Boosting

Can we make dumb learners smart?

Aarti Singh

Machine Learning 10-601
Nov 29, 2011

Slides Courtesy: Carlos Guestrin, Freund & Schapire

ML MACHINE LEARNING DEPARTMENT

Carnegie Mellon.
School of Computer Science

# Why boost weak learners?

Goal: Automatically categorize type of call requested
(Collect, Calling card, Person-to-person, etc.)

- yes I'd like to place a collect call long
  distance please (Collect)
- operator I need to make a call but I need to
  bill it to my office (ThirdNumber)
- yes I'd like to place a call on my master card
  please (CallingCard)

- **Easy to find "rules of thumb" that are "often" correct.**
  E.g. If 'card' occurs in utterance, then predict 'calling card'

- **Hard to find single highly accurate prediction rule.**

Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners** e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
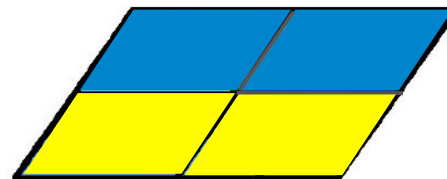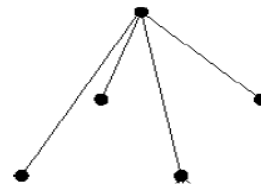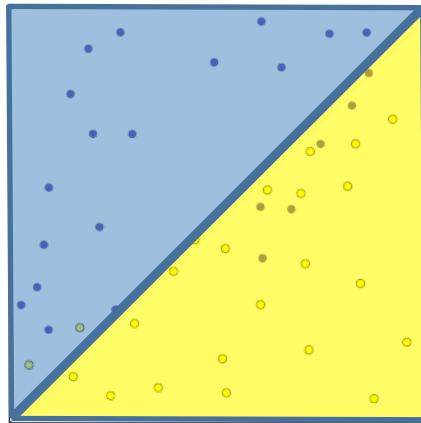
Are good ☺ - Low variance, don't usually overfit

**Are bad** ☹ - High bias, 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!

$h_1(X)$

$h_2(X)$
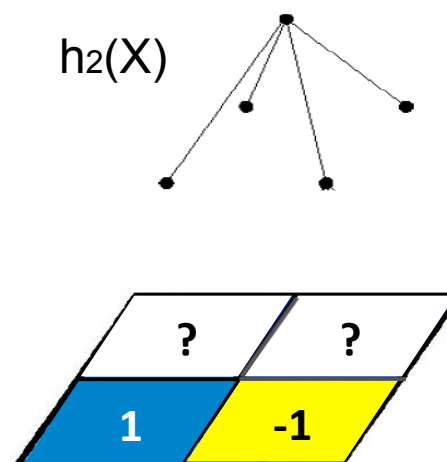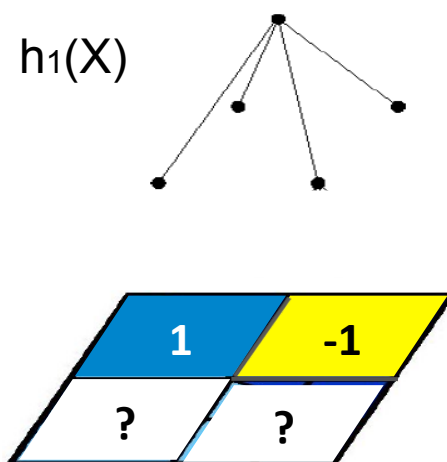
$H: X \rightarrow Y (-1,1)$

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

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

weights

1	-1
?	?

?	?
1	-1

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? α_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 – α_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: $(x_1, y_1), \ldots, (x_m, y_m)$ 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 $\alpha_t \in \mathbb{R}$. **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: $(x_1, y_1), \ldots, (x_m, y_m)$ 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 $\alpha_t \in \mathbb{R}$.   **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: $(x_1, y_1), \ldots, (x_m, y_m)$ 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 $\alpha_t \in \mathbb{R}$. **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 α_t to choose for hypothesis h_t?

Weight Update Rule:

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

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

[Freund & Schapire'95]

Weighted training error

$$\epsilon_t = P_{i\sim D_t(i)}[h_t(\mathbf{x}^i) \neq y^i] = \sum_{i=1}^{m} D_t(i)\underbrace{\delta(h_t(x_i) \neq y_i)}$$

Does h_t get i^{th} point wrong

ϵ_t = 0 if h_t perfectly classifies all weighted data pts α_t = ∞

ϵ_t = 1 if h_t perfectly wrong => -h_t perfectly right α_t = -∞

ϵ_t = 0.5 α_t = 0

Boosting Example (Decision Stumps)

D_1

D_2

D_3

h_1

h_2

h_3
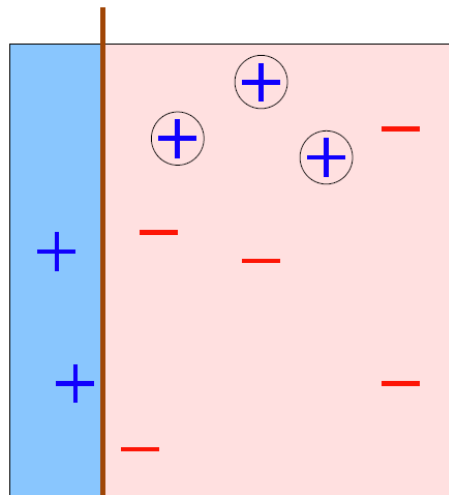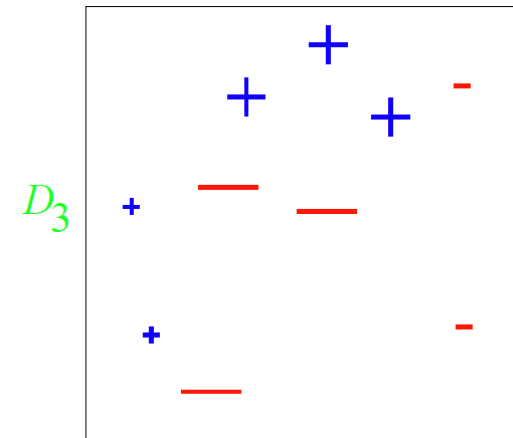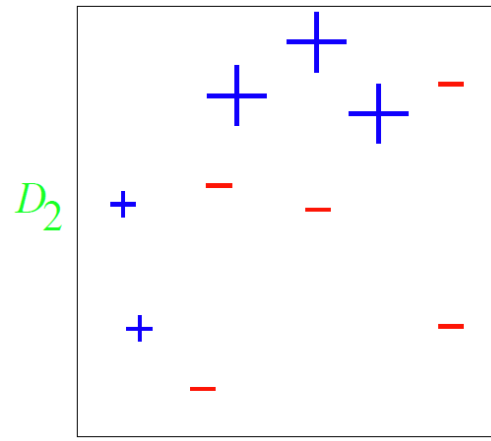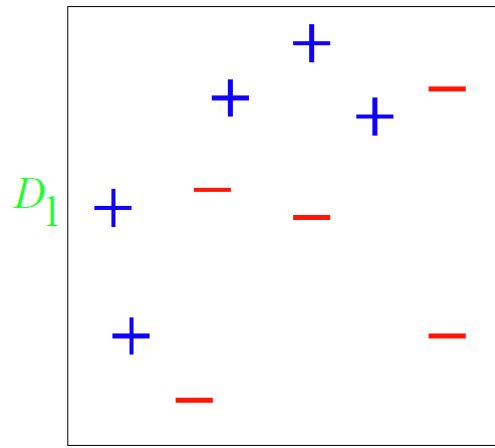
$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$
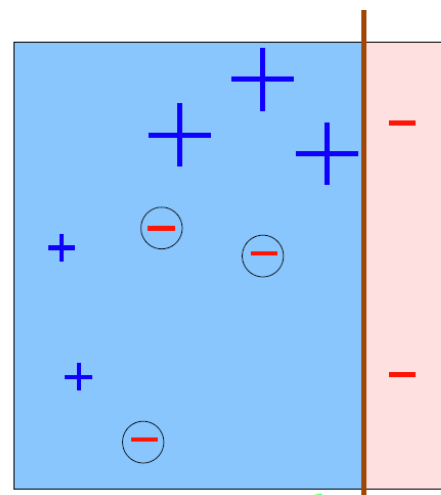
$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$
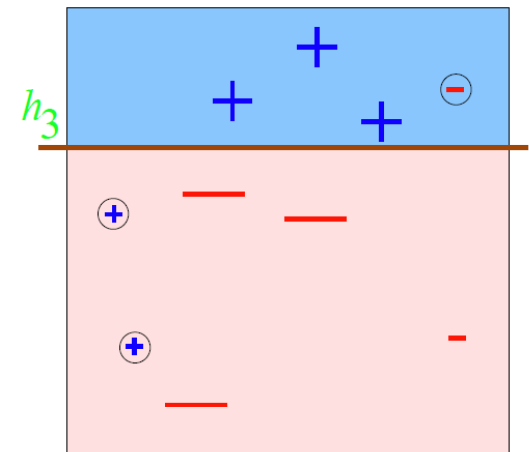
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

12

Boosting Example (Decision Stumps)

$$H_{\text{final}}$$

$$= \text{sign}\left(0.42 \quad + 0.65 \quad + 0.92 \right)$$
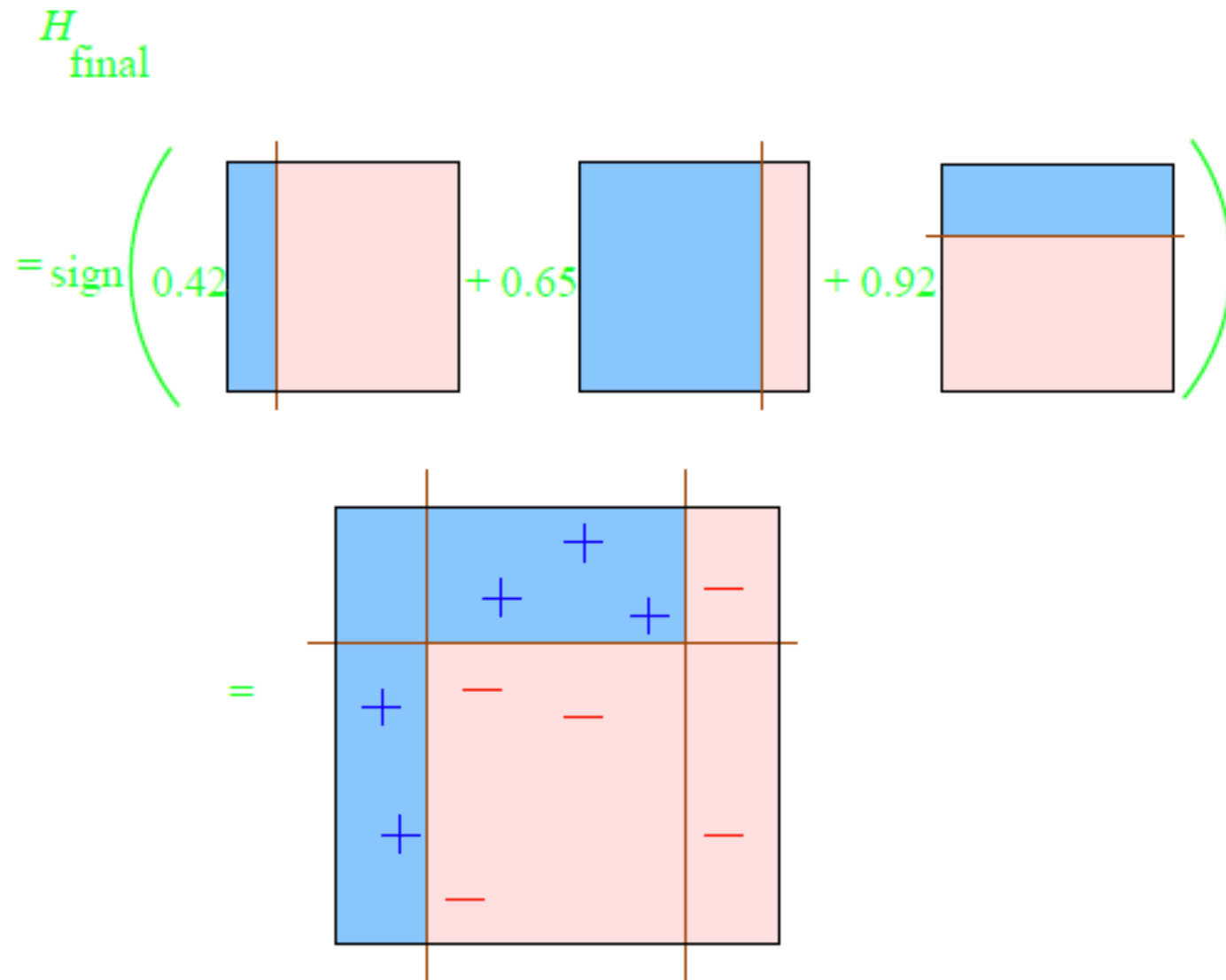
Analyzing training error

Analysis reveals:

- What α_t to choose for hypothesis h_t? $\boxed{\alpha_t = \frac{1}{2}\ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)}$

ε_t - weighted training error

- If each weak learner h_t is slightly better than random guessing ($\varepsilon_t < 0.5$), 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:

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

Convex upper bound

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

$y_i = 1$

exp loss $\exp(-y_i f(x_i))$

If boosting can make upper bound $\to 0$, then training error $\to 0$

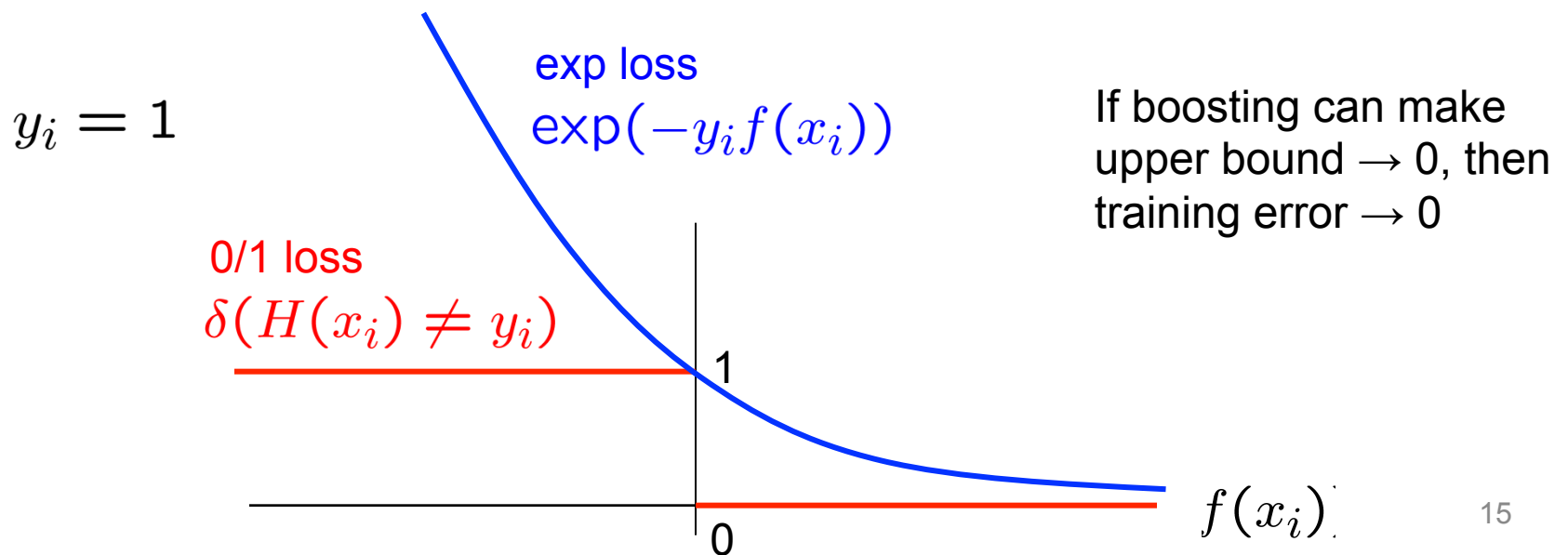0/1 loss $\delta(H(x_i) \neq y_i)$

1

0

$f(x_i)$

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

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

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

16

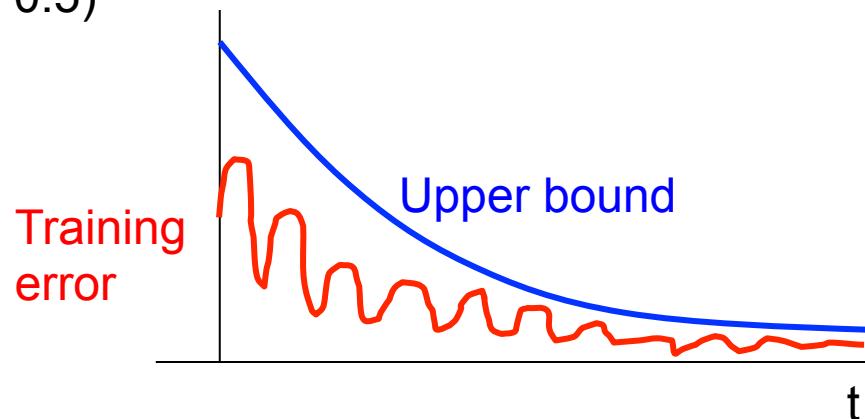# Analyzing training error

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 $\quad f(x)=\sum_{t}\alpha_t h_t(x); H(x)=sign(f(x))$

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

Training error

Upper bound

t

What α_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 $\quad f(x) = \sum_{t}\alpha_t h_t(x); \, H(x) = sign(f(x))$

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?

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]:

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

Proof:

$$Z_t = \sum_{i:y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} + \sum_{i:y_i = h_t(x_i)} D_t(i) e^{-\alpha_t}$$

$$= \epsilon_t e^{\alpha_t} + (1 - \epsilon_t) e^{-\alpha_t}$$

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

What α_t to choose for hypothesis h_t?

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]:

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

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:

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

Using 1-x ≤ e^{-x}

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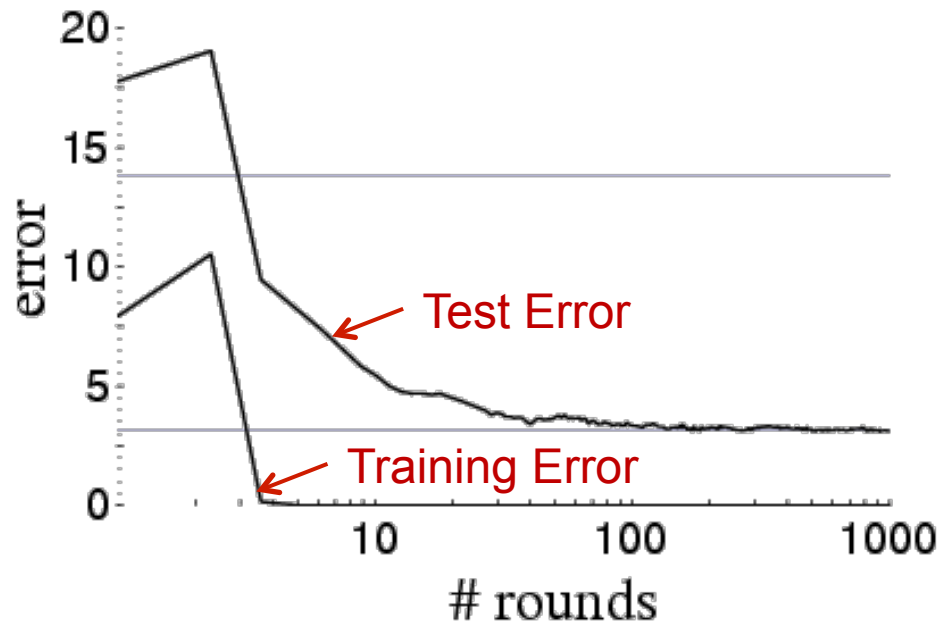
grows as ε$_t$ moves
away from 1/2

If each classifier is (at least slightly) better than random   ε$_t$ < 0.5

AdaBoost will achieve zero _training error_ exponentially fast (in number of rounds T) !!

What about test error?

21

Boosting results – Digit recognition

[Schapire, 1989]

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

Margin Based Bounds

[Schapire, Freund, Bartlett, Lee'98]

$$error_{true}(H) \leq \hat{\Pr}\left[\mathrm{margin}_f(x,y) \leq \theta\right] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$
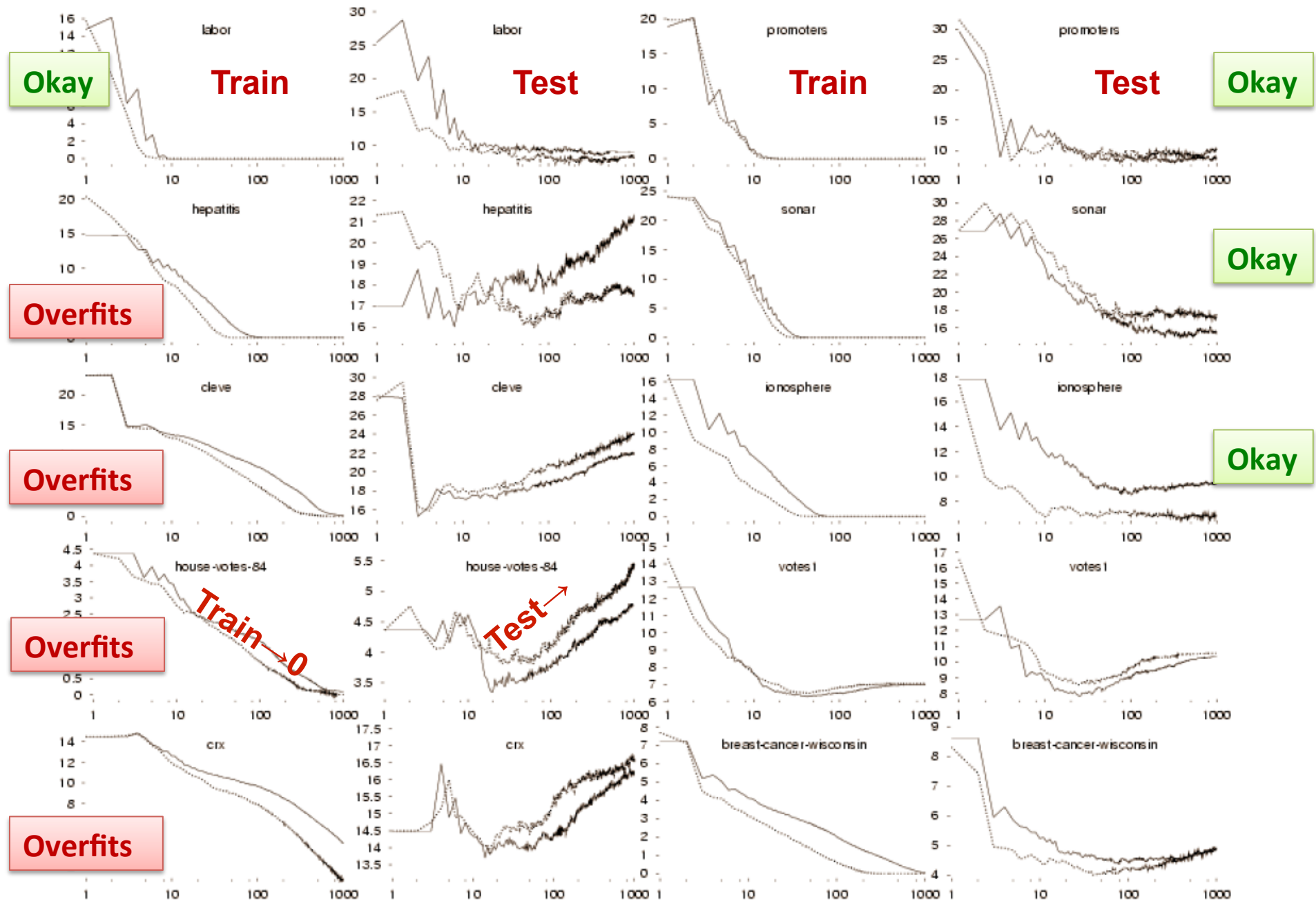
With high probability

Boosting increases the margin very aggressively since it concentrates on the hardest examples.

If margin is large, more weak learners agree and hence more rounds does not necessarily imply that final classifier is getting more complex.

Bound is independent of number of rounds T!

Boosting can still overfit if margin is too small or weak learners are too complex

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

Okay   **Train**   **Test**   **Train**   **Test**   Okay

Overfits — hepatitis / sonar   Okay

Overfits — cleve / ionosphere   Okay

Overfits — house-votes-84 / votes1   *Train→0*   *Test→*

Overfits — crx / breast-cancer-wisconsin

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

log loss

exp loss

$y_i = 1$

0/1 loss

Both smooth approximations of 0/1 loss!

1

$f(x_i)$

0

25

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 $w_0, w_1, w_2...$

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 α_t learned per iteration t incrementally

26

Hard & Soft Decision

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

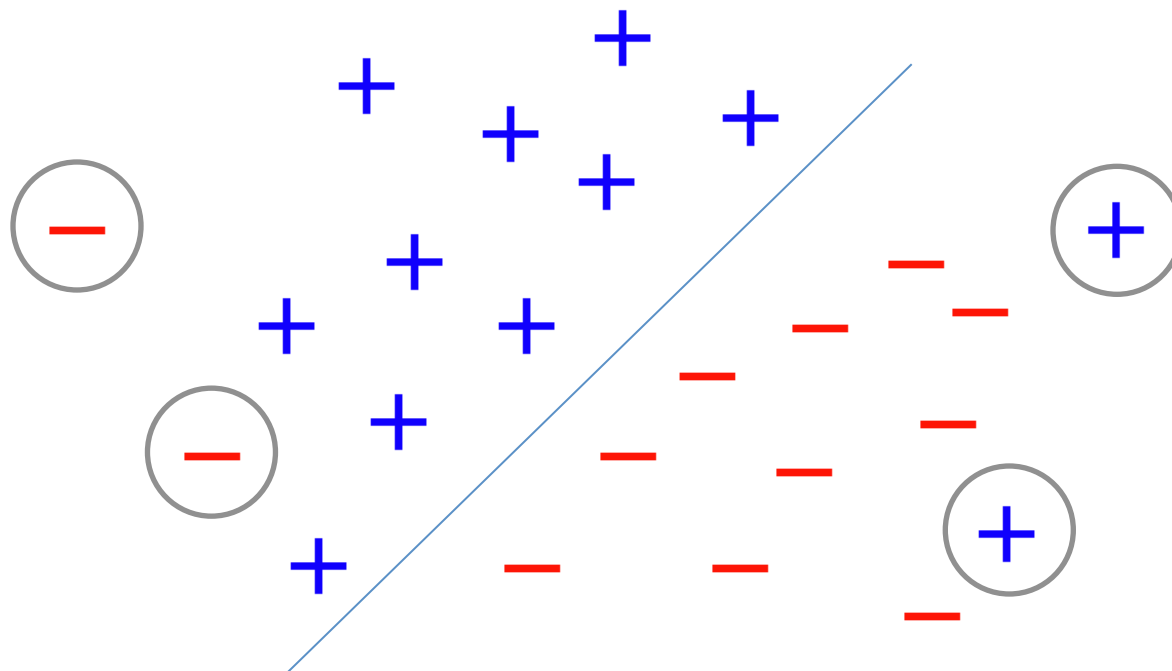Hard Decision/Predicted label: $\quad H(x) = sign(f(x))$

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

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
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