Boosting

Can we make dumb learners smart?

Aarti Singh

Co-instructor: Geoff Gordon

Machine Learning 10-701/15-781
Oct 22, 2014

Slides Courtesy: Carlos Guestrin, Freund & Schapire

ML MACHINE LEARNING DEPARTMENT
Carnegie Mellon.
School of Computer Science

Why boost weak learners?

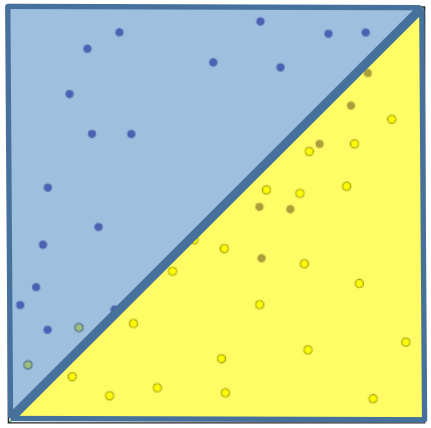
Goal: Automatically categorize type of call requested (Collect, Calling card, Person-to-person, etc.)

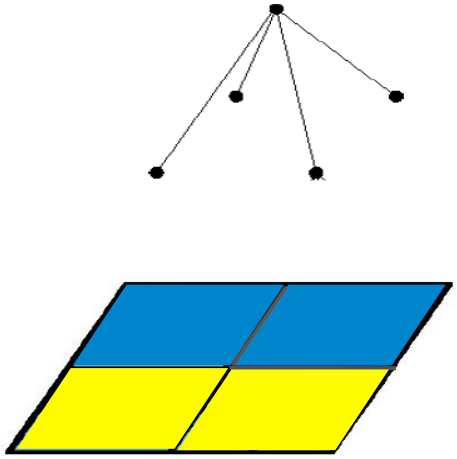
- yes I'd like to place a collect call long distance please (Collect)
- operator I need to make a call but I need to bill it to my office (ThirdNumber)
- yes I'd like to place a call on my master card please (CallingCard)

- **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)





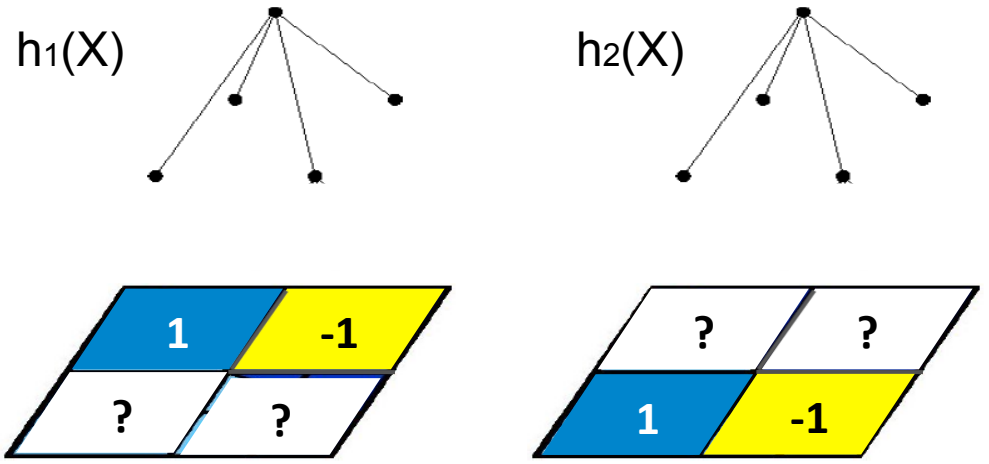
Are good ☺ - Low variance, don't usually overfit

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!



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

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

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

weights

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.
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

Naïve bayes, decision stump

Magic (+ve)

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

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

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

[Freund & Schapire'95]

Weighted training error

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

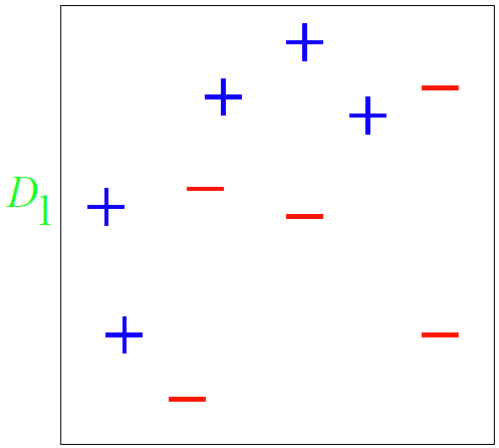
Does h_t get i^{th} point wrong

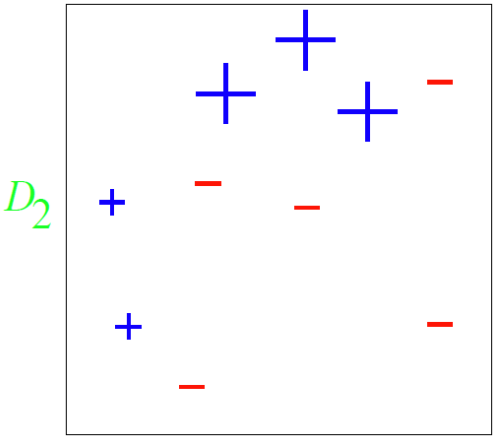
ε_t = 0 if h_t perfectly classifies all weighted data pts — $\alpha_t = \infty$

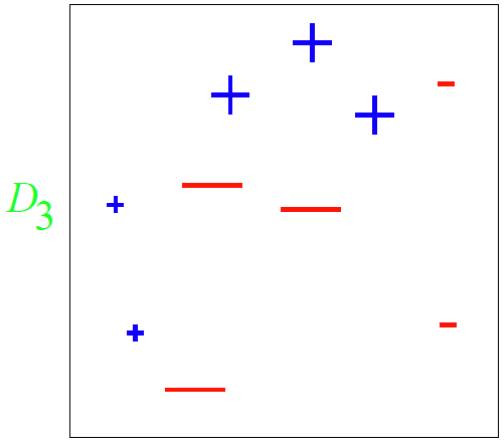
ε_t = 1 if h_t perfectly wrong => $-h_t$ perfectly right — $\alpha_t = -\infty$

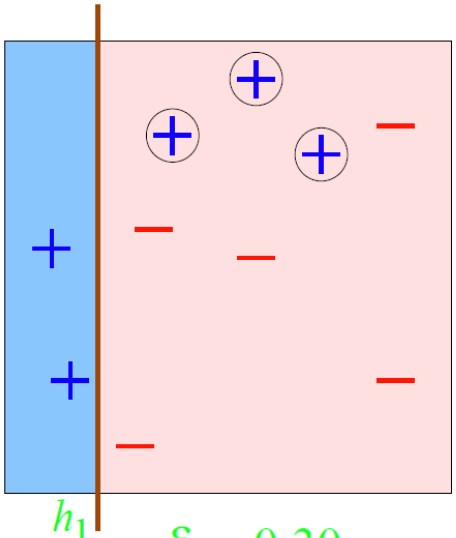
ε_t = 0.5 — $\alpha_t = 0$

Boosting Example (Decision Stumps)



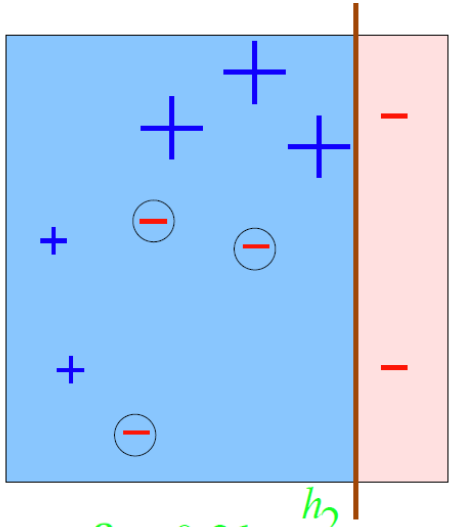






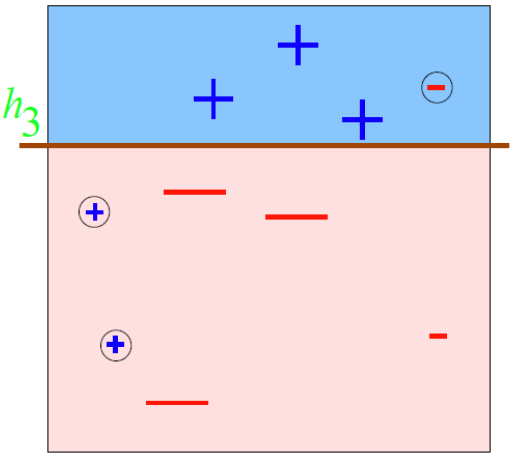
$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$



$\varepsilon_2 = 0.21$

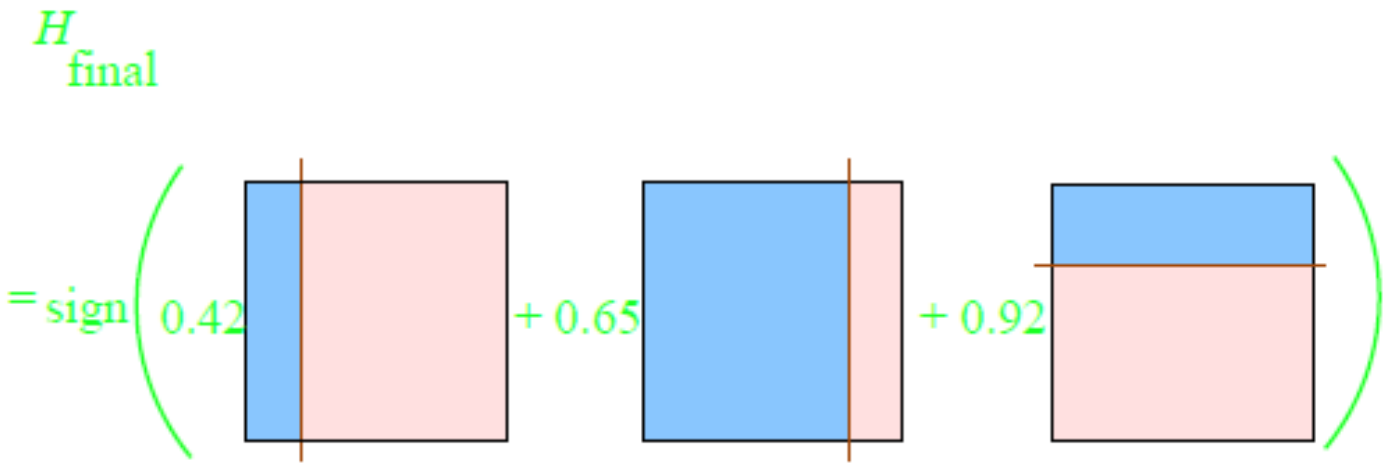
$\alpha_2 = 0.65$

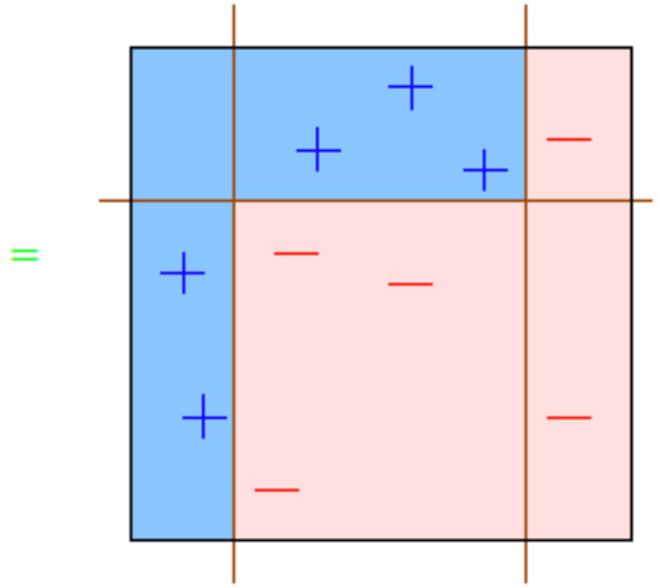


$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

Boosting Example (Decision Stumps)





Analyzing training error

Analysis reveals:

- What α_t to choose for hypothesis h_t?

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

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

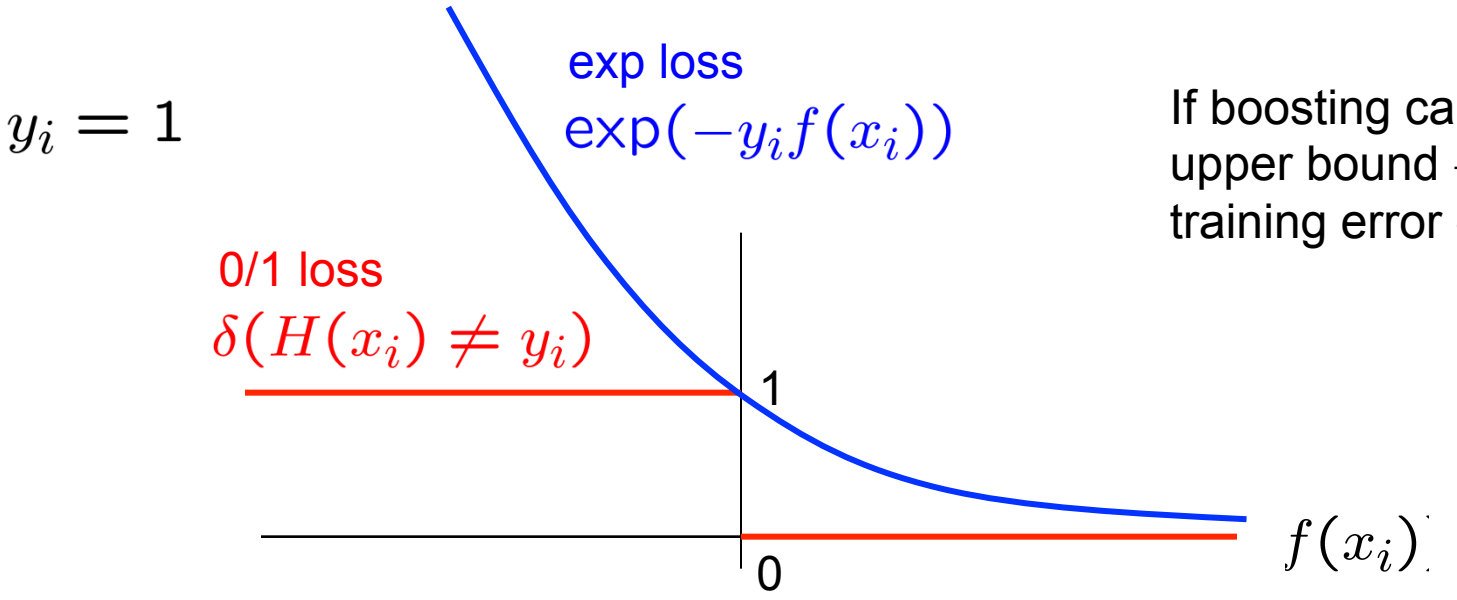
Analyzing training error

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



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

Proof: **Using Weight Update Rule**

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

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

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

…

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

Wts of all pts add to 1

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

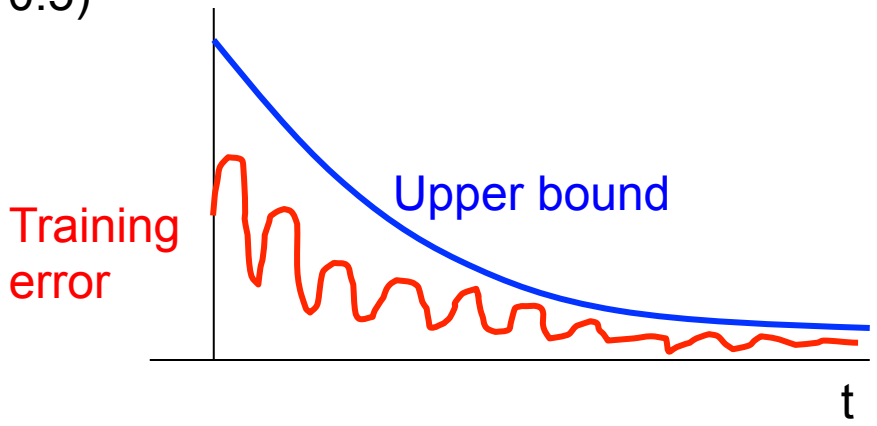
Analyzing training error

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 $\varepsilon_t \sim 0.5$)



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

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]:

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

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?

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]:

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

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:

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

$$\leq \exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)$$

grows as ε_t moves away from 1/2

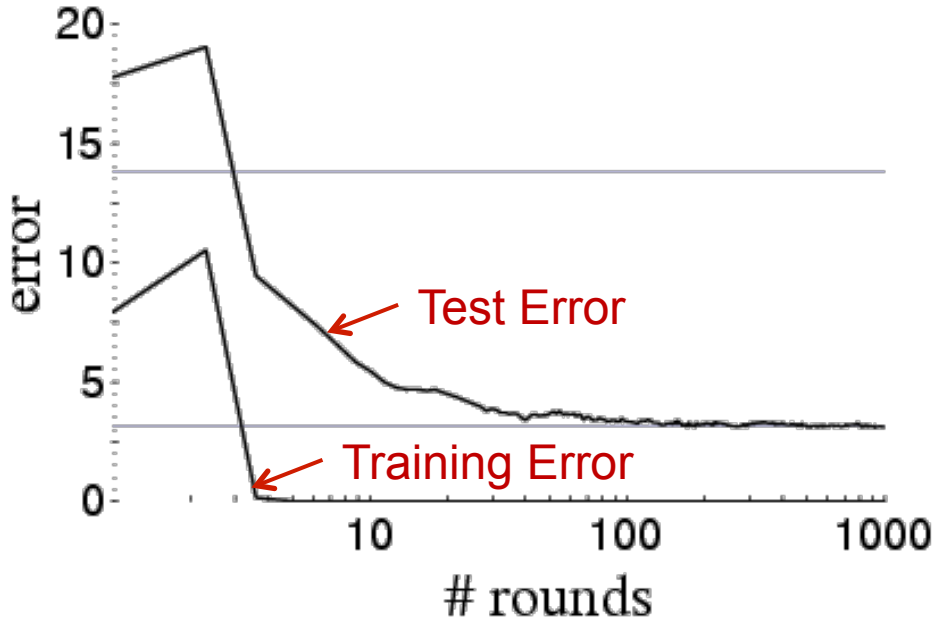
If each classifier is (at least slightly) better than random $\varepsilon_t < 0.5$

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]

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

	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]

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

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

Margin Based Bounds

[Schapire, Freund, Bartlett, Lee'98]

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

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 (can perform arbitrarily close to random guessing) or weak learners are too complex

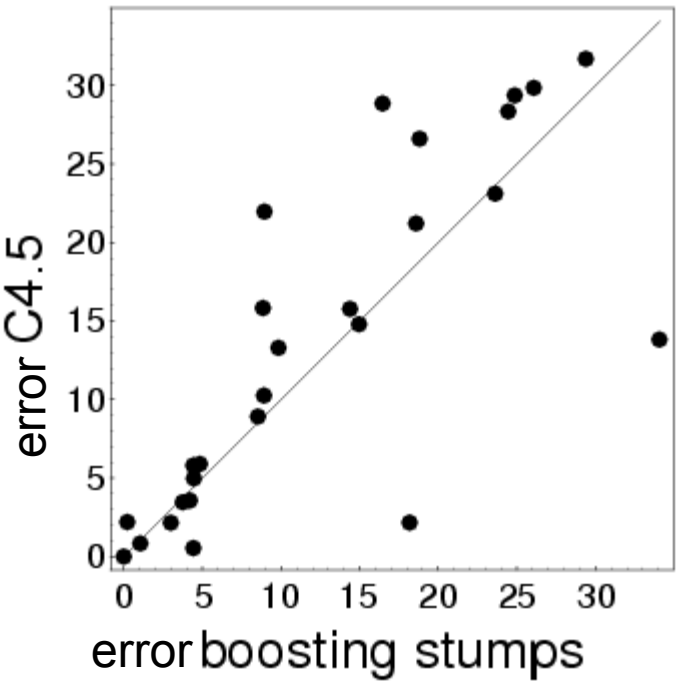
Boosting: Experimental Results

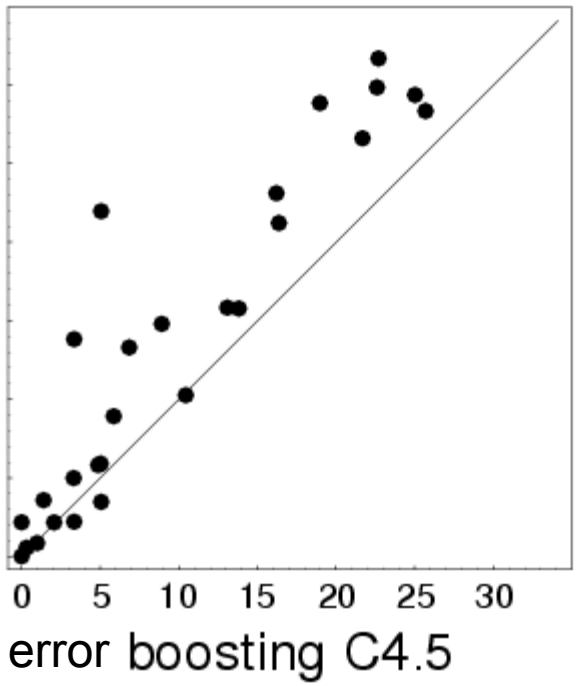
[Freund & Schapire, 1996]

Comparison of C4.5 (decision trees) vs Boosting decision stumps (depth 1 trees)

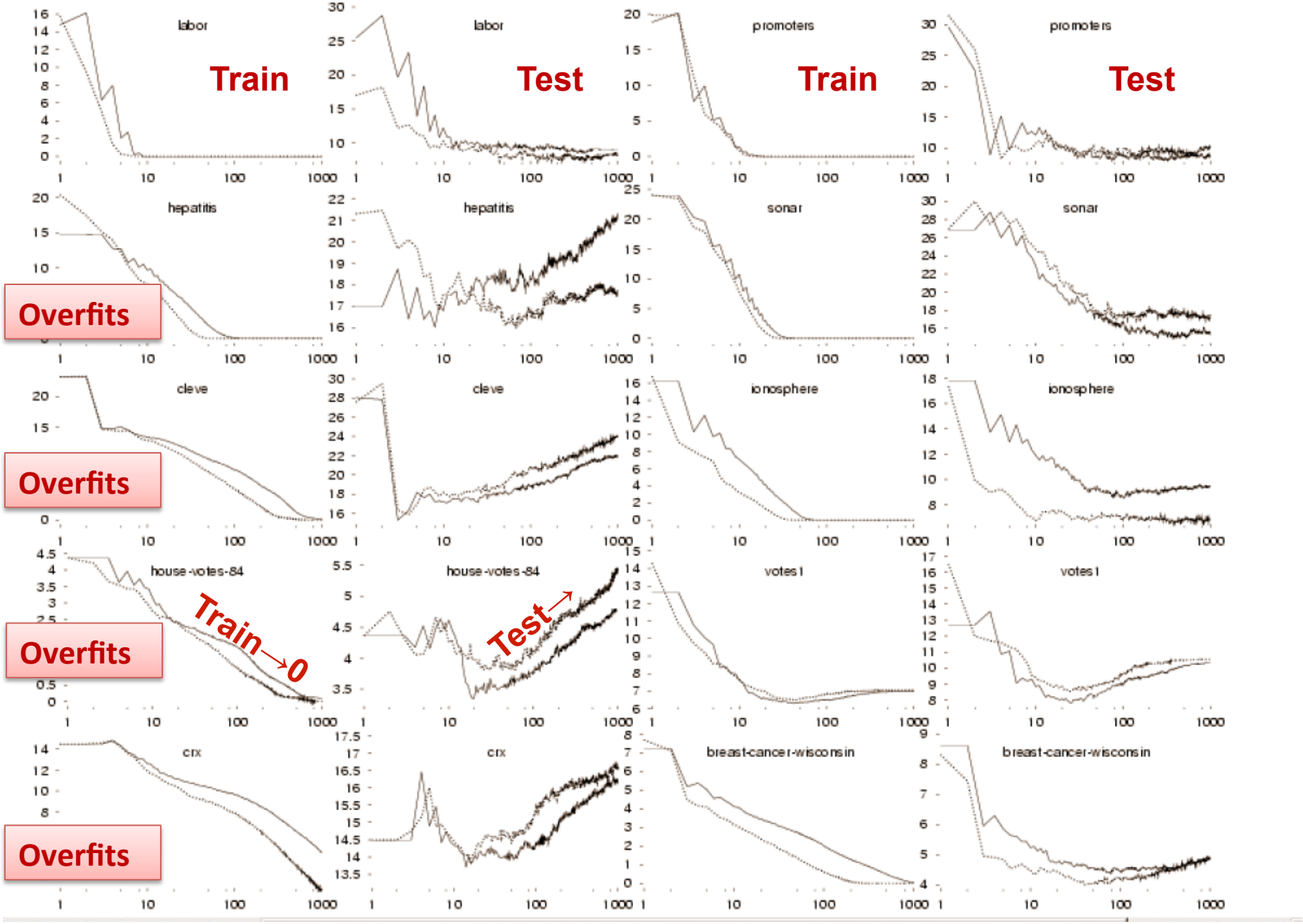
C4.5 vs Boosting C4.5

27 benchmark datasets





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



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
- Most popular application of Boosting:
  - Boosted decision stumps!
  - Very simple to implement, very effective classifier