Boosting: Foundations and Algorithms

Rob Schapire

Example: Spam Filtering

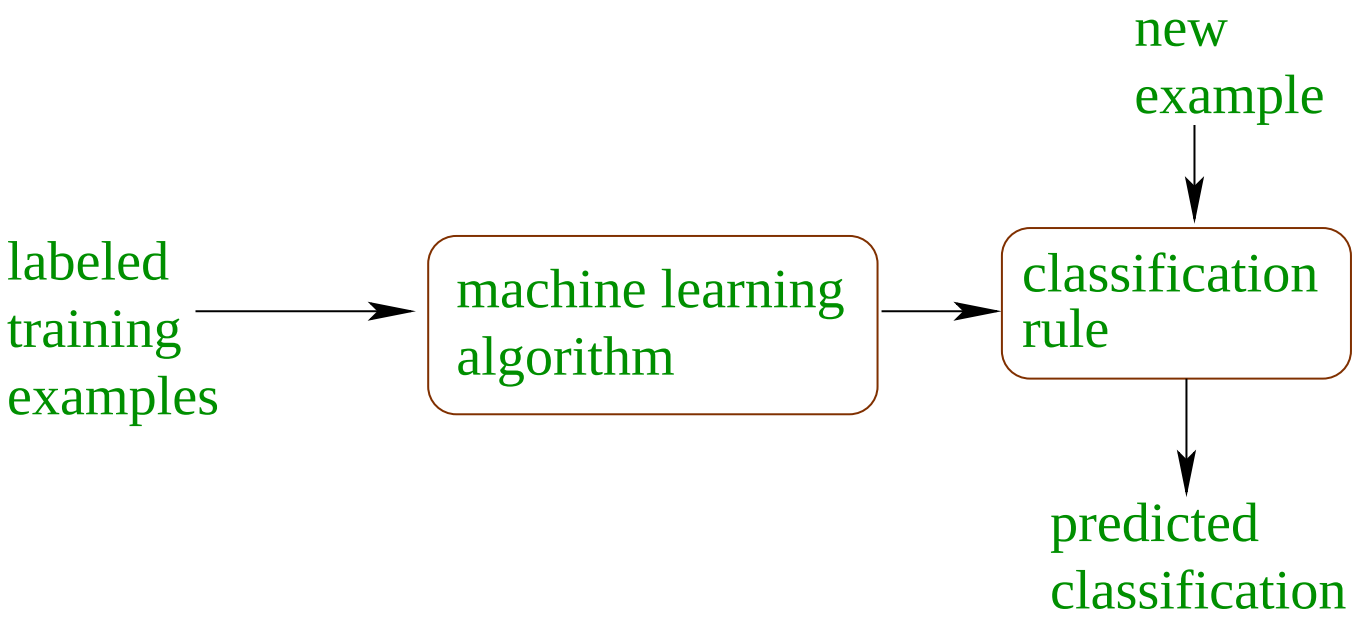
- problem: filter out spam (junk email)
- gather large collection of examples of spam and non-spam:

From: yoav@ucsd.edu	Rob, can you review a paper...	non-spam
From: xa412@hotmail.com	Earn money without working!!!! ...	spam
⋮	⋮	⋮

- goal: have computer learn from examples to distinguish spam from non-spam

Machine Learning

- studies how to automatically learn to make accurate predictions based on past observations
- classification problems:
  - classify examples into given set of categories



Examples of Classification Problems

- text categorization (e.g., spam filtering)
- fraud detection
- machine vision (e.g., face detection)
- natural-language processing
  (e.g., spoken language understanding)
- market segmentation
  (e.g.: predict if customer will respond to promotion)
- bioinformatics
  (e.g., classify proteins according to their function)
  ⋮

Back to Spam

- main observation:
  - easy to find "rules of thumb" that are "often" correct
    - *If* 'viagra' *occurs in message, then predict* 'spam'
  - hard to find single rule that is very highly accurate

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%

Early History

- [Valiant '84]:
  - introduced theoretical ("PAC") model for studying machine learning
- [Kearns & Valiant '88]:
  - open problem of finding a boosting algorithm
- 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

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:

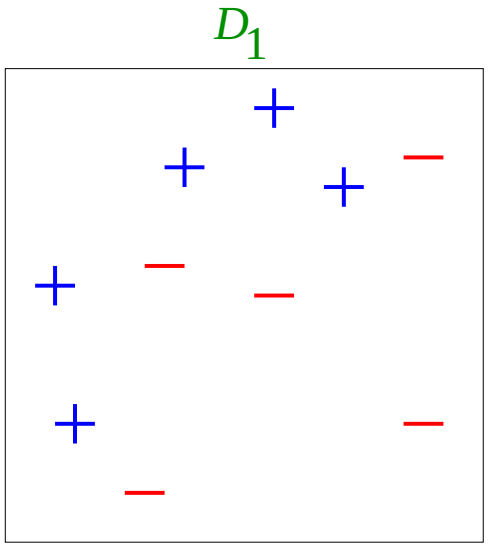
$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i)) \end{aligned}$$

where $Z_t =$ normalization factor

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

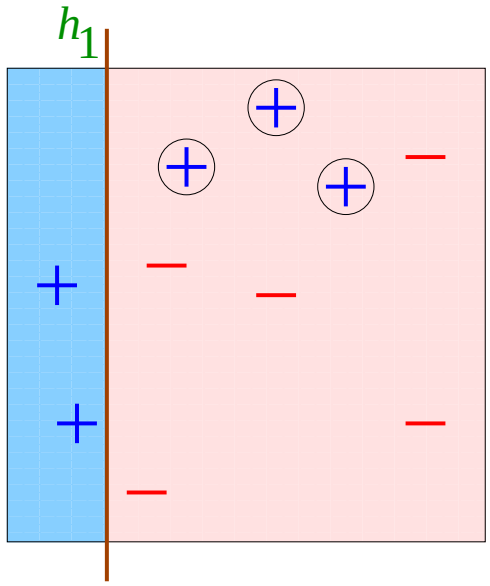
- final classifier:
  - $H_{\text{final}}(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$

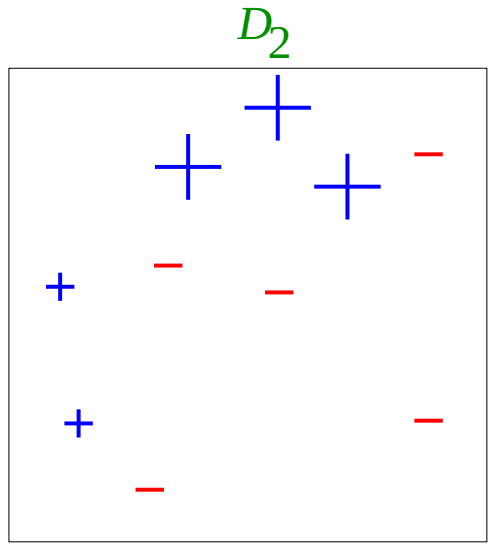
Toy Example



weak classifiers = vertical or horizontal half-planes

Round 1

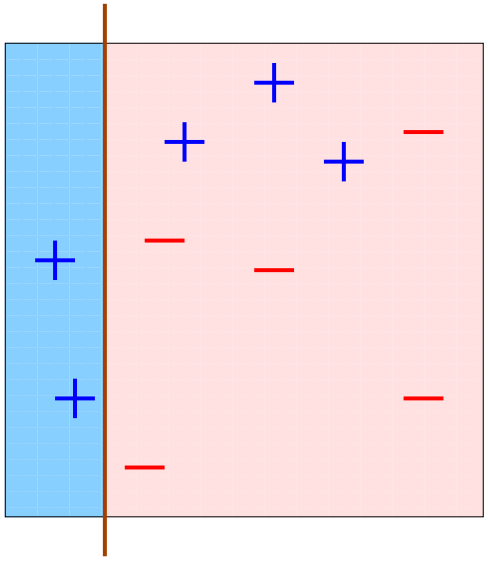


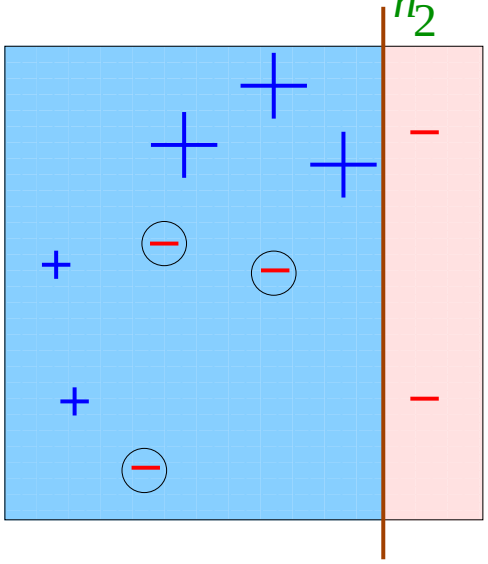


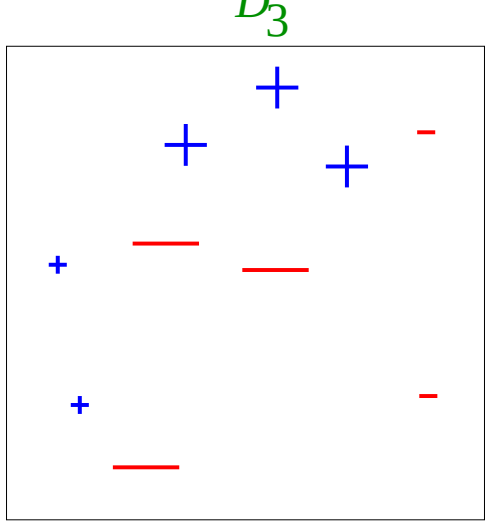
$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

Round 2



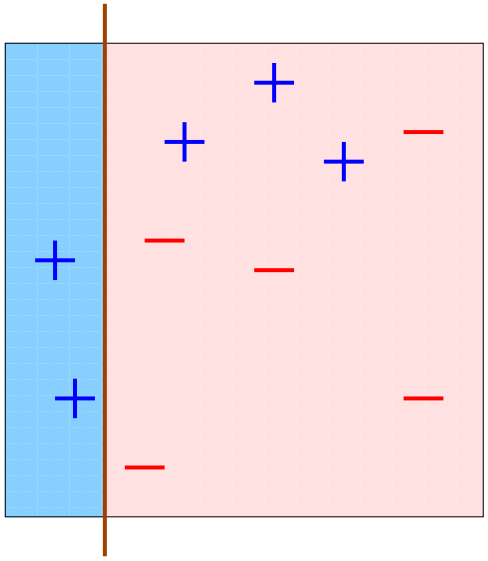


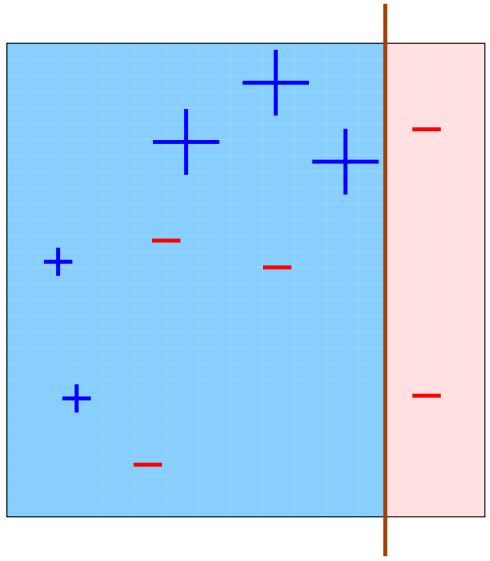


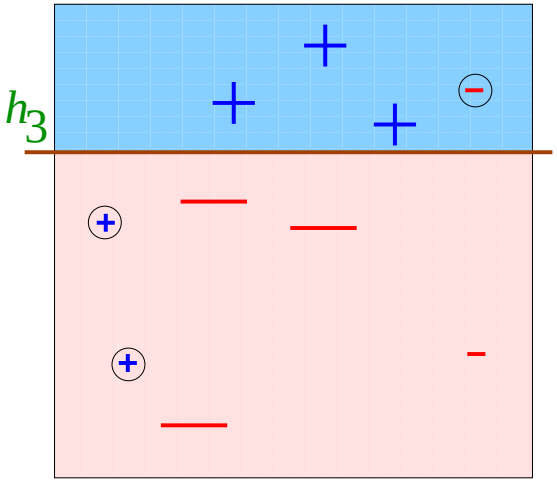
$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

Round 3



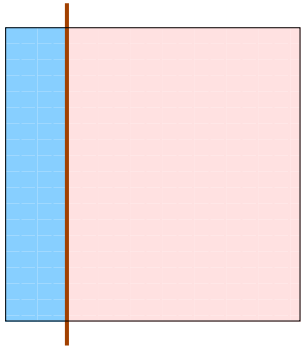
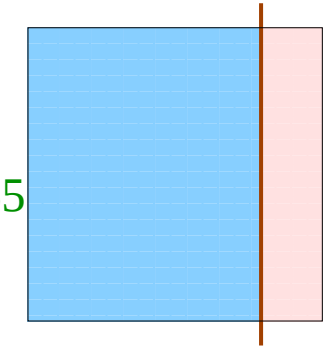
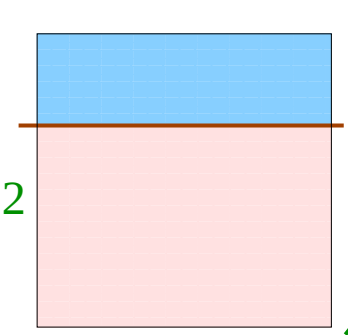


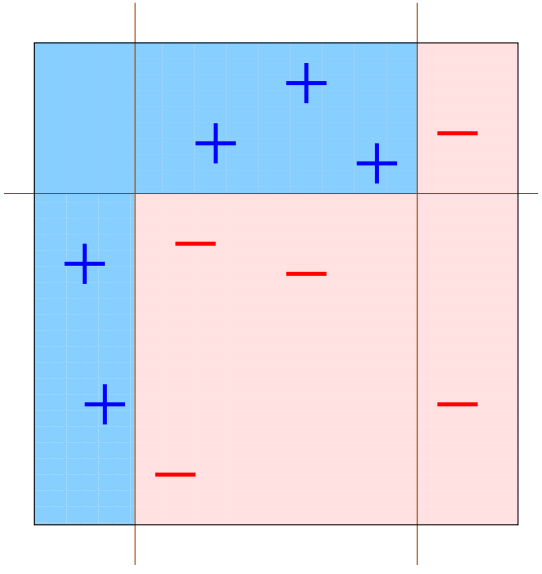


$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

Final Classifier

$H_{\text{final}} = \text{sign}\left(0.42 \right.$  $+ 0.65$  $+ 0.92$  $\left.\right)$

$=$ 

AdaBoost (recap)

- given training set $(x_1, y_1), \ldots, (x_m, y_m)$
  where $x_i \in X$, $y_i \in \{-1, +1\}$
- initialize $D_1(i) = 1/m$ $(\forall i)$
- for $t = 1, \ldots, T$:
  - train weak classifier $h_t : X \to \{-1, +1\}$ with error
    $\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$
  - $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$
  - update $\forall i$:

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

    where $Z_t =$ normalization factor

- $H_{\text{final}}(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Analyzing the Training Error

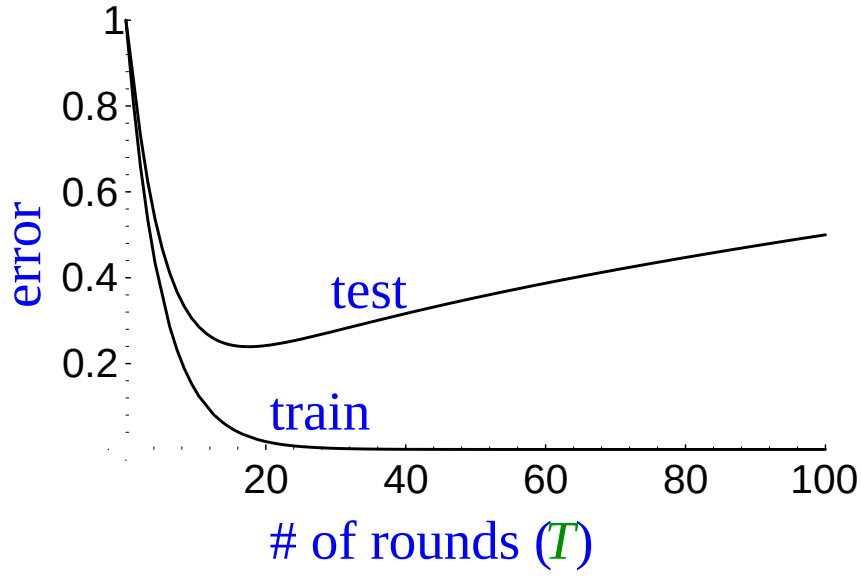
[with Freund]

- Theorem:
  - write ϵ_t as $\frac{1}{2} - \gamma_t$ [γ_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 $\gamma_t \gg \gamma$

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

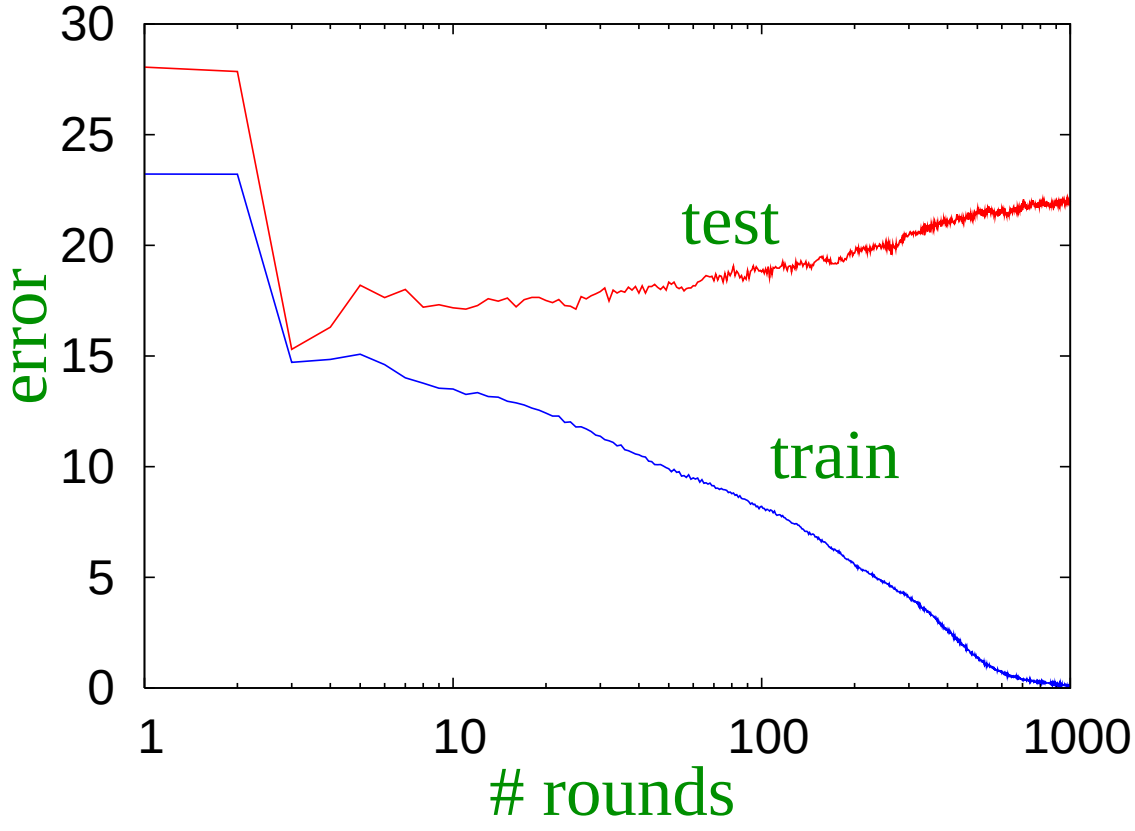
Technically...

- 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 $= \mathrm{E}\,[\text{test error}]$
- predicts overfitting

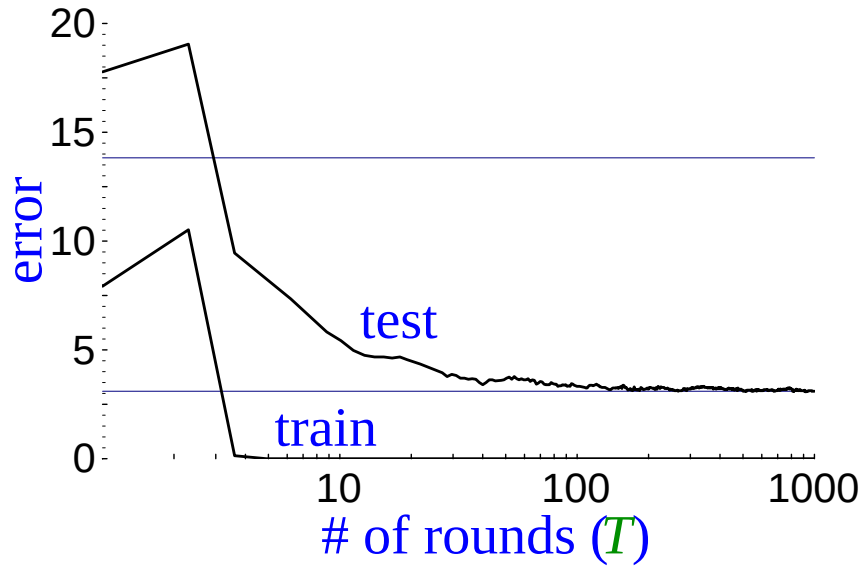
Overfitting Can Happen



(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!

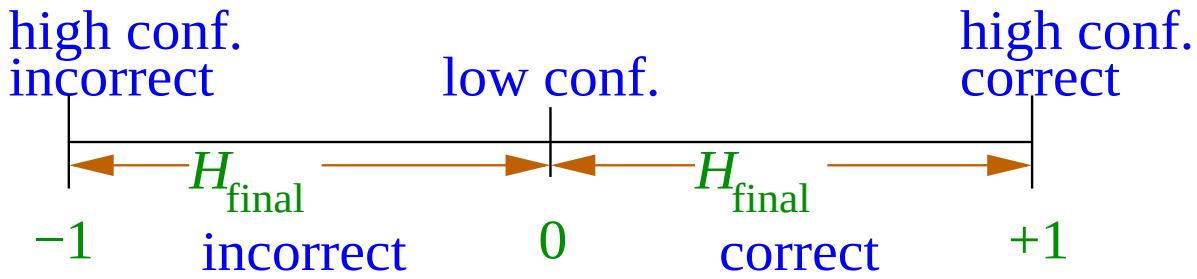
	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

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

A Better Story: The Margins Explanation

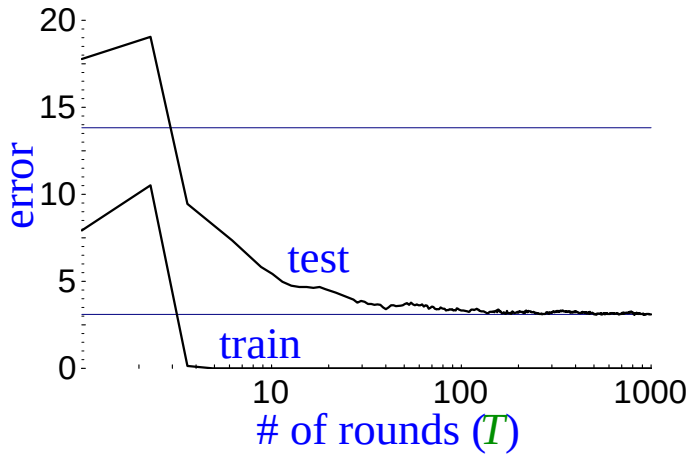
[with Freund, Bartlett & Lee]

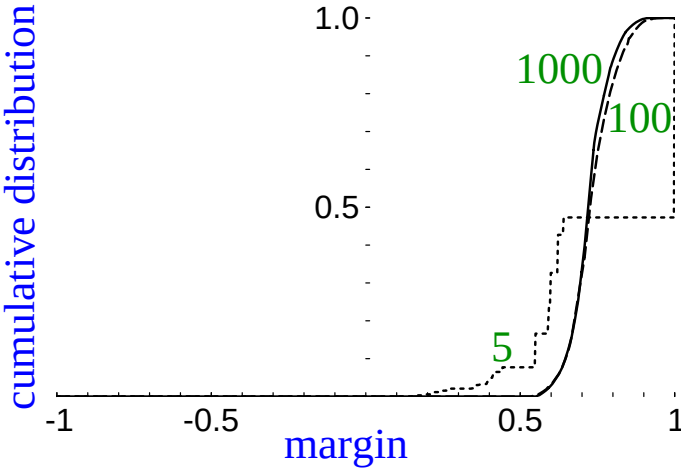
- key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications
- recall: H_{final} is weighted majority vote of weak classifiers
- measure confidence by margin = strength of the vote
  = (weighted fraction voting correctly)
  −(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)
- Theorem: boosting tends to increase margins of training examples (given weak learning assumption)
  - moreover, larger edges $\Rightarrow$ larger 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

More Theory

- many other ways of understanding AdaBoost:
  - as playing a repeated two-person matrix game
    - weak learning assumption and optimal margin have natural game-theoretic interpretations
    - special case of more general game-playing algorithm
  - as a method for minimizing a particular loss function via numerical techniques, such as coordinate descent
  - using convex analysis in an "information-geometric" framework that includes logistic regression and maximum entropy
  - as a universally consistent statistical method
- can also derive optimal boosting algorithm, and extend to continuous time

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
  - $\longrightarrow$ 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

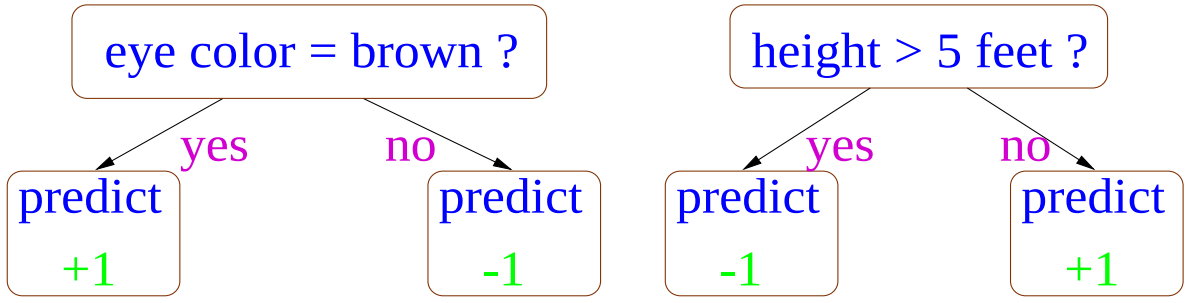
Caveats

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

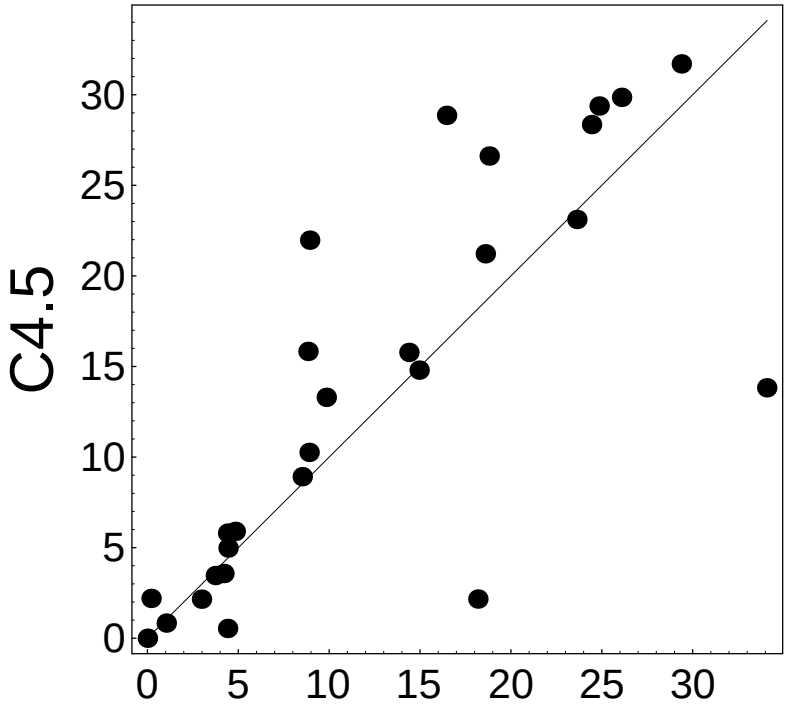
UCI Experiments

[with Freund]

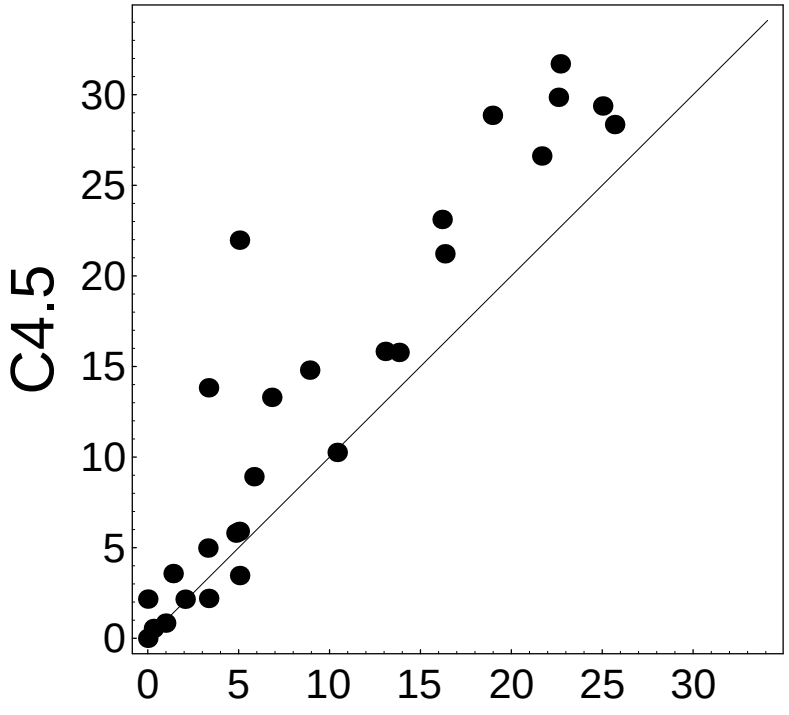
- tested AdaBoost on UCI benchmarks
- used:
  - C4.5 (Quinlan's decision tree algorithm)
  - "decision stumps": very simple rules of thumb that test on single attributes



UCI Results



boosting Stumps

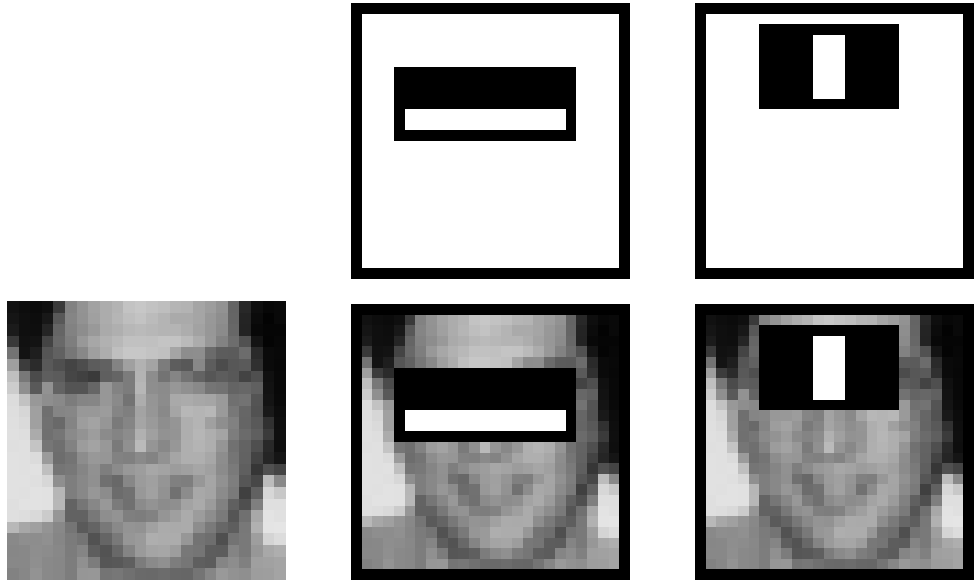


boosting C4.5

Application: Detecting Faces

[Viola & Jones]

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



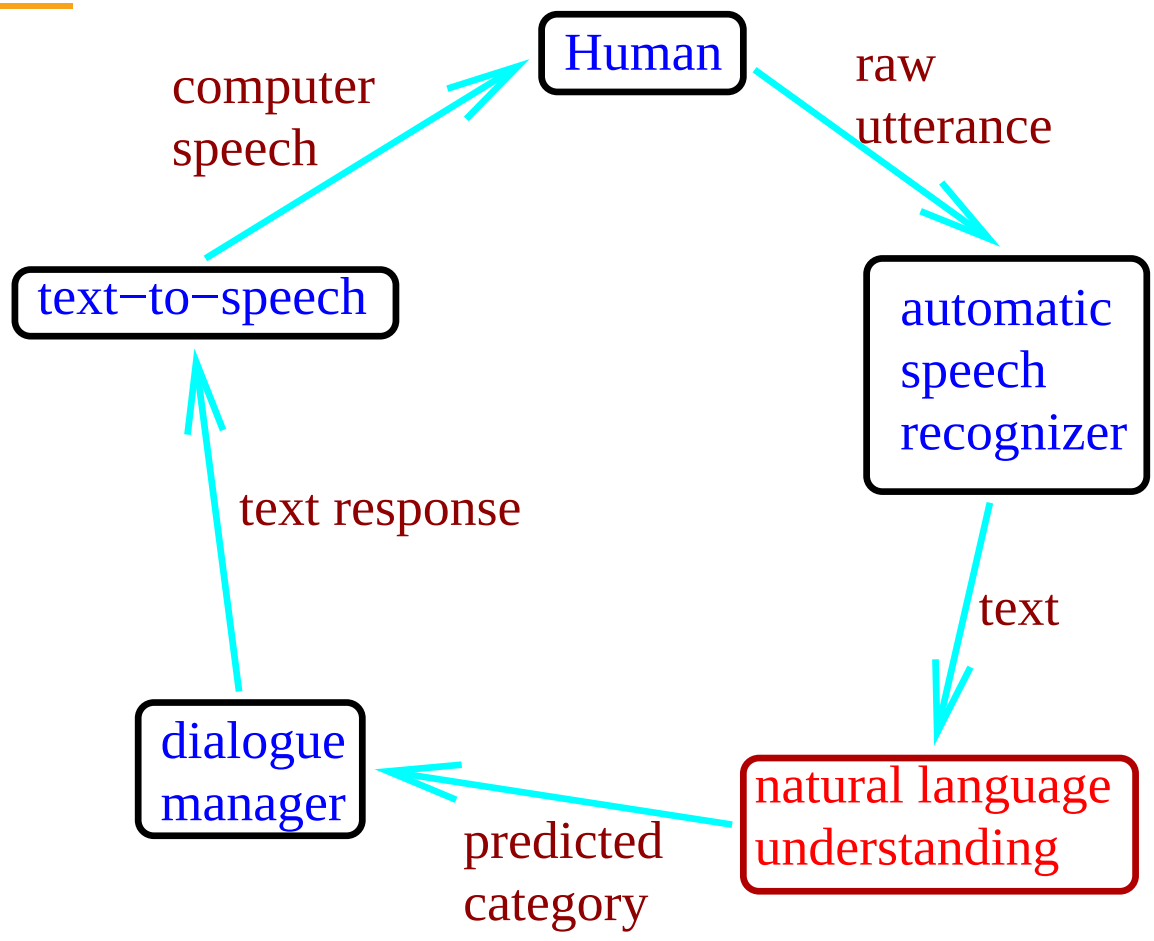
- many clever tricks to make extremely fast and accurate

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