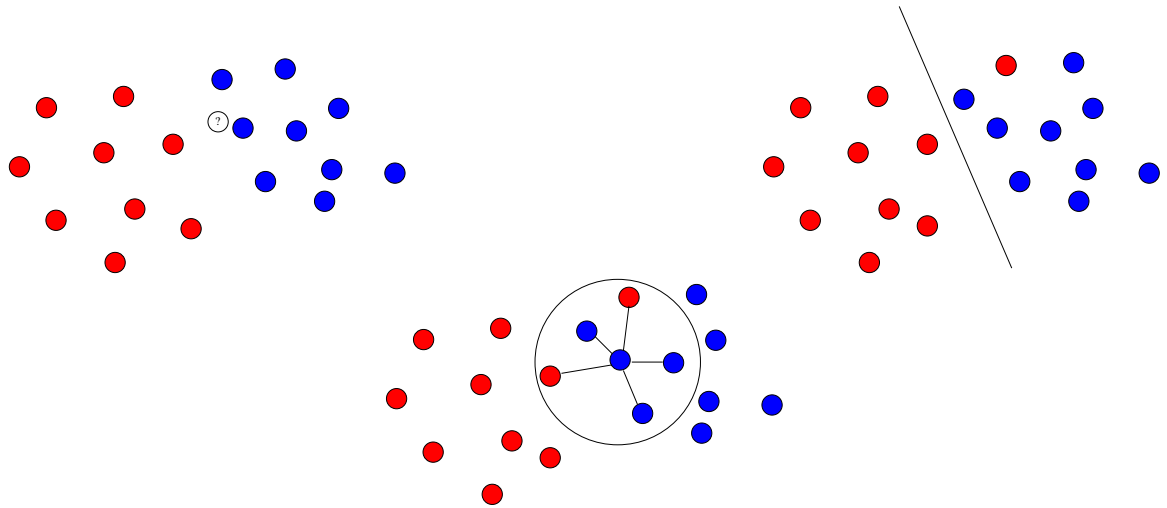
Ensemble Methods

Marc Sebban

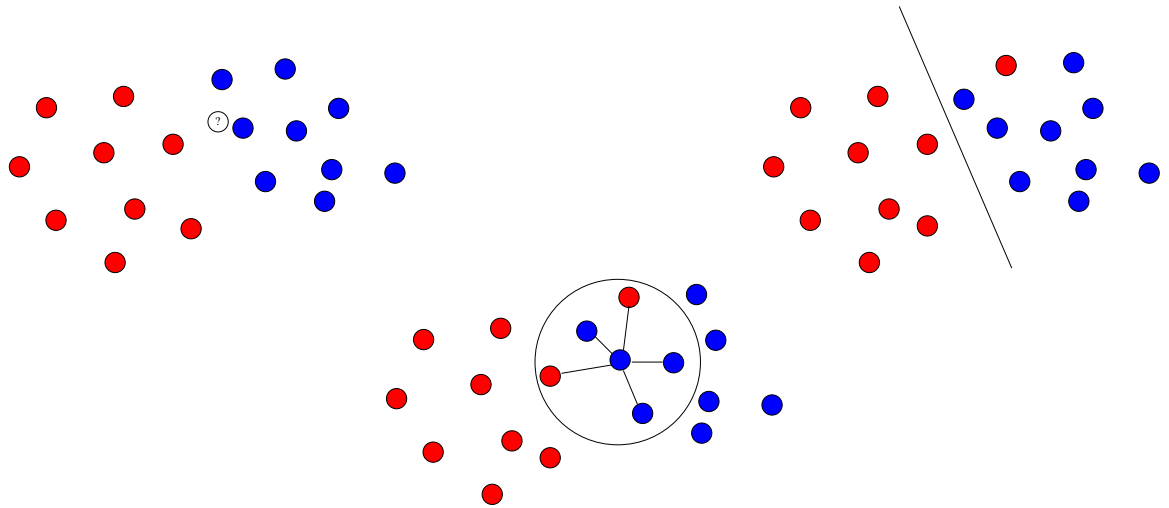
Hubert Curien lab, UMR CNRS 5516
University of Jean Monnet Saint-Étienne (France)

Outline

Many classifiers can be induced from the same task

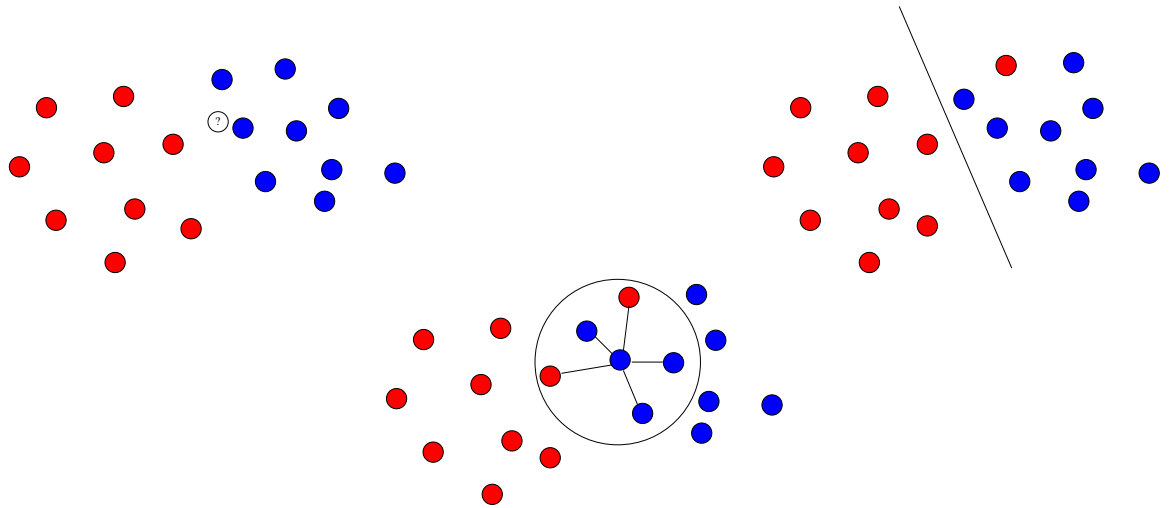


Many classifiers can be induced from the same task



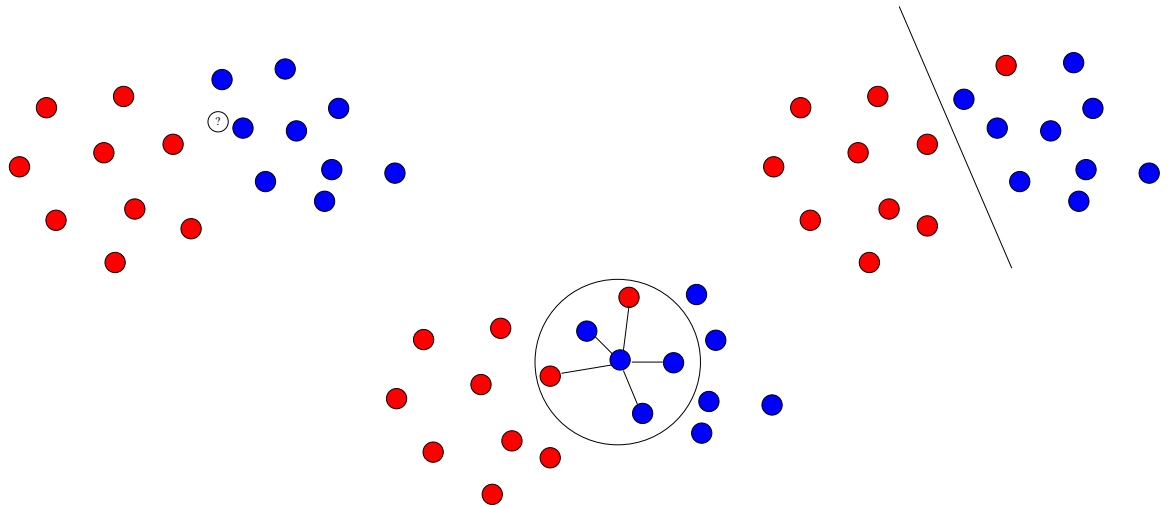
- Different learning algorithms (*e.g.* k-NNs, linear separator, decision trees, SVMs, etc.).

Many classifiers can be induced from the same task



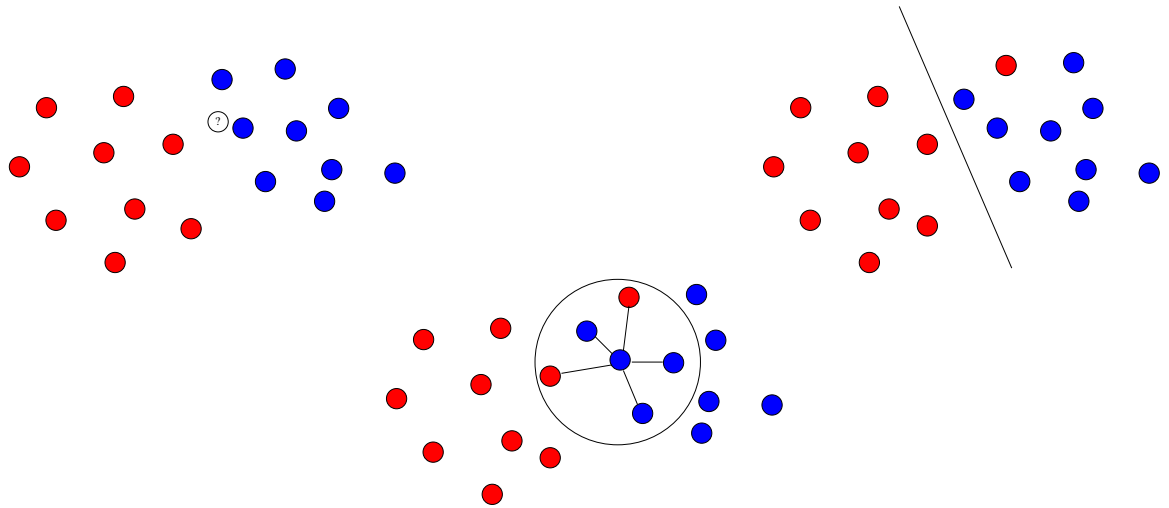
- Different learning algorithms (*e.g.* k-NNs, linear separator, decision trees, SVMs, etc.).
- Different hyperparameters (*e.g.* number of neighbors k, regularization parameter λ, learning rate α, degree d of the polynomial, etc.).

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, regularization parameter λ, learning rate α, degree d of the polynomial, etc.).
- Different (randomly drawn) training sets S.

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, regularization parameter λ, learning rate α, degree d of the polynomial, etc.).
- 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.

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.

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.

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.

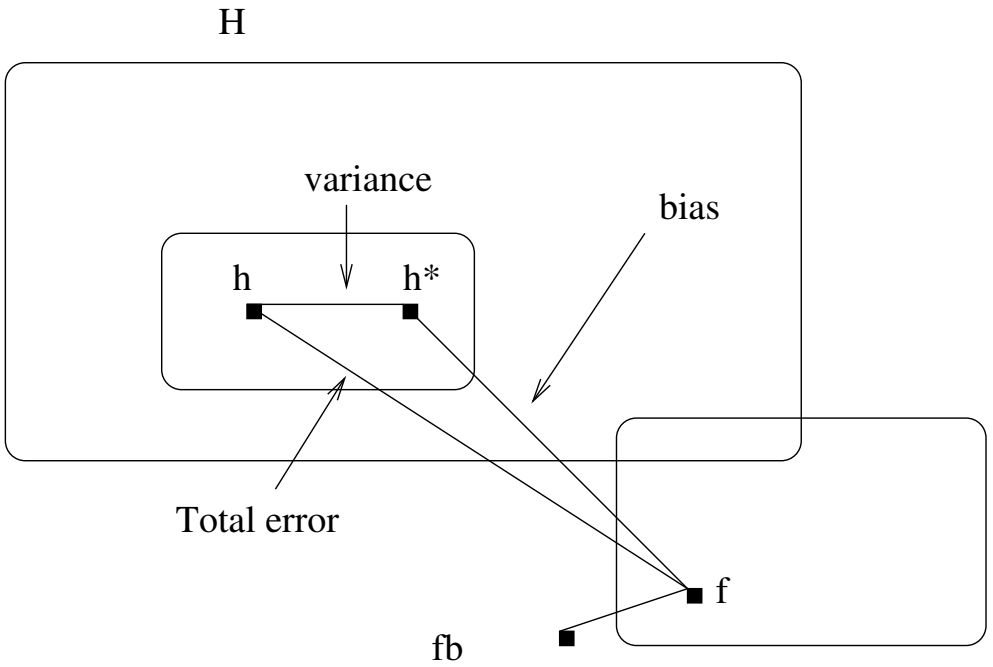
Necessary and sufficient conditions for an ensemble of classifiers to be efficient:

- 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.

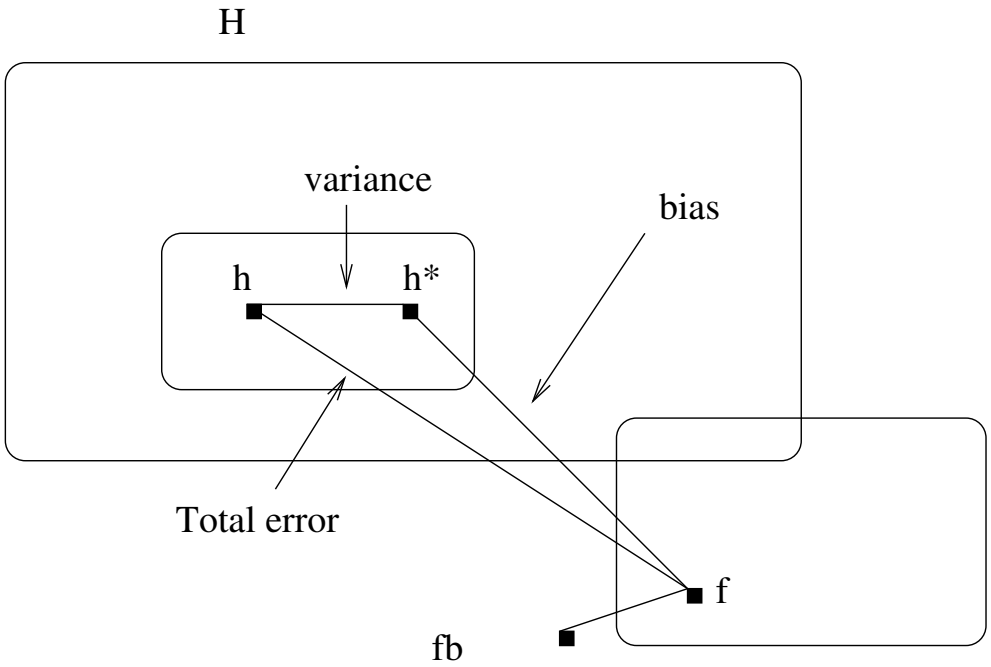
Question

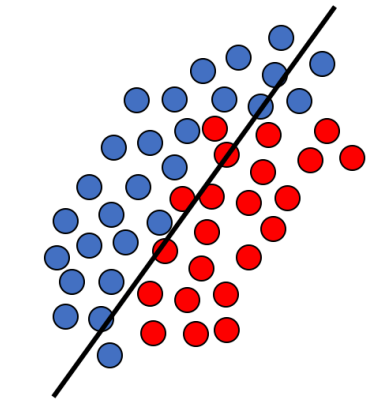
Is it possible to construct (theoretically) good ensembles?

Bias/Variance trade-off



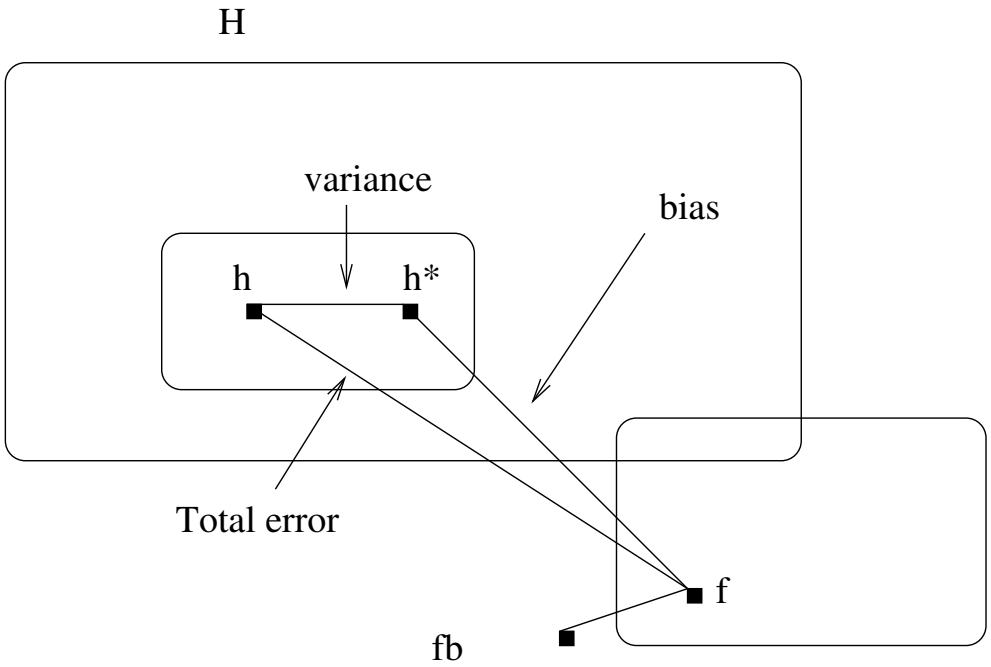
Bias/Variance trade-off

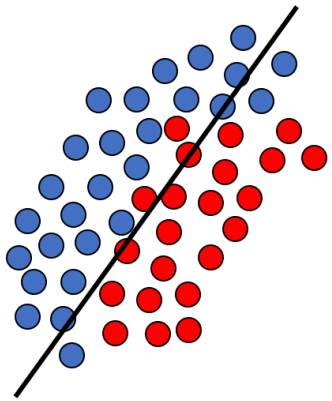




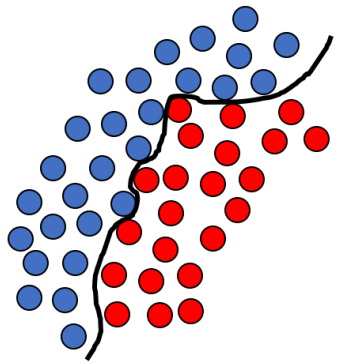
Large bias and small variance

Bias/Variance trade-off





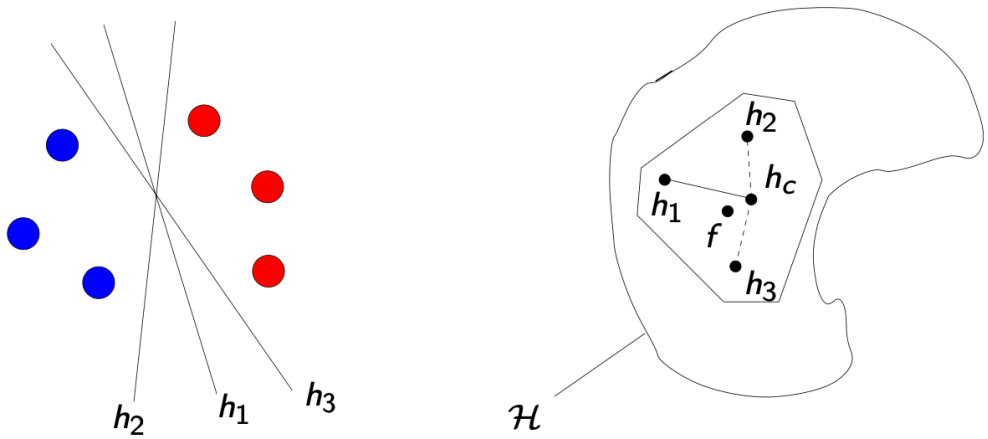
Large bias and small variance



Large variance and small bias

Limitations of a single classifier

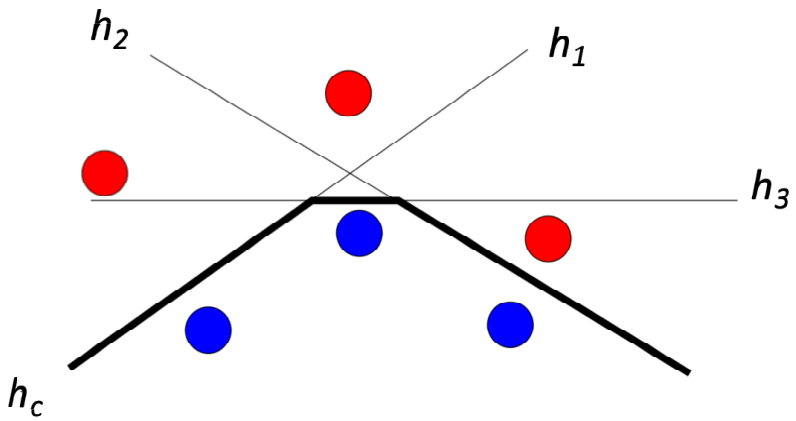
Statistical problem (variance): Without sufficient data, the learning algorithm can find many different hypotheses in $\mathcal{H}$ that all give the same empirical accuracy on S.

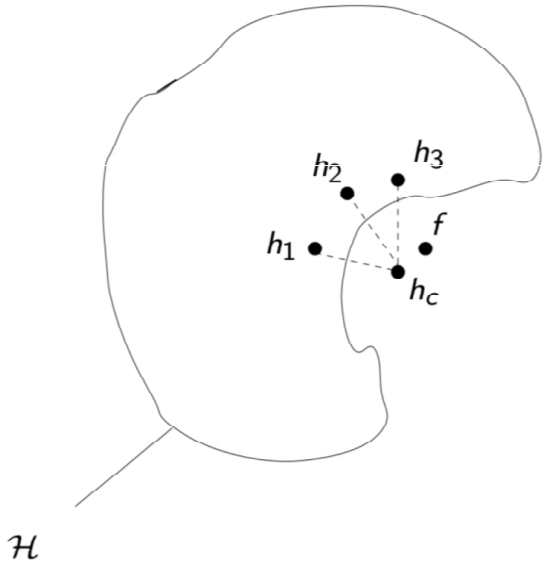


By constructing an ensemble h_c out of all of these accurate classifiers, the algorithm can "average" their votes and reduce the risk of choosing the wrong

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 $\mathcal{H}$.

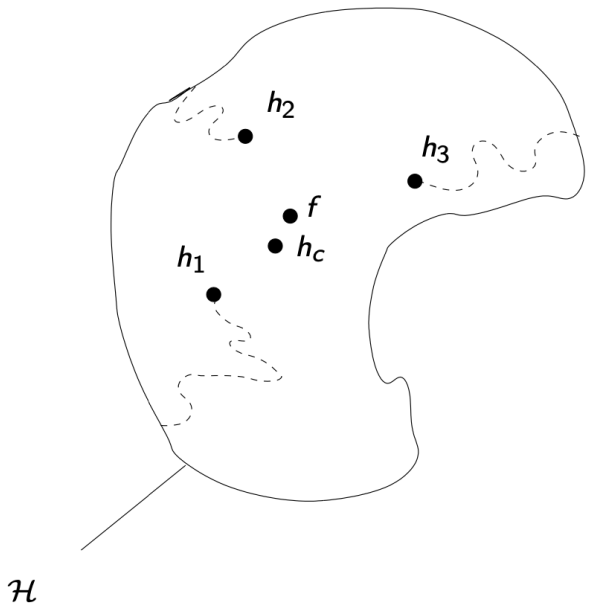




By forming weighted sums of hypotheses drawn from $\mathcal{H}$, it may be possible to expand the space of representable functions.

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.

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.

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.

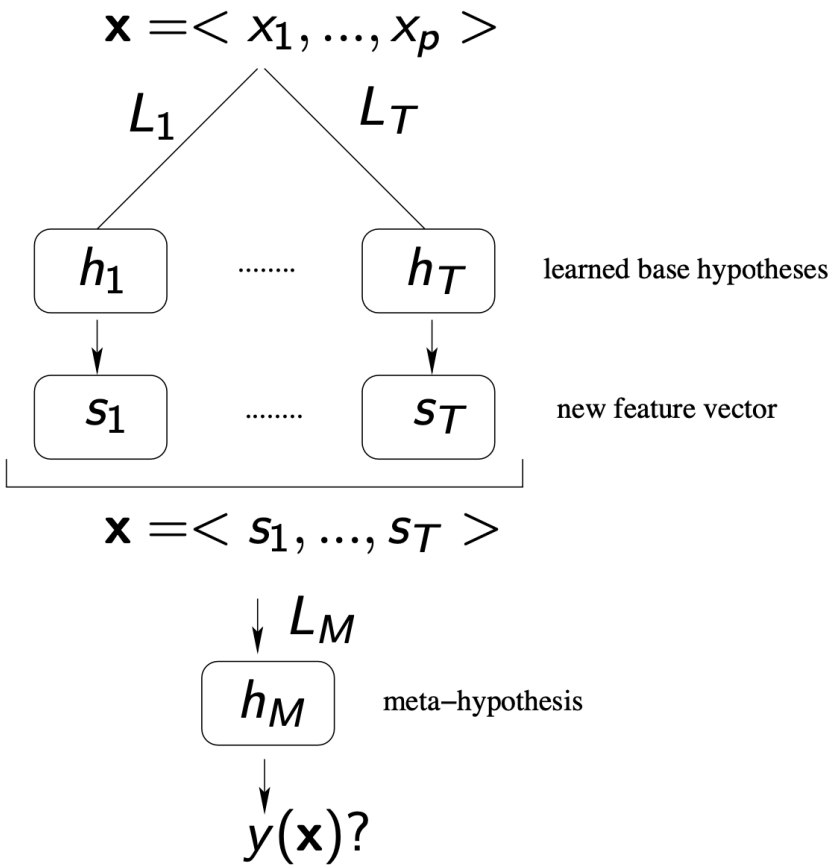
Heterogeneous ensemble methods

Heterogeneous ensemble methods

The diversity comes from the learning algorithms.

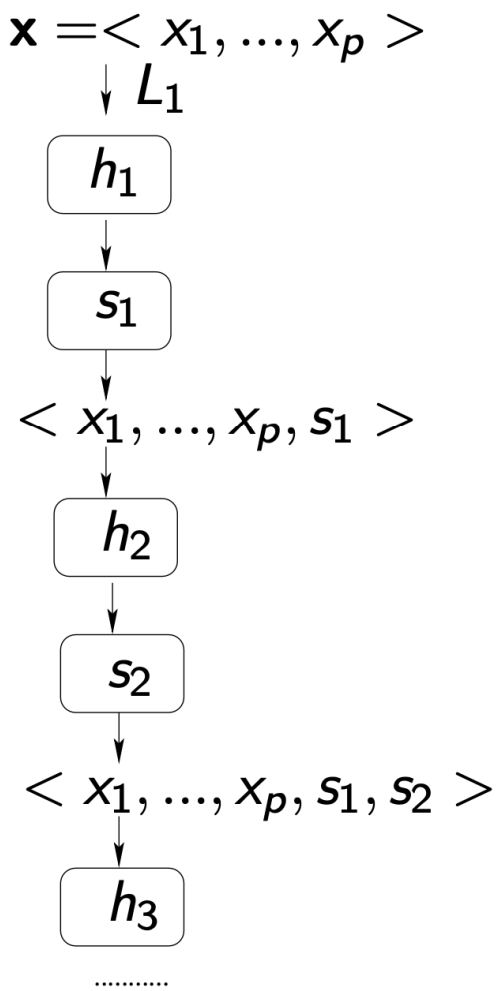
- **Stacking**
- **Cascade Generalization**

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

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

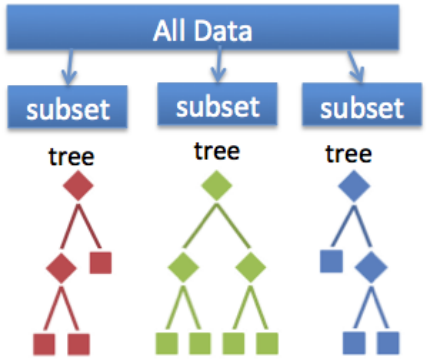
Input: *A learning sample* $S = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_m}, y_m)\}$
Input: *A total number* T *of bagging rounds*
Input: *A learning algorithm* L *returning a binary classifier*
Output: *A combined classifier*
for all t *from* 1 *to* T **do**
  $S_t = Resample(S)$ // *Randomly sample* 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

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 (e.g. ID3 with entropy).
- While growing the tree, a **random subset of F features** is selected from the p original ones.



Introduction to boosting

Robert Schapire

Boosting aims at learning a combination of weak classifiers where the update of the distribution is driven by "hard" examples.

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.

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.

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?

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?

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**.

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**.

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**.

Boosting combines *weak* hypotheses into a *strong* hypothesis (from a PAC theory point of view).

Strong vs Weak Learnability

$$\forall h \in \mathcal{H}, \forall \mathcal{D}_{\mathcal{Z}}, \forall \gamma \geq 0, \forall \delta \leq 1, P(|\mathcal{R}(h) - \mathcal{R}(h^*)| \leq \epsilon) \geq 1 - \delta$$

Strong vs Weak Learnability

$$\forall h \in \mathcal{H}, \forall \mathcal{D}_{\mathcal{Z}}, \forall \gamma \geq 0, \forall \delta \leq 1, P(|\mathcal{R}(h) - \mathcal{R}(h^*)| \leq \epsilon) \geq 1 - \delta$$

Strong PAC Learnability

A concept class $\mathcal{F}$ is **strongly PAC learnable** using a hypothesis class $\mathcal{H}$ if there exists an algorithm A such that for any $f \in \mathcal{F}$, for any distribution $D_{\mathcal{Z}}$ over the input space, for any $\epsilon \in (0, \frac{1}{2})$ and $\delta \in (0, \frac{1}{2})$, given access to a **polynomial (in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$) number of examples** drawn i.i.d. from $D_{\mathcal{Z}}$ and labeled by f, A outputs a function $h \in \mathcal{H}$ such that $P(\mathcal{R}(h) \leq \epsilon) \geq 1 - \delta$.

Strong vs Weak Learnability

$$\forall h \in \mathcal{H}, \forall D_{\mathcal{Z}}, \forall \gamma \geq 0, \forall \delta \leq 1, P(|\mathcal{R}(h) - \mathcal{R}(h^*)| \leq \epsilon) \geq 1 - \delta$$

Strong PAC Learnability

A concept class $\mathcal{F}$ is **strongly PAC learnable** using a hypothesis class $\mathcal{H}$ if there exists an algorithm A such that for any $f \in \mathcal{F}$, for any distribution $D_{\mathcal{Z}}$ over the input space, for any $\epsilon \in (0, \frac{1}{2})$ and $\delta \in (0, \frac{1}{2})$, given access to a **polynomial (in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$) number of examples** drawn i.i.d. from $D_{\mathcal{Z}}$ and labeled by f, A outputs a function $h \in \mathcal{H}$ such that $P(\mathcal{R}(h) \leq \epsilon) \geq 1 - \delta$.

Weak PAC Learnability

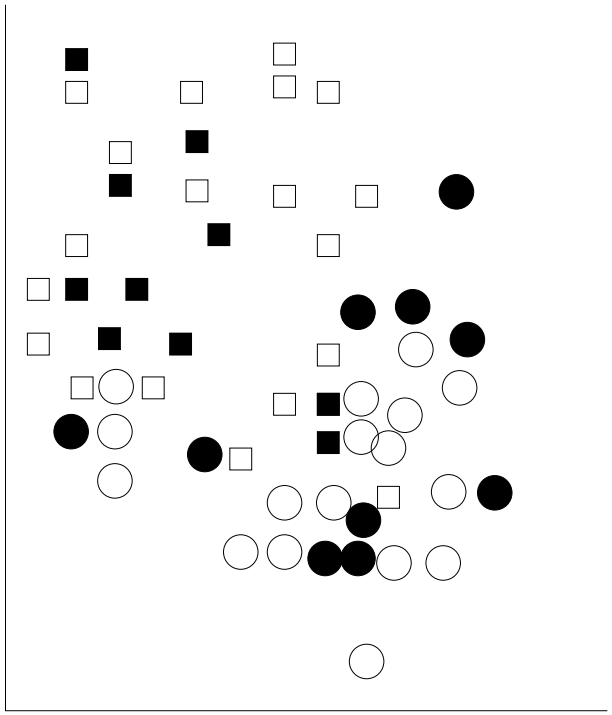
A concept class $\mathcal{F}$ is **weakly PAC learnable** using a hypothesis class $\mathcal{H}$ if there exists an algorithm A and a value $\gamma > 0$ such that for any $f \in \mathcal{F}$, for any distribution $D_{\mathcal{Z}}$ over the input space, for any $\delta \in (0, \frac{1}{2})$, given access to a **polynomial (in $\frac{1}{\delta}$) number of examples** drawn i.i.d. from $D_{\mathcal{Z}}$ and labeled by f, A outputs a function $h \in \mathcal{H}$ such that $P(\mathcal{R}(h) \leq \frac{1}{2} - \gamma) \geq 1 - \delta$.

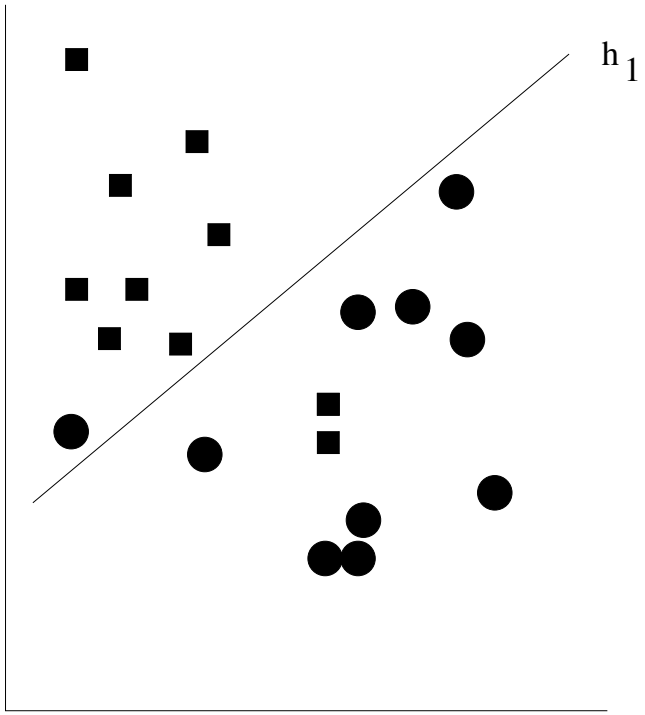
Strong vs Weak Learnability

- We will sometimes refer to γ as the **advantage of A** (over random guessing).
- It's clear that strong learnability implies weak learnability.
- The question we want to answer is whether weak learnability implies strong learnability, i.e., if $\mathcal{F}$ is weakly learnable using $\mathcal{H}$, must there exist some $\mathcal{H}'$ such that $\mathcal{F}$ is (strongly) learnable using $\mathcal{H}'$?

First boosting algorithm (1/4)

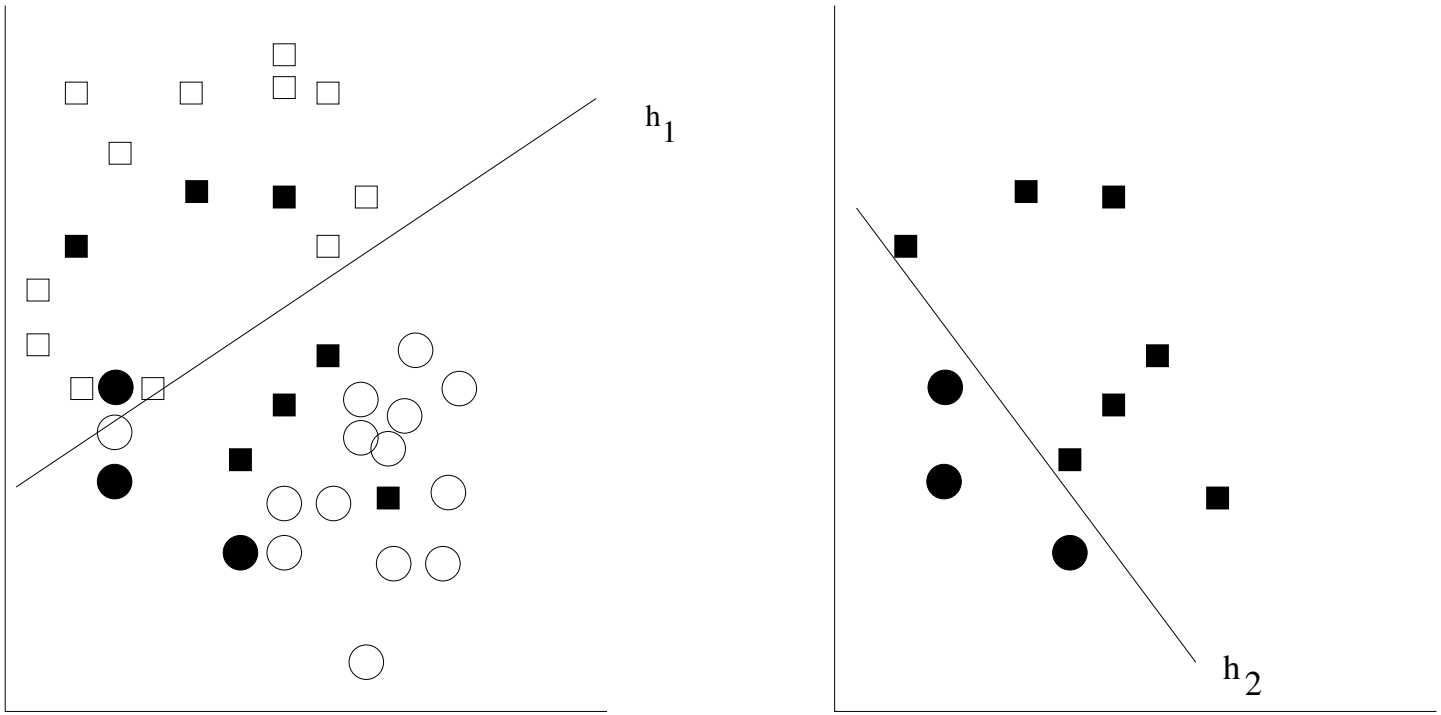
Step 1: Extract from S a learning sample S_1. Use a learning algorithm L to produce a first hypothesis h_1.





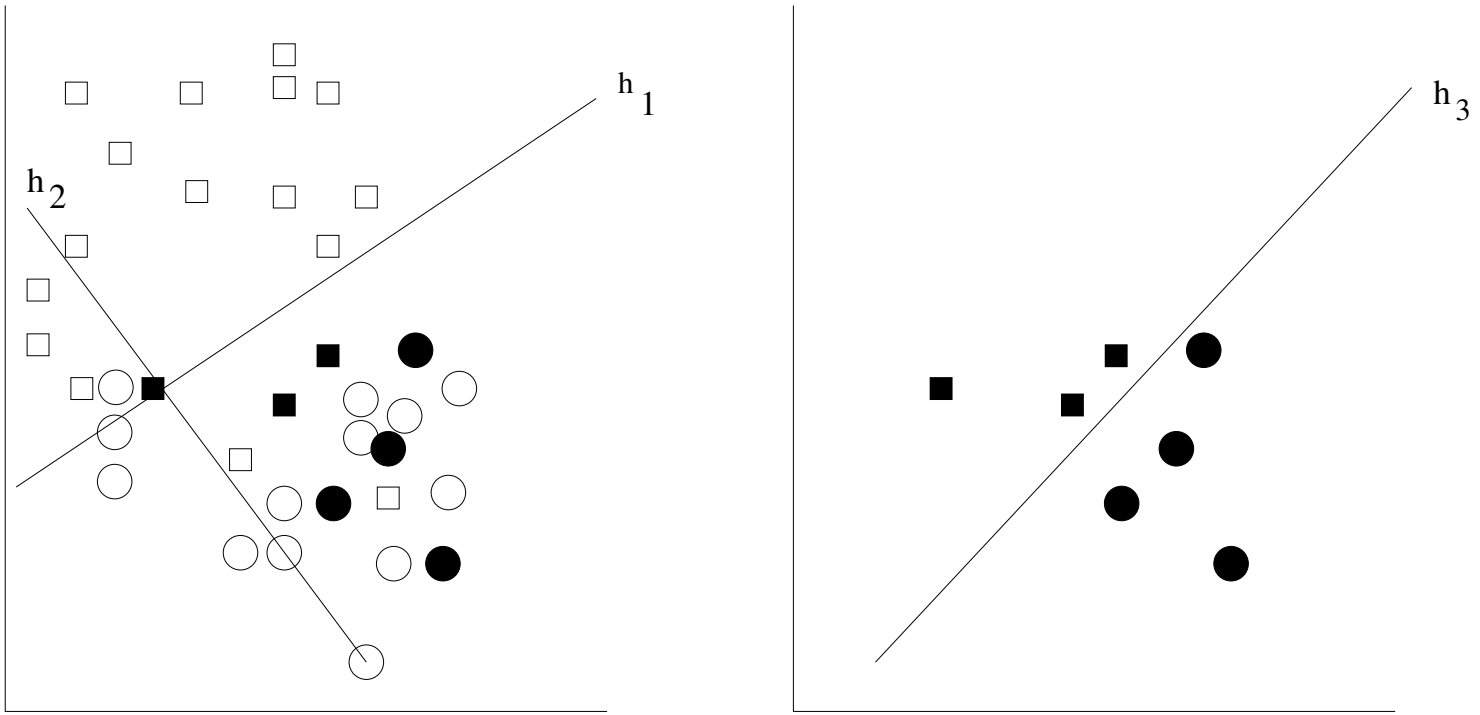
First boosting algorithm (2/4)

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.

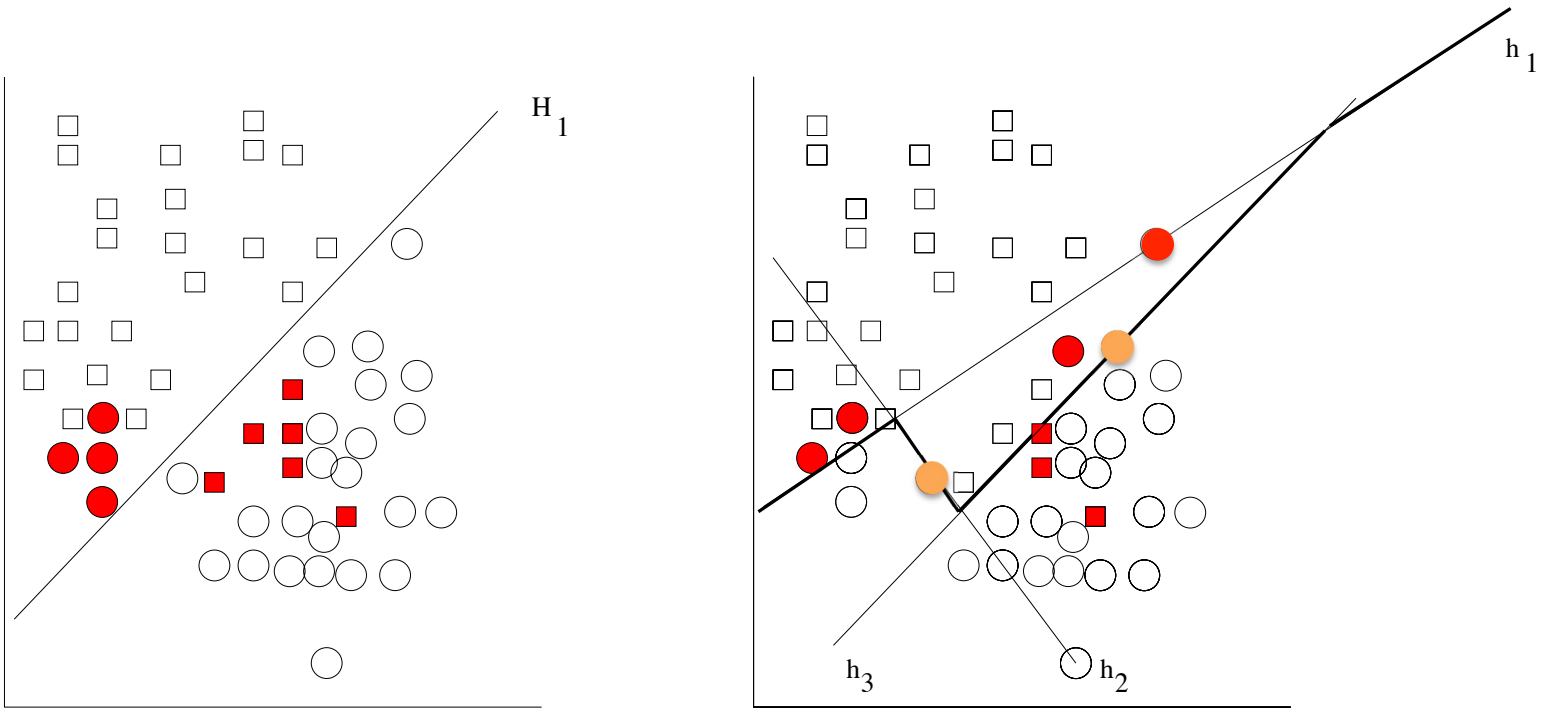


First boosting algorithm (3/4)

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

Adaboost

Input: *A learning sample S, a number of iterations T, a weak learner* L
Output: *A global hypothesis H_T*

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**
  $D_1(\mathbf{x_i}) = 1/m$;

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**
  $h_t = \mathrm{L}(S, \mathbf{D}_t)$;

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**
  $h_t = \mathrm{L}(S, \mathbf{D}_t)$;
  $\hat{\epsilon}_t = \sum_{\mathbf{x_i} \ t.q. \ y_i \neq h_t(\mathbf{x_i})} D_t(\mathbf{x_i})$;

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**
  $h_t = \mathrm{L}(S, \mathbf{D}_t)$;
  $\hat{\epsilon}_t = \sum_{\mathbf{x_i} \ t.q. \ y_i \neq h_t(\mathbf{x_i})} D_t(\mathbf{x_i})$;
  $\alpha_t = \frac{1}{2} \ln \frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t}$;

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*/

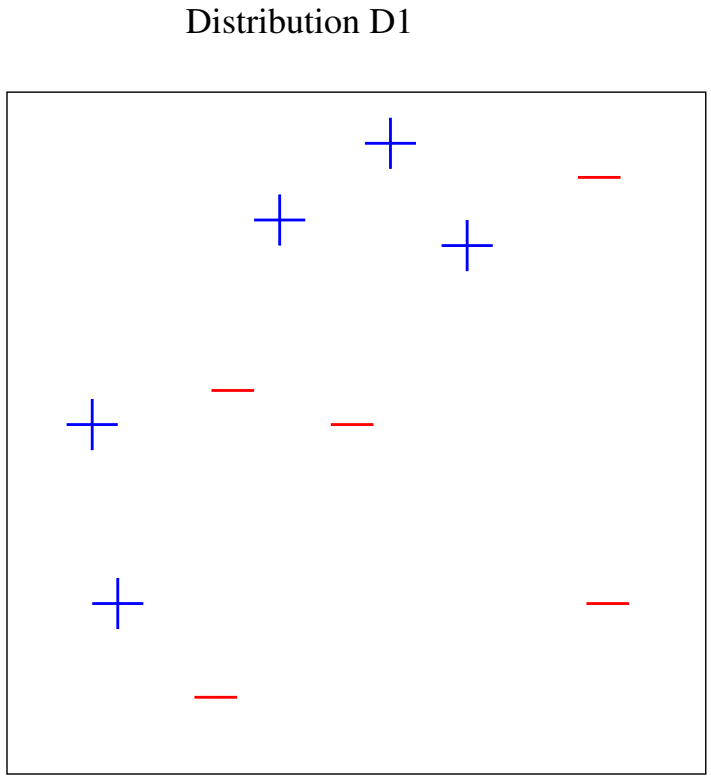
Adaboost

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*/
$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$;

Adaboost

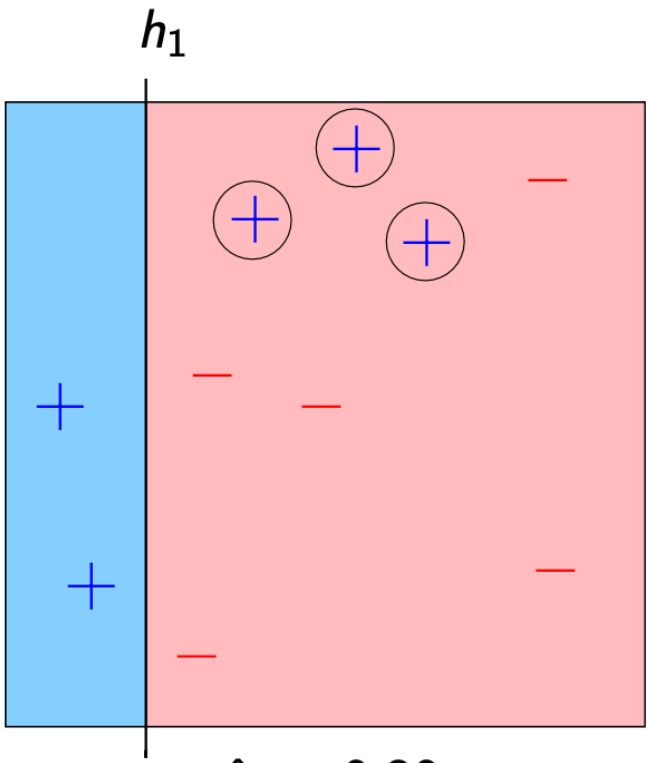
Input: *A learning sample* S, *a number of iterations* T, *a weak learner* L
Output: *A global hypothesis* H_T
for all i *from* 1 *to* m **do**
  $D_1(\mathbf{x_i}) = 1/m$;
for all t *from* 1 *to* T **do**
  $h_t = \mathrm{L}(S, \mathbf{D}_t)$;
  $\hat{\epsilon}_t = \sum_{\mathbf{x_i}\ t.q.\ y_i \neq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

Learning sample S



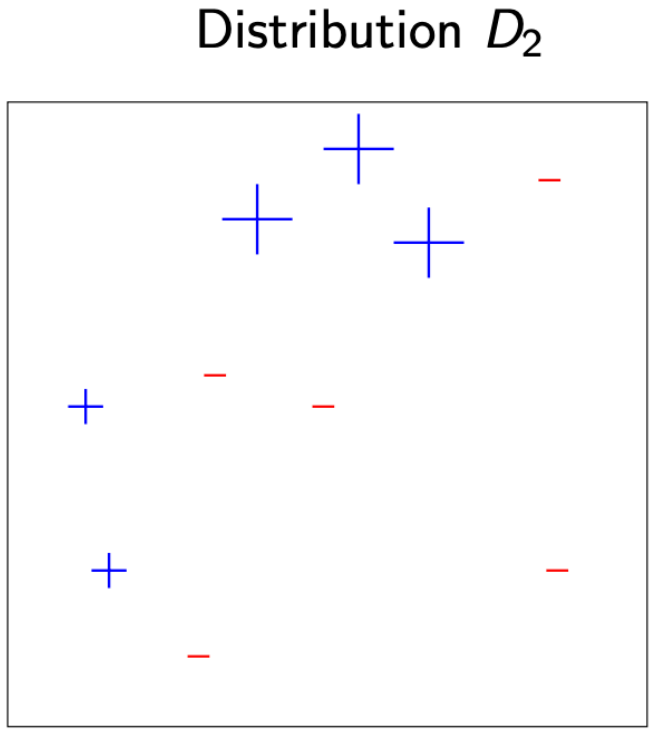
Weak Hypotheses: linear separators parallel to the axis

Step 1

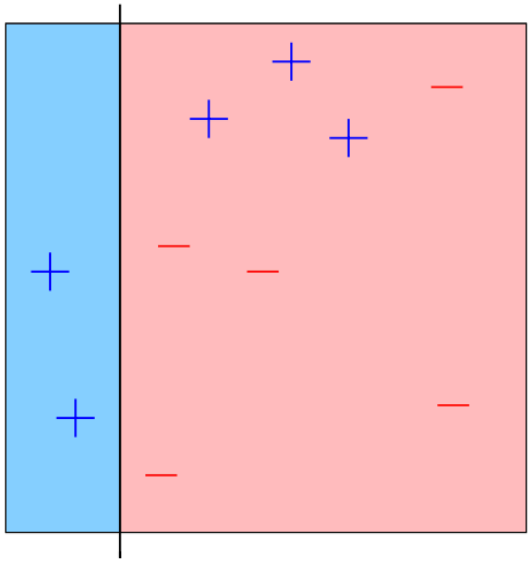


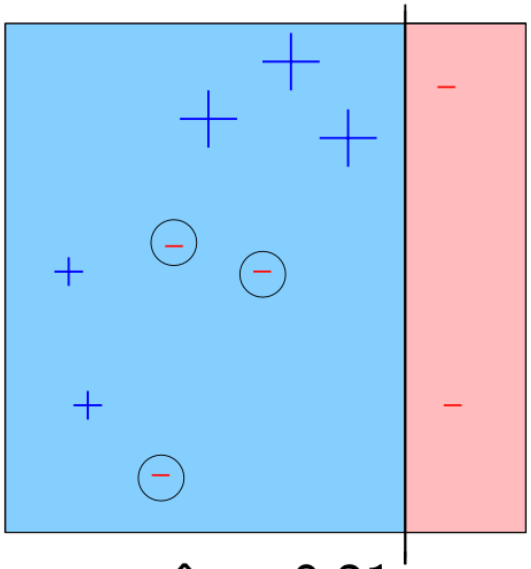
$\hat{\epsilon}_1 = 0.30$

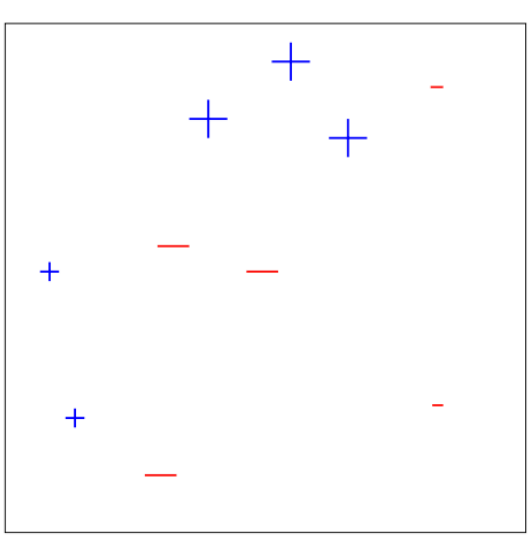
$\alpha_1 = 0.42$



Step 2

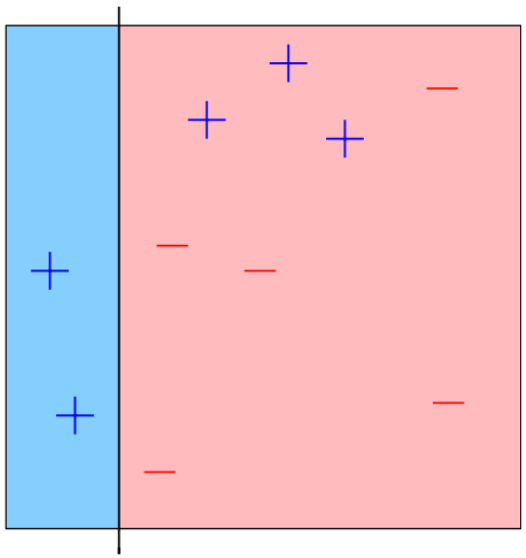


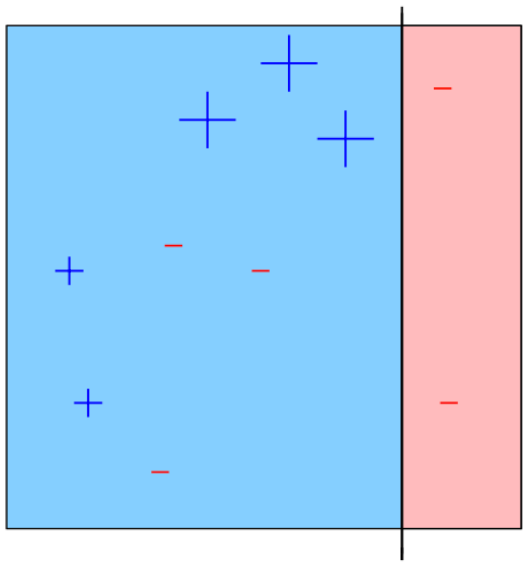


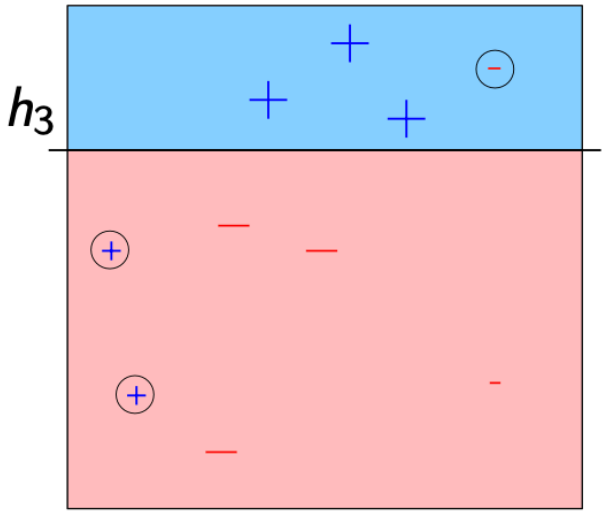


$\hat{\epsilon}_2 = 0.21$
$\alpha_2 = 0.65$

Step 3

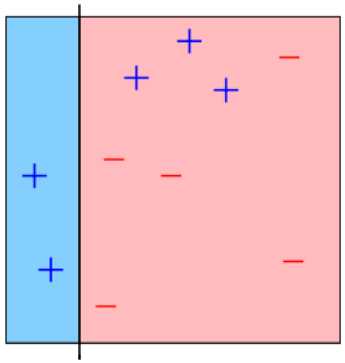
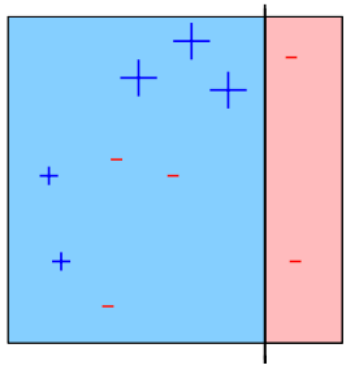
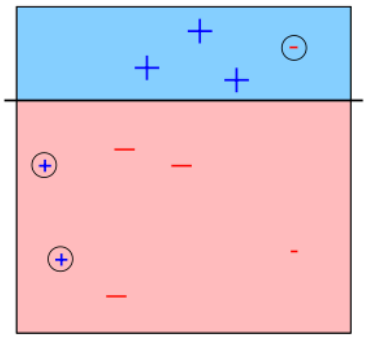


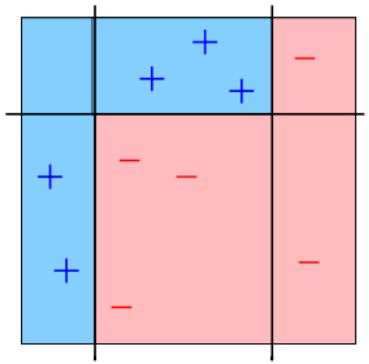




$\hat{\epsilon}_3 = 0.14$
$\alpha_3 = 0.92$

Final Classifier

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

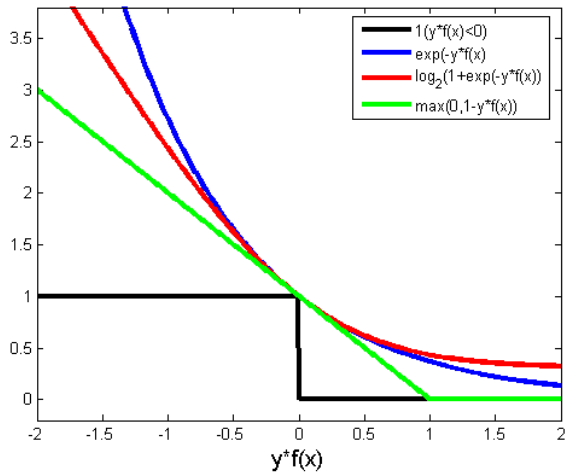
$=$ 

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

h_t behaves like random guessing on the new distribution D_{t+1}.

Corollary

A step $t+1$, AdaBoost is forced with h_{t+1} to learn something new about the underlying labelling function which was not captured by h_t.

Theorem 1 (Reminder)

Upper bound on the empirical error of H_T

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

Theorem 4

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). γ_t is the advantage of h_t over random guessing.

- This theorem means that the empirical risk exponentially decreases towards 0 with the number T of iterations.
- if $\forall t : \gamma_t \geq \gamma > 0$ then $\hat{\epsilon}_T \leq e^{-2\gamma^2 T}$

Theoretical results on the empirical risk

Proof.

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

Theoretical results on the empirical risk

Proof.

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

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

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,

$$\prod_t (Z_t) = \prod_t (2\sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)}$$

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,

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

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,

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

Theoretical results on the empirical risk

Proof.

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

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

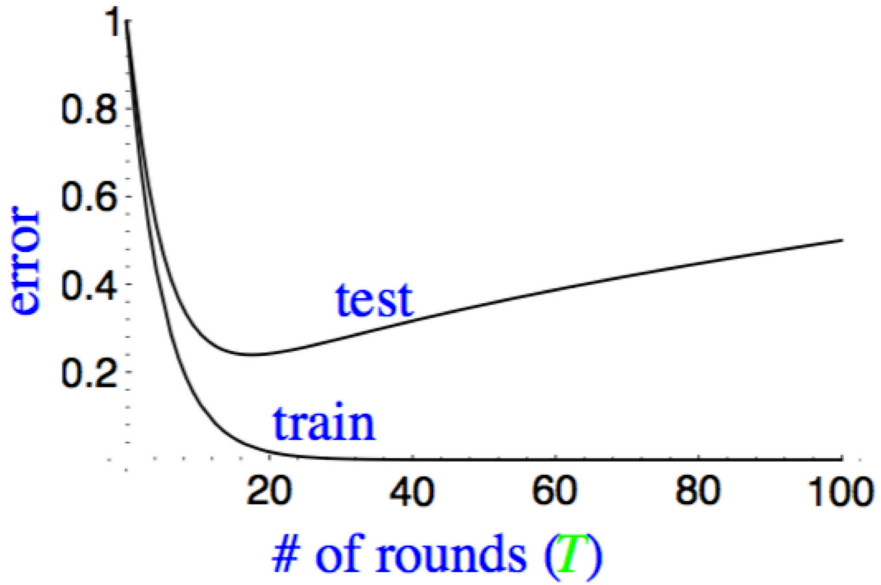
Then,

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

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

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

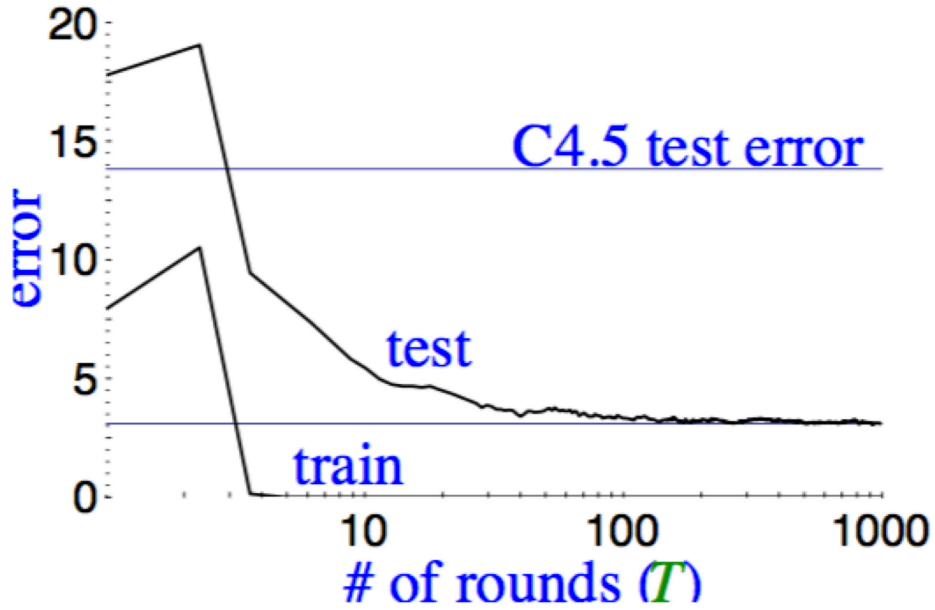
Expected behavior of boosting



Expected behavior

- $\hat{\epsilon}_{H_T}$ *continues to drop (or reaches zero).*
- ϵ_{H_T} *first decreases; then* H_T *becomes too complex:*
  - *overfitting*
  - *Occam's razor* $\rightarrow$ *hard to know when to stop training.*

Observed behavior of boosting



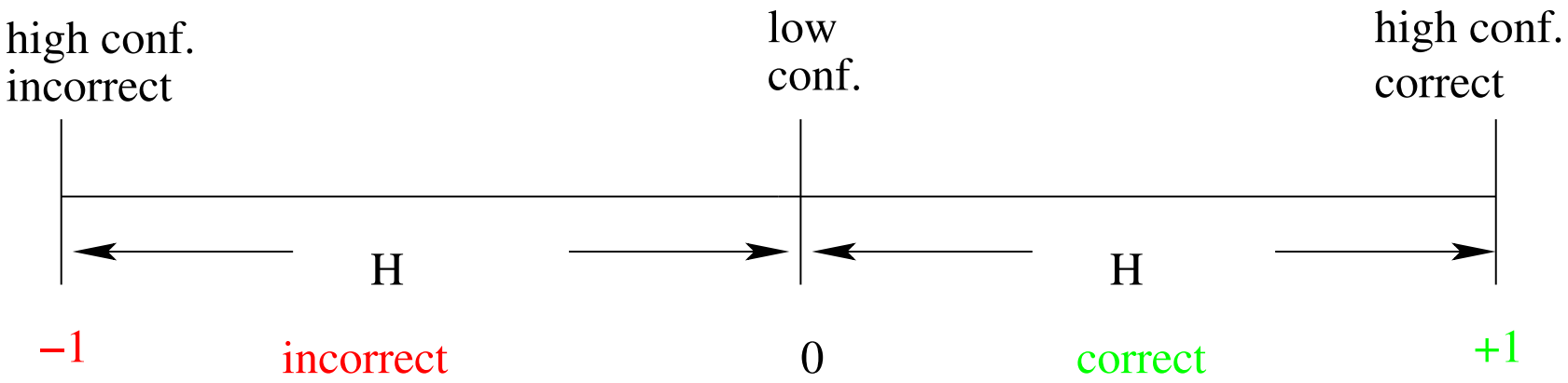
Actual typical run

- $\hat{\epsilon}_{H_T}$ continues to drop (or reaches zero).
- ϵ_{H_T} drops and continues to decrease even when $\hat{\epsilon}_T$ has reached 0!!
- Occam's razor wrongly predicts "simpler" rule is better

A better story: The margin explanation

Key idea

- Training error only measures whether classifications are right or wrong.
- Should also consider confidence of classifications.
- Recall: H_T is weighted majority vote of weak classifiers.
- Measure confidence by margin = strength of the vote.

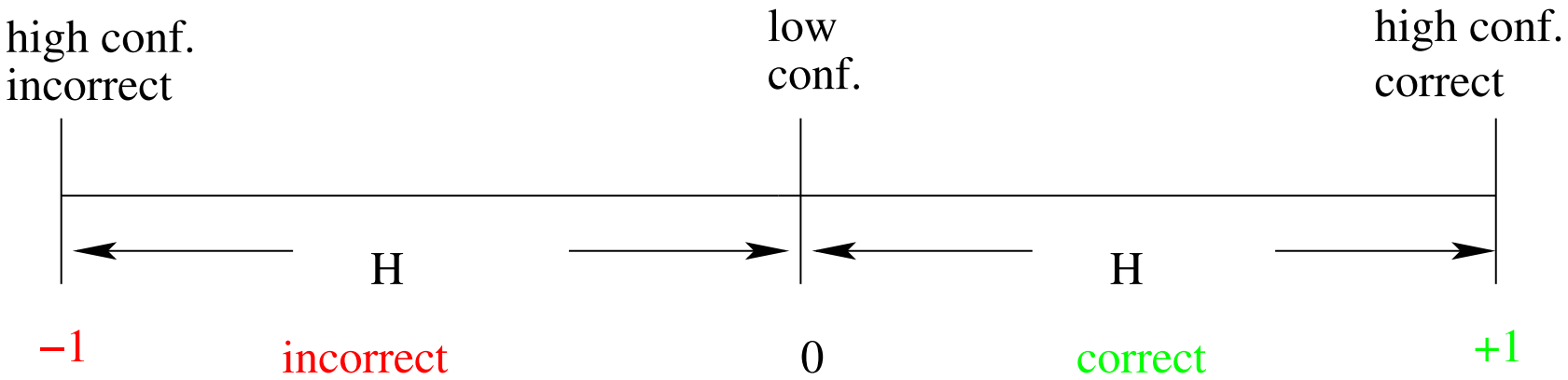


Explanation in terms of margins

Definition

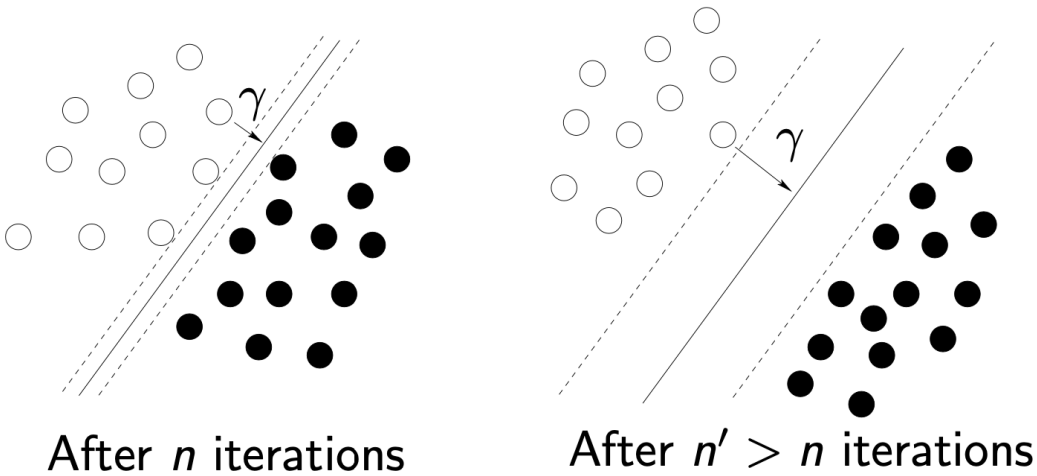
The margin of an example is defined by

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



Theoretical Evidence: Analyzing Boosting Using Margins

- **Theorem 5**: large margins $\Rightarrow$ better bound on generalization error
- **Theorem 6**: boosting tends to increase margins of training examples (given weak learning assumption)
- So, although final classifier is getting larger,
  - margins are likely to be increasing,
  - final classifier actually getting close to a simpler classifier, driving down the test error.

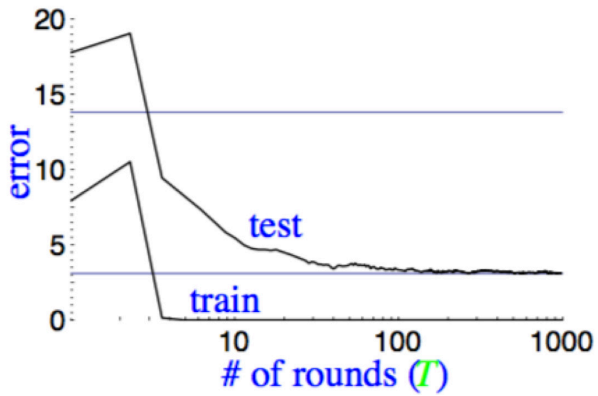


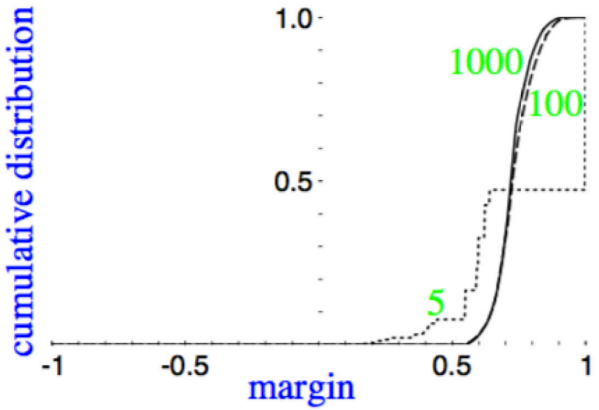
After n iterations

After $n' > n$ iterations

Empirical evidence: The margin distribution

margin distribution = cumulative distribution of margins of training examples.





	# rounds		
	5	100	1000
train 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

Theoretical results in generalization

Theorem 5

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 6

$\hat{Pr}(margin(\mathbf{x}) \leq \theta)$ exponentially decreases towards 0 with T if error of h_t on $D_t < \frac{1}{2} - \theta$ $(\forall t)$.

Margin maximization

Theorem 6

$\hat{Pr}(margin(\mathbf{x}) \leq \theta)$ exponentially decreases towards 0 with T if error of h_t on $D_t < \frac{1}{2} - \theta$ $(\forall t)$.
More formally:
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
$-y \sum_t \alpha_t h_t(\mathbf{x}) + \theta \sum_t \alpha_t \geq 0$

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

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
$-y \sum_t \alpha_t h_t(\mathbf{x}) + \theta \sum_t \alpha_t \geq 0$

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

$$\hat{Pr}(\frac{yf(\mathbf{x})}{\sum_t \alpha_t} \leq \theta) = \frac{1}{m} \sum_{(x,y)} [\frac{yf(\mathbf{x})}{\sum_t \alpha_t} \leq \theta] \quad \leq \quad \frac{1}{m} \sum_{(x,y)} exp^{-y \sum_t \alpha_t h_t(\mathbf{x}) + \theta \sum_t \alpha_t}$$

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
$-y \sum_t \alpha_t h_t(\mathbf{x}) + \theta \sum_t \alpha_t \geq 0$

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

$$\hat{Pr}(\frac{yf(\mathbf{x})}{\sum_t \alpha_t} \leq \theta) = \frac{1}{m} \sum_{(x,y)} [\frac{yf(\mathbf{x})}{\sum_t \alpha_t} \leq \theta] \leq \frac{1}{m} \sum_{(x,y)} exp^{-y \sum_t \alpha_t h_t(\mathbf{x}) + \theta \sum_t \alpha_t}$$

$$= \frac{exp^{\theta \sum_t \alpha_t}}{m} \sum_{(x,y)} exp^{-y \sum_t \alpha_t h_t(\mathbf{x})}$$

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:

$$= exp^{\theta \sum_t \alpha_t} \prod_t Z_t$$

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

Proof.

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

$$
\begin{aligned}
&= exp^{\theta \sum_t \alpha_t} \prod_t Z_t \\
&= exp^{\sum_t \theta \frac{1}{2} ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})} 2^T \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} = 2^T \prod_t exp^{ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})^{\frac{\theta}{2}}} \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} \\
&= \prod_t 2\sqrt{\frac{(1-\hat{\epsilon}_t)^{\theta}\hat{\epsilon}_t(1-\hat{\epsilon}_t)}{\hat{\epsilon}_t^{\theta}}} = \prod_t 2\sqrt{\hat{\epsilon}_t^{1-\theta}(1-\hat{\epsilon}_t)^{1+\theta}}
\end{aligned}
$$

Proof.

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

$$
\begin{aligned}
&= exp^{\theta \sum_t \alpha_t} \prod_t Z_t \\
&= exp^{\sum_t \theta \frac{1}{2} ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})} 2^T \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} = 2^T \prod_t exp^{ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})^{\frac{\theta}{2}}} \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} \\
&= \prod_t 2\sqrt{\frac{(1-\hat{\epsilon}_t)^{\theta}\hat{\epsilon}_t(1-\hat{\epsilon}_t)}{\hat{\epsilon}_t^{\theta}}} = \prod_t 2\sqrt{\hat{\epsilon}_t^{1-\theta}(1-\hat{\epsilon}_t)^{1+\theta}} \\
&= \prod_t 2\sqrt{(\frac{1}{2}-\gamma)^{1-\theta}(\frac{1}{2}+\gamma)^{1+\theta}}
\end{aligned}
$$

Proof.

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

$$
\begin{aligned}
&= exp^{\theta \sum_t \alpha_t} \prod_t Z_t \\
&= exp^{\sum_t \theta \frac{1}{2} ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})} 2^T \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} = 2^T \prod_t exp^{ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})^{\frac{\theta}{2}}} \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} \\
&= \prod_t 2\sqrt{\frac{(1-\hat{\epsilon}_t)^{\theta}\hat{\epsilon}_t(1-\hat{\epsilon}_t)}{\hat{\epsilon}_t^{\theta}}} = \prod_t 2\sqrt{\hat{\epsilon}_t^{1-\theta}(1-\hat{\epsilon}_t)^{1+\theta}} \\
&= \prod_t 2\sqrt{(\frac{1}{2}-\gamma)^{1-\theta}(\frac{1}{2}+\gamma)^{1+\theta}} \\
&= \prod_t 2\sqrt{\frac{1}{2^{1-\theta}}(1-2\gamma)^{1-\theta}\frac{1}{2^{1+\theta}}(1+2\gamma)^{1+\theta}} = \left(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}}\right)^T
\end{aligned}
$$

Proof.

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

$$
\begin{aligned}
&= exp^{\theta \sum_t \alpha_t} \prod_t Z_t \\
&= exp^{\sum_t \theta \frac{1}{2} ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})} 2^T \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} = 2^T \prod_t exp^{ln(\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t})^{\frac{\theta}{2}}} \prod_t \sqrt{\hat{\epsilon}_t(1-\hat{\epsilon}_t)} \\
&= \prod_t 2\sqrt{\frac{(1-\hat{\epsilon}_t)^{\theta}\hat{\epsilon}_t(1-\hat{\epsilon}_t)}{\hat{\epsilon}_t^{\theta}}} = \prod_t 2\sqrt{\hat{\epsilon}_t^{1-\theta}(1-\hat{\epsilon}_t)^{1+\theta}} \\
&= \prod_t 2\sqrt{(\frac{1}{2}-\gamma)^{1-\theta}(\frac{1}{2}+\gamma)^{1+\theta}} \\
&= \prod_t 2\sqrt{\frac{1}{2^{1-\theta}}(1-2\gamma)^{1-\theta}\frac{1}{2^{1+\theta}}(1+2\gamma)^{1+\theta}} = \left(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}}\right)^T
\end{aligned}
$$

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

Sketch of proof

- Observe that $(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta} = (1-4\gamma^2)\left(\frac{1+2\gamma}{1-2\gamma}\right)^{\theta}$
- This is an increasing function of θ since $\frac{1+2\gamma}{1-2\gamma} > 1$ as a consequence of $1/2 > \gamma > 0$
- If $\theta < \gamma$ it can be strictly upper bounded as $(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta} < (1-2\gamma)^{1-\gamma}(1+2\gamma)^{1+\gamma}$
- the function $\gamma \to (1-2\gamma)^{1-\gamma}(1+2\gamma)^{1+\gamma}$ is strictly upper bounded by 1 over $[0, 1/2]$.
- Thus if $\theta < \gamma$, then $(1-2\gamma)^{1-\gamma}(1+2\gamma)^{1+\gamma} < 1$, meaning the right-hand side of the theorem decreases exponentially fast.

Theoretical results in generalization

Theorem 4

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

Theorem 5

$\hat{Pr}(margin(\mathbf{x}) \leq \theta)$ exponentially decreases towards 0 with T if $\theta \leq \gamma$, that is if error of h_t on $D_t < \frac{1}{2} - \theta$ $(\forall t)$.

- If γ is too small (i.e. each classifier is too weak), θ is also small $(\theta \leq \gamma)$ and the second term is large $\rightarrow$ the bound is loose.
- If γ is too large, the classifier is too strong $\rightarrow \theta$ can be large to reduce the 2nd term, but this requires more iterations to reduce the first term. However, Adaboost can stop early (when $\hat{\epsilon}_t = 0$).

Practical advantages of AdaBoost

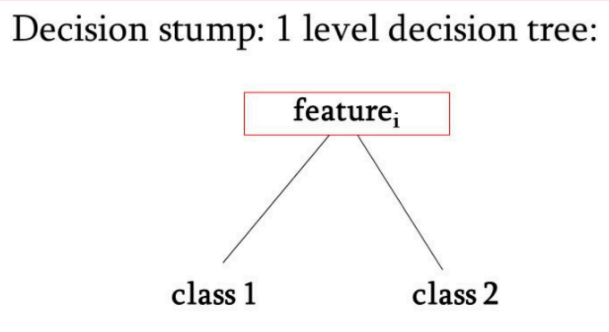
PROS

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

CONS

Performances of ADABOOST depends on data and weak learner:

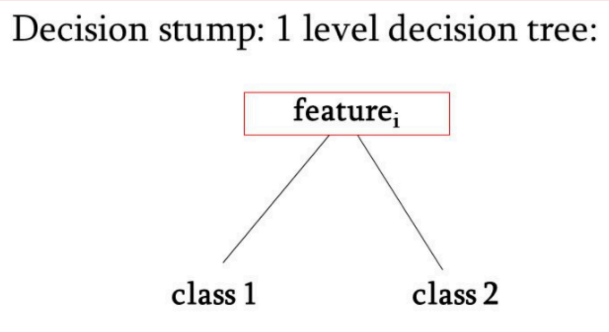
- While ADABOOST works very well with Decision Stumps, pruned decision trees (C4.5),



CONS

Performances of ADABOOST depends on data and weak learner:

- While ADABOOST works very well with Decision Stumps, pruned decision trees (C4.5),

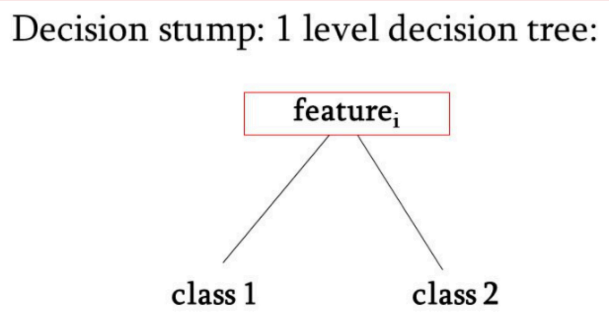


- it can fail if:
  - weak classifiers too strong ($\gamma_t \to 0.5$, *e.g.* kNN, non-pruned decision trees (ID3)) $\to$ overfitting + no diversity $\to$ stops after a few iterations ($\hat{\epsilon}_t = 0$).

CONS

Performances of ADABOOST depends on data and weak learner:

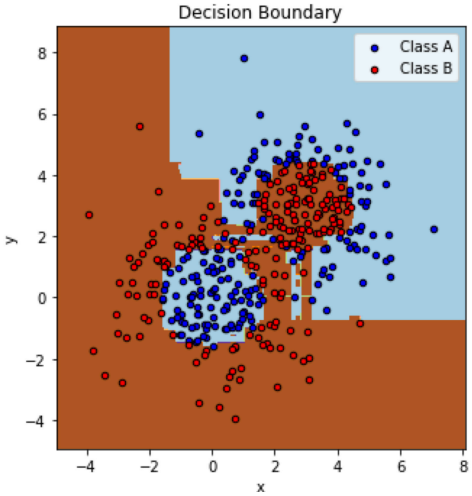
- While ADABOOST works very well with Decision Stumps, pruned decision trees (C4.5),

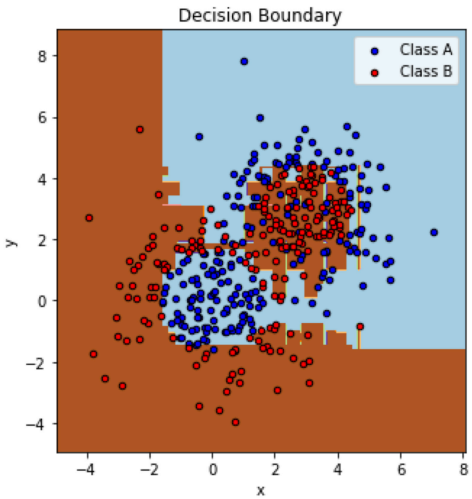


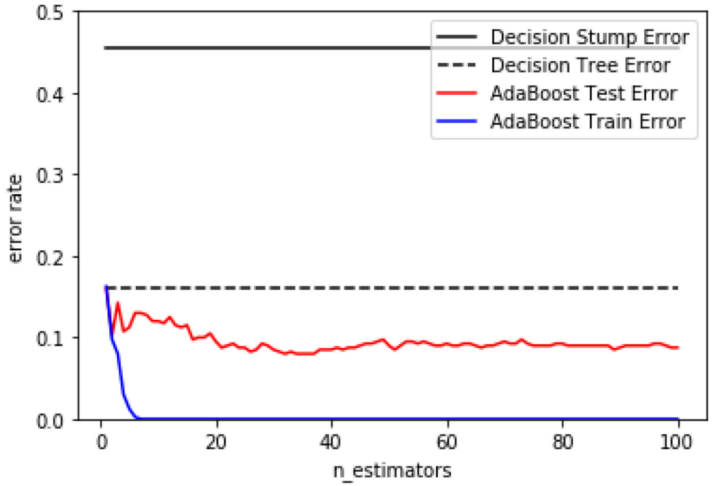
- it can fail if:
  - weak classifiers too strong ($\gamma_t \to 0.5$, *e.g.* kNN, non-pruned decision trees (ID3)) $\to$ overfitting + no diversity $\to$ stops after a few iterations ($\hat{\epsilon}_t = 0$).
  - weak classifier too weak ($\gamma_t \to 0$ too quickly and therefore the condition $\theta < \gamma_t$ is not satisfied) $\to$ underfitting or low margins.

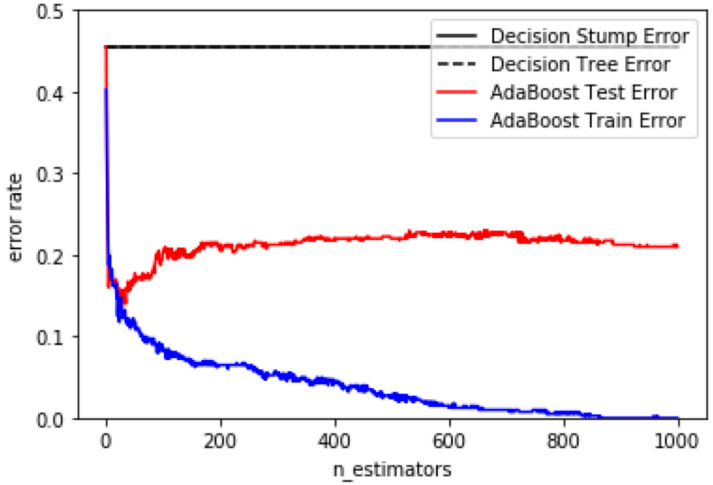
CONS (ctd)

Empirically, ADABOOST is especially sensitive to the presence of outliers $\rightarrow$ exponential increase of their weights $\rightarrow$ overfitting.



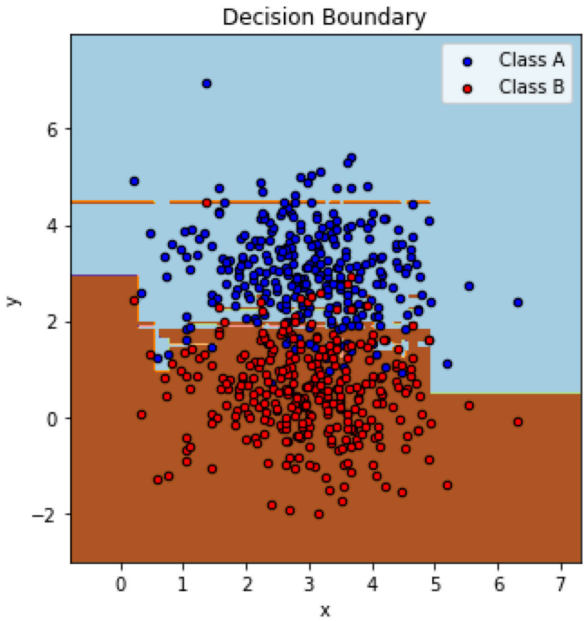


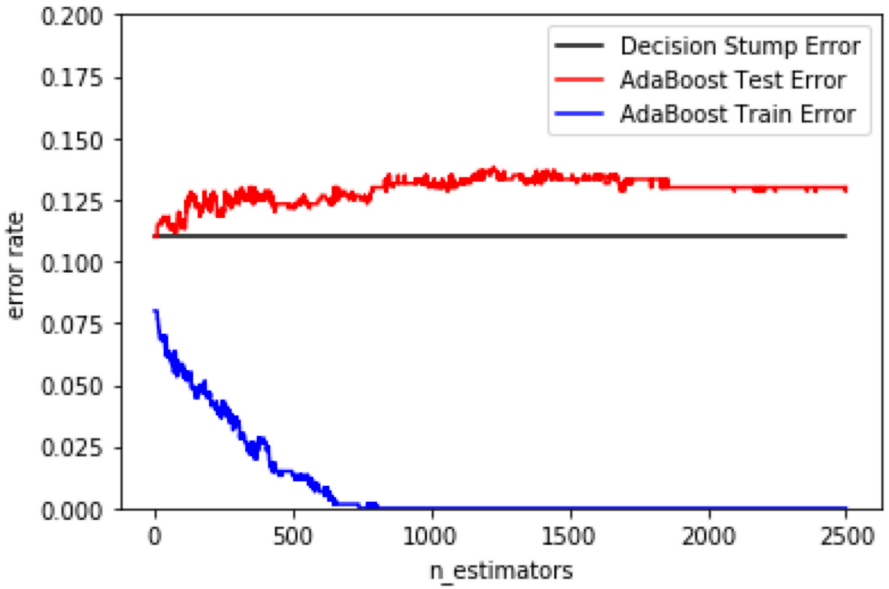




CONS (ctd)

ADABOOST is also sensitive to the presence of overlapping (bayesian error)
$\rightarrow$ slows down the convergence + overfitting.





SAMME: a multi-class boosting algorithm

(Zhu et al. 2000)

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**
  $H_T(\mathbf{x}) = sign\left(f(\mathbf{x})\right)$

SAMME

Input: *A learning set* S, K *classes, 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**

  $h_t = \mathrm{L}(S, \mathbf{D}_t)$;

  $\hat{\epsilon}_t = \sum_{\mathbf{x_i} \ t.q. \ y_i \neq h_t(\mathbf{x_i})} D_t(\mathbf{x_i})$;

  $\alpha_t = \ln \frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t} + \ln(K-1)$ *(equals 0 if random, i.e. when* $\hat{\epsilon}_t = \frac{K-1}{K}$*)*;

  for all *i from* 1 *to* m **do**

    $D_{t+1}(\mathbf{x_i}) = D_t(\mathbf{x_i}) \exp\left(\alpha_t [y_i \neq h_t(\mathbf{x_i})]\right) / Z_t$;

    /* Z_t *is a normalization coefficient*/

$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$;

Return H_T **such that**

  $H_T(\mathbf{x}) = sign\left(f(\mathbf{x})\right)$

Gradient Boosting

Adaboost versus Gradient Boosting

Adaboost

- Fit an additive model (ensemble) $f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$.
- In each stage, introduce a weak learner to compensate the shortcomings of existing learners.
- "Shortcomings" are identified by high-weight data points.
- Gradient Descent with a special loss function (exponential loss).

Adaboost versus Gradient Boosting

Adaboost

- Fit an additive model (ensemble) $f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$.
- In each stage, introduce a weak learner to compensate the shortcomings of existing learners.
- "Shortcomings" are identified by high-weight data points.
- Gradient Descent with a special loss function (exponential loss).

Gradient Boosting

- Fit an additive model (ensemble) $f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$.
- In each stage, introduce a weak learner to compensate the shortcomings of existing learners.
- "Shortcomings" are identified by gradients.
- Generalize to a variety of loss functions.

Gradient Boosting

Let's play a game...

- You are given a set of labeled data $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$.
- The task is to fit the model $f(x)$ to minimize the **square loss** $\ell(y, f(x)) = \frac{1}{2}(y - f(x))^2$.
- Let's suppose we have a model $f(x)$ at a given stage which makes some mistakes: $f(x_1) = 0.8$ while $y_1 = 0.9$ and $f(x_2) = 1.4$ while $y_2 = 1.3$
- Is it possible to improve $f(x)$ by learning a weak classifier $h(x)$ s.t.:
  - $f(x_1) + h(x_1) = y_1$
  - $f(x_2) + h(x_2) = y_2$
  - ...
  - $f(x_n) + h(x_n) = y_n$

Gradient Boosting

- The task is now to fit a model $h(x)$ to minimize the so-called residuals, i.e. the parts that existing $f(x)$ cannot do well:
  - $h(x_1) = y_1 - f(x_1)$
  - $h(x_2) = y_2 - f(x_2)$
  - ...
  - $h(x_n) = y_n - f(x_n)$

The role of $h(x)$ is to compensate the shortcoming of $f(x)$. We can typically use a **regression tree** to achieve this regression task.

Gradient Boosting

How is this related to gradient descent?

- Loss function: $\ell(y, f(x)) = \frac{1}{2}(y - f(x))^2$.
- We want to minimize $J = \sum_i \ell(y_i, f(x_i))$ by adjusting $f(x_1), f(x_2), ..., f(x_n)$.
- We get:
  - $\frac{\partial J}{\partial f(x_i)} = \frac{\partial \sum_i \ell(y_i, f(x_i))}{\partial f(x_i)} = f(x_i) - y_i$.
  - So we can interpret residuals as negative gradients: $y_i - f(x_i) = -\frac{\partial J}{\partial f(x_i)}$.

Gradient Boosting

How is this related to gradient descent?

- Loss function: $\ell(y, f(x)) = \frac{1}{2}(y - f(x))^2$.
- We want to minimize $J = \sum_i \ell(y_i, f(x_i))$ by adjusting $f(x_1), f(x_2), ..., f(x_n)$.
- We get:
  - $\frac{\partial J}{\partial f(x_i)} = \frac{\partial \sum_i \ell(y_i, f(x_i))}{\partial f(x_i)} = f(x_i) - y_i$.
  - So we can interpret residuals as negative gradients: $y_i - f(x_i) = -\frac{\partial J}{\partial f(x_i)}$.

The benefit of formulating this algorithm using gradients is that it allows us to consider **other loss functions** (convex and differentiable), and thus other task (ranking, classification) and derive the corresponding algorithms in the same way.

Gradient Boosting Algorithm

Gradient Boosting

Input: *A learning set* S, *a weak learner* L, *a loss function* ℓ
Output: *A global hypothesis* $f(x)$
Start with an initial model $f(x)$;
forall t *from* 1 *to* T **do**
    Calculate negative gradients $-g_i = -\frac{\partial \ell(y_i, f(x_i))}{\partial f(x_i)} \; \forall (x_i, y_i) \in S$;
    Fit with L *a model* $h(x)$ *to negative gradients* $-g_i$;
    $f(x) := f(x) + \alpha h_t(x)$;
    where α *is a step length;*