column player $\rightarrow$ weak learner
4. matrix M:
    1. row $\rightarrow$ example (x_i, y_i)
    2. column $\rightarrow$ weak classifier g_j
    3. $$M(i,j) = \begin{cases} 1 & \text{if } y_i = g_j(x) \\ 0 & \text{else} \end{cases}$$



Boosting and the Minmax Theorem

1. if:
    1. $\forall$ distributions over examples $\exists h$ with accuracy $\geq \frac{1}{2} + \gamma$
2. then:
    1. $\min_P \max_j M(P, j) \geq \frac{1}{2} + \gamma$
3. by minmax theorem:
    1. $\max_Q \min_i M(i, Q) \geq \frac{1}{2} + \gamma > \frac{1}{2}$
4. which means:
    1. $\exists$ weighted majority of classifiers which correctly classifies all examples with positive margin(2γ)
5. optimal margin $\leftrightarrow$ "value" of game

AdaBoost and Game Theory

1. AdaBoost is special case of general algorithm for solving games through repeated play
2. can show
   1. distribution over examples converges to (approximate) minmax strategy for boosting game
   2. weights on weak classifiers converge to (approximate) maxmin strategy
3. different instantiation of game-playing algorithm gives on-line learning algorithms (such as weighted majority algorithm)

Boost and Exponential Loss

1. many (most?) learning algorithms minimize a "loss" function
   1. e.g. least squares regression
2. training error proof sows AdaBoost actually minimizes

$$\prod_t Z_t = \frac{1}{m}\sum_i \exp(-y_i f(x_i))$$

   where $f(x) = \sum_t \alpha_t h_t(x)$
3. on each round, AdaBoost greedily chooses α_t and h_t to minimize loss.
4. we can prove that AdaBoost provably minimizes exponential loss.

Coordinate Descent

1. $\{g_1, \cdots, g_N\}$ =space of all weak classifiers
2. want to find $\lambda_1, \cdots, \lambda_N$ to minimize

$$L(\lambda_1, \cdots, \lambda_N) = \sum_i \exp(-y_i \sum_j \lambda_j g_j(x_i))$$

3. AdaBoost is actually doing coordinate descent on this optimization problem:
   1. initially, all $\lambda_j = 0$
   2. each round: choose one coordinate λ_j (corresponding to h_t) and update (increment by α_t)
   3. choose update causing biggest decrease in loss
4. powerful technique for minimizing over huge space of functions

Functional Gradient Descent

1. want to minimize

$$L(f) = L(f(x_1), \cdots, f(x_m)) = \sum_i \exp(-y_i f(x_i))$$

2. say have current estimate f and want to improve
3. to do gradient descent, would like update

$$f \leftarrow f - \alpha \nabla_f L(f)$$

4. but update resticted in class of weak classifiers

$$f \leftarrow f + \alpha h_t$$

5. so choose h_t "closest" to $-\nabla_f L(f)$
6. equivalent to AdaBoost