Machine Learning

Lecture 4: Ensemble Methods

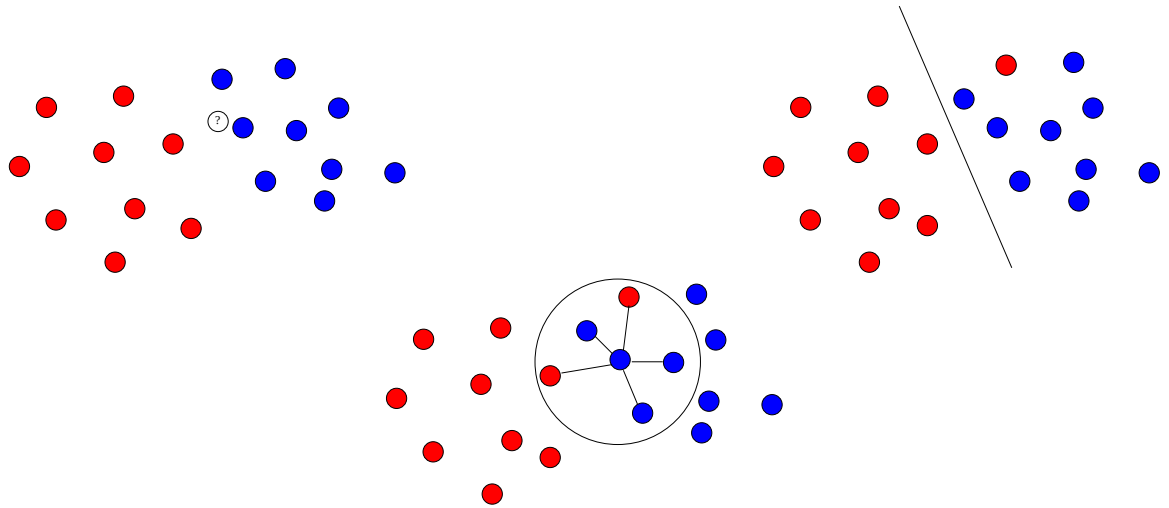
Marc Sebban & Amaury Habrard

Hubert Curien lab, UMR CNRS 5516
University of Jean Monnet Saint-Étienne (France)

Academic year 2016-2017

Outline

Many classifiers can be induced from the same task



- Different learning algorithms (*e.g.* k-NNs, linear separator, decision trees, SVMs, etc.).
- Different hyperparameters (*e.g.* number of neighbors k).
- Different (randomly drawn) training sets S.
- Different representations of the same learning set.

Select the best one or combine them?

Model selection versus ensemble methods

Rather than selecting the best model (w.r.t. some cross-validation procedure), why not try combining the whole set of classifiers and taking advantage of their diversity?

$\rightarrow$ **Ensemble methods**

Ensemble Methods

Definition

Ensemble methods are learning algorithms that construct a set of classifiers $h_1, \ldots, h_T$ whose individual decisions are combined in some way to classify new examples.

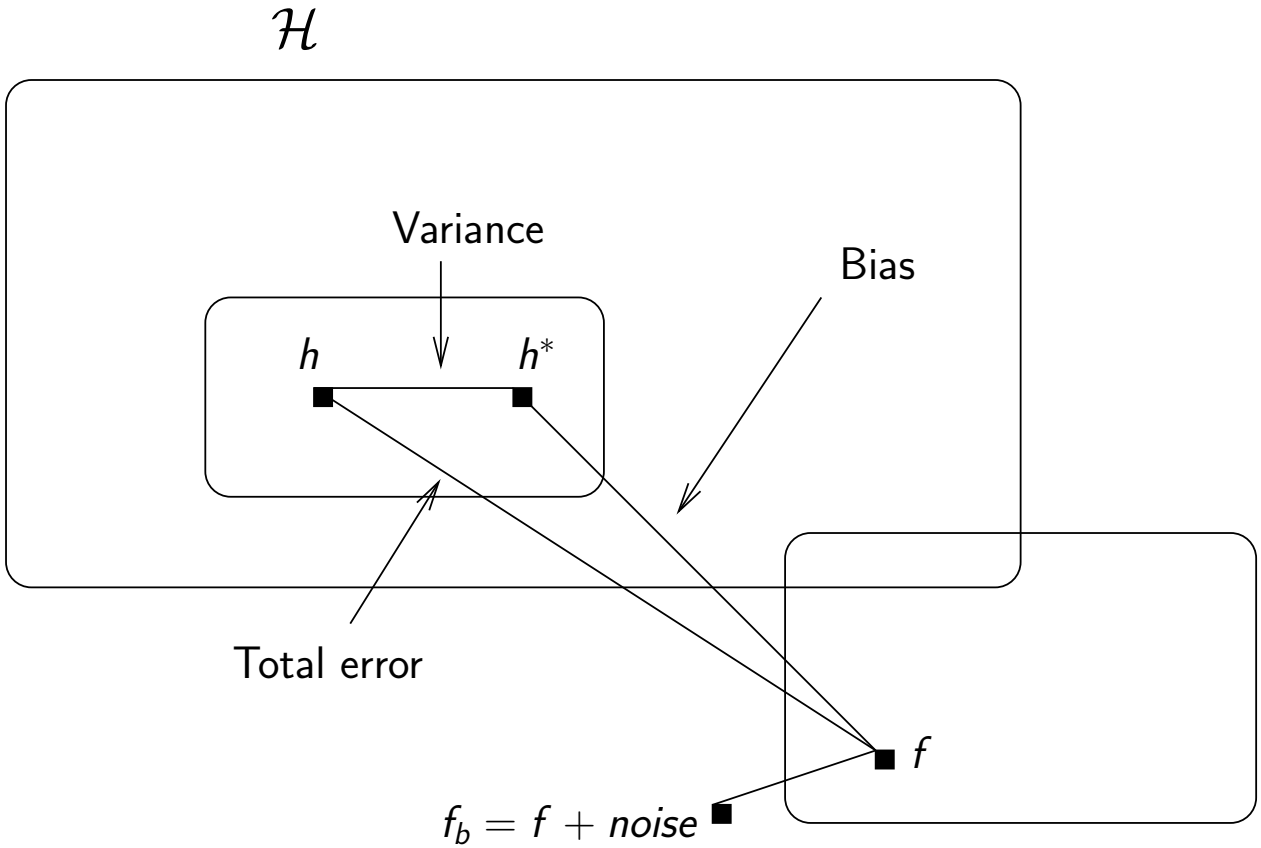
Necessary and sufficient conditions for an ensemble of classifiers to be efficient:

- the individual classifiers (or hypotheses) are accurate, *i.e.* they have an error rate of better than random guessing.
- the classifiers are diverse, *i.e.* they make different errors on new data points.

Question

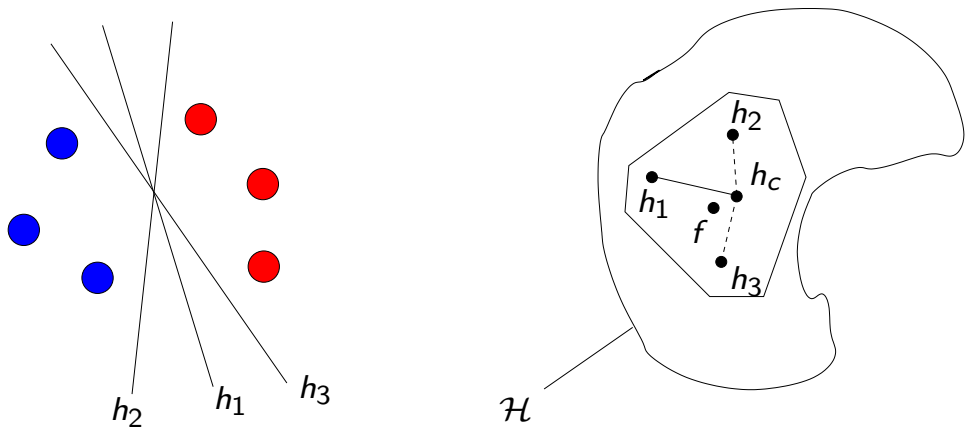
Is it possible to construct (theoretically) good ensembles?

Bias/Variance trade-off



Limitations of a single classifier

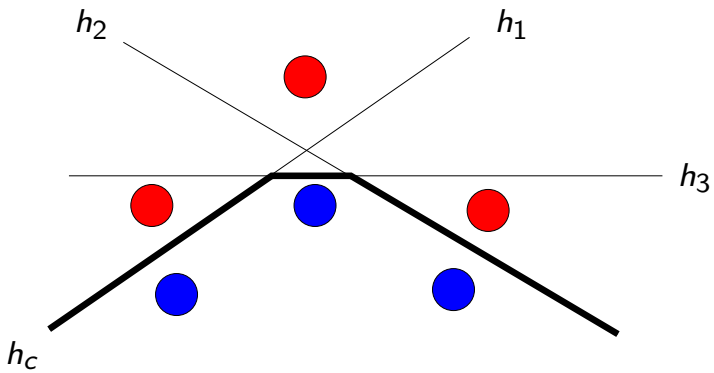
Statistical problem (variance): Without sufficient data, the learning algorithm can find many different hypotheses in $\mathcal{H}$ that all give the same empirical accuracy on S.

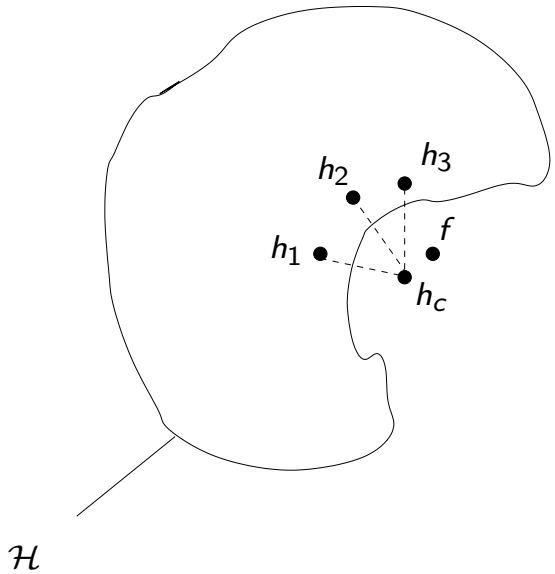


By constructing an ensemble h_c out of all of these accurate classifiers, the algorithm can "average" their votes and reduce the risk of choosing the wrong

Limitations of a single classifier

Representational problem (bias): In most applications of machine learning, the true function f cannot be represented by any of the hypotheses in $\mathcal{H}$.

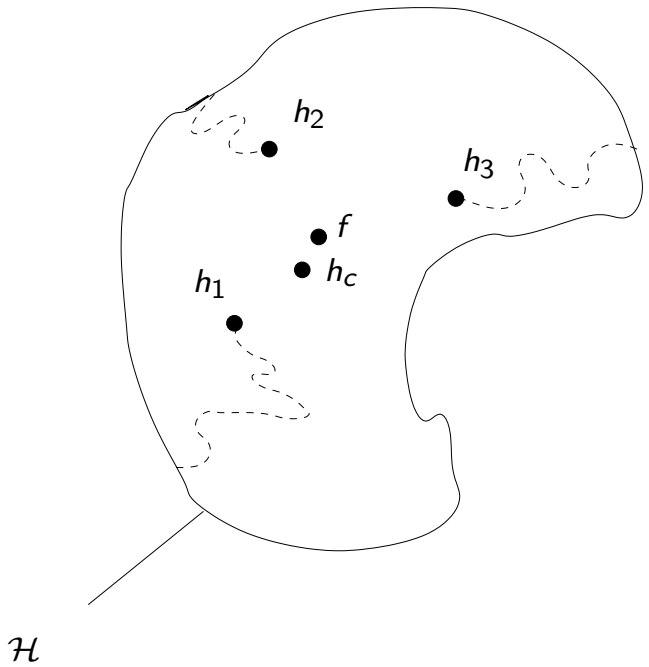




By forming weighted sums of hypotheses drawn from $\mathcal{H}$, it may be possible to expand the space of representable functions.

Limitations of a single classifier

Computational problem: Many learning algorithms work by performing some form of local search that may get stuck in local optima. An ensemble constructed by running the local search from many different starting points may provide a better approximation to the unknown function.



Ensemble Methods

There are two main categories of ensemble methods which depend on the origin of the diversity brought by the hypotheses.

- **Heterogeneous ensemble methods**: several classifiers $h_1, \ldots h_T$ are generated by applying different learning algorithms $L_1, \ldots, L_T$ to a single training dataset, *i.e.* to a constant distribution D of the training data.

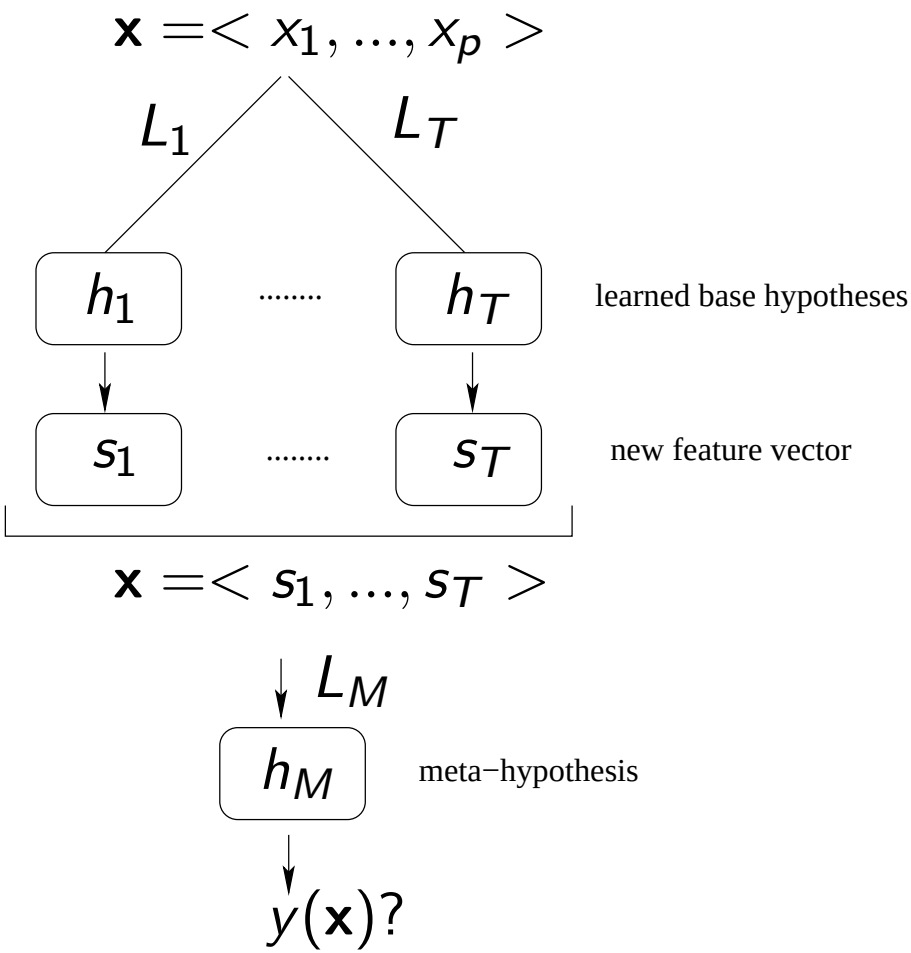
- **Homogeneous ensemble methods**: several hypotheses $h_1, \ldots h_T$ are generated from a single learning algorithm L. The diversity of the hypotheses is obtained by modifying the statistical distribution D_t of the training examples used to build h_t.

Heterogeneous ensemble methods

The diversity comes from the learning algorithms

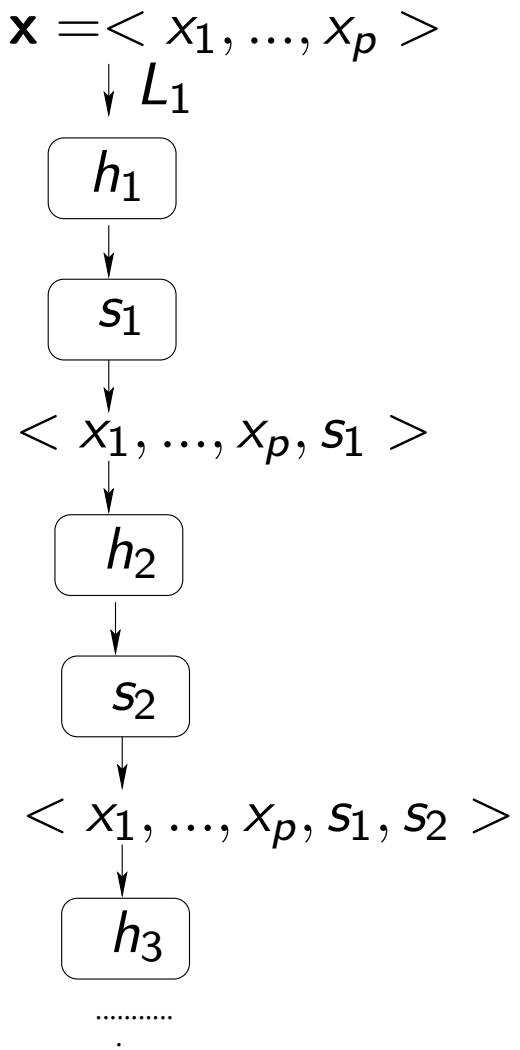
- **Stacking**
- **Cascade Generalization**

Stacking



1. Learn T hypotheses $h_1, \ldots, h_T$ with T different learning algorithms $L_1, \ldots, L_T$.
2. The decisions (scores) of $h_1, \ldots, h_T$ on $\mathbf{x}$ are seen as new features
3. Learn a meta hypothesis in this new T dimensional space.

Cascade Generalization



1. Learn a hypothesis h_1 with a learning algorithm L_1. Classify the learning examples with h_1.
2. Learn a hypothesis h_2 with a learning algorithm L_2 from the original features and the label (or the score) predicted at the previous step. Classify the learning examples with h_2.
3. Repeat the process.

Homogeneous ensemble methods

- The diversity comes from the training examples.
- We consider the problem of combining classifiers built from different sets of training data.
- The family of hypothesis is usually kept unchanged (i.e. same learning algorithm).
- Homogeneous ensemble methods:
  - **Bagging**
  - **Random Forests**
  - **Boosting**

Bagging

Bagging

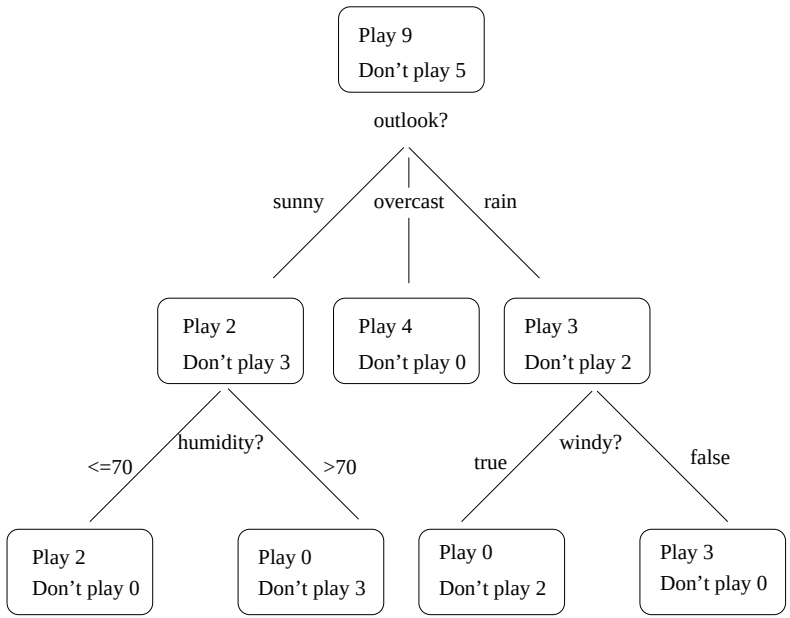
Input: *A learning sample* $S = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_m}, y_m)\}$
Input: *A total number* T *of bagging rounds*
Input: *A learning algorithm* L *returning a binary classifier*
Output: *A combined classifier*
for all t *from* 1 *to* T **do**
  $S_t = Resample(S)$ // *Randomly sample* S *with replacement;*
  $h_t(\mathbf{x}) = L(S_t)$ // *Build a classifier on* S_t *using learning algorithm* L*;*
Return H_T **such that**
  $H_T(\mathbf{x}) = sign\left(\sum_t h_t(\mathbf{x})\right)$

Random Forests

Definition

A decision tree is a tree that can be learned by splitting S into subsets based on a feature value test. This process is repeated on each derived subset in a recursive manner.

Example: the decision to play or not play tennis based on weather forecast.



Random Forests

Definition

Decision Trees + Random Feature Selection + Bagging = Random Forests

- **Aim**: generate diversity in decision trees.
- The general approach is like bagging:
  - build model on successive resampling (with replacement) of S;
  - make a majority vote to form the combined classifier.
- Decision trees are built with no pruning.
- While growing the tree, a random subset of F features is selected from the p original ones (typical values are $F = \lceil log_2 p \rceil$ or $F = \lceil \sqrt{p} \rceil$).

Introduction to boosting

Robert Schapire

Boosting

Let us start from an example....

- **Aim**: A horse-racing gambler, hoping to maximize his winnings, decides to create a computer program that will accurately predict the winner of a horse race.
- **Strategy 1**: ask a highly successful expert gambler to explain his betting strategy. Not surprisingly, the expert is unable to articulate a large set of rules for selecting a horse.
- **Strategy 2**: But, when presented with the data for a specific set of races, he is able to express some rules such as:
  - h_1 : ``Bet on the horse that has recently won the most races''.
  - h_2 : ``Bet on the horse with the most favored odds''.

Boosting

In order to use these rules to maximum advantage, there are two problems faced by the gambler:

1. How to choose the collections of races presented to the expert so as to extract rules that will be the most useful?
2. Once we have collected many rules, how to combine them into a single, highly accurate prediction rule?

Solutions:

1. If the combination is not weighted and the learning examples are randomly selected $\rightarrow$ **Bagging**.
2. If the combination is weighted and the selection of the learning examples is driven by a "hard" examples $\rightarrow$ **Boosting**.

Strong vs Weak Learnability

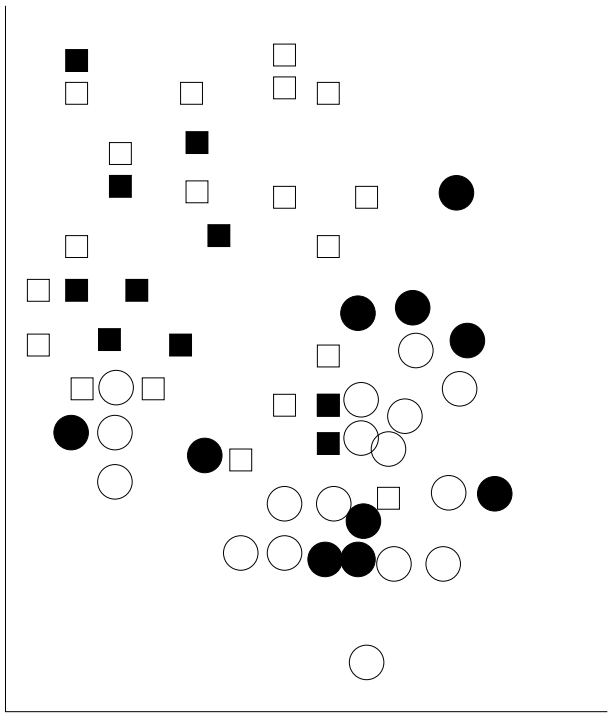
Definition

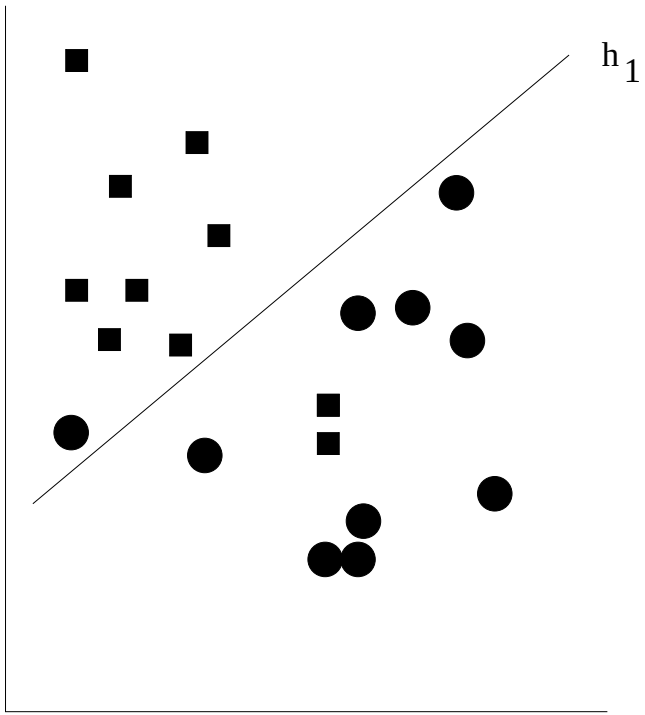
Boosting is a general method for improving (under some constraints) the accuracy of any given learning algorithm.

Boosting combines *weak* hypotheses (*i.e.* just better than a random guessing) into a *strong* hypothesis (from a PAC theory point of view).

First boosting algorithm (1/4)

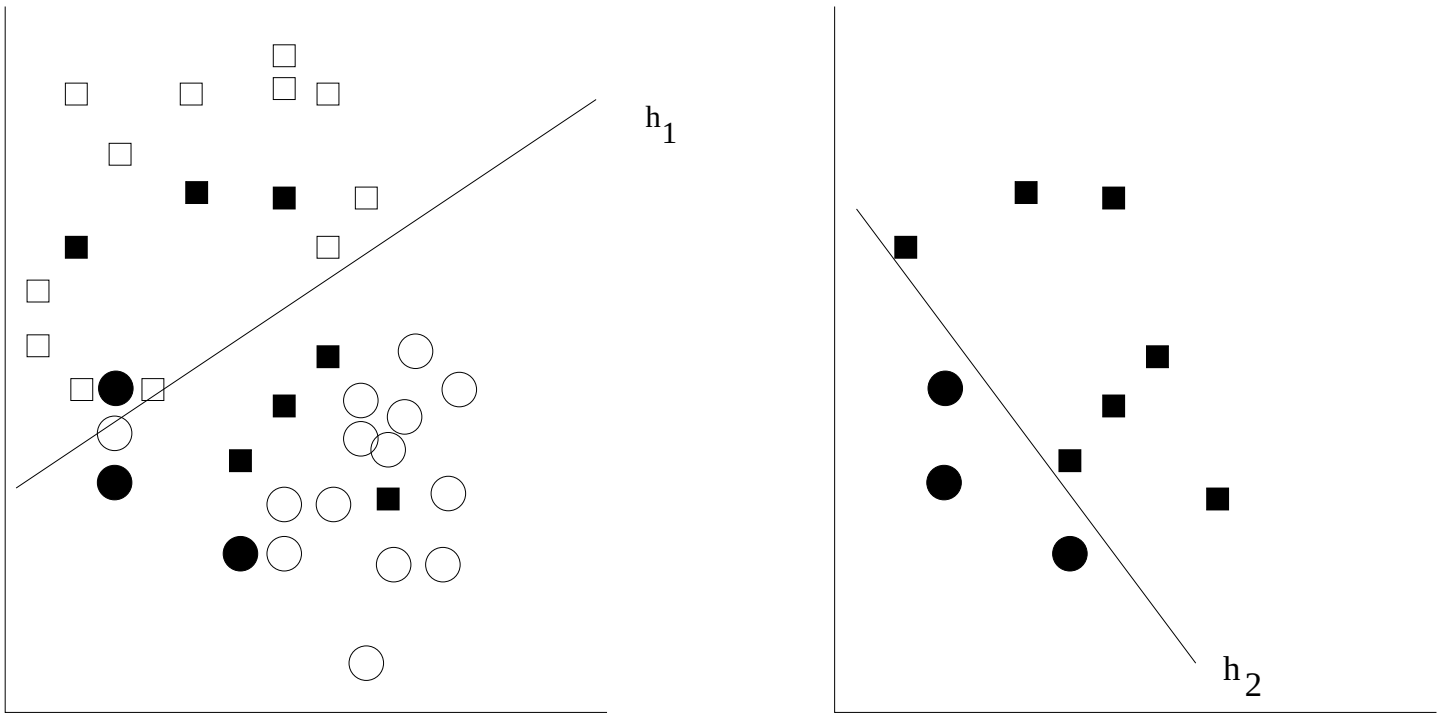
Step 1: Extract from S a learning sample S_1. Use a learning algorithm L to produce a first hypothesis h_1.





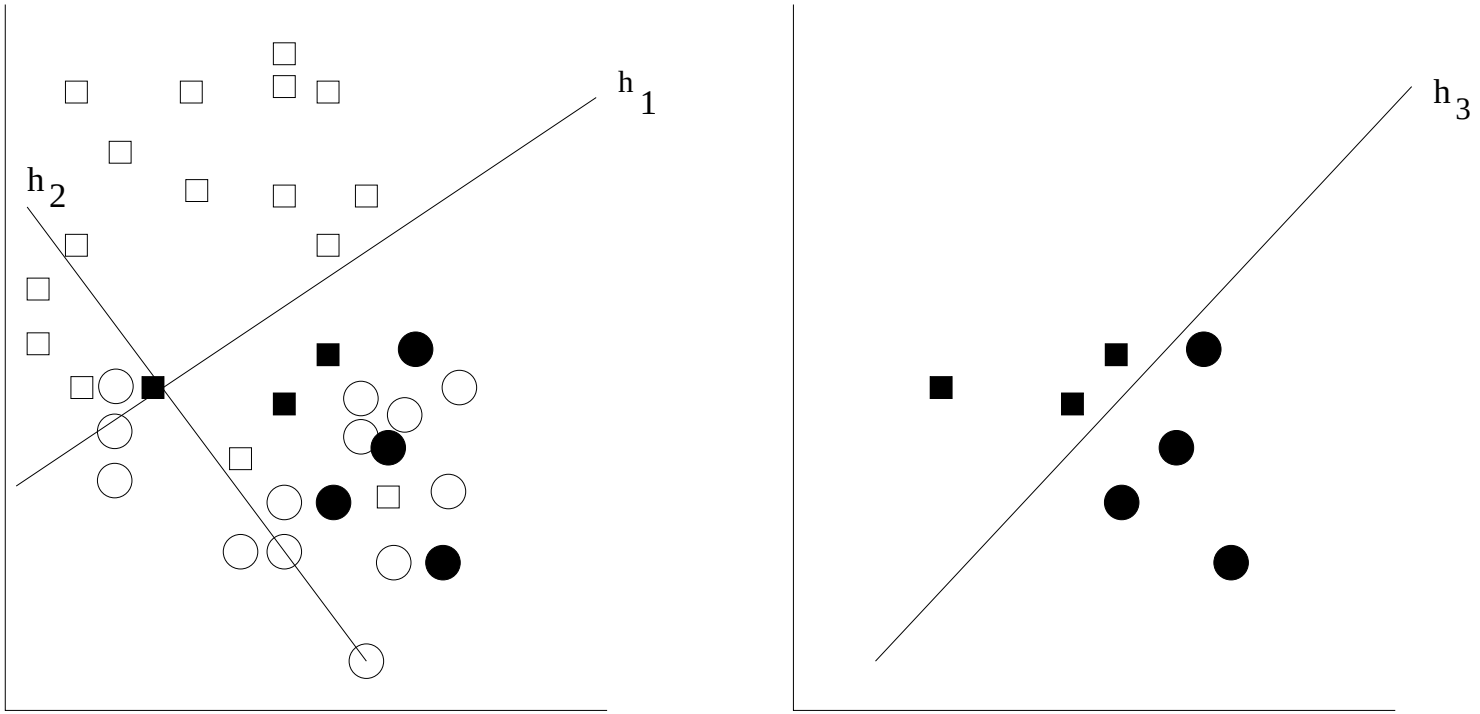
First boosting algorithm (2/4)

Step 2: Generate a second learning sample S_2, in which an instance has a roughly equal chance of being correctly or incorrectly classified by h_1. L is used again to infer a new hypothesis h_2.

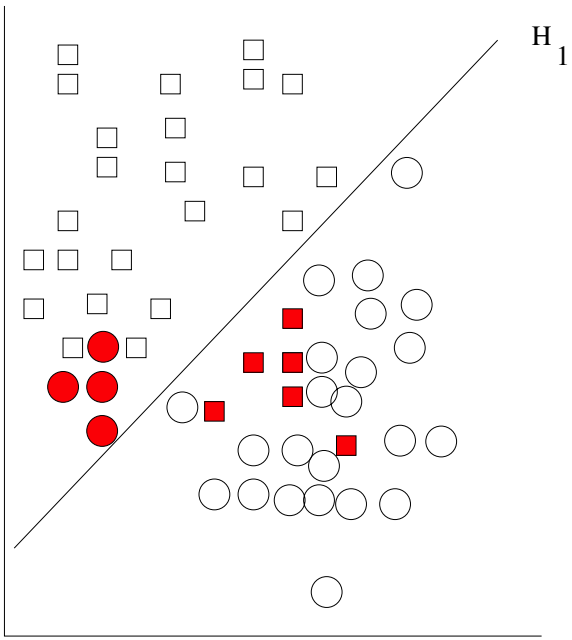


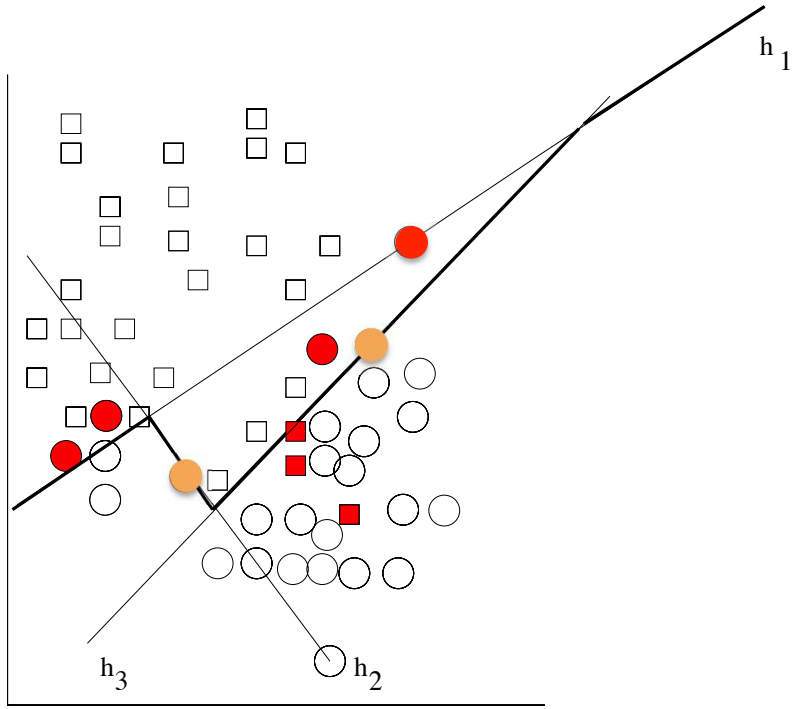
First boosting algorithm (3/4)

Step 3: Generate a third learning sample S_3 by removing from S the instances on which h_1 and h_2 agree. Once again, L is used to induce a third hypothesis h_3.



First boosting algorithm (4/4)





The final hypothesis takes the "majority vote" of h_1, h_2 and h_3.

Adaboost

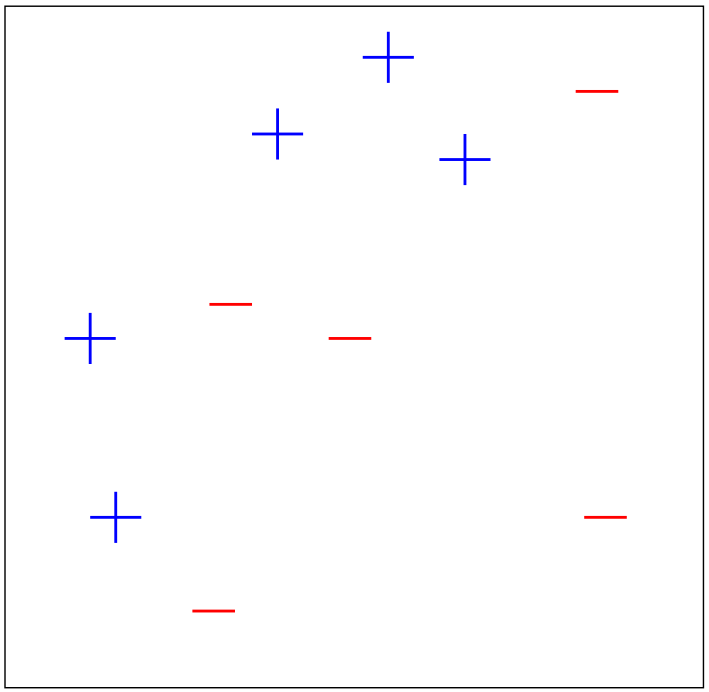
Adaboost

Input: *A learning sample* S, *a number of iterations* T, *a weak learner* L
Output: *A global hypothesis* H_T
for all i *from* 1 *to* m **do**
  $D_1(\mathbf{x_i}) = 1/m$;
for all t *from* 1 *to* T **do**
  $h_t = \mathrm{L}(S, \mathbf{D}_t)$;
  $\hat{\epsilon}_t = \sum_{\mathbf{x_i} \ t.q. \ y_i \neq h_t(\mathbf{x_i})} D_t(\mathbf{x_i})$;
  $\alpha_t = \frac{1}{2} \ln \frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t}$;
  for all i *from* 1 *to* m **do**
    $D_{t+1}(\mathbf{x_i}) = D_t(\mathbf{x_i}) \exp\left(-\alpha_t y_i h_t(\mathbf{x_i})\right) / Z_t$;
    /* Z_t *is a normalization coefficient*/
$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$;
Return H_T **such that**
  $H_T(\mathbf{x}) = sign\left(f(\mathbf{x})\right)$

Toy example (1/5)

Learning sample S

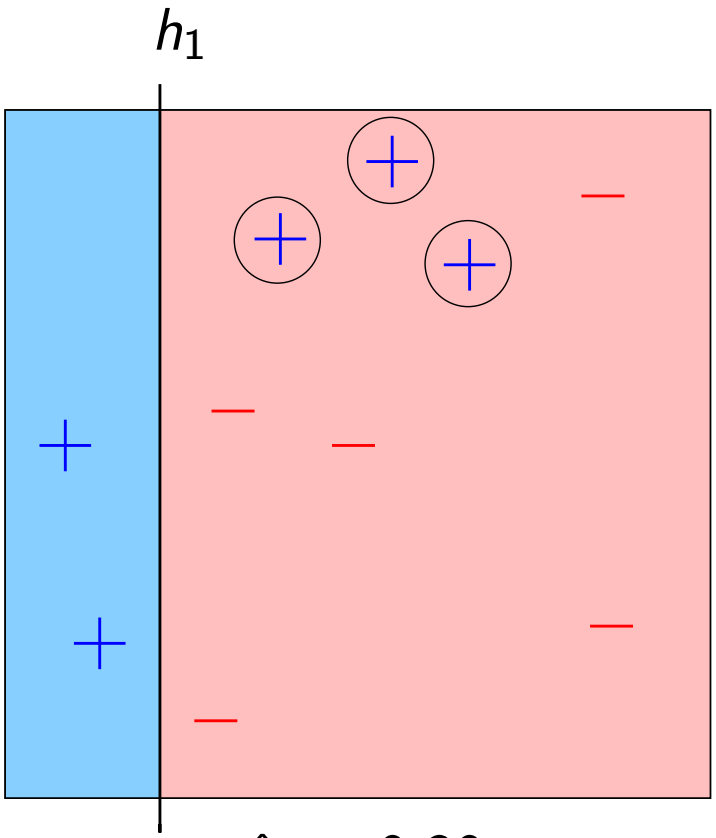
Distribution D_1



Weak Hypotheses: linear separators parallel to the axis

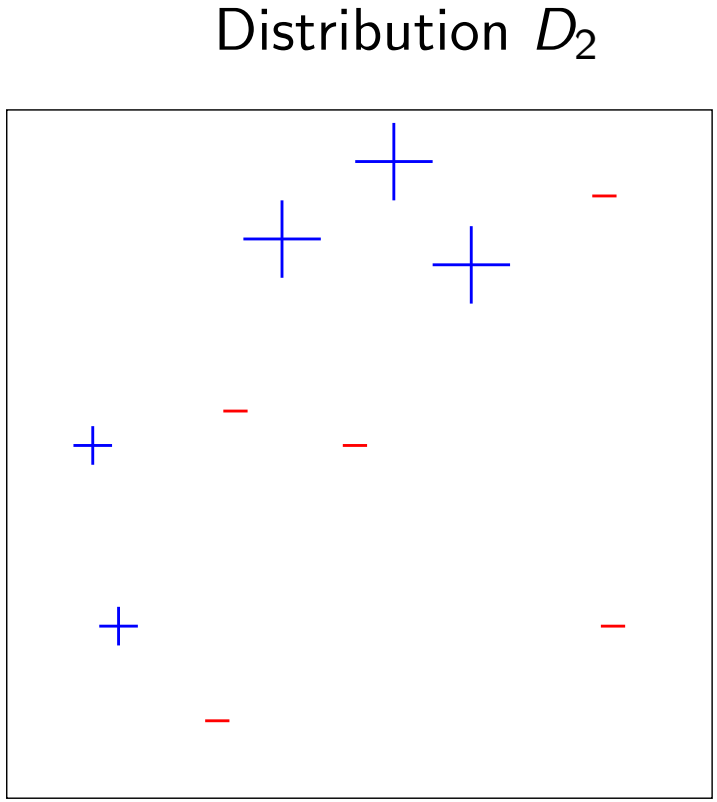
Toy example (2/5)

Step 1



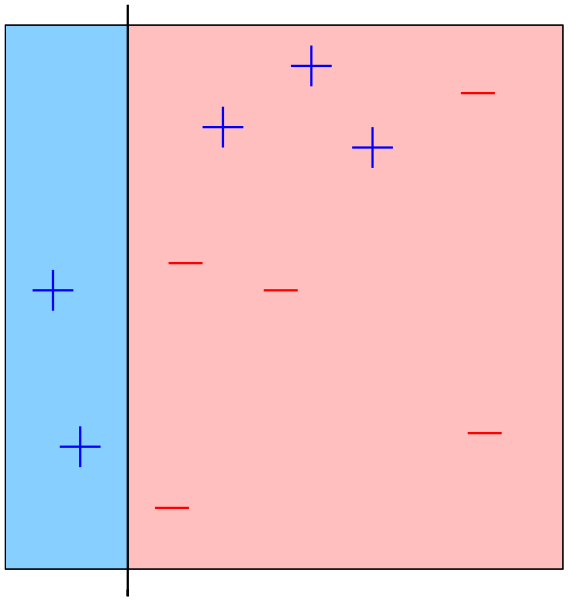
$\hat{\epsilon}_1 = 0.30$

$\alpha_1 = 0.42$

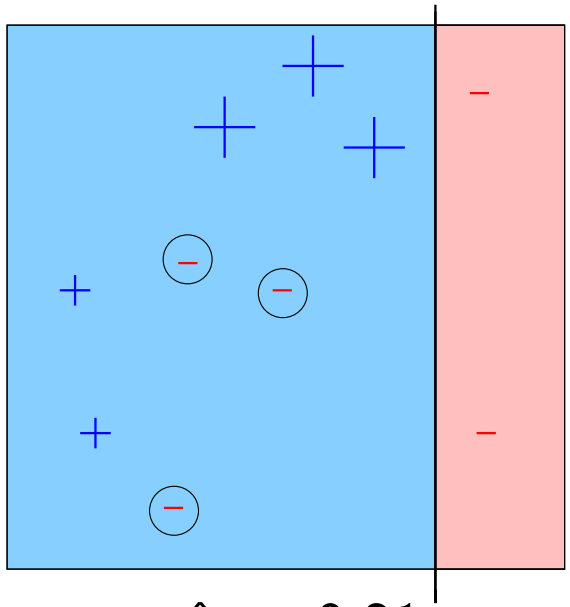


Toy example (3/5)

Step 2

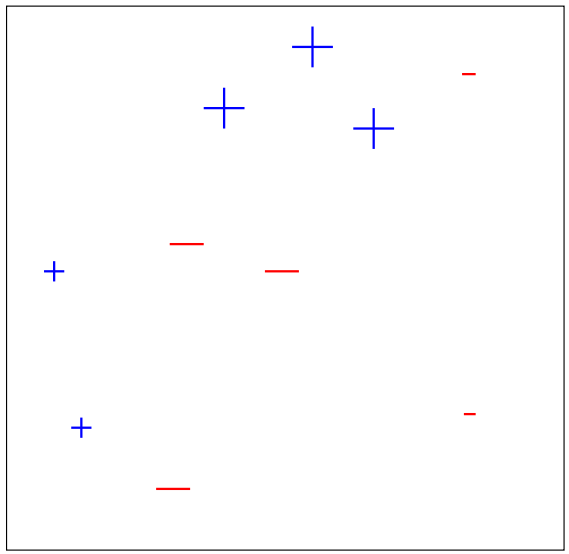


h_2



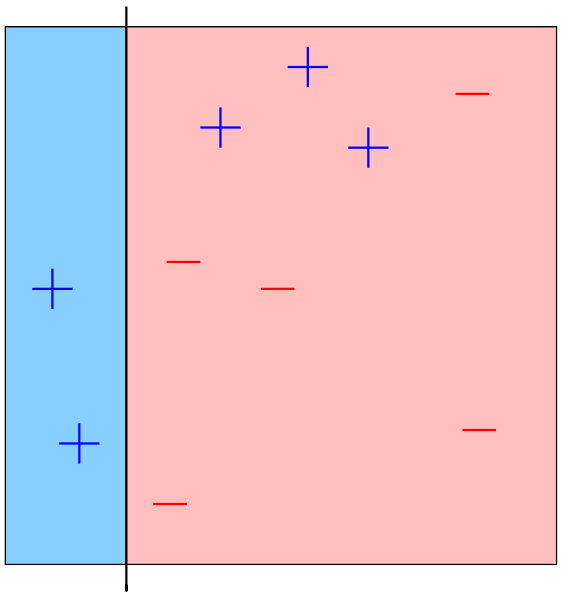
$\hat{\epsilon}_2 = 0.21$
$\alpha_2 = 0.65$

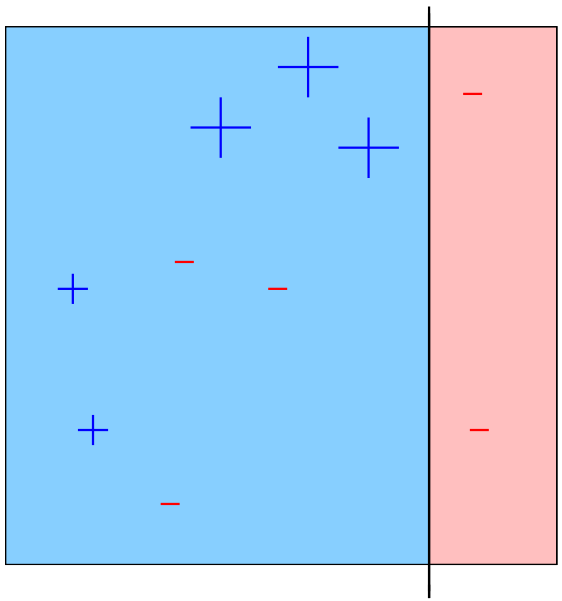
Distribution D_3

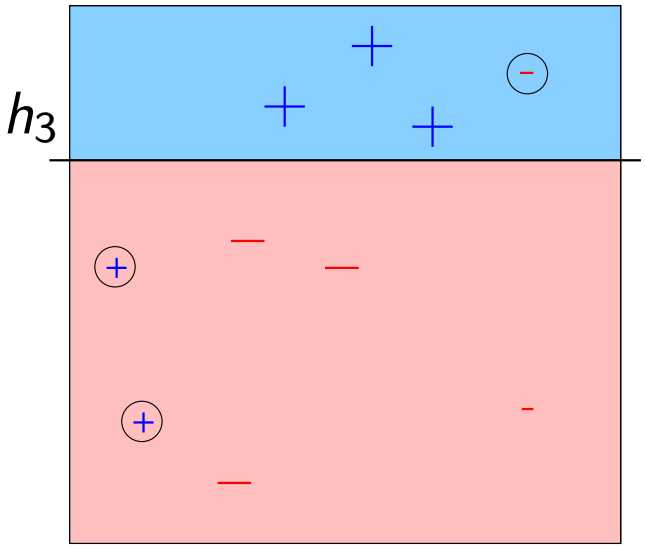


Toy example (4/5)

Step 3



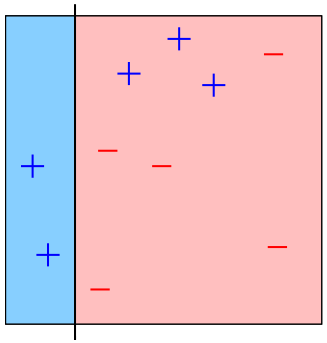
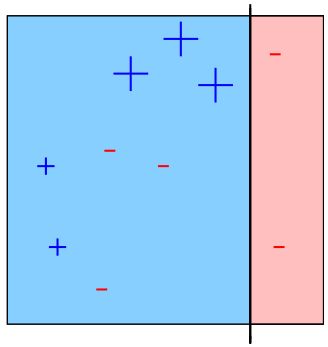
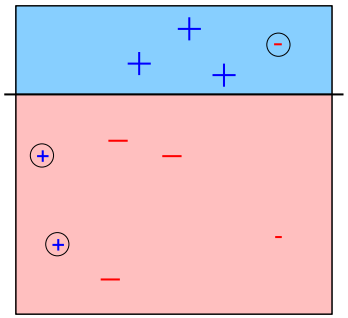


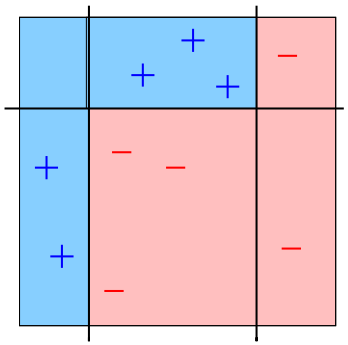


$\hat{\epsilon}_3 = 0.14$
$\alpha_3 = 0.92$

Toy example (5/5)

Final Classifier

$H_{final} = \text{sign}(\quad 0.42\times$  $+0.65\times$  $+0.92\times$  $)$

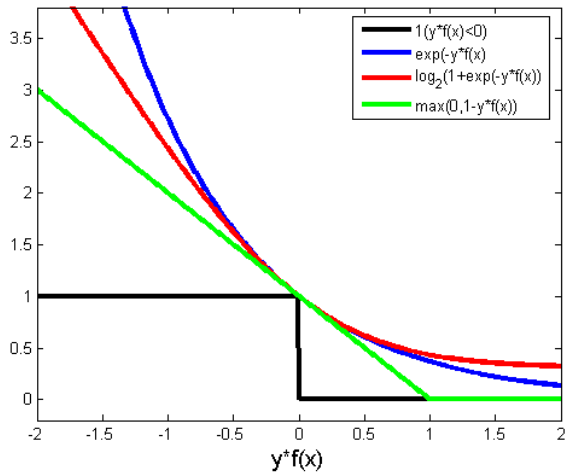
$=$ 

Theoretical results on the empirical risk

Theorem 1

Upper bound on the empirical error of H_T

$$\hat{\epsilon}_{H_T} = \frac{1}{m}\sum_i [H(\mathbf{x_i}) \neq y_i] \leq \frac{1}{m}\sum_i exp(-y_i f(\mathbf{x_i})) = \prod_t Z_t$$



This theorem means that to minimize the empirical error, we have to minimize the product of the Z_t.

The previous theorem is proven in two steps.

Step 1: $\hat{\epsilon}_{H_T} \leq \frac{1}{m} \sum_i exp(-y_i f(\mathbf{x_i}))$

Proof.

$$\begin{aligned}
\hat{\epsilon}_{H_T} &= \frac{1}{m} \sum_i [H(\mathbf{x_i}) \neq y_i] \\
&= \frac{1}{m} \sum_i [y_i f(\mathbf{x_i}) < 0] \\
&= \frac{1}{m} \sum_i [-y_i f(\mathbf{x_i}) > 0] \\
&= \frac{1}{m} \sum_i [exp(-y_i f(\mathbf{x_i})) > 1] \\
&\leq \frac{1}{m} \sum_i exp(-y_i f(\mathbf{x_i}))
\end{aligned}$$

Theoretical results on the empirical risk

Step 2: $\frac{1}{m}\sum_i exp(-y_i f(\mathbf{x_i})) = \prod_t Z_t$. To simplify, let us replace $\mathbf{x_i}$ by i.

Proof.

$$
\begin{aligned}
D_{T+1}(i) &= \frac{D_T(i)exp(-\alpha_T y_i h_T(i))}{Z_T} \\
&= \frac{\frac{D_{T-1}(i)exp(-\alpha_{T-1} y_i h_{T-1}(i))}{Z_{T-1}} exp(-\alpha_T y_i h_T(i))}{Z_T} \\
&= \frac{D_1(i)exp(\sum_t -\alpha_t y_i h_t(i))}{\prod_{t=1}^T Z_t} \\
&= \frac{1}{m}\frac{exp(\sum_{t=1}^T -\alpha_t y_i h_t(i))}{\prod_{t=1}^T Z_t} \\
&= \frac{1}{m}\frac{exp(-y_i f(i))}{\prod_{t=1}^T Z_t}
\end{aligned}
$$

Theoretical results on the empirical risk

Proof.

since $\sum_i D_{T+1}(i) = 1$ because it is a statistical distribution, we get

$$\prod_{t=1}^{T} Z_t = \frac{1}{m} \sum_i exp(-y_i f(i))$$

We need to minimize each Z_t to minimize the empirical risk of the final combination.

Adaboost

Input: *A learning sample* S, *a number of iterations* T, *a weak learner* L
Output: *A global hypothesis* H_T
for all i *from* 1 *to* m **do**
  $D_1(\mathbf{x_i}) = 1/m$;
for all t *from* 1 *to* T **do**
  $h_t = \mathrm{L}(S, \mathbf{D}_t)$;
  $\hat{\epsilon}_t = \sum_{\mathbf{x_i} \ t.q. \ y_i \neq h_t(\mathbf{x_i})} D_t(\mathbf{x_i})$;
  $\alpha_t = \frac{1}{2} \ln \frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t}$;
  for all $i = 1$ *from* 1 *to* m **do**
    $D_{t+1}(\mathbf{x_i}) = D_t(\mathbf{x_i}) \exp\left(-\alpha_t y_i h_t(\mathbf{x_i})\right) / Z_t$;
    /* Z_t *is a normalization coefficient*/
$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$;
Return H_T **such that**
  $H_T(\mathbf{x}) = sign\left(f(\mathbf{x})\right)$

Theoretical results on the empirical risk

Theorem 2

To minimize Z_t, the confidence coefficient α_t must be set to:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \hat{\epsilon}_t}{\hat{\epsilon}_t} \right)$$

Exercise

We know that

$$Z_t = \sum_{\mathbf{x} \in S} D_t(\mathbf{x}) e^{-\alpha_t y(\mathbf{x}) h_t(\mathbf{x})}.$$

Let us assume that $y(\mathbf{x})$ and $h_t(x) \in \{-1, +1\}$. Let W^b be defined as follows:

$$\forall b \in \{-1, +1\}, \; W^b = \sum_{\mathbf{x} \in S : y(\mathbf{x}) h_t(\mathbf{x}) = b} D_t(\mathbf{x})$$

Use W^b to discard the $\sum$ in Z_t and prove Theorem 2.

Adaboost

Input: *A learning sample* S, *a number of iterations* T, *a weak learner* L
Output: *A global hypothesis* H_T
for all i *from* 1 *to* m **do**
  $D_1(\mathbf{x_i}) = 1/m$;
for all t *from* 1 *to* T **do**
  $h_t = \mathrm{L}(S, \mathbf{D}_t)$;
  $\hat{\epsilon}_t = \sum_{\mathbf{x_i} \ t.q. \ y_i \neq h_t(\mathbf{x_i})} D_t(\mathbf{x_i})$;
  $\alpha_t = \frac{1}{2} \ln \frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t}$;
  for all $i = 1$ *from* 1 *to* m **do**
    $D_{t+1}(\mathbf{x_i}) = D_t(\mathbf{x_i}) \exp\left(-\alpha_t y_i h_t(\mathbf{x_i})\right) / Z_t$;
    /* Z_t *is a normalization coefficient*/
$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$;
Return H_T **such that**
  $H_T(\mathbf{x}) = sign\left(f(\mathbf{x})\right)$

Theoretical results on the empirical risk

Theorem 3

Exponential decrease of the empirical risk

$$\prod_t (Z_t) = \prod_t \sqrt{1 - 4\gamma_t^2} < \exp\left(-2 \sum_t \gamma_t^2\right)$$

where $\hat{\epsilon}_t = \frac{1}{2} - \gamma_t$ (weak hypothesis)

- This theorem means that the empirical risk exponentially decreases towards 0 with the number T of iterations.

Theoretical results on the empirical risk

Proof.

$$Z_t = \sum_{\mathbf{x}} D_t(\mathbf{x}) exp^{-\alpha_t y_{\mathbf{x}} h_t(\mathbf{x})} = W^{+1} e^{-\alpha_t} + W^{-1} e^{\alpha_t}$$

$$= (1-\hat{\epsilon}_t) e^{-\frac{1}{2}\ln\left(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t}\right)} + \hat{\epsilon}_t e^{\frac{1}{2}\ln\left(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t}\right)} = 2\sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)}$$

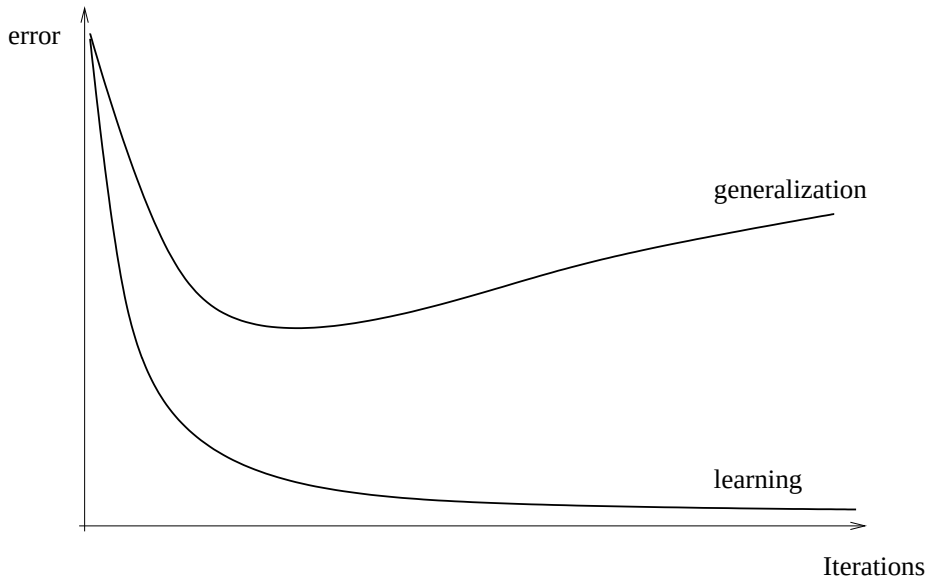
Then,

$$\prod_t (Z_t) = \prod_t (2\sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} = \prod_t \sqrt{4(\frac{1}{2}-\gamma_t)(1-\frac{1}{2}+\gamma_t)} = \prod_t \sqrt{1-4\gamma_t^2}$$

$$= e^{ln(\prod_t \sqrt{1-4\gamma_t^2})} = e^{\sum_t \frac{1}{2} ln(1-4\gamma_t^2)} \leq e^{-2\sum_t \gamma_t^2}$$

because $ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \ldots$

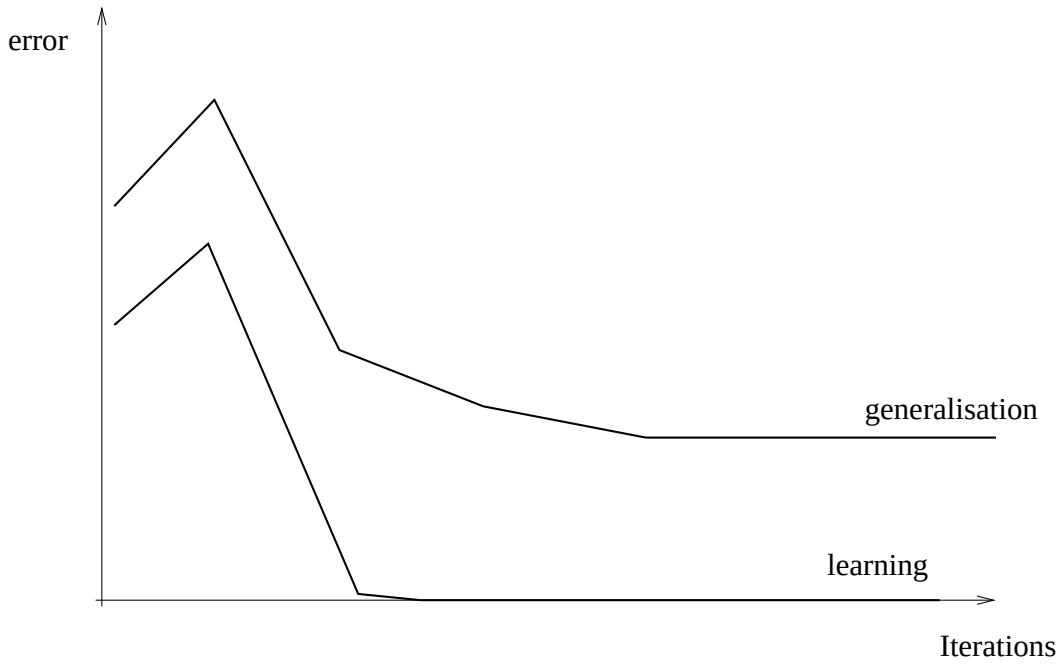
Expected behavior of boosting



Expected behavior

- $\hat{\epsilon}_{H_T}$ *decreases towards (eventually) 0.*
- ϵ_{H_T} *first decreases; then* H_T *becomes too complex* $\rightarrow$ *overfitting*

Observed behavior of boosting



Observed behavior

- $\hat{\epsilon}_{H_T}$ *decreases towards (eventually) 0.*
- ϵ_{H_T} *drops and continues to decrease even when* $\hat{\epsilon}_T$ *has reached 0!!*

Experimental proof

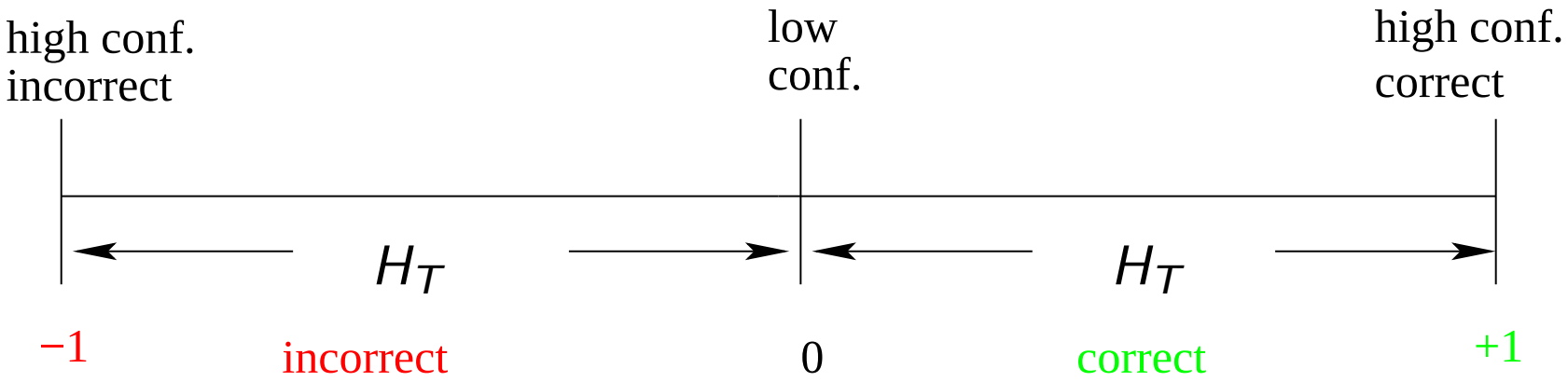
Experimental "proof" with AdaBoost

Explanation in terms of margins

Definition

The margin of an example is defined by

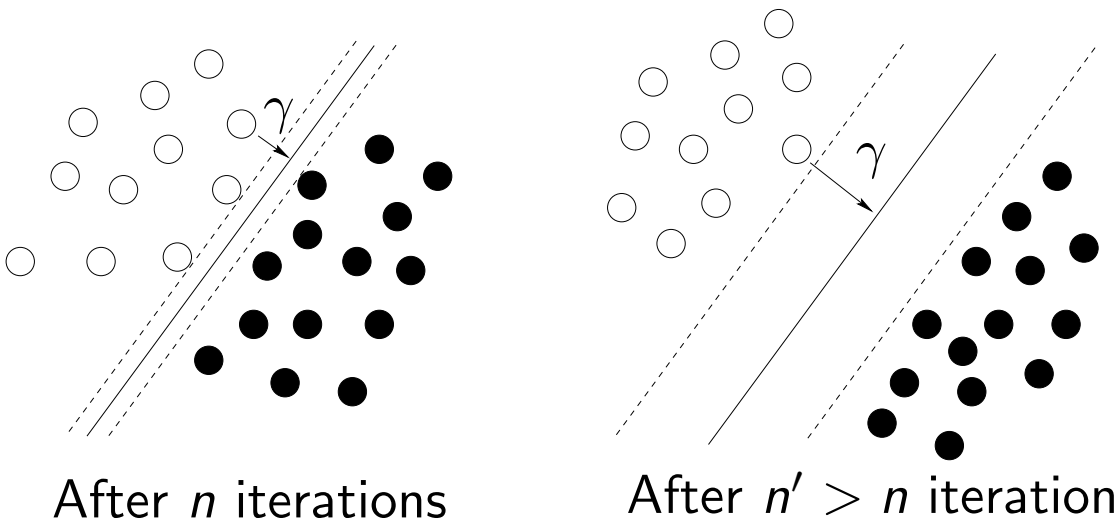
$$margin(\mathbf{x}) = \frac{yf(x)}{\sum_t \alpha_t} = \frac{y \sum_t \alpha_t h_t(\mathbf{x})}{\sum_t \alpha_t}$$



Behavior of Adaboost in terms of margins

Relationship between margins and generalization ability

- Larger margins on the training set translate into a tighter upper bound on the generalization error.
- Despite the increase of its complexity, the final performing classifier is becoming easier and easier to build because of the increase of the margins.



After n iterations

After $n' > n$ iterations

Behavior of Adaboost in terms of margins

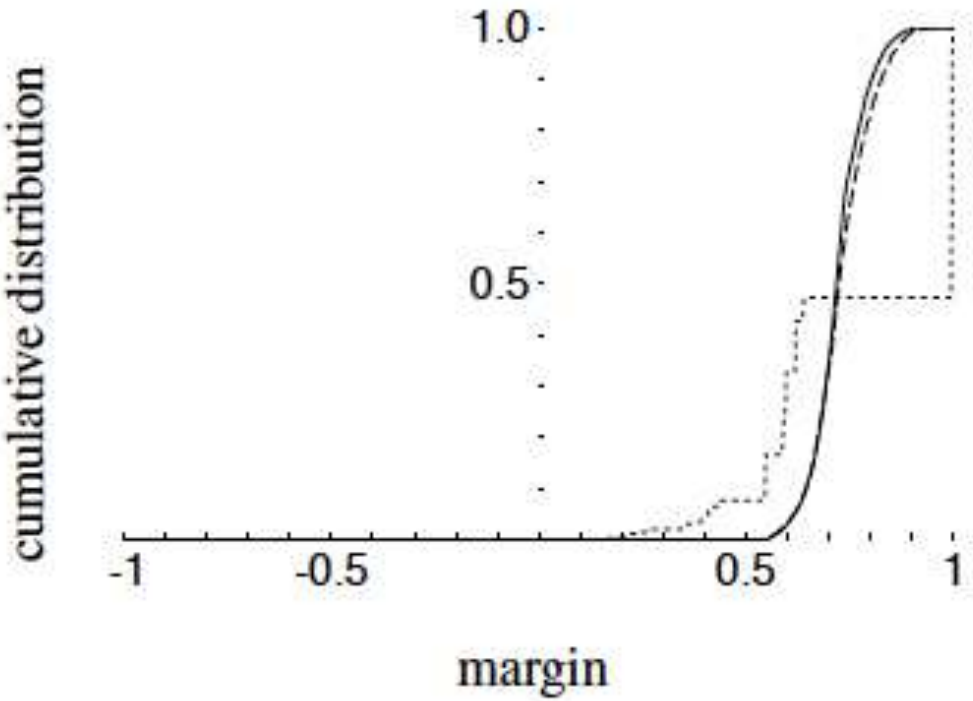


Figure : Cumulative distribution of margins of the training examples after 5, 100 and 1000 iterations, indicated by short-dashed, long-dashed and solid curves, respectively.

Theoretical results in generalization

Theorem 4

Let $\mathcal{H}$ be a class of classifiers with VC dim d_h (*i.e.* the capacity of $\mathcal{H}$). For any $\delta > 0$ and $\theta > 0$, with probability $1 - \delta$, any classifier ensemble H_T built from m learning examples satisfies:

$$\epsilon_{H_T} \leq \hat{Pr}(margin(\mathbf{x}) \leq \theta) + \mathcal{O}\left(\sqrt{\frac{d_h}{m}\frac{log^2(m/d_h)}{\theta^2} + log(1/\delta)}\right)$$

This bound depends on:

- constant parameters m, d_h, θ and δ.
- the distribution of the margins of the learning examples $\hat{Pr}$.

Theorem 5

$\hat{Pr}(margin(\mathbf{x}) \leq \theta)$ exponentially decreases towards 0 with T.

Margin maximization

Theorem

Let margin$(\mathbf{x})$ *be the margin of an example:*

$$\hat{Pr}(margin(\mathbf{x}) \leq \theta) \leq \left(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}} \right)^T$$

If $\theta < \gamma$*, this bound exponentially decreases with* T*.*

Margin maximization

Proof.

If $\frac{yf(\mathbf{x})}{\sum_t \alpha_t} \leq \theta$ then $y \sum_t \alpha_t h_t(\mathbf{x}) \leq \theta \sum_t \alpha_t$, then
$-y \sum_t \alpha_t h_t(\mathbf{x}) + \theta \sum_t \alpha_t \geq 0$

$$\Leftrightarrow exp^{-y \sum_t \alpha_t h_t(\mathbf{x}) + \theta \sum_t \alpha_t} \geq 1$$

$$\begin{aligned}
\hat{Pr}(\frac{yf(\mathbf{x})}{\sum_t \alpha_t} \leq \theta) = \frac{1}{m} \sum_{(x,y)} [\frac{yf(\mathbf{x})}{\sum_t \alpha_t} \leq \theta] &\leq \frac{1}{m} \sum_{(x,y)} exp^{-y \sum_t \alpha_t h_t(\mathbf{x}) + \theta \sum_t \alpha_t} \\
&= \frac{exp^{\theta \sum_t \alpha_t}}{m} \sum_{(x,y)} exp^{-y \sum_t \alpha_t h_t(\mathbf{x})} \\
&= exp^{\theta \sum_t \alpha_t} \prod_t Z_t \text{ (see Th.1)}
\end{aligned}$$

Proof.

By replacing α_t and Z_t by their expressions with $\hat{\epsilon}_t$, we get:

$$= exp^{\sum_t \theta \frac{1}{2} ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})} 2^T \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} = 2^T \prod_t exp^{ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})^{\frac{\theta}{2}}} \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)}$$

$$= \prod_t 2\sqrt{\frac{(1-\hat{\epsilon}_t)^\theta \hat{\epsilon}_t(1-\hat{\epsilon}_t)}{\hat{\epsilon}_t^\theta}} = \prod_t 2\sqrt{\hat{\epsilon}_t^{1-\theta}(1-\hat{\epsilon}_t)^{1+\theta}}$$

$$= \prod_t 2\sqrt{(\frac{1}{2}-\gamma)^{1-\theta}(\frac{1}{2}+\gamma)^{1+\theta}}$$

$$= \prod_t 2\sqrt{\frac{1}{2^{1-\theta}}(1-2\gamma)^{1-\theta}\frac{1}{2^{1+\theta}}(1+2\gamma)^{1+\theta}} = \left(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}}\right)^T$$

If $\theta < \gamma$ then the expression between brackets is < 1 and so the probability to have a small margin decreases with T.

□

First conclusions

- AdaBoost works in practice...
- and is theoretically well-founded!

Some platforms:

- Yoav Freund's home page
  `http://www.cs.ucsd.edu/~yfreund/adaboost`
- Ran El-Yaniv's home page
  `http://www.cs.technion.ac.il/~rani/LocBoost/`
- Weka (University of Waikato) in JAVA
  `http://www.cs.waikato.ac.nz/ml/weka/`
- Scikit-Learn in Python
  `http://scikit-learn.org/stable/`

Practical advantages of AdaBoost

- Fast.
- Simple and easy to program.
- No parameters to tune (except T).
- Flexible - can combine with any learning algorithm.
- Provably effective.

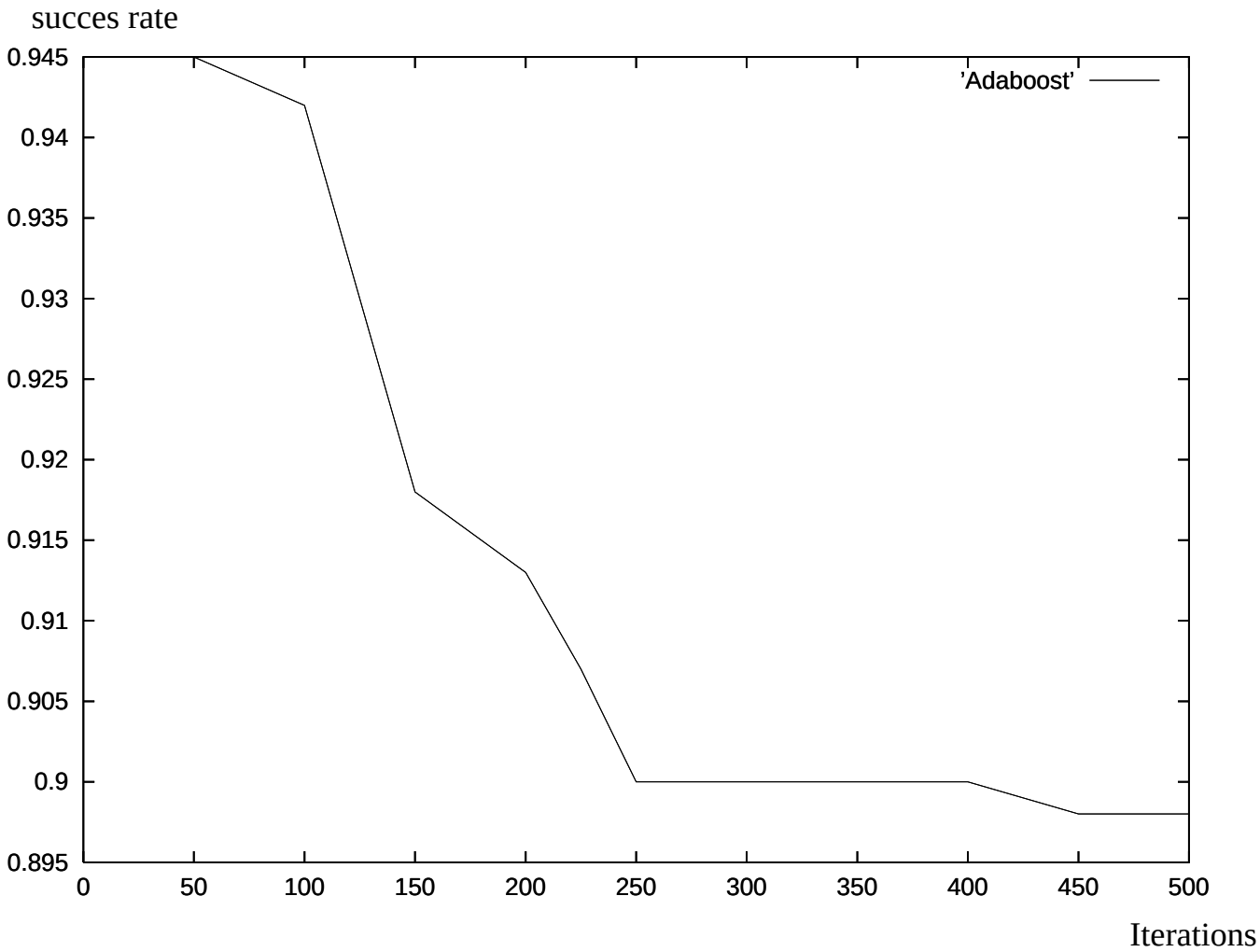
Caveats

Performances of ADABOOST depends on data and weak learner:

- ADABOOST works very well with Decision Stumps
- But it can fail if:
  - weak classifiers too strong (*e.g.* kNN, ID3) $\rightarrow$ overfitting + no diversity $\rightarrow$ stops after a few iterations ($\hat{\epsilon}_t = 0$).
  - weak classifier too weak ($\gamma_t \rightarrow 0$ too quickly and therefore the condition $\theta < \gamma_t$ is not satisfied) $\rightarrow$ underfitting or low margins.
- empirically, ADABOOST seems especially susceptible to the:
  - Presence of outliers $\rightarrow$ exponential increase of their weights $\rightarrow$ overfitting.
  - Presence of large bayesian error $\rightarrow$ slows down the convergence.

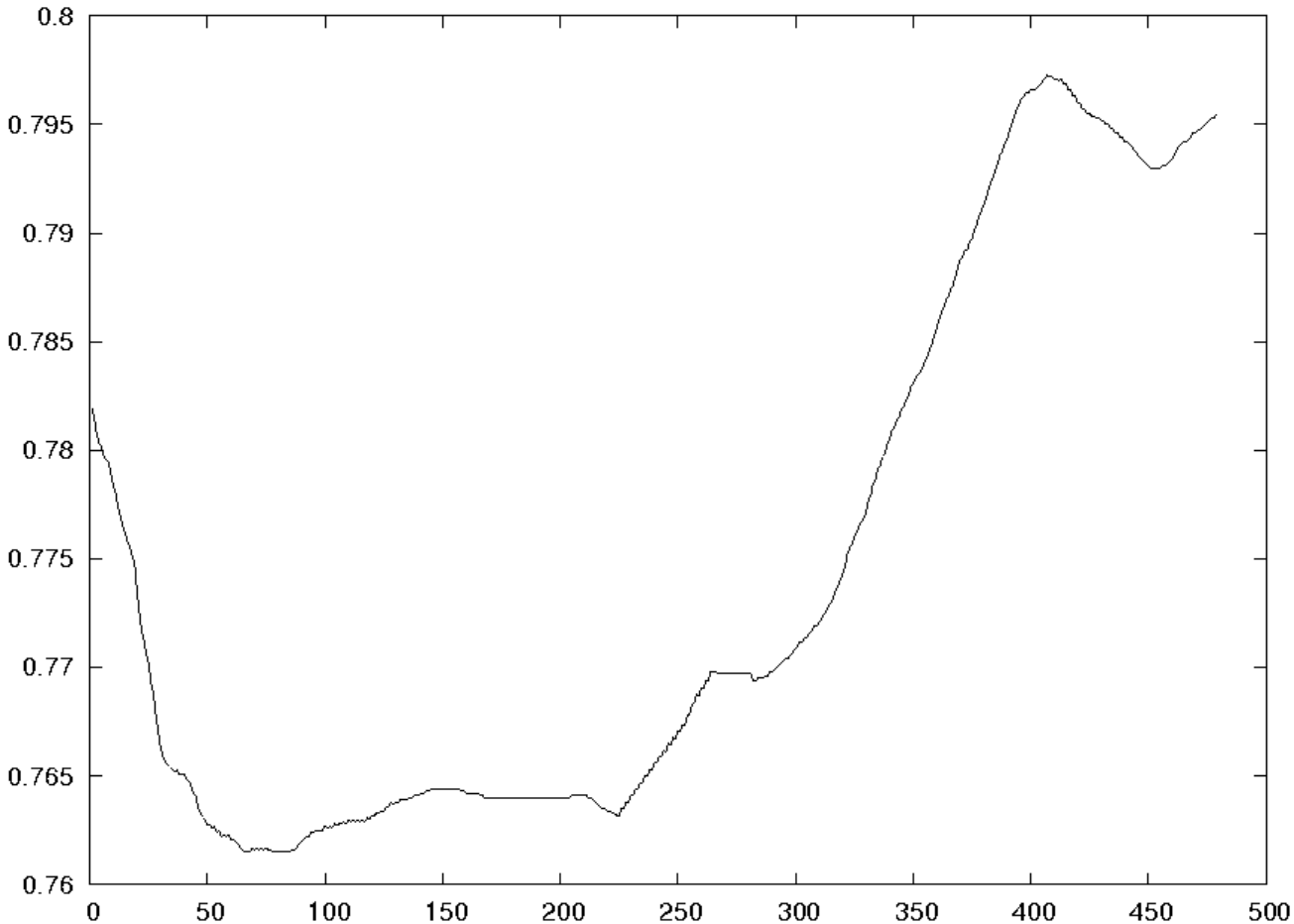
Impact of outliers on ADABOOST behavior

Illustration: If ADABOOST is run on a linearly separable problem containing some noise (by swapping -1 and $+1$), we get:



Hard predictions can slow learning

Illustration: If ADABOOST is run from overlapping distributions, we get:



Edge versus Margin

Double optimization of Adaboost

In a boosting algorithm, we optimize two sets of weights:

- a distribution D_t over the learning examples.
- a distribution α_t over the weak hypotheses.

Edge versus Margin

Definition

The *Edge* E_t of a hypothesis h_t for a given distribution D_t on the m training examples is

$$E_t = \sum_{i=1}^{m} y_i h_t(x_i) \times D_t(x_i)$$

Definition

The *Margin* M_i of a learning example x_i after T iterations of Adaboost is

$$M_i = y_i \sum_{t=1}^{T} h_t(x_i) \times \alpha_t$$

Edge versus Margin

The two objectives of boosting:

- w.r.t. the Edge E_t
  - Edges of past hypotheses should be small after update. In other words, the new hypothesis h_{t+1} must learn something new (i.e. diversity)
  - Therefore,we aim at **minimizing the maximum edge** of past hypotheses.
- w.r.t. the Margin M_i
  - Choose a convex combination of weak hypotheses that **maximizes the minimum margin**.

Connection between the two objectives?

Edge versus Margin

Duality

Linear Programming Duality

$$\min_D \max_t E_t = \max_\alpha \min_i M_i$$

$$min_D max_{t=1,\dots,T-1} \sum_{i=1}^{m} y_i h_t(x_i) D_t(x_i) = max_\alpha min_{i=1,\dots,m} y_i \sum_{t=1}^{T-1} h_t(x_i)\alpha_t$$