A Tutorial on Boosting

Yoav Freund
Rob Schapire

www.research.att.com/~yoav
www.research.att.com/~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

- select small subset of examples
- derive rough rule of thumb
- examine 2nd set of examples
- derive 2nd rule of thumb
- repeat T times
- questions:
  - how to choose subsets of examples to examine on each round?
  - how to combine all the rules of thumb into single prediction rule?
- boosting = general method of converting rough rules of thumb into highly accurate prediction rule

Tutorial outline

- first half (Rob): behavior on the training set
  - background
  - AdaBoost
  - analyzing training error
  - experiments
  - connection to game theory
  - confidence-rated predictions
  - multiclass problems
  - boosting for text categorization
- second half (Yoav): understanding AdaBoost's generalization performance

The Boosting Problem

- “strong” PAC algorithm
  - for any distribution
  - $\forall \epsilon > 0, \delta > 0$
  - given polynomially many random examples
  - finds hypothesis with error $\leq \epsilon$ with probability $\geq 1 - \delta$
- “weak” PAC algorithm
  - same, but only for $\epsilon \geq \frac{1}{2} - \gamma$
- [Kearns & Valiant ’88]:
  - does weak learnability imply strong learnability?

Early Boosting Algorithms

- [Schapire '89]:
  - first provable boosting algorithm
    - call weak learner three times on three modified distributions
    - get slight boost in accuracy
    - apply recursively
- [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 using AdaBoost:

[Drucker & Cortes '95]	[Schapire & Singer '98]
[Jackson & Craven '96]	[Maclin & Opitz '97]
[Freund & Schapire '96]	[Bauer & Kohavi '97]
[Quinlan '96]	[Schwenk & Bengio '98]
[Breiman '96]	[Dietterich '98]
	⋮

- continuing development of theory and algorithms:

[Schapire, Freund, Bartlett & Lee '97]	[Schapire & Singer '98]
[Breiman '97]	[Mason, Bartlett & Baxter '98]
[Grove & Schuurmans '98]	[Friedman, Hastie & Tibshirani '98]
	⋮

A Formal View 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 hypothesis ("rule of thumb")

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

    with small error ϵ_t on D_t:

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

AdaBoost

[Freund & Schapire]

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

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

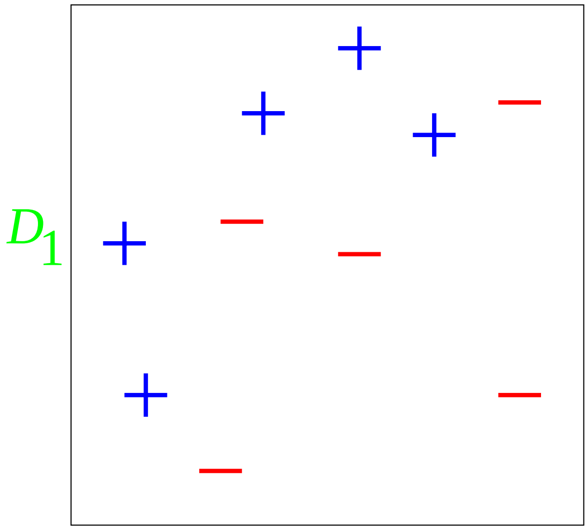
$$= \frac{D_t(i)}{Z_t} \cdot \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where Z_t = normalization constant

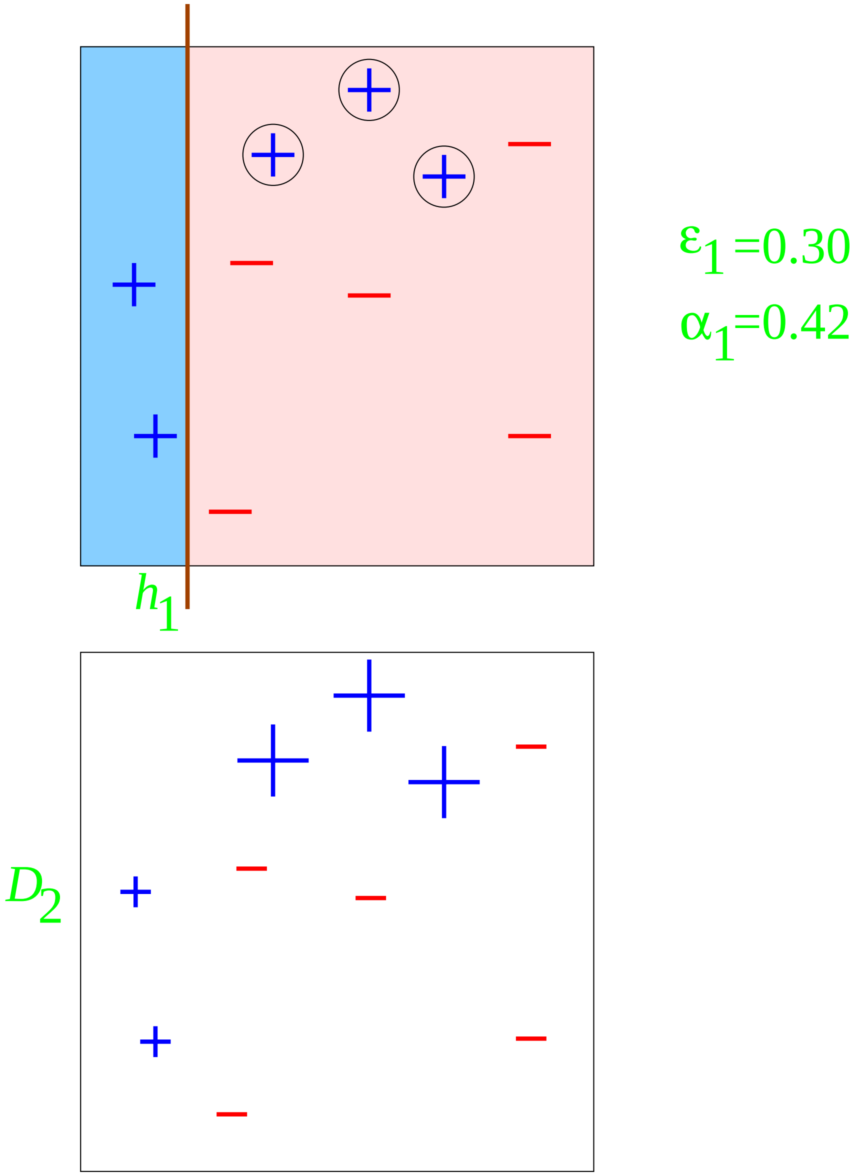
$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

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

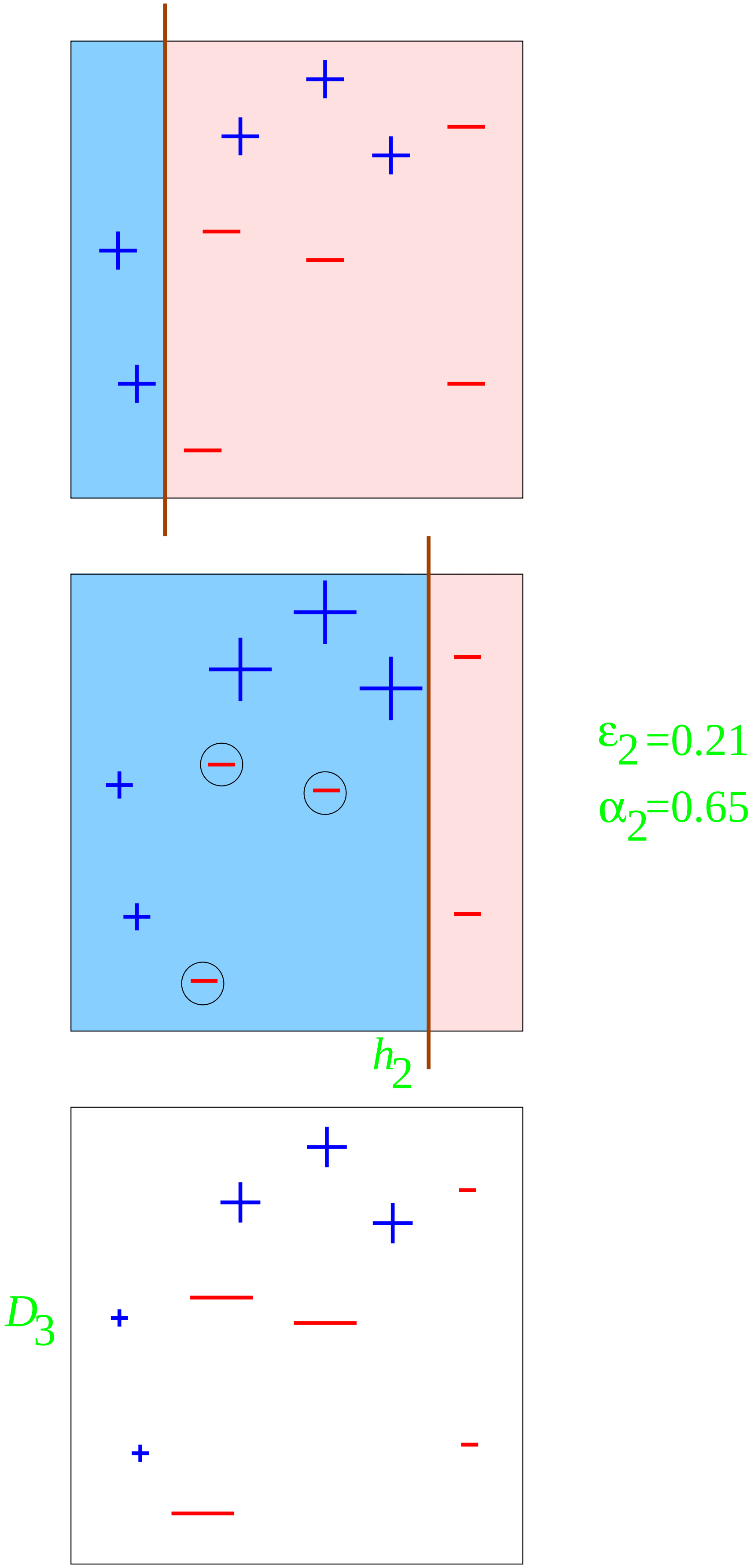
Toy Example



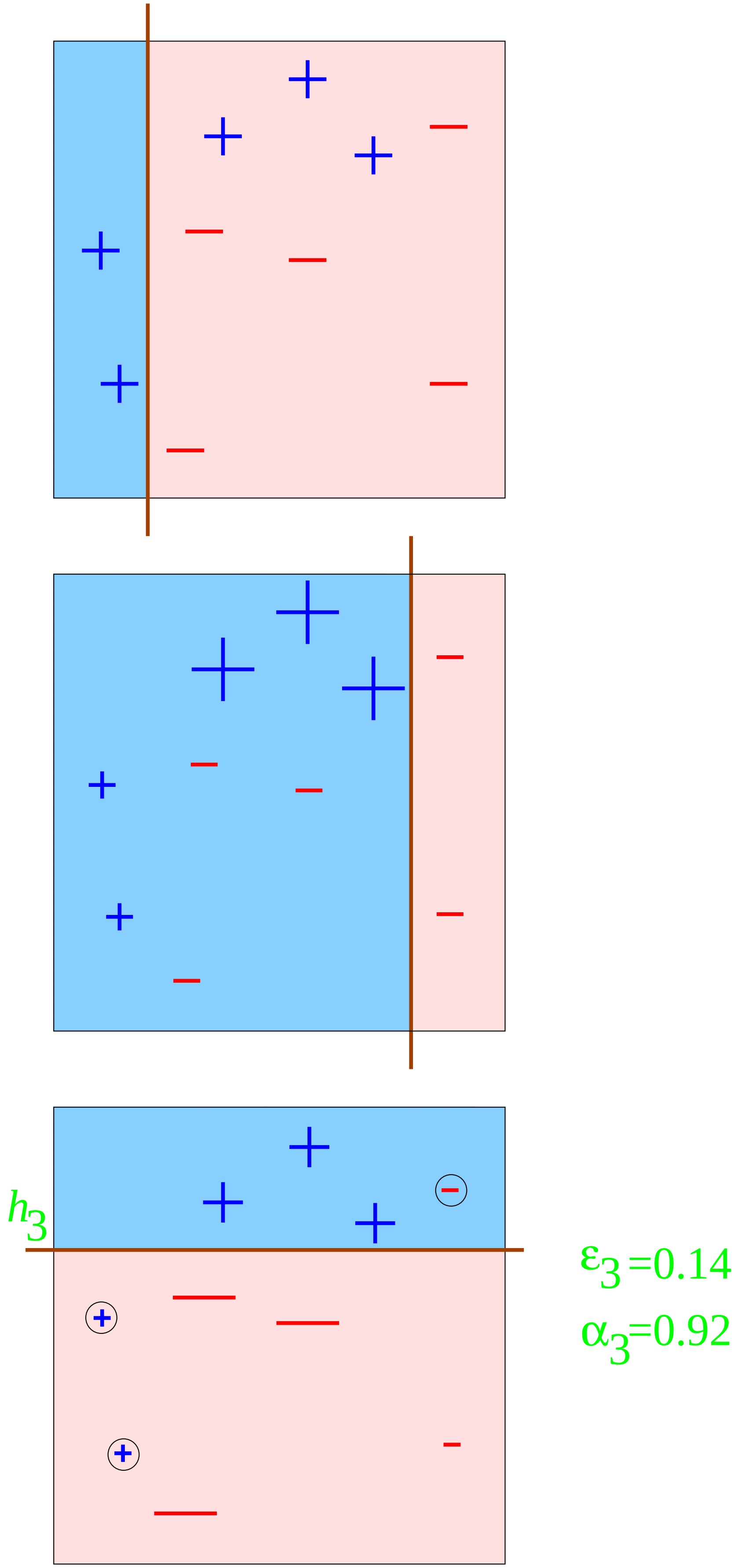
Round 1



Round 2

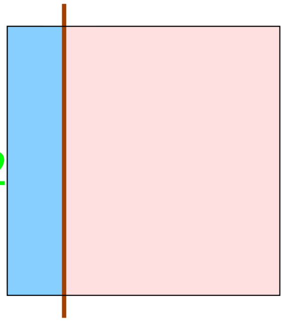
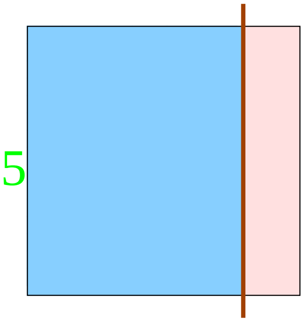
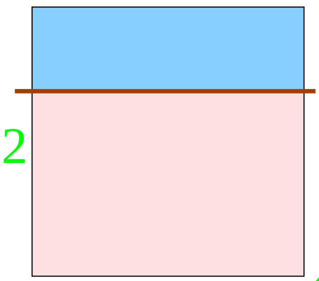


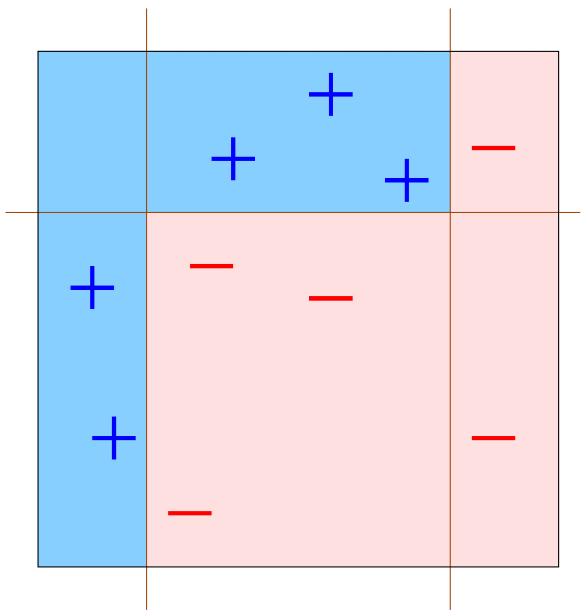
Round 3



Final Hypothesis

H_{final}

$= \text{sign}\Big($ 0.42  $+\ 0.65$  $+\ 0.92$  $\Big)$

$=$ 

* See demo at
`www.research.att.com/~yoav/adaboost`

Analyzing the training error

- Theorem:
  - run AdaBoost
  - let $\epsilon_t = 1/2 - \gamma_t$
  - 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 training error$(H_{\text{final}}) \le e^{-2\gamma^2 T}$
- 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 recursion:

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

$$= \frac{1}{m} \cdot \frac{e^{-y_i f(x_i)}}{\prod_t Z_t}$$

- *Step 2*: training error$(H_{\text{final}}) \le \prod_t Z_t$
- Proof:
  - $H_{\text{final}}(x) \neq y \Rightarrow y f(x) \le 0 \Rightarrow e^{-y f(x)} \ge 1$
  - so:

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

$$\le \frac{1}{m} \sum_i e^{-y_i f(x_i)}$$

$$= \sum_i D_{\text{final}}(i) \prod_t Z_t$$

$$= \prod_t Z_t$$

Proof (cont.)

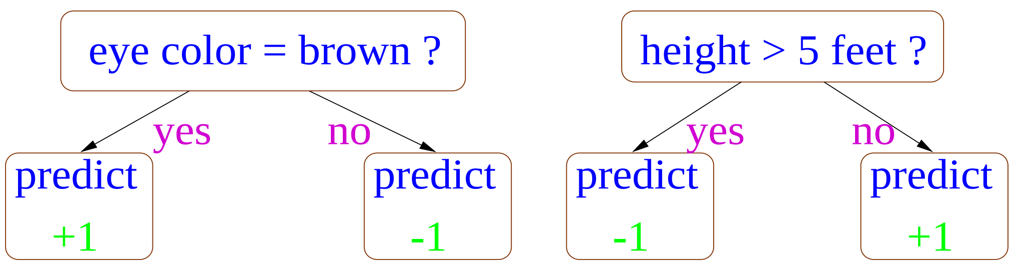
- *Step 3*: $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$
- Proof:

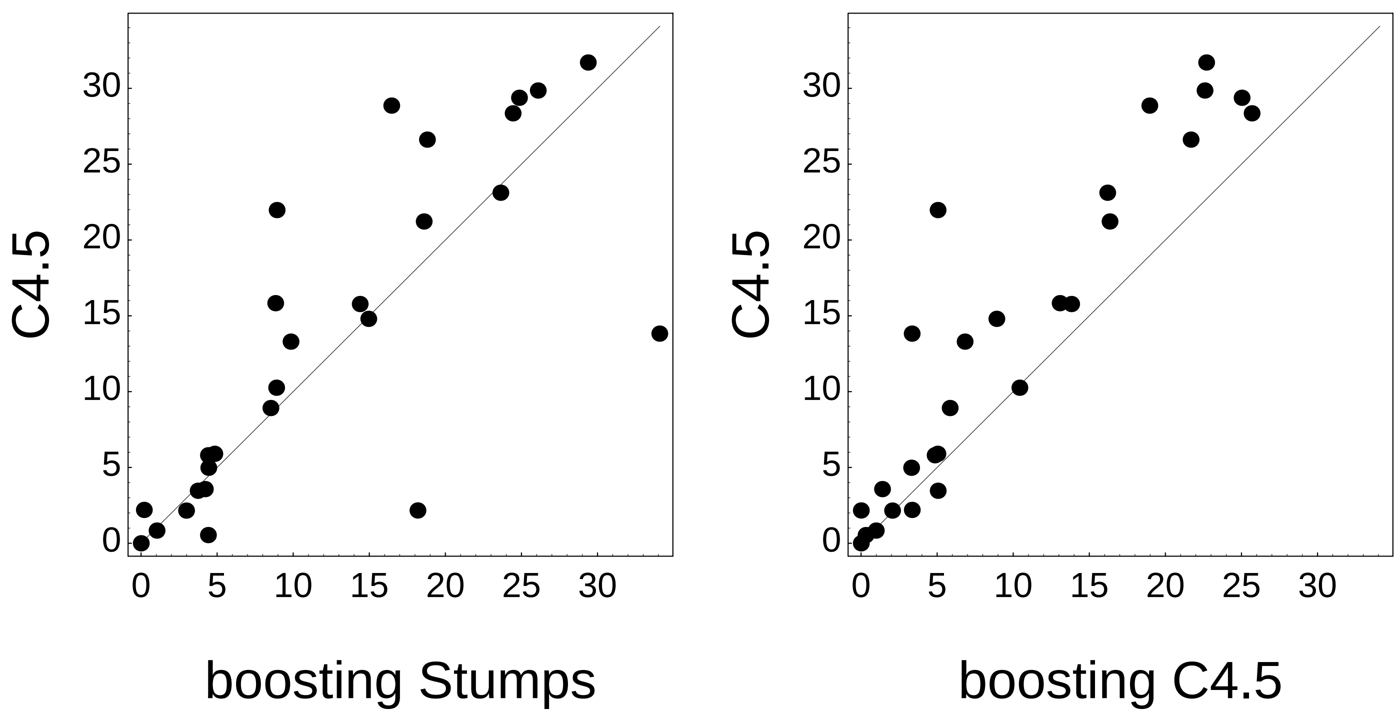
$$\begin{aligned} Z_t &= \sum_i D_t(i)\exp(-\alpha_t\, y_i\, h_t(x_i)) \\ &= \sum_{i:y_i \neq h_t(x_i)} D_t(i)e^{\alpha_t} + \sum_{i:y_i = h_t(x_i)} D_t(i)e^{-\alpha_t} \\ &= \epsilon_t\, e^{\alpha_t} + (1-\epsilon_t)\, e^{-\alpha_t} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \end{aligned}$$

UCI Experiments

[Freund & Schapire]

- 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





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

The Boosting Game

- row player $\leftrightarrow$ booster
- column player $\leftrightarrow$ weak learner
- matrix **M**:
  - row $\leftrightarrow$ example (x_i, y_i)
  - column $\leftrightarrow$ weak hypothesis h
  - $\mathbf{M}(i,h) = \begin{cases} 1 & \text{if } y_i = h(x_i) \\ 0 & \text{else} \end{cases}$

Boosting and the Minmax Theorem

- if:
  - $\forall$ distributions over examples
    $\exists h$ with accuracy $\geq \frac{1}{2} - \gamma$
- then:
  - $\min_{\mathbf{P}} \max_{h} \mathbf{M}(\mathbf{P}, h) \geq \frac{1}{2} - \gamma$
- 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 hypotheses which correctly classifies all examples

<u>AdaBoost and Game Theory</u>

[Freund & Schapire]

- 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 hypotheses converge to (approximate) maxmin strategy
- different instantiation of game-playing algorithm gives <u>on-line learning algorithms</u> (such as weighted majority algorithm)

Confidence-rated Predictions

[Schapire & Singer]

- useful to allow weak hypotheses 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 hypotheses

- questions:
  - how to choose h_t’s (specifically, how to assign confidences to predictions)
  - how to choose α_t’s

Confidence-rated Predictions (cont.)

- Theorem:

$$\text{training error}(H_{\text{final}}) \le \prod_t Z_t$$

- Proof: same as before
- therefore, on each round t, should choose h_t and α_t to minimize:

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

- given h_t, can find α_t which minimizes Z_t
  - analytically (sometimes)
  - numerically (in general)
- should design weak learner to minimize Z_t
  - e.g.: for decision trees, criterion gives:
    - splitting rule
    - assignment of confidences at leaves

Minimizing Exponential Loss

- AdaBoost attempts to minimize:

$$\prod_{t=1}^{T} Z_t = \frac{1}{m}\sum_i \exp(-y_i f(x_i))$$
$$= \frac{1}{m}\sum_i \exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right) \qquad \textbf{(*)}$$

- really a steepest descent procedure:
  - each round, add term $\alpha_t h_t$ to sum to minimize $(*)$
- why this loss function?
  - upper bound on training (classification) error
  - easy to work with
  - connection to logistic regression

    [Friedman, Hastie & Tibshirani]

Multiclass Problems

- 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 <u>if</u> $\forall t : \epsilon_t \leq 1/2$
  - in practice, not usually a problem for "strong" weak learners (e.g., C4.5)
  - significant problem for "weak" weak learners (e.g., decision stumps)

Reducing to Binary Problems

[Schapire & Singer]

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

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

AdaBoost.MH

- formally:

$$h_t : X \times Y \to \{-1, +1\} \text{(or } \mathbb{R}\text{)}$$

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

where $v_i(y) = \begin{cases} +1 & \text{if } y_i = y \\ -1 & \text{if } y_i \neq y \end{cases}$

$$H_{\text{final}}(x) = \arg\max_{y \in Y} \sum_t \alpha_t h_t(x, y)$$

- can prove:

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

Using Output Codes

[Schapire & Singer]

- alternative: reduce to "random" binary problems
- 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

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

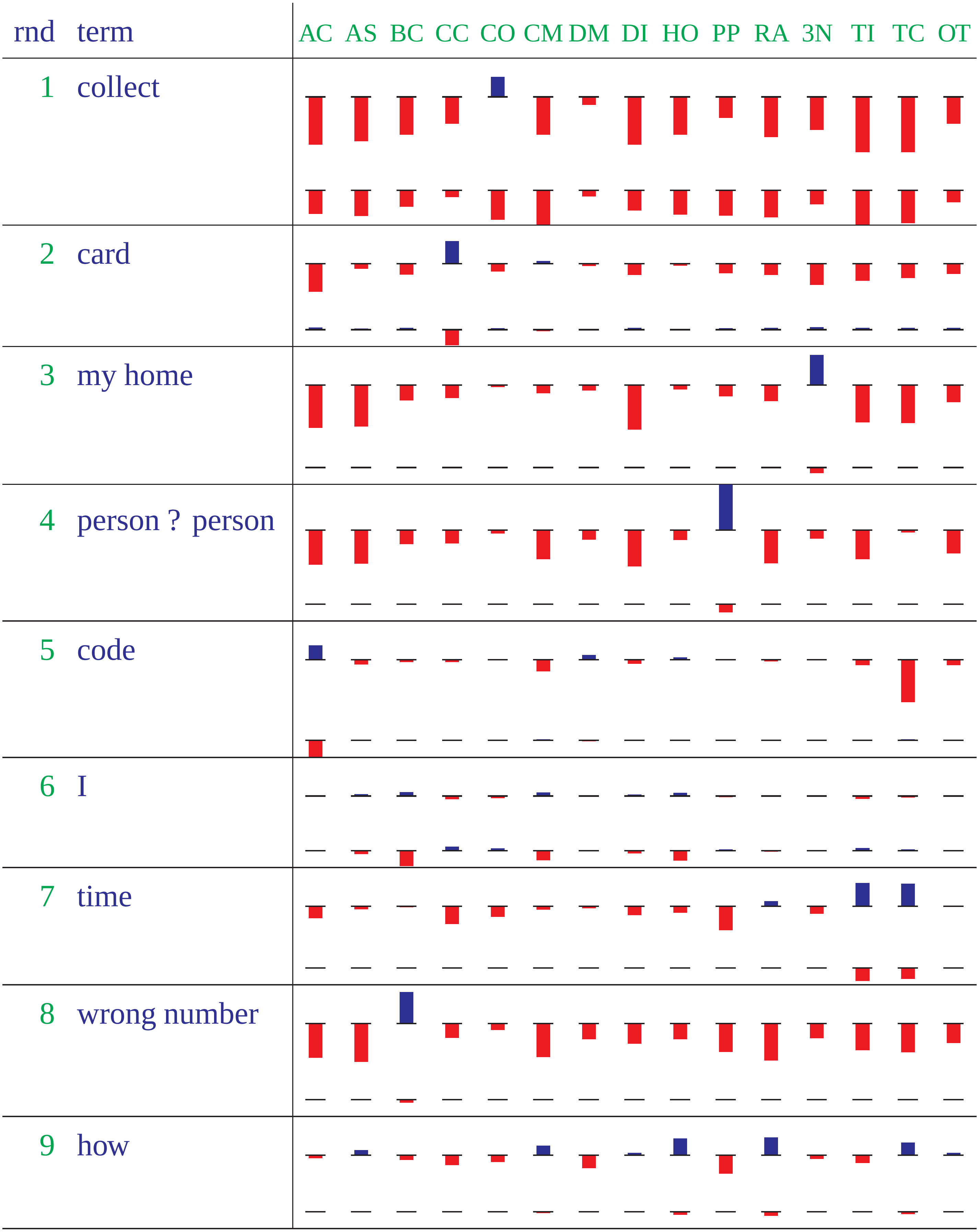
- to classify new example x:
  - evaluate hypothesis on $(x, \pi_1), \ldots, (x, \pi_4)$
  - choose label "most consistent" with results
- training error bounds independent of # of classes
- may be more efficient for very large # of classes

Example: Boosting for Text Categorization

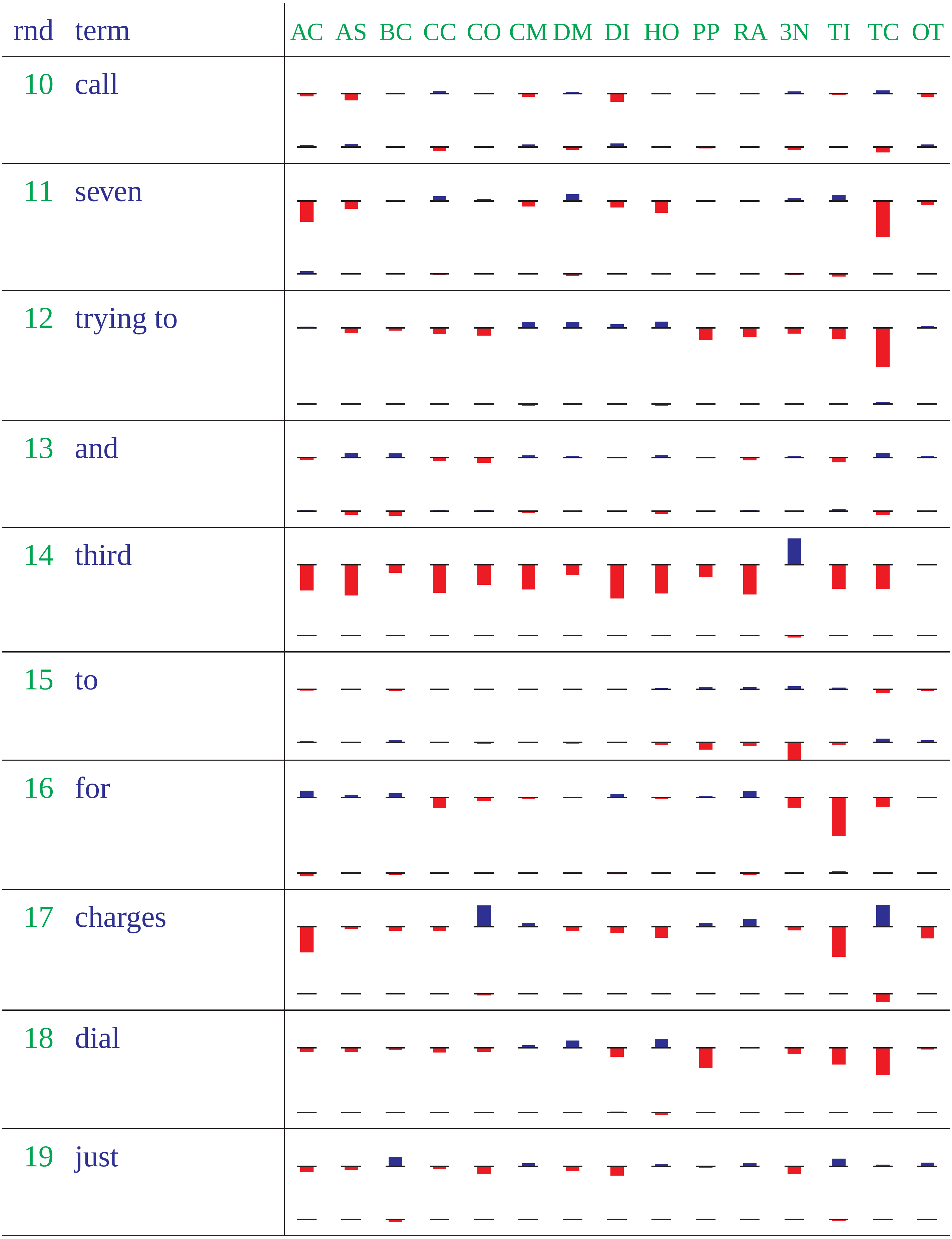
[Schapire & Singer]

- weak hypotheses: very simple weak hypotheses 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 (hand-transcribed)
  - 1000 test examples (both hand-transcribed and from speech recognizer)
  - categories: AreaCode, AttService, BillingCredit, CallingCard, Collect, Competitor, DialForMe, Directory, HowToDial, PersonToPerson, Rate, ThirdNumber, Time, TimeCharge, Other.

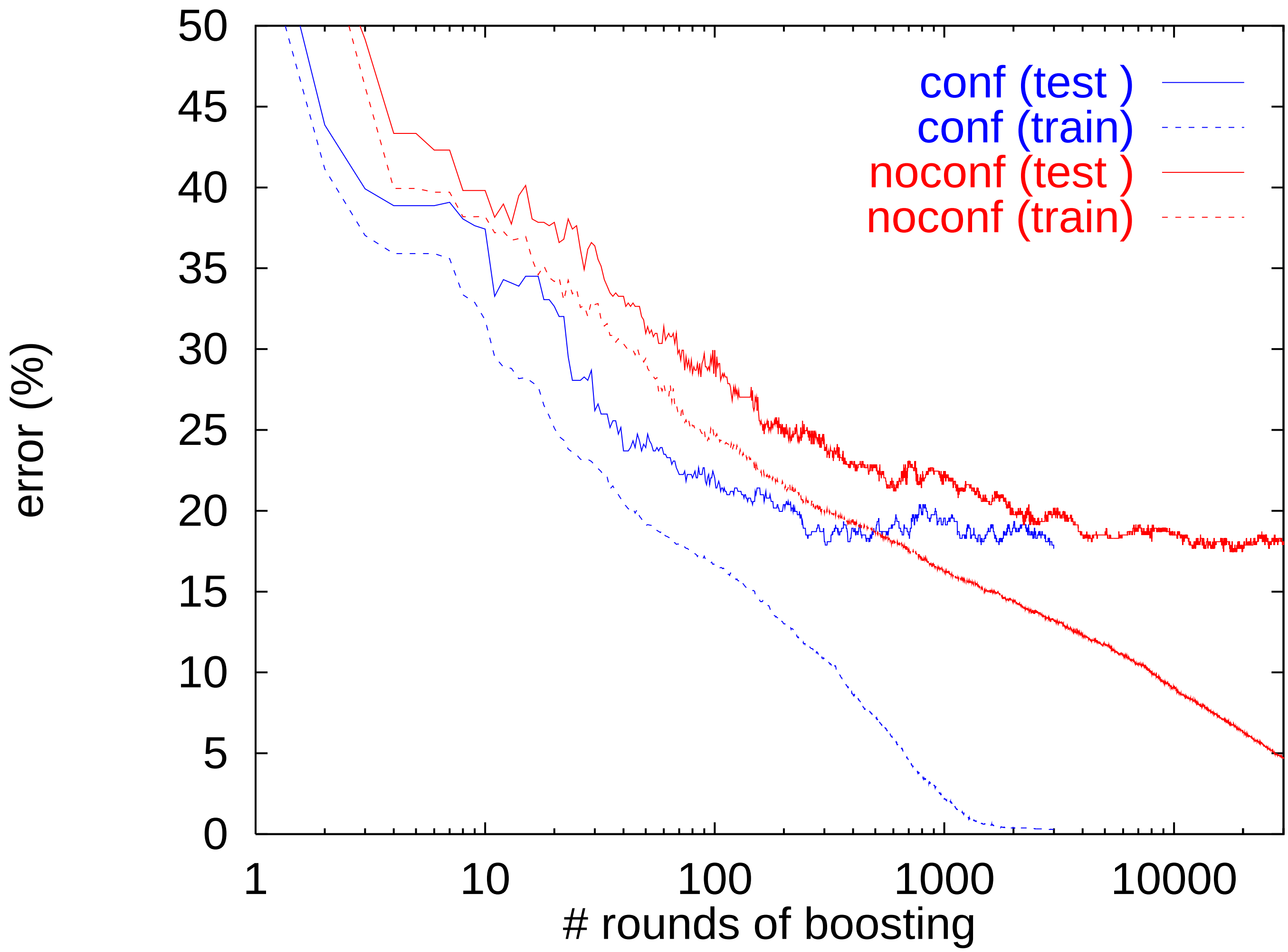
Weak Hypotheses



More Weak Hypotheses



Learning Curves



- test error reaches 20% for the first time on round...
  - 1,932 without confidence ratings
  - 191 with confidence ratings
- test error reaches 18% for the first time on round...
  - 10,909 without confidence ratings
  - 303 with confidence ratings

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)
- yes I like to make a long distance call and charge it to my home phone that's where I'm calling at my home (DialForMe)
- I like to make a call and charge it to my ameritech (Competitor)