

# APPRENTISSAGE PAR COMBINAISON DE CLASSIFIEURS ELEMENTAIRES (« dopage » ou « Boosting »)

Pr. Fabien Moutarde  
Centre de Robotique (CAOR)  
MINES Paris Tech (ENSMP)  
PSL Research University

Fabien.Moutarde@mines-paristech.fr  
<http://people.mines-paristech.fr/fabien.moutarde>

Apprentissage par dopage (« boosting ») Pr. Fabien Moutarde, CAOR, MINES ParisTech, PSL Fév.2017 1



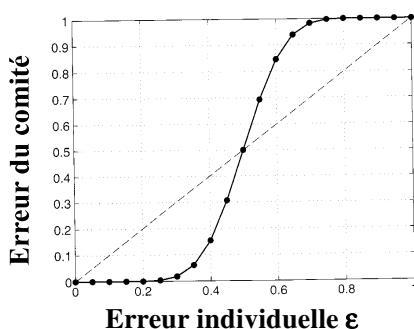
## Principe essentiel : l'union fait la force

**Réunir un « comité d'experts »**  
chacun peut se tromper, mais en combinant les avis,  
on a plus de chance d'avoir la bonne prédiction !!...

### Justification théorique :

- supposons N classifieurs *indépendants*, faisant chacun une erreur  $E_{gen} = \epsilon$
- si on décide « à la majorité », alors on se trompe si et seulement si plus de la moitié du « comité » se trompe

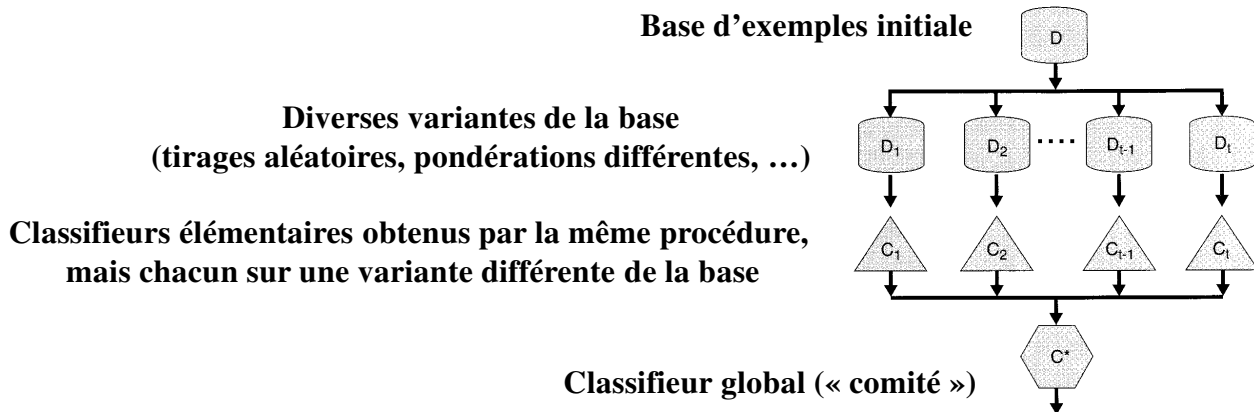
$$\rightarrow Erreur_{ensemble} = \sum_{k=N/2}^N C_k^N \epsilon^k (1-\epsilon)^{N-k}$$



**Décision fortement améliorée,  
(sous réserve que  $\epsilon < 0.5$  !!)...  
...et ce d'autant plus qu'on augmente  
le nombre N d'experts**

Apprentissage par dopage (« boosting ») Pr. Fabien Moutarde, CAOR, MINES ParisTech, PSL Fév.2017 2

- Utiliser des algos totalement différents
- Même algo mais avec des paramètres et/ou initialisations différent(e)s
- **Faire varier l'ensemble d'apprentissage**



→ Méthodes très générales, applicables  
avec n'importe quel algorithme « élémentaire »

## Le « bagging »

### Variantes de la base d'apprentissage obtenues par tirages aléatoires avec remise depuis la base initiale

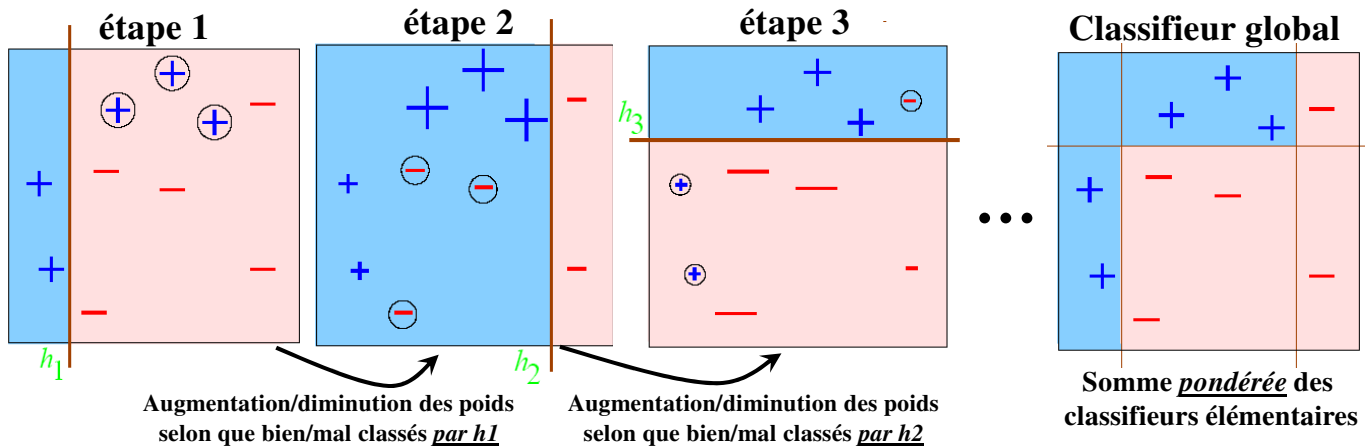
(sorte de « *bootstrap* » → duplication/disparition aléatoires de certains exemples selon variantes)

- Utile et efficace en particulier si l'algo de base utilisé est « instable » (au sens « sensible au bruit des données ») car différences entre variantes de base → classifieurs élémentaires très différents
- Evite « over-fitting » (sur-apprentissage), car on « moyenne » des classifieurs construits avec différentes réalisations aléatoires des mêmes données

## Méthode itérative pour l'ajout des classifieurs :

variantes de la base d'apprentissage obtenues par des pondérations successives des mêmes exemples

(calculées pour « se focaliser » sur les exemples « difficiles », i.e. mal classés par plusieurs classifieurs déjà assemblés)



## L'algorithme adaBoost

### adaBoost (« adaptive Boosting »)

- Base d'exemples initiale :  $S = \{ (x_1, u_1), \dots, (x_k, u_k) \}$ , avec  $u_i \in \{+1, -1\}$ ,  $i=1, k$
- Poids initiaux :  $w_0(x_i) = 1/m$  pour tout  $i=1, k$  (ou  $1/2p$  pr pos et  $1/2n$  pr neg)
- Pour chaque étape (« round »)  $t$  de 1 à  $T$ , faire :
  1. apprendre/choisir 1 règle de classification  $h_t$  sur  $(S, w_t)$  par l'algorithme A
  2. calculer l'erreur pondérée de  $h_t$  sur  $(S, w_t)$  :  $\varepsilon_t = \sum_{i=1}^k w_t(x_i) \times \|h_t(x_i) - u_i\|$
  3. en déduire la « fiabilité » de  $h_t$  :  $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$  [ $\alpha_t > 0$  si  $\varepsilon_t < 0.5$ , et  $\rightarrow +\infty$  si  $\varepsilon_t \rightarrow 0$ ]
  4. modifier les poids, i.e. pour  $i$  de 1 à  $k$  faire :

$$w_{t+1}(x_i) = \frac{w_t(x_i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{si } h_t(x_i) = u_i \text{ (i.e. } x_i \text{ bien classé)} \\ e^{+\alpha_t} & \text{si } h_t(x_i) \neq u_i \text{ (i.e. } x_i \text{ mal classé)} \end{cases}$$

- Fournir en sortie le classifieur global :  $H(x) = \text{signe}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$

Freund et Schapire (qui ont proposé l'algo) ont démontré le théorème suivant :

**Si chaque classifieur élémentaire assemblé a une erreur  $< 0.5$ , alors l'erreur empirique de  $H_T$  sur  $S$  décroît exponentiellement avec le nombre  $T$  d'étapes**

Plus précisément,  $E_{emp}(H_T) = \frac{1}{k} \sum_{i=1}^k \|H_T(x_i) - u_i\|$  est bornée ainsi :

$$E_{emp}(H_T) \leq \prod_{t=1}^T [2\sqrt{\varepsilon_t(1-\varepsilon_t)}] = \prod_{t=1}^T \sqrt{1-4\gamma_t^2}$$

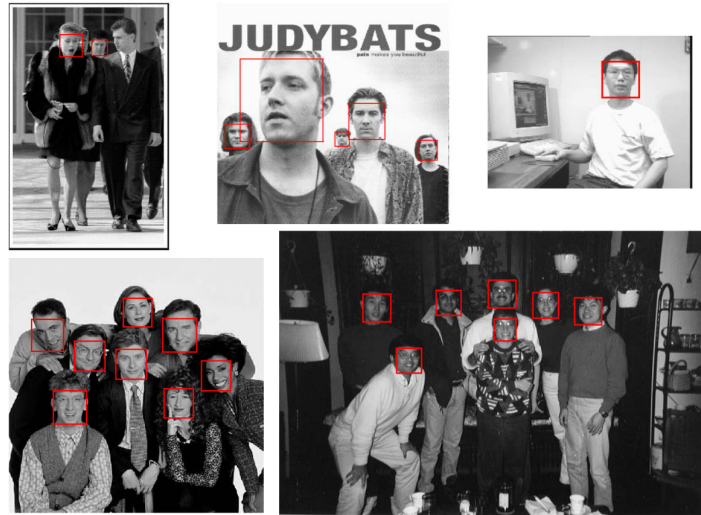
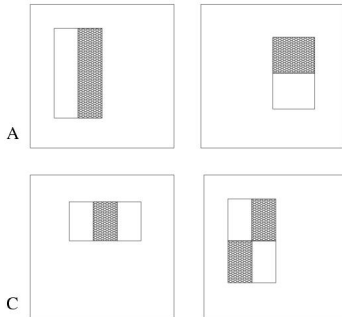
(où  $\gamma_t = 0,5 - \varepsilon_t$  est l'amélioration de  $h_t$  par rapport à une décision aléatoire)

## Avantages et inconvénients du boosting

- On peut obtenir de très bons classifieurs simplement en assemblant un grand nombre de classifieurs « faibles » (c.-à-d. faisant juste un peu mieux que décision aléatoire, i.e. avec  $\varepsilon$  juste légèrement inférieur à 0.5)
- Peut s'utiliser avec absolument n'importe quel algorithme  $A$  (le « weak learner ») de choix/construction du classifieur
- En particulier, le « weak learner » peut par exemple sélectionner un « feature » (notamment visuel dans cas classification d'image) parmi une famille  $\Rightarrow$  adaBoost sert alors simultanément d'algo d'apprentissage et d'outil de « sélection de features » (cf. travaux de Viola et Jones).
- Inconvénients :
  - durée apprentissage souvent longue
  - risque potentiel de sur-apprentissage (over-fitting)

# « Success story »

- **Reconnaissance visuelle de catégorie d'objet par sélection-boosting de « Haar features » ; exemple initial = détection de visages par Viola&Jones (2001)**

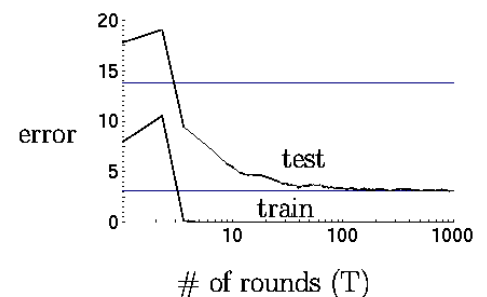


*Classifieurs faibles = comparaison des sommes de valeurs de pixels dans rectangles adjacents*

*Résultat classifieur fort appliqué à multiples sous-fenêtres à diverses échelles et positions (« window-scanning »)*

## Boosting et marges

**Courbe d'erreur typique du boosting :**  
**l'erreur de généralisation continue à diminuer longtemps après que l'erreur empirique sur la base d'apprentissage est devenue nulle !!**

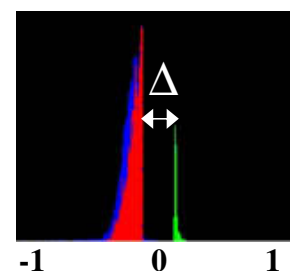


**Ceci s'explique parce que le boosting, même une fois l'erreur nulle sur la base, continue à augmenter les « marges » des exemples, et donc la séparation entre positifs et négatifs...**

Marge  $m$  du classifieur global  $H_T$  sur exemple  $x_i$  :

$$m(H_T, x_i) = u_i \sum_{t=1}^T \alpha_t h_t(x_i) / \sum_{t=1}^T \alpha_t$$

$m(x_i) \in [-1; +1]$ , et  $x_i$  bien classé par  $H \Leftrightarrow m(x_i) > 0$ ,  
 mais **plus  $|m|$  grandit, plus on augmente la séparation  $\Delta$  entre exemples positifs et négatifs**



- L'écart entre l'erreur empirique la généralisation peut être borné par :

$$E_{gen}(H_T) < E_{emp}(H_T) + O\left(\sqrt{\frac{T\delta}{n}}\right)$$

(où T est le nombre d'étapes de boosting, n le nombre d'exemples, et  $\delta$  la VC-dimension de l'espace des  $h_t$ )

⇒ Risque over-fitting ?? (cf aussi survalorisation des exemples « ambigus »)

- Heureusement une borne plus intéressante est :

$$E_{gen}(H_T) < \Pr(m(H_T, x) \leq \theta) + O\left(\sqrt{\frac{\delta}{n\theta^2}}\right)$$

indépendant de T !!

→ si  $\Pr(m(H_T, x) < \theta)$  très faible pour un  $\theta$  assez grand, alors on a une garantie de bonne généralisation...

## Résumé sur le boosting

- Trois principes essentiels :

1. Combiner les avis/estimations de différents experts
2. Modifier, avant chaque ajout d'un expert, la distribution des exemples en surpondérant au fur et à mesure les exemples mal classés
3. Au final, utiliser une moyenne des « votes » des experts, pondérée par leurs fiabilités respectives

- Permet d'obtenir un très bon classifieur en associant des classifieurs très « faibles »
- Problème essentiel : déterminer le « weak learner », i.e. l'algo employé à chaque étape (« round ») de boosting pour choisir/entraîner le classifieur faible  $h_t$