Where to learn better, P_X is changed

***Ensemble methods**: boosting, bagging, random forests, and Co*

***LUPI**: Learning Using Privileged Information*

Antoine Cornuéjols

AgroParisTech – INRAE UMR MIA Paris-Saclay

antoine.cornuejols@agroparistech.fr

- Here we investigate scenarios where the learning agent itself **manipulates** the input distribution P_X

Questions

- **How** to change the input distribution P_X?
- **Why** changing it?
- **Why** could it produce **good** learning **performances**?

Questions

- **How** to change the input distribution P_X? → Boosting / LUPI

- **Why** changing it?

- **Why** could it produce **good** learning **performances**?

Outline

1. Ensemble methods: boosting

2. The LUPI framework

3. Illustration on Early Classification of Time Series

4. Conclusions

- Find a **better solution**

  by combining “weak” solutions

« Ensemble » methods

Questions

1. Which **agents** ?
2. What kinds of **communication** between them if iterations?
3. How to **combine** their results?
4. How to ensure **convergence** ?
5. If convergence, **towards what**?

Nothing new under the sun

Motivation

- « *The wisdom of crowd* »

  [James Surowiecki, 2004]

  – Estimate the **weight of a bag** in a local market

    787 participants

    [Francis Galton[1], 1906 (85 years old)]

[1] anthropologue, explorateur, géographe, inventeur, météorologue, proto-généticien, psychométricien et statisticien

Motivation

- « *The wisdom of crowd* »

  [James Surowiecki, 2004]

  – Estimate the **weight of a bag** in a local market

    787 participants

    - Le **best estimation** = more than **1%** error
    - **Mean** of the estimates = less than **0.1%** error

    [Francis Galton[1], 1906 (85 years old)]

[1] anthropologist, explorer, geographer, inventor, meteorologist, proto-geneticist, psychometrician and statistician

- « Weak experts »

"Noisy" Estimations

– Unbiased

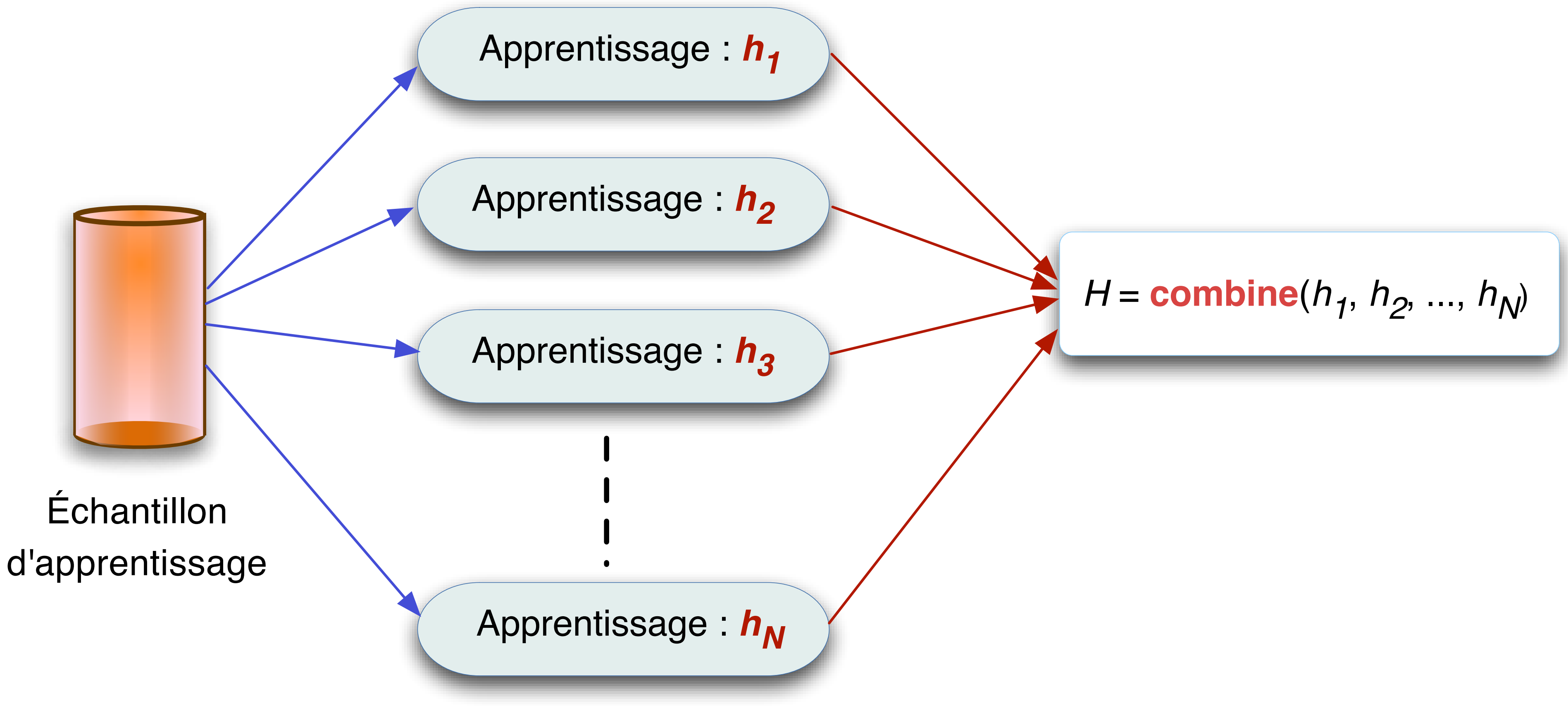
– Symmetrical

– Independent

Simple combination:

the mean

General framework: learning



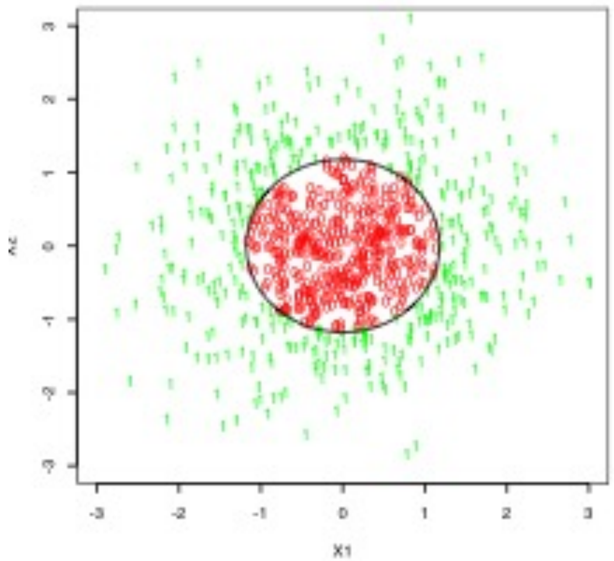
Boosting

Illustration

- Given $\mathcal{X}$ an input space with **10 dimensions**
- Independent descriptors with gaussian distribution gaussienne
- The **target concept** is defined by:

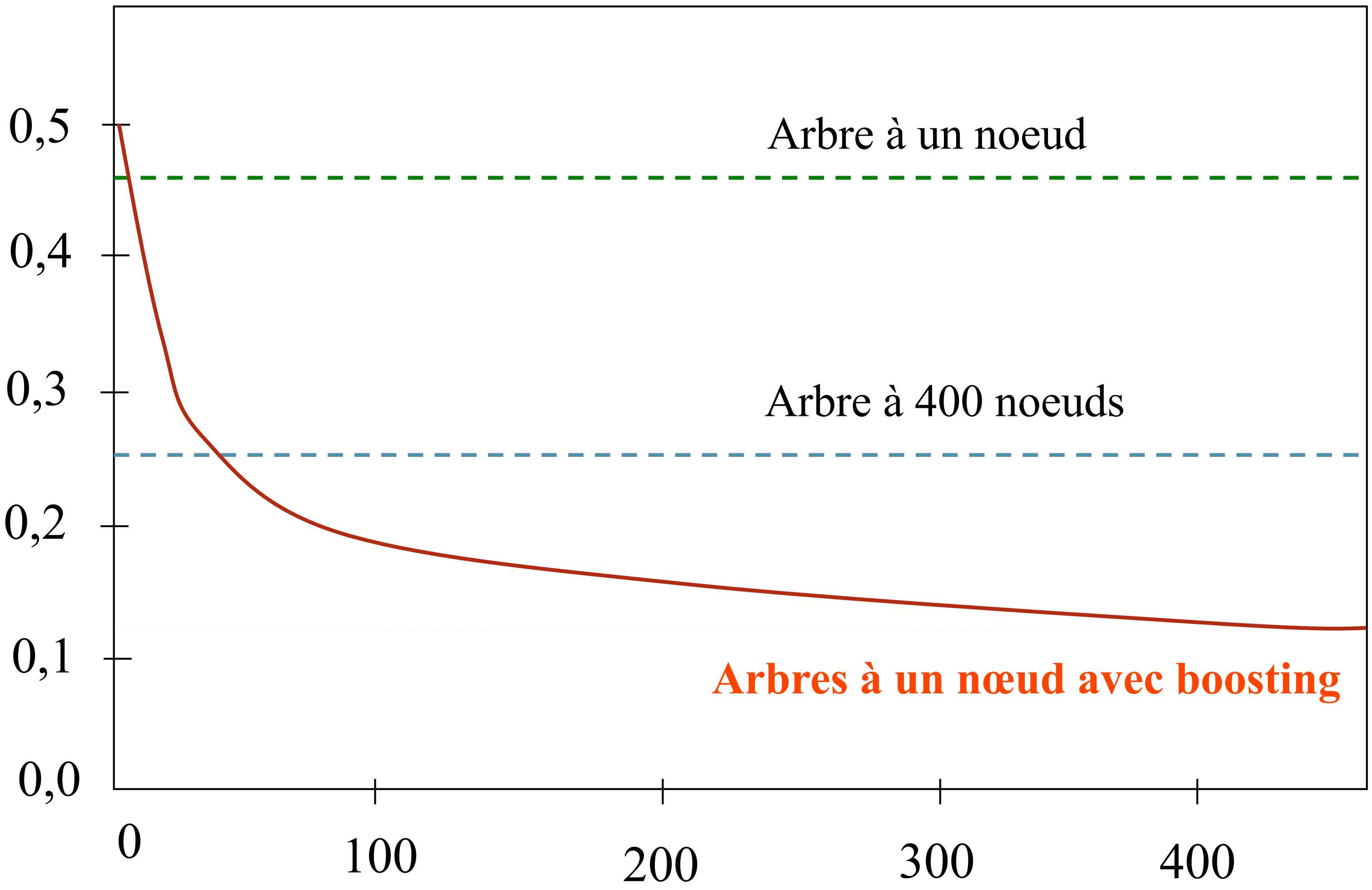
$$u = \begin{cases} 1 & \text{si } \sum_{j=1,10} x_j^2 > \chi^2_{10}(0,5) \\ -1 & \text{sinon} \end{cases}$$

With: $\chi^2_{10}(0,5) = 9,34$

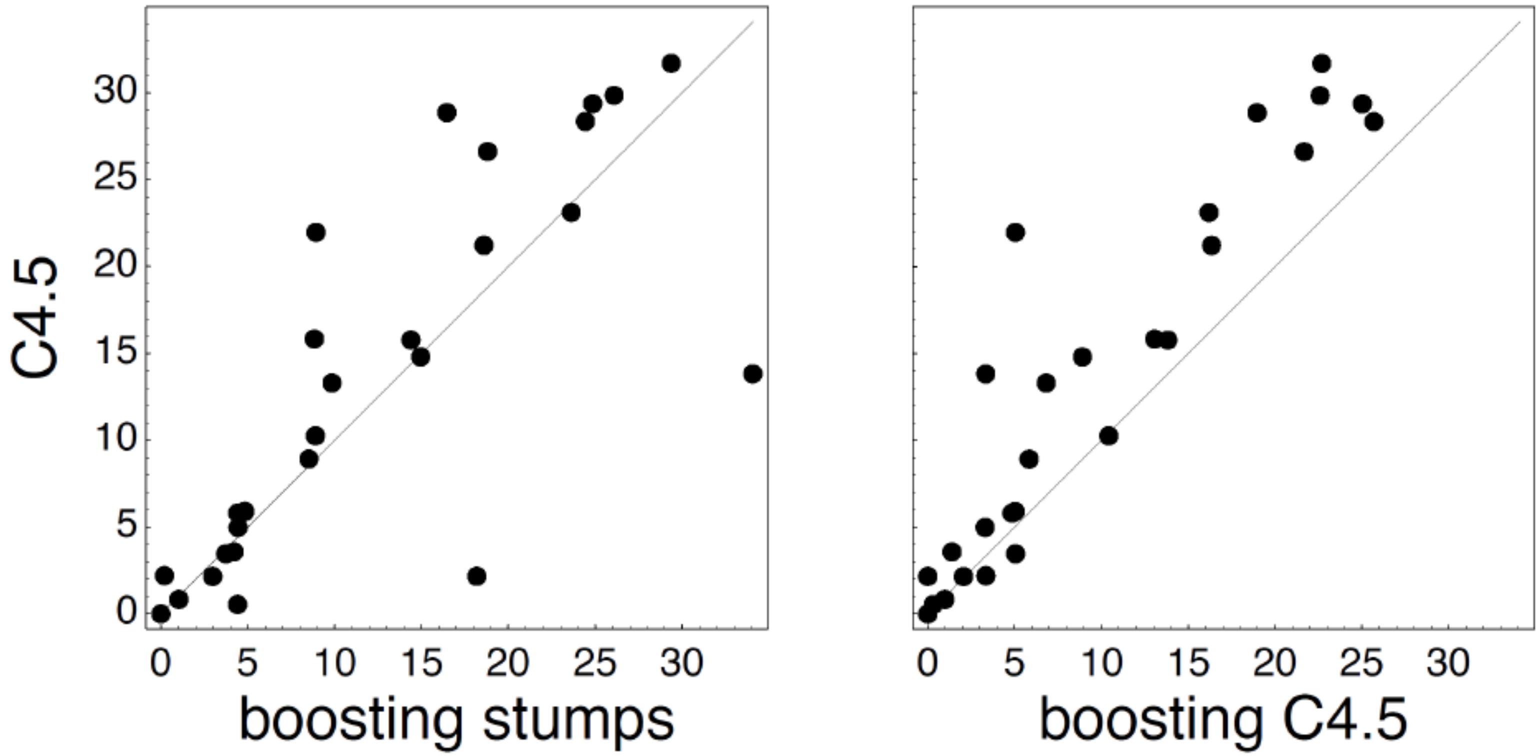


- 2000 *training examples* (1000+;1000-)
- 10000 *test examples*
- Learn **decision trees**

Illustration

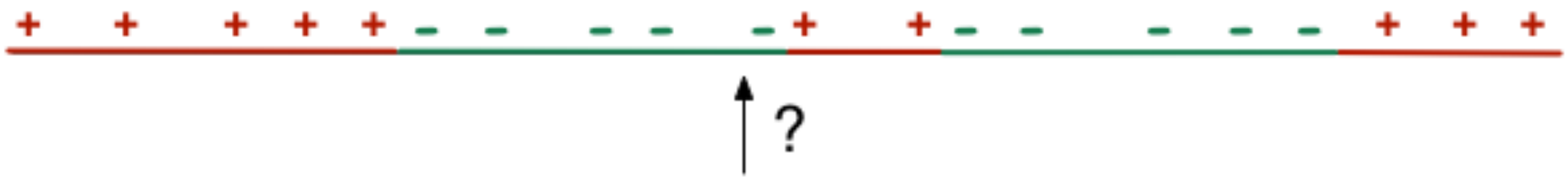


Performances using boosting



Test error rate on 27 benchmark problems
x-axis: boosting; y-axis: base-line (C4.5)

Simple example



- What is the best linear separator?

Simple example

- Error rate = 5/20 = 0.25

Simple example

- Error rate (h_1) = 5/20 = 0.25

What if I could combine it with other separators?

And with many others!

Simple example

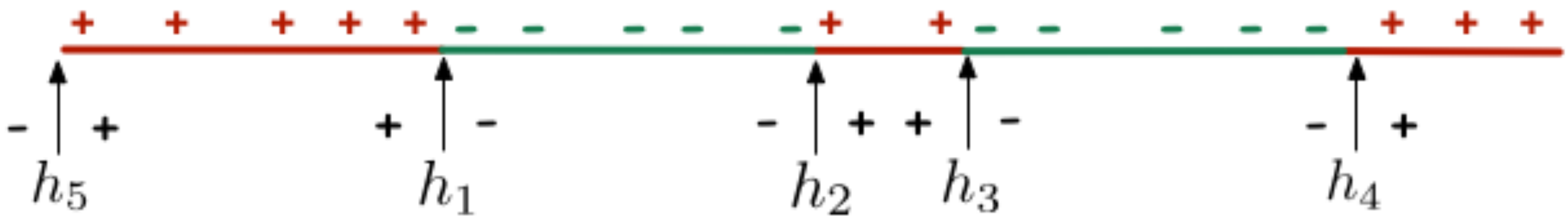
What if I could combine it with other linear separators?

Or with many others!

For instance using a **weighted vote** :

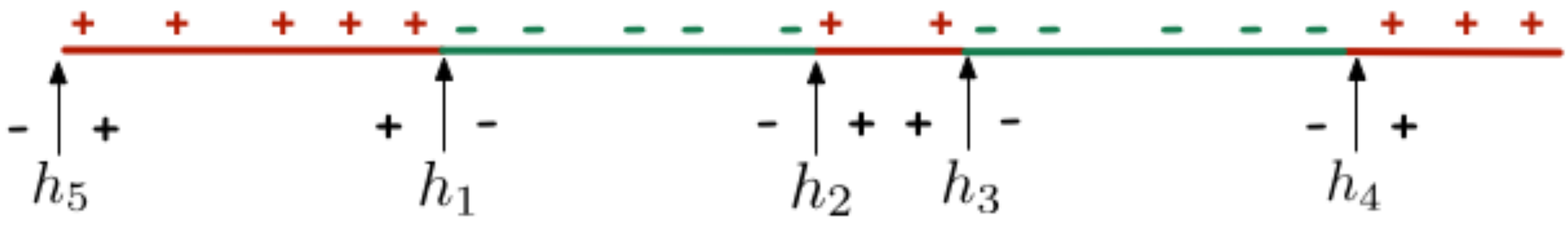
$$H(\mathbf{x}) = \text{sign}\left\{\sum_{i=1}^{l} \alpha_i \, h_i(\mathbf{x})\right\}$$

Simple example



$$H(x) = \text{sign}\{\, 0.549\, h_1(x) + 0.347\, h_2(x) + 0.310\, h_3(x) + 0.406\, h_4(x) + 0.503\, h_5(x) \,\}$$

Simple example



$$H(x) = \text{sign}\{\, 0.549\, h_1(x) + 0.347\, h_2(x) + 0.310\, h_3(x) + 0.406\, h_4(x) + 0.503\, h_5(x) \,\}$$

- How to find that kind of combination?

The boosting algorithm

The boosting algorithm

A theoretical question

- « **Strong** » learning (PAC learning)
  - A function class $\mathcal{F}$ is learnable (in a strong sense) if there exists a learning algorithm $\mathcal{A}$ which, for all distributions $\mathcal{D}_{\mathcal{X}}$ on $\mathcal{X}$, and for all functions f in $\mathcal{F}$ is such that:

$$\forall\, \varepsilon, \delta \;:\; \exists\, m(\varepsilon, \delta) \;\; \text{st.} \;\; \text{Prob}[R(h_{\mathcal{S}}) > \varepsilon] \;\leq\; \delta$$

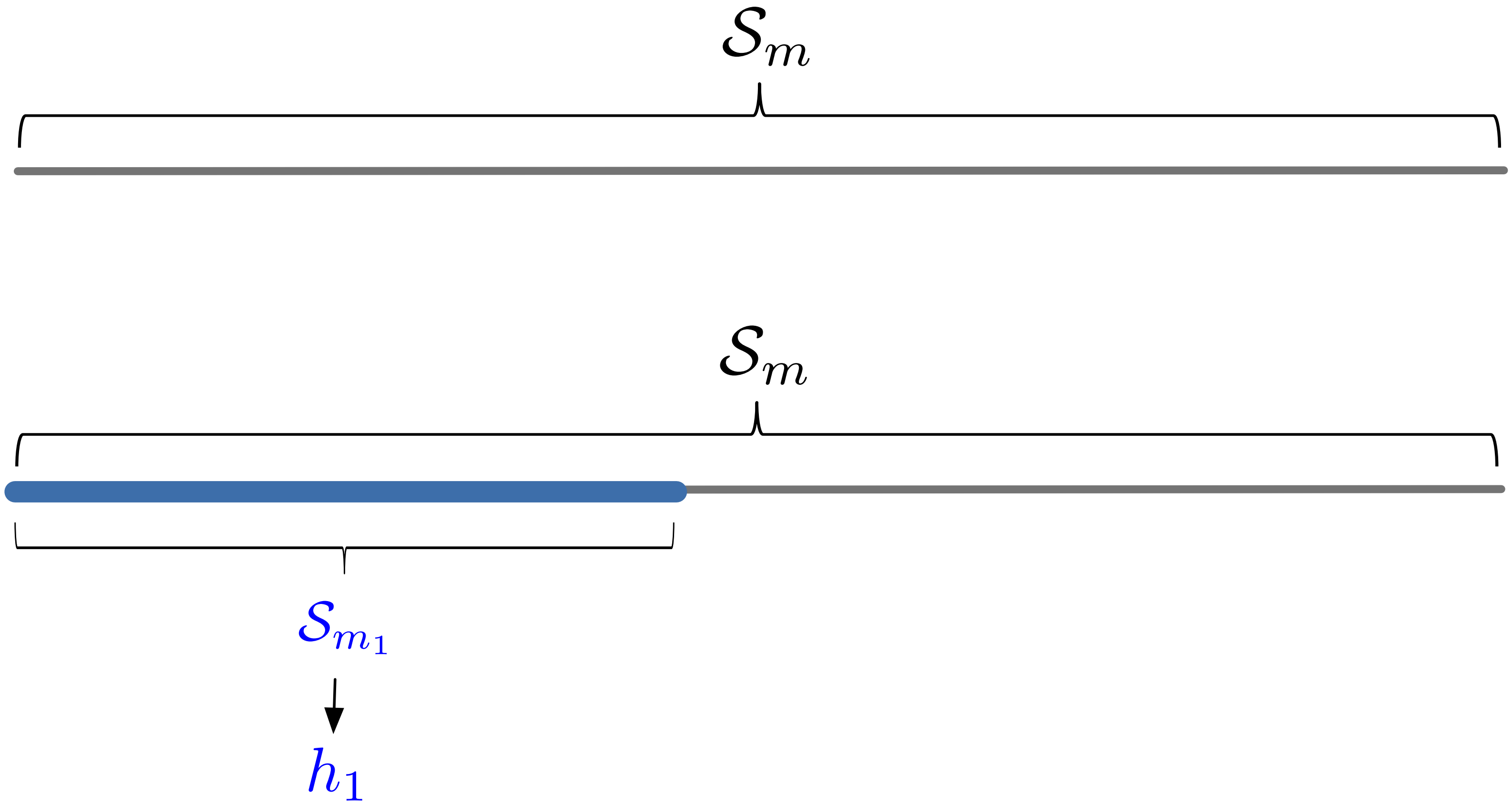
- « γ **weak** » learning
  - A function class $\mathcal{F}$ is learnable (in a weak sense) if , for $\gamma > 0$, there exists a learning algorithm $\mathcal{A}$ which, for all distributions $\mathcal{D}_{\mathcal{X}}$ on $\mathcal{X}$, and for all functions f in $\mathcal{F}$ is such that:

$$\forall\, \delta \;:\; \exists\, m(\delta) \;\; \text{st.} \;\; \text{Prob}[R(h_{\mathcal{S}}) > 1/2 - \gamma] \;\leq\; \delta$$

- Are these two function classes **different**?

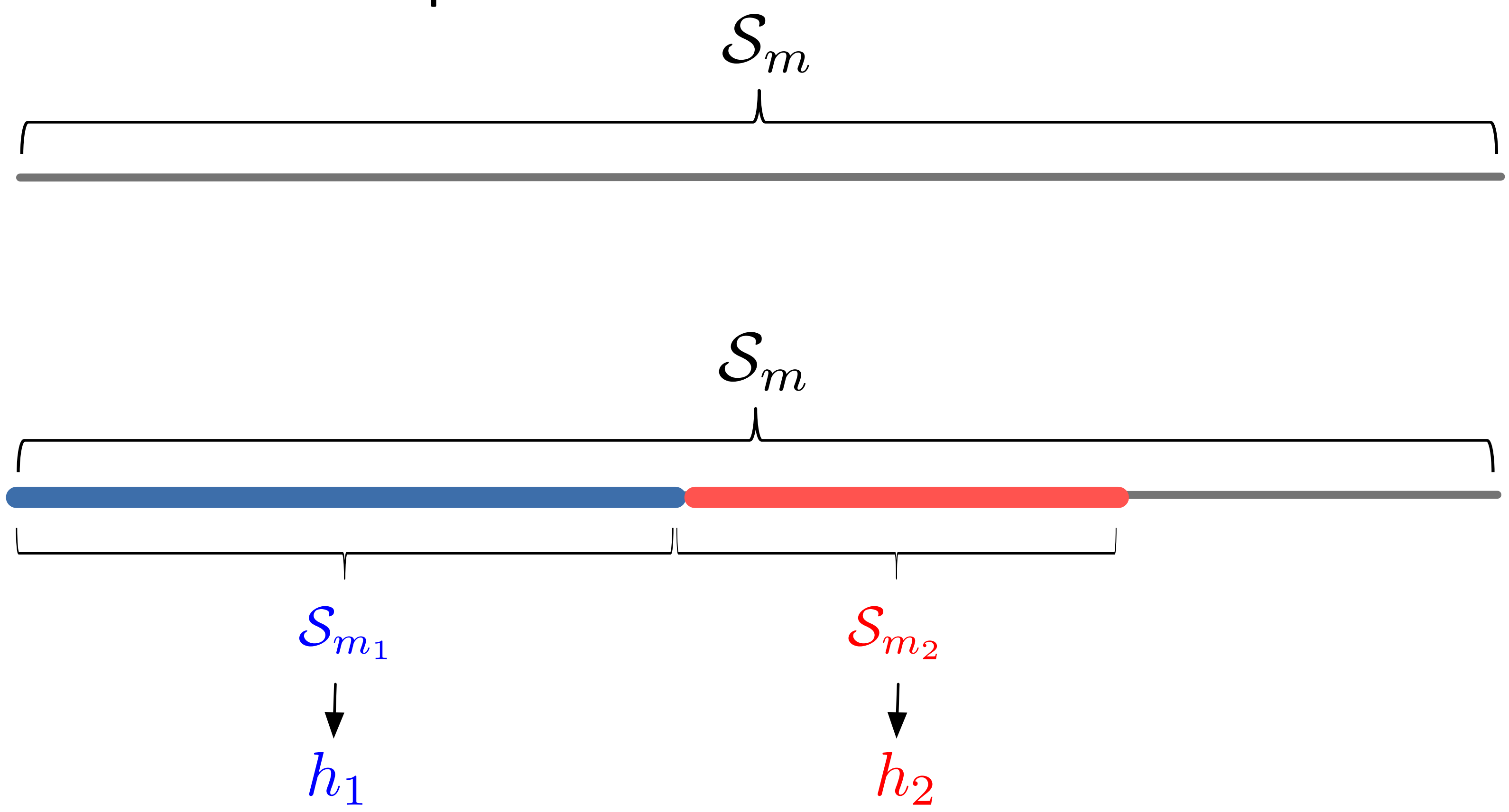
How to generate learners?

- A historical recipe



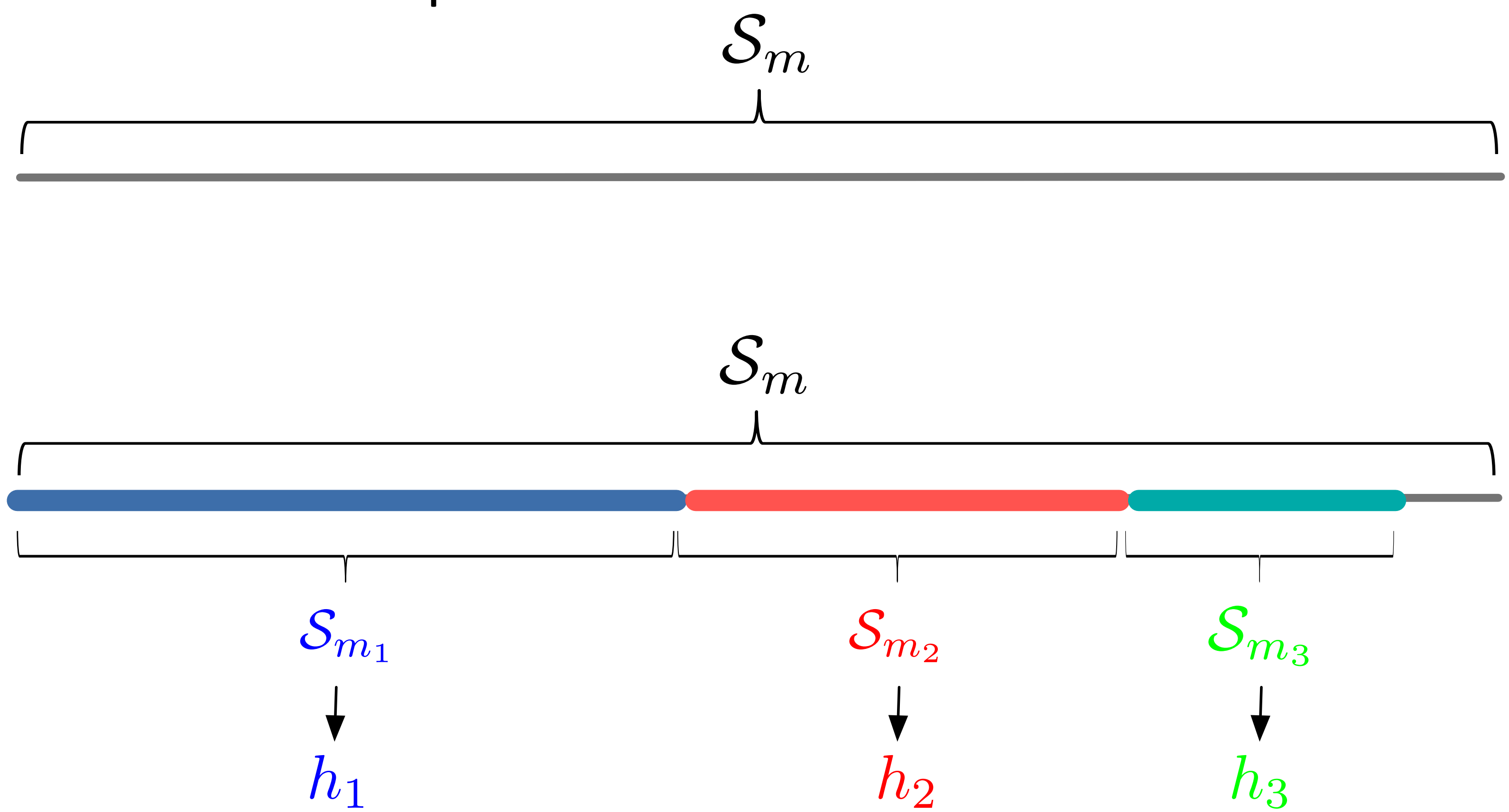
How to generate learners?

- A historical recipe



How to generate learners?

- A historical recipe



$$H(\mathbf{x}) \;=\; \operatorname{sign}\Big(h_1(\mathbf{x}) + h_2(\mathbf{x}) + h_3(\mathbf{x})\Big)$$

Questions

- How to generate **uncorrelated weak learners**?
- How to **combine** their predictions ?

Boosting

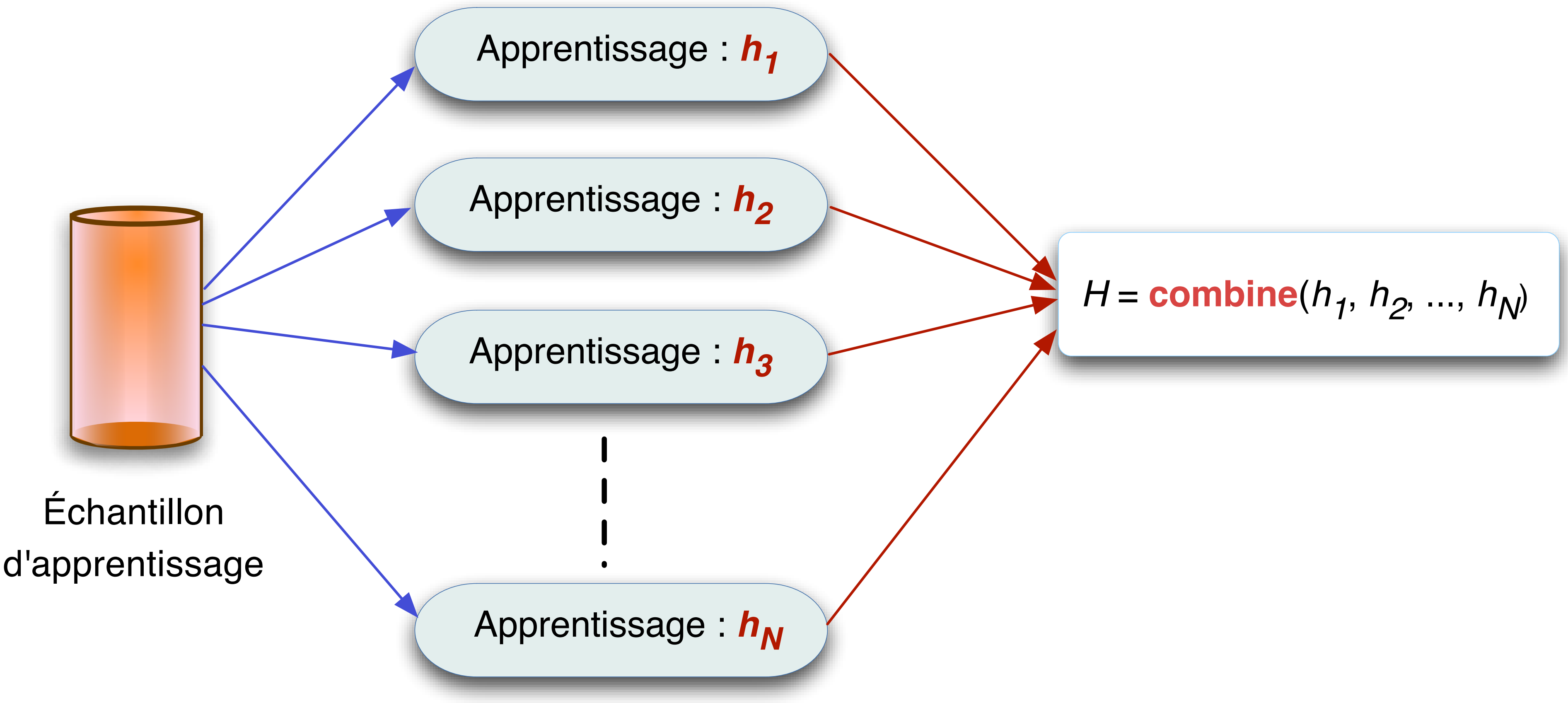
- **boosting** = a general method that allows the conversion of weak learning algorithms into a strong learning algorithm

- More precisely:

  - Given a "weak" learning algorithm which can **always** produce an hypothesis of error rate $\leq 1/2-\gamma$

  - A boosting algorithm can build (in a proven way) a decision rule (hypothesis) of error rate $\leq \varepsilon$

General schema: learning



Questions

- How to select the weak learners at each step?
  - Focus on the "hardest" examples

    (Those on which the previous learners have been the less efficient)

- How to combine the weak prediction rules into a single one?
  - Use a (weighted) vote

How to generate learners?

- Modifiy the learning sample after each learning iteration
  - *By **lowering** the weight of the **correctly labeled** examples*
  - *By **Increasing** --------------------- **incorrectly** ---------------*
  - By how much?

Boosting: formal view

- Given the training set $S = \{(x_1,y_1),\ldots,(x_m,y_m)\}$
- $y_i \in \{-1,+1\}$ being the label of example $x_i \in S$

- For all $t = 1,\ldots,T$:

  Compute the current distribution D_t over$\{1,\ldots,m\}$

  Find a weak hypothesis

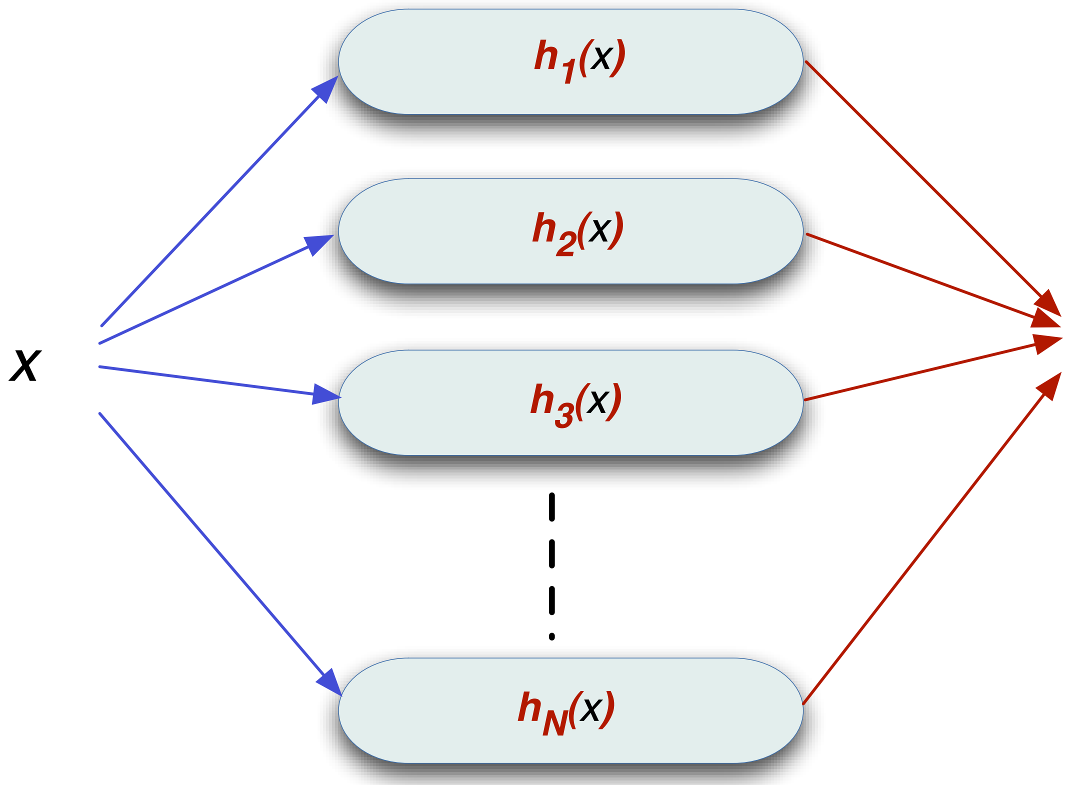
  $h_t : S \to \{-1,+1\}$

  with small error ε_t on D_t:

$$\varepsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$$

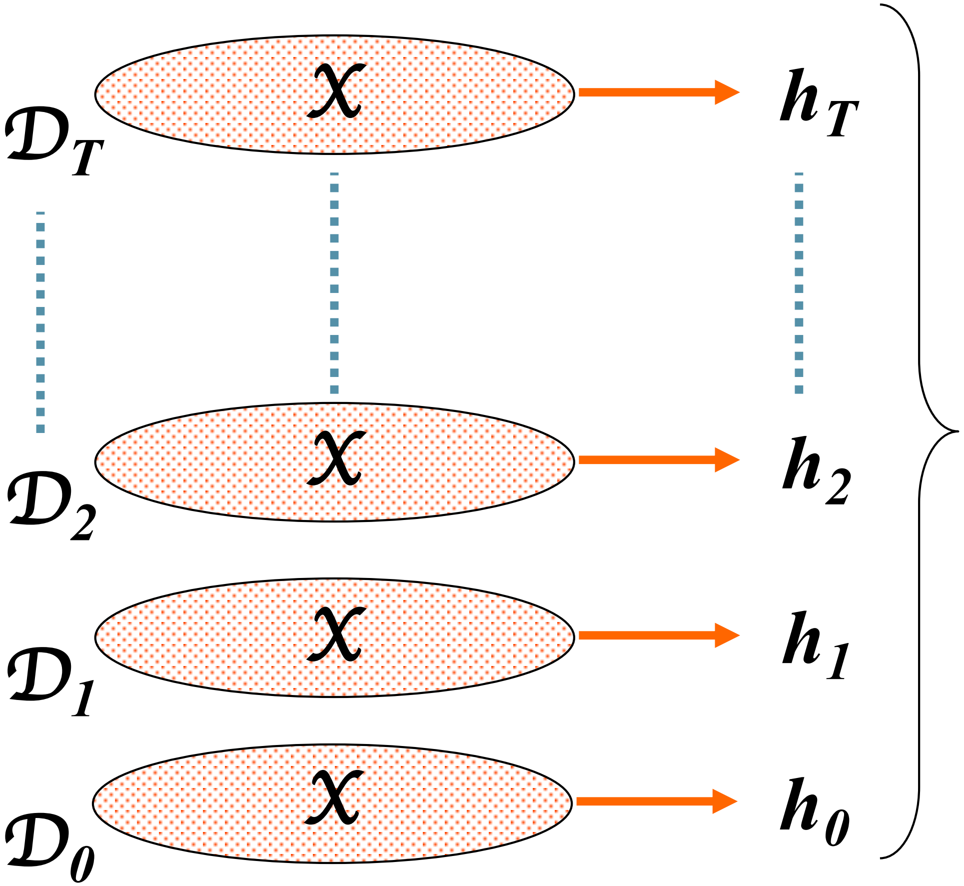
- Return the <u>final hypothesis</u> H

General scheme: prediction



$$H(\mathbf{x}) = \sum_{i=1}^{N} w_i \, h_i(\mathbf{x})$$

General principle



$$H_{finale}(\boldsymbol{x}) \;=\; \mathrm{sgn}\left[\sum_{t=0}^{T} \alpha_t \,.\, h_t(\boldsymbol{x})\right]$$

- How to **compute** $\mathcal{D}_{t+1}$ **from** $\mathcal{D}_t$**?**
- How to **compute** the **weight** α_t **?**

AdaBoost [Freund&Schapire '97]

- Define D_t: $$D_1(i) = \frac{1}{m}$$

Given D_t and h_t:

$$D_{t+1} = \frac{D_t}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$
$$= \frac{D_t}{Z_t} \cdot \exp(-\alpha_t \cdot y_i \cdot h_t(x_i))$$

where: Z_t = normalization constant

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) > 0$$

- Final hypothesis: $$\boxed{H_{\text{final}}(x) = \text{sgn}\left(\sum_t \alpha_t h_t(x)\right)}$$

AdaBoost

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) > 0$$

$$D_{t+1} = \frac{D_t}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$H_{\text{final}}(x) = \operatorname{sgn}\left(\sum_t \alpha_t h_t(x)\right)$$

Simple example

- Error rate = 5/20 = 0.25

$$\alpha_i = \frac{1}{2}\ln\frac{1-\varepsilon}{\varepsilon} = \frac{1}{2}\ln\frac{0.75}{0.25} = 0.549$$

Simple example

- New weights of the training examples

- Examples correctly labeled

$$p_b(x) = \frac{e^{-\alpha}}{Z} = \frac{e^{-0.549}}{Z} = \frac{0.577}{Z}$$

- Examples incorrectly labeled

$$p_m(x) = \frac{e^{\alpha}}{Z} = \frac{e^{0.549}}{Z} = \frac{1.732}{Z}$$

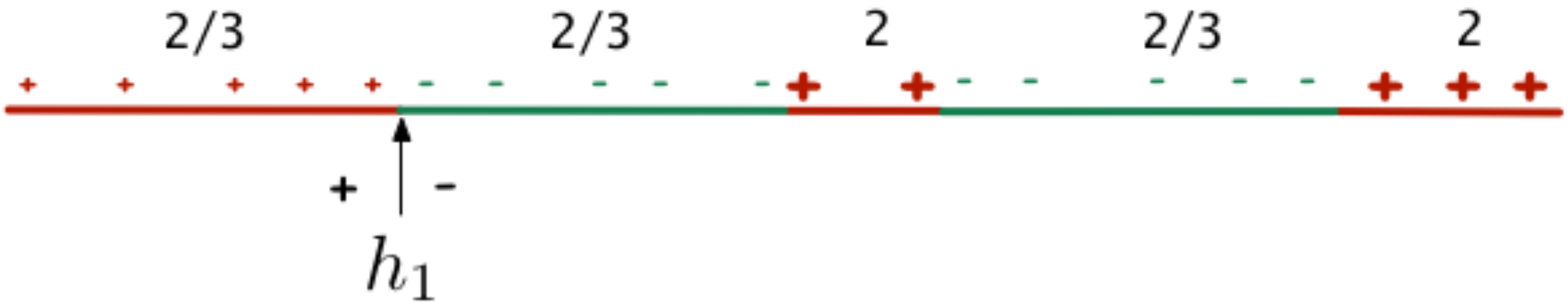
$$Z = \frac{(15 \times 0.577) + (5 \times 1.732)}{20} = \frac{8.660 + 8.660}{20} = \frac{17.32}{20} = 0.866$$

$$\boxed{p_b(x) = 0.666 = 2/3}$$

$$\boxed{p_m(x) = 2}$$

Simple example

- **Nouvelle pondération** des exemples d'apprentissage



- Examples **correctly** labeled

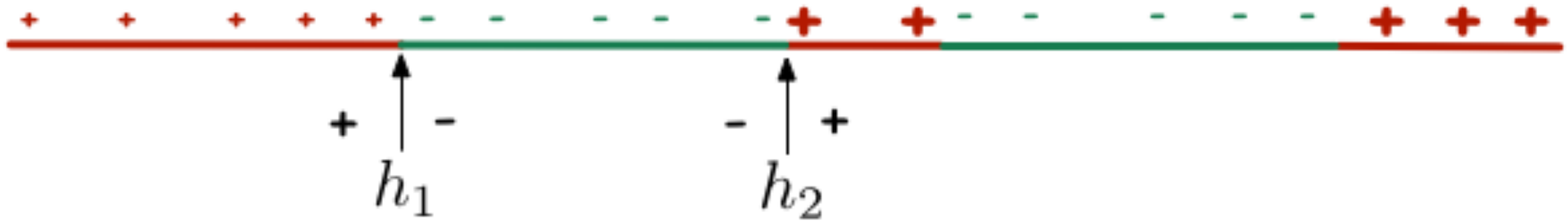
$$\boxed{p_b(x) = \frac{1}{2\,(1-\varepsilon)}} = \frac{1}{2 \times 0.75} = \frac{1}{1.5} = \frac{2}{3}$$

- Examples **incorrectly** labeled

$$\boxed{p_m(x) = \frac{1}{2\,\varepsilon}} = \frac{1}{2 \times 0.25} = \frac{1}{0.5} = 2$$

$$\boxed{Z = 2\,\varepsilon^{1/2}\,(1-\varepsilon)^{1/2}}$$

Simple example



- Error rate:

$$\varepsilon_2 = \frac{10 \times 2/3}{20} = \frac{1}{3}$$

$$\alpha_2 = \frac{1}{2} \ln \frac{1-\varepsilon_2}{\varepsilon_2} = \frac{1}{2} \ln \frac{2/3}{1/3} = 0.347$$

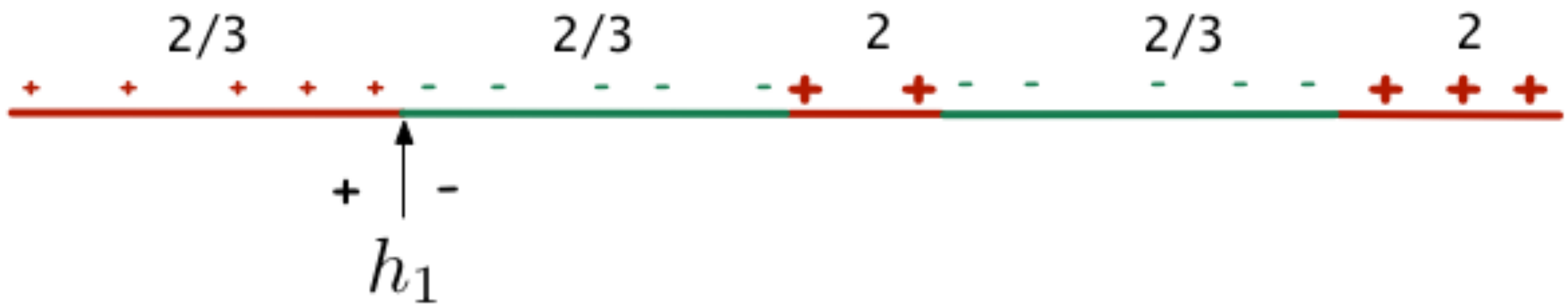
- Sous-pondération des bien classés :

$$p_b(x) = \frac{1}{2\,(1-\varepsilon)} = \frac{1}{2 \times 2/3} = \frac{3}{4} = 0.75$$

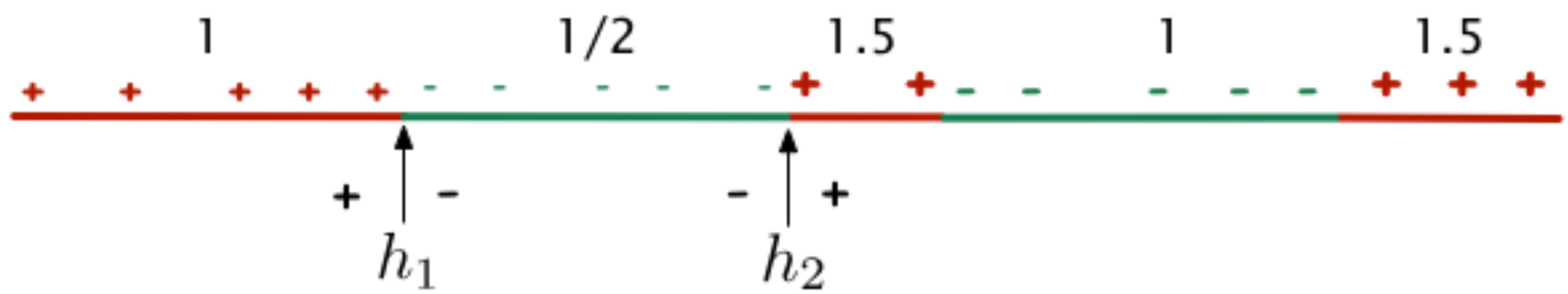
- Sur-pondération des mal classés :

$$p_m(x) = \frac{1}{2\,\varepsilon} = \frac{1}{2 \times 1/3} = \frac{3}{2} = 1.5$$

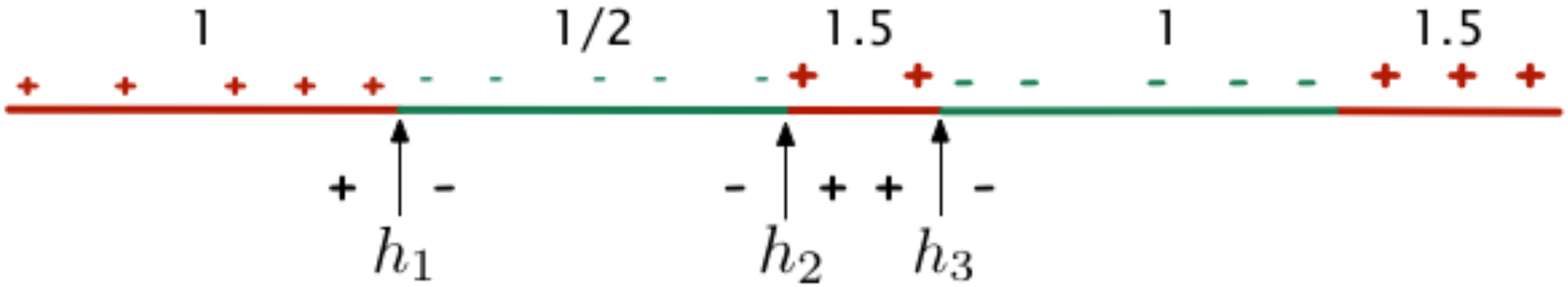
Simple example



- Sous-pondération des bien classés : $$p_b(x) = \frac{1}{2\,(1-\varepsilon)} = \frac{1}{2 \times 2/3} = \frac{3}{4} = 0.75$$
- Sur-pondération des mal classés : $$p_m(x) = \frac{1}{2\,\varepsilon} = \frac{1}{2 \times 1/3} = \frac{3}{2} = 1.5$$



Simple example

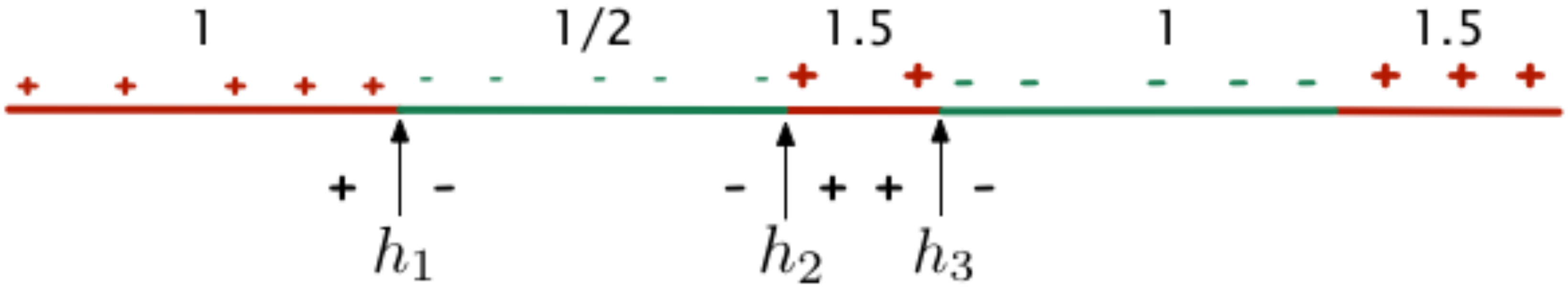


- Taux d'erreur : $\varepsilon_3 = \frac{(5 \times 1/2) + (3 \times 1.5)}{20} = \frac{7}{20} = 0.35$

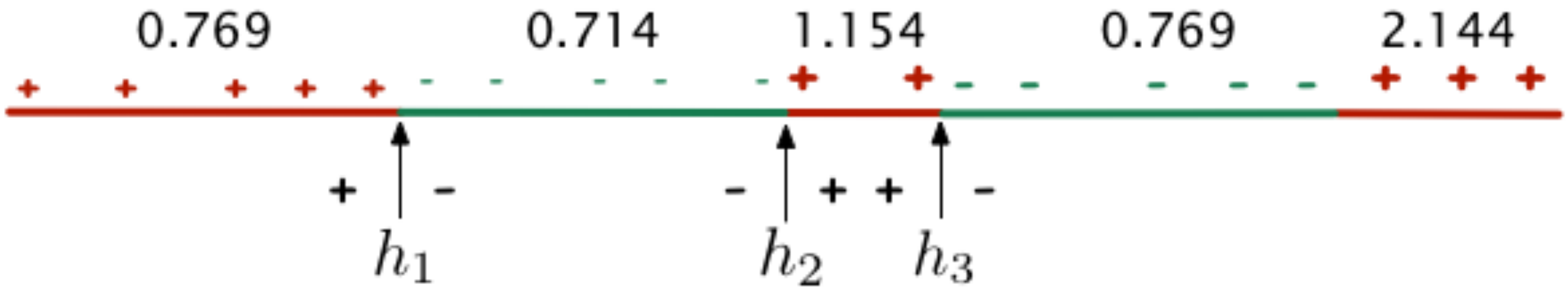
$$\alpha_3 = \frac{1}{2} \ln \frac{1 - \varepsilon_2}{\varepsilon_2} = \frac{1}{2} \ln \frac{0.65}{0.35} = 0.310$$

- Sous-pondération des bien classés : $p_b(x) = \frac{1}{2\,(1-\varepsilon)} = \frac{1}{2 \times 0.65} = \frac{1}{1.3} = 0.769$
- Sur-pondération des mal classés : $p_m(x) = \frac{1}{2\,\varepsilon} = \frac{1}{2 \times 0.35} = \frac{1}{0.7} = 1.429$

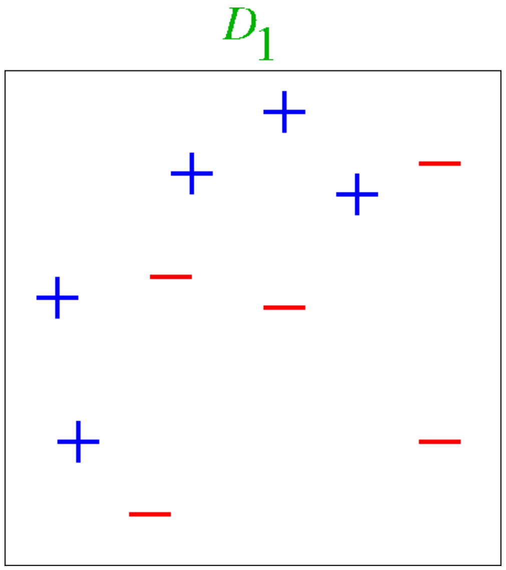
Simple example



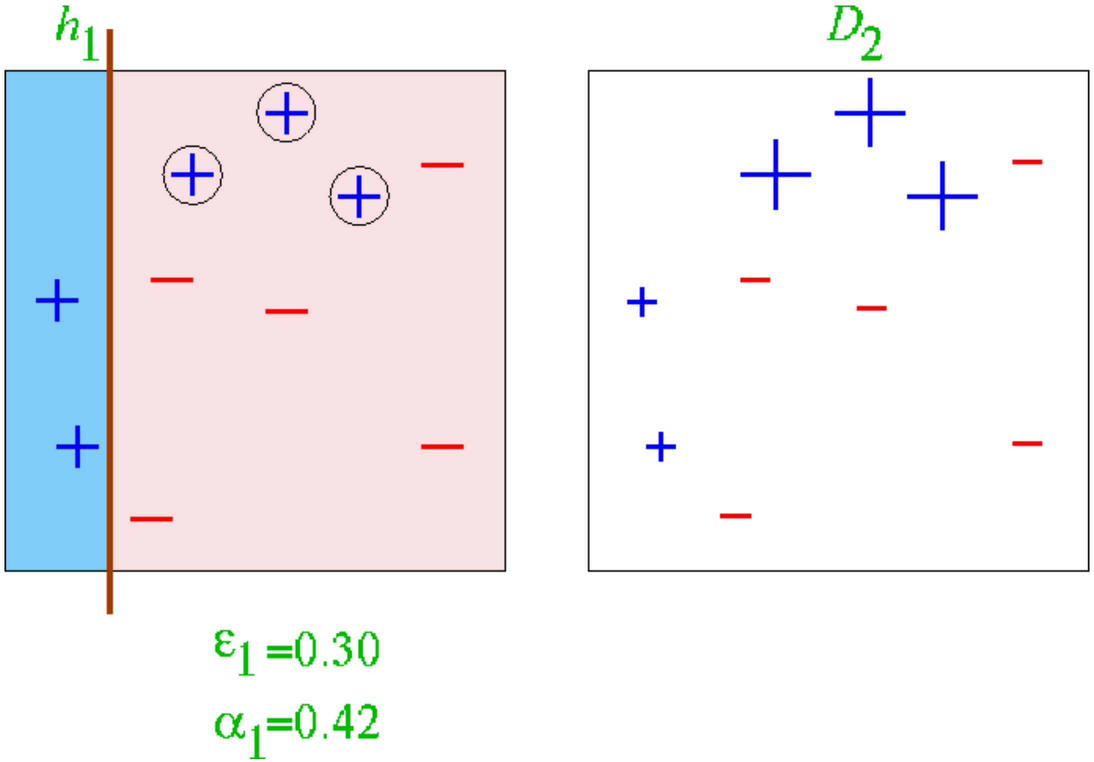
- Sous-pondération des bien classés : $$p_b(x) = \frac{1}{2\,(1-\varepsilon)} = \frac{1}{2 \times 0.65} = \frac{1}{1.3} = 0.769$$
- Sur-pondération des mal classés : $$p_m(x) = \frac{1}{2\,\varepsilon} = \frac{1}{2 \times 0.35} = \frac{1}{0.7} = 1.429$$



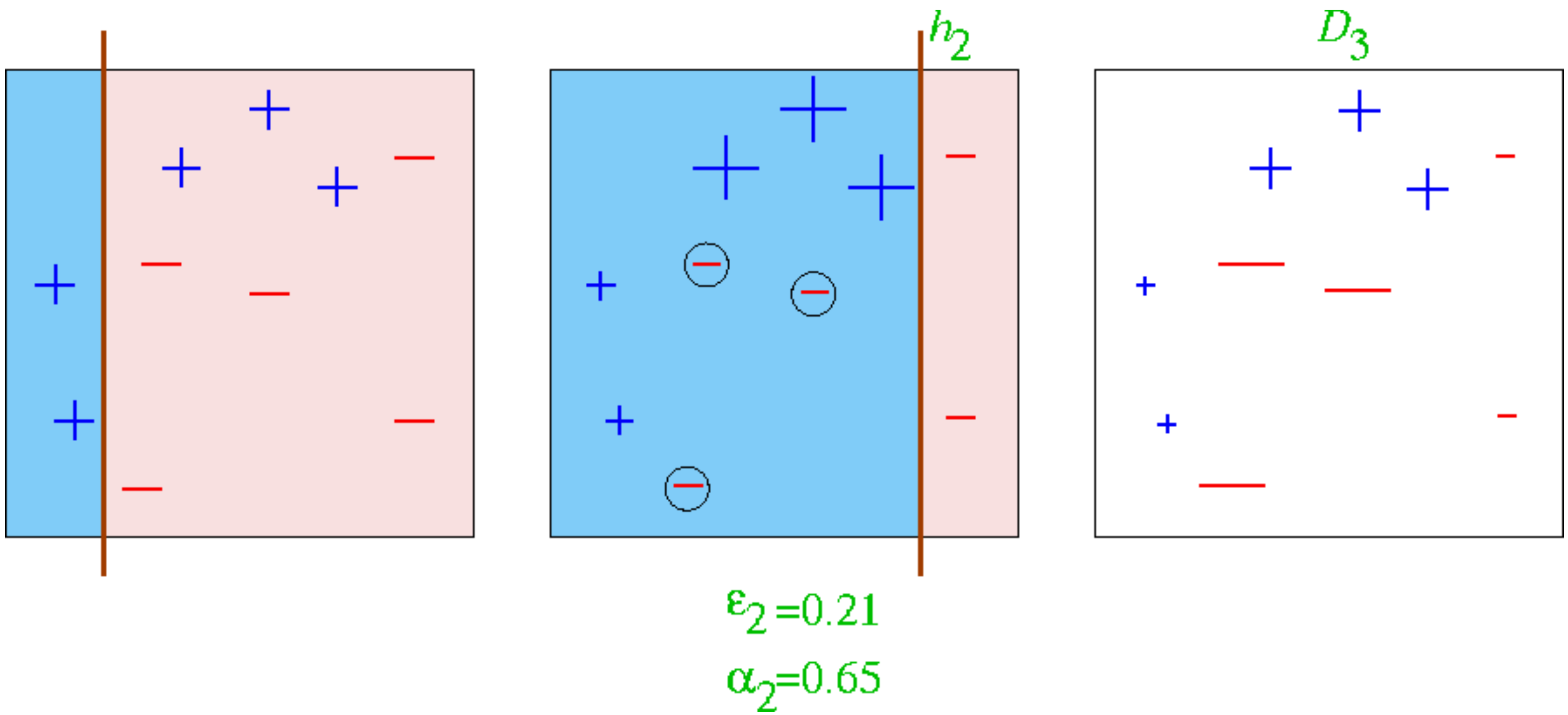
Toy example



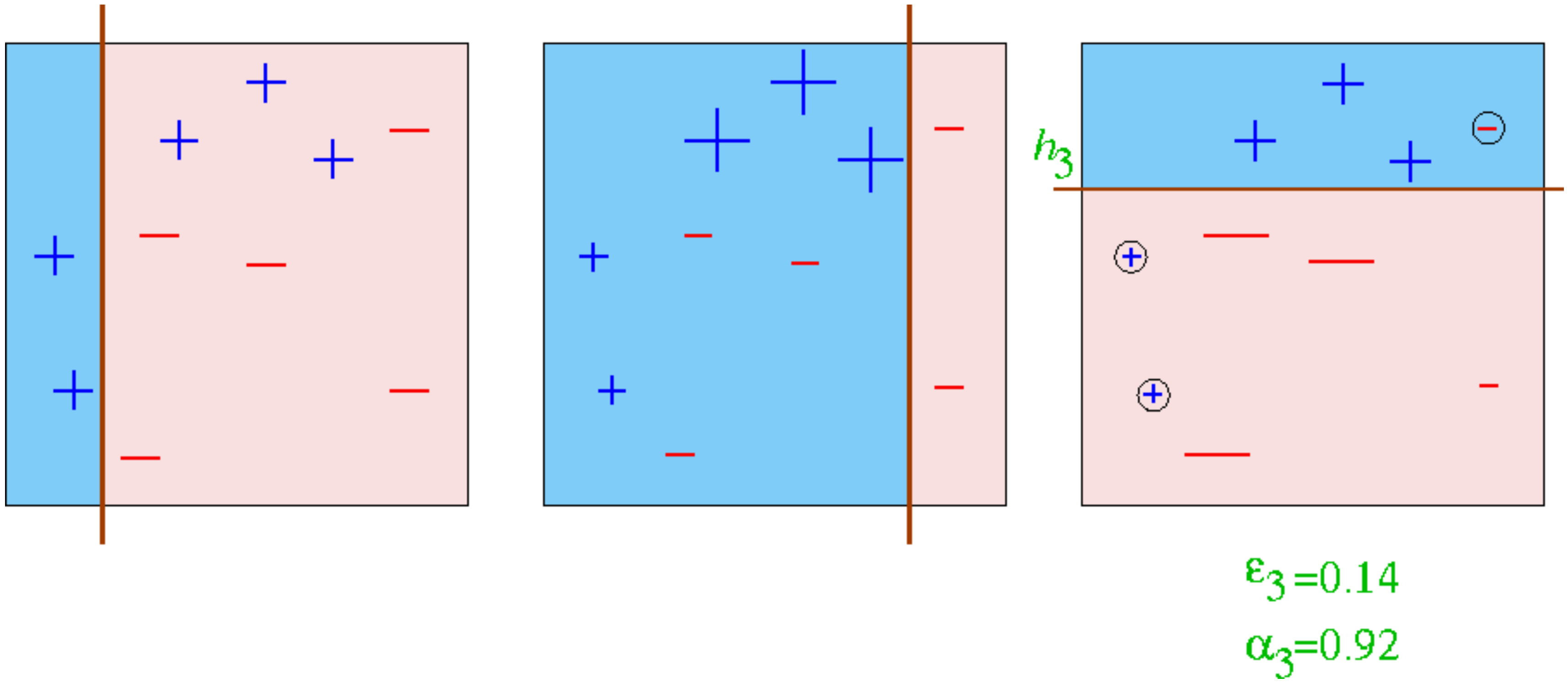
Step 1



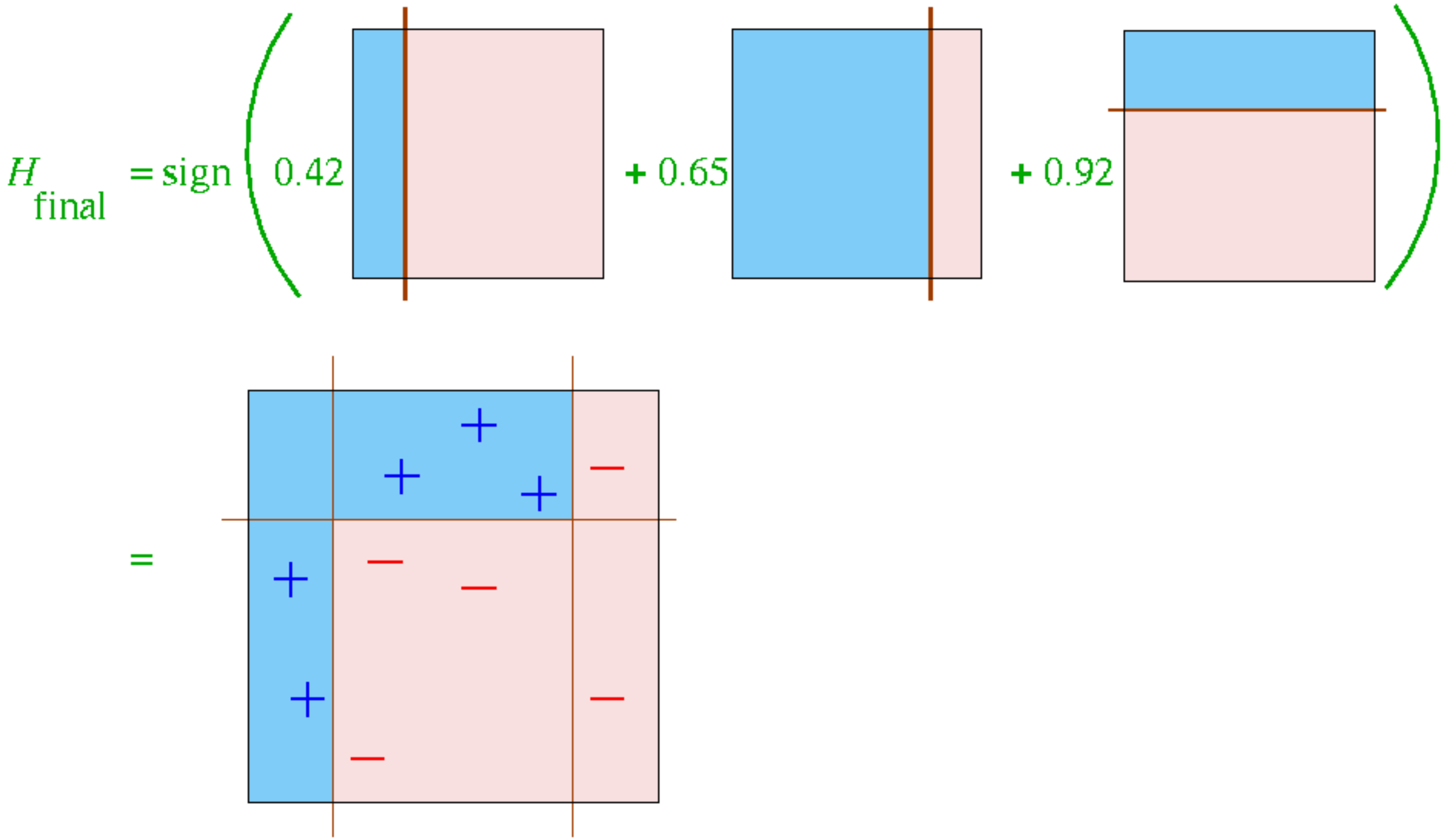
Step 2



Step 3

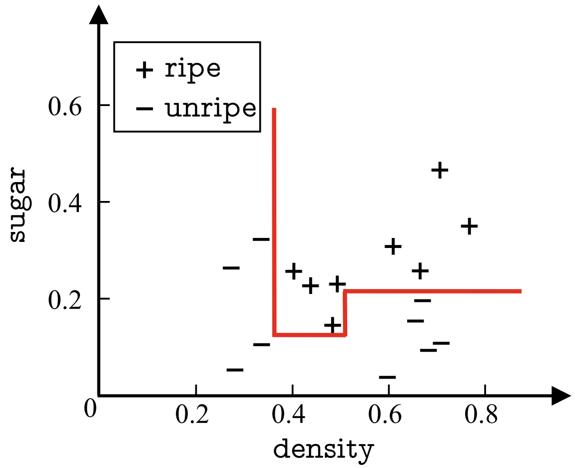


Final Hypothesis



Autre illustration du boosting sur jeu de données

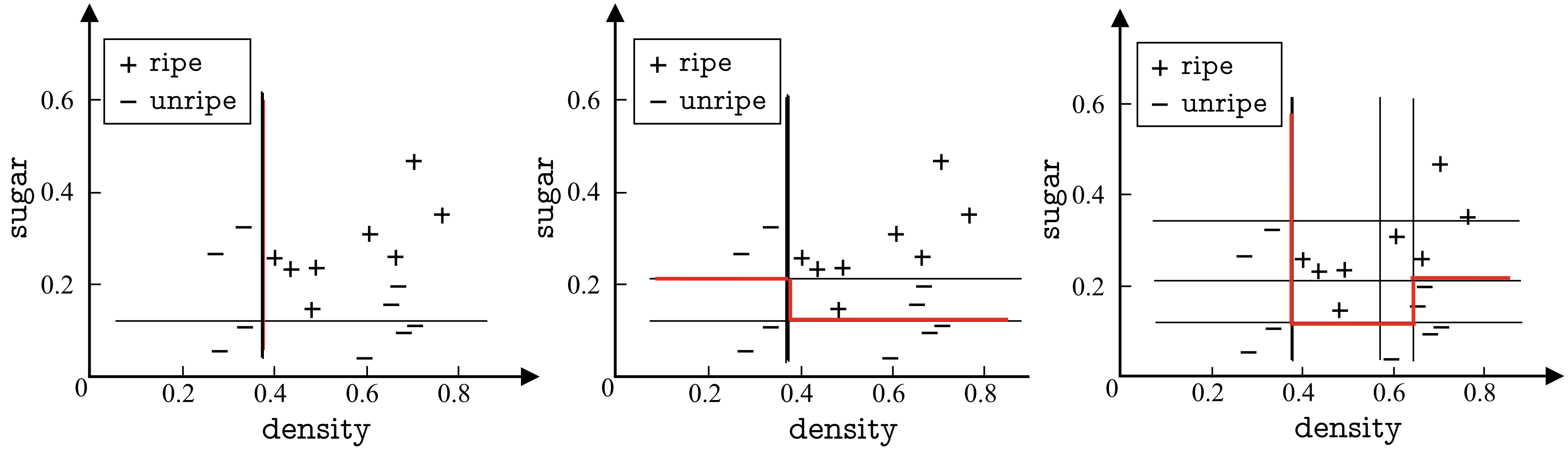
ID	density	sugar	ripe
1	0.697	0.460	true
2	0.774	0.376	true
3	0.634	0.264	true
4	0.608	0.318	true
5	0.556	0.215	true
6	0.403	0.237	true
7	0.481	0.149	true
8	0.437	0.211	true
9	0.666	0.091	false
10	0.243	0.267	false
11	0.245	0.057	false
12	0.343	0.099	false
13	0.639	0.161	false
14	0.657	0.198	false
15	0.360	0.370	false
16	0.593	0.042	false
17	0.719	0.103	false



Apprentissage par

arbre de décisions

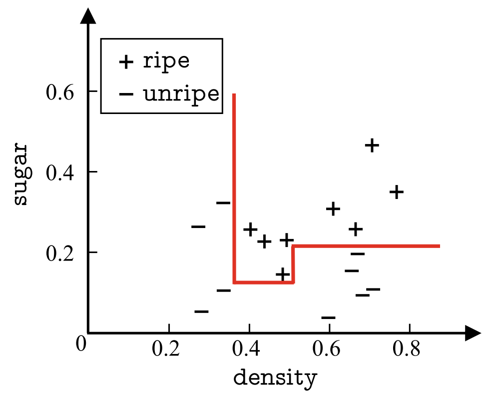
*Autre illustration du **boosting** sur jeu de données*



(a) 3 base learners. (b) 5 base learners. (c) 11 base learners.

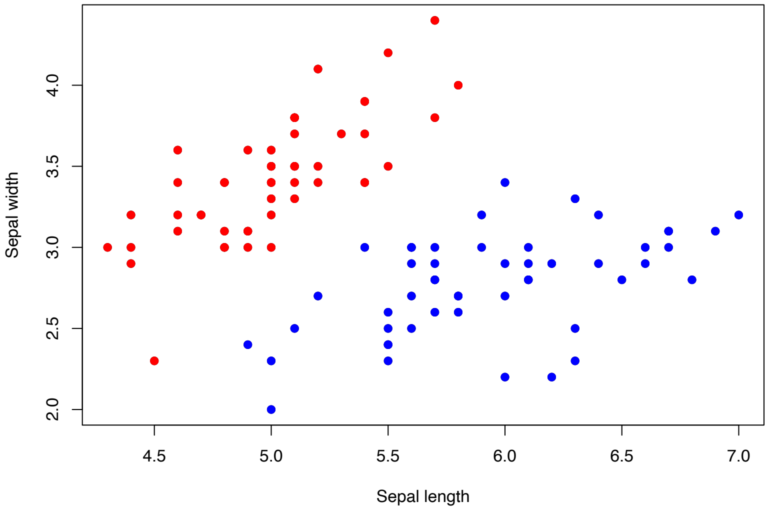
Here "**base learner**" = decision stump

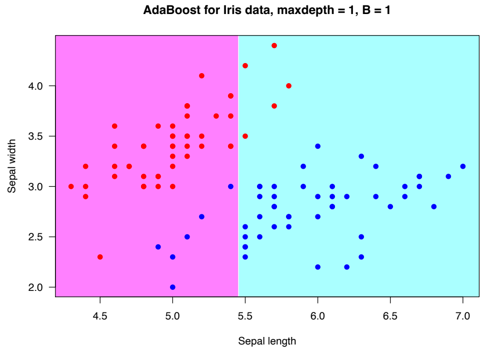
Arbre de décisions

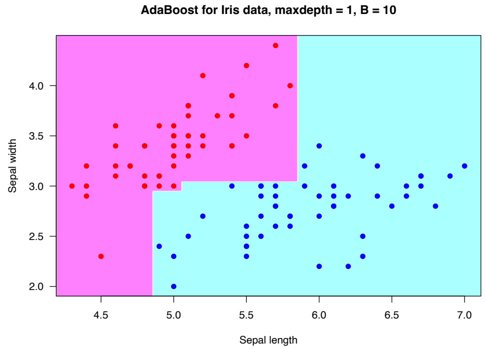


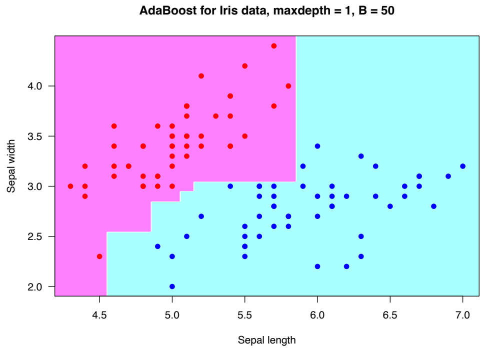
From [Zhi-Hua ZHOU « Machine Learning ». Springer, 2021]

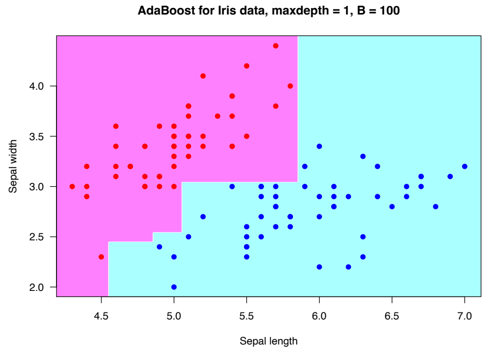
Boosting sur jeu de données « Iris » (Setosa vs. Versicolor)



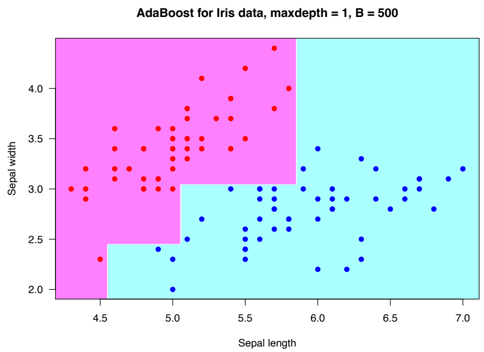




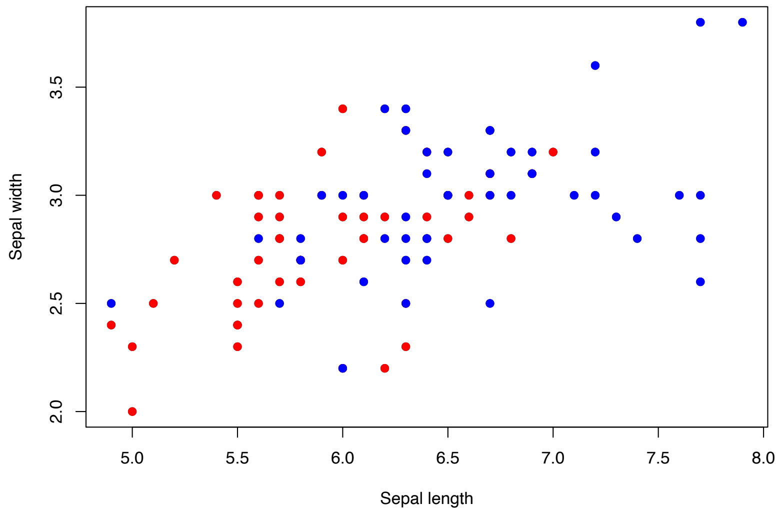


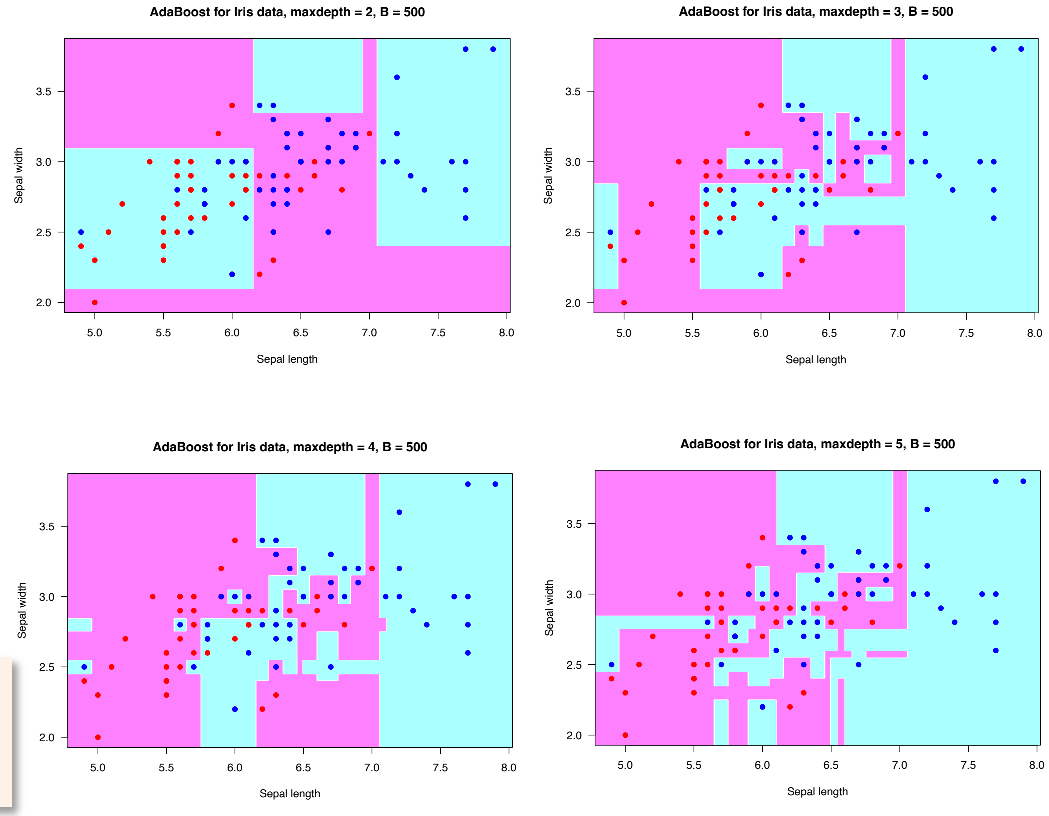


Effet du nombre **d'itérations**



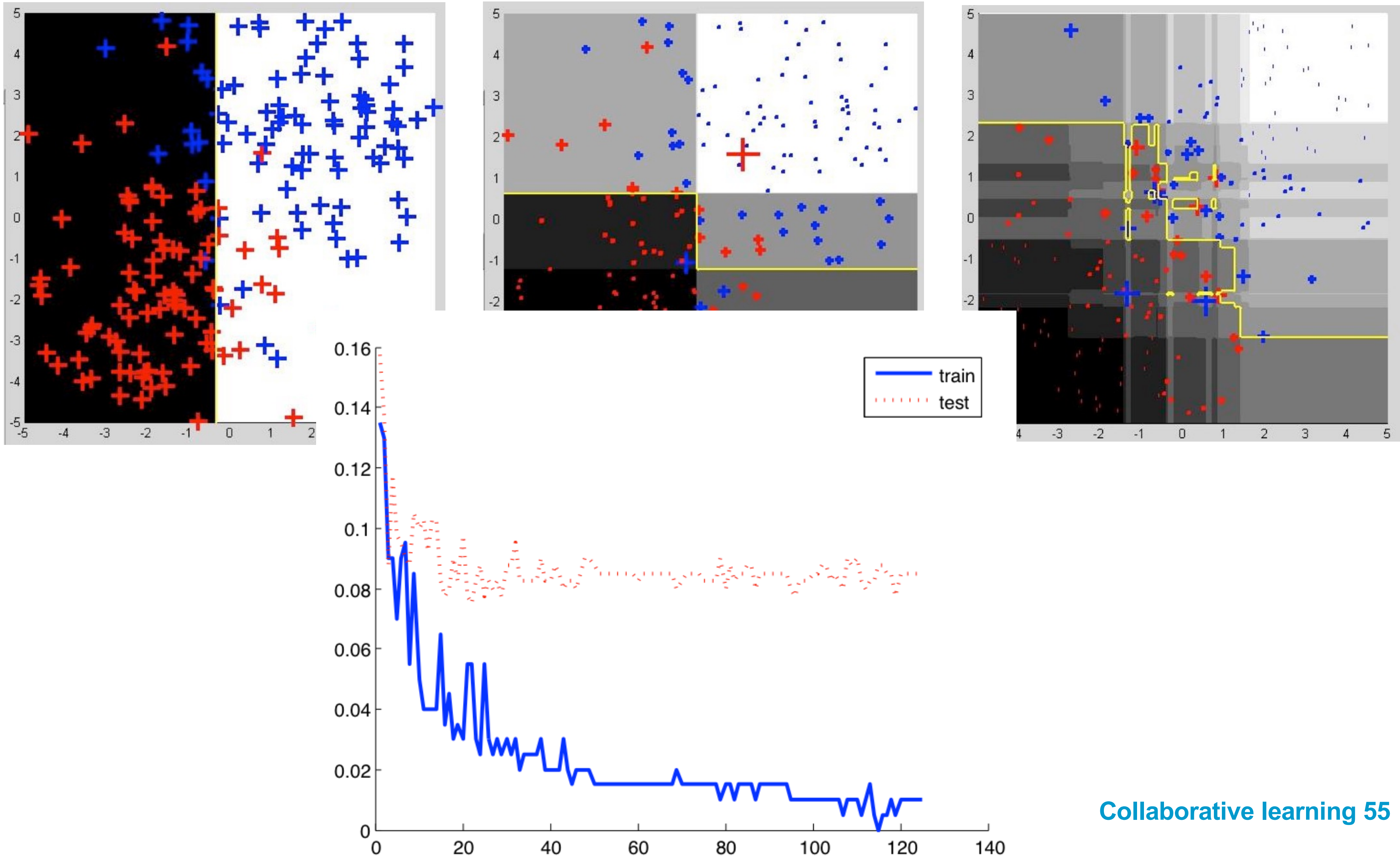
Boosting sur jeu de données « Iris » (Versicolor vs. Virginica)





Effet de la **profondeur** des arbres

Boostingdemo of Richard Stapenhurst

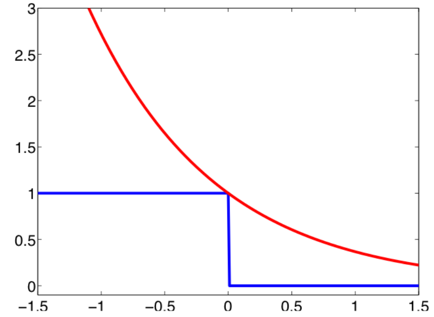


Theoretical analysis of boosting

Derivation of the boosting algorithm

Derivation of the boosting algorithm

- Another derivation of boosting
  - By choosing a *surrogate loss function* with an exponential form



$$\ell(h(\mathbf{x}), y) \;=\; e^{-y \cdot h(\mathbf{x})}$$

Soit : $\quad H_{T-1} \;=\; \alpha_1 \, h_1(\mathbf{x}) + \alpha_2 \, h_2(\mathbf{x}) + \ldots + \alpha_{T-1} \, h_{T-1}(\mathbf{x})$

On veut ajouter : $\quad \alpha_T \, h_T(\mathbf{x})$

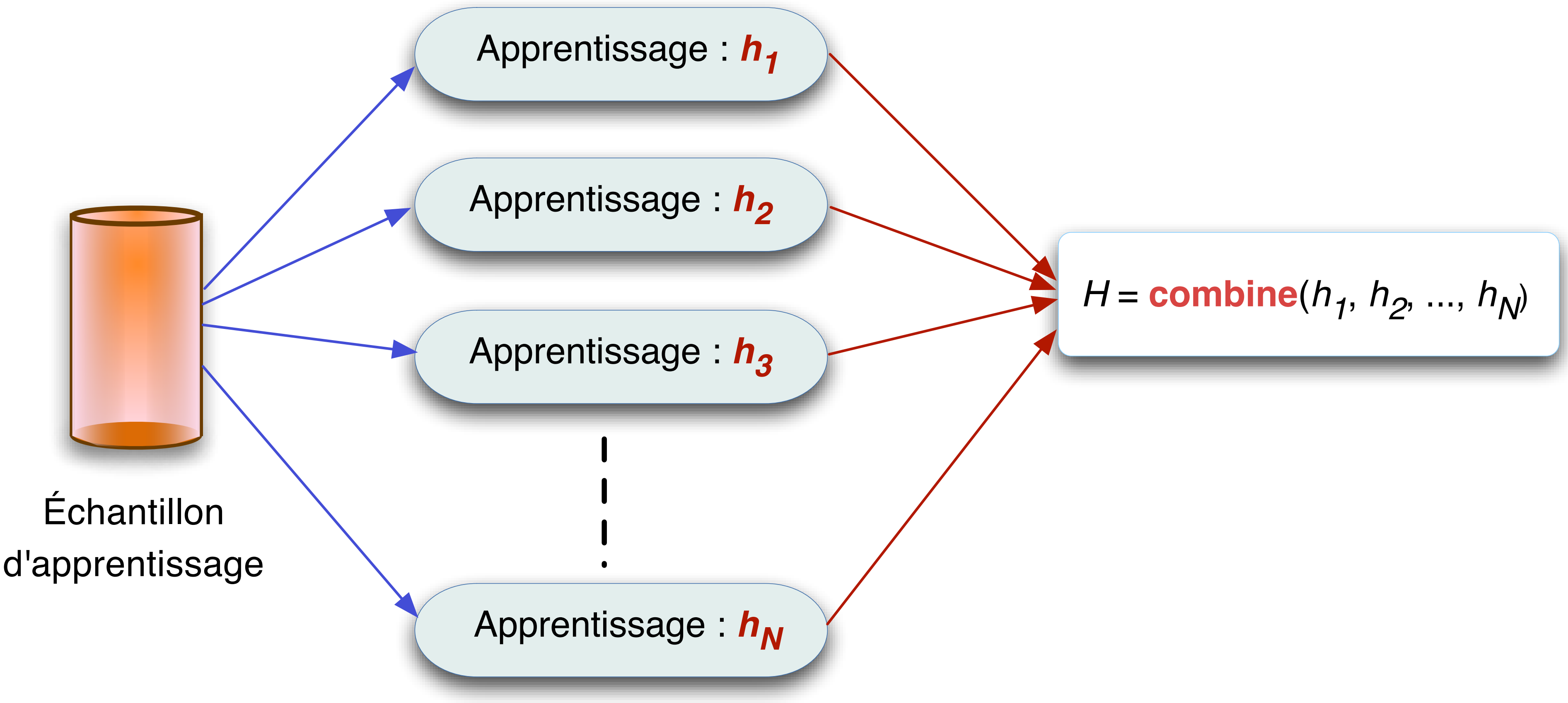
$$\boxed{R_{\text{Emp}}(H_T) \;=\; \sum_{i=1}^{m} e^{-y_i \left[H_{T-1}(\mathbf{x}_i) + \alpha_T \, h_T(\mathbf{x}_i)\right]}}$$

$$= \sum_{i=1}^{m} e^{-y_i \, H_{T-1}(\mathbf{x}_i)} \cdot e^{-\alpha_T \, y_i \, h_T(\mathbf{x}_i)}$$

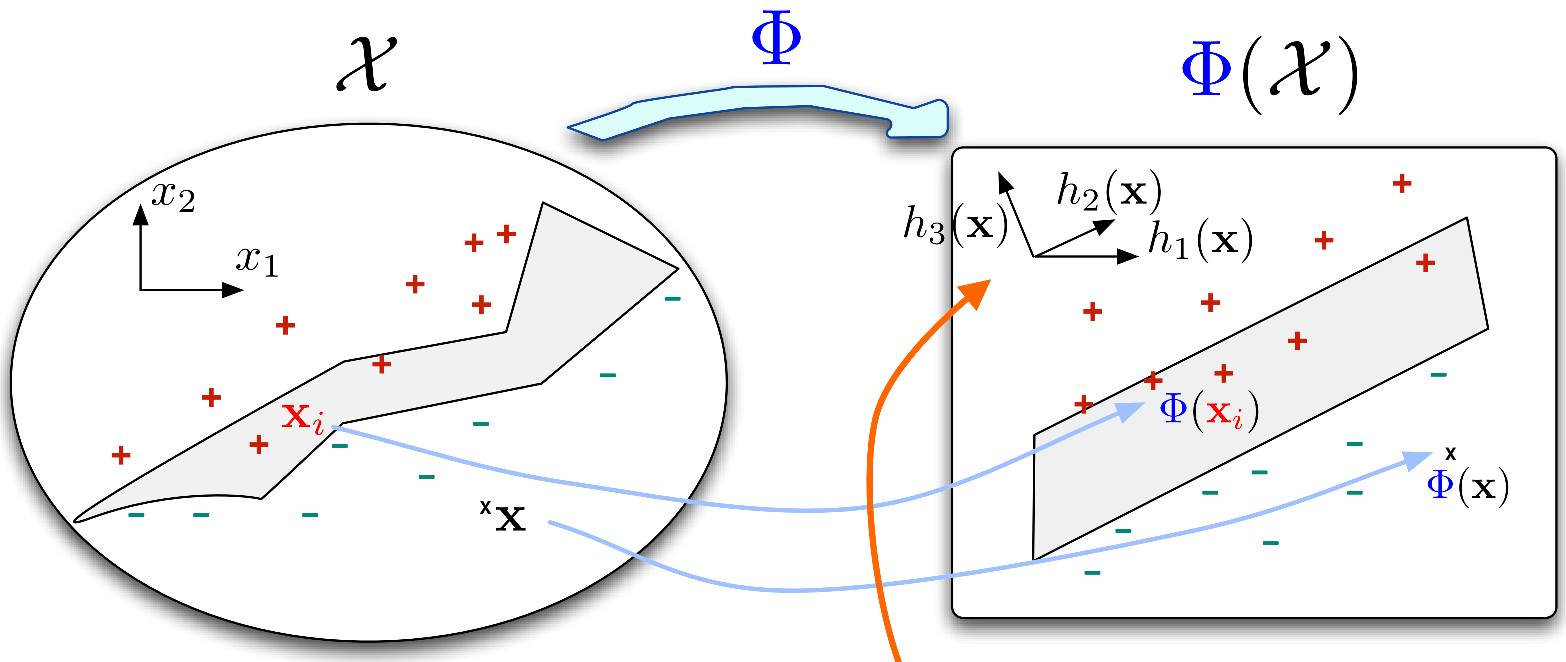
$$= \sum_{i=1}^{m} W_{T-1}(\mathbf{x}_i) \cdot e^{-\alpha_T \, y_i \, h_T(\mathbf{x}_i)}$$

$$\boxed{\frac{\partial R_{\text{Emp}}(H_T)}{\partial \alpha} \;\propto\; e^{-\alpha} \underbrace{(1 - \varepsilon_T)}_{\substack{\text{poids des exemples} \\ \text{correctement prédits}}} + e^{\alpha} \underbrace{\varepsilon_T}_{\substack{\text{poids des exemples} \\ \text{incorrectement prédits}}}} \;\rightarrow\; \boxed{\alpha_T \;=\; \frac{1}{2} \log \frac{1 - \varepsilon_T}{\varepsilon_T}}$$

General scenario: learning



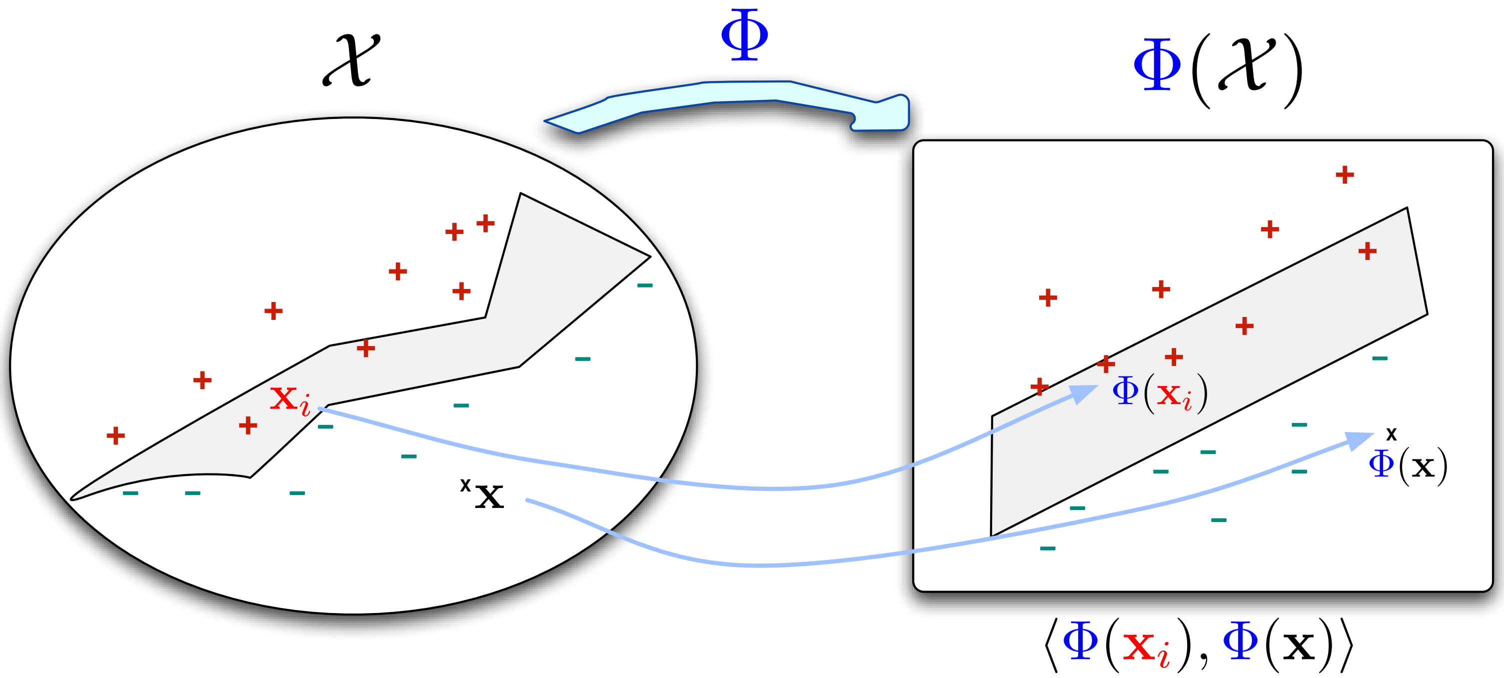
*Boosting and **redescription***



$$H(\mathbf{x}) \;=\; \mathrm{sign}\left\{\sum_{t=1}^{T} \alpha_t \; h_t(\mathbf{x})\right\}$$

- **Iterative** construction of the redescription space

SVM and kernel methods



$$h^*(\mathbf{x}) = \operatorname{sign}\left\{ \sum_{i \in \mathcal{P}_S} \alpha_i^\star \, y_i \, \kappa(\mathbf{x}_i \, , \, \mathbf{x}) + w_0^\star \right\}$$

Bounds on training error

and

On generalization error

*Bound on the **training** error*

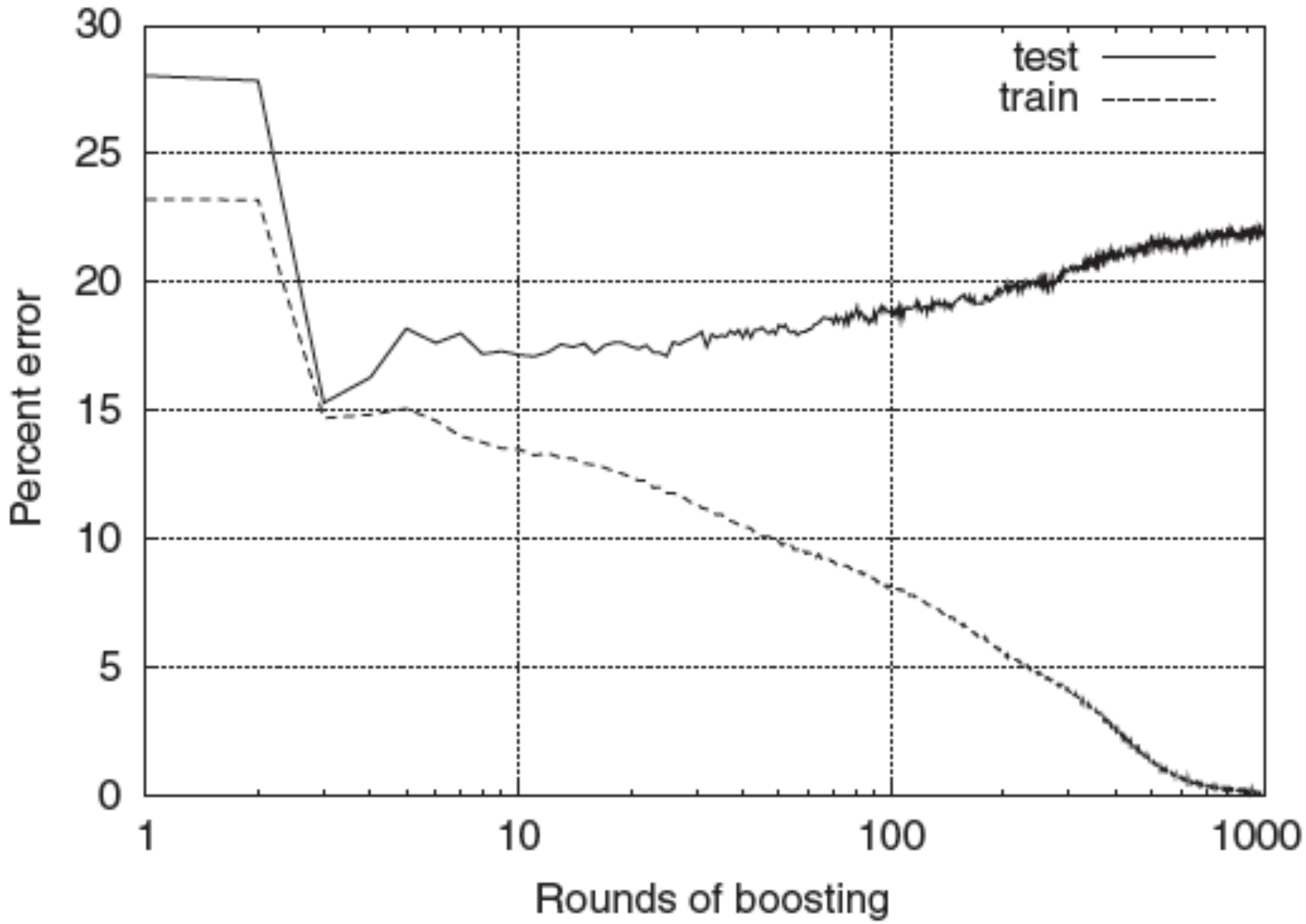
- Theorem:
  - write ϵ_t as $1/2 - \gamma_t$ [γ_t = "edge"]
  - then

$$
\begin{aligned}
\text{training error}(H_{\text{final}}) &\leq \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] \\
&= \prod_t \sqrt{1-4\gamma_t^2} \\
&\leq \exp\left(-2\sum_t \gamma_t^2\right)
\end{aligned}
$$

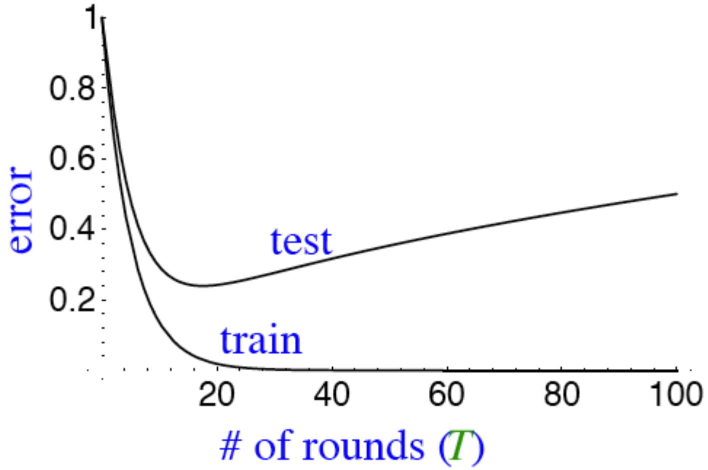
- so: if $\forall t: \ \gamma_t \geq \gamma > 0$

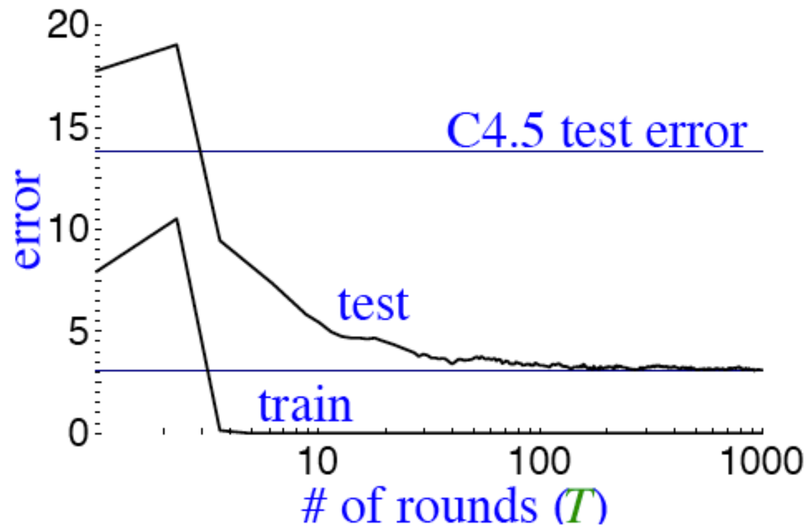
  then $\text{training error}(H_{\text{final}}) \leq e^{-2\gamma^2 T}$
- AdaBoost is adaptive:
  - does not need to know γ or T a priori
  - can exploit $\gamma_t \gg \gamma$

Evolution of the error curves (learning & test)



How to explain the evolution of the generalization error?





- The test error **does not increase**, even after 1000 steps (2.10^6 test nodes !!)
  - Boosting C4.5 on the « letter » dataset

Arguments to **explain**

the **properties** of boosting

(the unreasonable power of boosting)

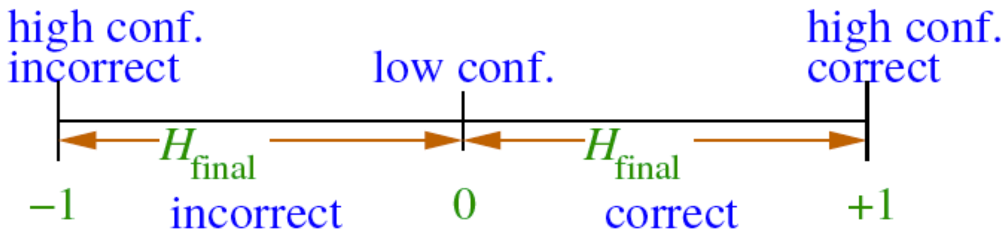
The "margin" explanation

- Idea:
  - The test error is a rough indicator of the prediction performance
  - One should also take into account the **confidence** of the prediction
  - It is possible to estimate this confidence by the **margin**

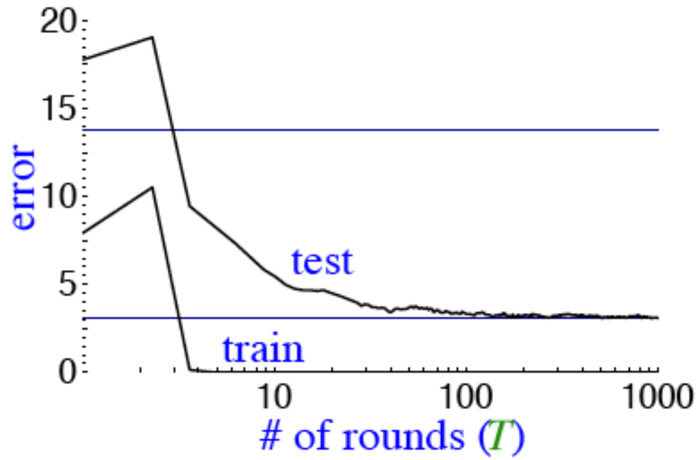
    = weights of the classifiers with **correct** predictions

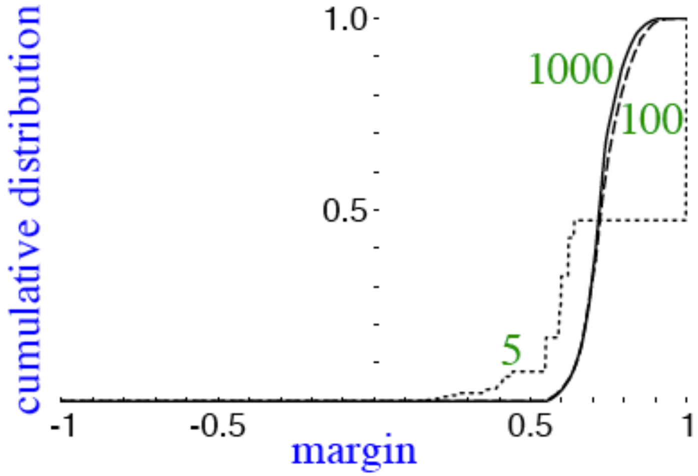
    (on the training examples)

    \- weights of the classifiers with **incorrect** predictions



Margin distribution on the $\mathbf{x}_i$





	# 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

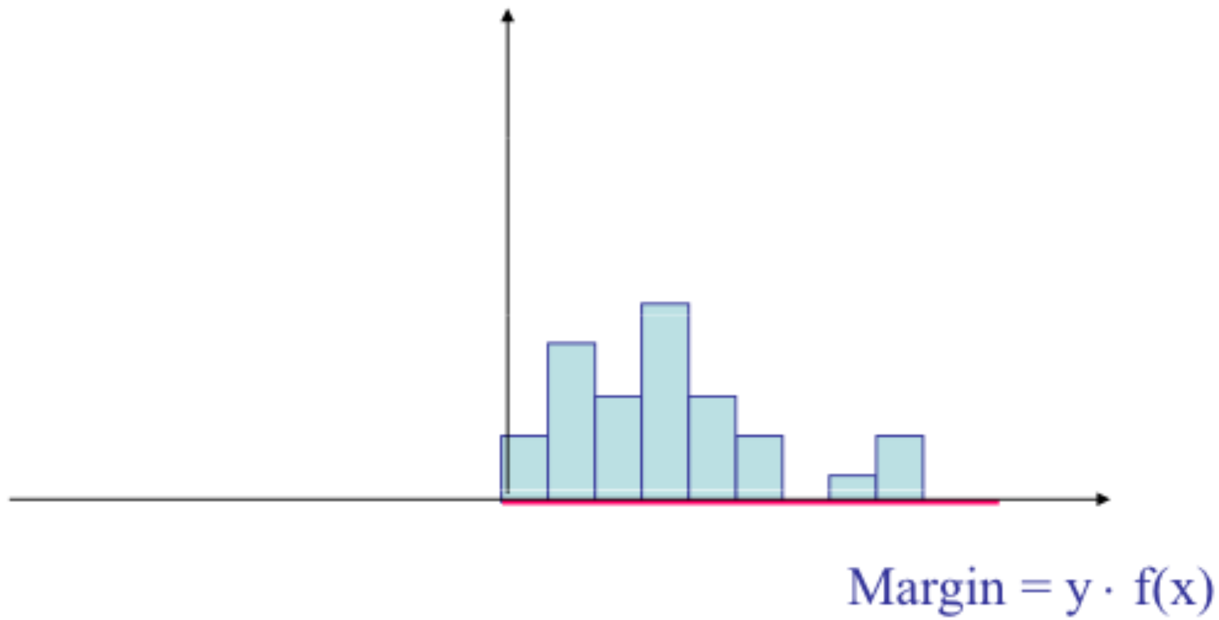
The argument of the margin maximization

- At each step, AdaBoost would put **more weight on the examples x_i with small margin** while continuing to improve the margin on the other examples

- The final hypothesis would be a **complex one** but **with a large margin**

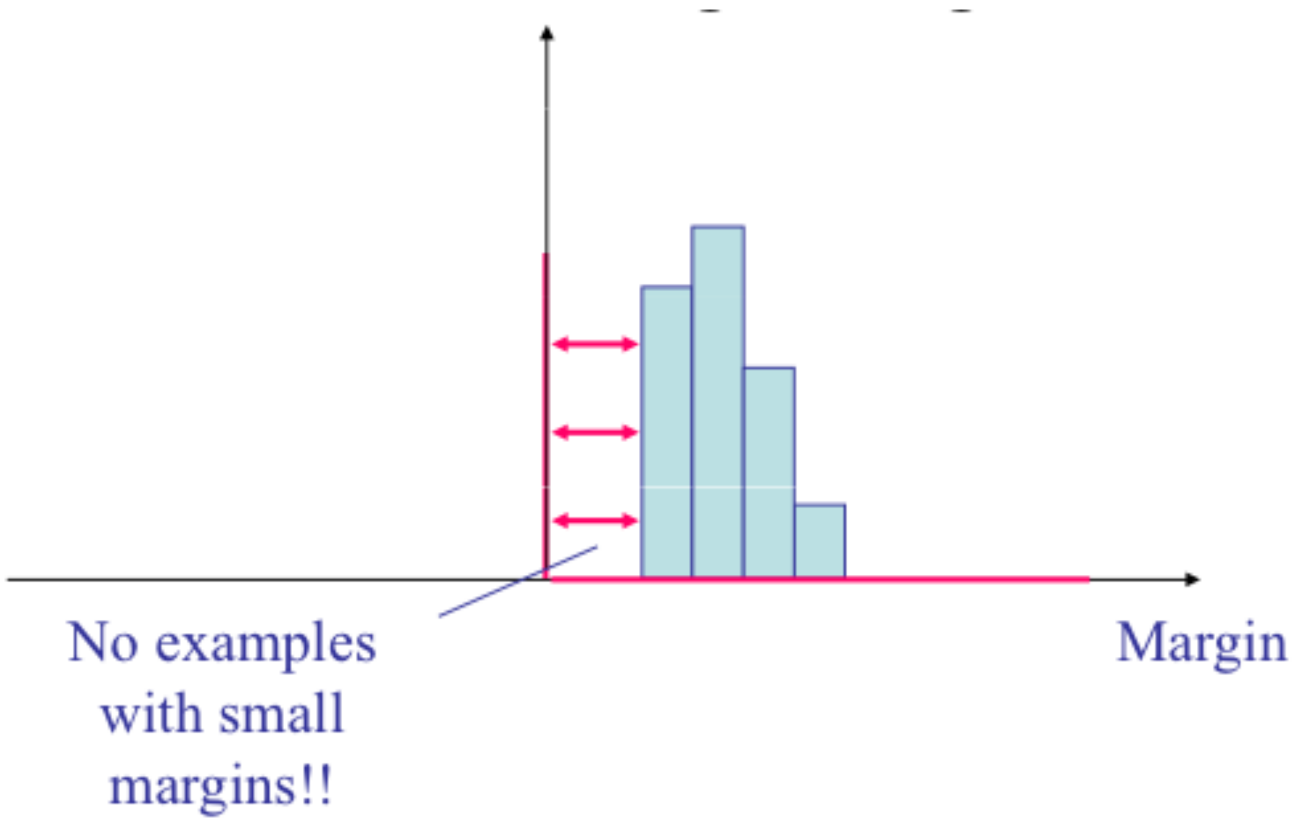
  (and so with **generalization error** close to the **training one**)

The argument of the margin maximization



Histogram of functional margin for ensemble just after achieving zero training error

The argument of the margin maximization



Even after zero training error the margin of examples increases.
This is one reason that the generalization error may continue decreasing.

Generalization bounds

$$R_{Réel} \leq R_{Emp} + \mathcal{O}\left(\sqrt{\frac{T \cdot d_{\mathcal{H}}}{m}}\right)$$

Formally

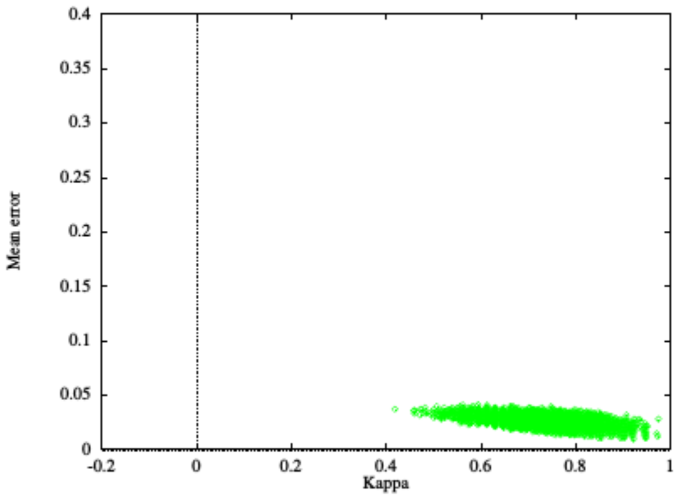
- with high probability, $\forall\theta > 0$:

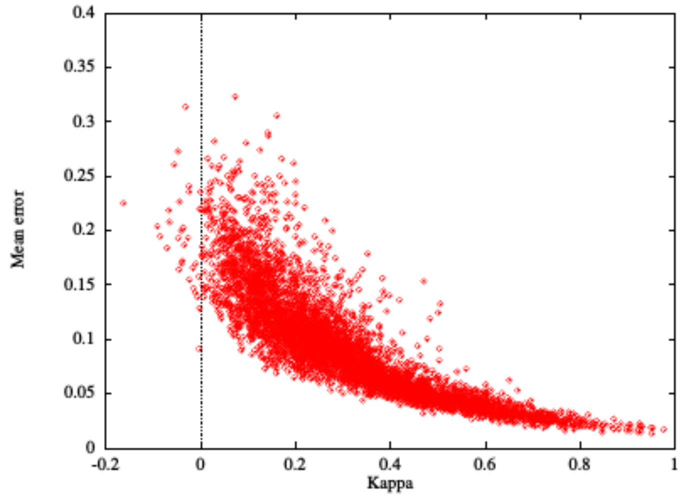
$$\text{generalization error} \leq \hat{\Pr}[\text{margin} \leq \theta] + \tilde{O}\left(\frac{\sqrt{d/m}}{\theta}\right)$$

($\hat{\Pr}[\,] =$ empirical probability)

  - bound depends on
    - $m = \#$ training examples
    - $d =$ "complexity" of weak classifiers
    - entire distribution of margins of training examples
- $\hat{\Pr}[\text{margin} \leq \theta] \rightarrow 0$ exponentially fast (in T) if (error of h_t on D_t) $< 1/2 - \theta$ $(\forall t)$
  - so: if weak learning assumption holds, then all examples will quickly have "large" margins

AdaBoost produit des hypothèses bien plus *diverses* que les autres méthodes d'ensemble [Dietterich, 2000] :





Poursuivre sur cette piste :

- méthode favorisant la diversité [Melville and Mooney, 2004],
- mesures de la diversité [Kuncheva and Whitaker, 2003],

...

Assessment

Advantages of AdaBoost

- **Low** computational **cost**
- **Easy** to use
- **A single parameter**: the number of steps: T
- Can be (and has been) **applied** in very numerous domains
- **No overfitting** (in general) because of the margin maximization
- Can be adapted to **regression** problems $h_t : \mathcal{X} \rightarrow \mathcal{R}$;
  the class is defined by the sign of $h_t(x)$ and the confidence by $| h_t(x) |$
- Can be adapted to the **multi-class** case where $y_i \in \{1,..,c\}$
- Allows one to **uncover outliers**

When is boosting not appropriate?

- Reminder: **No-free-lunch-theorem**

- Boosting is NOT recommended when
  - There is **not enough data**
  - The **set** of weak learners is **too limited**
  - The weak learners are too stable (but more true for bagging)
  - The **weak learners** are **too strong!**
    - They can overfit
  - **Noise** in the data (but ...)

Applications using boosting

Robust real-time face detection [Viola & Jones, 2004]

- **Images** 384 x 288 (grey level)
- Detect **visages** at every scale
- In **real time** (15 images / s) on a smartphone!!

- Problems
  - Identify the **relevant descriptors**
  - Compute them **fastly**
  - Use (combine) them in an very efficient way
    - Low FN rate
    - Low FP rate

Les descripteurs

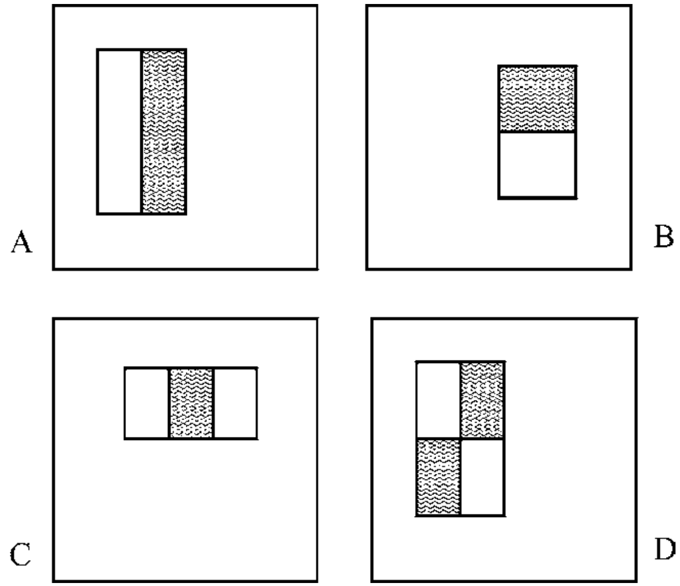
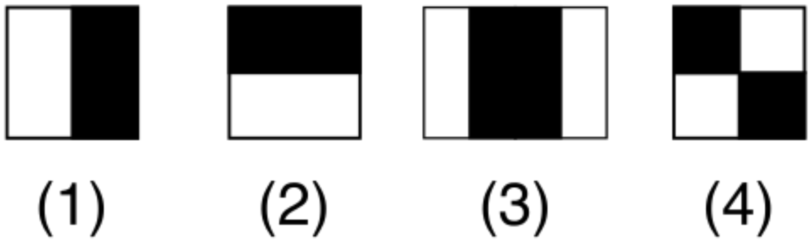
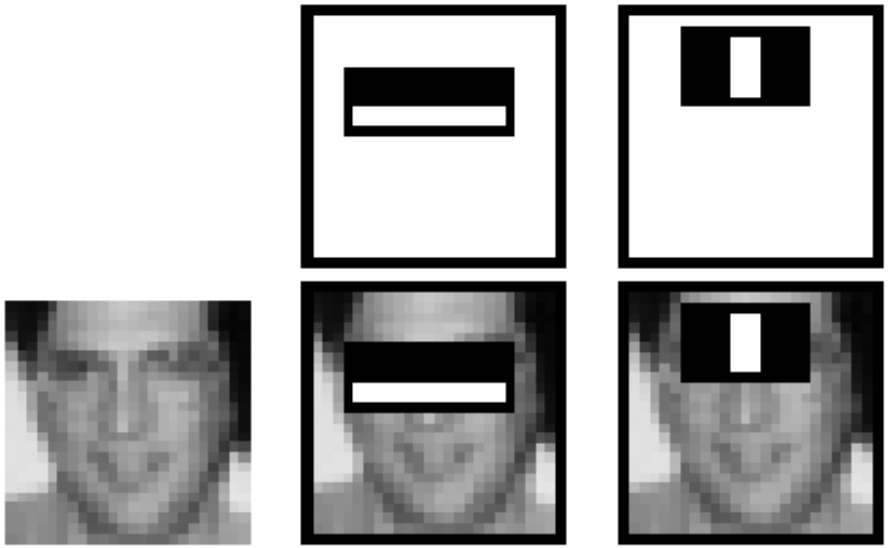


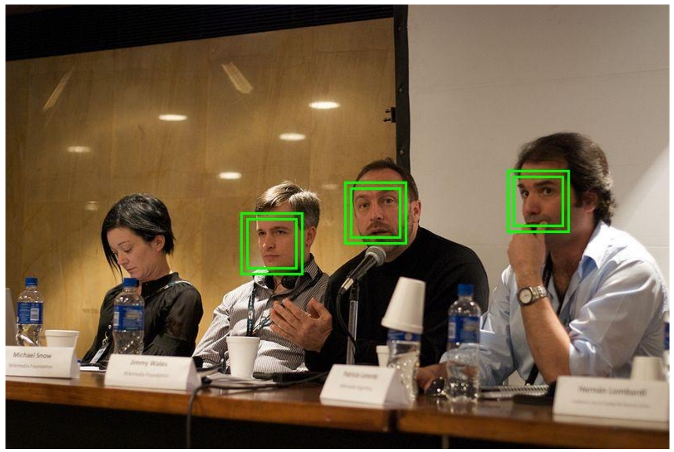
Figure 1. Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

- More than 3 000 000 000!
  - All scales
  - Thresholds to be defined

Useful Features Learned by Boosting





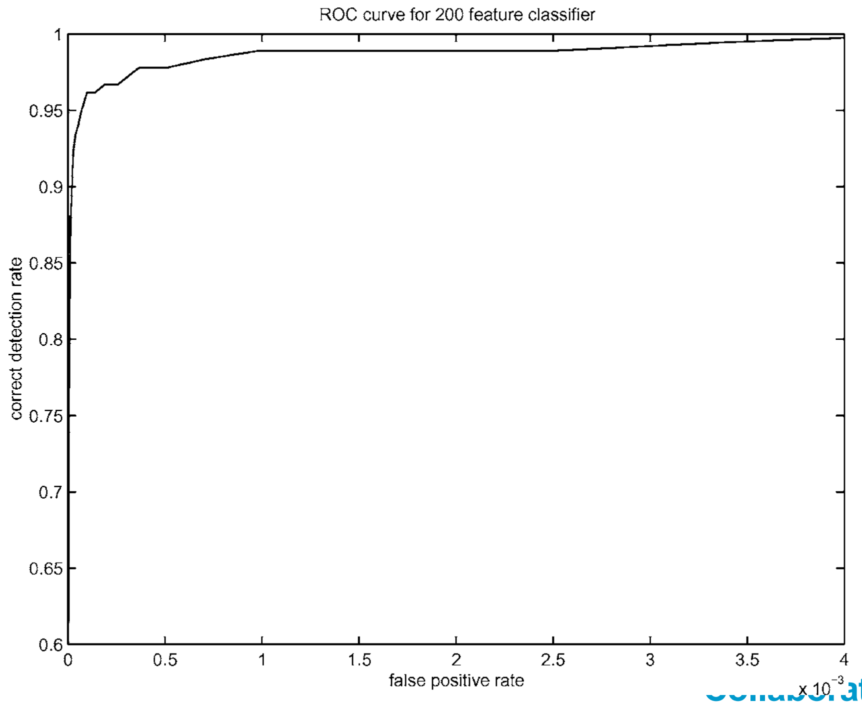


Example of the training set

Positive instances

Selection of the useful descriptors

- Using AdaBoost
  - Descriptors are found as decision stumps
  - Le boosting select them
    - 200 in this study



Still computationally too costly: ~0.7*s*

Detectors are organized in cascade

- **Eliminate** the negative as soon as possible

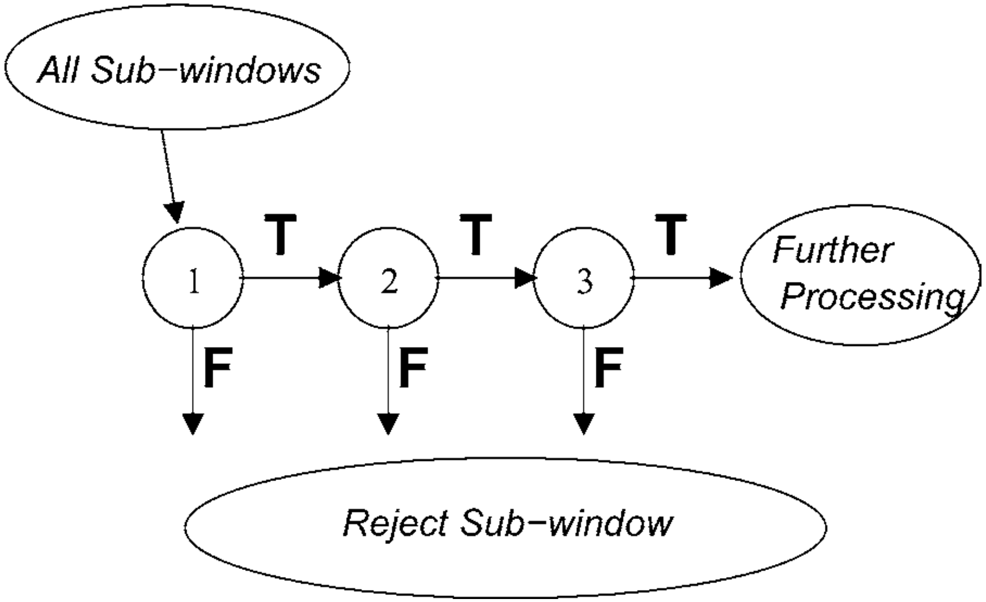
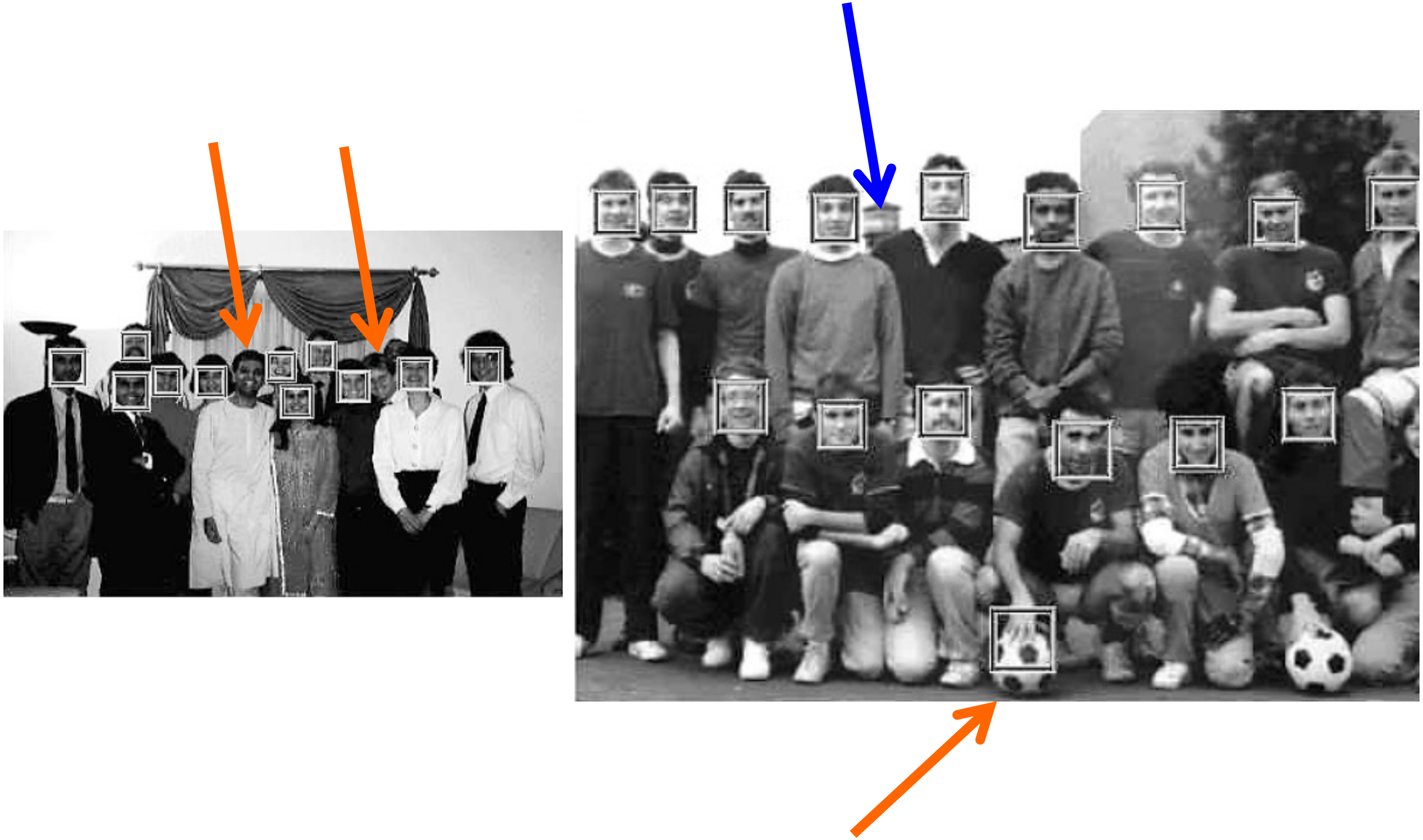


Figure 6. Schematic depiction of a the detection cascade. A series of classifiers are applied to every sub-window. The initial classifier eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation. After several stages of processing the number of sub-windows have been reduced radically. Further processing can take any form such as additional stages of the cascade (as in our detection system) or an alternative detection system.

Face detection using boosting



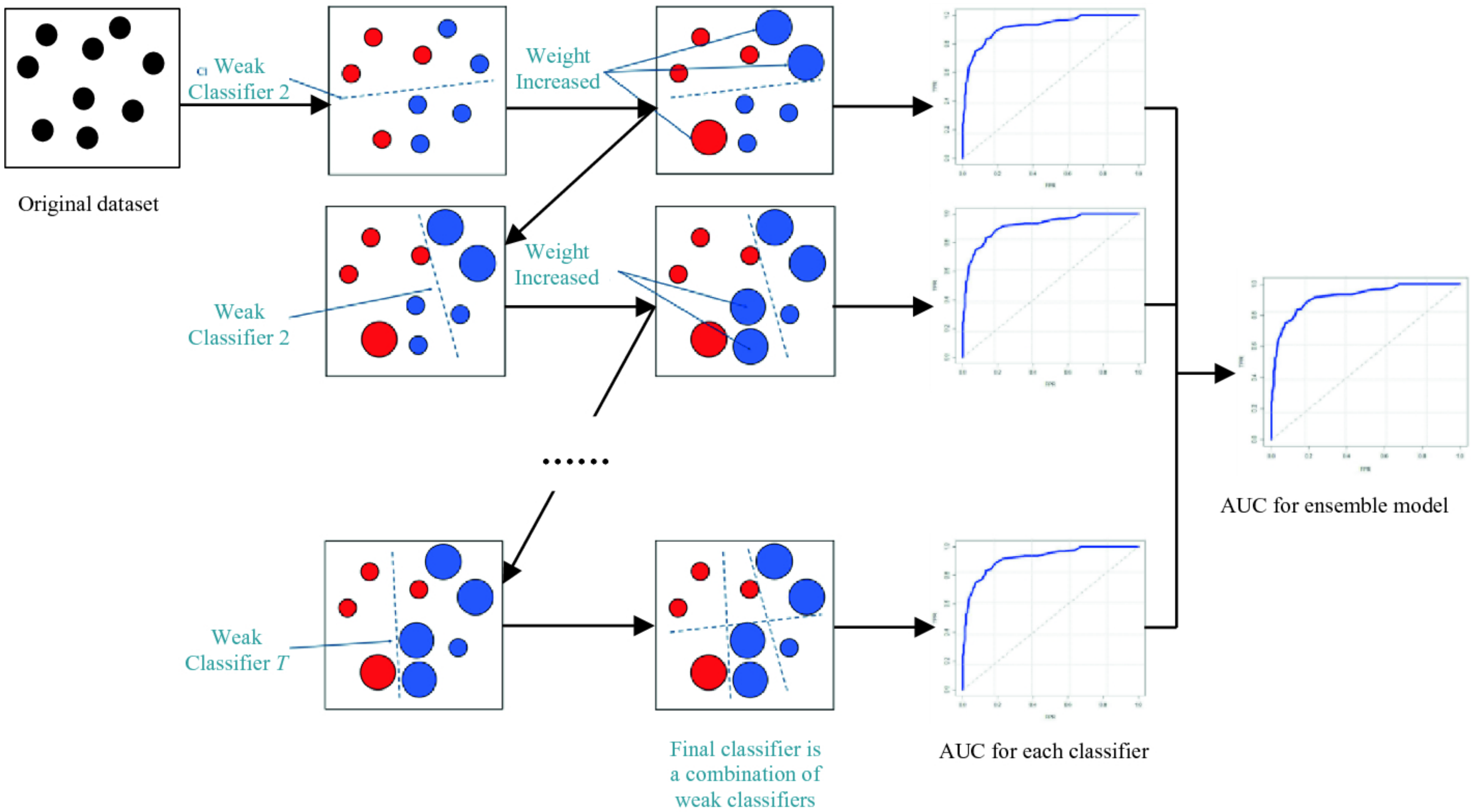
Gradient Tree Boosting

Boosted Trees

- For classification or regression
- Using decision **trees**
- The successive trees are found (and weighted) using **boosting**

- In general: **more powerful than Random Forests**
  - E.g. the AI4Industry challenge (2021)
    - *Regression for **predictive maintenance***
    - Won the challenge before deep NNs and Random Forests

Boosted Trees



https://datascience.eu/fr/apprentissage-automatique/gradient-boosting-ce-que-vous-devez-savoir/

Other algorithms

(*Xgboost or Tree Boosting*)

Xgboost aka. Extreme Gradient boosting

- Characteristics
  - Gradient Tree Boosting
  - Optimized to be very efficient
  - Lots of parameters (good and bad)

Conclusions on Ensemble Methods

- Modifying the input distribution during learning yields a richer **diversity** of (weak) learners

- *Boosting* makes the learners **dependent** upon each other

  Better than *bagging*

  - **All** are used in the final prediction
  - In co-learning, we will see another method of changing the input distribution, using **only the final classifiers** to make the prediction

Some references

- Bob Shapire and Yoav Freund
  Boosting: Foundations and Algorithms
  MIT Press, 2012

- Ron Meir and Gunnar Rätsch
  An introduction to Boosting and Leveraging
  In *Advanced Lectures on Machine Learning* (LNAI-2600), 2003
  http://www.boosting.org/papers/MeiRae03.pdf

- Zhi-Hua Zhou
  Ensemble Methods. Foundations and Algorithms
  CRC Press, 2012

- Vincent Barra, Antoine Cornuéjols et Laurent Miclet
  Apprentissage artificiel. Concepts et algorithmes. De Bayes et Hume au deep learning
  Eyrolles, 2021

Outline

1. Ensemble methods: boosting

2. The LUPI framework

3. Illustration on Early Classification of Time Series

4. Conclusions

Learning Using Privileged Information

Inspired by learning at school

- The goal is to learn a function $h : \mathbf{x} \in \mathcal{X} \rightarrow y \in \{-1, +1\}$

- Suppose that at learning time there is more available information than at test time

$$\mathcal{S}^* \; = \; \{(\underbrace{\mathbf{x}_i, \mathbf{x}_i^*}_{\mathcal{X}'}, y_i)\}_{1 \leq i \leq m}$$

- **Can we then improve the generalization performance** wrt. the one obtained with *S* only?

V. Vapnik and A. Vashist (2009) **“A new learning paradigm: Learning using privileged information”.**
Neural Networks, vol. 22, no. 5, pp. 544–557, 2009

Learning Using Privileged Information

Illustration in computer vision

V. Sharmanska, N. Quadrianto, and Ch. Lamper (2014) "**Learning to transfer privileged information**".
ArXiv preprint arXiv:1410.0389, 2014

*Bounds between the **real** risk and the **empirical** risk*

- $\mathcal{H}$ finite, realisable case

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \;\leq\; R_{\text{Emp}}(h) \;+\; \frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m} \right] \;>\; 1 - \delta$$

- $\mathcal{H}$ finite, **non** realisable case

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \;\leq\; R_{\text{Emp}}(h) \;+\; \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2\, m}} \right] \;>\; 1 - \delta$$

instead of 600.10^6 training examples, same performance wit

First *approach to LUPI*

- "At the core of our work lies the insight that privileged information allows us to **distinguish between easy and hard examples** in the training set.
- **Assuming** that examples that are easy or hard with respect to the privileged information **will also be easy or hard with respect to the original data**, we enable information transfer from the privileged to the original data modality.
- More specifically, we first define and identify which samples are easy and which are hard for the classification task, and **incorporate the privileged information** into the sample **weights** that encodes its easiness or hardness." (more weight on the easy examples)

One solution: SVM+

- The classical optimization problem

$$\begin{cases} \min \frac{1}{2}\langle \omega, \omega \rangle + C \sum_{i=1}^{m} \xi_i \\ \text{s.t. } y_i[\langle \omega, x_i \rangle + b] \geq 1 - \xi_i, \qquad i = 1, \dots, m. \end{cases}$$

- is changed into

$$\begin{cases} \min \frac{1}{2}\left[\langle \omega, \omega \rangle + \gamma \langle \omega^*, \omega^* \rangle\right] + C \sum_{i=1}^{m} [\langle \omega^*, x^* \rangle + b^*] \\ \text{s.t. } y_i[\langle \omega, x_i \rangle + b] \geq 1 - [\langle \omega^*, x_i^* \rangle + b^*], \qquad i = 1, \dots, m, \\ \quad [\langle \omega^*, x_i^* \rangle + b^*] \geq 0, \qquad i = 1, \dots, m, \end{cases}$$

C and γ are hyperparameters

- Intuition:
  - Identify the **difficult examples** (outliers)
  - Thus coming back to the **realizable case**

    and obtain **convergence rates** of **1/n** instead of 1/sqrt(n)

Second *approach to LUPI*

Suppose that in $\mathcal{X}'$, there exists a **good hypothesis space** $\mathcal{H}'$ with very limited capacity (otherwise, why would the teacher be interested?), then the student is expected to identify easily a good hypothesis $h' : \mathcal{X}' \rightarrow \mathcal{Y}$. And the whole problem is thus to "project" this hypothesis in $\mathcal{X} \rightarrow \mathcal{Y}$

...

Can you imagine other applications where privileged information could be available at training time (and not at testing time)?

Outline

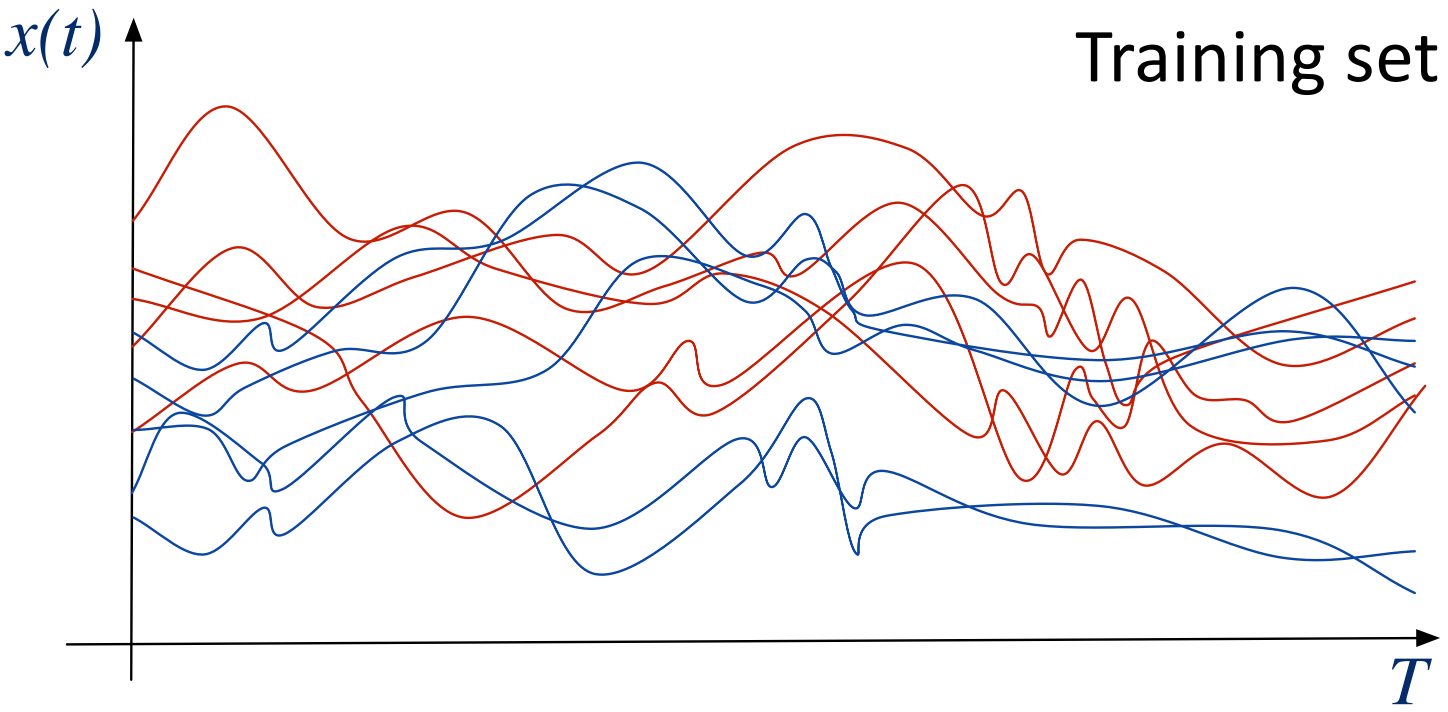
1. Ensemble methods: boosting

2. The LUPI framework

3. Illustration on Early Classification of Time Series

4. Conclusions

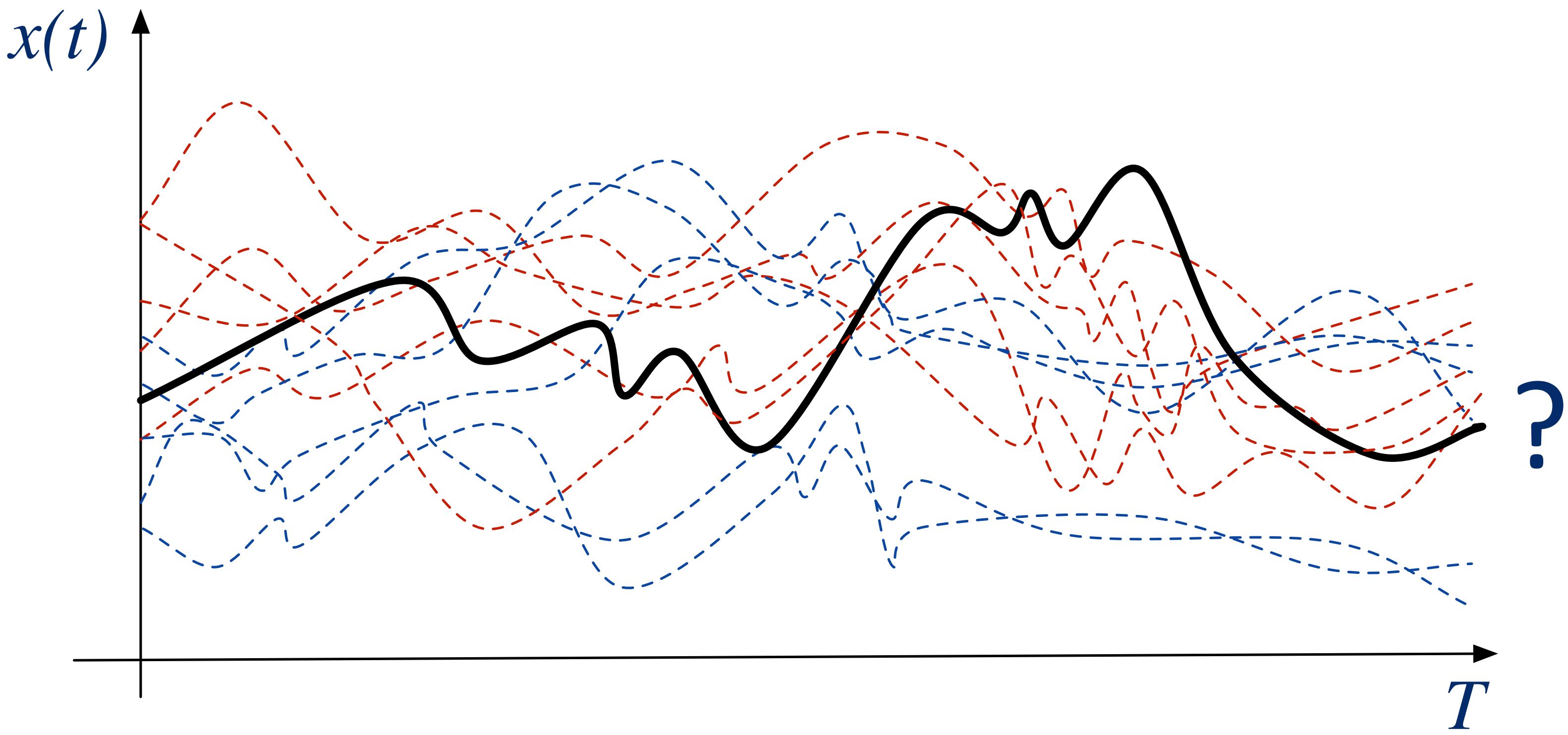
Classification of time series



...

Standard ***classification*** *of time series*

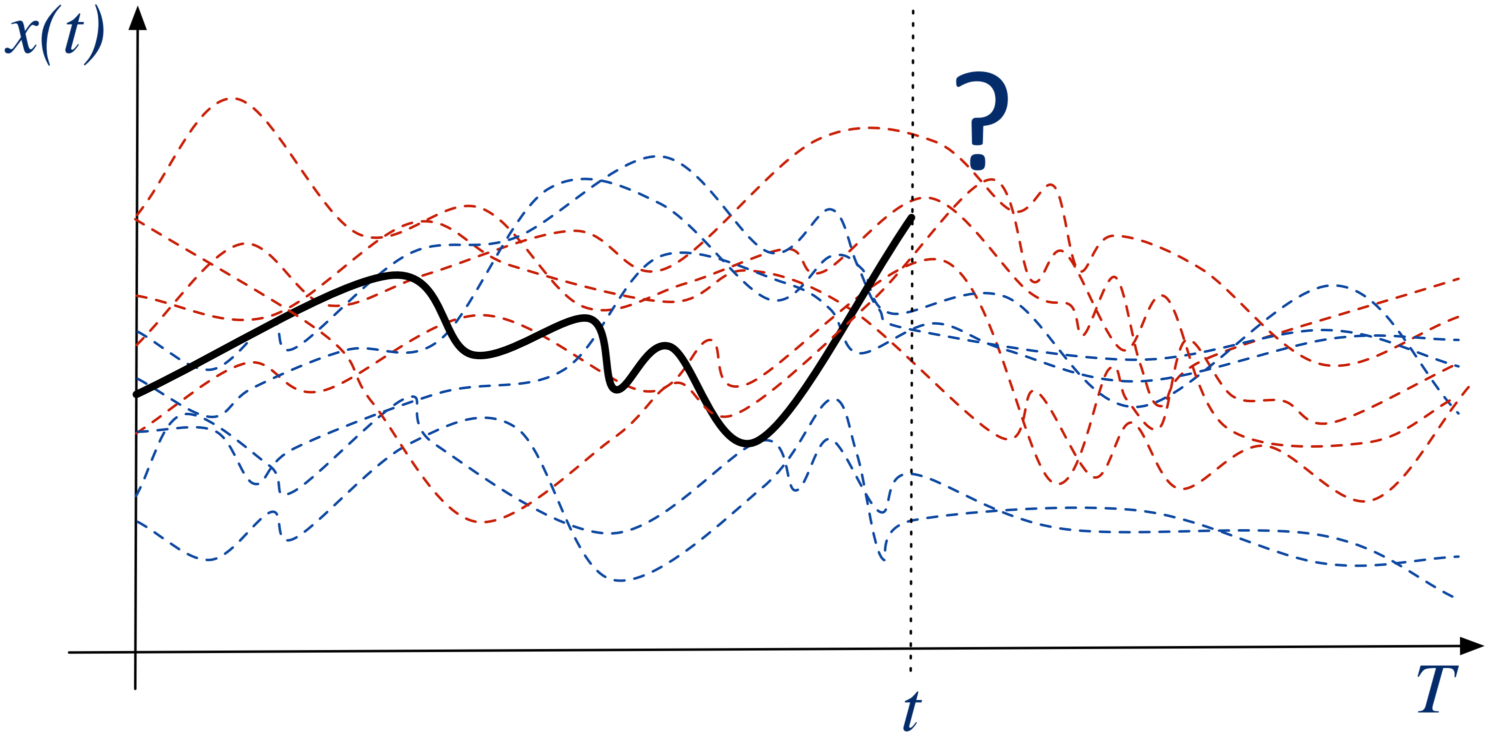
- What is the class of the new time series $\boldsymbol{x_T}$?



- Monitoring of ***consumer actions on a web site***: will buy or not
- Monitoring of a ***patient state***: critical or not
- Prediction of daily ***electrical consumption***: high or low

Early classification of time series

- What is the class of the new **incomplete** time series $\boldsymbol{x_t}$?

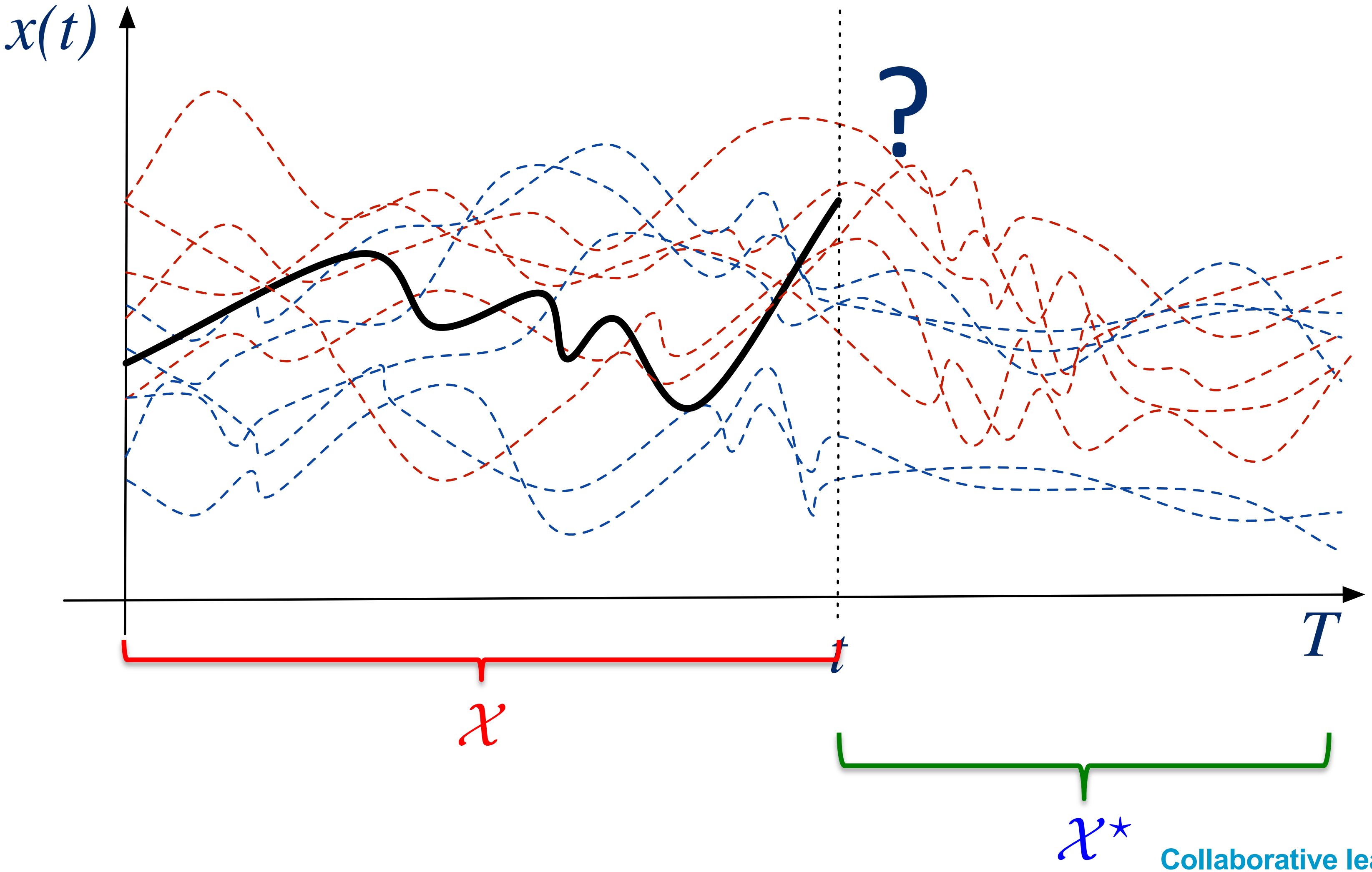


New decision problems: early classification

- Data **stream**
- **Classification** task
- As **early** as possible
- A **trade-off**
  - Classification **performance** (better if $t \nearrow$)
  - Cost of **delaying** prediction (lower if $t \searrow$)

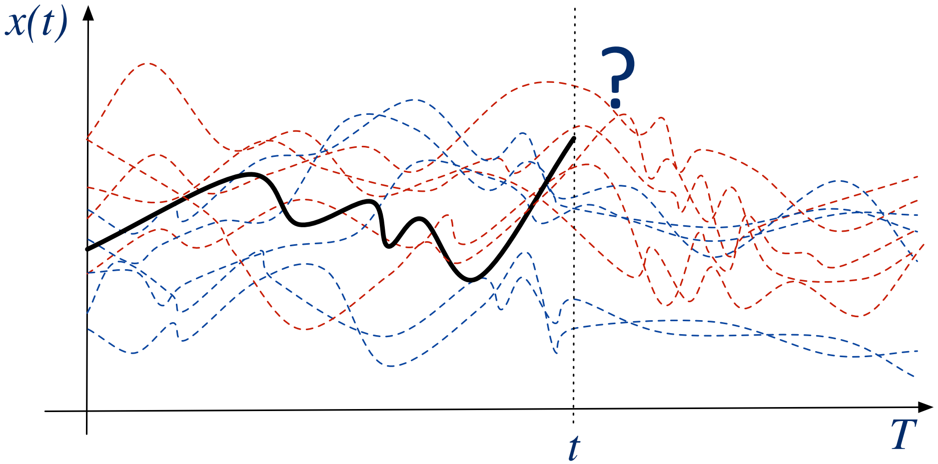
Early ***classification*** *of time series*

- What is the class of the new **incomplete** time series $\boldsymbol{x_t}$?



Early classification of time series

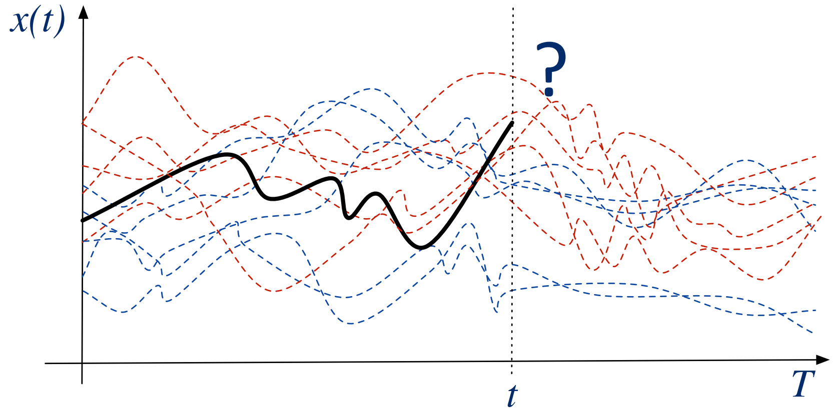
- What is the class of the new **incomplete** time series $\boldsymbol{x_t}$?



- **A LUPI framework**

Early classification and LUPI

- This is a LUPI setting

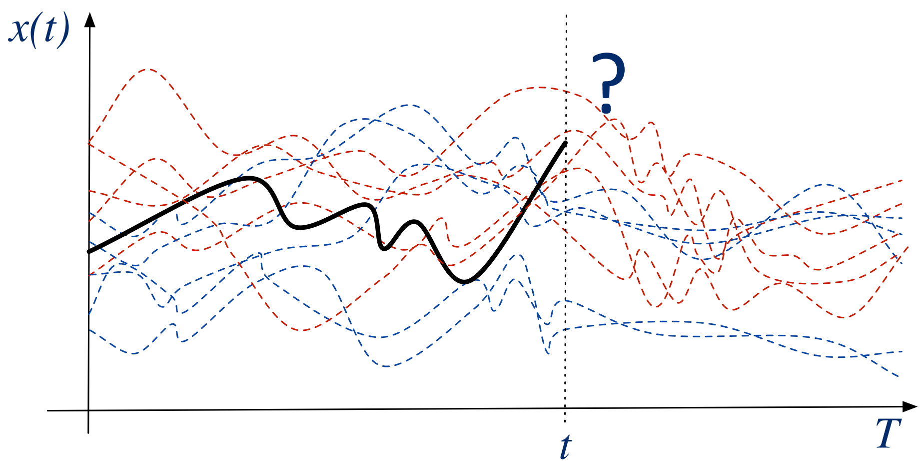


  - **How** to take advantage of this?

Early classification of time series

Online decision problem

- With option to defer at each time step
  - If the **expected future gain** in performance overcomes the **cost of delaying** decision

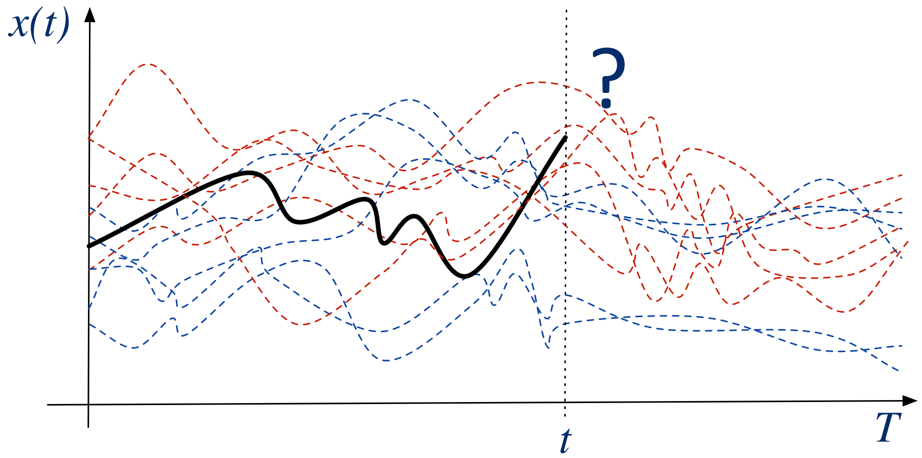


Early classification of time series

Online decision problem

- With option to defer at each time step
  - If the **expected future gain** in performance overcomes the **cost of delaying** decision

Role of LUPI



Decision making (1)

- Given an incoming sequence $\mathbf{x}_t = \langle x_1, x_2, \ldots, x_t \rangle$ where $x_t \in \mathbb{R}$
- And given:
  - A *miss-classification cost function* $C_t(\hat{y}|y) : \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}$
  - A *delaying decision cost function* $C(t) : \mathbb{N} \longrightarrow \mathbb{R}$
- **What is the optimal time to make a decision?**

Expected cost for a decision at time *t*

$$f(\mathbf{x}_t) = \underbrace{\sum_{y \in \mathcal{Y}} P(y|\mathbf{x}_t) \sum_{\hat{y} \in \mathcal{Y}} P(\hat{y}|y, \mathbf{x}_t)\, C_t(\hat{y}|y)}_{\text{expected miss-classification cost given } \mathbf{x}_t} + C(t)$$

Decision making (1)

- Given an incoming sequence $\mathbf{x}_t = \langle x_1, x_2, \ldots, x_t \rangle$ where $x_t \in \mathbb{R}$
- And given:
  - A *miss-classification cost function* $C_t(\hat{y}|y) : \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}$
  - A *delaying decision cost function* $C(t) : \mathbb{N} \longrightarrow \mathbb{R}$
- **What is the optimal time to make a decision?**

Expected cost for a decision at time *t*

$$f(\mathbf{x}_t) = \underbrace{\sum_{y \in \mathcal{Y}} P(y|\mathbf{x}_t) \sum_{\hat{y} \in \mathcal{Y}} P(\hat{y}|y, \mathbf{x}_t)\, C_t(\hat{y}|y)}_{\text{expected miss-classification cost given } \mathbf{x}_t} + C(t)$$

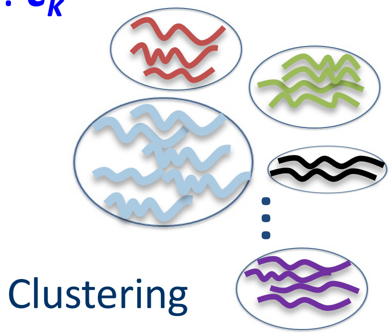
Optimal time: $$t^* = \underset{t \in \{1,\ldots,T\}}{\mathrm{ArgMin}}\ f(\mathbf{x}_t)$$

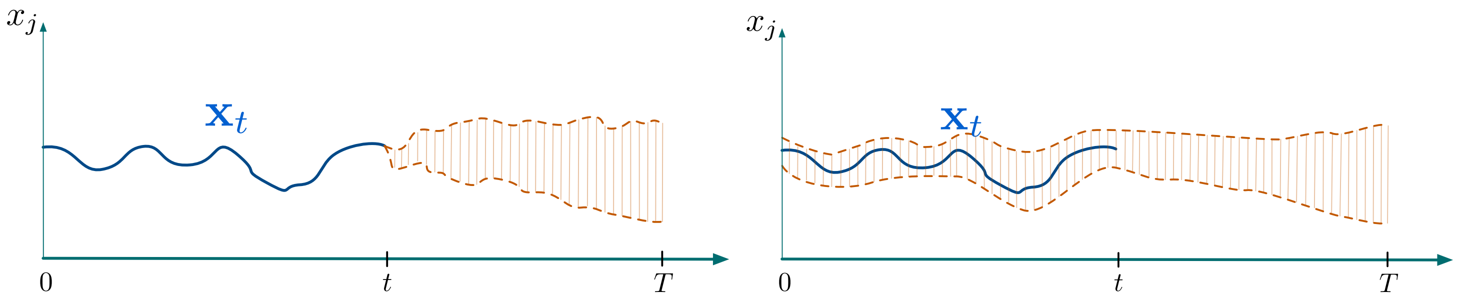
The principle

1. During **training**:

– identify **meaningful subsets** of time sequences in the training set: ***c_k***

$$P(y|\mathbf{x}_t) \;\rightarrow\; P(y|\mathfrak{c}_k)$$



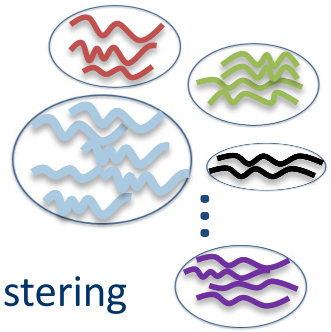


(a) (b)

Fig. 1: (a) Given an incomplete time series $\mathbf{x}_t$, the objective is to try to guess the "envelope" of its foreseeable futures. Various methods can be used to do so. (b) The incoming time series $\mathbf{x}_t$ is viewed as a member of or close to some group(s) of times series, and this is used to guess the "envelope" of its foreseeable futures.

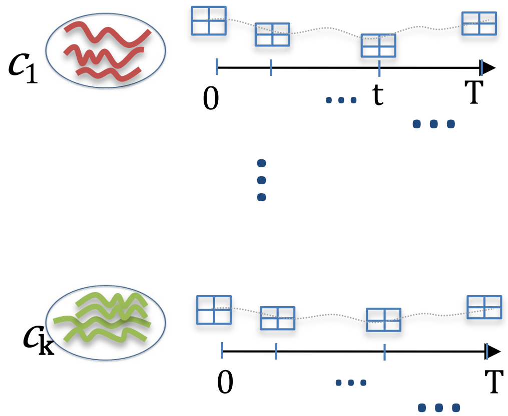
The principle

1. During **training**:
   - identify **meaningful subsets** of time sequences in the training set: ***c_k***
   - **For each of these subsets** c_k, and for each time step t
     - Estimate the **confusion matrices**



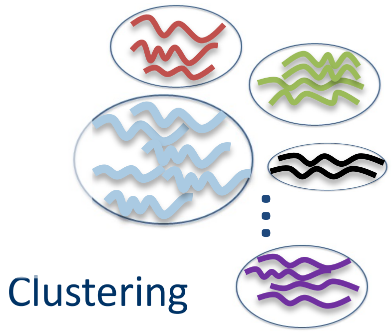
Clustering

- *T* classifiers are **learnt** $h_t(\mathbf{x}_t) : \mathcal{X}_t \rightarrow \mathcal{Y}$
- And their confusion matrices $P_t(\hat{y}|y, \mathfrak{c}_k)$ are **estimated** on a test set



The principle

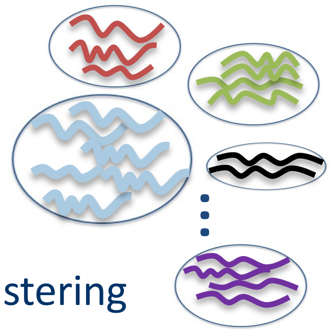
1. During **training**:
   - identify **meaningful subsets** of time sequences in the training set: $\boldsymbol{c_k}$
   - For each of these subsets c_k, and for each time step t
     - Estimate the **confusion matrices** $P_t(\hat{y}|y, \mathfrak{c}_k)$



2. **Testing**: For any new incomplete incoming sequence x_t
   - Identify the **most likely subset**: the closer class of shapes to x_t

The principle

1. During **training**:
   - identify **meaningful subsets** of time sequences in the training set: c_k
   - For each of these subsets c_k, and for each time step t
     - Estimate the **confusion matrices** $P_t(\hat{y}|y, \mathfrak{c}_k)$



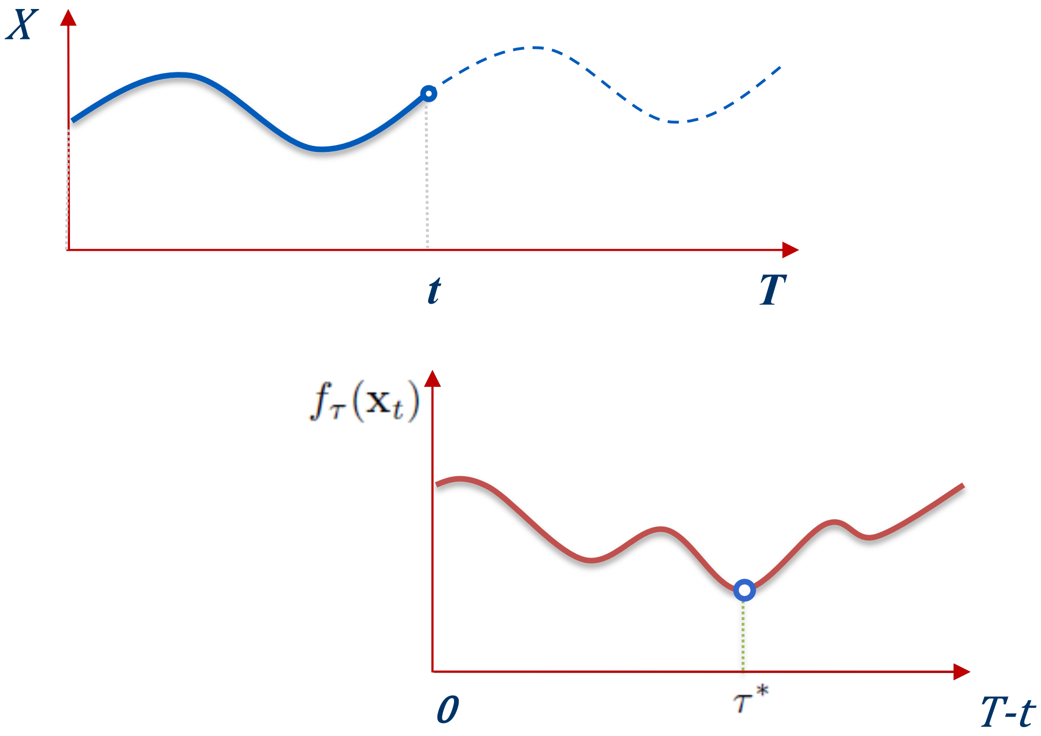
2. **Testing**: For any new incomplete incoming sequence x_t
   - Identify the most likely subset: the closer shape to x_t
   - Compute the **expected cost of decision** for all future time steps

$$f_\tau(\mathbf{x}_t) = \underbrace{\sum_{\mathfrak{c}_k \in \mathcal{C}} P(\mathfrak{c}_k|\mathbf{x}_t) \sum_{y \in \mathcal{Y}} P(y|\mathfrak{c}_k) \sum_{\hat{y} \in \mathcal{Y}} P_{t+\tau}(\hat{y}|y, \mathfrak{c}_k)\, C(\hat{y}|y)}_{\text{expected miss-classification cost given } \mathbf{x}_t} + C(t+\tau)$$

A non myopic decision process

- Optimal estimated time relative to current time *t*

$$\tau^* = \underset{\tau \in \{0,\ldots,T-t\}}{\mathrm{ArgMin}} \; f_\tau(\mathbf{x}_t)$$

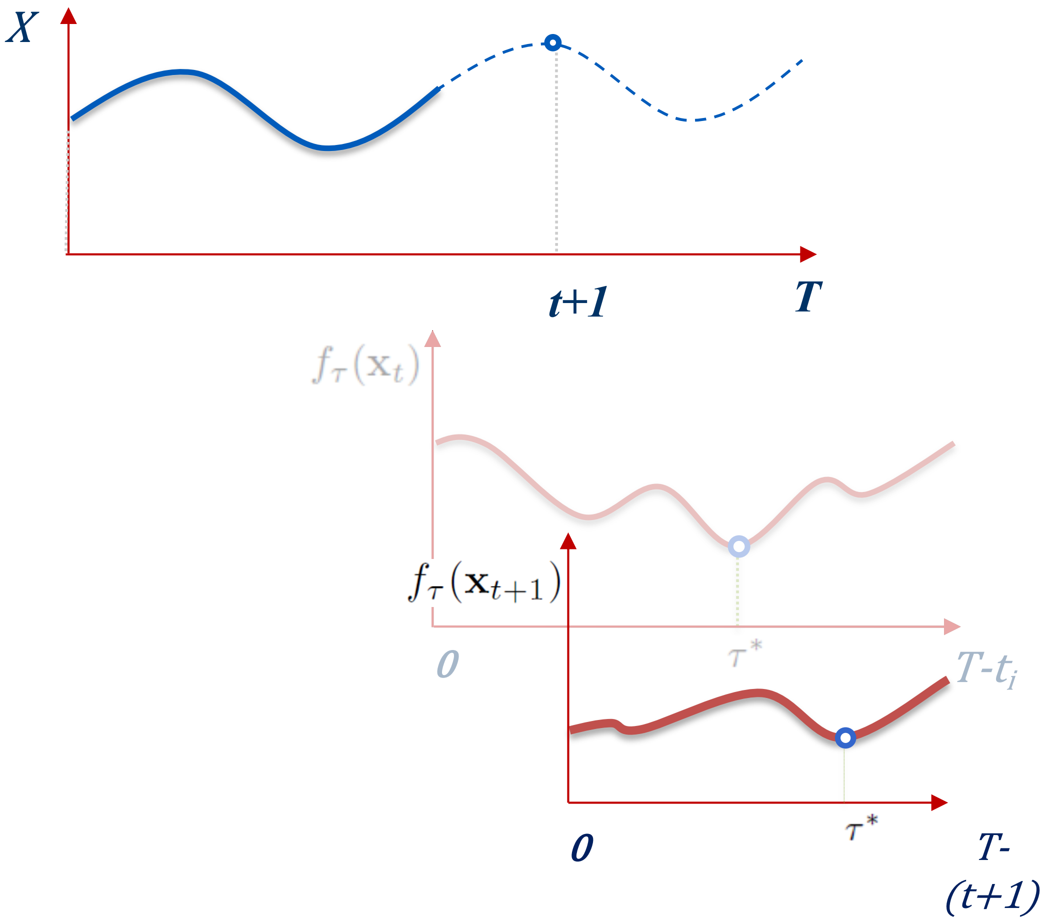


→ Continue monitoring

A non myopic decision process

- Optimal estimated time relative to current time *t*

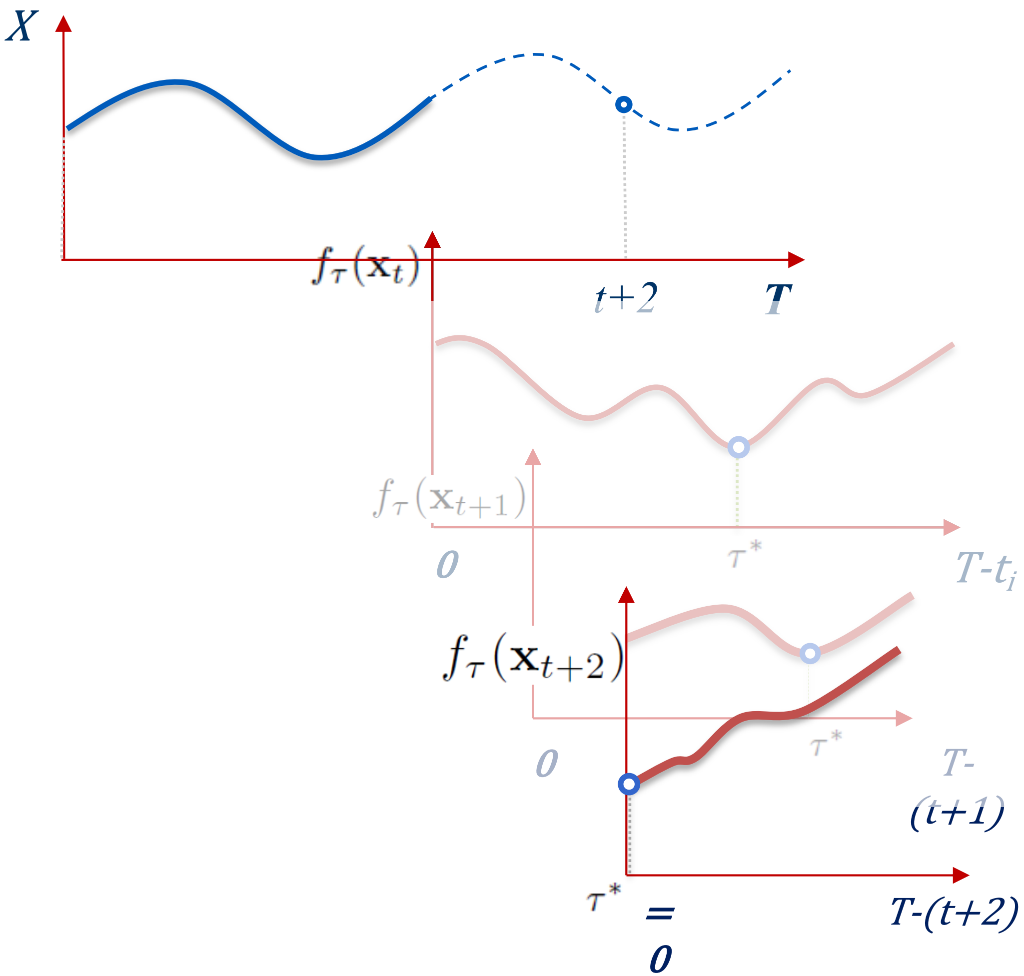
$$\tau^* = \underset{\tau \in \{0,\ldots,T-t\}}{\text{ArgMin}} \; f_\tau(\mathbf{x}_t)$$



Continue monitoring

A non myopic decision process

- Optimal estimated time relative to current time *t* $\tau^* = \underset{\tau \in \{0,\ldots,T-t\}}{\text{ArgMin}} \; f_\tau(\mathbf{x}_t)$

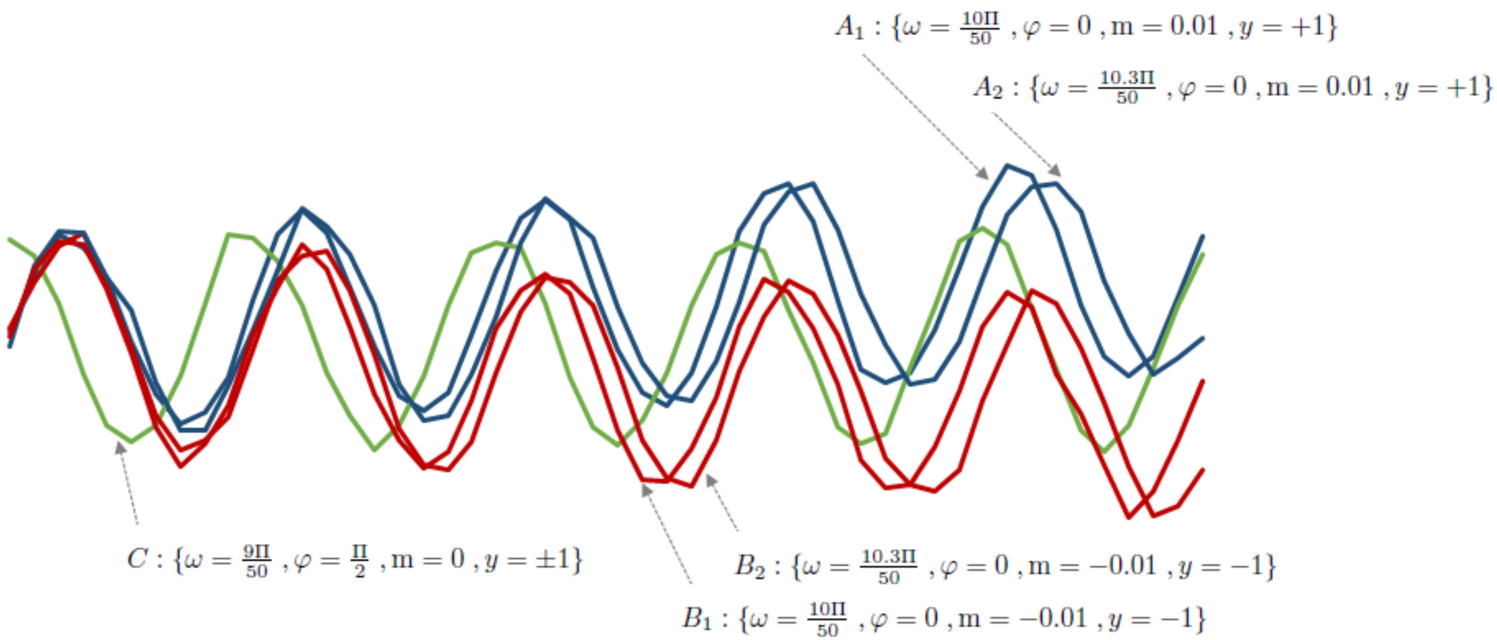


Take
decision

Experiments: Controlled data

- Control of
  - The time-dependent **information** provided: the **slopes** of the **classes**
  - The **shapes** of time series within each class
  - The **noise level**

$$\mathbf{x}_t = \underbrace{t \times \text{slope} \times \text{class}}_{\text{information gain}} + \underbrace{\mathrm{x}_{max} \sin(\omega_i \times t + \varphi_j)}_{\text{sub shape within class}} + \underbrace{\eta(t)}_{\text{noise factor}}$$



Results: effect of the *noise level*

Increasing the **noise level increases** the waiting time, and then it's no longer worth it

$C(t)$	$\pm b$ / $\varepsilon(t)$	0.02 $\bar{\tau}^\star$	0.02 $\sigma(\tau^\star)$	0.02 AUC	0.05 $\bar{\tau}^\star$	0.05 $\sigma(\tau^\star)$	0.05 AUC	0.07 $\bar{\tau}^\star$	0.07 $\sigma(\tau^\star)$	0.07 AUC
0.01	*0.2*	**9.0**	2.40	0.99	**9.0**	2.40	0.99	**10.0**	0.0	1.00
	0.5	**13.0**	4.40	0.98	**13.0**	4.40	0.98	**15.0**	0.18	1.00
	1.5	**24.0**	10.02	0.98	**32.0**	2.56	1.00	**30.0**	12.79	0.99
	5.0	**26.0**	7.78	0.84	**30.0**	18.91	0.87	**30.0**	19.14	0.88
	10.0	**38.0**	18.89	0.70	**48.0**	1.79	0.74	**46.0**	5.27	0.75
	15.0	**23.0**	15.88	0.61	**32.0**	13.88	0.64	**29.0**	17.80	0.62
	20.0	**7.0**	8.99	0.52	**11.0**	11.38	0.55	**4.0**	1.22	0.52
0.05	0.2	8.0	2.00	0.98	8.0	2.00	0.98	9.0	0.0	1.00
	0.5	10.0	2.80	0.96	8.0	4.0	0.98	14.0	0.41	0.99
	1.5	5.0	0.40	0.68	20.0	0.42	0.95	14.0	4.80	0.88
	5.0	8.0	3.87	0.68	6.0	1.36	0.64	5.0	0.50	0.65
	10.0	4.0	0.29	0.56	4.0	0.25	0.56	4.0	0.34	0.57
	15.0	4.0	0.0	0.54	4.0	0.25	0.56	4.0	0.0	0.55
	20.0	4.0	0.0	0.52	4.0	0.0	0.52	4.0	0.0	0.52
0.10	0.2	6.0	0.80	0.95	7.0	1.60	0.94	8.0	0.40	0.96
	0.5	6.0	0.80	0.84	9.0	2.40	0.93	10.0	0.0	0.95
	1.5	4.0	0.0	0.67	5.0	0.43	0.68	6.0	0.80	0.74
	5.0	4.0	0.07	0.64	4.0	0.05	0.64	4.0	0.11	0.64
	10.0	4.0	0.0	0.56	48.0	1.79	0.74	4.0	0.22	0.56
	15.0	4.0	0.0	0.55	4.0	0.0	0.55	4.0	0.0	0.55
	20.0	4.0	0.0	0.52	11.0	11.38	0.55	4.0	0.0	0.52

Table 1. Experimental results in function of the waiting cost $C(t) = \{0.01, 0.05, 0.1\} \times t$, the noise level $\varepsilon(t)$ and the trend parameter b.

Results: effect of the *waiting cost*

Increasing the **waiting cost** **reduces** the waiting time

$C(t)$	$\pm b$ $\varepsilon(t)$	0.02 $\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC	0.05 $\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC	0.07 $\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC
	0.2	9.0	2.40	0.99	9.0	2.40	0.99	**10.0**	0.0	1.00
	0.5	13.0	4.40	0.98	13.0	4.40	0.98	**15.0**	0.18	1.00
0.01	1.5	24.0	10.02	0.98	32.0	2.56	1.00	**30.0**	12.79	0.99
	5.0	26.0	7.78	0.84	30.0	18.91	0.87	**30.0**	19.14	0.88
	10.0	38.0	18.89	0.70	48.0	1.79	0.74	**46.0**	5.27	0.75
	15.0	23.0	15.88	0.61	32.0	13.88	0.64	**29.0**	17.80	0.62
	20.0	7.0	8.99	0.52	11.0	11.38	0.55	**4.0**	1.22	0.52
	0.2	8.0	2.00	0.98	8.0	2.00	0.98	**9.0**	0.0	1.00
	0.5	10.0	2.80	0.96	8.0	4.0	0.98	**14.0**	0.41	0.99
0.05	1.5	5.0	0.40	0.68	20.0	0.42	0.95	**14.0**	4.80	0.88
	5.0	8.0	3.87	0.68	6.0	1.36	0.64	**5.0**	0.50	0.65
	10.0	4.0	0.29	0.56	4.0	0.25	0.56	**4.0**	0.34	0.57
	15.0	4.0	0.0	0.54	4.0	0.25	0.56	**4.0**	0.0	0.55
	20.0	4.0	0.0	0.52	4.0	0.0	0.52	**4.0**	0.0	0.52
	0.2	6.0	0.80	0.95	7.0	1.60	0.94	**8.0**	0.40	0.96
	0.5	6.0	0.80	0.84	9.0	2.40	0.93	**10.0**	0.0	0.95
0.10	1.5	4.0	0.0	0.67	5.0	0.43	0.68	**6.0**	0.80	0.74
	5.0	4.0	0.07	0.64	4.0	0.05	0.64	**4.0**	0.11	0.64
	10.0	4.0	0.0	0.56	48.0	1.79	0.74	**4.0**	0.22	0.56
	15.0	4.0	0.0	0.55	4.0	0.0	0.55	**4.0**	0.0	0.55
	20.0	4.0	0.0	0.52	11.0	11.38	0.55	**4.0**	0.0	0.52

Table 2. Experimental results in function of the waiting cost $C(t) = \{0.01, 0.05, 0.1\} \times t$, the noise level $\varepsilon(t)$ and the trend parameter b.

Results: effect of the difference between classes

slope →

Increase of the difference between classes

The **performance** increases (AUC)

The *waiting time* is not much changed in these experiments

$C(t)$	$\pm b$ / $\varepsilon(t)$	**0.02** $\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC	**0.05** $\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC	**0.07** $\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC
0.01	0.2	*9.0*	2.40	**0.99**	*9.0*	2.40	**0.99**	*10.0*	0.0	**1.00**
	0.5	13.0	4.40	0.98	13.0	4.40	0.98	15.0	0.18	1.00
	1.5	24.0	10.02	0.98	32.0	2.56	1.00	30.0	12.79	0.99
	5.0	26.0	7.78	0.84	30.0	18.91	0.87	30.0	19.14	0.88
	10.0	38.0	18.89	0.70	48.0	1.79	0.74	46.0	5.27	0.75
	15.0	23.0	15.88	0.61	32.0	13.88	0.64	29.0	17.80	0.62
	20.0	7.0	8.99	0.52	11.0	11.38	0.55	4.0	1.22	0.52
0.05	0.2	8.0	2.00	0.98	8.0	2.00	0.98	9.0	0.0	1.00
	0.5	*10.0*	2.80	**0.96**	*8.0*	4.0	**0.98**	*14.0*	0.41	**0.99**
	1.5	5.0	0.40	0.68	20.0	0.42	0.95	14.0	4.80	0.88
	5.0	8.0	3.87	0.68	6.0	1.36	0.64	5.0	0.50	0.65
	10.0	4.0	0.29	0.56	4.0	0.25	0.56	4.0	0.34	0.57
	15.0	4.0	0.0	0.54	4.0	0.25	0.56	4.0	0.0	0.55
	20.0	4.0	0.0	0.52	4.0	0.0	0.52	4.0	0.0	0.52
0.10	0.2	6.0	0.80	0.95	7.0	1.60	0.94	8.0	0.40	0.96
	0.5	6.0	0.80	0.84	9.0	2.40	0.93	10.0	0.0	0.95
	1.5	*4.0*	0.0	**0.67**	*5.0*	0.43	**0.68**	*6.0*	0.80	**0.74**
	5.0	4.0	0.07	0.64	4.0	0.05	0.64	4.0	0.11	0.64
	10.0	4.0	0.0	0.56	48.0	1.79	0.74	4.0	0.22	0.56
	15.0	4.0	0.0	0.55	4.0	0.0	0.55	4.0	0.0	0.55
	20.0	4.0	0.0	0.52	11.0	11.38	0.55	4.0	0.0	0.52

Table 3. Experimental results in function of the waiting cost $C(t) = \{0.01, 0.05, 0.1\} \times t$, the noise level $\varepsilon(t)$ and the trend parameter b.

Are the decision times optimal?

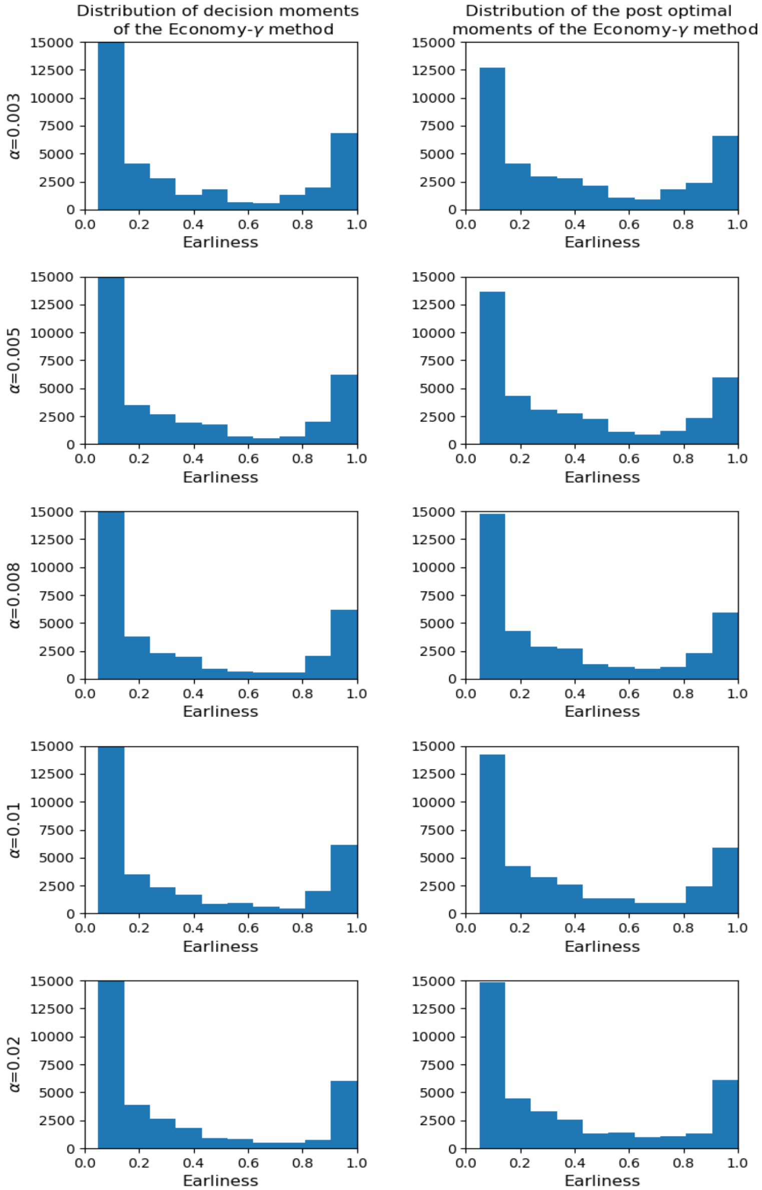


FIGURE B.2: The distribution of decision moments (left column) and post optimal moments (right column) for ECONOMY-γ with $\alpha \in \{0.003, 0.005, 0.008, 0.01, 0.02\}$

Outline

1. Ensemble methods: boosting

2. The LUPI framework

3. Illustration on Early Classification of Time Series

4. Conclusions

- Having more information at training time than at testing time might be useful
  - This information is **not directly available** in the test examples
  - **But** the learnt *hypothesis* (LUPI à la Vapnik)
    or the *decision rule* (ECTS) can use it

A kind of transfer learning

Ensemble methods

*for **unsupervised** learning?*

*Collaborative methods for **clustering***

- **What scenario** could motivate collaborative clustering?
  - Same descriptors but ≠ **examples** that you can not communicate
    -> *horizontal* scenario
  - Same examples but ≠ **descriptors** -> *vertical* scenario
  - *Hybrid* scenario

- **Goals**
  - **Consensus** clustering (*vertical* scenario)
  - Look for improved **local clusterings**

- What kind of **information** can be **exchanged**?
  - The **number** of clusters
  - The **centers** or prototypes of the clusters
  - Variance

Collaborative methods for clustering

- **Why** should that work? (bring better local clusterings)
  - More **confidence** in your results
    - As if you had **more information**
  - If disagreement ...
    - Escape **local minima**
    - Modifies your **local bias**

- **In which circumstances**, this should bring an improvement in performance?
  - **Are we certain** that collaborative clustering methods are **better**?

Collaborative methods for clustering

- Assumption
  - fortuitous and not meaningful solutions will **cancel each other** out
  - the **real structure** in the data should **emerge**

[A.P. Topchy, A.K. Jain, W.F. Punch, *Combining multiple weak clusterings*, in International Conference on Data Mining (ICDM), IEEE Computer Society, 2003, pp. 331–338.] proves this if the clusterings are **noisy versions** of the "true" clustering

Collaborative methods for clustering

- True state of the research

Largely an **open problem** still

Cornuéjols, A., Wemmert, C., Gançarski, P., & Bennani, Y. (2018). **Collaborative clustering: Why, when, what and how**. *Information Fusion*, *39*, 81-95.

Ingredients?

1. How to **select** "experts"?
   - What is an **expert**?
   - What is a **good panel** of experts?
2. How to **weight** them?
3. How to **combine** their results?

References

Références bibliographiques

- Bob Shapire and Yoav Freund
  Boosting: Foundations and Algorithms
  MIT Press, 2012

- Ron Meir and Gunnar Rätsch
  An introduction to Boosting and Leveraging
  In *Advanced Lectures on Machine Learning* (LNAI-2600), 2003
  http://www.boosting.org/papers/MeiRae03.pdf

- Zhi-Hua Zhou
  Ensemble Methods. Foundations and Algorithms
  CRC Press, 2012

- Vincent Barra, Antoine Cornuéjols & Laurent Miclet
  Apprentissage artificiel. Concepts et algorithmes. De Bayes et Hume au deep learning
  Eyrolles, 2021