Theory and Applications of Boosting

Rob Schapire

Example: "How May I Help You?"

[Gorin et al.]

- goal: automatically categorize type of call requested by phone customer (Collect, CallingCard, PersonToPerson, etc.)
  - `yes I'd like to place a collect call long distance please` (Collect)
  - `operator I need to make a call but I need to bill it to my office` (ThirdNumber)
  - `yes I'd like to place a call on my master card please` (CallingCard)
  - `I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill` (BillingCredit)
- observation:
  - easy to find "rules of thumb" that are "often" correct
    - e.g.: "IF '`card`' occurs in utterance
      THEN predict 'CallingCard' "
  - hard to find single highly accurate prediction rule

The Boosting Approach

- devise computer program for deriving rough rules of thumb
- apply procedure to subset of examples
- obtain rule of thumb
- apply to 2nd subset of examples
- obtain 2nd rule of thumb
- repeat T times

Key Details

- how to choose examples on each round?
  - 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

Boosting

- boosting = general method of converting rough rules of thumb into highly accurate prediction rule
- technically:
  - assume given "weak" learning algorithm that can consistently find classifiers ("rules of thumb") at least slightly better than random, say, accuracy $\geq 55\%$ (in two-class setting) ["weak learning assumption"]
  - given sufficient data, a boosting algorithm can provably construct single classifier with very high accuracy, say, 99%

Outline of Tutorial

- basic algorithm and core theory
- fundamental perspectives
- practical extensions
- advanced topics

Preamble: Early History

Strong and Weak Learnability

- boosting's roots are in "PAC" learning model [Valiant '84]
- get random examples from unknown, arbitrary distribution
- strong PAC learning algorithm:
  - for any distribution
    with high probability
    given polynomially many examples (and polynomial time)
    can find classifier with arbitrarily small generalization error
- weak PAC learning algorithm
  - same, but generalization error only needs to be slightly better than random guessing ($\frac{1}{2} - \gamma$)
- [Kearns & Valiant '88]:
  - does weak learnability imply strong learnability?

If Boosting Possible, Then...

- can use (fairly) wild guesses to produce highly accurate predictions
- if can learn "part way" then can learn "all the way"
- should be able to improve any learning algorithm
- for any learning problem:
    - either can always learn with nearly perfect accuracy
    - or there exist cases where cannot learn even slightly better than random guessing

First Boosting Algorithms

- [Schapire '89]:
  - first provable boosting algorithm
- [Freund '90]:
  - "optimal" algorithm that "boosts by majority"
- [Drucker, Schapire & Simard '92]:
  - first experiments using boosting
  - limited by practical drawbacks
- [Freund & Schapire '95]:
  - introduced "AdaBoost" algorithm
  - strong practical advantages over previous boosting algorithms

Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error
  and the margins theory
- experiments and applications

Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error
  and the margins theory
- experiments and applications

A Formal Description of Boosting

- given training set $(x_1, y_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \ldots, T$:
    - construct distribution D_t on $\{1, \ldots, m\}$
    - find weak classifier ("rule of thumb")

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

      with small error ϵ_t on D_t:

      $$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$
- output final classifier H_{final}

AdaBoost

[with Freund]

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

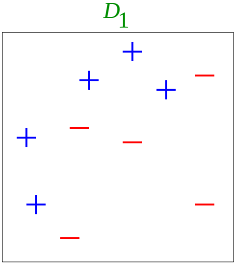
$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i)) \end{aligned}$$

where $Z_t =$ normalization factor

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

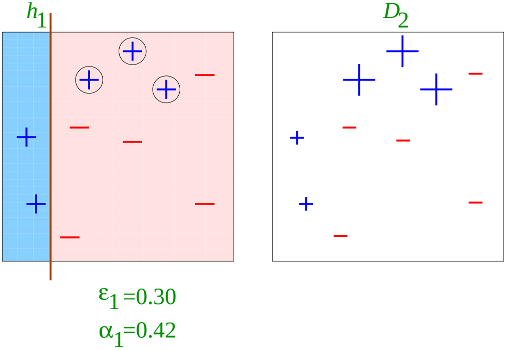
- final classifier:
    - $H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$

Toy Example

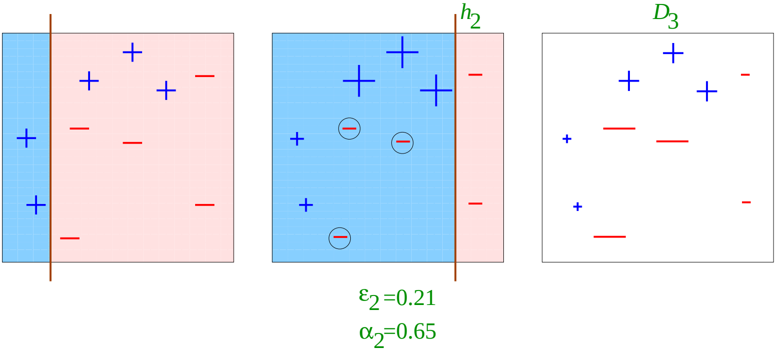


weak classifiers = vertical or horizontal half-planes

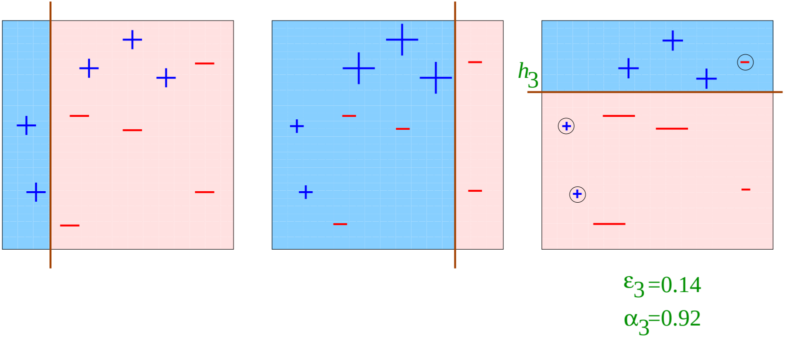
Round 1



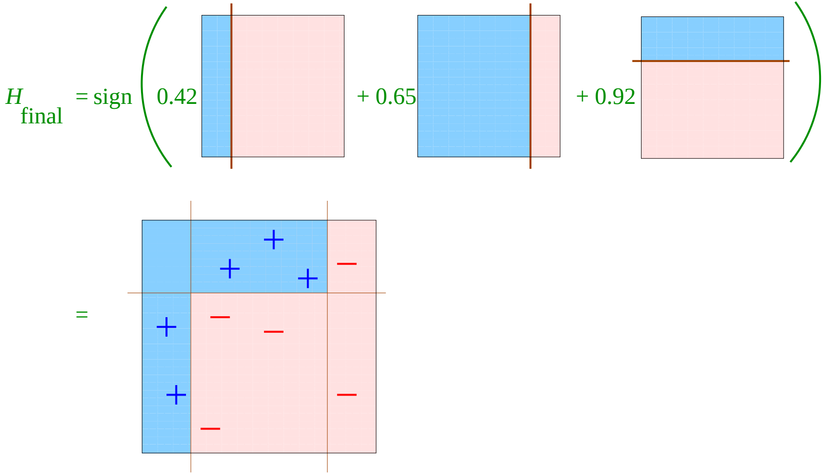
Round 2



Round 3



Final Classifier



Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error
  and the margins theory
- experiments and applications

Analyzing the Training Error

[with Freund]

- Theorem:
  - write ϵ_t as $\frac{1}{2} - \gamma_t$ [$\gamma_t =$ "edge"]
  - then

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

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

Proof

- let $F(x) = \sum_t \alpha_t h_t(x) \Rightarrow H_{\text{final}}(x) = \text{sign}(F(x))$
- *Step 1*: unwrapping recurrence:

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

Proof (cont.)

- *Step 2*: $\text{training error}(H_{\text{final}}) \leq \prod_t Z_t$
- Proof:

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

Proof (cont.)

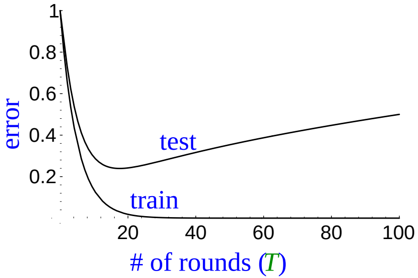
- *Step 3*: $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$
- Proof:

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

Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error
  and the margins theory
- experiments and applications

How Will Test Error Behave? (A First Guess)



expect:

- training error to continue to drop (or reach zero)
- test error to increase when H_{final} becomes "too complex"
  - "Occam's razor"
  - overfitting
    - hard to know when to stop training

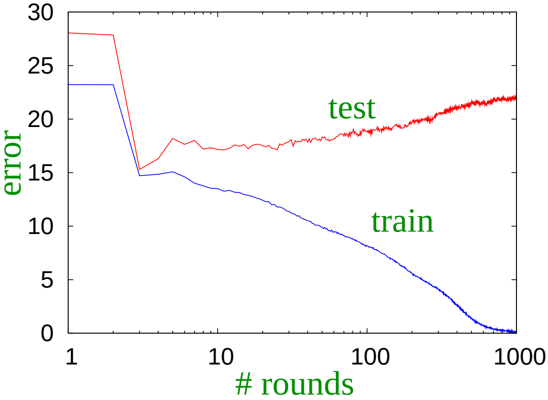
Technically...

- with high probability:

$$\text{generalization error} \leq \text{training error} + \tilde{O}\left(\sqrt{\frac{dT}{m}}\right)$$

  - bound depends on
    - $m = \#$ training examples
    - $d =$ "complexity" of weak classifiers
    - $T = \#$ rounds
- generalization error $= \mathrm{E}\left[\text{test error}\right]$
- predicts overfitting

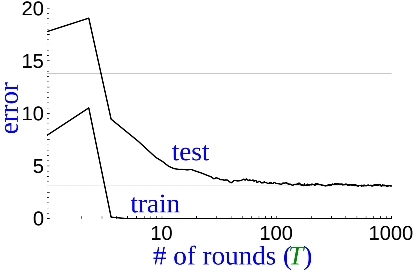
Overfitting Can Happen



(boosting "stumps" on heart-disease dataset)

- but often doesn't...

Actual Typical Run



(boosting C4.5 on "letter" dataset)

- test error does not increase, even after 1000 rounds
  - (total size $>$ 2,000,000 nodes)
- test error continues to drop even after training error is zero!

	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

- Occam's razor wrongly predicts "simpler" rule is better

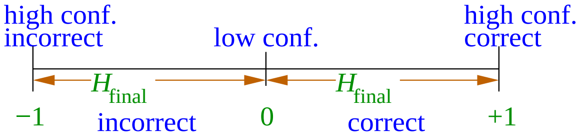
A Better Story: The Margins Explanation

[with Freund, Bartlett & Lee]

- key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications
- recall: H_{final} is weighted majority vote of weak classifiers
- measure confidence by margin = strength of the vote

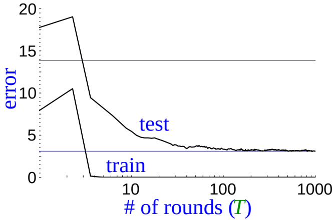
  = (weighted fraction voting correctly)

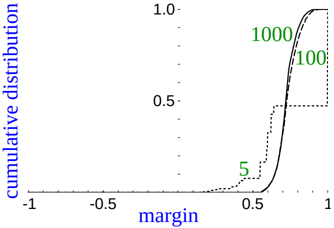
  −(weighted fraction voting incorrectly)



Empirical Evidence: The Margin Distribution

- margin distribution
  = cumulative distribution of margins of training examples





	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

Theoretical Evidence: Analyzing Boosting Using Margins

- Theorem: large margins $\Rightarrow$ better bound on generalization error (independent of number of rounds)
  - proof idea: if all margins are large, then can approximate final classifier by a much smaller classifier (just as polls can predict not-too-close election)
- Theorem: boosting tends to increase margins of training examples (given weak learning assumption)
  - moreover, larger edges $\Rightarrow$ larger margins
  - proof idea: similar to training error proof
- 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

More Technically...

- with high probability, $\forall\theta > 0$:

$$\text{generalization error} \le \hat{\Pr}[\text{margin} \le \theta] + \tilde{O}\left(\frac{\sqrt{d/m}}{\theta}\right)$$

($\hat{\Pr}[\,] =$ empirical probability)

  - bound depends on
    - $m = \#$ training examples
    - $d =$ "complexity" of weak classifiers
    - entire distribution of margins of training examples
- $\hat{\Pr}[\text{margin} \le \theta] \to 0$ exponentially fast (in T) if $\epsilon_t < \frac{1}{2} - \theta$ ($\forall t$)
  - so: if weak learning assumption holds, then all examples will quickly have "large" margins

Consequences of Margins Theory

- predicts good generalization with no overfitting if:
    - weak classifiers have large edges (implying large margins)
    - weak classifiers not too complex relative to size of training set
- e.g., boosting decision trees resistant to overfitting since trees often have large edges and limited complexity
- overfitting may occur if:
    - small edges (underfitting), or
    - overly complex weak classifiers
- e.g., heart-disease dataset:
    - stumps yield small edges
    - also, small dataset

Improved Boosting with Better Margin-Maximization?

- can design algorithms more effective than AdaBoost at maximizing the minimum margin
- in practice, often perform worse [Breiman]
- why??
- more aggressive margin maximization seems to lead to:
  - more complex weak classifiers (even using same weak learner); or
  - higher minimum margins, but margin distributions that are lower overall

[with Reyzin]

Comparison to SVM's

- both AdaBoost and SVM's:
  - work by maximizing "margins"
  - find linear threshold function in high-dimensional space
- differences:
  - margin measured slightly differently (using different norms)
  - SVM's handle high-dimensional space using kernel trick; AdaBoost uses weak learner to search over space

Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error
  and the margins theory
- experiments and applications

Practical Advantages of AdaBoost

- fast
- simple and easy to program
- no parameters to tune (except T)
- flexible — can combine with any learning algorithm
- no prior knowledge needed about weak learner
- provably effective, provided can consistently find rough rules of thumb
  - → shift in mind set — goal now is merely to find classifiers barely better than random guessing
- versatile
  - can use with data that is textual, numeric, discrete, etc.
  - has been extended to learning problems well beyond binary classification

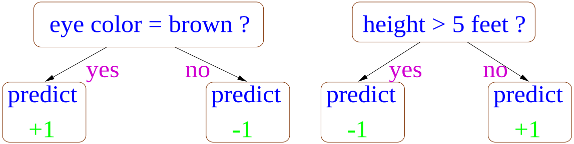
Caveats

- performance of AdaBoost depends on data and weak learner
- consistent with theory, AdaBoost can fail if
  - weak classifiers too complex
    - → overfitting
  - weak classifiers too weak ($\gamma_t \to 0$ too quickly)
    - → underfitting
    - → low margins → overfitting
- empirically, AdaBoost seems especially susceptible to uniform noise

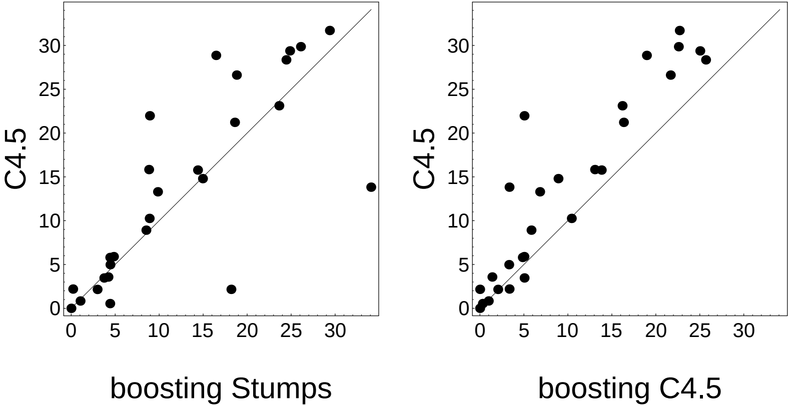
UCI Experiments

[with Freund]

- tested AdaBoost on UCI benchmarks
- used:
  - C4.5 (Quinlan's decision tree algorithm)
  - "decision stumps": very simple rules of thumb that test on single attributes



UCI Results

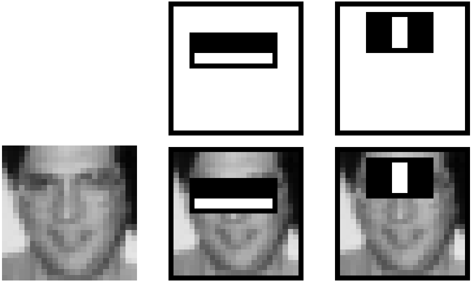


C4.5
30
25
20
15
10
5
0
0
5
10
15
20
25
30
boosting Stumps
C4.5
boosting C4.5

Application: Detecting Faces

[Viola & Jones]

- problem: find faces in photograph or movie
- weak classifiers: detect light/dark rectangles in image



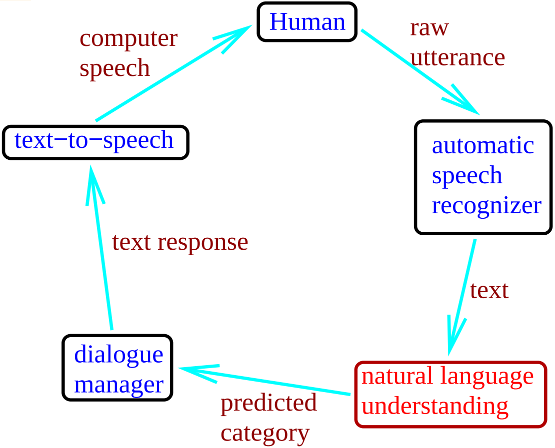
- many clever tricks to make extremely fast and accurate

Application: Human-computer Spoken Dialogue

[with Rahim, Di Fabbrizio, Dutton, Gupta, Hollister & Riccardi]

- application: automatic "store front" or "help desk" for AT&T Labs' Natural Voices business
- caller can request demo, pricing information, technical support, sales agent, etc.
- interactive dialogue

How It Works



- NLU's job: classify caller utterances into 24 categories (demo, sales rep, pricing info, yes, no, etc.)
- weak classifiers: test for presence of word or phrase

Problem: Labels are Expensive

- for spoken-dialogue task
    - getting examples is cheap
    - getting labels is expensive
        - must be annotated by humans
- how to reduce number of labels needed?

Active Learning

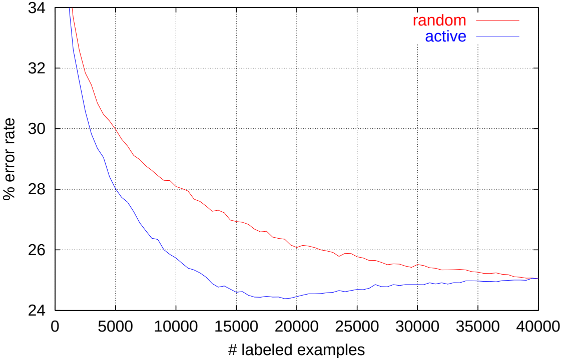
[with Tur & Hakkani-Tür]

- idea:
  - use selective sampling to choose which examples to label
  - focus on least confident examples [Lewis & Gale]
- for boosting, use (absolute) margin as natural confidence measure [Abe & Mamitsuka]

Labeling Scheme

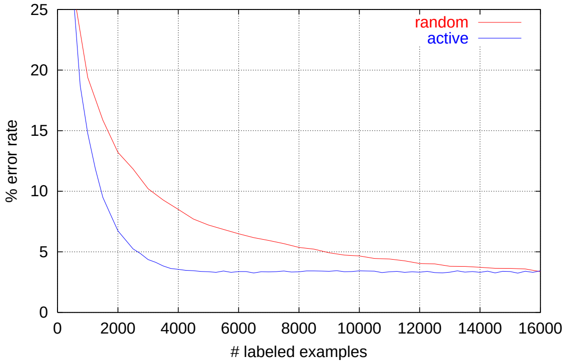
- start with pool of unlabeled examples
- choose (say) 500 examples at random for labeling
- run boosting on all labeled examples
    - get combined classifier F
- pick (say) 250 additional examples from pool for labeling
    - choose examples with minimum $|F(x)|$ (proportional to absolute margin)
- repeat

Results: How-May-I-Help-You?



	first reached		% label
% error	random	active	savings
28	11,000	5,500	50
26	22,000	9,500	57
25	40,000	13,000	68

Results: Letter



% error	first reached random	first reached active	% label savings
10	3,500	1,500	57
5	9,000	2,750	69
4	13,000	3,500	73

Fundamental Perspectives

- game theory
- loss minimization
- an information-geometric view

Fundamental Perspectives

- game theory
- loss minimization
- an information-geometric view

Just a Game

[with Freund]

- can view boosting as a game, a formal interaction between booster and weak learner
- on each round t:
  - booster chooses distribution D_t
  - weak learner responds with weak classifier h_t
- game theory: studies interactions between all sorts of "players"

Games

- game defined by matrix $\mathbf{M}$:

	Rock	Paper	Scissors
Rock	$1/2$	1	0
Paper	0	$1/2$	1
Scissors	1	0	$1/2$

- row player ("Mindy") chooses row i
- column player ("Max") chooses column j (simultaneously)
- Mindy's goal: minimize her loss $\mathbf{M}(i, j)$
- assume (wlog) all entries in $[0, 1]$

Randomized Play

- usually allow randomized play:
  - Mindy chooses distribution $\mathbf{P}$ over rows
  - Max chooses distribution $\mathbf{Q}$ over columns (simultaneously)
- Mindy's (expected) loss

$$\begin{aligned} &= \sum_{i,j} \mathbf{P}(i)\mathbf{M}(i,j)\mathbf{Q}(j) \\ &= \mathbf{P}^{\top}\mathbf{M}\mathbf{Q} \equiv \mathbf{M}(\mathbf{P},\mathbf{Q}) \end{aligned}$$

- i, j = "pure" strategies
- $\mathbf{P}$, $\mathbf{Q}$ = "mixed" strategies
- m = # rows of $\mathbf{M}$
- also write $\mathbf{M}(i,\mathbf{Q})$ and $\mathbf{M}(\mathbf{P},j)$ when one side plays pure and other plays mixed

Sequential Play

- say Mindy plays before Max
- if Mindy chooses $\mathbf{P}$ then Max will pick $\mathbf{Q}$ to maximize $\mathbf{M}(\mathbf{P},\mathbf{Q})$ $\Rightarrow$ loss will be

$$L(\mathbf{P}) \equiv \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P},\mathbf{Q})$$

- so Mindy should pick $\mathbf{P}$ to minimize $L(\mathbf{P})$ $\Rightarrow$ loss will be

$$\min_{\mathbf{P}} L(\mathbf{P}) = \min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P},\mathbf{Q})$$

- similarly, if Max plays first, loss will be

$$\max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\mathbf{Q})$$

Minmax Theorem

- playing second (with knowledge of other player's move) cannot be worse than playing first, so:

$$\underbrace{\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}, \mathbf{Q})}_{\text{Mindy plays first}} \geq \underbrace{\max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P}, \mathbf{Q})}_{\text{Mindy plays second}}$$

- von Neumann's minmax theorem:

$$\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) = \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P}, \mathbf{Q})$$

- in words: no advantage to playing second

Optimal Play

- minmax theorem:

$$\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P},\mathbf{Q}) = \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\mathbf{Q}) = \text{value } v \text{ of game}$$

- optimal strategies:
  - $\mathbf{P}^* = \arg\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P},\mathbf{Q}) =$ minmax strategy
  - $\mathbf{Q}^* = \arg\max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\mathbf{Q}) =$ maxmin strategy
- in words:
  - Mindy's minmax strategy $\mathbf{P}^*$ guarantees loss $\leq v$ (regardless of Max's play)
  - optimal because Max has maxmin strategy $\mathbf{Q}^*$ that can force loss $\geq v$ (regardless of Mindy's play)
- e.g.: in RPS, $\mathbf{P}^* = \mathbf{Q}^* =$ uniform
- solving game $=$ finding minmax/maxmin strategies

Weaknesses of Classical Theory

- seems to fully answer how to play games — just compute minmax strategy (e.g., using linear programming)
- weaknesses:
  - game **M** may be unknown
  - game **M** may be extremely large
  - opponent may not be fully adversarial
    - may be possible to do better than value v
    - e.g.:

      Lisa (thinks):

      Poor predictable Bart, always takes Rock.

      Bart (thinks):

      Good old Rock, nothing beats that.

Repeated Play

- if only playing once, hopeless to overcome ignorance of game $\mathbf{M}$ or opponent
- but if game played repeatedly, may be possible to learn to play well
- goal: play (almost) as well as if knew game and how opponent would play ahead of time

Repeated Play (cont.)

- $\mathbf{M}$ unknown
- for $t = 1, \ldots, T$:
  - Mindy chooses $\mathbf{P}_t$
  - Max chooses $\mathbf{Q}_t$ (possibly depending on $\mathbf{P}_t$)
  - Mindy's loss $= \mathbf{M}(\mathbf{P}_t, \mathbf{Q}_t)$
  - Mindy observes loss $\mathbf{M}(i, \mathbf{Q}_t)$ of each pure strategy i
- want:

$$\underbrace{\frac{1}{T}\sum_{t=1}^{T}\mathbf{M}(\mathbf{P}_t, \mathbf{Q}_t)}_{\text{actual average loss}} \le \underbrace{\min_{\mathbf{P}} \frac{1}{T}\sum_{t=1}^{T}\mathbf{M}(\mathbf{P}, \mathbf{Q}_t)}_{\text{best loss (in hindsight)}} + [\text{“small amount”}]$$

Multiplicative-weights Algorithm (MW)

[with Freund]

- choose $\eta > 0$
- initialize: $\mathbf{P}_1 =$ uniform
- on round t:

$$\mathbf{P}_{t+1}(i) = \frac{\mathbf{P}_t(i)\ \exp\left(-\eta\,\mathbf{M}(i, \mathbf{Q}_t)\right)}{\text{normalization}}$$

- idea: decrease weight of strategies suffering the most loss
- directly generalizes [Littlestone & Warmuth]
- other algorithms:
  - [Hannan'57]
  - [Blackwell'56]
  - [Foster & Vohra]
  - [Fudenberg & Levine]

  ⋮

Analysis

- Theorem: can choose η so that, for any game $\mathbf{M}$ with m rows, and any opponent,

$$\underbrace{\frac{1}{T}\sum_{t=1}^{T}\mathbf{M}(\mathbf{P}_t,\mathbf{Q}_t)}_{\text{actual average loss}} \quad\le\quad \underbrace{\min_{\mathbf{P}}\frac{1}{T}\sum_{t=1}^{T}\mathbf{M}(\mathbf{P},\mathbf{Q}_t)}_{\text{best average loss }(\le v)} \;+\Delta_T$$

where $\Delta_T = O\left(\sqrt{\frac{\ln m}{T}}\right) \to 0$

- regret Δ_T is:
  - logarithmic in # rows m
  - independent of # columns
- therefore, can use when working with very large games

Solving a Game

[with Freund]

- suppose game $\mathbf{M}$ played repeatedly
    - Mindy plays using MW
    - on round t, Max chooses best response:

$$\mathbf{Q}_t = \arg\max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}_t, \mathbf{Q})$$

- let

$$\overline{\mathbf{P}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{P}_t, \quad \overline{\mathbf{Q}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{Q}_t$$

- can prove that $\overline{\mathbf{P}}$ and $\overline{\mathbf{Q}}$ are Δ_T-approximate minmax and maxmin strategies:

$$\max_{\mathbf{Q}} \mathbf{M}(\overline{\mathbf{P}}, \mathbf{Q}) \leq v + \Delta_T$$

and

$$\min_{\mathbf{P}} \mathbf{M}(\mathbf{P}, \overline{\mathbf{Q}}) \geq v - \Delta_T$$

Boosting as a Game

- Mindy (row player) $\leftrightarrow$ booster
- Max (column player) $\leftrightarrow$ weak learner
- matrix $\mathbf{M}$:
  - row $\leftrightarrow$ training example
  - column $\leftrightarrow$ weak classifier
  - $\mathbf{M}(i,j) =$
    $$\begin{cases} 1 & \text{if } j\text{-th weak classifier correct on } i\text{-th training example} \\ 0 & \text{else} \end{cases}$$
  - encodes which weak classifiers correct on which examples
  - huge # of columns — one for every possible weak classifier

Boosting and the Minmax Theorem

- γ-weak learning assumption:
  - for every distribution on examples
  - can find weak classifier with weighted error $\leq \frac{1}{2} - \gamma$
- equivalent to:

$$(\text{value of game } \mathbf{M}) \geq \frac{1}{2} + \gamma$$

- by minmax theorem, implies that:
  - $\exists$ some weighted majority classifier that correctly classifies all training examples with margin $\geq 2\gamma$
  - further, weights are given by maxmin strategy of game **M**

Idea for Boosting

- maxmin strategy of **M** has perfect (training) accuracy and large margins
- find approximately using earlier algorithm for solving a game
    - i.e., apply MW to **M**
- yields (variant of) AdaBoost

AdaBoost and Game Theory

- summarizing:
  - weak learning assumption implies maxmin strategy for $\mathbf{M}$ defines large-margin classifier
  - AdaBoost finds maxmin strategy by applying general algorithm for solving games through repeated play
- consequences:
  - weights on weak classifiers converge to (approximately) maxmin strategy for game $\mathbf{M}$
  - (average) of distributions D_t converges to (approximately) minmax strategy
  - margins and edges connected via minmax theorem
  - explains why AdaBoost maximizes margins
- different instantiation of game-playing algorithm gives online learning algorithms (such as weighted majority algorithm)

Fundamental Perspectives

- game theory
- loss minimization
- an information-geometric view

AdaBoost and Loss Minimization

- many (most?) learning and statistical methods can be viewed as minimizing loss (a.k.a. cost or objective) function measuring fit to data:
    - e.g. least squares regression $\sum_i (F(x_i) - y_i)^2$
- AdaBoost also minimizes a loss function
- helpful to understand because:
    - clarifies goal of algorithm and useful in proving convergence properties
    - decoupling of algorithm from its objective means:
        - faster algorithms possible for same objective
        - same algorithm may generalize for new learning challenges

What AdaBoost Minimizes

- recall proof of training error bound:
  - $\text{training error}(H_{\text{final}}) \le \prod_t Z_t$
  - $Z_t = \epsilon_t e^{\alpha_t} + (1-\epsilon_t)e^{-\alpha_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$
- closer look:
  - α_t chosen to minimize Z_t
  - h_t chosen to minimize ϵ_t
    - same as minimizing Z_t
      (since increasing in ϵ_t on $[0, 1/2]$)
- so: both AdaBoost and weak learner minimize Z_t on round t
  - equivalent to greedily minimizing $\prod_t Z_t$

AdaBoost and Exponential Loss

- so AdaBoost is greedy procedure for minimizing exponential loss

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

where

$$F(x) = \sum_t \alpha_t h_t(x)$$

- why exponential loss?
  - intuitively, strongly favors $F(x_i)$ to have same sign as y_i
  - upper bound on training error
    - smooth and convex (but very loose)
- how does AdaBoost minimize it?

Coordinate Descent

[Breiman]

- $\{g_1, \ldots, g_N\}$ = space of all weak classifiers
- then can write $F(x) = \sum_t \alpha_t h_t(x) = \sum_{j=1}^{N} \lambda_j g_j(x)$
- want to find $\lambda_1, \ldots, \lambda_N$ to minimize

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

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

Functional Gradient Descent

[Mason et al.][Friedman]

- want to minimize

$$\mathcal{L}(F) = \mathcal{L}(F(x_1), \ldots, F(x_m)) = \sum_i \exp(-y_i F(x_i))$$

- say have current estimate F and want to improve
- to do gradient descent, would like update

$$F \leftarrow F - \alpha \nabla_F \mathcal{L}(F)$$

- but update restricted in class of weak classifiers

$$F \leftarrow F + \alpha h_t$$

- so choose h_t "closest" to $-\nabla_F \mathcal{L}(F)$
- equivalent to AdaBoost

Estimating Conditional Probabilities

[Friedman, Hastie & Tibshirani]

- often want to estimate probability that $y = +1$ given x
- AdaBoost minimizes (empirical version of):

$$\mathrm{E}_{x,y}\left[e^{-yF(x)}\right] = \mathrm{E}_x\left[\Pr\left[y = +1|x\right] e^{-F(x)} + \Pr\left[y = -1|x\right] e^{F(x)}\right]$$

where x, y random from true distribution

- over all F, minimized when

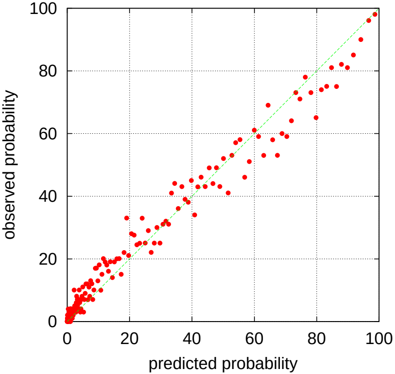
$$F(x) = \frac{1}{2} \cdot \ln\left(\frac{\Pr\left[y = +1|x\right]}{\Pr\left[y = -1|x\right]}\right)$$

or

$$\Pr\left[y = +1|x\right] = \frac{1}{1 + e^{-2F(x)}}$$

- so, to convert F output by AdaBoost to probability estimate, use same formula

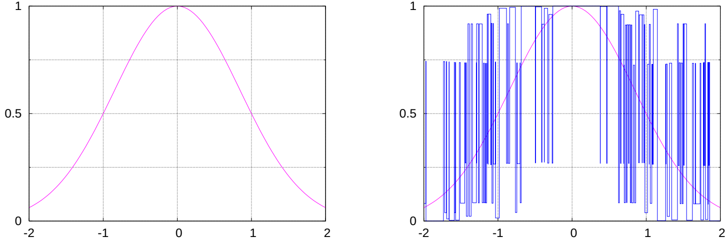
Calibration Curve



- order examples by F value output by AdaBoost
- break into bins of fixed size
- for each bin, plot a point:
    - x-value: average estimated probability of examples in bin
    - y-value: actual fraction of positive examples in bin

A Synthetic Example

- $x \in [-2, +2]$ uniform
- $\Pr[y = +1 | x] = 2^{-x^2}$
- $m = 500$ training examples



- if run AdaBoost with stumps and convert to probabilities, result is poor
  - extreme overfitting

Regularization

- AdaBoost minimizes

$$L(\boldsymbol{\lambda}) = \sum_i \exp\left(-y_i \sum_j \lambda_j g_j(x_i)\right)$$

- to avoid overfitting, want to constrain $\boldsymbol{\lambda}$ to make solution "smoother"
- (ℓ_1) regularization:

$$\begin{aligned} &\text{minimize: } \; L(\boldsymbol{\lambda}) \\ &\text{subject to: } \|\boldsymbol{\lambda}\|_1 \le B \end{aligned}$$

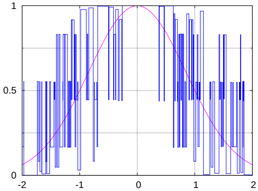
- or:

$$\text{minimize: } \; L(\boldsymbol{\lambda}) + \beta \|\boldsymbol{\lambda}\|_1$$

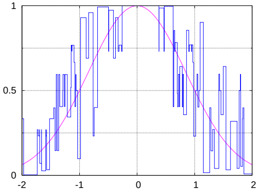
- other norms possible
  - ℓ_1 ("lasso") currently popular since encourages sparsity

[Tibshirani]

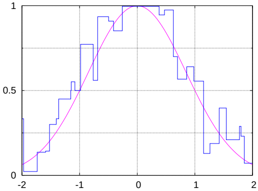
Regularization Example



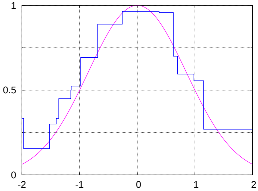
$\beta = 10^{-3}$



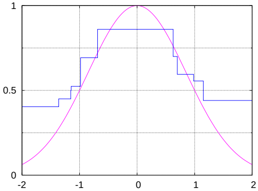
$\beta = 10^{-2.5}$



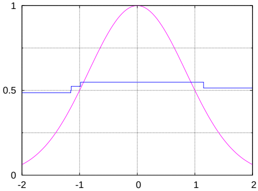
$\beta = 10^{-2}$



$\beta = 10^{-1.5}$



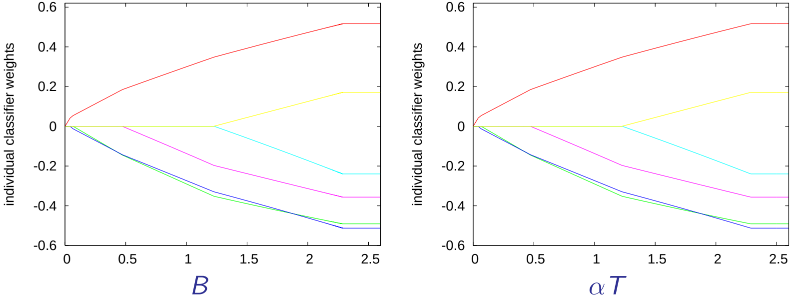
$\beta = 10^{-1}$



$\beta = 10^{-0.5}$

Regularization and AdaBoost

[Hastie, Tibshirani & Friedman; Rosset, Zhu & Hastie]



- Experiment 1: regularized solution vectors $\boldsymbol{\lambda}$ plotted as function of B
- Experiment 2: AdaBoost run with α_t fixed to (small) α
  - solution vectors $\boldsymbol{\lambda}$ plotted as function of αT
- plots are identical!
- can prove under certain (but not all) conditions that results will be the same (as $\alpha \to 0$) [Zhao & Yu]

Regularization and AdaBoost

- suggests stopping AdaBoost early is akin to applying ℓ_1-regularization
- caveats:
    - does not strictly apply to AdaBoost (only variant)
    - not helpful when boosting run "to convergence" (would correspond to very weak regularization)
- in fact, in limit of vanishingly weak regularization ($B \to \infty$), solution converges to maximum margin solution

[Rosset, Zhu & Hastie]

Benefits of Loss-Minimization View

- immediate generalization to other loss functions and learning problems
    - e.g. squared error for regression
    - e.g. logistic regression (by only changing one line of AdaBoost)
- sensible approach for converting output of boosting into conditional probability estimates
- helpful connection to regularization
- basis for proving AdaBoost is statistically "consistent"
    - i.e., under right assumptions, converges to best possible classifier [Bartlett & Traskin]

A Note of Caution

- tempting (but incorrect!) to conclude:
  - AdaBoost is just an algorithm for minimizing exponential loss
  - AdaBoost works only because of its loss function

  ∴ more powerful optimization techniques for same loss should work even better
- incorrect because:
  - other algorithms that minimize exponential loss can give very poor generalization performance compared to AdaBoost
- for example...

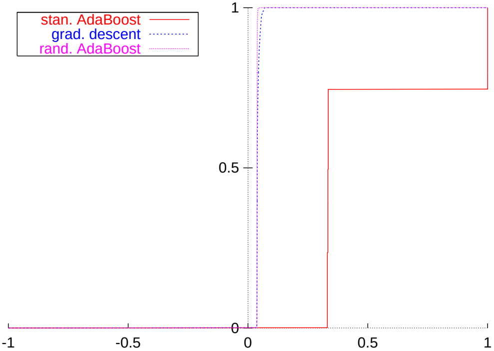
An Experiment

- data:
    - instances x uniform from $\{-1, +1\}^{10,000}$
    - label $y =$ majority vote of three coordinates
    - weak classifier = single coordinate (or its negation)
    - training set size $m = 1000$
- algorithms (all provably minimize exponential loss):
    - standard AdaBoost
    - gradient descent on exponential loss
    - AdaBoost, but in which weak classifiers chosen at random
- results:

exp. loss	% test error [# rounds]					
	stand. AdaB.		grad. desc.		random AdaB.	
10^{-10}	0.0	[94]	40.7	[5]	44.0	[24,464]
10^{-20}	0.0	[190]	40.8	[9]	41.6	[47,534]
10^{-40}	0.0	[382]	40.8	[21]	40.9	[94,479]
10^{-100}	0.0	[956]	40.8	[70]	40.3	[234,654]

An Experiment (cont.)

- conclusions:
  - not just what is being minimized that matters, but how it is being minimized
  - loss-minimization view has benefits and is fundamental to understanding AdaBoost
  - but is limited in what it says about generalization
- results are consistent with margins theory



Fundamental Perspectives

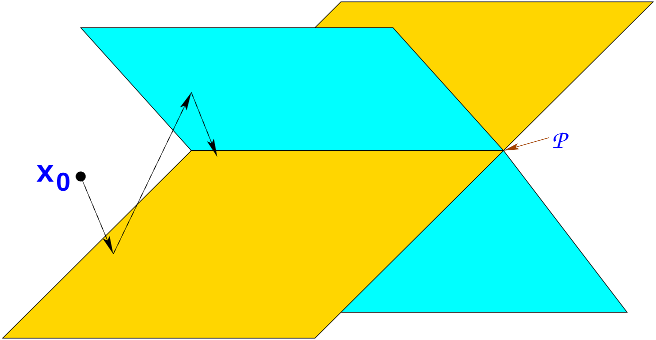
- game theory
- loss minimization
- an information-geometric view

A Dual Information-Geometric Perspective

- loss minimization focuses on function computed by AdaBoost (i.e., weights on weak classifiers)
- dual view: instead focus on distributions D_t (i.e., weights on examples)
- dual perspective combines geometry and information theory
- exposes underlying mathematical structure
- basis for proving convergence

An Iterative-Projection Algorithm

- say want to find point closest to $\mathbf{x}_0$ in set $\mathcal{P} = \{$ intersection of N hyperplanes $\}$
- algorithm: [Bregman; Censor & Zenios]
  - start at $\mathbf{x}_0$
  - repeat: pick a hyperplane and project onto it



- if $\mathcal{P} \neq \emptyset$, under general conditions, will converge correctly

AdaBoost is an Iterative-Projection Algorithm

[Kivinen & Warmuth]

- points = distributions D_t over training examples
- distance = relative entropy:

$$\mathrm{RE}(P \parallel Q) = \sum_i P(i) \ln \left(\frac{P(i)}{Q(i)} \right)$$

- reference point $\mathbf{x}_0 =$ uniform distribution
- hyperplanes defined by all possible weak classifiers g_j:

$$\sum_i D(i) y_i g_j(x_i) = 0 \Leftrightarrow \Pr_{i \sim D} [g_j(x_i) \neq y_i] = \tfrac{1}{2}$$

  - intuition: looking for “hardest” distribution

AdaBoost as Iterative Projection (cont.)

- algorithm:
  - start at $D_1 =$ uniform
  - for $t = 1, 2, \ldots$:
    - pick hyperplane/weak classifier $h_t \leftrightarrow g_j$
    - $D_{t+1} =$ (entropy) projection of D_t onto hyperplane

$$= \arg \min_{D : \sum_i D(i) y_i g_j(x_i) = 0} \mathrm{RE}\left(D \parallel D_t\right)$$

- claim: equivalent to AdaBoost
- further: choosing h_t with minimum error $\equiv$ choosing farthest hyperplane

Boosting as Maximum Entropy

- corresponding optimization problem:

$$\min_{D \in \mathcal{P}} \mathrm{RE}\left(D \;\|\; \textsf{uniform}\right) \leftrightarrow \max_{D \in \mathcal{P}} \textsf{entropy}(D)$$

- where

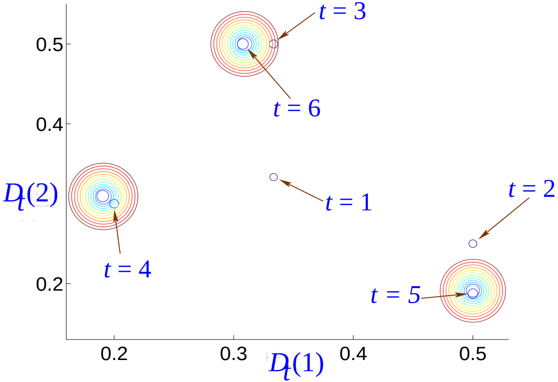
$$\begin{aligned} \mathcal{P} &= \text{feasible set} \\ &= \left\{ D : \sum_i D(i) y_i g_j(x_i) = 0 \;\; \forall j \right\} \end{aligned}$$

- $\mathcal{P} \neq \emptyset \Leftrightarrow$ weak learning assumption does not hold
  - in this case, $D_t \rightarrow$ (unique) solution
- if weak learning assumption does hold then
  - $\mathcal{P} = \emptyset$
  - D_t can never converge
  - dynamics are fascinating but unclear in this case

Visualizing Dynamics

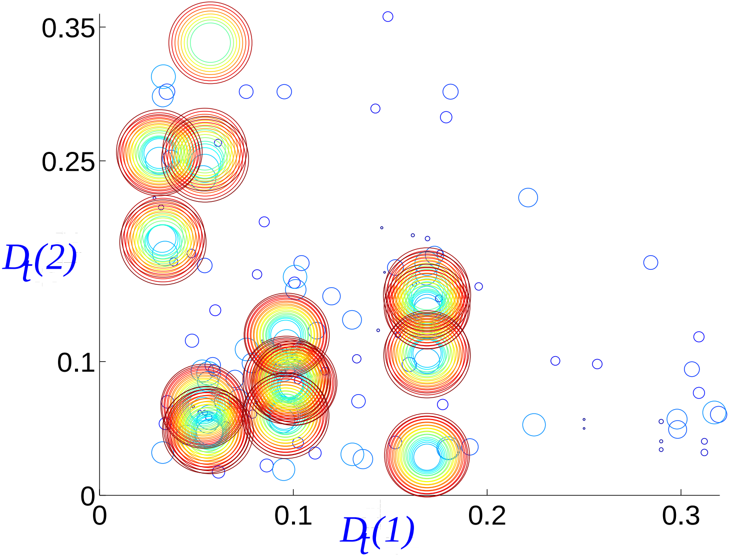
[with Rudin & Daubechies]

- plot one circle for each round t:
    - center at $(D_t(1), D_t(2))$
    - radius $\propto t$ (color also varies with t)

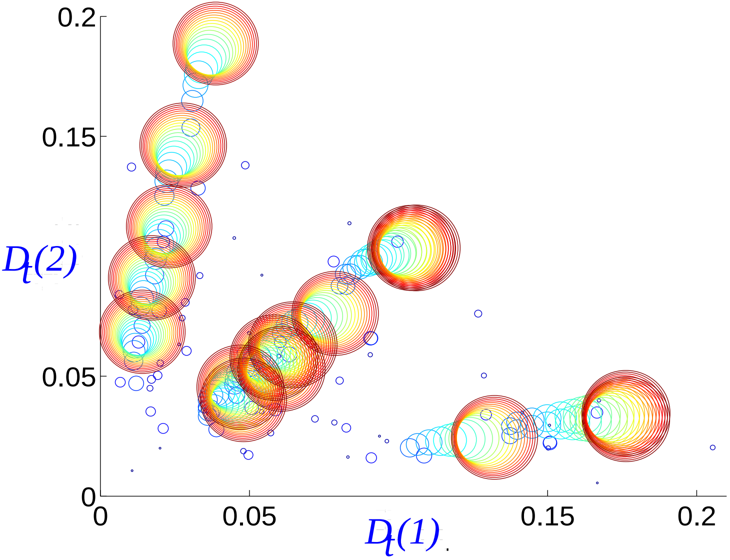


- in all cases examined, appears to converge eventually to cycle
    - open if always true

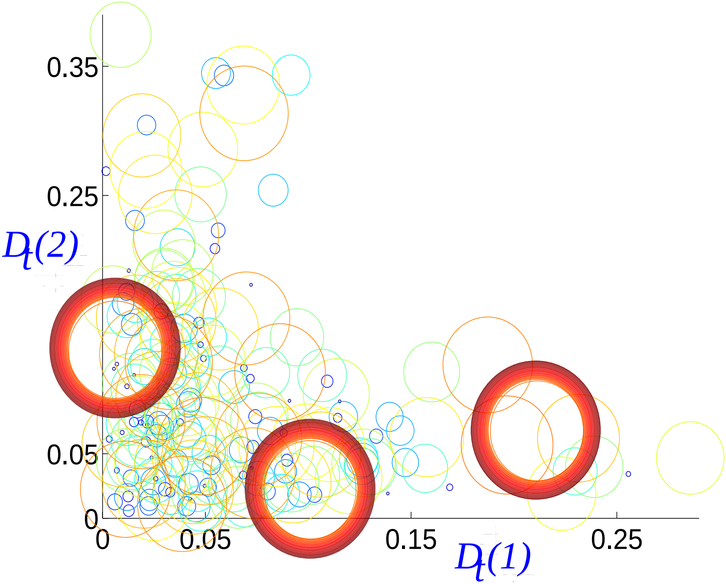
More Examples



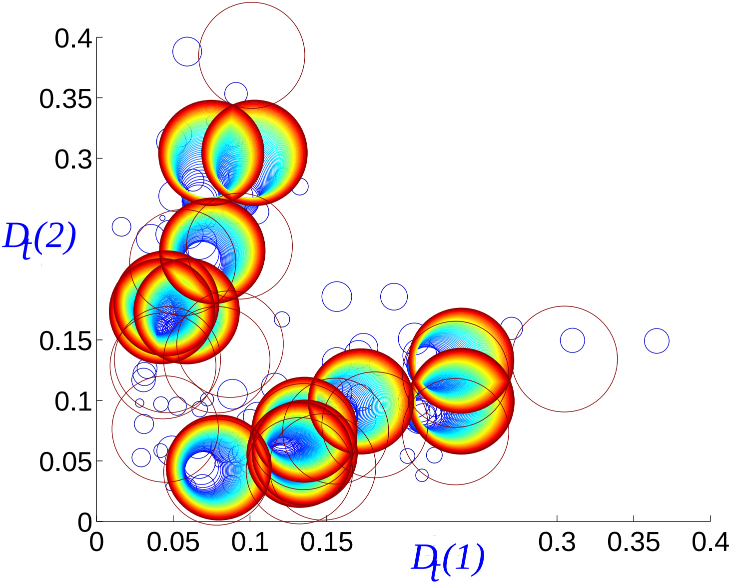
More Examples



More Examples



More Examples



Unifying the Two Cases

[with Collins & Singer]

- two distinct cases:
  - weak learning assumption holds
    - $\mathcal{P} = \emptyset$
    - dynamics unclear
  - weak learning assumption does not hold
    - $\mathcal{P} \neq \emptyset$
    - can prove convergence of D_t's
- to unify: work instead with unnormalized versions of D_t's
  - standard AdaBoost: $D_{t+1}(i) = \dfrac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{\text{normalization}}$
  - instead:

$$d_{t+1}(i) = d_t(i)\exp(-\alpha_t y_i h_t(x_i))$$

$$D_{t+1}(i) = \frac{d_{t+1}(i)}{\text{normalization}}$$

  - algorithm is unchanged

Reformulating AdaBoost as Iterative Projection

- points = nonnegative vectors $\mathbf{d}_t$
- distance = unnormalized relative entropy:

$$\mathrm{RE}(\mathbf{p} \parallel \mathbf{q}) = \sum_i \left[p(i) \ln \left(\frac{p(i)}{q(i)} \right) + q(i) - p(i) \right]$$

- reference point $\mathbf{x}_0 = \mathbf{1}$ (all 1's vector)
- hyperplanes defined by weak classifiers g_j:

$$\sum_i d(i) y_i g_j(x_i) = 0$$

- resulting iterative-projection algorithm is again equivalent to AdaBoost

Reformulated Optimization Problem

- optimization problem:

$$\min_{\mathbf{d}\in\mathcal{P}} \mathrm{RE}\left(\mathbf{d} \;\|\; \mathbf{1}\right)$$

- where

$$\mathcal{P} = \left\{\mathbf{d} : \sum_i d(i) y_i g_j(x_i) = 0 \;\; \forall j\right\}$$

- note: feasible set $\mathcal{P}$ never empty (since $\mathbf{0} \in \mathcal{P}$)

Exponential Loss as Entropy Optimization

- all vectors $\mathbf{d}_t$ created by AdaBoost have form:

$$d(i) = \exp\left(-y_i \sum_j \lambda_j g_j(x_i)\right)$$

- let $\mathcal{Q} = \{$ all vectors $\mathbf{d}$ of this form $\}$
- can rewrite exponential loss:

$$\begin{aligned}\inf_{\boldsymbol{\lambda}} \sum_i \exp\left(-y_i \sum_j \lambda_j g_j(x_i)\right) &= \inf_{\mathbf{d} \in \mathcal{Q}} \sum_i d(i) \\ &= \min_{\mathbf{d} \in \overline{\mathcal{Q}}} \sum_i d(i) \\ &= \min_{\mathbf{d} \in \overline{\mathcal{Q}}} \mathrm{RE}\left(\mathbf{0} \;\|\; \mathbf{d}\right)\end{aligned}$$

- $\overline{\mathcal{Q}}$ = closure of $\mathcal{Q}$

Duality

[Della Pietra, Della Pietra & Lafferty]

- presented two optimization problems:
  - $\min_{\mathbf{d} \in \mathcal{P}} \mathrm{RE}\left(\mathbf{d} \;\|\; \mathbf{1}\right)$
  - $\min_{\mathbf{d} \in \overline{\mathcal{Q}}} \mathrm{RE}\left(\mathbf{0} \;\|\; \mathbf{d}\right)$
- which is AdaBoost solving? Both!
- problems have same solution
- moreover: solution given by unique point in $\mathcal{P} \cap \overline{\mathcal{Q}}$
- problems are convex duals of each other

Convergence of AdaBoost

- can use to prove AdaBoost converges to common solution of both problems:
    - can argue that $\mathbf{d}^* = \lim \mathbf{d}_t$ is in $\mathcal{P}$
    - vectors $\mathbf{d}_t$ are in $\mathcal{Q}$ always $\Rightarrow \mathbf{d}^* \in \overline{\mathcal{Q}}$

    $\therefore$ $\mathbf{d}^* \in \mathcal{P} \cap \overline{\mathcal{Q}}$

    $\therefore$ $\mathbf{d}^*$ solves both optimization problems
- so:
    - AdaBoost minimizes exponential loss
    - exactly characterizes limit of unnormalized "distributions"
    - likewise for normalized distributions when weak learning assumption does not hold
- also, provides additional link to logistic regression
    - only need slight change in optimization problem

[with Collins & Singer; Lebannon & Lafferty]

Practical Extensions

- multiclass classification
- ranking problems
- confidence-rated predictions

Practical Extensions

- multiclass classification
- ranking problems
- confidence-rated predictions

Multiclass Problems

[with Freund]

- say $y \in Y = \{1, \ldots, k\}$
- direct approach (AdaBoost.M1):

$$h_t : X \to Y$$

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

$$H_{\mathrm{final}}(x) = \arg\max_{y \in Y} \sum_{t : h_t(x) = y} \alpha_t$$

- can prove same bound on error if $\forall t : \epsilon_t \leq 1/2$
    - in practice, not usually a problem for "strong" weak learners (e.g., C4.5)
    - significant problem for "weak" weak learners (e.g., decision stumps)
- instead, reduce to binary

Reducing Multiclass to Binary

[with Singer]

- say possible labels are $\{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}\}$
- each training example replaced by five $\{-1, +1\}$-labeled examples:

$$x \quad , \quad \mathrm{c} \rightarrow \begin{cases} (x, \mathrm{a}) & , \quad -1 \\ (x, \mathrm{b}) & , \quad -1 \\ (x, \mathrm{c}) & , \quad +1 \\ (x, \mathrm{d}) & , \quad -1 \\ (x, \mathrm{e}) & , \quad -1 \end{cases}$$

- predict with label receiving most (weighted) votes

AdaBoost.MH

- can prove:

$$\text{training error}(H_{\text{final}}) \leq \frac{k}{2} \cdot \prod Z_t$$

  - reflects fact that small number of errors in binary predictors can cause overall prediction to be incorrect
- extends immediately to multi-label case
  (more than one correct label per example)

Using Output Codes

[with Allwein & Singer][Dietterich & Bakiri]

- alternative: choose "codeword" for each label

	π_1	π_2	π_3	π_4
a	$-$	$+$	$-$	$+$
b	$-$	$+$	$+$	$-$
c	$+$	$-$	$-$	$+$
d	$+$	$-$	$+$	$+$
e	$-$	$+$	$-$	$-$

- each training example mapped to one example per column

$$x \quad , \quad \mathrm{c} \rightarrow \left\{ \begin{array}{lcl} (x, \pi_1) & , & +1 \\ (x, \pi_2) & , & -1 \\ (x, \pi_3) & , & -1 \\ (x, \pi_4) & , & +1 \end{array} \right.$$

- to classify new example x:
  - evaluate classifier on $(x, \pi_1), \ldots, (x, \pi_4)$
  - choose label "most consistent" with results

Output Codes (cont.)

- training error bounds independent of # of classes
- overall prediction robust to large number of errors in binary predictors
- but: binary problems may be harder

Practical Extensions

- multiclass classification
- ranking problems
- confidence-rated predictions

Ranking Problems

[with Freund, Iyer & Singer]

- other problems can also be handled by reducing to binary
- e.g.: want to learn to rank objects (say, movies) from examples
- can reduce to multiple binary questions of form:
  "is or is not object A preferred to object B?"
- now apply (binary) AdaBoost ⇒ "RankBoost"

Application: Finding Cancer Genes

[Agarwal & Sengupta]

- examples are genes (described by microarray vectors)
- want to rank genes from most to least relevant to leukemia
- data sizes:
  - 7129 genes total
  - 10 known relevant
  - 157 known irrelevant

Top-Ranked Cancer Genes

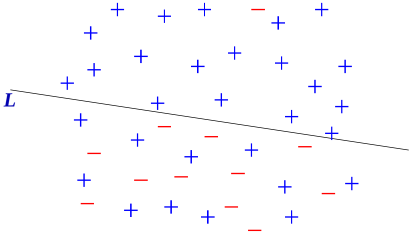
	Gene	**Relevance Summary**
1.	KIAA0220	□
2.	G-gamma globin	♦
3.	Delta-globin	♦
4.	Brain-expressed HHCPA78 homolog	□
5.	Myeloperoxidase	♦
6.	Probable protein disulfide isomerase ER-60 precursor	□
7.	NPM1 Nucleophosmin	♦
8.	CD34	♦
9.	Elongation factor-1-beta	×
10.	CD24	♦

■ = known therapeutic target
□ = potential therapeutic target
♦ = known marker
◊ = potential marker
× = no link found

Practical Extensions

- multiclass classification
- ranking problems
- confidence-rated predictions

“Hard” Predictions Can Slow Learning



- ideally, want weak classifier that says:

$$h(x) = \begin{cases} +1 & \text{if } x \text{ above } L \\ \textit{“don't know”} & \text{else} \end{cases}$$

- problem: cannot express using “hard” predictions
- if must predict ±1 below L, will introduce many “bad” predictions
  - need to “clean up” on later rounds
- dramatically increases time to convergence

Confidence-Rated Predictions

[with Singer]

- useful to allow weak classifiers to assign confidences to predictions
- formally, allow $h_t : X \to \mathbb{R}$

$$\begin{aligned} \operatorname{sign}(h_t(x)) &= \text{prediction} \\ |h_t(x)| &= \text{“confidence”} \end{aligned}$$

- use identical update:

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

  and identical rule for combining weak classifiers

- question: how to choose α_t and h_t on each round

Confidence-Rated Predictions (cont.)

- saw earlier:

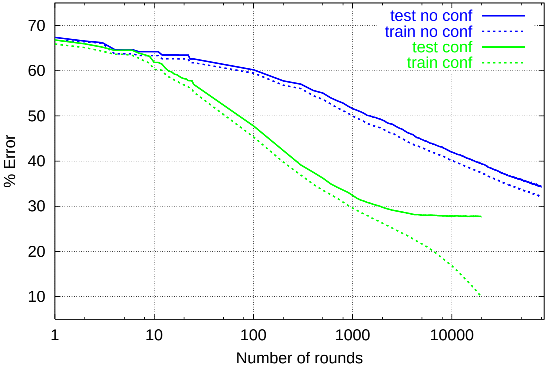
$$\text{training error}(H_{\text{final}}) \le \prod_t Z_t = \frac{1}{m} \sum_i \exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right)$$

- therefore, on each round t, should choose $\alpha_t h_t$ to minimize:

$$Z_t = \sum_i D_t(i) \exp(-\alpha_t \ y_i \ h_t(x_i))$$

- in many cases (e.g., decision stumps), best confidence-rated weak classifier has simple form that can be found efficiently

Confidence-Rated Predictions Help a Lot



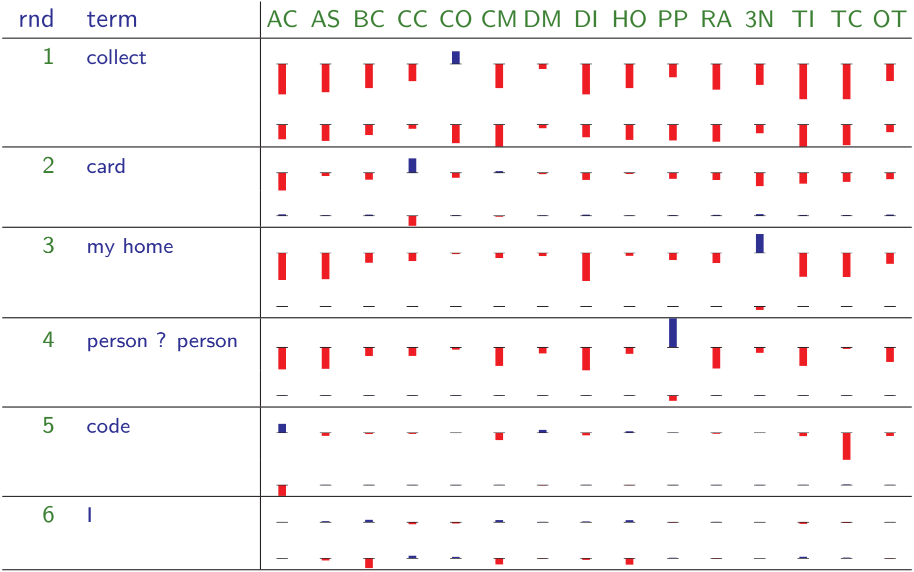
	round first reached		
% error	conf.	no conf.	speedup
40	268	16,938	63.2
35	598	65,292	109.2
30	1,888	>80,000	–

Application: Boosting for Text Categorization

[with Singer]

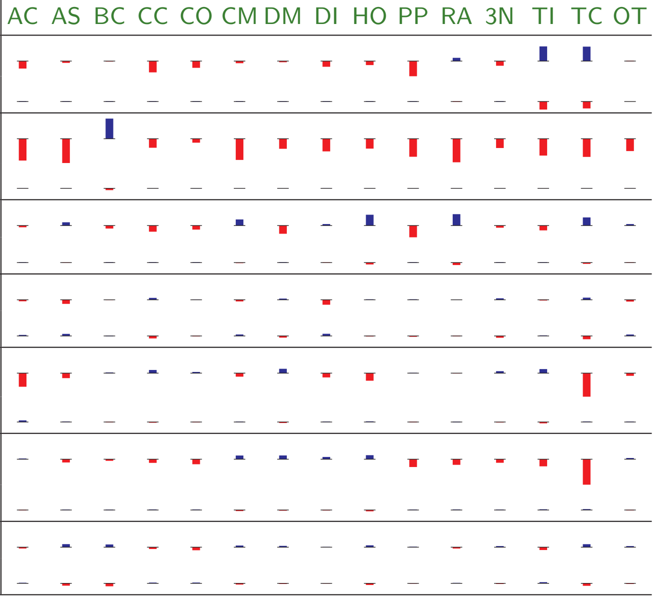
- weak classifiers: very simple weak classifiers that test on simple patterns, namely, (sparse) *n*-grams
    - find parameter α_t and rule h_t of given form which minimize Z_t
    - use efficiently implemented exhaustive search
- "How may I help you" data:
    - 7844 training examples
    - 1000 test examples
    - categories: AreaCode, AttService, BillingCredit, CallingCard, Collect, Competitor, DialForMe, Directory, HowToDial, PersonToPerson, Rate, ThirdNumber, Time, TimeCharge, Other.

Weak Classifiers



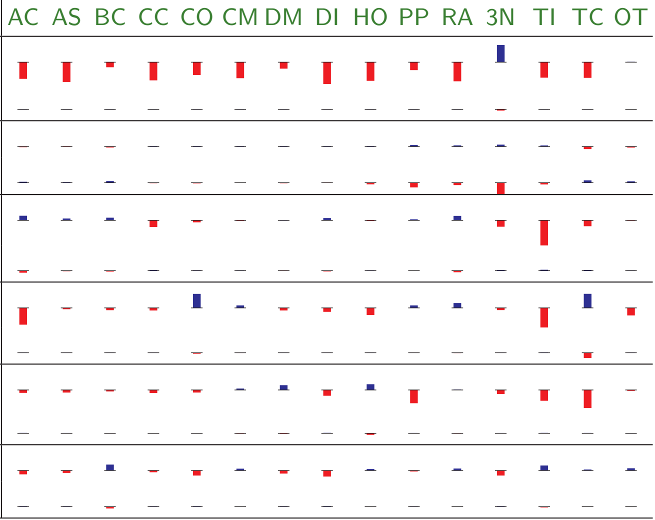
More Weak Classifiers

rnd	term
7	time
8	wrong number
9	how
10	call
11	seven
12	trying to
13	and



More Weak Classifiers

rnd	term
14	third
15	to
16	for
17	charges
18	dial
19	just



Finding Outliers

examples with most weight are often outliers (mislabeled and/or ambiguous)

- I'm trying to make a credit card call (Collect)
- hello (Rate)
- yes I'd like to make a long distance collect call please (CallingCard)
- calling card please (Collect)
- yeah I'd like to use my calling card number (Collect)
- can I get a collect call (CallingCard)
- yes I would like to make a long distant telephone call and have the charges billed to another number (CallingCard DialForMe)
- yeah I can not stand it this morning I did oversea call is so bad (BillingCredit)
- yeah special offers going on for long distance (AttService Rate)
- mister allen please william allen (PersonToPerson)
- yes ma'am I I'm trying to make a long distance call to a non dialable point in san miguel philippines (AttService Other)

Advanced Topics

- optimal accuracy
- optimal efficiency
- boosting in continuous time

Advanced Topics

- optimal accuracy
- optimal efficiency
- boosting in continuous time

Optimal Accuracy

[Bartlett & Traskin]

- usually, impossible to get perfect accuracy due to intrinsic noise or uncertainty
- Bayes optimal error = best possible error of any classifier
  - usually > 0
- can prove AdaBoost's classifier converges to Bayes optimal if:
  - enough data
  - run for many (but not too many) rounds
  - weak classifiers "sufficiently rich"
- "universally consistent"
- related results: [Jiang], [Lugosi & Vayatis], [Zhang & Yu], ...
- means:
  - AdaBoost can (theoretically) learn "optimally" even in noisy settings
  - but: does not explain why works when run for very many rounds

Boosting and Noise

[Long & Servedio]

- can construct data source that "breaks" AdaBoost with even tiny amount of noise (say, 1%)
  - Bayes optimal error $= 1\%$ (obtainable by classifier of same form as AdaBoost)
  - AdaBoost provably has error $\geq 50\%$
- holds even if:
  - given unlimited training data
  - use any method for minimizing exponential loss (holds for most other convex losses as well)
- shows:
  - consistency result can fail badly if weak classifiers "not rich enough"
  - boosting susceptible to noise
- on "real-world" datasets, AdaBoost often works anyway
- various theoretical algorithms proposed for handling noise (e.g., [Kalai & Servedio], [Long & Servedio])

Advanced Topics

- optimal accuracy
- optimal efficiency
- boosting in continuous time

Optimal Efficiency

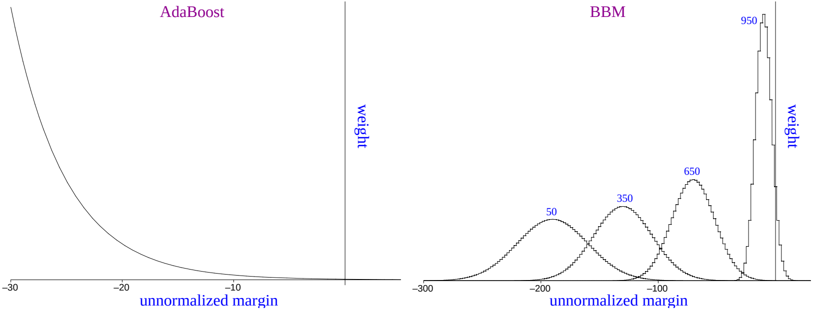
[Freund]

- for AdaBoost, saw: training error $\leq e^{-2\gamma^2 T}$
- is AdaBoost most efficient boosting algorithm?
  no!
- given T rounds and γ-weak learning assumption, boost-by-majority (BBM) algorithm is provably exactly best possible:

$$\text{training error} \leq \sum_{j=0}^{\lfloor T/2 \rfloor} \binom{T}{j} \left(\tfrac{1}{2}+\gamma\right)^{j} \left(\tfrac{1}{2}-\gamma\right)^{T-j}$$

  (probability of $\leq T/2$ heads in T coin flips if probability of heads $= \frac{1}{2}+\gamma$)
- AdaBoost's training error is like Chernoff approximation of BBM's

Weighting Functions: AdaBoost versus BBM



- both put more weight on harder examples, but BBM "gives up" on very hardest examples
  - may make more robust to noise
- problem: BBM not adaptive
  - need to know γ and T a priori

Advanced Topics

- optimal accuracy
- optimal efficiency
- boosting in continuous time

Boosting in Continuous Time

[Freund]

- idea: let γ get very small so that γ-weak learning assumption eventually satisfied
- need to make T correspondingly large
- if scale time to begin at $\tau = 0$ and end at $\tau = 1$, then each boosting round takes time $1/T$
- in limit $T \to \infty$, boosting is happening in continuous time

BrownBoost

- algorithm has sensible limit called "BrownBoost" (due to connection to Brownian motion)
- harder to implement, but potentially more resistant to noise and outliers, e.g.:

dataset	noise	AdaBoost	BrownBoost
letter	0%	3.7	4.2
	10%	10.8	7.0
	20%	15.7	10.5
satimage	0%	4.9	5.2
	10%	12.1	6.2
	20%	21.3	7.4

Conclusions

- from different perspectives, AdaBoost can be interpreted as:
  - a method for boosting the accuracy of a weak learner
  - a procedure for maximizing margins
  - an algorithm for playing repeated games
  - a numerical method for minimizing exponential loss
  - an iterative-projection algorithm based on an information-theoretic geometry
- none is entirely satisfactory by itself, but each useful in its own way
- taken together, create rich theoretical understanding
  - connect boosting to other learning problems and techniques
  - provide foundation for versatile set of methods with many extensions, variations and applications

References

Coming soon:

- Robert E. Schapire and Yoav Freund.
  Boosting: Foundations and Algorithms.
  MIT Press, 2012.

Survey articles:

- Ron Meir and Gunnar Rätsch.
  An Introduction to Boosting and Leveraging.
  In *Advanced Lectures on Machine Learning (LNAI2600)*, 2003.
  http://www.boosting.org/papers/MeiRae03.pdf
- Robert E. Schapire.
  The boosting approach to machine learning: An overview.
  In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.
  http://www.cs.princeton.edu/∼schapire/boost.html