The Boosting Approach to Machine Learning

Maria-Florina Balcan

10/26/2015

Boosting

- General method for improving the accuracy of any given learning algorithm.
- Works by creating a series of challenge datasets s.t. even modest performance on these can be used to produce an overall high-accuracy predictor.
  - Works amazingly well in practice --- Adaboost and its variations one of the top 10 algorithms.
  - Backed up by solid foundations.

<u>Readings</u>:

- The Boosting Approach to Machine Learning: An Overview. Rob Schapire, 2001
- Theory and Applications of Boosting. NIPS tutorial.

http://www.cs.princeton.edu/~schapire/talks/nips-tutorial.pdf

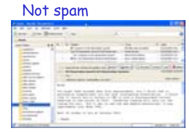
<u>Plan for today:</u>

- Motivation.
- A bit of history.
- Adaboost: algo, guarantees, discussion.
- Focus on supervised classification.

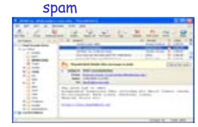
An Example: Spam Detection

- E.g., classify which emails are spam and which are important.

Not spam



spam

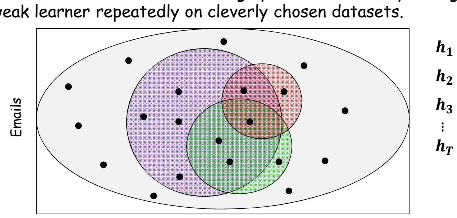


Key observation/motivation:

- Easy to find rules of thumb that are often correct.
  - E.g., "If buy now in the message, then predict spam."
  - E.g., "If say good-bye to debt in the message, then predict spam."
- Harder to find single rule that is very highly accurate.

An Example: Spam Detection

- Boosting: meta-procedure that takes in an algo for finding rules of thumb (weak learner). Produces a highly accurate rule, by calling the weak learner repeatedly on cleverly chosen datasets.



- apply weak learner to a subset of emails, obtain rule of thumb
- apply to 2nd subset of emails, obtain 2nd rule of thumb
- apply to 3[rd] subset of emails, obtain 3rd rule of thumb
- repeat T times; combine weak rules into a single highly accurate rule.

Boosting: Important Aspects

How to choose examples on each round?

- Typically, concentrate on "hardest" examples (those most often misclassified by previous rules of thumb)

How to combine rules of thumb into single prediction rule?

- take (weighted) majority vote of rules of thumb

Historically….

Weak Learning vs Strong/PAC Learning

- [Kearns & Valiant '88]: defined weak learning: being able to predict better than random guessing (error $\leq \frac{1}{2} - \gamma$) , consistently.
- Posed an open pb: "Does there exist a boosting algo that turns a weak learner into a strong PAC learner (that can produce arbitrarily accurate hypotheses)?"
- Informally, given "weak" learning algo that can consistently find classifiers of error $\leq \frac{1}{2} - \gamma$, a boosting algo would provably construct a single classifier with error $\leq \epsilon$.

Weak Learning vs Strong/PAC Learning

Strong (PAC) Learning

- $\exists$ algo A
- $\forall\, c \in H$
- $\forall D$
- $\forall\, \epsilon > 0$
- $\forall\, \delta > 0$
- A produces h s.t.:

  $\Pr[err(h) \geq \epsilon] \leq \delta$

Weak Learning

- $\exists$ algo A
- $\exists \gamma > 0$
- $\forall\, c \in H$
- $\forall D$
- $\forall\, \epsilon > \frac{1}{2} - \gamma$
- $\forall\, \delta > 0$
- A produces h s.t.

  $\Pr[err(h) \geq \epsilon] \leq \delta$

- [Kearns & Valiant '88]: defined weak learning & posed an open pb of finding a boosting algo.

Surprisingly….

Weak Learning =Strong (PAC) Learning

Original Construction [Schapire '89]:

- poly-time boosting algo, exploits that we can learn a little on every distribution.

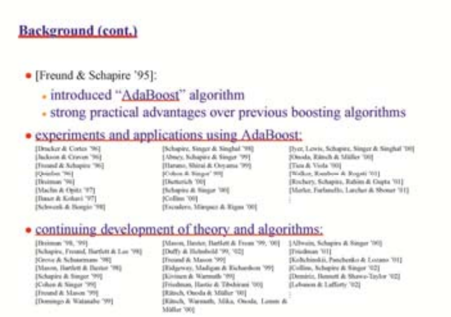
  - A modest booster obtained via calling the weak learning algorithm on 3 distributions.

    Error $= \beta < \frac{1}{2} - \gamma \rightarrow$ error $3\beta^2 - 2\beta^3$

  - Then amplifies the modest boost of accuracy by running this somehow recursively.

- Cool conceptually and technically, not very practical.

An explosion of subsequent work


Background (cont.)

- [Freund & Schapire '95]:
  - introduced "AdaBoost" algorithm
  - strong practical advantages over previous boosting algorithms
- experiments and applications using AdaBoost:
  [Drucker & Cortes '96] [Jackson & Craven '96] [Freund & Schapire '96] [Quinlan '96] [Breiman '96] [Maclin & Opitz '97] [Bauer & Kohavi '97] [Schwenk & Bengio '98] [Schapire, Singer & Singhal '98] [Abney, Schapire & Singer '99] [Haruno, Shirai & Ooyama '99] [Cohen & Singer '99] [Dietterich '00] [Schapire & Singer '00] [Collins '00] [Escudero, Màrquez & Rigau '00] [Iyer, Lewis, Schapire, Singer & Singhal '00] [Onoda, Rätsch & Müller '00] [Tieu & Viola '00] [Walker, Rambow & Rogati '01] [Rochery, Schapire, Rahim & Gupta '01] [Merler, Furlanello, Larcher & Sboner '01] …
- continuing development of theory and algorithms:
  [Breiman '98, '99] [Schapire, Freund, Bartlett & Lee '98] [Grove & Schuurmans '98] [Mason, Bartlett & Baxter '98] [Schapire & Singer '99] [Cohen & Singer '99] [Freund & Mason '99] [Domingo & Watanabe '99] [Mason, Baxter, Bartlett & Frean '99, '00] [Duffy & Helmbold '99, '02] [Freund & Mason '99] [Ridgeway, Madigan & Richardson '99] [Kivinen & Warmuth '99] [Friedman, Hastie & Tibshirani '00] [Rätsch, Onoda & Müller '00] [Rätsch, Warmuth, Mika, Onoda, Lemm & Müller '00] [Allwein, Schapire & Singer '00] [Friedman '01] [Koltchinskii, Panchenko & Lozano '01] [Collins, Schapire & Singer '02] [Demiriz, Bennett & Shawe-Taylor '02] [Lebanon & Lafferty '02] …

Adaboost (Adaptive Boosting)

"A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting"

[Freund-Schapire, JCSS'97]

Godel Prize winner 2003

Informal Description Adaboost

- Boosting: turns a weak algo into a strong (PAC) learner.

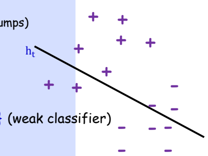
Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$; $x_i \in X, y_i \in Y = \{-1,1\}$

weak learning algo A (e.g., Naïve Bayes, decision stumps)

- For $t=1,2, \ldots, T$
  - Construct D_t on $\{x_1, \ldots, x_m\}$
  - Run A on D_t producing $h_t: X \rightarrow \{-1,1\}$ (weak classifier)

$\epsilon_t = P_{x_i \sim D_t}(h_t(x_i) \neq y_i)$ error of h_t over D_t

- Output $H_{final}(x) = \mathrm{sign}(\sum_{t=1} \alpha_t h_t(x))$



Roughly speaking D_{t+1} increases weight on x_i if h_t incorrect on x_i ; decreases it on x_i if h_t correct.

Adaboost (Adaptive Boosting)

- Weak learning algorithm A.
- For $t=1,2, \ldots, T$
  - **Construct D_t on $\{x_1, \ldots, x_m\}$**
  - Run A on D_t producing h_t

<u>Constructing D_t</u>

- D_1 uniform on $\{x_1, \ldots, x_m\}$ [i.e., $D_1(i) = \frac{1}{m}$]
- Given D_t and h_t set

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{\{-\alpha_t\}} \text{ if } y_i = h_t(x_i)$$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{\{\alpha_t\}} \text{ if } y_i \neq h_t(x_i)$$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{\{-\alpha_t y_i h_t(x_i)\}}$$

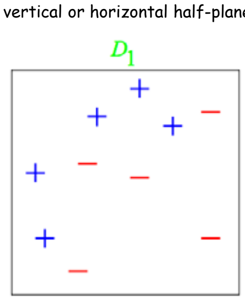
$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

D_{t+1} puts half of weight on examples x_i where h_t is incorrect & half on examples where h_t is correct

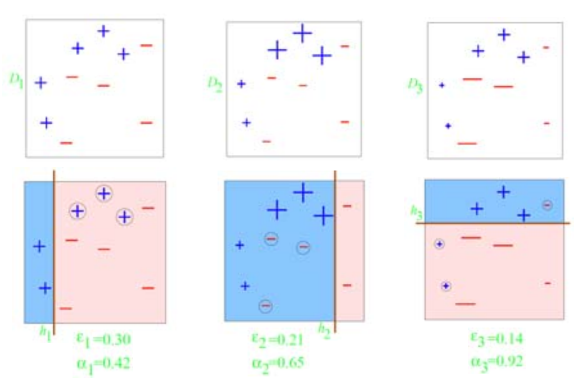
<u>Final hyp</u>: $H_{final}(x) = \mathrm{sign}(\sum_t \alpha_t h_t(x))$

Adaboost: A toy example

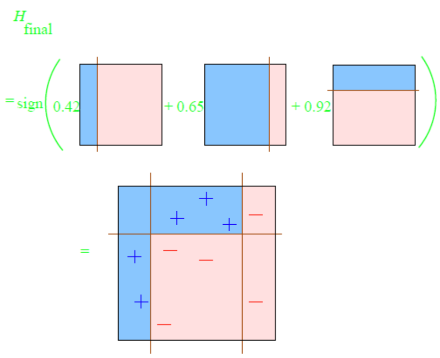
Weak classifiers: vertical or horizontal half-planes (a.k.a. decision stumps)



Adaboost: A toy example



Adaboost: A toy example



Adaboost (Adaptive Boosting)

- Weak learning algorithm A.
- For t=1,2, … ,T
  - **Construct D_t on $\{x_1, \ldots, x_m\}$**
  - Run A on D_t producing h_t

Constructing D_t

- D_1 uniform on $\{x_1, \ldots, x_m\}$ [i.e., $D_1(i) = \frac{1}{m}$]
- Given D_t and h_t set

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

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

D_{t+1} puts half of weight on examples x_i where h_t is incorrect & half on examples where h_t is correct

Final hyp: $H_{final}(x) = \text{sign}(\sum_t \alpha_t h_t(x))$

Nice Features of Adaboost

- Very general: a meta-procedure, it can use any weak learning algorithm!!! (e.g., Naïve Bayes, decision stumps)
- Very fast (single pass through data each round) & simple to code, no parameters to tune.
- Shift in mindset: goal is now just to find classifiers a bit better than random guessing.
- Grounded in rich theory.
- Relevant for big data age: quickly focuses on "core difficulties", well-suited to distributed settings, where data must be communicated efficiently [Balcan-Blum-Fine-Mansour COLT'12].

Analyzing Training Error

Theorem $\epsilon_t = 1/2 - \gamma_t$ (error of h_t over D_t)

$$err_S(H_{final}) \leq \exp\left[-2\sum_t \gamma_t^2\right]$$

So, if $\forall t, \gamma_t \geq \gamma > 0$, then $err_S(H_{final}) \leq \exp[-2\gamma^2 T]$

The training error drops exponentially in T!!!

To get $err_S(H_{final}) \leq \epsilon$, need only $T = O\left(\frac{1}{\gamma^2}\log\left(\frac{1}{\epsilon}\right)\right)$ rounds

Adaboost is adaptive

- Does not need to know γ or T a priori
- Can exploit $\gamma_t \gg \gamma$

Understanding the Updates & Normalization

Claim: D_{t+1} puts half of the weight on x_i where h_t was incorrect and half of the weight on x_i where h_t was correct.

Recall $D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{\{-\alpha_t y_i h_t(x_i)\}}$

Probabilities are equal!

$$\Pr_{D_{t+1}}[y_i \neq h_t(x_i)] = \sum_{i: y_i \neq h_t(x_i)} \frac{D_t(i)}{Z_t} e^{\alpha_t} = \epsilon_t \frac{1}{Z_t} e^{\alpha_t} = \frac{\epsilon_t}{Z_t}\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \frac{\sqrt{\epsilon_t(1-\epsilon_t)}}{Z_t}$$

$$\Pr_{D_{t+1}}[y_i = h_t(x_i)] = \sum_{i: y_i = h_t(x_i)} \frac{D_t(i)}{Z_t} e^{-\alpha_t} = \frac{1-\epsilon_t}{Z_t} e^{-\alpha_t} = \frac{1-\epsilon_t}{Z_t}\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} = \frac{\sqrt{(1-\epsilon_t)\epsilon_t}}{Z_t}$$

$$Z_t = \sum_{i:} D_t(i) e^{-\alpha_t y_i h_t(x_i)} = \sum_{i: y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i: y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t}$$

$$= (1-\epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Analyzing Training Error: Proof Intuition

Theorem $\epsilon_t = 1/2 - \gamma_t$ (error of h_t over D_t)

$$err_S(H_{final}) \leq \exp\left[-2 \sum_t \gamma_t^2\right]$$

- On round t, we increase weight of x_i for which h_t is wrong.
- If H_{final} incorrectly classifies x_i,
  - Then x_i incorrectly classified by (wtd) majority of h_t's.
  - Which implies final prob. weight of x_i is large.

    Can show probability $\geq \frac{1}{m}\left(\frac{1}{\prod_t Z_t}\right)$
- Since sum of prob. $= 1$, can't have too many of high weight.

  Can show # incorrectly classified $\leq m \left(\prod_t Z_t\right)$.

  And $\left(\prod_t Z_t\right) \to 0$.

Analyzing Training Error: Proof Math

Step 1: unwrapping recurrence: $D_{T+1}(i) = \frac{1}{m}\left(\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}\right)$
where $f(x_i) = \sum_t \alpha_t h_t(x_i)$. [Unthresholded weighted vote of h_i on x_i]

Step 2: $\text{errs}_S(H_{final}) \leq \prod_t Z_t$.

Step 3: $\prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \leq e^{-2\sum_t \gamma_t^2}$

Analyzing Training Error: Proof Math

Step 1: unwrapping recurrence: $D_{T+1}(i) = \frac{1}{m}\left(\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}\right)$
where $f(x_i) = \sum_t \alpha_t h_t(x_i)$.

Recall $D_1(i) = \frac{1}{m}$ and $D_{t+1}(i) = D_t(i)\frac{\exp(-y_i \alpha_t h_t(x_i))}{Z_t}$

$$D_{T+1}(i) = \frac{\exp(-y_i \alpha_T h_T(x_i))}{Z_T} \times D_T(i)$$

$$= \frac{\exp(-y_i \alpha_T h_T(x_i))}{Z_T} \times \frac{\exp(-y_i \alpha_{T-1} h_{T-1}(x_i))}{Z_{T-1}} \times D_{T-1}(i)$$

$\ldots \ldots.$

$$= \frac{\exp(-y_i \alpha_T h_T(x_i))}{Z_T} \times \cdots \times \frac{\exp(-y_i \alpha_1 h_1(x_i))}{Z_1} \frac{1}{m}$$

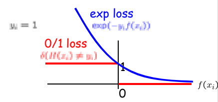
$$= \frac{1}{m}\frac{\exp(-y_i(\alpha_1 h_1(x_i) + \cdots + \alpha_T h_T(x_T)))}{Z_1 \cdots Z_T} = \frac{1}{m}\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}$$

Analyzing Training Error: Proof Math

Step 1: unwrapping recurrence: $D_{T+1}(i) = \frac{1}{m}\left(\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}\right)$
where $f(x_i) = \sum_t \alpha_t h_t(x_i)$.

Step 2: $\mathrm{err}_S(H_{final}) \le \prod_t Z_t$.

$$\mathrm{err}_S(H_{final}) = \frac{1}{m}\sum_i 1_{y_i \neq H_{final}(x_i)}$$
$$= \frac{1}{m}\sum_i 1_{y_i f(x_i) \le 0}$$
$$\le \frac{1}{m}\sum_i \exp(-y_i f(x_i))$$
$$= \sum_i D_{T+1}(i) \prod_t Z_t \;= \prod_t Z_t.$$



Analyzing Training Error: Proof Math

Step 1: unwrapping recurrence: $D_{T+1}(i) = \frac{1}{m}\left(\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}\right)$
where $f(x_i) = \sum_t \alpha_t h_t(x_i)$.

Step 2: $\mathrm{err}_S(H_{final}) \le \prod_t Z_t$.

Step 3: $\prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \le e^{-2\sum_t \gamma_t^2}$

Note: recall $Z_t = (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$

α_t minimizer of $\alpha \to (1-\epsilon_t)e^{-\alpha} + \epsilon_t e^{\alpha}$

Analyzing Training Error: Proof Math

Step 1: unwrapping recurrence: $D_{T+1}(i) = \frac{1}{m}\left(\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}\right)$
where $f(x_i) = \sum_t \alpha_t h_t(x)$.

Step 2: #mistakes$(H_{final}) = \sum_i 1_{y_i \neq H_{final}(x_i)}$
$= \sum_i 1_{y_i f(x_i) \le 0}$

Step 3: Each mistake of H_{final} has $D_{T+1}(i) \ge \frac{1}{m}\left(\frac{1}{\prod_t Z_t}\right)$, so total number of mistakes $\le m(\prod_t Z_t)$.

Step 4: $\prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \le e^{-2\sum_t \gamma_t^2}$

Analyzing Training Error: Proof Intuition

- Why does $(\prod_t Z_t) \to 0$?
- On round t, we have $1-\epsilon_t$ probability mass that h_t gets correct and ϵ_t that h_t gets incorrect.
- Our reweighting replaces these with their geometric mean $\sqrt{\epsilon_t(1-\epsilon_t)}$, which is **less than $\frac{1}{2}$**.
- So we normalize by $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$, which is less than 1.
- If $\epsilon_t = \frac{1}{2}$, then geometric mean would be $\frac{1}{2}$, would normalize by 1, and get nowhere, but that makes sense since h_t is just guessing!

Analyzing Training Error: Proof Intuition

Theorem $\epsilon_t = 1/2 - \gamma_t$ (error of h_t over D_t)

$$err_S(H_{final}) \leq \exp\left[-2\sum_t \gamma_t^2\right]$$

- On round t, we increase weight of x_i for which h_t is wrong.
- If H_{final} incorrectly classifies x_i,
  - Then x_i incorrectly classified by (wtd) majority of h_t's.
  - Which implies final prob. weight of x_i is large.

    Can show probability $\geq \frac{1}{m}\left(\frac{1}{\prod_t Z_t}\right)$
- Since sum of prob. $= 1$, can't have too many of high weight.

  Can show # incorrectly classified $\leq m\,(\prod_t Z_t)$.

  And $(\prod_t Z_t) \to 0$.

Generalization Guarantees

Theorem $err_S(H_{final}) \leq \exp\left[-2\sum_t \gamma_t^2\right]$ where $\epsilon_t = 1/2 - \gamma_t$

How about generalization guarantees? 

Original analysis [Freund&Schapire'97]

- H space of weak hypotheses; d=VCdim(H)

  H_{final} is a weighted vote, so the hypothesis class is:

  G={all fns of the form $\text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$ }

Theorem [Freund&Schapire'97]

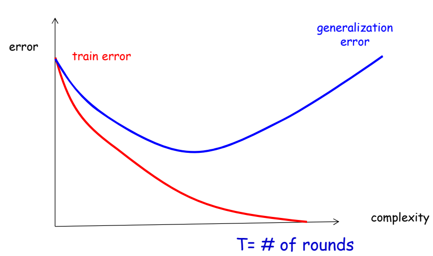
$\forall\, g \in G, err(g) \leq err_S(g) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$ T= # of rounds

Key reason: $\text{VC}dim(G) = \tilde{O}(dT)$ plus typical VC bounds.

Generalization Guarantees

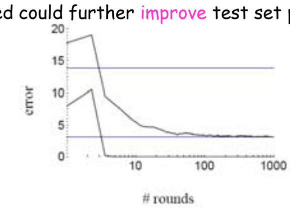
Theorem [Freund&Schapire'97]

$\forall\, g \in co(H), err(g) \leq err_S(g) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$ where d=VCdim(H)

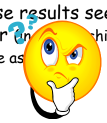
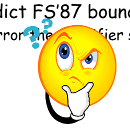


Generalization Guarantees

- Experiments with boosting showed that the test error of the generated classifier usually does not increase as its size becomes very large.
- Experiments showed that continuing to add new weak learners after correct classification of the training set had been achieved could further improve test set performance!!!



Generalization Guarantees

- Experimith boosting showed that the test error of the gene ssifier usually does not  as its size bec y large.
- Experiments showed that continuing to add weak learners after correct classification of the training set had been achieved could further improve test set performance!!!
- These results seem to contradict FS'87 bound and Occam's razor  hieve good test error  ier should be as simple as

How can we explain the experiments?

R. Schapire, Y. Freund, P. Bartlett, W. S. Lee. present in "*Boosting the margin: A new explanation for the effectiveness of voting methods*" a nice theoretical explanation.

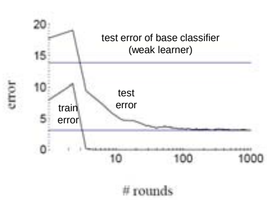
Key Idea:

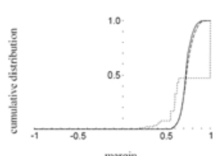
Training error does not tell the whole story.

We need also to consider the classification confidence!!

Boosting didn't seem to overfit…(!)

…because it turned out to be increasing the *margin* of the classifier





Error Curve, Margin Distr. Graph - Plots from [SFBL98]

Classification Margin

- H space of weak hypotheses. Define the convex hull of H to be

$co(H) = \{f = \sum_{t=1}^{T} \alpha_t h_t, \alpha_t \geq 0, \sum_{t=1}^{T} \alpha_t = 1, h_t \in H\}$

- Let $f \in co(H), f = \sum_{t=1}^{T} \alpha_t h_t, \alpha_t \geq 0, \sum_{t=1}^{T} \alpha_t = 1.$

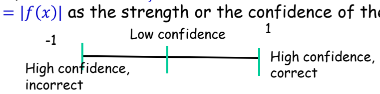
The majority vote rule H_f given by f (given by $H_f = sign(f(x))$) predicts wrongly on example (x, y) iff $yf(x) \leq 0$.

Definition: margin of H_f (or of f) on example (x, y) to be $yf(x)$.

$$yf(x) = y\sum_{t=1}^{T}[\alpha_t h_t(x)] = \sum_{t=1}^{T}[y\alpha_t h_t(x)] = \sum_{t: y=h_t(x)} \alpha_t - \sum_{t: y\neq h_t(x)} \alpha_t$$

The margin is positive iff $y = H_f(x)$.

See $|yf(x)| = |f(x)|$ as the strength or the confidence of the vote.



Gen. error as a function of margin Distributions

Assume that the examples are generated i.i.d. according to some distr. D over $X \times \{-1,1\}$; denote by $\Pr_D[\cdot]$ the probability when (x,y) is chosen from D.

If S is a training set (a sample of size m, $S = \{(x_1,y_1),\cdots(x_m,y_m)\}$), then we denote by $\Pr_S[\cdot]$ the probability when (x,y) is chosen uniformly at random from S.

Theorem 2: if H is finite, then with prob $\geq 1-\delta$, $\forall f \in co(H)$, $\forall \theta > 0$,

$$\Pr_D[yf(x) \leq 0] \leq \Pr_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\sqrt{\frac{\ln m \ln|H|}{\theta^2} + \ln\frac{1}{\delta}}\right)$$

Theorem 3: if H has VC-dim d, then w.prob $\geq 1-\delta$, $\forall f \in co(H)$, $\forall \theta > 0$,

$$\Pr_D[yf(x) \leq 0] \leq \Pr_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\sqrt{\frac{d \ln^2\frac{m}{d}}{\theta^2} + \ln\frac{1}{\delta}}\right)$$

Boosting and Margins

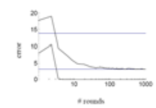
Theorem 2: If H is finite, then with prob $\geq 1-\delta$, $\forall f \in co(H)$, $\forall \theta > 0$,

$$\Pr_D[yf(x) \leq 0] \leq \Pr_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\sqrt{\frac{\ln m \ln|H|}{\theta^2} + \ln\frac{1}{\delta}}\right)$$

Theorem 3: VCdim$(H) = d$, then with prob $\geq 1-\delta$, $\forall f \in co(H)$, $\forall \theta > 0$,

$$\Pr_D[yf(x) \leq 0] \leq \Pr_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\sqrt{\frac{d \ln^2\frac{m}{d}}{\theta^2} + \ln\frac{1}{\delta}}\right)$$

Note: bound does **not** depend on the # of rounds of boosting, depends only on the complex. of the weak hyp space and the margin!

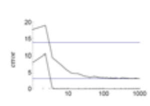


Boosting and Margins

Theorem 3: VCdim$(H) = d$, then with prob $\geq 1-\delta$, $\forall f \in co(H)$, $\forall \theta > 0$,

$$\Pr_D[yf(x) \leq 0] \leq \Pr_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\sqrt{\frac{d \ln^2\frac{m}{d}}{\theta^2} + \ln\frac{1}{\delta}}\right)$$

- If all training examples have large margins, then we can approximate the final classifier by a much smaller classifier.
- Can use this to prove that better margin ➔ smaller test error, regardless of the number of weak classifiers.
- Can also prove that boosting tends to increase the margin of training examples by concentrating on those of smallest margin.
- Although final classifier is getting larger, margins are likely to be increasing, so the final classifier is actually getting closer to a simpler classifier, driving down test error.



Boosting summary

- Shift in mindset: goal is now just to find classifiers a bit better than random guessing.
- Backed up by solid foundations.
- Adaboost work and its variations well in practice with many kinds of data (one of the top 10 algorithms).
- Very general: can use any given weak learning algorithm!!!
- Adaboost is very fast (single pass through data each round) & simple to code, no parameters to tune.
- Relevant for big data age: quickly focuses on "core difficulties", so well-suited to distributed settings, where data must be communicated efficiently [Balcan-Blum-Fine-Mansour COLT'12].