

The Boosting Approach to Machine Learning

Rob Schapire
Princeton University

www.cs.princeton.edu/~schapire

The Boosting Approach

- devise computer program for deriving rough rules of thumb
- apply procedure to subset of emails
- obtain rule of thumb
- apply to 2nd subset of emails
- 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
 - more technically:
 - given “weak” learning algorithm that can consistently find classifier with error $\leq 1/2 - \gamma$
 - a boosting algorithm can provably construct single classifier with error $\leq \epsilon$
- (ϵ, γ small)

This Talk

- introduction to AdaBoost
- analysis of training error
- analysis of generalization error based on theory of margins
- applications, experiments and extensions

Background

- [Valiant '84]:
 - introduced theoretical (“PAC”) model for studying machine learning
- [Kearns & Valiant '88]:
 - open problem of finding a boosting algorithm
- [Schapire '89], [Freund '90]:
 - first polynomial-time boosting algorithms
- [Drucker, Schapire & Simard '92]:
 - first experiments using boosting

Background (cont.)

- [Freund & Schapire '95]:
 - introduced “AdaBoost” algorithm
 - strong practical advantages over previous boosting algorithms

- experiments and applications using AdaBoost:

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

[Schapire, Singer & Singhal '98]
[Abney, Schapire & Singer '99]
[Haruno, Shirai & Ooyama '99]
[Cohen & Singer '99]
[Dietterich '00]
[Schapire & Singer '00]
[Collins '00]
[Escudero, Mårquez & Rigau '00]

[Iyer, Lewis, Schapire, Singer & Singhal '00]
[Onoda, Rätsch & Müller '00]
[Tieu & Viola '00]
[Walker, Rambow & Rogati '01]
[Rochery, Schapire, Rahim & Gupta '01]
[Merler, Furlanello, Larcher & Sboner '01]
⋮

- continuing development of theory and algorithms:

[Breiman '98, '99]
[Schapire, Freund, Bartlett & Lee '98]
[Grove & Schuurmans '98]
[Mason, Bartlett & Baxter '98]
[Schapire & Singer '99]
[Cohen & Singer '99]
[Freund & Mason '99]
[Domingo & Watanabe '99]

[Mason, Baxter, Bartlett & Freaun '99, '00]
[Duffy & Helmbold '99, '02]
[Freund & Mason '99]
[Ridgeway, Madigan & Richardson '99]
[Kivinen & Warmuth '99]
[Friedman, Hastie & Tibshirani '00]
[Rätsch, Onoda & Müller '00]
[Rätsch, Warmuth, Mika, Onoda, Lemm & Müller '00]

[Allwein, Schapire & Singer '00]
[Friedman '01]
[Koltchinskii, Panchenko & Lozano '01]
[Collins, Schapire & Singer '02]
[Demiriz, Bennett & Shawe-Taylor '02]
[Lebanon & Lafferty '02]
⋮

A Formal Description of Boosting

- given training set $(x_1, y_1), \dots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \dots, T$:
 - construct distribution D_t on $\{1, \dots, m\}$
 - find weak classifier (“rule of thumb”)
$$h_t : X \rightarrow \{-1, +1\}$$
with small error ϵ_t on D_t :
$$\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$$
- output final classifier H_{final}

AdaBoost

[with Freund]

- constructing D_t :

- $D_1(i) = 1/m$

- given D_t and h_t :

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

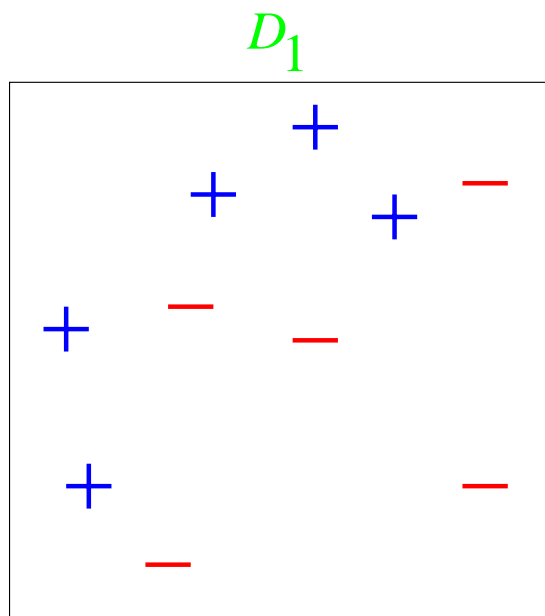
where $Z_t =$ normalization constant

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

- final classifier:

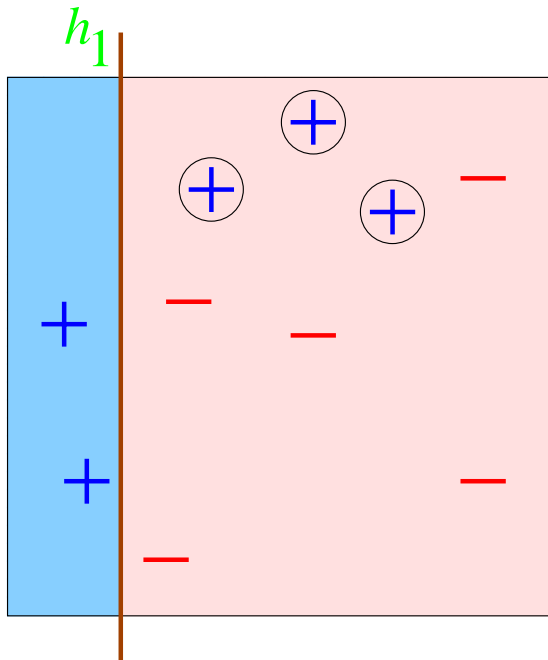
- $H_{\text{final}}(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$

Toy Example



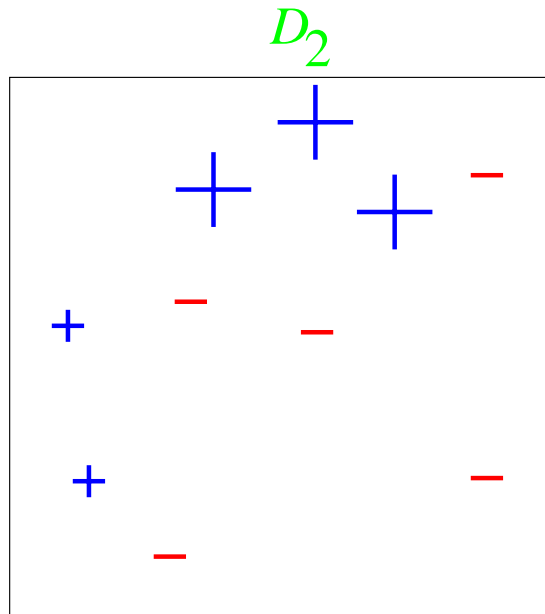
weak classifiers = vertical or horizontal half-planes

Round 1

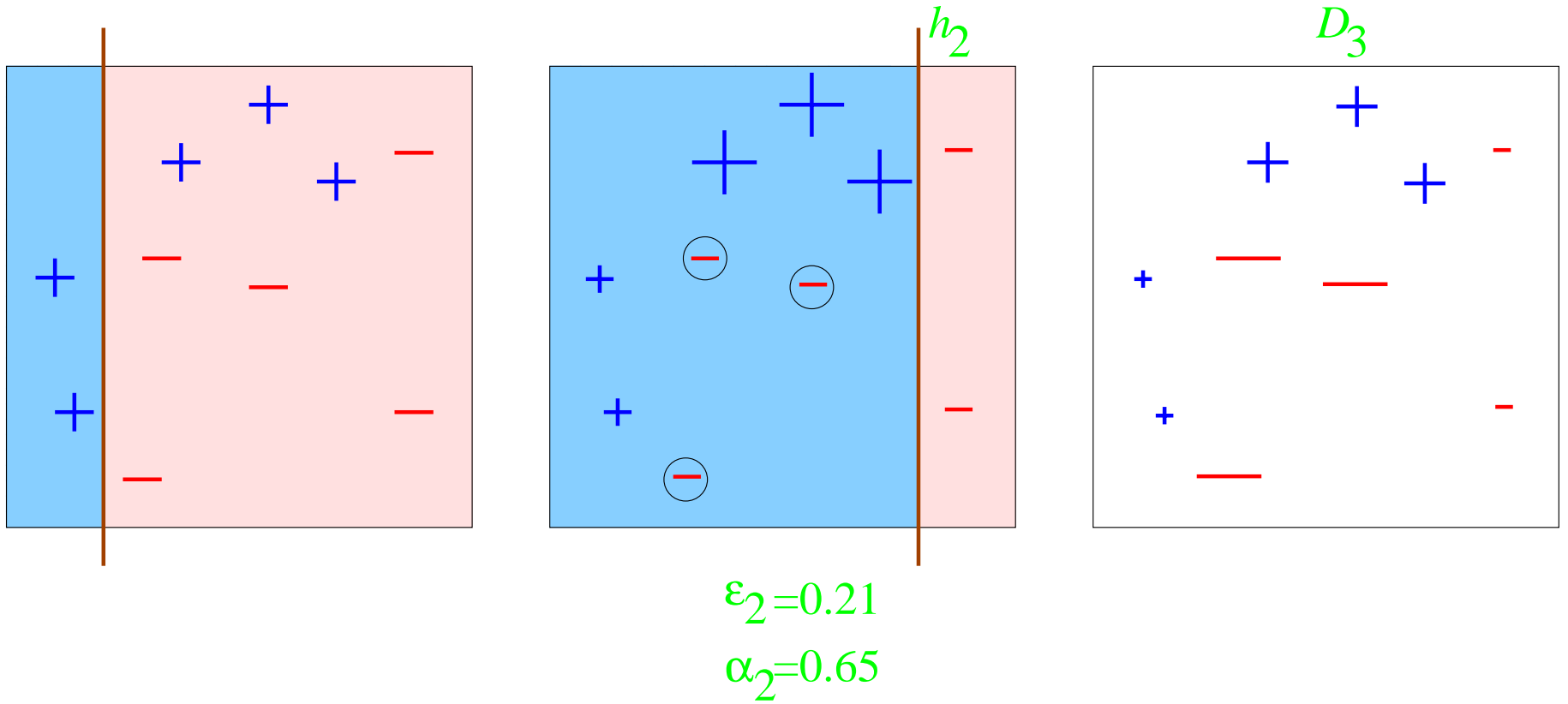


$$\varepsilon_1 = 0.30$$

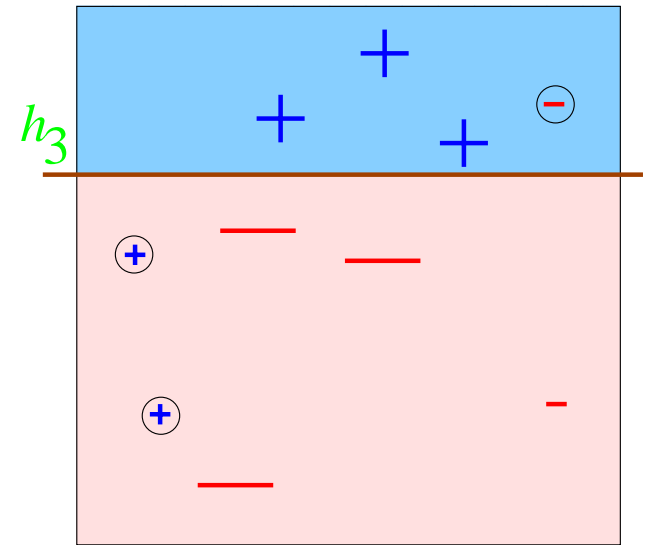
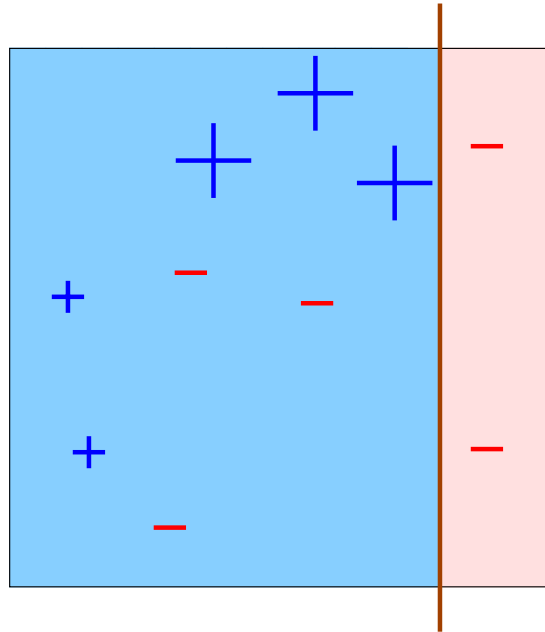
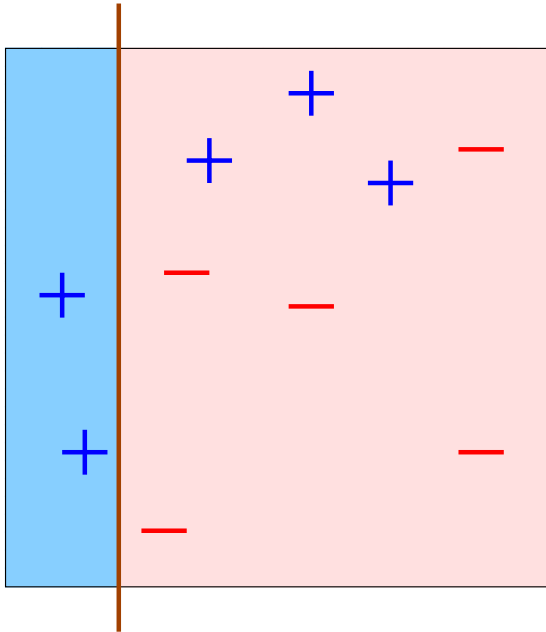
$$\alpha_1 = 0.42$$



Round 2



Round 3

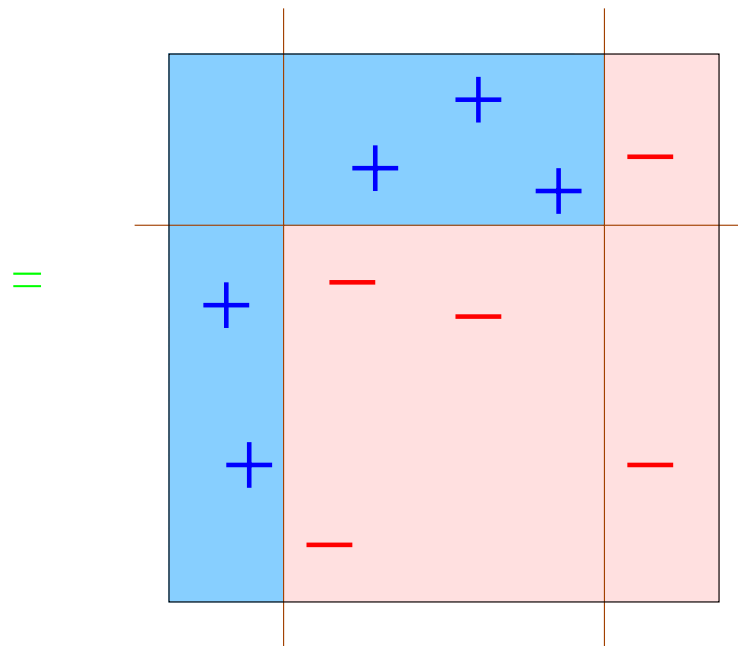


$$\epsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

Final Classifier

$$H_{\text{final}} = \text{sign} \left(0.42 \left(\begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} \right) + 0.65 \left(\begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} \right) + 0.92 \left(\begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} \right) \right)$$



Analyzing the Training Error

- Theorem:

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

$$\text{training error}(H_{\text{final}}) \leq \exp\left(-2 \sum_t \gamma_t^2\right)$$

- 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 Intuition

- on round t :
increase weight of examples incorrectly classified by h_t
- if x_i incorrectly classified by H_{final}
then x_i incorrectly classified by (weighted) majority of h_t 's
- ∴ if x_i incorrectly classified by H_{final}
then x_i must have “large” weight under final distribution D_{T+1}
- ∴ number of incorrectly classified examples “small”
(since total weight ≤ 1)

Proof

- Step 1: unwrapping recurrence:

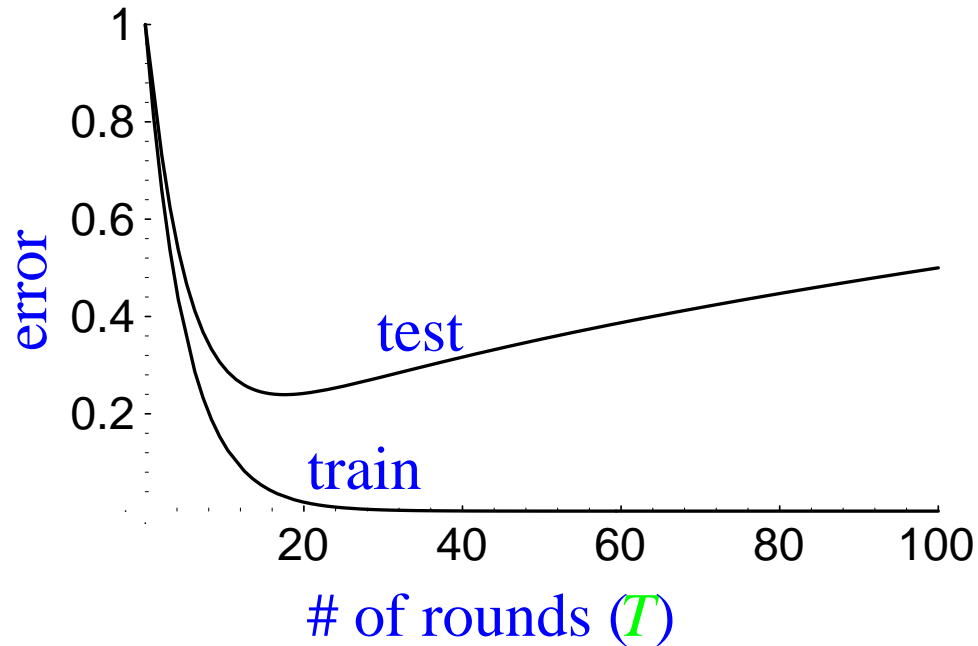
$$D_{T+1}(i) = \frac{1}{m} \frac{\exp(-y_i f(x_i))}{\prod_t Z_t}$$

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

- Step 2: training error(H_{final})
$$\begin{aligned} &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i \neq H_{\text{final}}(x_i) \\ 0 & \text{else} \end{cases} \\ &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i f(x_i) \leq 0 \\ 0 & \text{else} \end{cases} \\ &\leq \frac{1}{m} \sum_i \exp(-y_i f(x_i)) \\ &= \sum_i D_{T+1}(i) \prod_t Z_t \\ &= \prod_t Z_t \end{aligned}$$

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

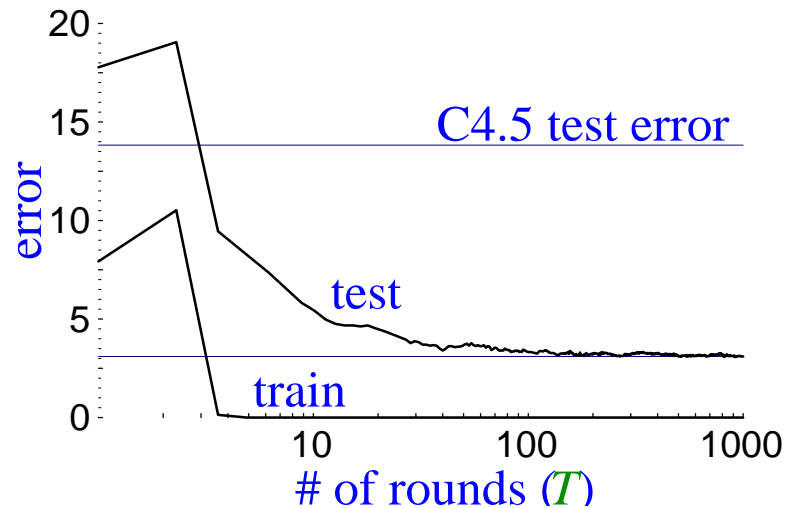
A First Attempt at Analyzing the Generalization Error



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

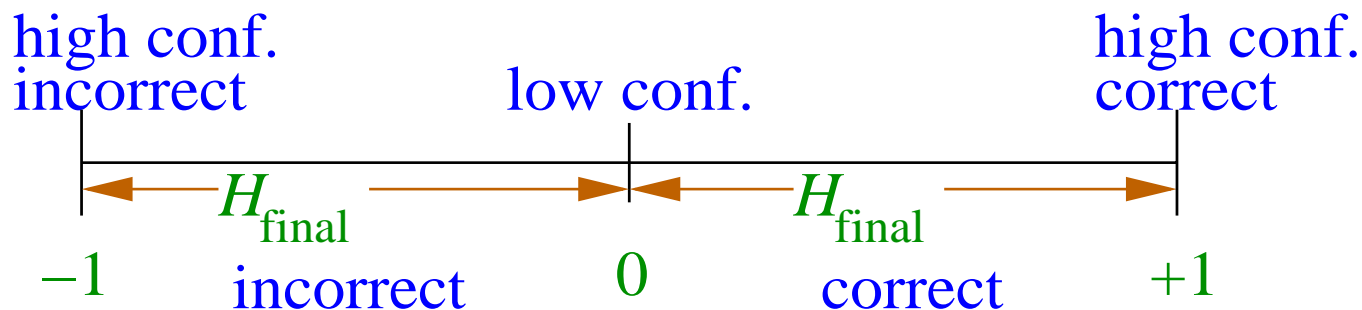
A Better Story: Theory of Margins

[with Freund, Bartlett & Lee]

- key idea:
 - training error only measures whether classifications are right or wrong
 - should also consider confidence of classifications
- can write: $H_{\text{final}}(x) = \text{sign}(f(x))$

where $f(x) = \frac{\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \in [-1, +1]$

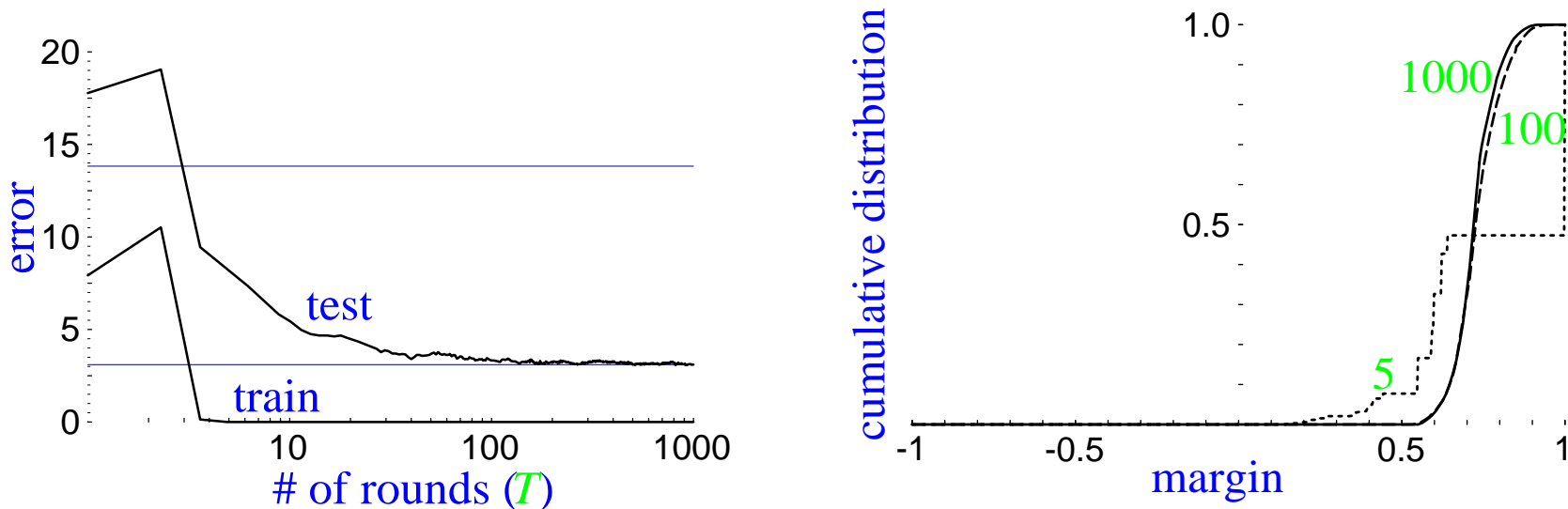
- define margin of example (x, y) to be $y f(x)$
= measure of confidence of classifications



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

- if all training examples have large margins, then can approximate final classifier by a much smaller classifier
 - (similar to how polls can predict outcome of a not-too-close election)
- can use this to prove that larger margins \Rightarrow better test error, regardless of number of weak classifiers
- can also prove that boosting tends to increase margins of training examples by concentrating on those with smallest margin
- so:
 - although final classifier is getting larger,
 - margins are likely to be increasing,
 - so final classifier is actually getting close to a simpler classifier,
 - driving down the test error

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

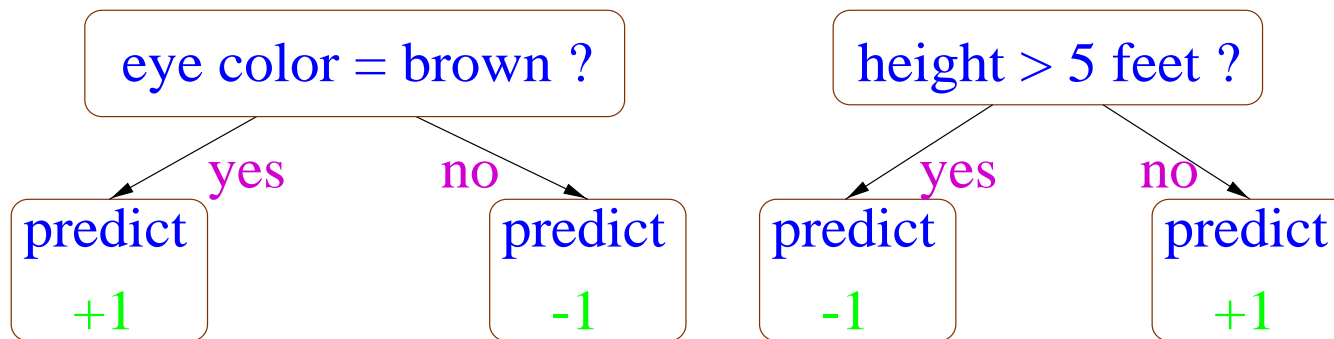
Caveats

- 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 ($\gamma_t \rightarrow 0$ too quickly)
 - underfitting
 - low margins → overfitting
- empirically, AdaBoost seems especially susceptible to uniform noise

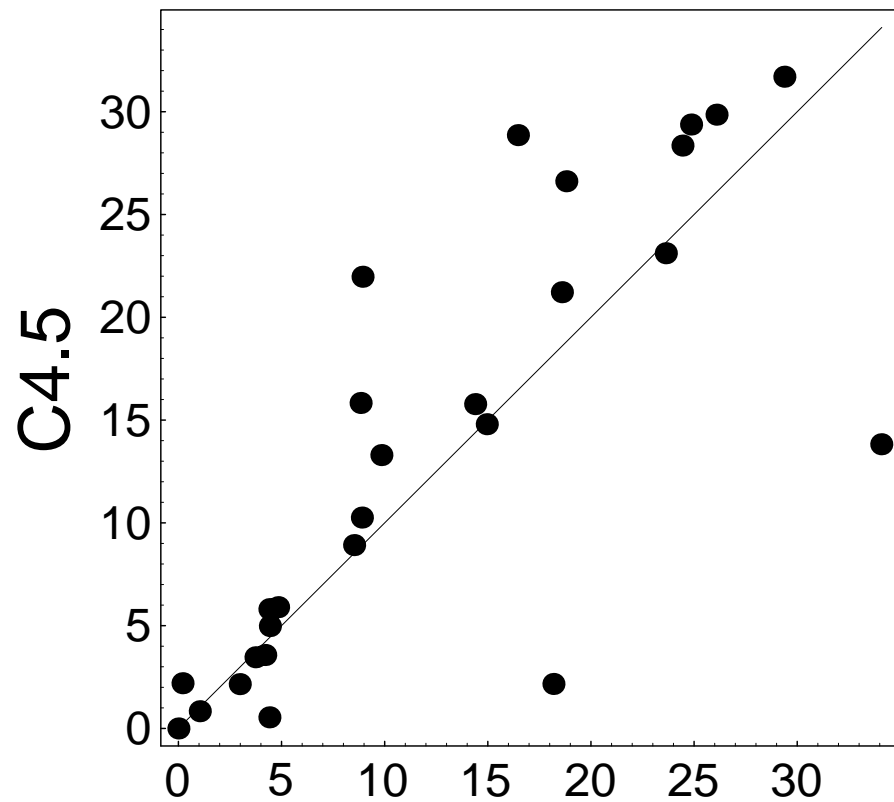
UCI Experiments

[with Freund]

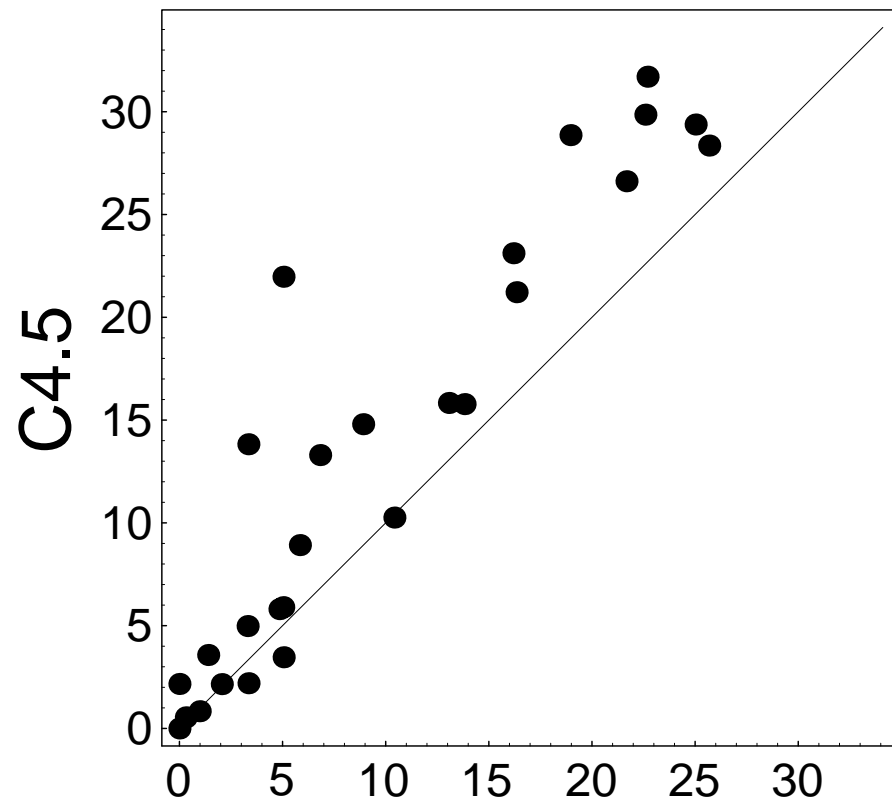
- tested AdaBoost on UCI benchmarks
- used:
 - C4.5 (Quinlan's decision tree algorithm)
 - “decision stumps”: very simple rules of thumb that test on single attributes



UCI Results



boosting Stumps



boosting C4.5

Multiclass Problems

[with Freund & Singer]

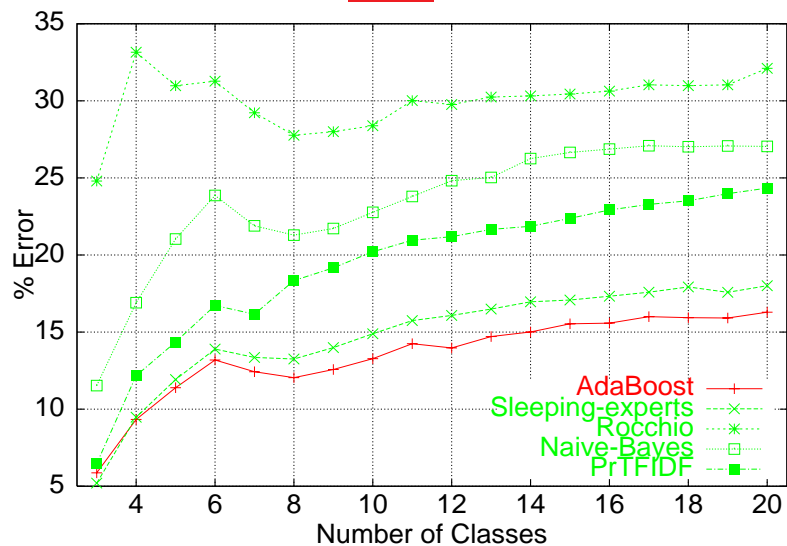
- most direct extension effective only if all weak classifiers have error $\leq 1/2$
 - difficult to achieve for “weak” weak learners
- instead, reduce to binary problem by creating several binary questions for each example:
 - “*does or does not example x belong to class 1?*”
 - “*does or does not example x belong to class 2?*”
 - ⋮

Application: Text Categorization

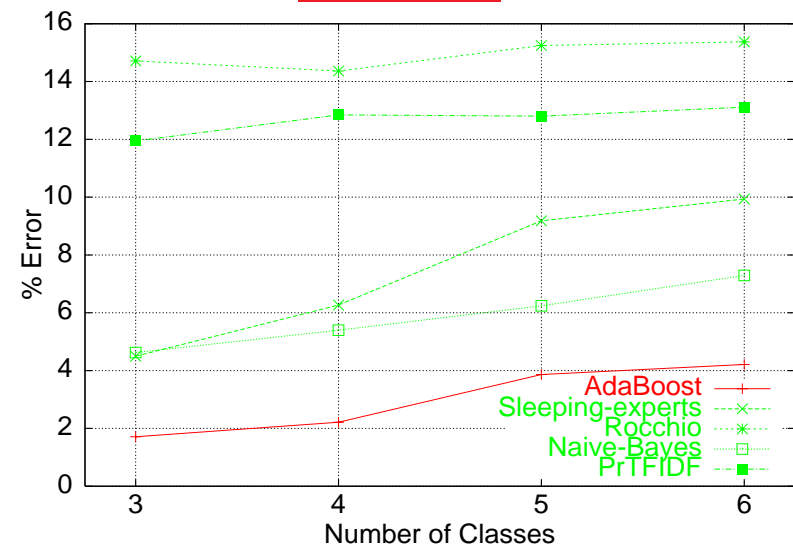
[with Singer]

- weak classifiers are decision stumps
 - test for presence of word or short phrase in document
 - e.g.:
 - “If the word *Clinton* appears in the document predict document is about *politics*”
- in our experiments, consistently beat or tied tested competitors

AP



Reuters



Extension: Confidence-rated Predictions

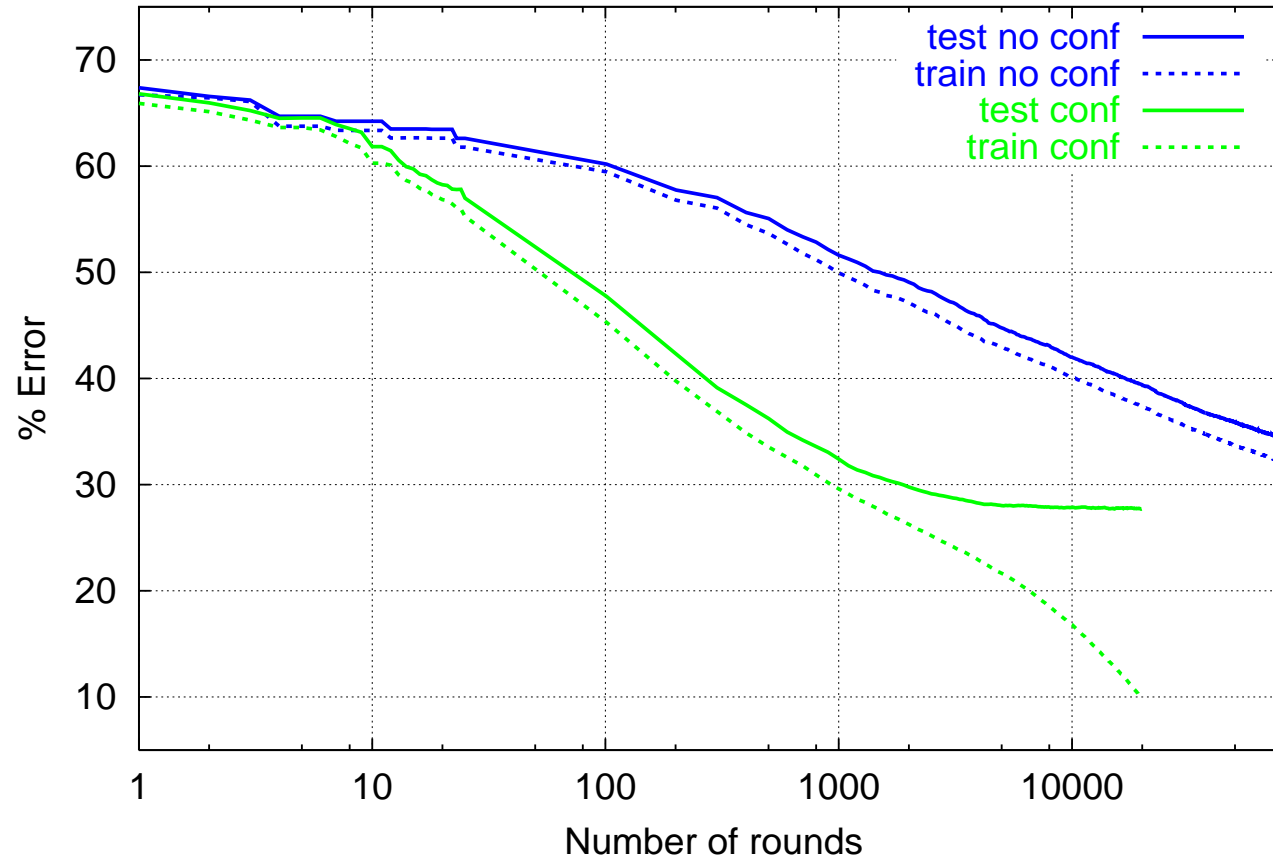
[with Singer]

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

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

- proposed general principle for:
 - modifying AdaBoost
 - designing weak learner to find (confidence-rated) h_t 's
- sometimes makes learning much faster since removes need to undo under-confident predictions of earlier weak classifiers

Confidence-rated Predictions Help a Lot

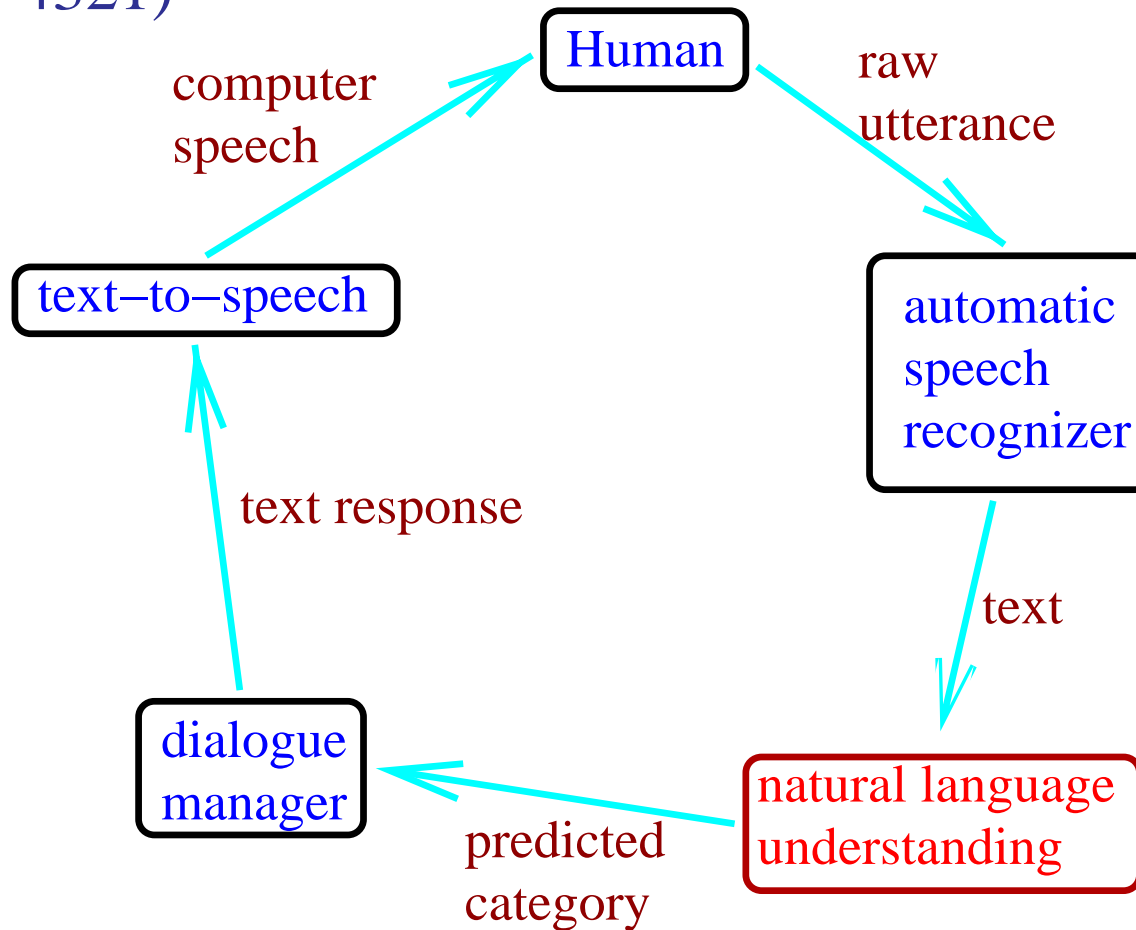


% error	round first reached		speedup
	conf.	no conf.	
40	268	16,938	63.2
35	598	65,292	109.2
30	1,888	>80,000	—

Application: Human-computer Spoken Dialogue

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

- phone “helpdesk” for AT&T’s Natural Voices text-to-speech business (1-877-741-4321)



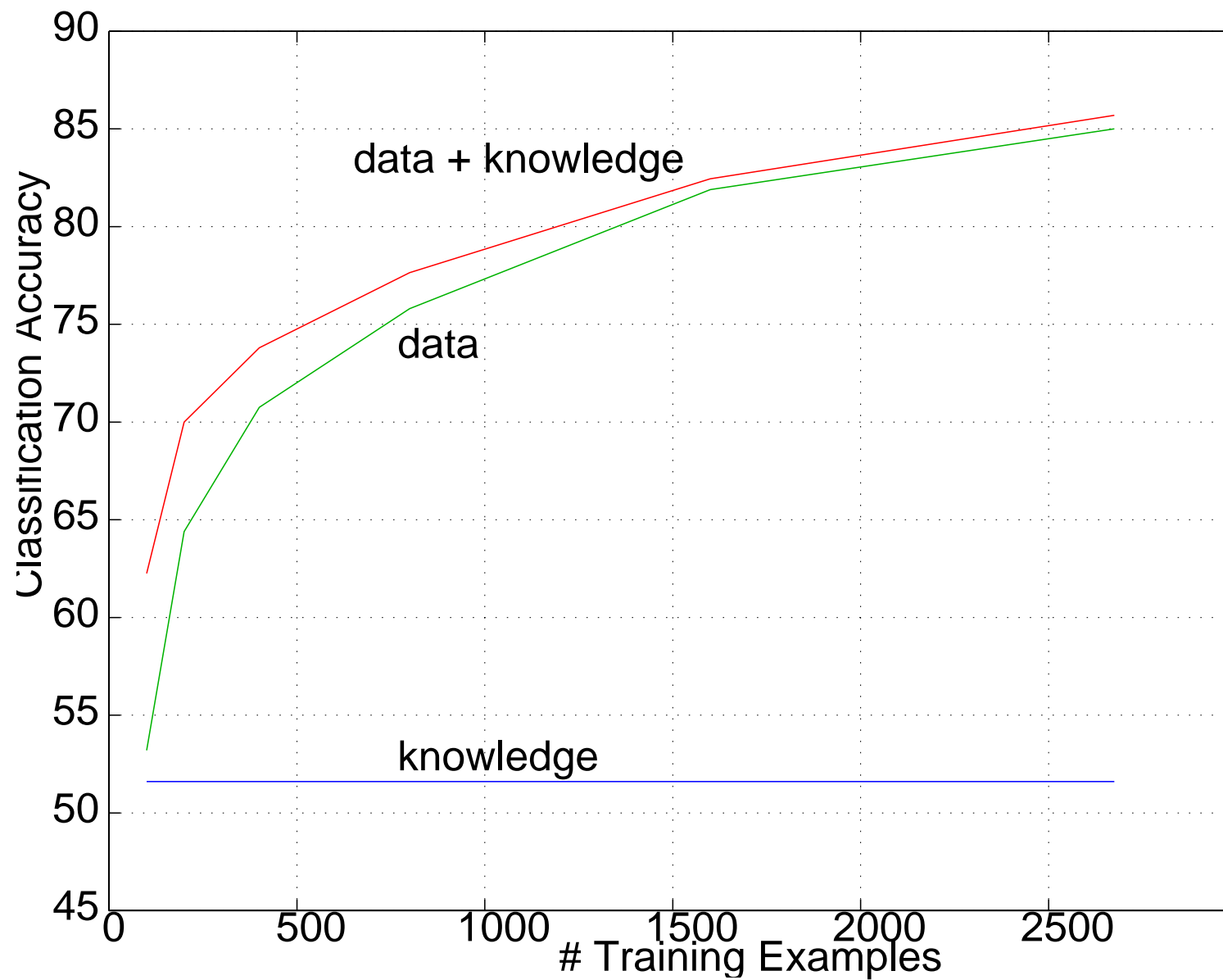
- NLU’s job: classify caller utterances into 24 categories (demo, sales rep, pricing info, yes, no, etc.)

Incorporating Human Knowledge

[with Rochery, Rahim & Gupta]

- boosting is data-driven
 - so works best with lots of data
- for rapid deployment, can't wait to gather lots of data
 - want to compensate with human knowledge
- idea: balance fit to data against fit to prior, human-built model

Results



Application: Bidding Agents

[with Stone, Csirik, Littman & McAllester]

- trading agent competition (TAC)
- 8 agents in each game
- must purchase flights, hotel rooms and entertainment tickets for 8 clients in complicated, interacting auctions
- value of one good depends on price of others
 - e.g., need both ingoing and outgoing flights
- so: need to predict prices, especially of hotel rooms
 - used boosting
- second place in tournament using straight scores
 - (first place with “handicapped” scores)

Predicting Hotel Prices

- predicting real numbers (prices)
- want to estimate entire distribution of prices, given current conditions
- main ideas:
 - reduce to multiple binary classification problems:
 - “*is price above or below \$100?*”
 - “*is price above or below \$150?*”
 - ⋮
 - extract probabilities using modification of boosting for logistic regression [Collins, Schapire & Singer, '02] [Duffy & Helmbold '99]
- can be applied to any conditional density estimation problem

Conclusions

- boosting is a useful new 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
- other stuff:
 - theoretical connections to:
 - game theory and linear programming
 - support-vector machines
 - logistic regression
 - convex analysis and Bregman distances
 - tool for data cleaning:
 - very effective at finding outliers (mis-labeled or ambiguously labeled examples)