Advanced Machine Learning & Perception

Instructor: Tony Jebara

Topic 10

- Combining Multiple Classifiers
- Voting
- Mixture of Experts
- Boosting
- Adaboost

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

Combining Multiple Learners

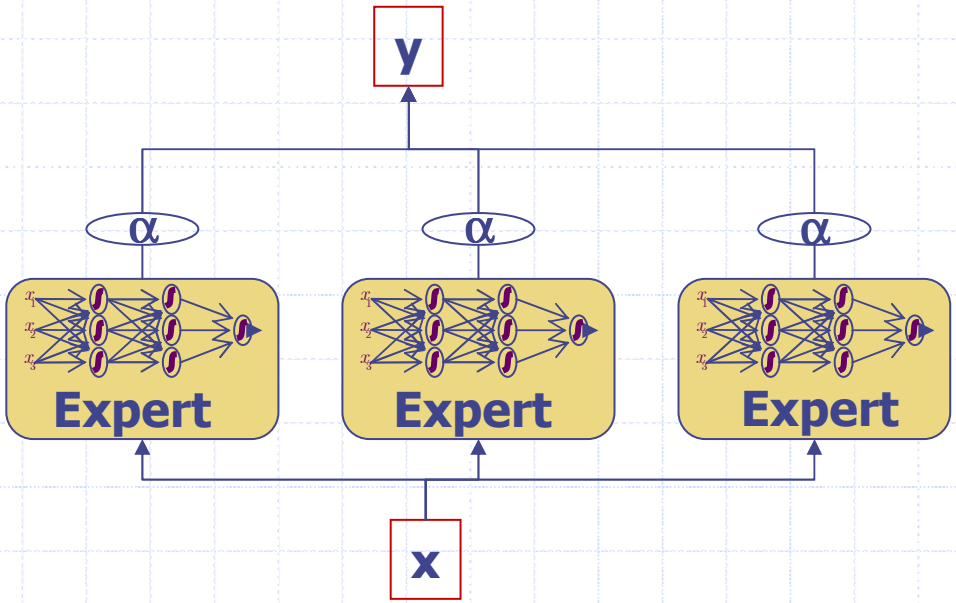
- Have many simple learners
- Also called base learners or weak learners which have a classification error of <0.5
- Combine or vote them to get a higher accuracy
- No free lunch: there is no guaranteed best approach here
- Different approaches:
  - Voting
    - combine learners with fixed weight
  - 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 $h_t(x)$
- Average their prediction with weights

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

- Like mixture of experts but weight is constant with input



Mixtures of Experts

- Mixture of Gaussians is a complicated generative model from many simple Gaussians work together
- In many learning problems, a complicated classifier or regression model can be seen as many simpler classification / regression mini-models working together
- Consider complicated general question X, want answer Y
- We have a panel of simple specialized Experts:

 **Doctor**  **Lawyer**  **Mechanic**

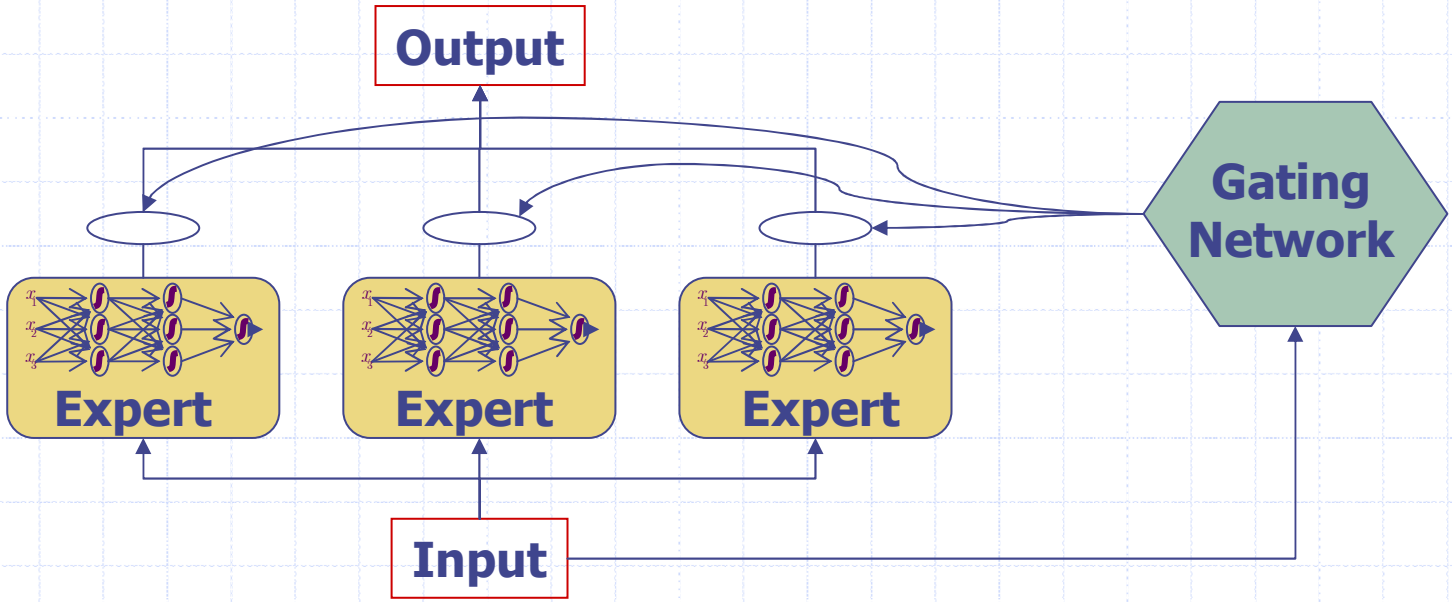
- Each expert takes X and gives an answer Y
- We want to take their answers and combine them
- Gate or Vote on each Expert's answer depending on their expertise or confidence in domain X (**car,** medicine, law)

5%  **No** **5%**  **No** **90%**  **Yes**

Mixture of Experts

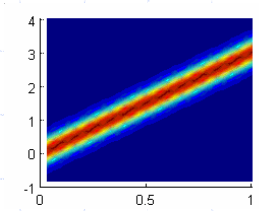
- Have T classifiers or experts $h_t(x)$ and a gating fn $\alpha_t(x)$
- Average their prediction with variable weights
- But, adapt parameters of the gating function and the experts (fixed total number T of experts)

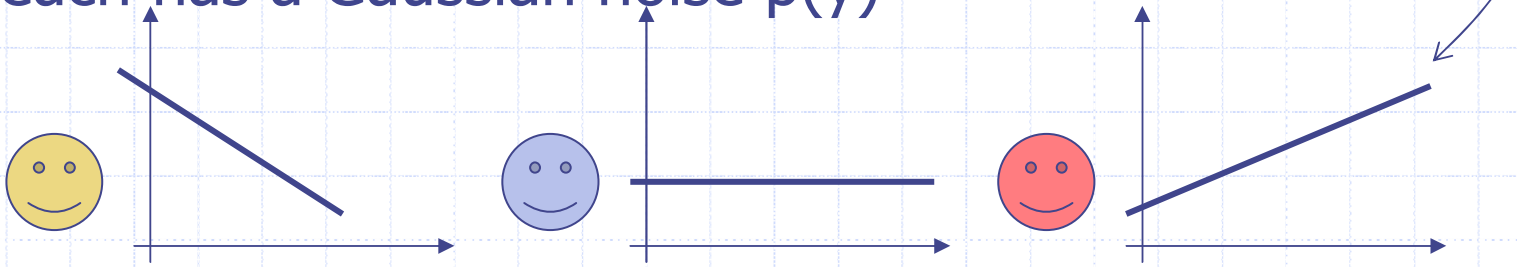
$$f(x) = \sum_{t=1}^{T} \alpha_t(x)\, h_t(x) \;\; where\, \alpha_t(x) \geq 0\, and \sum_{t=1}^{T} \alpha_t(x) = 1$$



Mixture of Experts

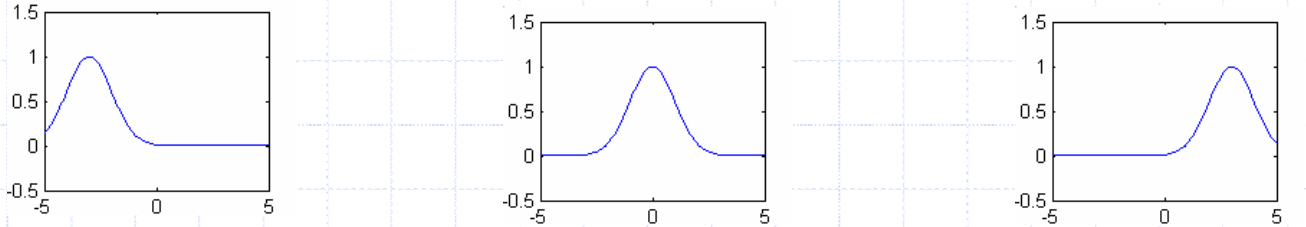
- Suppose we have several linear experts
- Also each has a Gaussian noise p(y)





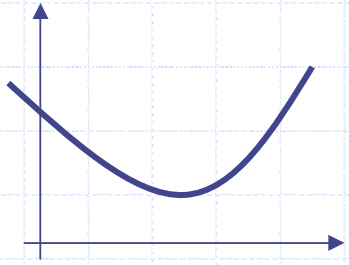
$$N\left(y \mid \mu_k + \theta_k^T x, \sigma_k^2\right) \qquad N\left(y \mid \mu_k + \theta_k^T x, \sigma_k^2\right) \qquad N\left(y \mid \mu_k + \theta_k^T x, \sigma_k^2\right)$$

- Also have Gaussian gates that vote for each expert



$$p(y \mid x) = \sum_k Gate_k(x)\, Expert_k(y \mid x)$$

$$= \sum_k \frac{\pi_k N\left(x \mid \mu_k, \Sigma_k\right)}{\sum_j \pi_j N\left(x \mid \mu_k, \Sigma_k\right)} N\left(y \mid \mu_k + \theta_k^T x, \sigma_k^2\right)$$

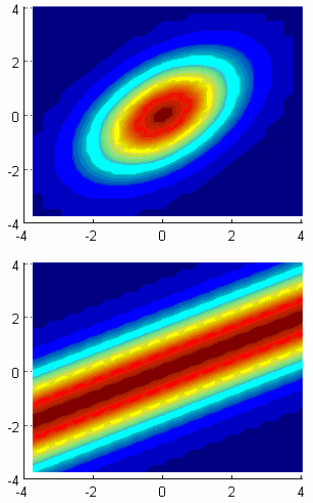


Mixtures of Experts

- Suppose each Expert is a Gaussian which does regression
- Recall a Gaussian conditioned gives linear model

$$p(x,y) = \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\left(\begin{bmatrix}x\\y\end{bmatrix}-\begin{bmatrix}\mu_x\\\mu_y\end{bmatrix}\right)^T \begin{bmatrix}\Sigma_{xx} & \Sigma_{yx}\\\Sigma_{xy} & \Sigma_{yy}\end{bmatrix}^{-1} \left(\begin{bmatrix}x\\y\end{bmatrix}-\begin{bmatrix}\mu_x\\\mu_y\end{bmatrix}\right)\right)$$

$$p(y \mid x) = N\left(y \mid \mu_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x-\mu_x), \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}\right)$$



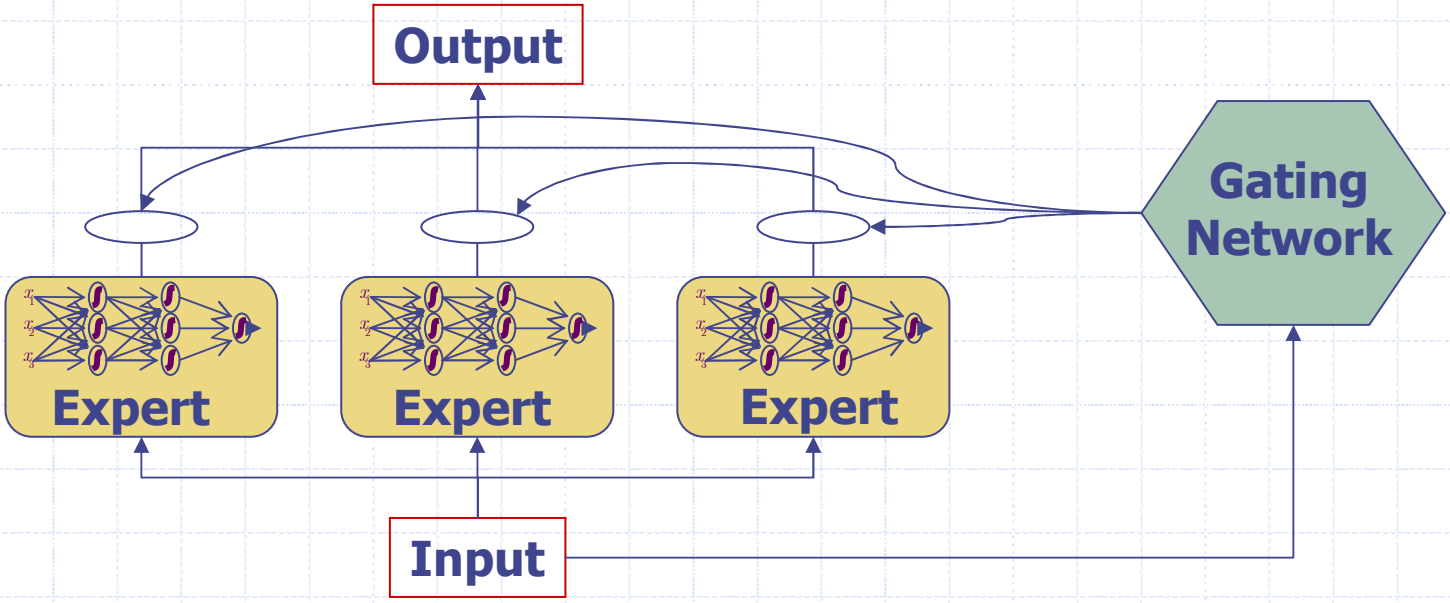
- We can view a conditioned mixture of Gaussians as a mixture of linear experts and Gaussian gates...

$$p(y \mid x) = \sum_k \frac{\pi_k N_k(x \mid \mu_x, \Sigma_{xx})}{\sum_j \pi_j N_j(x \mid \mu_x, \Sigma_{xx})} N_k\left(y \mid \mu_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x-\mu_x), \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}\right)$$

- But, we only 'trust' them for certain values of X
- Can now consider other non-Gaussian experts and non Gaussian gates, for example each expert = NeuralNet

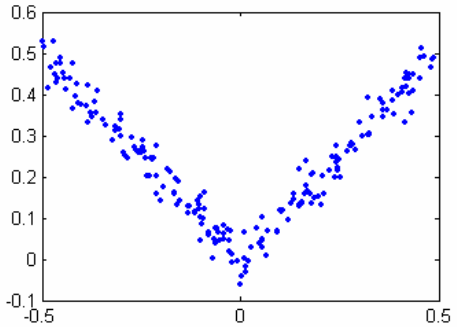
Mixture of Experts

- Use EM for Max Likelihood $l(\theta) = \sum_i \log \sum_k Gate_k\left(x^i\right) Expert_k\left(y^i \mid x^i\right)$
- Gates can be Gaussian, Neural Net, etc.
- Each Expert can be linear, Neural Network, etc.
- Use EM with Gates & Experts to conquer and divide data
- Update the Gates & Experts with weighted data
- Weighted max likelihood/gradient descent/backprop

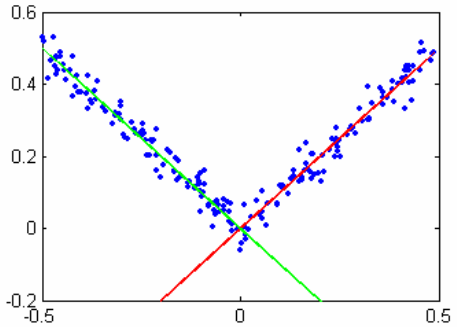


Learning Mixture of Experts

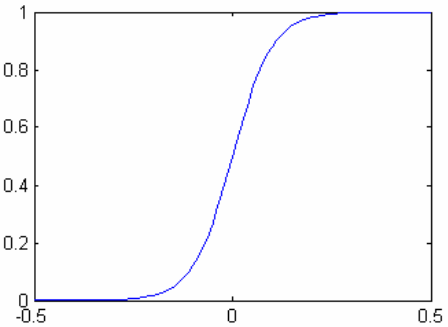
- Example: linear experts with softmax gating
- Softmax gates (Logistic): $Gate_k\left(x^i\right) = \frac{\exp\left(\phi_k^T x + b_k\right)}{\sum_k \exp\left(\phi_k^T x + b_k\right)}$
- Linear experts (Gaussian): $Expert_k\left(y \mid x\right) = N\left(y \mid \mu_k + \theta_k^T x, \sigma_k^2\right)$



Complex Data



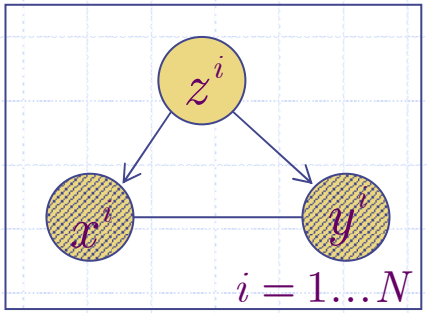
Green & Red Experts

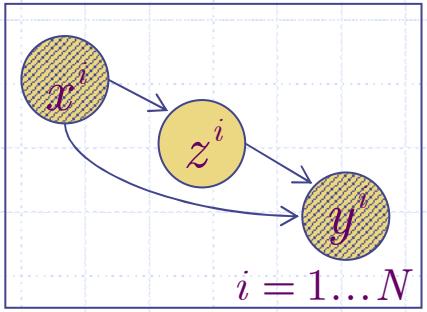


Red Gate

- Experts get data conditionally on Gates
- If we new which expert got which data, we could train them separately. Conquer and Divide just like EM…

EM for Mixture of Experts





- In mixture of Gaussians, roll the dice with α to pick which bump and then sample from it to get (x,y)
- In mixture of experts, look at x to get $\alpha(x)$. Roll the dice to pick an expert. Gives y sample from x (with noise).

$$\sum_k \alpha_k(x) = 1 \qquad \alpha_k(x) \geq 0$$

- E-step:

$$\tau_k^i = \frac{Gate_k\left(x^i\right) Expert_k\left(y^i \mid x^i\right)}{\sum_k Gate_k\left(x^i\right) Expert_k\left(y^i \mid x^i\right)}$$

- M-step:

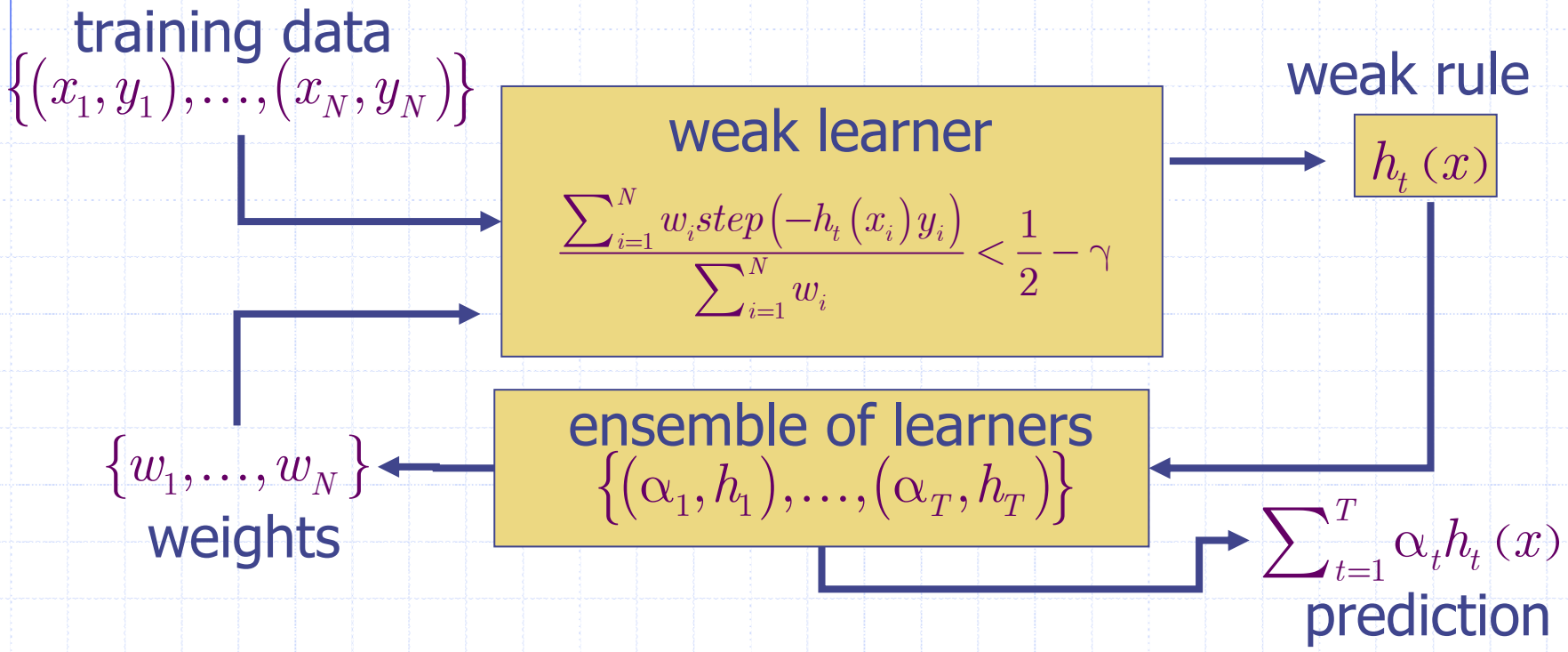
Learn Individually
Could be Neural Nets
Get weighted Backprop

Expert: $\max_\theta \sum_i \sum_k \tau_k^i \log Expert_k\left(y^i \mid x^i\right)$

Gate: $\max_\theta \sum_i \sum_k \tau_k^i \log Gate_k\left(x^i\right)$

Boosting

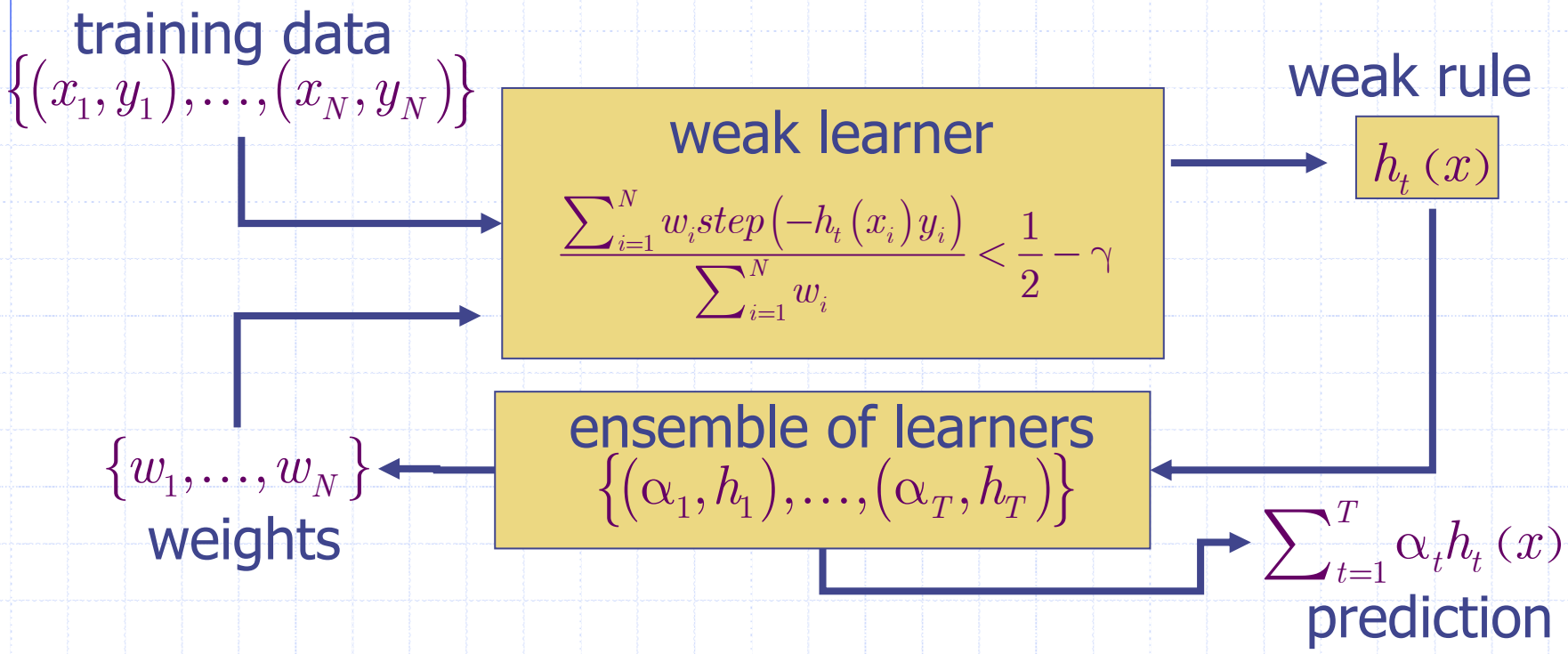
- Actively find complementary or synergistic weak learners
- Train next learner based on mistakes of previous ones.
- Average prediction with fixed weights
- Find next learner by training on weighted versions of data.



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

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

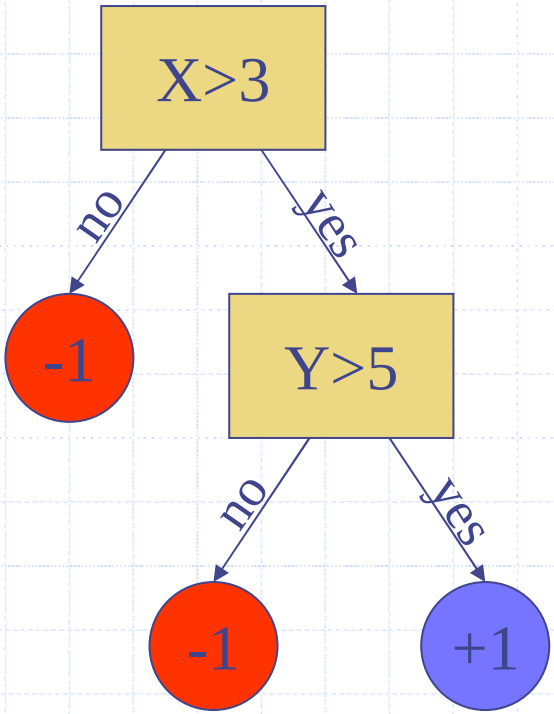


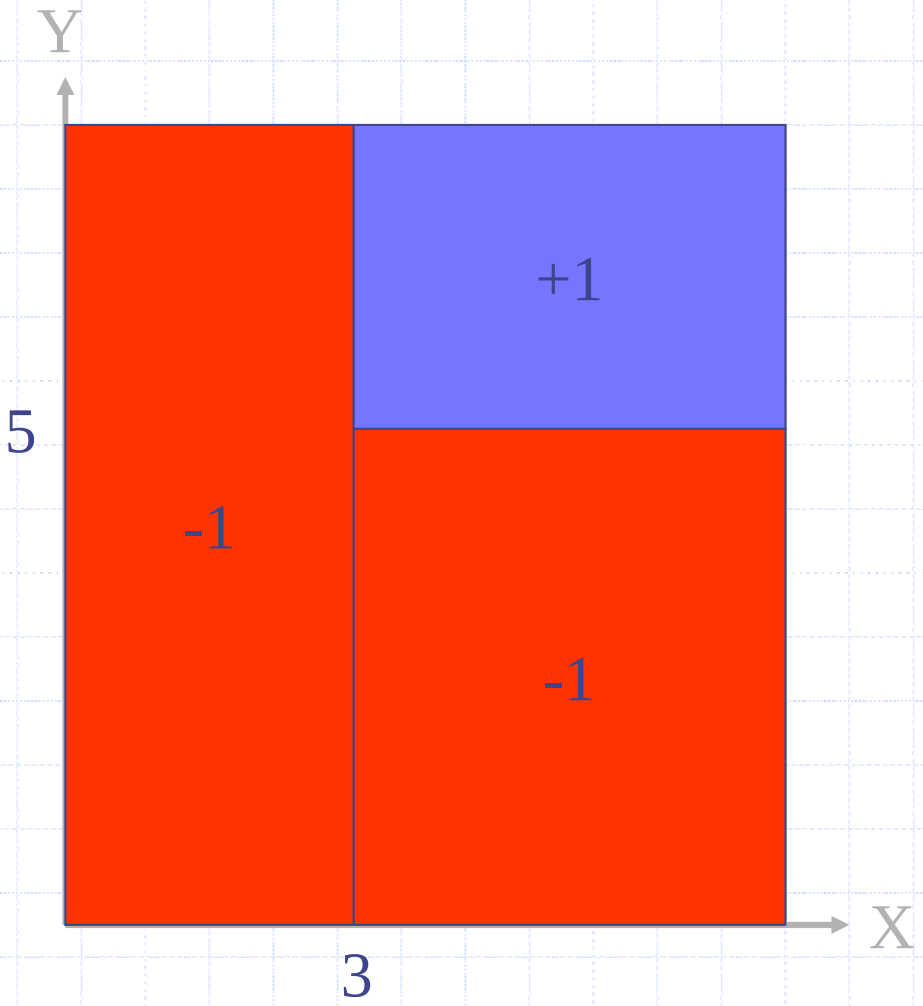
AdaBoost

- Choose base learner & α_t to $\min_{\alpha_t, h_t} \sum_{i=1}^{N} \exp\left(-y_i \alpha_t h_t\left(x_i\right)\right)$
- Recall error of base classifier h_t must be $\varepsilon_t = \frac{\sum_{i=1}^{N} w_i step\left(-h_j\left(x_i\right) y_i\right)}{\sum_{i=1}^{N} w_i} < \frac{1}{2} - \gamma$
- For binary h, Adaboost puts this weight on weak learners: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ (instead of the more general rule)
- Adaboost picks the following for the weights on data for the next round (here Z is the normalizer to sum to 1)

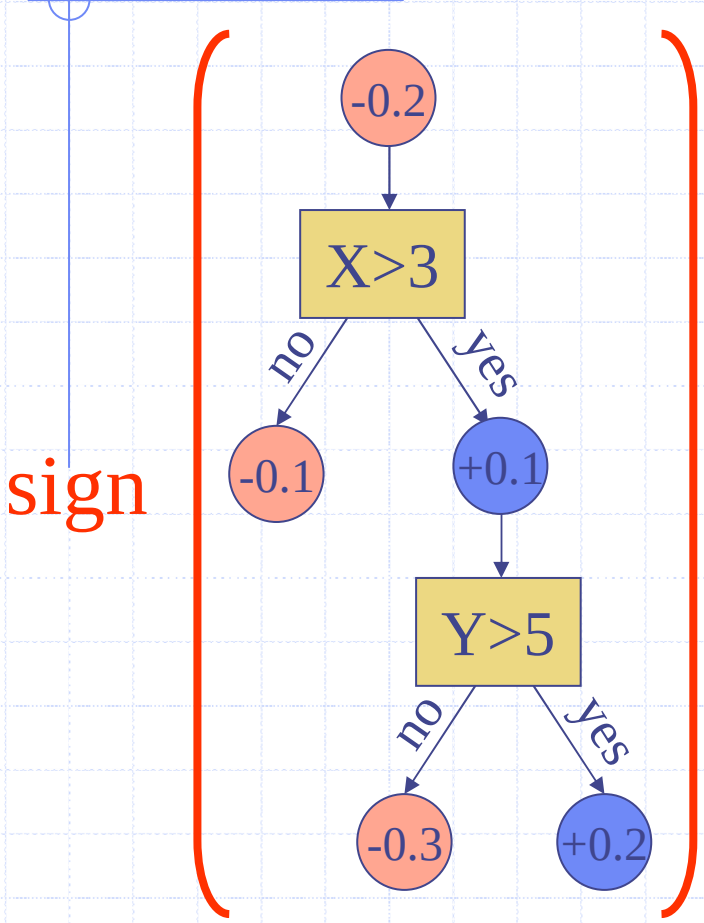
$$w_i^{t+1} = \frac{w_i^t \exp\left(-\alpha_t y_i h_t\left(x_i\right)\right)}{Z_t}$$

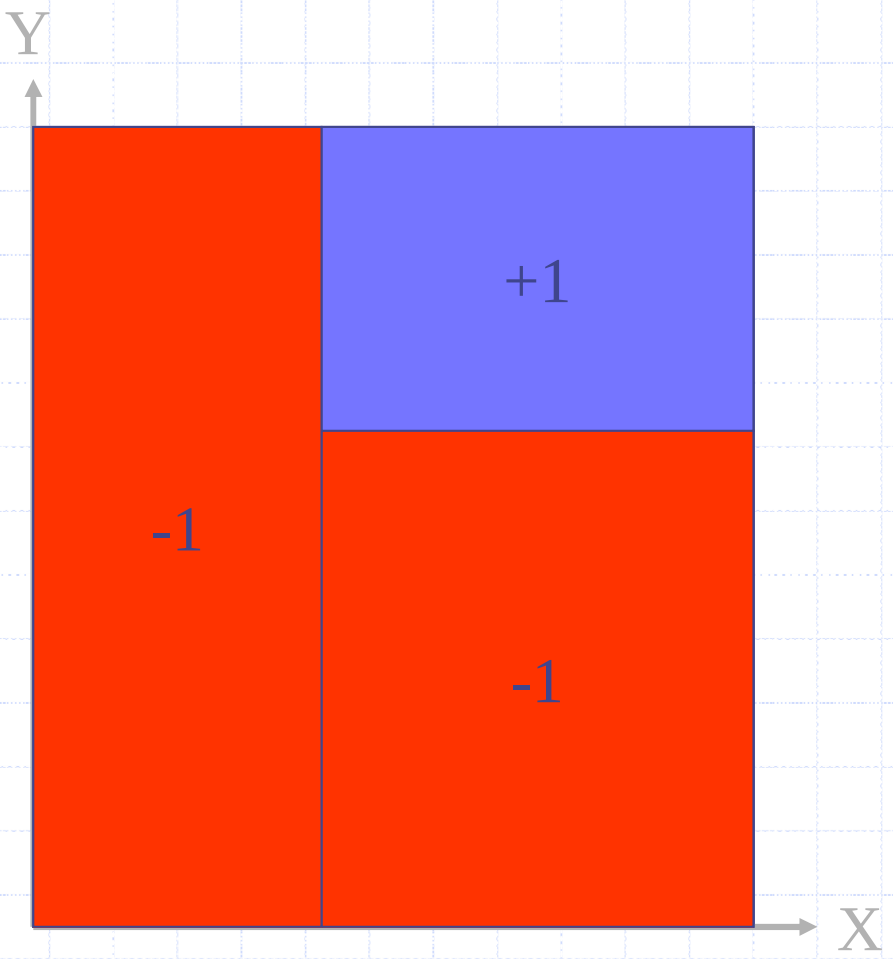
Decision Trees



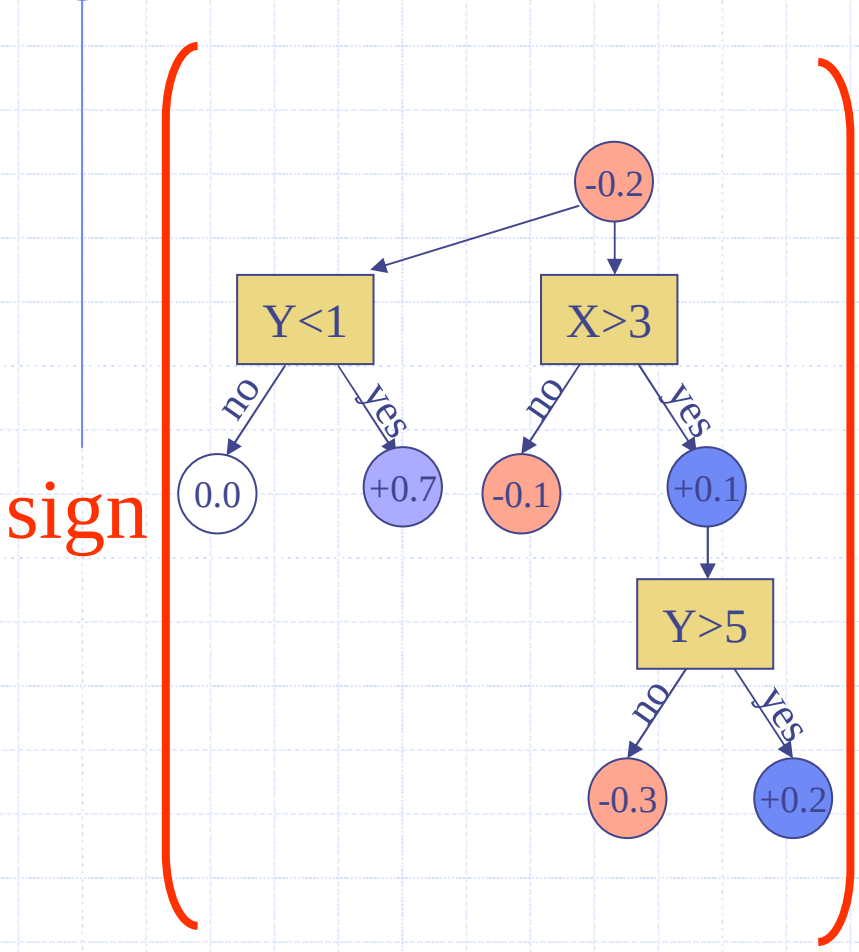


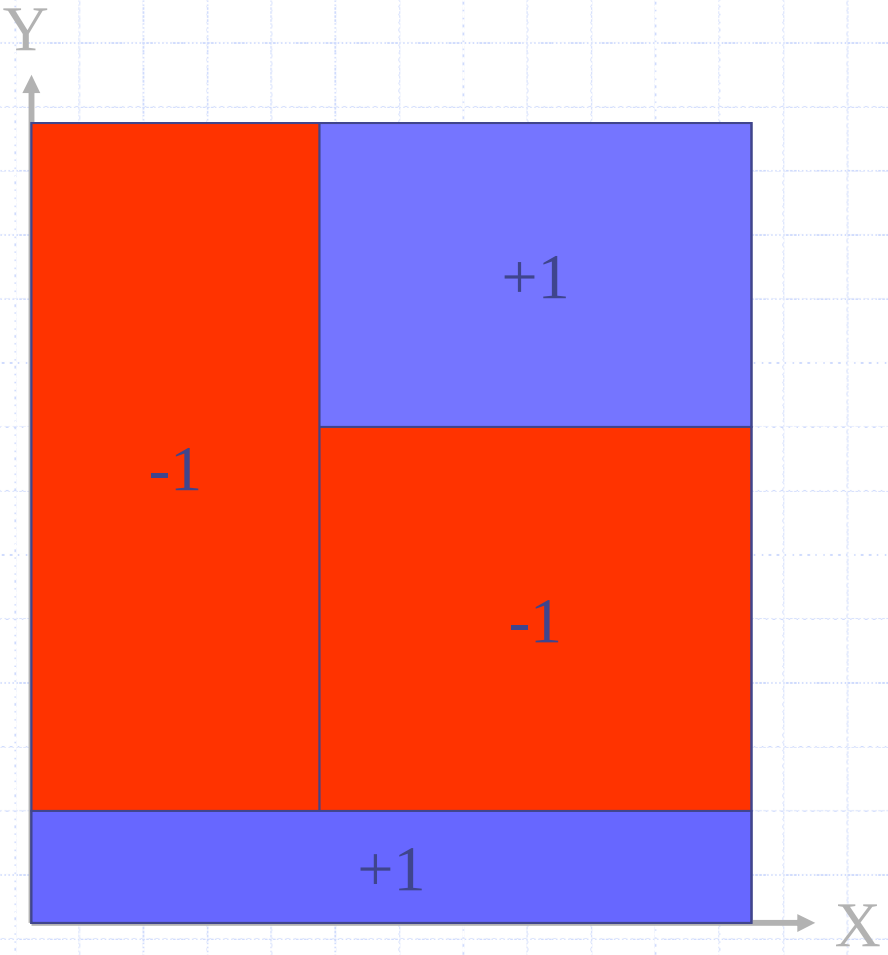
Decision tree as a sum





An alternating decision tree

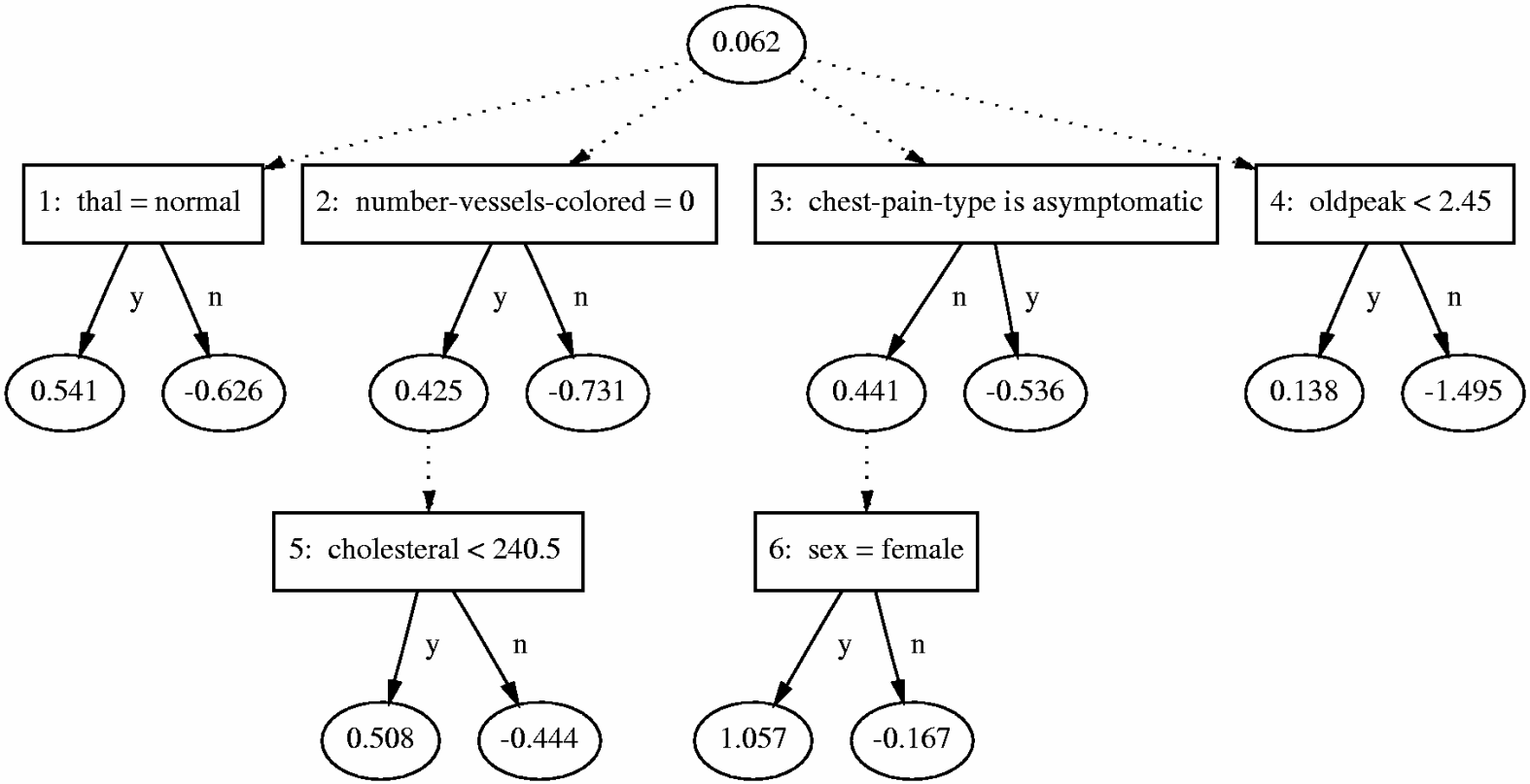




Example: Medical Diagnostics

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

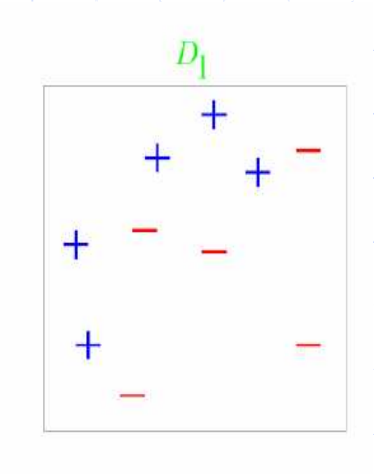
Ad-Tree Example



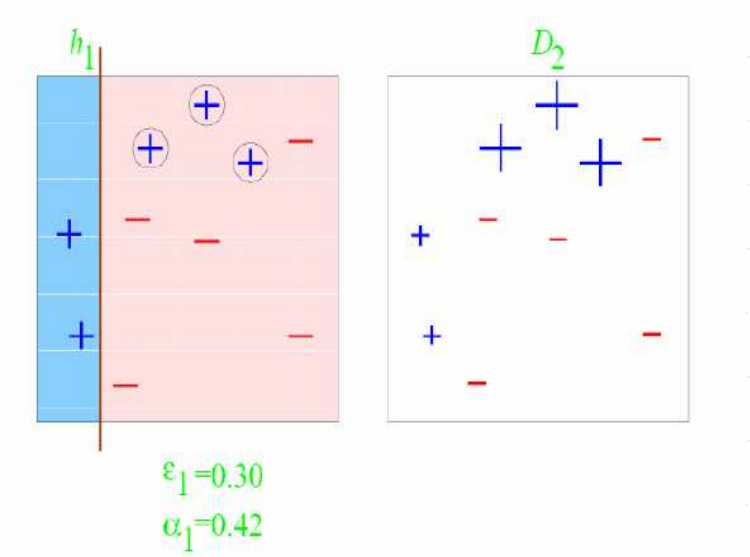
Cross-validated accuracy

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%

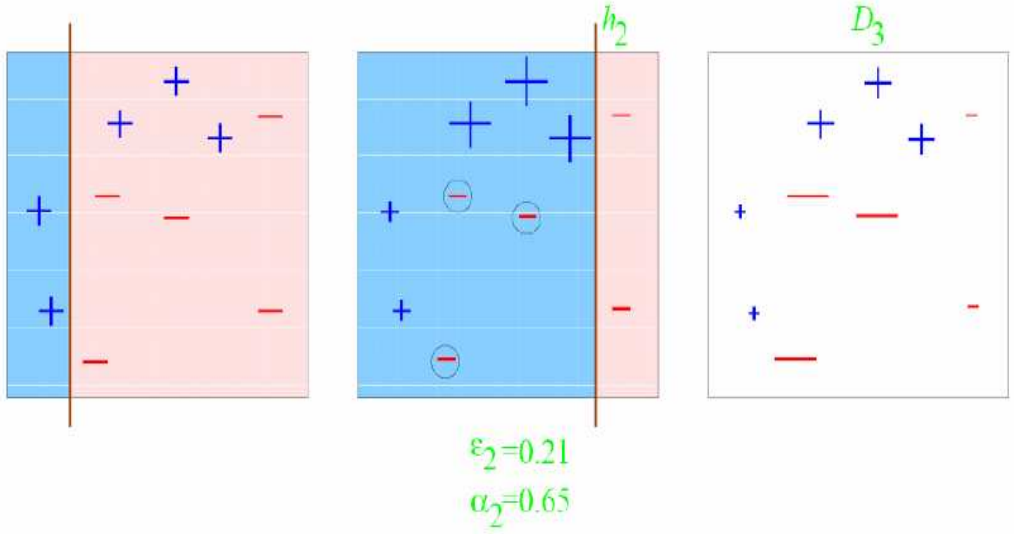
AdaBoost Demo



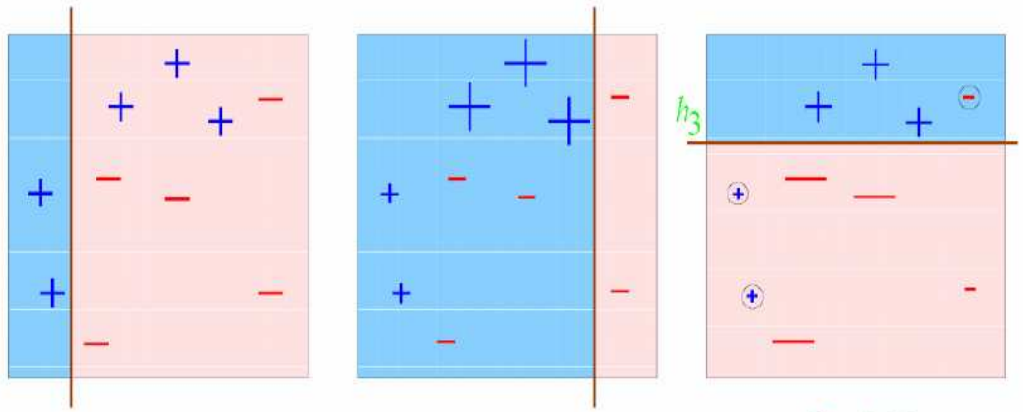
AdaBoost Demo



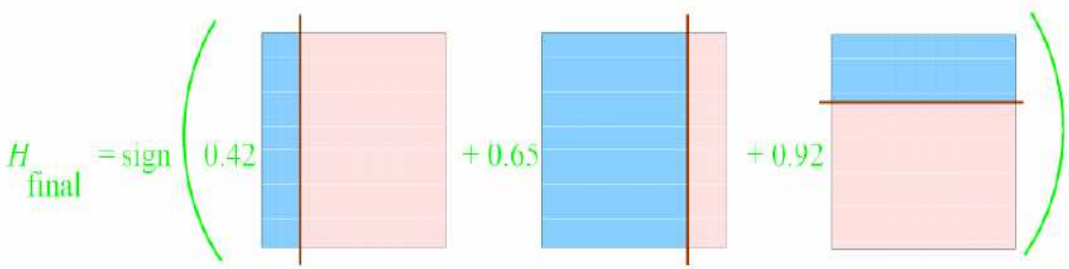
AdaBoost Demo

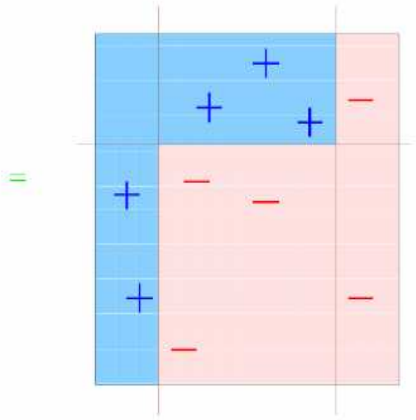


AdaBoost Demo



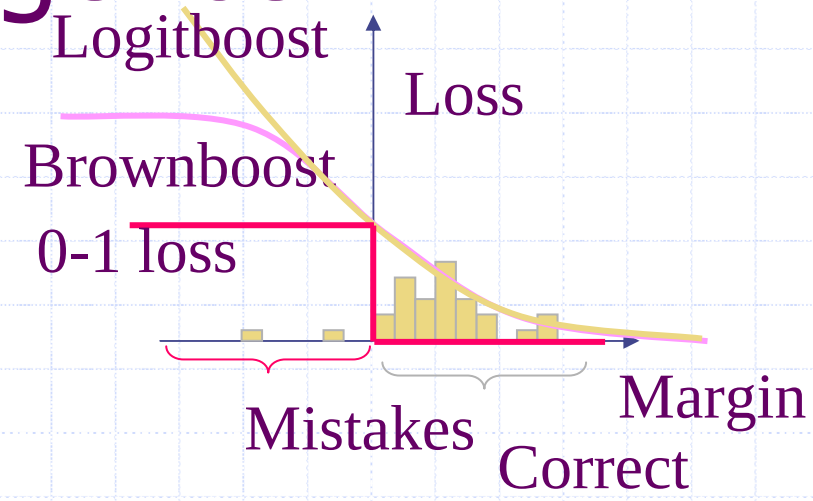
AdaBoost Demo





AdaBoost Convergence

- Rationale?
- Consider bound on the training error:



$$R_{emp} = \frac{1}{N}\sum_{i=1}^{N} step\left(-y_i f\left(x_i\right)\right)$$

$$\leq \frac{1}{N}\sum_{i=1}^{N} \exp\left(-y_i f\left(x_i\right)\right) \quad \textbf{exp bound on step}$$

$$= \sum_{i=1}^{N} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t\left(x_i\right)\right) \quad \textbf{definition of f(x)}$$

$$= \prod_{t=1}^{T} Z_t \quad \textbf{recursive use of Z}$$

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

AdaBoost Convergence

- Convergence? Consider the binary h_t case.

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

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

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

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

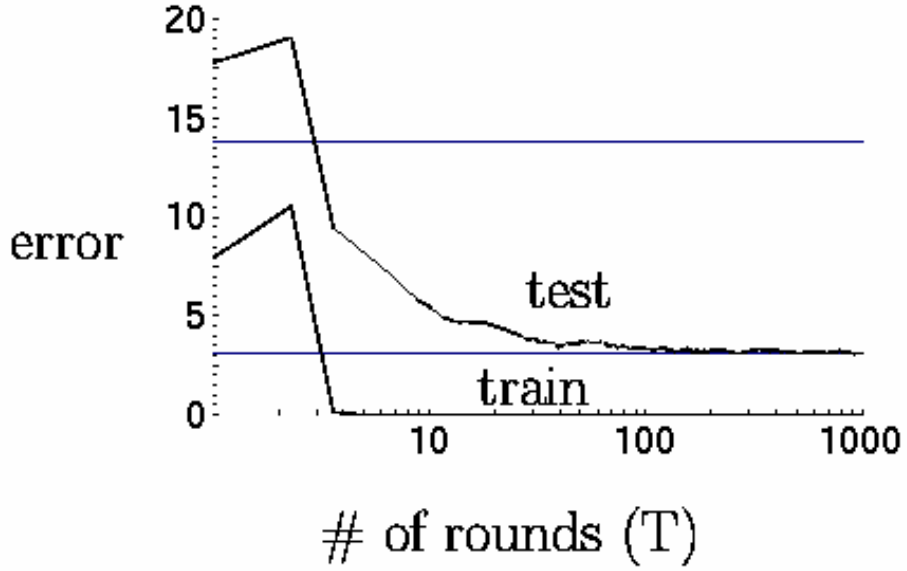
$$= \prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)} = \prod_{t=1}^{T} \sqrt{(1-4\gamma_t^2)} \le \exp\left(-2\sum_t \gamma_t^2\right)$$

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

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

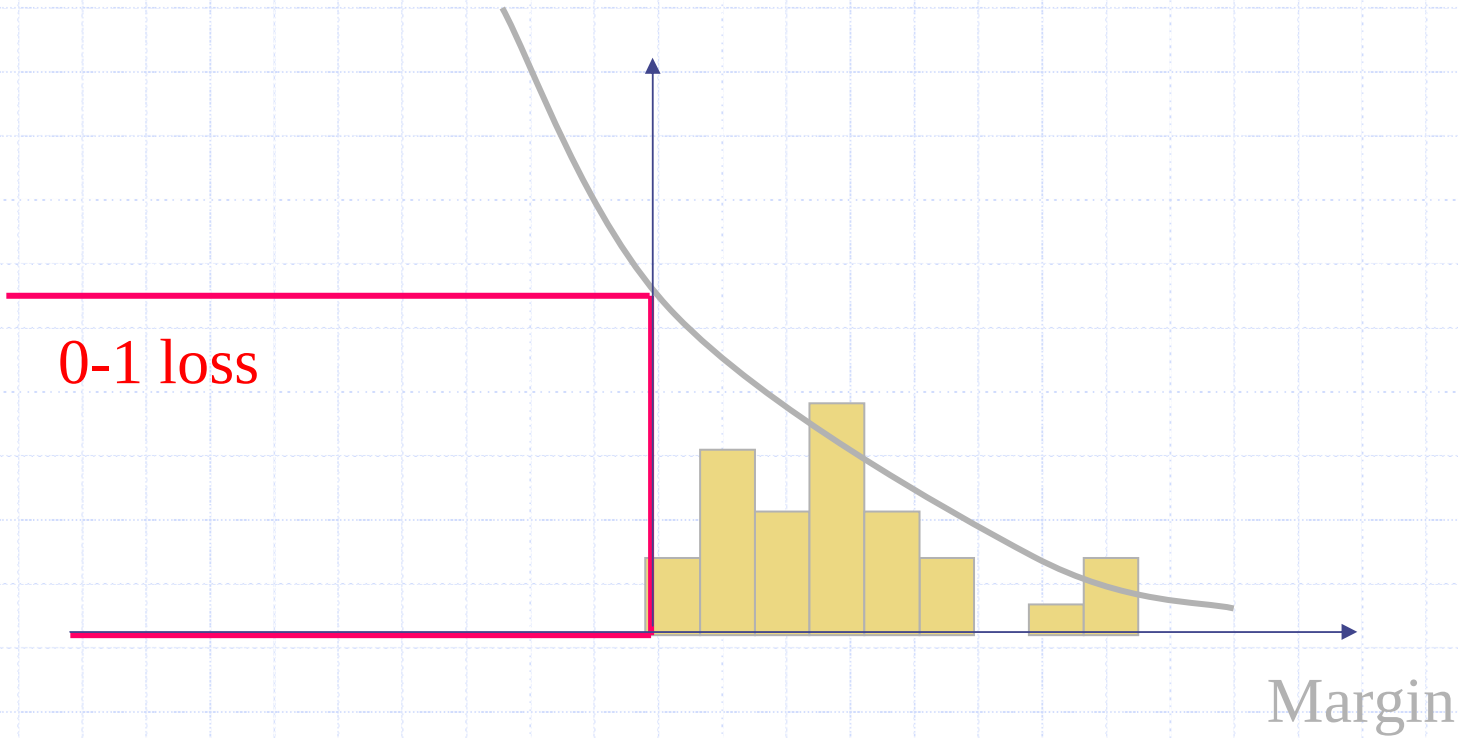
Curious phenomenon

Boosting decision trees

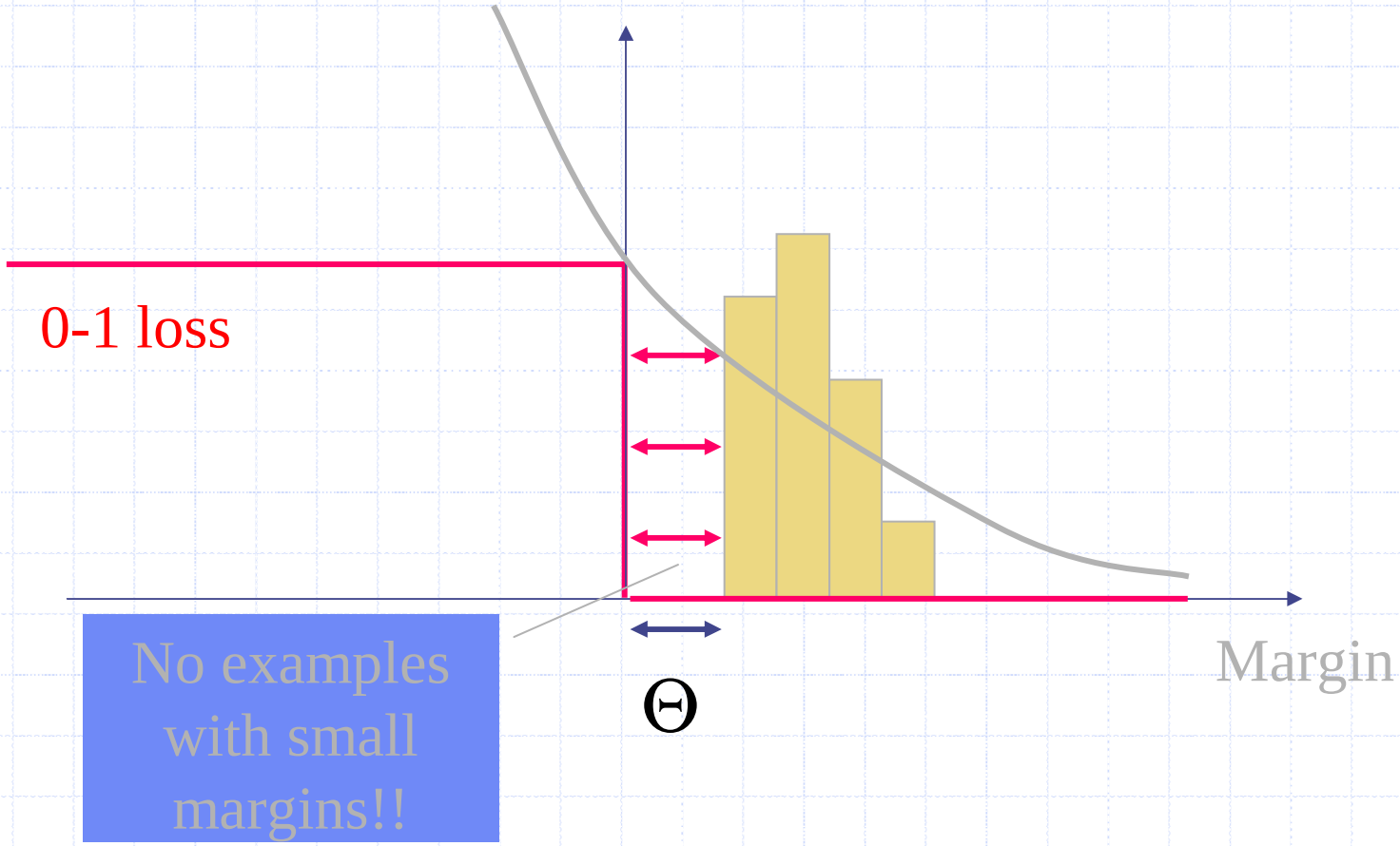


Using <10,000 training examples we fit >2,000,000 parameters

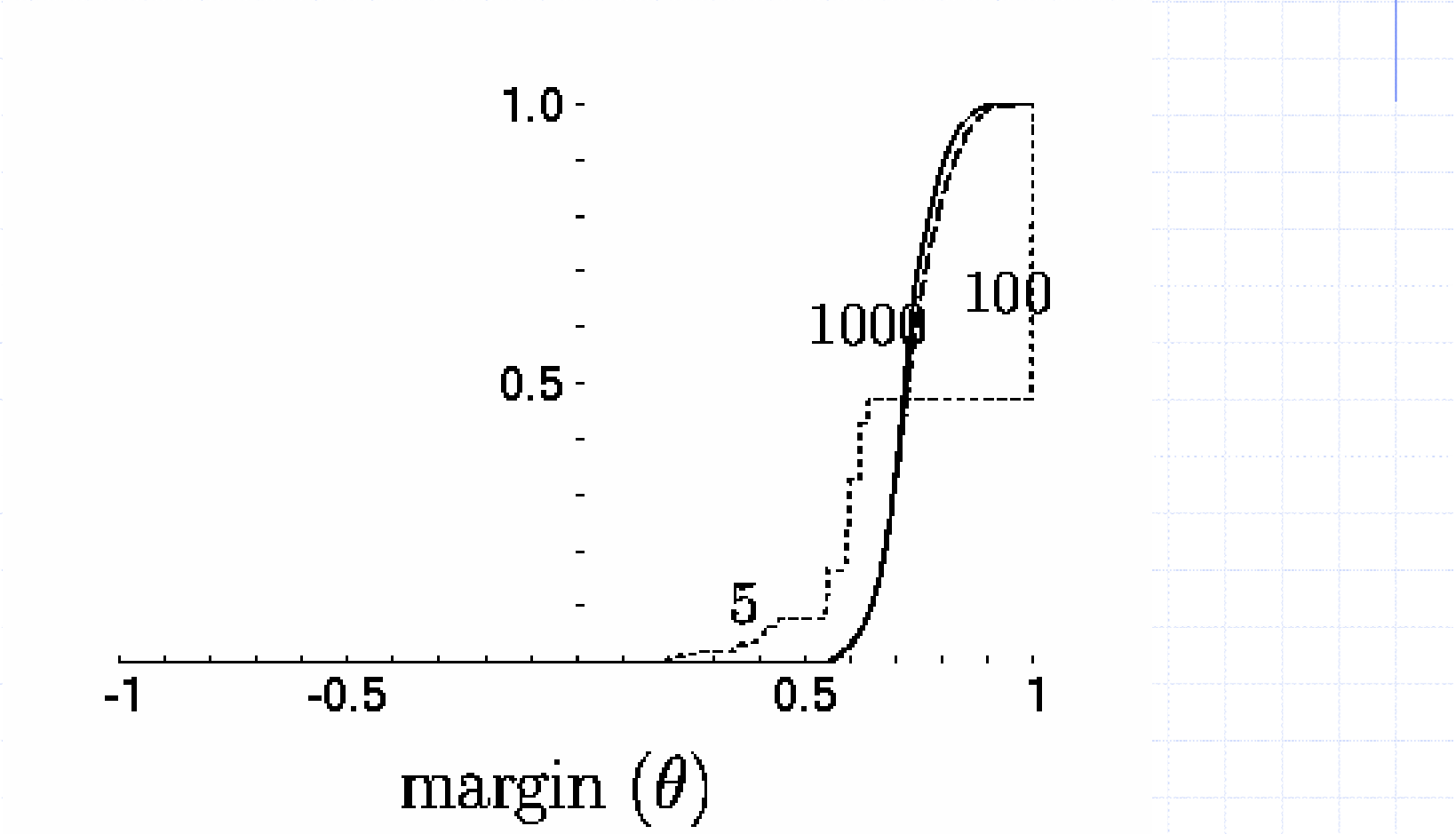
Explanation using margins



Explanation using margins



Experimental Evidence



AdaBoost Generalization

- 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)

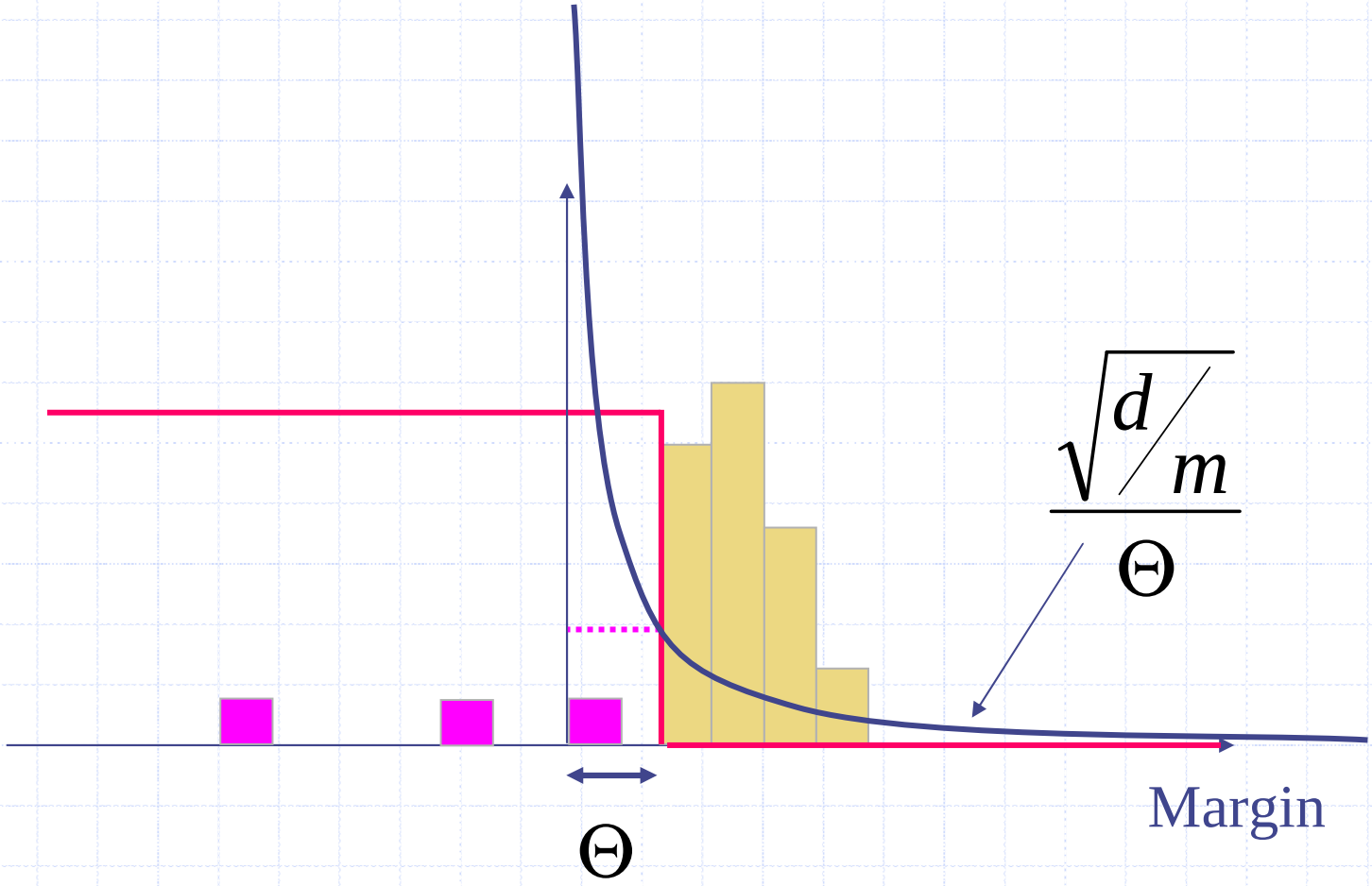
- But, more iterations à overfitting!
- A margin analysis is possible, redefine margin as:

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

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

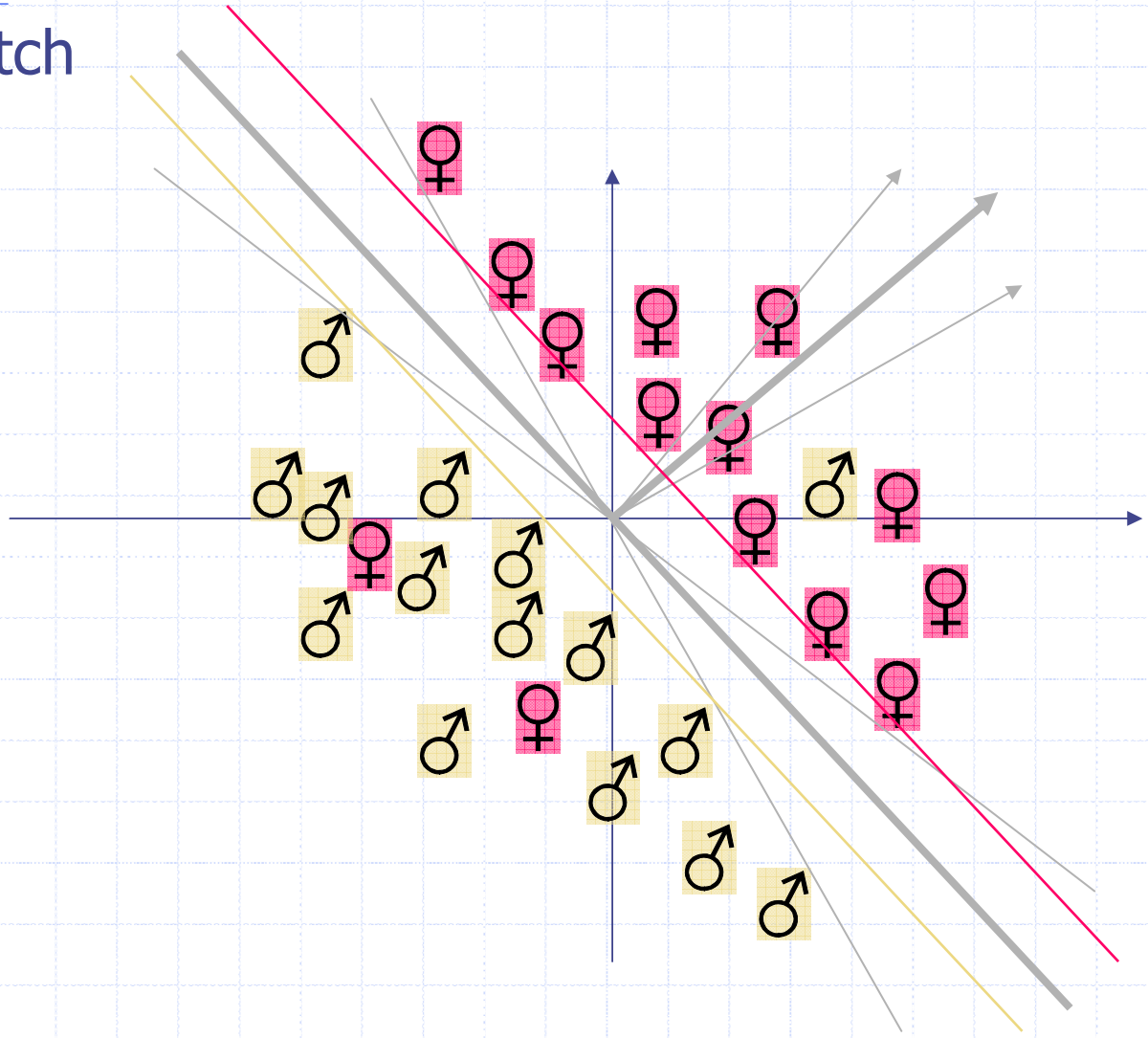
AdaBoost Generalization

- Suggests this optimization problem:



AdaBoost Generalization

- 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%