GLOBOOST
Combinaisons de moindres généralisés

Produire des moindres généralisés et les combiner au sein d'un vote pondéré

Fabien Torre

GRAppA - Mostrare - Université Charles de Gaulle - Lille 3
Domaine universitaire du « Pont de bois »
B.P. 60149
F-59653 Villeneuve d'Ascq Cedex
fabien.torre@univ-lille3.fr

RÉSUMÉ. *Nous explorons dans cet article l'utilisation de moindres généralisés corrects comme apprenant dans des techniques de* boosting *(Freund* et al.*, 1996). Les premières expérimentations sur des problèmes classiques montrent qu'*ADABOOST *instancié avec un apprenant à base de moindres généralisés obtient des taux d'erreur plus faibles que* C4.5, GLOBO *(Torre, 1999) et* ADABOOST *muni d'un apprenant plus classique. Constatant que notre nouvel apprenant peut être coûteux en temps de calcul, nous proposons un mode de génération des hypothèses qui peut être distribué sur différentes machines et une définition du poids des hypothèses* a posteriori. *Cela aboutit au nouvel algorithme* GLOBOOST *et, à nouveau, les expérimentations montrent que* GLOBOOST *obtient des performances comparables à celles d'*ADABOOST.

ABSTRACT. *The primary goal of this paper is to propose a new learner for boosting algorithms, namely* least general generalization*. First experiments conducted on benchmarks show that* ADABOOST *boosting* least general generalization *obtains smaller error than reference systems. The computation time needed by these experiments leads us to define a method that could be easily distributed. This method, called GloBoost, is able to produce hypotheses independently of one another and then give a weight to each produced hypothesis. New experiments are then conducted and show low error rates for both* ADABOOST *and* GLOBOOST*. Moreover,* GLOBOOST *has the advantage to be naturally distributable to different computers.*

MOTS-CLÉS : *boosting, méthodes d'ensemble, moindres généralisés, algorithmes distribués.*

KEYWORDS: *boosting, ensemble methods, leveraging, least general generalization, distributed algorithms.*

RSTI - RIA – 19/2005. Apprentissage automatique, pages 769 à 797

1. Introduction

Le *boosting* est une technique qui a des origines théoriques ancrées dans le modèle PAC (Valiant, 1984). Dans ce cadre, un résultat fort a été montré : tout algorithme d'apprentissage produisant des hypothèses faisant mieux qu'une procédure de choix aléatoire peut être *boostée* jusqu'à atteindre une erreur aussi faible que voulu et ce avec une probabilité aussi grande que souhaité et en temps polynomial.

Deux preuves de ce résultat ont été proposées (Schapire, 1990; Freund, 1995) ; elles ont pour point commun de définir chacune un algorithme réalisant le *boosting* d'un apprenant faible, toujours dans le cadre PAC. L'idée générale de ces algorithmes est d'utiliser plusieurs fois l'apprenant faible sur des distributions différentes, puis de combiner les différentes hypothèses obtenues au sein d'un vote.

L'algorithme utilisé dans (Freund, 1995) a de plus la particularité de se rapprocher d'un cadre plus réaliste que le cadre PAC (par exemple, en constituant son échantillon au tout début de l'algorithme). Cette démarche a finalement conduit à l'algorithme ADABOOST (Freund *et al.*, 1995; Freund *et al.*, 1997). Son principe est d'associer un poids à chaque exemple, puis de solliciter l'apprenant faible sur ces exemples et ces poids pour obtenir une hypothèse qui classe correctement les exemples de poids forts ; ensuite, les poids des exemples sont modifiés en fonction de cette hypothèse (les poids des bien classés sont diminués et ceux des mal classés augmentés) et le processus itéré.

L'algorithme ADABOOST a permis de tester les idées du *boosting* sur des données réelles (Freund *et al.*, 1996), tout comme BOOSTEXTER la version dédiée à la classification de textes (Schapire *et al.*, 2000). Ces expérimentations, et bien d'autres ensuite, démontrent qu'ADABOOST est capable d'atteindre des erreurs en généralisation très faibles.

Dans cet article, nous nous intéressons à un constructeur d'hypothèses qui, à notre connaissance, n'a jamais été utilisé comme apprenant faible dans un algorithme de *boosting* : le calcul de *moindres généralisés corrects*. Ces moindres généralisés ont montré leur capacité à capturer des sous-concepts, en particulier sur des problèmes très disjonctifs (Webb *et al.*, 1992; Domingos, 1996; Torre, 1999). Outre l'aspect prédictif, les moindres généralisés permettent la production de théories compréhensibles par un expert.

Leur défaut majeur est le temps de calcul qu'ils nécessitent et la question se pose alors de pouvoir distribuer le calcul effectué par ADABOOST sur plusieurs machines. Or, comme nous l'avons signalé, ADABOOST produit une hypothèse en fonction des précédentes ce qui, *a priori*, empêche sa parallélisation. Motivé par ce constat, nous présentons dans cet article une utilisation des moindres généralisés par ADABOOST, ainsi que l'algorithme GLOBOOST. Celui-ci, à l'instar de ADABOOST, produit des hypothèses pondérées dont la combinaison permet la classification de nouveaux exemples. La carac-

téristique notable de GLOBOOST est que les hypothèses ne sont plus produites en fonction les unes des autres mais indépendamment ; en particulier, GLOBOOST n'oriente pas l'apprenant en jouant sur la distribution des exemples. Il ne s'agit donc plus à proprement parler d'un algorithme de *boosting*.

Le plan de l'article est le suivant. Nous revenons à la section 2 sur la technique du *boosting*, les questions qu'elle soulève et les éléments de réponse disponibles. À la section 3, nous rappelons ce qu'est un *moindre généralisé correct* (Torre, 1999) et discutons de son utilisation comme apprenant faible d'ADABOOST. La section 4 présente les résultats expérimentaux obtenus en suivant cette voie ainsi que les performances sur les mêmes données de systèmes plus classiques. Nous évoquons ensuite, section 5, la possibilité de remplacer la génération des hypothèses d'ADABOOST par une méthode de production plus facilement parallélisable et une pondération des hypothèses qui soit détachée de cette génération. À la section 6, nous montrons à nouveau des résultats expérimentaux. Enfin, à la section 7 nous dressons un bilan de ce travail relativement aux travaux précédents et une liste des pistes ouvertes.

2. Quelques éléments de *boosting*

Il ne s'agit pas ici de donner une vision exhaustive des travaux sur le *boosting* (pour cela, le lecteur pourra consulter les états de l'art sur le sujet (Schapire, 2002; Meir *et al.*, 2003)), mais simplement de montrer que certaines questions relatives aux conditions et aux modes d'application de cette technique restent ouvertes.

Dans le cadre PAC (Valiant, 1984), une classe de concepts est dite *apprenable* s'il existe un algorithme capable, pour tout concept de cette classe et toute distribution des exemples, de fournir une hypothèse dont l'erreur par rapport au concept cible peut être aussi proche de 0 que voulu et ce avec une probabilité aussi proche de 1 que souhaité. Précisons qu'en plus l'algorithme doit fournir l'hypothèse en temps polynomial dans les inverses de l'erreur et de la confiance choisies.

Cette notion d'apprenabilité est dite *forte*, par opposition à la version dite *faible* qui apparaît dans (Kearns *et al.*, 1989) : une classe de concepts est dite *faiblement apprenable* s'il existe un algorithme capable, pour tout concept de cette classe et toute distribution des exemples, de fournir une hypothèse dont l'erreur est strictement inférieure à $\frac{1}{2}$ avec une confiance non nulle. Autrement dit, il est simplement exigé que l'apprenant faible fournisse une hypothèse qui fasse mieux qu'un classifieur aléatoire.

Vient ensuite la question du lien entre ces deux notions d'apprenabilité et la réponse est surprenante : elles sont équivalentes. Une première preuve de ce résultat est donnée dans (Schapire, 1990) (voir (Kearns *et al.*, 1994) pour une présentation limpide de cette preuve) puis une seconde dans (Freund, 1995). Chacune de ces deux preuves repose sur un algorithme auquel on fournit un apprenant faible et qui fait appel un nombre polyno-

mial de fois à cet apprenant faible pour finalement produire une hypothèse forte. Dans les deux cas, cette hypothèse forte prend la forme d'un vote qui combine les hypothèses faibles produites par l'apprenant faible.

À la lumière de ces preuves, l'équivalence des deux notions d'apprenabilité revient à dire que, dans le cadre PAC, il est possible de *booster* un apprenant faisant à peine mieux que le hasard pour atteindre une erreur aussi petite que voulue.

Un algorithme pratique nommé ADABOOST et inspiré de la preuve de (Freund, 1995) a finalement été proposé (Freund *et al.*, 1995; Freund *et al.*, 1997). ADABOOST est présenté à l'algorithme 1 dans sa version à deux classes et avec un apprenant faible pouvant fournir des hypothèses qui s'abstiennent. Cette version est reprise de (Schapire *et al.*, 1999) et définit pour chaque hypothèse trois quantités : W_+ la somme des poids des exemples bien classés, W_- la somme des poids des exemples mal classés et W_0 la somme des poids des exemples sur lesquels l'hypothèse s'abstient.

Algorithm 1 ADABOOST

Entrées : n exemples x_i et leurs classes $y_i \in \{-1, +1\}$; T un nombre d'itérations à effectuer ; A un apprenant faible acceptant en entrée un échantillon d'exemples étiquetés et pondérés.

Sortie : H le classifieur final.

for $i = 1$ to n **do**
  $w_i = 1/n$ {initialisation des poids des exemples}
end for
for $t = 1$ to T **do**
  $h_t = A(\{(x_i, y_i, w_i)\})$
  for $b \in \{-, 0, +\}$ **do**
    $W_b = \sum_{i:\text{sign}(y_i h_t(x_i))=b} w_i$
  end for
  $\alpha_t = \dfrac{1}{2}\log\left(\dfrac{W_+ + \frac{1}{2}W_0}{W_- + \frac{1}{2}W_0}\right)$
  $Z_t = \displaystyle\sum_i^n [w_i.\exp(-\alpha_t y_i h_t(x_i))]$
  for $i = 1$ to n **do**
    $w_i = \dfrac{w_i.\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
  end for
end for
return $H(x) = \text{sign}\left(\displaystyle\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Dans la réalité, il est difficile de réunir les conditions théoriques énoncées dans le cadre PAC pour que le *boosting* soit effectivement garanti :

– on n'est jamais certain de disposer d'assez d'exemples et dans la majorité des cas, on ne peut pas comme dans le cadre PAC en réclamer plus à un oracle ;

– on ignore si le nombre d'étapes de *boosting* choisi est suffisant ;

– on est rarement sûr de disposer d'un apprenant faible, c'est-à-dire d'un algorithme fournissant pour toute distribution une hypothèse faisant mieux que l'aléatoire.

De plus, dans la pratique, on pourrait s'attendre à une dégradation de l'erreur en généralisation au cours du *boosting*, un trop grand nombre d'étapes ayant pour effet de sur-spécialiser le système de vote.

Malgré toutes ces réserves, l'observation de très bonnes performances obtenues par ADABOOST sur des données réelles est souvent rapportée. Les conditions imposées par le cadre PAC et les théorèmes de *boosting* n'étant clairement pas remplies, il faut trouver une autre explication à ces bons résultats.

Dans (Breiman, 1996b), l'erreur d'un classifieur est décomposée en biais et variance puis il est avancé que, comme toute méthode de type *arcing* (*adaptively resample and combine*) et comme le *bagging* (Breiman, 1996a), ADABOOST diminuerait la composante due à la variance. Cet argument est battu en brèche dans (Freund *et al.*, 1998) où sont présentées des expérimentations où l'on voit ADABOOST faire chuter à la fois les deux composantes de l'erreur mais toujours sans explication du phénomène.

Une observation sur les algorithmes de *boosting* est souvent rapportée : l'erreur en généralisation décroît encore après que l'erreur en apprentissage soit stable ou même nulle. L'explication est que même si tous les exemples d'apprentissage sont déjà bien classés, la poursuite du *boosting* tend à maximiser les *marges* (Schapire *et al.*, 1998). Dans le cadre des techniques de vote, la marge d'un exemple est la somme des poids des hypothèses couvrant cet exemple : le signe indique la classe prédite par les votants, tandis que la valeur absolue traduit une certaine confiance dans cette classification. Une interprétation géométrique consiste à voir la marge d'un exemple comme sa distance à la séparatrice définie par le classifieur. D'un point de vue théorique, des bornes sur l'erreur en généralisation ont été établies qui font intervenir les marges des exemples d'apprentissage (Schapire *et al.*, 1998; Koltchinskii *et al.*, 2002) : plus ces marges sont grandes, plus la borne est petite. À la suite de cette *théorie de la marge*, certains ont cherché à maximiser explicitement la marge (Rätsch *et al.*, 2002; Rätsch *et al.*, 2003; Grove *et al.*, 1998) et ont dû constater que leurs systèmes étaient moins performants qu'ADABOOST. Sur cette thématique, citons encore (Harries, 1999) qui parvient à construire un classifieur donnant à chacun des exemples d'apprentissage une marge maximale (c'est-à-dire un marge de 1) mais ce classifieur a des performances toujours moins bonnes qu'ADABOOST. Cette dernière expérience permet de rejeter l'explication des performances d'ADABOOST re-

posant uniquement sur la maximisation de la marge : cette caractéristique est sans doute importante mais ne peut, à elle seule, expliquer les résultats observés.

Nous pouvons également nous interroger sur la signification des poids α_t qu'ADABOOST associe aux hypothèses produites. À la vue de l'algorithme, il s'agit d'une valeur déterminée sur une distribution artificielle, elle-même fonction des échecs et réussites des hypothèses précédentes. Cependant, nous avons noté lors d'expériences sur des données très simples que l'erreur en généralisation diminuait encore alors que l'apprenant faible avait déjà fourni toutes les hypothèses possibles. Autrement dit, lorsqu'une hypothèse apparaît plusieurs fois, elle vote finalement avec un poids, cumul de tous ses α_t, qui a peut-être un caractère plus absolu. À cet égard, les études sur la dynamique de ADABOOST (Rudin *et al.*, 2004) montrent que, dans certains cas, l'algorithme converge vers la solution qui maximise les marges. Cependant, il est montré également qu'une convergence vers une solution non optimale du point de vue des marges peut survenir (ce qui affirme à nouveau que la maximisation des marges ne suffit pas à garantir la meilleure erreur en généralisation). Dernière situation évoquée par ces travaux : la convergence peut ne pas ne se produire du tout, auquel cas des cycles sont observés. Au final, la possibilité d'approcher les valeurs des α_t par un processus non adaptatif est encore une question ouverte.

Une dernière question est celle de l'apprenant faible : comment bien le choisir ? À nouveau les éléments de réponse sont limités. Les auteurs d'ADABOOST indiquent que la définition simple et claire d'un apprenant faible est perdue avec le passage du cadre PAC aux problèmes réels et que la seule caractérisation d'un bon apprenant dont nous disposions est : *an algorithm that gives small errors when run under Adaboost* (Freund *et al.*, 1998). En particulier, faut-il autoriser ou même encourager l'apprenant faible à se tromper sur une partie des exemples d'apprentissage (les exemples de poids faibles, bien classés par les hypothèses produites précédemment) ? La réponse n'est pas évidente, surtout si l'on prend en compte qu'ADABOOST traite de manière privilégiée des données non bruitées (son comportement naturel sur des données bruitées est de s'acharner sur les exemples difficiles à classer, donc sur le bruit).

Dans cet article, nous proposons justement d'utiliser un apprenant faible qui ne produit que des hypothèses correctes : celles-ci peuvent donc s'abstenir sur une partie des exemples mais en aucun cas se tromper sur un exemple. Il s'agit de *moindres généralisés maximalement corrects*.

3. Booster des moindres généralisés avec AdaBoost

3.1. *Moindres généralisés corrects*

Comme son nom l'indique, le moindre généralisé de deux exemples est l'hypothèse la plus spécifique possible qui couvre ces deux exemples. En attribut-valeur, cette hypothèse est unique et peut être simplement définie comme suit :

– si les deux exemples partagent la même valeur d'un attribut catégoriel, celle-ci est conservée dans le moindre généralisé ; si ces valeurs sont distinctes, le moindre généralisé fait apparaître une valeur indéterminée pour cet attribut ;

– pour un attribut continu, on construit le plus petit intervalle incluant les deux valeurs présentes dans les exemples à généraliser.

Cette procédure apparaît clairement sur l'exemple suivant :

	âge	**fumeur**	**sexe**	***classe***
$\mathbf{e_1}$	25	non	homme	positif
$\mathbf{e_2}$	35	non	femme	positif
$\mathbf{g_1}$	$[25, 35]$	non	?	positif

Naturellement, cette opération peut se généraliser pour s'appliquer à deux hypothèses ou, cas qui nous intéresse le plus, à une hypothèse et un exemple :

	âge	**fumeur**	**sexe**	***classe***
$\mathbf{g_1}$	$[25, 35]$	non	?	positif
$\mathbf{e_3}$	30	oui	homme	positif
$\mathbf{g_2}$	$[25, 35]$	?	?	positif
$\mathbf{e_4}$	40	non	femme	positif
$\mathbf{g_3}$	$[25, 40]$	?	?	positif

Du point de vue de l'apprentissage supervisé, il est intéressant de généraliser ensemble des exemples d'une même classe, en prenant garde à ne pas couvrir d'exemples d'autres classes. Ce calcul, qui fournit la caractérisation d'un sous-concept au sein d'une classe, est celui d'un *moindre généralisé maximalement correct* et est décrit formellement par l'algorithme 2 : étant donné un exemple-graine, on essaye de généraliser, un à un, les exemples de la même classe avec comme critère d'acceptation la non couverture d'exemples d'autres classes (*correction*). La généralisation produite est *maximalement correcte* dans le sens où plus aucun exemple d'apprentissage ne peut lui être ajoutée sans perdre la correction. Par conséquent, à moins que les exemples permettent une solution purement conjonctive, l'hypothèse construite ne couvre pas tous les exemples de

Algorithm 2 MG_Correct_v1

Entrées : s un exemple graine, $P = \{p_1, \ldots, p_n\}$ un ensemble *ordonné* de n exemples de la même classe que la graine, N un ensemble de contre exemples. L'algorithme utilise l'opération élémentaire de `MoindreGénéralisé` entre une hypothèse et un exemple, ainsi qu'un test de subsomption noté $\succeq$ permettant de décider si une hypothèse couvre ou non un exemple (ces deux opérations sont à définir en fonction du langage de description choisi).

Sortie : g une généralisation de P, maximalement correcte par rapport à N.

$g = s$
for $i = 1$ to n **do**
  $g' =$ `MoindreGénéralisé`(g, p_i) {Généralisation entre deux hypothèses.}
  if $(\forall e \in N : g' \not\succeq e)$ **then**
    $g = g'$ {Si g' est correcte, elle devient la généralisation courante.}
  end if
end for
return g

la classe cible. Notons également que cette hypothèse est fonction de l'ordre dans lequel les exemples sont testés pour l'ajout.

L'exemple suivant montre trois exécutions possibles de l'algorithme 2 (les exemples qui provoquent une perte de correction pour la généralisation courante sont barrés) :

graine	autres positifs	$\rightarrow$	paquet maximalement correct	règle
p_1	$p_5\ p_8\ \not{p_2}\ p_{14} \ldots$	$\rightarrow$	$\{p_1, p_5, p_8, p_{14}, \ldots\}$	g_1
p_2	$\not{p_{14}}\ p_3\ \not{p_1}\ p_{12} \ldots$	$\rightarrow$	$\{p_2, p_3, p_{12}, \ldots\}$	g_2
p_3	$p_7\ \not{p_4}\ \not{p_{18}}\ \not{p_{12}} \ldots$	$\rightarrow$	$\{p_3, p_7, \ldots\}$	g_3

On y voit l'importance de l'ordre de présentation des exemples et également que la contrainte de correction amène des généralisations qui, prises isolément, ne couvrent pas tous les exemples positifs. Autrement dit, une telle généralisation ne peut conclure que sur une seule classe, la classe des exemples sur lesquels elle a été construite. Pour tous les autres exemples, ceux de la même classe qui ne sont pas couverts et ceux d'autres classes, la généralisation ne fournit pas de classe, elle s'abstient.

Cette brique de base a déjà été utilisée dans l'algorithme GLOBO (Torre, 1999) (cf. algorithme 3). Dans ce cas, chaque exemple joue successivement le rôle de graine et pour chacun, la liste des exemples de même classe est mélangée aléatoirement. Puis GLOBO opère une couverture minimale pour ne garder que le nombre minimum d'hypothèses permettant de couvrir tous les exemples. Le but de la couverture minimale dans GLOBO est de produire finalement un ensemble réduit de règles qui soit appréhendable par un

Algorithm 3 GLOBO

Entrées : n exemples x_i et leurs classes $y_i \in \{-1, +1\}$.
Sortie : H un ensemble de règles.

$H' = \emptyset$
for $i = 1$ to n **do**
  $P = \{x_j | y_j = y_i \wedge i \neq j\}$
  $N = \{x_j | y_j \neq y_i\}$
  mélanger P ***aléatoirement***
  $h_i =$ `MG_Correct_v1`(x_i,P,N) {Appel à l'algorithme 2}
  ajouter h_i à H'
end for
$H = \emptyset$
while $(\exists x_i, \forall h_j \in H, h_j \not\succeq x_i)$ **do**
  $h = \text{ArgMax}_{h_i \in H'} |\{x_j : h_i \succeq x_j \wedge \nexists h_k \in H : h_k \succeq x_j\}|$
  ajouter h à H
end while
return H

expert humain. Cette approche s'est révélée pertinente puisque GLOBO a remporté le PTE Challenge (Srinivasan *et al.*, 1997; Srinivasan *et al.*, 1999) dans la catégorie des systèmes produisant une théorie compréhensible.

Dans cet article, notre volonté est d'abandonner la compréhensibilité pour gagner en capacité prédictive et cela en utilisant le calcul de moindre généralisé correct comme apprenant faible dans des techniques de *boosting*.

3.2. *Instanciation de l'algorithme* ADABOOST

Nous proposons dans cette section une nouvelle version du calcul de moindres généralisés corrects destinée à devenir un apprenant faible pour ADABOOST. Cela implique de prendre en compte les poids des exemples, autrement dit de favoriser la couverture des exemples de poids élevé. Avec cet objectif, l'idée de notre apprenant faible est de réutiliser l'algorithme 2 en triant au préalable les exemples candidats à la généralisation par poids décroissants. Précisons immédiatement que, quels que soient les poids, nous ne relâchons pas notre critère de correction : la généralisation produite ne couvrira aucun contre-exemple, même de poids faible. Cette nouvelle version est présentée à l'algorithme 4.

Cet apprenant peut être fourni en l'état à ADABOOST (algorithme 1) dont l'utilisation ne nécessite pas d'autre information. Remarquons simplement que la formule de α_t (poids des hypothèses) utilisée dans ADABOOST se simplifie en prenant en compte que

Algorithm 4 MG_Correct_v2 (apprenant faible)

Entrées : n exemples (x_i, y_i, w_i) avec leurs étiquettes et poids.
Sortie : g une généralisation maximalement correcte d'exemples de poids élevés.
target = classe choisie au hasard
seed = l'exemple de la classe *target* ayant le poids le plus fort
$P = \{x_i | y_i = \textit{target}, x_i \neq \textit{seed}\}$
$N = \{x_i | y_i \neq \textit{target}\}$
trier P ***par poids décroissants***
return `MG_Correct_v1`(*seed*,P,N) {Appel à l'algorithme 2}

les moindres généralisés ainsi calculés ne font pas d'erreur de classification (ils classifient bien ou s'abstiennent). En effet, $W_- = 0$ et par suite $W_0 = 1 - W_+$ et finalement

$$\alpha_t = \frac{1}{2}\log\left(\frac{1+W_+}{1-W_+}\right)$$

La section suivante présente les résultats de ADABOOST ainsi instancié, algorithme que nous appelerons ADABOOSTMG dans la suite (ou en plus concis ABMG, dans les tableaux de résultats).

4. Expérimentations avec ADABOOST

4.1. *Protocole expérimental*

ADABOOSTMG est maintenant comparé aux algorithmes suivants : le classique C4.5 ; GLOBO, système à base de moindres généralisés mais sans *boosting* présenté à l'algorithme 3 ; et ADABOOST (algorithme 1) utilisant comme apprenant faible des *decision stumps*, arbres de décision limités à un seul niveau, utilisés comme hypothèses faibles par exemple dans (Freund *et al.*, 1996). Dans ce cas, le travail de l'apprenant faible consiste simplement à choisir parmi toutes les hypothèses possibles, celle qui se comporte le mieux vis-à-vis de la distribution courante. Ce dernier système a été simulé à l'aide du système ADTREE (De Comité *et al.*, 2001) en mode BOOSTEXTER et sera désigné dans la suite par ADABOOSTDS (ou en plus concis ABDS, dans les tableaux de résultats).

Ces systèmes sont évalués sur vingt problèmes classiques de l'UCI (Blake *et al.*, 1998), représentant un éventail assez large des problèmes que l'on peut rencontrer : attributs tous continus, tous discrets, ou mélange des deux types, avec deux classes à prédire ou plus, avec ou sans valeurs manquantes, hautement disjonctif ou pas.

Pour chacun de ces problèmes, on utilise une validation croisée 10 fois, le découpage étant le même pour toutes les expérimentations[1]. Finalement, pour chaque problème et pour chaque système, on mesure le taux d'erreur moyen sur les 10 blocs de test. Précisons que les systèmes stochastiques (GLOBO et ADABOOSTMG) sont exécutés dix fois sur chaque bloc et c'est la moyenne sur ces dix exécutions qui est utilisée pour mesurer l'erreur du système.

4.2. *Résultats pour 100 étapes de* boosting

Les taux d'erreur observés pour 100 étapes de *boosting* sont présentés dans le tableau 1(a) où l'on a en plus fait apparaître pour chaque système la moyenne des taux d'erreur sur l'ensemble des problèmes. Pour chacun des problèmes, l'erreur la plus faible est distinguée en gras.

Il y apparaît d'emblée que les systèmes à base d'arbres de décision (C4.5 et ADABOOSTDS) obtiennent les meilleurs résultats.

Le tableau 1(b) explicite cette impression en donnant les résultats des confrontations des méthodes prises deux à deux. Chaque ligne de ce tableau contient les résultats d'une méthode particulière : chaque case fournit le nombre de problèmes sur lesquels cette méthode a obtenu une erreur plus faible que la méthode en colonne, ainsi que le nombre de fois où la méthode considérée s'est montrée moins performante. Par souci de clarté, le nombre de nuls n'est pas rapporté mais peut être retrouvé aisément. Ces données apparaissent en italique si l'algorithme en ligne est meilleur que celui en colonne. Enfin, la dernière colonne totalise le nombre de victoires et de défaites obtenues par la méthode considérée.

À l'issue de ces expérimentations utilisant 100 tours de boosting, les méthodes à base de moindres généralisés sont battues. Notons cependant que ADABOOSTMG surpasse GLOBO sur 14 des 20 problèmes étudiés.

4.3. *Résultats pour 1 000 étapes de* boosting

Le tableau 2 présente les résultats d'expérimentations identiques où le nombre d'étapes de *boosting* est passé de 100 à 1 000 ; il amène les constatations suivantes :

– l'erreur de ADABOOSTMG diminue notablement sur quasiment tous les problèmes lorsque l'on passe de 100 étapes de *boosting* à 1 000 (l'erreur globale chute de 18,16 % à 12,70 %) ;

1. Les fichiers de la validation croisée ainsi que des résultats complémentaires sont disponibles à l'adresse : `http://www.grappa.univ-lille3.fr/~torre/Recherche/Experiments/`.

(a) Taux d'erreur

Problèmes	**C4.5**	**GLOBO**	**ADABOOSTDS**	**ADABOOSTMG**
audiology	18,20	20,76	**13,43**	25,78
breast-cancer	4,87	3,89	4,16	**3,02**
car	**7,69**	10,17	13,42	17,92
cmc	48,07	50,60	**44,80**	56,32
crx	**14,79**	16,50	15,80	17,82
dermatology	6,23	8,51	**3,51**	6,58
ecoli	15,89	23,88	**15,59**	23,24
glass	28,72	5,57	23,62	**4,70**
hepatitis	20,70	20,09	**17,64**	17,88
horse-colic	**13,63**	22,29	18,44	20,90
house-votes-84	**3,22**	7,39	4,62	56,15
ionosphere	7,96	8,74	**6,78**	8,51
iris	**5,33**	7,47	**5,33**	5,93
pima	29,29	27,04	**25,13**	29,68
promoters	18,17	22,60	**6,83**	12,93
sonar	28,97	31,09	**16,86**	25,25
tic-tac-toe	14,40	1,11	13,78	**0,23**
vowel	21,21	23,99	**20,71**	21,30
wine	8,83	8,94	14,18	**5,41**
zoo	7,51	5,55	4,25	**3,67**
Moyennes	*16,18*	*16,31*	***14,45***	*18,16*

(b) Confrontations

	C4.5	**GLOBO**	**ABDS**	**ABMG**	***Bilan***
C4.5	-	*14-6*	5-14	*12-8*	31-28
GLOBO	6-14	-	5-15	6-14	17-43
ABDS	*14-5*	*15-5*	-	*15-5*	**44-15**
ABMG	8-12	*14-6*	5-15	-	27-33

TABLEAU 1. *Résultats de* C4.5, GLOBO, ADABOOSTDS *et* ADABOOSTMG *sur vingt problèmes de l'UCI ; le nombre d'étapes de boosting vaut 100 pour* ADABOOSTDS *et* ADABOOSTMG

(a) Taux d'erreur

	ADABOOSTDS		ADABOOSTMG	
Problèmes	**100**	**1 000**	**100**	**1 000**
audiology	**13,43**	15,79	25,78	19,22
breast-cancer	4,16	4,16	**3,02**	3,10
car	13,42	13,36	17,92	**9,74**
cmc	**44,80**	44,87	56,32	49,61
crx	15,80	16,53	17,82	**14,47**
dermatology	**3,51**	4,38	6,58	4,63
ecoli	**15,59**	**15,59**	23,24	19,25
glass	23,62	22,53	4,70	**4,59**
hepatitis	**17,64**	18,23	17,88	18,39
horse-colic	18,44	21,71	20,90	**18,42**
house-votes-84	**4,62**	5,31	56,15	6,12
ionosphere	**6,78**	12,19	8,51	7,44
iris	**5,33**	**5,33**	5,93	**5,33**
pima	25,13	25,39	29,68	**25,13**
promoters	**6,83**	18,67	12,93	7,00
sonar	16,86	**15,03**	25,25	23,34
tic-tac-toe	13,78	1,67	0,23	**0,00**
vowel	20,71	14,65	21,30	**9,30**
wine	14,18	18,07	**5,41**	5,73
zoo	4,25	5,25	3,67	**3,21**
Moyennes	*14,45*	*14,94*	*18,16*	***12,70***

	$\mathbf{ABDS_{100}}$	$\mathbf{ABDS_{1000}}$	$\mathbf{ABMG_{100}}$	$\mathbf{ABMG_{1000}}$	***Bilan***
$\mathbf{ABDS_{100}}$	-	*12-5*	*15-5*	9-10	36-20
$\mathbf{ABDS_{1000}}$	5-12	-	*11-9*	7-12	23-33
$\mathbf{ABMG_{100}}$	5-15	9-11	-	3-17	17-43
$\mathbf{ABMG_{1000}}$	*10-9*	*12-7*	*17-3*	-	**39-19**

(b) Confrontations

	C4.5	**GLOBO**	$\mathbf{ABDS_{100}}$	$\mathbf{ABMG_{1000}}$	***Bilan***
C4.5	-	*14-6*	5-14	6-13	25-33
GLOBO	6-14	-	5-15	0-20	11-49
$\mathbf{ABDS_{100}}$	*14-5*	*15-5*	-	9-10	38-20
$\mathbf{ABMG_{1000}}$	*13-6*	*20-0*	*10-9*	-	**43-15**

TABLEAU 2. ADABOOSTMG *et* ADABOOSTDS *pour 100 et 1 000 étapes de* boosting

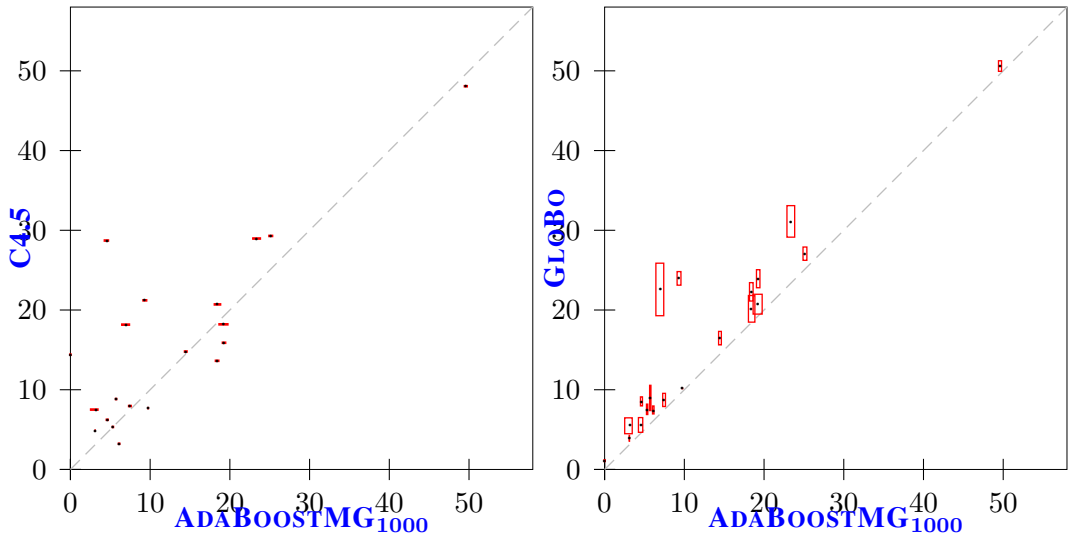

FIGURE 1. ADABOOSTMG *opposé à deux méthodes non boostées :* C4.5 *et* GLOBO

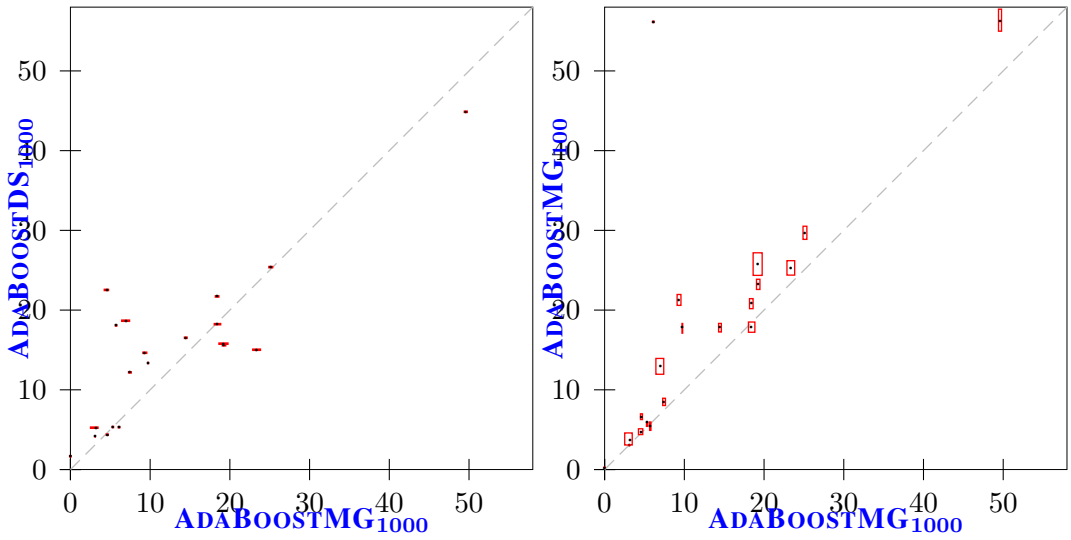

FIGURE 2. ADABOOSTMG$_{1\,000}$ *contre* ADABOOSTDS$_{1\,000}$ *et* ADABOOSTMG$_{100}$

– ADABOOSTMG surpasse ou égale C4.5 sur 14 problèmes sur 20 et cette supériorité est aussi apparente sur les taux d'erreur moyens (12,70 % contre 16,18 %) ;

– GLOBO est battu sur tous les problèmes par ADABOOSTMG et l'on mesure ainsi ce que nous avons gagné en abandonnant la compréhensibilité des théories produites par GLOBO (en moyenne, on passe de 16,31 % d'erreurs à 12,70 %) ;

– ADABOOSTDS supporte moins bien que ADABOOSTMG le passage à 1 000 étapes de *boosting* et tous les indicateurs de performance se détériorent par rapport aux résultats précédents ; que ce soit avec 100 ou 1 000 étapes de *boosting*, ADABOOSTDS est maintenant battu par ADABOOSTMG utilisant 1 000 étapes de *boosting*.

Un dernier moyen de comparer deux méthodes d'apprentissage vis-à-vis d'un ensemble de problèmes consiste à construire un graphique où chaque problème est représenté par un point dont les coordonnées correspondent aux erreurs respectives obtenues par les deux compétiteurs. En plus de ces points, nous pouvons faire apparaître les écarts types observés pour les méthodes stochastiques. Dans la suite, la méthode testée est systématiquement placée en abscisse et sa supériorité par rapport à la seconde méthode (en ordonnée) est mesurée par le nombre de points au-dessus de la droite $y = x$, ainsi que par les distances des points à cette droite.

Par exemple, les figures 1(a) et 1(b) montrent de tels graphiques pour ADABOOSTMG opposé à C4.5 puis à GLOBO. La première montre que la supériorité de ADABOOSTMG sur C4.5 vaut même en prenant en compte la variabilité du *boosting* de moindres généralisés. La seconde confirme que, sur tous nos problèmes, ADABOOSTMG surpasse GLOBO avec l'information supplémentaire que ADABOOSTMG présente des écarts types bien plus réduits que GLOBO.

La figure 2(a) illustre la confrontation ADABOOSTMG *vs.* ADABOOSTDS et, enfin, la figure 2(b) le passage de 100 à 1 000 étapes de *boosting* : outre la réduction des taux d'erreur, la réduction des écarts types est également observée.

Ces résultats nous encouragent à augmenter encore le nombre d'étapes de *boosting* pour continuer à faire baisser l'erreur. Malheureusement, le calcul de moindres généralisés corrects est coûteux : pour chacun, il faut essayer d'ajouter les exemples de la classe cible un à un et chacun de ces ajouts doit être validé sur l'ensemble des contre-exemples.

Ainsi, le plus long des apprentissages avec 1 000 étapes de *boosting* et l'utilisation de moindres généralisés prend une heure et vingt minutes sur une machine i686 à 2,5 Ghz pour une base qui contient moins de 1 000 exemples. Ce temps n'est pas prohibitif en soi et est sans doute comparable au temps requis par bon nombre d'apprenants faibles. Cependant, notre protocole implique de réaliser 100 fois cet apprentissage (10 blocs de validation croisée et 10 exécutions de chaque bloc) ce qui cette fois est difficile à mener dans un temps raisonnable.

Il est donc impératif, pour que la méthode soit viable, de rendre l'apprentissage plus rapide. Nous avons choisi pour cela de paralléliser la génération des hypothèses et donc de différer l'affectation de poids à ces hypothèses. Cette étude est l'objet de la section suivante.

5. Produire et combiner des moindres généralisés sans ADABOOST

Nous l'avons vu, la production d'un moindre généralisé correct est de complexité quadratique dans le nombre d'exemples disponibles et cela n'est pas acceptable pour un apprenant faible destiné à être appelé de nombreuses fois dans un algorithme de type ADABOOST.

Nous aurions pu, pour réduire le temps de calcul, échantillonner et distribuer les données à différentes unités de calcul. Nous avons préféré distribuer, non pas les données, mais le calcul des hypothèses, lequel se fera toujours sur l'ensemble complet des données disponibles. Cela signifie que nous ne pouvons plus jouer sur les poids des exemples pour orienter la construction d'une hypothèse en fonction du comportement de celles obtenues précédemment ; en l'occurrence les hypothèses sont produites de manière aléatoire. À proprement parler, il ne s'agit donc plus ici de *boosting*. GLOBOOST est sans doute plus proche du *bagging*, à ceci près que chaque hypothèse est apprise sur le même ensemble de données.

En résumé, nous nous ramenons ici à une production distribuée des hypothèses, ensuite combinées au sein d'un vote pondéré. Cela nécessite de savoir :

– produire les hypothèses indépendamment les unes des autres ;

– affecter un poids à chacune de ces hypothèses *a posteriori*.

Nous commençons par le deuxième point en attribuant de nouveaux poids aux hypothèses disponibles, quel que soit leur mode de production.

5.1. *Affectation de poids aux hypothèses produites*

Le premier de ces poids, dans la suite repéré par le nom *adalike*, cherche à imiter le mode de calcul des poids d'ADABOOST. Comme ADABOOST, nous commençons par définir le poids d'un exemple x_i, cela à partir de T_i le nombre de fois où cet exemple est couvert par les hypothèses produites :

$$w_i = \exp(-T_i)$$

Cette formule fait référence à ADABOOST (algorithme 1) qui multiplie le poids d'un exemple par $\exp(-\alpha_t)$ pour chaque hypothèse h_t couvrant l'exemple. Une fois ces poids attribués aux exemples, on les normalise en divisant chacun par $Z = \sum_i w_i$. Puis nous

définissons le poids d'une hypothèse h. W_+ est défini comme la somme des poids des exemples couverts par h et n est le nombre de fois où cette hypothèse est apparue pendant la production. Finalement, le poids *adalike* de h est calculé comme suit :

$$\alpha(h) = \frac{n}{2}\log\left(\frac{1+W_+}{1-W_+}\right)$$

La valeur de n intervient pour simuler le cumul des poids effectué par ADABOOST lorsqu'une même hypothèse apparaît plusieurs fois.

Nous considérons également les poids suivants, plus simples :

– *covering* : chaque hypothèse distincte vote avec un poids qui correspond au nombre d'exemples qu'elle couvre dans l'ensemble d'apprentissage (dans notre cas, les hypothèses sont correctes par construction et les exemples couverts par une hypothèse sont donc tous de la même classe) ;

– *frequency* : chaque hypothèse distincte vote avec un poids qui correspond à son nombre d'apparitions pendant la phase de production ;

– *uniform* : toutes les hypothèses distinctes votent avec un poids égal.

5.2. *Production d'hypothèses sans* ADABOOST

L'étape suivante est de trouver une alternative à ADABOOST pour produire des hypothèses. Nous proposons pour cela l'algorithme GLOBOOST. Comme nous l'avons précisé, l'objectif est, à terme, de partager ce travail de génération entre plusieurs machines : cela conduit à abandonner le caractère adaptatif d'ADABOOST et en particulier l'utilisation de poids sur les exemples. Ainsi, en jouant simplement sur l'ordre de généralisation des exemples, GLOBOOST produit des moindres généralisés corrects de façon purement stochastique et, par conséquent, sans aucun lien de dépendance entre eux (algorithme 5).

6. Évaluation expérimentale de GLOBOOST

Dans cette section, nous évaluons expérimentalement les idées de la section précédente : génération des hypothèses par GLOBOOST en remplacement d'ADABOOST et affectation de poids *a posteriori*. Le protocole et les données sont les mêmes qu'à la section 4.1.

6.1. *Évaluation expérimentale des poids proposés*

Dans un premier temps, nous conservons l'algorithme ADABOOST comme générateur d'hypothèses. Après la phase de *boosting*, nous obtenons un ensemble d'hypothèses

(a) Taux d'erreur

Problèmes	**adaboost**	**adalike**	**covering**	**frequency**	**uniform**
audiology	25,78	**24,14**	28,32	25,97	26,25
breast-cancer	3,02	4,17	**2,96**	4,63	4,65
car	17,92	**17,89**	18,14	18,39	18,39
cmc	56,32	56,31	**56,27**	56,62	56,62
crx	17,82	**17,12**	17,77	20,45	20,43
dermatology	6,58	**6,55**	7,20	7,18	7,32
ecoli	**23,24**	24,21	25,25	25,37	25,32
glass	**4,70**	**4,70**	4,83	**4,70**	4,83
hepatitis	17,88	**15,65**	18,98	20,67	20,72
horse-colic	**20,90**	21,06	21,24	23,16	23,40
house-votes-84	56,15	37,59	14,94	56,38	**6,20**
ionosphere	8,51	27,95	9,61	**8,14**	**8,14**
iris	5,93	**5,07**	6,00	6,27	6,27
pima	29,68	**29,61**	30,23	32,46	32,46
promoters	**12,93**	16,12	16,45	15,50	15,33
sonar	25,25	40,70	25,28	**24,26**	**24,26**
tic-tac-toe	**0,23**	**0,23**	**0,23**	0,71	1,13
vowel	21,30	22,60	21,38	**20,14**	**20,14**
wine	5,41	16,40	**5,35**	5,46	**5,35**
zoo	3,67	**3,52**	5,07	4,20	5,40
Moyennes	*18,16*	*19,58*	*16,78*	*19,03*	***16,63***

(b) Confrontations

	adaboost	**adalike**	**covering**	**frequency**	**uniform**	***Bilan***
adaboost	-	8-10	*14-5*	*16-3*	*15-5*	**53-23**
adalike	*10-8*	-	*12-7*	*14-5*	*14-6*	50-26
covering	5-14	7-12	-	*12-8*	*12-6*	36-40
frequency	3-16	5-14	8-12	-	*8-5*	24-47
uniform	5-15	6-14	6-12	5-8	-	22-49

TABLEAU 3. *Résultats selon les poids attribués aux moindres généralisés corrects produits par* ADABOOSTMG *pour 100 étapes de* boosting

(a) Taux d'erreur

Problèmes	**adaboost**	**adalike**	**covering**	**frequency**	**uniform**
audiology	19,22	**19,17**	26,89	20,66	21,72
breast-cancer	3,10	10,36	**3,00**	3,96	3,94
car	9,74	**9,56**	9,74	15,83	10,20
cmc	49,61	49,24	**48,98**	51,25	51,03
crx	**14,47**	26,50	14,60	21,22	20,71
dermatology	**4,63**	12,93	5,92	5,51	7,15
ecoli	**19,25**	23,92	21,02	22,46	21,62
glass	**4,59**	8,65	6,33	**4,59**	6,33
hepatitis	**18,39**	22,74	19,28	21,21	23,06
horse-colic	18,42	**18,17**	19,54	19,80	18,86
house-votes-84	6,12	34,84	**5,07**	30,06	5,80
ionosphere	**7,44**	57,79	9,23	7,64	7,56
iris	**5,33**	6,67	6,00	6,00	6,80
pima	25,13	**24,75**	25,55	29,37	29,39
promoters	**7,00**	29,38	12,83	8,02	13,63
sonar	23,34	49,52	22,53	**21,83**	**21,83**
tic-tac-toe	**0,00**	**0,00**	**0,00**	0,72	2,71
vowel	9,30	34,43	9,85	9,17	**9,16**
wine	5,73	45,90	**4,62**	5,23	4,67
zoo	**3,21**	3,55	6,78	4,59	5,58
Moyennes	***12,70***	*24,40*	*13,89*	*15,46*	*14,59*

(b) Confrontations

	adaboost	**adalike**	**covering**	**frequency**	**uniform**	***Bilan***
adaboost	-	*14-5*	*13-6*	*16-3*	*16-4*	**59-18**
adalike	5-14	-	5-14	7-13	9-11	26-52
covering	6-13	*14-5*	-	*11-8*	*13-6*	44-32
frequency	3-16	*13-7*	8-11	-	9-10	33-44
uniform	4-16	*11-9*	6-13	*10-9*	-	31-47

TABLEAU 4. *Résultats selon les poids attribués aux moindres généralisés corrects produits par* ADABOOSTMG *pour 1 000 étapes de* boosting

Algorithm 5 GLOBOOST

Entrées : n exemples (x_i, y_i) avec étiquettes ; T un nombre d'itérations.
Sortie : H le classifieur final.

```
for t = 1 to T do
    target = classe choisie au hasard
    seed = un exemple de la classe target choisi au hasard
    P = {x_i | y_i = target, x_i ≠ seed}
    N = {x_i | y_i ≠ target}
    mélanger P aléatoirement
    h_t = MG_Correct_v1(seed,P,N) {Appel à l'algorithme 2, page 776}
end for
for all t tel que 1 ≤ t ≤ T do
    α_t = poids(h_t) {fonction à instancier}
end for
return H(x) = sign ( Σ_{t=1}^{T} α_t.h_t(x) )
```

avec leurs poids attribués par ADABOOST et nous leurs affectons les poids supplémentaires définis précédemment. On obtient ainsi des systèmes de vote distincts même si tous sont fondés sur les mêmes votants.

Notre intuition, forgée par les différents travaux qui cherchent à redéfinir les poids pour maximiser la marge (Schapire *et al.*, 1998; Rätsch *et al.*, 2002; Rätsch *et al.*, 2003; Grove *et al.*, 1998; Harries, 1999), est que ADABOOST a un comportement quasi optimal dans le réglage des poids des hypothèses. Il s'agit donc ici de quantifier ce que nous perdons en abandonnant les poids des hypothèses fixés par ADABOOST.

Les résultats pour 100 (respectivement 1 000) étapes de *boosting* sont présentés au tableau 3 (respectivement tableau 4). La première constatation est que ADABOOST reste le meilleur pour fixer les poids des hypothèses qu'il produit, en particulier lorsque le nombre d'étapes de *boosting* augmente.

A contrario, notre imitation *adalike* a un meilleur comportement pour un nombre d'étapes de *boosting* de 100 : ce poids obtient les taux d'erreur les plus faibles sur 10 des vingt problèmes considérés et est le meilleur de tous les poids dans le confrontations deux à deux. Malheureusement les performances de *adalike* se dégradent avec l'augmentation du nombre d'étapes de *boosting* au point de devenir le moins bon de tous les poids étudiés.

Les poids *covering*, *frequency* et même *uniform* présentent également des résultats intéressants et qui en plus résistent au passage à 1 000 étapes de *boosting* ; *covering* en particulier semble très proche du poids d'ADABOOST pour 1 000 étapes de *boosting*. Il est important de noter que, même si ces poids sont finalement moins bons que le poids

(a) Taux d'erreur

Problèmes	**adalike**	**covering**	**frequency**	**uniform**
audiology	**26,27**	28,82	26,73	27,14
breast-cancer	3,86	4,34	**3,81**	3,82
car	**22,12**	22,22	22,25	22,27
cmc	54,19	**54,16**	54,25	54,24
crx	15,12	**14,17**	14,45	14,45
dermatology	7,42	5,84	5,49	**5,13**
ecoli	23,49	23,94	23,49	**23,46**
glass	**4,59**	4,73	**4,59**	4,73
hepatitis	**16,99**	17,33	18,04	18,18
horse-colic	19,74	19,52	**19,43**	**19,43**
house-votes-84	9,98	6,25	5,26	**5,00**
ionosphere	18,28	7,47	**7,22**	**7,22**
iris	**6,33**	6,73	6,67	6,67
pima	27,33	**26,65**	27,02	27,02
promoters	15,90	16,47	**14,38**	14,98
sonar	36,88	26,06	**25,85**	**25,85**
tic-tac-toe	**0,54**	**0,54**	0,58	2,00
vowel	28,81	29,07	**28,22**	**28,22**
wine	14,17	4,39	4,32	**4,27**
zoo	**3,92**	5,39	4,12	5,60
Moyennes	*17,80*	*16,20*	***15,81***	*15,98*

(b) Confrontations

	adalike	**covering**	**frequency**	**uniform**	***Bilan***
adalike	-	*10-9*	7-12	8-12	25-33
covering	9-10	-	6-14	7-12	22-36
frequency	*12-7*	*14-6*	-	*8-5*	**34-18**
uniform	*12-8*	*12-7*	5-8	-	29-23

TABLEAU 5. *Résultats de* GLOBOOST *et des poids* adalike, covering, frequency, uniform *pour 100 étapes*

adaboost, ils maintiennent de meilleures performances que celles des systèmes C4.5 et GLOBO évalués à la section 4, tableau 1(a).

En conclusion de ces expérimentations, nous disposons de poids qui peuvent raisonnablement se substituer aux poids traditionnels d'ADABOOST.

(a) Taux d'erreur

Problèmes	**adalike**	**covering**	**frequency**	**uniform**
audiology	18,10	24,35	**17,07**	20,67
breast-cancer	5,91	4,39	**3,65**	3,65
car	**10,45**	11,37	11,52	11,84
cmc	50,05	**47,02**	47,67	47,63
crx	24,15	**13,90**	13,93	13,91
dermatology	13,14	4,14	3,94	**3,08**
ecoli	23,72	19,48	**19,19**	19,28
glass	8,69	6,33	**4,55**	6,33
hepatitis	18,70	**17,16**	17,62	17,84
horse-colic	21,81	16,98	**16,54**	16,65
house-votes-84	29,72	4,38	5,00	**4,20**
ionosphere	57,47	**6,36**	7,05	7,07
iris	12,33	5,87	**5,67**	5,73
pima	30,15	**24,59**	24,88	24,88
promoters	27,60	10,88	**6,03**	9,07
sonar	49,21	**23,90**	23,96	23,96
tic-tac-toe	0,24	0,39	**0,14**	7,08
vowel	25,95	14,51	13,44	**13,42**
wine	47,49	**3,98**	4,37	4,03
zoo	**3,71**	6,44	4,05	4,74
Moyennes	*23,93*	*13,32*	***12,51***	*13,25*

(b) Confrontations

	adalike	**covering**	**frequency**	**uniform**	***Bilan***
adalike	-	4-16	2-18	4-16	10-50
covering	*16-4*	-	9-11	9-10	34-25
frequency	*18-2*	*11-9*	-	*12-6*	**41-17**
uniform	*16-4*	*10-9*	6-12	-	32-25

TABLEAU 6. *Résultats de* GLOBOOST *et des poids* adalike, covering, frequency, uniform *pour 1 000 étapes*

6.2. *Évaluation expérimentale de* GLOBOOST

Toujours avec le protocole et les données de la section 4.1, nous évaluons l'algorithme GLOBOOST muni des différents poids dont nous disposons (*adalike*, *covering*, *frequency* et *uniform*). Les résultats sont présentés aux tableaux 5 et 6.

À nouveau, notre poids *adalike* fait apparaître des valeurs aberrantes et supporte mal l'augmentation du nombre d'étapes de *boosting*. Par contre, les poids *covering*, *frequency* et *uniform* confirment leurs bons résultats au point de concurrencer ADABOOST-MG. De plus, cette tendance se confirme lorsque l'on augmente le nombre d'étapes de 100 à 1 000. Même le poids *uniform* obtient des résultats très honorables en regard de sa simplicité. Ce résultat est surprenant pour nous : ces poids faisaient baisser les performances d'ADABOOST lorsque celui-ci était utilisé comme générateur et on s'attendait à ce que les taux d'erreur augmentent un peu plus avec un générateur purement aléatoire. Au contraire, ces poids se révèlent mieux adaptés à ce mode de production des hypothèses.

Précisons que, dans le cas de la génération purement aléatoire de GLOBOOST, la fréquence et la couverture sont fortement liées : une hypothèse ayant un fort support sortira souvent, et inversement, une hypothèse fréquente couvre nécessairement beaucoup d'exemples. Ceci explique les résultats proches des poids basés sur ces quantités.

Finalement, c'est le poids *frequency* qui apparaît comme le meilleur de tous : que ce soit à 100 ou à 1 000 étapes de *boosting*, ce poids gagne toutes ces confrontations et obtient le taux d'erreur moyen le plus faible.

7. Discussion

7.1. *Bilan*

Nous avons montré que les résultats sur le *boosting* laissaient une large place à l'investigation, en particulier sur les apprenants faibles qui peuvent être utilisés. Nous avons proposé d'utiliser dans ce rôle la production de *moindres généralisés corrects*. Devant la complexité de ce calcul et l'usage intensif que doit en faire ADABOOST, nous avons proposé une génération des hypothèses parallélisable et avons démontré que des poids simples pouvaient ensuite être affectés aux hypothèses. Toutes ces idées ont été validées expérimentalement.

Aujourd'hui, le système GLOBOOST implémente la production aléatoire vue à l'algorithme 5 avec le poids *frequency*. Ce poids offre de bonnes performances et n'occasionne pas de calculs importants après la génération.

Le tableau 7 et la figure 3 dressent le bilan de nos expérimentations pour cet algorithme. En résumé GLOBOOST présente des taux d'erreur et des écarts types faibles ; de

(a) Taux d'erreur

Problèmes	**C4.5**	**GLOBO**	**ABDS**	**ABMG**	**GLOBOOST**
audiology	18,20	20,76	**15,79**	19,22	17,07
breast-cancer	4,87	3,89	4,16	**3,10**	3,65
car	**7,69**	10,17	13,36	9,74	11,52
cmc	48,07	50,60	**44,87**	49,61	47,67
crx	14,79	16,50	16,53	14,47	**13,93**
dermatology	6,23	8,51	4,38	4,63	**3,94**
ecoli	15,89	23,88	**15,59**	19,25	19,19
glass	28,72	5,57	22,53	4,59	**4,55**
hepatitis	20,70	20,09	18,23	18,39	**17,62**
horse-colic	**13,63**	22,29	21,71	18,42	16,54
house-votes-84	**3,22**	7,39	5,31	6,12	5,00
ionosphere	7,96	8,74	12,19	7,44	**7,05**
iris	**5,33**	7,47	**5,33**	**5,33**	5,67
pima	29,29	27,04	25,39	25,13	**24,88**
promoters	18,17	22,60	18,67	7,00	**6,03**
sonar	28,97	31,09	**15,03**	23,34	23,96
tic-tac-toe	14,40	1,11	1,67	**0,00**	0,14
vowel	21,21	23,99	14,65	**9,30**	13,44
wine	8,83	8,94	18,07	5,73	**4,37**
zoo	7,51	5,55	5,25	**3,21**	4,05
Moyennes	*16,18*	*16,31*	*14,94*	*12,70*	***12,51***

(b) Confrontations

	C4.5	**GLOBO**	**ABDS**	**ABMG**	**GLOBOOST**	***Bilan***
C4.5	-	*14-6*	7-12	6-13	5-15	32-46
GLOBO	6-14	-	7-13	0-20	1-19	14-66
ABDS	*12-7*	*13-7*	-	7-12	5-15	37-41
ABMG	*13-6*	*20-0*	*12-7*	-	7-13	52-26
GLOBOOST	*15-5*	*19-1*	*15-5*	*13-7*	-	**62-18**

TABLEAU 7. *Comparaison de* GLOBOOST *(*frequency*) avec* C4.5, GLOBO *et* ADABOOSTDS *et* ADABOOSTMG. GLOBOOST *et* ADABOOSTMG *utilisent ici 1 000 étapes de génération*

plus, il supporte favorablement l'augmentation du nombre d'étapes de *boosting*. Ces résultats sont tout à fait comparables à ceux obtenus avec ADABOOSTMG, avec l'avantage que le calcul peut être réparti sur plusieurs machines.

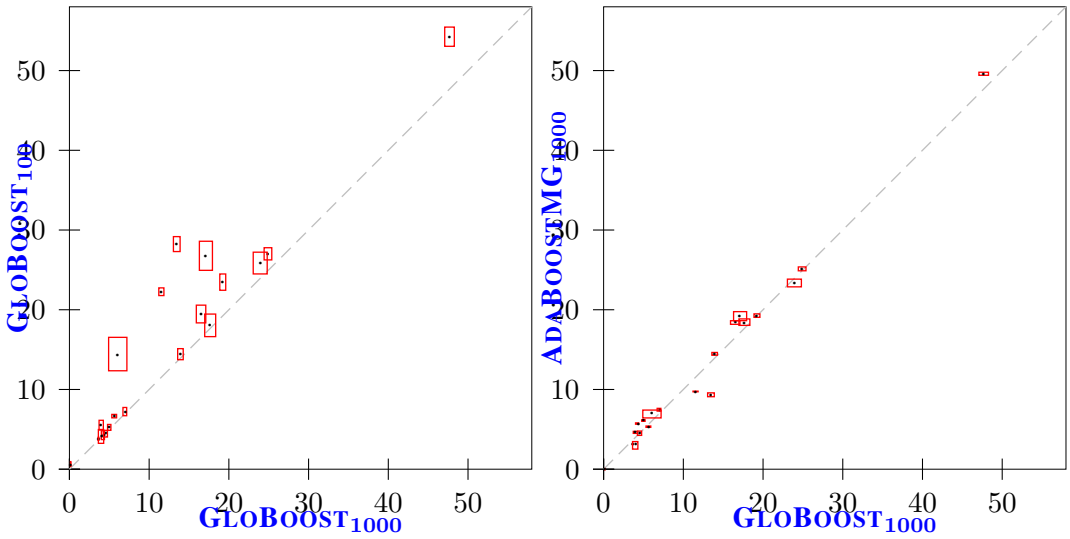

FIGURE 3. GLOBOOST$_{1\,000}$ *contre* ADABOOSTMG$_{1\,000}$ *et* GLOBOOST$_{100}$

7.2. *Travaux apparentés*

GLOBO et GLOBOOST sont proches de la famille AQ (Michalski., 1983) : l'idée de ce type d'algorithmes est de produire pour chaque exemple l'ensemble des hypothèses maximalement générales qui couvrent cet exemple (la graine) et qui soient correctes vis-à-vis des exemples de classe différente. Cet ensemble d'hypothèses est appelée l'*étoile* de l'exemple graine. L'étape suivante consiste à sélectionner dans l'étoile, l'hypothèse qui semble être la meilleure (en pratique, il n'est pas possible de produire complètement une étoile et ce critère, quel qu'il soit, doit être utilisé pour élaguer l'espace de recherche). Enfin, le processus est itéré en choisissant comme graine un exemple non encore couvert et ce, jusqu'à la couverture complète de tous les exemples. L'algorithme DLG (Webb *et al.*, 1992) est une version de AQ qui utilise les moindres généralisés pour produire les étoiles des exemples. Il reste biaisé par la mesure d'intérêt qui permet de sélectionner les hypothèses et par l'aspect glouton de AQ. GLOBO et GLOBOOST répondent à ces défauts en produisant les règles indépendamment les unes des autres et en différant la phase de sélection.

RISE (Domingos, 1996) est un autre système utilisant la construction d'un moindre généralisé comme brique de base ; dans ce cas, il est permis au moindre généralisé d'être incorrect. Son originalité repose sur le mode de classification d'un exemple : RISE considère la règle la plus proche de l'exemple à classer, au sens d'une distance. RISE a été boosté (Wiratunga *et al.*, 2002) : dans ce cas précis, les hypothèses faibles ne sont plus de simples règles incomplètes mais des théories complètes par rapport aux exemples. C'est le cas également des *Light Weight Rules* (Weiss *et al.*, 2000) où une

formule DNF est modifiée par une technique de type *générer et tester* jusqu'à couvrir tous les exemples d'une classe. Ici aussi, c'est un constructeur complexe qui est boosté.

Enfin, SLIPPER (Cohen *et al.*, 1999) n'utilise pas les moindres généralisés mais des règles assez proches dans la forme et qui de plus sont boostées avec ADABOOST : dans SLIPPER, l'apprenant faible choisit la règle qui minimise la valeur de Z_t apparaissant dans l'algorithme 1, comme préconisé dans (Schapire *et al.*, 1999). Cette stratégie permet de faire chuter très rapidement l'erreur observée (mais pas nécessairement l'erreur en généralisation). Notons simplement que, dans notre cadre, la correction des règles produites fait que l'erreur observée est nulle dès que suffisamment de règles ont été produites pour couvrir tous les exemples disponibles.

Au delà des méthodes elles-mêmes, notre démarche s'inscrit dans une tendance qui consiste à englober toutes les techniques de production et combinaison d'hypothèses dans un cadre plus large que celui du *boosting*. Une première tentative dans ce sens a consisté à rassembler les méthodes de type *arcing* (pour *adaptively resample and combine*, (Breiman, 1996b)), c'est-à-dire essentiellement le *bagging* et le *boosting*. Plus proche de nous, (Dietterich, 2000b) décrit des expérimentations semblables aux nôtres mais dédiées aux arbres de décisions (comparaison de *boosting*, *bagging* et de production aléatoire), travail qui conduit finalement à la notion de *méthodes d'ensemble* (Dietterich, 2000a). Certains préconisent de réserver le terme de *boosting* aux algorithmes effectivement capables de booster un apprenant faible dans le cadre PAC et de rassembler les autres techniques sous le terme de *leveraging* (Meir *et al.*, 2003).

D'un point de vue plus théorique, des travaux sont menés pour voir toutes ces méthodes comme des algorithmes de type *Monte Carlo* et par là expliquer leurs performances (Esposito *et al.*, 2003; Esposito *et al.*, 2004).

7.3. *Perspectives*

La question se pose maintenant de savoir si les résultats présentés dans cet article sont propres aux moindres généralisés corrects. La seule particularité de notre apprenant faible est de fournir des hypothèses qui ne se trompent pas : peut-être que cette caractéristique rend les poids et les performances d'ADABOOST plus faciles à approcher. Autrement dit, autoriser un moindre généralisé à être incorrect pourrait améliorer les performances d'ADABOOST. Précisons que cela va dans le sens de notre souci d'efficacité : un apprenant incorrect n'a plus besoin de considérer tous les exemples mais seulement ceux de poids forts.

Dans l'idée d'améliorer l'apprenant faible fourni à ADABOOST, il semble aisé de contraindre un peu plus notre construction de moindres généralisés corrects pour focaliser sur les exemples de poids forts (même si c'est le principe de l'algorithme 4, cette mise en œuvre reste assez lâche).

Nous avons vu que ADABOOSTMG et GLOBOOST, tous deux basés sur les moindres généralisés corrects, s'amélioraient avec l'augmentation du nombre d'hypothèses produites. Il faudra donc poursuivre dans cette voie, cela étant plus facile en pratique pour la version parallèle de GLOBOOST.

La recherche de nouveaux poids reste elle aussi ouverte. En particulier, il faut déterminer pourquoi notre poids *adalike* se comporte bien pour des étapes de *boosting* en nombre faible mais mal sur certains problèmes lorsque ce nombre augmente.

Une perspective qui nous semble importante est celle de la compréhensibilité du classifieur produit. Il s'agit maintenant de savoir si, à partir des hypothèses produites par ADABOOST ou GLOBOOST, on peut revenir à des théories compréhensibles comme celles de GLOBO. La solution la plus simple est d'appliquer la couverture minimale de GLOBO sur les hypothèses produites mais il est peu probable que les performances de la théorie obtenue restent proches de celles du système de votants. Il serait donc préférable de chercher un ensemble d'hypothèses d'assez petite taille pour être appréhendable par un humain mais en conservant au maximum le pouvoir prédictif du classifieur complet. Précisons que, tout proche de cette problématique, le problème de fournir un nombre minimal de votants a été suggéré dans (Quinlan, 1999) et une solution proposée dans (Long, 2002), solution qui implique de calculer d'abord toutes les hypothèses faibles possibles (dans ce cas, des *decision stumps*).

Enfin, un dernier objectif est de poursuivre la parallélisation en distribuant également les données : la génération des hypothèses serait distribuée comme nous l'avons décrit, mais en plus elle se ferait sur des échantillons de l'ensemble de données initial. Cela conduira à des hypothèses incorrectes mais nous avons vu que cela pouvait être un avantage. À noter qu'une fois cette distribution mise en œuvre, le poids *frequency* sera un candidat privilégié puisqu'il ne nécessite pas de calcul sur l'ensemble de données, comme ce serait le cas par exemple pour le poids *covering*.

Remerciements

Ce travail a été soutenu par :

– l'unité de recherche INRIA Futurs ;

– le Laboratoire d'Informatique Fondamentale de Lille (UMR CNRS 8022) ;

– le Contrat de Plan État-Région Nord/Pas-de-Calais (CPER 2000-2006, programme TAC, projet COCOA) ;

– l'Action Concertée Incitative « Masse de données » (ACI-MDD, FNS).

8. Bibliographie

Blake C., Merz C., UCI Repository of machine learning databases [http ://www.ics.uci.edu/∼mlearn/MLRepository.html], 1998.

Breiman L., Bagging Predictors, *Machine Learning*, vol. 24, n° 2, p. 123-140, 1996a.

Breiman L., Bias, Variance, and Arcing Classifiers, 1996b. Technical Report 460, Statistics Department, University of California.

Cohen W. W., Singer Y., A simple, fast, and effective rule learner, *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, AAAI/MIT Press, p. 335-342, 1999.

De Comité F., Gilleron R., Tommasi M., Learning multi-label alternating decision trees and applications, G. Bisson (ed.), *Actes de la Conférence d'Apprentissage Automatique*, p. 195-210, 2001.

Dietterich T. G., Ensemble Methods in Machine Learning, J. Kittler, F. Roli (eds), *First International Workshop on Multiple Classifier Systems*, vol. 1857 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 1-15, 2000a.

Dietterich T. G., An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees : Bagging, Boosting, and Randomization, *Machine Learning*, vol. 40, n° 2, p. 139-158, 2000b.

Domingos P., Unifying Instance-Based and Rule-Based Induction, *Machine Learning*, vol. 24, n° 2, p. 141-168, 1996.

Esposito R., Saitta L., Monte Carlo Theory as an Explanation of Bagging and Boosting, G. Gottlob, T. Walsh (eds), *Proceeding of the Eighteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufman, p. 499-504, 2003.

Esposito R., Saitta L., A Monte Carlo Analysis of Ensemble Classification, R. Greiner, D. Schuurmans (eds), *Proceedings of the twenty-first International Conference on Machine Learning*, ACM Press, New York, NY, Banff, Canada, p. 265-272, July, 2004.

Freund Y., Boosting a Weak Learning Algorithm by Majority, *Information and Computation*, vol. 121, n° 2, p. 256-285, 1995.

Freund Y., Schapire R. E., A decision-theoretic generalization of on-line learning and an application to boosting, *European Conference on Computational Learning Theory*, p. 23-37, 1995.

Freund Y., Schapire R. E., Experiments with a New Boosting Algorithm, *International Conference on Machine Learning*, p. 148-156, 1996.

Freund Y., Schapire R. E., A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences*, vol. 55, n° 1, p. 119-139, 1997.

Freund Y., Schapire R. E., Discussion of the paper Arcing Classifiers by Leo Breiman, *The Annals of Statistics*, vol. 26, p. 824-832, 1998.

Grove A. J., Schuurmans D., Boosting in the Limit : Maximizing the Margin of Learned Ensembles, *AAAI/IAAI*, p. 692-699, 1998.

Harries M., Boosting a strong learner : evidence against the minimum margin, *Proc. 16th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, p. 171-180, 1999.

Kearns M. J., Vazirani U. V., *An Introduction to Computational Learning Theory*, MIT Press, 1994.

Kearns M., Valiant L. G., Cryptographic limitations on learning Boolean formulae and finite automata, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, p. 433-444, 1989.

Koltchinskii V., Panchenko D., Empirical Margin Distributions and Bounding the Generalization Error of Combined Classifiers, *Annals of Statistics*, vol. 30, n° 1, p. 1-50, 2002.

Long P. M., Minimum Majority Classification and Boosting, *Proceedings of the The Eighteenth National Conference on Artificial Intelligence*, 2002.

Meir R., Rätsch G., An Introduction to Boosting and Leveraging, S. Mendelson, A. Smola (eds), *Advanced Lectures on Machine Learning*, number 2600 in *LNAI*, Springer-Verlag, p. 119-184, January, 2003.

Michalski. R. S., *A Theory and Methodology of Inductive Learning*, Springer-Verlag, p. 83-134, 1983.

Quinlan J. R., Some Elements of Machine Learning, 1999. Invited talk (ILP / ICML).

Rudin C., Daubechies I., Schapire R. E., The Dynamics of AdaBoost : Cyclic Behavior and Convergence of Margins, *Machine Learning Research*, vol. 5, p. 1557-1595, 2004.

Rätsch G., Warmuth M., Maximizing the Margin with Boosting, J. Kivinen, R. H. Sloan (eds), *Proceedings of the Annual Conference on Computational Learning Theory (COLT)*, vol. 2375 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 334-350, 2002.

Rätsch G., Warmuth M. K., Efficient Margin Maximizing with Boosting, 2003. Soumis à Journal of Machine Learning Research (JMLR).

Schapire R. E., The Strength of Weak Learnability, *Machine Learning*, vol. 5, p. 197-227, 1990.

Schapire R. E., The boosting approach to machine learning : An overview, *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

Schapire R. E., Freund Y., Bartlett P., Lee W. S., Boosting the margin : a new explanation for the effectiveness of voting methods, *The Annals of Statistics*, vol. 26, p. 1651-1686, 1998.

Schapire R. E., Singer Y., Improved Boosting Algorithms using Confidence-Rated Predictions, *Machine Learning*, vol. 37, n° 3, p. 297-336, 1999.

Schapire R. E., Singer Y., BoosTexter : A Boosting-based System for Text Categorization, *Machine Learning*, vol. 39, n° 2/3, p. 135-168, 2000.

Srinivasan A., King R., Bristol D., An assessment of submissions made to the Predictive Toxicology Evaluation Challenge, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, p. 270-276, 1999.

Srinivasan A., King R. D., Muggleton S. H., Sternberg M. J. E., The Predictive Toxicology Evaluation Challenge, *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, p. 4-9, 1997.

Torre F., GloBo : un algorithme stochastique pour l'apprentissage supervisé et non-supervisé, M. Sebag (ed.), *Actes de la Première Conférence d'Apprentissage*, p. 161-168, 1999.

Valiant L. G., A theory of the Learnable, *Communications of the ACM*, vol. 27, p. 1134-1142, 1984.

Webb G. I., Agar J. W. M., Inducing Diagnostic Rules for Glomerular Disease with the DLG Machine Learning Algorithm, *Artificial Intelligence in Medicine*, vol. 4, p. 419-430, 1992.

Weiss S. M., Indurkhya N., Lightweight Rule Induction, P. Langley (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, Stanford, CA, p. 1135-1142, 2000.

Wiratunga N., Craw S., Rowe R., Learning to Adapt for Case-Based Design, S. Craw, A. D. Preece (eds), *Proceedings of the 6th European Conference on Advances in Case-Based Reasoning*, vol. 2416 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 421-435, 2002.