Ensemble Learning

Zhiyao Duan

EECS 349

Oct. 22, 2010

Boosting part is adapted from Robert Schapire's tutorial in 2005

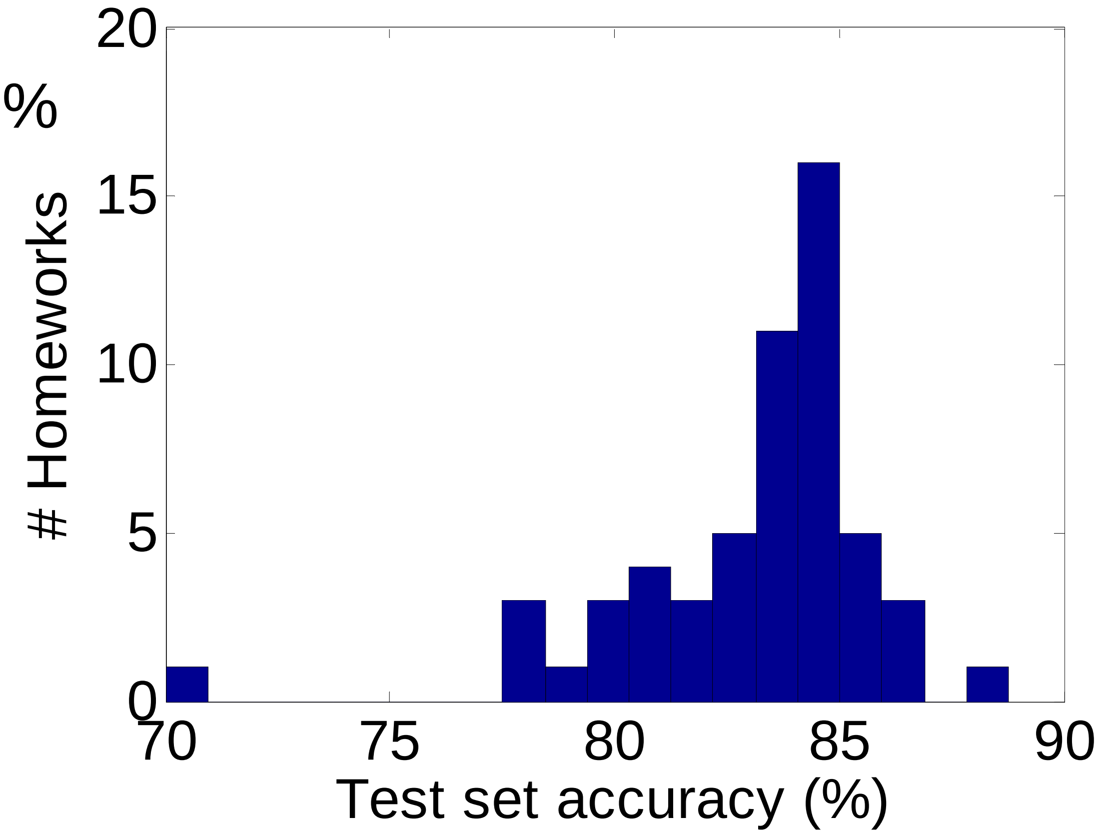
What is Ensemble Learning?

- Building a highly accurate classifier is difficult
- Building many not-so-accurate classifiers is easy
- Can we generate a single, highly accurate classifier from these not-so-accurate classifiers?

- Answer: Yes
- Why?
  - Many heads are better than one.
  - 三个臭皮匠，赛过诸葛亮。(Three Stooges, the top of *Zhuge Liang*.)

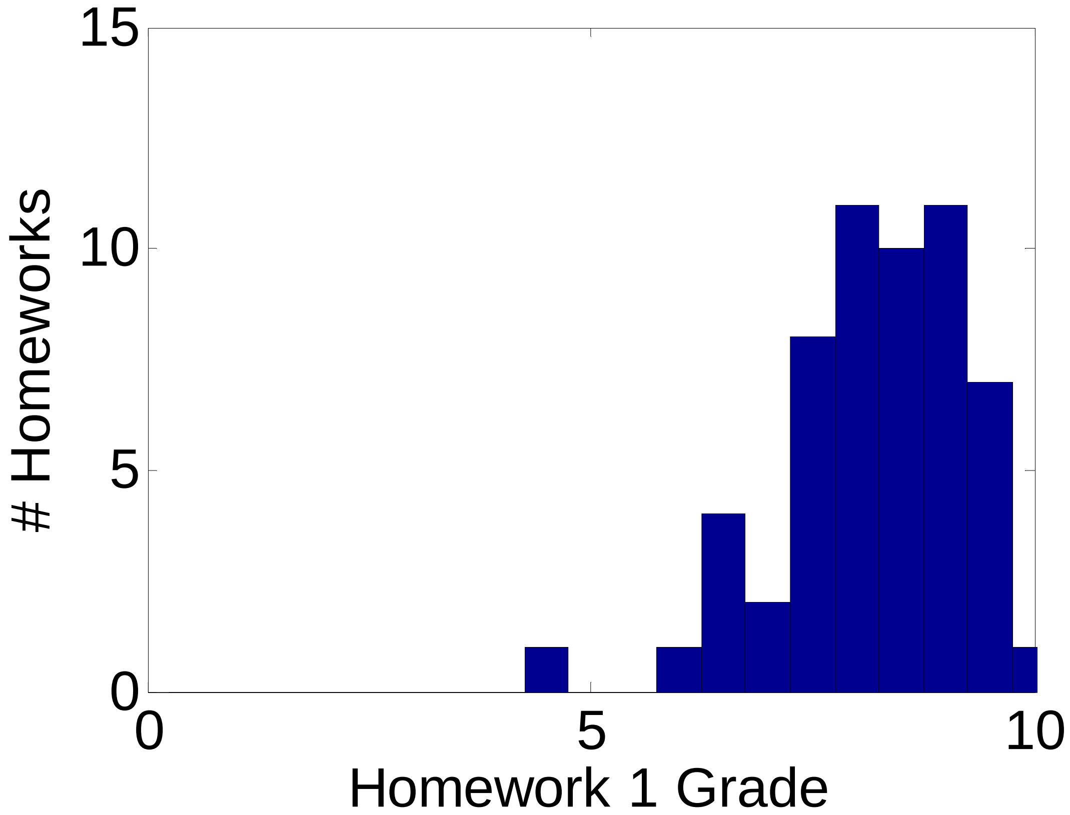
Another Evidence

- Homework 1 test set accuracy
  - Mean 83.07%
  - Min = 70.00%
  - Max = 88.77%
  - Median = 83.89%



BTW…

- Homework 1 grade
  - Mean: 8.20
  - Min: 4.5
  - Max: 10
  - Median: 8.50



General Steps

- 1. Build a number of weak classifiers from the training data
- 2. Each classifier predicts a classification label on a test instance
- 3. Combine these predicted labels to a single label as the final prediction for the test instance

- Question:
  - How to build these classifiers?
  - How to combine their predictions?

Outline

- Bagging
- Boosting
  - **AdaBoost**
- Random Subspace
- Random Forests

Bagging

- Bagging = **B**ootstrap **agg**regat**ing**
- [Breiman '96]

- "pull oneself over a fence by one's bootstraps"
- In statistics, boostrapping means "estimating properties of an estimator (e.g. variance) by measuring those properties when sampling from an approximation distribution."   ---- Wikipedia

Bagging

- Given L = {m training instances}
- For i = 1 to T
  - Sample m instances with replacement from L to form a new training set L_i
  - Train a classifier using L_i
- For a test instance, output a prediction label by majority vote of the T classifiers.

- Note: classifiers constructed in each round are independent with each other

Properties of Bagging

- Improves accuracy on unstable classification algorithms
  - A algorithm is said unstable if perturbing the training set can significantly change the classifier it constructs
  - E.g. C4.5, neural nets, linear regression, etc.
- Cannot improve accuracy on stable algorithms
  - E.g. Nearest neighbors, etc.

- “Bagging goes a ways toward making a silk purse out of a sow’s ear, especially if the sow’s ear is twitchy.” ---- L. Breiman

Boosting

- Construct a classifier using a weak learning algorithm based on previous classifiers
  - Create a training set which weights more on the "hardest" examples (those most often misclassified by previous classifiers)
  - Combine classifiers by weighted majority vote, putting more weights on accurate classifiers
- Assumptions:
  - The weak learning algorithm can consistently find classifier with error $\leq 1/2-\gamma$
- Conclusion:
  - A boosting algorithm can provably construct a single classifier with arbitrarily small error

A Formal View of Boosting

- Given <u>training set</u> $X=\{(x_1,y_1),\ldots,(x_m,y_m)\}$
- $y_i \in \{-1,+1\}$ correct label of instance $x_i \in X$
- For $t = 1,\ldots,T$:
  - construct a distribution D_t on $\{1,\ldots,m\}$
  - Find a <u>weak classifier</u> $f_t : X \rightarrow \{-1,+1\}$

    with small error ε_t on D_t: $\varepsilon_t = \Pr_{D_t}[f_t(x_i) \neq y_i]$
- Output a <u>final classifier</u> f_{final} that combines the weak classifiers in a good way

HOW?

AdaBoost [Freund & Schapire '95]

- constructing D_t: Size of the training set
  - $D_1(i) = 1/m$
  - given D_t and f_t:

Correct label   Predicted label

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

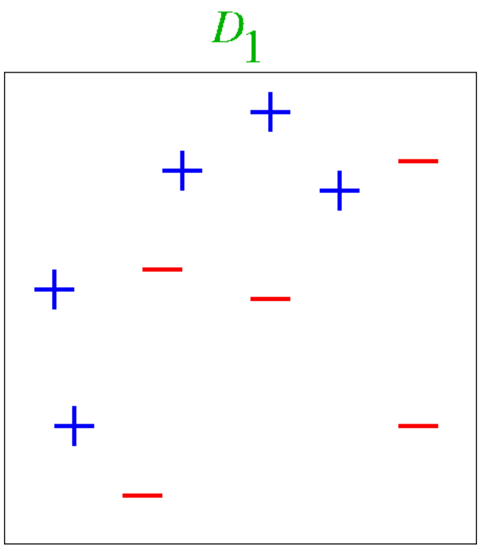
Normalization factor

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

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

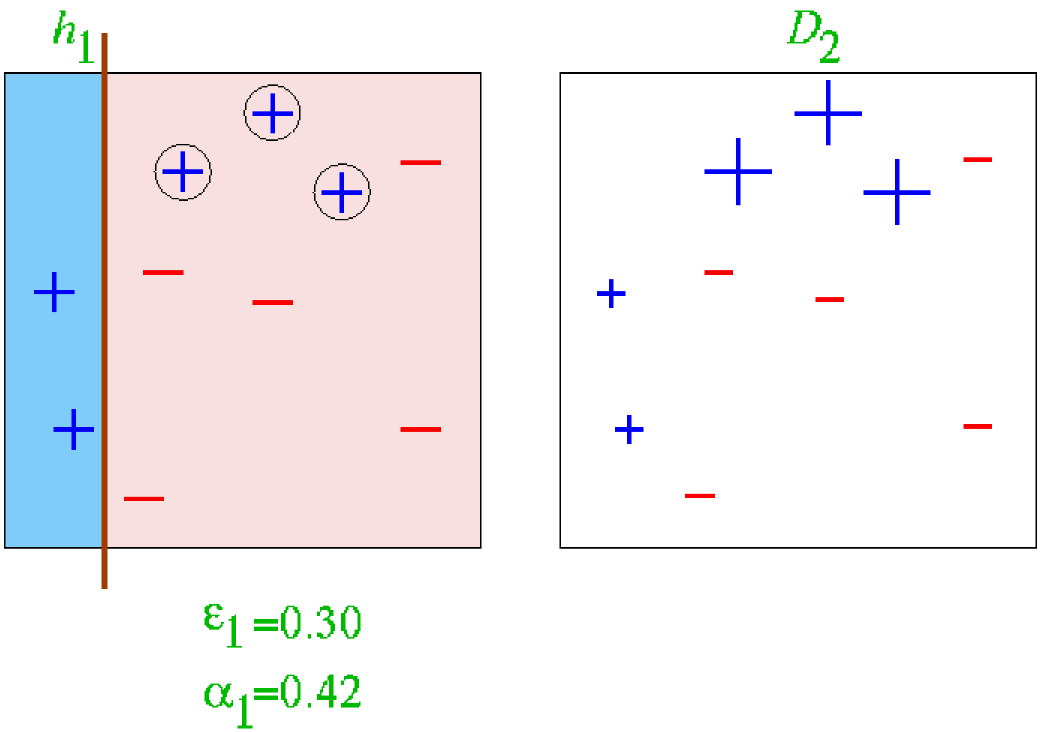
- final classifier: $f_{\text{final}}(x) = \text{sgn}\left(\sum_t \alpha_t f_t(x)\right)$

Toy Example

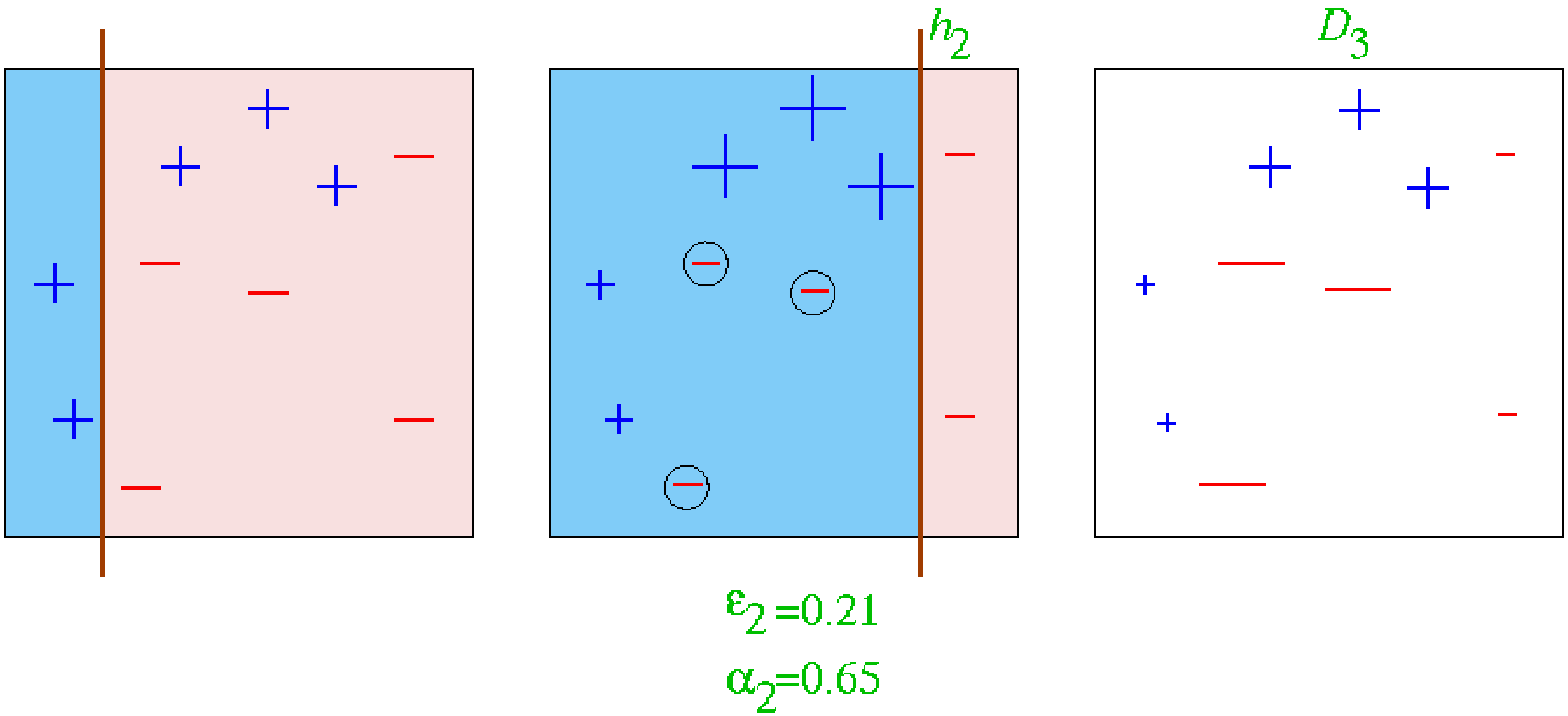


Weak classifiers: vertical or horizontal half planes

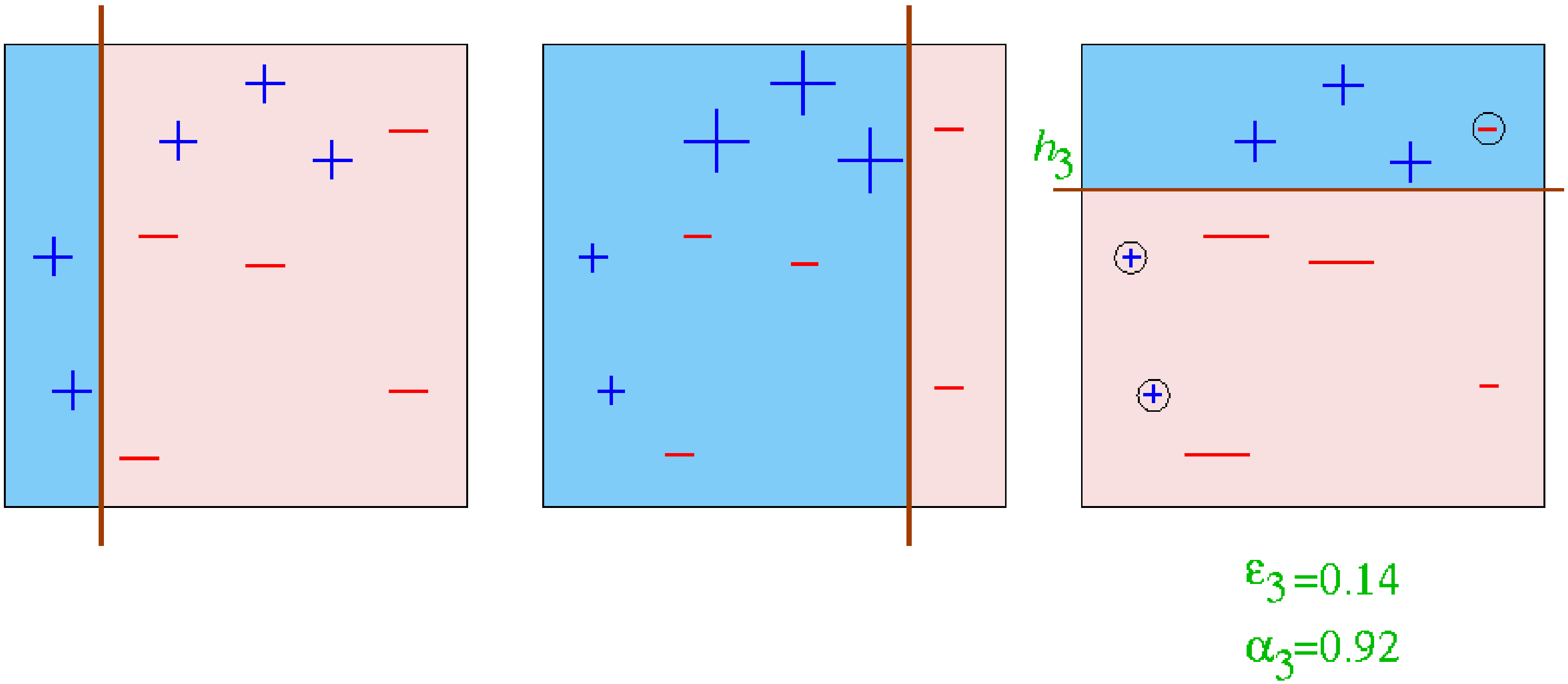
Round 1



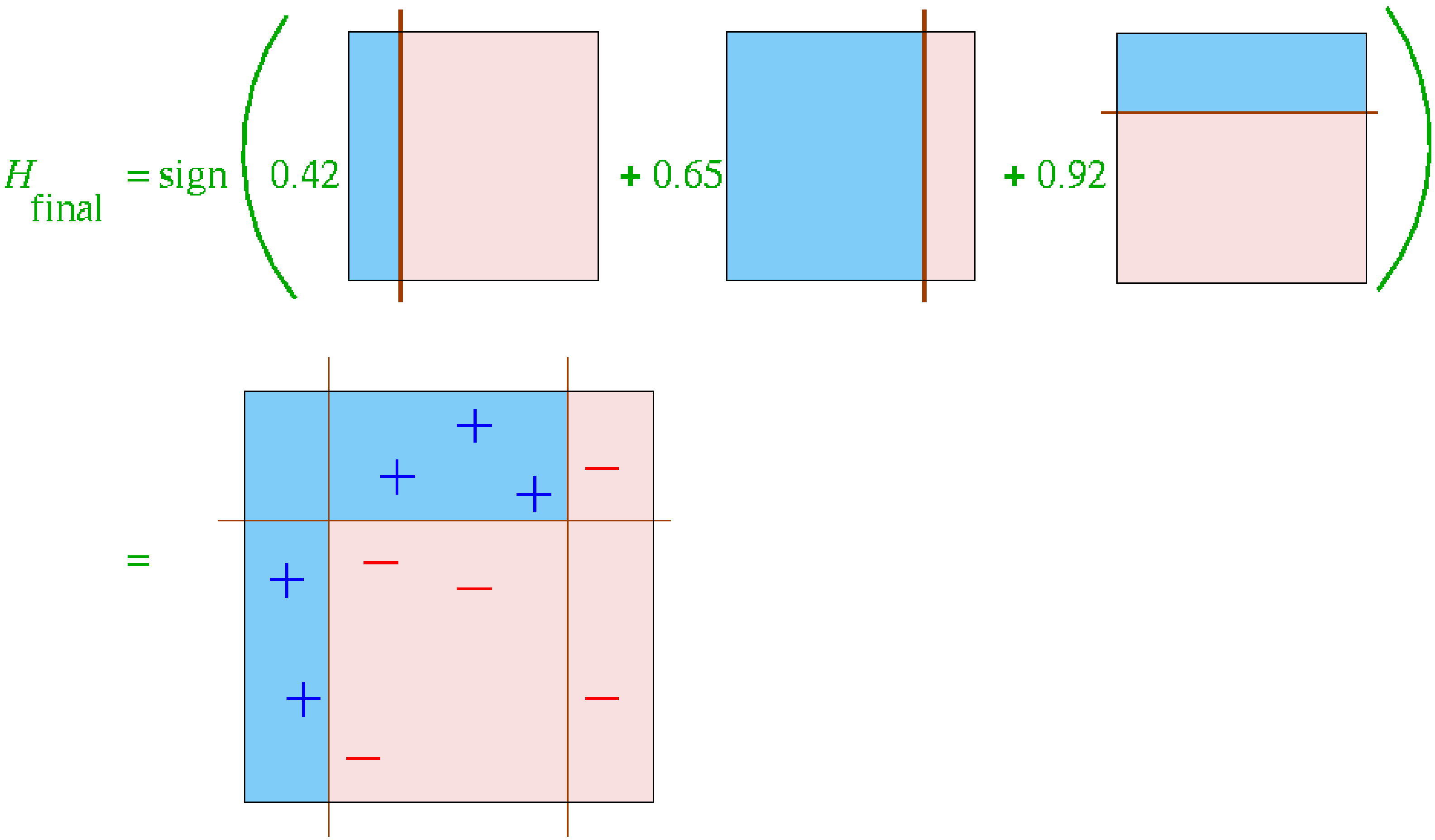
Round 2



Round 3



Final Classifier



Analyzing the Training Error

- Theorem [Freund&Schapire ’97]:
  - write ε_t as ½-γ_t
  - the $\text{training error}(f_{\text{final}}) \le \exp\left(-2\sum_t \gamma_t^2\right)$
- so if $\forall t$: $\gamma_t \ge \gamma > 0$ then

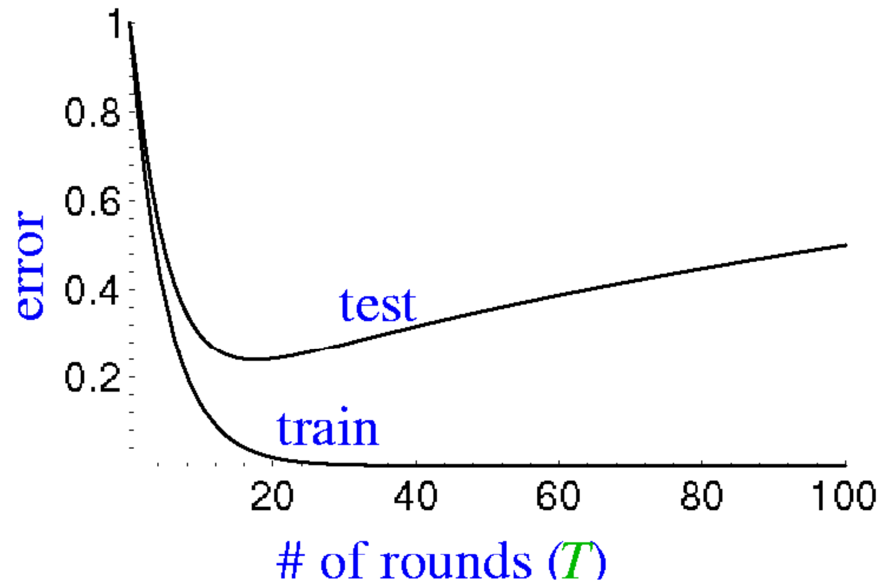
$$\text{training error}(f_{\text{final}}) \le \exp\left(-2\gamma^2 T\right)$$

- <u>Ada</u>Boost is <u>ada</u>ptive:
  - does not need to know γ or T a priori
  - can exploit $\gamma_t >> \gamma$

Proof

- Derive on the blackboard

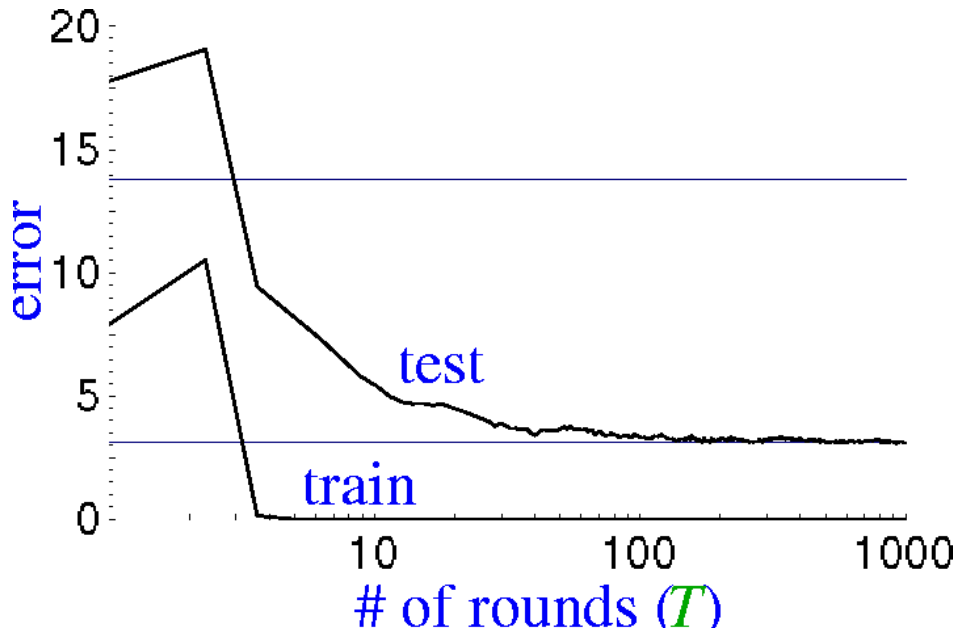
Guess the Test Error



We expect:

- training error to continue to drop (or reach zero)
- test error to <u>increase</u> when f_{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

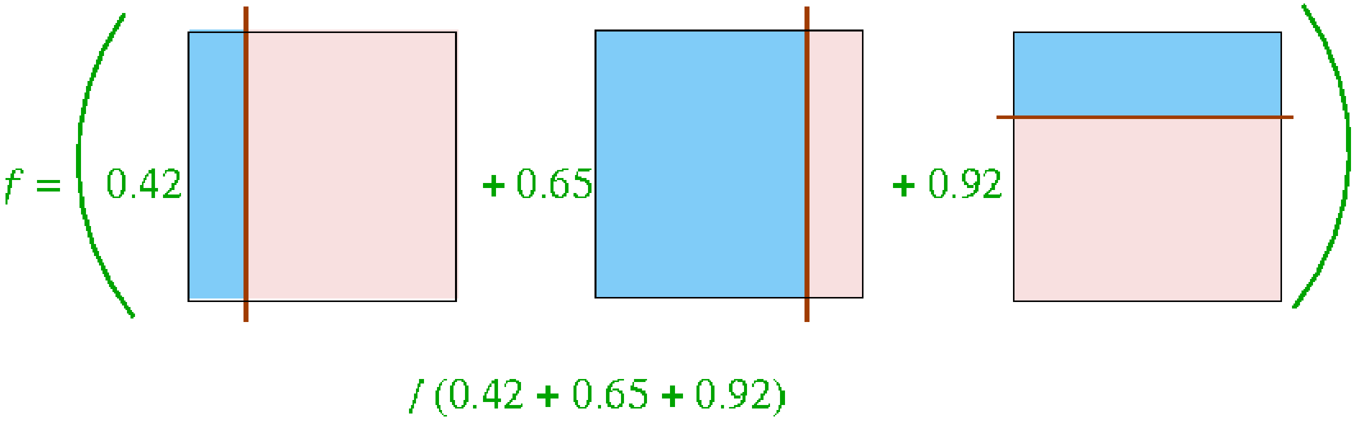
- Key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications

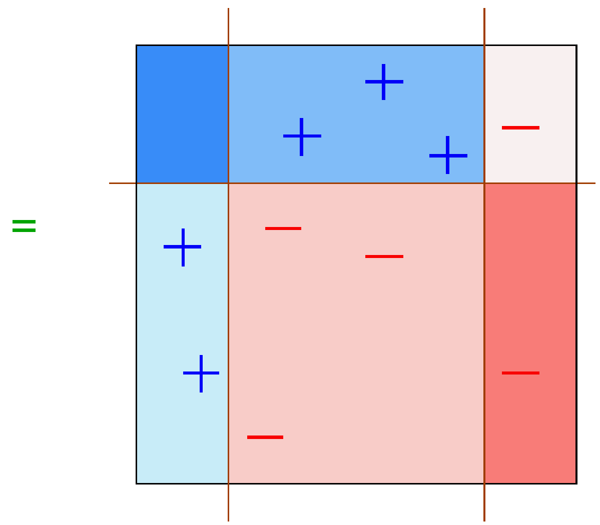
- Consider confidence (margin):

$$f_{\text{final}}(x) = \operatorname{sgn}(f(x)) \qquad f(x) = \frac{\sum_t \alpha_t f_t(x)}{\sum_t \alpha_t} \in [-1,1]$$

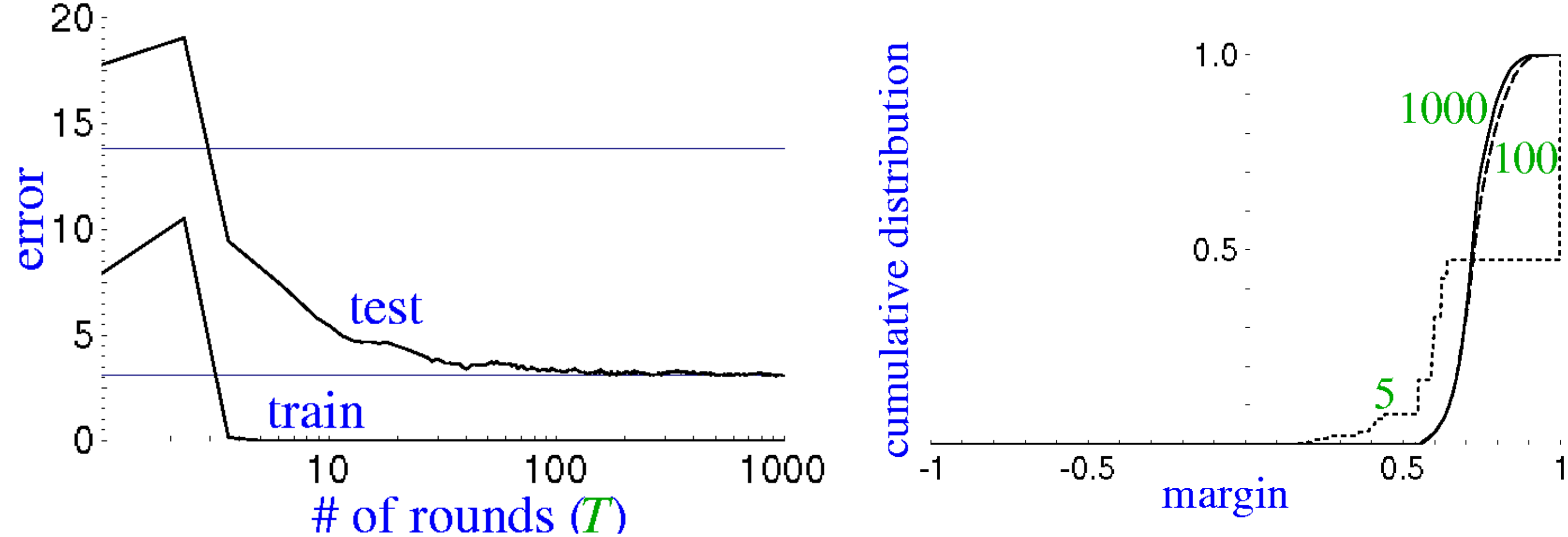
- Define: margin of $(x, y) = y \cdot f(x) \in [-1,1]$

Margins for Toy Example





The Margin Distribution



rounds	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

Analyzing Boosting Using Margins

- Theorem: boosting tends to increase margins of training examples
- 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
- Consequence: 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.

Practical Advantages of AdaBoost

- Simple + easy to program
- Flexible: can be combined with any classifier (neural net, C4.5, …)
- Only a single parameter to tune (T)
- No prior knowledge
- Provably effective (assuming weak learner)

Cons

- AdaBoost can fail if
  - weak classifier too complex (overfitting)
  - weak classifier is too weak ($\gamma_t \to 0$ too quickly),
- Empirically, AdaBoost seems especially susceptible to noise

Resources for Boosting

- A demo of AdaBoost:

http://cseweb.ucsd.edu/~yfreund/adaboost/

- A website:

http://www.boosting.org/

- A good bibliography list:

http://www.cs.princeton.edu/~schapire/boost.html

A good viedo lecture:

http://videolectures.net/mlss05us_schapire_b/

Random Subspace

- [Ho '98]
- Create the training set in each round by randomly choosing a subset of all attributes, i.e. using a random subspace of the feature space.
- Train a decision tree using this training set
- Majority vote by these trees

Random Forrests

- [Breiman ‘01]
- Use decision tree as the weak classifier
- Generate a number of trees
- For each tree, randomly select a subset of features to determine splitting at each node
- Majority vote using all the trees with equal weights

Properties

- Pros
  - as accurate as Adaboost and sometimes better
  - relatively robust to outliers and noise.
  - faster than bagging or boosting.
  - simple and easily parallelized.

Summary

- Ensemble learning:
  - Combine weak classifiers to obtain a strong classifier
- Bagging, Boosting: sample training instances
- Random Subspace, Random Forrests: sample features
- AdaBoost
  - Error on the training set can be arbitrarily small (given enough data and enough rounds)
  - Often resistant to overfitting
  - Margins are increased with more rounds
  - Performs well experimentally
  - Suspicious to noise

Thank you!