Advanced Machine Learning & Perception

Instructor: Tony Jebara

Boosting

- Combining Multiple Classifiers
- Voting
- Boosting
- Adaboost

- Based on material by Y. Freund, P. Long & R. Schapire

Combining Multiple Learners

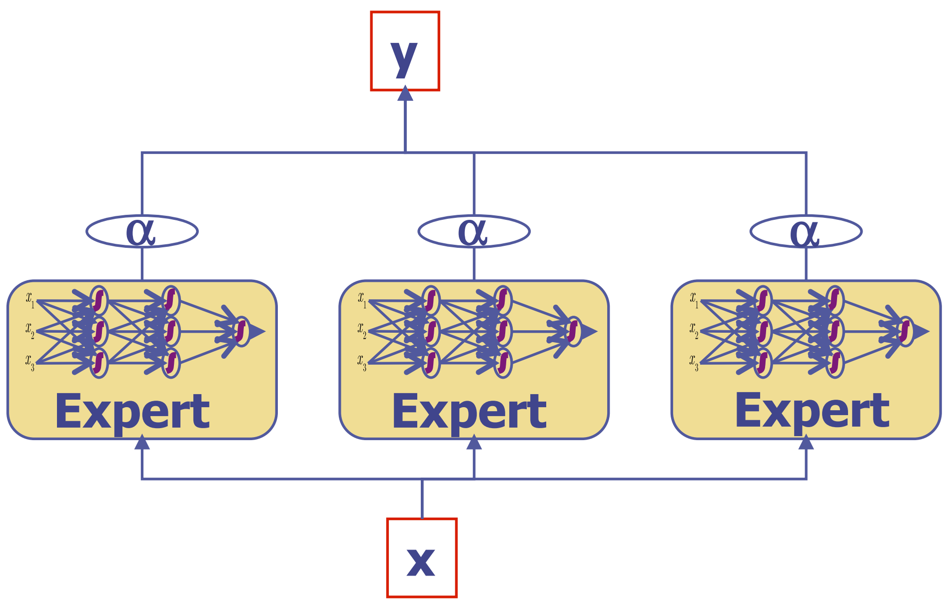
- Have many simple learners
- Also called base learners or weak learners which have a classification error of <0.5
- Combine or vote them to get a higher accuracy
- No free lunch: there is no guaranteed best approach here
- Different approaches:
  - Voting
    - combine learners with fixed weight
  - Mixture of Experts
    - adjust learners and a variable weight/gate fn
  - Boosting
    - actively search for next base-learners and vote
  - Cascading, Stacking, Bagging, etc.

Voting

- Have T classifiers $h_t\left(x\right)$
- Average their prediction with weights

$$f\left(x\right) = \sum_{t=1}^{T} \alpha_t h_t\left(x\right) \; where \, \alpha_t \geq 0 \, and \sum_{t=1}^{T} \alpha_t = 1$$

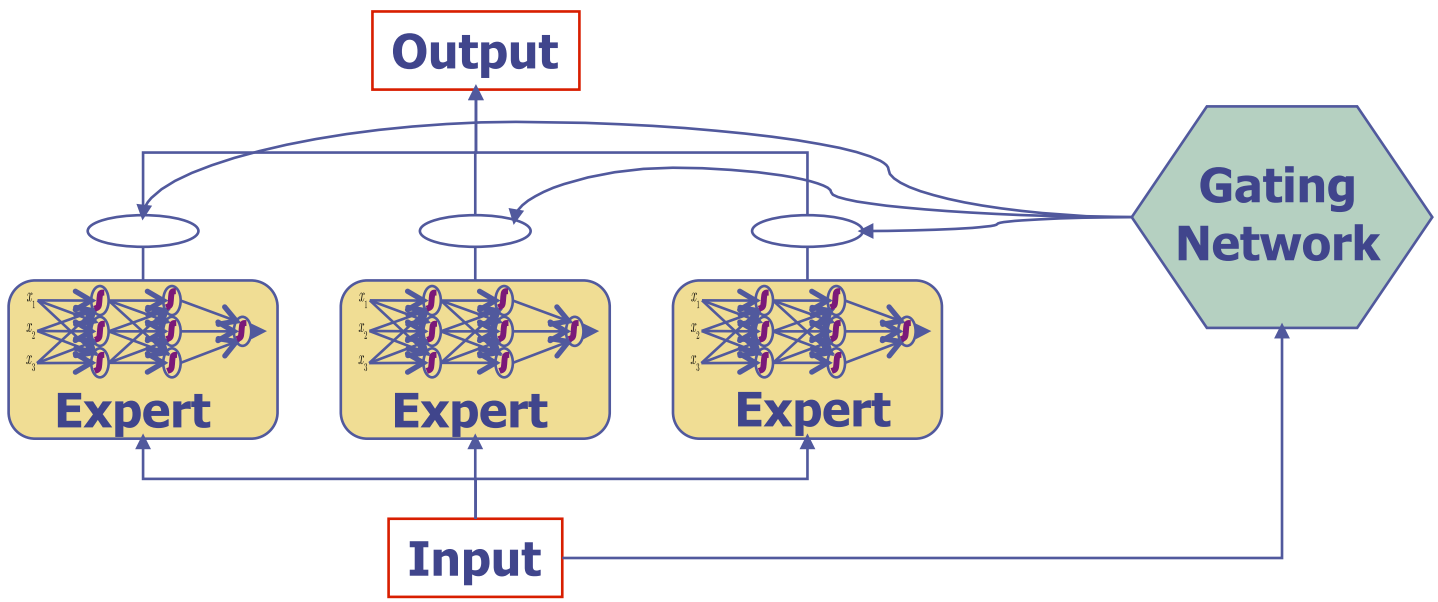
- Like mixture of experts but weight is constant with input



Mixture of Experts

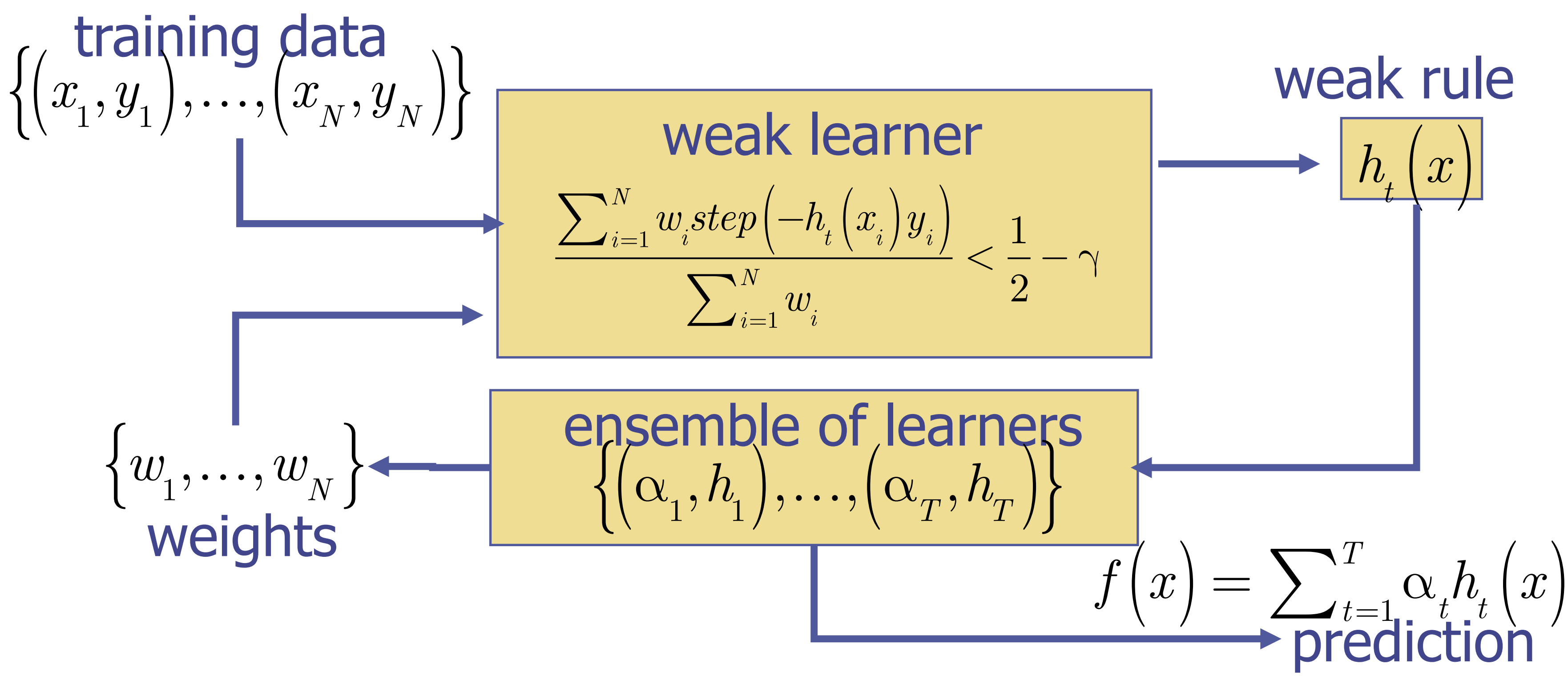
- Have T classifiers or experts $h_t(x)$ and a gating fn $\alpha_t(x)$
- Average their prediction with variable weights
- But, adapt parameters of the gating function and the experts (fixed total number T of experts)

$$f(x) = \sum_{t=1}^{T} \alpha_t(x) h_t(x) \; where \, \alpha_t(x) \geq 0 \, and \sum_{t=1}^{T} \alpha_t(x) = 1$$



Boosting

- Actively find complementary or synergistic weak learners
- Train next learner based on mistakes of previous ones.
- Average prediction with fixed weights
- Find next learner by training on weighted versions of data.

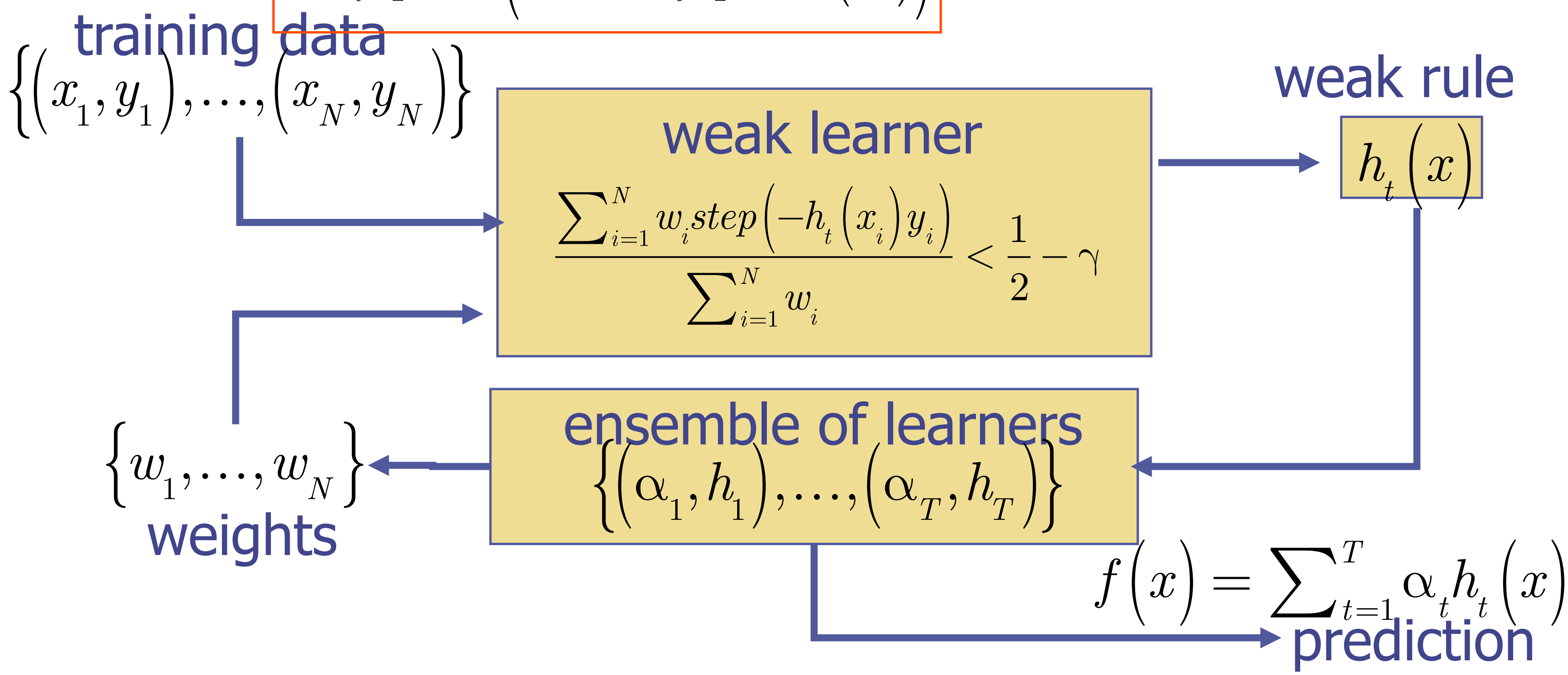


AdaBoost

- Most popular weighting scheme
- Define margin for point i as $y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)$
- Find an h_t and find weight α_t to min the cost function

$$\boxed{\sum_{i=1}^{N} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right)}$$

sum exp-margins



AdaBoost

- Choose base learner & α_t:

$$\boxed{\min_{\alpha_t, h_t} \sum_{i=1}^{N} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t\left(x_i\right)\right)}$$

- Recall error of base classifier h_t must be

$$\varepsilon_t = \frac{\sum_{i=1}^{N} w_i step\left(-h_t\left(x_i\right) y_i\right)}{\sum_{i=1}^{N} w_i} < \frac{1}{2} - \gamma$$

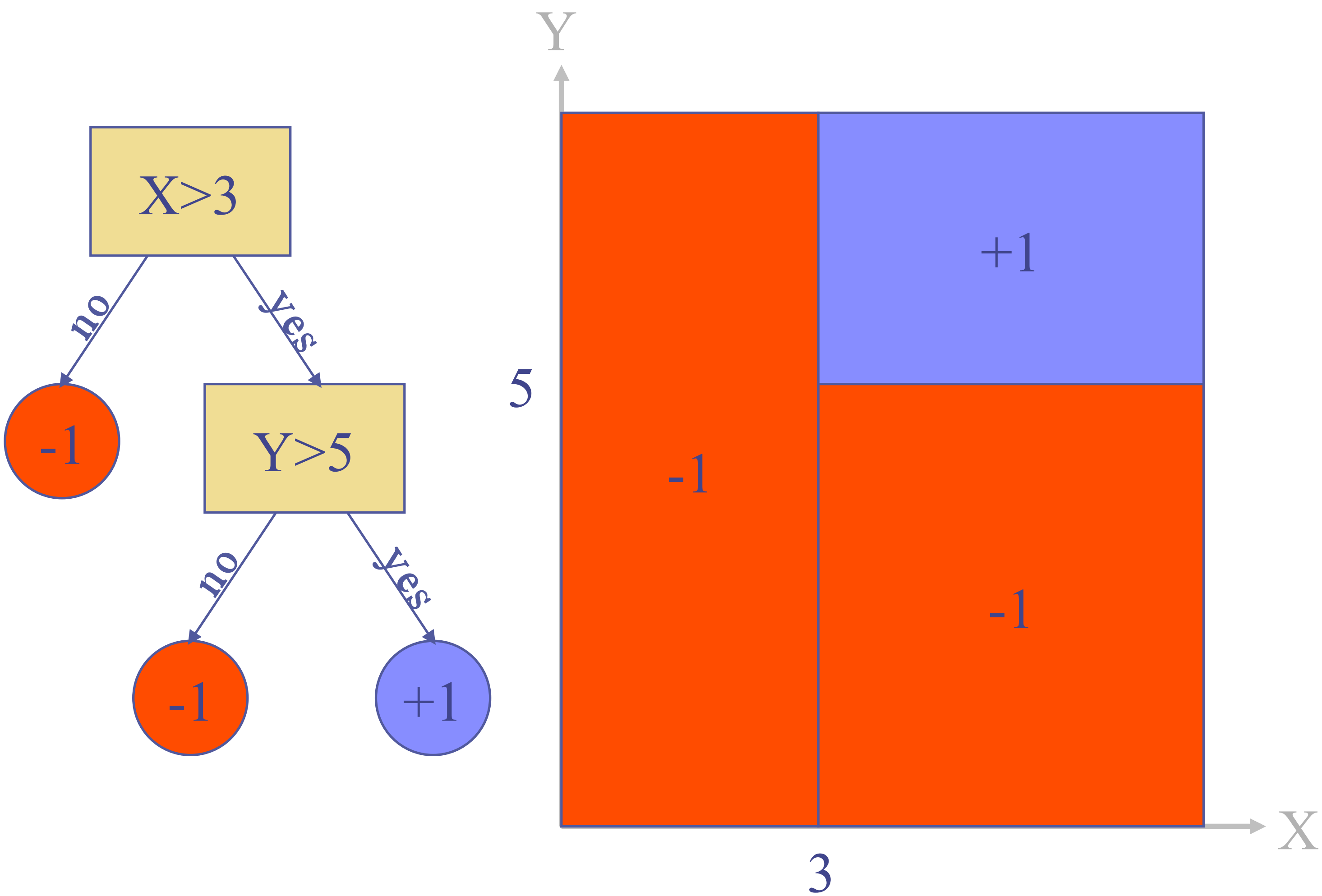
- For <u>binary</u> h, Adaboost puts this weight on weak learners: (instead of the more general rule)

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

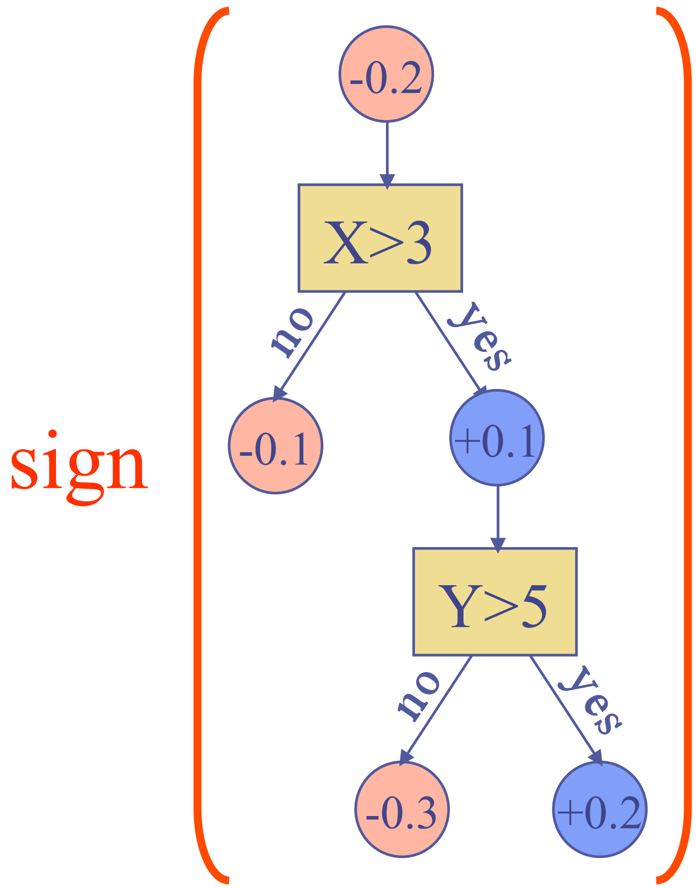
- Adaboost picks the following for the weights on data for the next round (here Z is the normalizer to sum to 1)

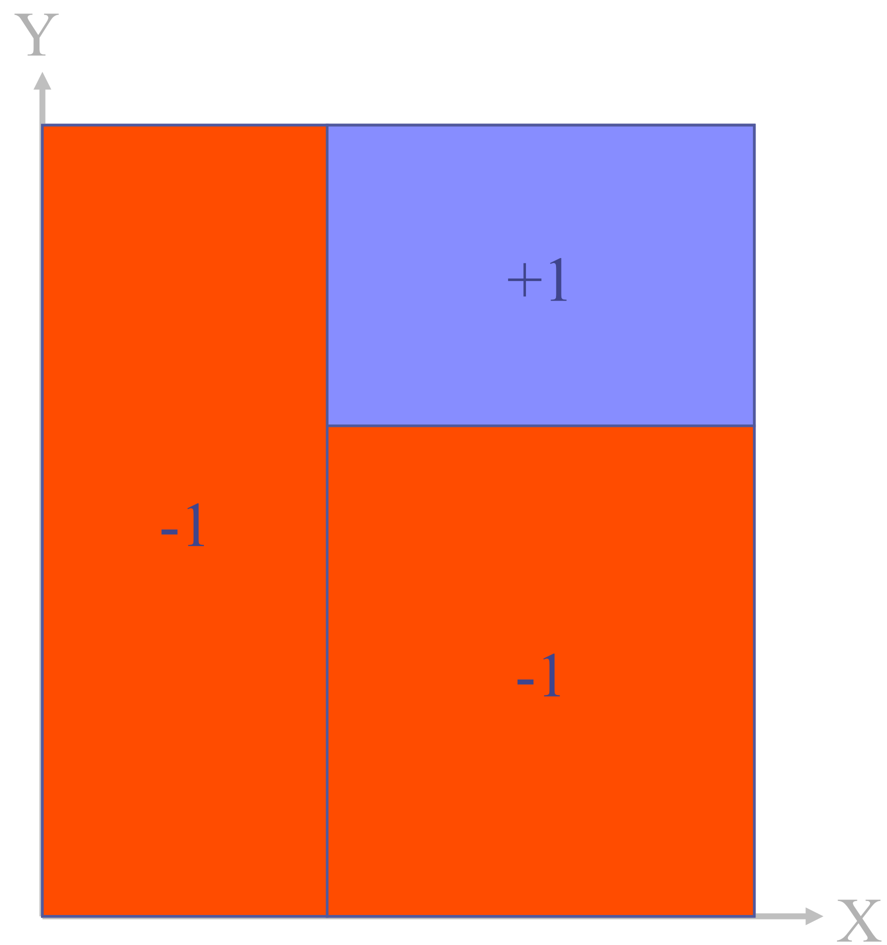
$$\boxed{w_i^{t+1} = \frac{w_i^t \exp\left(-\alpha_t y_i h_t\left(x_i\right)\right)}{Z_t}}$$

Decision Trees

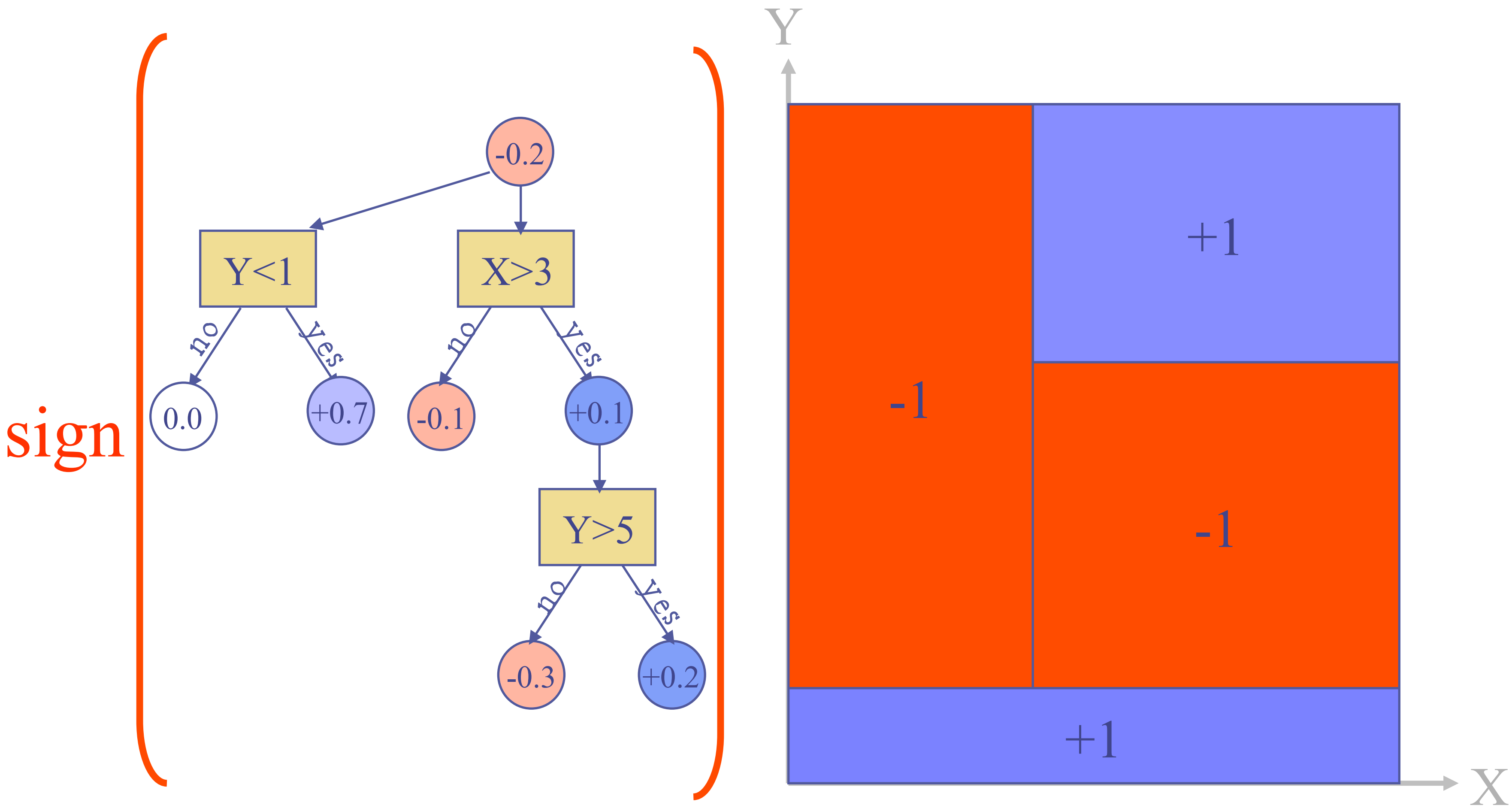


Decision tree as a sum





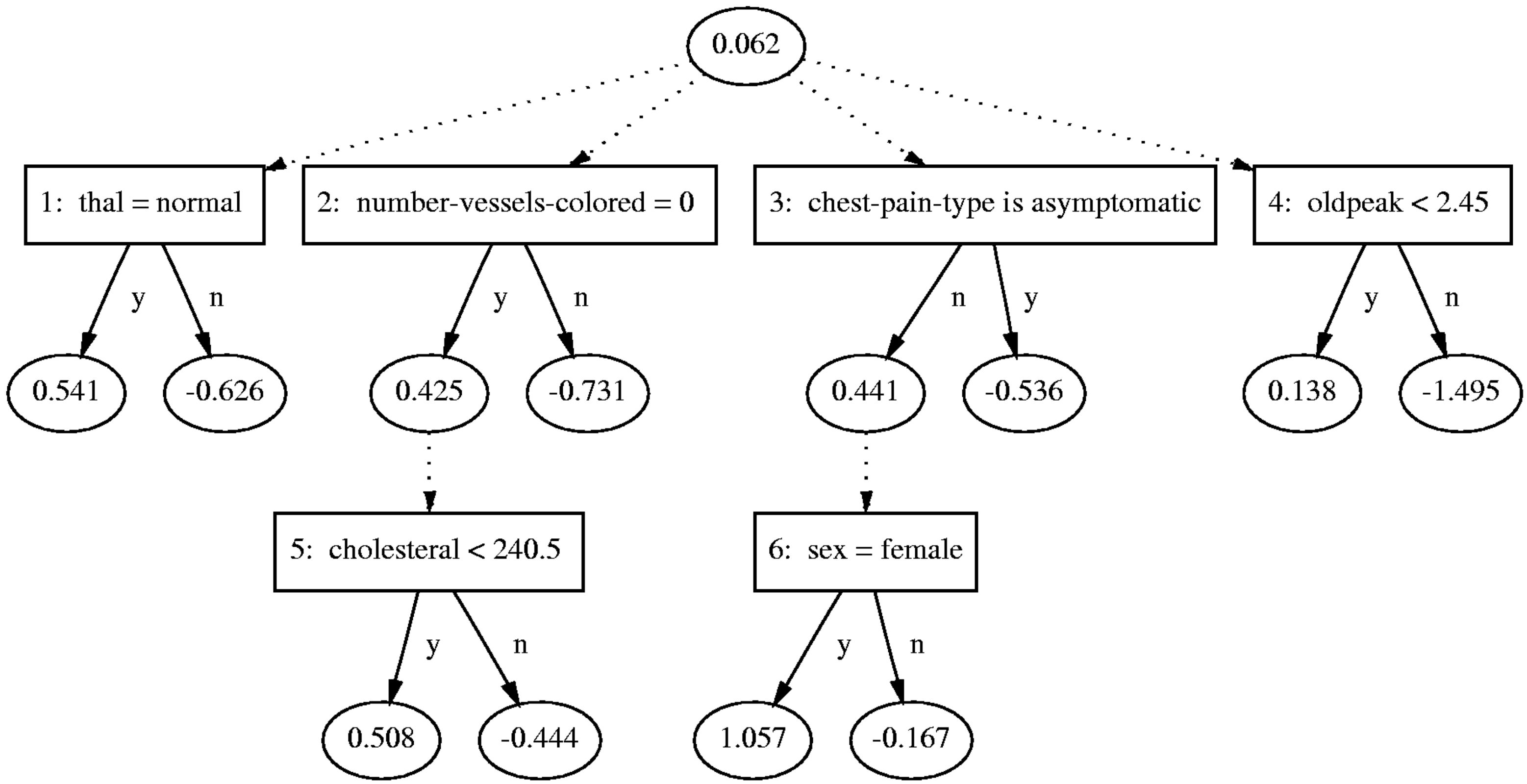
An alternating decision tree



Example: Medical Diagnostics

- `Cleve` dataset from UC Irvine database.
- Heart disease diagnostics (+1=healthy,-1=sick)
- 13 features from tests (real valued and discrete).
- 303 instances.

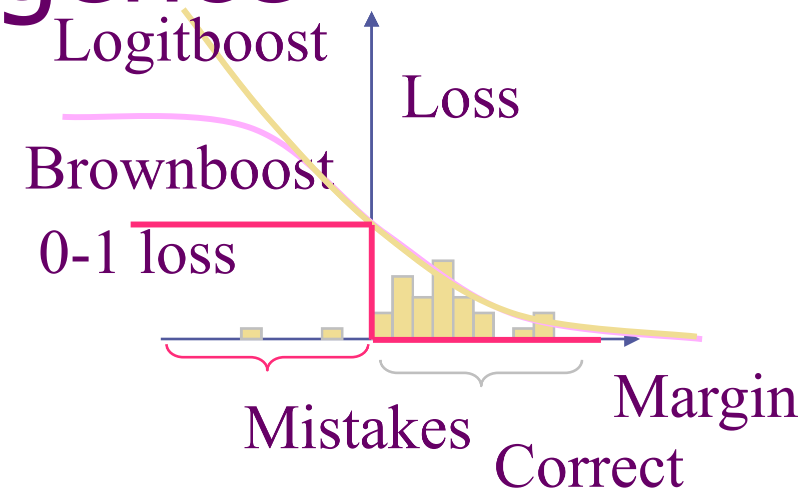
Ad-Tree Example



Cross-validated accuracy

Learning algorithm	Number of splits	Average test error	Test error variance
ADtree	6	17.0%	0.6%
C5.0	27	27.2%	0.5%
C5.0 + boosting	446	20.2%	0.5%
Boost Stumps	16	16.5%	0.8%

AdaBoost Convergence



- Rationale?
- Consider bound on the training error:

$$
\begin{aligned}
R_{emp} &= \frac{1}{N}\sum_{i=1}^{N} step\left(-y_i f\left(x_i\right)\right) \\
&\leq \frac{1}{N}\sum_{i=1}^{N} \exp\left(-y_i f\left(x_i\right)\right) && \textbf{exp bound on step} \\
&= \sum_{i=1}^{N} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t\left(x_i\right)\right) && \textbf{definition of f(x)} \\
&= \prod_{t=1}^{T} Z_t && \textbf{recursive use of Z}
\end{aligned}
$$

- Adaboost is essentially doing gradient descent on this.
- Convergence?

AdaBoost Convergence

- Convergence? Consider the binary h_t case.

$$R_{emp} \le \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} \sum_{i=1}^{N} w_i^t \exp\left(-\alpha_t y_i h_t\left(x_i\right)\right)$$

$$= \prod_{t=1}^{T} \sum_{i=1}^{N} w_i^t \exp\left(\ln\left(\left(\frac{\varepsilon_t}{1-\varepsilon_t}\right)^{\frac{1}{2} y_i h_t\left(x_i\right)}\right)\right)$$

$$= \prod_{t=1}^{T} \sum_{i=1}^{N} w_i^t \left(\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}\right)^{y_i h_t\left(x_i\right)}$$

$$= \prod_{t=1}^{T} \left(\sum_{correct} w_i^t \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} + \sum_{incorrect} w_i^t \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}\right)$$

$$= \prod_{t=1}^{T} 2\sqrt{\varepsilon_t\left(1-\varepsilon_t\right)} = \prod_{t=1}^{T} \sqrt{\left(1-4\gamma_t^2\right)} \le \exp\left(-2\sum_t \gamma_t^2\right)$$

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

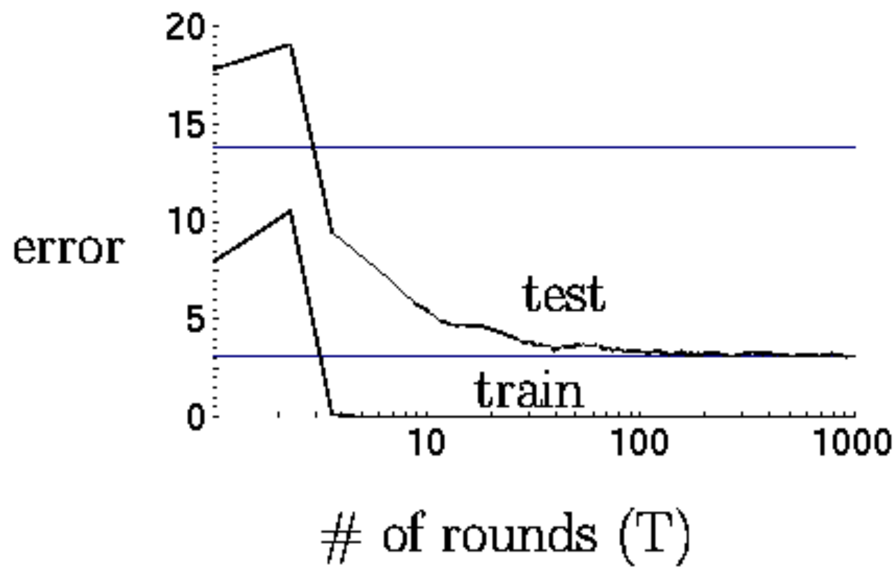
$$\varepsilon_t = \tfrac{1}{2} - \gamma_t \le \tfrac{1}{2} - \gamma$$

$$R_{emp} \le \exp\left(-2\sum_t \gamma_t^2\right) \le \exp\left(-2T\gamma^2\right)$$

So, the final learner converges exponentially fast in T if each weak learner is at least better than gamma!

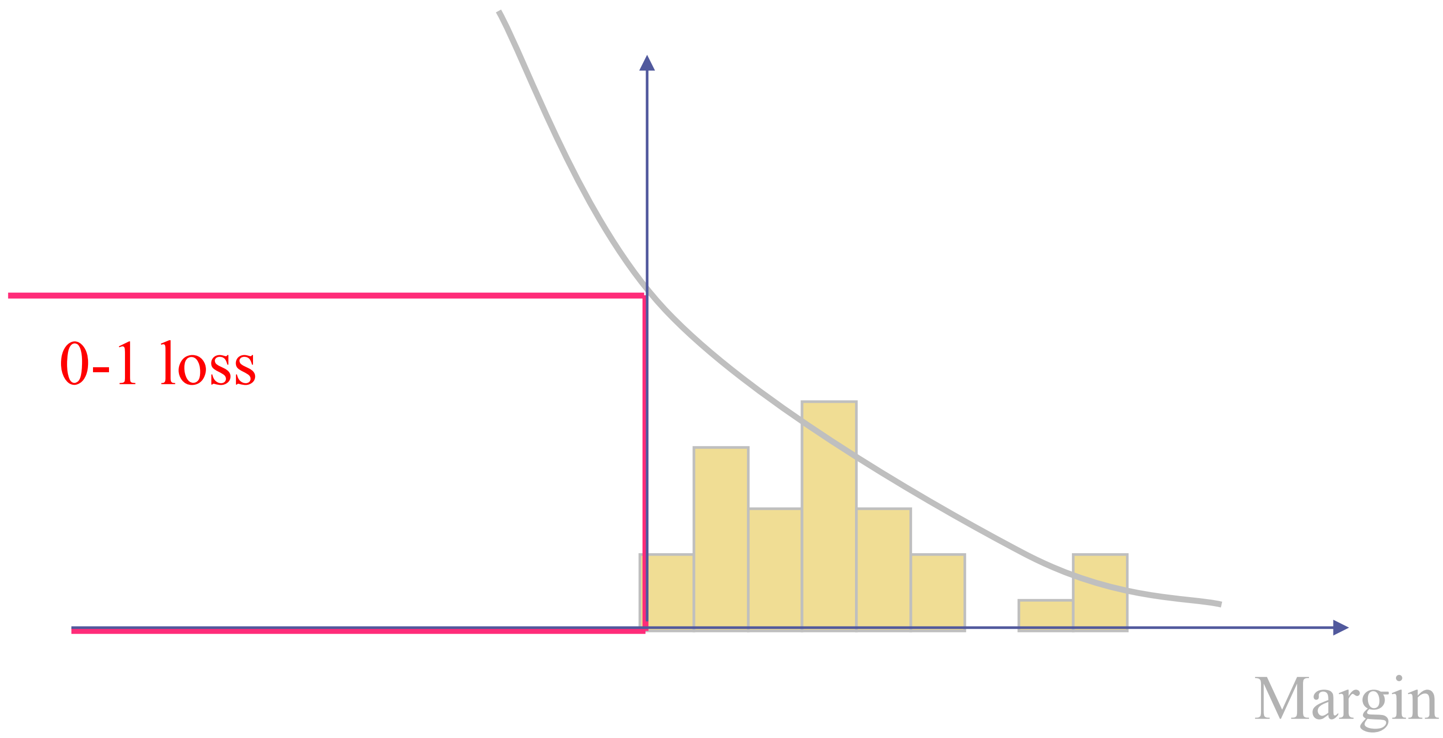
Curious phenomenon

Boosting decision trees

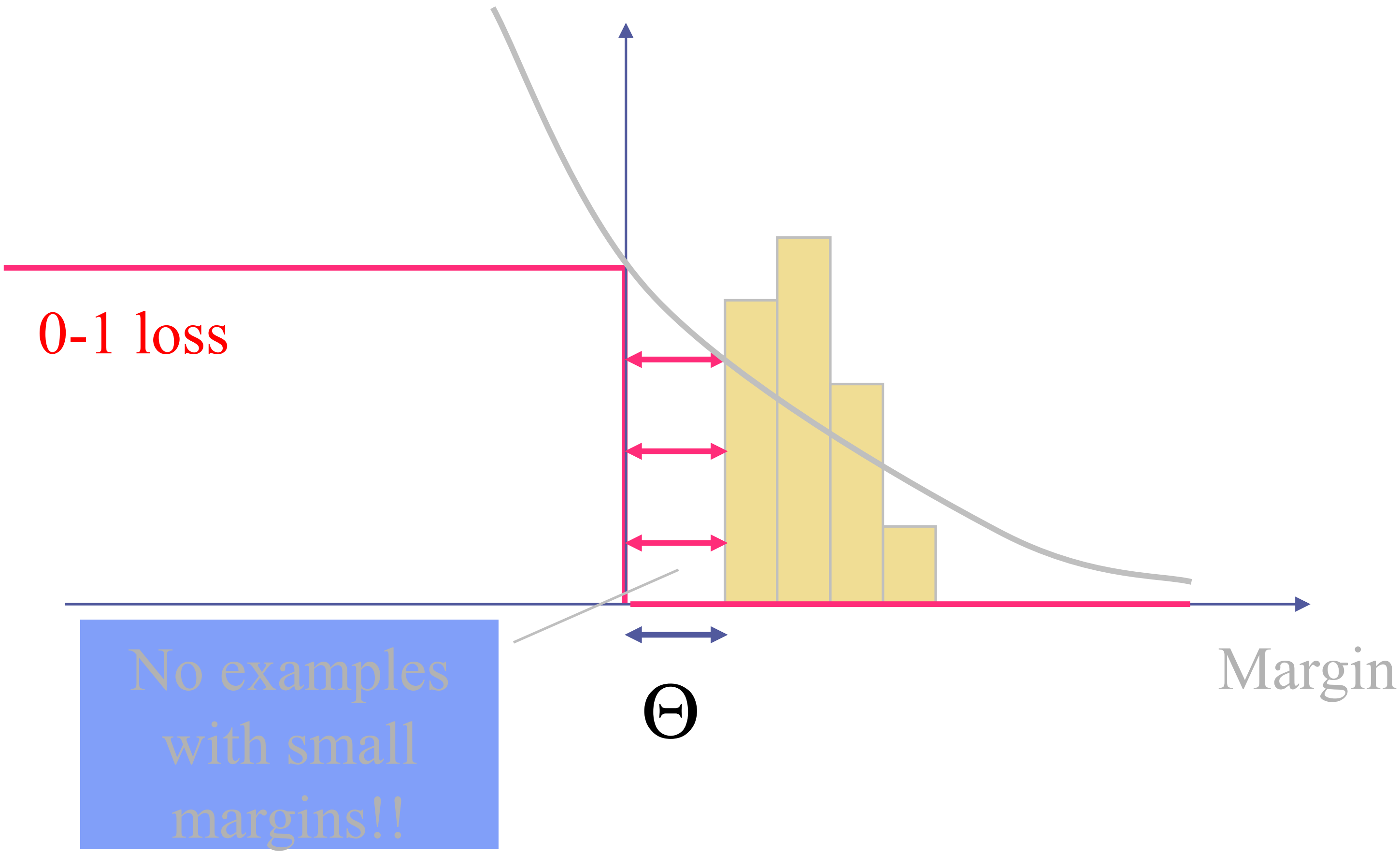


Using <10,000 training examples we fit >2,000,000 parameters

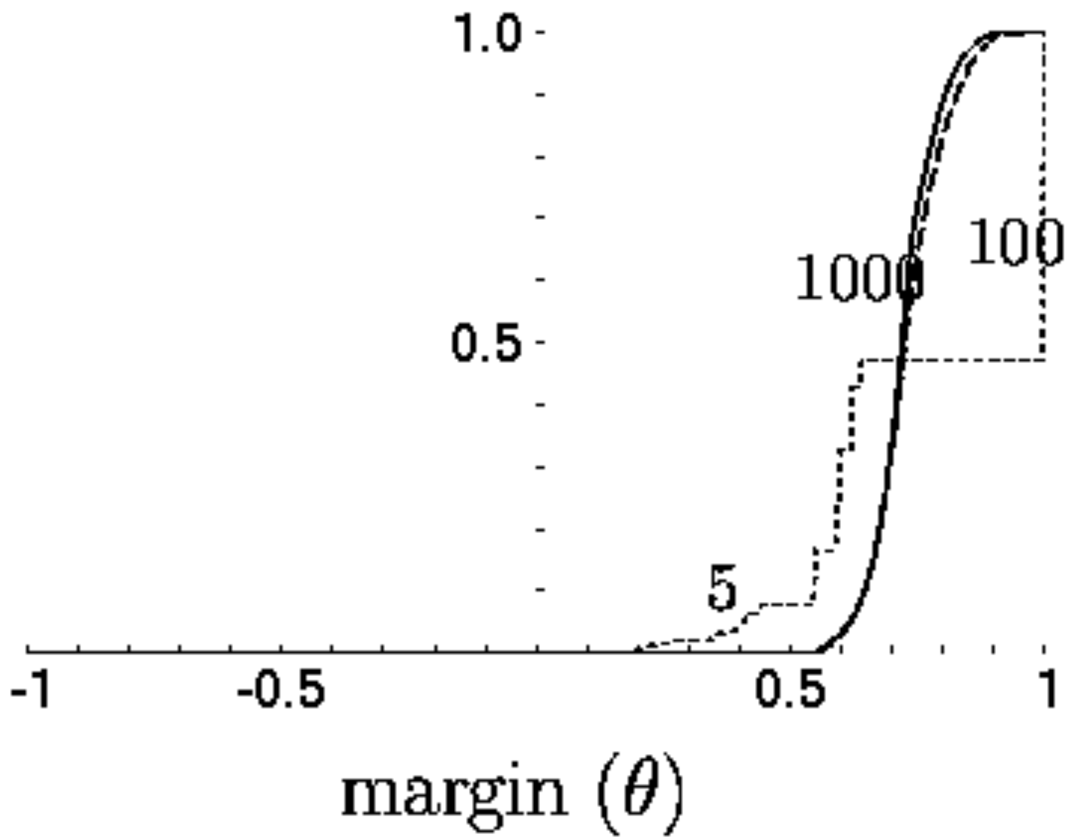
Explanation using margins



Explanation using margins



Experimental Evidence



AdaBoost Generalization Bound

- Also, a VC analysis gives a generalization bound:

$$R \leq R_{emp} + O\left(\sqrt{\frac{Td}{N}}\right)$$

(where d is VC of base classifier)

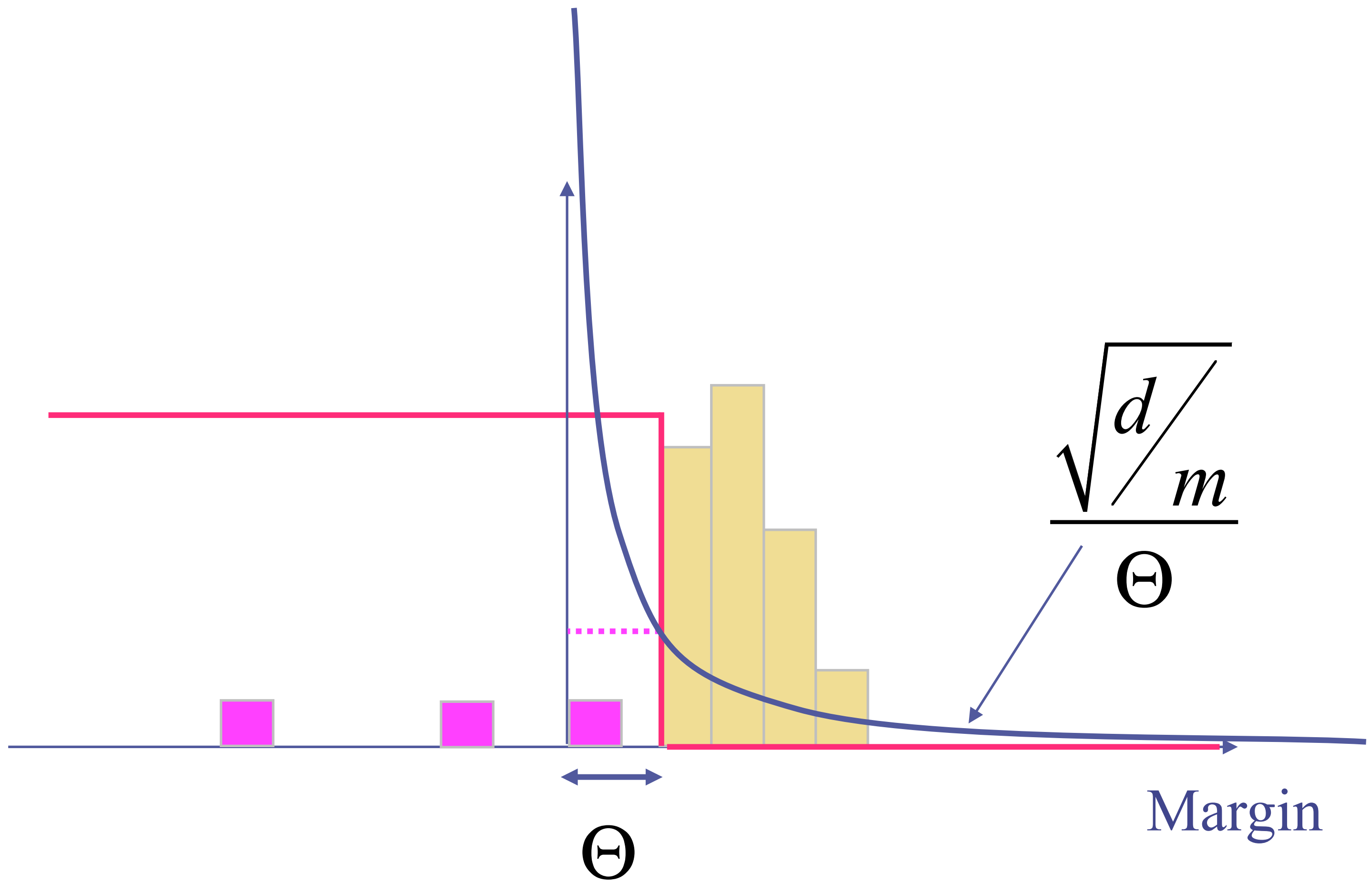
- But, more iterations → overfitting!
- A margin analysis is possible, redefine margin as:

$$mar_f\left(x,y\right) = \frac{y\sum_t \alpha_t h_t\left(x\right)}{\sum_t \left|\alpha_t\right|}$$

Then have $R \leq \frac{1}{N}\sum_{i=1}^{N} step\left(\theta - mar_f\left(x_i, y_i\right)\right) + O\left(\sqrt{\frac{d}{N\theta^2}}\right)$

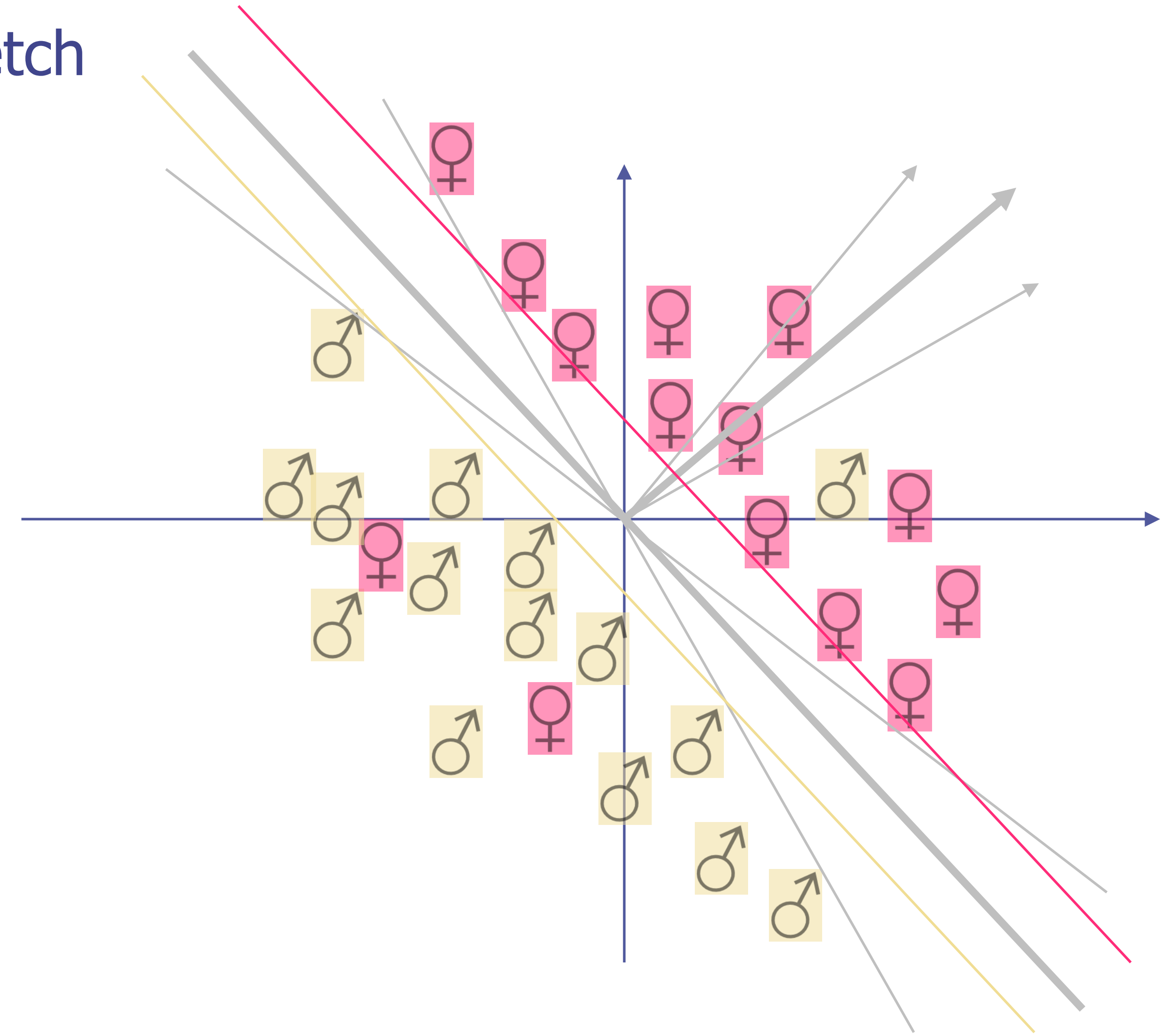
AdaBoost Generalization Bound

- Suggests this optimization problem:



AdaBoost Generalization Bound

- Proof Sketch



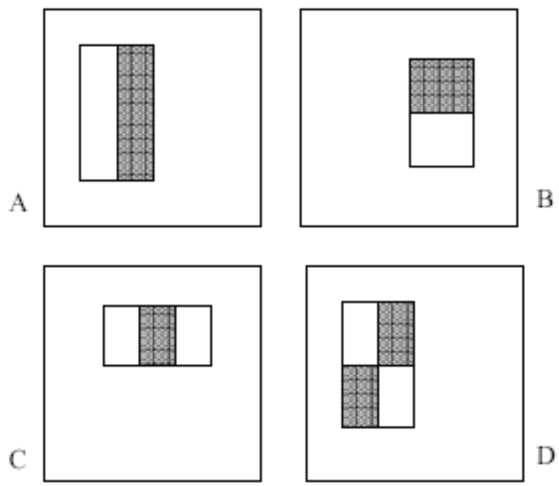
UCI Results

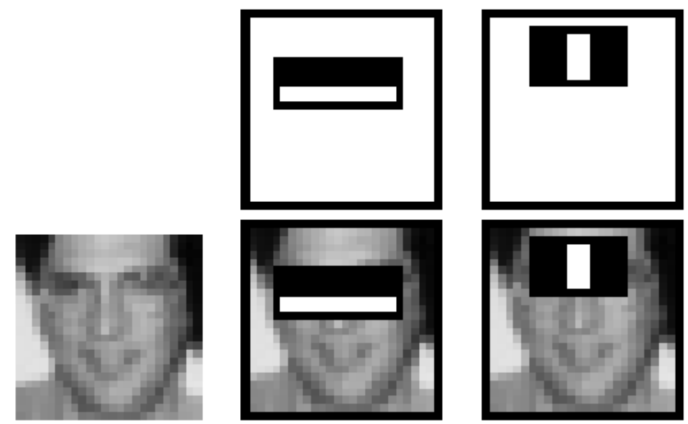
% test error rates

Database	Other	Boosting	Error reduction
Cleveland	27.2 (DT)	16.5	39%
Promoters	22.0 (DT)	11.8	46%
Letter	13.8 (DT)	3.5	74%
Reuters 4	5.8, 6.0, 9.8	2.95	~60%
Reuters 8	11.3, 12.1, 13.4	7.4	~40%

Boosted Cascade of Stumps

- Consider classifying an image as face/notface
- Use weak learner as stump that averages of pixel intensity
- Easy to calculate, white areas subtracted from black ones

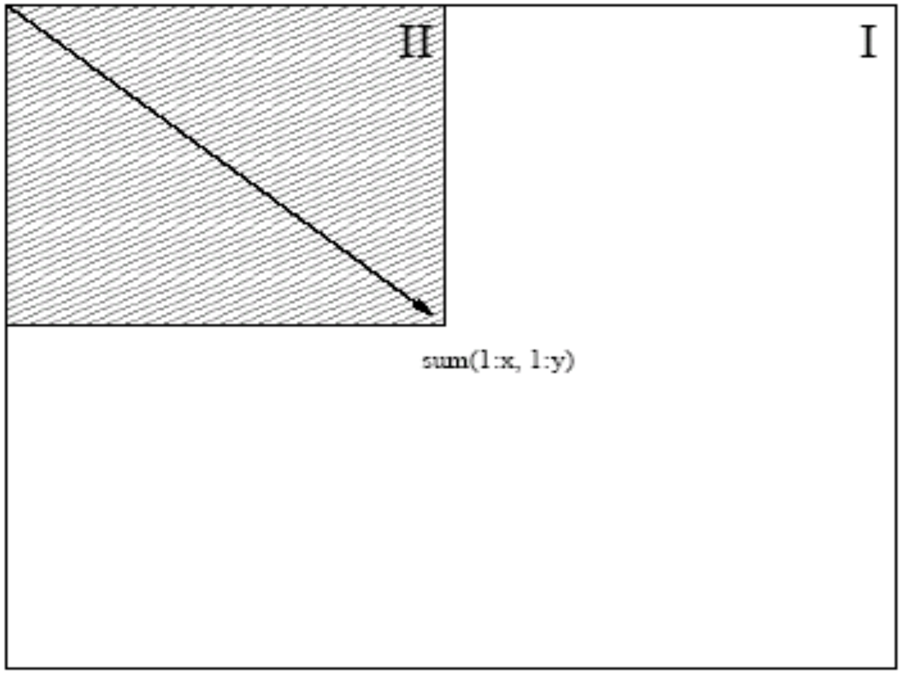




- A special representation of the sample called the integral image makes feature extraction faster.

Boosted Cascade of Stumps

- Summed area tables



- A representation that means any rectangle's values can be calculated in four accesses of the integral image.

Boosted Cascade of Stumps

- Summed area tables

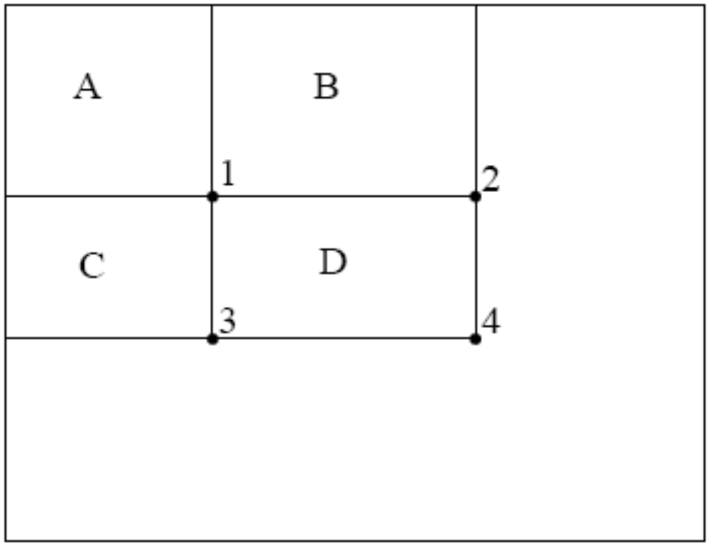
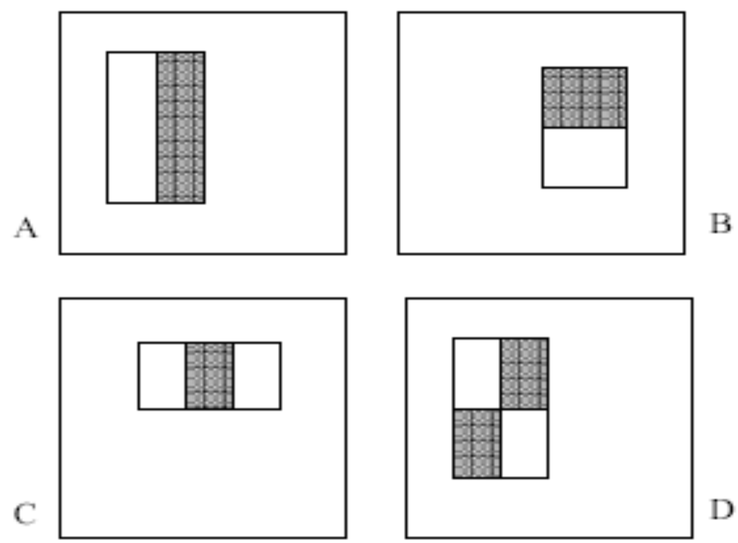


Figure 3: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$.

Boosted Cascade of Stumps

- The base size for a sub window is 24 by 24 pixels.
- Each of the four feature types are scaled and shifted across all possible combinations
- In a 24 pixel by 24 pixel sub window there are ~160,000 possible features to be calculated.



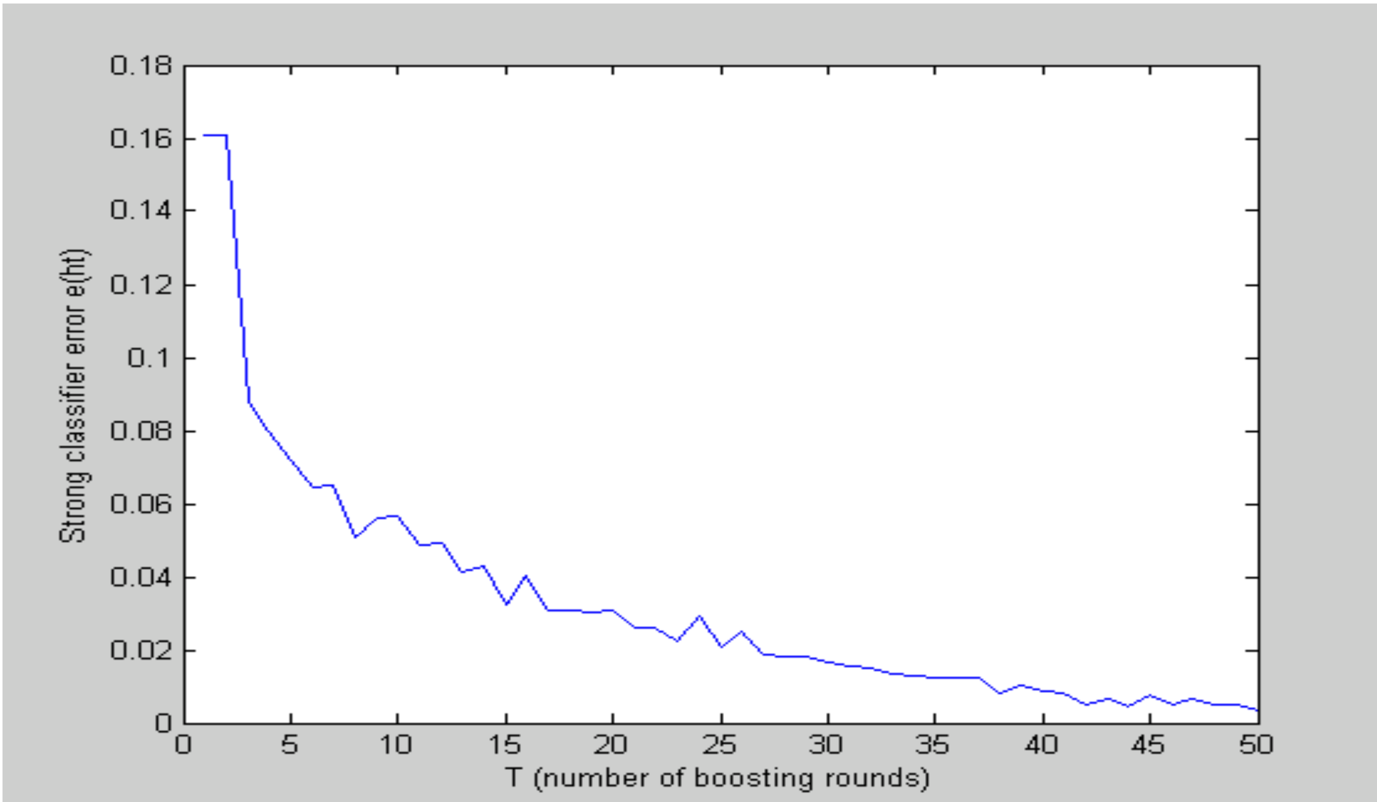
Boosted Cascade of Stumps

•Viola-Jones algorithm, with K attributes (e.g., K = 160,000) we have 160,000 different decision stumps to choose from

At each stage of boosting
•given reweighted data from previous stage
•Train all K (160,000) single-feature perceptrons
•Select the single best classifier at this stage
•Combine it with the other previously selected classifiers
•Reweight the data
•Learn all K classifiers again, select the best, combine, reweight
•Repeat until you have T classifiers selected
•Very computationally intensive!

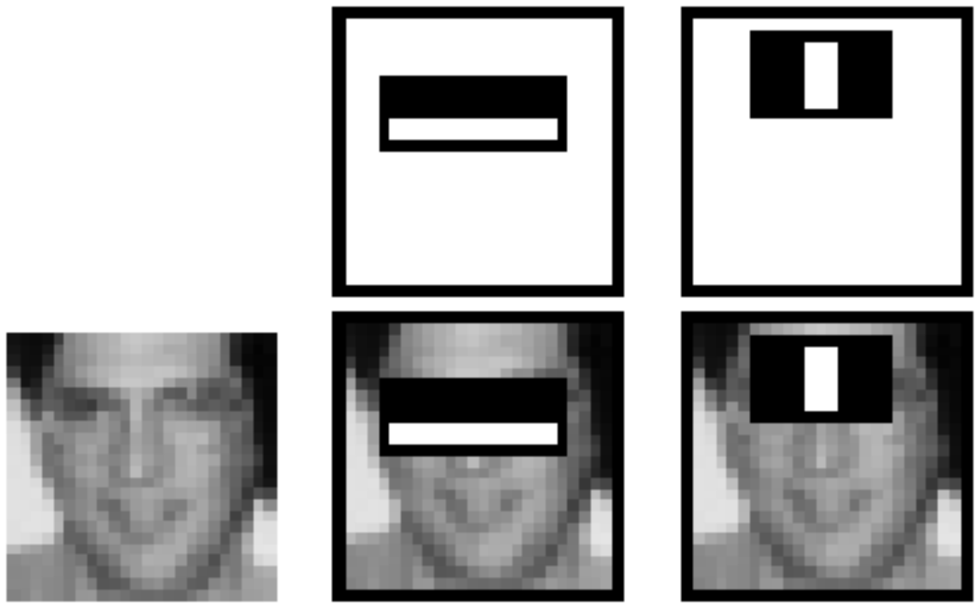
Boosted Cascade of Stumps

- Reduction in Error as Boosting adds Classifiers



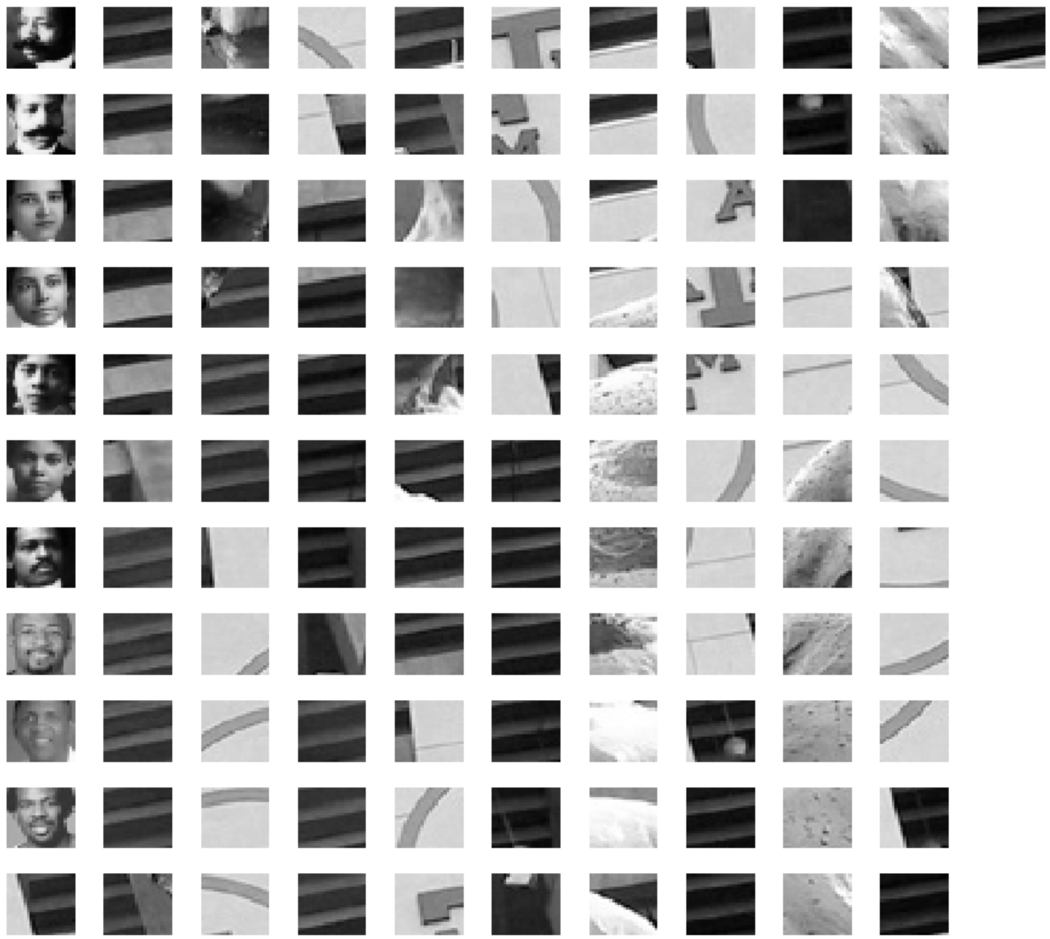
Boosted Cascade of Stumps

- First (e.g. best) two features learned by boosting



Boosted Cascade of Stumps

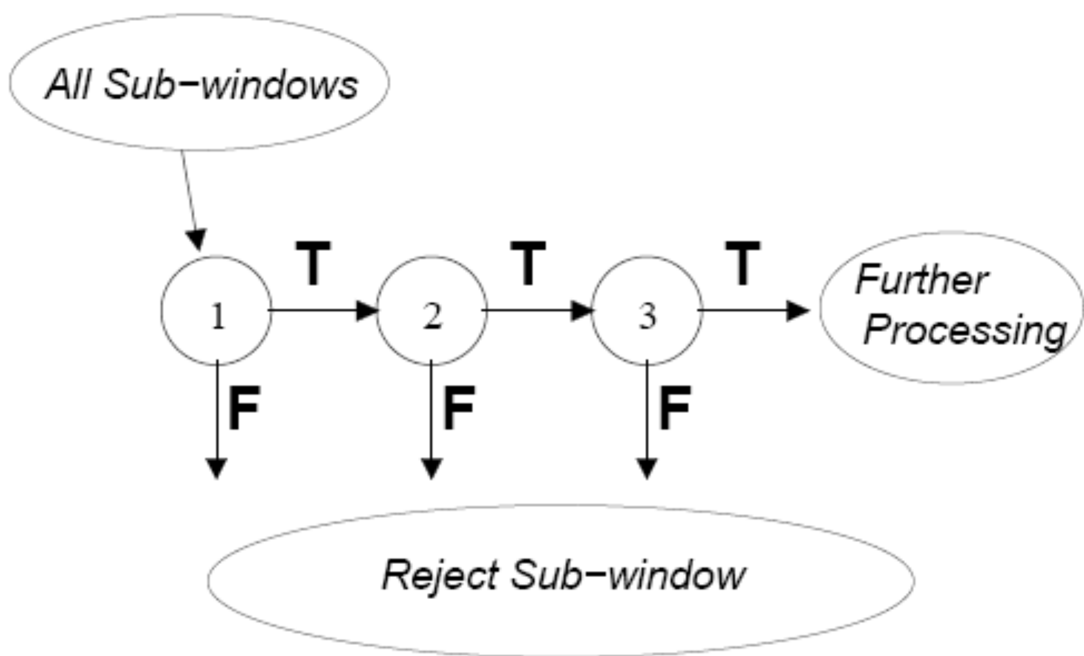
- Example training data



Boosted Cascade of Stumps

- To find faces, scan all squares at different scales, slow ☹

- Boosting finds ordering on weak learners (best ones first)
- Idea: cascade stumps to avoid too much computation! ☺



Boosted Cascade of Stumps

•Training time = weeks (with 5k faces and 9.5k non-faces)

•Final detector has 38 layers in the cascade, 6060 features

•700 Mhz processor:
•Can process a 384 x 288 image in 0.067 seconds (in 2003 when paper was written)

Boosted Cascade of Stumps

- Results