# Ensemble learning

Cours 7
ARF Master DAC

Nicolas Baskiotis
nicolas.baskiotis@lip6.fr
http://webia.lip6.fr/~baskiotisn

équipe MLIA, Laboratoire d'Informatique de Paris 6 (LIP6)
Université Pierre et Marie Curie (UPMC)

S2 (2016-2017)

# Ensemble Learning

## Principe

- Idée simple : considérer plusieurs (beaucoup) de classifieurs
- Avantage : réduit la variance si les classifieurs sont indépedants ! $Var(\hat{X}) = \frac{Var(X)}{n}$
- Mais qu'un jeu de données disponible...

$\Rightarrow$ Différentes techniques d'échantillonage et d'aggrégation pour varier les classifieurs appris

- Inférence : vote majoritaire pondéré sur l'ensemble des classifieurs.

# Plan

# Bagging

## Bootstrap Aggregation — Breiman, 1994

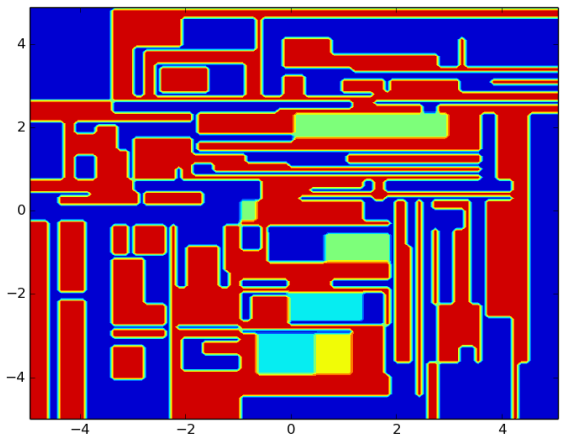
- constitution des $N$ ensembles par tirage aléatoire **avec remise** d'un ensemble de même taille que l'original :
  $E \Rightarrow \{E_1, E_2, \ldots E_N\}$, avec $|E_i| = |E| = N$
- Apprendre $f_1, \ldots f_N$ sur ces ensembles d'apprentissage
- Classer $\mathbf{x}$ par moyennage ou vote de $f_1(\mathbf{x}), \ldots f_N(\mathbf{x})$
- Chaque donnée a une probabilité de $(1 - 1/n)^n$ d'être dans un $E_i$ donné.

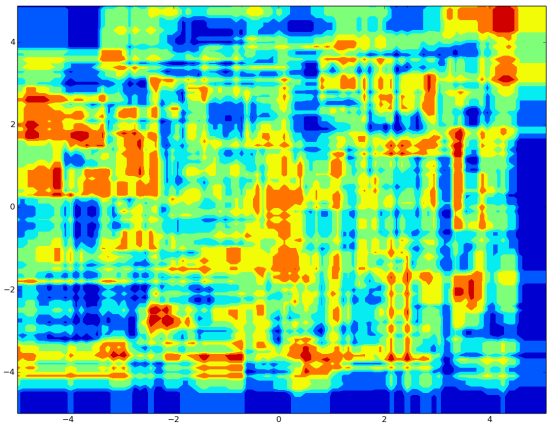
$\Rightarrow$ $E_i$ contient en moyenne $1 - (1 - 1/n)^n\% = 63.2\%$ des instances initiales.

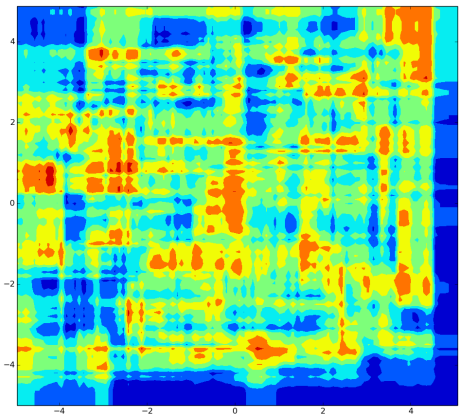
# Un exemple : plusieurs arbres = une fôret

## Principe

- A l'origine pour des considérations computationelles
- Deux facteurs d'aléa :
  - chaque arbre est appris sur un ensemble bootstrap de l'initial (bagging)
  - à chaque nœud, un sous-ensemble des dimensions est considéré uniquement, tiré aléatoirement.
- Décision au vote majoritaire (ou en moyenne pour la régression).
- Remarques : Effet de la profondeur ? Sur-apprentissage ?





# Plan

# Boosting

## Principe

- Pourquoi faire de l'aléatoire quand on connait où on fait l'erreur ?
- Trouver plein de petits classifieurs approximativement bons sur de petites régions de l'espace.
- Idée : aggréger plein de petits classifieurs, appris séquentiellement, chacun corrigeant les erreurs des précédents.

$\Rightarrow$ Besoin de classifieurs faibles ! (pourquoi ?)

## Questions

- Qu'est ce qu'un classifieur faible ?

# Boosting

## Principe

- Pourquoi faire de l'aléatoire quand on connait où on fait l'erreur ?
- Trouver plein de petits classifieurs approximativement bons sur de petites régions de l'espace.
- Idée : aggréger plein de petits classifieurs, appris séquentiellement, chacun corrigeant les erreurs des précédents.

$\Rightarrow$ Besoin de classifieurs faibles ! (pourquoi ?)

## Questions

- Qu'est ce qu'un classifieur faible ?

$\Rightarrow$ classifieur peu expressif, arbres de faibles profondeurs, perceptrons ...

- Comment prendre en compte les erreurs ?

# Boosting

## Principe

- Pourquoi faire de l'aléatoire quand on connait où on fait l'erreur ?
- Trouver plein de petits classifieurs approximativement bons sur de petites régions de l'espace.
- Idée : aggréger plein de petits classifieurs, appris séquentiellement, chacun corrigeant les erreurs des précédents.

$\Rightarrow$ Besoin de classifieurs faibles ! (pourquoi ?)

## Questions

- Qu'est ce qu'un classifieur faible ?

$\Rightarrow$ classifieur peu expressif, arbres de faibles profondeurs, perceptrons . . .

- Comment prendre en compte les erreurs ?

$\Rightarrow$ Considérer une distribution des exemples $D_t$ différente à chaque pas de temps

- Comment combiner les classifieurs ?

# Boosting

## Principe

- Pourquoi faire de l'aléatoire quand on connait où on fait l'erreur ?
- Trouver plein de petits classifieurs approximativement bons sur de petites régions de l'espace.
- Idée : aggréger plein de petits classifieurs, appris séquentiellement, chacun corrigeant les erreurs des précédents.

$\Rightarrow$ Besoin de classifieurs faibles ! (pourquoi ?)

## Questions

- Qu'est ce qu'un classifieur faible ?

$\Rightarrow$ classifieur peu expressif, arbres de faibles profondeurs, perceptrons . . .

- Comment prendre en compte les erreurs ?

$\Rightarrow$ Considérer une distribution des exemples $D_t$ différente à chaque pas de temps

- Comment combiner les classifieurs ?

$\Rightarrow$ Somme pondérée des classifieurs

- Combien de classifieurs apprendre ?

# AdaBoost

## Principe

- $E = \{x^i, y^i\}$ un ensemble de données, distribution $D_t(i) = w_t^i$ sur ces données au temps $t$ : $\sum_i w_t^i = 1$
- $\mathbf{h} = \{h_1, \cdots, h_T\}$ un ensemble de classifieurs,
- $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_T\}$ un ensemble de réels,
- $f_T(x) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) =< \boldsymbol{\alpha}, \mathbf{h} >$, $F_T(\mathbf{x}) = sign(f_T(\mathbf{x}))$ le classifieur pondéré.
- Objectif : trouver $(\mathbf{h}^\star, \boldsymbol{\alpha}^\star) = argmin_{\mathbf{h},\boldsymbol{\alpha}} \frac{1}{N} \sum_i 1_{F(\mathbf{x}^i) \neq y^i}$

## Algorithme

1. Initialiser la distribution : $D_0(i) = \frac{1}{|E|}$
2. Apprendre $h_t$ sur $D_t$
3. Calculer l'erreur $\epsilon_t = \sum_i D_t(i) 1_{h_t(\mathbf{x}^i) \neq y^i}$
4. Fixer $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$
5. Fixer $D_{t+1}(i) = \frac{1}{Z_t} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}^i)}$

# Remarques

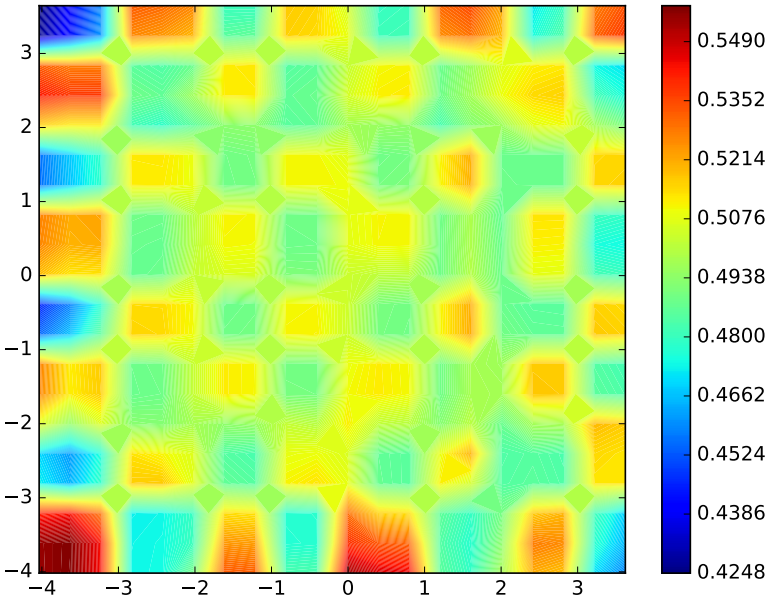
## Considérations sur les poids

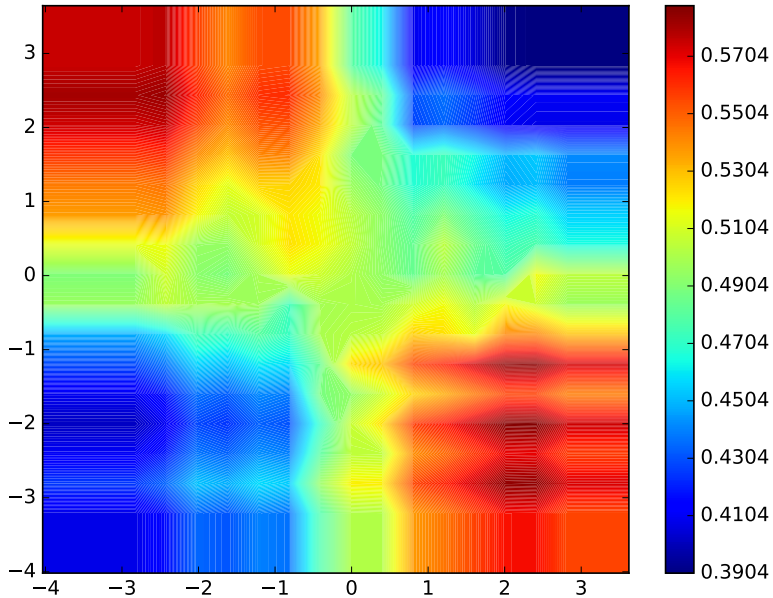
- $\epsilon_t < \frac{1}{2} \Rightarrow \alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t}) > 0$
- $\epsilon(h_a) < \epsilon(h_b) \Rightarrow \alpha_a > \alpha_b$
- $e^{-y\alpha_t h_t(\mathbf{x})} = \begin{cases} e^{-\alpha_t} < 1 & \text{si } h_t(\mathbf{x}) = y \\ e^{\alpha_t} > 1 & \text{si } h_t(\mathbf{x}) \neq y \end{cases}$

## Considérations sur la distribution

- $D_{t+1}(i) = \frac{1}{Z_t} D_t(i) e^{-\alpha_t y^i h_t(\mathbf{x}^i)} = \frac{1}{Z_t Z_{t-1}} D_{t-1}(i) e^{-y^i(\alpha_t h_t(\mathbf{x}^i) + \alpha_{t-1} h_{t-1}(\mathbf{x}^i))}$
  $\cdots = \frac{1}{Z_t \cdots Z_1} D_1(i) e^{-y^i(\alpha_t h_t(\mathbf{x}^i) + \cdots + \alpha_1 h_1(\mathbf{x}^i))}$
- On montre que $Z = Z_1 \cdots Z_t = \frac{1}{N} \sum_{i=1}^{N} e^{-y^i f_t(\mathbf{x}^i)}$
- Et que $Err(F) \leq Z$

# Illustrations



# Conclusions

## Sur le bagging

- Très utilisé ! (kinect, les gagnants de netflix)
- Facile à mettre en place, peut traiter de grosses masses de données (parallélisation), en apprentissage et en inférence

## Boosting

- Classifieurs faibles : Stump (arbre à un niveau), naive bayes, perceptron,. . .
- Adaptable sous beaucoup d'autres formes (gradient tree boosting, gradient boosting)
- Adapté au très grande masse de données et données sparse (ciblage publicitaire par exemple)