

Ensemble Learning

Based mainly on slides from
Cong Li

Ensembles Methods in ML

- **Ensemble Methods in Machine Learning**
- Bagging
- Boosting

Different Classifiers (1)

- **Different Classifiers**
 - Conduct classification on a same set of class labels
 - May use different input or have different parameters
 - May produce different output for a certain example
- **Learning Different Classifiers**
 - Use different training examples
 - Use different features

Different Classifiers (2)

■ Performance

- Each of the classifiers is not perfect
- Complementary
 - Examples which are not correctly classified by one classifier may be correctly classified by the other classifiers

■ Potential Improvements?

- Utilize the complementary property

Ensembles of Classifiers


















































- **Idea**

- Combine the classifiers to improve the performance

- **Ensembles of Classifiers**

- Combine the classification results from different classifiers to produce the final output
 - Unweighted voting
 - Weighted voting

Example: Weather Forecast

Reality							
1							
2							
3							
4							
5							
Combine							

Ensemble Learning

- **Ensemble Learning**
 - Relatively new field in machine learning
 - Achieve state-of-the-art performance
- **Central Issues in Ensemble Learning**
 - How to create classifiers with complementary performances
 - How to conduct voting

Outline

- Ensemble Methods in Machine Learning
- **Bagging**
- Boosting

Bagging

- **An Important Strategy for Ensemble Learning**
 - Create different training sets
- **Bootstrap AGGREGatING**
 - Take repeated bootstrap samples to create a sequence of training sets
 - Train classifiers using the training sets
 - Classification by majority voting

Replicating Data Sets

- **Original Training Set**

- $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$

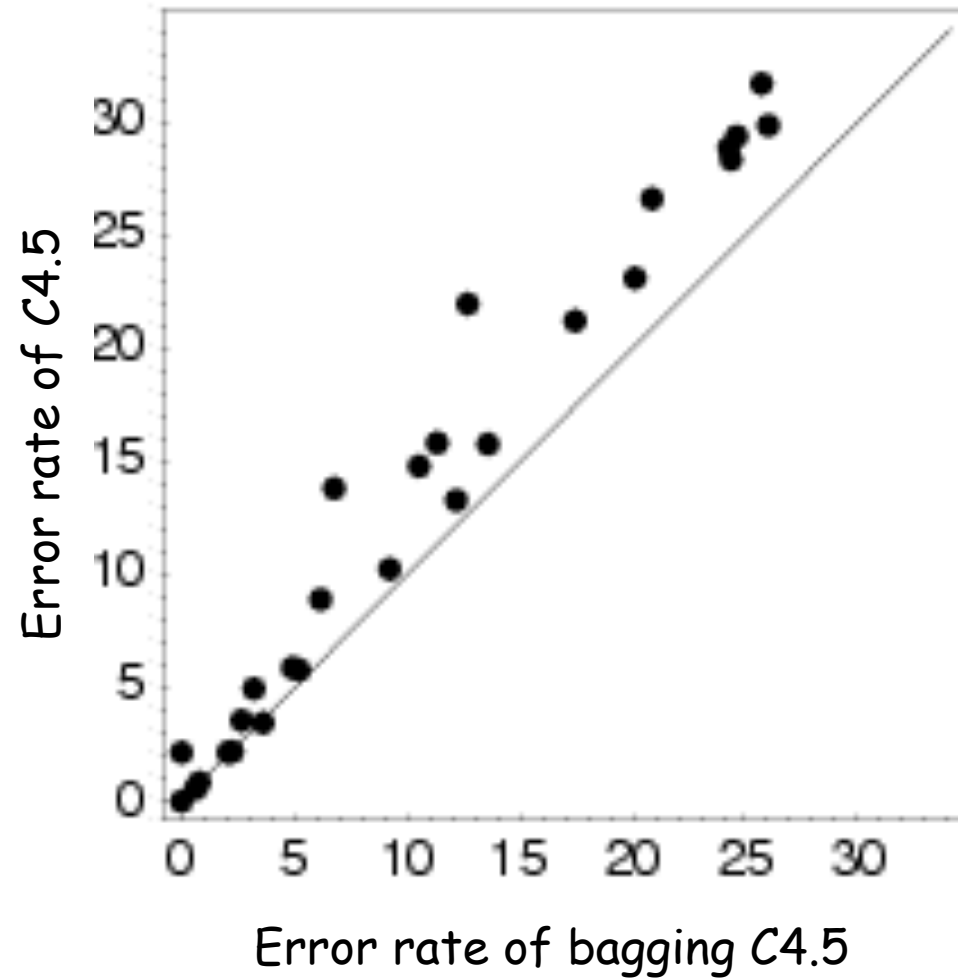
- **Sample with Replacement**

- Randomly draw m examples according to the uniform distribution on the original training set
- Allow duplicating and missing
- Used for training classifiers

Performance

- **Data Set**
 - 27 data sets from UCI ML Repository
- **Methods for Comparison**
 - Decision tree classifier: C4.5
 - Bagging: ensembles of 100 C4.5 classifiers

Results (Freund and Schapire 1996)



Seems to improve performance

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

Outline

- Ensemble Methods in Machine Learning
- Bagging
- **Boosting**

Boosting

- Ensemble Methods in Machine Learning
- Bagging
- **Boosting**
 - Basic Idea
 - AdaBoost Algorithm
 - Performance
 - Why AdaBoost Works

Strong and Weak Learners

■ Strong (PAC) Learner

- Take labeled data for training
- Produce one classifier which should be nearly accurate
- Objective of machine learning

■ Weak (PAC) Learner

- Take labeled data for training
- Produce a classifier which is more accurate than random guessing

Boosting

■ Learners

- Strong learners are very difficult to construct
- Constructing weaker Learners is relatively easy

■ Strategy

- Create a strong learner from weak learner
- Boost weak classifiers to a strong learner

Construct Weak Classifiers

- **Using Different Data Distribution**
 - Start with uniform weighting
 - During each step of learning
 - Increase weights of the examples which are not correctly learned by the weak learner
 - Decrease weights of the examples which are correctly learned by the weak learner
- **Idea**
 - Focus on difficult examples which are not correctly classified in the previous steps

Combine Weak Classifiers

■ **Weighted Voting**

- Construct strong classifier by weighted voting of the weak classifiers

■ **Idea**

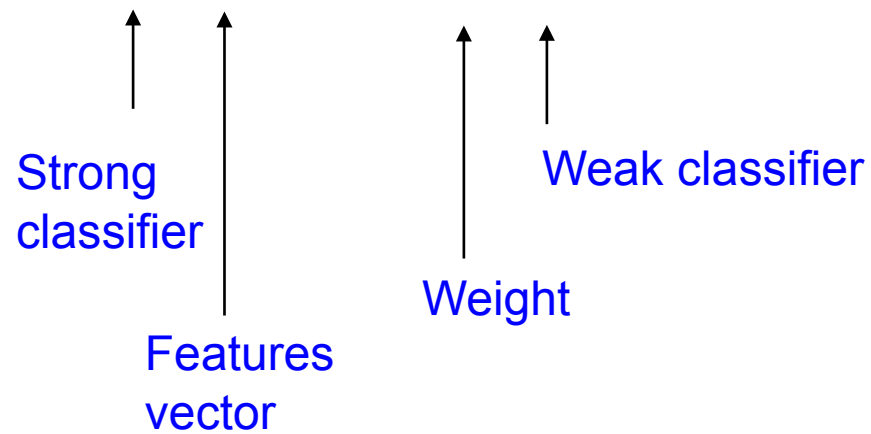
- Better weak classifier gets a larger weight
- Iteratively add weak classifiers
 - Increase accuracy of the combined classifier through minimization of a cost function

Boosting

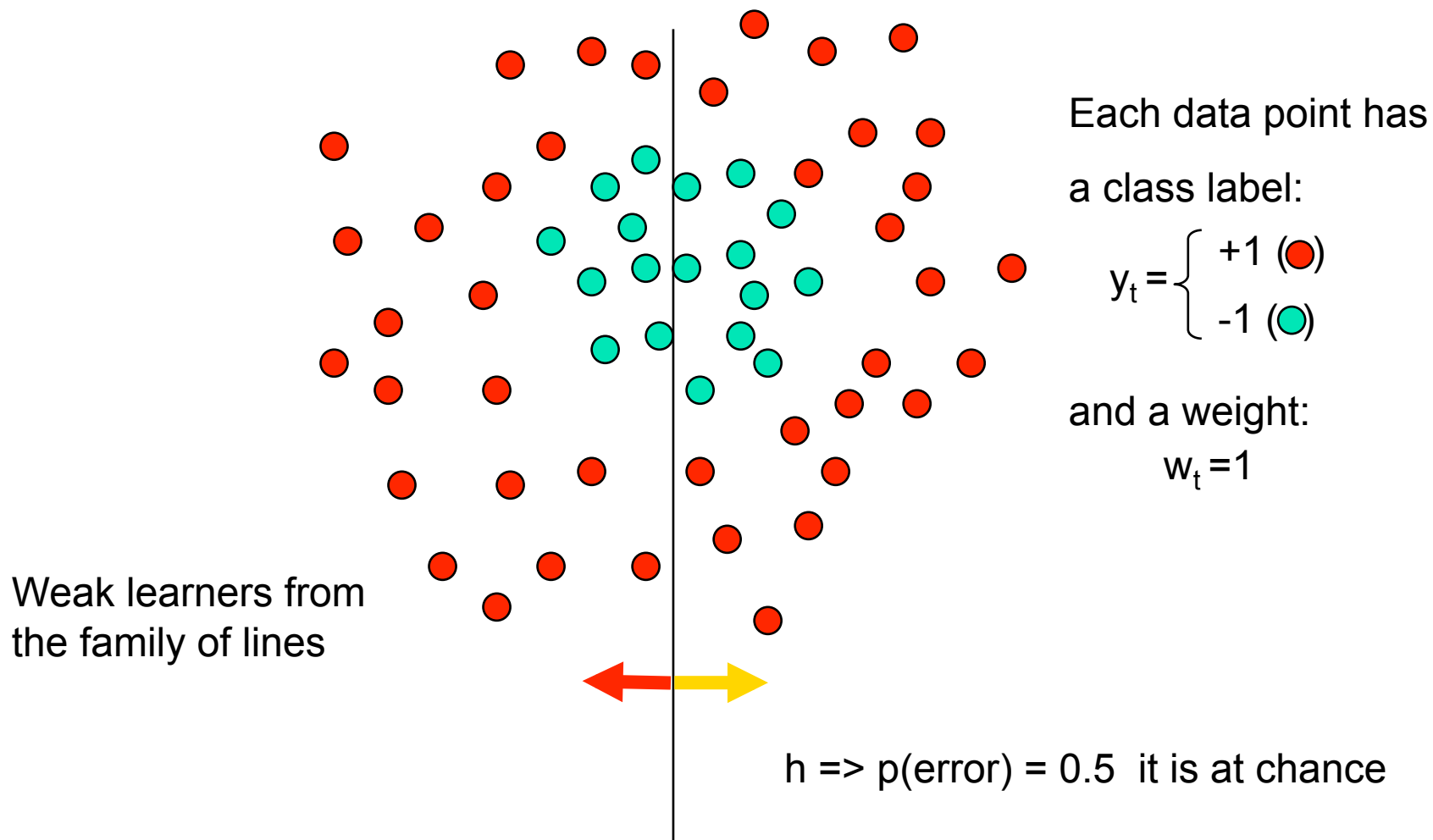
- Ensemble Methods in Machine Learning
- Bagging
- **Boosting**
 - Basic Idea
 - **AdaBoost Algorithm**
 - Performance
 - Why AdaBoost Works

Principle of Adaboost

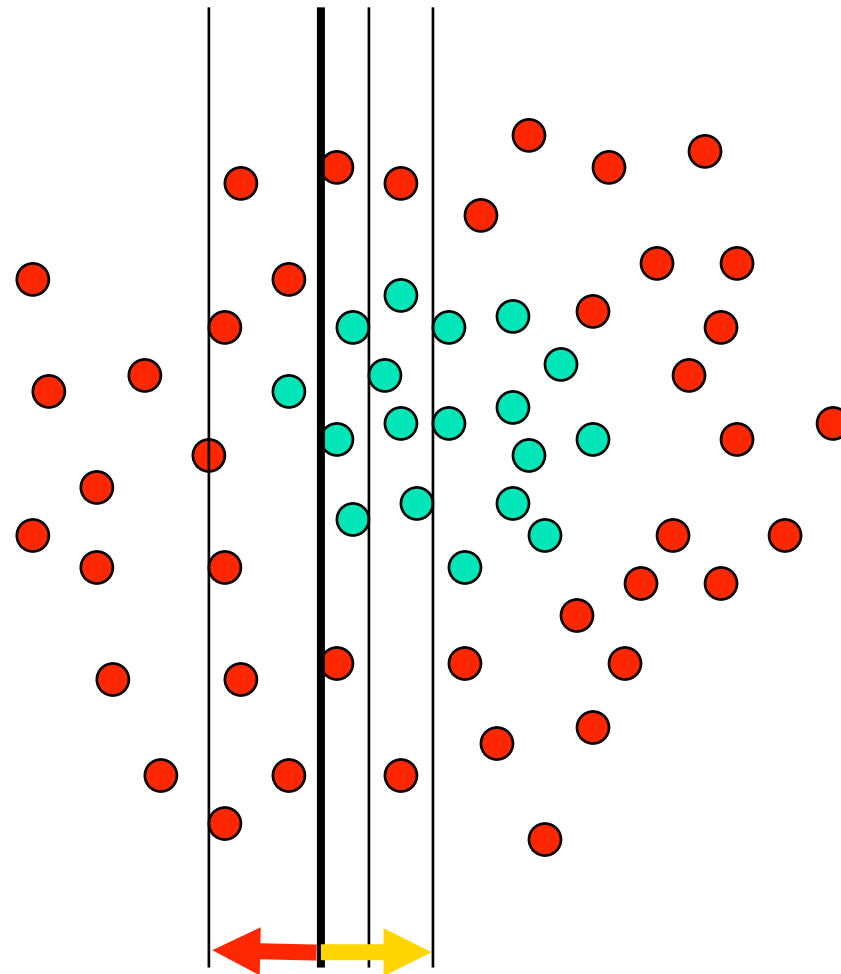
$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$



Toy Example - taken from Antonio Torralba @MIT



Toy example



Each data point has

a class label:

$$y_t = \begin{cases} +1 & (\text{red circle}) \\ -1 & (\text{cyan circle}) \end{cases}$$

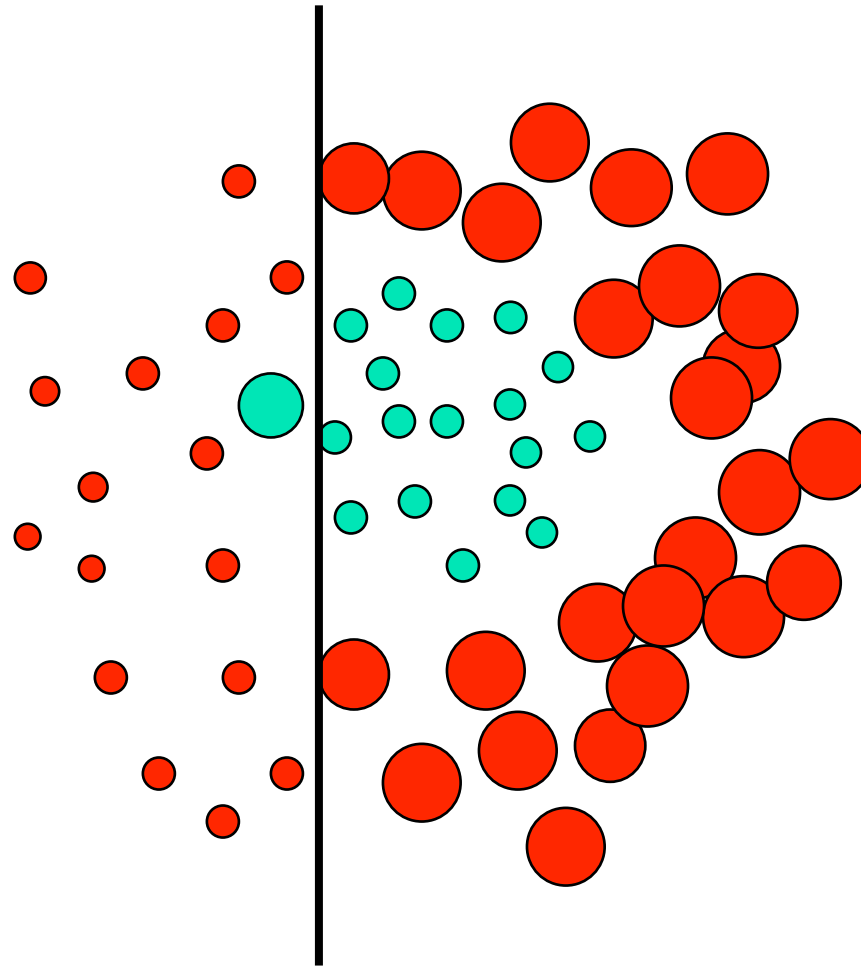
and a weight:

$$w_t = 1$$

This one seems to be the best

This is a '**weak classifier**': It performs slightly better than chance.

Toy example



Each data point has
a class label:

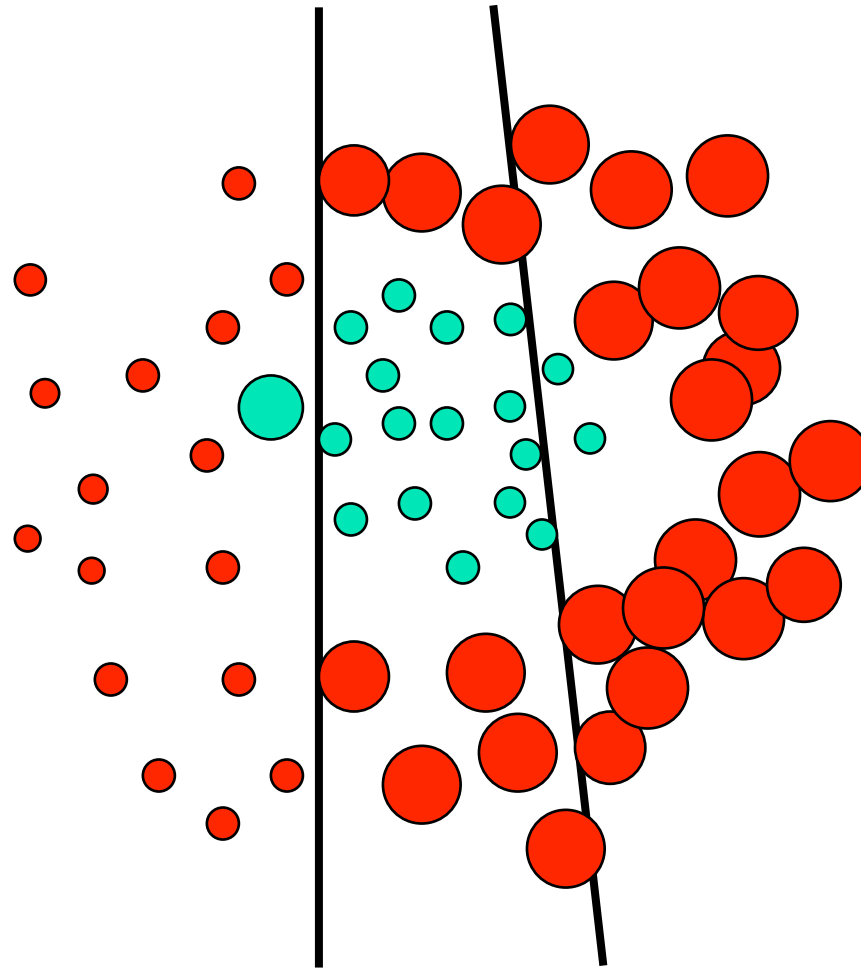
$$y_t = \begin{cases} +1 & (\text{red circle}) \\ -1 & (\text{cyan circle}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has
a class label:

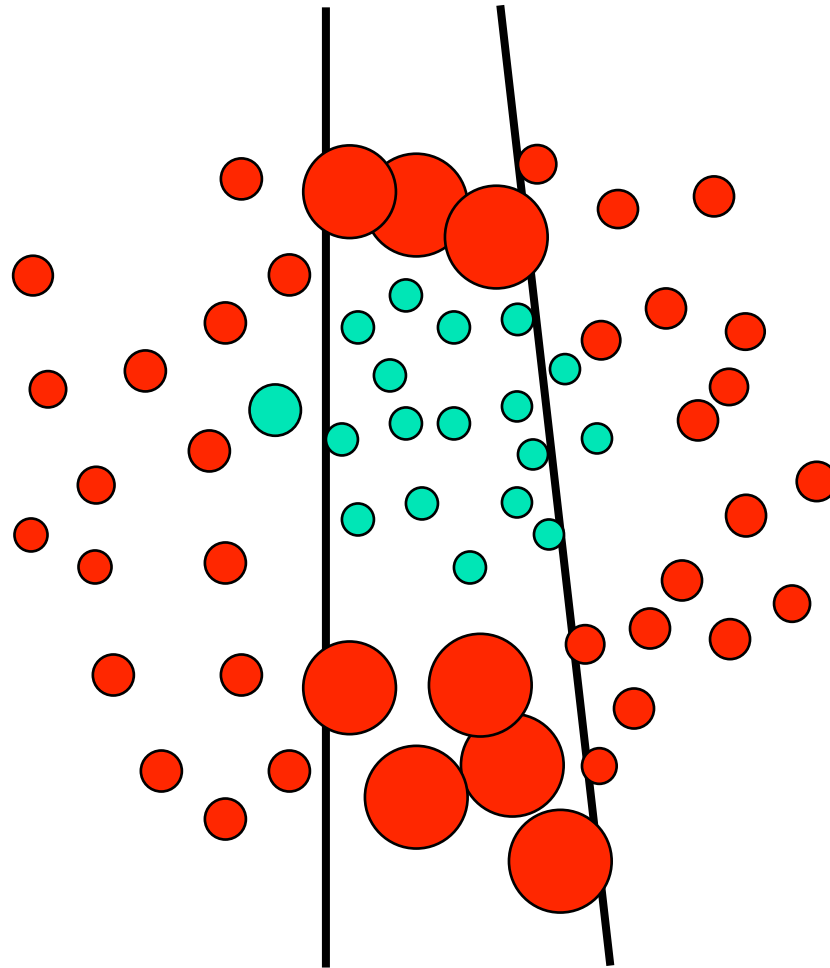
$$y_t = \begin{cases} +1 & (\text{red circle}) \\ -1 & (\text{cyan circle}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has
a class label:

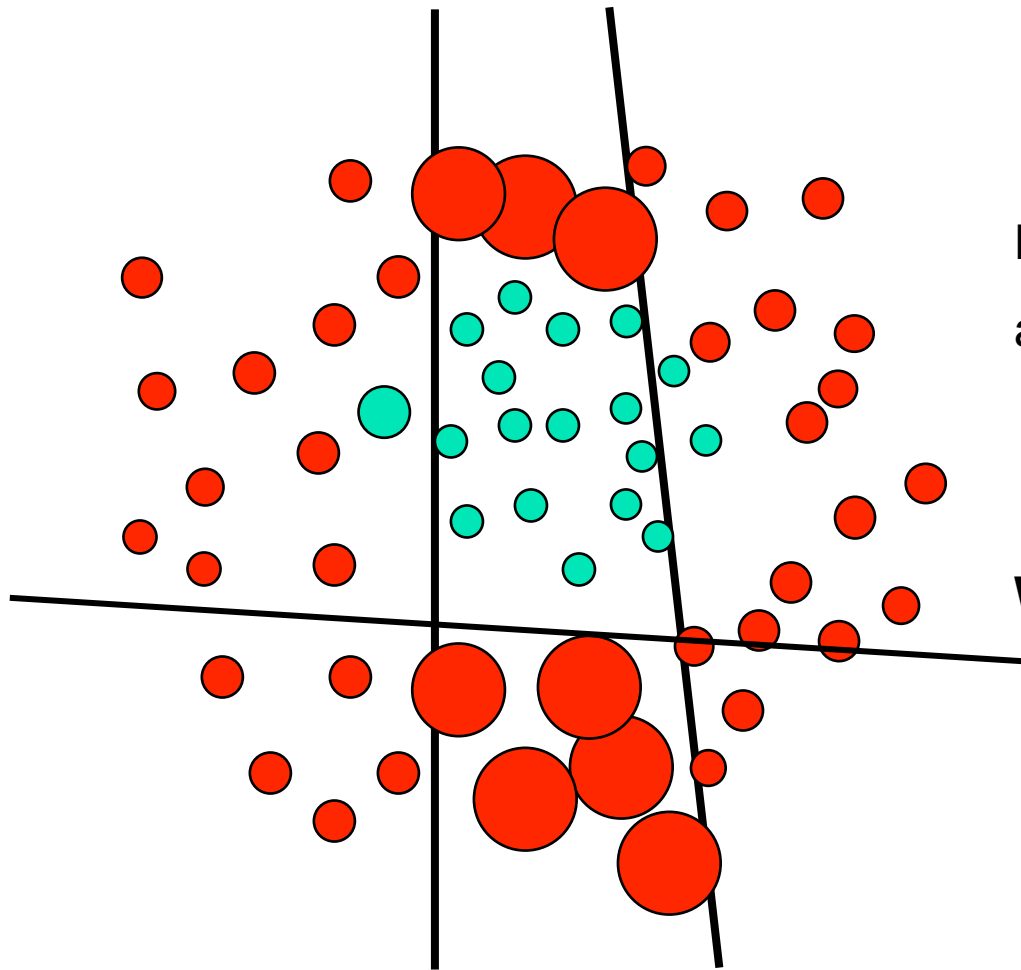
$$y_t = \begin{cases} +1 & (\text{red circle}) \\ -1 & (\text{cyan circle}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has
a class label:

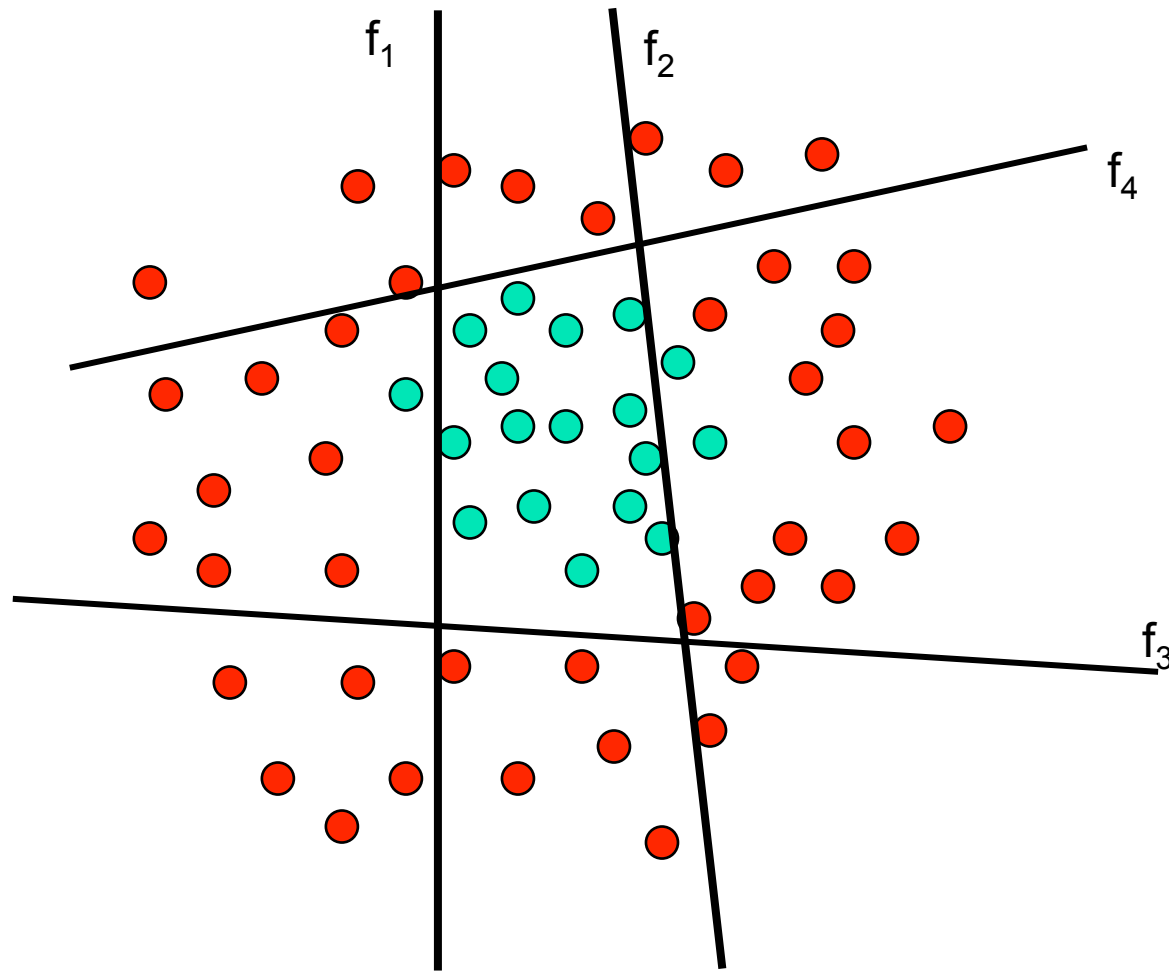
$$y_t = \begin{cases} +1 & (\text{red circle}) \\ -1 & (\text{cyan circle}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



The strong (non-linear) classifier is built as the combination of all the weak (linear) classifiers.

Formal Procedure of AdaBoost

- given training set $(x_1, y_1), \dots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \dots, T$:
 - construct distribution D_t on $\{1, \dots, m\}$
 - find weak classifier (“rule of thumb”)
$$h_t : X \rightarrow \{-1, +1\}$$
with small error ϵ_t on D_t :
$$\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$$
- output final classifier H_{final}

Procedure of Adaboost

- constructing D_t :

- $D_1(i) = 1/m$

- given D_t and h_t :

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

where $Z_t =$ normalization constant

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

- final classifier:

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

AdaBoost: Final

- **Output**

$$H(\mathbf{x}) = \text{sgn}\left[\sum_t \alpha_t h_t(\mathbf{x})\right]$$

- **Margin Classifier**

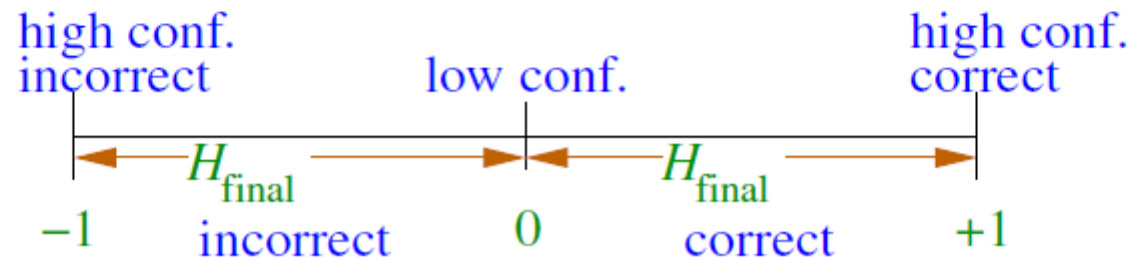
- Margin in majority vote classifiers

$$\text{Margin}(\mathbf{x}^{(i)}, y^{(i)}) = y^{(i)} \frac{\sum_t \alpha_t h_t(\mathbf{x}^{(i)})}{\sum_t \alpha_t}$$

- Related to generalization error
- AdaBoost **often** optimizes the margins

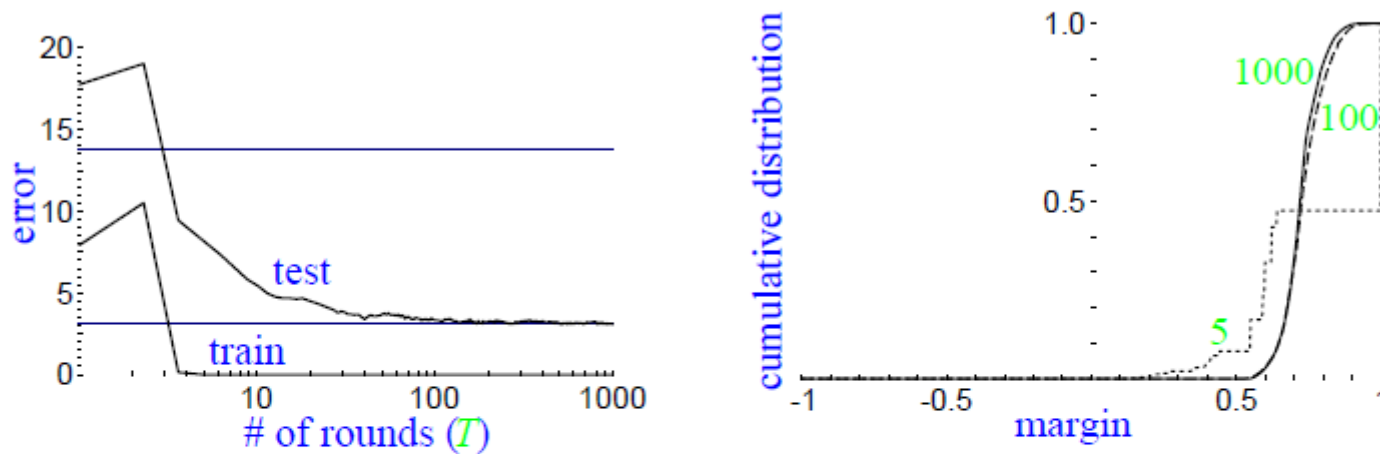
The margin explanation

- 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
= (fraction voting correctly) – (fraction voting incorrectly)



The margin explanation

- 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

Boosting

- Ensemble Methods in Machine Learning
- Bagging
- **Boosting**
 - Basic Idea
 - AdaBoost Algorithm
 - Performance
 - Why AdaBoost Works

Performance

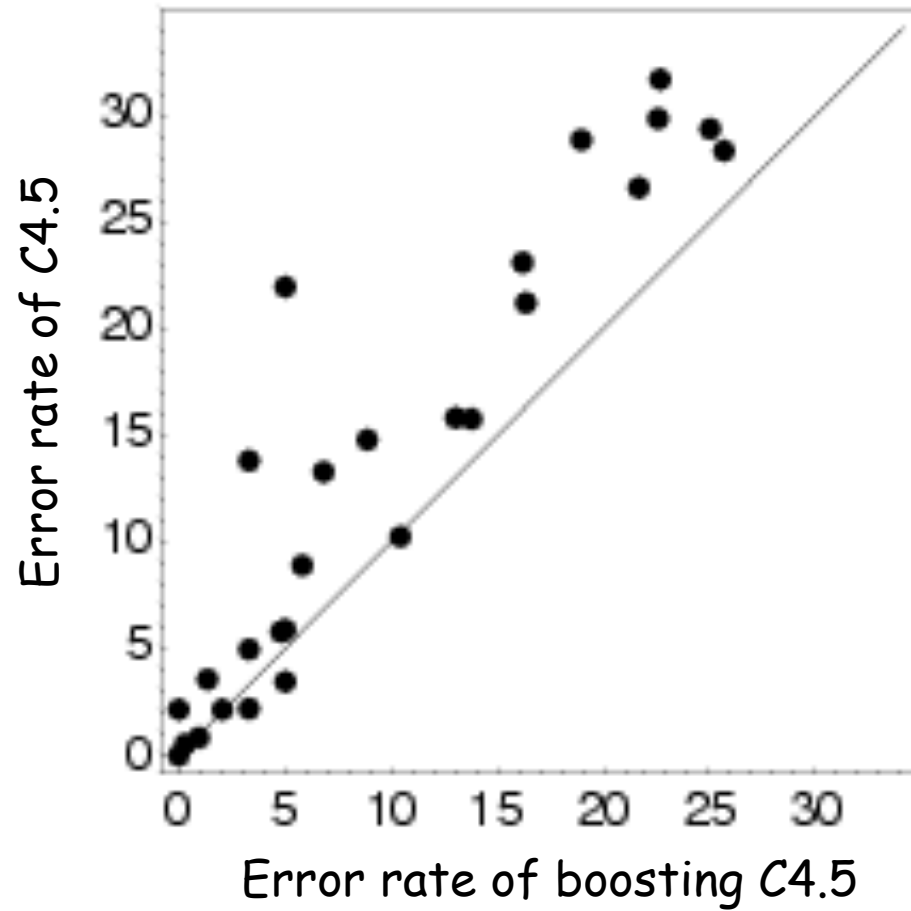
- **Data Set**

- 27 data sets from UCI ML Repository

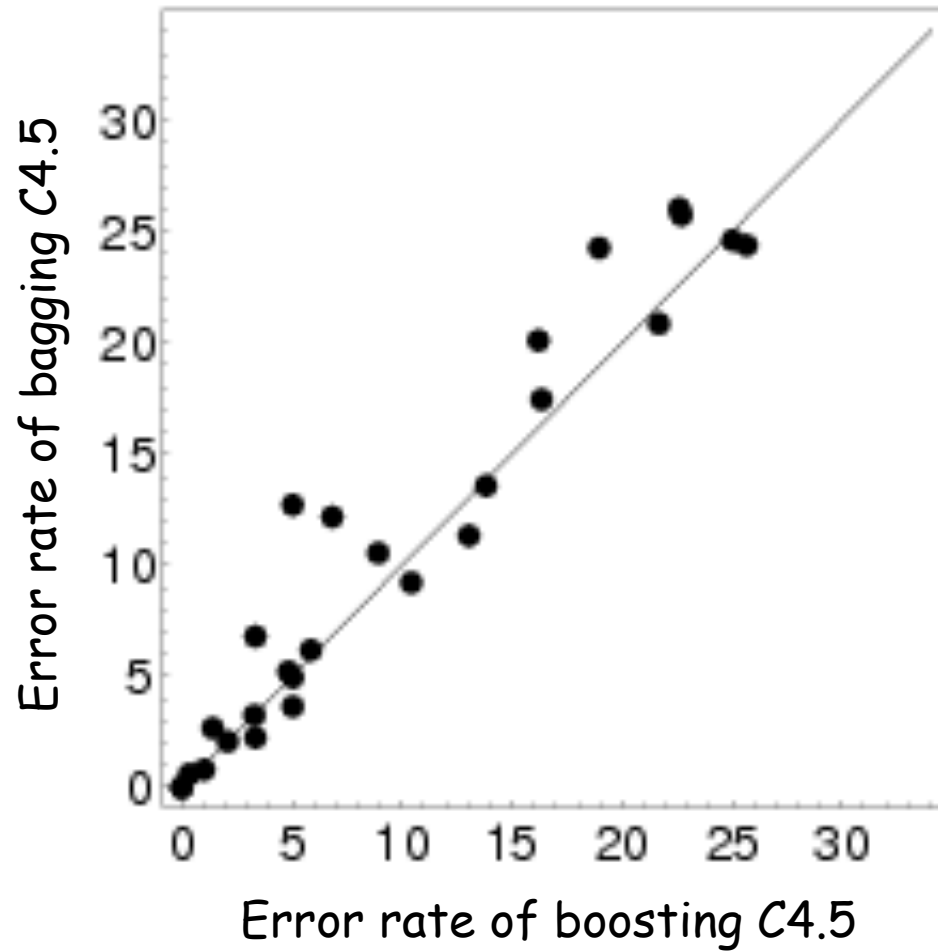
- **Methods for Comparison**

- Decision tree classifier: C4.5
- Bagging: ensembles of 100 C4.5 classifiers
- Boosting: AdaBoost using C4.5 as the weak learner

Results (Freund and Schapire 1996)



Results (Freund and Schapire 1996)



Boosting

- Ensemble Methods in Machine Learning
- Bagging
- **Boosting**
 - Basic Idea
 - AdaBoost Algorithm
 - Performance
 - **Why AdaBoost Works**

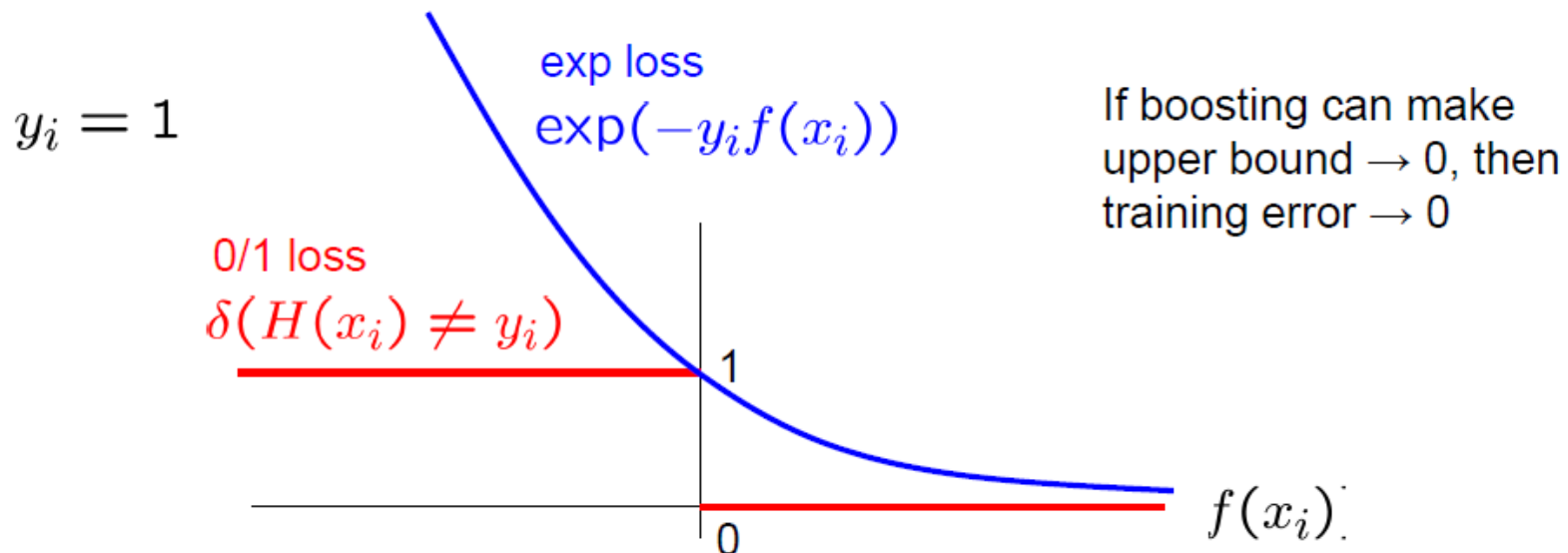
Analyzing training error

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^m \exp(-y_i f(x_i))$$

Convex
upper
bound

Where $f(x) = \sum_t \alpha_t h_t(x)$; $H(x) = \text{sign}(f(x))$



Analyzing training error

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^m \exp(-y_i f(x_i)) = \prod_t Z_t$$

Where $f(x) = \sum_t \alpha_t h_t(x)$; $H(x) = \text{sign}(f(x))$

Proof: Using Weight Update Rule

$$\begin{aligned} D_1(i) &= 1/m. \\ D_2(i) &= \frac{1}{m} \frac{e^{-\alpha_1 y_i h_1(x_i)}}{Z_1} \\ D_3(i) &= \frac{1}{m} \frac{e^{-\alpha_1 y_i h_1(x_i)} e^{-\alpha_2 y_i h_2(x_i)}}{Z_1 Z_2} \\ &\dots \end{aligned}$$

$$D_{T+1}(i) = \frac{1}{m} \frac{\exp(-y_i f(x_i))}{\prod_t Z_t}$$

Wts of all pts add to 1

$$\sum_{i=1}^m D_{T+1}(i) = 1$$

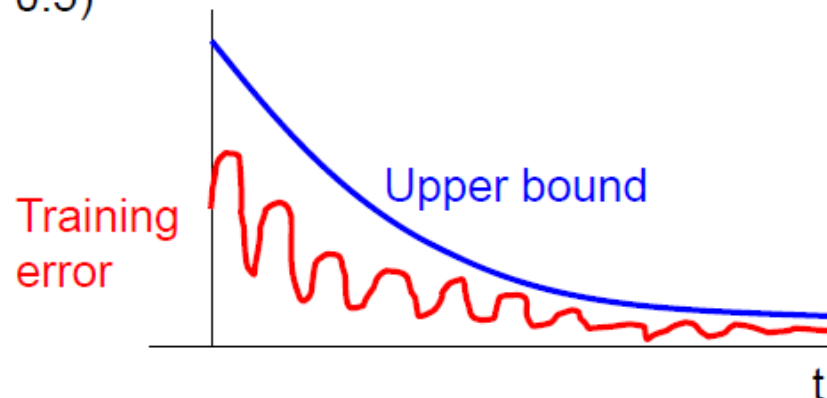
Analyzing training error

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^m \exp(-y_i f(x_i)) = \prod_t Z_t$$

Where $f(x) = \sum_t \alpha_t h_t(x)$; $H(x) = \text{sign}(f(x))$

If $Z_t < 1$, training error decreases exponentially (even though weak learners may not be good $\varepsilon_t \sim 0.5$)



What α_t to choose?

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^m \exp(-y_i f(x_i)) = \prod_t Z_t$$

Where $f(x) = \sum_t \alpha_t h_t(x)$; $H(x) = \text{sign}(f(x))$

If we minimize $\prod_t Z_t$, we minimize our training error

We can tighten this bound greedily, by choosing α_t and h_t on each iteration to minimize Z_t .

$$Z_t = \sum_{i=1}^m D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

What α_t to choose?

We can minimize this bound by choosing α_t on each iteration to minimize Z_t .

$$Z_t = \sum_{i=1}^m D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

For boolean target function, this is accomplished by [Freund & Schapire '97]:

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

Proof:
$$\begin{aligned} Z_t &= \sum_{i: y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} + \sum_{i: y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} \\ &= \epsilon_t e^{\alpha_t} + (1 - \epsilon_t) e^{-\alpha_t} \end{aligned}$$

$$\frac{\partial Z_t}{\partial \alpha_t} = \epsilon_t e^{\alpha_t} - (1 - \epsilon_t) e^{-\alpha_t} = 0 \quad \Rightarrow \quad e^{2\alpha_t} = \frac{1 - \epsilon_t}{\epsilon_t}$$

What α_t to choose?

We can minimize this bound by choosing α_t on each iteration to minimize Z_t .

$$Z_t = \sum_{i=1}^m D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

For boolean target function, this is accomplished by [Freund & Schapire '97]:

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

Proof:

$$\begin{aligned} Z_t &= \sum_{i: y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} + \sum_{i: y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} \\ &= \epsilon_t e^{\alpha_t} + (1 - \epsilon_t) e^{-\alpha_t} \\ &= 2\sqrt{\epsilon_t(1 - \epsilon_t)} = \sqrt{1 - (1 - 2\epsilon_t)^2} \end{aligned}$$

Dumb classifiers made smart

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \prod_t Z_t = \prod_t \sqrt{1 - (1 - 2\epsilon_t)^2}$$
$$\leq \exp\left(-2 \sum_{t=1}^T \underbrace{(1/2 - \epsilon_t)^2}_{\substack{\text{grows as } \epsilon_t \text{ moves} \\ \text{away from } 1/2}}\right)$$

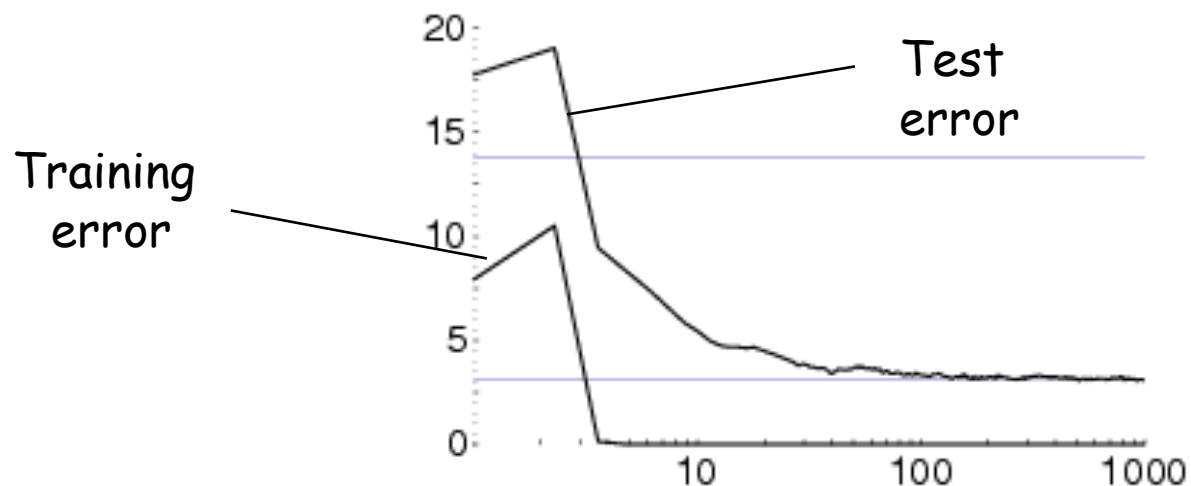
If each classifier is (at least slightly) better than random $\epsilon_t < 0.5$

AdaBoost will achieve zero training error exponentially fast (in number of rounds T) !!

What about test error?

Training Errors vs Test Errors

Performance on 'letter' dataset
(Schapire et al. 1997)



Training error drops to 0 on round 5

Test error continues to drop after round 5
(from 8.4% to 3.1%)

Summary

- AdaBoost is a sequential algorithm that minimizes an upper bound of the empirical classification error by selecting the weak classifiers and their weights. These are “pursued” one-by-one with each one being selected to maximally reduce the upper bound of error.
- AdaBoost defines a distribution of weights over the data samples. These weights are updated each time a new weak classifier is added such that samples misclassified by this new weak classifiers are given more weight. In this manner, currently misclassified samples are emphasized more during the selection of the subsequent weak classifier.
- The empirical error will converge to zero at an exponential rate.

Summary

- It is fast to evaluate (linear-additive) and can be fast to train (depending on weak learner).
- T is the only parameter to tune.
- It is flexible and can be combined with any weak learner.
- It is provably effective if it can consistently find the weak classifiers (that do better than random).
- Since it can work with any weak learner, it can handle the gamut of data.

Caveats

- Performance depends on the data and the weak learner.
- It can fail if
 - The weak classifiers are too complex and overfit.
 - The weak classifiers are too weak, essentially underfitting.
- AdaBoost seems, empirically, to be especially susceptible to uniform noise.