Lecture 13

Oct-27-2007

Bagging

- Generate T **random sample** from training set by **bootstrapping**
- Learn a sequence of classifiers $h_1, h_2, \ldots, h_T$ from each of them, using base learner *L*
- To classify an unknown sample X, let each classifier predict.
- Take simple **majority vote** to make the final prediction.

Simple scheme, works well in many situations!

Bias/Variance for classifiers

- Bias arises when the classifier cannot represent the true function – that is, the classifier underfits the data

- Variance arises when the classifier overfits the data – minor variations in training set cause the classifier to overfit differently

- Clearly you would like to have a low bias and low variance classifier!
  - Typically, low bias classifiers (overfitting) have high variance
  - high bias classifiers (underfitting) have low variance
  - We have a trade-off

Effect of Algorithm Parameters on Bias and Variance

- k-nearest neighbor: increasing k typically increases bias and reduces variance
- decision trees of depth D: increasing D typically increases variance and reduces bias

Why does bagging work?

- Bagging takes the average of multiple models --- reduces the variance
- This suggests that bagging works the best with low bias and high variance classifiers

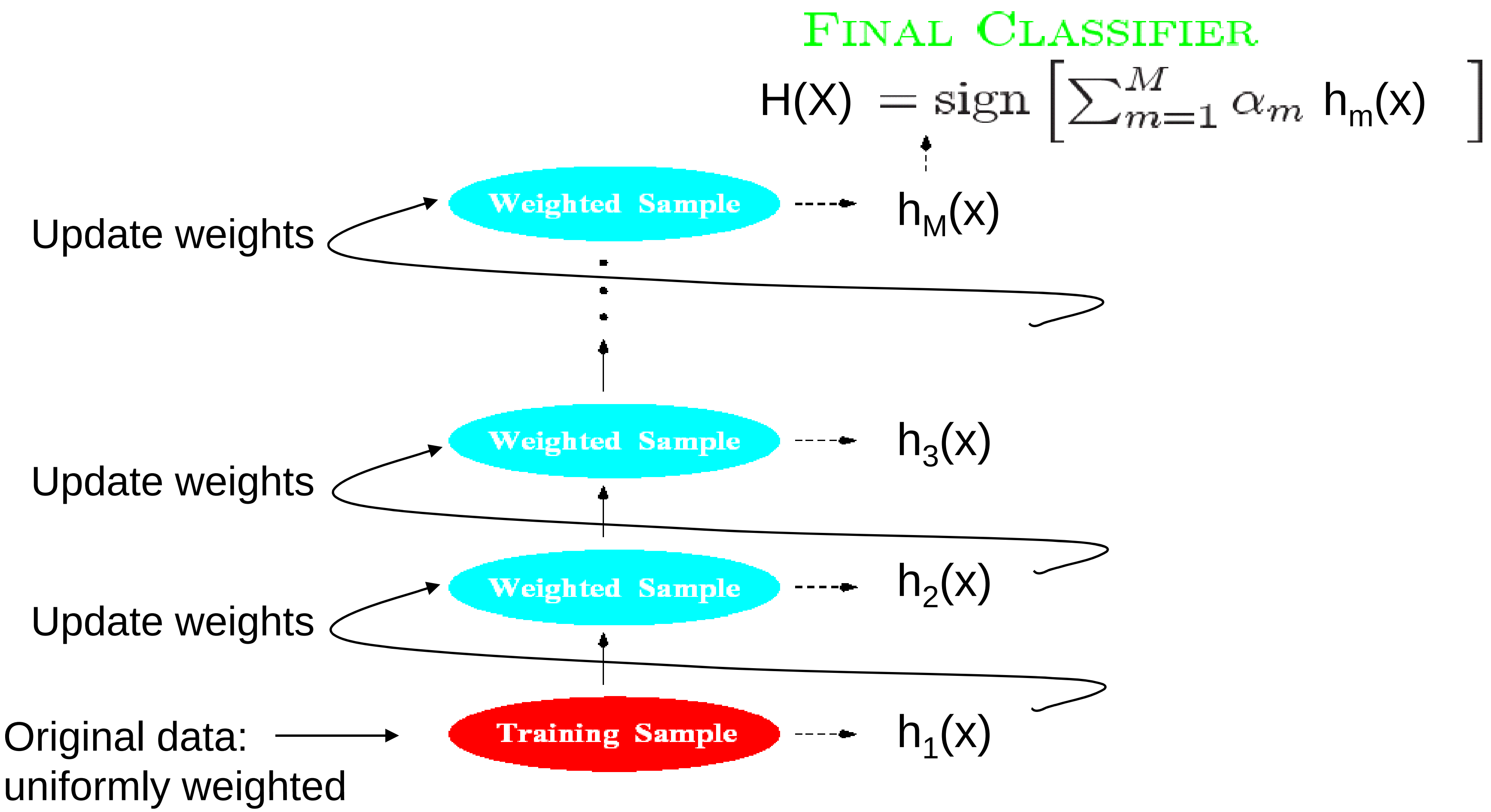
Boosting

- Also an ensemble method: the final prediction is a combination of the prediction of multiple classifiers.
- What is different?
  - Its iterative.

    Boosting: Successive classifiers depends upon its predecessors - look at **errors from previous classifiers** to decide what to **focus** on for the next iteration over data

    Bagging : Individual classifiers were independent.
  - All training examples are used in each iteration, but with different weights – more weights on difficult sexamples. (the ones on which we committed mistakes in the previous iterations)

Adaboost: Illustration



The AdaBoost Algorithm

Input: a set S, of m labeled examples: $S = \{(x_i, y_i), i = 1, 2, \ldots, m\}$,

labels $y_i \in Y = \{1, \ldots, K\}$

Learn (a learning algorithm)

a constant L.

The AdaBoost Algorithm

Input: a set S, of m labeled examples: $S = \{(x_i, y_i), i = 1, 2, \ldots, m\}$,

labels $y_i \in Y = \{1, \ldots, K\}$

Learn (a learning algorithm)

a constant L.

[1] **initialize for all** i**:** $w_1(i) := 1/m$ — *initialize the weights*

[2] **for** $\ell = 1$ **to** L **do**

[3] **for all** i: $p_\ell(i) := w_\ell(i)/(\Sigma_i w_\ell(i))$ — *compute normalized weights*

[4] $h_\ell := \text{Learn}(p_\ell)$ — *call* Learn *with normalized weights.*

[5] $\epsilon_\ell := \Sigma_i p_\ell(i)[\![h_\ell(x_i) \neq y_i]\!]$ — *calculate the error of* h_ℓ

[7] **if** $\epsilon_\ell > 1/2$ **then**

[8] $L := \ell - 1$

[9] **exit**

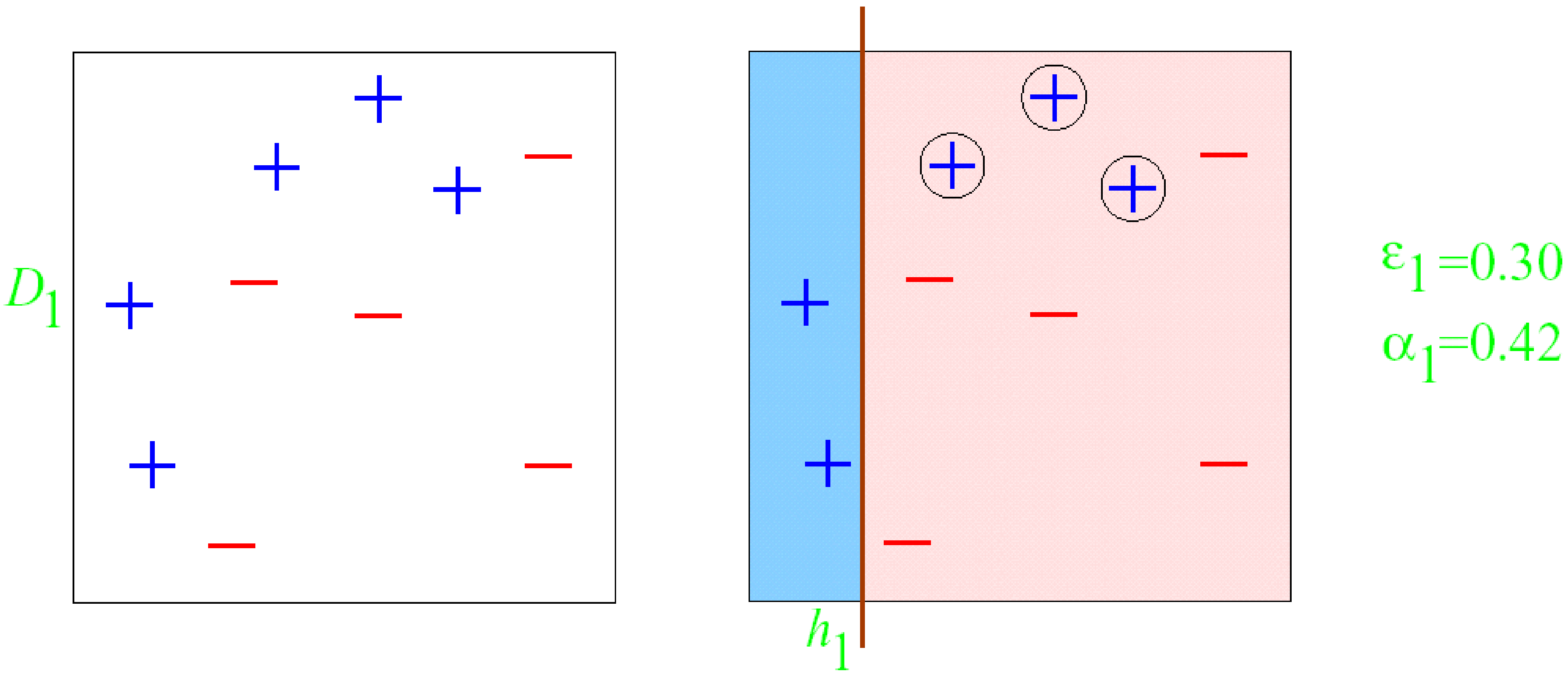
[10] $\beta_\ell := \epsilon_\ell/(1 - \epsilon_\ell)$

[11] **for all** i: $w_{\ell+1}(i) := w_\ell(i)\beta_\ell^{1-[\![h_\ell(x_i) \neq y_i]\!]}$ — *compute new weights*

Output: $h_f(x) = \underset{y \in Y}{\text{argmax}} \sum_{\ell=1}^{L} \left(\log \frac{1}{\beta_\ell}\right) [\![h_\ell(x) = y]\!]$

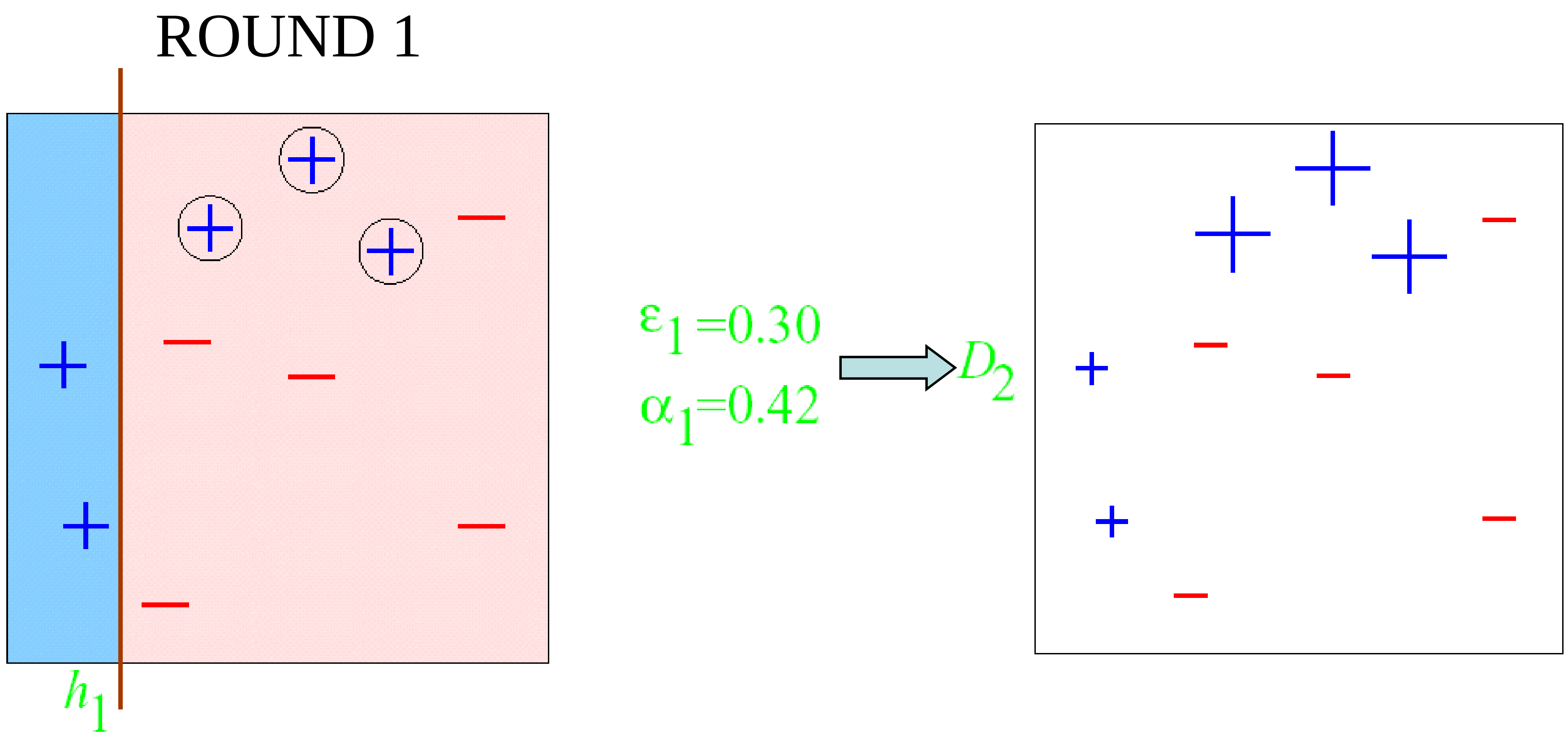
AdaBoost(Example)

Original Training set : Equal Weights to all training samples



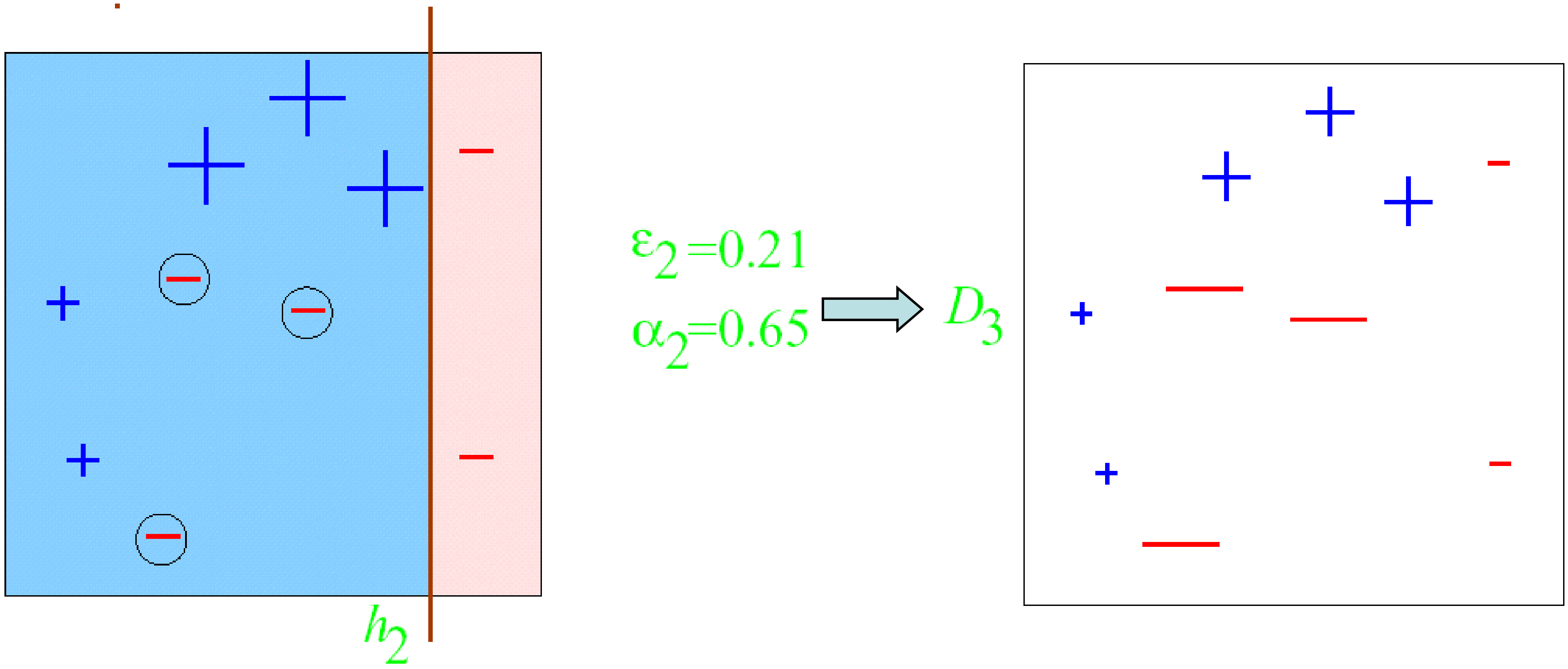
Taken from "**A Tutorial on Boosting" by** Yoav Freund and Rob Schapire

AdaBoost(Example)



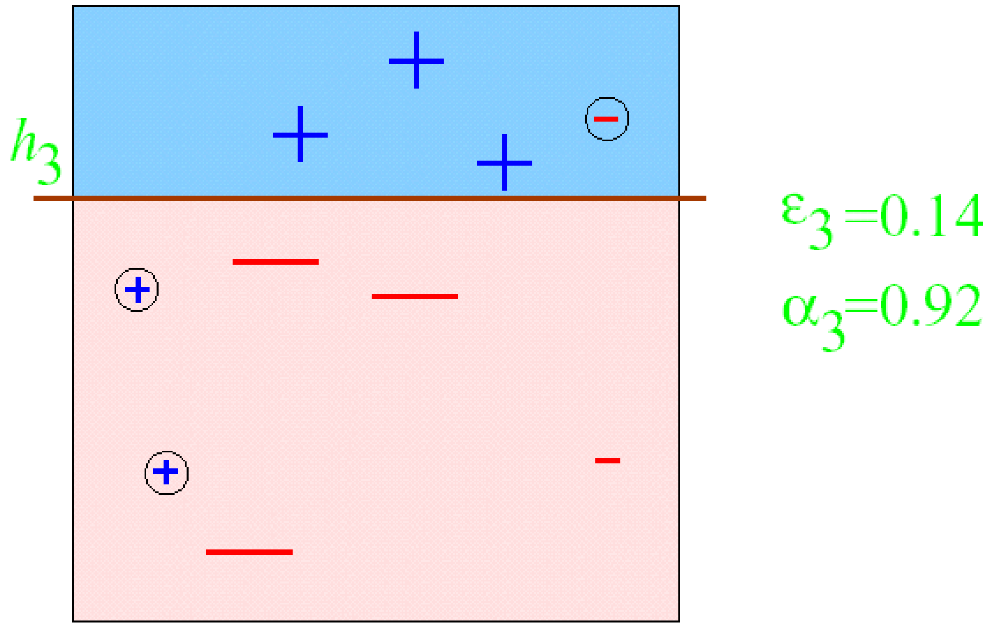
AdaBoost(Example)

ROUND 2



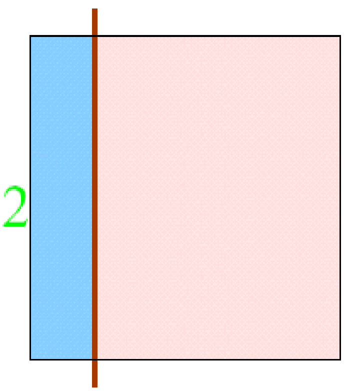
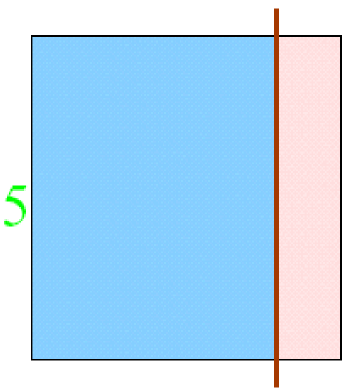
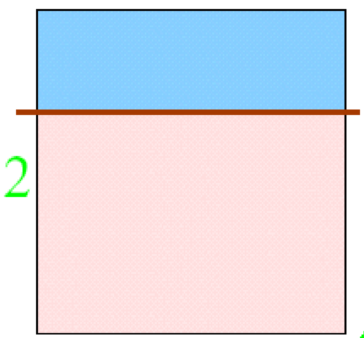
AdaBoost(Example)

ROUND 3



AdaBoost(Example)

H_{final}

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

Weighted Error

- Adaboost calls ***L*** with a set of prespecified weights
- It is often straightforward to convert a base learner ***L*** to take into account an input distribution *D*.

Decision trees?

K Nearest Neighbor?

Naïve Bayes?

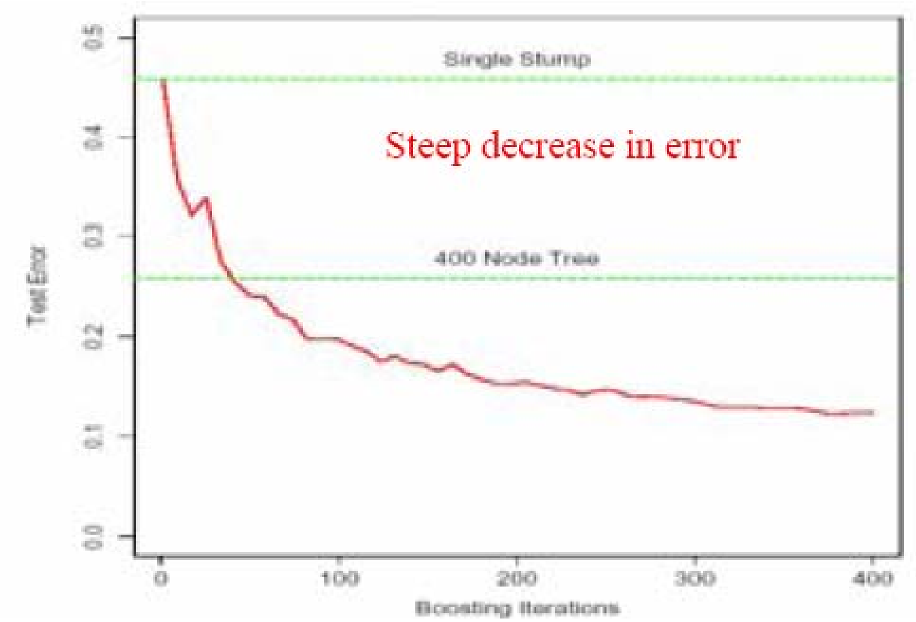
- When it is not straightforward we can resample the training data S according to D and then feed the new data set into the learner.

Boosting Decision Stumps

Decision stumps: very simple rules of thumb that test condition on a single attribute.

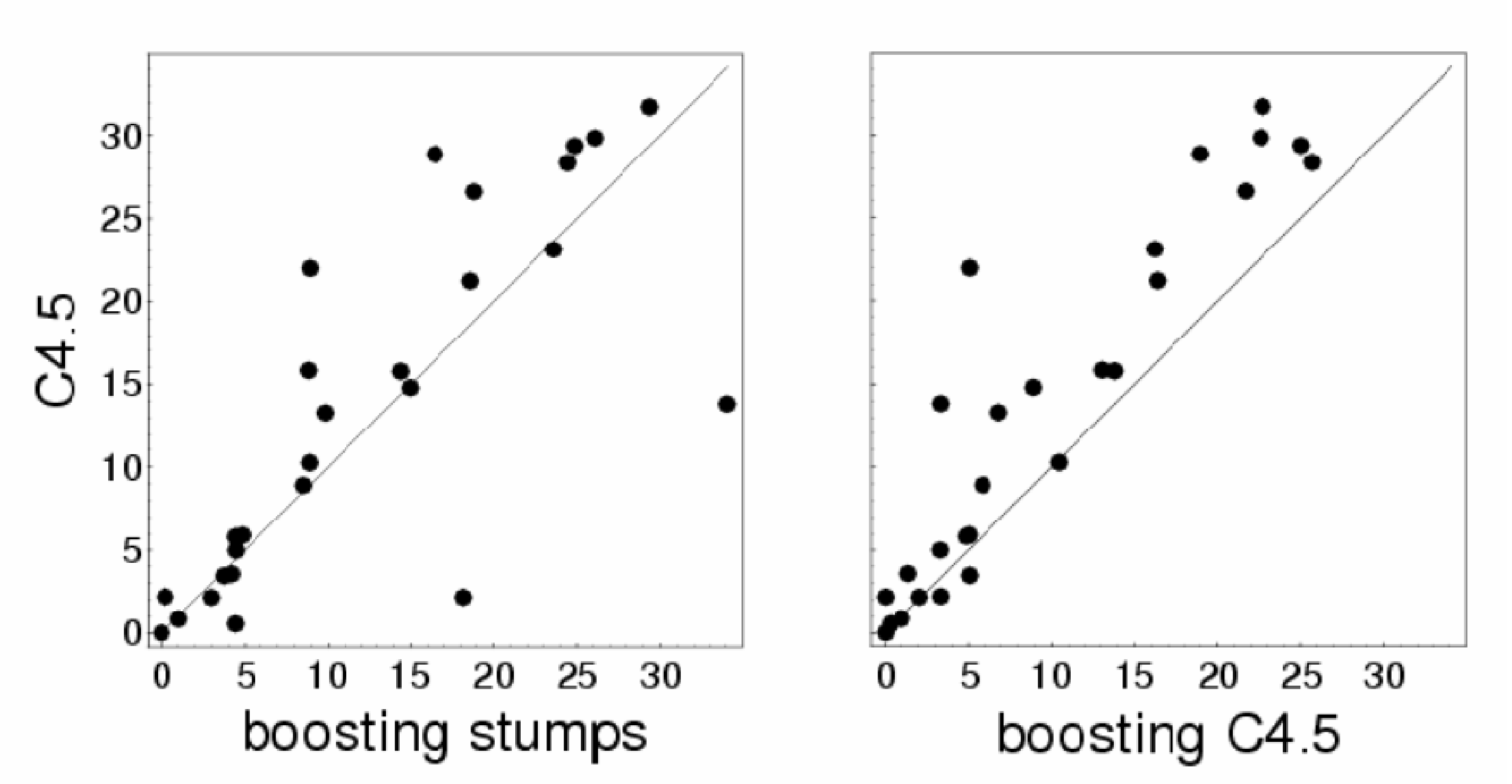
Among the most commonly used base classifiers – truly weak!

Boosting with decision stumps has been shown to achieve better performance compared to unbounded decision trees.

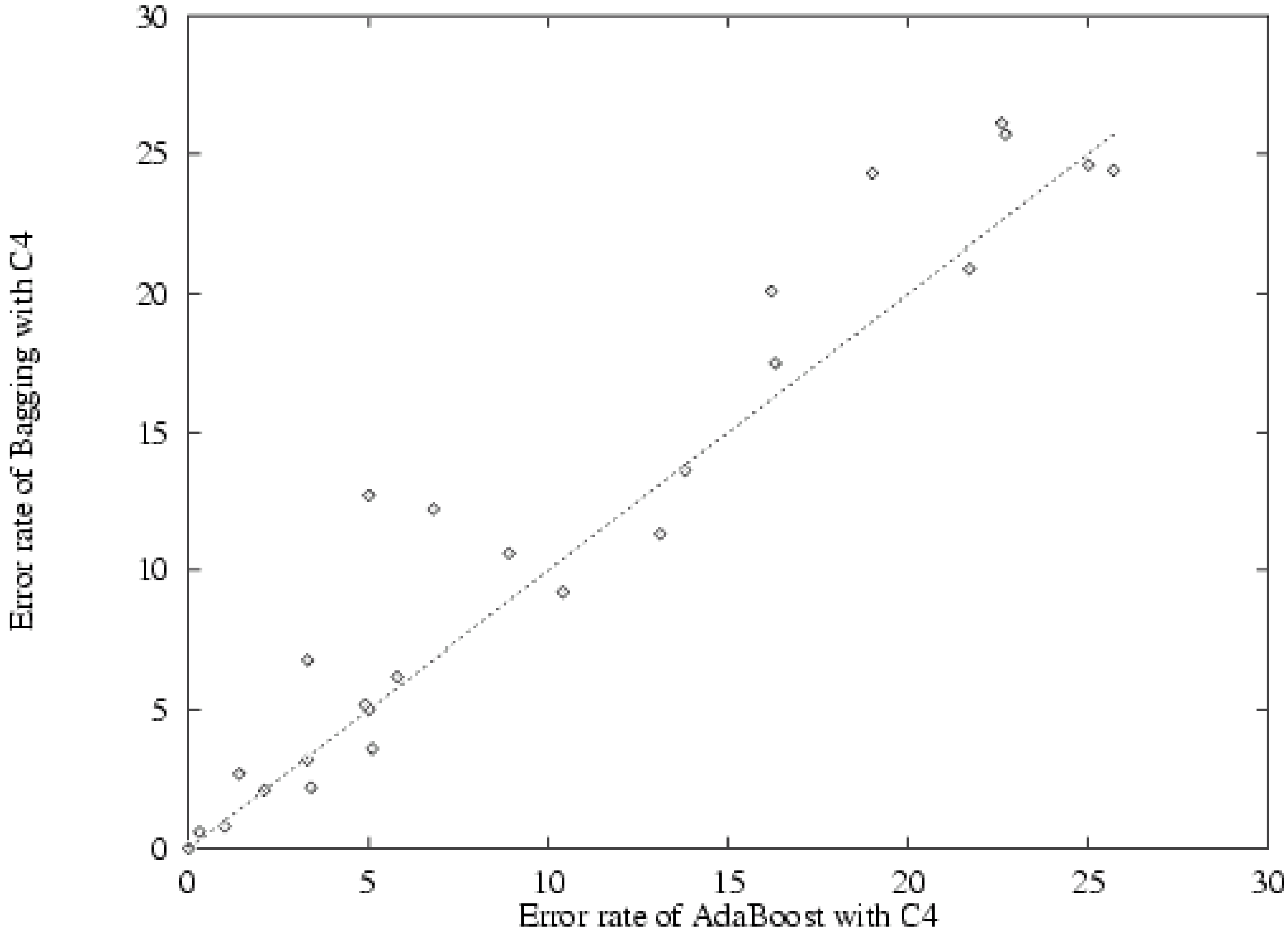


Boosting Performance

- Comparing C4.5, boosting decision stumps, boosting C4.5 using 27 UCI data set
  - C4.5 is a popular decision tree learner

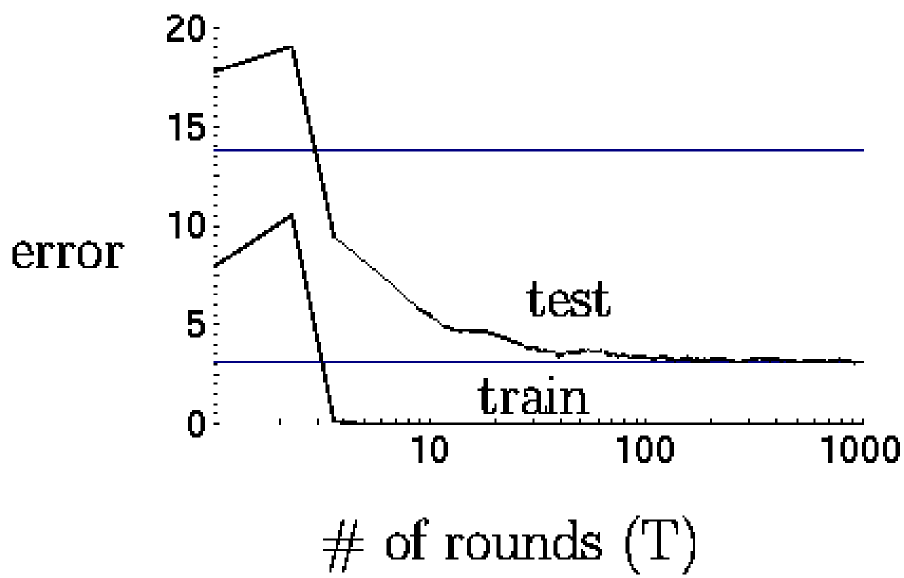


Boosting vs Bagging of Decision Trees



Overfitting?

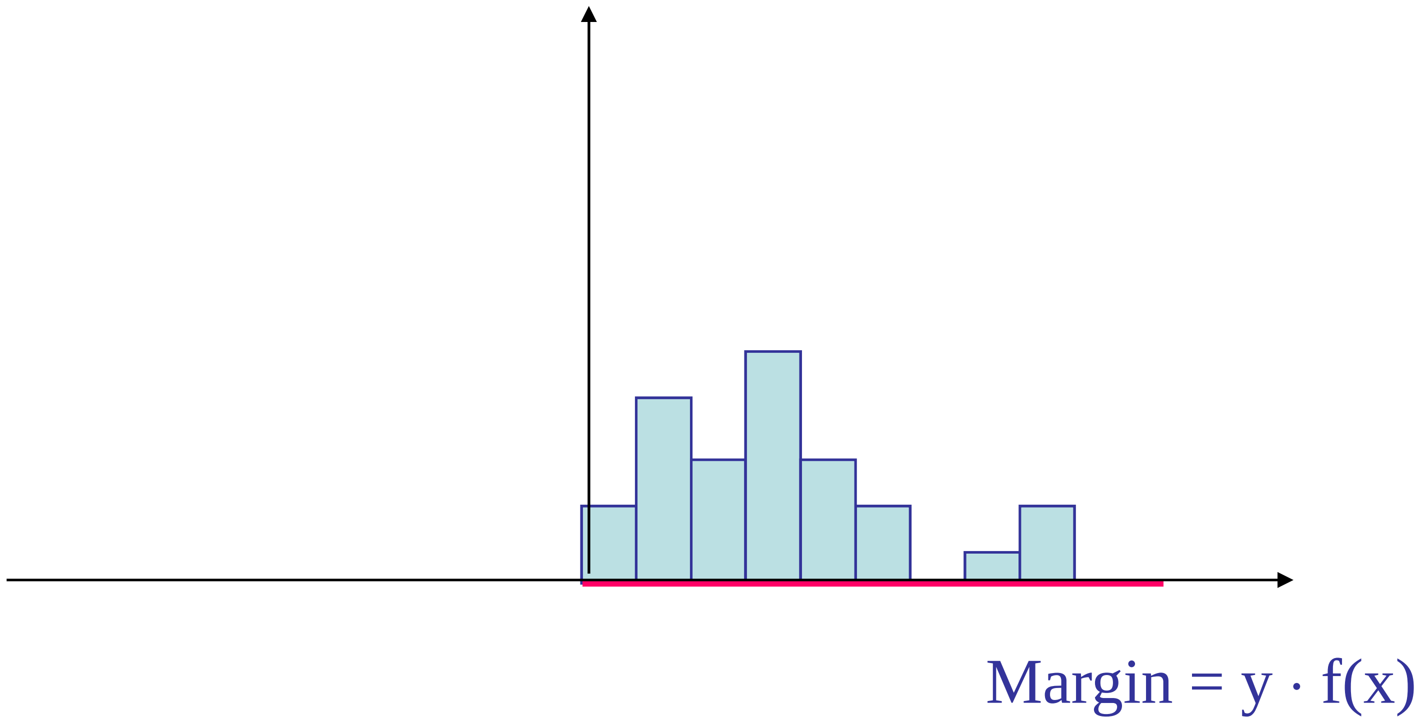
- Boosting drives training error to zero, will it overfit?
- Curious phenomenon



- Boosting is often robust to overfitting (not always)
- Test error continues to decrease even after training error goes to zero

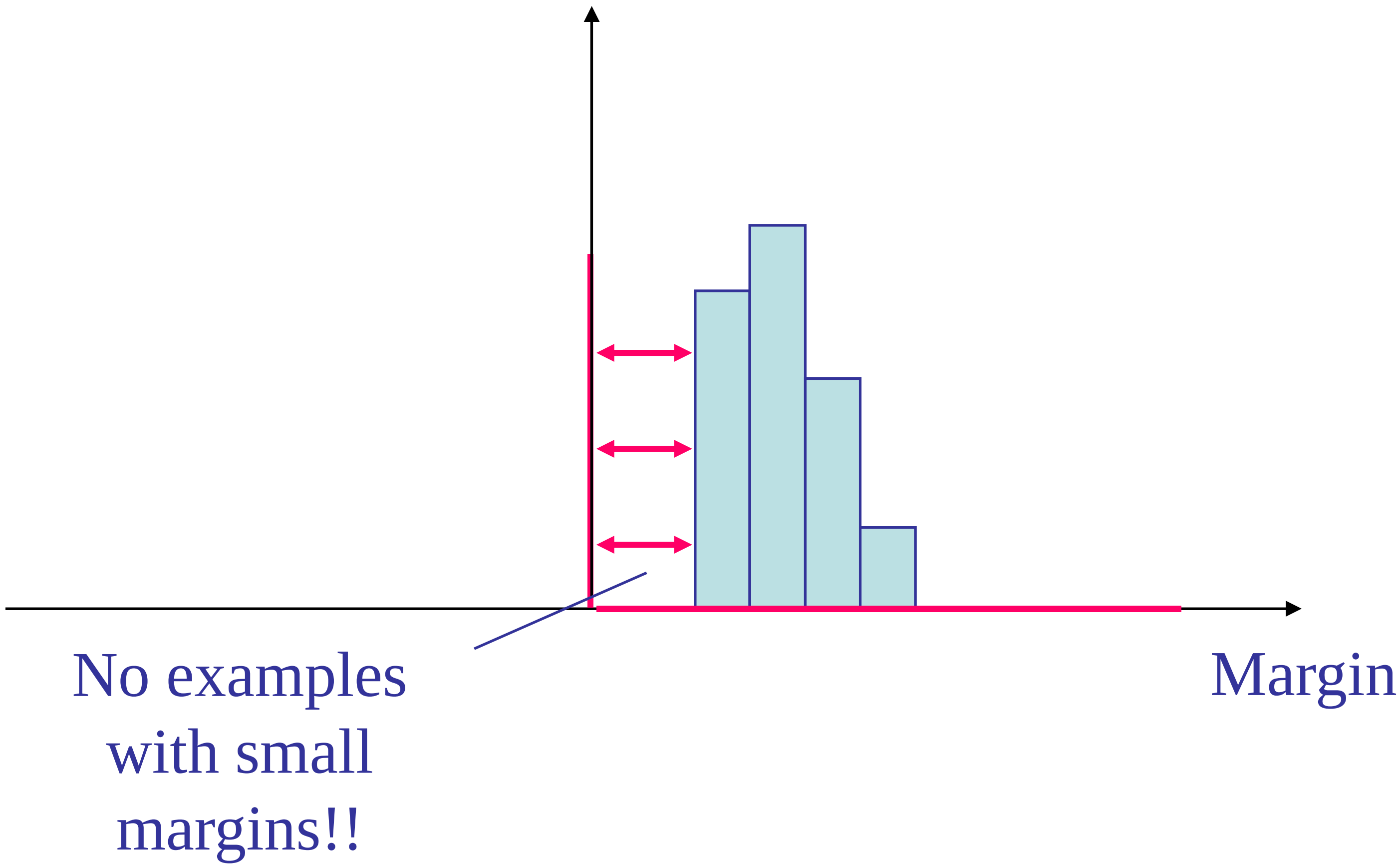
Explanation with Margins

$$f(x) = \sum_{l=1}^{L} w_l \cdot h_l(x)$$



Histogram of functional margin for ensemble just after achieving zero training error

Effect of Boosting: Maximizing Margin



Even after zero training error the margin of examples increases.
This is one reason that the generalization error may continue decreasing.

Bias/variance analysis of Boosting

- In the early iterations, boosting is primary a bias-reducing method
- In later iterations, it appears to be primarily a variance-reducing method

What you need to know about ensemble methods?

- Bagging: a randomized algorithm based on bootstrapping
  - What is bootstrapping
  - Variance reduction
  - What learning algorithms will be good for bagging?
- Boosting:
  - Combine weak classifiers (i.e., slightly better than random)
  - Training using the same data set but different weights
  - How to update weights?
  - How to incorporate weights in learning (DT, KNN, Naïve Bayes)
  - One explanation for not overfitting: maximizing the margin