

SPLEX

Statistiques pour la classification et fouille de données en génomique

Apprentissage statistique

Pierre-Henri WUILLEMIN

Decision-axe IA-LIP6
pierre-henri.wuillemin@lip6.fr

Quelques points d'organisation

- **Intervenants en cours** : Nataliya Sokolovska, Pierre-Henri Wuillemin
- TME en python (Nataliya Sokolovska)
- Cours (55–65 201bis) – TME (14–15 509)
- **Évaluation** : 2 examens répartis ($70\% = 35\% + 35\%$), une note de TME (30%)

Un (tout petit) peu d'histoire

La classification est un thème majeur de l'histoire de la biologie.

● Andreas Caesalpinus (Césalpin),

De plantis libri XVI, Florence, 1583 : Première taxonomie.

- A. Arides (arbres et arbustes)
 - 1. Fruits étoiles-pennes (4 régiments: chêne, noyer, pêcher, prunier)
 - 2. Fruits bipartites (figuier, saule)
 - 3. Fruits imparies (baies, myrte)
 - 4. Fruits quadrangulaires charnus de cardinal, (aspirin)
 - 5. Fruits multicarpelles (poivrier, Conifères)
- B. Arborescentes, plantes vivaces et annuelles
 - 1. Fruits mono-permes (vératres, renoncules, graminées, tiges)
 - 2. Fruits bipartites (Labiées, Légumineuses, Crocusiers)
 - 3. Fruits imparies (fougères, Iris, violettes)
 - 4. Fruits tuberculeux (Broméliacées, Zygophyllacées)
 - 5. Fruits polypérimes (Composées, Asteracées, Malvacées, ortie, rosacées)
- C. Plantes sans graines
 - 1. Fougères
 - 2. Ricophyses
 - 3. Algues
 - 4. Champignons.



● Buffon, 1749

« Le seul moyen de faire une méthode instructive et naturelle est de mettre ensemble les choses qui se ressemblent et de séparer celles qui diffèrent les unes des autres. »



● XIX^{ème} siècle : Théorie statistique

- Quetelet (1796–1874)
- Recensement américain, 1890 (utilisation de la carte perforée).

● XX^{ème} siècle : Analyse de données, apprentissage statistique

Quel est le problème ?

Description de la tâche de classification

- Identifier la classe d'appartenance des objets à partir de certains de leurs attributs (*features*),
- Attribuer une classe à un objet à partir de ces attributs.

Buts de la classification

- Connaissance (apprentissage) de la structure des ensembles d'objets
- Processus de décisions automatiques

Applications

- Diagnostic médical
- Prêt bancaire, Marketing
(*profiling*)
- Fusion de senseurs
- Contrôle de process
- Reconnaissance de formes
(*pattern recognition*)
- etc.

Analyse de données et Data Mining

▶ Définition (Analyse de données)

Ensemble de méthodes d'explorations de données considérées comme des points dans un espace vectoriel multi-dimensionnel.

Par exemple : Analyse en Composantes Principales (ACP), régression linéaire, etc.

▶ Définition (Data Mining)

Organisation et exploration de données considérées comme des points dans un espace vectoriel multi-dimensionnel dans le but d'apprendre une structure ou d'apprendre à prédire.

- Prédire : apprentissage par l'exemple afin de mimer un comportement.
 - Discrimination,
 - Régression,
 - Classification supervisée, etc.
- Apprendre une structure : extraire de l'information.
 - Classification non supervisée,
 - Détection de motifs, etc.

Formalisation

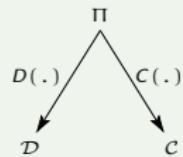
- Soit Π une population (un ensemble d'objets),
- chaque élément de Π est déterminé par d attributs (supposément quantitatifs). On nomme $\mathcal{D} \subseteq \mathbb{R}^d$ l'espace descriptif,
- chaque élément de Π appartient à une classe de \mathcal{C} .

Autrement dit,

Relations fonctionnelles

$\exists D : \Pi \rightarrow \mathcal{D}$ et $C : \Pi \rightarrow \mathcal{C}$ telles que $\forall \pi \in \Pi$,

- $D(\pi)$ est le vecteur des attributs déterminant π ,
- $C(\pi)$ est la classe de π .



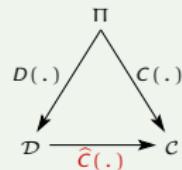
On ne connaît en fait un objet π que par l'intermédiaire de sa description $D(\pi)$.

Tâche de classification

Trouver $\hat{C} : \mathcal{D} \rightarrow \mathcal{C}$ telle que

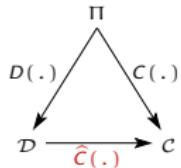
$\forall \pi, \hat{C}(D(\pi)) \approx C(\pi)$ avec "le moins d'erreur possible"

\hat{C} est appelé le **classifieur**.



Trouver \hat{C} : méthodologie en mode supervisé

On suppose que \mathcal{D} , \mathcal{C} et D sont parfaitement connus **mais pas C .**



➊ Apprentissage sur un base d'exemples

Soit $\Pi_a \subset \Pi$ pour lequel on connaît parfaitement la restriction $C|_{\Pi_a}$,
on peut alors **estimer** \hat{C} sur Π_a .

➋ Validation sur un base d'exemples

Soit $\Pi_v \subset \Pi \setminus \Pi_a$ pour lequel on connaît parfaitement la restriction $C|_{\Pi_v}$,
On peut alors **évaluer** \hat{C} sur Π_v et **construire des indicateurs de qualité**.

➌ Prédiction

$\forall \pi \in \Pi \setminus (\Pi_a \cup \Pi_v)$, on ne connaît pas $C(\pi)$: **on le prédit par $\hat{C}(D(\pi))$.**

Nature des données

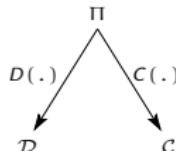
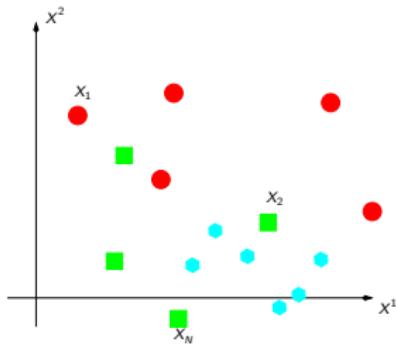
 Chaque donnée $(D, \Pi_a, C|_{\Pi_a}, \Pi_v, C|_{\Pi_v},)$ peut être fausse, bruitée, incomplète, non indépendante, etc.

L'estimation est une approximation et la construction de l'estimateur peut être une heuristique !

Simplifier les notations

Généralement, le problème de classification prendra la forme suivante :

- Soit une v.a. X (de dimension d) à valeurs dans \mathbb{R} ($\mathcal{D} = \mathbb{R}^d$),
 - discrète (*classification* : $\mathcal{C} \subset \mathbb{Z}$)
 - continue (*régression* : $\mathcal{C} \subseteq \mathbb{R}$).
- Soit une v.a. Y (de dimension 1)
- $\Pi_a \cup \Pi_v$ est une base de N observations (X_i, Y_i) de ces variables.



	$D(\pi)$		$C(\pi)$
π_1	X_1^1	X_1^2	Y_1
π_2	X_2^1	X_2^2	Y_2
π_3	X_3^1	X_3^2	Y_3
...
π_n	X_N^1	X_N^2	Y_N

Le problème de la classification (et de la régression)

Soit x une instantiation de X , quelle valeur prendrait $y = \hat{C}(x)$?

Deux approches de la classification supervisée

Approches non paramétriques

- Pas d'hypothèse sur le modèle des données,
- Seuls les exemples mènent vers \widehat{C} .

Exemple : méthode des *k plus proches voisins, fenêtre de Parzen.*

Approches paramétriques

- À partir de l'**hypothèse forte** de l'existence d'un modèle,
- Phase d'estimation des paramètres du modèle : $\widehat{\Theta}$,
- \widehat{C} est alors $\widehat{C}_{\widehat{\Theta}}$.

Exemple : Régression, Classification bayésienne

Pour la suite du cours (sauf si précisé), on simplifiera la tâche de classification à la discrimination entre 2 classes seulement : Y peut prendre les valeurs -1 et 1 .

Le classifieur \widehat{C} se décompose alors en 2 fonctions : $g : \mathbb{R}^d \rightarrow \mathbb{R}$ et $\widehat{C} = \sigma(g)$.

Évaluations et Comparaisons de Classifieurs

Évaluation d'un classifieur

Quelles propriétés peut-on demander pour \hat{C} ?

Critère : précision

Un classifieur doit minimiser le taux d'erreur lors de son utilisation. (?)

Critère : compréhensibilité

Un classifieur représentant une base doit être intelligible :

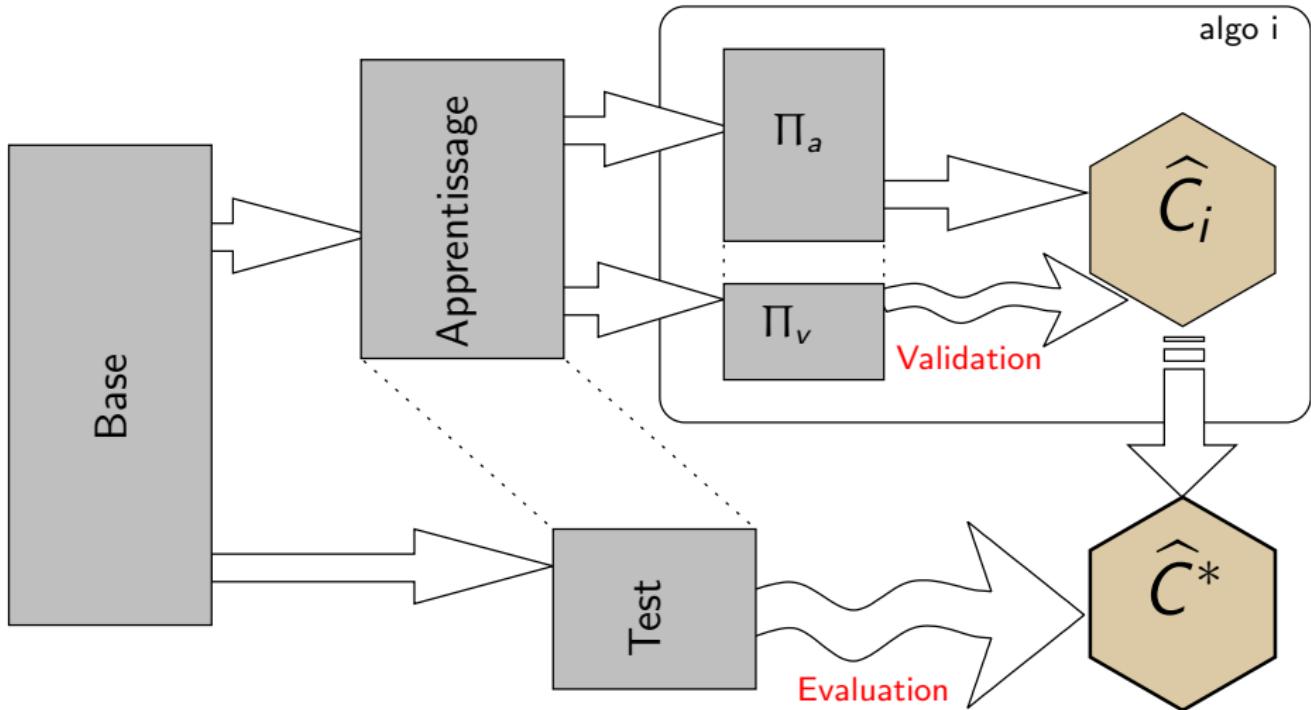
- expliquable (et donc exploitable),
- compréhensible (par un expert),
- paramètrable (par un expert).

Critère : rapidité

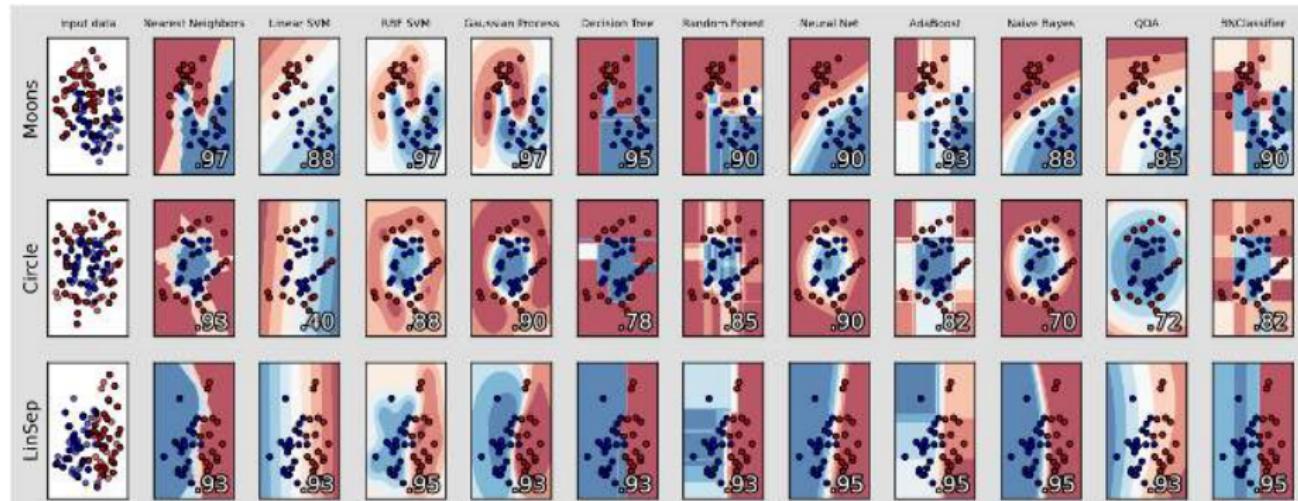
Les opérations sur le classifieur doivent être facilitées.

- Construction aisée,
- Mise à jour aisée (incrémentale),
- Critère de classification rapide.

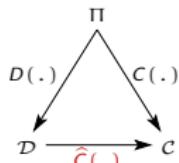
Méthodologie



Différents classifieurs



Évaluation de \widehat{C}



Ce qu'on voudrait connaître, c'est l'erreur que l'on fera en utilisant \widehat{C} au lieu de C .

Erreur en généralisation

$$e(\widehat{C}) = \mathbb{E}_x (C(x) \neq \widehat{C}(x))$$

En régression, on utilisera plutôt la distance quadratique :

$$e(\widehat{C}) = \mathbb{E}_x (C(x) - \widehat{C}(x))^2$$

Toutefois, cette erreur en généralisation n'est pas atteignable. Il faut donc l'estimer.

Erreur en apprentissage, erreur en validation

$$e_a(\widehat{C}) = \frac{1}{\|\Pi_a\|} \sum_{x \in \Pi_a} \delta(C(x) - \widehat{C}(x)) \quad e_v(\widehat{C}) = \frac{1}{\|\Pi_v\|} \sum_{x \in \Pi_v} \delta(C(x) - \widehat{C}(x))$$

où $\delta(x) = \begin{cases} 1 & \text{si } x \neq 0 \\ 0 & \text{si } x = 0 \end{cases}$

Mais ces erreurs expérimentales n'indiquent a priori rien d'autres que des hypothèses sur $e(\widehat{C})$.

Erreurs des erreurs de classification

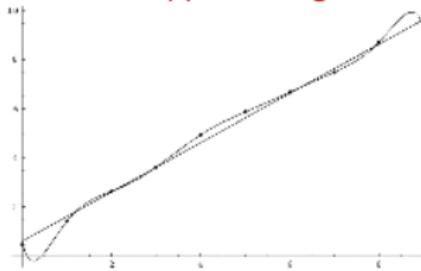
erreur d'apprentissage

Sur la base $\Pi_a = \bigcup_{i=1}^{N_a} \{x_i^{(a)}\}$:

$$\epsilon_a = \frac{1}{N_a} \sum_{i=1}^{N_a} \delta \left(\hat{C}(x_i^{(a)}) - C(x_i^{(a)}) \right)$$

⚠️ On peut rendre $\epsilon_a = 0$.

⚠️ \hat{C} peut se sur-adAPTER à Π_a :
sur-apprentissage

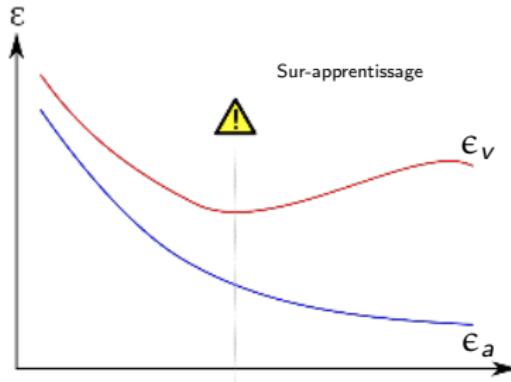


erreur de test de validation

Sur la base $\Pi_v = \bigcup_{i=1}^{N_v} \{x_i^{(v)}\}$:

$$\epsilon_v = \frac{1}{N_v} \sum_{i=1}^{N_v} \delta \left(\hat{C}(x_i^{(v)}) - C(x_i^{(v)}) \right)$$

⚠️ $\epsilon_v = 0$ est très dépendant de Π_v
(biais, approximation, etc.).



Estimation du taux d'erreur de \hat{C}

Estimation sur Π_a

- ⊕ Estimation aisée !
- ⊖ Sous-estimation évidente
- ⊖ Favorise le surapprentissage

Estimation sur Π_v

- ⊕ Estimation aisée !
- ⊖ Gâchis d'une partie des données classées
- ⊖ Données issues de la même base que $\Pi_a \Rightarrow$ risque de biais

Cross-Validation, Leave-one-out, bootstrap

Avec $C(\Pi^*)$ bien connu, construire plusieurs Π_v différents et apprendre sur $\Pi_a = \Pi^* \setminus \Pi_v \dots$

- ⊕ Plus de perte de données,
- ⊕ Une certaine robustesse de l'évaluation,
- ⊖ Cher et lent !!

LOOCV : Leave-One-Out Cross Validation

LOOCV est un algorithme qui permet de se passer de Π_v en contre-partie d'un temps de calcul beaucoup plus important.

LOOCV

Data : Π_a

- 1 **for each** $\pi \in \Pi_a$ **do**
- 2 Calculer $\widehat{C}_{-\pi}$ en apprenant sur $\Pi_a \setminus \{\pi\}$
- 3 $e_\pi = |\widehat{C}_{-\pi}(\pi) - C(\pi)|$
- 4 Calculer \widehat{C} en apprenant sur Π_a
- 5 alors

$$e_{\text{LOOCV}}(\widehat{C}) = \frac{1}{\|\Pi_a\|} \sum_{\pi \in \Pi_a} e_\pi$$

Il existe bien évidemment des variantes de cet algorithme, moins gourmands car utilisant une partition en k sous-ensembles de Π_a plutôt que d'en isoler chaque singleton : *k-fold Cross Validation*.

Autres indicateurs sur Π_v

Matrice de confusion

Sur une base de test :

	$C(x) = \ominus$	$C(x) = \oplus$	
$\widehat{C}(x) = \ominus$	VN	FN	$VN + FN$
$\widehat{C}(x) = \oplus$	FP	VP	$VP + FP$
	$VN + FP$	$FN + VP$	N

Indicateurs

- Taux d'erreur : $e_v(\widehat{C}) = \frac{FP+FN}{N}$
- Accuracy** (taux de bon classement) : $\frac{VP+VN}{N}$
- Spécificité** (taux de vrai négatifs) : $\frac{VN}{VN+FP}$
- 1-Spécificité** (taux de faux positifs) : $\frac{FP}{VN+FP}$
- Sensibilité, Rappel** (taux de vrai positifs) : $\frac{VP}{VP+FN}$
- Précision** : $\frac{VP}{VP+FP}$



High accuracy, but
low precision



High precision, but
low accuracy

Coûts d'affectation

Il peut être intéressant de briser la symétrie entre FP et FN .

Matrice d'affectation

	$C(x) = \ominus$	$C(x) = \oplus$
$\hat{C}(x) = \ominus$	0	γ_{FN}
$\hat{C}(x) = \oplus$	γ_{FP}	0

Généralement, on priviliege les erreurs de première espèce : $\gamma_{FP} \gg \gamma_{FN}$

On peut alors définir un coût d'un classifieur :

Indicateur de coût

$$cout(\hat{C}) = \frac{FP \cdot \gamma_{FP} + FN \cdot \gamma_{FN}}{N}$$

Limite des indicateurs issus de la matrice de confusion

Comment comparer

\widehat{C}_1	(-)	(+)
(-)	40	10
(+)	10	40

et

\widehat{C}_2	(-)	(+)
(-)	45	5
(+)	20	30

?

- $e_V(\widehat{C}_1) = 0.2$ et $e_V(\widehat{C}_2) = 0.25$

$$\Rightarrow \widehat{C}_1 \succ \widehat{C}_2$$

- Avec la matrice de coût :

	(-)	(+)
(-)	0	10
(+)	1	0

$$cout(\widehat{C}_1) = 1.1 \text{ et } cout(\widehat{C}_2) = 0.7$$

$$\Rightarrow \widehat{C}_1 \prec \widehat{C}_2$$

PS : Comparer e_V revient à utiliser une matrice de coût avec $\gamma_{FP} = \gamma_{FN} = 1$.

Doit-on tester pour toutes les matrices de coût possible ?

Comment comparer globalement les modèles sans coût ?

Cas de classes déséquilibrées

Matrice de confusion – COIL'00 – Challenge police d'assurance

Sur une base de test :

\widehat{C}	$\widehat{-}$	$\widehat{+}$
$\widehat{-}$	3731	31
$\widehat{+}$	229	9

- $e_V(\widehat{C}) = \frac{229+31}{4000}$

- $e_V(\text{Toujours } \widehat{-}) = \frac{31+9}{4000}$

$\widehat{C} \prec \text{Toujours } \widehat{-}$

Au lieu de se limiter à des critères sur l'affectation proposée, il s'agirait de mesurer la *propension à être dans une classe*.

Un classifieur est souvent composé de :

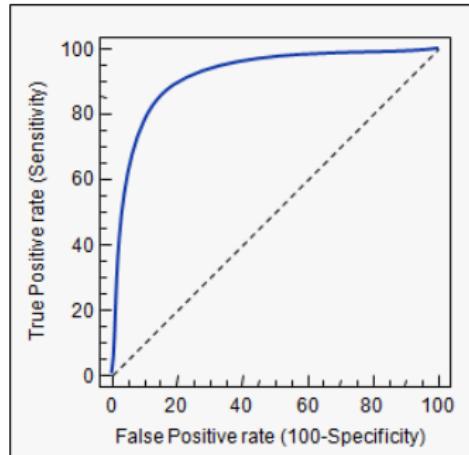
- Une fonction $g : \mathcal{D} \longrightarrow \mathbb{R}$
- Une fonction $\sigma : \mathbb{R} \longrightarrow \mathcal{C}$ (fonction signe si classificateur binaire)

Quand $\widehat{C}_i(x) = \sigma(g_i(x))$, il faudrait comparer g_1 et g_2 !!!

Courbe de ROC

Courbe de ROC : *Receiver Operating Characteristic*

- Outil d'évaluation et de comparaison de classifieurs,
- Outil graphique (aisément lisible),
- Indépendant des matrices de coûts,
- Utilisable dans le cas de classes déséquilibrées,
- Indicateur synthétique associé (aisément interprétable).



Construction de la courbe de ROC

Soit $\hat{C}(x) = \sigma(g(x))$.

Soit $B_>$ la base des $x \in \Pi_a$, classés par valeur croissante de $g(x)$.

On construit alors une famille de classifieurs $\hat{C}_k, k \in \{0, \dots, |\Pi_a|\}$, définis ainsi :

$$\forall k \in \{0, \dots, |\Pi_a|\}, \hat{C}_k = \{\text{classer } \ominus \text{ les } x \text{ de rang } \leq k \text{ dans } B_>\}$$

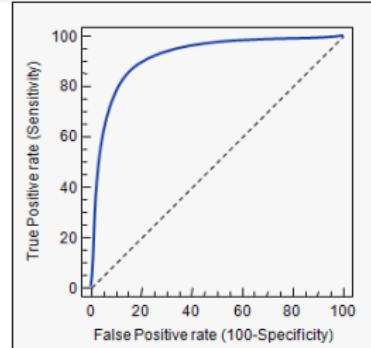
Rappel :

- 1-Spécificité (TFP) : $\frac{FP}{VN+FP}$
- Sensibilité (TVP) : $\frac{VP}{VP+FN}$

- \hat{C}_0 : sensibilité=1 et spécificité=0
- $\hat{C}_{|\Pi_a|}$: sensibilité=0 et spécificité=1

Construction de la courbe de ROC

$\forall k \in \{0, \dots, |\Pi_a|\}$, afficher le point
(1-spécificité(C_k), sensibilité(C_k))



construction de la courbe de ROC (1)

Classer les données
selon un score décroissant

Individu	Score (+)	Classe
1	1	+
2	0.95	+
3	0.9	+
4	0.85	-
5	0.8	+
6	0.75	-
7	0.7	-
8	0.65	+
9	0.6	-
10	0.55	-
11	0.5	-
12	0.45	+
13	0.4	-
14	0.35	-
15	0.3	-
16	0.25	-
17	0.2	-
18	0.15	-
19	0.1	-
20	0.05	-

Positifs = 6
Négatifs = 14

Seuil = 1

	*positif	*négatif	Total
positif	1	5	6
négatif	0	14	14
Total	1	19	20

$$TVP = 1/6 = 0.2 ; TFP = 0/14 = 0$$

Seuil = 0.95

	*positif	*négatif	Total
positif	2	4	6
négatif	0	14	14
Total	2	18	20

$$TVP = 2/6 = 0.33 ; TFP = 0/14 = 0$$

Seuil = 0.9

	*positif	*négatif	Total
positif	3	3	6
négatif	0	14	14
Total	3	17	20

$$TVP = 3/6 = 0.5 ; TFP = 0/14 = 0$$

Seuil = 0.85

	*positif	*négatif	Total
positif	3	3	6
négatif	1	13	14
Total	4	16	20

$$TVP = 3/6 = 0.5 ; TFP = 1/14 = 0.07$$

Seuil = 0

	*positif	*négatif	Total
positif	6	0	6
négatif	14	0	14
Total	20	0	20

$$TVP = 6/6 = 1 ; TFP = 14/14 = 1$$

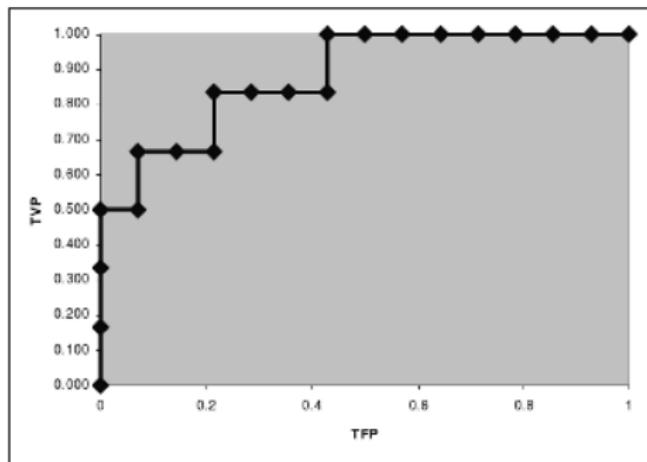
construction de la courbe de ROC (2)

Mettre en relation
TFP (abscisse) et TVP (ordonnée)

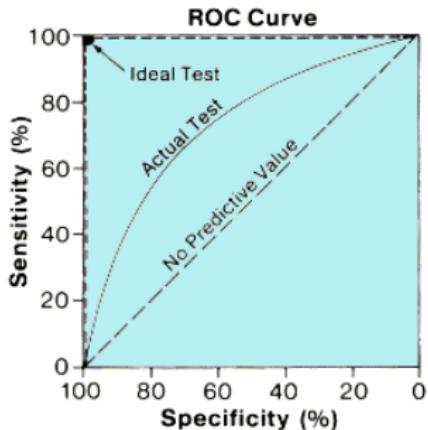
Individu	Score (+)	Classe	TFP	TVP
1	1	+	0.000	0.167
2	0.95	-	0.000	0.333
3	0.9	+	0.000	0.500
4	0.85	-	0.071	0.500
5	0.8	-	0.071	0.667
6	0.75	-	0.143	0.667
7	0.7	-	0.214	0.667
8	0.65	-	0.214	0.833
9	0.6	-	0.286	0.833
10	0.55	-	0.357	0.833
11	0.5	-	0.429	0.833
12	0.45	+	0.429	1.000
13	0.4	-	0.500	1.000
14	0.35	-	0.571	1.000
15	0.3	-	0.643	1.000
16	0.25	-	0.714	1.000
17	0.2	-	0.786	1.000
18	0.15	-	0.857	1.000
19	0.1	-	0.929	1.000
20	0.05	-	1.000	1.000



Courbe ROC



Utilisation de la courbe de ROC (1)



➊ Tout classifieur est un point dans l'espace ROC.

- ➌ (0, 1) : Classifieur parfait.
- ➌ (1, 1) : Classifieur "toujours \oplus ".
- ➌ (0, 0) : Classifieur "toujours \ominus ".
- ➌ (q , q) : Classifieur " $\text{proba}(\hat{C}(x) = \oplus) = q$ ".

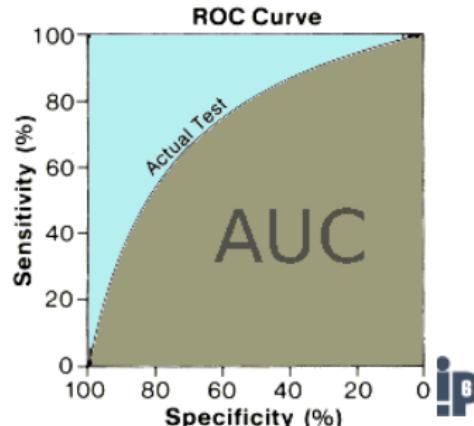
➋ Une famille de classifieurs \hat{C}_k doit avoir sa courbe de ROC **au dessus** de la première diagonale ($x = y$).

Puisque cette première diagonale correspond aux classifieurs aléatoires.

➌ critère AUC Area Under Curve

- ➌ Pour la diagonale : $AUC = \frac{1}{2}$
- ➌ Pour le test idéal, $AUC = 1$
- ➌ Pour un test quelconque, $\frac{1}{2} \leq AUC \leq 1$. Plus il est grand, mieux c'est.

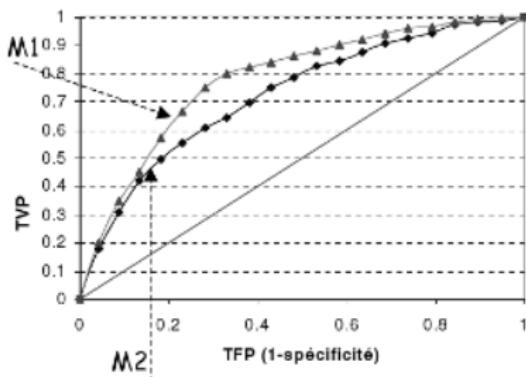
AUC indique la probabilité pour que \hat{C} interclasse un positif et un négatif.



Utilisation de ROC (2) : comparaison de classifieurs

1 Dominance

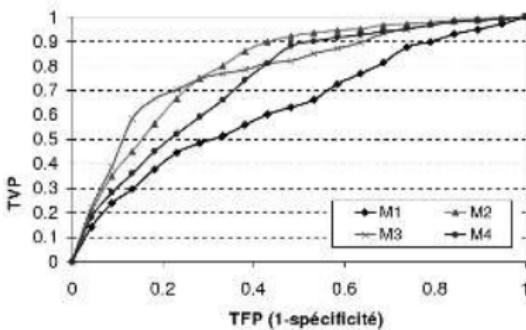
Si $ROC_{M1} \succ ROC_{M2}$ alors il ne peut pas exister de situation (quelque soit le coût) où $M2$ serait un meilleur classifieur que $M1$.



2 Enveloppe convexe

Les classifieurs ne participant jamais à l'enveloppe convexe sont dominés et peuvent être éliminés.

- $M1$ dominé
- $M4$ dominé par $\max(M2, M3)$



Aggrégations de Classieurs

Introduction au bootstrap

Principe

Problème : Quelle est la distribution d'un estimateur calculé à partir d'un échantillon L ? Si on connaît la distribution de l'échantillon ou si on peut faire l'hypothèse d'une distribution gaussienne, pas de problèmes. Mais sinon ?

But du bootstrap : Remplacer des hypothèses probabilistes pas toujours vérifiées ou même invérifiables par des simulations et donc beaucoup de calcul.

Exemple : $L = \{x_1, \dots, x_n\}$ suffit à calculer \bar{x} , estimateur de μ . Comment estimer sa précision (son écart-type) ? son biais ?

- ① iid + TCL $\Rightarrow \bar{X} \sim \mathcal{N}(\mu, \frac{\sigma^2}{n})$.
- ② Soit $(L_j)_{j \in \mathcal{J}}$ une famille d'échantillons.
 \Rightarrow on peut estimer ces statistiques à partir des $(\bar{x}_j)_{j \in \mathcal{J}}$.
- ③ Soit L l'unique échantillon utilisable \Rightarrow calculer un ensemble d'échantillons *bootstrap*.

Bootstrap : points de repères

- Bradley Efron, 1979. Efron & Tibshirani, 1993.

- Origine du nom : Inspirée du baron de Münchausen (Rudolph Erich Raspe) qui se sortit de sables mouvants par traction sur ses *tirants de bottes*.



- Popularisé avec la montée en puissance de l'ordinateur dans les calculs statistiques.
- Basé sur une théorie mathématique bien rôdée.

Échantillon bootstrap

General Bootstrap Algorithm

- Data :** L échantillon, θ paramètre à estimer sur L ,
- 1 **for** $b \in \{1, \dots, B\}$ **do**
 - 2 Tirer aléatoirement L_b , un échantillon **avec remise** dans L
 - 3 Estimer $\hat{\theta}_b$ sur l'échantillon L_b
 - 4 $\tilde{L} = \{\hat{\theta}_1, \dots, \hat{\theta}_B\}$ est un échantillon *bootstrap* issu de L .

$X=(3.12, 0, 1.57,$ $19.67, 0.22, 2.20)$ Mean=4.46	$X1=(1.57, 0.22, 19.67,$ $0, 0, 2.2, 3.12)$ Mean=4.13
	$X2=(0, 2.20, 2.20,$ $2.20, 19.67, 1.57)$ Mean=4.84
	$X3=(0.22, 3.12, 1.57,$ $3.12, 2.20, 0.22)$ Mean=1.74

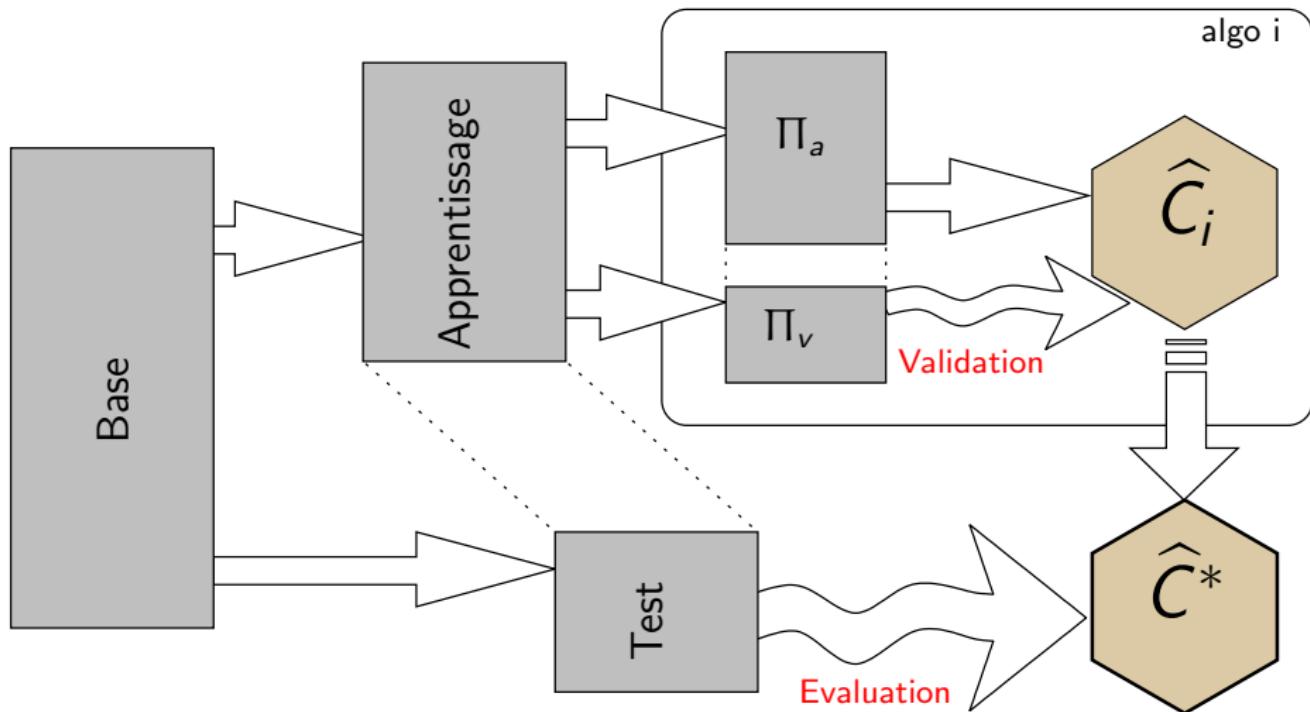
On peut alors utiliser \tilde{L} pour estimer des statistiques de la distribution de l'estimateur de θ :

- $\sigma_{\hat{\theta}}^2 \approx \hat{\sigma}_B^2 = \frac{1}{B-1} \sum_{b=1}^B \left(\hat{\theta}_b - \frac{1}{B} \sum_{b=1}^B \hat{\theta}_b \right)^2$
- Intervalle de confiance, biais, quantiles, tests d'hypothèses, etc.

Considérations sur le bootstrap

-  \tilde{L} ne remplace pas L .
 \tilde{L} est utilisé pour estimer des paramètres d'un estimateur basé sur L !!
- Le bootstrap ne peut pas s'appliquer si :
 - Échantillon L trop faible
 - Données trop bruitées
 - Dans le cas de dépendances structurelles dans L (par exemple, séries temporelles, problèmes 2D,etc.)
- Comment choisir $B = |\hat{L}|$?
- Un grand nombre de versions légèrement différentes.
Par exemple, **Jackknife** : extraire les n sous-listes de L de taille $n - 1$.

Méthodologie et agrégations



- Est-ce que choisir \hat{C}^* parmi les \hat{C}_i a un sens ?
- Y a-t-il vraiment un 'meilleur' ?

Aggrégations

Soit un certain nombre de classifieurs $(\hat{C}_i)_{i \in \mathcal{I}}$ pour une même classification C . Chaque C_i est plus ou moins précis. Ils sont également plus ou moins stables.

Stabilité

Un classifieur \hat{C} est instable quand de petites modifications dans Π_a peuvent entraîner de grandes modifications dans la classification proposée par \hat{C} .

Les statistiques nous ont appris que moyenner les résultats est un bon moyen de réduire la variance (l'instabilité).

Est-il possible de combiner les $(\hat{C}_i)_{i \in \mathcal{I}}$ de manière à obtenir un meilleur classifieur ?

Aggrégation majoritaire

En considérant chaque \hat{C}_i comme un 'expert', une manière d'aggréger leurs avis est de les faire voter et de choisir l'avis majoritaire :

$$\hat{C}_{\text{maj}}(x) = \arg \max_{c \in \{\oplus, \ominus\}} \left| \left\{ \hat{C}_i(x) = c, i \in \mathcal{I} \right\} \right|$$

Aggrégations avec poids

On peut raffiner l'aggrégation majoritaire en introduisant des poids associés à l'expert.

Aggrégation majoritaire avec poids

Pour chaque \hat{C}_i , on note w_i son poids associé, alors $\hat{C}_w(x) = \arg \max_{c \in \{\oplus, \ominus\}} \sum_{\substack{i \in \mathcal{I} \\ (\hat{C}_i(x)=c)}} w_i$

Pour fixer les poids, on s'attend à ce qu'un expert qui donne souvent la bonne réponse ait un poids 'élevé'.

Adaptation des poids

Data : $\beta \in [0, 1]$

- 1 $\forall i, w_i = 1$
- 2 **foreach** $\pi \in \Pi_a$ **do**
- 3 **if** $\hat{C}_i(\pi) \neq C(\pi)$ **then**
- 4 $w_i = \beta \cdot w_i$

- Si $\beta = 0$, algorithme dit de Halving.
- Si $\beta = \frac{1}{2}$, avec k le nbr minimal d'erreurs faites par un \hat{C}_i sur Π_a , n le nombre de \hat{C}_i , et M le nombre d'erreurs faite par C_w ,

$$M \leq 2.4(k + \log_2(n))$$

Bagging

Pour le bagging, on considère qu'il faut constituer des ensembles $\Pi_a^{(i)}$ spécifiques pour estimer chaque classifieur \widehat{C}_i (qui peuvent être alors de même type) et moyenner les résultats : ça ressemble à du **bootstrap**.

Bootstrap aggregating

Soit $\left(\Pi_a^{(i)}\right)_{i \in \mathcal{I}}$ une famille d'échantillons bootstrap (avec remise), alors :

$$\forall i \in \mathcal{I}, \widehat{C}_i \text{ est estimé sur } \Pi_a^{(i)} \quad \text{et} \quad C_{\text{bagging}}(x) = \widehat{C}_{\text{maj}}(x)$$

Si les \widehat{C}_i sont effectivement des classificateurs de même type, heuristiquement, il doivent être instables afin que chaque échantillon bootstrap fournit un classificateur différent.

Convergence et généralisation

- Pas de preuve de convergence.
- Pas de borne connue pour l'erreur de généralisation.
- En pratique, souvent excellents résultats (XGBoost, Random Forest).

Modèles additifs

- Dans les classifieurs majoritaires à poids, la classification s'effectue comme espérance de plusieurs classifieurs, éventuellement pondérés.
- Il s'agit donc de choisir chaque classifieur et son poids associé de manière à améliorer le résultat global.
- **idée** : au lieu de construire les classifieurs 'indépendamment', il serait pertinent de construire incrémentalement chaque classifieur afin qu'il s'intéresse particulièrement aux points pour l'instant mal classés.
- **idée** : ça pourrait marcher même avec des classifieurs faibles.

▶ Définition (Classifieur faible (*weak classifier*))

Un classifieur faible est un classifieur à peine plus précis que le classifieur aléatoire.

- C'est le principe des algorithmes de **boosting**.

Boosting : principes

principe : un nouveau classifieur doit se concentrer sur les éléments de Π_a mal classés pour l'instant.

Boosting schématique

- ① Affecter un poids à chaque $\pi \in \Pi_a$
- ② Proposer un classifieur faible (" $x_i < \dots$ ", "règle d'or", "rule of thumb")
- ③ Modifier les poids pour tenir compte de ce nouveau classifier
- ④ Recommencer T fois les 2 derniers points.
- ⑤ Combiner tous les classifieurs dans $\hat{C}_{\text{boosting}}$

Deux questions (au moins) :

- Comment mettre à jour les poids ?
- Comment combiner correctement tous ces classifieurs ?

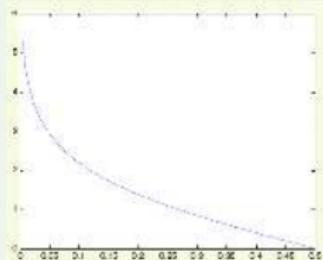
Erreur en généralisation

On rappelle : $e_P(\hat{C}) = \mathbb{E}_P(C(x) \neq \hat{C}(x))$ où P probabilité sur Π .

- En particulier, si \hat{C} est un classifieur faible, $e(\hat{C}) \leq \frac{1}{2} - \gamma$, $\gamma > 0$: \hat{C} est un peu meilleur que le classifieur aléatoire qui se trompe une fois sur deux.
- Pour $\Pi_a = \{x_1, \dots, x_n\}$, on notera $D(x_i)$ le poids associé à x_i . $D(\cdot)$ probabilité ($\sum_i D(x_i) = 1$).

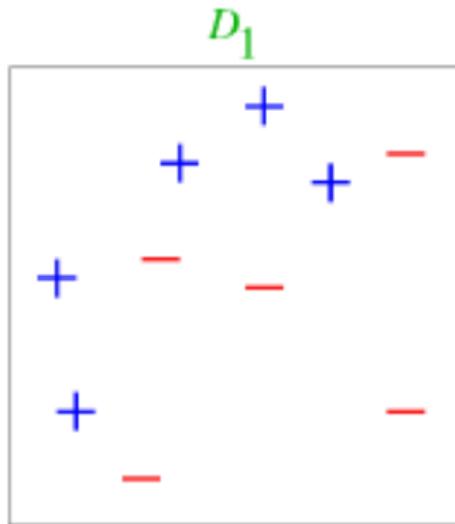
AdaBoost (Freund & Schapire, 1995)

- 1 $\forall x_i, D_1(x_i) = \frac{1}{n}$
- 2 soit \hat{C}_1 un premier classifieur faible
- 3 **repeat**
- 4 Calculer $\epsilon_t = e_{D_t}(\hat{C}_t)$ et $\beta_t = \frac{\epsilon_t}{1-\epsilon_t} < 1$
- 5 $D_{t+1}(x_i) \propto \begin{cases} D_t(x_i) & \text{si } \hat{C}_t(x_i) \neq C(x_i) \\ \beta_t \cdot D_t(x_i) & \text{sinon} \end{cases}$
- 6 Estimer \hat{C}_{t+1} un classifieur faible minimisant $e_{D_{t+1}}$
- 7 **until** condition d'arrêt (T fois par exemple);
- 8 $C_{\text{boosting}}(x) = \sigma \left(\sum_t \alpha_t \hat{C}_t(x) \right)$ avec $\alpha_t = \frac{1}{2} \log \frac{1}{\beta_t}$

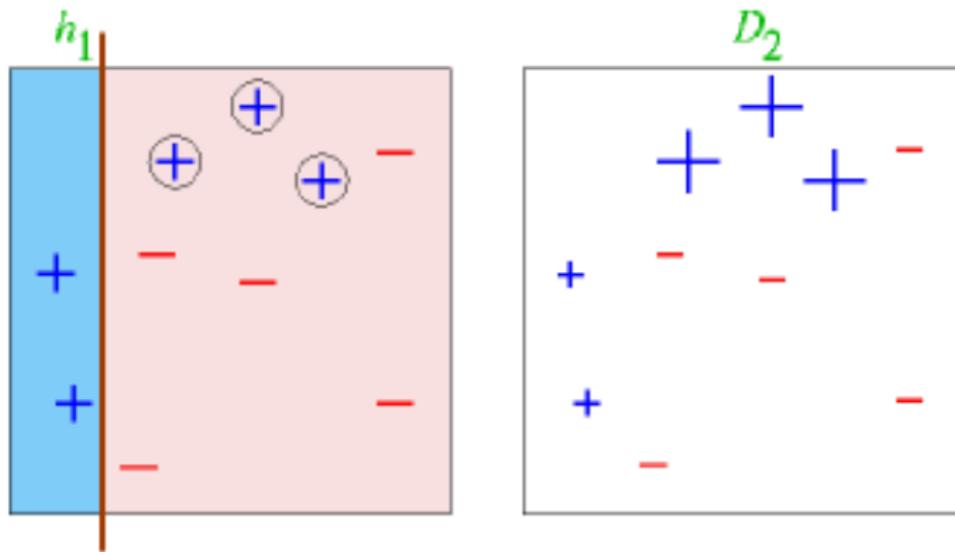


α_t en fonction de $e_{D_t}(\hat{C}_t)$

AdaBoost : un exemple ("A Tutorial On Boosting", Freung,Y. & Schapire,R.)



AdaBoost : un exemple ("A Tutorial On Boosting", Freung,Y. & Schapire,R.)

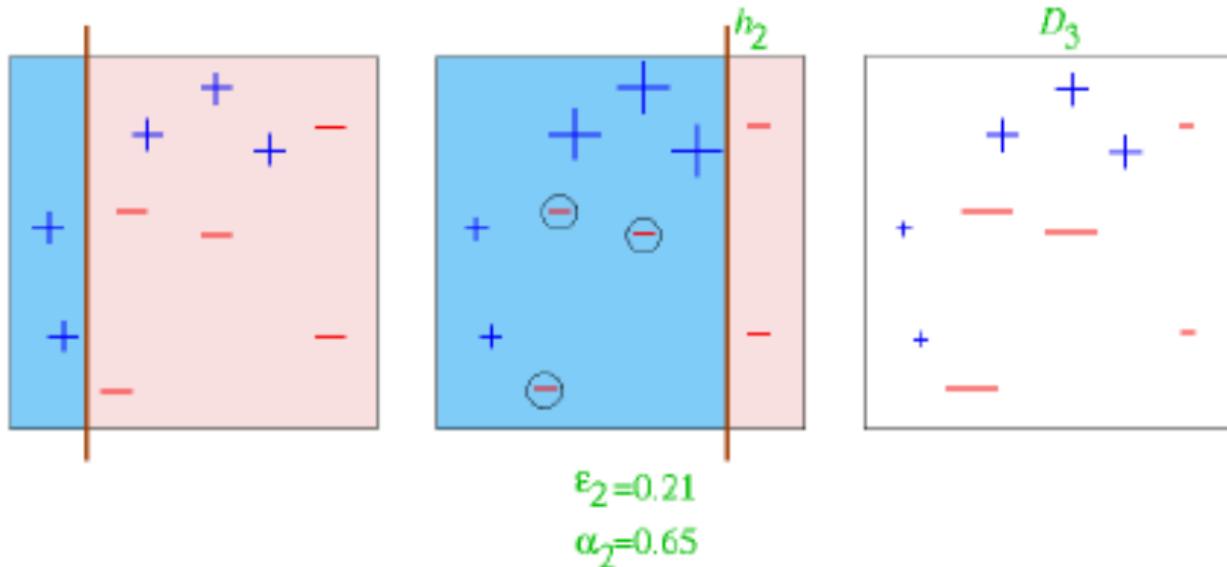


$$\epsilon_1 = 0.30$$

$$\alpha_1 = 0.42$$

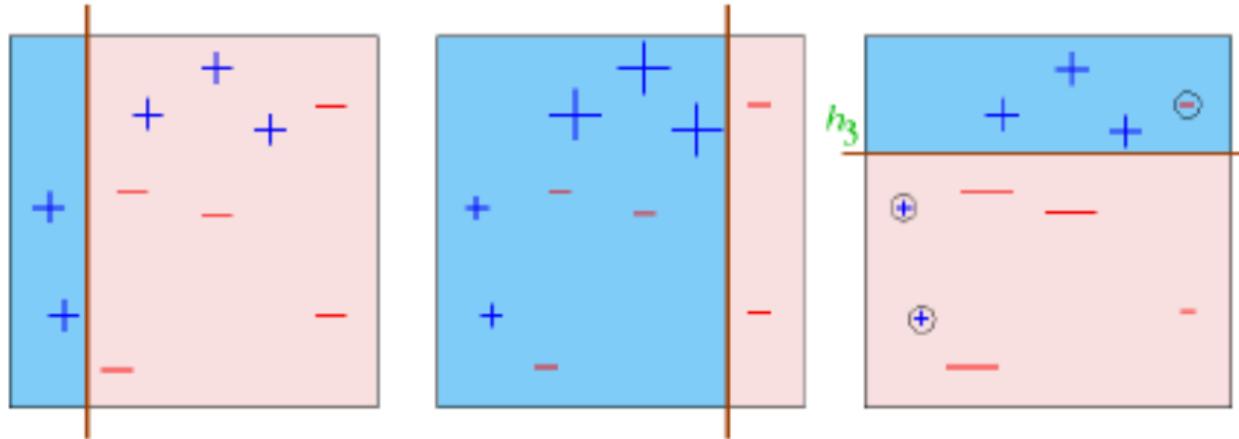
$$\alpha_1 = \frac{1}{2} \log \frac{1 - \epsilon_1}{\epsilon_1}$$

AdaBoost : un exemple ("A Tutorial On Boosting", Freung,Y. & Schapire,R.)



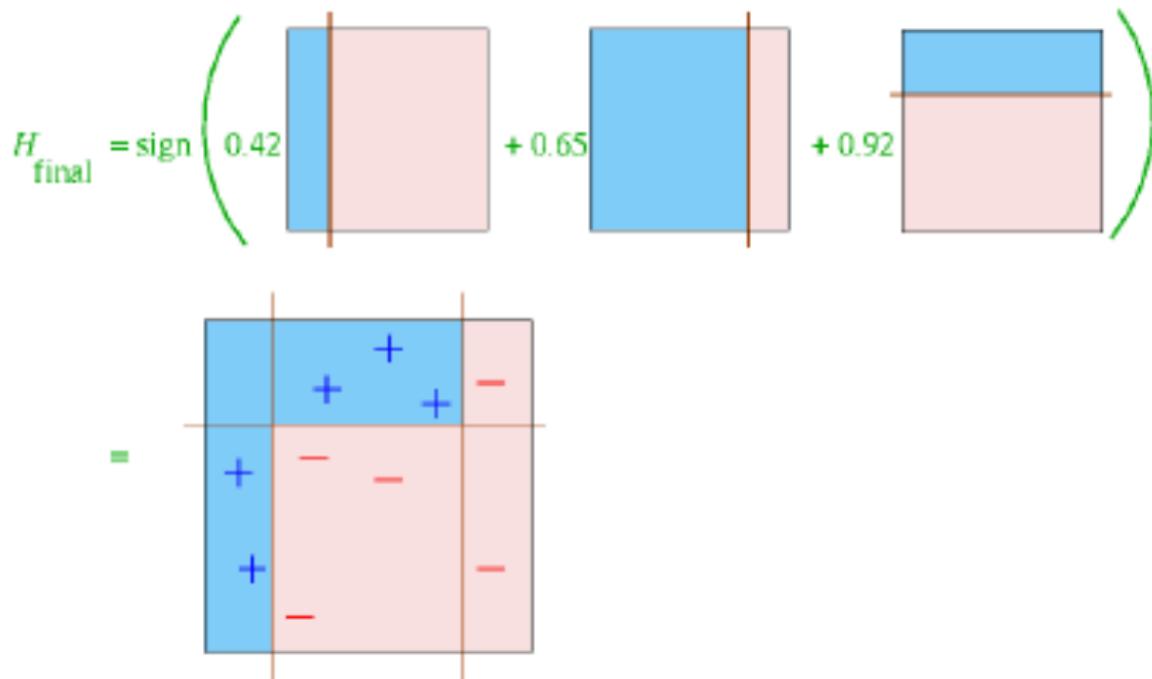
$$\alpha_2 = \frac{1}{2} \log \frac{1 - \epsilon_2}{\epsilon_2}$$

AdaBoost : un exemple ("A Tutorial On Boosting", Freung,Y. & Schapire,R.)



$$\alpha_3 = \frac{1}{2} \log \frac{1 - \epsilon_3}{\epsilon_3}$$

AdaBoost : un exemple ("A Tutorial On Boosting", Freung,Y. & Schapire,R.)



bagging vs boosting

Les 2 méthodes ont en commun une augmentation de la stabilité du classifieur, au prix d'un temps de calcul plus important. Voici quelques généralités à ne pas prendre pour vérités absolues :

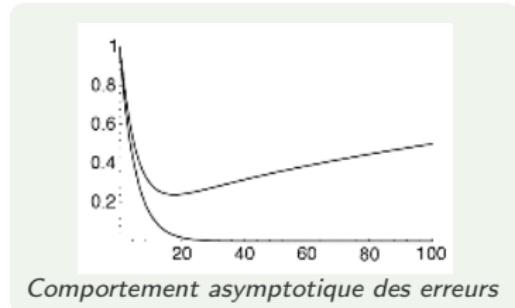
- Bagging est souvent plus rapide que boosting.
- A contrario, la réduction d'erreur est moindre pour bagging.
- Bagging est une méthode qui fonctionne avec des ensembles de classificateurs 'raisonnables' mais peu stables (arbres de décisions par exemple).
- Boosting utilise des classificateurs très simples et donc très faibles.
- AdaBoost n'a plus qu'un paramètre : le nombre de classificateurs que l'on veut mettre.
- Boosting est très sensible au bruit i.e. à un grand nombre de données mal classées (puisque il va leur donner un grand poids).
- Comme les classificateurs faibles ont un biais important, on peut considérer le boosting comme une méthode de réduction du biais.

Boosting : erreur en généralisation

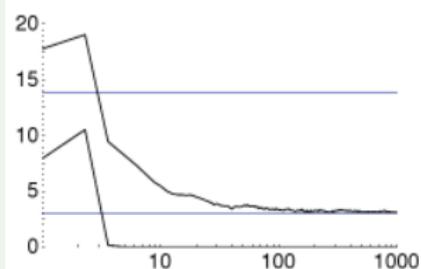
Que se passe-t-il si on augmente beaucoup le nombre de classificateurs faibles ?

Généralement, on s'attend à :

- Erreur d'apprentissage (sur Π_a) qui tend vers 0.
- Erreur en généralisation qui remonte (sur-apprentissage, *overfitting*)



Expérimentalement, pour AdaBoost



- L'erreur de test (meilleure approximation de l'erreur en généralisation) n'augmente pas !
- Même quand l'erreur d'apprentissage est nulle !

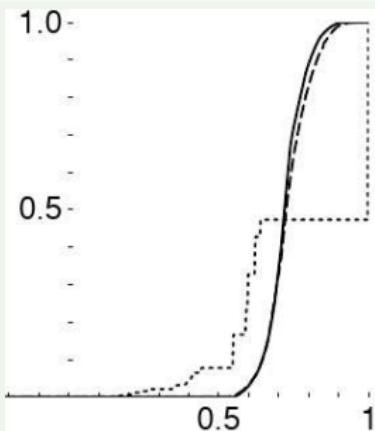
Boosting et marges

Le boosting prend la forme : $C_{\text{boosting}}(x) = \sigma \left(\sum_t \alpha_t \hat{C}_t(x) \right)$ avec $\alpha_t = \log \frac{1}{\beta_t}$.

La marge est définie par $\min_{x \in \Pi_a} (f(x) \cdot C(x))$.

On se souvient également qu'une marge importante indique une robustesse du classifieur, et donc une plus faible erreur en généralisation.

Boosting et marge



- Quand l'erreur de test ne diminue plus, on observe une **augmentation de la marge**.
- Ce comportement est théoriquement prouvable.

Résumé

- Boosting et bagging combinent les résultats d'ensembles de classifieurs.
- A priori, ces méthodes améliorent les résultats en diminuant le biais ou la variance des classifieurs.
- Bagging est principalement une méthode de réduction de la variance, utilisé avec des classifieurs à part entières.
- Boosting se concentre sur les cas particuliers difficiles. Il réduit principalement le biais et augmente la marge. Mais il est sensible au bruit.