

# Boosting

**Dr. Jianlin Cheng**

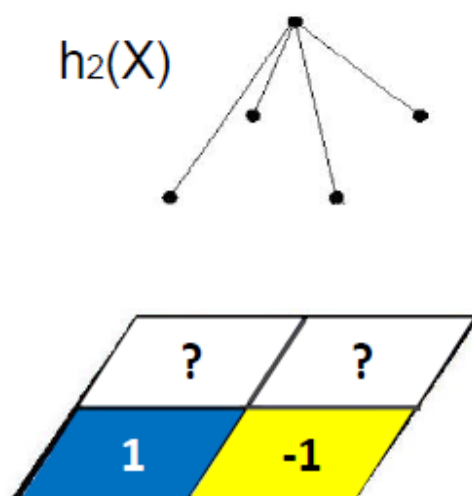
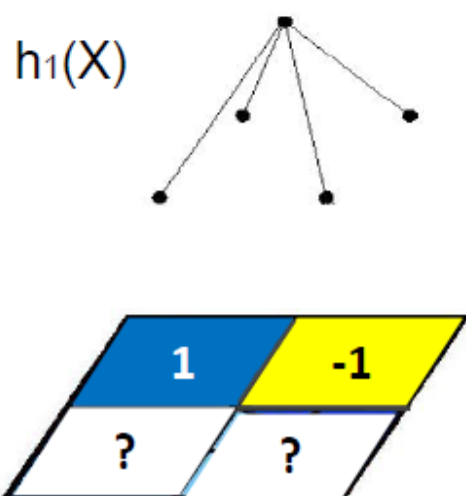
**Computer Science Department  
University of Missouri, Columbia  
Fall, 2015**

Slides Adapted from Book and CMU, Stanford Machine Learning Courses



# Voting (Ensemble Methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**
- **Output class:** (Weighted) vote of each classifier
  - Classifiers that are most “sure” will vote with more conviction
  - Classifiers will be most “sure” about a particular part of the space
  - On average, do better than single classifier!



$$H: X \rightarrow Y (-1, 1)$$

$$H(X) = h_1(X) + h_2(X)$$

$$H(X) = \text{sign}\left(\sum_t \alpha_t h_t(X)\right)$$

↓  
weights

# Voting (Ensemble Methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**
- **Output class:** (Weighted) vote of each classifier
  - Classifiers that are most “sure” will vote with more conviction
  - Classifiers will be most “sure” about a particular part of the space
  - On average, do better than single classifier!
- **But how do you ???**
  - force classifiers  $h_t$  to learn about different parts of the input space?
  - weigh the votes of different classifiers?  $\alpha_t$

# Boosting [Schapire'89]

- **Idea:** given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote
- On each iteration  $t$ :
  - weight each training example by how incorrectly it was classified
  - Learn a weak hypothesis –  $h_t$
  - A strength for this hypothesis –  $\alpha_t$
- Final classifier:  $H(X) = \text{sign}(\sum \alpha_t h_t(X))$
- **Practically useful**
- **Theoretically interesting**

# Learning from weighted data

- **Consider a weighted dataset**
  - $D(i)$  – weight of  $i$  th training example  $(\mathbf{x}^i, y^i)$
  - Interpretations:
    - $i$  th training example counts as  $D(i)$  examples
    - If I were to “resample” data, I would get more samples of “heavier” data points
- **Now, in all calculations, whenever used,  $i$  th training example counts as  $D(i)$  “examples”**
  - e.g., in MLE redefine  $Count(Y=y)$  to be weighted count

## Unweighted data

$$Count(Y=y) = \sum_{i=1}^m \mathbf{1}(Y^i=y)$$

## Weights $D(i)$

$$Count(Y=y) = \sum_{i=1}^m D(i) \mathbf{1}(Y^i=y)$$

# AdaBoost [Freund & Schapire'95]

Given:  $(x_1, y_1), \dots, (x_m, y_m)$  where  $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize  $D_1(i) = 1/m$ . **Initially equal weights**

For  $t = 1, \dots, T$ :

- Train weak learner using distribution  $D_t$ . **Naïve bayes, decision stump**

- Get weak classifier  $h_t : X \rightarrow \mathbb{R}$ .

- Choose  $\alpha_t \in \mathbb{R}$ . **Magic (+ve)**

- Update:

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

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

**Increase weight  
if wrong on pt i  
 $y_i h_t(x_i) = -1 < 0$**

where  $Z_t$  is a normalization factor

# AdaBoost [Freund & Schapire'95]

Given:  $(x_1, y_1), \dots, (x_m, y_m)$  where  $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize  $D_1(i) = 1/m$ . **Initially equal weights**

For  $t = 1, \dots, T$ :

- Train weak learner using distribution  $D_t$ . **Naïve bayes, decision stump**

- Get weak classifier  $h_t : X \rightarrow \mathbb{R}$ .

- Choose  $\alpha_t \in \mathbb{R}$ . **Magic (+ve)**

- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight  
if wrong on pt i  
 $y_i h_t(x_i) = -1 < 0$**

where  $Z_t$  is a normalization factor

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

**Weights for all  
pts must sum to 1  
 $\sum_t D_{t+1}(i) = 1$**



# AdaBoost [Freund & Schapire'95]

Given:  $(x_1, y_1), \dots, (x_m, y_m)$  where  $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize  $D_1(i) = 1/m$ . **Initially equal weights**

For  $t = 1, \dots, T$ :

- Train weak learner using distribution  $D_t$ . **Naïve bayes, decision stump**
- Get weak classifier  $h_t : X \rightarrow \mathbb{R}$ .
- Choose  $\alpha_t \in \mathbb{R}$ . **Magic (+ve)**

- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight  
if wrong on pt i  
 $y_i h_t(x_i) = -1 < 0$**

where  $Z_t$  is a normalization factor

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right).$$

# What $\alpha_t$ to choose for hypothesis $h_t$ ?

Weight Update Rule: 
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right) \quad \text{[Freund & Schapire'95]}$$

## Weighted training error

$$\epsilon_t = P_{i \sim D_t(i)} [h_t(\mathbf{x}^i) \neq y^i] = \sum_{i=1}^m D_t(i) \underbrace{\delta(h_t(x_i) \neq y_i)}_{\text{Does } h_t \text{ get } i^{\text{th}} \text{ point wrong}}$$

$\epsilon_t = 0$  if  $h_t$  perfectly classifies all weighted data pts

$\epsilon_t = 1$  if  $h_t$  perfectly wrong  $\Rightarrow$   $-h_t$  perfectly right

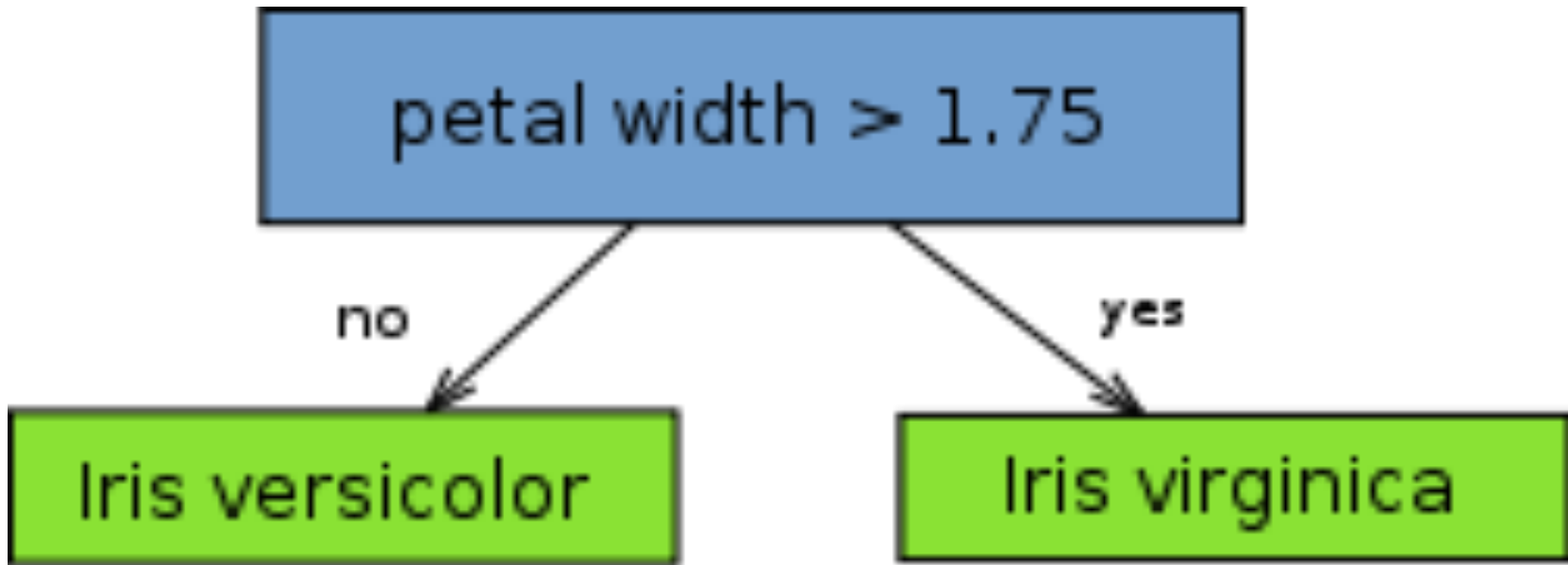
$\epsilon_t = 0.5$

$\alpha_t = \infty$

$\alpha_t = -\infty$

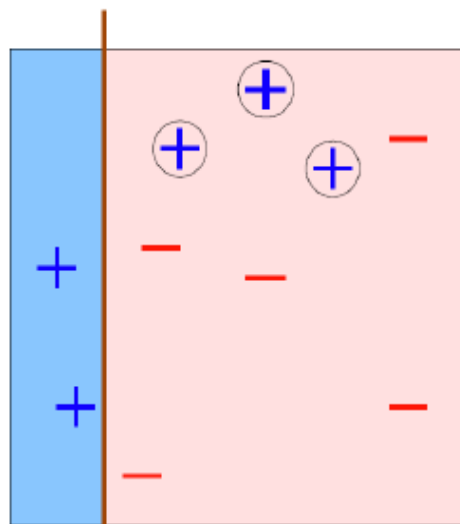
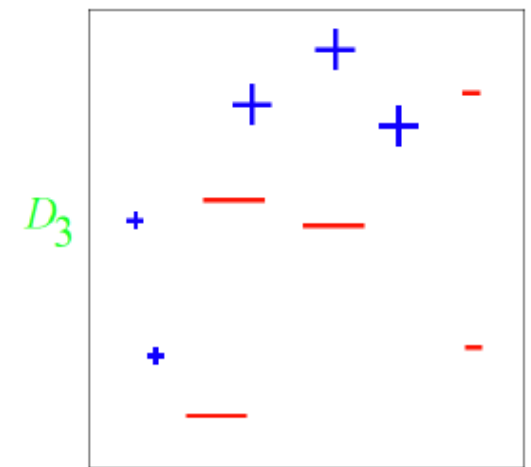
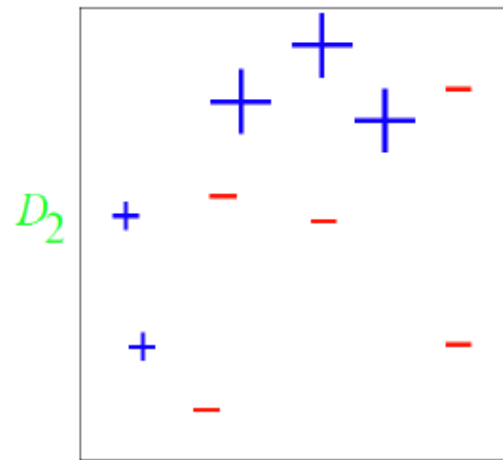
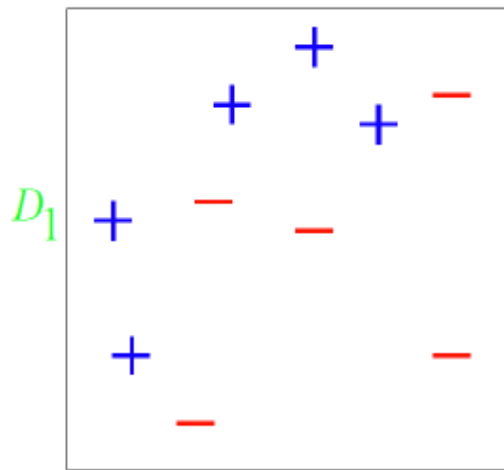
$\alpha_t = 0$

# Decision Stump

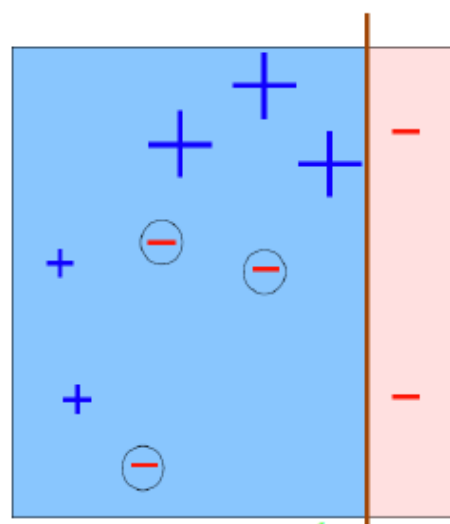


Source: Wikipedia

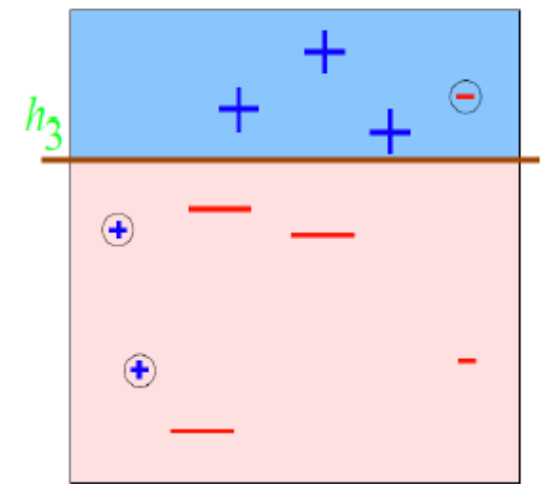
# Boosting Example (Decision Stumps)



$h_1$   
 $\epsilon_1 = 0.30$   
 $\alpha_1 = 0.42$



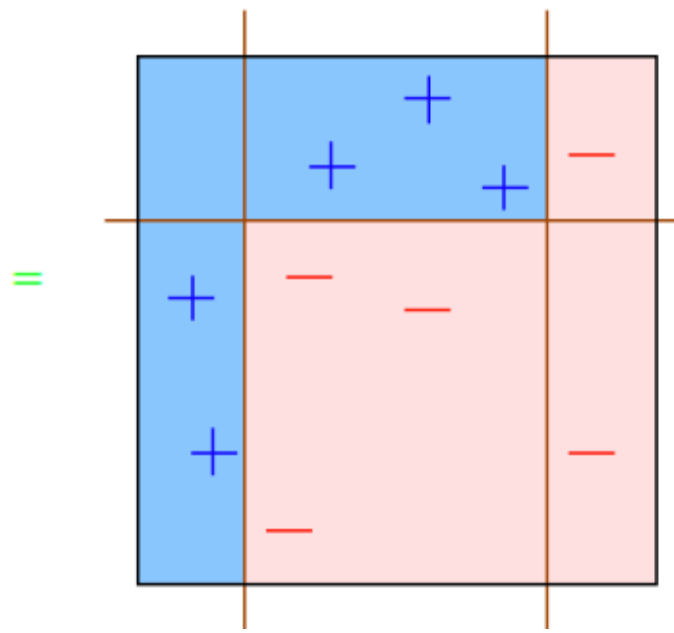
$\epsilon_2 = 0.21$   
 $\alpha_2 = 0.65$   
 $h_2$



$h_3$   
 $\epsilon_3 = 0.14$   
 $\alpha_3 = 0.92$

# Boosting Example (Decision Stumps)

$$H_{\text{final}} = \text{sign} \left( 0.42 \left[ \begin{array}{|c|} \hline \text{blue} \\ \hline \text{red} \end{array} \right] + 0.65 \left[ \begin{array}{|c|} \hline \text{blue} \\ \hline \text{red} \end{array} \right] + 0.92 \left[ \begin{array}{|c|} \hline \text{blue} \\ \hline \text{red} \end{array} \right] \right)$$



# Analyzing training error

Analysis reveals:

- What  $\alpha_t$  to choose for hypothesis  $h_t$ ?

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

$\epsilon_t$  - weighted training error

- If each weak learner  $h_t$  is slightly better than random guessing ( $\epsilon_t < 0.5$ ), then training error of AdaBoost decays exponentially fast in number of rounds  $T$ .

$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \exp \left( -2 \sum_{t=1}^T (1/2 - \epsilon_t)^2 \right)$$

**Training Error**

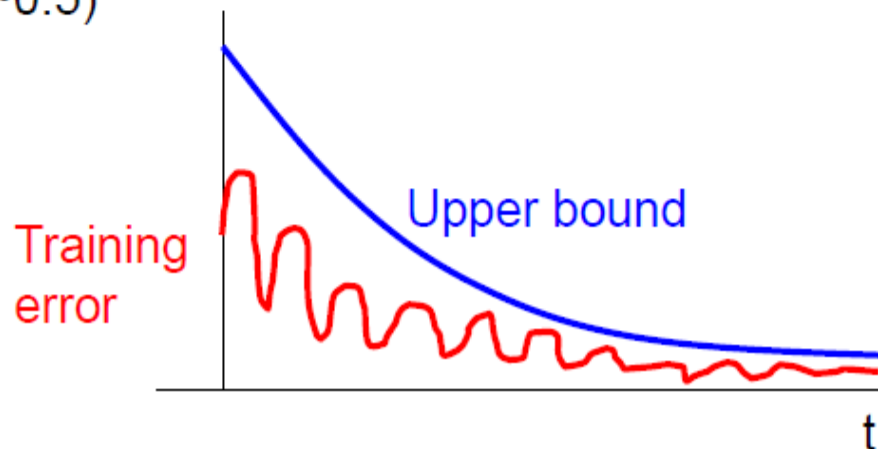
# Analyzing training error

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 $\alpha_t$ to choose for hypothesis $h_t$ ?

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  $\alpha_t$  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 $\alpha_t$ to choose for hypothesis $h_t$ ?

We can minimize this bound by choosing  $\alpha_t$  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 $\alpha_t$ to choose for hypothesis $h_t$ ?

We can minimize this bound by choosing  $\alpha_t$  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:

$$\frac{1}{m} \sum_{i=1}^m \delta(H(x_i) \neq y_i) \leq \prod_t Z_t = \prod_t \sqrt{1 - (1 - 2\epsilon_t)^2}$$

$$\leq \exp\left(-2 \sum_{t=1}^T \underbrace{(1/2 - \epsilon_t)^2}_{\text{grows as } \epsilon_t \text{ moves away from } 1/2}\right)$$

grows as  $\epsilon_t$  moves away from 1/2

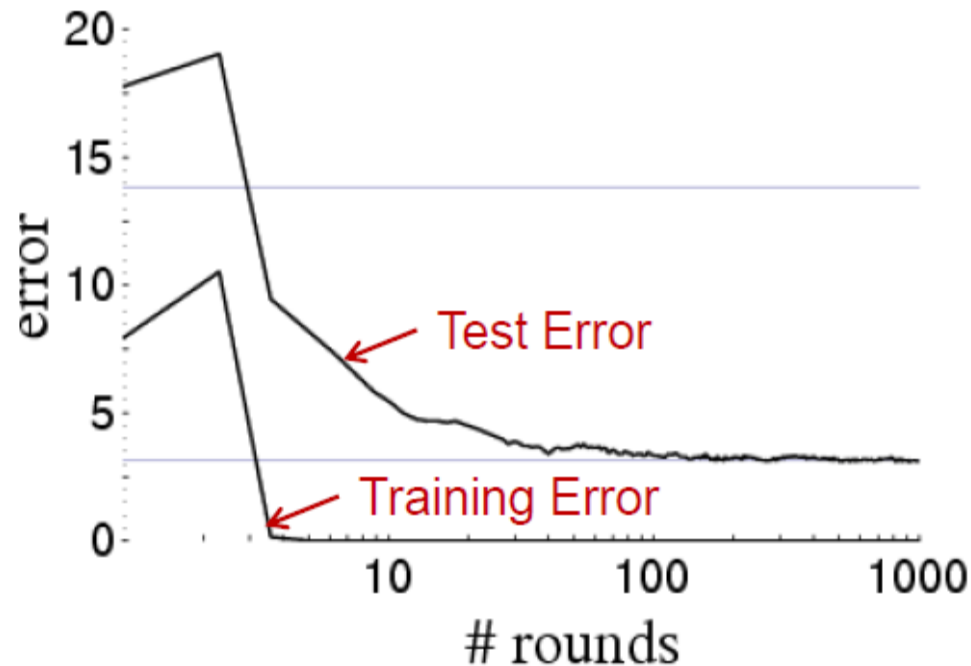
If each classifier is (at least slightly) better than random  $\epsilon_t < 0.5$

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

What about test error?

# Boosting results – Digit recognition

[Schapire, 1989]



- Boosting often, **but not always**
  - Robust to overfitting
  - Test set error decreases even after training error is zero

# Generalization Error Bounds

[Freund & Schapire'95]

$$error_{true}(H) \leq error_{train}(H) + \tilde{O} \left( \sqrt{\frac{Td}{m}} \right)$$

	bias	variance	
tradeoff	large	small	T small
	small	large	T large

- T – number of boosting rounds
- d – VC dimension of weak learner, measures complexity of classifier
- m – number of training examples

# Generalization Error Bounds

[Freund & Schapire'95]

$$error_{true}(H) \leq error_{train}(H) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right) \quad \text{With high probability}$$

Boosting can overfit if T is large

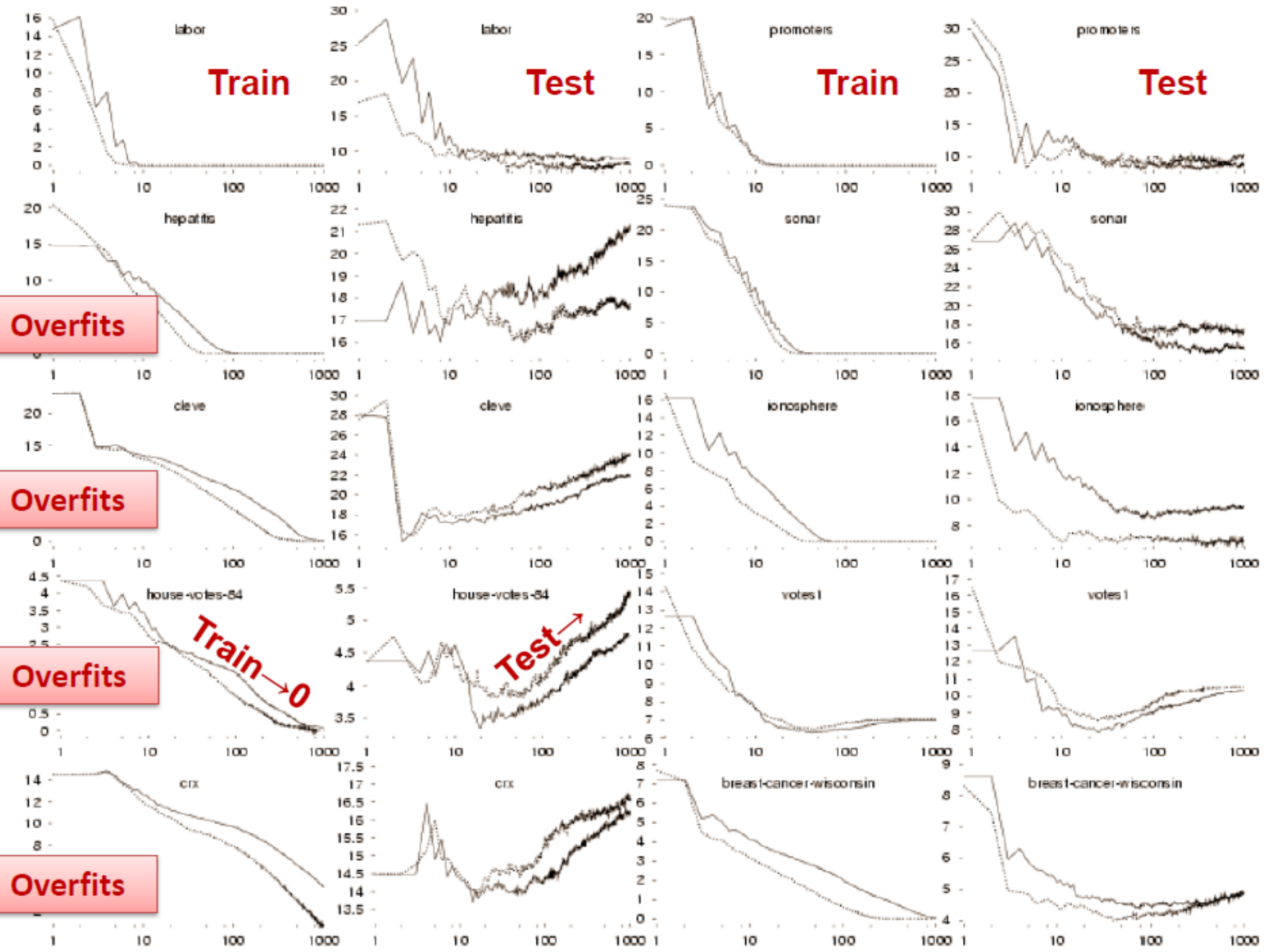
Boosting often,

**Contradicts experimental results**

- Robust to overfitting
- Test set error decreases even after training error is zero

Need better analysis tools – margin based bounds

AdaBoost and AdaBoost.MH on Train (left) and Test (right) data from Irvine repository. [Schapire and Singer, ML 1999]



# Boosting and Logistic Regression

Logistic regression assumes:

$$P(Y = 1|X) = \frac{1}{1 + \exp(f(x))}$$

$$f(x) = w_0 + \sum_j w_j x_j$$

And tries to maximize data likelihood:

$$P(\mathcal{D}|f) \stackrel{\text{iid}}{=} \prod_{i=1}^m \frac{1}{1 + \exp(-y_i f(x_i))}$$

Equivalent to minimizing log loss

$$-\log P(\mathcal{D}|f) = \sum_{i=1}^m \ln(1 + \exp(-y_i f(x_i)))$$



# Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss

$$\sum_{i=1}^m \ln(1 + \exp(-y_i f(x_i)))$$

$$f(x) = w_0 + \sum_j w_j x_j$$

Boosting minimizes similar loss function!!

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

$$f(x) = \sum_t \alpha_t h_t(x)$$

Weighted average of weak learners



**Both smooth approximations of 0/1 loss!**

# Boosting and Logistic Regression

## Logistic regression:

- Minimize log loss

$$\sum_{i=1}^m \ln(1 + \exp(-y_i f(x_i)))$$

- Define

$$f(x) = \sum_j w_j x_j$$

where  $x_j$  predefined features

(linear classifier)

- Jointly optimize over all weights  $w_0, w_1, w_2, \dots$

## Boosting:

- Minimize exp loss

$$\sum_{i=1}^m \exp(-y_i f(x_i))$$

- Define

$$f(x) = \sum_t \alpha_t h_t(x)$$

where  $h_t(x)$  defined dynamically to fit data

(not a linear classifier)

- Weights  $\alpha_t$  learned per iteration  $t$  incrementally

# Hard & Soft Decision

Weighted average of weak learners  $f(x) = \sum_t \alpha_t h_t(x)$

Hard Decision/Predicted label:  $H(x) = \text{sign}(f(x))$

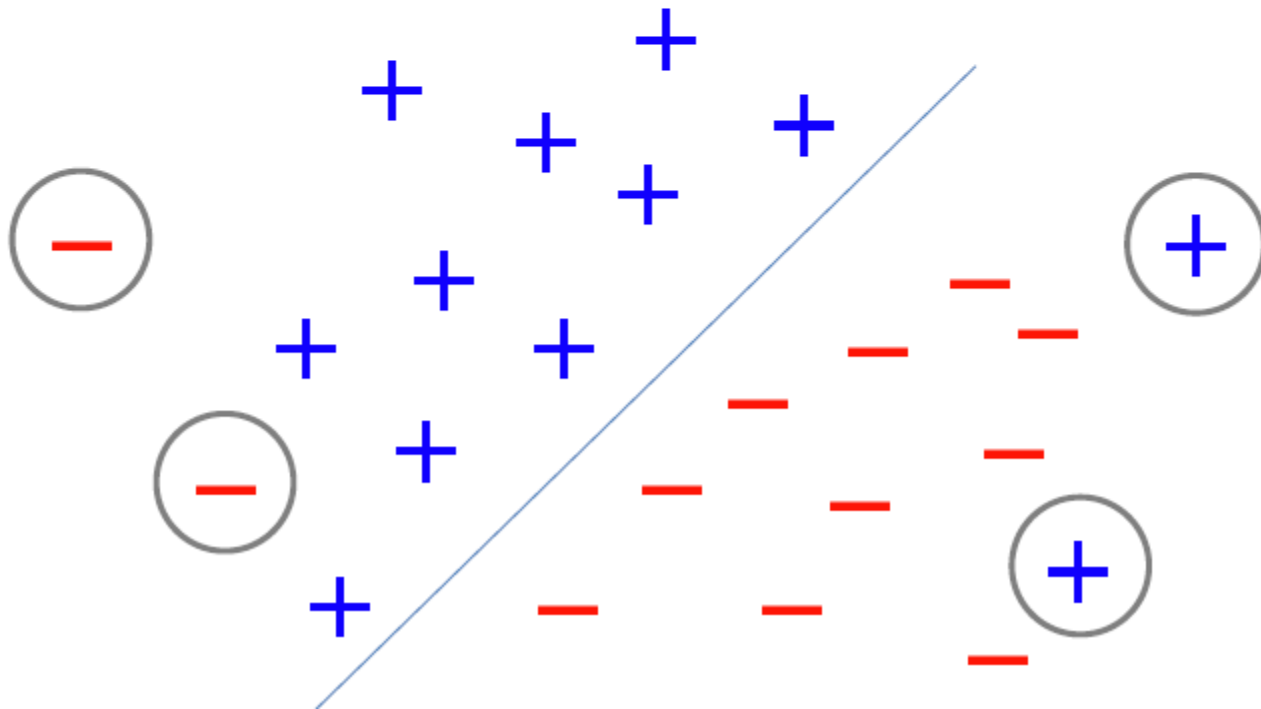
Soft Decision:  
(based on analogy with  
logistic regression)

$$P(Y = 1|X) = \frac{1}{1 + \exp(-f(x))}$$

# Effect of Outliers

**Good 😊** : Can identify outliers since focuses on examples that are hard to categorize

**Bad 😞** : Too many outliers can degrade classification performance dramatically increase time to convergence



# Bagging

[Breiman, 1996]

Related approach to combining classifiers:

1. Run independent weak learners on bootstrap replicates (sample with replacement) of the training set
2. Average/vote over weak hypotheses

## Bagging

vs.

## Boosting

Resamples data points

Reweights data points (modifies their distribution)

Weight of each classifier is the same

Weight is dependent on classifier's accuracy

Only variance reduction

Both bias and variance reduced – learning rule becomes more complex with iterations

# Boosting Summary

- Combine weak classifiers to obtain very strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can eventually provide zero training error
- AdaBoost algorithm
- Boosting v. Logistic Regression
  - Similar loss functions
  - Single optimization (LR) v. Incrementally improving classification (B)
- Most popular application of Boosting:
  - Boosted decision stumps!
  - Very simple to implement, very effective classifier

# Boosting Demo