Boosting

Machine Learning 10-601B

Seyoung Kim

Many of these slides are derived fromTom Mitchell, Ziv-Bar Joseph. Thanks!

Simple Learners

- Simple (a.k.a. weak) learners are good
  - e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  - don't usually overfit
- Simple (a.k.a. weak) learners are bad
  - can't solve hard learning problems
- Can we make weak learners always good???
  - No!!!
  - But often yes…

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 to learn about different parts of the input space?
  - weight the votes of different classifiers?

Boosting [Schapire, 1989]

- Idea: given a weak learner, run it multiple times on (reweighted) training data, then let the learned classifiers vote

- On each iteration *t*:
  - weight each training example by how incorrectly it was classified
  - Learn a hypothesis – h_t
  - A strength for this hypothesis – α_t

- Final classifier:

  - A linear combination of the votes of the different classifiers weighted by their strength

$$H(X) = \text{sign}(\Sigma \alpha_t h_t(X))$$

- **Practically useful**
- **Theoretically interesting**

Learning from weighted data

- **Sometimes not all data points are equal**
  - Some data points are more equal than others

- **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., MLE for Naïve Bayes, redefine *Count(Y=y)* to be weighted count

Learning From Weighted Data

- Consider a weighted dataset
  - D(i) – weight of i th training example (xi,yi)
  - Interpretatons:
    - *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

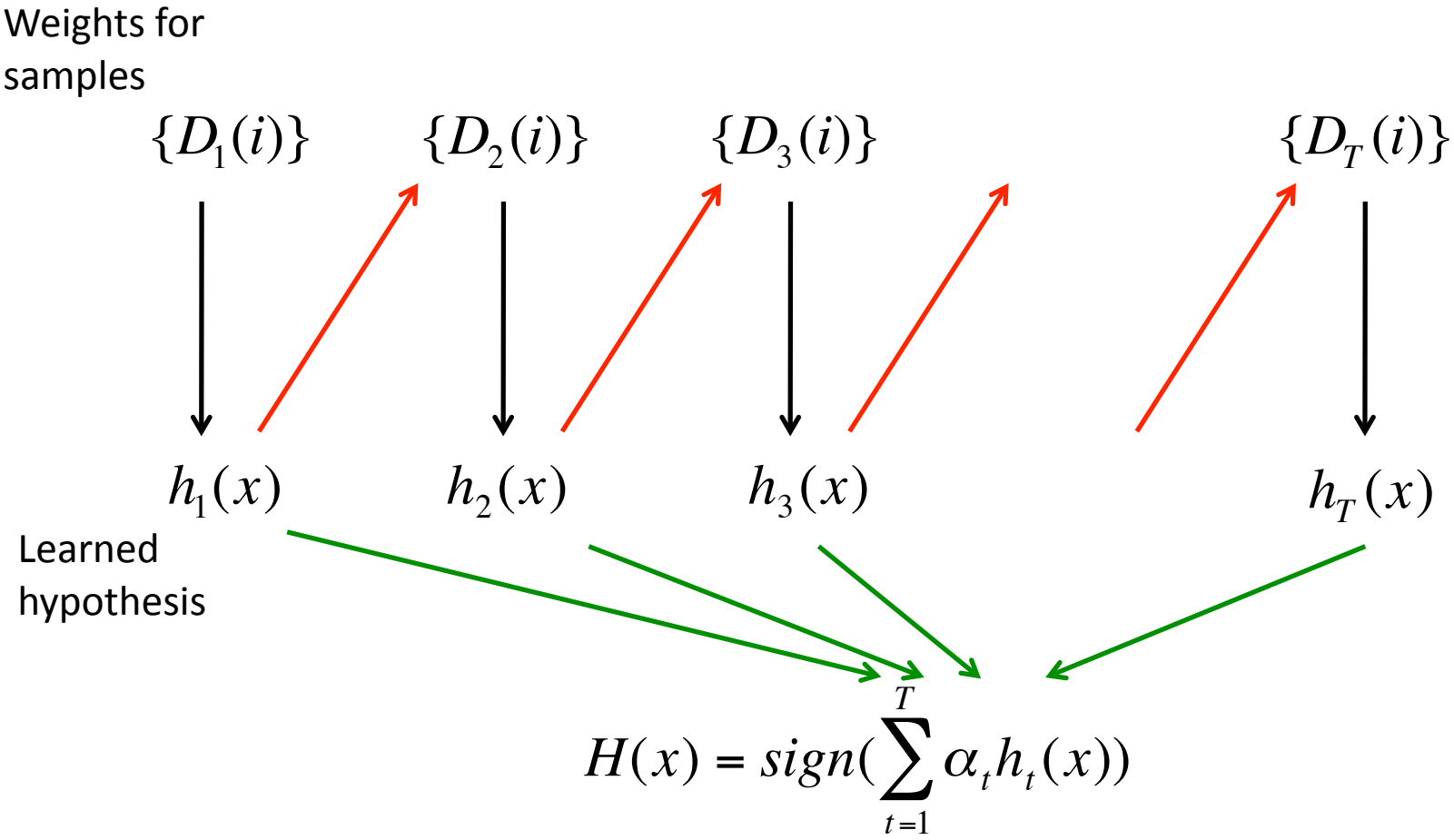
Unweighted data

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

Weights D(i)

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

Boosting


Weights for samples
$\{D_1(i)\}$
$\{D_2(i)\}$
$\{D_3(i)\}$
$\{D_T(i)\}$
$h_1(x)$
$h_2(x)$
$h_3(x)$
$h_T(x)$
Learned hypothesis
$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

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

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

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

- Train weak learner using distribution D_t. Naïve Bayes, decision stump
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

Why?

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

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 samples should sum to 1

$\sum_i D_{t+1}(i)=1$

Output the final classifier:

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

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

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

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

- Train weak learner using distribution D_t. Naïve Bayes, decision stump
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

Why?

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

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

Increase weight
if wrong on sample i

Output the final classifier:

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

What αt to choose for hypothesis h_t?

[Schapire, 1989]

- Weight update rule:

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

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$ [Freund & Schapire '97]

$$\epsilon_t = \sum_{i=1}^{m} D_t(i)\delta(h_t(x_i) \neq y_i)$$

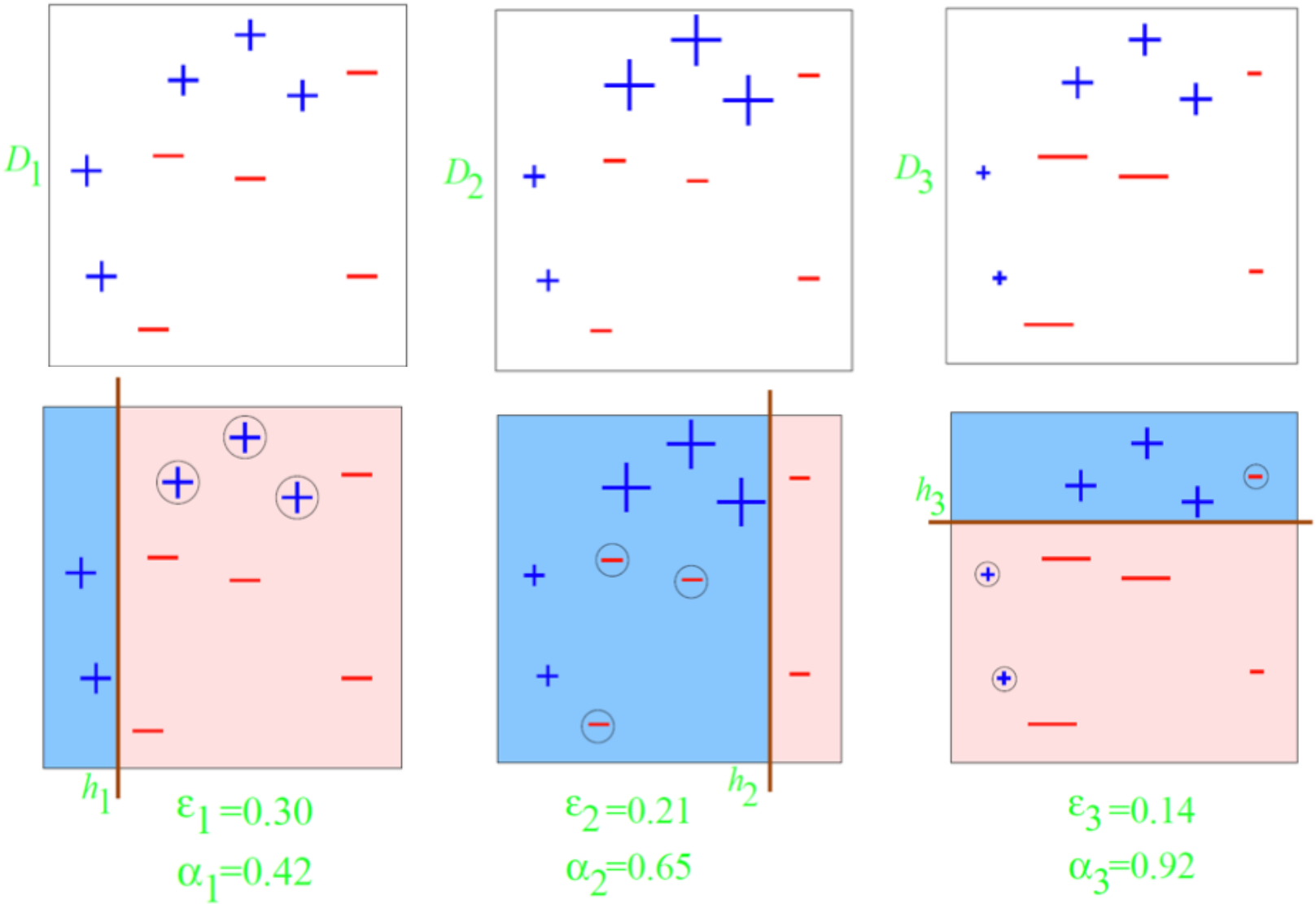
Weighted training error

ε_t = 0 if h_t perfectly classifies all weighted data pts — $\alpha_t = \infty$

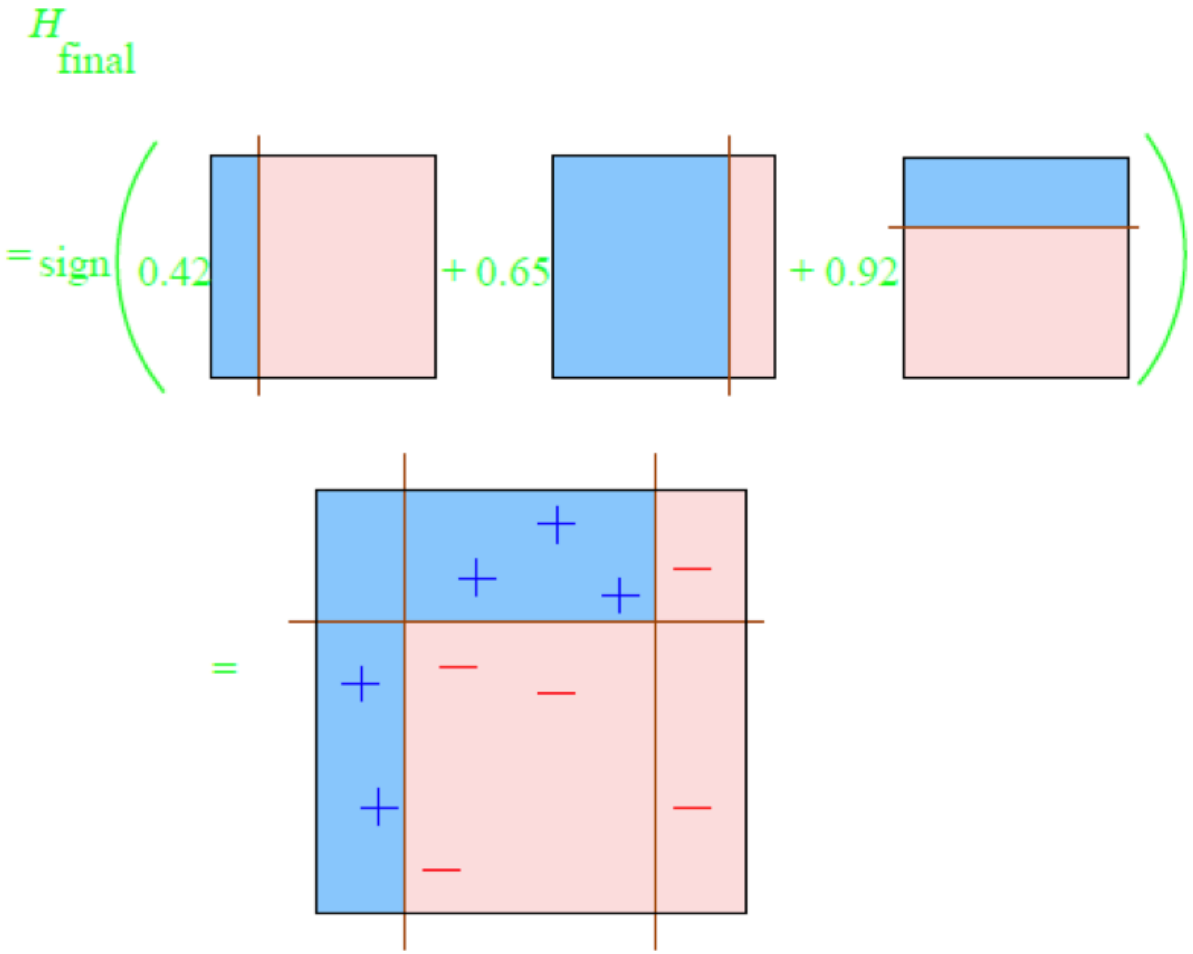
ε_t = 1 if h_t perfectly wrong => $-h_t$ perfectly right — $\alpha_t = -\infty$

ε_t = 0.5 — α_t = 0

Boosting Example (Decision Stump)



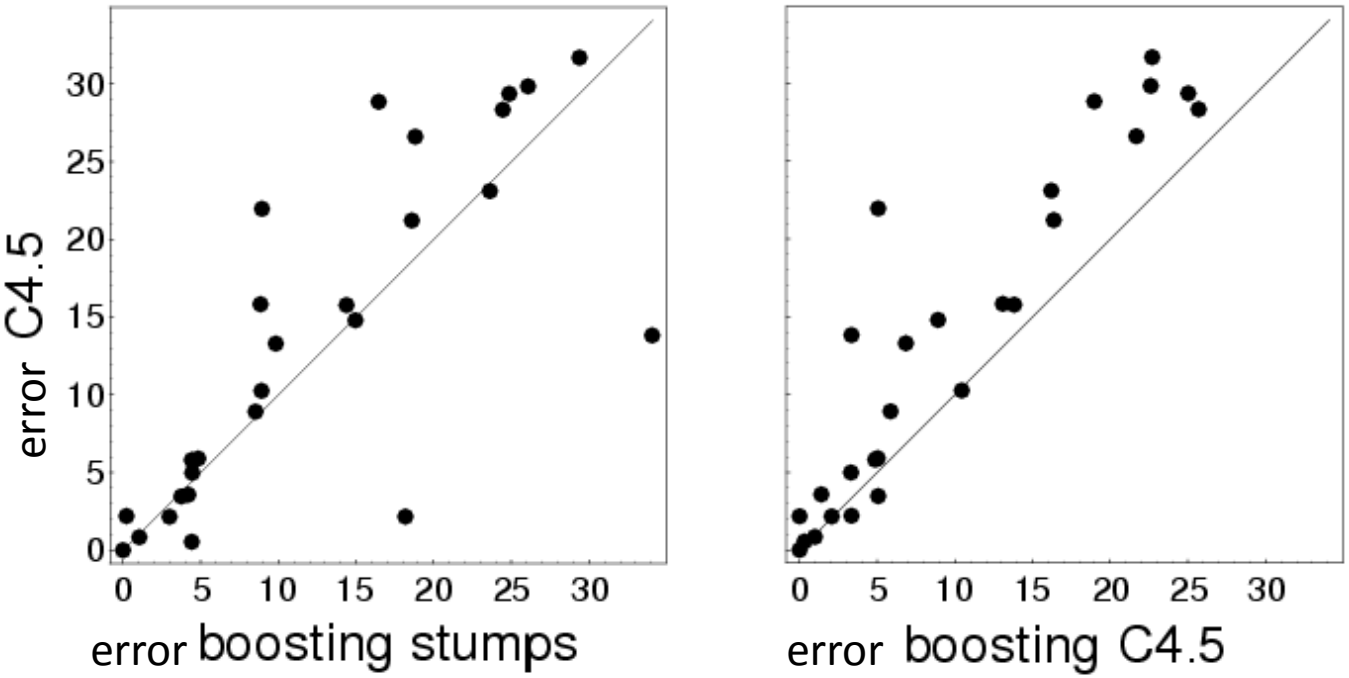
Boosting Example



Boosting: Experimental Results

[Freund & Schapire, 1996]

Comparison of C4.5, Boosting C4.5, Boosting decision stumps (depth 1 trees), 27 benchmark datasets



Analyzing Training Error

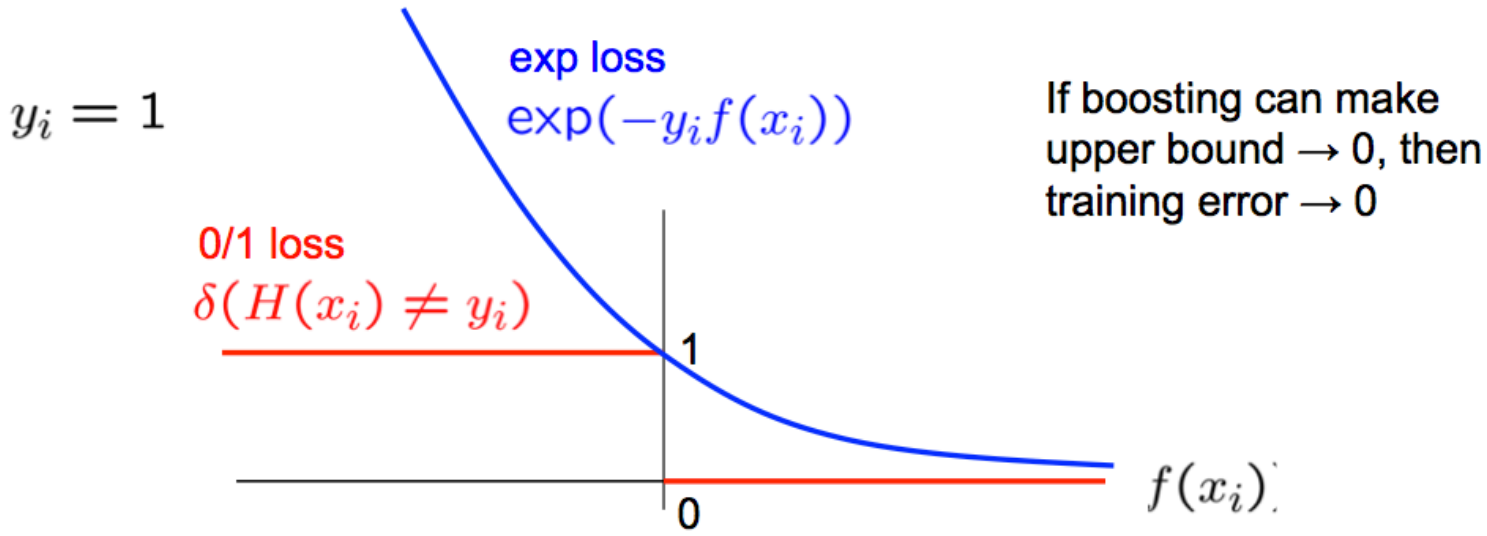
- Training error of final classifier is bounded by:

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

Convex upper bound

where

$$f(x)=\sum \alpha_t h_t(x);\ H(x)=sign(f(x))$$



If boosting can make upper bound → 0, then training error → 0

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 $f(x) = \sum_t \alpha_t h_t(x); H(x) = sign(f(x))$

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

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 $f(x) = \sum_t \alpha_t h_t(x)$; $H(x) = sign(f(x))$

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

Proof: **Using Weight Update Rule**

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

$$D_1(i) = 1/m.$$

$$D_2(i) = \frac{1}{m}\frac{e^{-\alpha_1 y_i h_1(x_i)}}{Z_1}$$

$$D_3(i) = \frac{1}{m}\frac{e^{-\alpha_1 y_i h_1(x_i)} e^{-\alpha_2 y_i h_2(x_i)}}{Z_1 Z_2}$$

$$\vdots$$

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

Wts of all pts add to 1

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

Analyzing Training Error

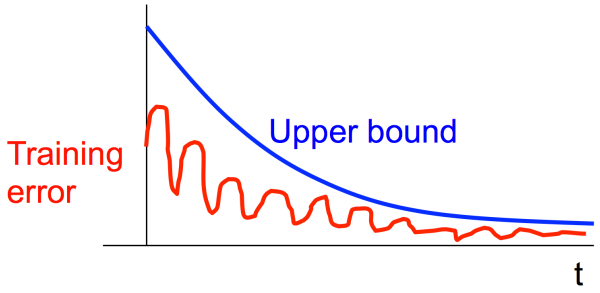
- Training error of final classifier is bounded by:

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

where

$$f(x) = \sum_t \alpha_t h_t(x);\; H(x) = sign(f(x))$$

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



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

$$f(x) = \sum_t \alpha_t h_t(x); \; H(x) = sign(f(x))$$

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 for hypothesis h_t?

[Schapire, 1989]

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:

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

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

What α_t to choose for hypothesis h_t?

[Schapire, 1989]

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:

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

Strong, weak classifiers

- Training error of the final classifier is bounded by

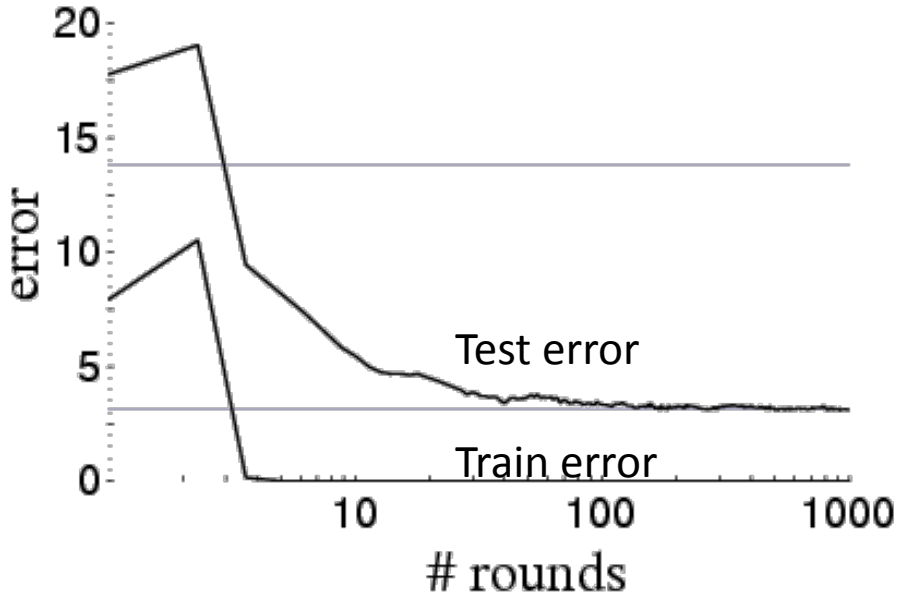
Using $1\text{-}x \leq e^{-x}$

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

- If each classifier is (at least slightly) better than random ($\varepsilon_t < 0.5$), AdaBoost will achieve zero training error exponentially fast (in number of rounds T) !!

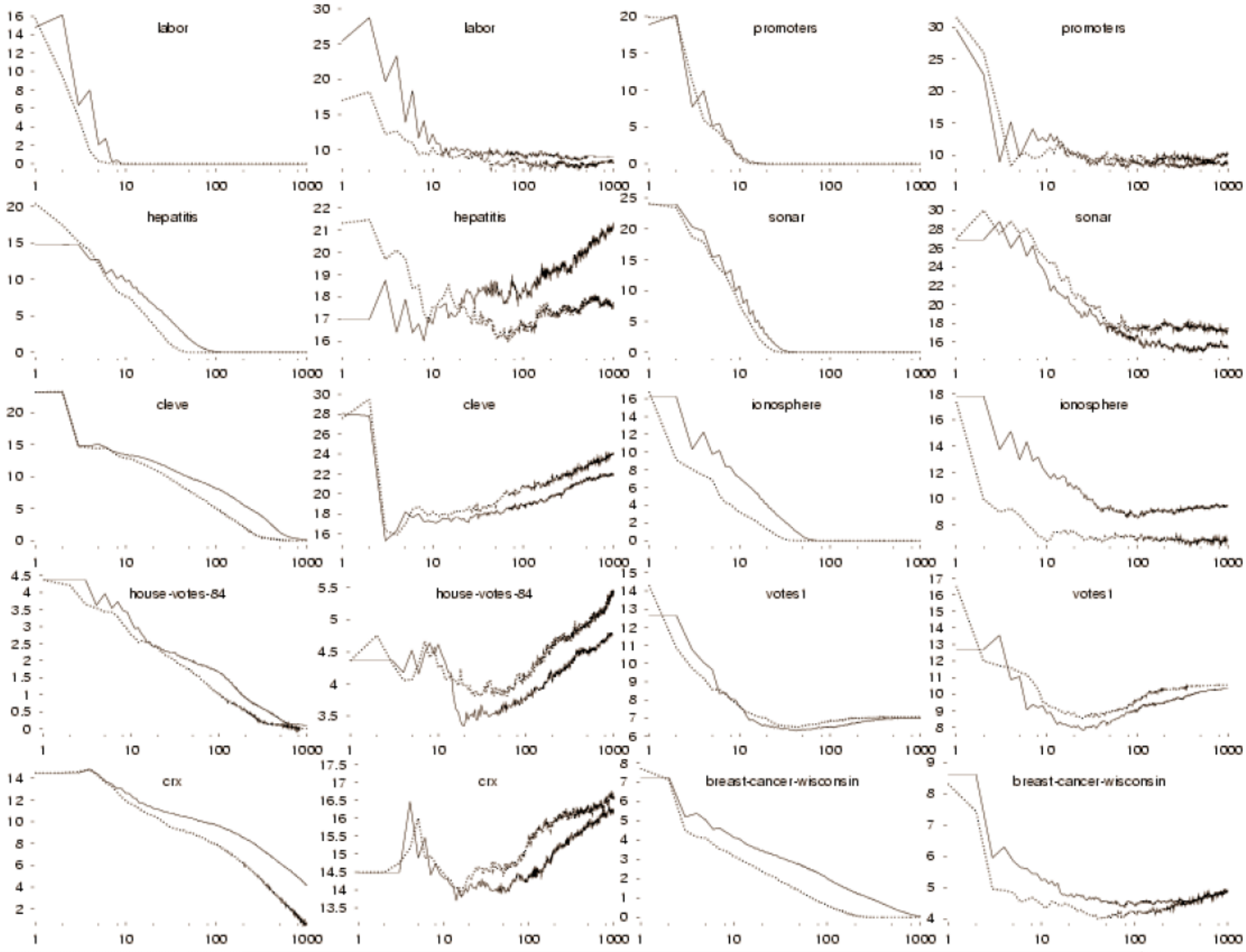
Boosting results – Digit recognition

[Schapire, 1989]



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

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:

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

And tries to maximize data likelihood:

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

Equivalent to minimizing log loss

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

Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss

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

Boosting minimizes similar loss function!!

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

Both smooth approximations of 0/1 loss!

Logistic regression and Boosting

Logistic regression:

- Minimize loss fn

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

- Define

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

where x_j predefined

Boosting:

- Minimize loss fn

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

- Define

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

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

(not a linear classifier)

- Weights α_t learned incrementally over t

Bagging

- 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	**Boosting**
Resamples data points	Reweights data points (modifies their distribution)
Weight of each classifier is the same	Weight is dependent on classifier's accuracy

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

What you need to know about Boosting

- 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 vs Logistic Regression
  - Similar loss functions
  - Single optimization (LR) vs Incrementally improving classification (B)
- Most popular application of Boosting:
  - Boosted decision stumps!
  - Very simple to implement, very effective classifier