Bagging and Boosting

Lecture 20

Some slides are due to Robin Dhamankar

Vandi Verma & Sebastian Thrun

Ensemble Learning: Bagging and Boosting

- So far we have talked about design of a single classifier that generalizes well (want to “learn” f(x))
- From statistics, we know that it is good to average your predictions (reduces variance)
- Bagging
  - reshuffle your training data to create k different training sets and learn $f_1(x), f_2(x), \ldots, f_k(x)$
  - Combine the k different classifiers by majority voting

$$f_{FINAL}(x) = \mathrm{sign}[\Sigma\ 1/k\ f_i(x)\ ]$$

- Boosting
  - Assign different weights to training samples in a “smart” way so that different classifiers pay more attention to different samples
  - Weighted majority voting, the weight of individual classifier is proportional to its accuracy
  - Ada-boost (1996) was influenced by bagging, and it is superior to bagging

Bagging

- Generate a random sample from training set by selecting ***l*** elements (out of ***n*** elements available) with replacement
- Repeat the sampling procedure, getting a sequence of ***k*** independent training sets
- A corresponding sequence of classifiers $f_1(x),f_2(x),\dots,f_k(x)$ is constructed for each of these training sets, using the same classification algorithm
- To classify an unknown sample x, let each classifier predict.
- The *bagged classifier* $f_{FINAL}(x)$ then combines the predictions of the individual classifiers to generate the final outcome, frequently this combination is simple voting

Boosting: motivation

- It is usually hard to design an accurate classifier which generalizes well
- However it is usually easy to find many “rule of thumb” *weak* classifiers
  - A classifier is weak if it is only slightly better than random guessing
- Can we combine several weak classifiers to produce an accurate classifier?
  - Question people have been working on since 1980’s

Ada Boost

- Let's assume we have 2-class classification problem, with $y_i \in \{-1,1\}$
- Ada boost will produce a discriminant function:

$$\boldsymbol{g}(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t \boldsymbol{f}_t(\boldsymbol{x})$$

- where $f_t(x)$ is the "weak" classifier
- As usual, the final classifier is the sign of the discriminant function, that is $f_{final}(x) = \text{sign}[g(x)]$

Idea Behind Ada Boost

- Algorithm is iterative
- Maintains distribution of weights over the training examples
- Initially distribution of weights is uniform
- At successive iterations, the weight of misclassified examples is increased, forcing the weak learner to focus on the hard examples in the training set

More Comments on Ada Boost

- Ada boost is very simple to implement, provided you have an implementation of a “weak learner”
- Will work as long as the “basic” classifier $f_t(x)$ is at least slightly better than random
- Can be applied to boost any classifier, not necessarily weak

Ada Boost *(slightly modified from the original version)*

- $d(x)$ is the distribution of weights over the N training points $\sum d(x_i)=1$
- Initially assign uniform weights $d_0(x_i) = 1/N$ for all x_i
- At each iteration t :
  - Find best weak classifier $f_t(x)$ using weights $d_t(x)$
  - Compute the error rate ε_t as

    $\varepsilon_t = \sum_{i=1\ldots N} d_t(x_i) \cdot I[y_i \neq f_t(x_i)]$
  - assign weight α_t the classifier f_t's in the final hypothesis

    $$\alpha_t = \log((1 - \varepsilon_t)/\varepsilon_t)$$
  - For each x_i , $d_{t+1}(x_i) = d_t(x_i) \cdot \exp[\alpha_t \cdot I(y_i \neq f_t(x_i))]$
  - Normalize $d_{t+1}(x_i)$ so that $\sum_{i=1} d_{t+1}(x_i) = 1$
- $f_{FINAL}(x) = \text{sign}\left[\sum \alpha_t f_t(x) \right]$

Ada Boost

- At each iteration t :
  - Find best weak classifier $f_t(x)$ using weights $d_t(x)$
  - Compute ε_t the error rate as

    $\varepsilon_t = \sum d_t(x_i) \cdot I[y_i \neq f_t(x_i)]$
  - assign weight α_t the classifier f_t's in the final hypothesis

    $$\alpha_t = \log((1 - \varepsilon_t)/\varepsilon_t)$$
  - For each x_i, $d_{t+1}(x_i) = d_t(x_i) \cdot \exp[\alpha_t \cdot I(y_i \neq f_t(x_i))]$
  - Normalize $d_{t+1}(x_i)$ so that $\sum_{t+1} d(x_i) = 1$
- $f_{FINAL}(x) = \text{sign}[\sum \alpha_t f_t(x)]$

- If the classifier does not take weighted samples, this step can be achieved by sampling from the training samples according to the distribution $d_t(x)$

Ada Boost

- At each iteration t :
  - Find best weak classifier $f_t(x)$ using weights $d_t(x)$
  - Compute ε_t the error rate as

$$\varepsilon_t = \sum d_t(x_i) \cdot \mathrm{I}[y_i \neq f_t(x_i)]$$

  - assign weight α_t the classifier f_t's in the final hypothesis

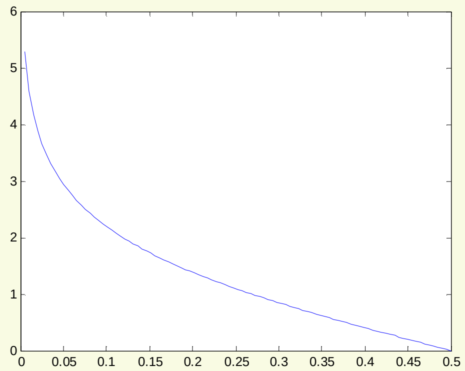
$$\alpha_t = \log((1 - \varepsilon_t)/\varepsilon_t)$$

  - For each x_i, $d_{t+1}(x_i) = d_t(x_i) \cdot \exp[\alpha_t \cdot \mathrm{I}(y_i \neq f_t(x_i))]$
  - Normalize $d_{t+1}(x_i)$ so that $\sum d_{t+1}(x_i) = 1$
- $f_{FINAL}(x) = \mathrm{sign}[\sum \alpha_t f_t(x)]$

- Since the weak classifier is better than random, we expect $\varepsilon_t < 1/2$

Ada Boost

- At each iteration t :
  - Find best weak classifier $f_t(x)$ using weights $d_t(x)$
  - Compute ε_t the error rate as
    $\varepsilon_t = \sum d(x_i) \cdot I(y_i \neq f_t(x_i))$
  - assign weight α_t the classifier f_t's in the final hypothesis
    $$\alpha_t = \log((1 - \varepsilon_t)/\varepsilon_t)$$
  - For each x_i, $d_{t+1}(x_i) = d_t(x_i) \cdot \exp[\alpha_t \cdot I(y_i \neq f_t(x_i))]$
  - Normalize $d_{t+1}(x_i)$ so that $\sum d_{t+1}(x_i) = 1$
- $f_{FINAL}(x) = \text{sign}[\sum \alpha_t f_t(x)]$



- Recall that $\varepsilon_t < ½$
- Thus $(1- \varepsilon_t)/ \varepsilon_t > 1 \Rightarrow \alpha_t > 0$
- The smaller is ε_t, the larger is α_t, and thus the more importance (weight) classifier $f_t(x)$ gets in the final classifier
  $$f_{FINAL}(x) = \text{sign}[\sum \alpha_t f_t(x)]$$

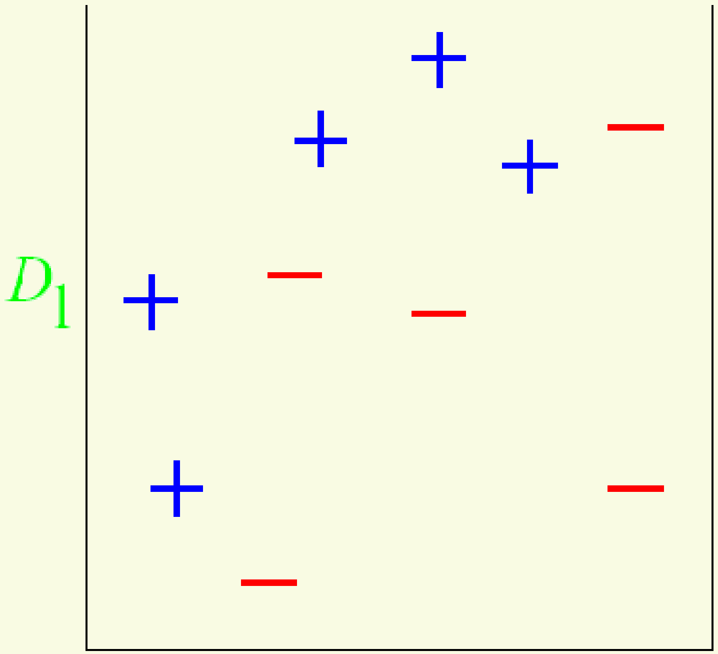
Ada Boost

- At each iteration t :
  - Find best weak classifier $f_t(x)$ using weights $d_t(x)$
  - Compute ε_t the error rate as
    $\varepsilon_t = \sum d_t(x_i) \cdot I(y_i \neq f_t(x_i))$
  - assign weight α_t the classifier f_t's in the final hypothesis
    $$\alpha_t = \log((1 - \varepsilon_t)/\varepsilon_t)$$
  - For each x_i, $d_{t+1}(x_i) = d_t(x_i) \cdot \exp[\alpha_t \cdot I(y_i \neq f_t(x_i))]$
  - Normalize $d_{t+1}(x_i)$ so that $\sum d_{t+1}(x_i) = 1$
- $f_{FINAL}(x) = \text{sign}\left[\sum \alpha_t f_t(x)\right]$

- Weight of misclassified examples is increased and the new $d_{t+1}(x_i)$'s are normalized to be a distribution again

AdaBoost Example

from “**A Tutorial on Boosting” by** Yoav Freund and Rob Schapire

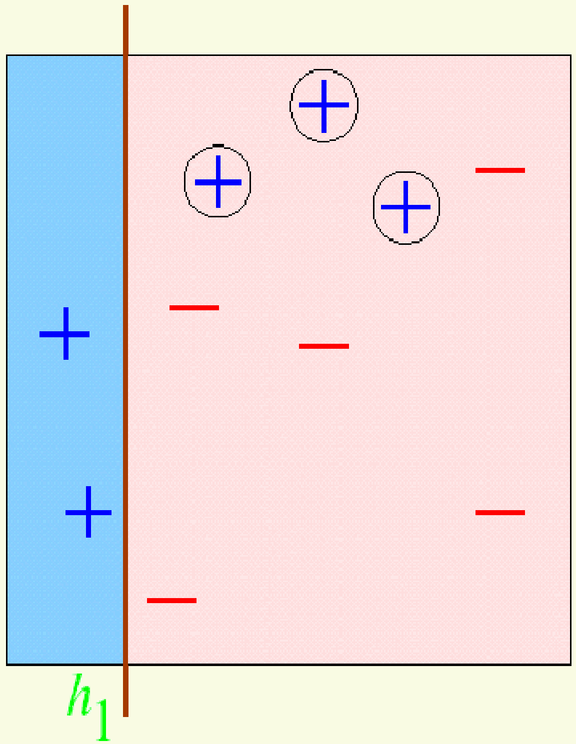


Original Training set : equal weights to all training samples

Note: in the following slides, $h_t(x)$ is used instead of $f_t(x)$, and D instead of d

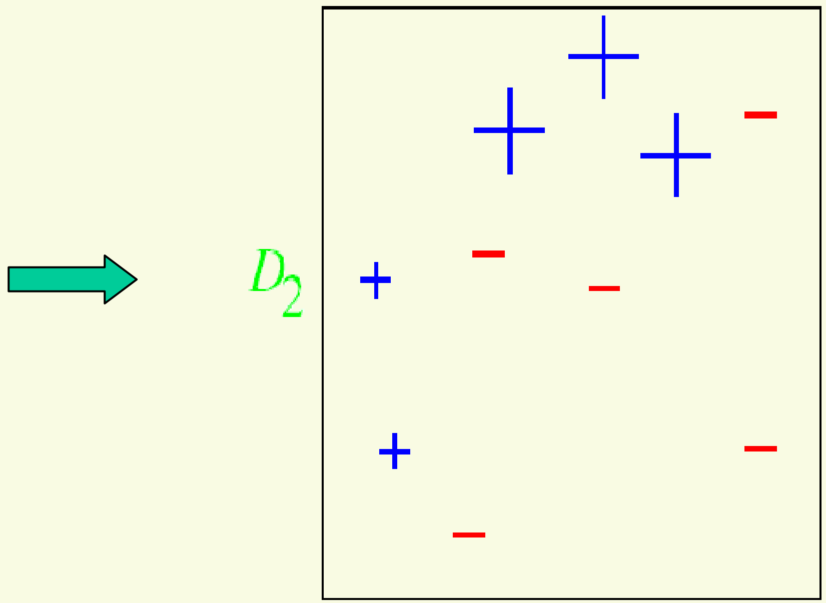
AdaBoost Example

ROUND 1



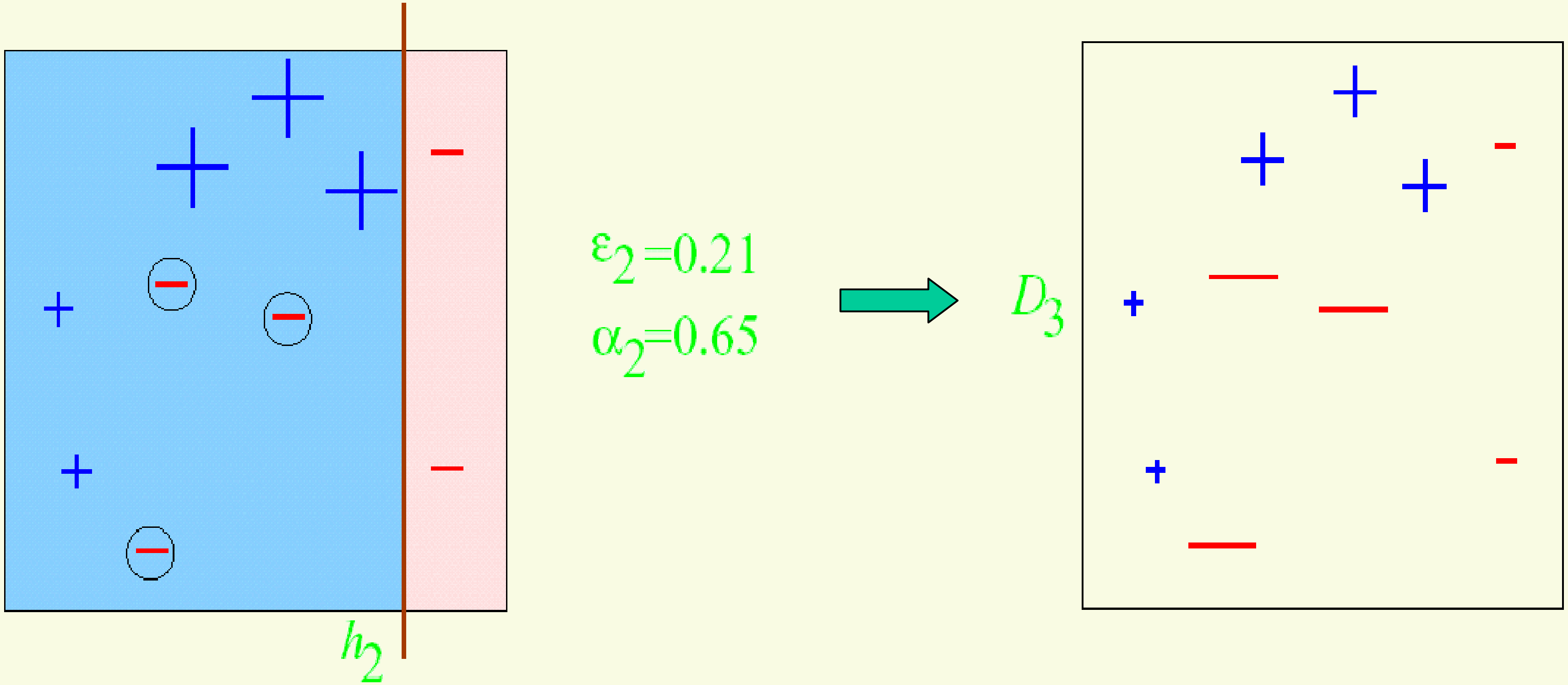
$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$



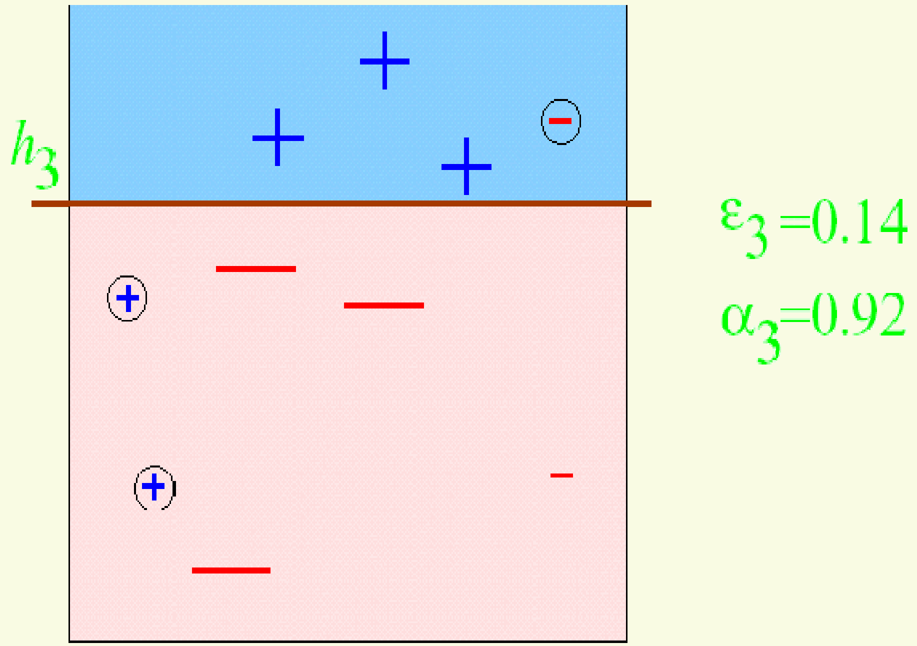
AdaBoost Example

ROUND 2

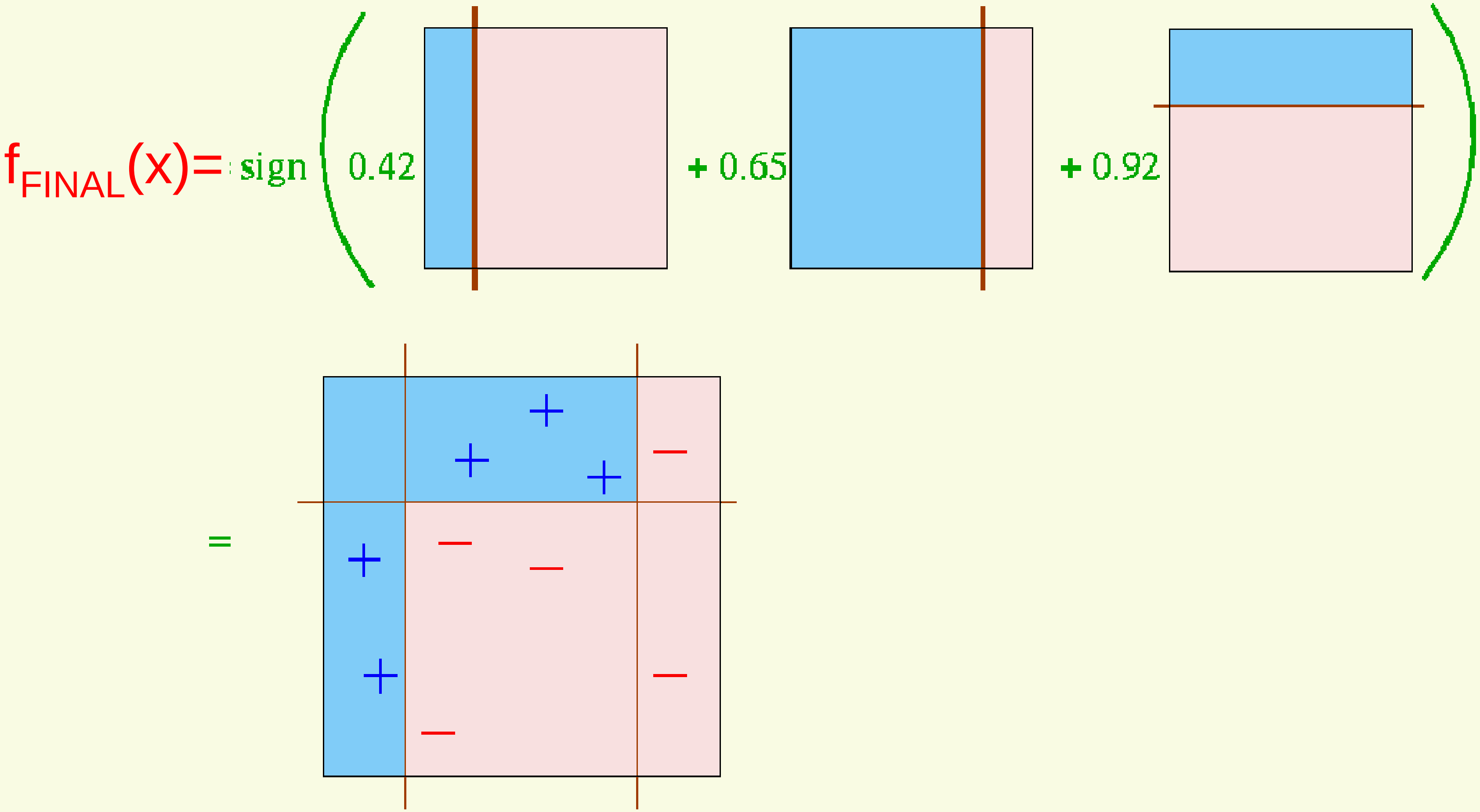


AdaBoost Example

ROUND 3



AdaBoost Example



AdaBoost Comments

- It can be shown that the training error drops exponentially fast, if each weak classifier is slightly better than random

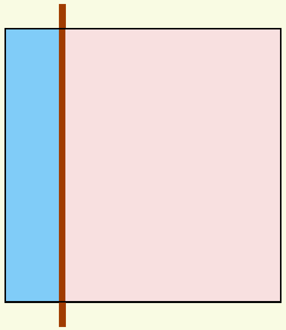
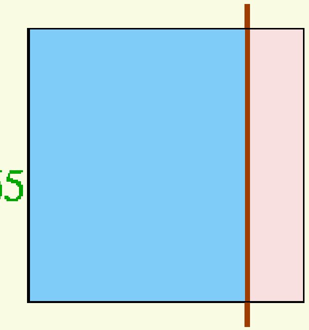
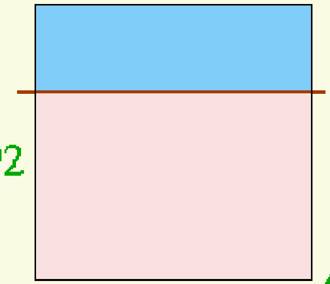
$$\boldsymbol{Err_{train}} \leq \mathbf{exp}\left(-2\sum_t \gamma_t^2\right)$$

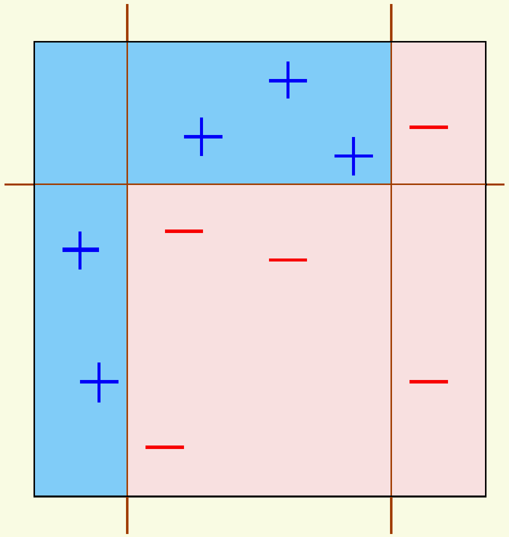
- Here $\gamma_t = \varepsilon_t - 1/2$, where ε_t is classification error at round $\boldsymbol{t}$ (weak classifier f_t)

AdaBoost Comments

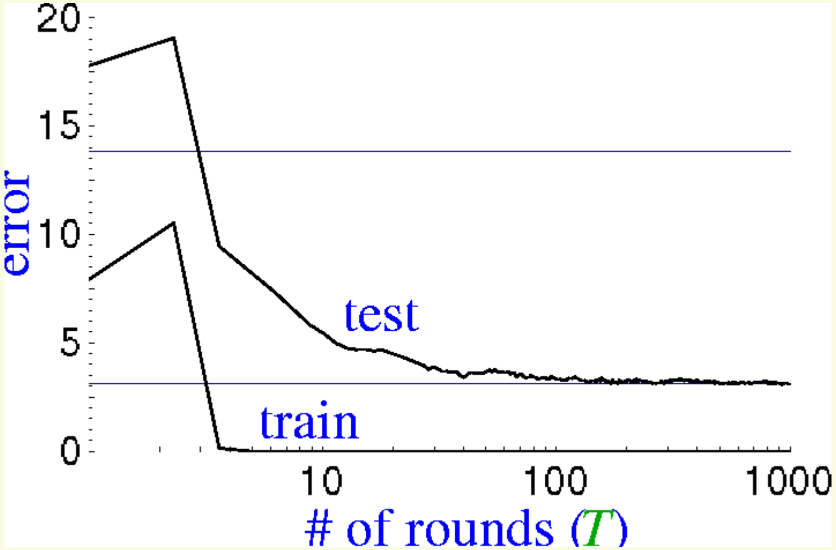
- But we are really interested in the generalization properties of $f_{FINAL}(x)$, not the training error
- AdaBoost was shown to have excellent generalization properties in practice, in fact in the beginning researchers thought it does not overfit data
  - It turns out it does overfit data eventually, if you run it really long
- It can be shown that boosting “aggressively” increases the margins of training examples, as iterations proceed
  - margins continue to increase even when training error reaches zero
  - Helps to explain empirically observed phenomena: test error continues to drop even after training error reaches zero

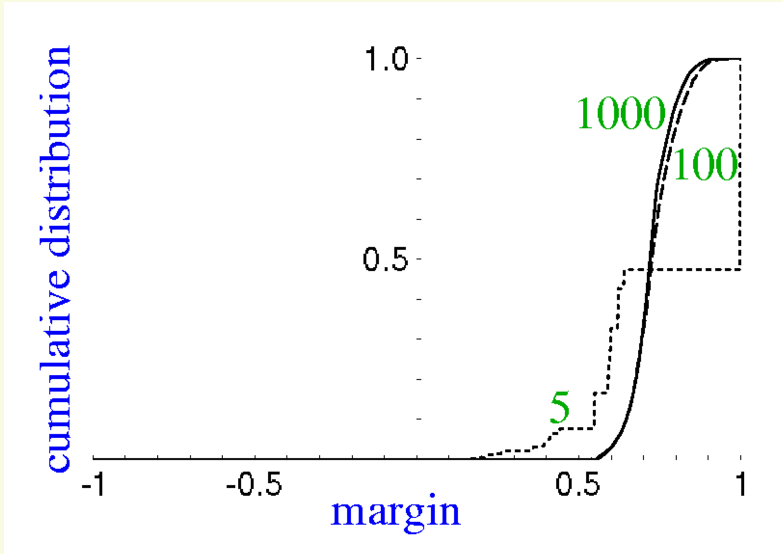
AdaBoost Example

$f_{FINAL}(x) = \text{sign}\Big($ 0.42  + 0.65  + 0.92  $\Big)$

= 

The Margin Distribution





epoch	5	100	1000
training error	0.0	0.0	0.0
test error	8.4	3.3	3.1
%margins≤0.5	7.7	0.0	0.0
Minimum margin	0.14	0.52	0.55

Practical Advantages of AdaBoost

- fast
- simple
- Has only one parameter to tune (T)
- flexible: can be combined with any classifier
- provably effective (assuming weak learner)
  - shift in mind set: goal now is merely to find hypotheses that are better than random guessing
- finds outliers
  - The hardest examples are frequently the “outliers”

Caveats

- performance depends on data & weak learner
- AdaBoost can fail if
  - weak hypothesis too complex (overfitting)
  - weak hypothesis too weak ($\gamma_t \rightarrow 0$ too quickly),
    - underfitting
    - Low margins $\rightarrow$ overfitting
- empirically, AdaBoost seems especially susceptible to noise