Theory and Applications of Boosting

Rob Schapire
Princeton University

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)

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

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)
  - given sufficient data, a boosting algorithm can provably construct single classifier with very high accuracy, say, 99%

Outline of Tutorial

- brief background
- basic algorithm and core theory
- other ways of understanding boosting
- experiments, applications and extensions

Brief Background

Strong and Weak Learnability

- boosting's roots are in "PAC" (Valiant) learning model
- 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?

Early 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

AdaBoost

- [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 et al. '00]
[Onoda, Rätsch & Müller '00]
[Tieu & Viola '00]
[Walker, Rambow & Rogati '01]
[Rochery, Schapire, Rahim & Gupta '01]
[Merler, Furlanello, Larcher & Sboner '01]
[Di Fabbrizio, Dutton, Gupta et al. '02]
[Qu, Adam, Yasui et al. '02]
[Tur, Schapire & Hakkani-Tür '03]
[Viola & Jones '04]
[Middendorf, Kundaje, Wiggins et al. '04]
⋮

- 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]
[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 et al. '00]
[Allwein, Schapire & Singer '00]
[Friedman '01]
[Koltchinskii, Panchenko & Lozano '01]
[Collins, Schapire & Singer '02]
[Demiriz, Bennett & Shawe-Taylor '02]
[Lebanon & Lafferty '02]
[Wyner '02]
[Rudin, Daubechies & Schapire '03]
[Jiang '04]
[Lugosi & Vayatis '04]
[Zhang '04]
⋮

Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error based on margins theory

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 $x_i \in X$

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:

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

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:
  - $D_1(i) = 1/m$

AdaBoost

[with Freund]

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

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

AdaBoost

[with Freund]

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

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

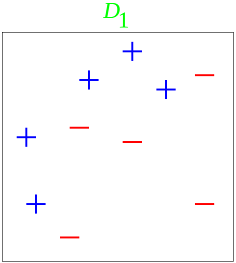
$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where Z_t = normalization constant

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

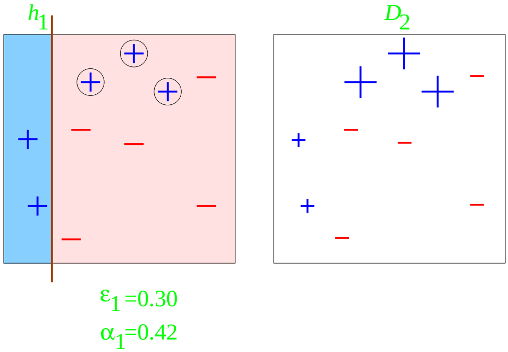
- final classifier:
  - $H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$

Toy Example

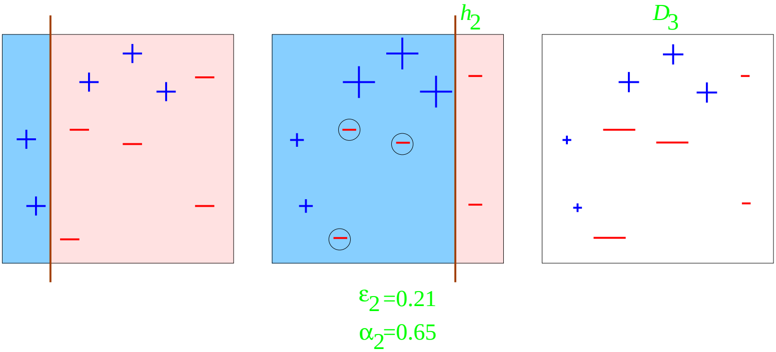


weak classifiers = vertical or horizontal half-planes

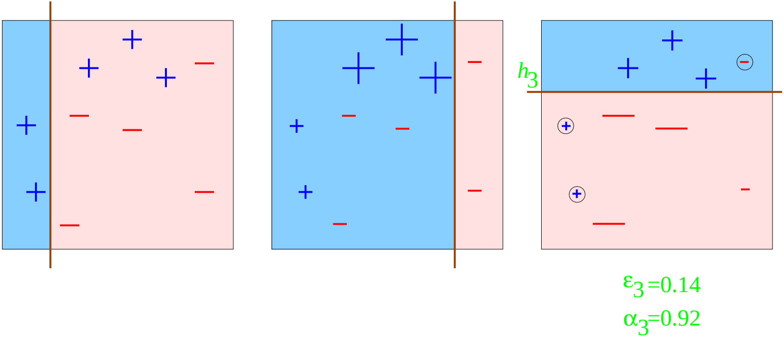
Round 1



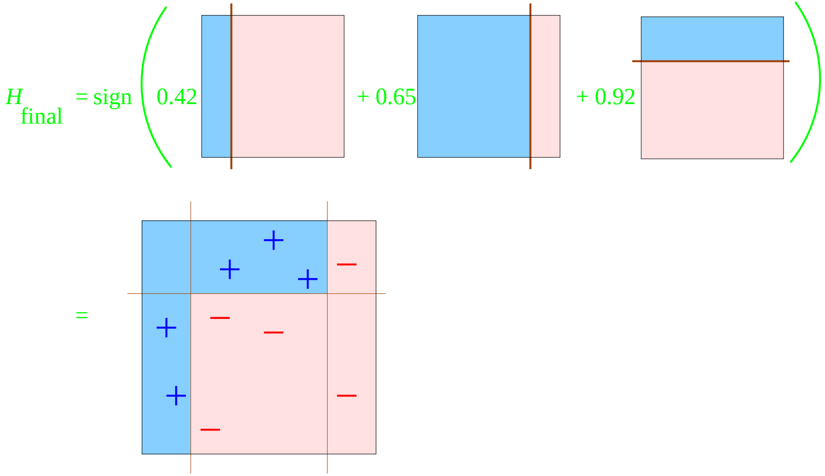
Round 2



Round 3



Final Classifier



Analyzing the training error

- Theorem:
  - write ϵ_t as $1/2 - \gamma_t$

Analyzing the training error

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

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

Analyzing the training error

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

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

- so: if $\forall t: \ \gamma_t \ge \gamma > 0$
  then $\text{training error}(H_{\text{final}}) \le e^{-2\gamma^2 T}$
- 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:

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

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

Proof (cont.)

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

Proof (cont.)

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

$$\text{training error}(H_{\text{final}}) = \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i \ne H_{\text{final}}(x_i) \\ 0 & \text{else} \end{cases}$$

Proof (cont.)

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

$$\text{training error}(H_{\text{final}}) = \frac{1}{m}\sum_i \begin{cases} 1 & \text{if } y_i \neq H_{\text{final}}(x_i) \\ 0 & \text{else} \end{cases}$$

$$= \frac{1}{m}\sum_i \begin{cases} 1 & \text{if } y_i f(x_i) \leq 0 \\ 0 & \text{else} \end{cases}$$

Proof (cont.)

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

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

Proof (cont.)

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

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

Proof (cont.)

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

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

Proof (cont.)

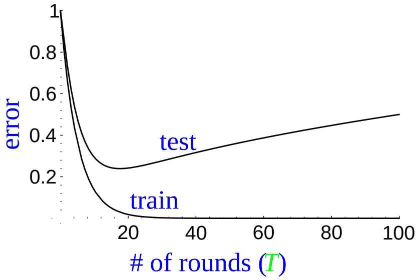
- *Step 3*: $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$

Proof (cont.)

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

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

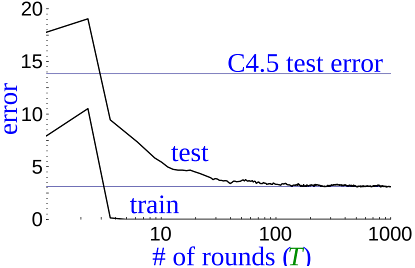
How Will Test Error Behave? (A First Guess)



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

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

A Better Story: The Margins Explanation

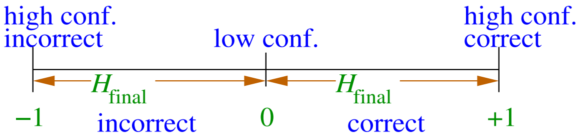
[with Freund, Bartlett & Lee]

- 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

A Better Story: The Margins Explanation

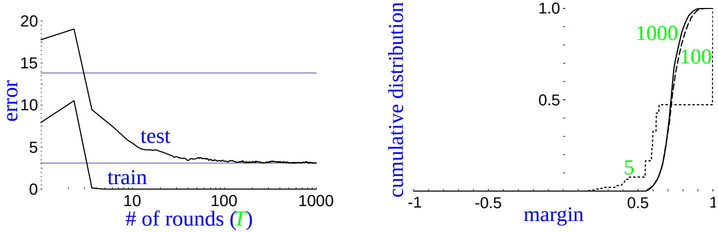
[with Freund, Bartlett & Lee]

- 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
  = (fraction voting correctly) − (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)

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)

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)

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)
  - proof idea: similar to training error proof

Theoretical Evidence: Analyzing Boosting Using Margins

- Theorem: large margins ⇒ 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)
  - 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 (error of h_t on D_t) $< 1/2 - \theta$ ($\forall t$)
  - so: if weak learning assumption holds, then all examples will quickly have "large" margins

Other Ways of Understanding AdaBoost

- game theory
- loss minimization
- estimating conditional probabilities

Game Theory

- 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 chooses row i
- column player chooses column j (simultaneously)
- row player's goal: minimize loss $\mathbf{M}(i, j)$

Game Theory

- 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 chooses row i
- column player chooses column j (simultaneously)
- row player's goal: minimize loss $\mathbf{M}(i,j)$
- usually allow randomized play:
  - players choose distributions $\mathbf{P}$ and $\mathbf{Q}$ over rows and columns
- learner's (expected) loss

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

The Minmax Theorem

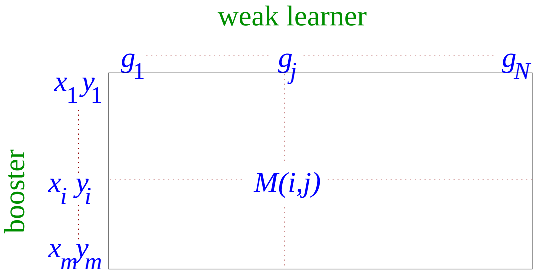
- von Neumann's minmax theorem:

$$\begin{aligned} \min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P},\mathbf{Q}) &= \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\mathbf{Q}) \\ &= v \\ &= \text{“value” of game } \mathbf{M} \end{aligned}$$

- in words:
  - $v = \min \max$ means:
    - row player has strategy $\mathbf{P}^*$ such that $\forall$ column strategy $\mathbf{Q}$ loss $\mathbf{M}(\mathbf{P}^*,\mathbf{Q}) \le v$
  - $v = \max \min$ means:
    - this is optimal in sense that column player has strategy $\mathbf{Q}^*$ such that $\forall$ row strategy $\mathbf{P}$ loss $\mathbf{M}(\mathbf{P},\mathbf{Q}^*) \ge v$

The Boosting Game

- let $\{g_1, \ldots, g_N\}$ = space of all weak classifiers
- row player $\leftrightarrow$ booster
- column player $\leftrightarrow$ weak learner
- matrix **M**:
  - row $\leftrightarrow$ example (x_i, y_i)
  - column $\leftrightarrow$ weak classifier g_j
  - $M(i,j) = \begin{cases} 1 & \text{if } y_i = g_j(x_i) \\ 0 & \text{else} \end{cases}$



Boosting and the Minmax Theorem

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

AdaBoost and Game Theory

[with Freund]

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

AdaBoost and Exponential Loss

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

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

where $f(x) = \sum_t \alpha_t h_t(x)$

- on each round, AdaBoost greedily chooses α_t and h_t to minimize loss

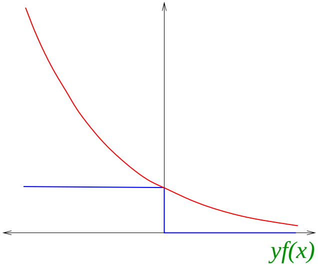
AdaBoost and Exponential Loss

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

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

where $f(x) = \sum_t \alpha_t h_t(x)$

- on each round, AdaBoost greedily chooses α_t and h_t to minimize loss
- exponential loss is an upper bound on 0-1 (classification) loss
- AdaBoost provably minimizes exponential loss



Coordinate Descent

[Breiman]

- $\{g_1, \ldots, g_N\}$ = space of all weak classifiers
- want to find $\lambda_1, \ldots, \lambda_N$ to minimize

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

Coordinate Descent

[Breiman]

- $\{g_1, \ldots, g_N\}$ = space of all weak classifiers
- 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

[Friedman][Mason et al.]

- want to minimize

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

Functional Gradient Descent

[Friedman][Mason et al.]

- want to minimize

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

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

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

Functional Gradient Descent

[Friedman][Mason et al.]

- want to minimize

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

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

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

- but update restricted in class of weak classifiers

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

Functional Gradient Descent

[Friedman][Mason et al.]

- want to minimize

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

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

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

- but update restricted in class of weak classifiers

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

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

Benefits of Model Fitting View

- immediate generalization to other loss functions
  - 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

Benefits of Model Fitting View

- immediate generalization to other loss functions
  - 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
- caveat: wrong to view AdaBoost as just an algorithm for minimizing exponential loss
  - other algorithms for minimizing same loss will (provably) give very poor performance
  - thus, this loss function cannot explain why AdaBoost "works"

Estimating Conditional Probabilities

[Friedman, Hastie & Tibshirani]

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

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

where x, y random from true distribution

Estimating Conditional Probabilities

[Friedman, Hastie & Tibshirani]

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

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

where x, y random from true distribution

- over all f, minimized when

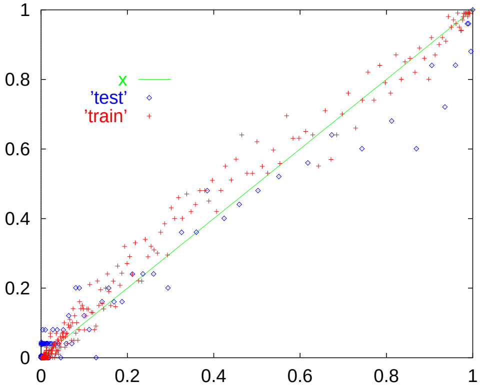
$$f(x) = \frac{1}{2}\cdot\ln\left(\frac{\mathrm{P}\left[y=+1|x\right]}{\mathrm{P}\left[y=-1|x\right]}\right)$$

or

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

- 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 size r
- for each bin, plot a point:
    - x-value: average estimated probability of examples in bin
    - y-value: actual fraction of positive examples in bin

Other Ways to Think about AdaBoost

- dynamical systems
- statistical consistency
- maximum entropy

Experiments, Applications and Extensions

- basic experiments
- multiclass classification
- confidence-rated predictions
- text categorization / spoken-dialogue systems
- incorporating prior knowledge
- active learning
- face detection

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

    $\longrightarrow$ overfitting
  - weak classifiers too weak ($\gamma_t \rightarrow 0$ too quickly)

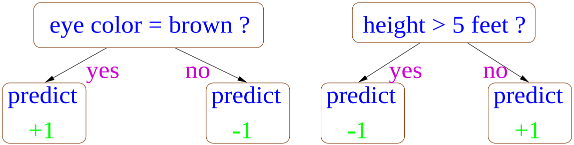
    $\longrightarrow$ underfitting

    $\longrightarrow$ low margins $\longrightarrow$ overfitting
- empirically, AdaBoost seems especially susceptible to uniform noise

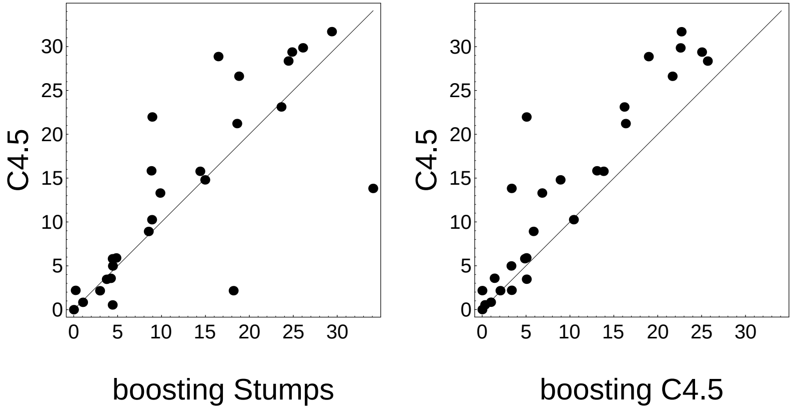
UCI Experiments

[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



Multiclass Problems

[with Freund]

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

$$h_t : X \to Y$$

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

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

Multiclass Problems

[with Freund]

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

$$h_t : X \to Y$$

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

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

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

$$x \quad , \quad c \rightarrow \left\{ \begin{array}{lcl} (x, a) & , & -1 \\ (x, b) & , & -1 \\ (x, c) & , & +1 \\ (x, d) & , & -1 \\ (x, e) & , & -1 \end{array} \right.$$

- 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 "code word" 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

$$x \quad , \quad c \rightarrow \begin{cases} (x, \pi_1) & , \quad +1 \\ (x, \pi_2) & , \quad -1 \\ (x, \pi_3) & , \quad -1 \\ (x, \pi_4) & , \quad +1 \end{cases}$$

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

Output Codes (cont.)

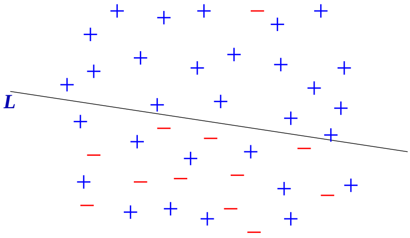
- training error bounds independent of # of classes
- overall prediction robust to large number of errors in binary predictors
- but: binary problems may be harder

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

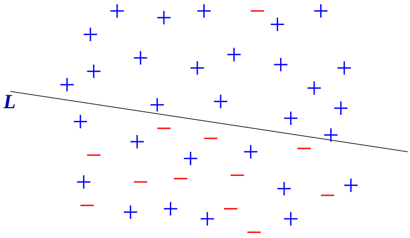
"Hard" Predictions Can Slow Learning



- ideally, want weak classifier that says:

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

"Hard" Predictions Can Slow Learning



- ideally, want weak classifier that says:

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

- 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 $h_t : X \to \mathbb{R}$

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

Confidence-rated Predictions

[with Singer]

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

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

- 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:

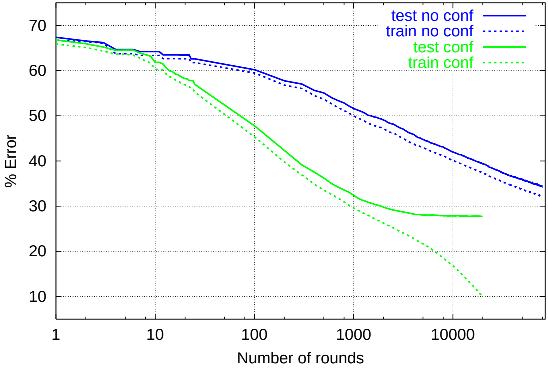
$$\text{training error}(H_{\text{final}}) \leq \prod_t Z_t = \frac{1}{m} \sum_i \exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right)$$

- 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



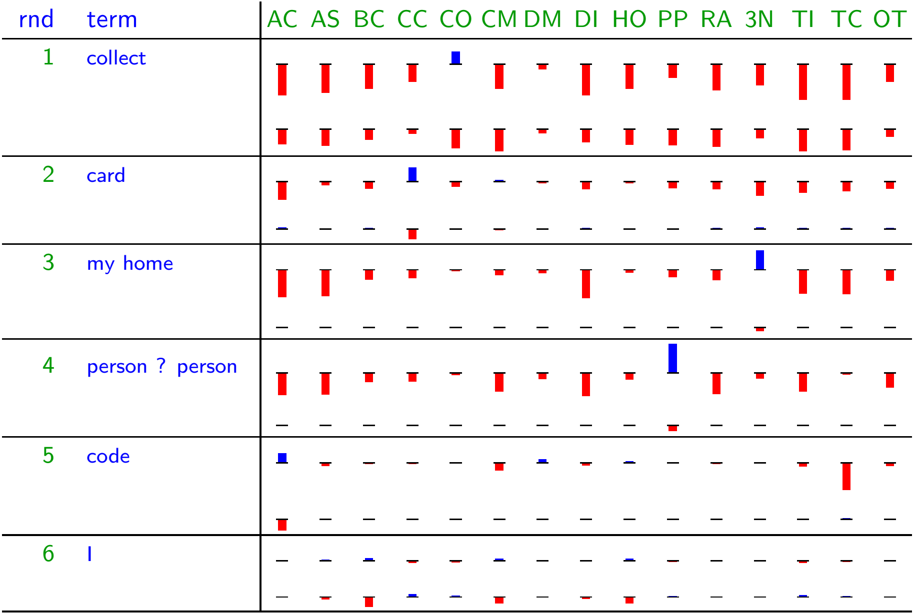
% error	round first reached conf.	round first reached no conf.	speedup
40	268	16,938	63.2
35	598	65,292	109.2
30	1,888	>80,000	–

Application: Boosting for Text Categorization

[with Singer]

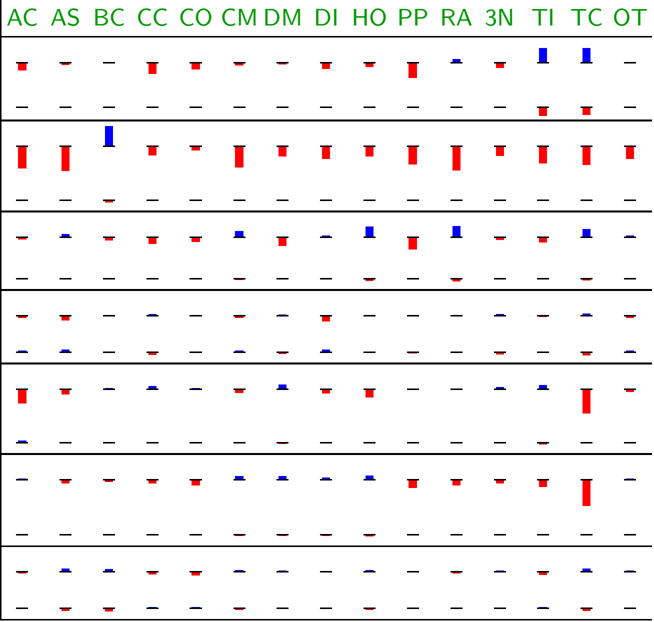
- weak classifiers: very simple weak classifiers that test on simple patterns, namely, (sparse) *n*-grams
  - find parameter α_t and rule h_t of given form which minimize Z_t
  - use efficiently implemented exhaustive search
- "How may I help you" data:
  - 7844 training examples
  - 1000 test examples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

Weak Classifiers



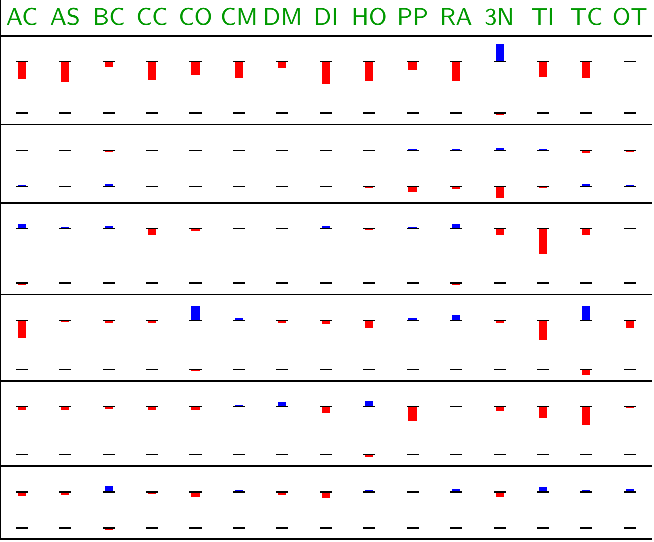
More Weak Classifiers

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)

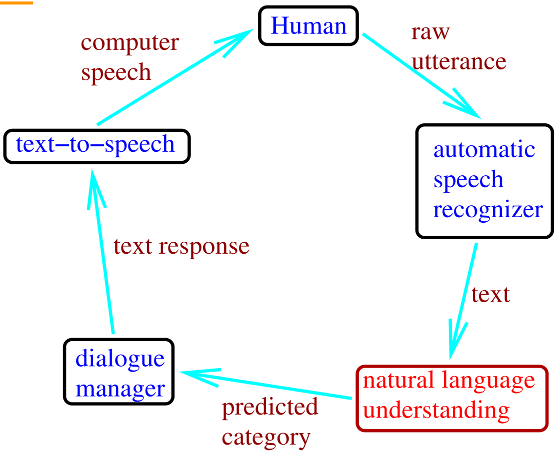
- `I'm trying to make a credit card call` (Collect)
- `hello` (Rate)
- `yes I'd like to make a long distance collect call please` (CallingCard)
- `calling card please` (Collect)
- `yeah I'd like to use my calling card number` (Collect)
- `can I get a collect call` (CallingCard)
- `yes I would like to 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)

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



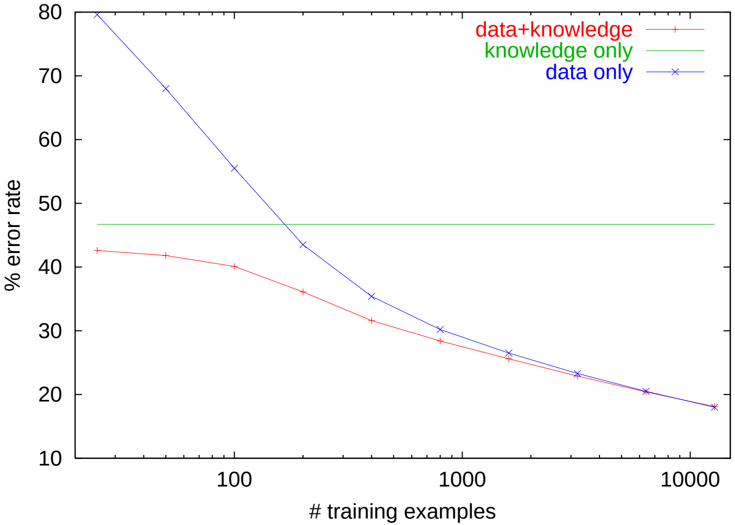
- NLU's job: classify caller utterances into 24 categories (demo, sales rep, pricing info, yes, no, etc.)
- weak classifiers: test for presence of word or phrase

Need for Prior, Human Knowledge

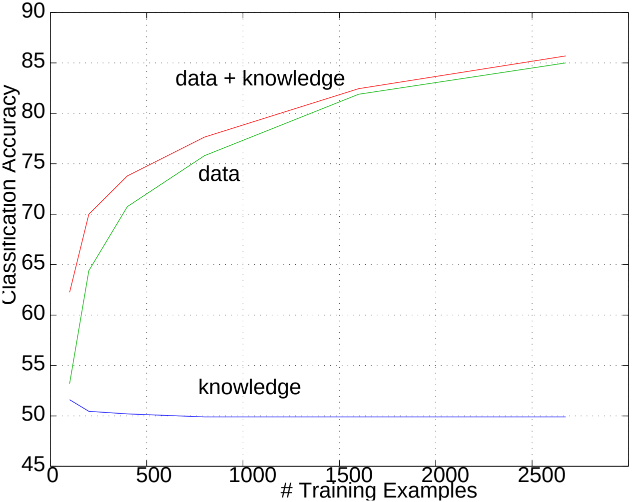
[with Rochery, Rahim & Gupta]

- building NLU: standard text categorization problem
- need lots of data, but for cheap, rapid deployment, can't wait for it
- bootstrapping problem:
  - need labeled data to deploy
  - need to deploy to get labeled data
- idea: use human knowledge to compensate for insufficient data
  - modify loss function to balance fit to data against fit to prior model

Results: AP-Titles



Results: Helpdesk



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

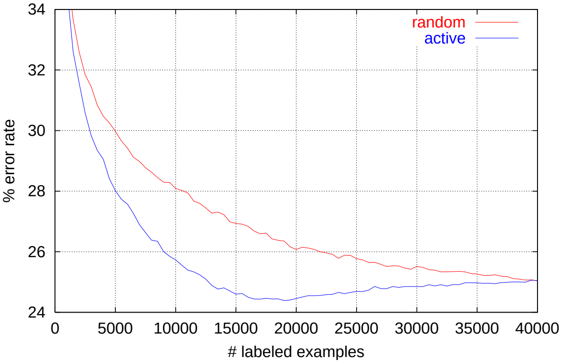
- idea:
  - use selective sampling to choose which examples to label
  - focus on least confident examples [Lewis & Gale]
- for boosting, use (absolute) margin $|f(x)|$ as natural confidence measure

[Abe & Mamitsuka]

Labeling Scheme

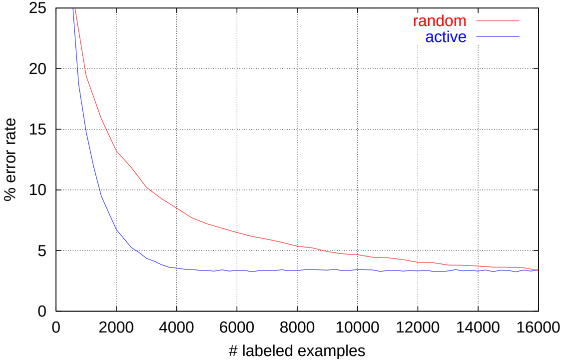
- start with pool of unlabeled examples
- choose (say) 500 examples at random for labeling
- run boosting on all labeled examples
  - get combined classifier f
- pick (say) 250 additional examples from pool for labeling
  - choose examples with minimum $|f(x)|$
- repeat

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



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

Results: Letter

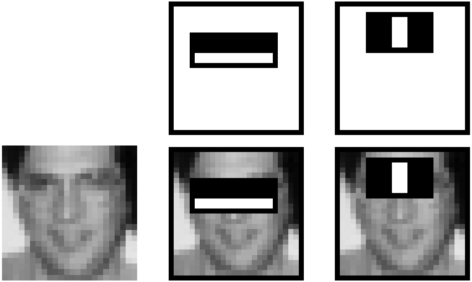


% error	first reached random	first reached active	% label savings
10	3,500	1,500	57
5	9,000	2,750	69
4	13,000	3,500	73

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

Conclusions

- boosting is a practical tool for classification and other learning problems
  - grounded in rich theory
  - performs well experimentally
  - often (but not always!) resistant to overfitting
  - many applications and extensions
- many ways to think about boosting
  - none is entirely satisfactory by itself,
    but each useful in its own way
  - considerable room for further theoretical and
    experimental work

References

- 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