# Adaboost

Derek Hoiem

March 31, 2004

# Outline

- Background
- Adaboost Algorithm
- Theory/Interpretations
- Practical Issues
- Face detection experiments

# What's So Good About Adaboost

- Improves classification accuracy
- Can be used with many different classifiers
- Commonly used in many areas
- Simple to implement
- Not prone to overfitting

# A Brief History

- Bootstrapping
- Bagging
- Boosting (Schapire 1989)
- Adaboost (Schapire 1995)

# Bootstrap Estimation

- Repeatedly draw *n* samples from *D*
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance

# Bagging - Aggregate Bootstrapping

- For i = 1 .. M
  - Draw $n^* < n$ samples from $D$ with replacement
  - Learn classifier $C_i$
- Final classifier is a vote of $C_1$ .. $C_M$
- Increases classifier stability/reduces variance

# Boosting (Schapire 1989)

- Randomly select $n_1 < n$ samples from $D$ without replacement to obtain $D_1$
  - Train weak learner $C_1$
- Select $n_2 < n$ samples from $D$ with half of the samples misclassified by $C_1$ to obtain $D_2$
  - Train weak learner $C_2$
- Select all samples from $D$ that $C_1$ and $C_2$ disagree on
  - Train weak learner $C_3$
- Final classifier is vote of weak learners

# Adaboost - Adaptive Boosting

- Instead of sampling, re-weight
  - Previous weak learner has only 50% accuracy over new distribution
- Can be used to learn weak classifiers
- Final classification based on weighted vote of weak classifiers

# Adaboost Terms

- Learner = Hypothesis = Classifier
- Weak Learner: < 50% error over any distribution
- Strong Classifier: thresholded linear combination of weak learner outputs

# Discrete Adaboost (DiscreteAB) (Friedman's wording)

**Discrete AdaBoost(Freund & Schapire 1996*b*)**

1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights $w_i$ on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

# Discrete Adaboost (DiscreteAB) (Freund and Schapire's wording)

**Algorithm AdaBoost**

**Input:** sequence of $N$ labeled examples $\langle (x_1, y_1), \ldots, (x_N, y_N) \rangle$
distribution $D$ over the $N$ examples
weak learning algorithm **WeakLearn**
integer $T$ specifying number of iterations

**Initialize** the weight vector: $w_i^1 = D(i)$ for $i = 1, \ldots, N$.

**Do for** $t = 1, 2, \ldots, T$

1. Set
$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$$
2. Call **WeakLearn**, providing it with the distribution $\mathbf{p}^t$; get back a hypothesis $h_t : X \to [0, 1]$.
3. Calculate the error of $h_t$: $\epsilon_t = \sum_{i=1}^N p_i^t |h_t(x_i) - y_i|$.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Set the new weights vector to be
$$w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$$

**Output** the hypothesis

$$h_f(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \left(\log \frac{1}{\beta_t}\right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases} .$$

# Adaboost with Confidence Weighted Predictions (RealAB)

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:

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

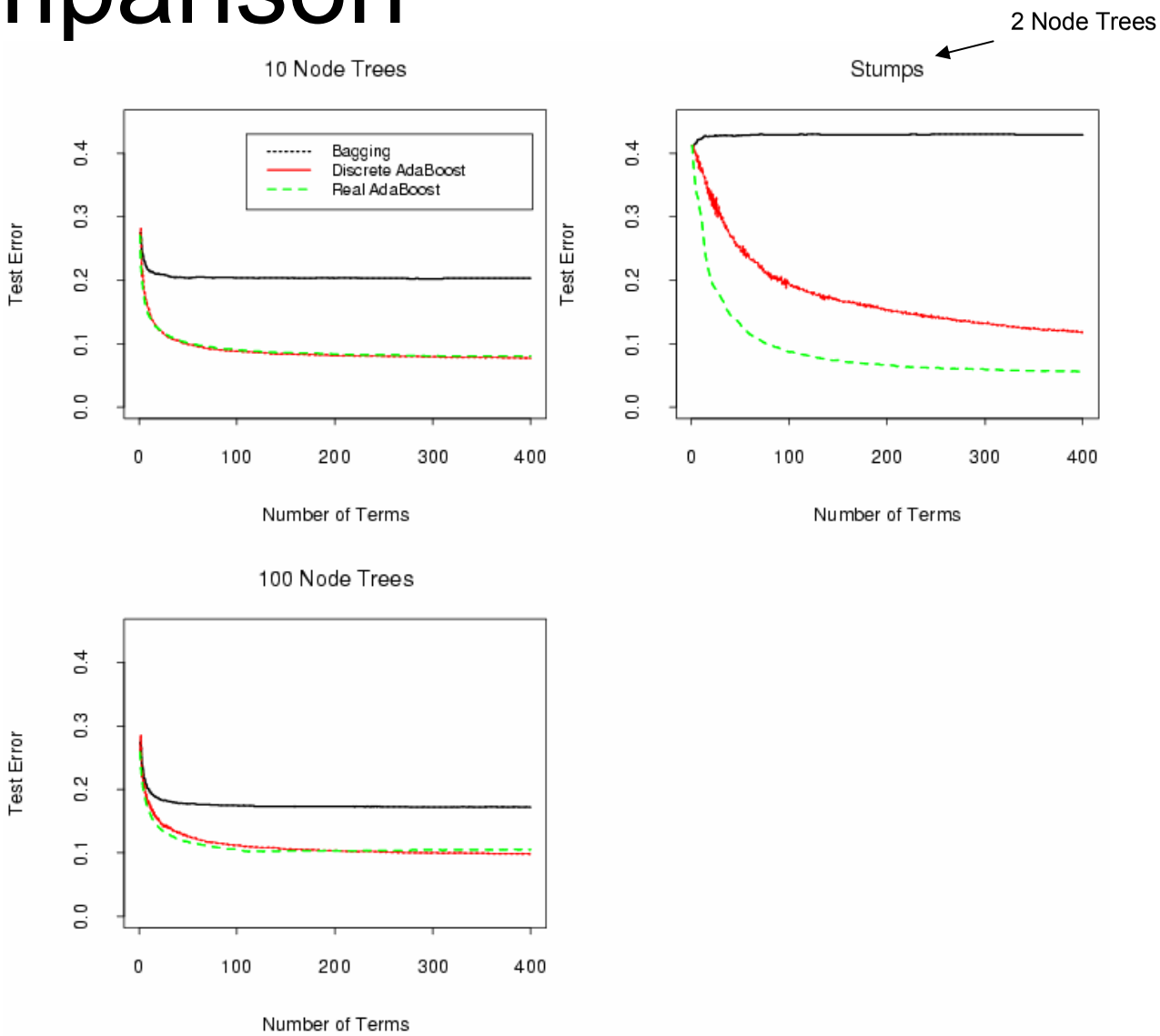
where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution).

Output the final classifier:

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

Figure 1: The boosting algorithm AdaBoost.

# Comparison


10 Node Trees
Stumps
2 Node Trees
Bagging
Discrete AdaBoost
Real AdaBoost
Test Error
Number of Terms
0.0
0.1
0.2
0.3
0.4
0
100
200
300
400
100 Node Trees

# Bound on Training Error (Schapire)

$$\begin{aligned}\frac{1}{m}\sum_i [\![H(x_i) \neq y_i]\!] &\leq \frac{1}{m}\sum_i \exp(-y_i f(x_i)) \\ &= \sum_i \left(\prod_t Z_t\right) D_{T+1}(i) \\ &= \prod_t Z_t \,.\end{aligned}$$

$$Z_t = \sum_i D_t(i)\exp(-\alpha_t y_i h_t(x_i))$$

# Finding a weak hypothesis

- Train classifier (as usual) on weighted training data
- Some weak learners can minimize Z by gradient descent

$$Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

- Sometimes we can ignore alpha (when the weak learner output can be freely scaled)

$$Z = \sum_i D(i) \exp(-y_i h(x_i))$$

# Choosing Alpha

- Choose alpha to minimize Z

$$Z(\alpha) = Z = \sum_i D(i) e^{-\alpha u_i}$$

- Results in 50% error rate in latest weak learner

$$\mathrm{E}_{i \sim D_{t+1}} [y_i h_t(x_i)] = 0$$

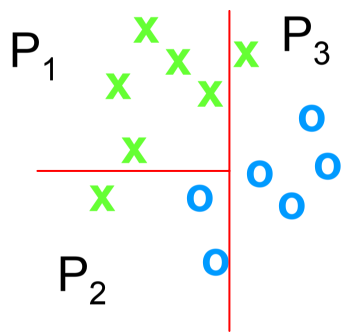
- In general, compute numerically

# Special Case

Domain-partitioning weak hypothesis (e.g. decision trees, histograms)

$$W_b^j = \sum_{i:x_i \in X_j \wedge y_i = b} D(i) = \Pr_{i \sim D}[x_i \in X_j \wedge y_i = b]$$

$$c_j = \frac{1}{2} \ln \left( \frac{W_+^j}{W_-^j} \right) \qquad Z = 2 \sum_j \sqrt{W_+^j W_-^j}$$




$W_x^1 = 5/13$, $W_o^1 = 0/13$

$W_x^2 = 1/13$, $W_o^2 = 2/13$

$W_x^3 = 1/13$, $W_o^3 = 4/13$

Z = 2 (0 + sqrt(2)/13 + 2/13) = .525

# Smoothing Predictions

- Equivalent to adding prior in partitioning case

$$c_j = \tfrac{1}{2} \ln \left( \frac{W_+^j + \varepsilon}{W_-^j + \varepsilon} \right)$$

- Confidence bound by

$$\tfrac{1}{2} \ln \left( \frac{1+\varepsilon}{\varepsilon} \right) \approx \tfrac{1}{2} \ln(1/\varepsilon)$$

# Justification for the Error Function

- Adaboost minimizes:

$$J(F) = E(e^{-yF(x)}) \qquad F(x) = \sum_{m=1}^{M} f_m(x)$$

- Provides a differentiable upper bound on training error (Shapire)

- Minimizing J(F) is equivalent up to 2nd order Taylor expansion about F = 0 to maximizing expected binomial log-likelihood (Friedman)

# Probabilistic Interpretation (Friedman)

**Lemma 1** *$E(e^{-yF(x)})$ is minimized at*

$$F(x) = \frac{1}{2}\log\frac{P(y=1|x)}{P(y=-1|x)}.$$

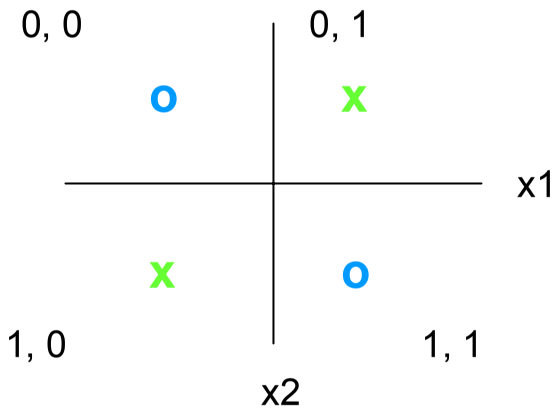
*Hence*

$$\begin{aligned} P(y=1|x) &= \frac{e^{F(x)}}{e^{-F(x)}+e^{F(x)}} \\ P(y=-1|x) &= \frac{e^{-F(x)}}{e^{-F(x)}+e^{F(x)}}. \end{aligned}$$

# Misinterpretations of the Probabilistic Interpretation

- Lemma 1 applies to the true distribution
  - Only applies to the training set
  - Note that $P(y|\mathbf{x}) = \{0, 1\}$ for the training set in most cases
- Adaboost will converge to the global minimum
  - Greedy process → local minimum
  - Complexity of strong learner limited by complexities of weak learners

# Example of Joint Distribution that Cannot Be Learned with Naïve Weak Learners




P(o | 0,0) = 1

P(x | 0,1) = 1

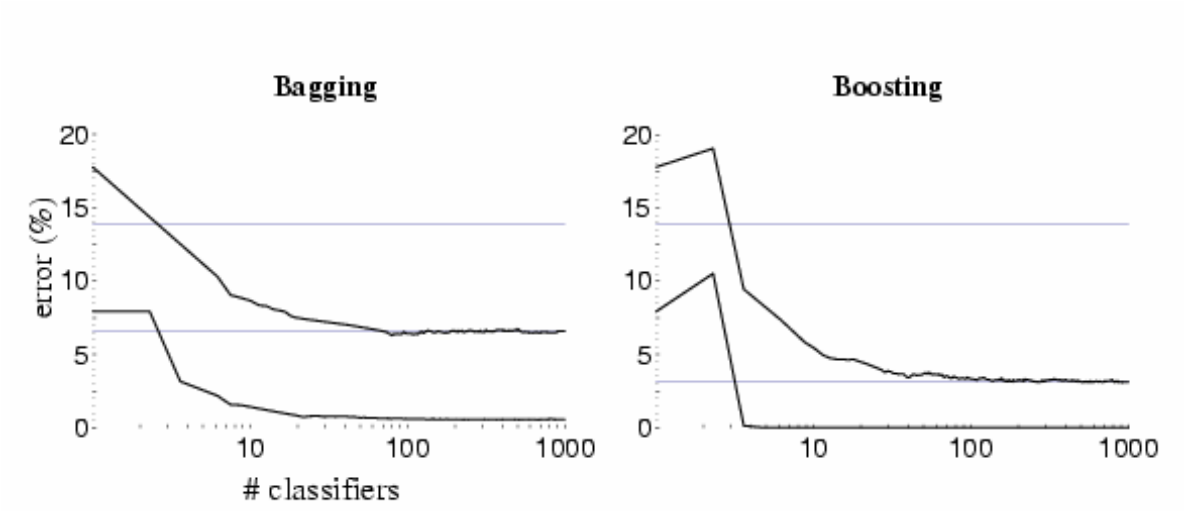
P(x | 1,0) = 1

P(o | 1,1) = 1

$$H(x1,x2) = a*x1 + b*(1-x1) + c*x2 + d*(1-x2) = (a-b)*x1 + (c-d)*x2 + (b+d)$$

Linear decision for non-linear problem when using naïve weak learners

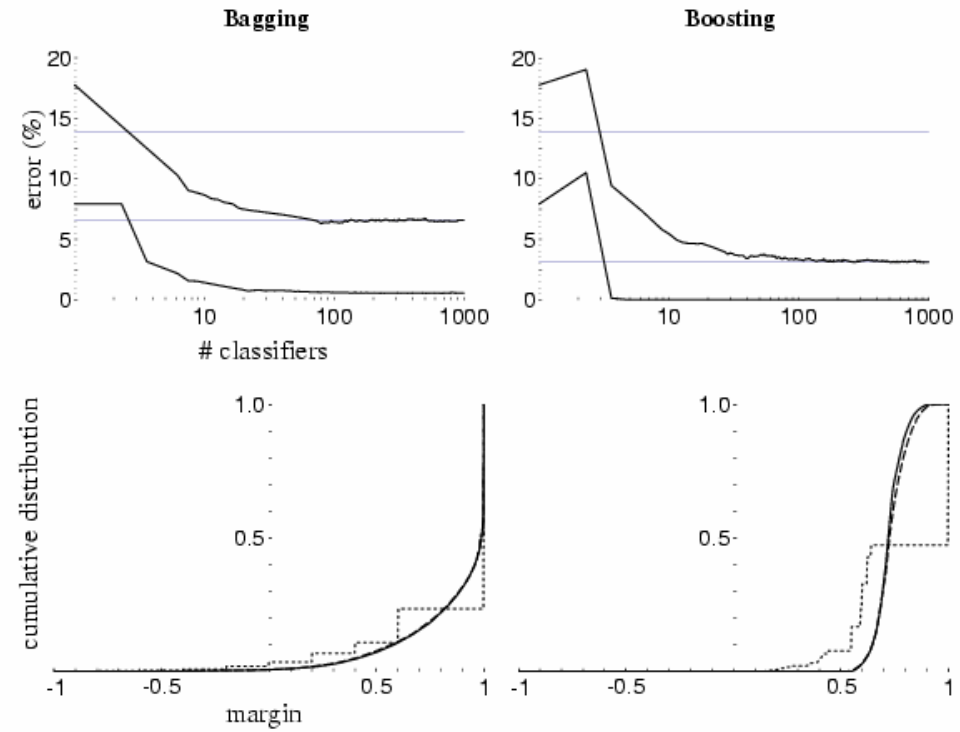
# Immunity to Overfitting?



# Adaboost as Logistic Regression (Friedman)

- Additive Logistic Regression: Fitting class conditional probability log ratio with additive terms
- DiscreteAB builds an additive logistic regression model via Newton-like updates for minimizing $E(e^{-yF(x)})$
- RealAB fits an additive logistic regression model by stage-wise and approximate optimization of $E(e^{-yF(x)})$
- Even after training error reaches zero, AB produces a “purer” solution probabilistically

# Adaboost as Margin Maximization (Schapire)



# Bound on Generalization Error

Confidence of correct decision

VC dimension

$$\Pr_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\left(\frac{d\log^2(m/d)}{\theta^2} + \log(1/\delta)\right)^{1/2}\right)$$

Confidence Margin

Number of training examples

Bound confidence term

To loose to be of practical value:

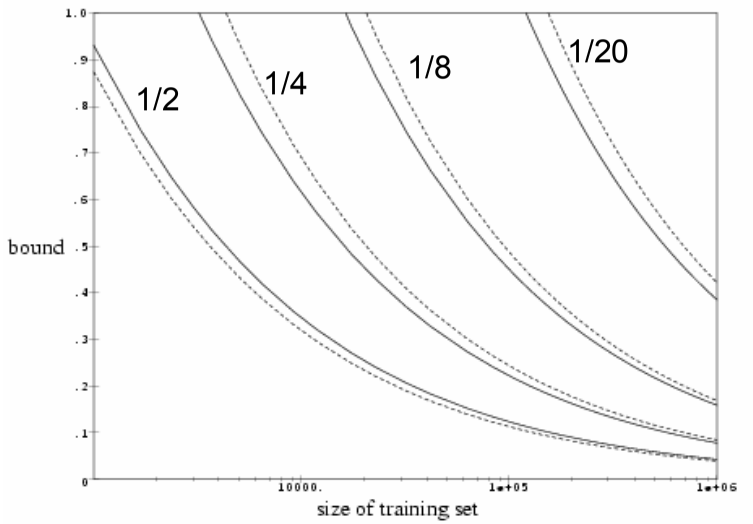



Figure 3: A few plots of the second and third terms in the bound given in Equation (8) (solid lines) and their approximation by the second term in Equation (9) (dotted lines). The horizontal axis denotes the number of training examples (with a logarithmic scale) and the vertical axis denotes the value of the bound. All plots are for $\delta = 0.01$ and $|\mathcal{H}| = 10^6$. Each pair of close lines corresponds to a different value of $\theta$, counting the pairs from the upper right to the lower left, the values of $\theta$ are 1/20, 1/8, 1/4 and 1/2.

# Maximizing the margin…

- But Adaboost doesn't necessarily maximize the margin on the test set (Ratsch)
- Ratsch proposes an algorithm (Adaboost*) that does so

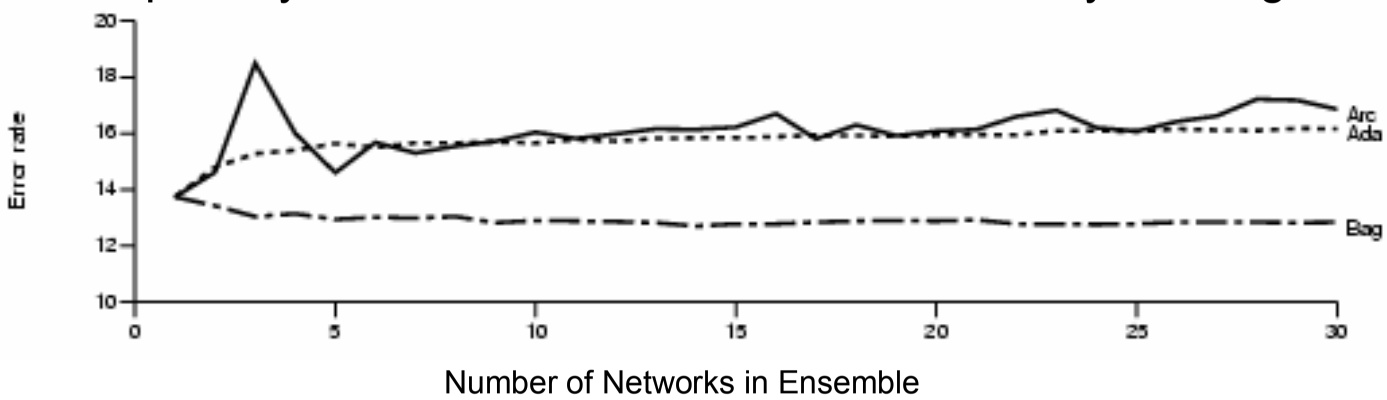
| | C4.5 | AdaBoost | Marginal AdaBoost | AdaBoost* |
|---|---|---|---|---|
| $E_{gen}$ | $7.4 \pm 0.11\%$ | $4.0 \pm 0.11\%$ | $3.6 \pm 0.10\%$ | $3.5 \pm 0.10\%$ |
| $\rho$ | — | $0.31 \pm 0.01$ | $0.58 \pm 0.01$ | $0.55 \pm 0.01$ |

Table 1: Estimated generalization performances and margins with confidence intervals for decision trees (C4.5), AdaBoost, Marginal AdaBoost and AdaBoost* on the toy data. All numbers are averaged over 200 splits into 100 training and 19900 test examples.

# Adaboost and Noisy Data

- Examples with the largest gap between the label and the classifier prediction get the most weight
- What if the label is wrong?

Opitz: Synthetic data with 20% one-sided noisy labeling



# WeightBoost (Jin)

- Uses input-dependent weighting factors for weak learners

*Table 2.* Classification errors for the WeightBoost, AdaBoost and Weight Decay

| Collection Name | C4.5 | AdaBoost | Weight Decay | $\epsilon$-Boost | Weight Boost |
|---|---|---|---|---|---|
| Ionosphere | 9.1% | 6.8% | 5.7% | 6.8% | 6.2% |
| German | 26.9% | 26.3% | 26.7% | 24.7% | 24.7% |
| Pima-Indians-Diabetes | 25.2% | 24.7% | 25.1% | 23.9% | 22.6% |
| Breast Cancer | 5.4% | 4.5% | 3.7% | 3.2% | 3.3% |
| wpbc | 28.8% | 26.3% | 21.1% | 21.1% | 19.9% |
| wdbc | 6.1% | 3.5% | 3.0% | 3.7% | 3.0% |
| Contra-ceptive | 31.5% | 31% | 29.8% | 30.4% | 27.6% |
| Spambase | 7.2% | 5.8% | 4.9% | 4.5% | 4.2% |

Text Categorization

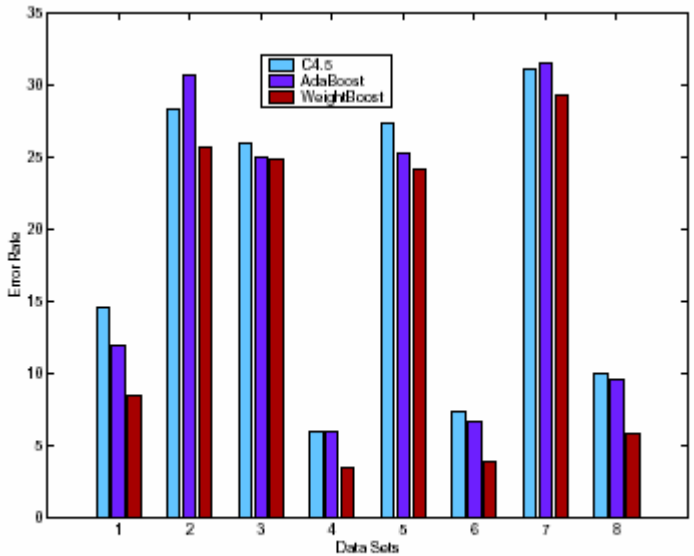



*Figure 1.* Classification Errors with 10% Noise

# BrownBoost (Freund)

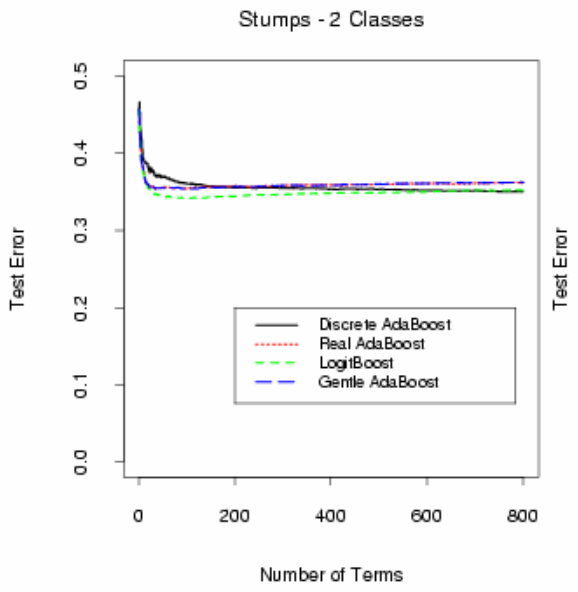
- Non-monotonic weighting function
  - Examples far from boundary decrease in weight
- Set a target error rate – algorithm attempts to achieve that error rate
- No results posted (ever)

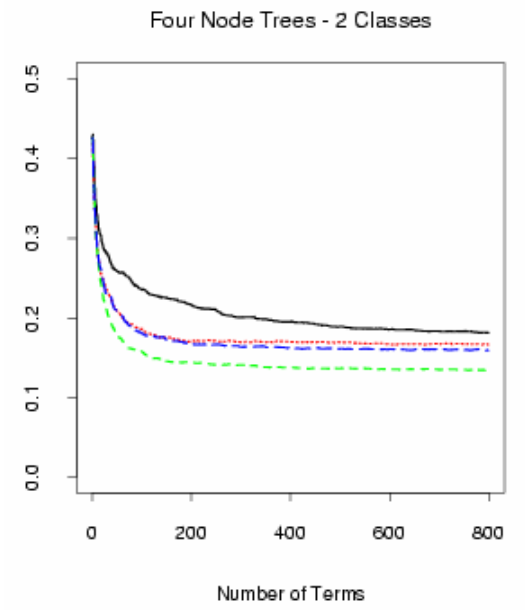
# Adaboost Variants Proposed By Friedman

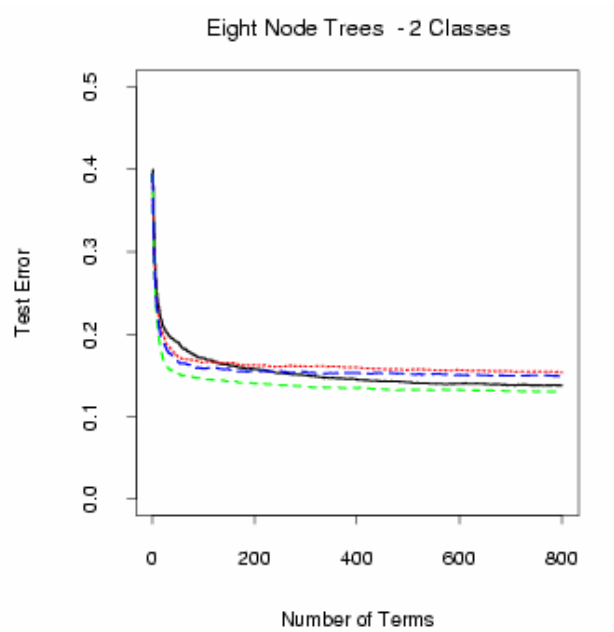
- LogitBoost
  - Solves $\min_{f(x)} E_{w(x)}\left(F(x) + \frac{1}{2}\frac{y^* - p(x)}{p(x)(1-p(x))} - (F(x)+f(x))\right)^2$
  - Requires care to avoid numerical problems

- GentleBoost
  - Update is $f_m(x) = P(y=1 \mid x) - P(y=0 \mid x)$ instead of $f_m(x) = \frac{1}{2}\log\frac{P_w(y=1|x)}{P_w(y=-1|x)}$
    - Bounded [0 1]

# Comparison (Friedman)



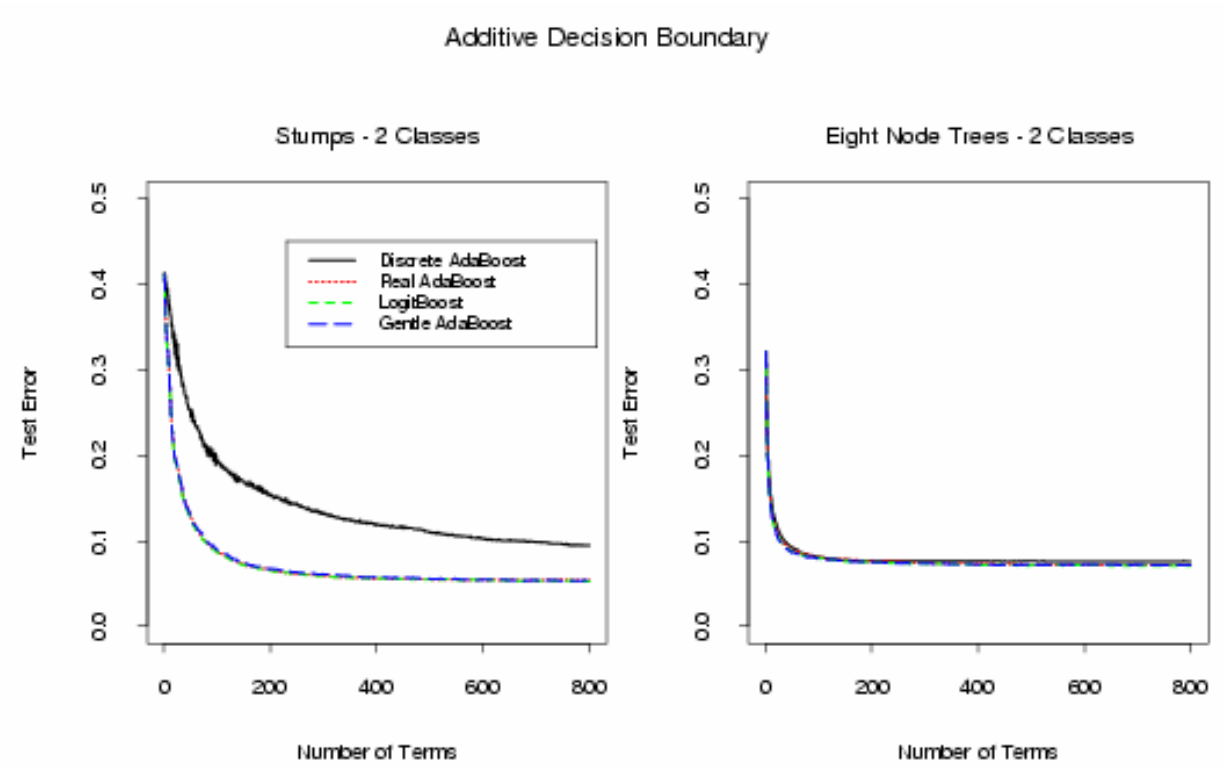

Synthetic, non-additive decision boundary

# Complexity of Weak Learner

- Complexity of strong classifier limited by complexities of weak learners

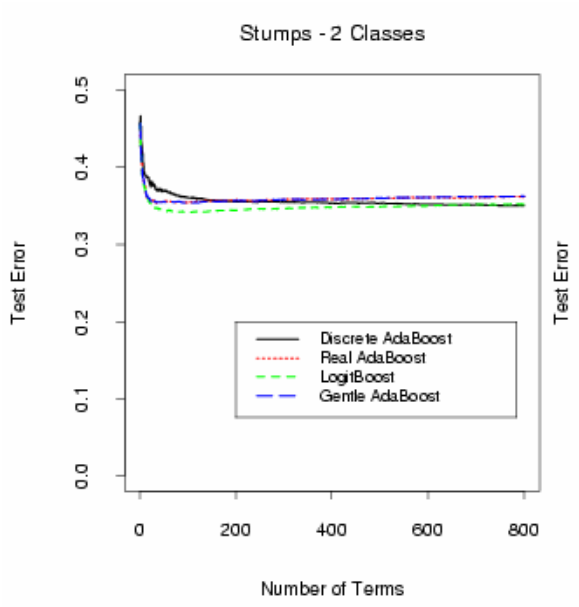
- Example:
  - Weak Learner: stubs → WL $Dim_{VC} = 2$
  - Input space: $R^N$ → Strong $Dim_{VC} = N+1$
  - N+1 partitions can have arbitrary confidences assigned

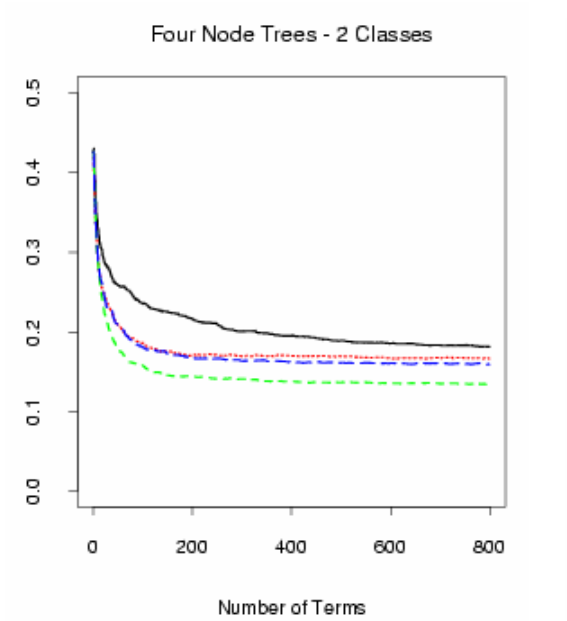
# Complexity of the Weak Learner

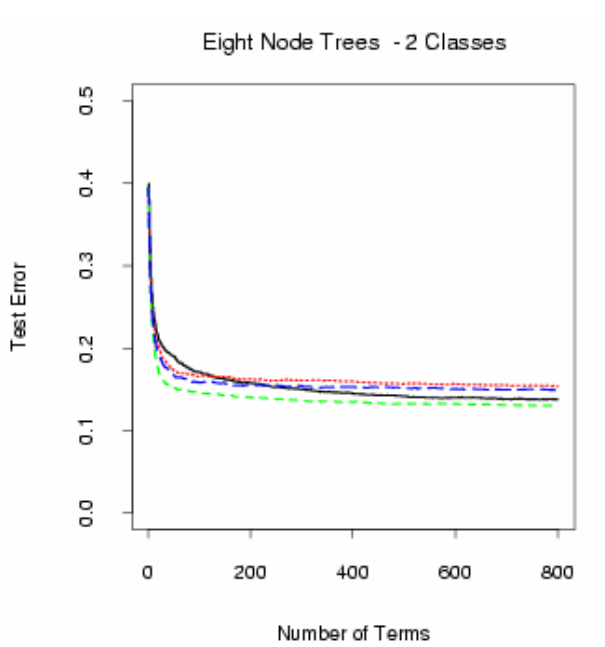


# Complexity of the Weak Learner

Non-Additive Decision Boundary



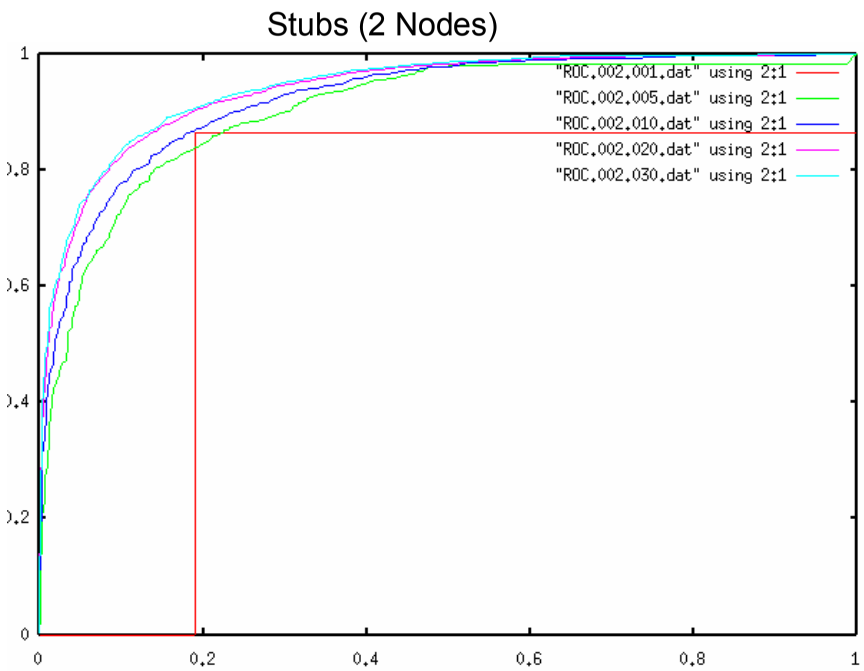



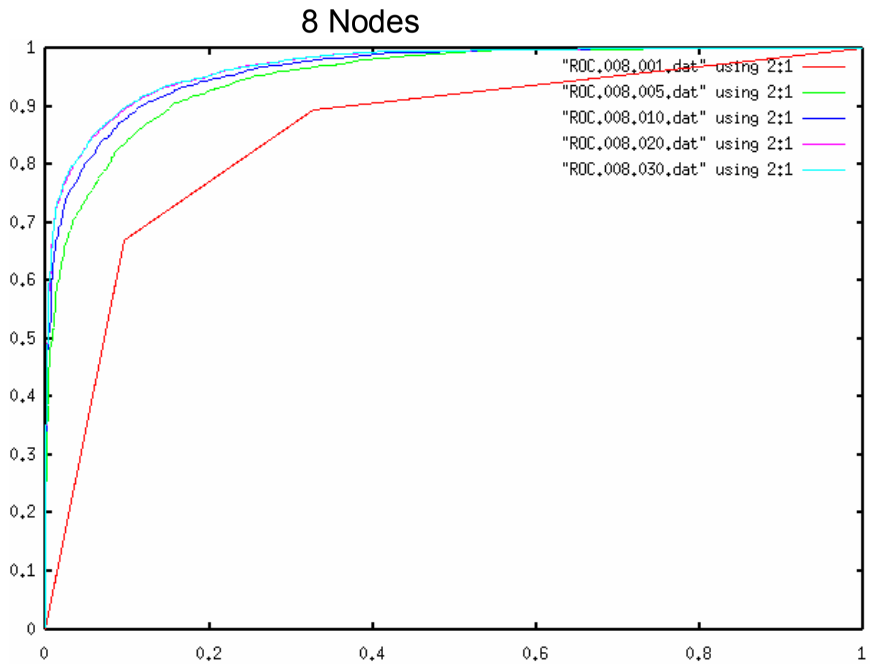



# Adaboost in Face Detection

| Detector | Adaboost Variant | Weak Learner |
|---|---|---|
| Viola-Jones | DiscreteAB | Stubs |
| FloatBoost | FloatBoost | 1-D Histograms |
| KLBoost | KLBoost | 1-D Histograms |
| Schneiderman | RealAB | One Group of N-D Histograms |

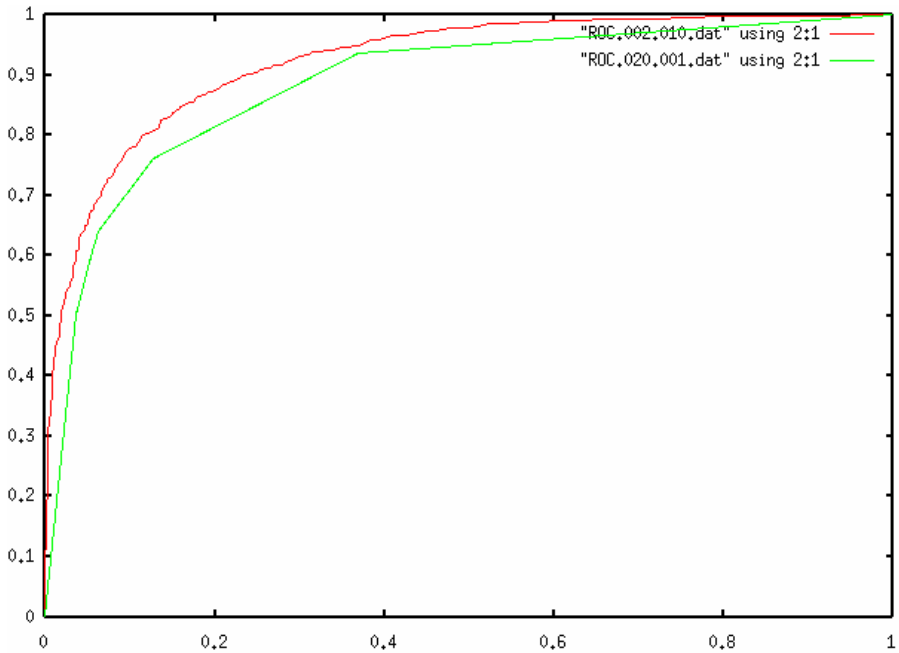
# Boosted Trees – Face Detection



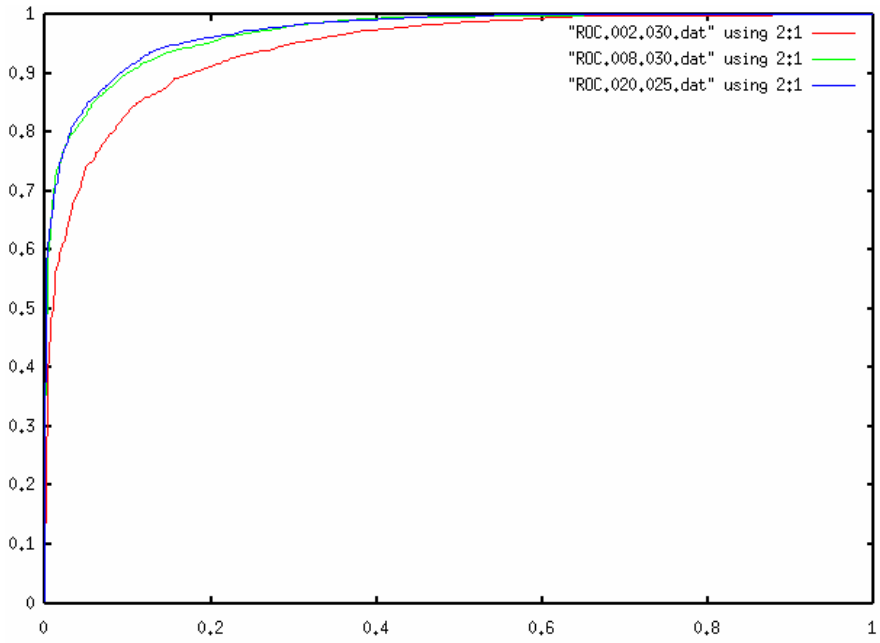



# Boosted Trees – Face Detection

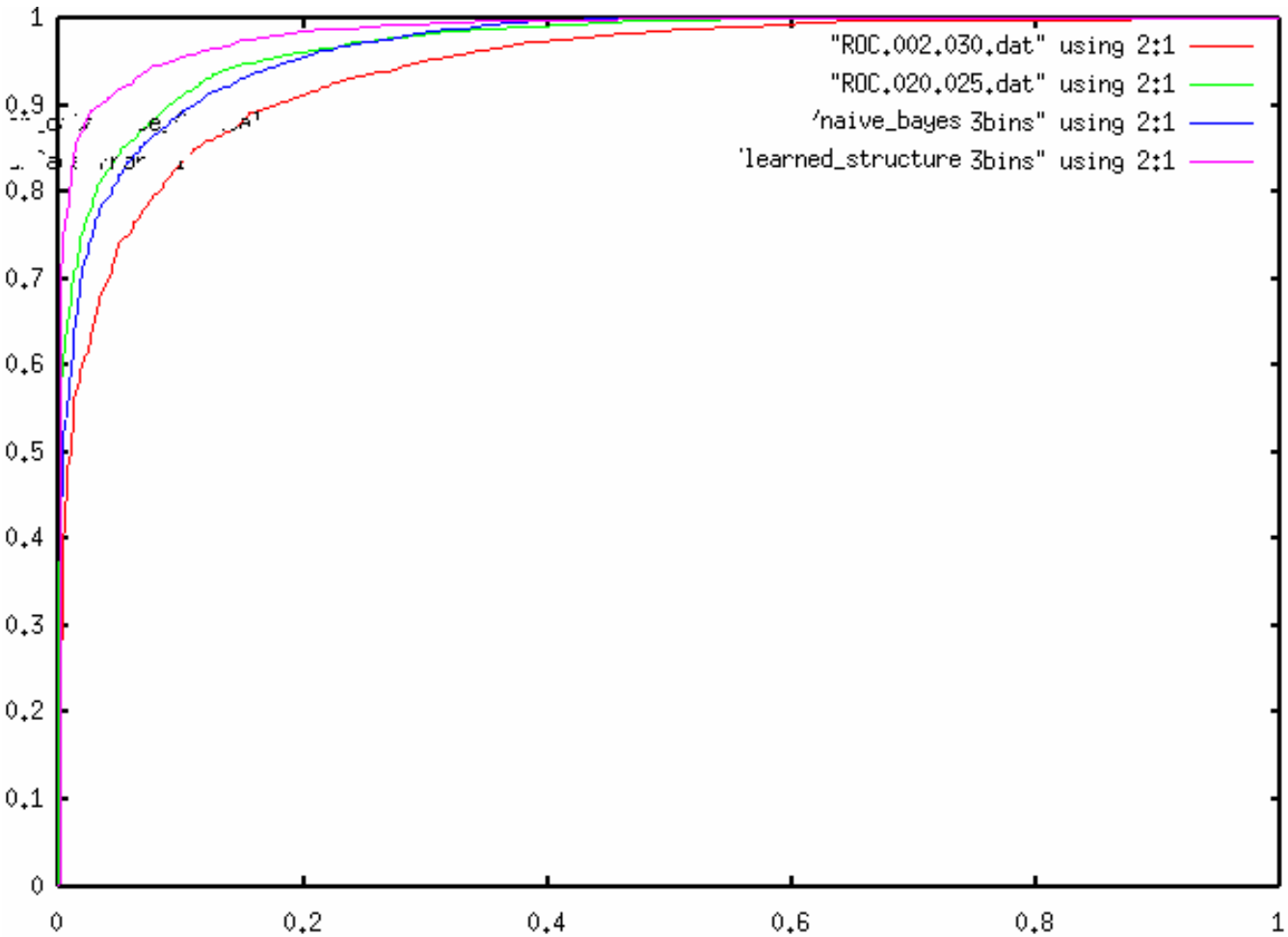
Ten Stubs vs. One 20 Node



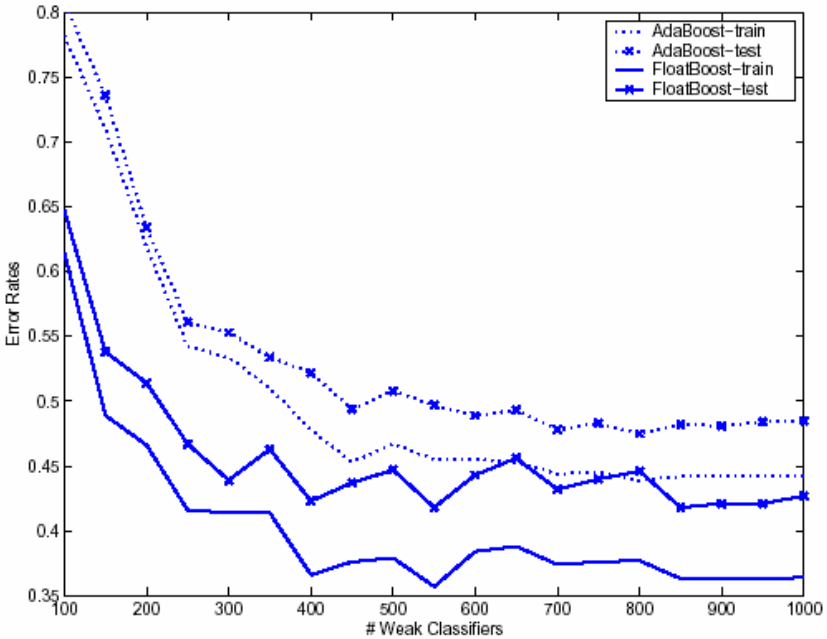

2, 8, and 20 Nodes



# Boosted Trees vs. Bayes Net



# FloatBoost (Li)



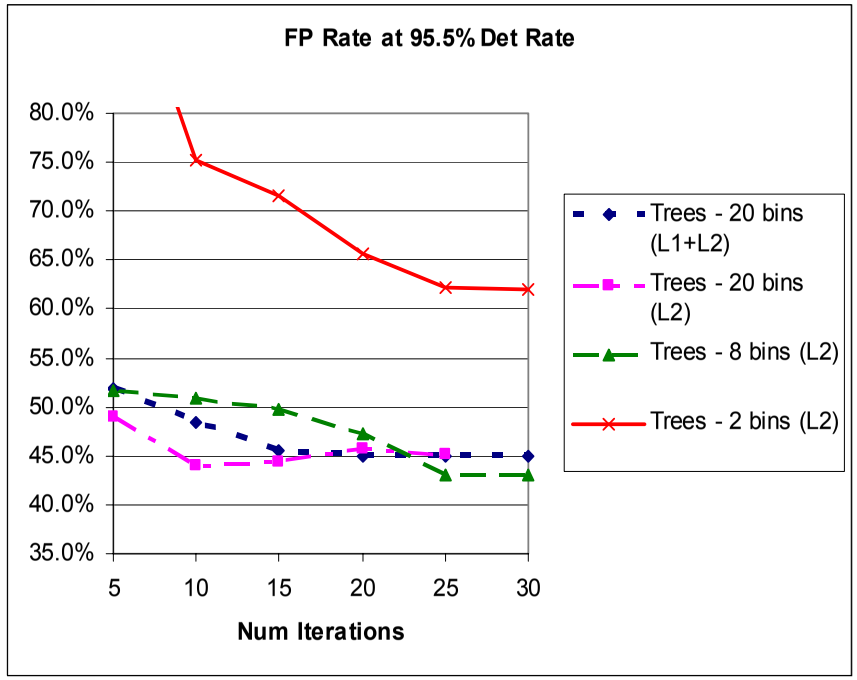




Figure 5: Error Rates of FloatBoost vs AdaBoost for frontal face detection.

1. Less greedy approach (FloatBoost) yields better results
2. Stubs are not sufficient for vision

# When to Use Adaboost

- Give it a try for any classification problem
- Be wary if using noisy/unlabeled data

# How to Use Adaboost

- Start with RealAB (easy to implement)
- Possibly try a variant, such as FloatBoost, WeightBoost, or LogitBoost
- Try varying the complexity of the weak learner
- Try forming the weak learner to minimize Z (or some similar goal)

# Conclusion

- Adaboost can improve classifier accuracy for many problems
- Complexity of weak learner is important
- Alternative boosting algorithms exist to deal with many of Adaboost's problems

# References


- Duda, Hart, ect – *Pattern Classification*
- Freund – “An adaptive version of the boost by majority algorithm”
- Freund – “Experiments with a new boosting algorithm”
- Freund, Schapire – “A decision-theoretic generalization of on-line learning and an application to boosting”
- Friedman, Hastie, etc – “Additive Logistic Regression: A Statistical View of Boosting”
- Jin, Liu, etc (CMU) – “A New Boosting Algorithm Using Input-Dependent Regularizer”
- Li, Zhang, etc – “Floatboost Learning for Classification”
- Opitz, Maclin – “Popular Ensemble Methods: An Empirical Study”
- Ratsch, Warmuth – “Efficient Margin Maximization with Boosting”
- Schapire, Freund, etc – “Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods”
- Schapire, Singer – “Improved Boosting Algorithms Using Confidence-Weighted Predictions”
- Schapire – “The Boosting Approach to Machine Learning: An overview”
- Zhang, Li, etc – “Multi-view Face Detection with Floatboost”

# Adaboost Variants Proposed By Friedman

- LogitBoost

**LogitBoost (2 classes)**

1. Start with weights $w_i = 1/N$ $i = 1, 2, \ldots, N$, $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Compute the working response and weights

$$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))}$$
$$w_i = p(x_i)(1 - p(x_i))$$

   (b) Fit the function $f_m(x)$ by a weighted least-squares regression of $z_i$ to $x_i$ using weights $w_i$.
   (c) Update $F(x) \leftarrow F(x) + \frac{1}{2} f_m(x)$ and $p(x) \leftarrow \frac{e^{F(x)}}{e^{F(x)} + e^{-F(x)}}$.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

**Algorithm 3:** *An adaptive Newton algorithm for fitting an additive logistic regression model.*

# Adaboost Variants Proposed By Friedman

- GentleBoost

**Gentle AdaBoost**

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$, $F(x) = 0$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the regression function $f_m(x)$ by weighted least-squares of $y_i$ to $x_i$ with weights $w_i$.
   (b) Update $F(x) \leftarrow F(x) + f_m(x)$
   (c) Update $w_i \leftarrow w_i e^{-y_i f_m(x_i)}$ and renormalize.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

**Algorithm 4:** *A modified version of the Real AdaBoost algorithm, using Newton stepping rather than exact optimization at each step*