# CSE 446: Boosting
## Winter 2012

Daniel Weld

Slides adapted from Tom Dietterich, Luke Zettlemoyer, Carlos Guestrin, Nick Kushmerick, Padraig Cunningham

## Ensembles of Classifiers

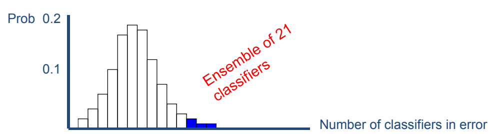
- Traditional approach: Use one classifier
- Can one do better?
- Approaches:
  - Cross-validated committees
  - Bagging
  - Boosting
  - Stacking

© Daniel S. Weld 2

## Ensembles of Classifiers

- Assume
  – Errors are independent (suppose 30% error)
  – Majority vote
- Probability that majority is wrong…

= area under binomial distribution




- If individual area is 0.3
- Area under curve for ≥11 wrong is 0.026
- Order of magnitude improvement!

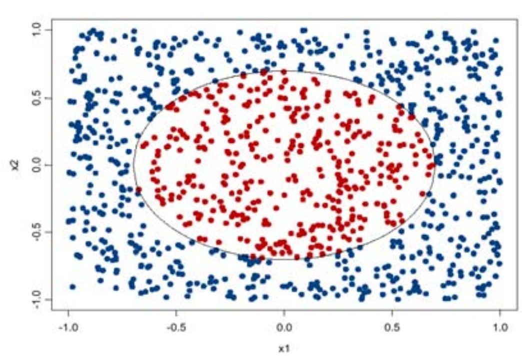
© Daniel S. Weld 3

## BAGGing = Bootstrap AGGregation

(Breiman, 1996)

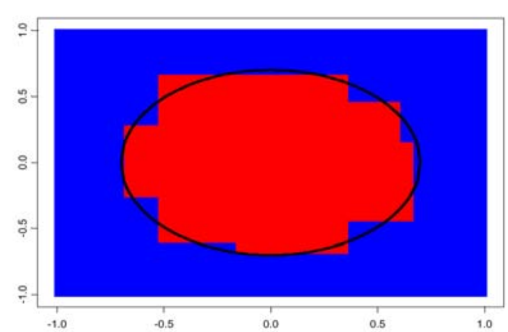
- For m = 1, 2, …, M:
  – $B_m \leftarrow$ randomly select N training instances with replacement
  – $C_m \leftarrow$ learn($B_m$) *[ID3, NB, kNN, neural net, …]*
- Combine the $C_m$ together
  - Uniform voting ($\alpha_m = {}^1/_K$ for all i)

## Bagging Example

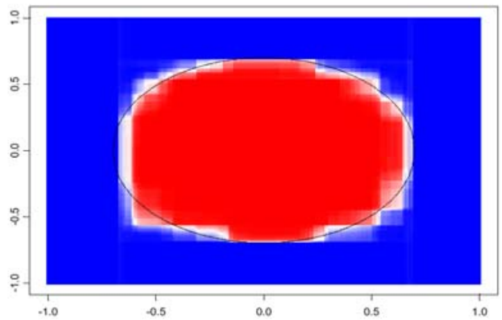



decision tree learning algorithm; very similar to ID3

## CART decision boundary



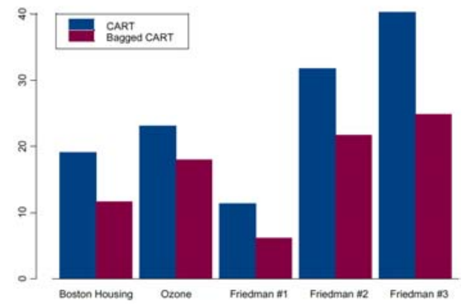
# 100 bagged trees




shades of blue/red indicate strength of vote for particular classification

# Regression results

Squared error loss




## Ensemble Creation III

## Boosting – Incorrect Version

- Maintain prob distribution over set of training ex
- Create M sets of training data iteratively:
- On iteration *m*
  - Draw m examples randomly **(~~like bagging~~)**
  - But use probability distribution to bias selection
  - Train classifier number *i* on this training set
  - Modify distribution: increase P of each error example
  - Assign confidence to classifier I = f(error)
- Create harder and harder learning problems...
- "~~Bagging~~ with ***optimized*** choice of examples"



## Ensemble Creation III

## Boosting

- Create 1st weight distribution (uniform) over training ex: $\{w_i^1\}$
- Create M classifiers iteratively:
  - On iteration *m*
  - Train $C_m$ by minimizing $\sum_i w_i^m \, I(y_m(x_i) \neq t_i)$ Indicator function: 0 or 1
  - Modify distribution: increase P of each error example
  - Assign confidence to classifier m = f(error)
- Combine
- Create harder and harder learning problems...
- ***Optimized*** choice of examples



## Bagging vs Boosting




## Bias, Variance, and Noise

- Variance: **$E[\,(h(x^*) - \underline{h(x^*)})^2\,]$**

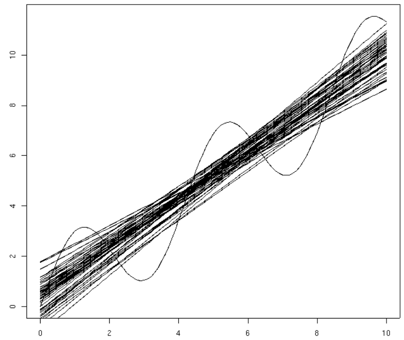
  Describes how much h(x*) varies from one training set S to another
- Bias: **$[\underline{h(x^*)} - f(x^*)]$**

  Describes the <u>average</u> error of h(x*).
- Noise: **$E[\,(y^* - f(x^*))^2\,] = E[\varepsilon^2] = \sigma^2$**

  Describes how much y* varies from f(x*)

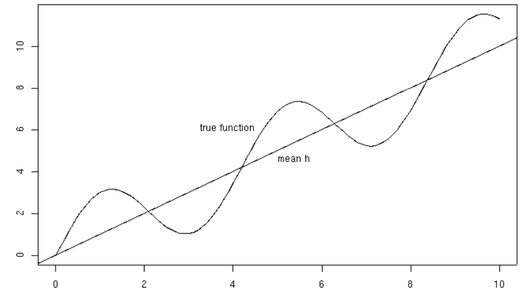
Slide from T Dietterich

## 50 fits (20 examples each)



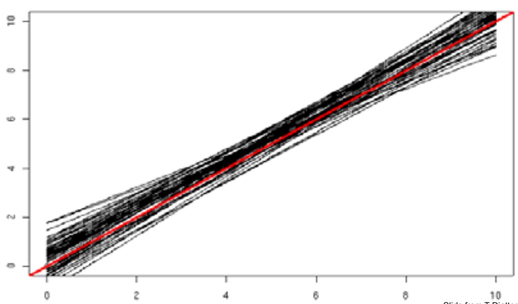
Slide from T Dietterich

## Bias



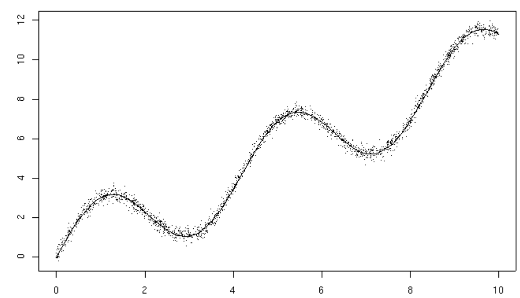

Slide from T Dietterich

## Variance



Slide from T Dietterich

## Noise



Slide from T Dietterich

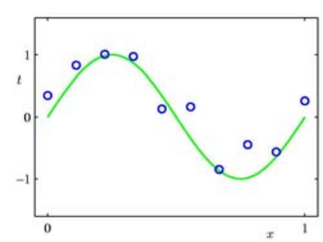
## Bias / Variance Tradeoff

Decreasing bias increases variance

Want the best compromise



## Polynomial Curve Fitting

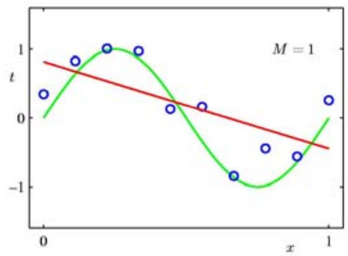



Sin($2\pi$x)

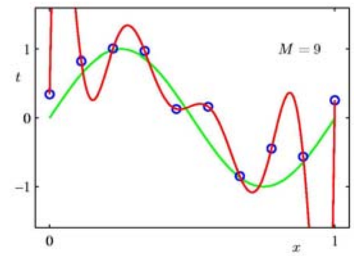
Hypothesis Space

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

## 1st Order Polynomial




## 9th Order Polynomial



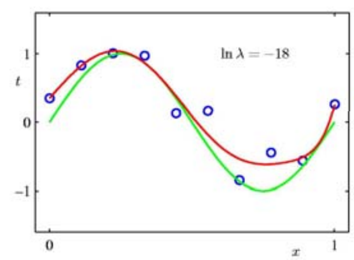

## Regularization

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

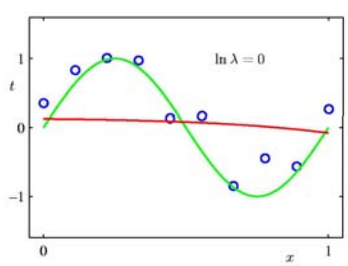
Penalize large coefficient values

Increasing λ trades bias for variance

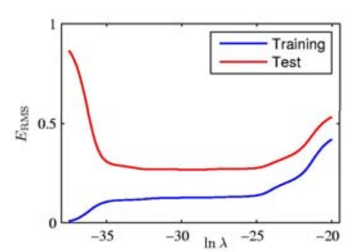
## Regularization: $\ln \lambda = -18$




## Regularization: $\ln \lambda = 0$




## Regularization: $E_{\mathrm{RMS}}$ *vs.* $\ln \lambda$



## Punchline

- Ensembles trade bias & variance
- Bagging reduces variance (bias almost unchanged)
  – Use with low bias learner, eg full decision tree
  – (Bagging decision stumps performs poorly)
- Boosting can reduce both
  – Often used with high bias learners, eg decision stumps

26

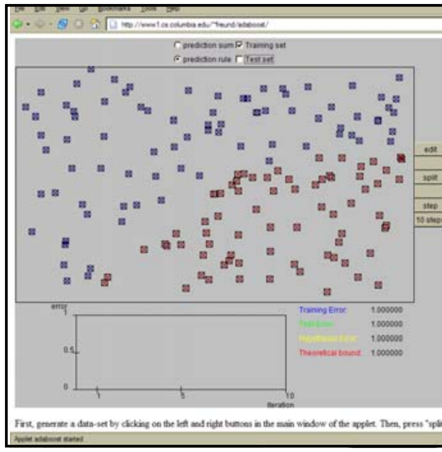
## Boosting [Schapire, 1989]

- Idea: run weak learner multiple times on (reweighted!) training data; weight learned classifiers ∝ their accuracy
- On each iteration $t$:
  – Learn a hypothesis, $h_t$, using distribution to weight examples
  – Compute a strength for this hypothesis – $\alpha_t$
  – Reweight training examples by how well they were classified
- Final classifier: $\boxed{h(x) = \text{sign}\left(\sum_i \alpha_i h_i(x)\right)}$
- **Practically useful**
- **Theoretically interesting**

## Boosting Applet

http://cseweb.ucsd.edu/~yfreund/adaboost/index.html

29

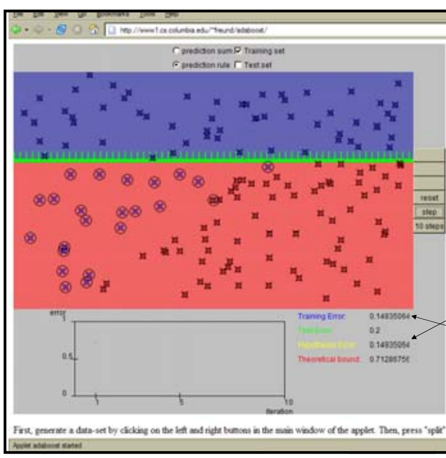


time = 0

blue/red = class

size of dot = weight

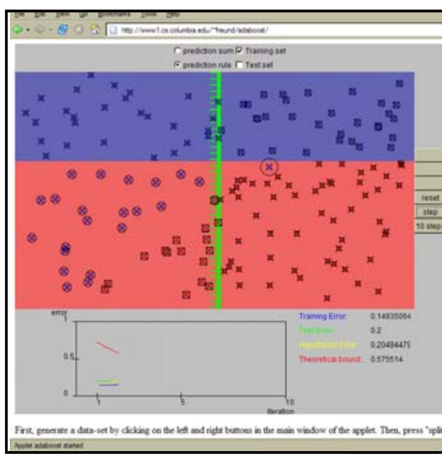
weak learner = Decision stub: horizontal or vertica



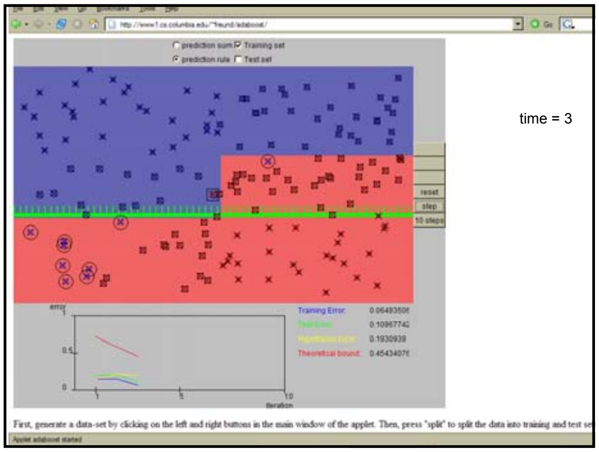
time = 1

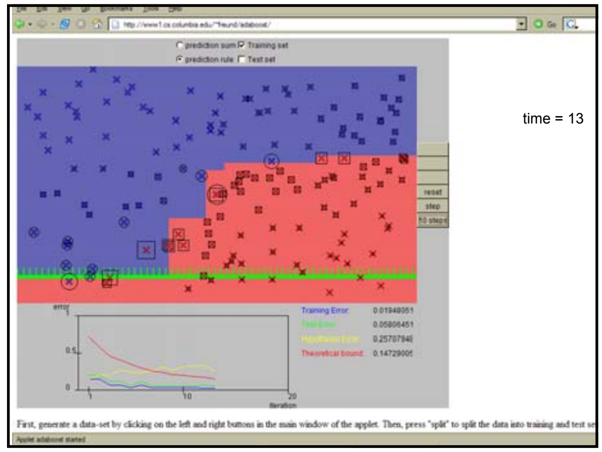
this hypothesis has error

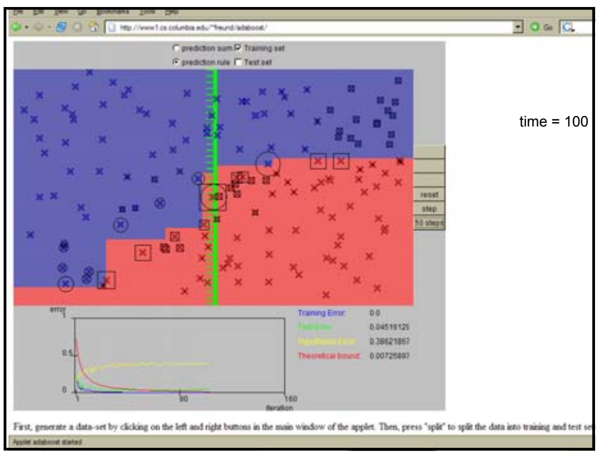
and so does this ensemble, since the ensemble contain just this one hypothes

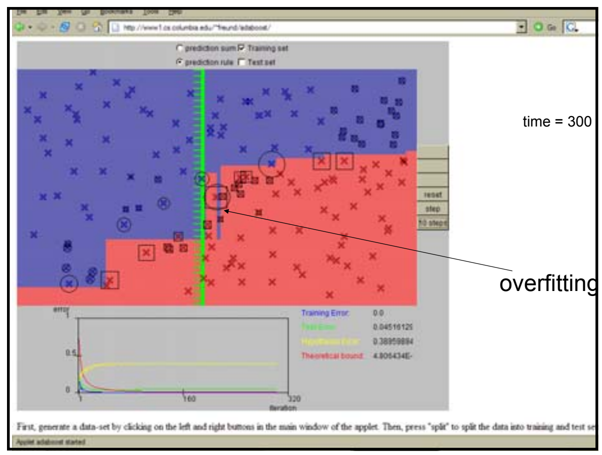


time = 2




# Learning from weighted data

- **Consider a weighted dataset**
  - D(j) – weight of *j* th training example ($\mathbf{x}^j$,$y^j$)
  - Interpretations:
    - *j* th training example counts as if it occurred D(j) times
    - If I were to "resample" data, I would get more samples of "heavier" data points
- **Now, always do weighted calculations:**
  - e.g., MLE for Naïve Bayes, redefine *Count(Y=y)* to be weighted count:

$$Count(Y = y) = \sum_{j=1}^{n} D(j)\delta(Y^j = y)$$

  - setting D(j)=1 (or any constant value!), for all j, will recreate unweighted case

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train base learner using distribution $D_t$.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

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

where $Z_t$ is a normalization factor

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

Output the final classifier:

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

Figure 1: The boosting algorithm AdaBoost.

How? Many possibilities. Will see one shortly!

Why? Reweight the data: examples $i$ that are misclassified will have higher weights!

- $y_i h_t(x_i) > 0 \rightarrow h_i$ correct
- $y_i h_t(x_i) < 0 \rightarrow h_i$ wrong
- $h_i$ correct, $\alpha_t > 0 \rightarrow D_{t+1}(i) < D_t(i)$
- $h_i$ wrong, $\alpha_t > 0 \rightarrow D_{t+1}(i) > D_t(i)$

Final Result: linear sum of "base" or "weak" classifier outputs.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train base learner using distribution $D_t$.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

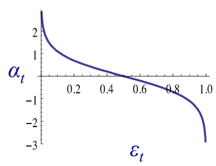
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$\boxed{\epsilon_t = P_{i \sim D_t(i)}[h_t(\mathbf{x}^i) \neq y^i]}$$

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

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

- $\varepsilon_t$ : error of $h_t$, weighted by $D_t$
  - $0 \leq \varepsilon_t \leq 1$
- $\alpha_t$ :
  - No errors: $\varepsilon_t$=0 → $\alpha_t$=∞
  - All errors: $\varepsilon_t$=1 → $\alpha_t$=−∞
  - Random: $\varepsilon_t$=0.5 → $\alpha_t$=0




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

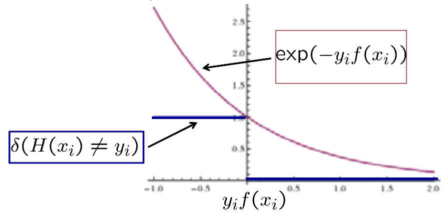
[Schapire, 1989]

Idea: choose $\alpha_t$ to minimize a bound on training error!

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

Where

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




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

[Schapire, 1989]

Idea: choose $\alpha_t$ to minimize a bound on training error!

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

Where

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

And

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

This equality isn't obvious! Can be shown with algebra (telescoping sums)!

**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 $Z_t$.
- $h_t$ is estimated as a black box, but can we solve for $\alpha_t$?

## Summary: choose $\alpha_t$ to minimize *error bound*

[Schapire, 1989]

We can squeeze this bound by choosing $\alpha_t$ on each iteration to minimize $Z_t$.

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

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

For boolean Y: differentiate, set equal to 0, there is a closed form solution! [Freund & Schapire '97]:

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

## Why Does Boosting Work So Well?

- On each iteration *t*:
  - Learn a classifier, $h_t$, using distribution to weight examples
  - Compute a strength for this classifier – $\alpha_t$
  - Reweight training examples by how well they were classified
- Final classifier:

$$\boxed{h(x) = \text{sign}\left(\sum_i \alpha_i h_i(x)\right)}$$

- Look familiar?
  - Another linear model, except…
  - Not in terms of original features
  - Creates ***new*** features (classifiers, $h_t$) while it learns weights

## Strong, weak classifiers

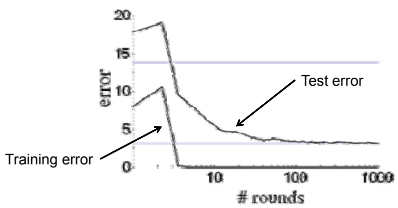
- If each classifier is (at least slightly) better than random: $\varepsilon_t < 0.5$
- Another bound on error:

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

- What does this imply about the training error?
  - Will reach zero!
  - Will get there exponentially fast!

## Boosting results – Digit recognition

[Schapire, 1989]

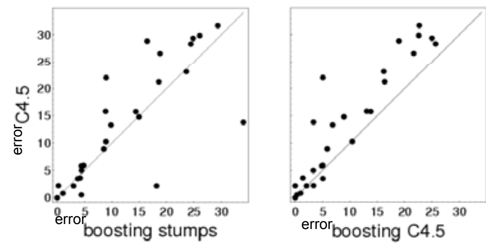


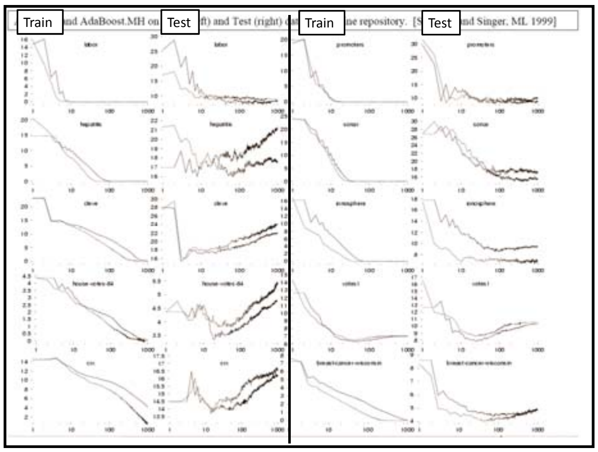

- Boosting:
  - Seems to be robust to overfitting
  - Test error can decrease even after training error is zero!!!

## Boosting: Experimental Results

[Freund & Schapire, 1996]

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




## What you need to know about Boosting

- Combine weak classifiers to get very strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can get zero training error
- AdaBoost algorithm
- Boosting v. Logistic Regression
  - Both linear model, boosting "learns" features
  - 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