
Machine Learning

Boosting

(based on Rob Schapire's IJCAI'99 talk and slides by B. Pardo)

Horse Race Prediction



How to Make \$\$\$ In Horse Races?

- Ask a professional.
- Suppose:
 - Professional cannot give single highly accurate rule
 - ...but presented with a set of races, can always generate better-than-random rules
- Can you get rich?

Idea

- 1) Ask expert for rule-of-thumb
 - 2) Assemble set of cases where rule-of-thumb fails (hard cases)
 - 3) Ask expert for a rule-of-thumb to deal with the hard cases
 - 4) Goto Step 2
- Combine all rules-of-thumb
 - Expert could be “weak” learning algorithm

Questions

- How to choose races on each round?
 - concentrate on “hardest” races
(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

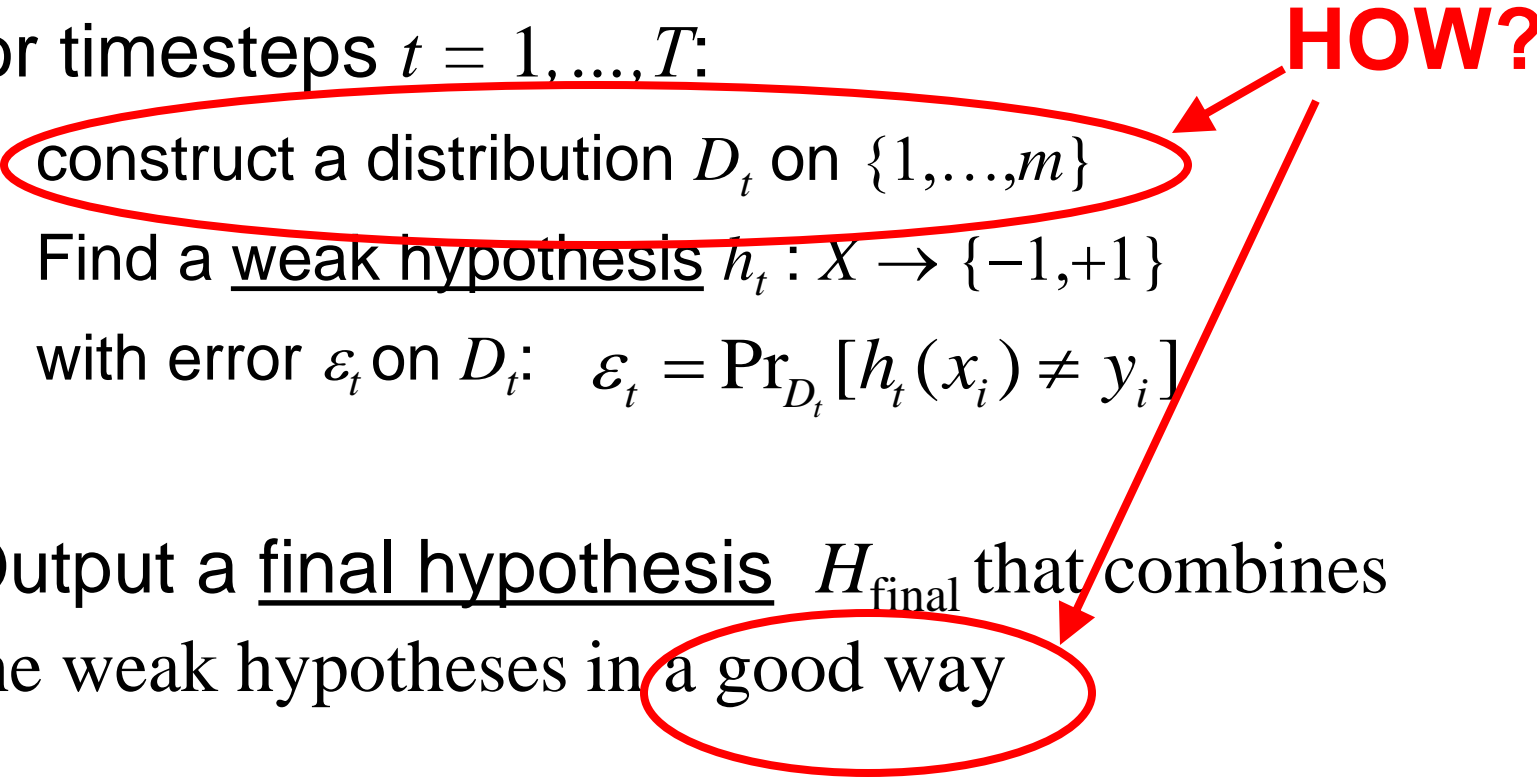
Boosting

- boosting = general method of converting rough rules of thumb into highly accurate prediction rule
- more technically:
 - given “weak” learning algorithm that can consistently find hypothesis (classifier) with error $\leq 1/2 - \gamma$
 - a boosting algorithm can provably construct single hypothesis with error $\leq \epsilon$

This Lecture

- Introduction to boosting (AdaBoost)
- Analysis of training error
- Analysis of generalization error based on theory of margins
- Extensions
- Experiments

A Formal View of Boosting

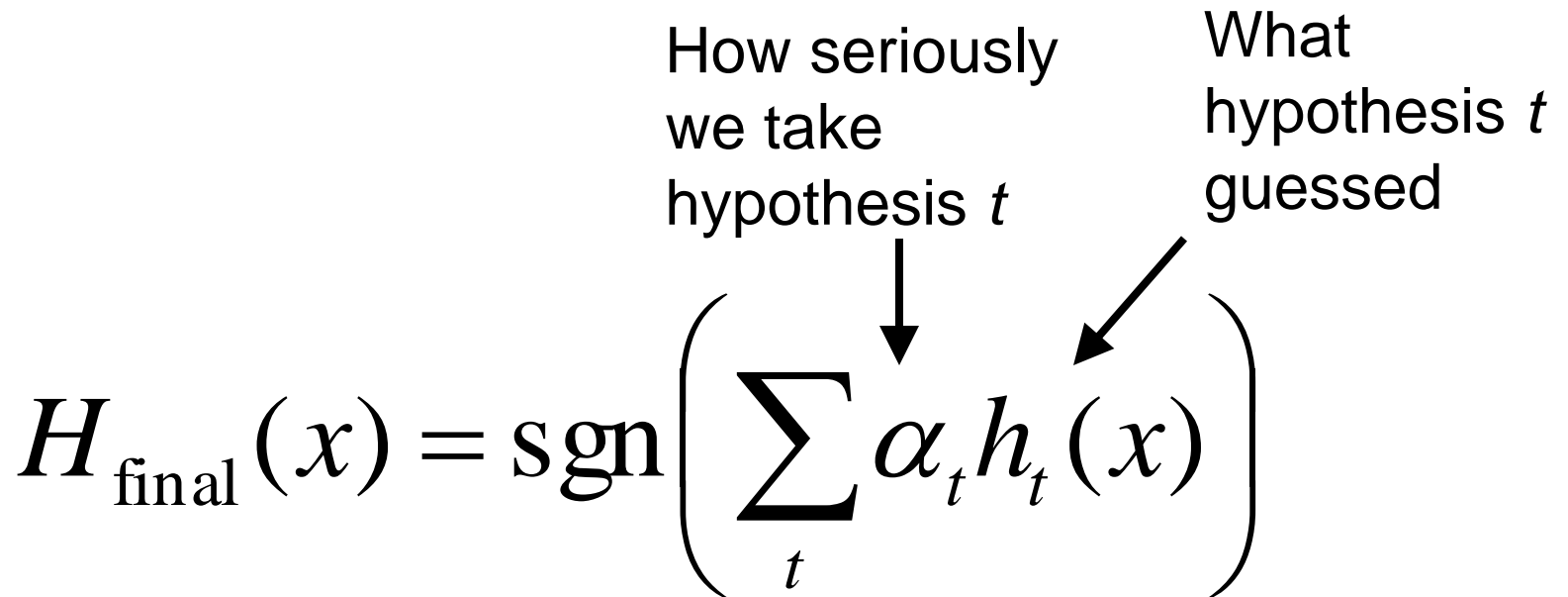
- Given training set $X = \{(x_1, y_1), \dots, (x_m, y_m)\}$
 - $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
 - for timesteps $t = 1, \dots, T$:
 - construct a distribution D_t on $\{1, \dots, m\}$
 - Find a weak hypothesis $h_t : X \rightarrow \{-1, +1\}$
with error ε_t on D_t : $\varepsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$
 - Output a final hypothesis H_{final} that combines the weak hypotheses in a good way
- 

Weighting the Votes

- H_{final} is a weighted combination of the choices from all our hypotheses.

How seriously we take hypothesis t


What hypothesis t guessed

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


The Hypothesis Weight

- α_t determines how “seriously” we take this particular classifier’s answer

The error on training distribution D_t


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

The Training Distribution

- D_t determines which elements in the training set we focus on.

$$D_1(i) = \frac{1}{m}$$

Size of the training set

The right answer What we guessed

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

Normalization factor

The Hypothesis Weight

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

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

AdaBoost [Freund&Schapire '97]

- constructing D_t :

- $D_1(i) = \frac{1}{m}$

- given D_t and h_t :

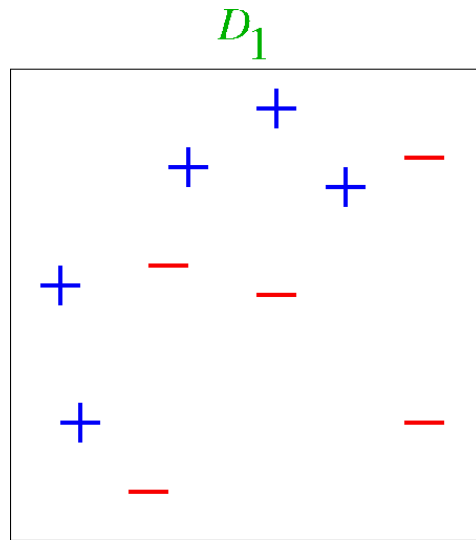
$$D_{t+1} = \frac{D_t}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$
$$= \frac{D_t}{Z_t} \cdot \exp(-\alpha_t \cdot y_i \cdot h_t(x_i))$$

where: Z_t = normalization constant

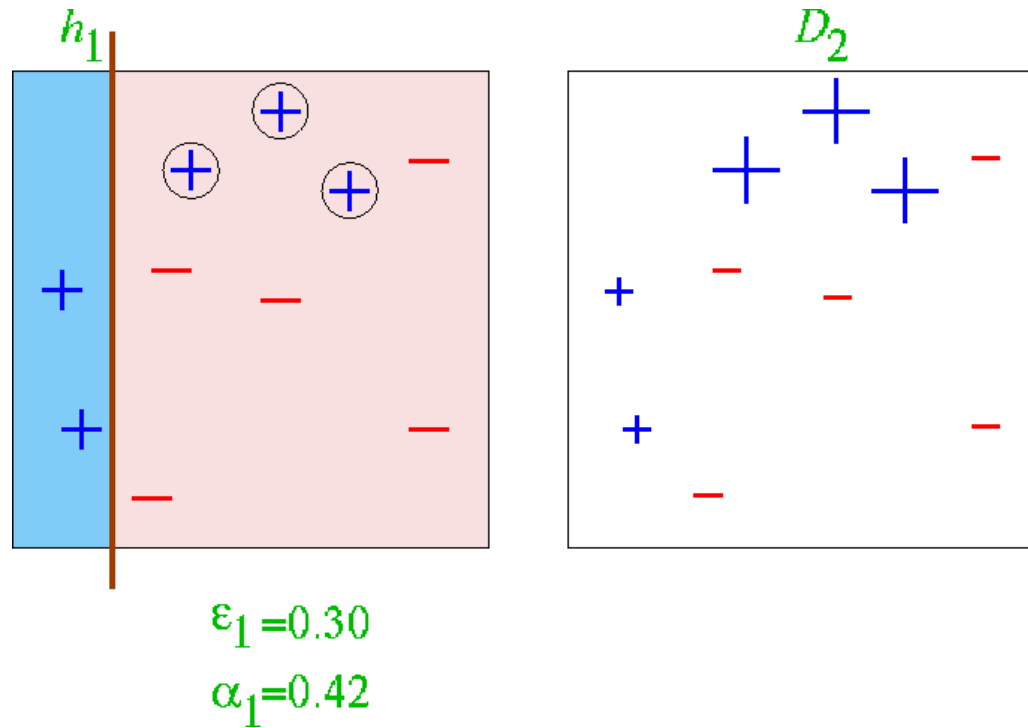
$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right) > 0$$

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

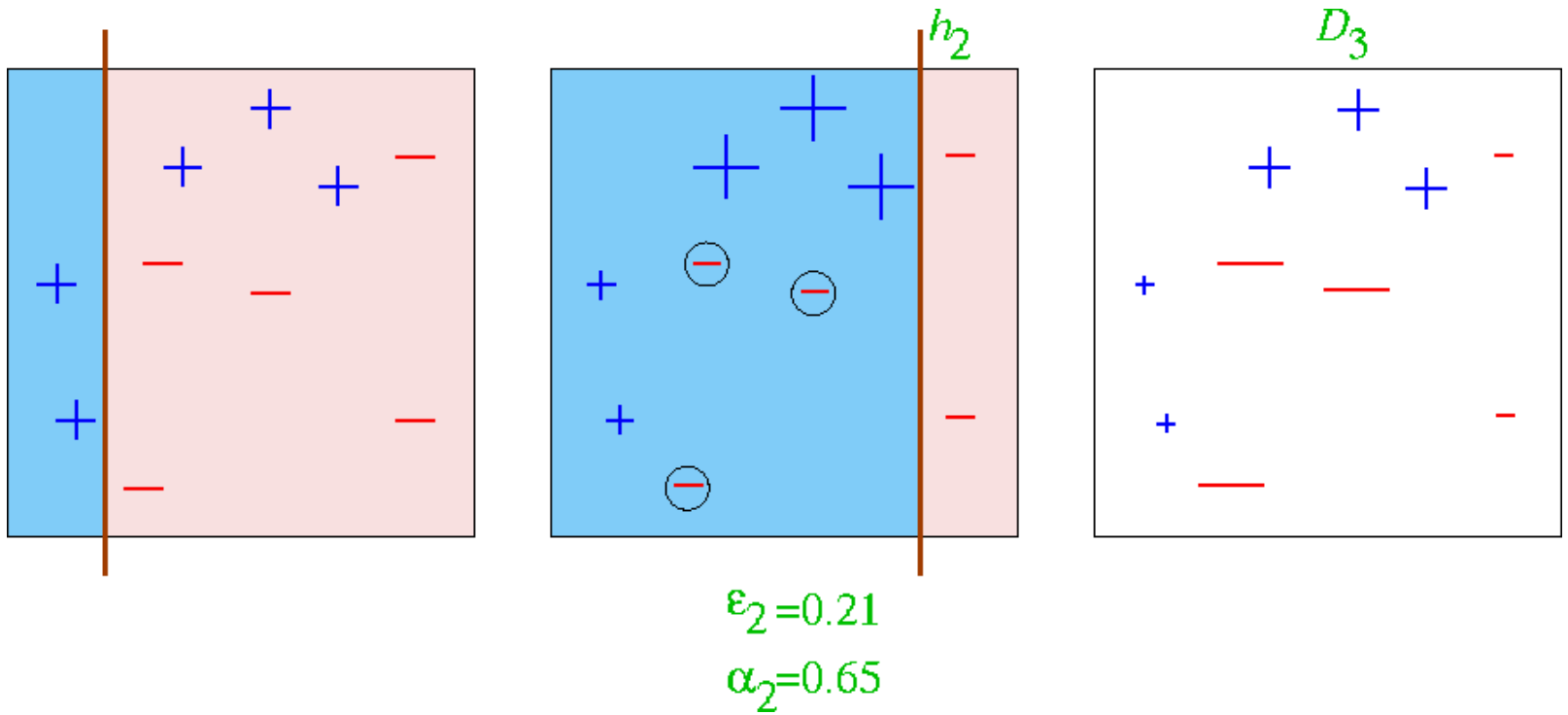
Toy Example



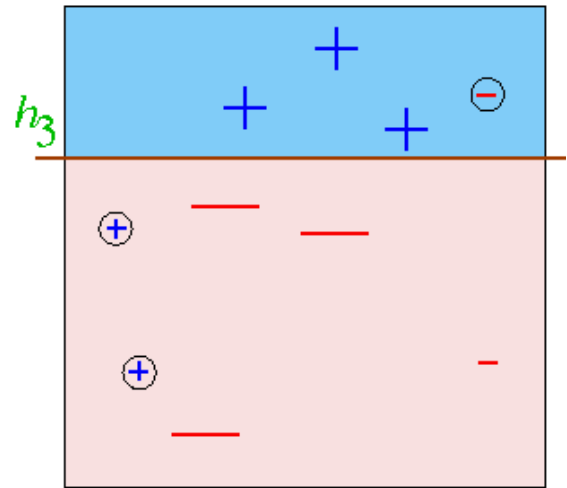
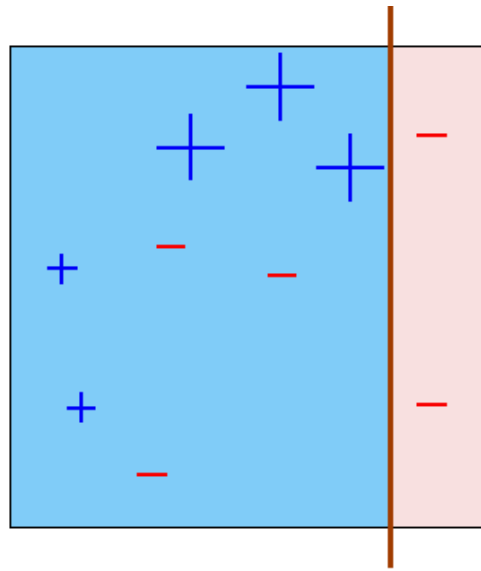
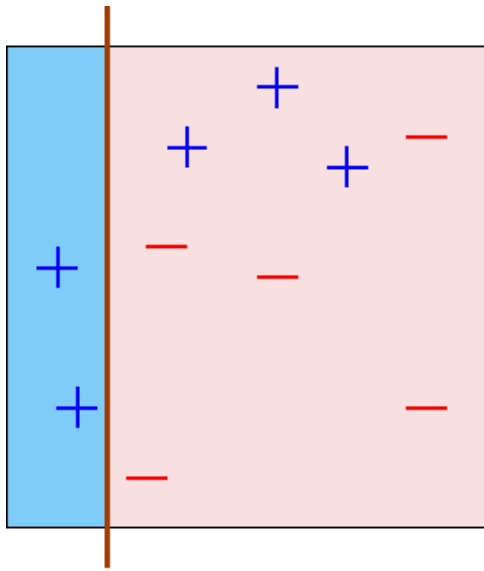
Round 1



Round 2



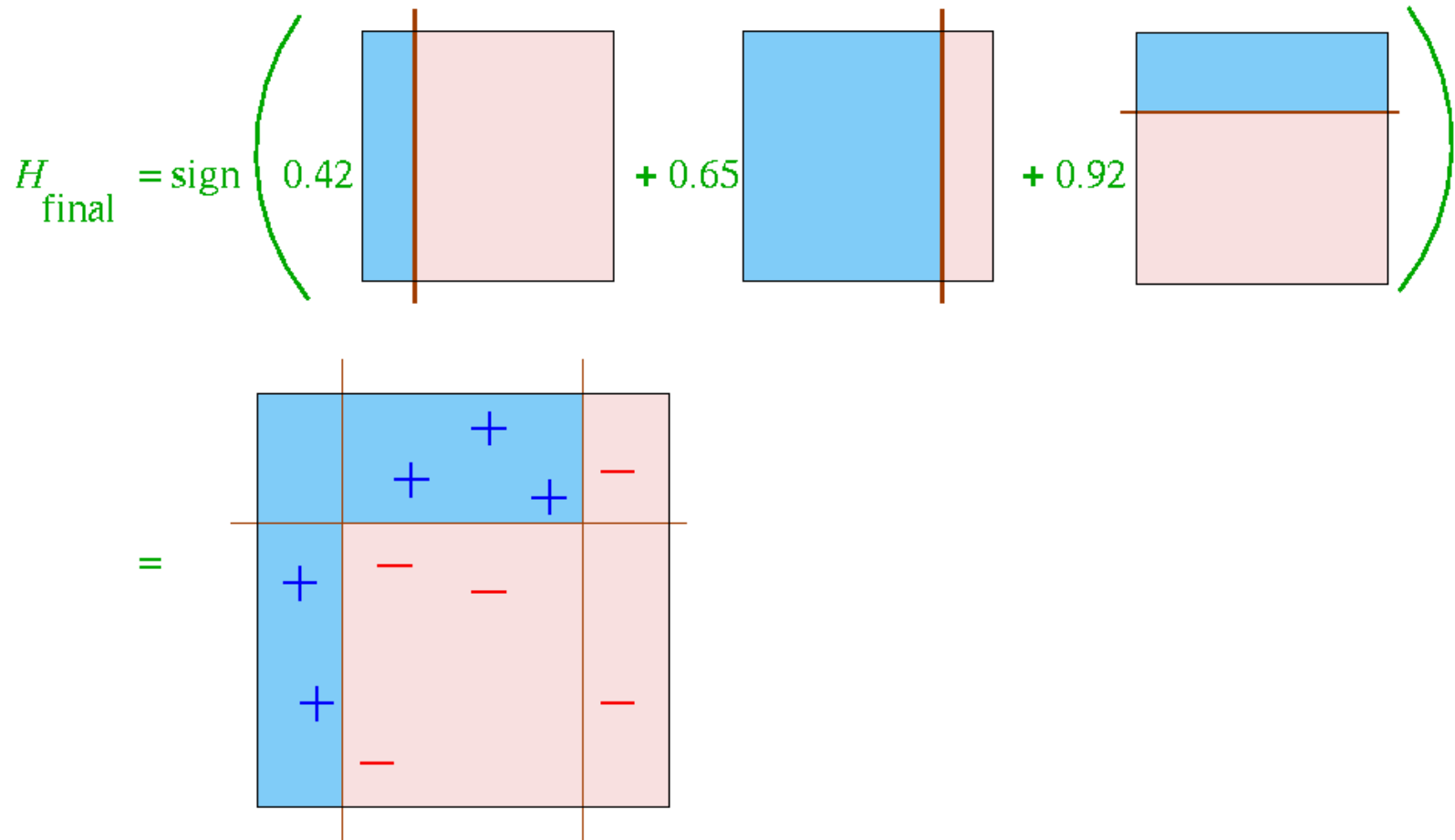
Round 3



$$\epsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

Final Hypothesis



Analyzing the Training Error

- Theorem [Freund&Schapire '97]:

write ε_t as $\frac{1}{2} - \gamma_t$

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

so if $\forall t: \gamma_t \geq \gamma > 0$ then

then, training error(H_{final}) $\leq e^{-2\gamma^2 T}$

Analyzing the Training Error

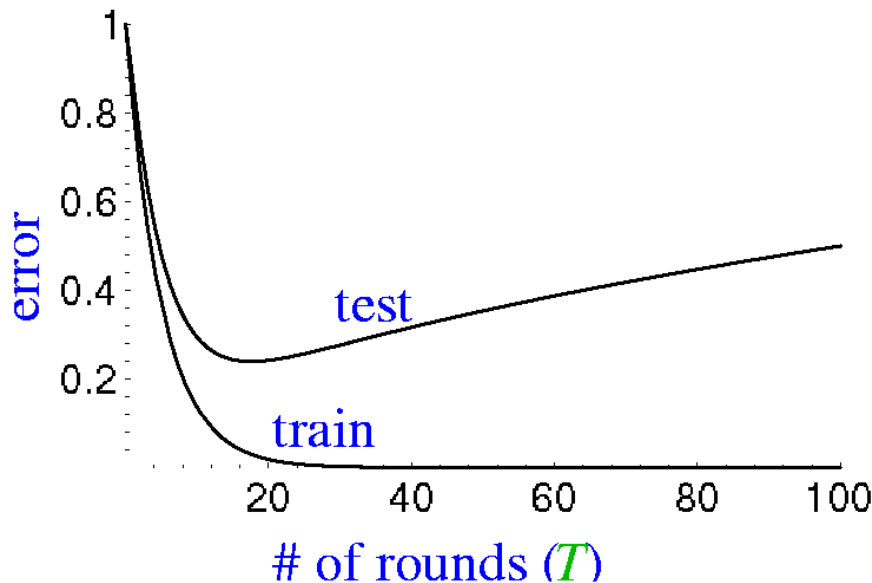
So what? This means AdaBoost is adaptive:

- does not need to know γ or T a priori
- Works as long as $\gamma_t > 0$

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
then x_i must have “large” weight under final dist. D_{T+1}
- since total weight ≤ 1 :
number of incorrectly classified examples “small”

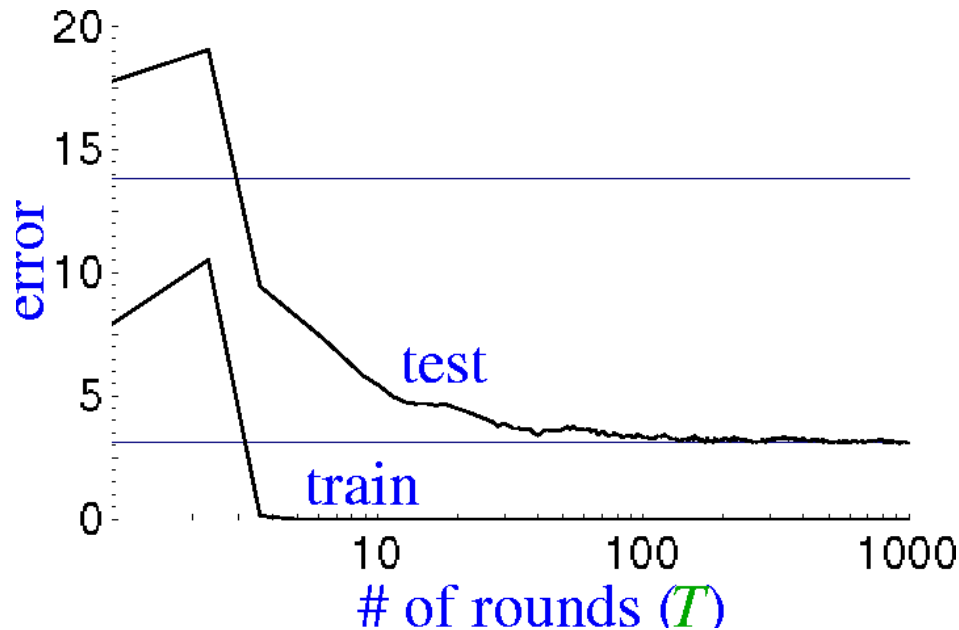
Analyzing Generalization Error



we expect:

- training error to continue to drop (or reach zero)
- test error to increase when H_{final} becomes “too complex” (Occam’s razor)

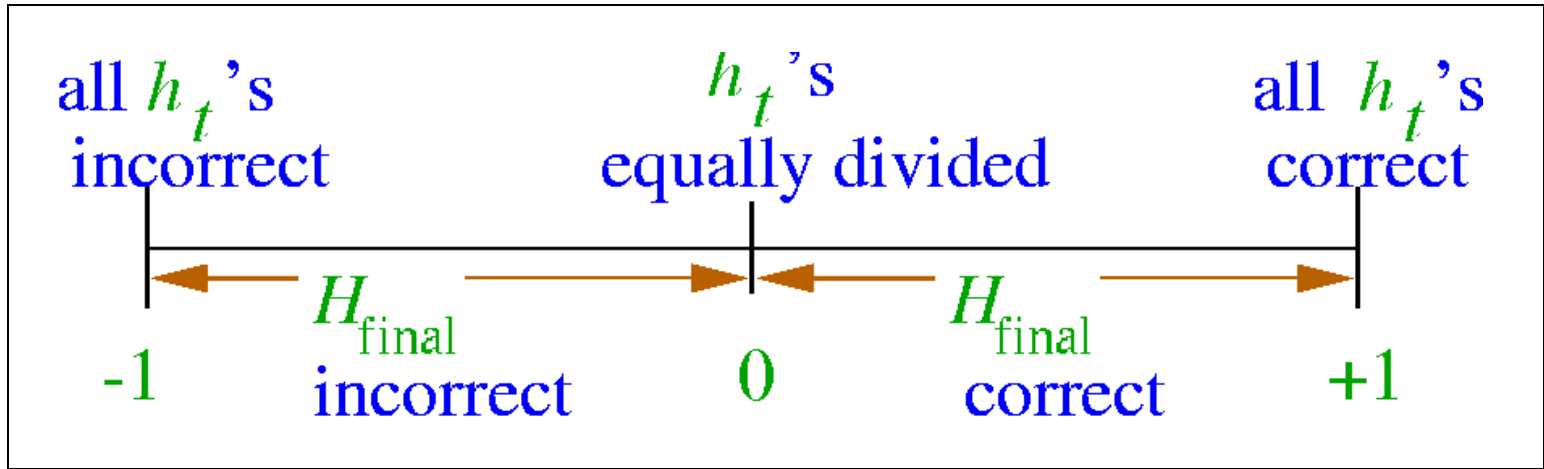
A Typical Run



(boosting on C4.5 on
“letter” dataset)

- Test error does not increase even after 1,000 rounds (~2,000,000 nodes)
- Test error continues to drop after training error is zero!
- Occam’s razor wrongly predicts “simpler” rule is better.

A Better Story: Margins



Key idea: Consider confidence (margin):

- with

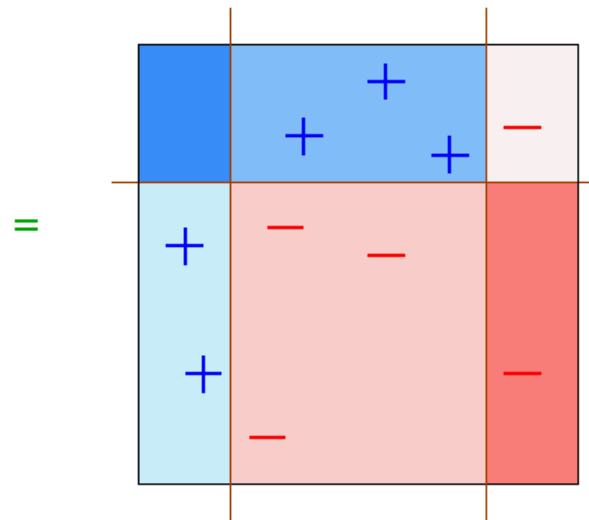
$$H_{\text{final}}(x) = \text{sgn}(f(x)) \quad f(x) = \frac{\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \in [-1, 1]$$

- define: margin of $(x, y) = y \cdot f(x)$

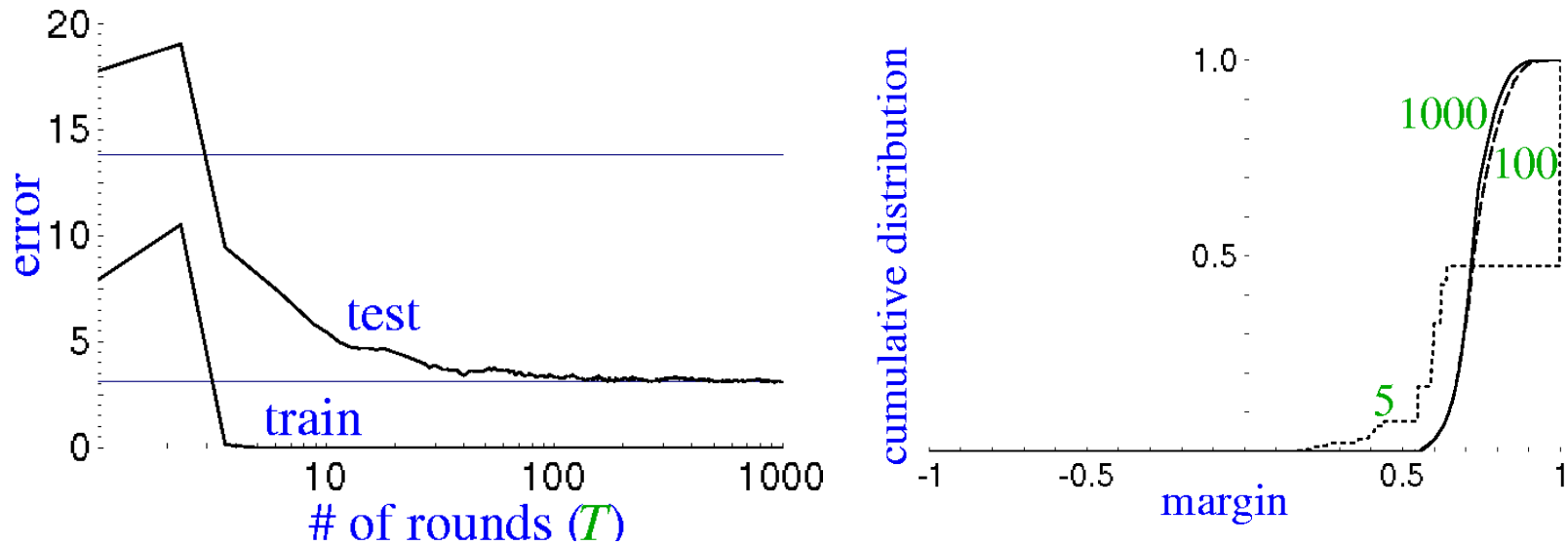
Margins for Toy Example

$$f = \left(\begin{array}{c} 0.42 \\ + 0.65 \\ + 0.92 \end{array} \right)$$

$/ (0.42 + 0.65 + 0.92)$



The Margin Distribution



epoch	5	100	1000
training 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

Boosting Maximizes Margins

- Can be shown to minimize

$$\sum_i e^{-y_i f(x_i)} = \sum_i e^{-y_i \sum_t \alpha_t h_t(x_i)}$$

\propto to margin of (x_i, y_i)



Analyzing Boosting Using Margins

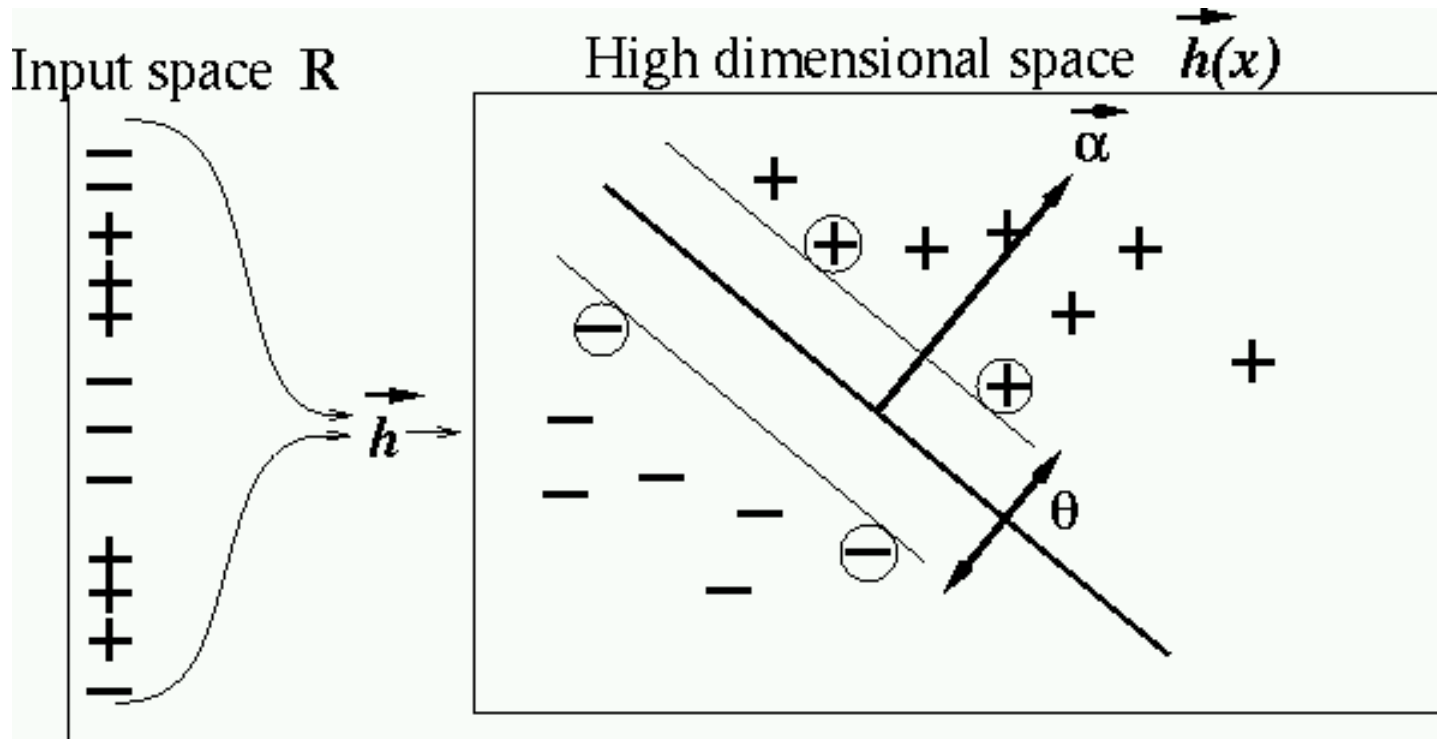
generalization error bounded by function of training sample margins:

$$\text{error} \leq \hat{\Pr}[\text{margin}_f(x, y) \leq \theta] + \tilde{O}\left(\sqrt{\frac{\text{VC}(H)}{m\theta^2}}\right)$$

- larger margin \Rightarrow better bound
- bound independent on # of epochs
- boosting tends to increase margins of training examples by concentrating on those with smallest margin

Relation to SVMs

SVM: map x into high-dim space, separate data linearly



Relation to SVMs (cont.)

$$H(x) = \begin{cases} +1 & \text{if } 2x^5 - 5x^2 + x > 10 \\ -1 & \text{otherwise} \end{cases}$$

$$\vec{h}(x) = (1, x, x^2, x^3, x^4, x^5)$$

$$\vec{\alpha} = (-10, 1, -5, 0, 0, 2)$$

$$H(x) = \begin{cases} +1 & \text{if } \vec{\alpha} \cdot \vec{h}(x) > 0 \\ -1 & \text{otherwise} \end{cases}$$

Relation to SVMs

- Both maximize margins:

$$\theta \doteq \max_w \min_i \frac{(\vec{\alpha} \cdot \vec{h}(x_i)) y_i}{\|\vec{\alpha}\|}$$

- SVM: $\|\vec{\alpha}\|_2$ Euclidean norm (L_2)
- AdaBoost: $\|\vec{\alpha}\|_1$ Manhattan norm (L_1)
- Has implications for optimization, PAC bounds

See [Freund et al '98] for details

Extensions: Multiclass Problems

- 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?”
 - “does or does not example x belong to class 3?”

•
•
•

Extensions: Confidences and Probabilities

- Prediction of hypothesis h_t : $\text{sgn}(h_t(x))$
- Confidence of hypothesis h_t : $|h_t(x)|$
- Probability of H_{final} : $\Pr_f[y = +1 | x] = \frac{e^{f(x)}}{e^{f(x)} + e^{-f(x)}}$

[Schapire&Singer '98], [Friedman, Hastie & Tibshirani '98]

Practical Advantages of AdaBoost

- (quite) fast
- simple + easy to program
- only a single parameter to tune (T)
- no prior knowledge
- flexible: can be combined with any classifier (neural net, C4.5, ...)
- provably effective (assuming weak learner)
 - shift in mind set: goal now is merely to find hypotheses that are better than random guessing
- finds outliers

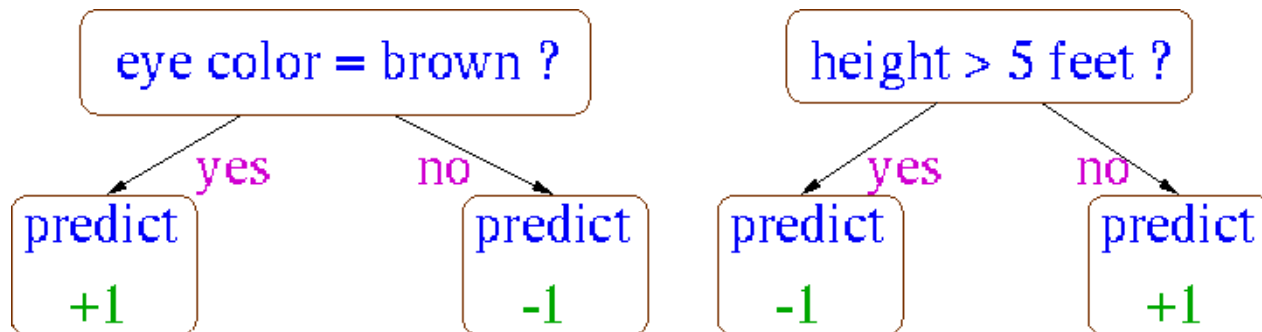
Caveats

- performance depends on data & weak learner
- AdaBoost can fail if
 - weak hypothesis too complex (overfitting)
 - weak hypothesis too weak ($\gamma_t \rightarrow 0$ too quickly),
 - underfitting
 - Low margins \rightarrow overfitting
- empirically, AdaBoost seems susceptible to noise

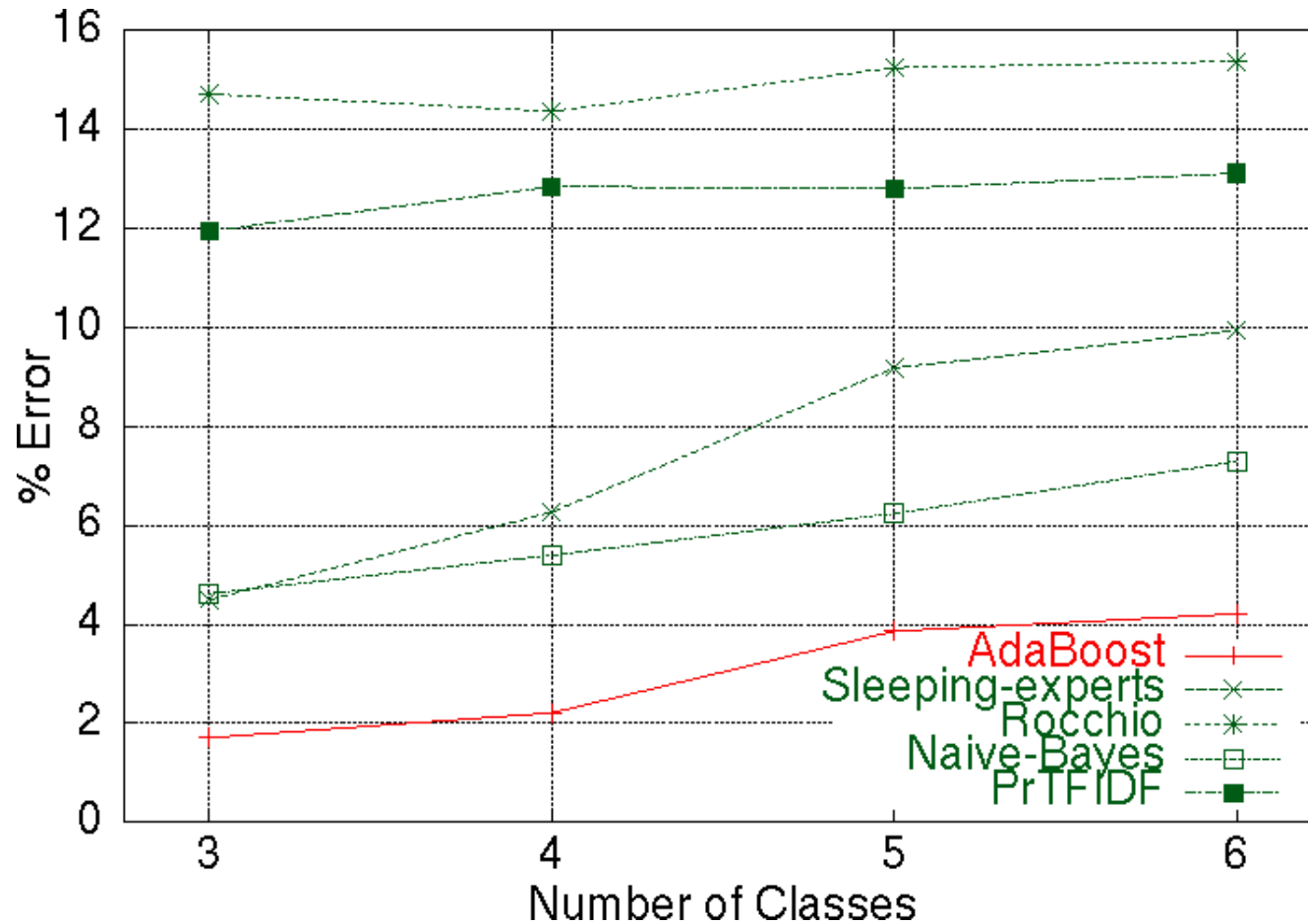
UCI Benchmarks

Comparison with

- C4.5 (Quinlan's Decision Tree Algorithm)
- Decision Stumps (only single attribute)



Text Categorization



database: Reuters

Conclusion

- boosting useful tool for classification problems
 - grounded in rich theory
 - performs well experimentally
 - often (but not always) resistant to overfitting
 - many applications
- but
 - slower classifiers
 - result less comprehensible
 - sometime susceptible to noise

Other Ensembles

- Bagging
- Stacking

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 using AdaBoost:

[Drucker&Cortes '95]	[Schapire&Singer '98]
[Jackson&Cravon '96]	[Maclin&Opitz '97]
[Freund&Schapire '96]	[Bauer&Kohavi '97]
[Quinlan '96]	[Schwenk&Bengio '98]
[Breiman '96]	[Dietterich'98]
- continuing development of theory & algorithms:

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