Ensemble methods:

boosting, bagging, random forests

…

Antoine Cornuéjols

AgroParisTech

- Trouver **une** meilleure solution

  en combinant des solutions « faibles »

Méthodes « d'ensemble »

Les questions

1. Quels **agents** ?
2. Que doivent-ils **échanger entre eux** si processus itératif ?
3. Comment **combiner** leurs avis ?
4. Quelles **conditions** de **convergence** ?
5. Et si convergence, **vers quoi** ?

Ce n’est pas nouveau

Motivation

- « *The wisdom of crowd* »

  [James Surowiecki, 2004]

  – Estimation du **poids d'un panier** dans un marché

  787 participants

  [Francis Galton[1], 1906 (85 ans)]

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

Motivation

- « *The wisdom of crowd* »

  [James Surowiecki, 2004]

  – Estimation du **poids d'un panier** dans un marché

    787 participants

    - Le **meilleur** = plus d'un **centième** d'erreur
    - Moyenne : moins d'un **millième** d'erreur

    [Francis Galton[1], 1906 (85 ans)]

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

Pourquoi / comment ça marche

- « Experts faibles »

Estimations « bruitées »

–Non biaisées

–Symétriques

–Indépendantes

Combinaison simple :

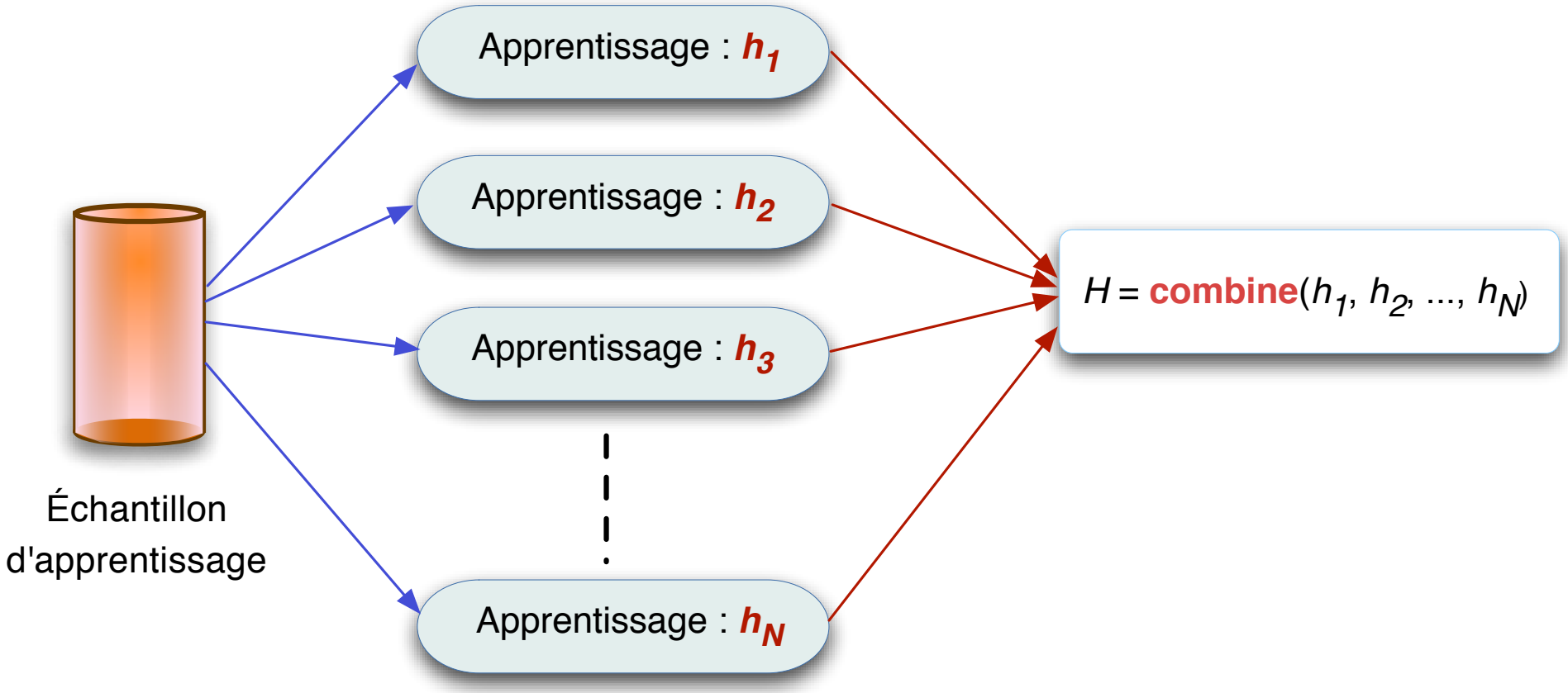
la moyenne

Plan

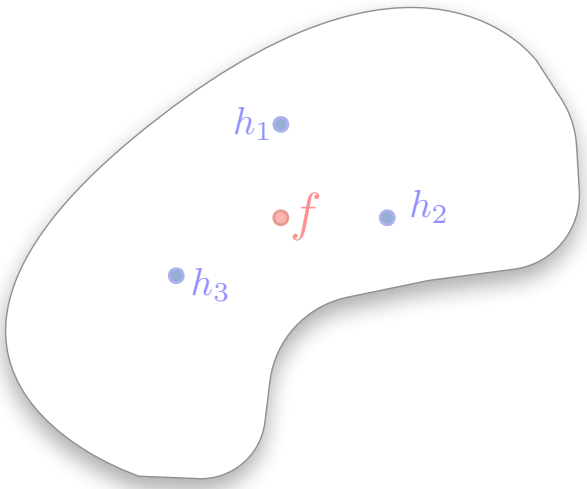
1. Méthodes d'ensemble
2. Le boosting
3. Le boosting : pourquoi ça marche
4. Le bagging
5. Les forêts aléatoires
6. XGBoost : le boosting d'arbres
7. Vers d'autre méthodes d'ensemble ?

Ensemble methods

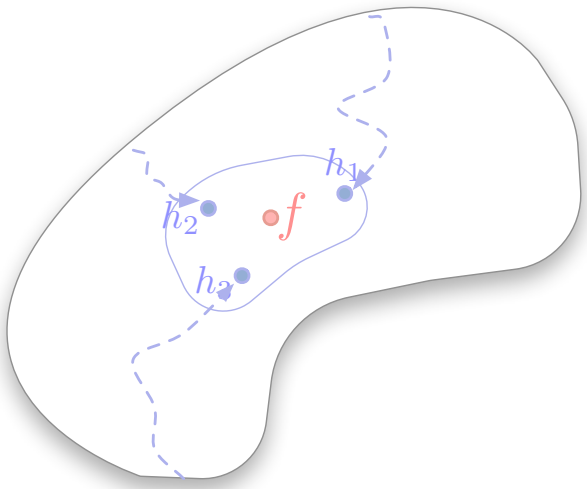
Schéma général : apprentissage



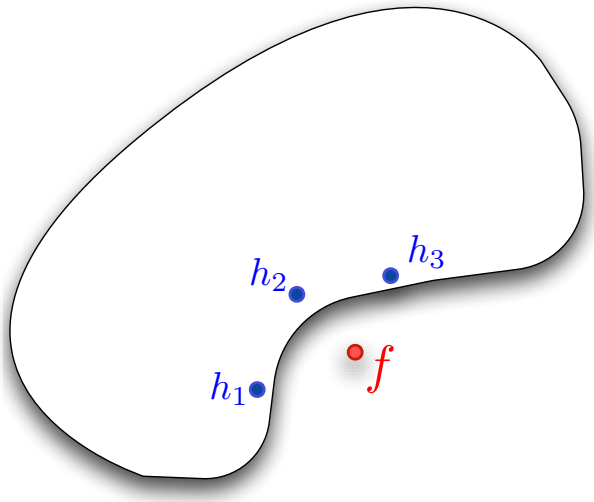
Justifications intuitives



Justification
statistique



Justification
computationnelle



Justification
représentationnelle

- Dietterich T. (2000) « Ensemble Methods in Machine Learning ». Proc. 1st Int. *Workshop on Multiple Classifier Systems*, Sardinia, Italy, 2000.

Succès applicatifs

- **KDD-Cup**
  - *Network intrusion detection (1999) ; molecular bioactivity & protein locale prediction (2001) ; (…) pulmonary embolisme detection (2006) ; customer relationship management (2009) ; educational data mining (2010) ; music recommandation (2011) ; …*
  - Tous les 1er prix et 2ème prix pour les **méthodes d'ensemble** (2009-2011 - ???)
- **Netflix prize**
  - *Improve accuracy about how much someone is going to enjoy a movie based on their preference*
  - 1 000 000 $ for a new algorithm improving on Netflix'one by more than 10%
  - Winner in 2009 for an **ensemble method**

Plan

1. Méthodes d'ensemble
2. Le boosting
3. Le boosting : pourquoi ça marche
4. Le bagging
5. Les forêts aléatoires
6. XGBoost : le boosting d'arbres
7. Vers d'autre méthodes d'ensemble ?

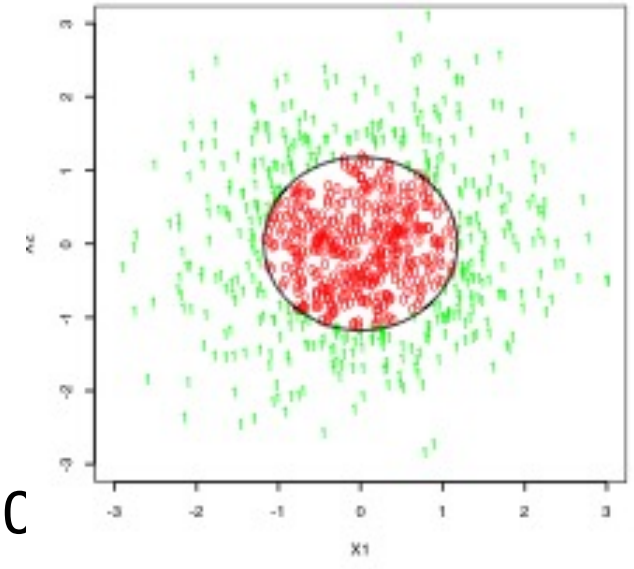
Boosting

Illustration

- Soit $\mathcal{X}$ un espace d'entrée à **10 dimensions**
- Les attributs sont indépendants et de distribution gaussienne
- Le **concept cible** est défini par :

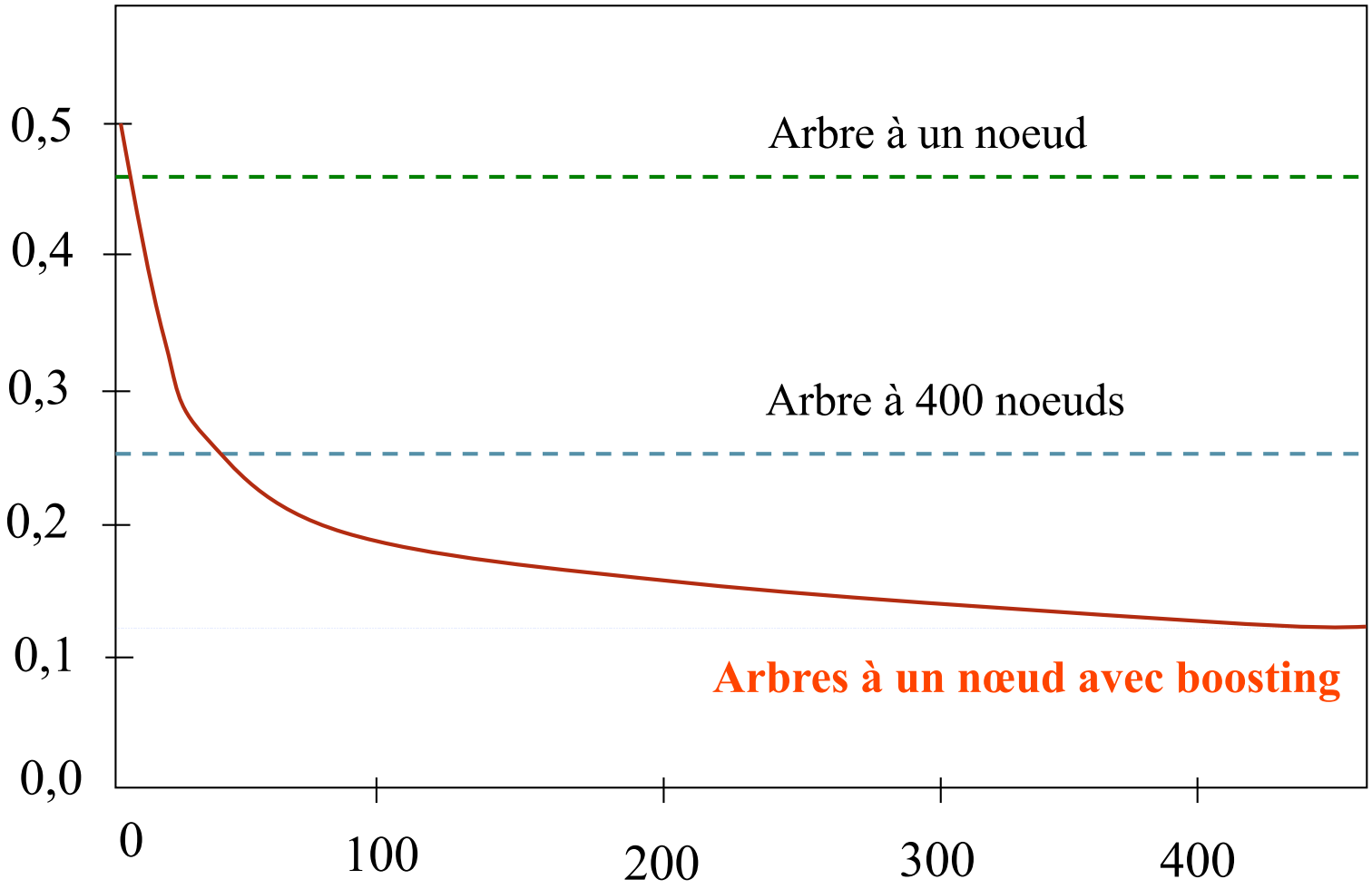
$$u = \begin{cases} 1 & \text{si } \sum_{j=1,10} x_j^2 > \chi^2_{10}(0,5) \\ -1 & \text{sinon} \end{cases}$$

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

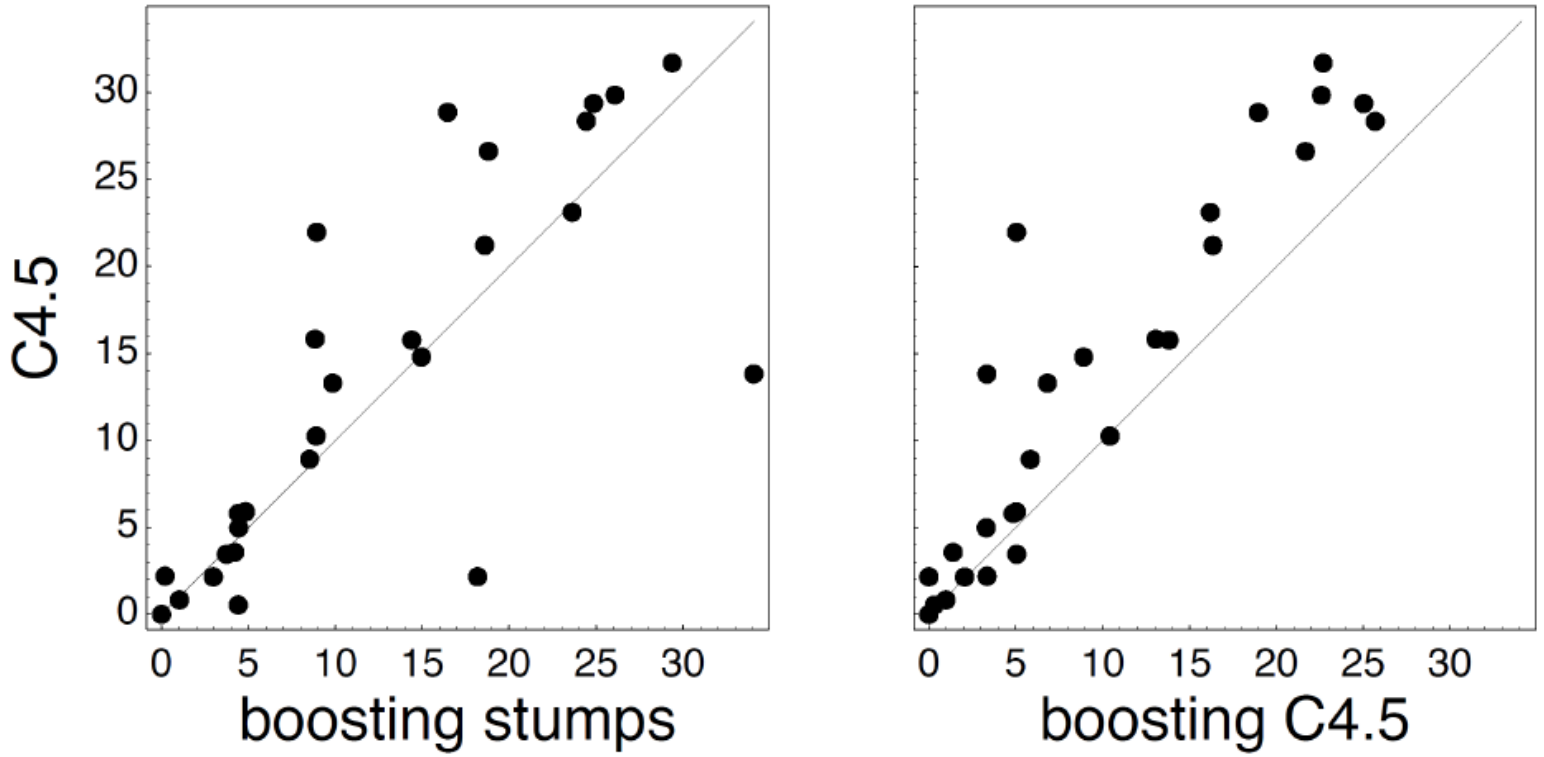


- 2000 *exemples d'apprentissage* (1000+;10
- 10000 *exemples de test*
- Apprentissage d'**arbres de décision**

Illustration

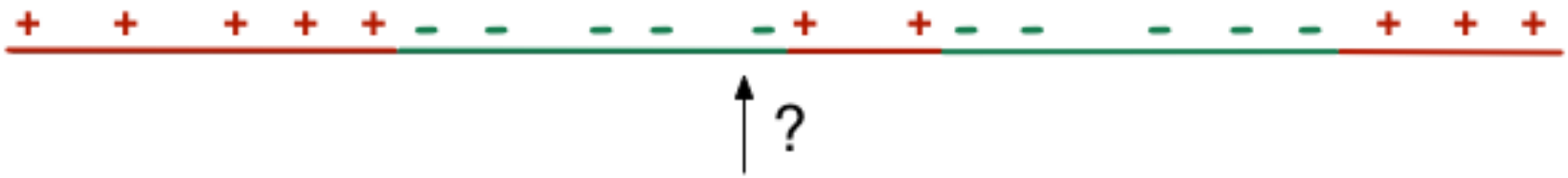


Performances du boosting



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

Exemple simple



- Quel est le meilleur séparateur linéaire ?

Exemple simple

- Taux d'erreur = 5/20 = 0.25

Exemple simple

- Taux d'erreur (h_1) = 5/20 = 0.25

Et si je pouvais combiner avec un autre séparateur linéaire ?

Ou même plusieurs autres !

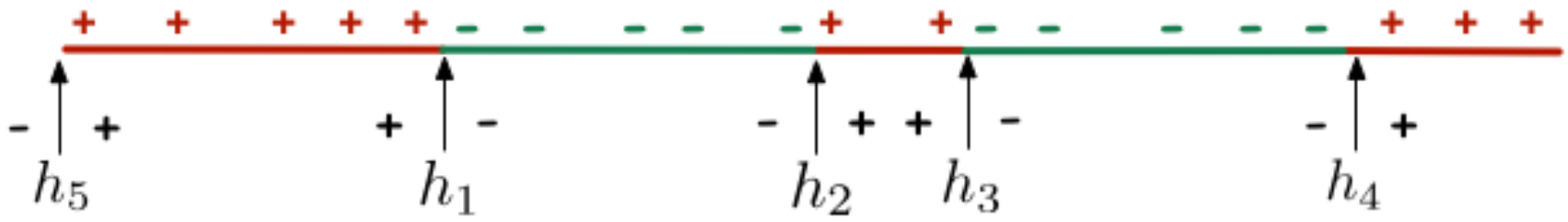
Exemple simple

Et si je pouvais combiner avec un autre séparateur linéaire ? Ou même plusieurs autres !

Par exemple en utilisant un **vote pondéré** :

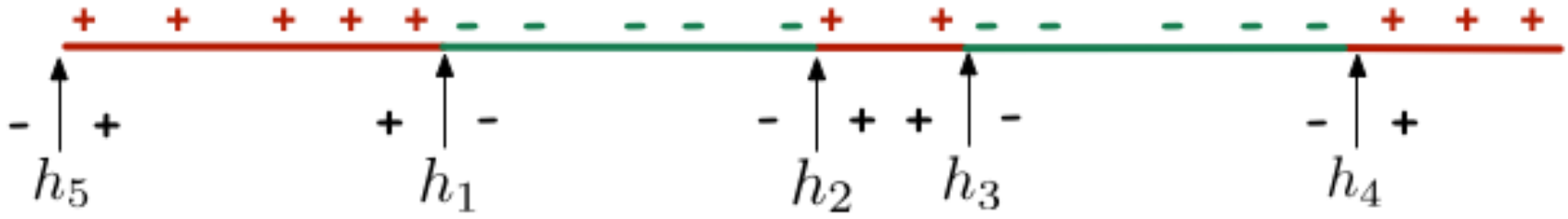
$$H(\mathbf{x}) = \text{sign}\left\{\sum_{i=1}^{l} \alpha_i \, h_i(\mathbf{x})\right\}$$

Exemple simple



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

Exemple simple



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

- Comment arriver à ce genre de combinaison ?

Algorithme du boosting

The boosting algorithm

Une question théorique

- Apprentissage « **fort** » (PAC learning)
  - Une classe de fonctions $\mathcal{F}$ est apprenable (au sens fort) si il existe un algorithme d'apprentissage $\mathcal{A}$ qui pour toute distribution $\mathcal{D}_X$ sur $\mathcal{X}$, et pour toute fonction f dans $\mathcal{F}$ est tel que :

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

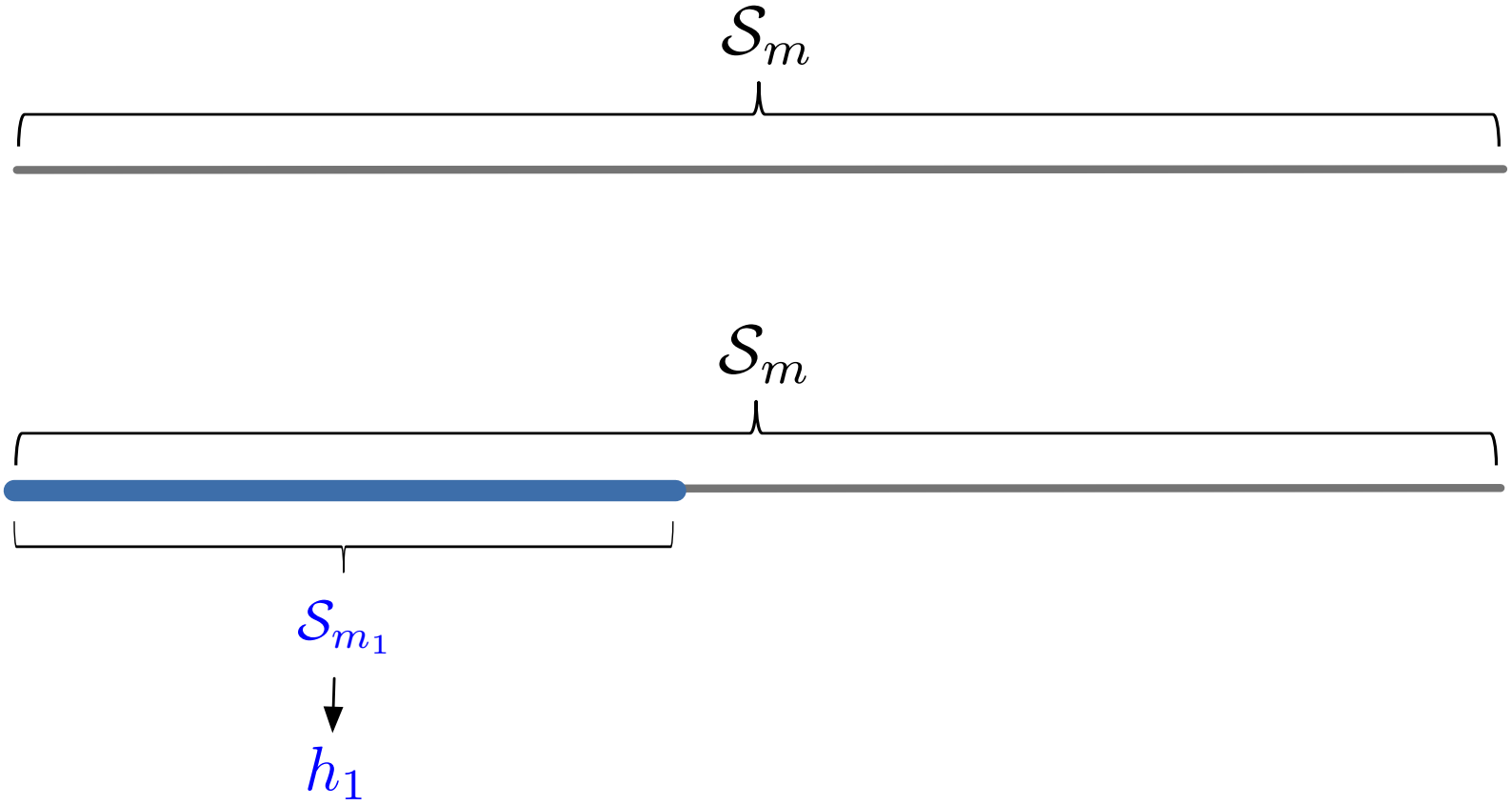
- Apprentissage « γ **faible** »
  - Une classe de fonctions $\mathcal{F}$ est apprenable (au sens faible) si, pour $\gamma > 0$, il existe un algorithme d'apprentissage $\mathcal{A}$ qui pour toute distribution $\mathcal{D}_X$ sur $\mathcal{X}$, et pour toute fonction f dans $\mathcal{F}$ est tel que :

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

- Sont-ils de nature différente ?

Comment engendrer les apprenants

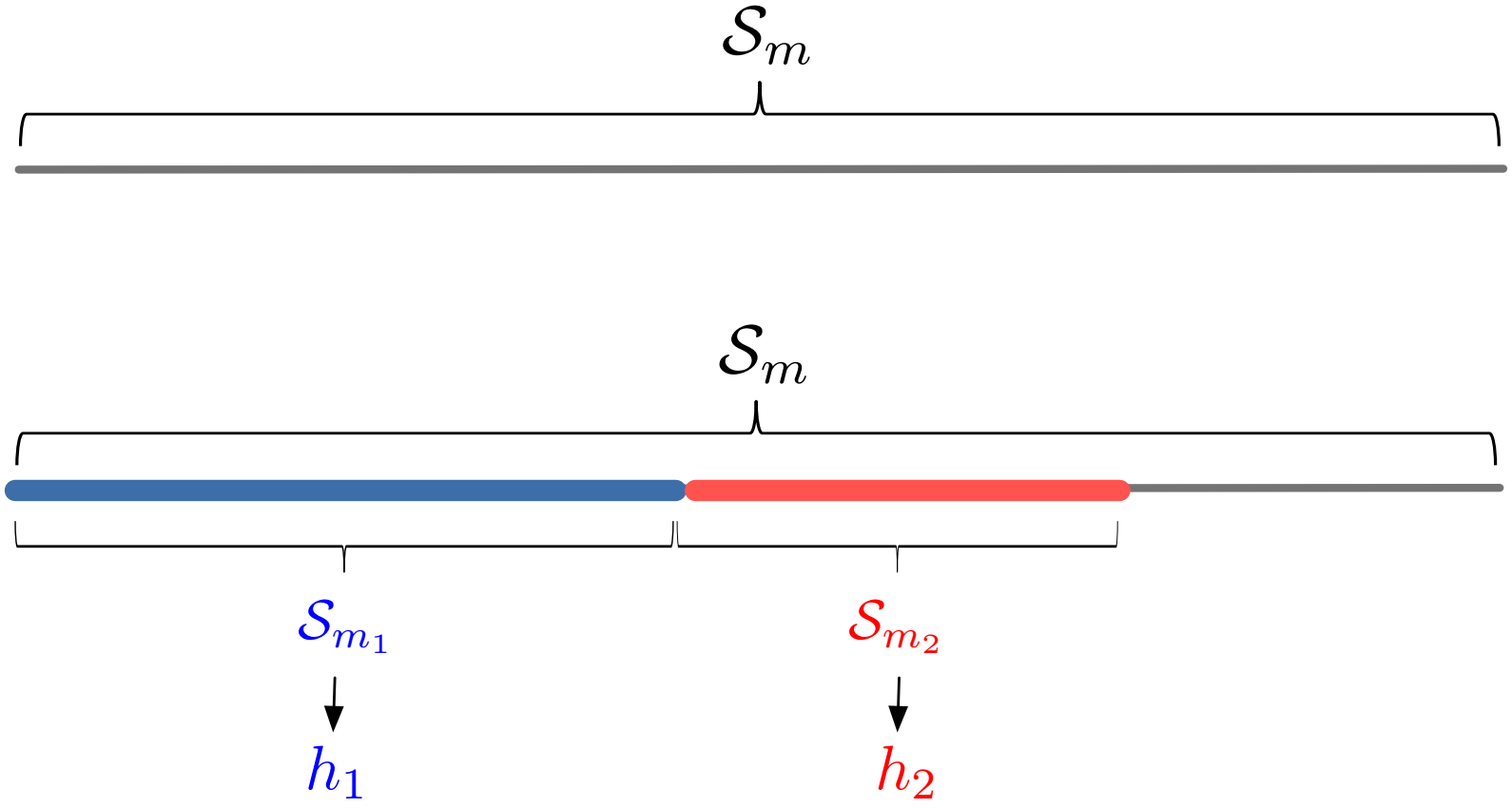
- Une recette historique



SCHAPIRE, Robert E. The strength of weak learnability. *Machine learning*, 1990, vol. 5, p. 197-227.

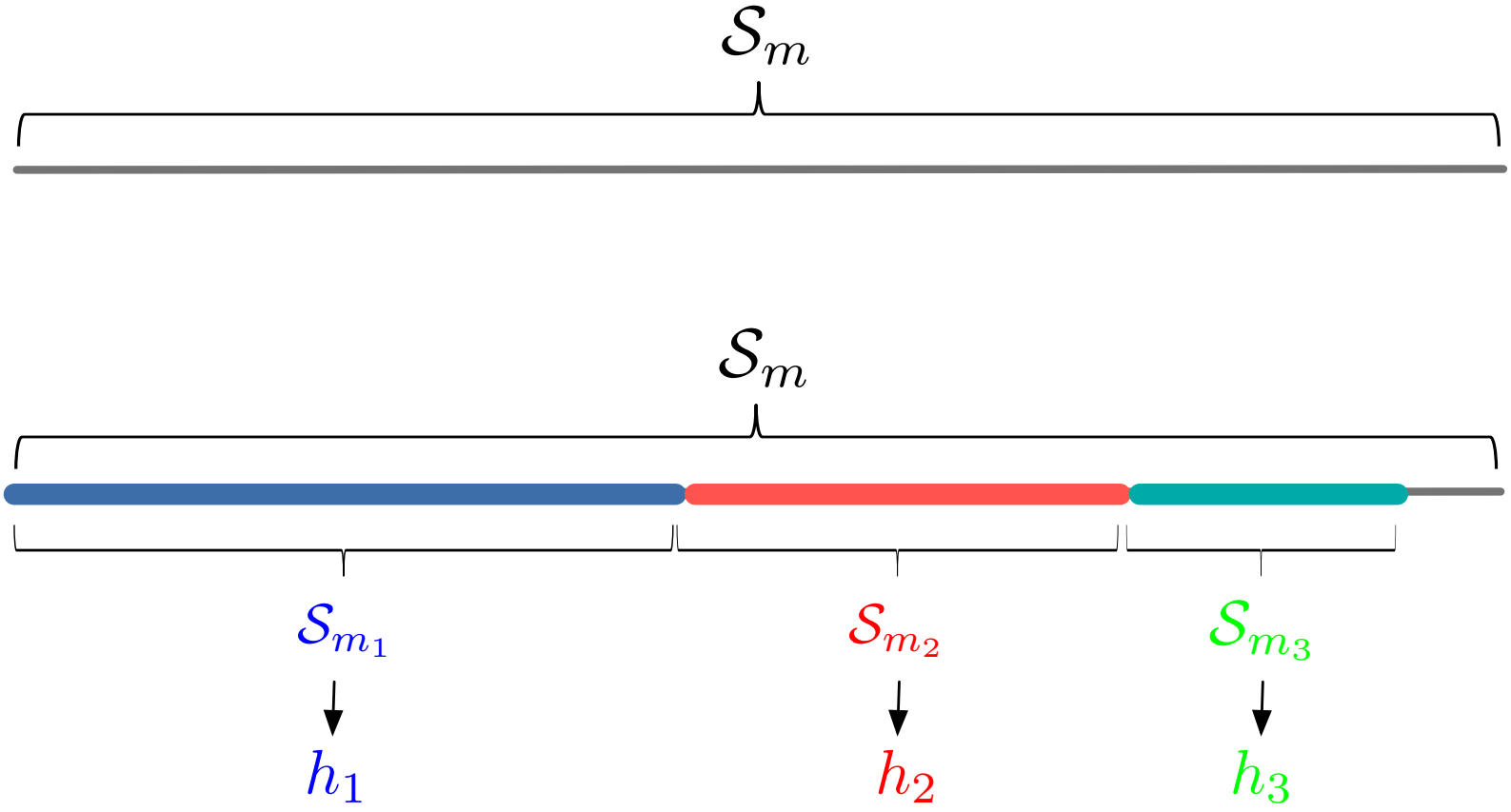
Comment engendrer les apprenants

- Une recette historique



Comment engendrer les apprenants

- Une recette historique



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

Questions

- Comment engendrer des **apprenants faibles décorrélés** ?
- Comment **combiner** leurs prédictions ?

Comment engendrer les apprenants (suite)

- Modifier l'échantillon d'apprentissage à chaque étape
  - *En **diminuant** l'importance des exemples **bien** classés*
  - *En **augmentant** ---------- **mal** --------*
  - De combien ?

Boosting

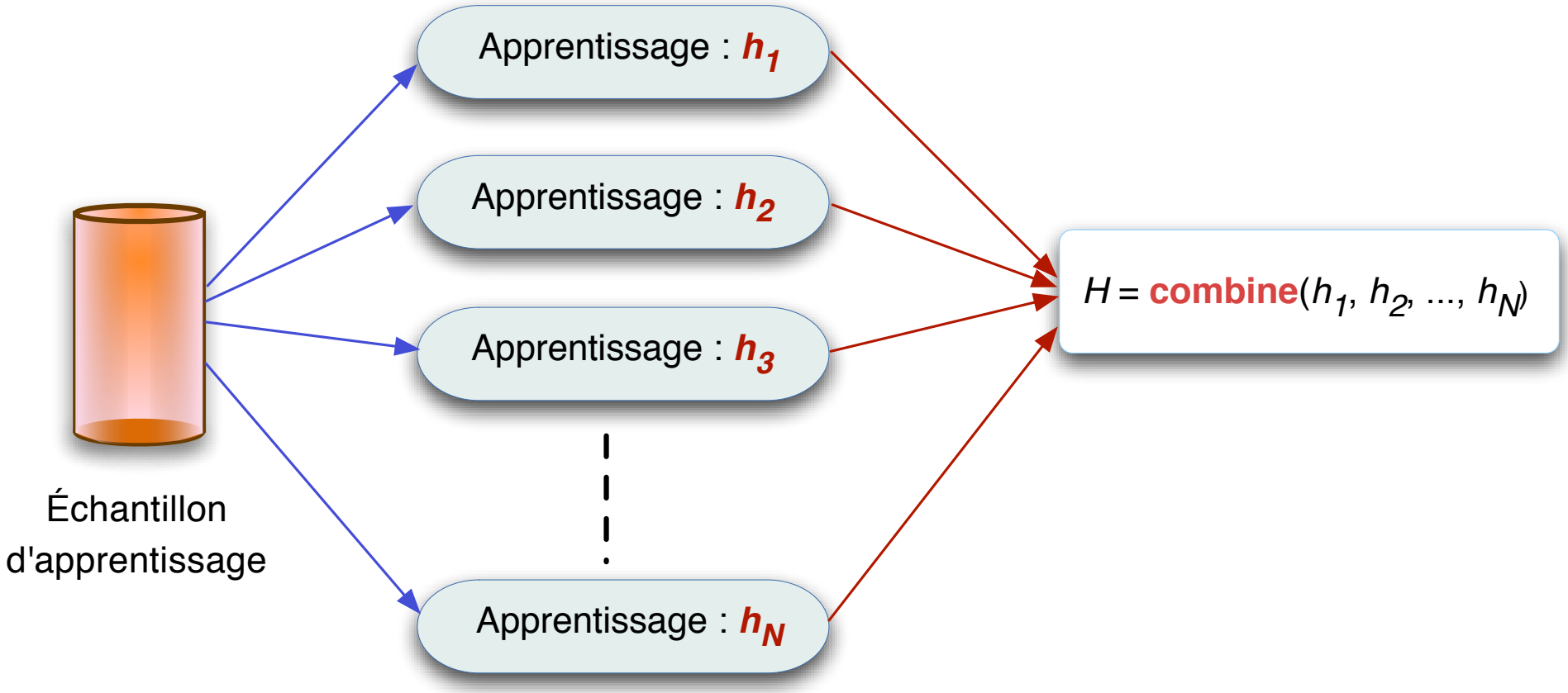
- **boosting** = méthode générale pour convertir des règles de prédiction peu performantes en une règle de prédiction (très) performante

- Plus précisément :

  – Étant donné un algorithme d'apprentissage "faible" qui peut toujours retourner une hypothèse de taux d'erreur $\leq 1/2-\gamma$

  – Un algorithme de boosting peut construire (de manière prouvée) une règle de décision (hypothèse) de taux d'erreur $\leq \varepsilon$

Schéma général : apprentissage



Questions

- Comment choisir les courses à chaque étape?
  - Se concentrer sur les courses les plus “difficiles”

    (celles sur lesquelles les heuristiques précédentes sont les moins performantes)

- Comment combiner les heuristiques (règles de prédiction) en une seule règle de prédiction ?
  - Prendre une vote (pondéré) majoritaire de ces règles

Boosting : vue formelle

- Étant donné l'échantillon d'apprentissage $S = \{(x_1,y_1),\ldots,(x_m,y_m)\}$
- $y_i \in \{-1,+1\}$ étiquette de l'exemple $x_i \in S$

- Pour $t = 1,\ldots,T$:

  Construire la distribution D_t sur $\{1,\ldots,m\}$

  Trouver l'hypothèse faible ("heuristique")

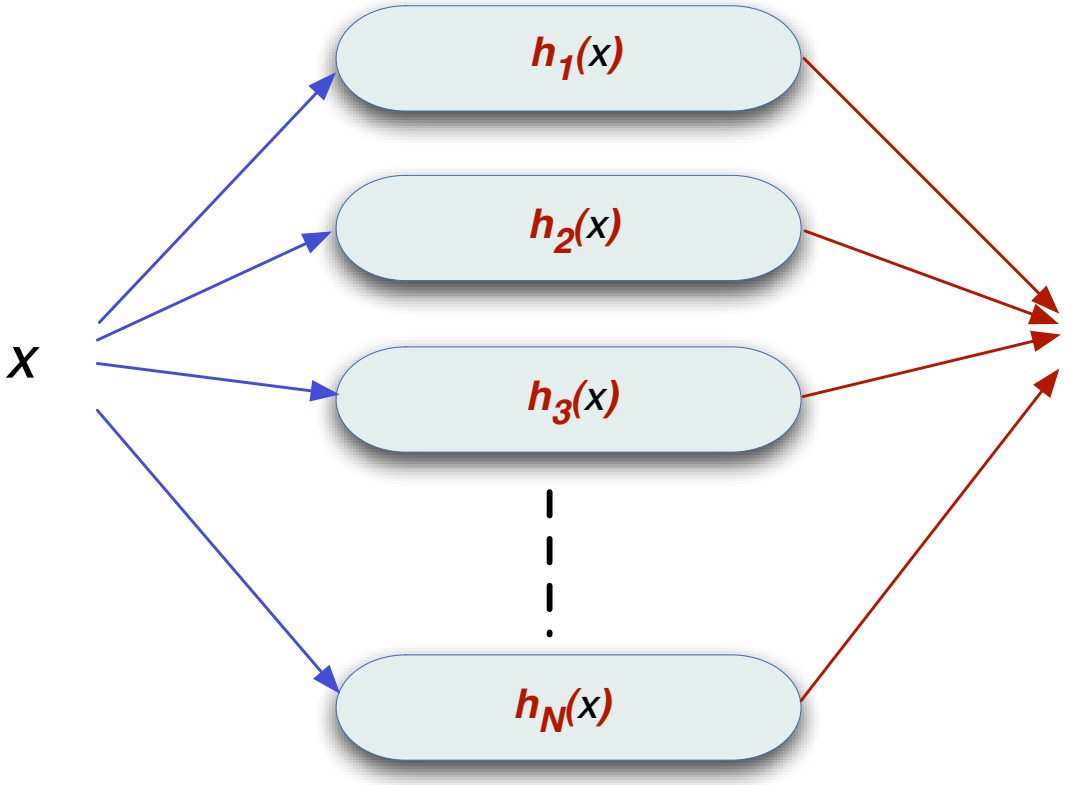
  $$h_t : S \rightarrow \{-1,+1\}$$

  avec erreur petite ε_t sur D_t:

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

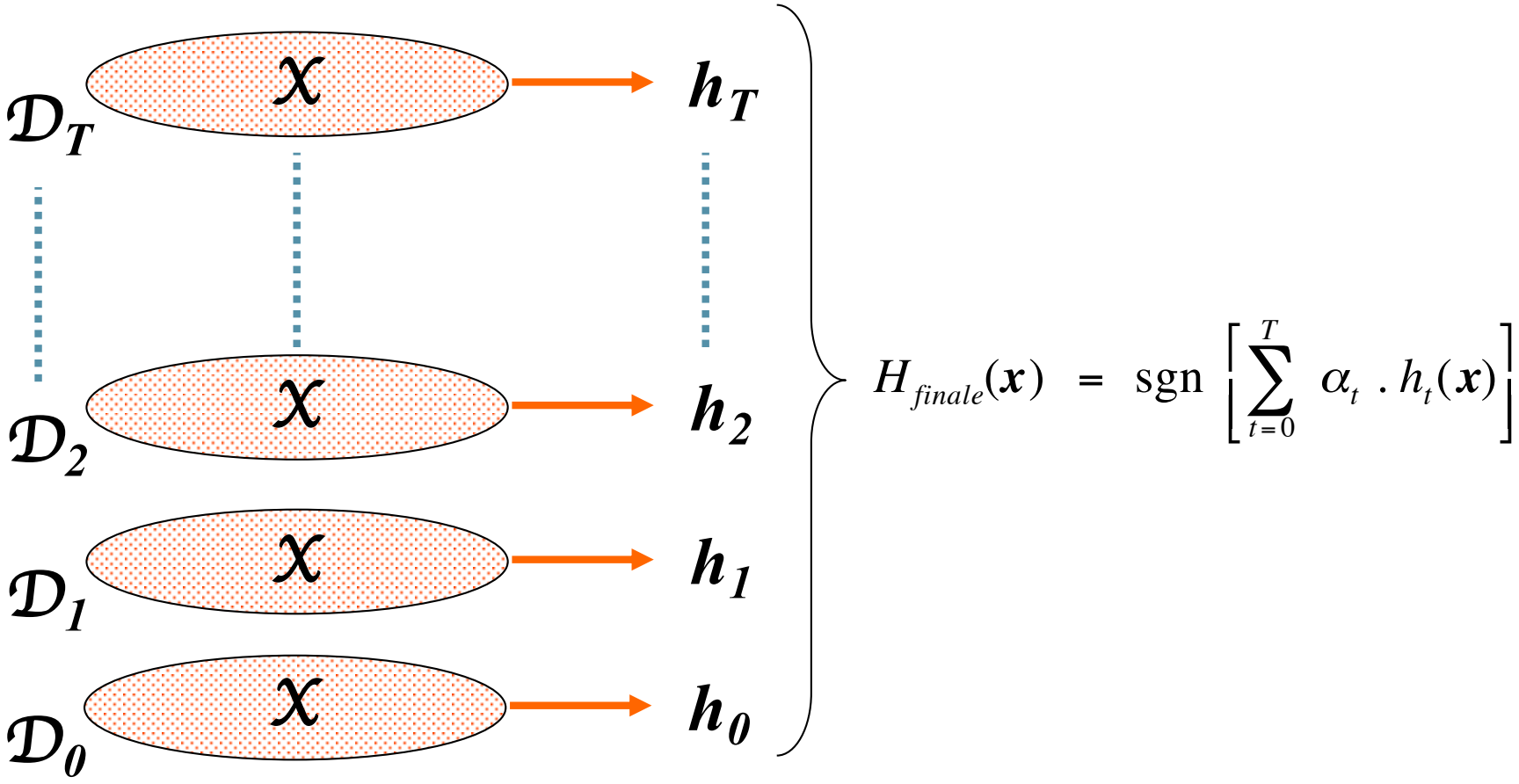
- Retourner l'hypothèse finale h_{final}

Schéma général : prédiction



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

Le principe général



- **Comment passer de $\mathcal{D}_t$ à $\mathcal{D}_{t+1}$?**
- **Comment calculer la pondération α_t ?**

AdaBoost [Freund&Schapire ’97]

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

Étant donnée D_t et h_t:

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

$$= \frac{D_t}{Z_t} \cdot \exp(-\alpha_t \cdot y_i \cdot h_t(x_i))$$

où: Z_t = constante de normalisation

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

- Hypothèse finale : $H_{\text{final}}(x) = \text{sgn}\left(\sum_t \alpha_t h_t(x)\right)$

FREUND, Yoav, SCHAPIRE, Robert E., *et al.* Experiments with a new boosting algorithm. In : *icml*. 1996. p. 148-156.

AdaBoost en plus gros

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

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

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

Exemple simple

- Taux d'erreur = 5/20 = 0.25

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

Exemple simple

- Nouvelle pondération des exemples d'apprentissage

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

- Exemples **mal** classés $$p_m(x) = \frac{e^{\alpha}}{Z} = \frac{e^{0.549}}{Z} = \frac{1.732}{Z}$$

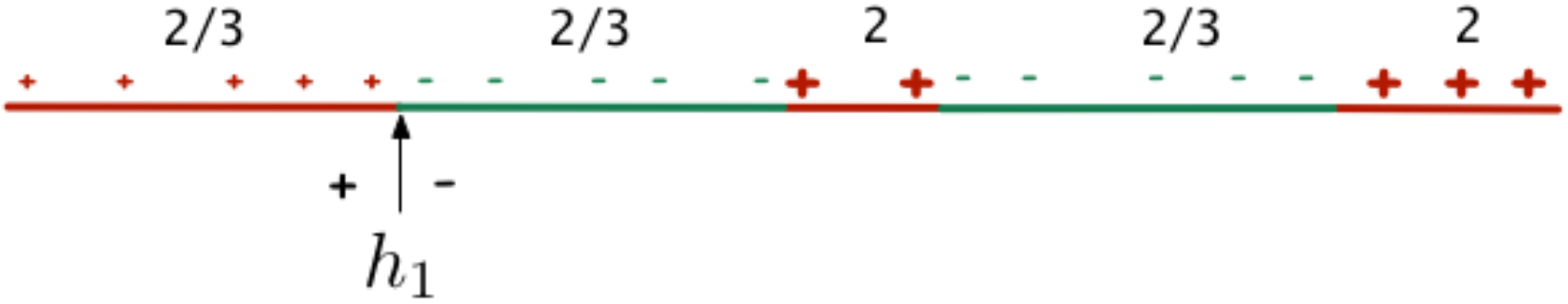
$$Z = \frac{(15 \times 0.577) + (5 \times 1.732)}{20} = \frac{8.660 + 8.660}{20} = \frac{17.32}{20} = 0.866$$

$$p_b(x) = 0.666 = 2/3$$

$$p_m(x) = 2$$

Exemple simple

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



- Exemples **bien** classés

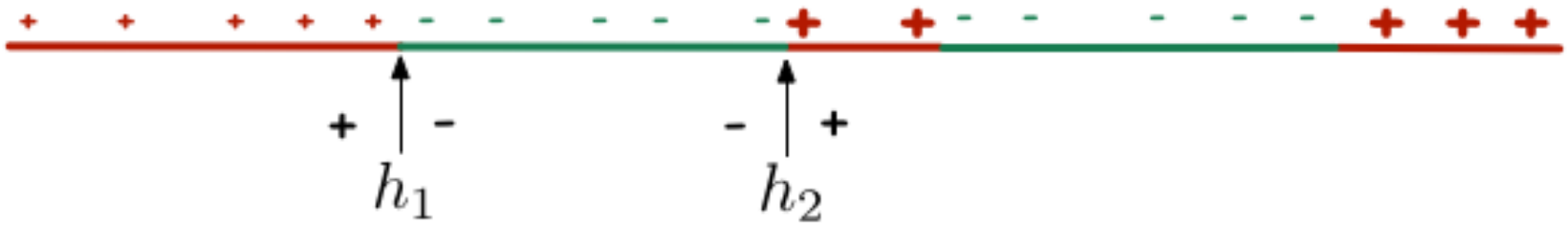
$$p_b(x) = \frac{1}{2\,(1-\varepsilon)} = \frac{1}{2 \times 0.75} = \frac{1}{1.5} = \frac{2}{3}$$

- Exemples **mal** classés

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

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

Exemple simple

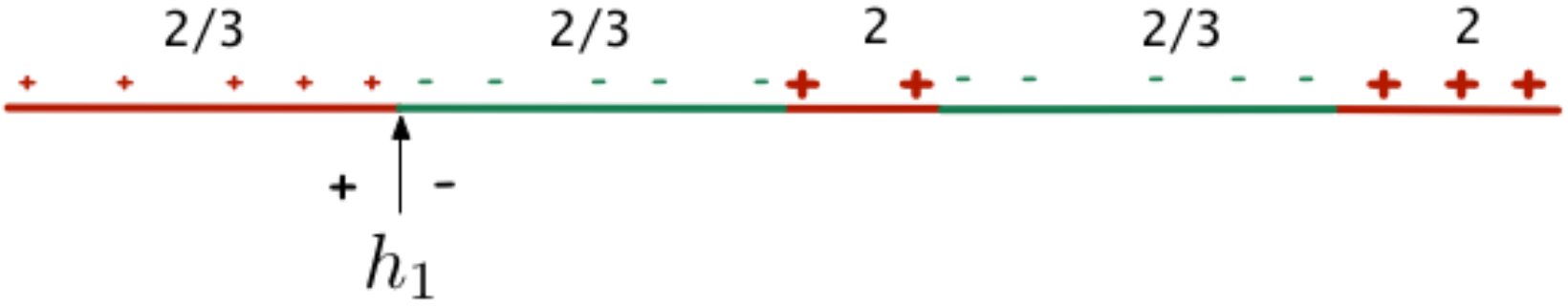


- Taux d'erreur : $\varepsilon_2 = \dfrac{10 \times 2/3}{20} = \dfrac{1}{3}$

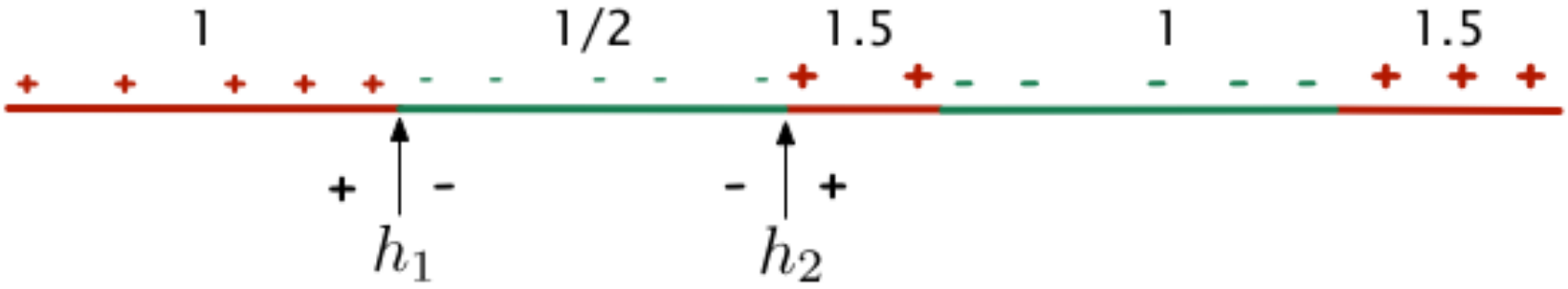
$$\alpha_2 = \frac{1}{2} \ln \frac{1-\varepsilon_2}{\varepsilon_2} = \frac{1}{2} \ln \frac{2/3}{1/3} = 0.347$$

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

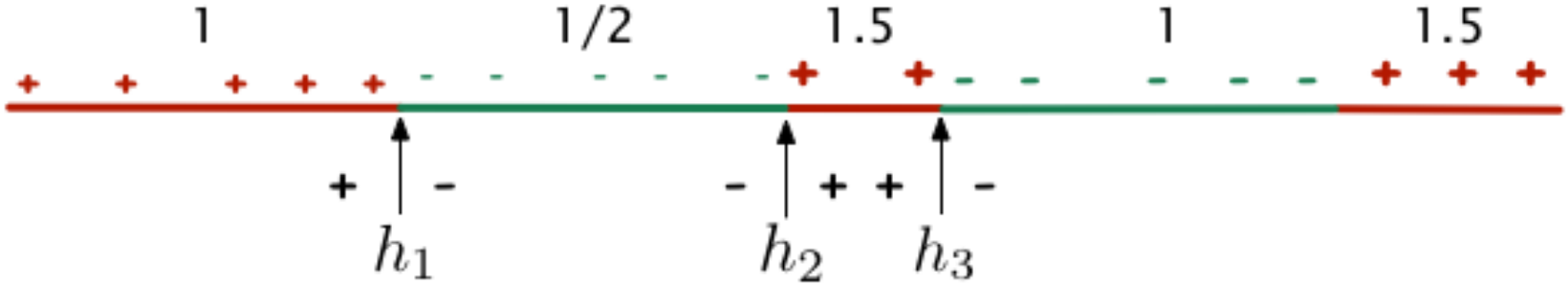
Exemple simple



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



Exemple simple

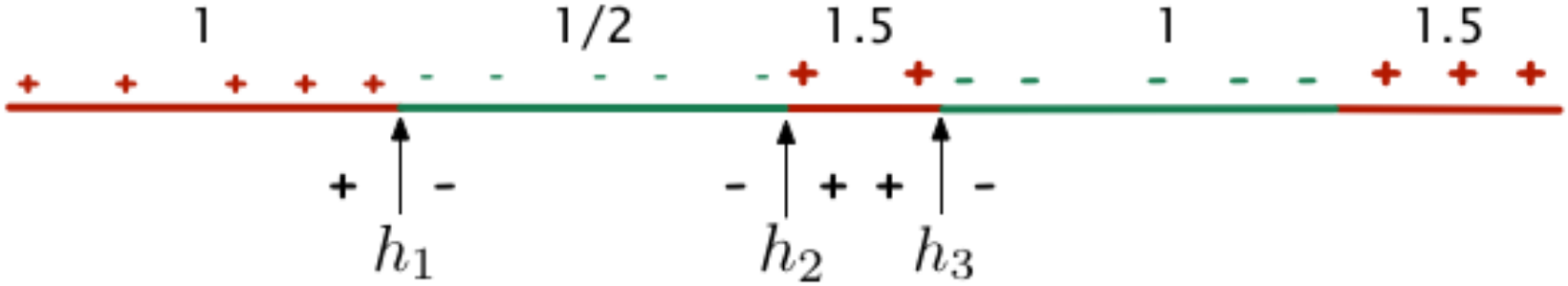


- Taux d'erreur : $\varepsilon_3 = \dfrac{(5 \times 1/2) + (3 \times 1.5)}{20} = \dfrac{7}{20} = 0.35$

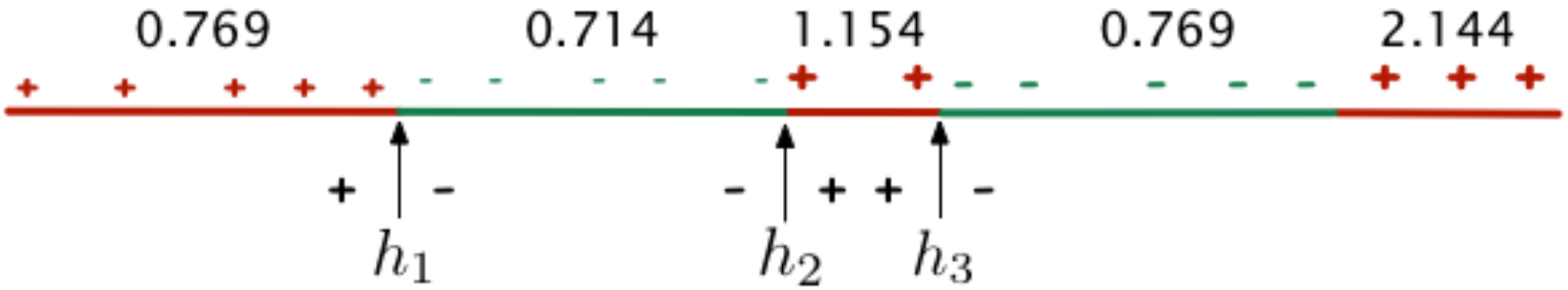
$$\alpha_3 = \frac{1}{2} \ln \frac{1 - \varepsilon_2}{\varepsilon_2} = \frac{1}{2} \ln \frac{0.65}{0.35} = 0.310$$

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

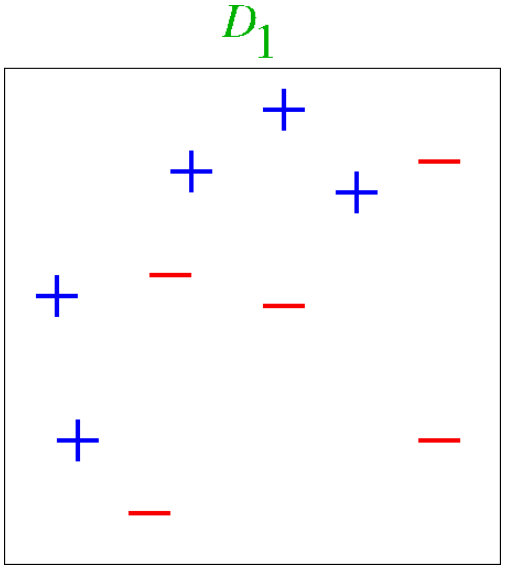
Exemple simple



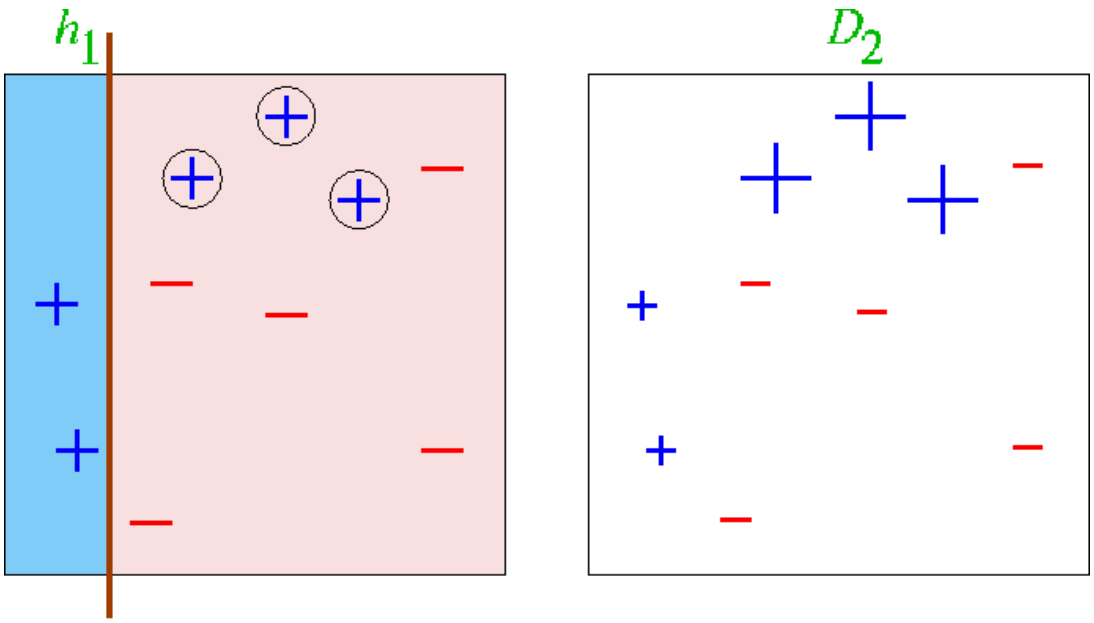
- Sous-pondération des bien classés : $p_b(x) = \frac{1}{2\,(1-\varepsilon)} = \frac{1}{2 \times 0.65} = \frac{1}{1.3} = 0.769$
- Sur-pondération des mal classés : $p_m(x) = \frac{1}{2\,\varepsilon} = \frac{1}{2 \times 0.35} = \frac{1}{0.7} = 1.429$



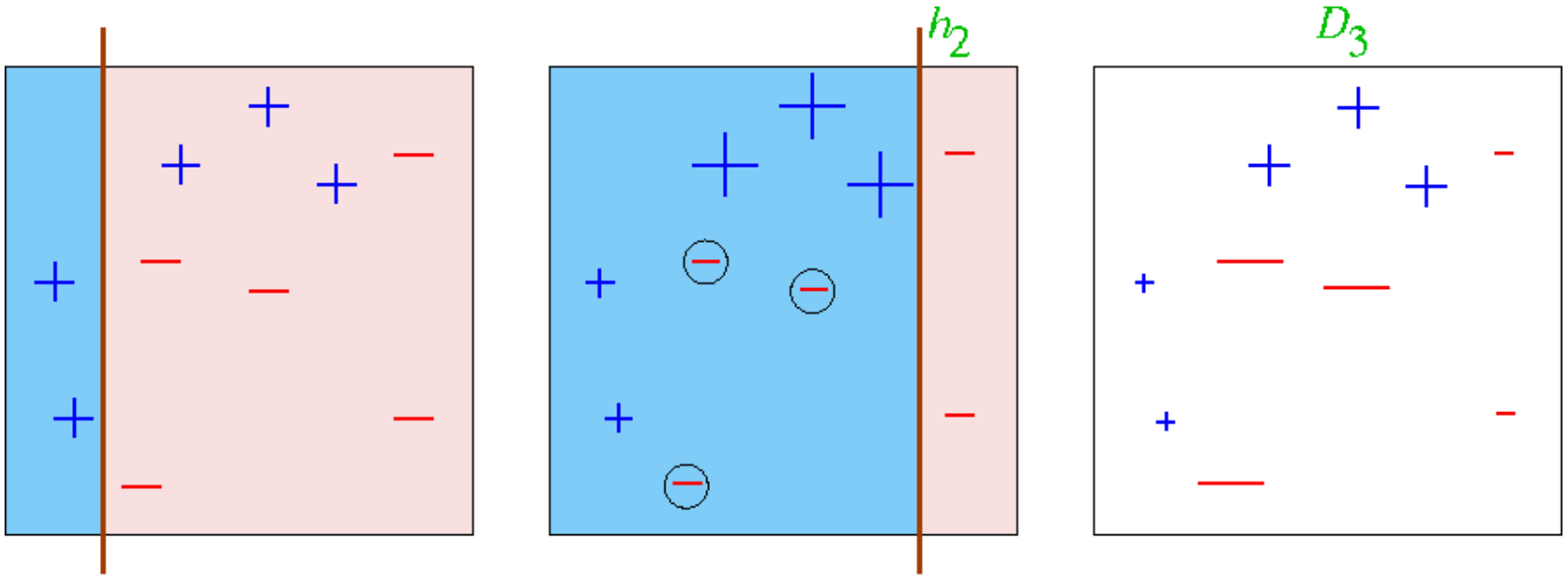
Exemple jouet



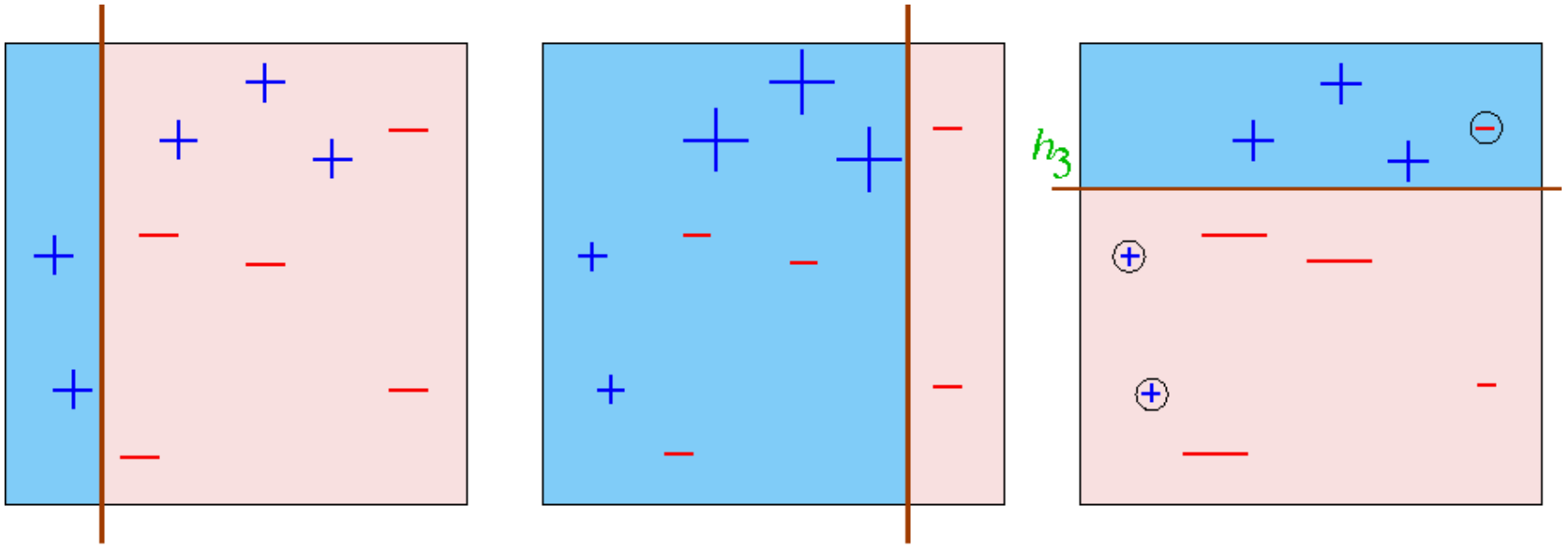
Étape 1



Étape 2



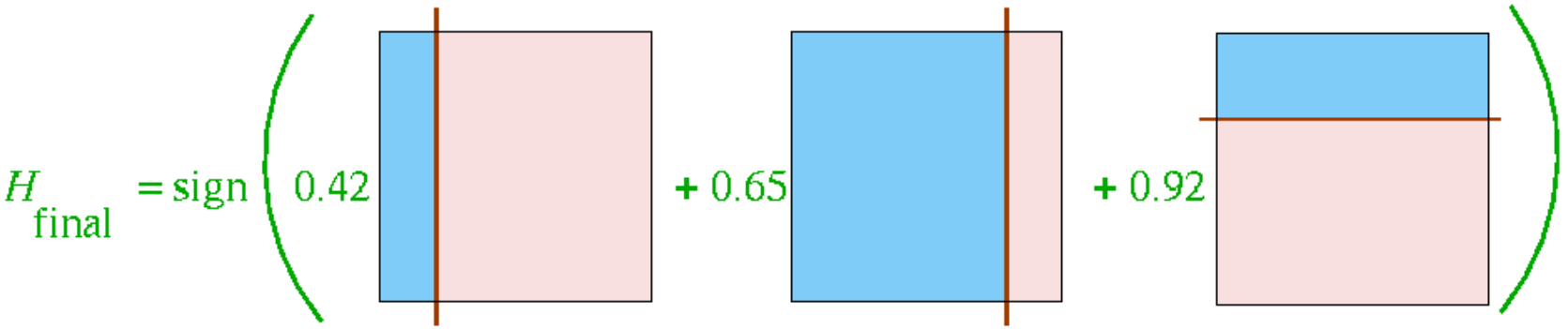
Étape 3

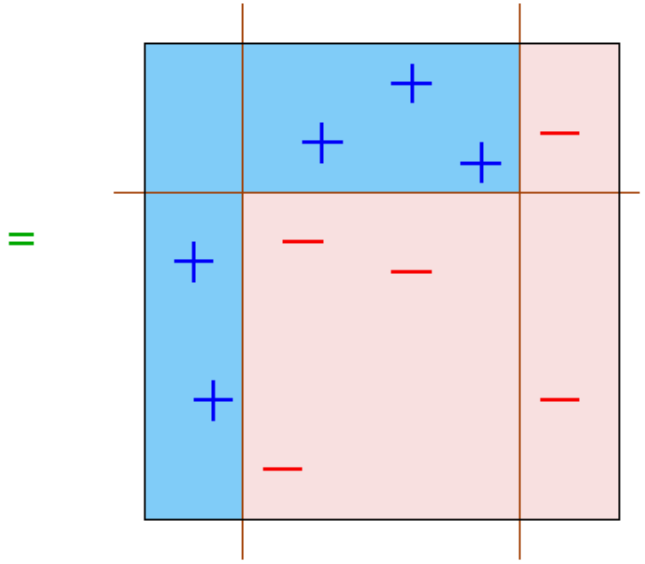


$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

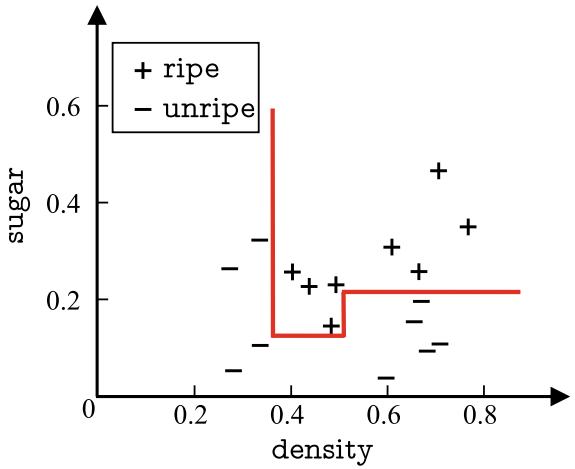
Hypothèse finale





Autre illustration du boosting sur jeu de données

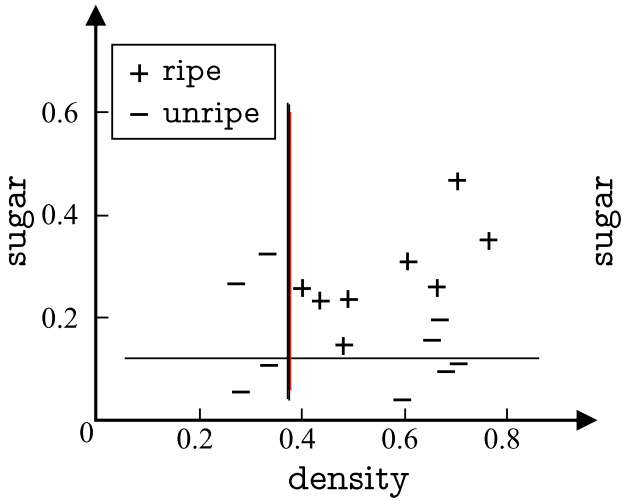
ID	density	sugar	ripe
1	0.697	0.460	true
2	0.774	0.376	true
3	0.634	0.264	true
4	0.608	0.318	true
5	0.556	0.215	true
6	0.403	0.237	true
7	0.481	0.149	true
8	0.437	0.211	true
9	0.666	0.091	false
10	0.243	0.267	false
11	0.245	0.057	false
12	0.343	0.099	false
13	0.639	0.161	false
14	0.657	0.198	false
15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



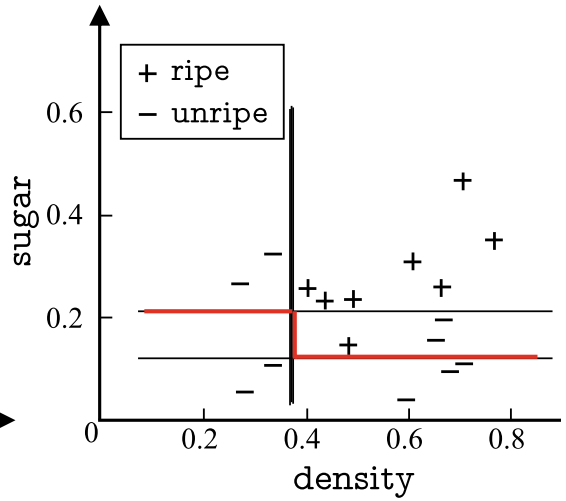
Apprentissage par

arbre de décisions

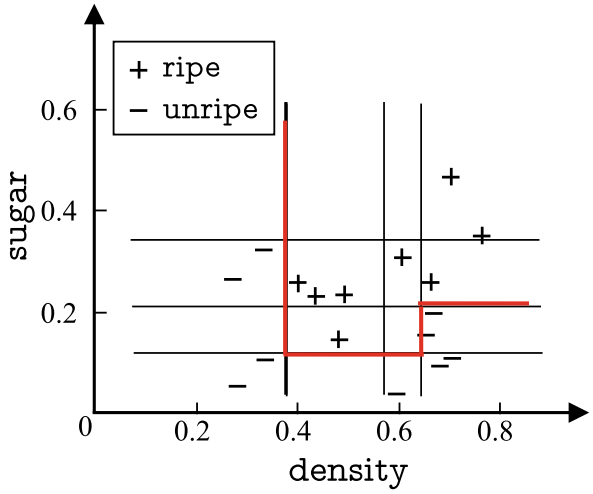
Autre illustration du boosting sur jeu de données



(a) 3 base learners.



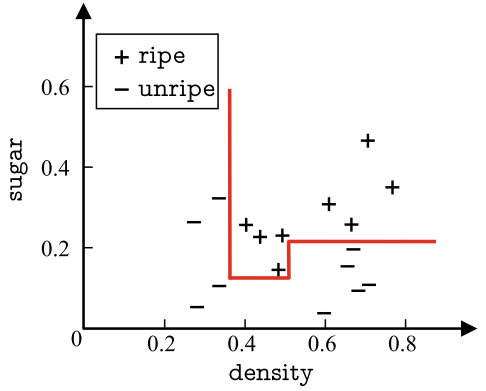
(b) 5 base learners.



(c) 11 base learners.

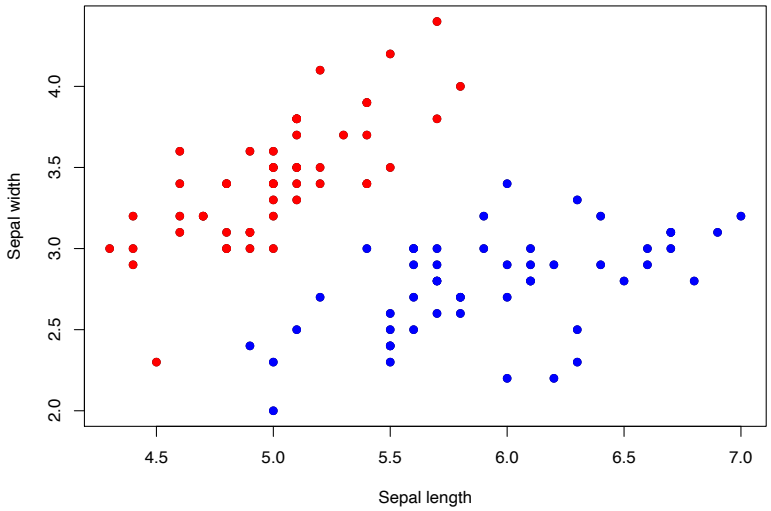
Here "**base learner**" = decision stump

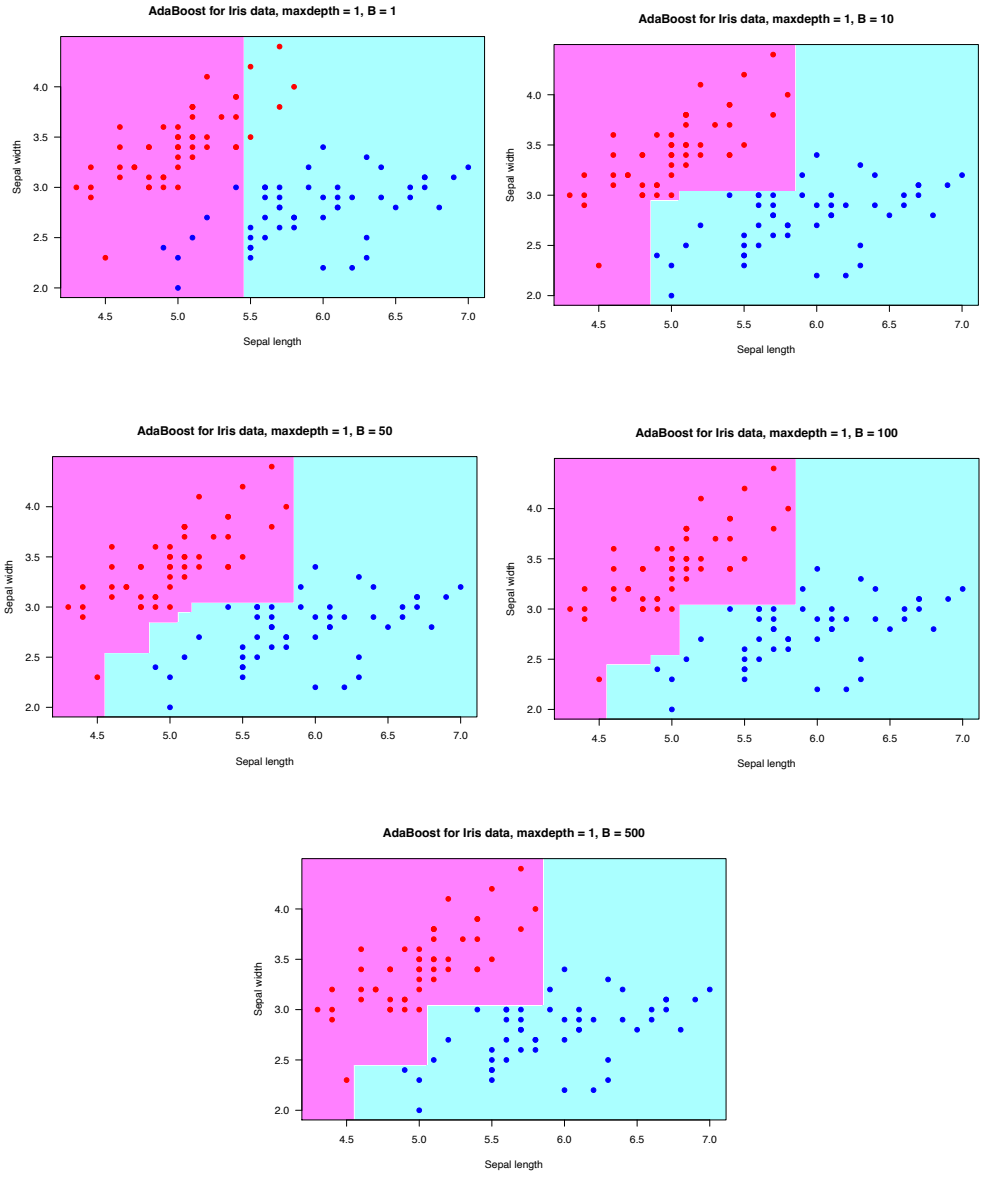
Arbre de décisions



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

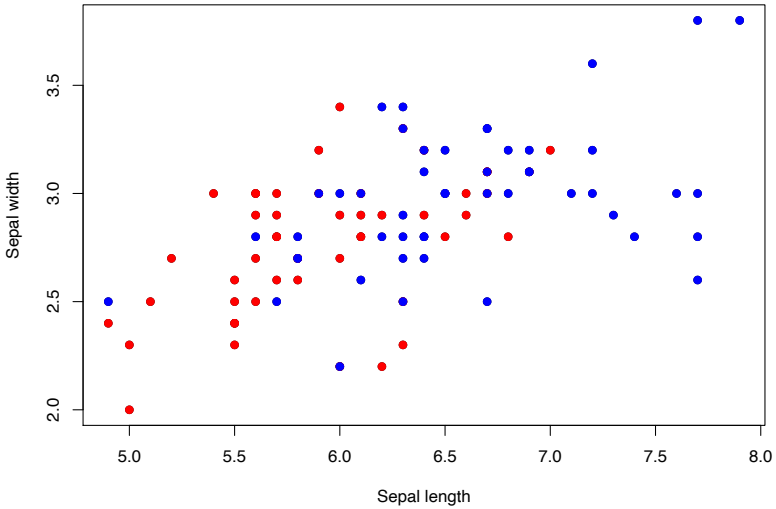
Boosting sur jeu de données « Iris » (Setosa vs. Versicolor)

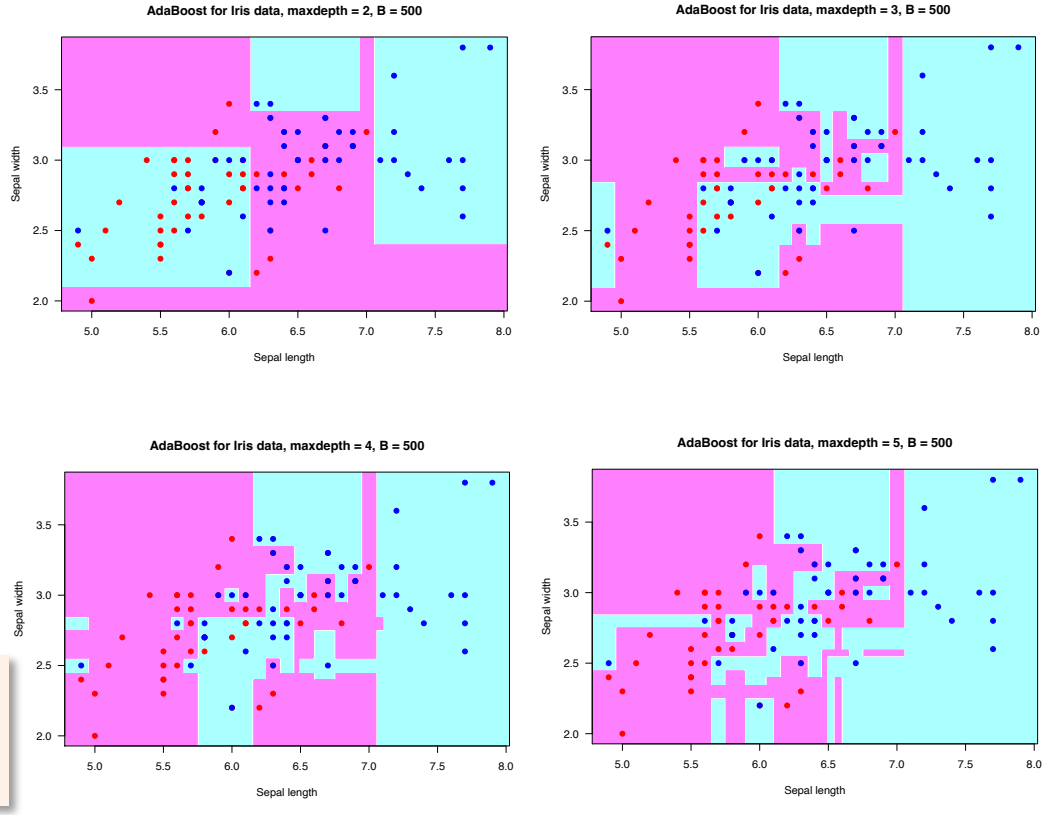




Effet du nombre **d'itérations**

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





Effet de la **profondeur** des arbres

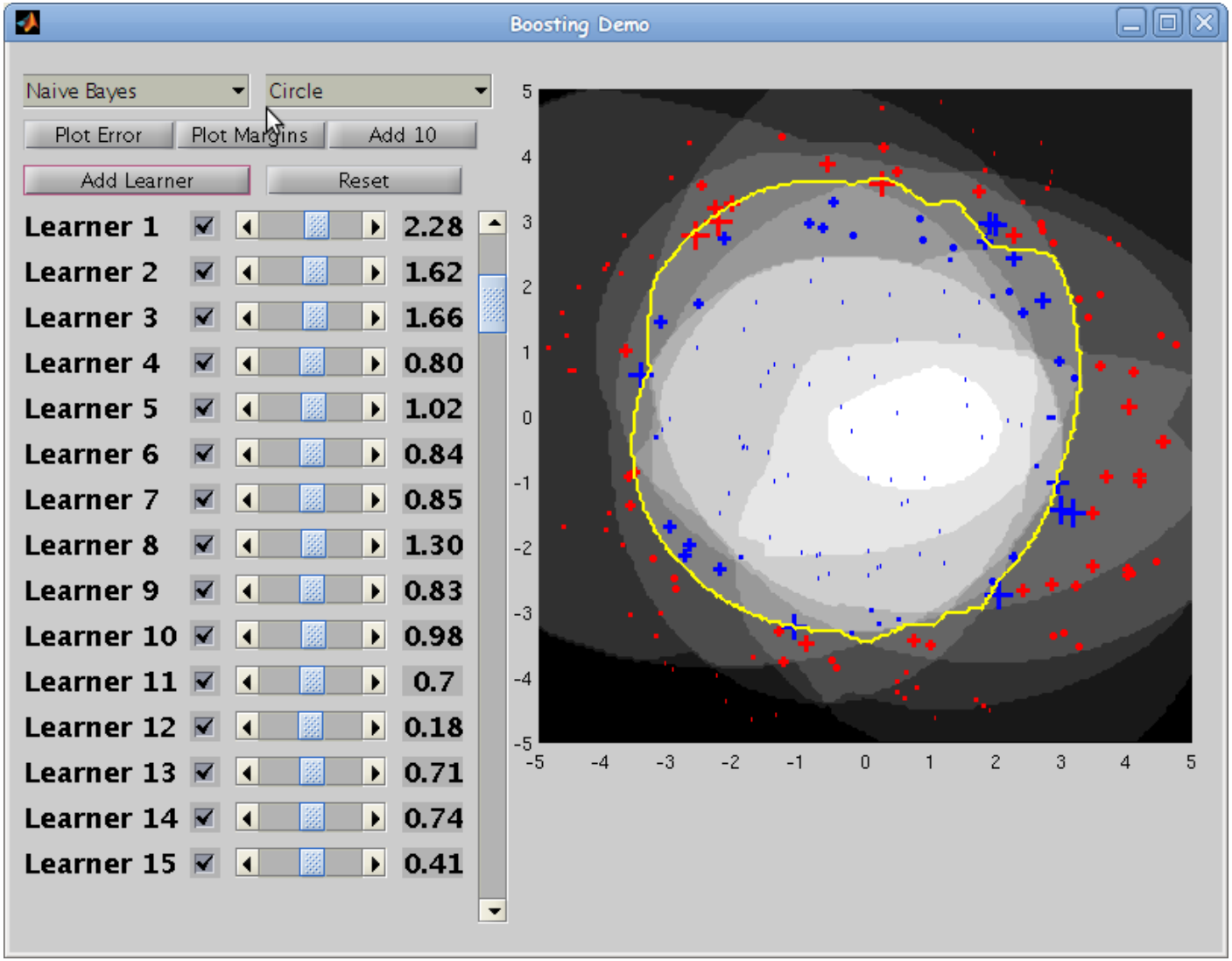
Exercice

Boosting : Des demos en ligne

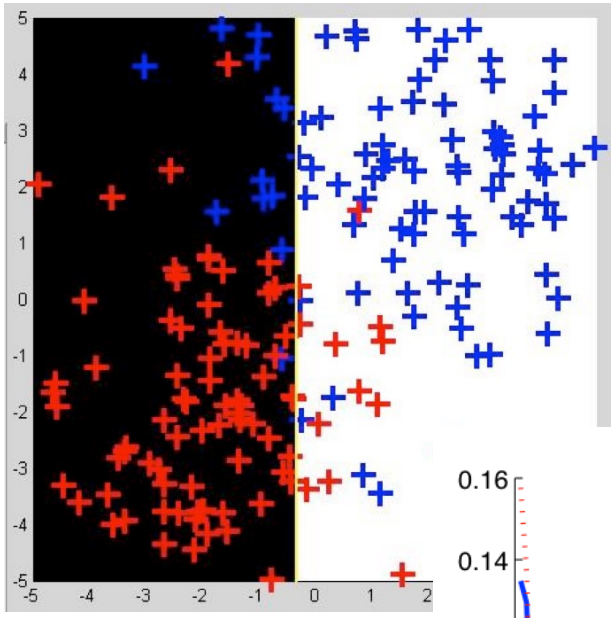
http://www.research.att.com/~yoav/adaboost/index.html

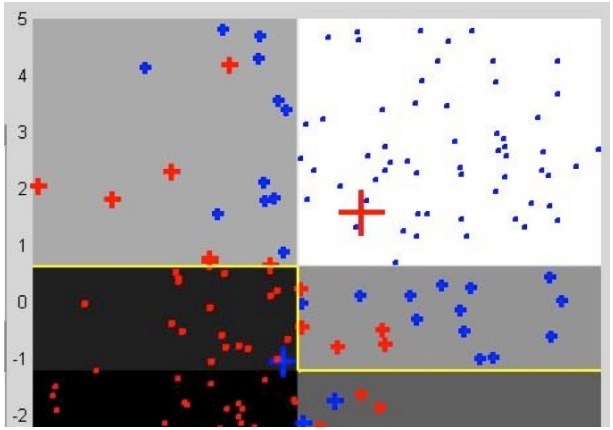
http://www.mathworks.com/matlabcentral/fx_files/29245/1/boosting_demo.png

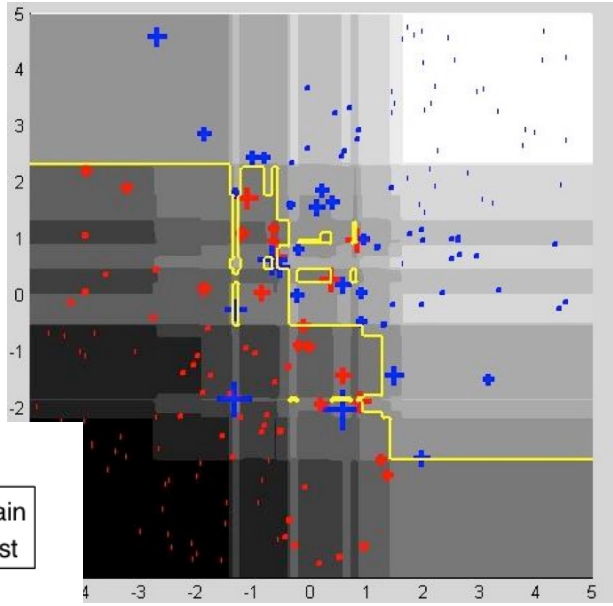
Boostingdemo de Richard Stapenhurst

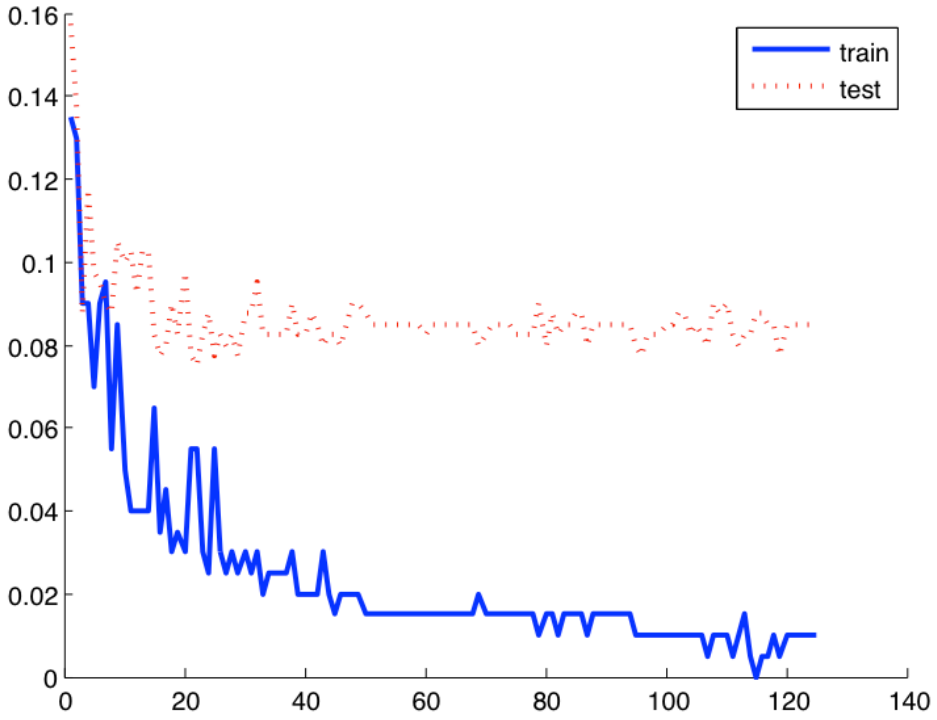


Boostingdemo de Richard Stapenhurst









Plan

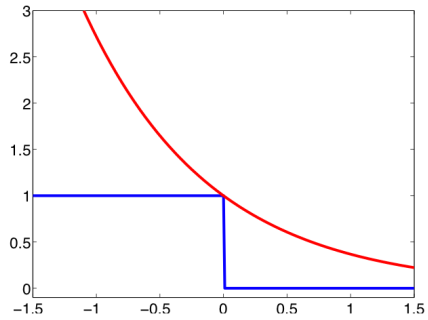
1. Méthodes d'ensemble
2. Le boosting
3. Le boosting : pourquoi ça marche
4. Le bagging
5. Les forêts aléatoires
6. XGBoost : le boosting d'arbres
7. Vers d'autre méthodes d'ensemble ?

Analyse théorique du boosting

Dérivation de l'algorithme du boosting

Dérivation de l'algorithme du boosting

- **Re-dérivation du boosting**
  - En choisissant une *fonction de perte surrogée* de forme exponentielle



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

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

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

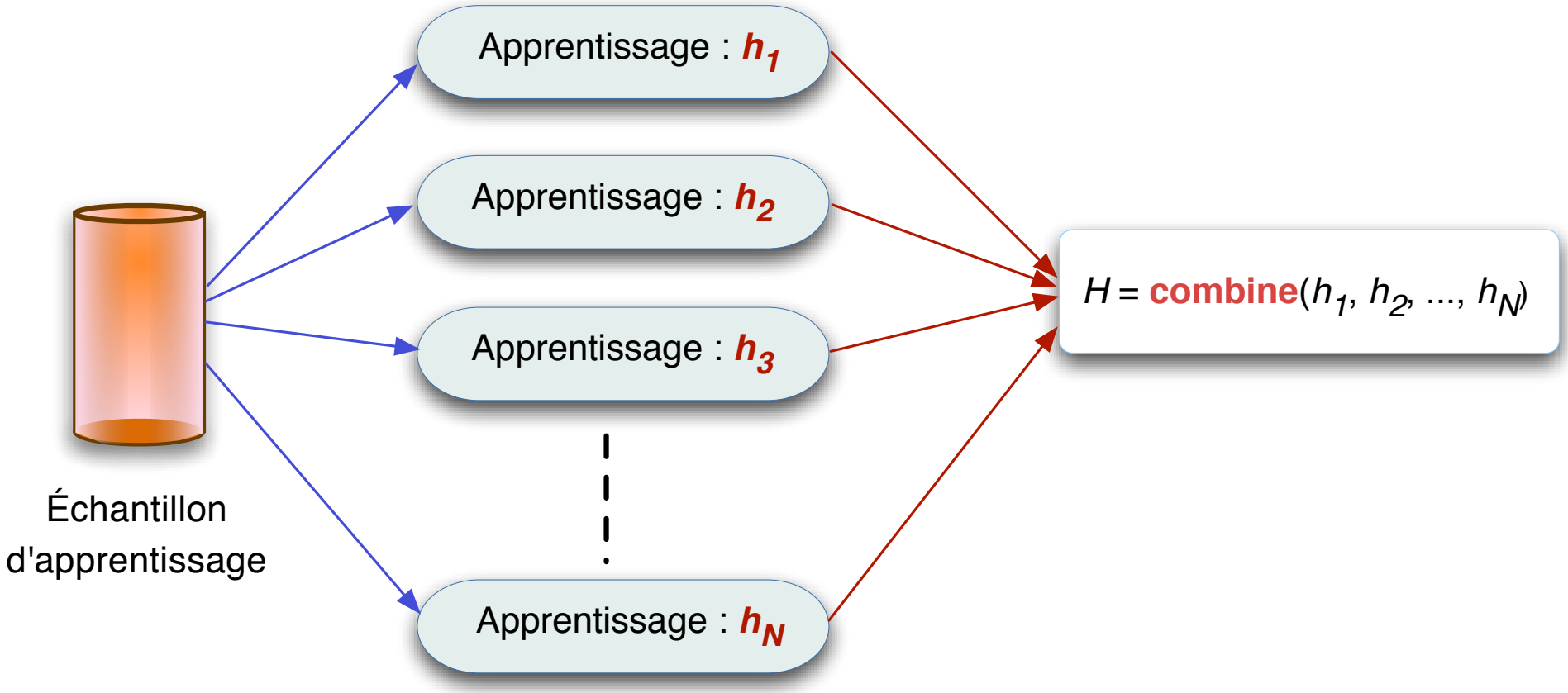
$$= \sum_{i=1}^{m} e^{-y_i\, H_{T-1}(\mathbf{x}_i)} \cdot e^{-\alpha_T\, y_i\, h_T(\mathbf{x}_i)}$$

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

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

$$\rightarrow \quad \alpha_T \;=\; \frac{1}{2} \log \frac{1-\varepsilon_T}{\varepsilon_T}$$

Schéma général : apprentissage



Bounds on training error

and

On generalization error

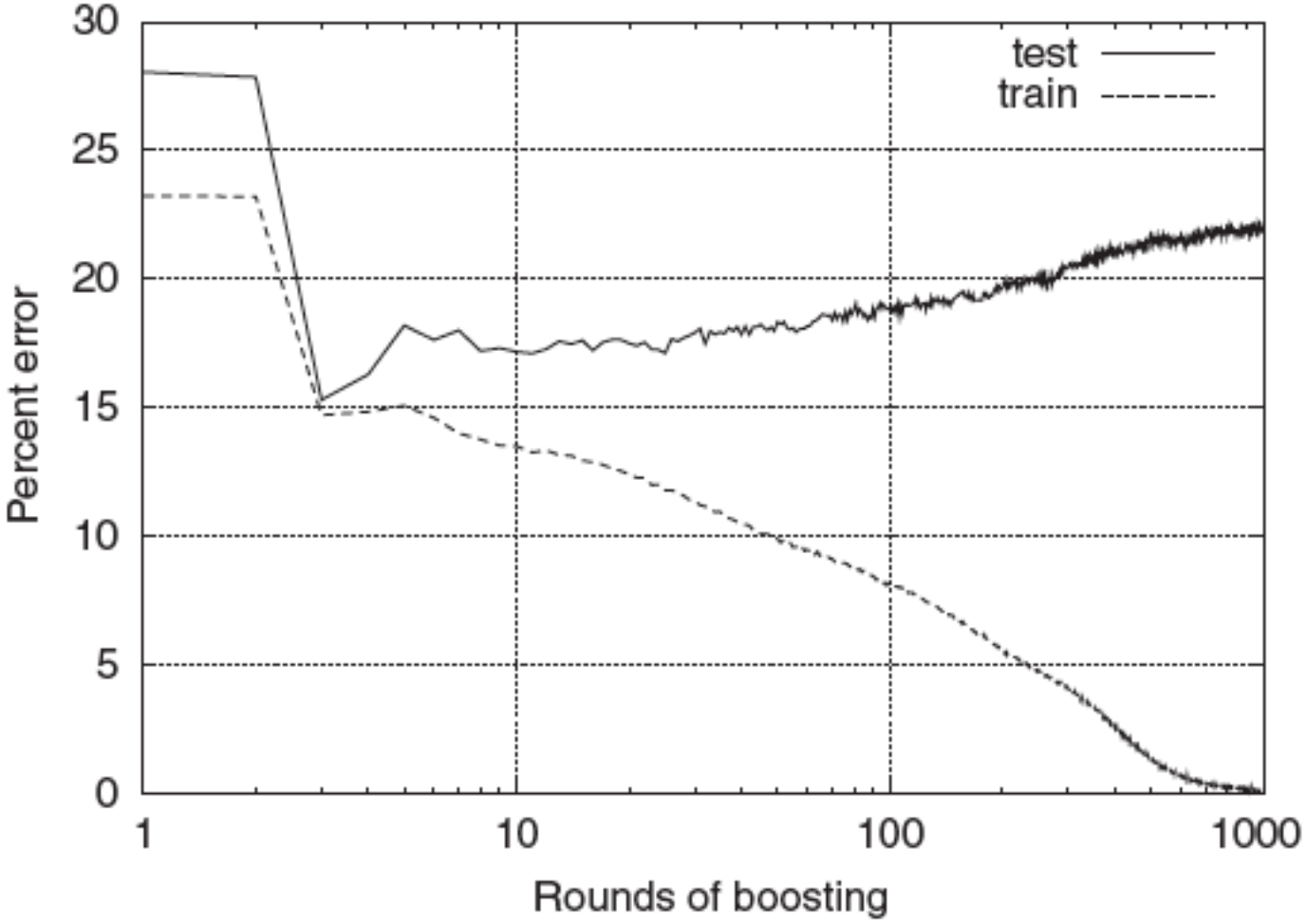
Bound on the training error

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

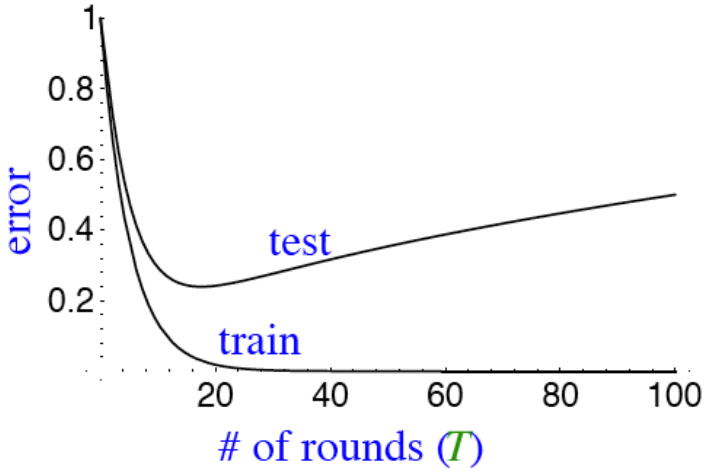
$$\begin{aligned} \text{training error}(H_{\text{final}}) &\leq \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)} \right] \\ &= \prod_t \sqrt{1-4\gamma_t^2} \\ &\leq \exp\left(-2 \sum_t \gamma_t^2 \right) \end{aligned}$$

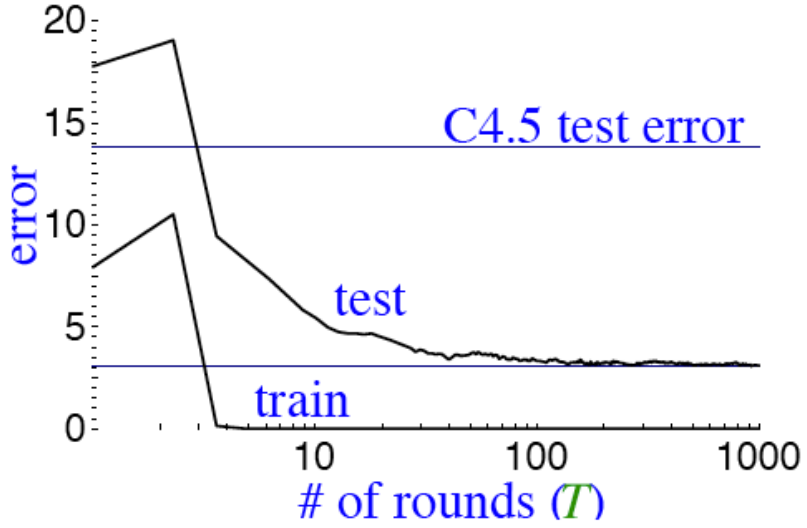
- so: if $\forall t : \ \gamma_t \geq \gamma > 0$
  then $\text{training error}(H_{\text{final}}) \leq e^{-2\gamma^2 T}$
- AdaBoost is adaptive:
  - does not need to know γ or T a priori
  - can exploit $\gamma_t \gg \gamma$

Evolution of the error curves (learning & test)



How to explain the evolution of the generalization error?





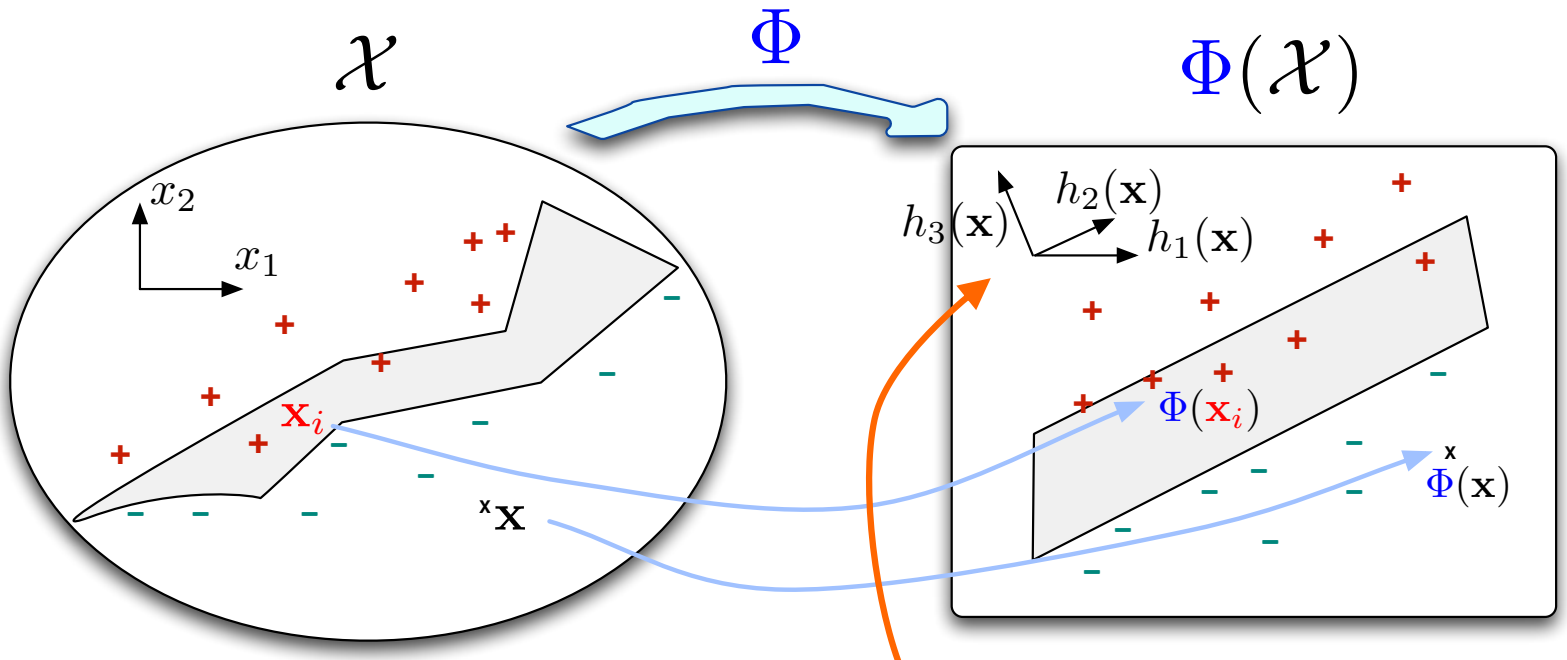
- The test error **does not increase**, even after 1000 steps (2.10^6 test nodes !!)
  - Boosting C4.5 on the « letter » dataset

Arguments pour **expliquer**

les propriétés du boosting

(the unreasonable power of boosting)

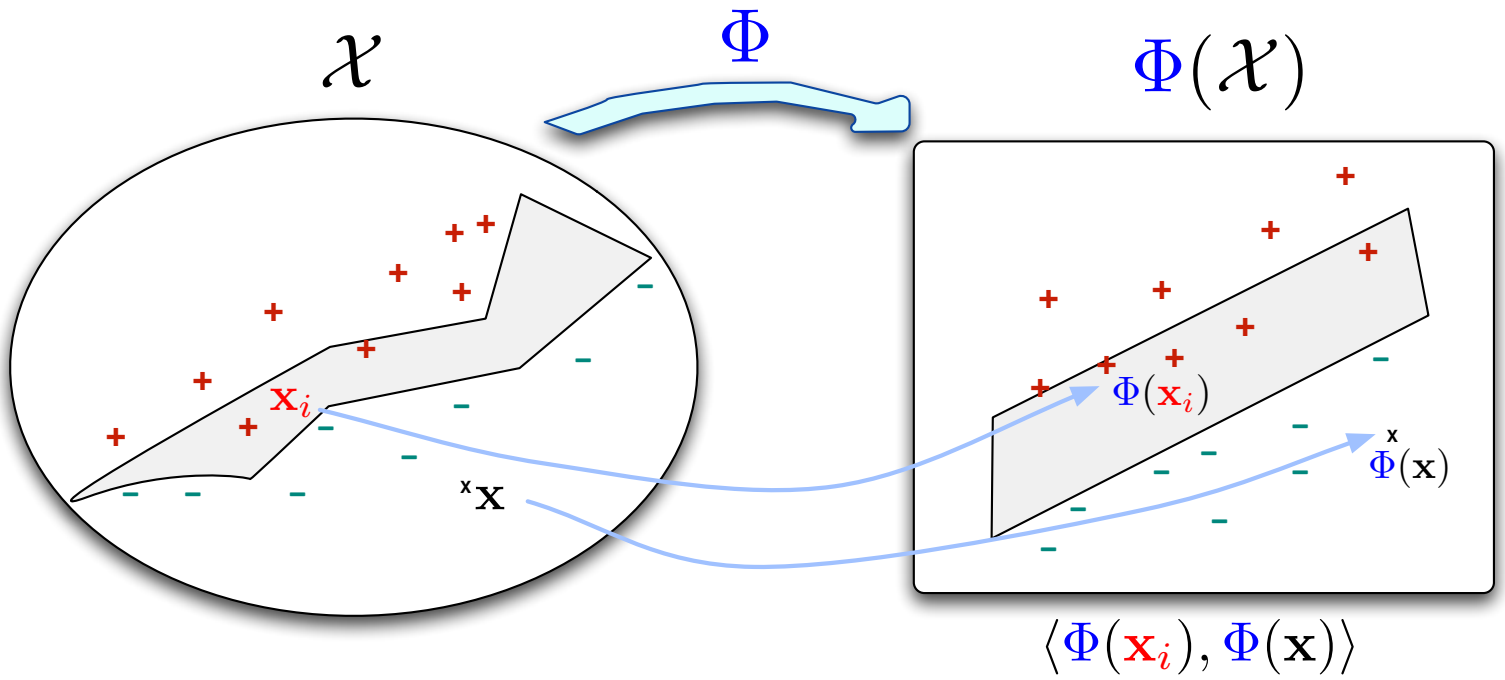
*Boosting and **redescription***



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

- **Iterative** construction of the redescription space

SVM and kernel methods



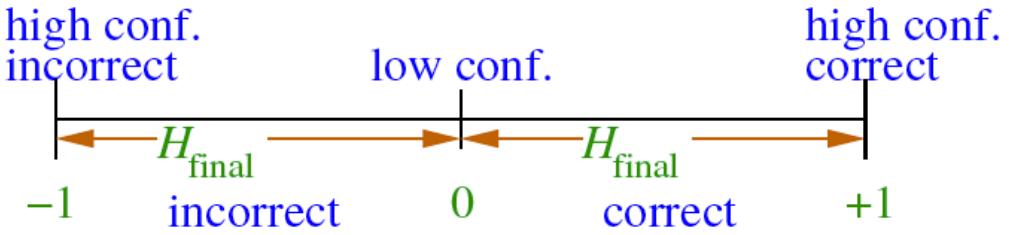
$$h^*(\mathbf{x}) \;=\; \text{sign}\left\{ \sum_{i \in \mathcal{P}_S} \alpha_i^\star \; y_i \; \kappa(\mathbf{x}_i \,,\, \mathbf{x}) \;+\; w_0^\star \right\}$$

Une explication par la « marge »

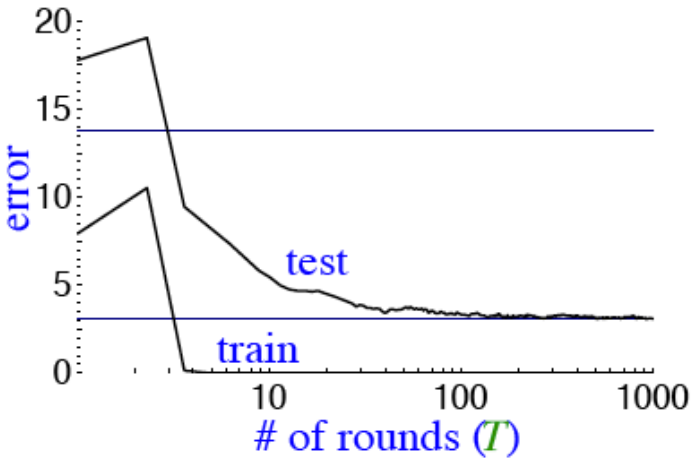
- Idée :
  - L'erreur en apprentissage ne mesure que le taux d'erreur en prédiction
  - Il faut aussi tenir compte de **la confiance** de la prédiction
  - On peut estimer cette confiance par la **marge**

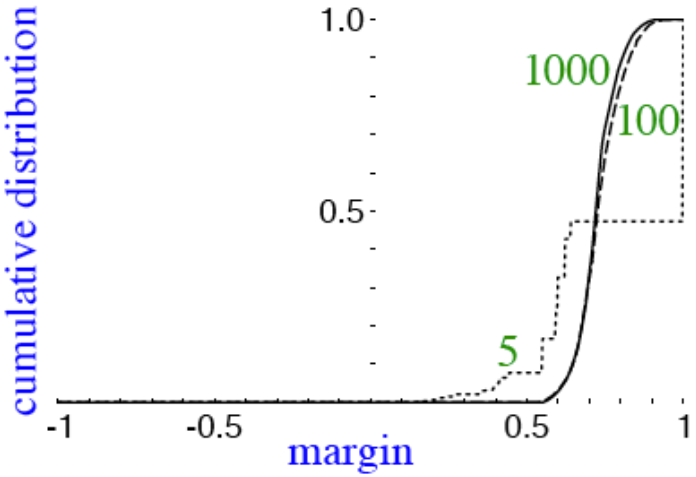
    = poids des classifieurs ayant voté correctement

    - poids des classifieurs en erreur



Étude de la distribution des marges des $\boldsymbol{x}_i$





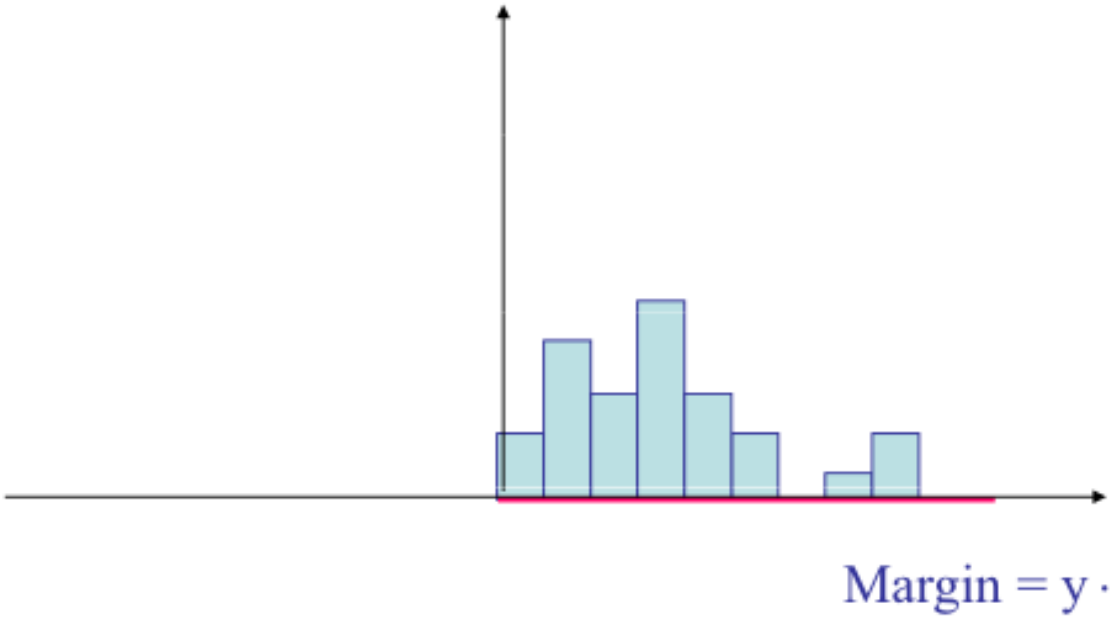
	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

Argument de la maximisation de la marge

- À chaque étape, AdaBoost placerait le **plus de poids sur les exemples x_i de plus faible marge** tout en continuant à améliorer la marge sur les autres

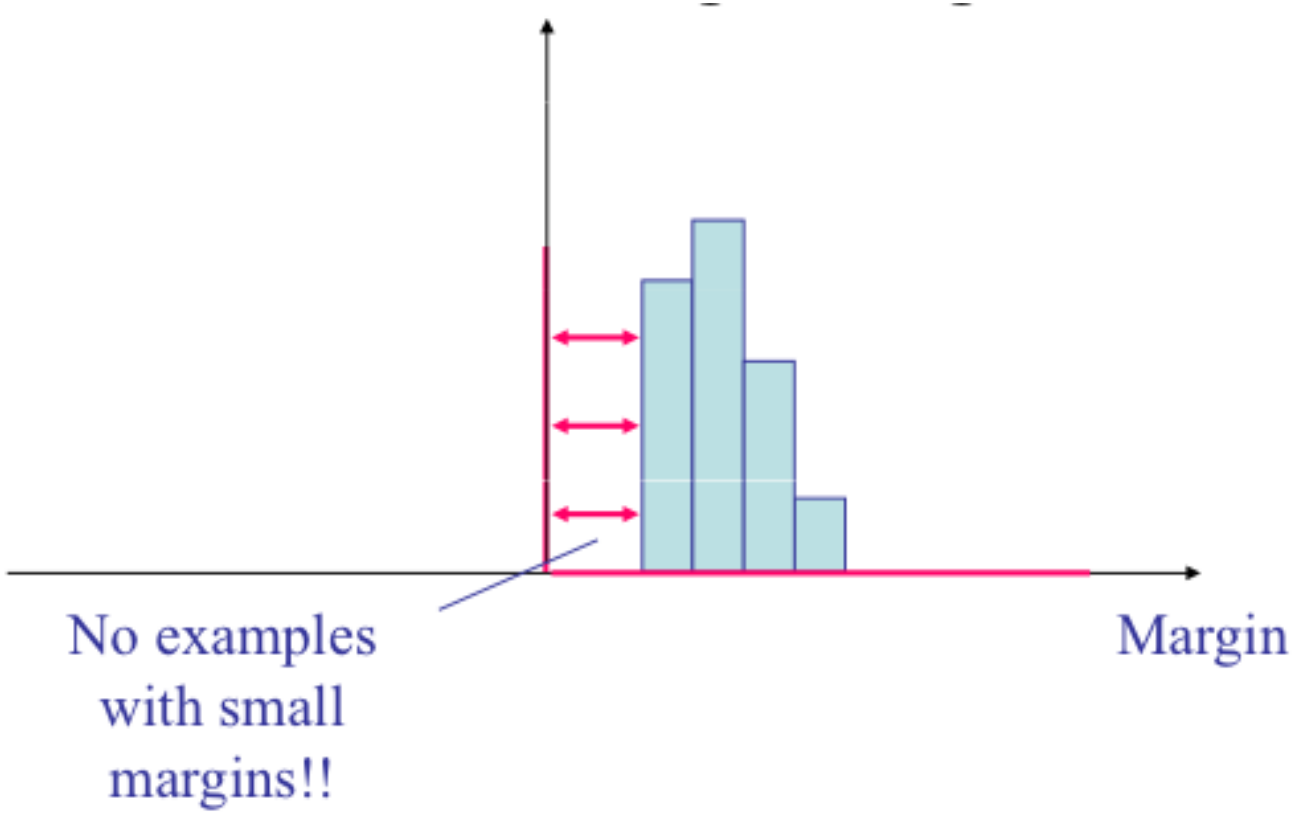
- L'hypothèse finale serait **complexe** mais **proche d'une hypothèse simple** d'erreur d'apprentissage proche (et donc d'erreur en généralisation faible aussi)

Analyse théorique : maximisation de la marge



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

Analyse théorique : maximisation de la marge



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

The intuitive argument

[Shapire & Freund, "*Boosting. Foundations and Algorithms*", MIT Press, 2012], p.97

- When a vote is organized among a very large number of voters, it is often possible to forecast the result of the vote by **randomly polling a tiny subset** of the electorate.

- The **larger the margin** between the candidates, the **smaller the poll** can be.

Well, a boosting ensemble **is not** a random selection of classifiers …

→ The theorems will apply to any selection of classifiers

Bornes en généralisation

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

Plus techniquement

- 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

Mais ...

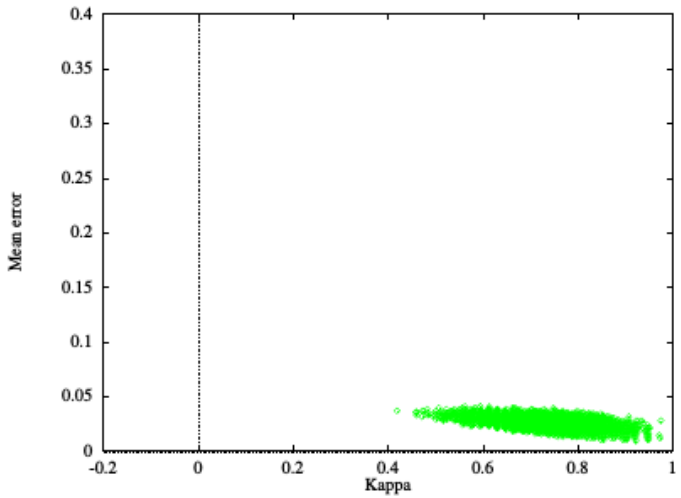
AdaBoost fait grandir les marges... sans les maximiser.

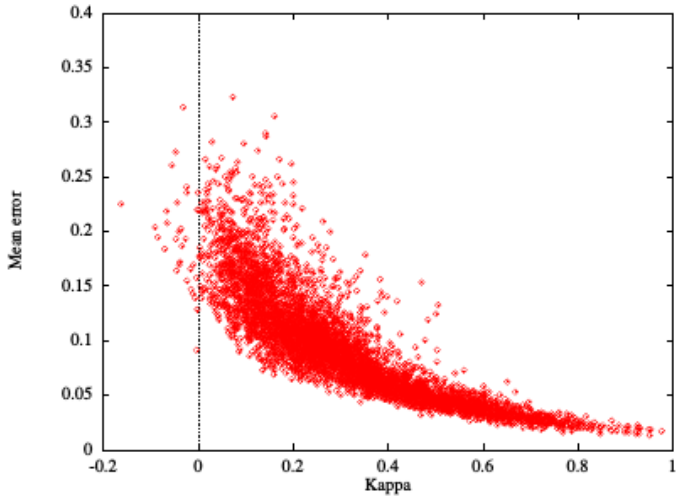
Travaux poussant l'idée plus loin :

- maximiser les marges [Rätsch and Warmuth, 2003],
- mettre toutes les marges à 1 [Harries, 1999].

Ces méthodes font moins bien que AdaBoost...
la maximisation des marges n'est pas la réponse.

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],

...

Autres perspectives pour **expliquer**

les propriétés du boosting

Autres perspectives

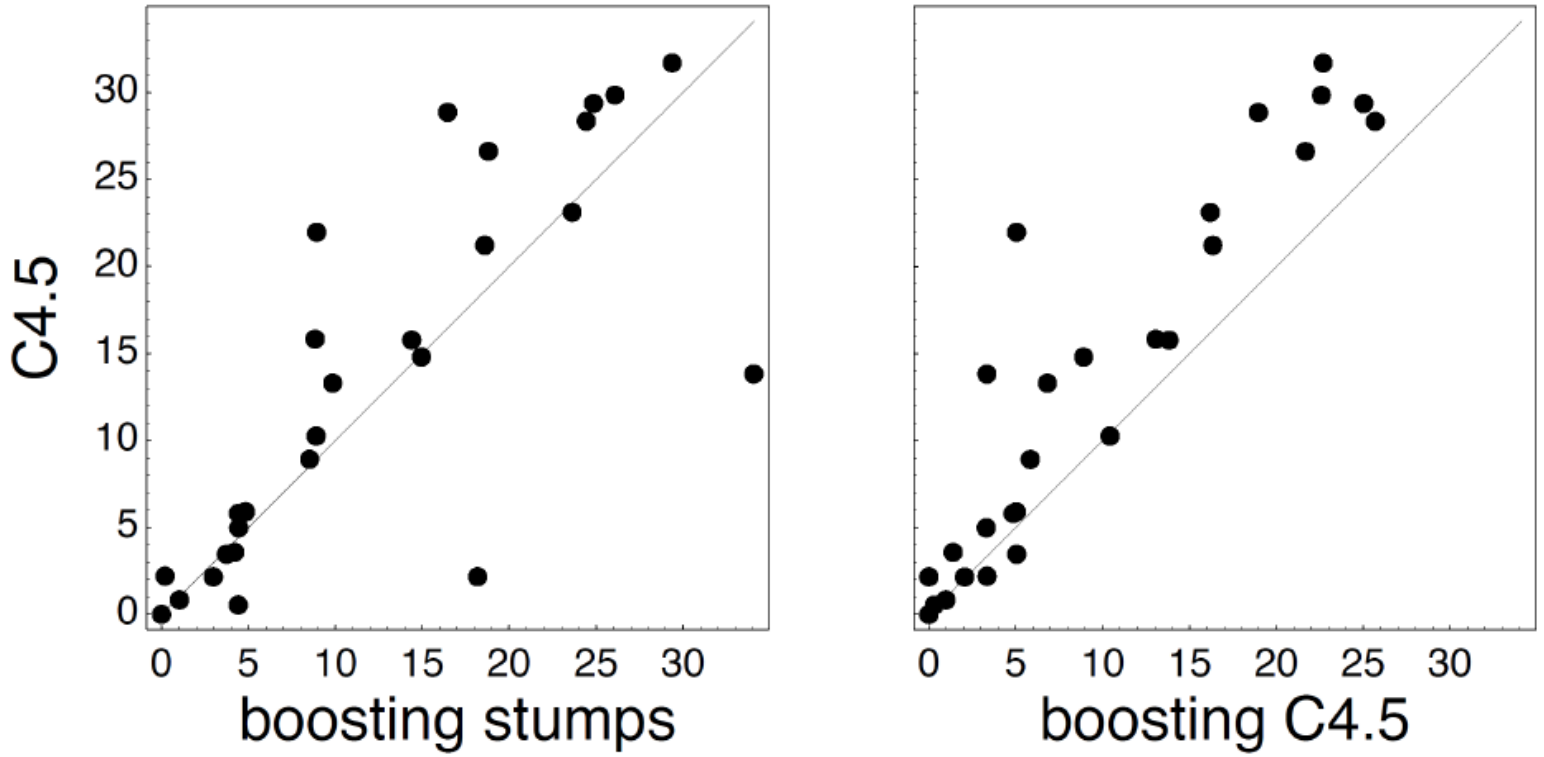
- La théorie des jeux
  - Inspiration originale pour développer le boosting
- Minimisation de la perte (surrogée)
  - Vue plus haut (et plus loin)
- Point de vue géométrie de l'information

Bilan

Avantages pratiques de AdaBoost

- (très) **rapide**
- **simple** + facile à programmer
- **Un seul paramètre** à régler : le nombre d'étapes de boosting (T)
- Applicable à de nombreux domaines par un bon choix de classifieur faible (neuro net, C4.5, …)
- **Pas de sur-spécialisation** par la maximisation des marges
- Peut être adapté au cas où $h_t : \mathcal{X} \rightarrow \mathcal{R}$; la classe est définie par le signe de $h_t(x)$; la confiance est donnée par $| h_t(x) |$
- Peut être adapté aux problèmes multi-classes où $y_i \in \{1,..,c\}$ et aux problèmes multi-étiquettes
- Permet de trouver les exemples aberrants (outliers)

Performances du boosting



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

Quand est-ce que ça ne marche pas

- Rappel : **No-free-lunch-theorem**

- Boosting inadapté quand
  - **Pas assez de données**
  - **Comité** (d'apprenants faibles) **trop restreint**
  - Apprenants faibles **trop stables** (quoique … argument surtout valable pour le bagging)
  - **Apprenants faibles trop forts !**
    - Peuvent faire du surapprentissage par eux-mêmes
  - Données **bruitées**

Aspects pratiques

Avantages	***Difficultés***
• Un **meta-algorithme** d'apprentissage : utiliser n'importe quel algorithme d'apprentissage faible • En principe, **un seul paramètre** à régler (le nombre T d'itérations) • **Facile et aisé à programmer** • **Performances théoriques garanties**	• Difficile d'incorporer des connaissances a priori • Difficile de savoir comment régulariser • Le meilleur choix d'un apprenti faible n'est pas évident • Les frontières de décision en utilisant des méthodes parallèles aux axes est souvent très irrégulière (non interprétable)

Boosting : résumé

- La prédiction finale est issue d'une combinaison (vote pondéré) de plusieurs prédictions
- Méthode :
  - Itérative
  - Chaque classifieur dépend des précédents

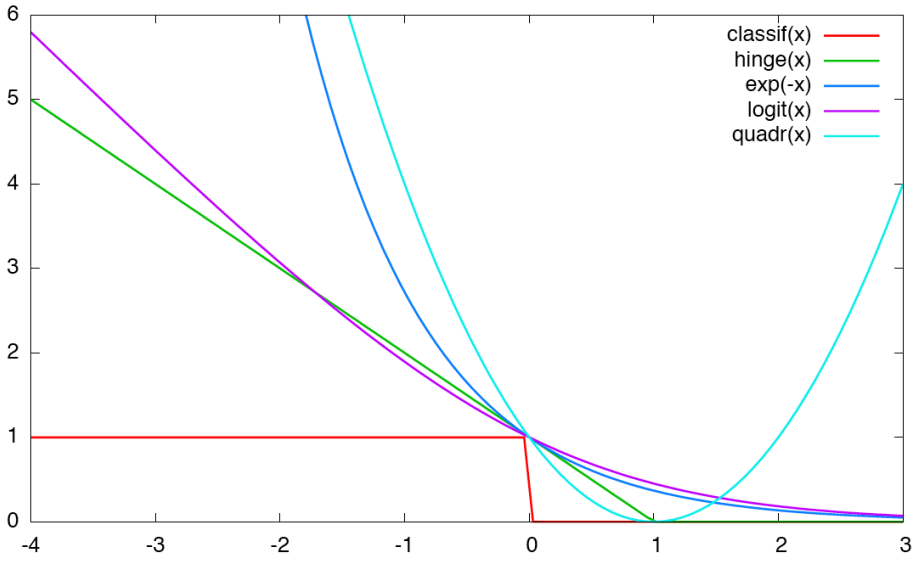
    (les classifieurs ne sont donc pas indépendants comme dans d'autres méthodes de vote)
  - Les exemples sont pondérés différemment
  - Le poids des exemples reflète la difficulté des classifieurs précédents à les apprendre

Autres algorithmes de boosting

Fonctions « surrogées »

Fonctions de la forme $\ell(x, y) = \varphi(-xy)$:

Nom	$\varphi(x)$
Exponentiel	$\exp(-x)$
Logit	$\log_2(1 + \exp(-x))$
Quadratique	$(1 - x)^2$
Quadratique tronqué	$(1 - x)_+^2$
Hinge (SVM)	$(1 - x)_+$



LogitBoost

- Utilisation de la fonction de perte logloss
  - Pas de poids exorbitant sur les points aberrants
  - Possibilité de calculer des probabilités associées aux étiquettes

$$p(y|\mathbf{x}) = \frac{e^{h(\mathbf{x})}}{e^{-h(\mathbf{x})} + e^{h(\mathbf{x})}} = \frac{1}{1 + e^{-2h(\mathbf{x})}}$$

Applications du boosting

Applications

Text classification	Schapire and Singer - Used stumps with normalized term frequency and multi-class encoding
OCR	Schwenk and Bengio (neural networks)
Natural language Processing	Collins; Haruno, Shirai and Ooyama
Image retrieval	Thieu and Viola
Medical diagnosis	Merle *et al.*
Fraud Detection	Rätsch & Müller 2001
Drug Discovery	Rätsch, Demiriz, Bennett 2002
Elect. Power Monitoring	Onoda, Rätsch & Müller 2000

Fuller list: Schapire's 2002, Meir & Rätsch 2003 review

Face detection idea

- 1) in Adaboost use parallel-axis (tree decision) classifier

  2) in Viola Jones, the weak classifier is the specially designed classifier described in the paper.

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

- **Images** 384 x 288 (niveaux de gris)
- Détecter des **visages** à toute échelle
- En **temps réel** (15 images / s) sur un smartphone !!

- Problèmes
  - Identifier des **descripteurs pertinents**
  - Les calculer **rapidement**
  - Les **organiser** pour faire un système performant
    - Faible FN
    - Faible FP

Les descripteurs

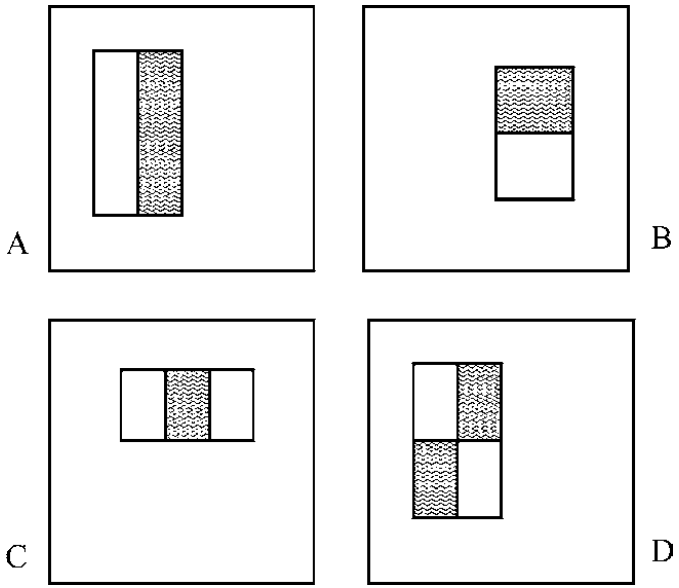
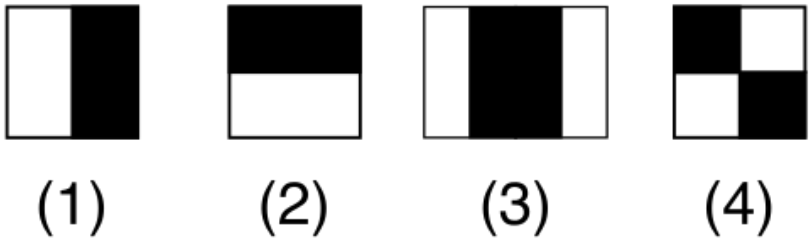
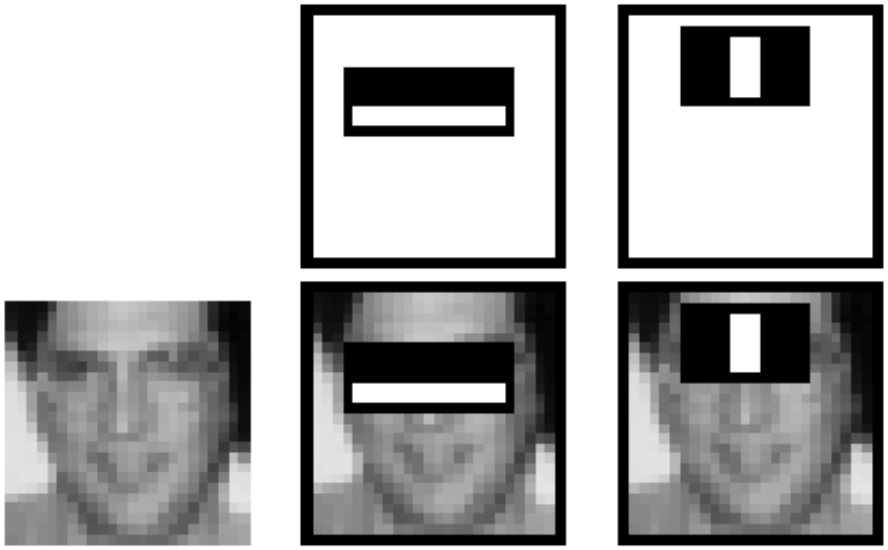


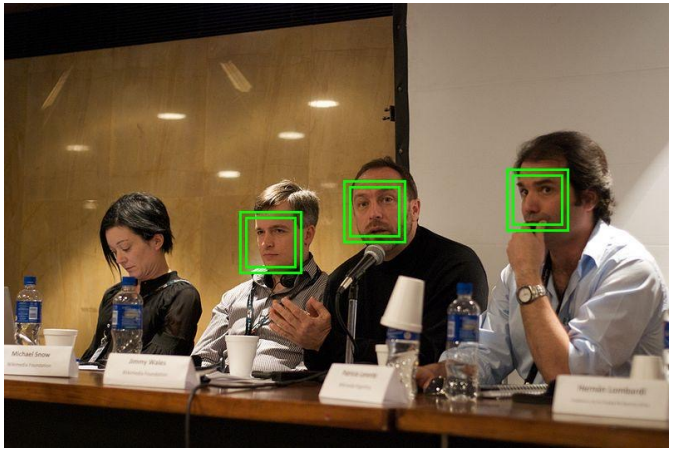
Figure 1. Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature,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

- Plus de 3 000 000 000 !
  - Toutes échelles
  - Seuil à définir

Useful Features Learned by Boosting







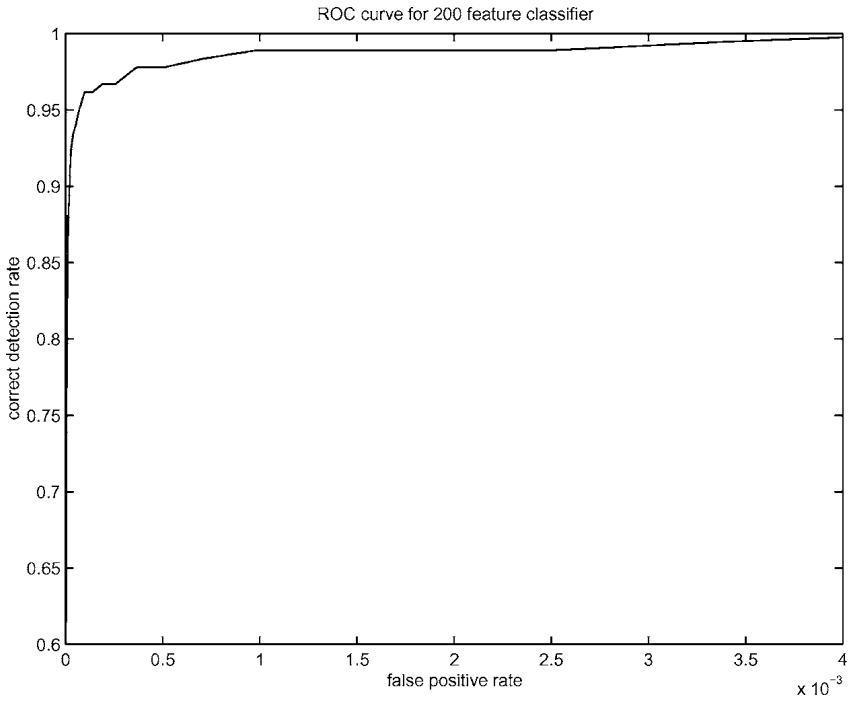
Adaboost v.2c

Example of the training set

Positive instances

Sélection des descripteurs utiles

- En utilisant AdaBoost
  - Les descripteurs sont associés à des « decision stumps »
  - Le boosting sélectionne une séquence de descripteurs
    - 200 dans cette étude



Mais trop coûteux

en temps calcul : ~0.7*s*

Organisation des détecteurs en cascade

- Éliminer le plus tôt possible les négatifs

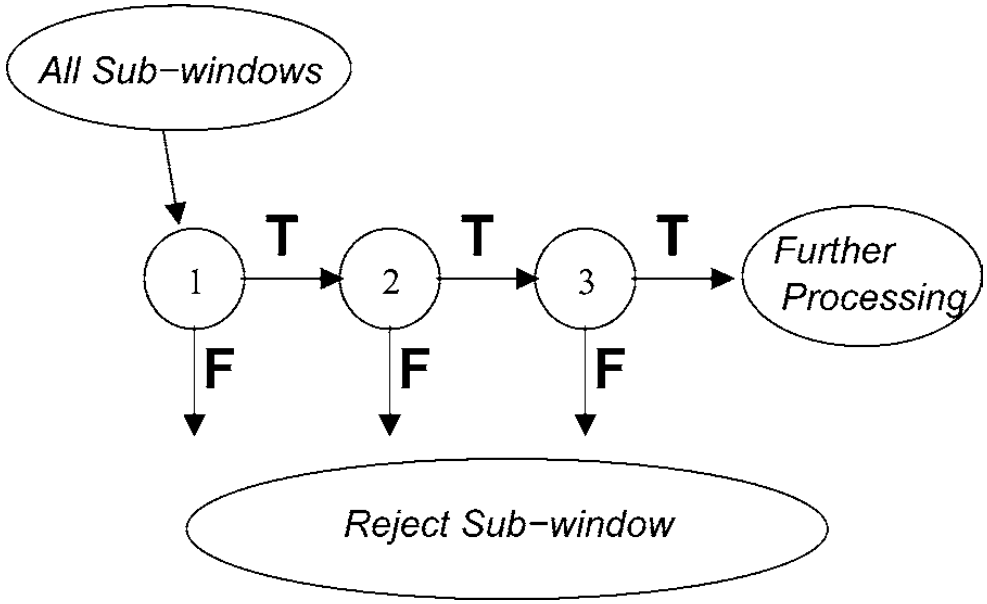
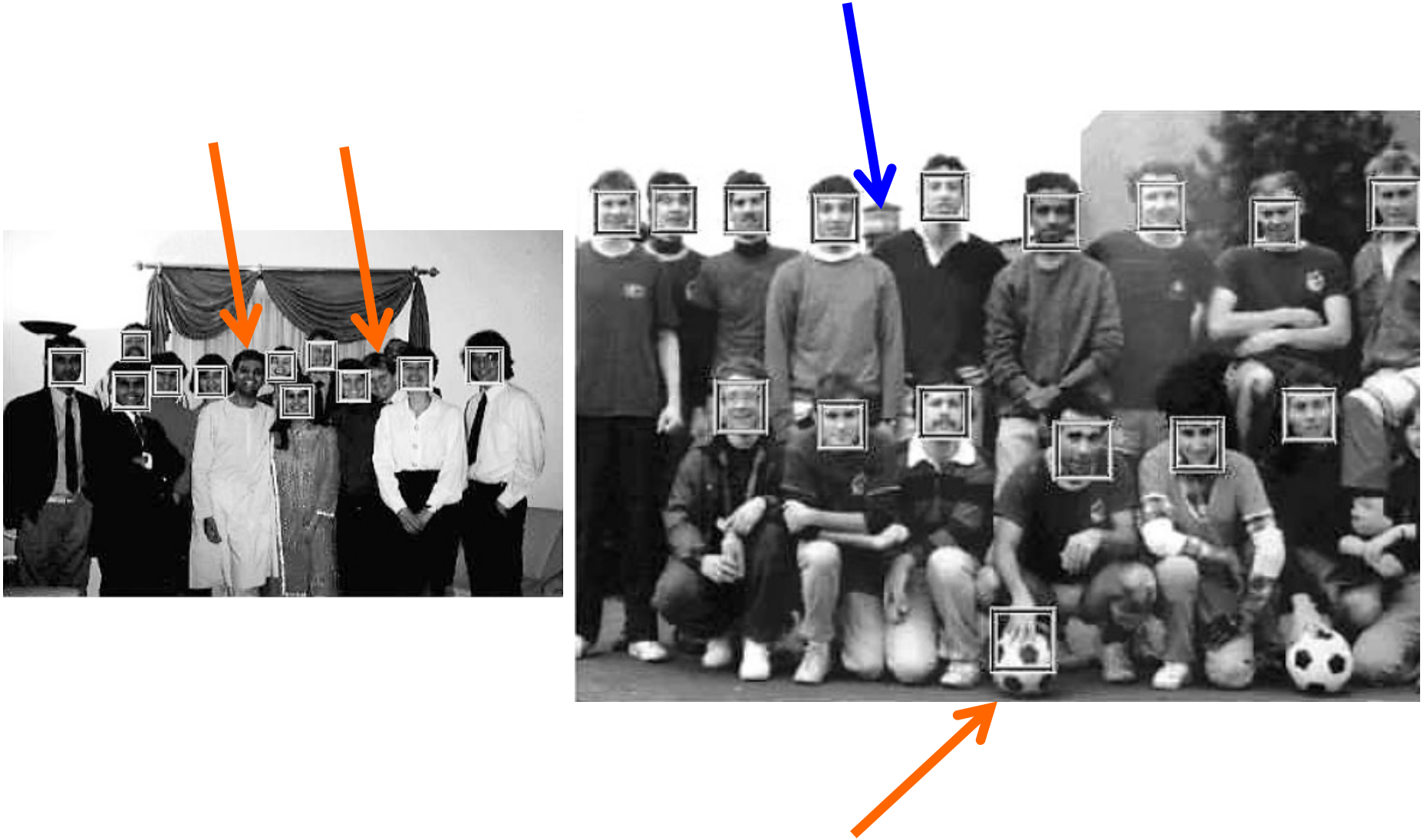


Figure 6. Schematic depiction of a the detection cascade. A series of classifiers are applied to every sub-window. The initial classifier eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation. After 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



Applications

- Reconnaissance du contour du rein

Plan

1. Méthodes d'ensemble
2. Le boosting
3. Le boosting : pourquoi ça marche
4. Le bagging
5. Les forêts aléatoires (random forests)
6. XGBoost : le boosting d'arbres
7. Vers d'autre méthodes d'ensemble ?

Bagging = Bootstrapping aggregation

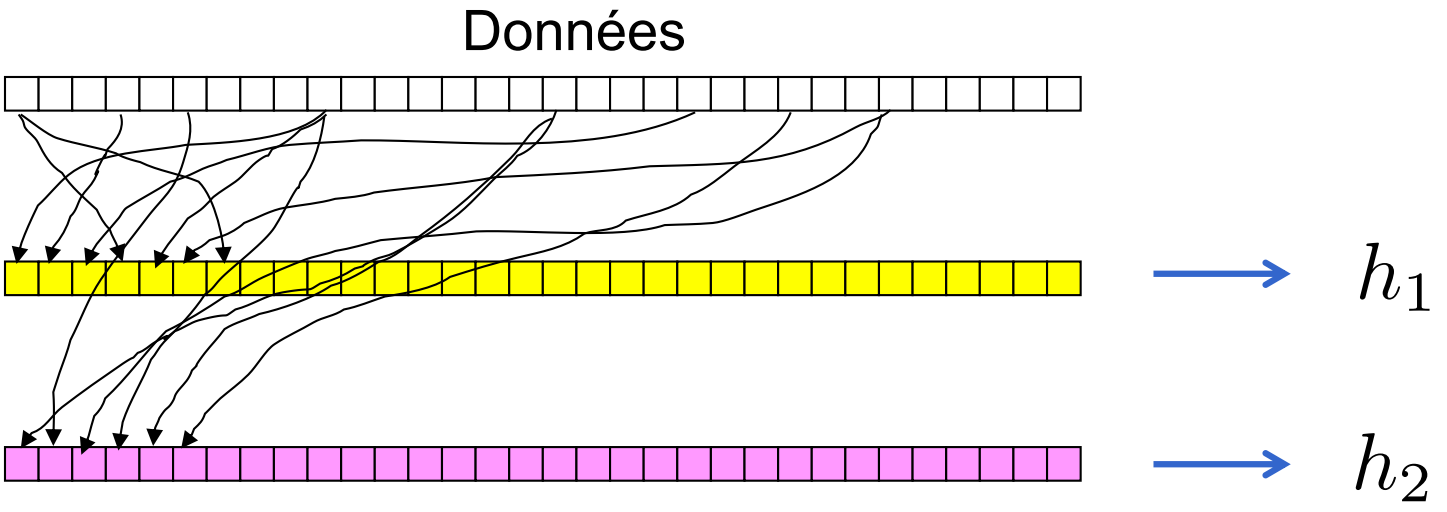
Bagging *[Breiman,96]*

- **Génération de *k* échantillons « indépendants »** par tirage avec remise dans l'échantillon S_m

- **Pour chaque échantillon**, apprentissage d'un classifieur en utilisant le même algorithme d'apprentissage

- La **prédiction finale** pour un nouvel exemple est obtenue par vote (simple) des classifieurs

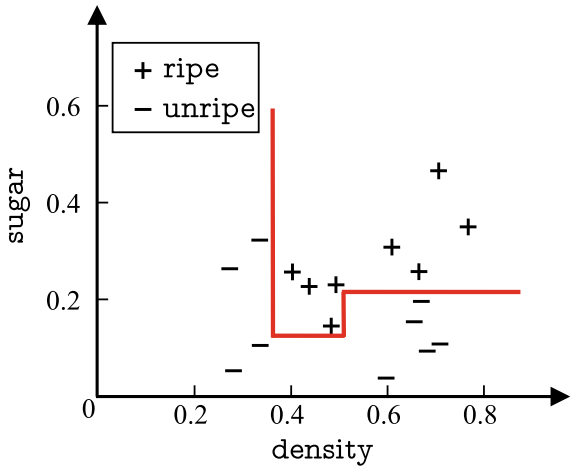
Le Bagging



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

Autre illustration du bagging sur jeu de données

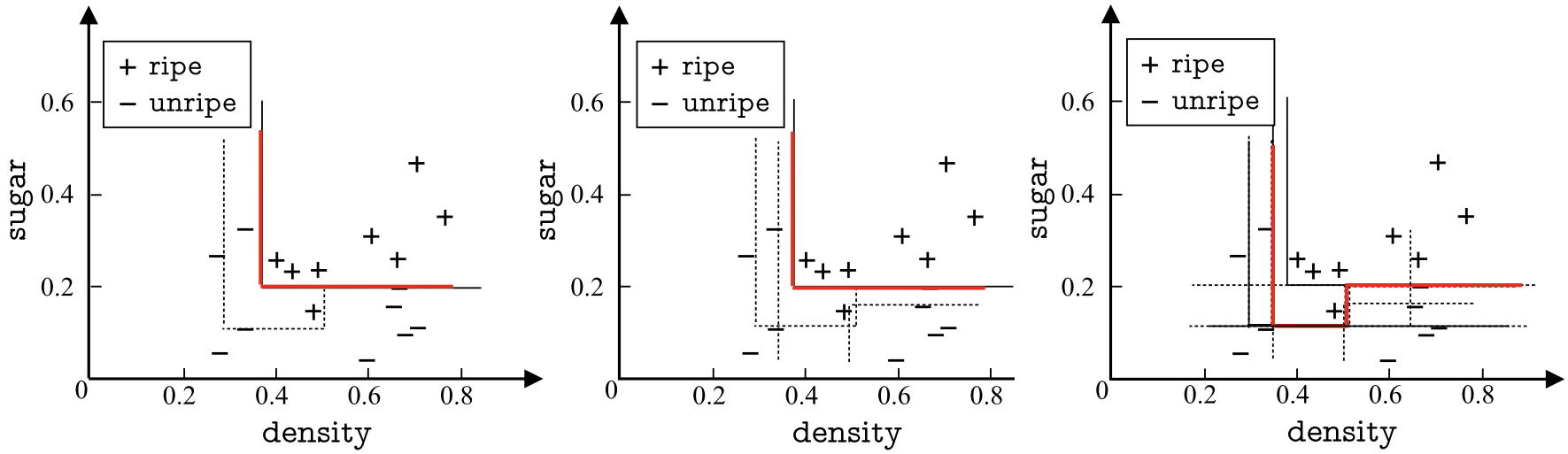
ID	density	sugar	ripe
1	0.697	0.460	true
2	0.774	0.376	true
3	0.634	0.264	true
4	0.608	0.318	true
5	0.556	0.215	true
6	0.403	0.237	true
7	0.481	0.149	true
8	0.437	0.211	true
9	0.666	0.091	false
10	0.243	0.267	false
11	0.245	0.057	false
12	0.343	0.099	false
13	0.639	0.161	false
14	0.657	0.198	false
15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



Apprentissage par

arbre de décisions

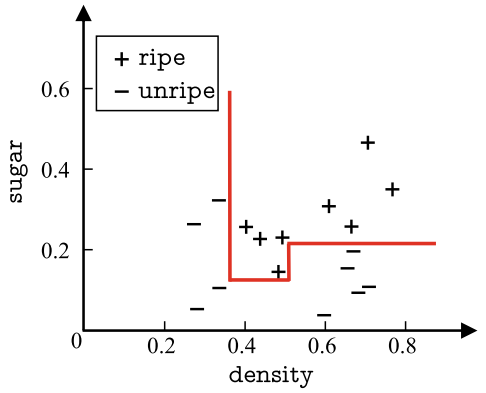
Autre illustration du bagging sur jeu de données



(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]

Bagging (suite)

- Il est souvent dit que :
  - Le bagging fonctionne en réduisant la variance
    en laissant le biais inchangé

- Mais, encore incomplètement compris
  - Voir [Yves Grandvalet : « Bagging equalizes influence »,
    Machine Learning , 55(3), pages 251-270, 2004.]

Plan

1. Méthodes d'ensemble
2. Le boosting
3. Le boosting : pourquoi ça marche
4. Le bagging
5. Les forêts aléatoires (random forests)
6. XGBoost : le boosting d'arbres
7. Vers d'autre méthodes d'ensemble ?

Les forêts aléatoires

(*Random forests*)

Du bagging aux forêts aléatoires

Il conserve le **biais** en diminuant la **variance**

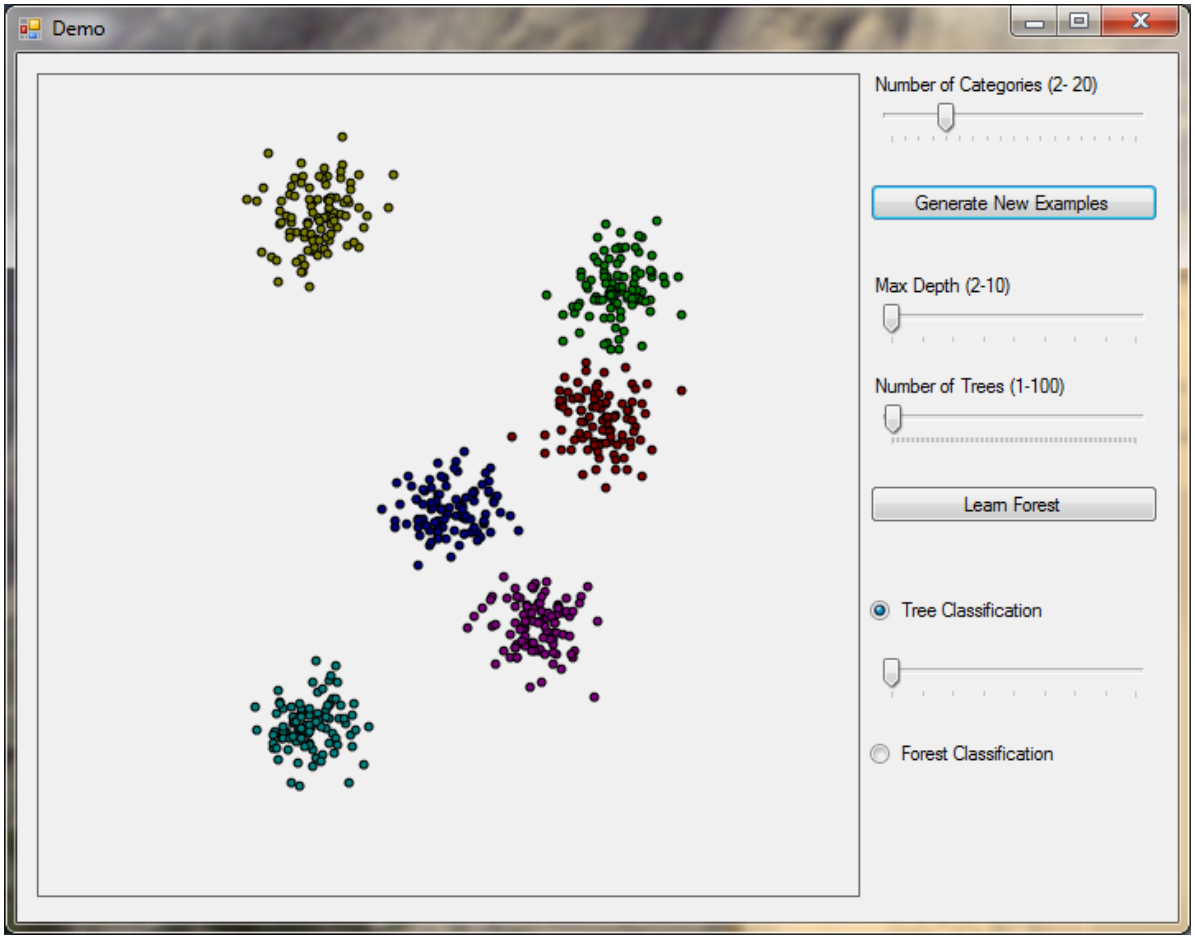
- Le **bagging** est favorisé par :
  - Des hypothèses faibles **différentes**
  - Des hypothèses faibles de **faible biais** (e.g. arbres de décision profonds)
  - On peut avoir un **nombre d'itérations élevé** sans sur-apprentissage (par augmentation de la marge)
- D'où les **forêts aléatoires**
  - On va s'arranger pour avoir des arbres de décisions
    - Profonds
    - Différents
    - Nombreux

Les forêts aléatoires

- Principe :
  - **réduire la corrélation entre apprenants faibles**

- **Apprendre** des arbres
  - Sur des **jeux de données différents** (comme le bagging)
  - Sur des **sous-ensembles d'attributs** tirés aléatoirement

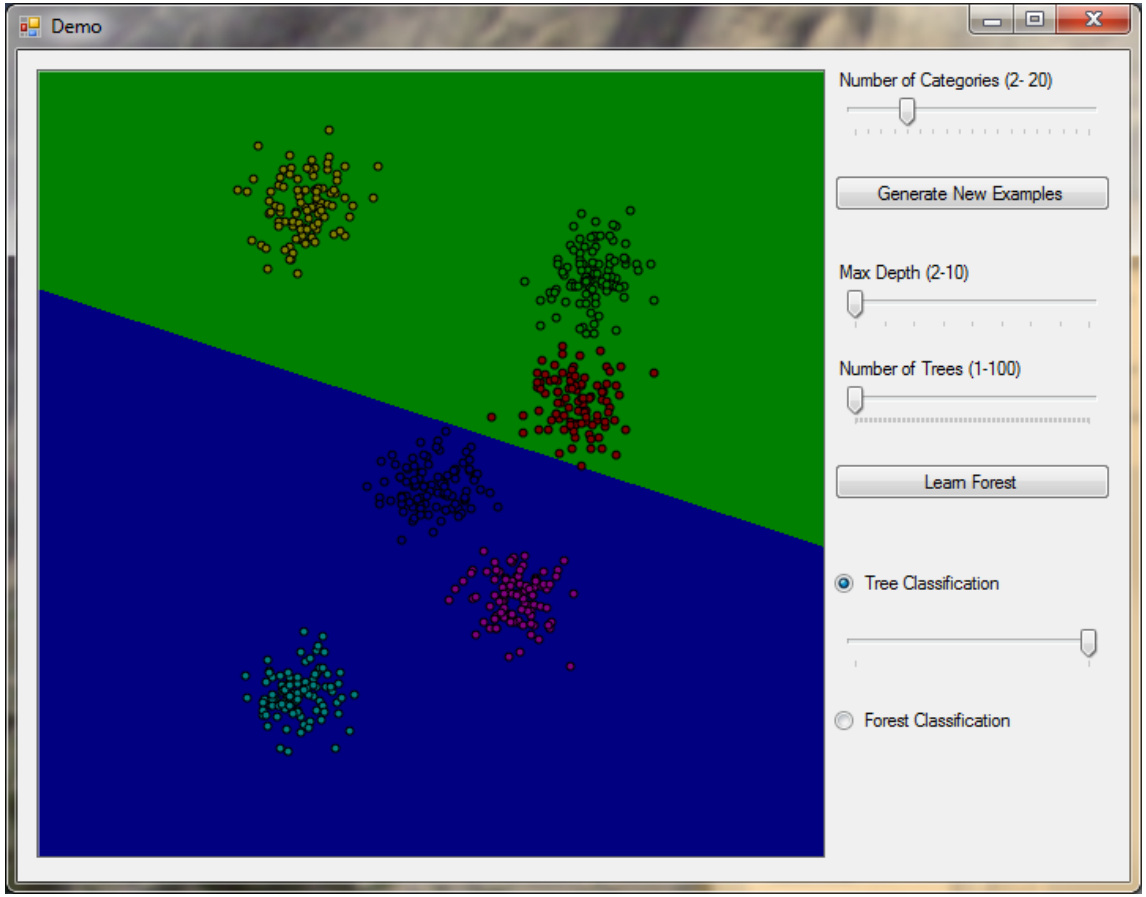
Toy Forest Classification Demo



6 classes in a **2** dimensional feature space.

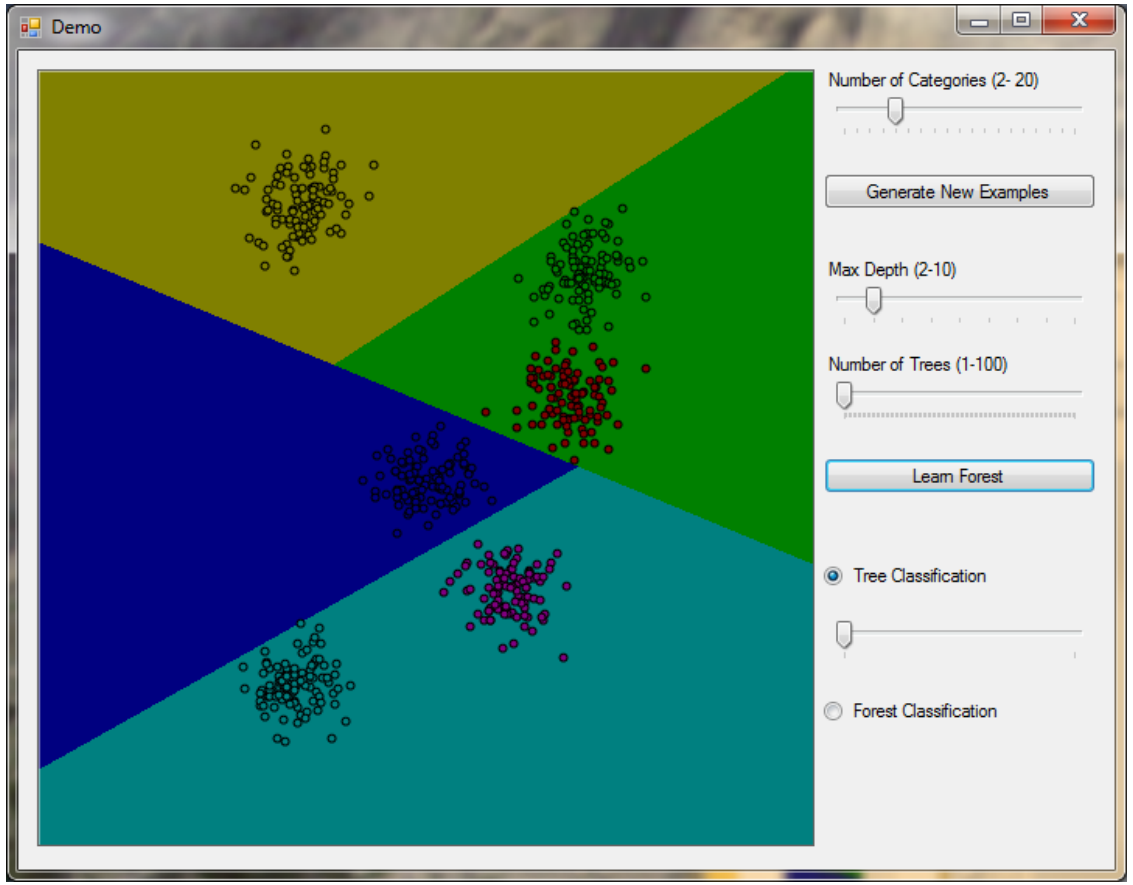
Split functions are lines in this space.

Toy Forest Classification Demo



With a **depth 2** tree, you can not separate all six classes.

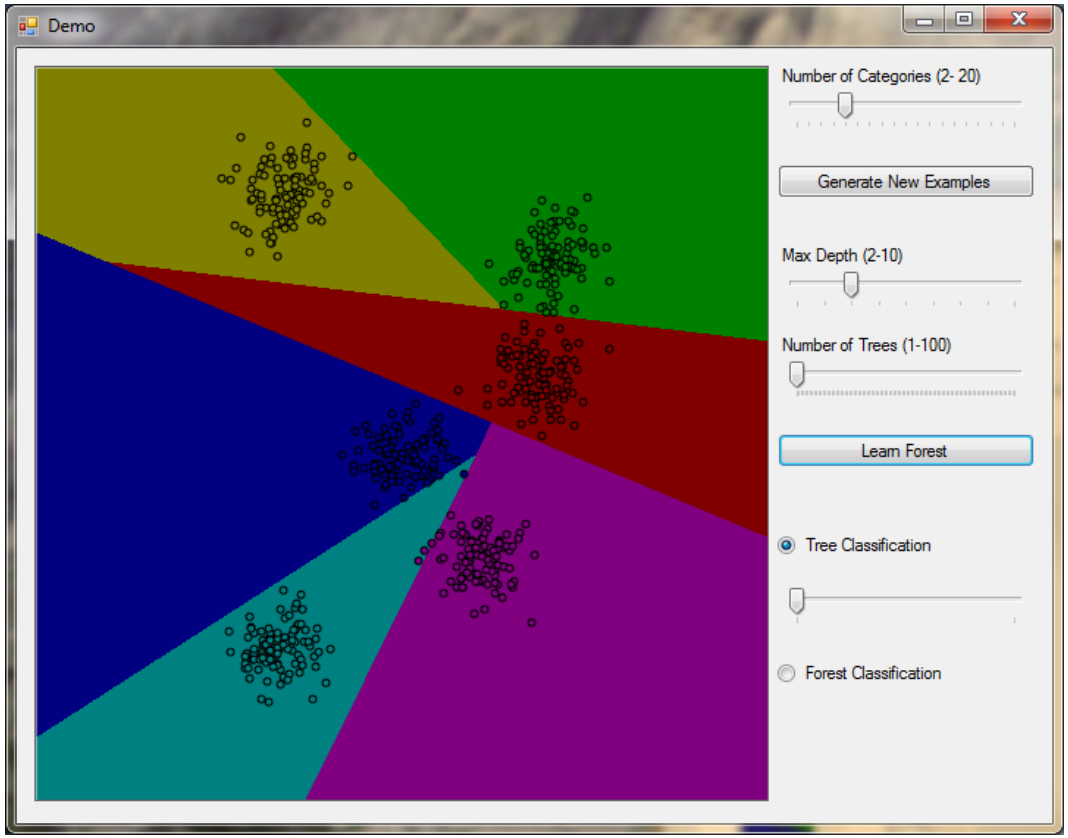
Toy Forest Classification Demo



With a **depth 3** tree, you are doing better,

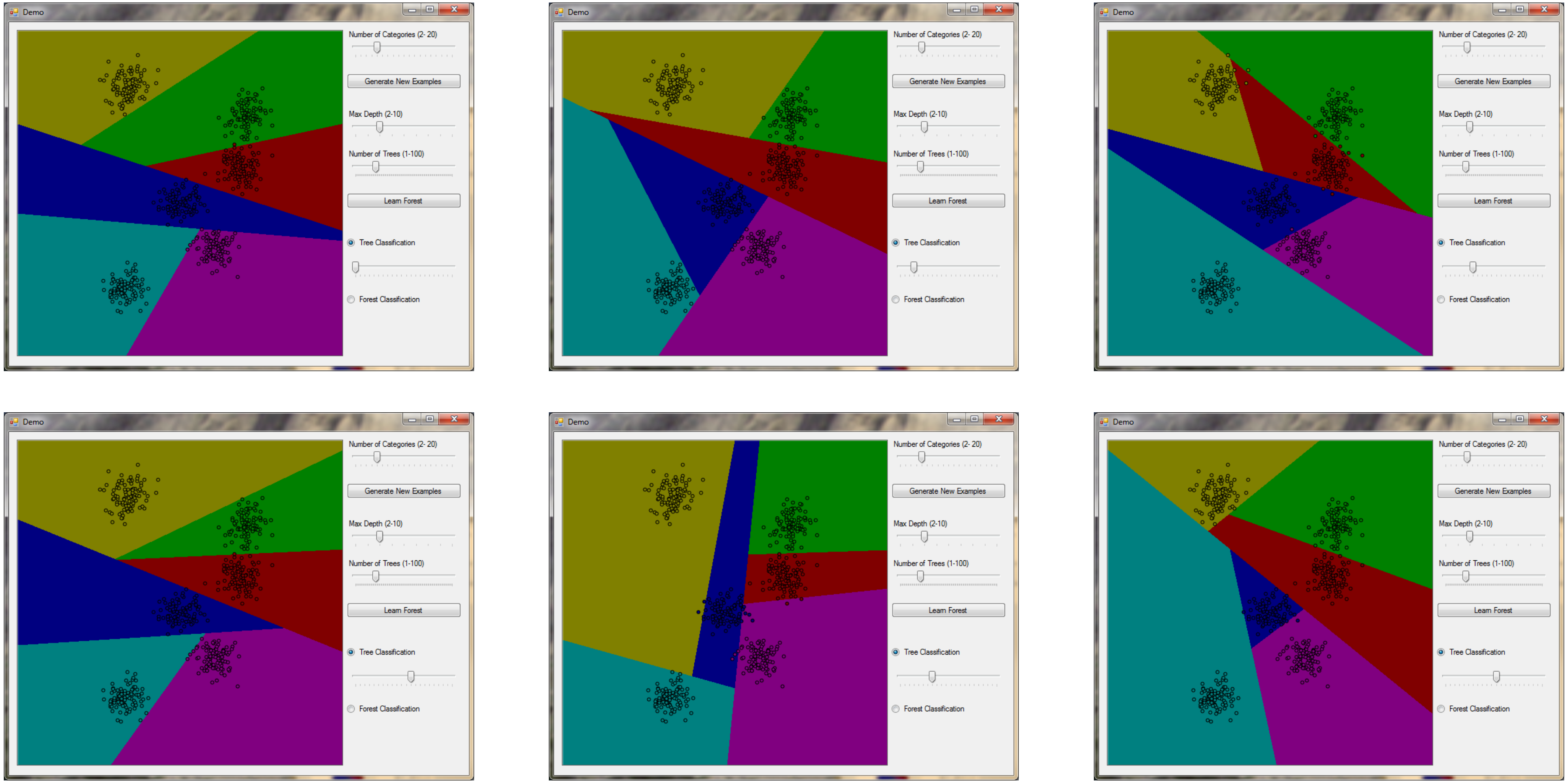
but still cannot separate all six classes.

Toy Forest Classification Demo



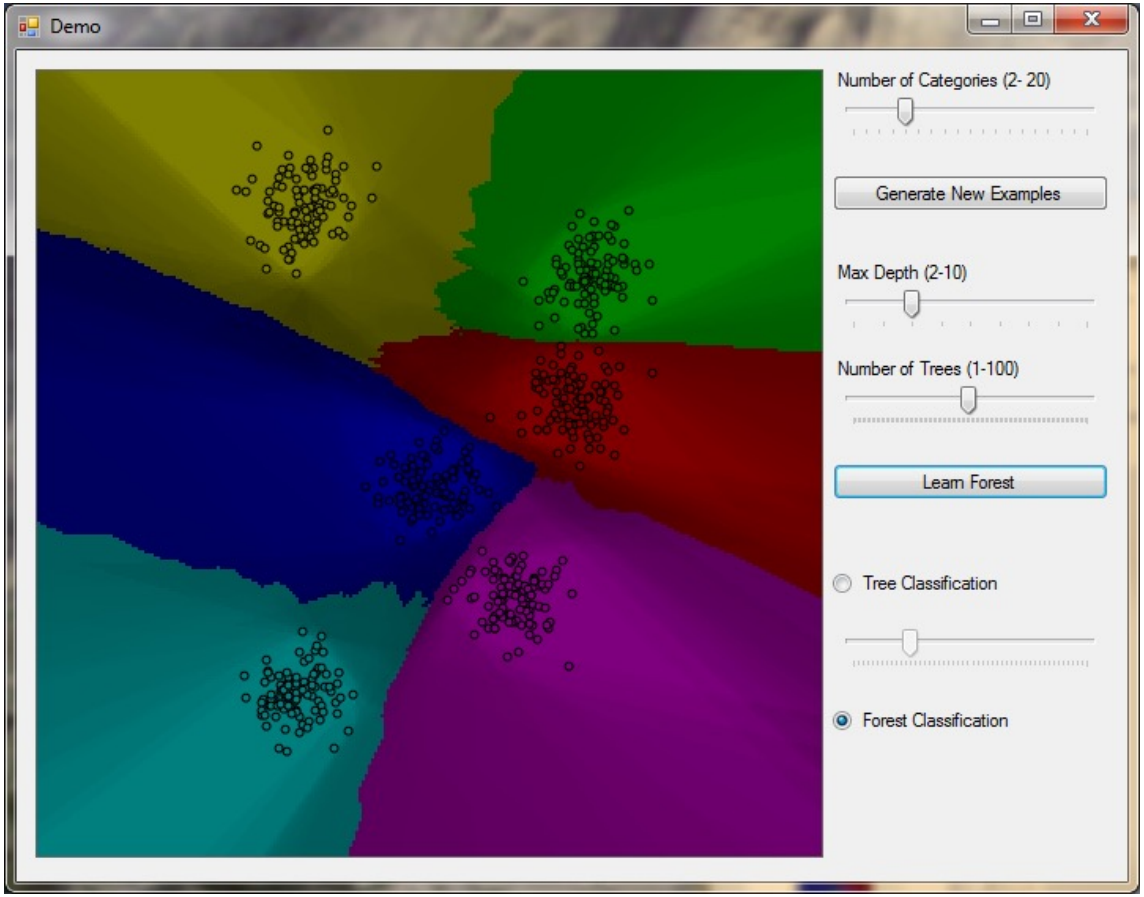
With a **depth 4** tree, you now have at least as many leaf nodes as classes,and so are able to classify most examples correctly.

Toy Forest Classification Demo



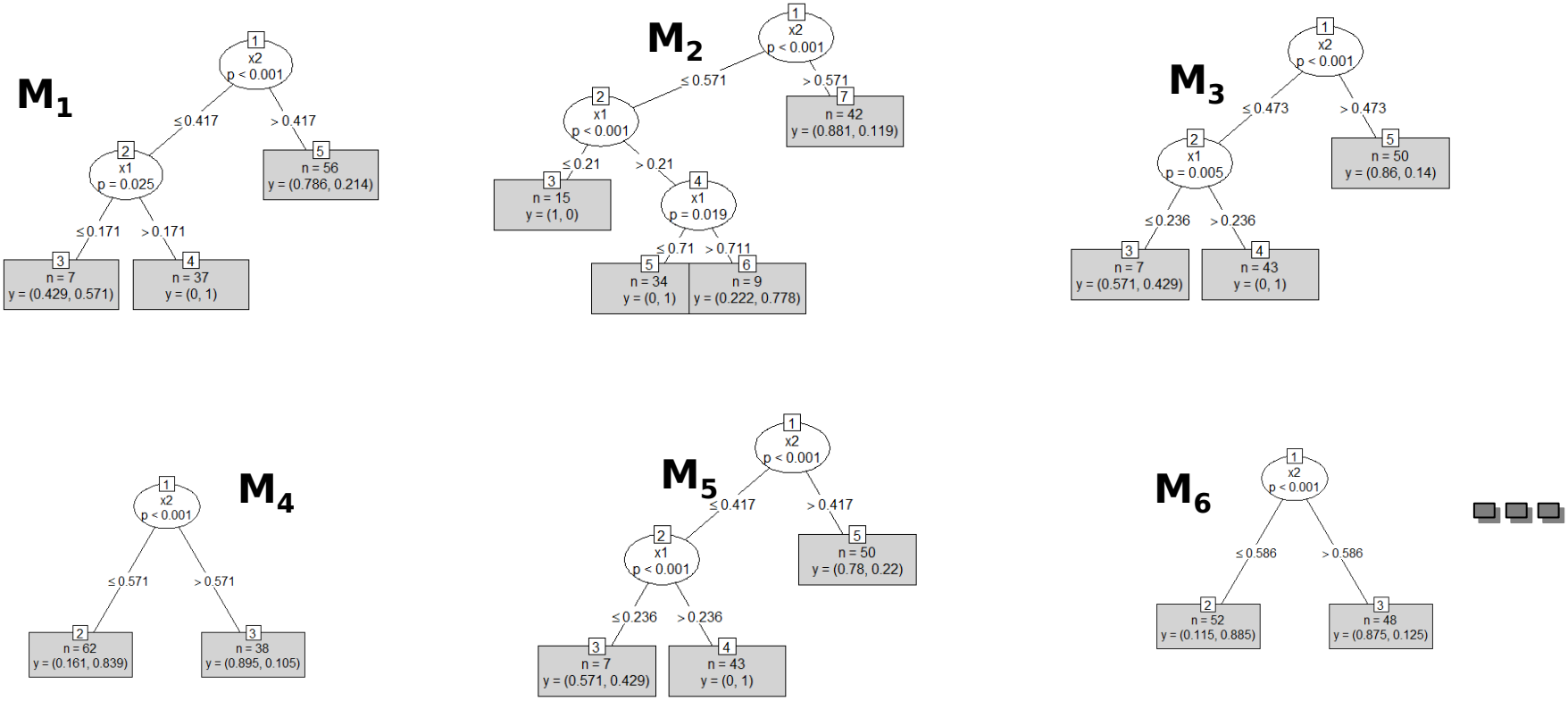
Different trees within a forest can give rise to very different decision boundaries, none of which is particularly good on its own.

Toy Forest Classification Demo



But **averaging together many trees** in a forest can result in decision boundaries that look very sensible, and are even quite close to the max margin classifier. (Shading represents entropy – darker is higher entropy).

Interprétabilité et sélection d'attributs

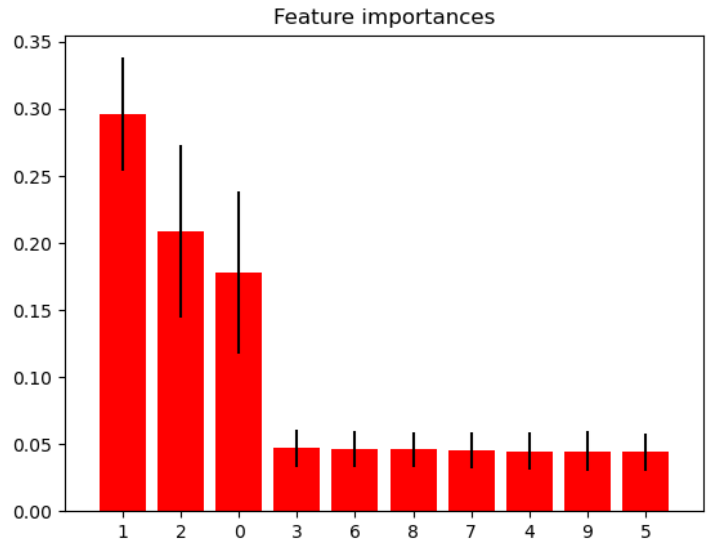


- Multitude d'arbres. Plus de lecture directe possible de l'importance des variables.
- On mesure la fréquence d'apparition

Interprétabilité et sélection d'attributs

- On **additionne** les importances de chaque attribut dans tous les arbres
  - Le **niveau** des attributs dans l'arbre
  - Le **gain** (d'entropie ou de Gini)
- On calcule **la moyenne** (ou la médiane) des importances
- Et on retient les attributs **dépassant cette valeur** seuil
  - Ou les ***N*** premiers attributs

```
print(__doc__)

import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import make_classification
from sklearn.ensemble import ExtraTreesClassifier

# Build a classification task using 3 informative features
X, y = make_classification(n_samples=1000,
                           n_features=10,
                           n_informative=3,
                           n_redundant=0,
                           n_repeated=0,
                           n_classes=2,
                           random_state=0,
                           shuffle=False)

# Build a forest and compute the impurity-based feature importances
forest = ExtraTreesClassifier(n_estimators=250,
                              random_state=0)

forest.fit(X, y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")

for f in range(X.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

# Plot the impurity-based feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
        color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.show()
```




```
Feature ranking:
1. feature 1 (0.295902)
2. feature 2 (0.208351)
3. feature 0 (0.177632)
4. feature 3 (0.047121)
5. feature 6 (0.046303)
6. feature 8 (0.046013)
7. feature 7 (0.045575)
8. feature 4 (0.044614)
9. feature 9 (0.044577)
10. feature 5 (0.043912)
```

https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

Bilan sur les forêts aléatoires

- Souvent **puissant**

- **Mais**
  - Perte d'interprétabilité
  - Et plus coûteux en temps calcul

Plan

1. Méthodes d'ensemble
2. Le boosting
3. Le boosting : pourquoi ça marche
4. Le bagging
5. Les forêts aléatoires (random forests)
6. XGBoost : le boosting d'arbres
7. Vers d'autre méthodes d'ensemble ?

Autres algorithmes

(*Xgboost ou Tree Boosting*)

Xgboost aka. eXtreme gradient boosting

- Références:
  - Boosting applied to decision trees with regularization terms [Friedman]
  - http://xgboost.readthedocs.io/en/latest/model.html

Plan

1. Méthodes d'ensemble
2. Le boosting
3. Le boosting : pourquoi ça marche
4. Le bagging
5. Les forêts aléatoires (random forests)
6. XGBoost : le boosting d'arbres
7. Vers d'autre méthodes d'ensemble ?

Le stacking

- On apprend *T* **hypothèses de base** sur un jeu de données
- On génère un **deuxième jeu** de données indépendant
- Les **sorties** des *T* hypothèses sont utilisés comme des **descripteurs** des données
- Un **méta algorithme** apprend à classer les exemples basé sur ces descripteurs

Le stacking : l'algorithme

Input: Training set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
First-level learning algorithms $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_T$;
Second-level learning algorithm $\mathcal{L}$.

Process:

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $h_t = \mathcal{L}_t(D)$;
3: **end for**
4: $D' = \varnothing$;
5: **for** $i = 1, 2, \ldots, m$ **do**
6: **for** $t = 1, 2, \ldots, T$ **do**
7: $z_{it} = h_t(\boldsymbol{x}_i)$;
8: **end for**
9: $D' = D' \cup ((z_{i1}, z_{i2}, \ldots, z_{iT}), y_i)$;
10: **end for**
11: $h' = \mathcal{L}(D')$.

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$.

Generate first-level learner h_t using first-level learning algorithm $\mathcal{L}_t$.

Generate second-level training set.

Generate second-level learner h' using second-level learning algorithm on $\mathcal{D}'$.

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

Error Correcting Output Codes (ECOC)

ECOC

- Apprendre à distinguer **10 classes**
- Codage
  - Sur 4 bits ?
  - **Sur 15 bits !**

Class	Code Word														
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
2	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
3	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
4	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
5	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
7	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
8	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
9	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1

7 bits d'écart au moins entre les codages des classes ici

ECOC

- On apprend 15 classifieurs binaires

$$h_0(\mathbf{x}) = \begin{cases} 0 & \text{si } y \in \{1,3,5,7,9\} \\ 1 & \text{si } y \in \{0,2,4,6,8\} \end{cases}$$

$$h_1(\mathbf{x}) = \begin{cases} 0 & \text{si } y \in \{1,2,3,6,7\} \\ 1 & \text{si } y \in \{0,4,5,8,9\} \end{cases}$$

$\bullet$
$\bullet$
$\bullet$

- Pour un nouvel ***x***, on calcule $h_1(\boldsymbol{x})$, $h_2(\boldsymbol{x})$, $h_3(\boldsymbol{x})$, $h_4(\boldsymbol{x})$, $h_5(\boldsymbol{x})$, $h_6(\boldsymbol{x})$, $h_7(\boldsymbol{x})$, $h_8(\boldsymbol{x})$, $h_9(\boldsymbol{x})$, $h_{10}(\boldsymbol{x})$, $h_{11}(\boldsymbol{x})$, $h_{12}(\boldsymbol{x})$, $h_{13}(\boldsymbol{x})$, $h_{14}(\boldsymbol{x})$, $h_{15}(\boldsymbol{x})$
- On décode $H(\boldsymbol{x})$ par plus proche voisin / au codage des 10 classes
  (e.g. 0 devrait être codé 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1 par les 15 hypothèses apprises)

ECOC = une méthode d'ensemble !!

- On a **transformé** un problème de classification à 10 classes

  en 15 problèmes de classification binaire

Marche bien !

Dietterich, T. G., & Bakiri, G. (1994). Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, *2*, 263-286.

Des méthodes d'ensemble

*en apprentissage **non** supervisé ?*

Quels ingrédients ?

1. Comment **sélectionner** des « experts » ?
   - Qu'est-ce qu'un **expert** ?
   - Qu'est-ce qu'un **bon panel** d'experts ?
2. Comment leur attribuer un **poids** (éventuellement) ?
3. Comment **combiner** leur avis ?

Références

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

- A. Cornuéjols, Y. Bennani, P. Gançarski and C. Wemmert (2018) "*Collaborative Clustering: Why, When, What and How*", *Information Fusion* (An International Journal on Multi-Sensor, Multi-Source Information Fusion), vol. 39, pp. 81-95.