Boosting

Adrian Barbu

Boosting – Combining Classifiers

- Given a set of “weak” classifiers
  - Aimed at solving the same problem
  - Different
    - Algorithm types
    - Parameters
    - Data types
    - Training sets
    - Sub-problems

- Combine them into a “strong” classifier
  - More powerful than any of them

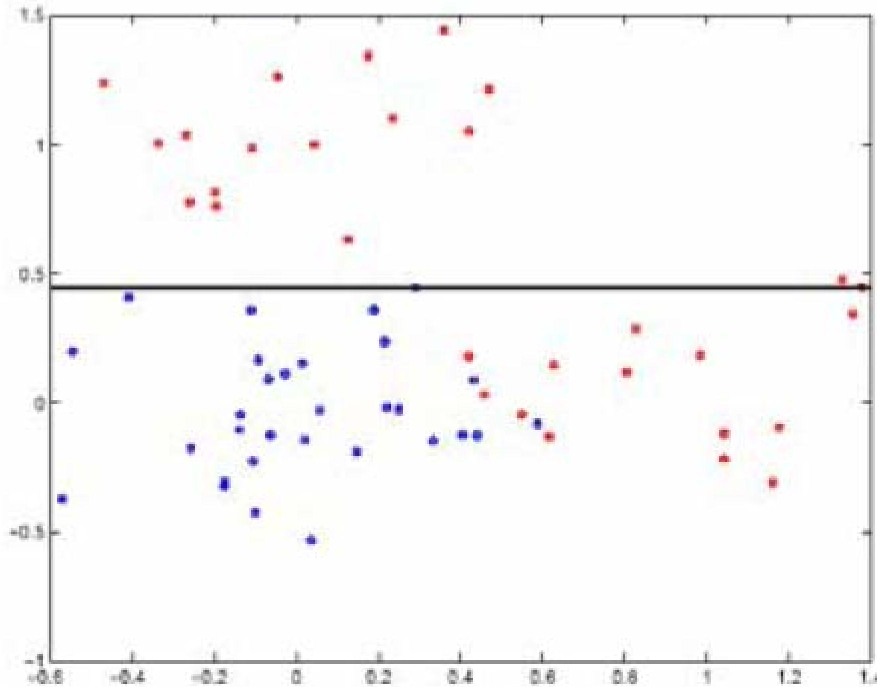
Early Methods

- Boosting by filtering (Schapire 1990)
  - Run weak learner on filtered sample sets (more and more difficult)
  - Combine weak hypotheses
  - Requires knowledge on the performance of weak learner
- Boosting by majority (Freund 1995)
  - Run weak learner on weighted example set
  - Combine weak hypotheses linearly
  - Requires knowledge on the performance of weak learner
- Bagging (Breiman 1996)
  - Train weak learner on bootstrap replicates of the training set
  - Average weak hypotheses or majority voting
  - Similar to Random Forest (what is the difference?)
  - Reduces variance

Weak Classifiers

- Component classifiers
  - Output y=$\pm$1
  - Based on one single feature x_k
  - Decision stump (cut down decision tree)

$$h(x,\theta) = \mathsf{sign}(wx_k + b), \qquad \theta = (k, w, b)$$



  - Should be better than random

Combining Weak Classifiers

- Weighted Majority Voting

$$\text{classify}(x) = \text{sign}(h(x))$$
$$h(x) = \alpha_1 h(x, \theta_1) + \ldots + \alpha_m h(x, \theta_m)$$

- Issues:
  - Criterion that is optimizing (loss function)
  - How to find best weak classifier combination and weights?

Measuring Error

- Loss Function:

$$\lambda(y, h(\mathbf{x}))$$

- Generalization Error:

$$L(h) = E[\lambda(y, h(\mathbf{x})]$$

- Want to find h to minimize *L(h)*
  - Don't know the true distribution that generated the data
- Minimize the sample average (empirical error) instead:

$$\widehat{L}(h) = \frac{1}{N} \sum_{i=1}^{N} \lambda(y_i, h(\mathbf{x}_i))$$

AdaBoost Loss

- Exponential Loss

$$\lambda(y, h(\mathbf{x})) = \exp(-yh(x))$$

- Empirical Error is:

$$\begin{aligned}
\hat{L}(h) &= \frac{1}{N}\sum_{i=1}^{N} \exp[-y_i \sum_{k=1}^{m} \alpha_k h(\mathbf{x}_i, \theta_k)] \\
&= \frac{1}{N}\sum_{i=1}^{N} \exp[-y_i \sum_{k=1}^{m-1} \alpha_k h(\mathbf{x}_i, \theta_k) - y_i \alpha_m h(\mathbf{x}_i, \theta_m)] \\
&= \frac{1}{N}\sum_{i=1}^{N} \exp[-y_i \sum_{k=1}^{m-1} \alpha_k h(\mathbf{x}_i, \theta_k)] \exp[-y_i \alpha_m h(\mathbf{x}_i, \theta_m)] \\
&= \frac{1}{N}\sum_{i=1}^{N} W_i^{m-1} \exp[-y_i \alpha_m h(\mathbf{x}_i, \theta_m)]
\end{aligned}$$

  - After m-1 iterations, each sample has a weight W_i^{m-1}
  - Correctly classified samples have small weight
  - Error is a weighted loss based on the last classifier

Linearization

- We can use a linear approximation when α is small

$$\exp[-y_i \alpha_m h(\mathbf{x}_i, \theta_m)] \approx 1 - y_i \alpha_m h(\mathbf{x}_i, \theta_m)$$

- We get

$$\hat{L}(h) \approx \frac{1}{N} \sum_{i=1}^{N} W_i^{m-1} [1 - y_i \alpha_m h(\mathbf{x}_i, \theta_m)]$$

$$= \frac{1}{N} [\sum_{i=1}^{N} W_i^{m-1} - \alpha_m \sum_{i=1}^{N} W_i^{m-1} y_i h(\mathbf{x}_i, \theta_m)]$$

- One way to minimize $\hat{L}(h)$ at stage m
  1. Maximize $\sum_{i=1}^{N} W_i^{m-1} y_i h(\mathbf{x}_i, \theta_m)$
  2. Find α_m that minimizes the original loss

either 1 or 2

$$\sum_{i=1}^{N} W_i^{m-1} \exp[-y_i \alpha_m h(\mathbf{x}_i, \theta_m)]$$

- This is **Boosting**

AdaBoost Training

- Start with N training samples $(\mathbf{x}_i, y_i)$
- Assign equal weights to all positives and to all negatives

$$W_i^0 = \frac{0.5}{|\{j, y_j = y_i\}|}$$

- Repeat m times:
  1. At step k find θ_k for which

$$\epsilon_k(\theta_k) = 0.5 - \frac{1}{2}\sum_{i=1}^{N} W_i^{k-1} y_i h(\mathbf{x}_i, \theta_k) = \sum_{i=1}^{N} W_i^{k-1} I[y_i \neq h(\mathbf{x}_i, \theta_k)]$$

     is minimum.
  2. Compute the k-th classifier weight $\alpha_k = 0.5 \ln \frac{1-\epsilon_k}{\epsilon_k}$
  3. Update the weights

$$W_i^k = W_i^{k-1} \exp[-\alpha_k y_i h(\mathbf{x}_i, \theta_k)]$$

     and re-normalize them
     - Weights of all positives must sum to 0.5, same for negatives

AdaBoost Classification

- Given a feature vector $\mathbf{x}$

$$\text{classify}(\mathbf{x}) = \text{sign}(h(\mathbf{x}))$$
$$h(\mathbf{x}) = \alpha_1 h(\mathbf{x}, \theta_1) + \ldots + \alpha_m h(\mathbf{x}, \theta_m)$$

- Can also use a normalized h

$$h(\mathbf{x}) = \frac{\alpha_1 h(\mathbf{x}, \theta_1) + \ldots + \alpha_m h(\mathbf{x}, \theta_m)}{\alpha_1 + \ldots + \alpha_m}$$

AdaBoost Summary

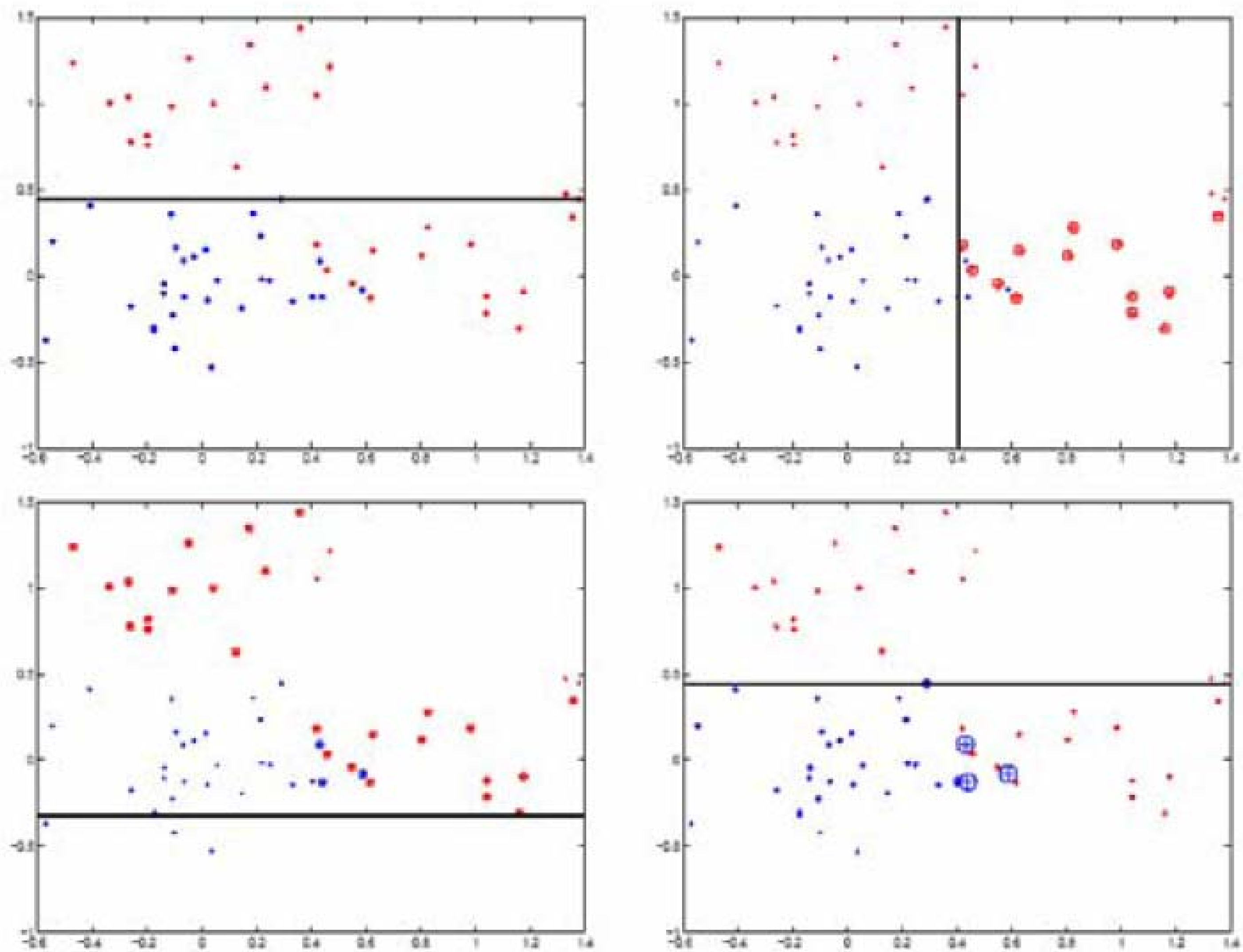
- Input:
  - *N* training samples $T_N = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$
  - a weak base learner $h = h(\mathbf{x}, \theta)$
- Initialize: equal weights to all positives and to all negatives

$$W_i^0 = \frac{0.5}{|\{j, y_j = y_i\}|}$$

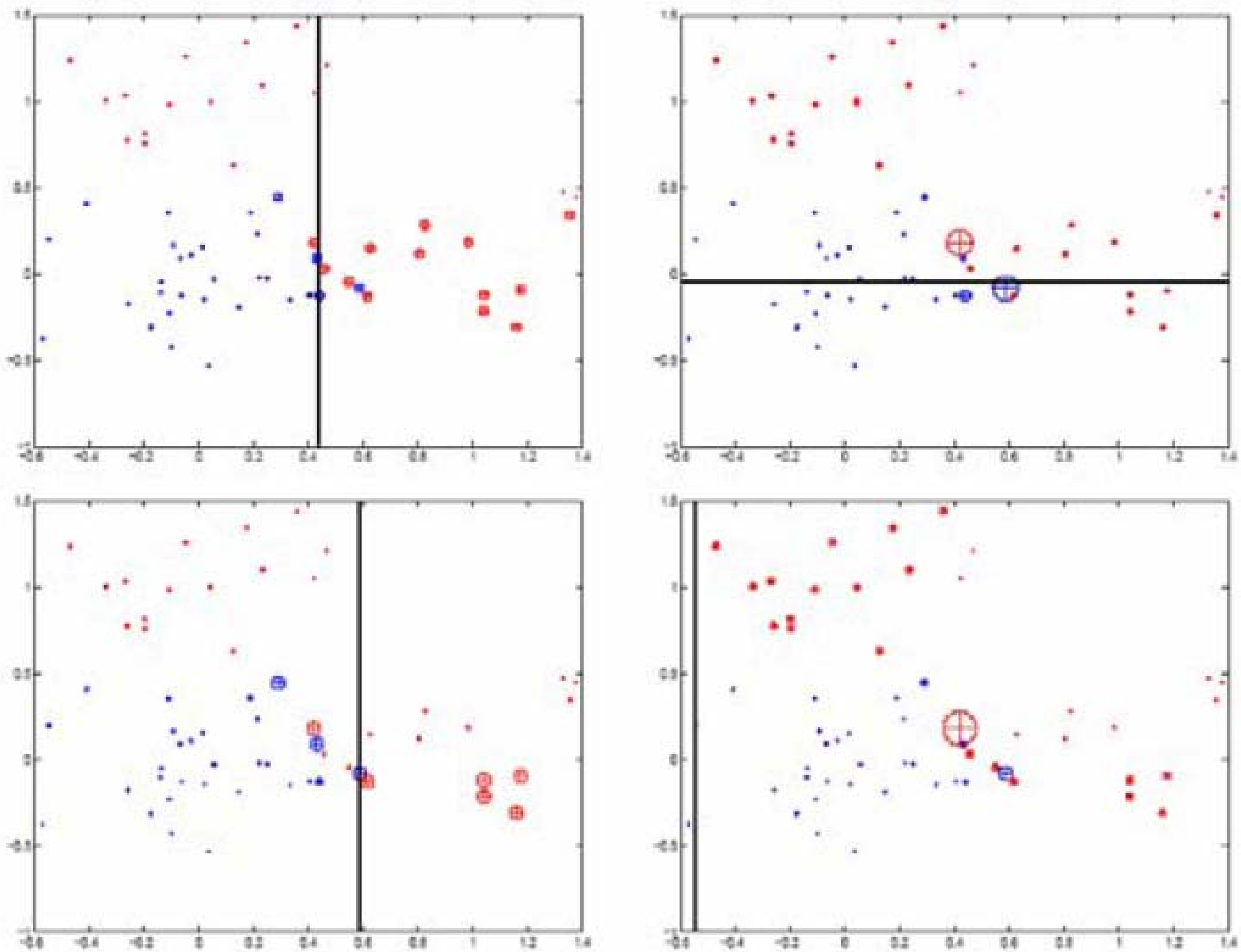
- Repeat for k = 1…m:
  1. Train base learner using the samples and weights W_i^{k-1} obtain weak classifier $h(\mathbf{x}, \theta_k)$
  2. Compute weighted error ϵ_k
  3. Compute classifier weight α_k
  4. Update sample weights for next iteration W_i^k
- Output: final classifier as a linear combination

$$h(x) = \alpha_1 h(x, \theta_1) + ... + \alpha_m h(x, \theta_m)$$

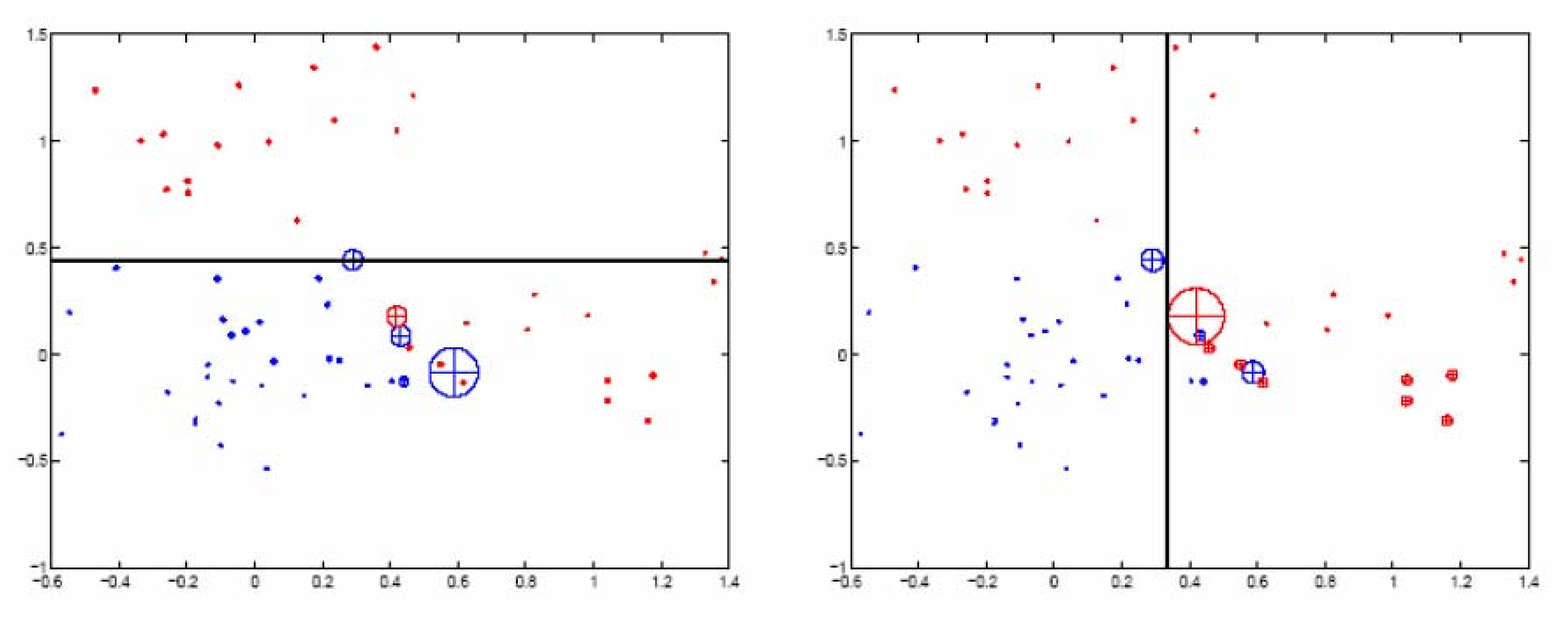
Example



Example (Continued)



Example (Continued)



Base Learners

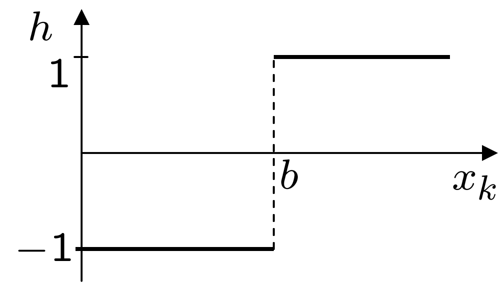
- Weak learners used in practice:
  - Decision stumps (axis parallel splits)
  - Histogram based weak classifier
  - Decision trees (e.g. C4.5 by Quinlan 1996)
  - Multi-layer neural networks
  - …

- Base learners need to train on weighted samples
  - Modify them to use weights along with the samples
  - Obtain representative training set by sampling (with replacement) according to the distribution defined by the weights

Histogram-Based Weak Classifier

- Decision stump $\theta = (k, w, b), w = \pm 1$

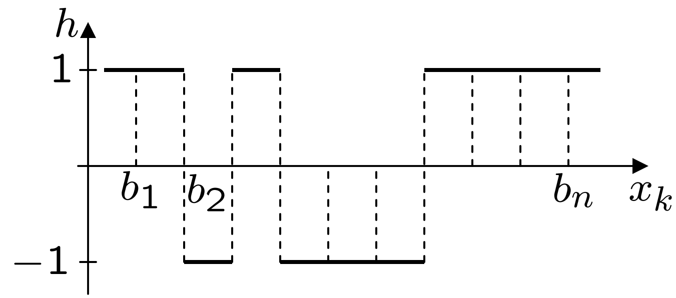
$$h(\mathbf{x}, \theta) = \begin{cases} 1 & \text{if } wx_k > b \\ -1 & \text{else} \end{cases}$$



- Histogram classifier

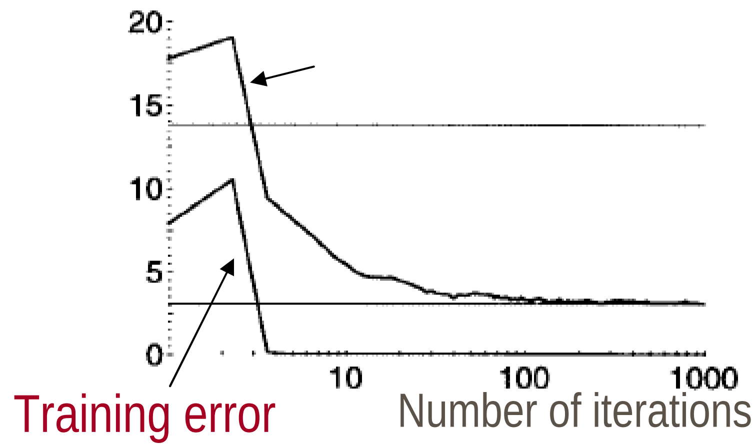
$$\theta = (k, b_1, ..., b_n, s_1, ..., s_n), s_i = \pm 1$$

$$h(\mathbf{x}, \theta) = \begin{cases} s_1 & \text{if } x_k < b_2 \\ \dots & \\ s_i & \text{if } b_i \le x_k < b_{i+1} \\ \dots & \\ s_n & \text{if } b_n \le x_k \end{cases}$$



Boosting Performance

```
pos=20000, neg=20000, nWeak=10, nFeatures=100511
i=0  best=21485 e=0.088 5,H ,4,3,4,34,8
i=1  best=21547 e=0.206 5,H ,4,1,3,38,8
i=2  best=49332 e=0.278 5,H ,10,7,6,30,6
i=3  best=21518 e=0.288 5,H ,4,1,4,36,8
i=4  best=37010 e=0.329 5,H ,7,13,7,16,4
i=5  best=5848 e=0.334 5,H ,0,3,6,28,4
i=6  best=21105 e=0.356 5,H ,4,7,2,22,12
i=7  best=21524 e=0.359 5,H ,4,3,3,36,8
i=8  best=91880 e=0.368 5,H ,20,7,4,26,4
i=9  best=50299 e=0.361 5,H ,11,15,5,8,3
1 detection rate=0.9820 false positive=0.1212
```



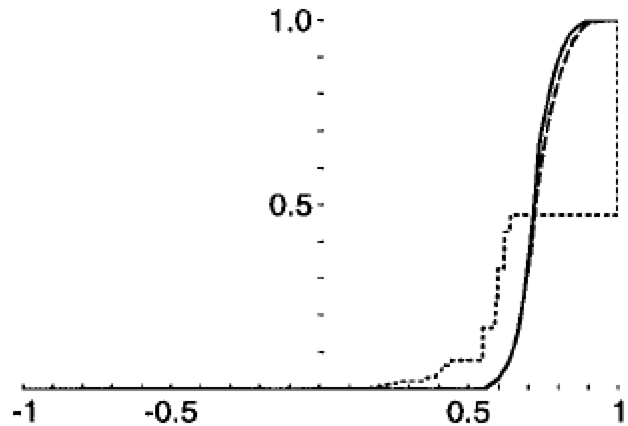
- The misclassification error of the best weak classifier at the last Boosting stage slowly grows over time
- Magic!
  - The training and testing errors are surprisingly small!
- Even after the training error goes to zero, boosting iterations can still improve the generalization error!!

Training Margin

- The margin shows how much agreement is between the weak classifiers

$$\text{margin}_h(\mathbf{x}_i, y_i) = y_i \frac{\alpha_1 h(\mathbf{x}_i, \theta_1) + ... + \alpha_m h(\mathbf{x}_i, \theta_m)}{\alpha_1 + ... + \alpha_m}$$

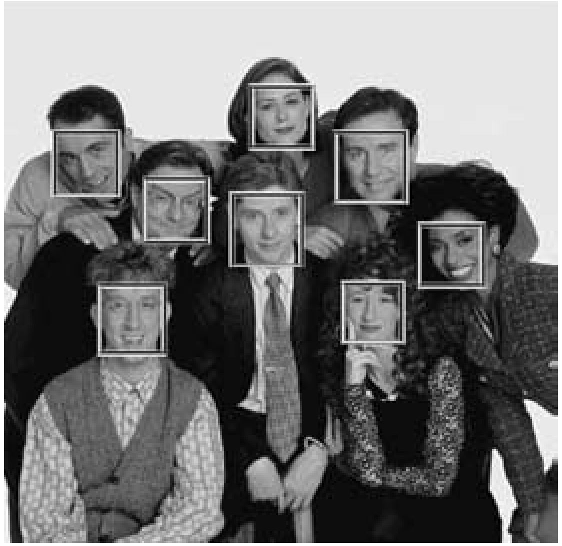
  - $\text{margin} \in [-1, 1]$, negative for misclassified samples
- As training progresses most margins are close to 1



CDF of margins

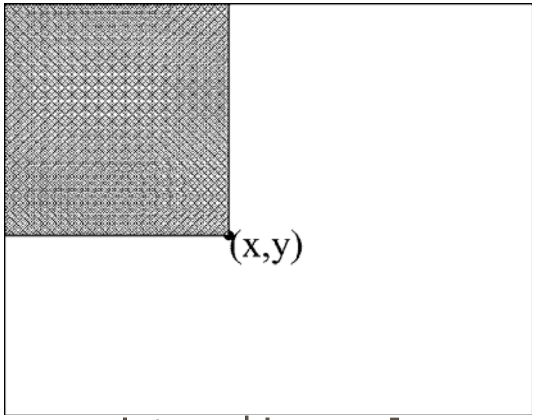
Application: Face Detection

- Viola-Jones, 2004
- Face Detection
  - Detect horizontal faces
  - Three parameters:
    - Position (x,y)
    - Scale s
  - Need to be fast: 0.2 sec/ image
  - Classifier inside a 24x24 pixel window
    - Translated at all positions
    - Rescaled at scales 1,2,…
  - Slow if applied at all locations and all scales

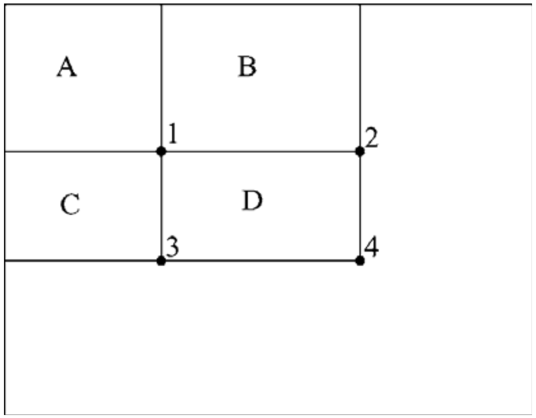


Integral Image

- Integral Image
  - Value at (x,y)= sum of intensities of all pixels in the rectangle (0,0)→(x,y)
  - Sum of pixels of $D=J_4+J_1-J_2-J_3$
  - Constant time
    - Independent of the size of D
  - Can have integral image of square intensities
    - Obtain variance inside rectangles in constant time



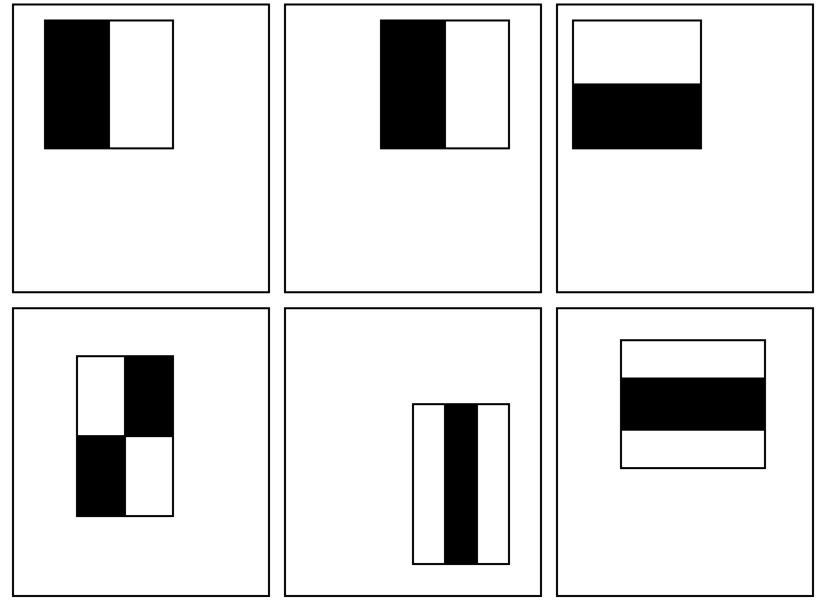
Integral Image J



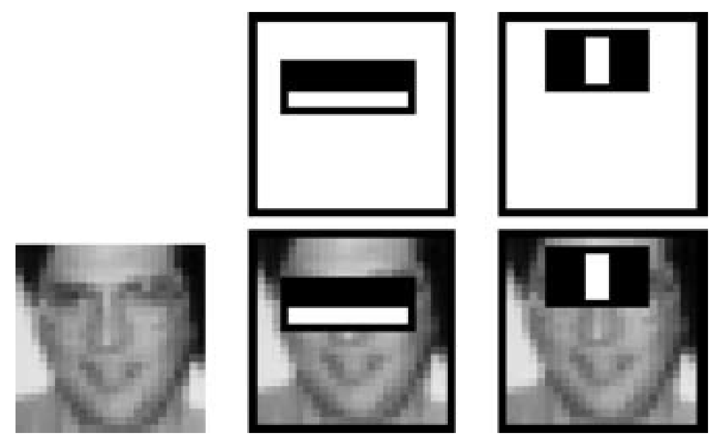
Sum of pixels of $D=J_4+J_1-J_2-J_3$

Haar Features

- Haar Features
  - Sum of pixels in white rectangles-sum of pixels in black rectangles
  - Rectangles relative the 24x24 pixel window
  - Many combinations of rectangles
    - Relative locations
    - Sizes
    - Type of combinations
  - Brightness constancy:
    - Sum of positive (white) rectangles= Sum of negative (black) rectangles
  - Can easily obtain >100k features



Haar Features

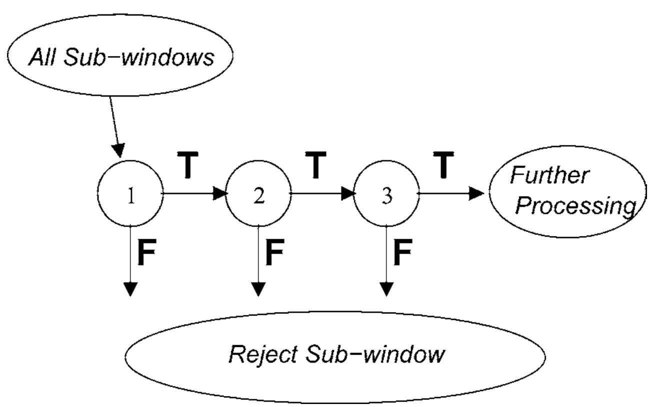


Best two features for face detection

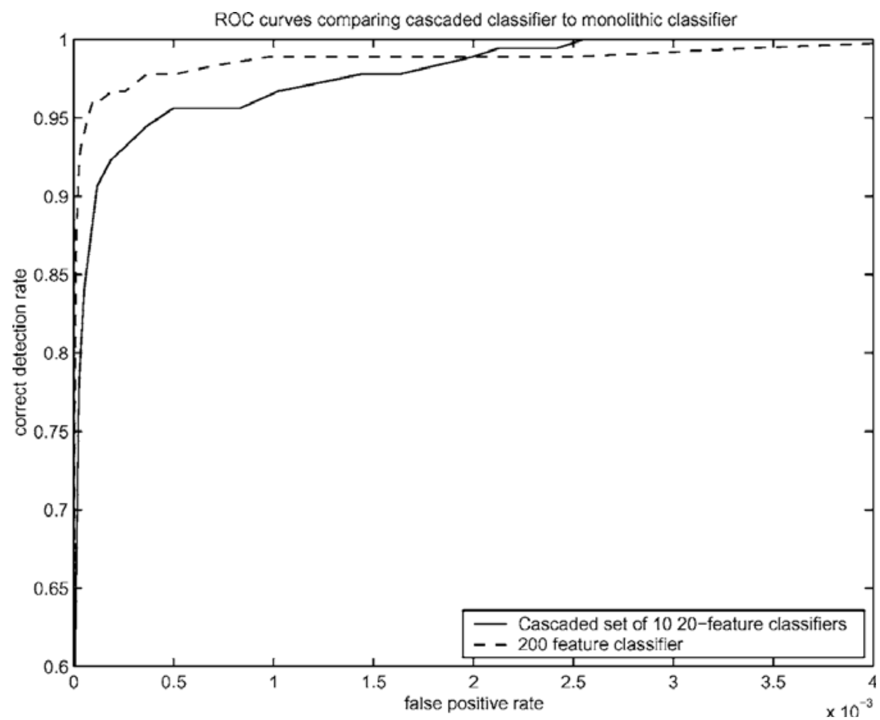
Viola, Jones, 2004

Cascade of AdaBoost Classifiers

- Problems with using a single classifier
  - Need many (~200) weak classifiers for good performance
  - Slow if applied at all locations and scales
- Cascade
  - Very few weak classifiers in first levels
  - Discards many windows quickly



Boosting Cascade

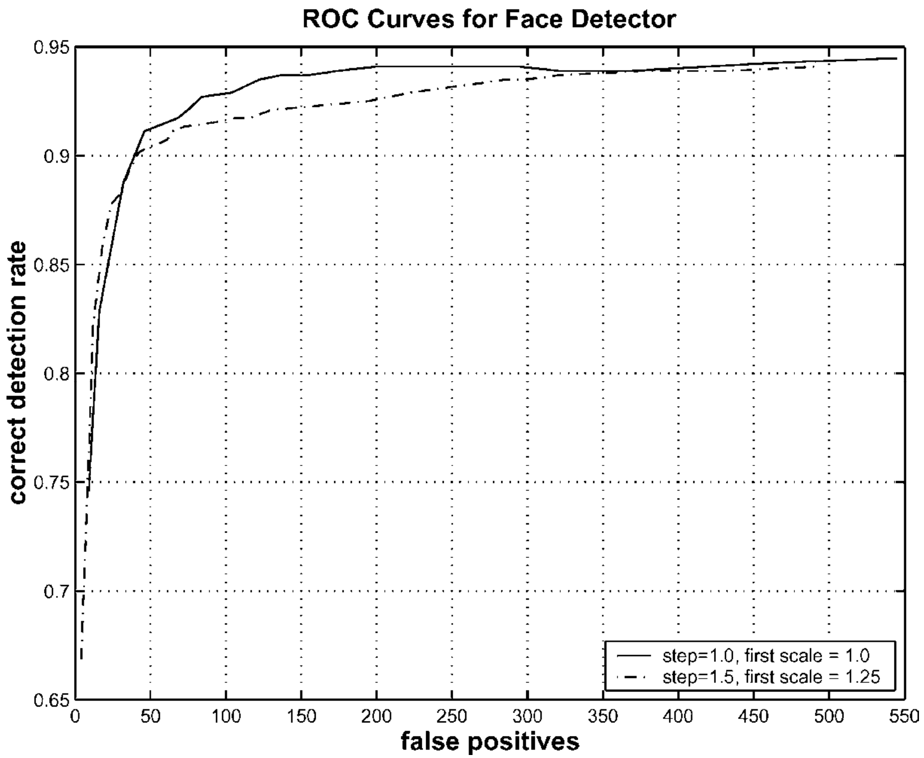


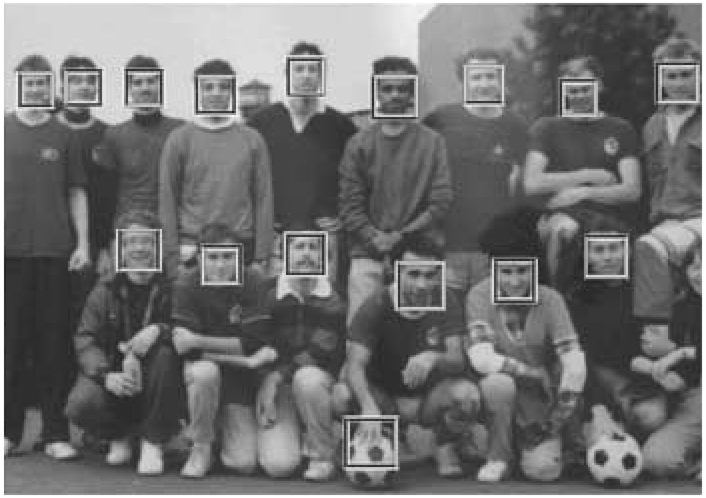
Training the Cascade

- 38 cascade Levels
- 5000 positive samples
  - Aligned and rescaled to 24x24 pixels
- 5000 negatives at each level
  - Random but have passed through all previous cascade levels
  - More difficult to differentiate from positives as level increases
- Level 1:
  - 2 weak classifiers
  - Detection ~100%, false alarm ~50%
- Level 2:
  - 10 weak classifiers
  - Detection ~100%, false alarm ~20%
- Level 3:
  - 25 weak classifiers
- …

Face Detection Results

- MIT+ CMU frontal face dataset
  - 130 images with 507 faces
  - Detection rate vs number of false positives





Boosting – Pros and Cons

- Pros:
  - Simple to implement
  - Can handle large datasets:
    - e.g. 40k samples x 100k features
  - Fast Learning
  - Feature selection → Fast Classification

- Cons:
  - Greedy feature selection might be suboptimal
  - For a small number of features, SVM is better

Training Error Bound

- Remember the training error at stage k is ϵ_k
- Let $\gamma_k = 0.5 - \epsilon_k$
- Then the overall training error is bounded by

$$Error \leq \exp(-2 \sum_{k=1}^{m} \gamma_k^2)$$

- If all $\epsilon_k < 0.5 - \epsilon$
- Then

$$Error \leq \exp(-2m\epsilon^2) \to 0$$

- Training error goes exponentially to zero

LogitBoost

- Logistic Loss

$$\lambda(y, h(\mathbf{x})) = -\log(1 + e^{-2yh(\mathbf{x})})$$

- Probability

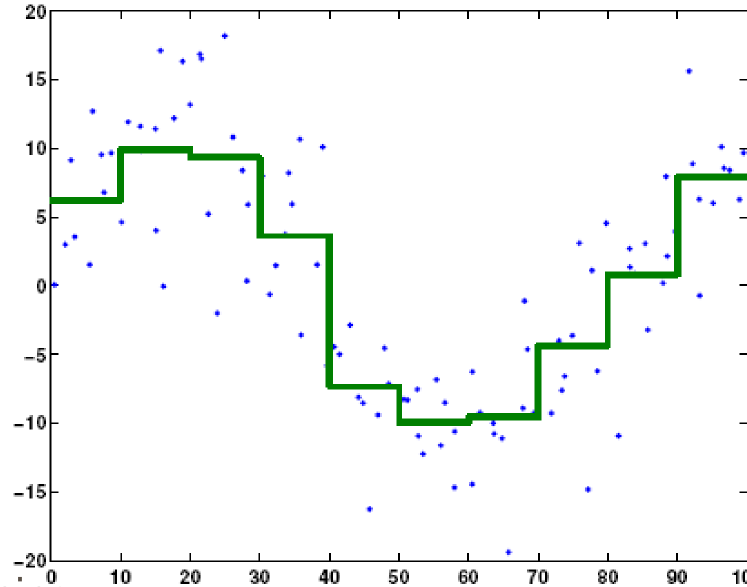
$$\underline{p(y = 1|\mathbf{x})} = \frac{e^{h(\mathbf{x})}}{e^{h(\mathbf{x})} + e^{-h(\mathbf{x})}} = \frac{1}{1 + e^{-2h(\mathbf{x})}}$$

- Like logistic regression but h is not linear

LogitBoost Training

- Start with N training samples $(\mathbf{x}_i, y_i)$
- Initialize $h(\mathbf{x}) = 0$
- Assign equal weights to all positives and to all negatives
- Repeat m times:
  1. Compute $p(\mathbf{x}_i) = \dfrac{e^{h(\mathbf{x}_i)}}{e^{h(\mathbf{x}_i)} + e^{-h(\mathbf{x}_i)}}$
  $$w_i = p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))$$
  $$z_i = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))}$$
  2. Find $h(\mathbf{x}, \theta_k)$ as the best weighted regression of z_i on x_i with weights w_i
  3. Update
  $$h(\mathbf{x}) \leftarrow h(\mathbf{x}) + \frac{1}{2} h(\mathbf{x}, \theta_k)$$

Finding $h(\mathbf{x}, \theta_k)$



- Find $h(\mathbf{x}, \theta_k)$ locally constant

$$\theta_k = (j, b_1, ..., b_n, v_1, ..., v_n)$$

$$h(\mathbf{x}, \theta_k) = \begin{cases} v_1 & \text{if } x_k < b_2 \\ \cdots & \\ v_i & \text{if } b_i \le x_k < b_{i+1} \\ \cdots & \\ v_n & \text{if } b_n \le x_k \end{cases}$$

- For each feature j
  - Divide the range of values of feature j into n bins
  - For each bin b, v_i is the weighted mean of z_j that fall into that bin

$$v_j = \frac{\sum_{i, x_i^j \in b} w_i z_i}{\sum_{i, x_i^j \in b} w_i}$$

- Find the j with smallest MSE
  - Better yet, find j that decreases loss most
- Could also use splines

Multi-Class LogitBoost Training (C classes)

- Start with N training samples $(\mathbf{x}_i, y_i)$
- Initialize $h_c(\mathbf{x}) = 0, c = 1, ..., C$
- Assign equal weights w_{ic} to all positives and to all negatives
- Repeat m times:

1. Compute
$$p_c(\mathbf{x}_i) = \frac{e^{h_c(\mathbf{x}_i)}}{\sum_{j=1}^{C} e^{h_j(\mathbf{x}_i)}}$$
$$w_{ic} = p_c(\mathbf{x}_i)(1 - p_c(\mathbf{x}_i))$$
$$z_{ic} = \frac{y_i - p_c(\mathbf{x}_i)}{p_c(\mathbf{x}_i)(1 - p_c(\mathbf{x}_i))}$$
2. Find $h(\mathbf{x}, \theta_{kc})$ as the best weighted regression of z_{ic} on x_i with weights w_{ic}
3. Change $h(\mathbf{x}, \theta_{kc}) \leftarrow \frac{C-1}{C}(h(\mathbf{x}, \theta_{kc}) - \frac{1}{C}\sum_{j=1}^{C} h(\mathbf{x}, \theta_{kj}))$
4. Update
$$h_c(\mathbf{x}) \leftarrow h_c(\mathbf{x}) + \frac{1}{2}h(\mathbf{x}, \theta_{kc})$$

Multi-Class LogitBoost Classification

- Observe

$$\sum_{c=1}^{C} h_c(\mathbf{x}) = 0, \ \forall \mathbf{x}$$

- Given a feature vector $\mathbf{x}$
  - Compute all $h_c(\mathbf{x}), c = 1, ..., C$
  - Obtain

$$\text{classify}(\mathbf{x}) = \arg\max_c (h_c(\mathbf{x}))$$

Real AdaBoost

- Start with N training samples $(\mathbf{x}_i, y_i)$
- Assign equal weights to all positives and to all negatives

$$W_i^0 = \frac{0.5}{|\{j, y_j = y_i\}|}$$

- Repeat m times:
  1. Find the class probability estimate on x_i with weights w_i
     $$p(\mathbf{x}, \theta_k) = P_W(y = 1|\mathbf{x}) \in [0, 1]$$
  2. Set
     $$h(\mathbf{x}, \theta_k) = \frac{1}{2} \log \frac{p(\mathbf{x}, \theta_k)}{1 - p(\mathbf{x}, \theta_k)}$$
  3. Update the classifier
     $$h(\mathbf{x}) \leftarrow h(\mathbf{x}) + h(\mathbf{x}, \theta_k)$$
  4. Update the weights
     $$W_i^k = W_i^{k-1} e^{-y_i h(\mathbf{x}_i, \theta_k)}$$
     and re-normalize them

Gentle AdaBoost

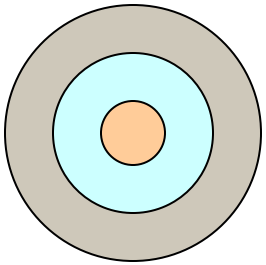
- Start with N training samples $(\mathbf{x}_i, y_i)$
- Assign equal weights to all positives and to all negatives

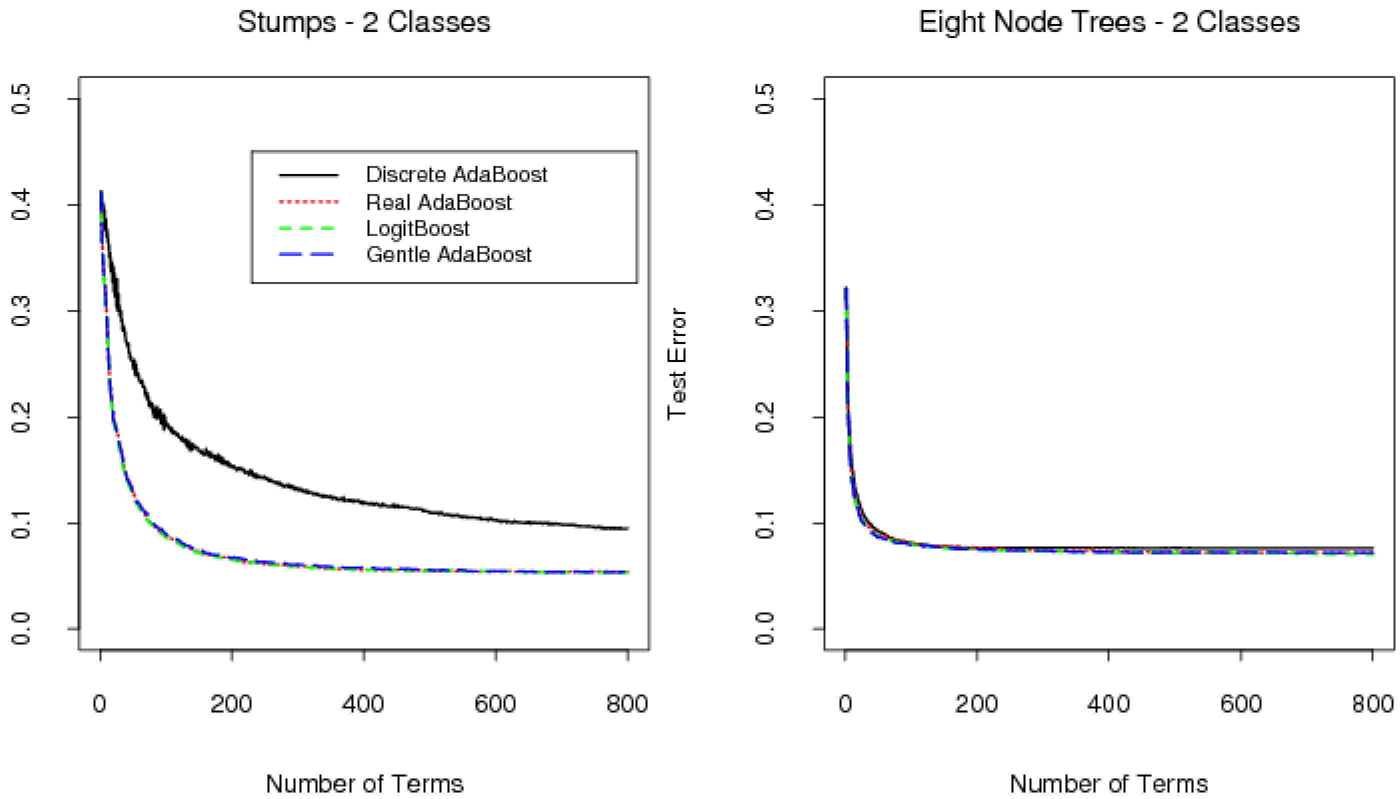
$$W_i^0 = \frac{0.5}{|\{j, y_j = y_i\}|}$$

- Repeat m times:
  1. Find $h(\mathbf{x}, \theta_k)$ as the best weighted regression of y_i on x_i with weights w_i
  2. Update the classifier
  $$h(\mathbf{x}) \leftarrow h(\mathbf{x}) + \frac{1}{2} h(\mathbf{x}, \theta_k)$$
  3. Update the weights
  $$W_i^k = W_i^{k-1} e^{-y_i h(\mathbf{x}_i, \theta_k)}$$
  and re-normalize them

Boosting Comparison

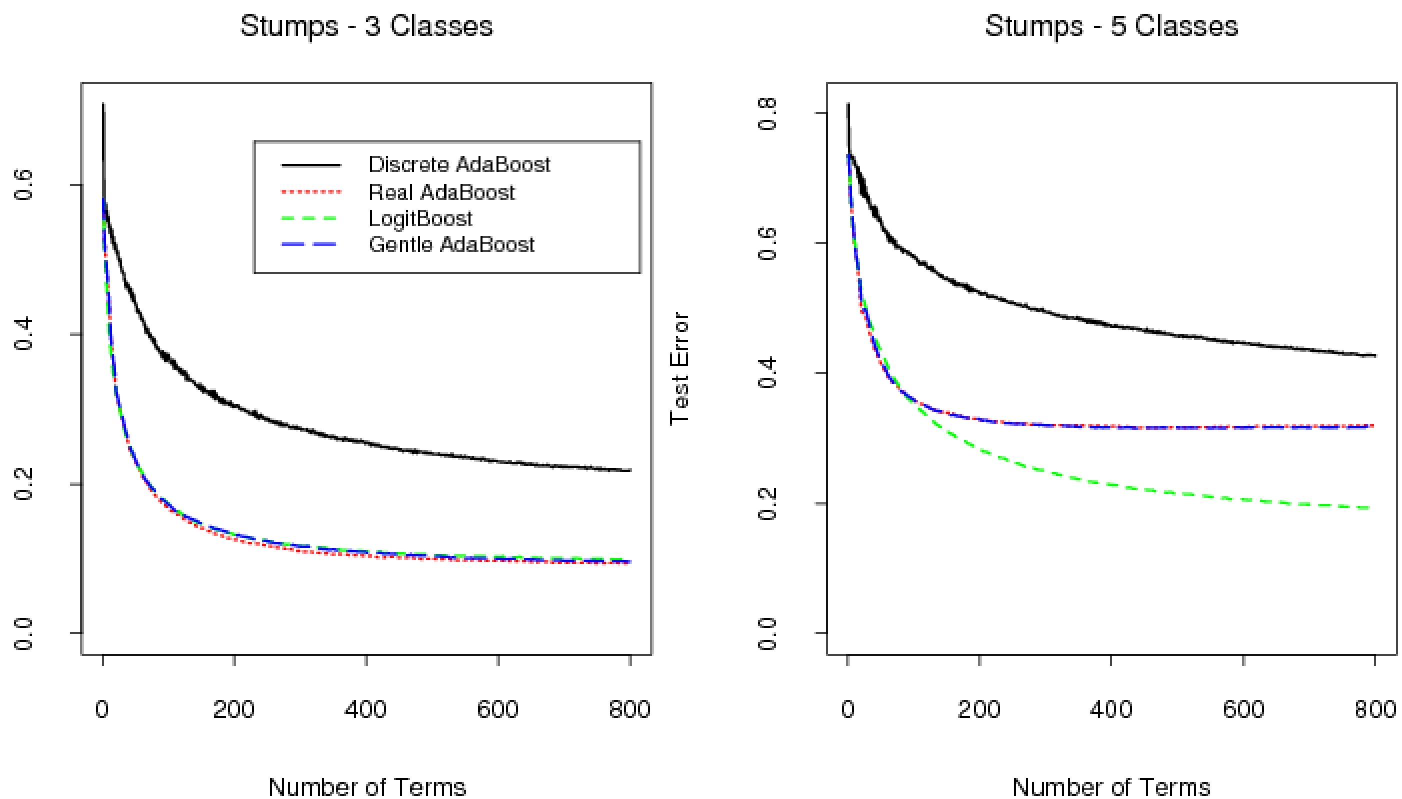
- Dataset:
  - Classes are between concentric hyper-spheres in 10D space
  - Additive boundary $\sum_i f_i(\mathbf{x}) = b$





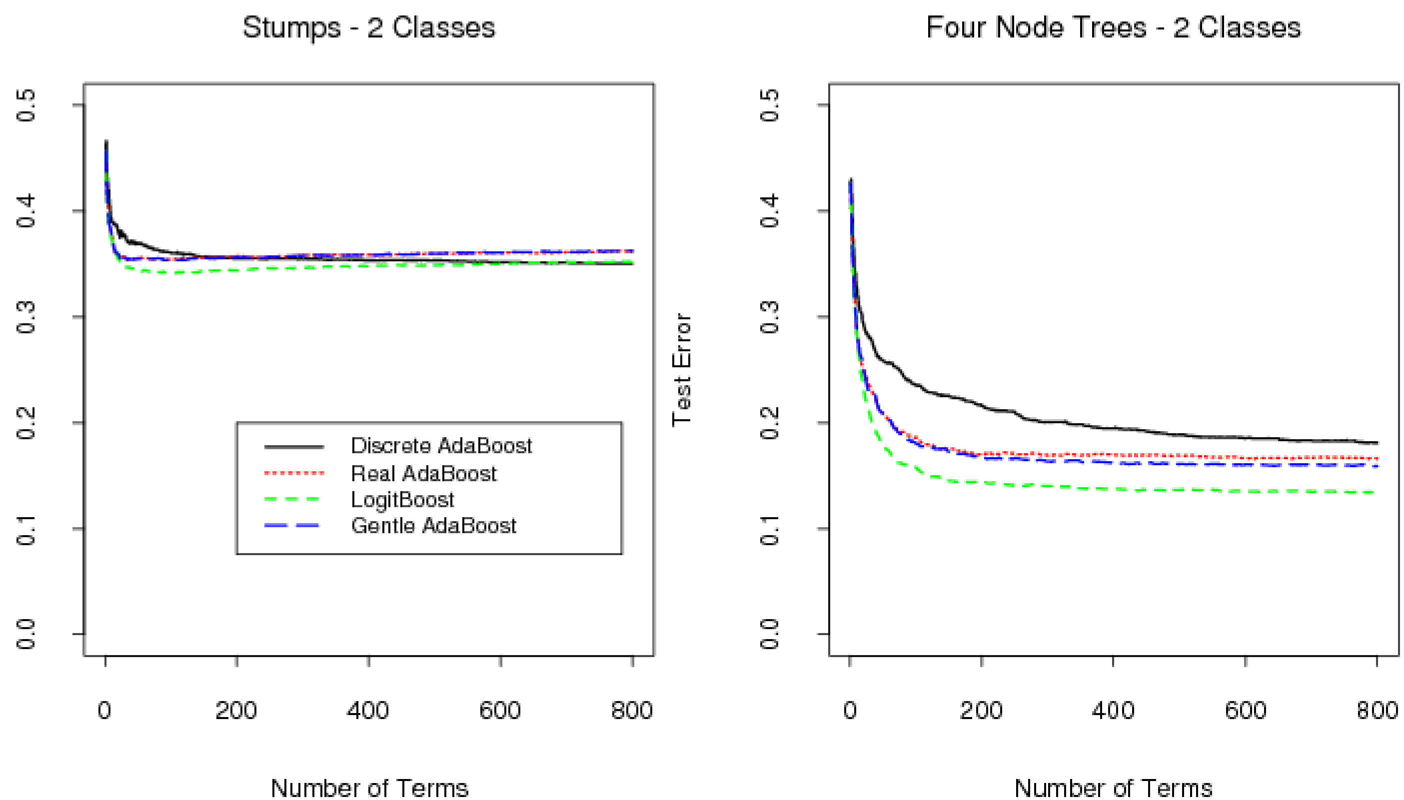
Friedman et al, Ann Stat 2000

Boosting Comparison



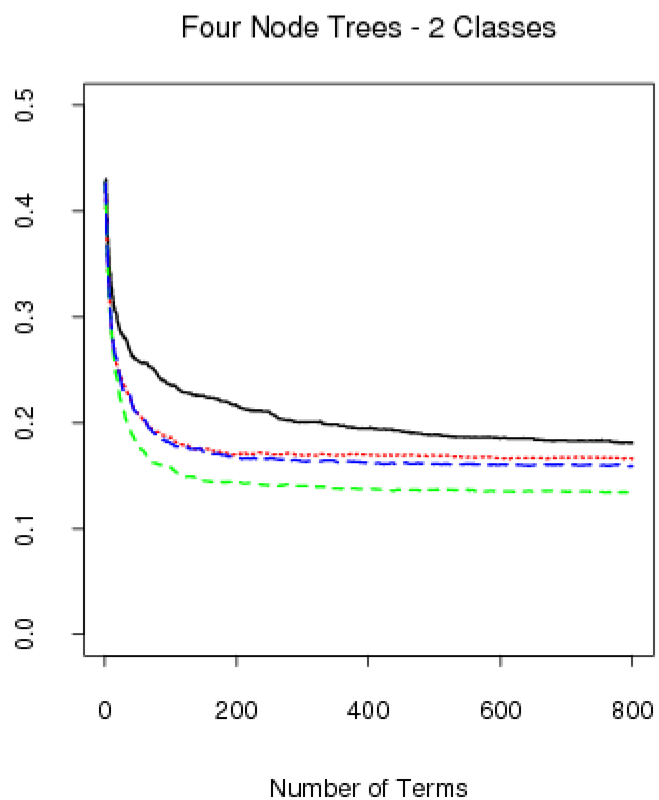
Boosting Comparison

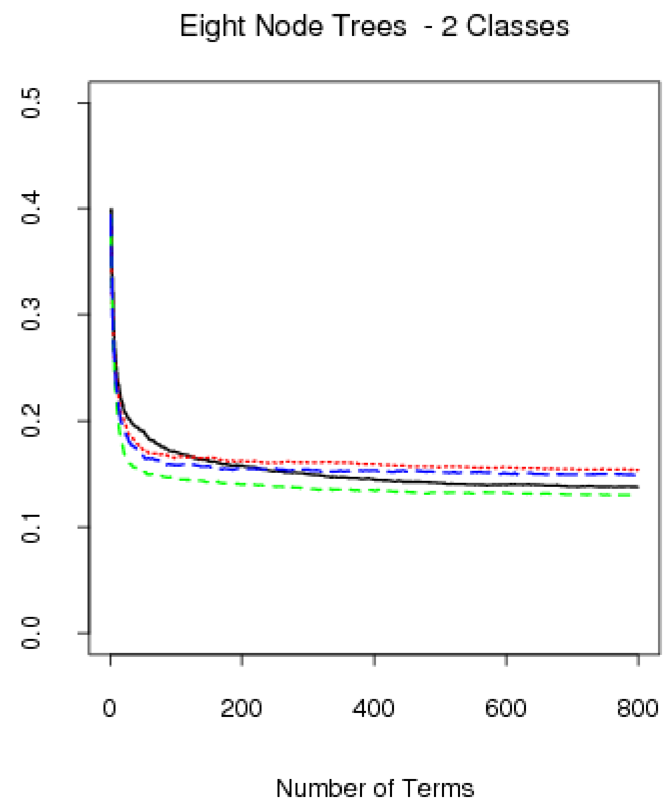
- Complex boundary with overlapping classes
  - Non-additive boundary $\log\left(\frac{\Pr[y=1\,|\,x]}{\Pr[y=-1\,|\,x]}\right) = 10\sum_{j=1}^{6} x_j\left(1+\sum_{l=1}^{6}(-1)^l x_l\right)$
- Bayes error 0.046



Boosting Comparison

- Trees are better than stumps when the boundary is not “additive”
- They can capture some feature interactions





Real-Data Comparison

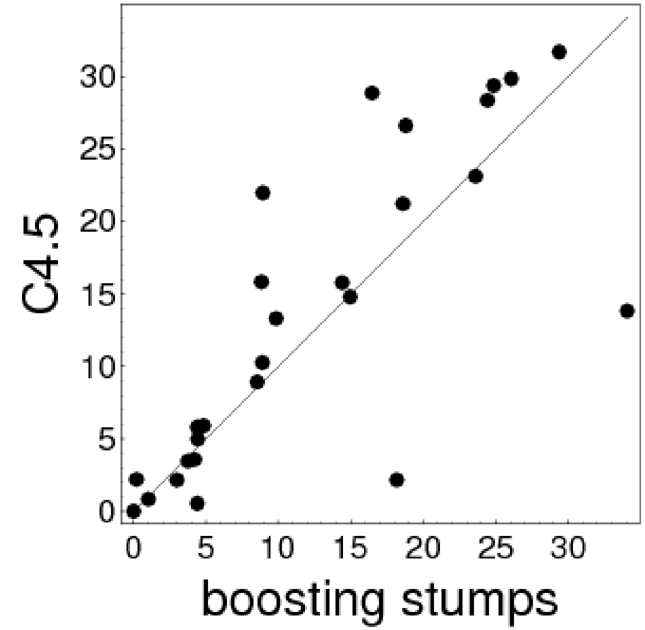
Method	Terminal Nodes	Iterations 20	50	100	200	Fraction
Satimage	CART error = .148					
LogitBoost	2	.140	.120	.112	.102	
Real AdaBoost	2	.148	.126	.117	.119	
Gentle AdaBoost	2	.148	.129	.119	.119	
Discrete AdaBoost	2	.174	.156	.140	.128	
LogitBoost	8	.096	.095	.092	.088	
Real AdaBoost	8	.105	.102	.092	.091	
Gentle AdaBoost	8	.106	.103	.095	.089	
Discrete AdaBoost	8	.122	.107	.100	.099	
Letter	CART error = .124					
LogitBoost	2	.250	.182	.159	.145	.06
Real AdaBoost	2	.244	.181	.160	.150	.12
Gentle AdaBoost	2	.246	.187	.157	.145	.14
Discrete AdaBoost	2	.310	.226	.196	.185	.18
LogitBoost	8	.075	.047	.036	.033	.03
Real AdaBoost	8	.068	.041	.033	.032	.03
Gentle AdaBoost	8	.068	.040	.030	.028	.03
Discrete AdaBoost	8	.080	.045	.035	.029	.03

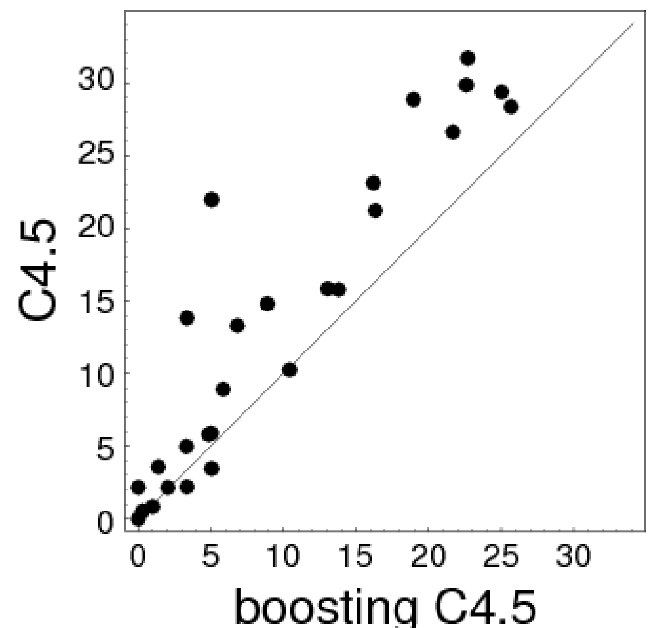
- CART=Decision Tree

Real-Data Comparison

- 27 Benchmark problems
- Test error C4.5 vs. test error boosting

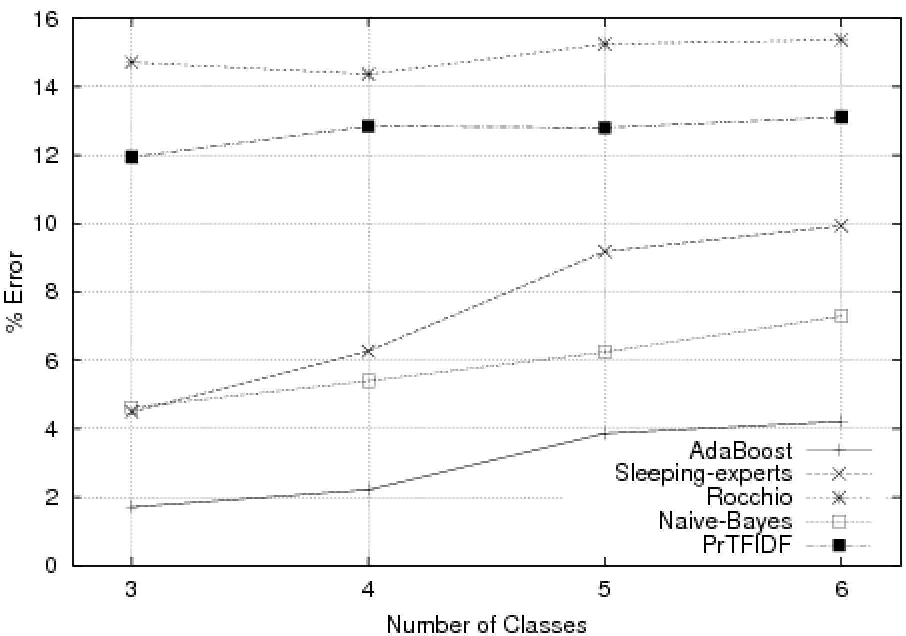
name	# examples train	# examples test	# classes	# attributes disc.	# attributes cont.	missing values
soybean-small	47	-	4	35	-	-
labor	57	-	2	8	8	×
promoters	106	-	2	57	-	-
iris	150	-	3	-	4	-
hepatitis	155	-	2	13	6	×
sonar	208	-	2	-	60	-
glass	214	-	7	-	9	-
audiology.stand	226	-	24	69	-	×
cleve	303	-	2	7	6	×
soybean-large	307	376	19	35	-	×
ionosphere	351	-	2	-	34	-
house-votes-84	435	-	2	16	-	×
votes1	435	-	2	15	-	×
crx	690	-	2	9	6	×
breast-cancer-w	699	-	2	-	9	×
pima-indians-di	768	-	2	-	8	-
vehicle	846	-	4	-	18	-
vowel	528	462	11	-	10	-
german	1000	-	2	13	7	-
segmentation	2310	-	7	-	19	-
hypothyroid	3163	-	2	18	7	×
sick-euthyroid	3163	-	2	18	7	×
splice	3190	-	3	60	-	-
kr-vs-kp	3196	-	2	36	-	-
satimage	4435	2000	6	-	36	-
agaricus-lepiot	8124	-	2	22	-	-
letter-recognit	16000	4000	26	-	16	-

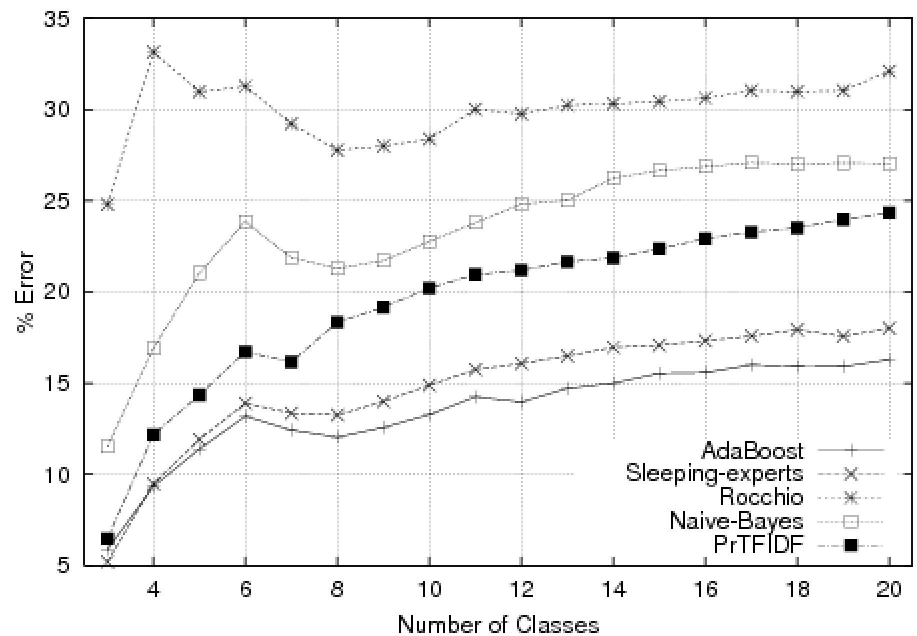




Text Categorization

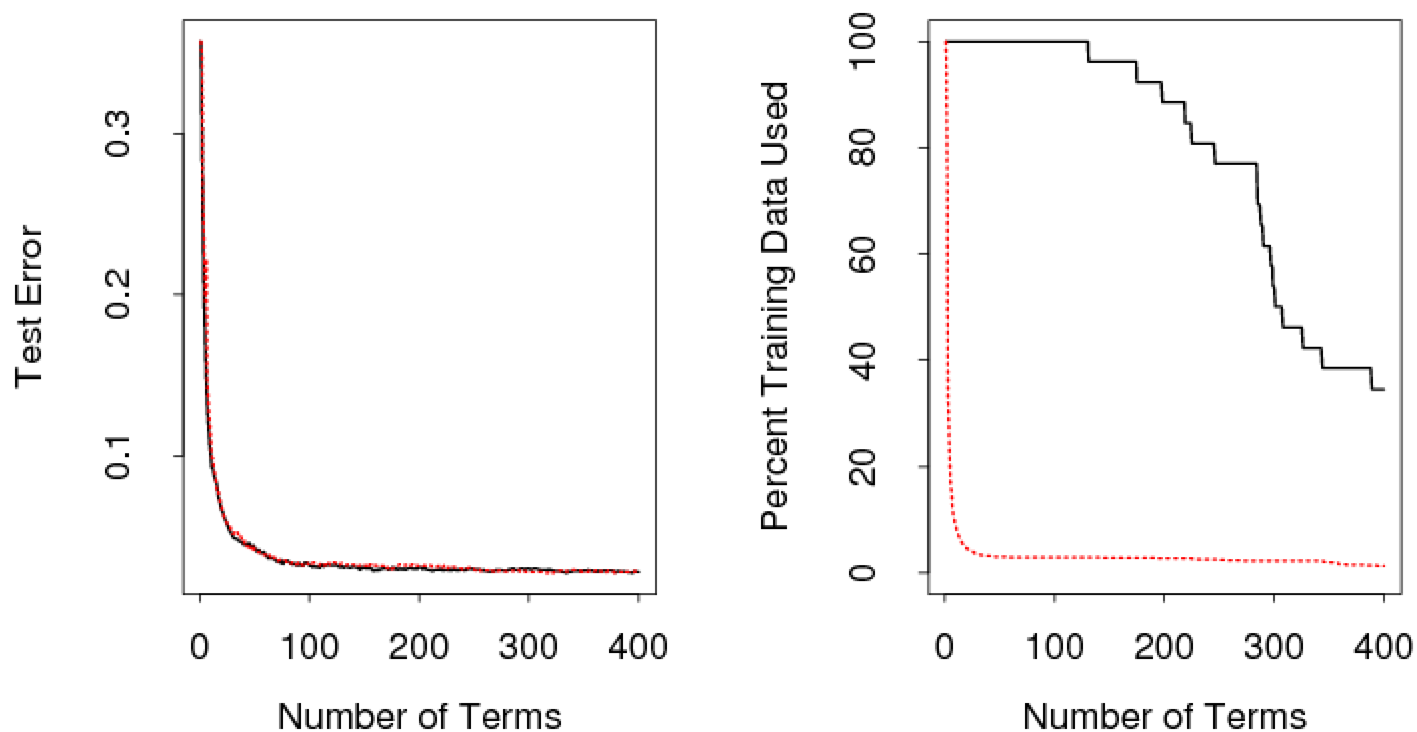
- 2 Text Datasets:
  - Reuters newswire articles
  - AP newswire articles





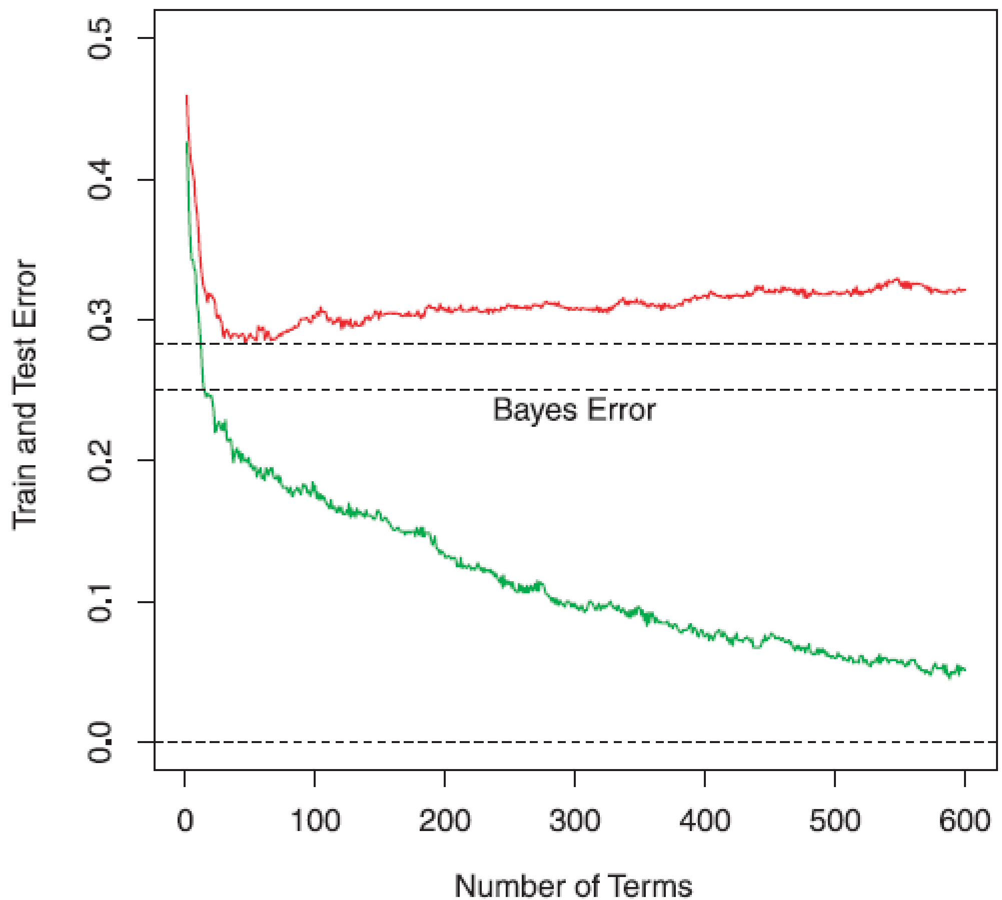
Weight Trimming

- Most weights decrease after a number of boosting iterations
- Idea:
  - Ignore samples with very small weight when training
  - Recompute all weights each time
  - Use LogitBoost



Overfitting

- 2 Isotropic 10D Gaussians
  - same mean, different variances
- Bayes error 0.25



Conclusion

- Boosting is a powerful method for classification
- Advantages:
  - Easy to implement
  - Fast to train
  - Can handle large training sets
  - Offers feature selection
  - Hard to overfit
  - Fast classification when using a cascaded approach

References

- Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197– 227, 1990
- Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995
- L Breiman - Bagging Predictors - *Machine Learning*, 1996
- J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 725–730. 1996
- J Friedman, T Hastie, R Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 2000
- Robert E. Schapire. The Boosting Approach to Machine Learning, *MSRI Workshop on Nonlinear Estimation and Classification*, 2002
- P Viola, MJ Jones. Robust Real-Time Face Detection - *International Journal of Computer Vision*, 2004