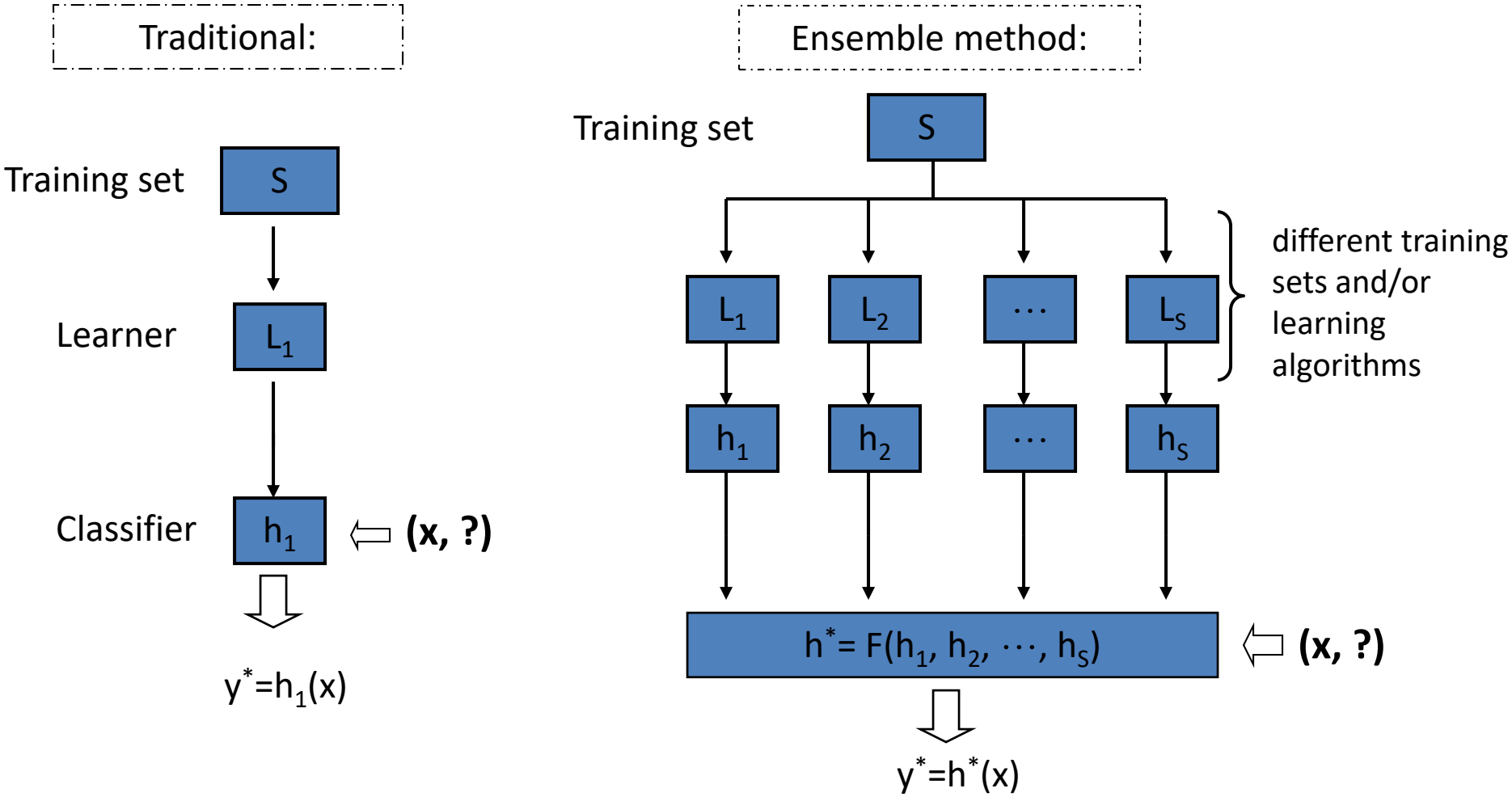
Ensemble Learning

Ensemble Learning

- So far we have seen learning algorithms that take a training (data) set and output a classifier (or clustering solution)
- What if we want more accuracy than current algorithms afford?
  - Develop new learning algorithm
  - Improve existing algorithms
- Another approach is to leverage the algorithms we have via ensemble methods
  - Instead of calling an algorithm just once
  - Call algorithm multiple times and combine the multiple outputs
- Can be used in both supervised and unsupervised learning
  - We will focus on supervised ensemble learning here

Supervised Ensemble Learning



Why Ensemble Learning?

- **INTUITION:** Combining predictions of multiple classifiers (an ensemble) via majority vote will be more accurate than a single classifier.
- Majority vote

Suppose we have 5 completely independent classifiers…

  - If accuracy is 70% for each
    - 10 (.7^3)(.3^2)+5(.7^4)(.3)+(.7^5)
    - **83.7% majority vote accuracy**
  - 101 such classifiers
    - **99.9% majority vote accuracy**

How to generate ensemble?

- There are a variety of methods developed
- We will look at two of them:
  - Bagging: use different samples of the examples to generate diverse classifiers
  - Boosting (Adaboost: adaptive boosting): iteratively build classifiers by making examples currently misclassified more important

Bagging: Bootstrap Aggregation
(Breiman, 1996)

S - Training data set.

1. Generate T bootstrap training sets $S_1, \ldots, S_T$ from S
   (see next slide for bootstrap procedure)

2. For each sample, learn a classifier (e.g., a decision tree)

3. Out put all classifiers, and use majority vote to make predictions

Generate a Bootstrap sample of S

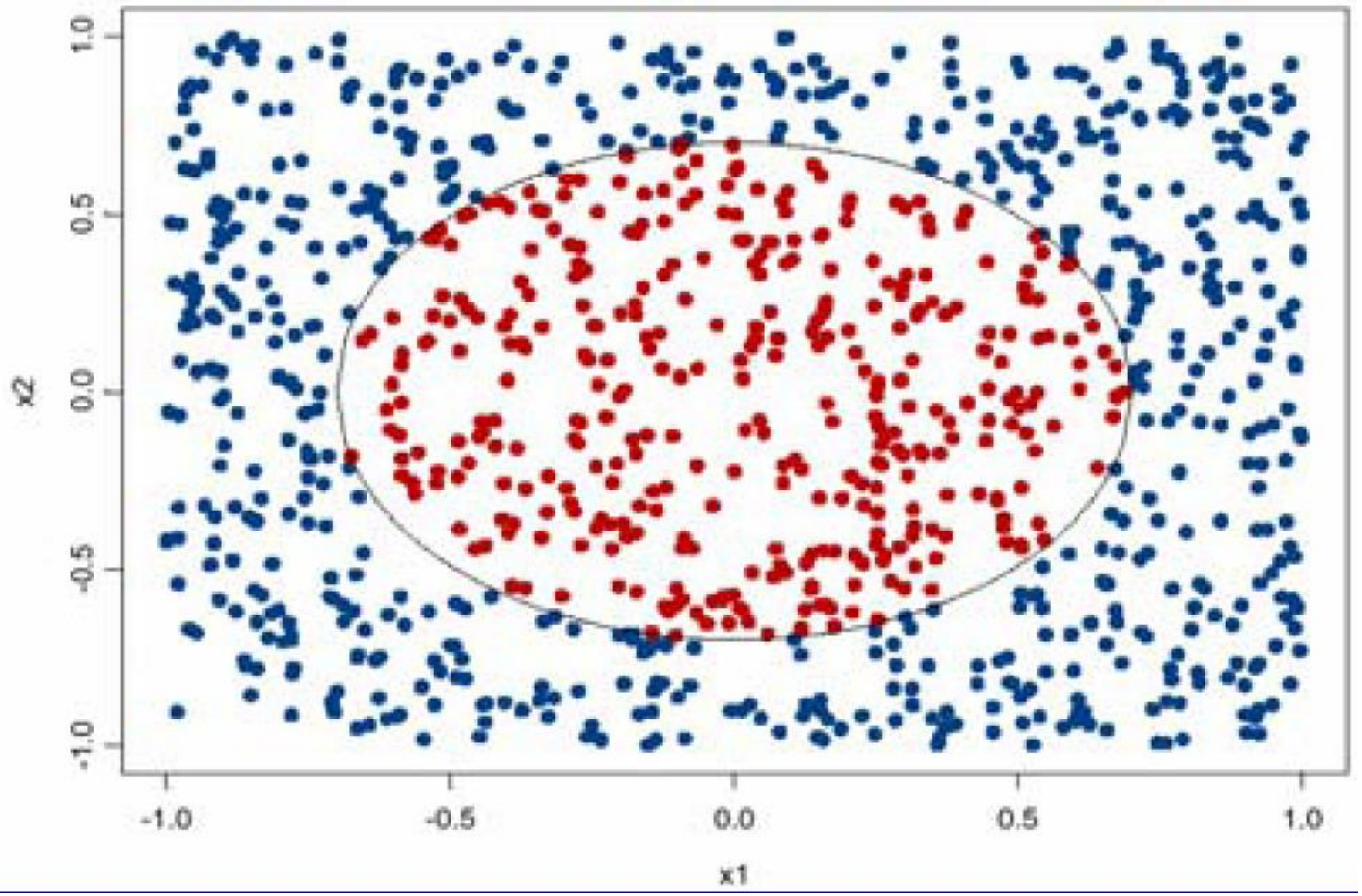
Start with an empty set S'

Repeated sample a random point from S to add to S' until |S'|=|S|

Return S'

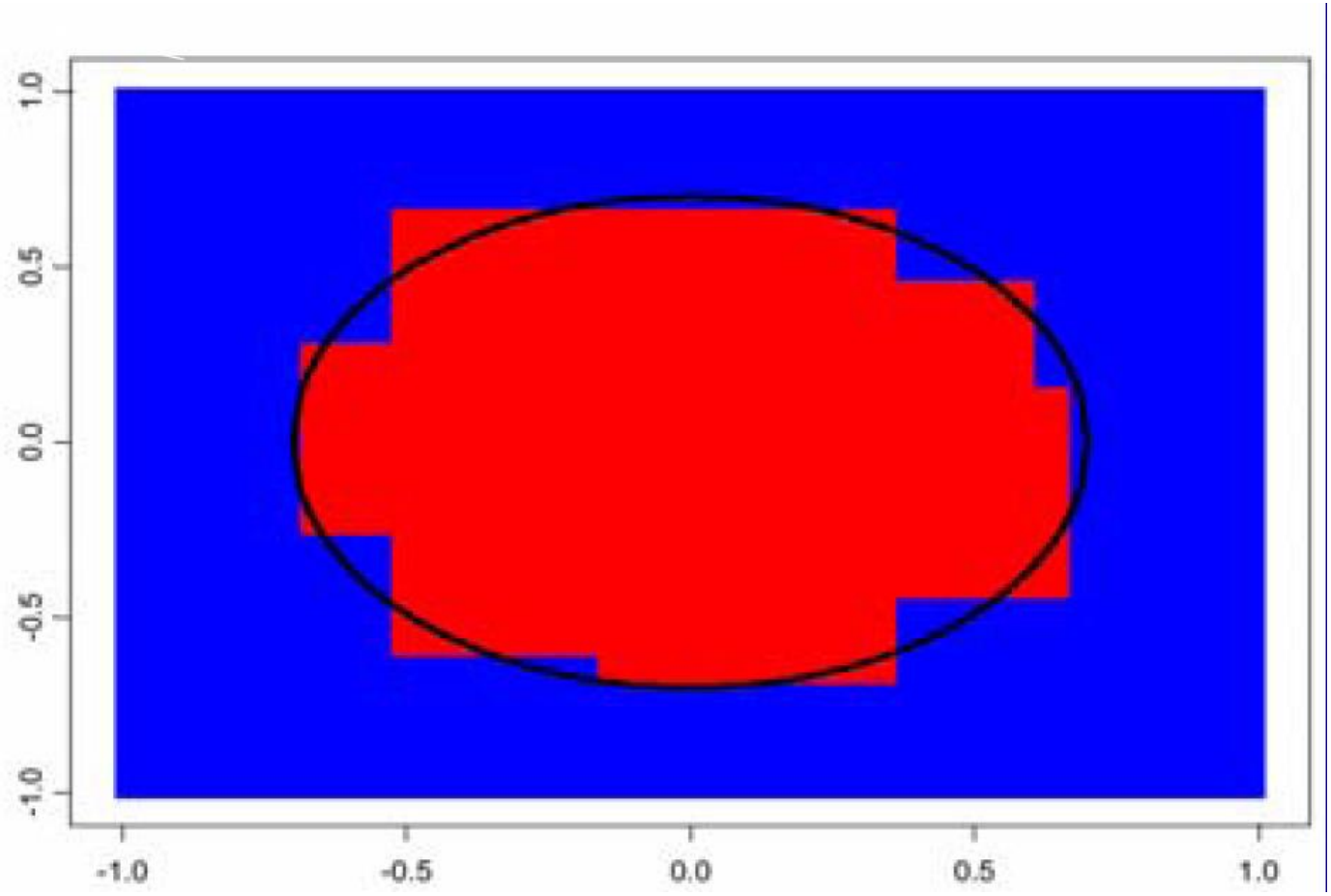
- This procedure is called **sampling with replacement**
  - Each time a point is drawn, it will not be removed
  - This means that we can have multiple copies of the same data point in my sample
  - size of S'= size of S
  - On average, 66.7% of the original points will appear in S'

Bagging Example



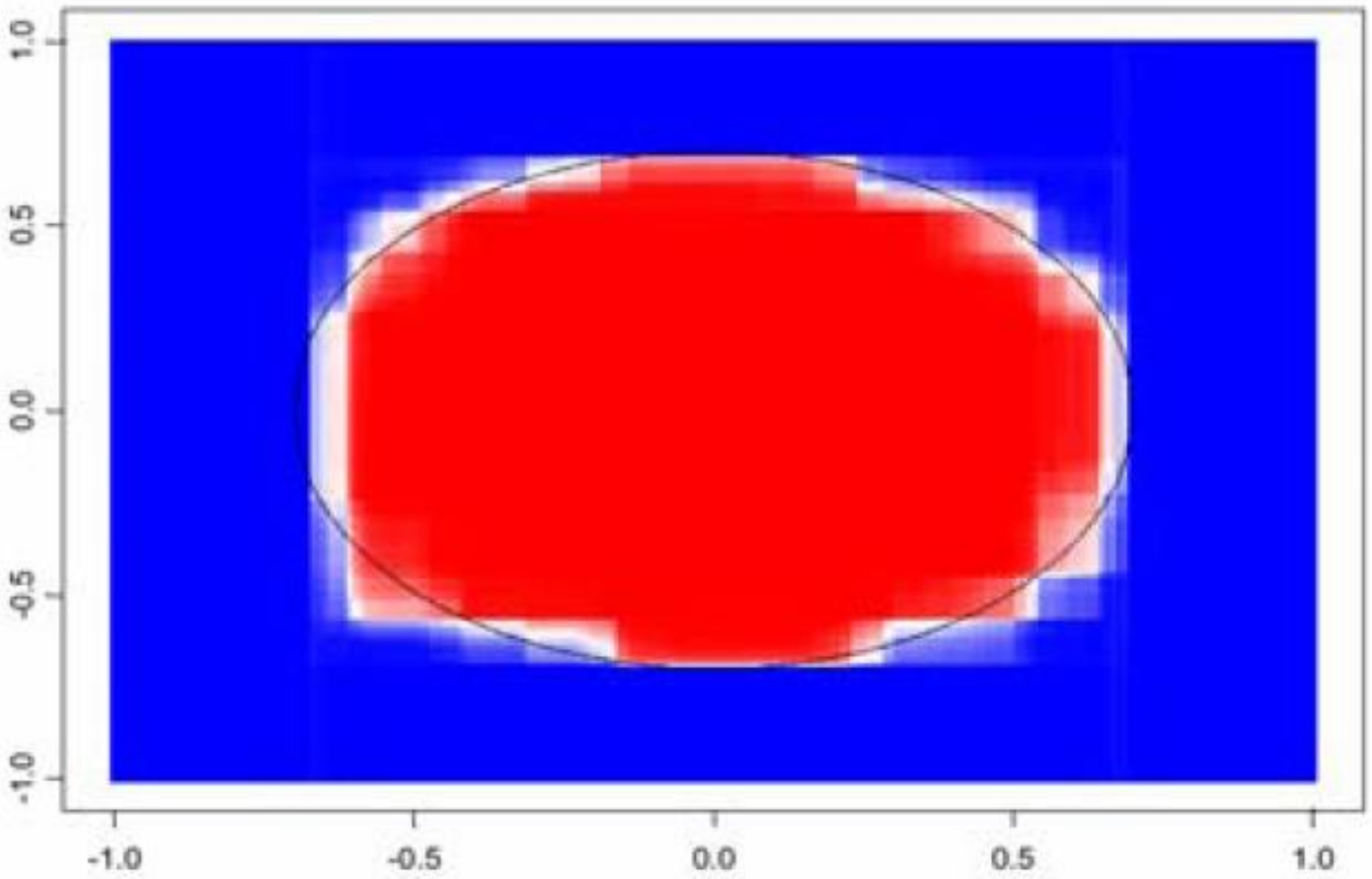
The true decision boundary

Decision Boundary by the CART Decision Tree Algorithm



Note that the decision tree has trouble representing this decision boundary

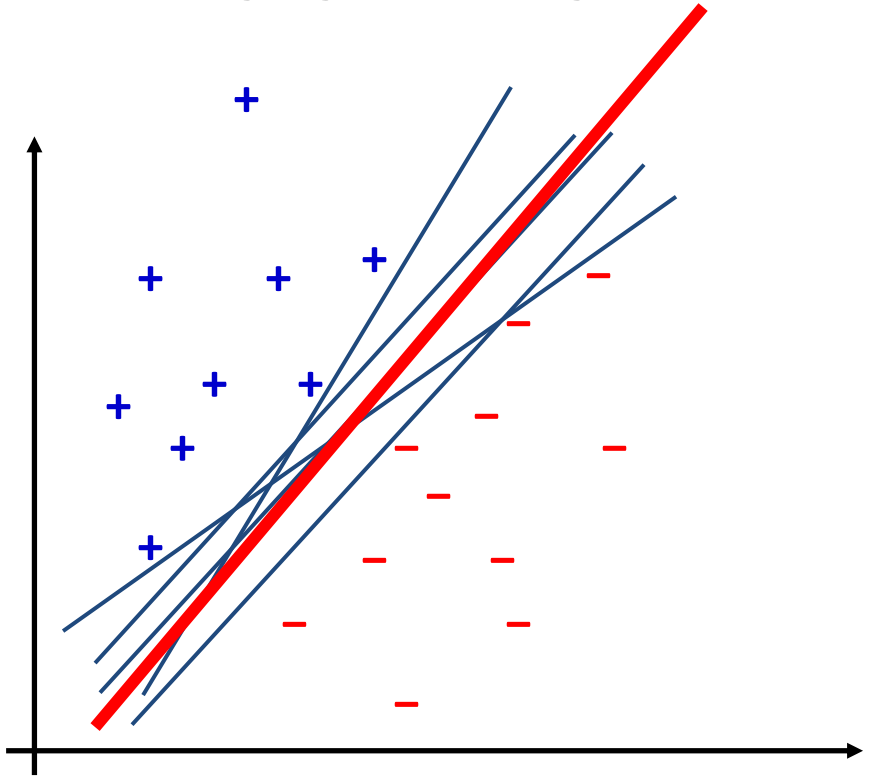
100 bagged trees



By averaging 100 trees, we achieve better approximation of the boundary, together with information regarding how confidence we are about our prediction.

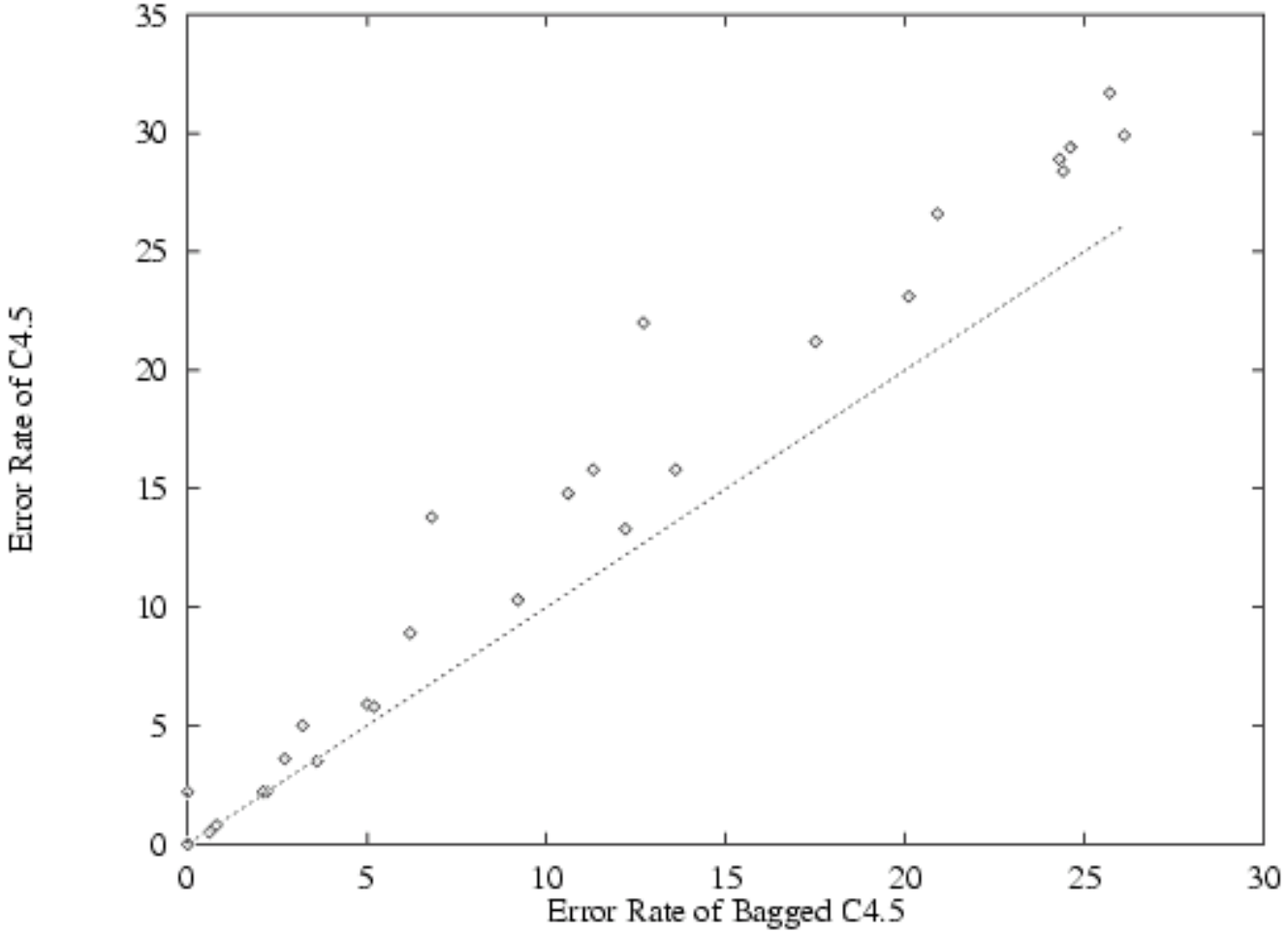
Another Example

- Consider bagging with the linear perceptron base learning algorithm
- Each bootstrap training set will give a different linear separator
- Voting is **similar** to averaging them together (not equivalent)



Empirical Results for Bagging Decision Trees

(Freund & Schapire)

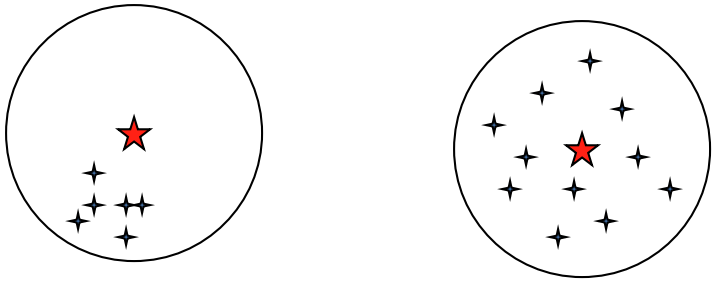


Each point represents the results of one data set

Why can bagging improve the classification accuracy?

The Concept of Bias and Variance

There are two main types of errors: bias and variance



Bias

Variance

When training on a randomly sampled training set, if the algorithm consistently miss the target – this is the bias

If algorithm on average is on target but randomly gets off target depending on the training set – this is the variance

Bias/Variance for classifiers

- Bias arises when the classifier cannot represent the true function – that is, the classifier underfits the data
  - If the hypothesis space does not contain the target function, then we have bias

- Variance arises when classifiers learned on minor variations of data result in significantly different classifiers – variations cause the classifier to overfit differently
  - If the hypothesis space has only one function then the variance is zero (but the bias is huge)

- 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

Why does bagging work?

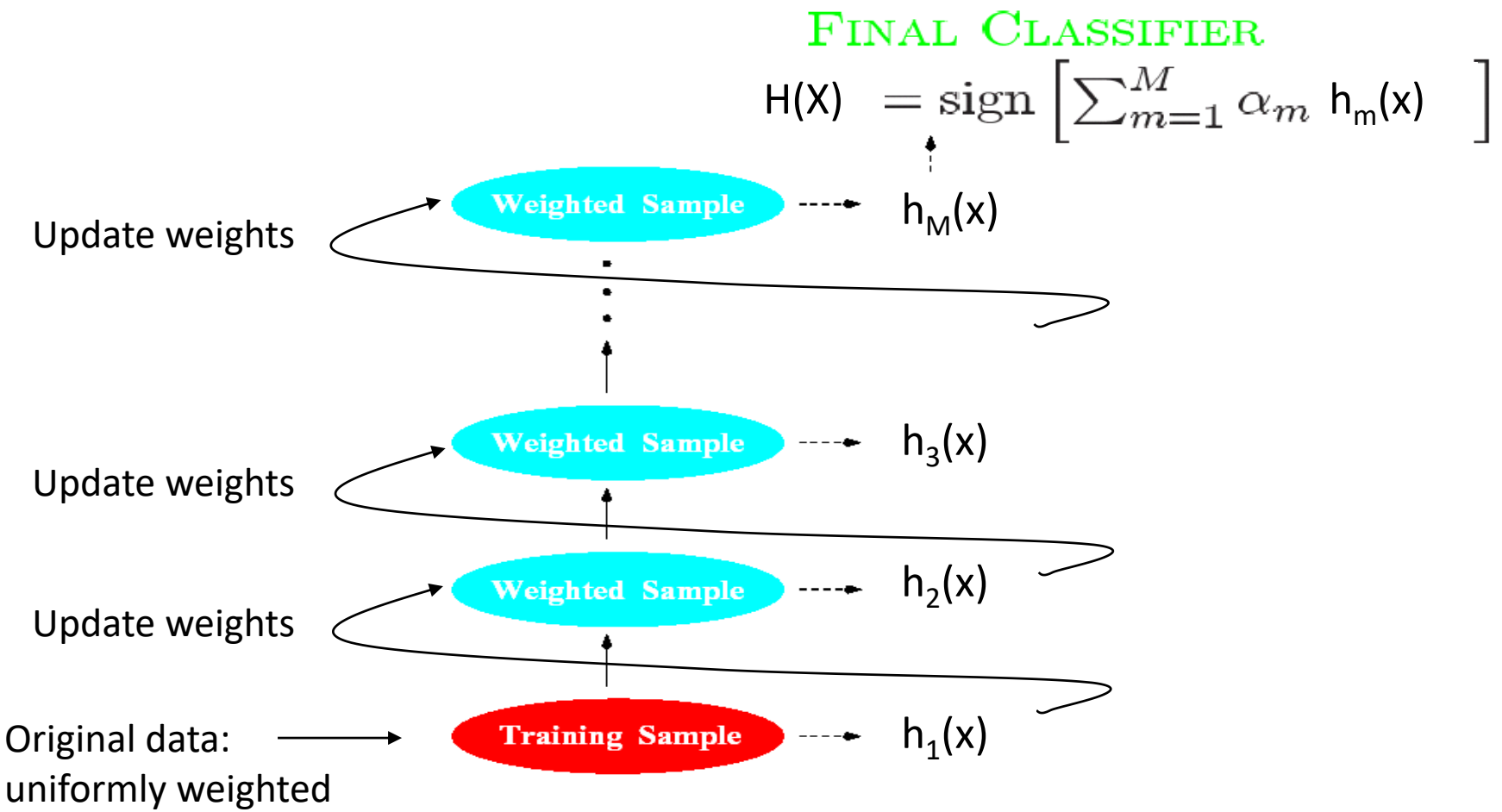
- Bagging takes the average of multiple models -- reduces the variance
- This suggests that bagging works the best with low bias and high variance classifiers such as …
  - Un-pruned decision trees
- Bagging typically will not hurt the performance

Random Forest

- A variant of bagged decision trees
- When learning decision tree from bootstrap samples, each step only consider a random subset of the features
  - The number of random features to consider is a hyper-parameter
- Introduce more variance in the classifiers than bagging
- Powerful and yet simple to use classifiers

Boosting

Adaboost: Illustration



AdaBoost algorithm:

Input: S - Set of N labeled training instances.

Output: $H(x) = sign\left[\sum_{t=1}^{M} \alpha_m \cdot h_m(x)\right]$

Initialize $D_1(i) = 1/N$, for all i from 1 to N. (uniform distribution)

FOR $t = 1, 2, \ldots, M$ **DO**

$h_t = Learn(S, D_t)$

$\varepsilon_t = error(h_t, S, D_t)$

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

;; if $\varepsilon_t < 0.5$ implies $\alpha_t > 0$

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

for i from 1 to N

Normalize D_{t+1} ;; can show that h_t has 0.5 error on D_{t+1}

Note that $\varepsilon_t < 0.5$ implies $\alpha_t > 0$ so weight is decreased for instances h_t predicts correctly and increases for incorrect instances

Weighted Error

• Adaboost calls ***Learn*** with a set of prespecified weights
• It is often straightforward to convert a base learner ***Learn*** to take into account the weights in *D*.

Decision trees?

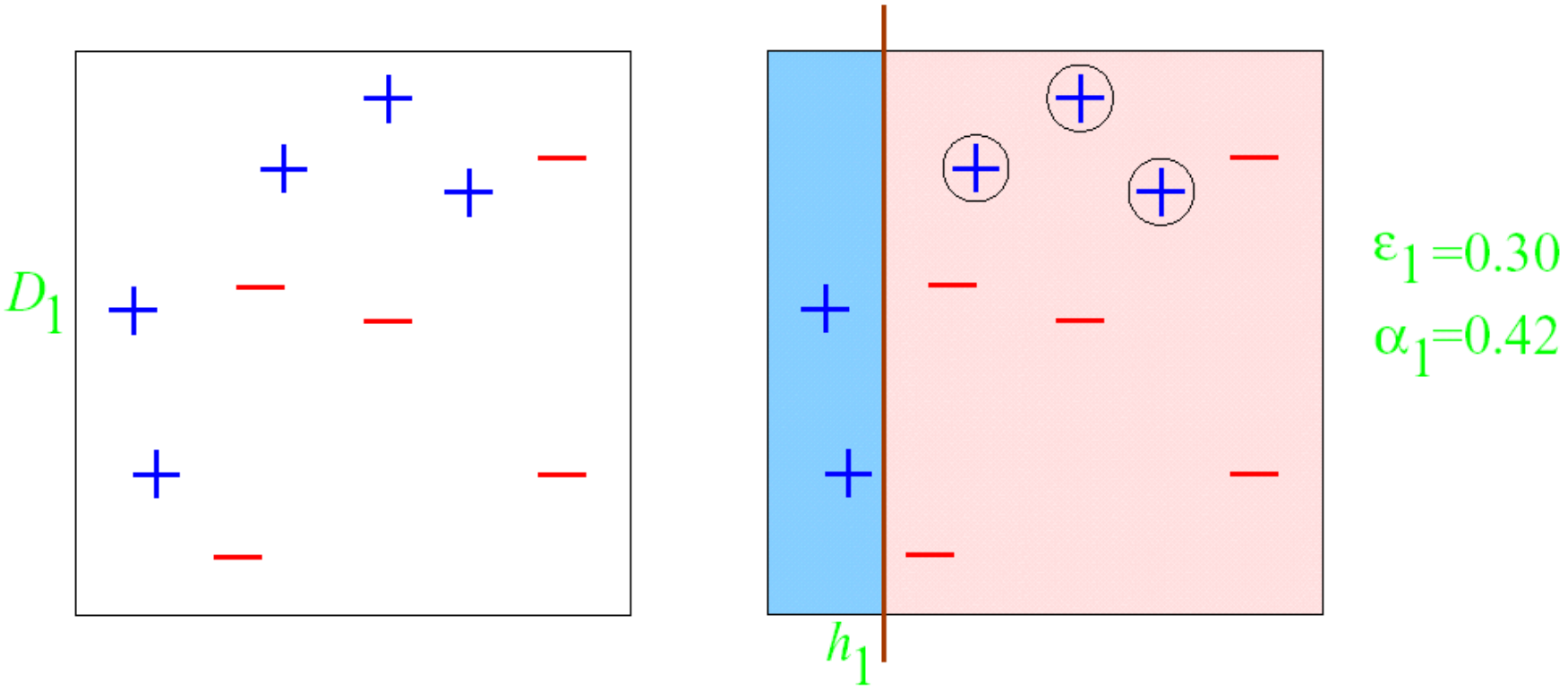
K Nearest Neighbor?

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.

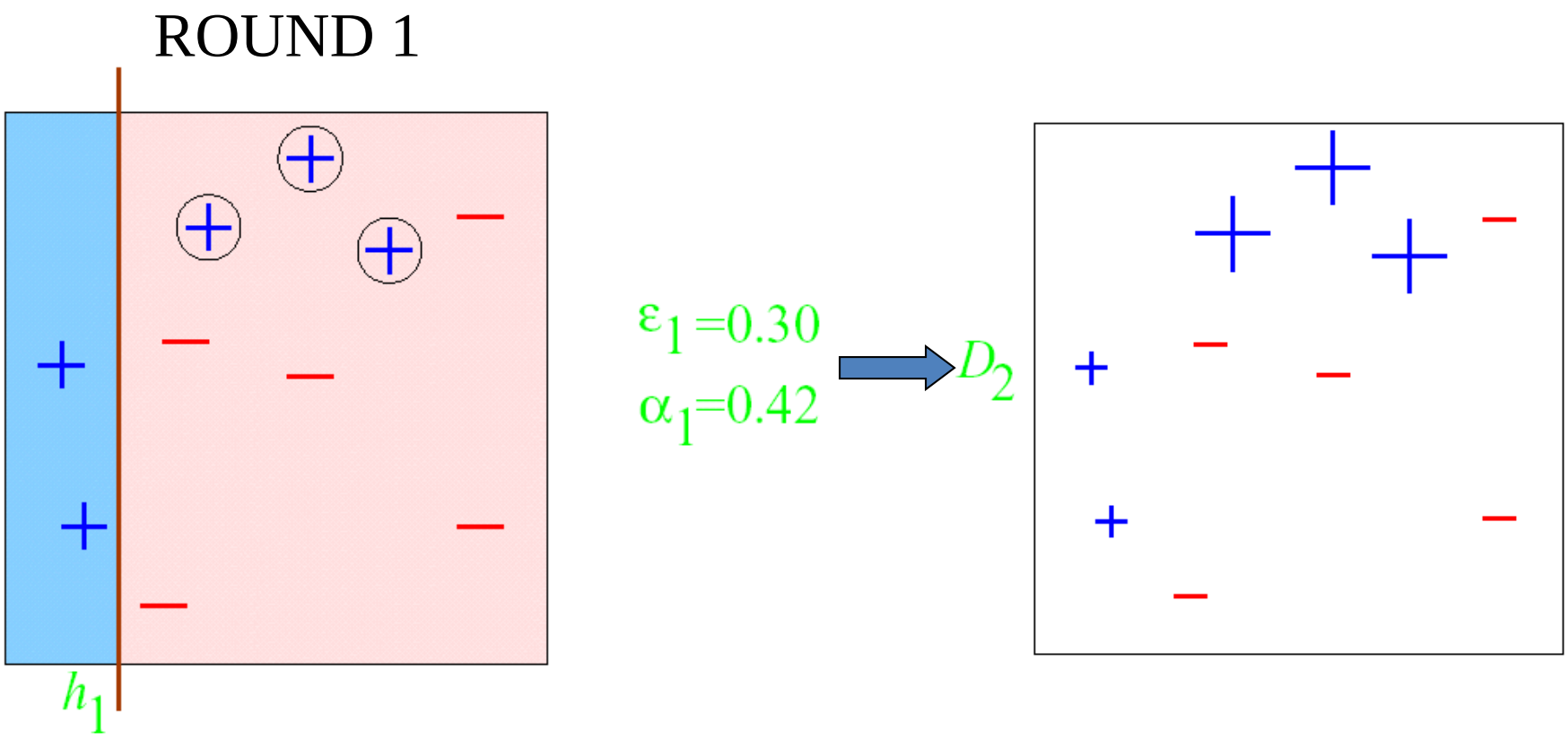
AdaBoost using Decision Stump (Depth-1 decision tree)

Original Training set : Equal
Weights to all training samples

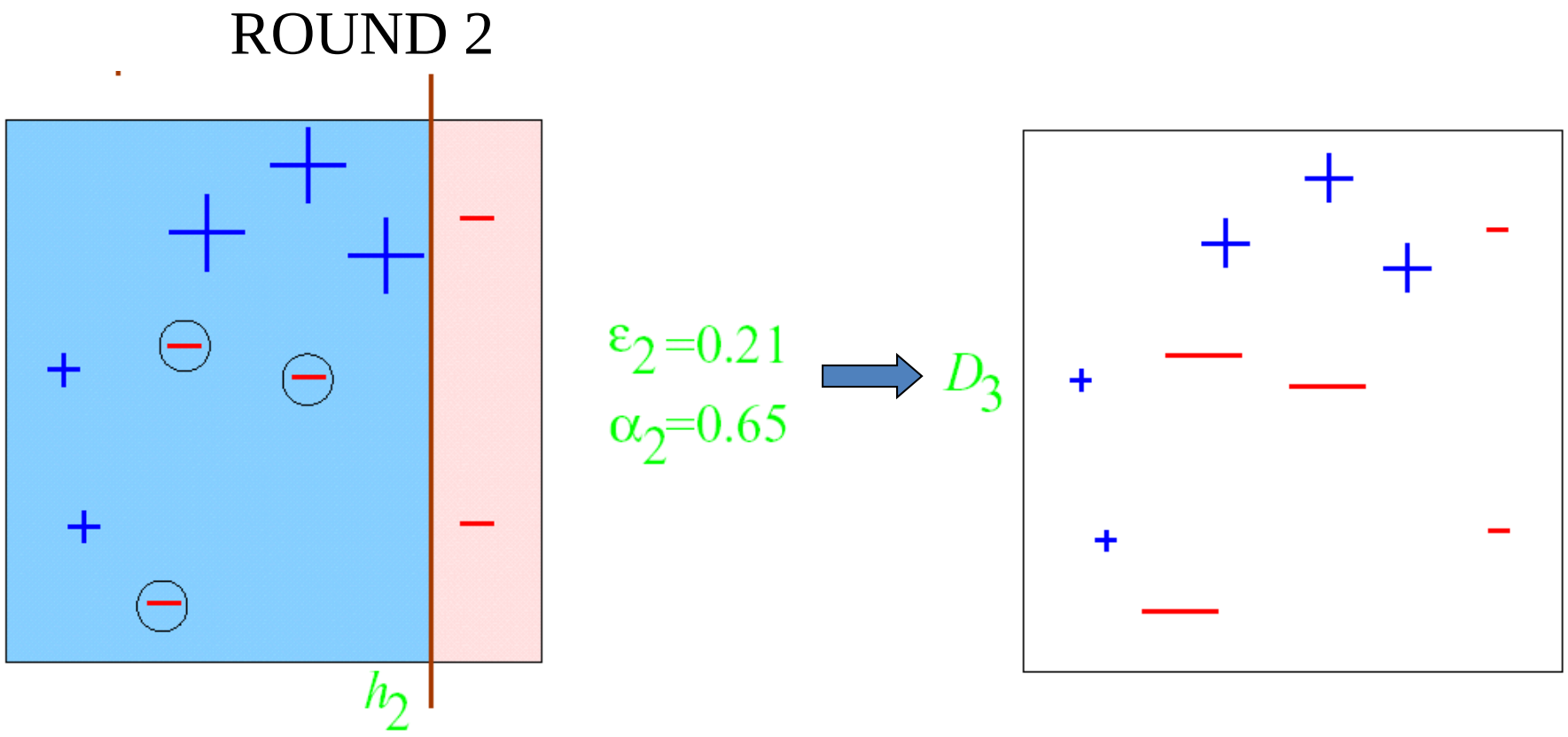


Taken from "**A Tutorial on Boosting" by** Yoav Freund and Rob Schapire

AdaBoost(Example)

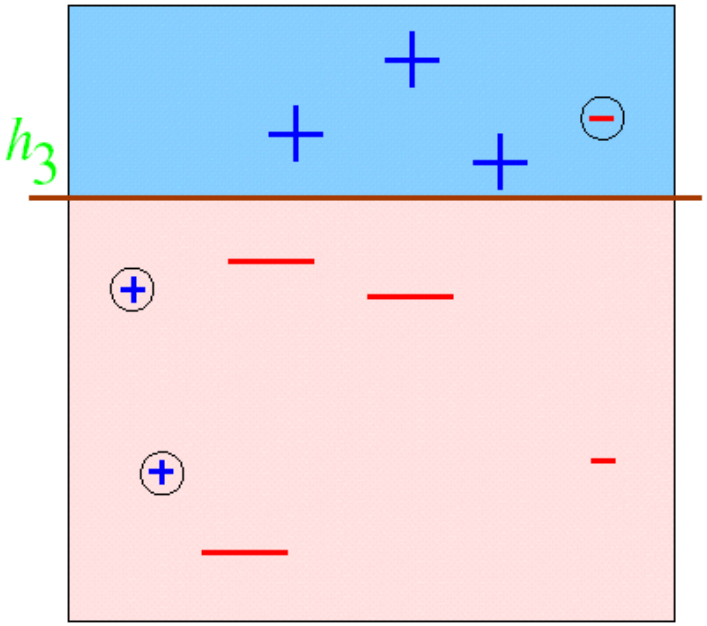


AdaBoost(Example)



AdaBoost(Example)

ROUND 3

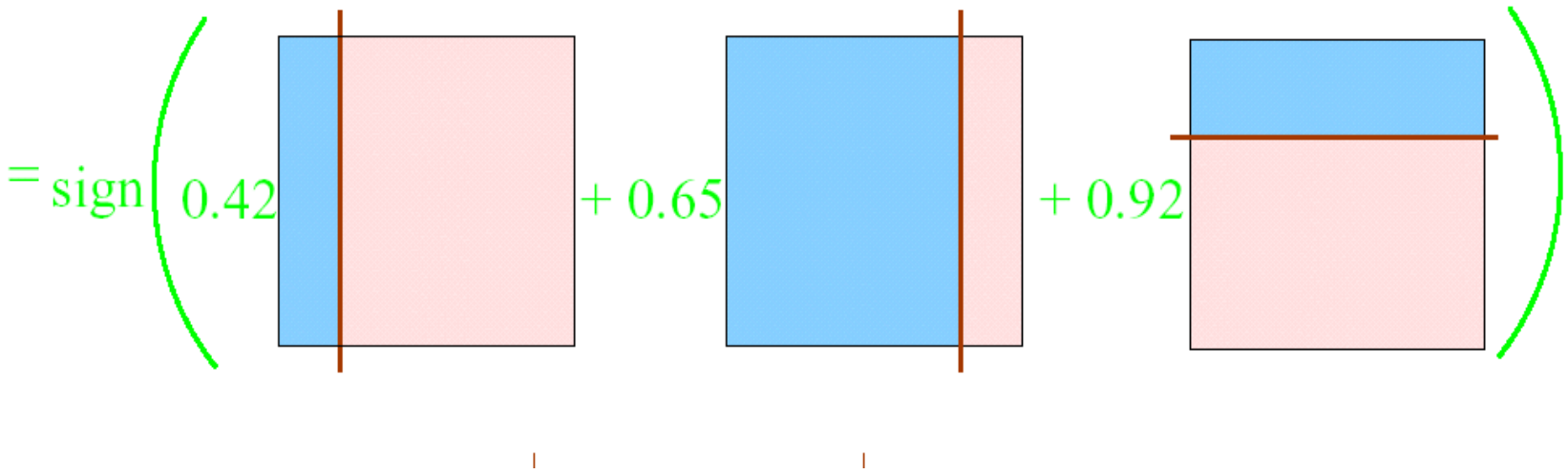


$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

AdaBoost(Example)

H_{final}

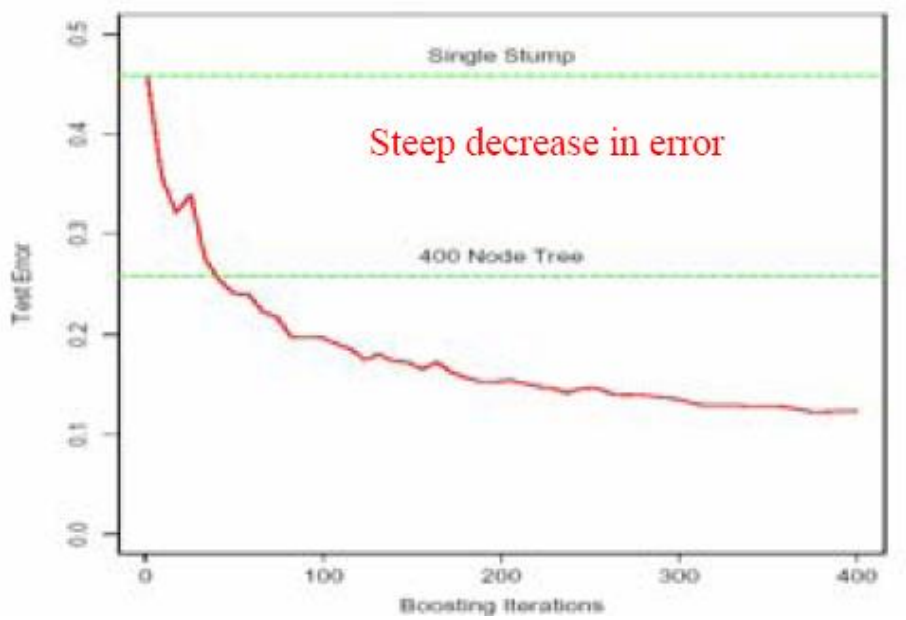


Boosting Decision Stumps

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

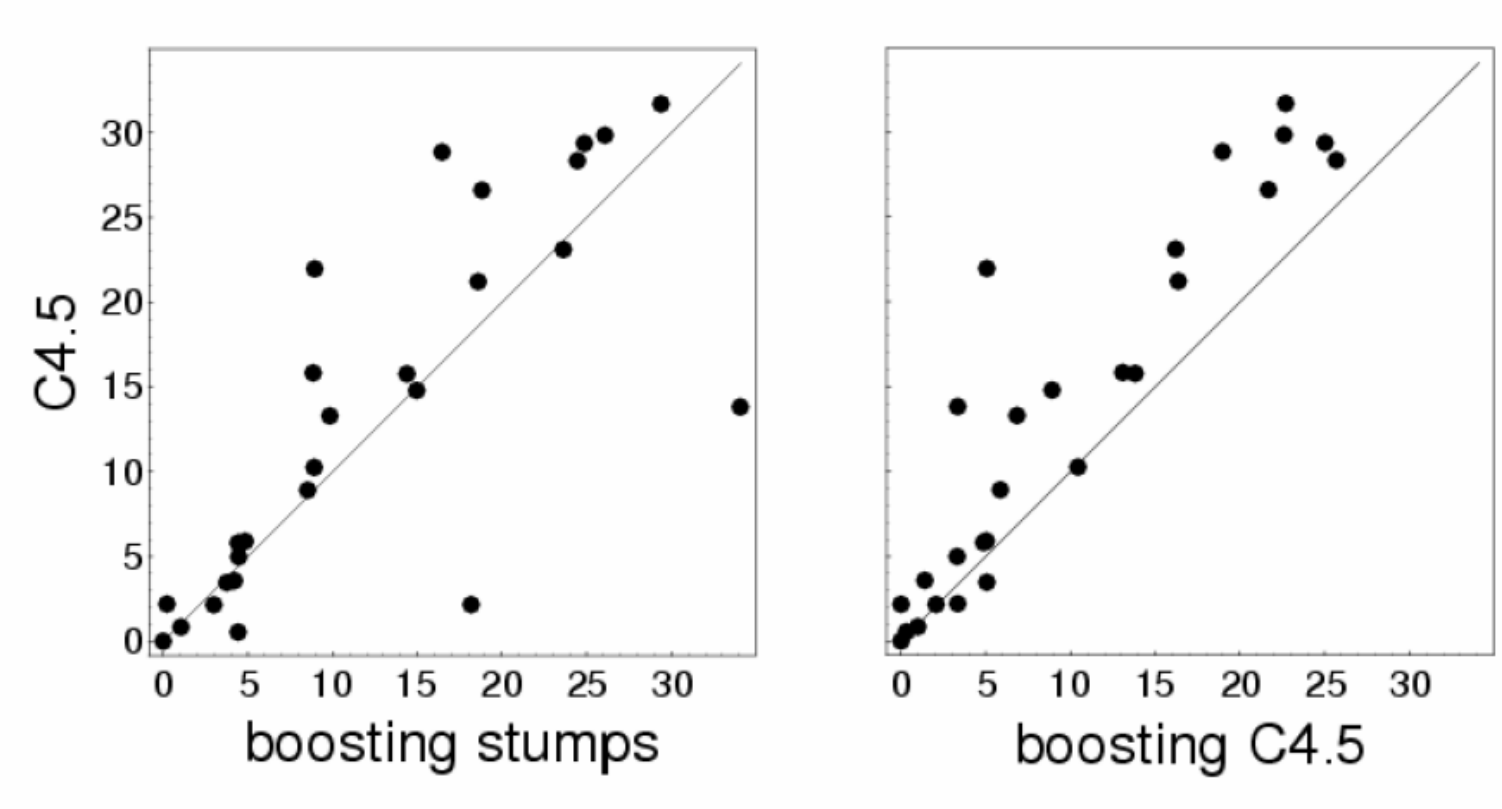
Among the most commonly used base classifiers – truly weak!

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

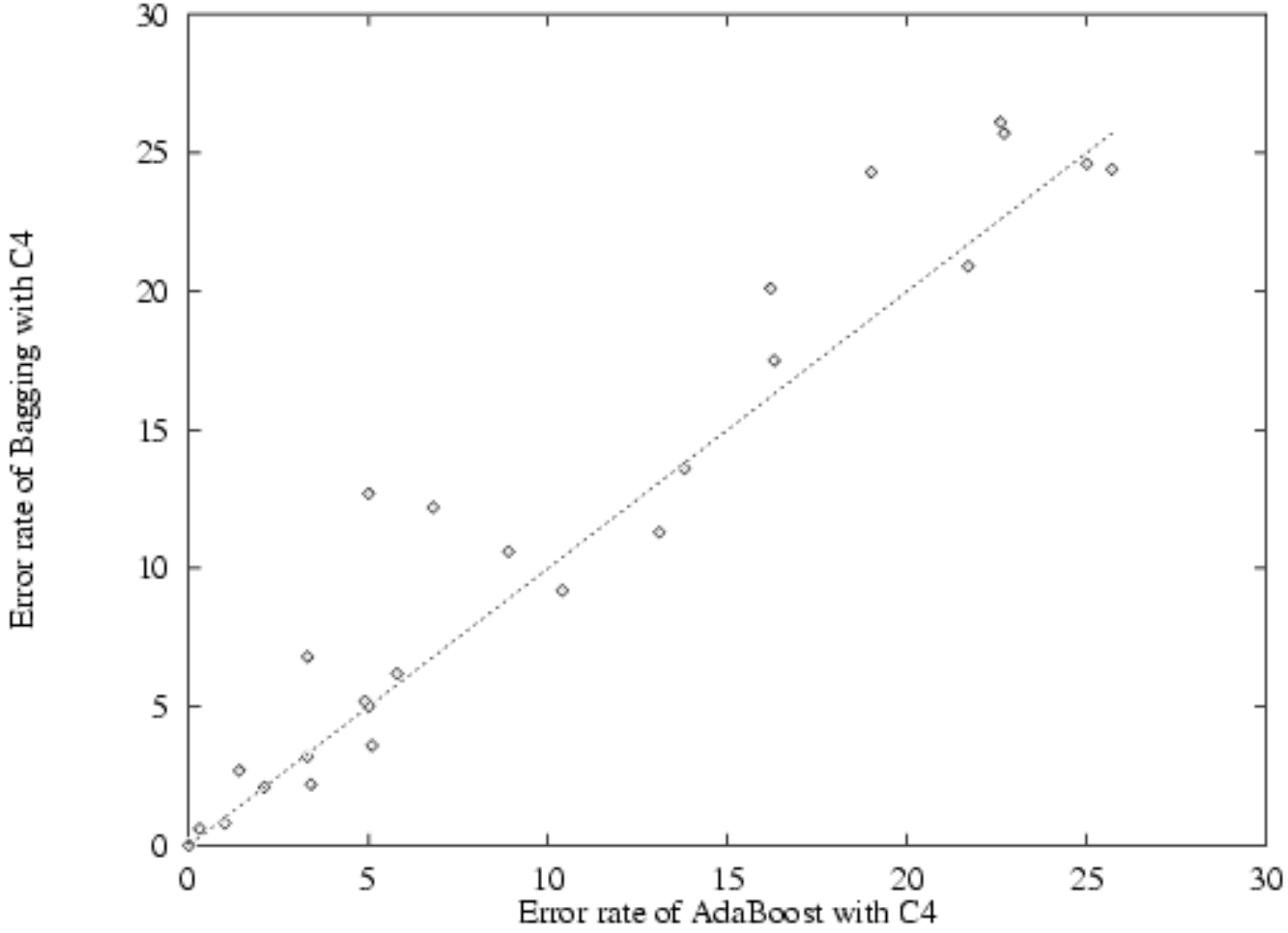


Boosting Performance

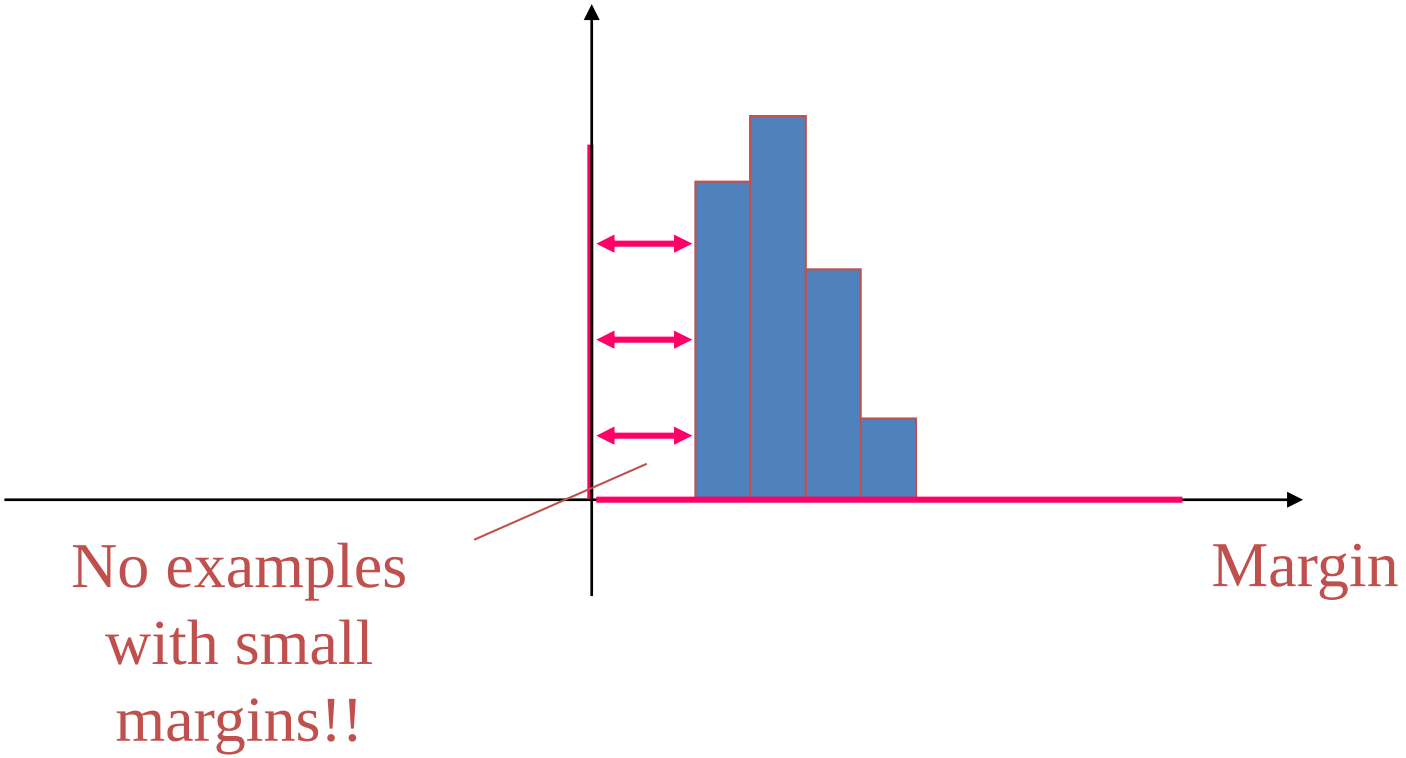
- Comparing C4.5, boosting decision stumps, boosting C4.5 using 27 UCI data set
  - C4.5 is a popular decision tree learner



Boosting vs Bagging of Decision Trees



Effect of Boosting: Maximizing Margin



Even after the boosted ensemble reaches zero training error, continued adding of more classifier tends to increase the margin of examples. This is one reason that the generalization error may continue decreasing for boosting.

Effect of Boosting on Bias/variance

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

Summary

- Bagging
  - Resample data points
  - Weight of each classifier is the same
  - Only variance reduction
  - Robust to noise and outliers
- Random forest
  - (Instance) Bagging + feature bagging
  - Increased diversity among the learned classifiers
  - Robust to noise and outliers
  - Efficient for large dataset
- Boosting
  - Reweight data points (modify data distribution)
  - Weight of classifier vary depending on accuracy
  - Reduces both bias and variance
  - Can hurt performance with noise and outliers