Méthodes d'ensemble en apprentissage automatique

Fabien Torre

Université de Lille

Mercredi 14 et 21 octobre 2009

Combinaisons d'hypothèses faibles

Idées issues du *boosting*

- on dispose d'un apprenant L ;
- on le contraint à produire diverses hypothèses h_t ;
- les h_t sont combinées au sein d'un vote.

Vocabulaire

En dehors du cadre PAC, par abus de langage, on dit que :

- L est un apprenant faible ;
- chaque h_t est une hypothèse faible.

Méthode d'ensemble

Éléments constitutifs

Une méthode d'ensemble se définit par :

- l'espace des hypothèses $\mathcal{H}$;
- un apprenant faible L ;
- une stratégie utilisant L pour produire les $h_{t=1\dots T} \in \mathcal{H}$;
- une méthode de combinaison des h_t.

Sortie

Une méthode d'ensemble fournit un classifieur H :

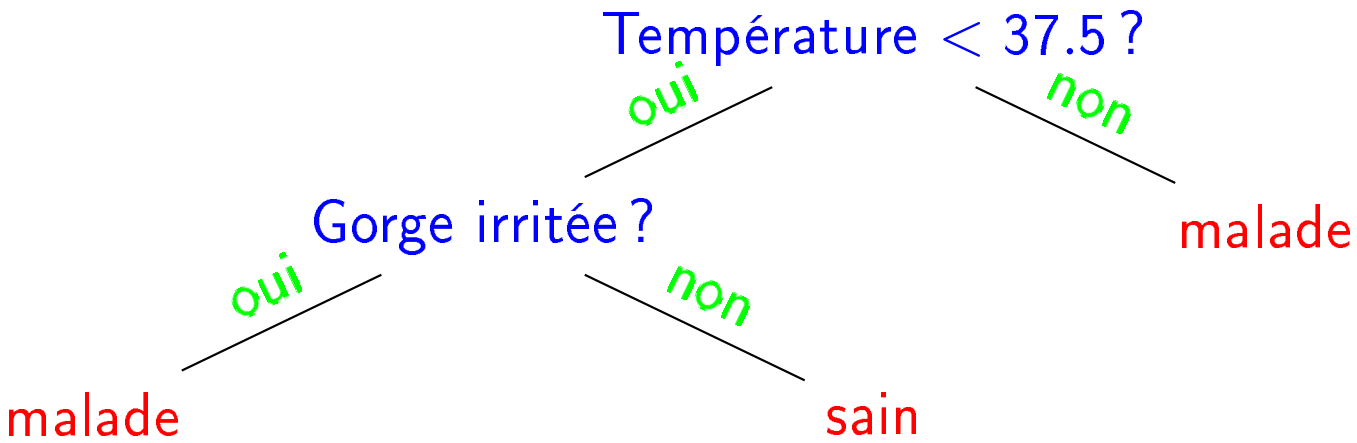
- $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t . h_t(x)\right)$
- $H = [(h_t, \alpha_t)]_{t=1\dots T} \in (\mathcal{H}, \mathbb{R}_+^*)^T$
- on a changé d'espace...

voir trois droites qui vaporisent 9 points

Rappel sur l'inférence d'arbres

Principes

- passer en revue tous les tests possibles ;
- choisir le test qui minimise le désordre (entropie, Gini, etc.) ;
- recommencer sur les fils, jusqu'à obtenir des feuilles pures.

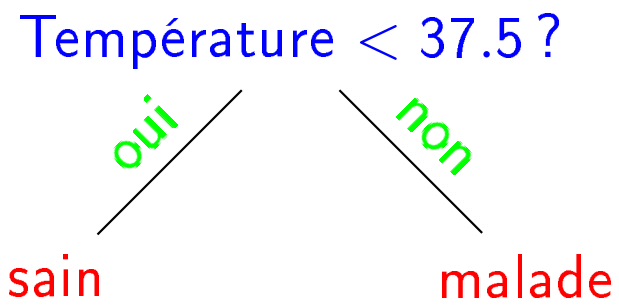


Apprentissage parfait possible...

Les *stumps*

Définition : Stumps

Arbres de décision à un seul nœud, on fait un test et on conclut sur une classe ou sur une autre.



- performances si l'on en prend plusieurs ?
- a-t-on toujours un *stump* faible à disposition ?

Moindres généralisés

Combinaison de théories

- DLG (s'arrête quand tous les exemples sont couverts) ;
- GloBo (opère une sélection minimale sur les règles produites).

Combinaison de règles

- MGC ;
- MGC + Laplace ;
- ne pas s'arrêter quand tous les exemples sont couverts ;
- ne pas faire une sélection minimale sur les règles produites ;
- en produire beaucoup et toutes les garder.

Algorithme MGC

Entrées : $E = [e_1, \ldots, e_n] \subseteq \mathcal{X}$ un ensemble *ordonné* de n exemples de la même classe, N un ensemble de contre exemples.

Sortie : $h \in \mathcal{H}$ une généralisation maximalement correcte de E.

1: $h = e_1$
2: **for** $i = 2$ to n **do**
3: $h' = \texttt{MG}(h, e_i)$ {Généralisation entre deux hypothèses.}
4: **if** $(\forall e \in N : h' \not\geq e)$ **then**
5: $h = h'$
6: **end if**
7: **end for**
8: **return** h

Dépendance à l'ordre des exemples.

Motivation du bagging : limiter la variance

Rappels

- décomposition de l'erreur en biais/variance ;
- la variance : plusieurs hypothèses de $\mathcal{H}$ sont équivalentes par rapport à A, il est difficile de choisir, l'apprenant est *instable*.

Idée : on ne choisit pas, on en prend plusieurs et on les fait voter.

Esquisse de la méthode

- apprendre chaque h_t à partir d'un échantillon tiré aléatoirement dans l'échantillon complet A ;
- poids identique pour chaque h_t.

Technique du bagging [Breiman, 1996]

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

for $t = 1$ to T **do**
    $A_t = \emptyset$
    for $i = 1$ to n **do**
        tirer un exemple au hasard dans A et le placer dans A_t
    end for
    $h_t = L(A_t)$ {appel à l'apprenant faible}
    ajouter h_t à H
end for
return H

Quels apprenants pour le bagging ?

Pistes

- soit on cherche un classifieur très instable ;
- soit on modifie des classifieurs stables pour les déstabiliser !

Une possibilité pour introduire de l'instabilité :
ajouter de l'aléatoire dans les algorithmes d'apprentissage !

Exemple : de l'aléatoire dans l'inférence d'arbres...

Random Forests [Breiman, 2001]

Un apprenant L aléatoire à base d'arbres de décision

À chaque nœud de l'arbre de décision :

- on choisit au hasard m attributs ;
- on prend le meilleur test constructible sur ces attributs.

Observations

- le résultat dépend de m ;
- L, à partir d'un même jeu de données, apprend un arbre différent à chaque appel ;
- utilisable avec le bagging.

Bagging et moindres-généralisés

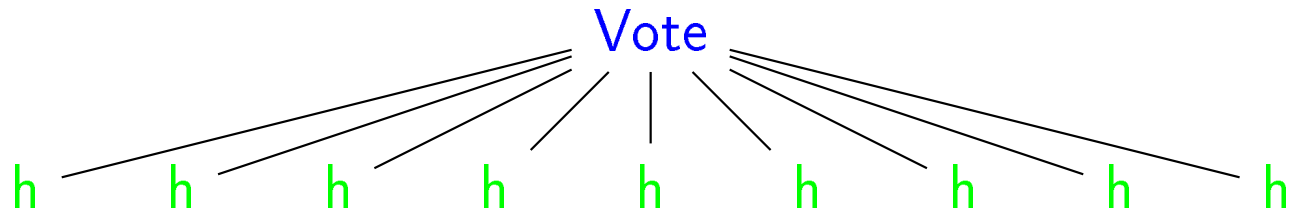
Entrées : n exemples x_i et leurs classes y_i ; T un nombre d'itérations à effectuer.

Sortie : H le classifieur final.

1: **for** $t = 1$ to T **do**
2: $A_t = \emptyset$
3: **for** $i = 1$ to n **do**
4: tirer un exemple au hasard et le placer dans A_t
5: **end for**
6: $target =$ une classe tirée au hasard
7: $P = \{x_i \in A_t | y_i = target\}$
8: $N = \{x_i \in A_t | y_i \neq target\}$
9: $h_t = \texttt{MGC}(P, N)$ {Appel à l'algorithme MGC}
10: ajouter h_t à H
11: **end for**
12: **return** H

Algorithme AdaBoost [Freund, 1995], intuitions

- AdaBoost est un algorithme de *boosting* plus simple que celui de [Schapire, 1990] car, comme le bagging, il opère sur un seul niveau, il n'y a pas de récursion :



- n appels groupés à l'oracle pour constituer un échantillon A : ce sont les exemples de A qui sont fournis à L à chaque appel ;
- l'instabilité est introduite en jouant sur la distribution sur A ;
- à l'étape t, L apprend h_t au mieux les exemples de poids forts, puis les poids des exemples mal classés par h_t sont augmentés et les poids des exemples bien classés par h_t sont diminués.

AdaBoost, version généralisée [Freund and Schapire, 1995] I

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

1: **for** $i = 1$ to n **do**
2: $\mathcal{D}_1[x_i] = 1/n$ {initialisation de la distribution initiale}
3: **end for**
4: **for** $t = 1$ to T **do**
5: $h_t = L(\mathcal{D}_t)$ {Appel à l'apprenant faible}
6: choisir $\alpha_t \in \mathbb{R}$ comme « poids » de h_t
7: **for** $i = 1$ to n **do**
8: $\mathcal{D}_{t+1}[x_i] = \mathcal{D}_t[x_i].\exp(-\alpha_t.y_i.h_t(x_i))$
9: **end for**
10: $Z_t = \sum_{i=1}^{n} \mathcal{D}_{t+1}[x_i] = \sum_{i=1}^{n} \mathcal{D}_t[x_i].\exp(-\alpha_t.y_i.h_t(x_i))$
11: **end for**

AdaBoost, version généralisée [Freund and Schapire, 1995] II

12: **for** $i = 1$ to n **do**

13: $\mathcal{D}_{t+1}[x_i] = \frac{\mathcal{D}_{t+1}[x_i]}{Z_t}$

14: **end for**

15: **return** $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t . h_t(x)\right)$

Conditions du *boosting*

- L apprenant faible ;
- n bien choisi en fonction de ϵ et δ ;
- T bien choisi en fonction de ϵ et δ.

AdaBoost en pratique

n et T doivent être assez grands, mais dans un cas réel :

- on ne sait pas forcément déterminer si L est faible ;
- pas d'oracle, donc pas de certitude d'avoir assez d'exemples ;
- comment choisir T ?

On perd la garantie de booster l'erreur en généralisation mais :

- AdaBoost peut travailler sur un échantillon donné sans se soucier de savoir si sa taille n est suffisante et avec une valeur de T arbitraire ;
- reste à voir les modalités du vote : définition des α_t ?

Choix des α_t

Choix initiaux

- $\alpha_t = \frac{1}{2}.\ln(\frac{1-\epsilon_t}{\epsilon_t})$; $\alpha_t \in]0;+\infty[$
- où $\epsilon_t = \sum_{i:y_i \neq h_t(x_i)} \mathcal{D}_t[x_i]$. $\epsilon_t \in]0;\frac{1}{2}[$

Minimisation de l'erreur empirique

$$\text{erreur}_A(H) = \frac{|\{i : H(x_i) \neq y_i\}|}{n} \leq \prod_{t=1}^{T} Z_t$$

minimiser chaque Z_t conduit à

- $\alpha_t = \frac{1}{2}.\ln(\frac{1+r_t}{1-r_t})$; $\alpha_t \in]0;+\infty[$
- où $r_t = \sum_{i=1}^{n} \mathcal{D}_t[x_i].y_i.h_t(x_i)$. $r_t \in]0;1[$

Version pour abstention [Schapire and Singer, 1999] I

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

1: **for** $i = 1$ to n **do**
2: $w_i = 1/n$ {on note maintenant $w_i = \mathcal{D}_t[x_i]$}
3: **end for**
4: **for** $t = 1$ to T **do**
5: $h_t = L([(x_i, y_i, w_i)])$
6: **for** $b \in \{-, 0, +\}$ **do**
7: $W_b = \sum_{i:\text{sign}(y_i h_t(x_i))=b} w_i$
8: **end for**
9: $\alpha_t = \frac{1}{2}\log\left(\frac{W_+ + \frac{1}{2}W_0}{W_- + \frac{1}{2}W_0}\right)$
10: $Z_t = \sum_i^n [w_i.\exp(-\alpha_t y_i h_t(x_i))]$

Version pour abstention [Schapire and Singer, 1999] II

11: **for** $i = 1$ to n **do**

12: $$w_i = \frac{w_i.\exp(-\alpha_t.y_i.h_t(x_i))}{Z_t}$$

13: **end for**

14: **end for**

15: **return** $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t.h_t(x)\right)$

Stratégie pour L

Toujours pour minimiser Z_t, produire le h de $\mathcal{H}$ qui minimise :

$$\sum_{i=1}^{n} \mathcal{D}_t[x_i].\exp(-\alpha_t.y_i.h_t(x_i))$$

quand c'est possible :

- on peut énumérer $\mathcal{H}$ à l'avance ;
- calculer la couverture de chaque $h \in \mathcal{H}$;
- puis, à chaque étape du *boosting*, choisir la meilleure h.

C'est par exemple possible pour les *stumps*.

Boosting d'arbres

Il faut prendre en compte les poids des exemples dans l'inférence d'arbres.

Implémentations

- *stumps* : *boostexter* [Schapire and Singer, 2000] ;
- JStumps : une implémentation Java du *boosting de stumps* ;
- arbres complets : C5.0 [Quinlan, 2004], commercial, algorithme non détaillé.

Boosting de moindres généralisés : AdaBoostMG I

Entrées : n exemples x_i et leurs classes $y_i \in \{-1, +1\}$; T un nombre d'itérations à effectuer.

Sortie : H le classifieur final.

1: **for** $i = 1$ to n **do**
2: $w_i = 1/n$ {initialisation des poids des exemples}
3: **end for**
4: **for** $t = 1$ to T **do**
5: *target* = la classe de l'exemple de poids le plus fort
6: $P = \{x_i | y_i = target\}$
7: $N = \{x_i | y_i \neq target\}$
8: trier P par poids décroissant
9: $h_t = \texttt{MGC}(P, N)$
10: **for** $b \in \{-, 0, +\}$ **do**
11: $W_b = \sum_{i:\text{sign}(y_i h_t(x_i))=b} w_i$

Boosting de moindres généralisés : AdaBoostMG II

end for

$\alpha_t = \frac{1}{2}\log\left(\frac{W_+ + \frac{1}{2}W_0}{W_- + \frac{1}{2}W_0}\right)$

$Z_t = \sum_i^n [w_i.\exp(-\alpha_t y_i h_t(x_i))]$

for $i = 1$ to n **do**

$w_i = \frac{w_i.\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

end for

end for

return $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Meilleure prise en compte des poids ?

Méthodes aléatoires

Principes

- ne pas toucher à l'échantillon ;
- ne pas jouer sur la distribution ;
- mais simplement rendre l'apprenant instable ;
- puis voter à égalité.

Arbres de décision [Dietterich, 2000b]

Un (autre) apprenant L aléatoire à base d'arbres de décision

À chaque nœud de l'arbre de décision :

- on évalue tous les tests possibles ;
- on tire au hasard parmi les 20 meilleurs.

GloBoost [Torre, 2005]

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

1: **for** $t = 1$ to T **do**
2: $target$ = classe choisie au hasard
3: $P = \{x_i | y_i = target\}$
4: $N = \{x_i | y_i \neq target\}$
5: mélanger P aléatoirement
6: $h_t = \texttt{MGC}(P,N)$ {Appel à l'algorithme MGC}
7: **end for**
8: **return** $H(x) = \text{sign}\left(\sum_{t=1}^{T} h_t(x)\right)$

Protocole

- Algorithmes en présence : C4.5, GloBo, *boosting* de *stumps*, *boosting* de moindres-généralisés, GloBoost ;
- utilisation de volata pour les méthodes à base de moindres généralisés ;
- 1 000 itérations pour les méthodes d'ensemble ;
- 20 problèmes du *repository UCI* [Blake and Merz, 1998] ;
- validations croisées 10 fois ;
- nombre d'erreurs moyen ;
- visualisation des performances.

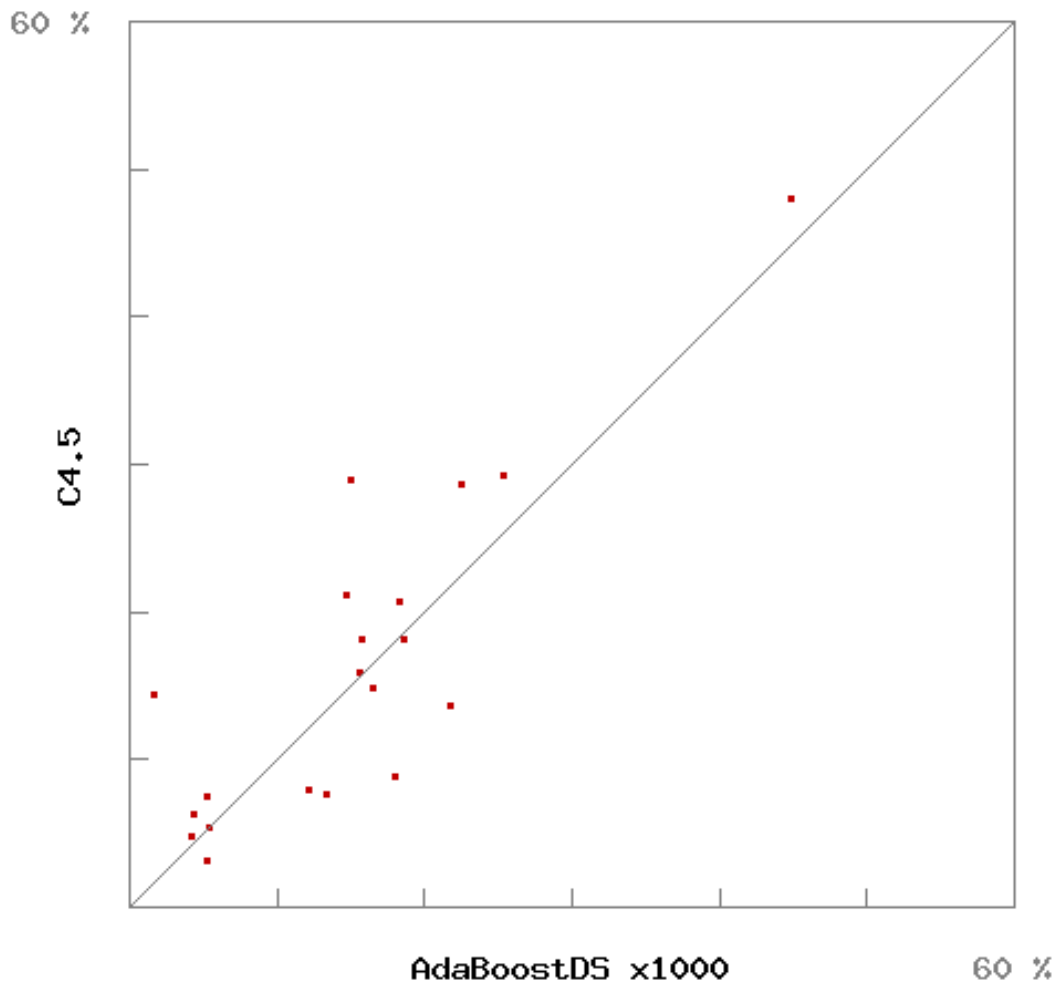
Tableau de résultats I

Problème	C4.5	GloBo	AB+S	AB+MG	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**

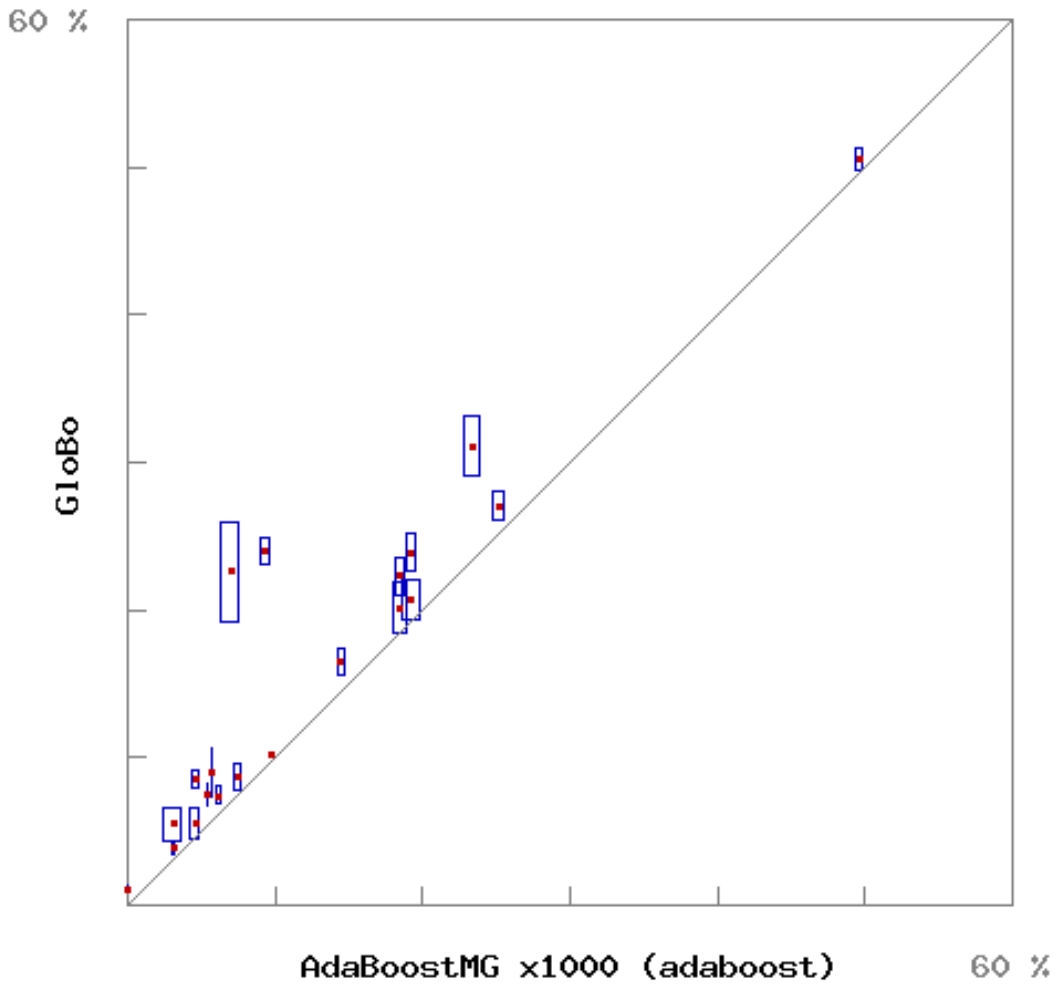
Tableau de résultats II

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**
	C4.5	GloBo	AB+S	AB+MG	GloBoost

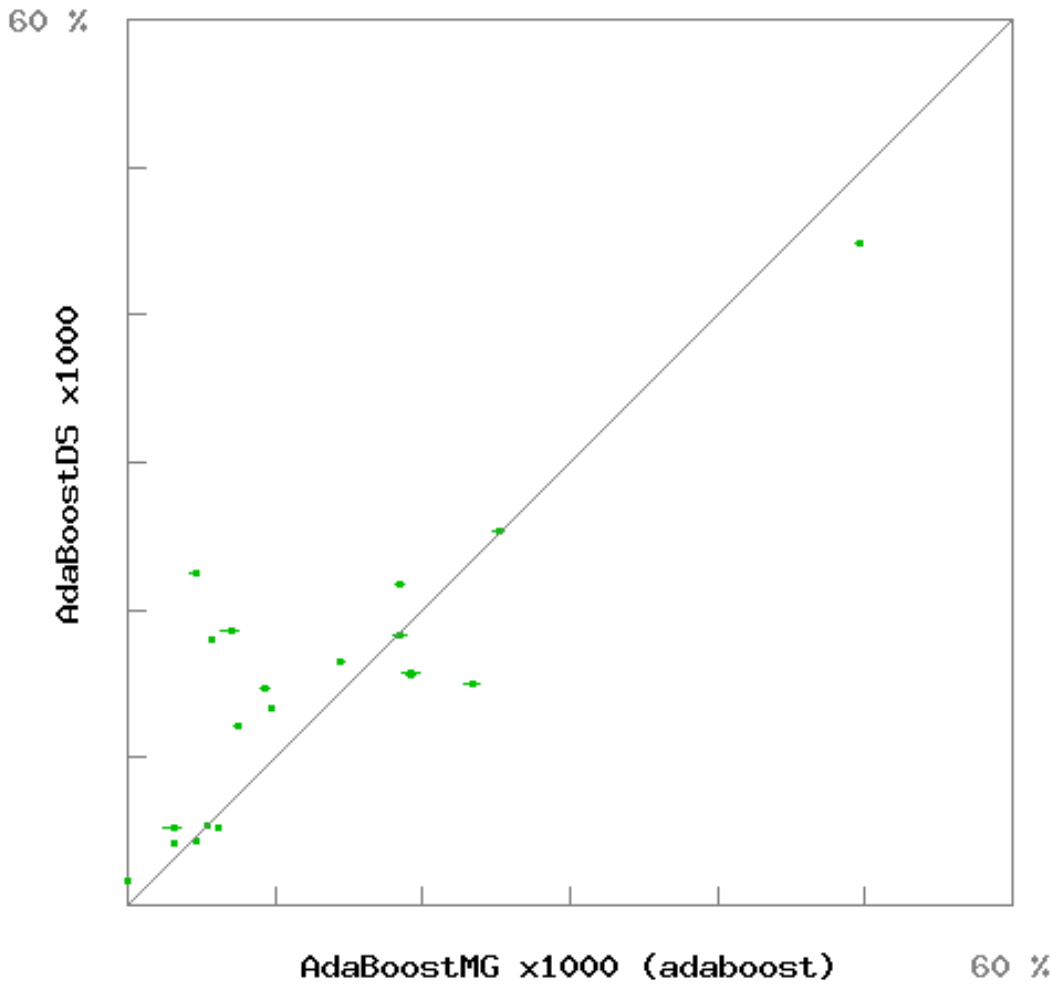
C4.5 vs. *boosting* de stumps



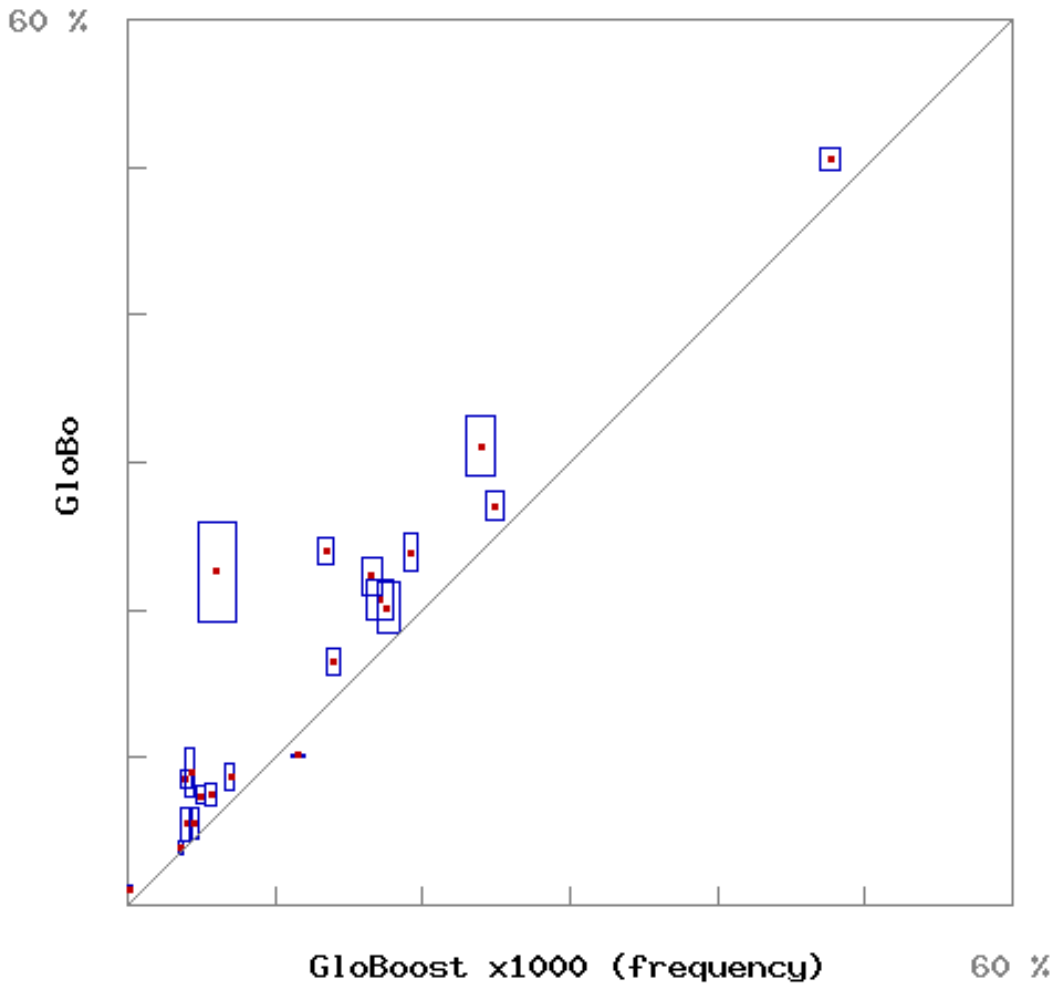
GloBo vs. AdaBoost-MG



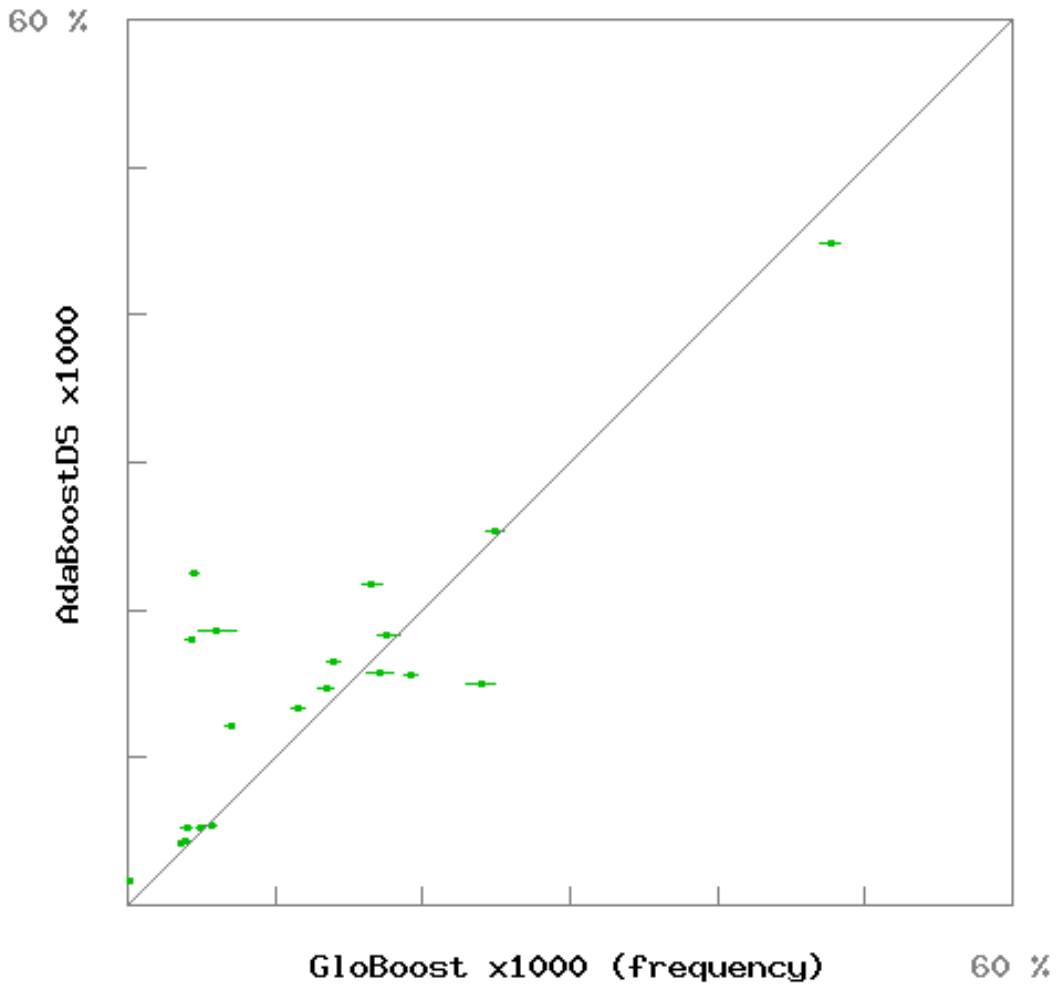
Boosting : MGC ou stumps ?



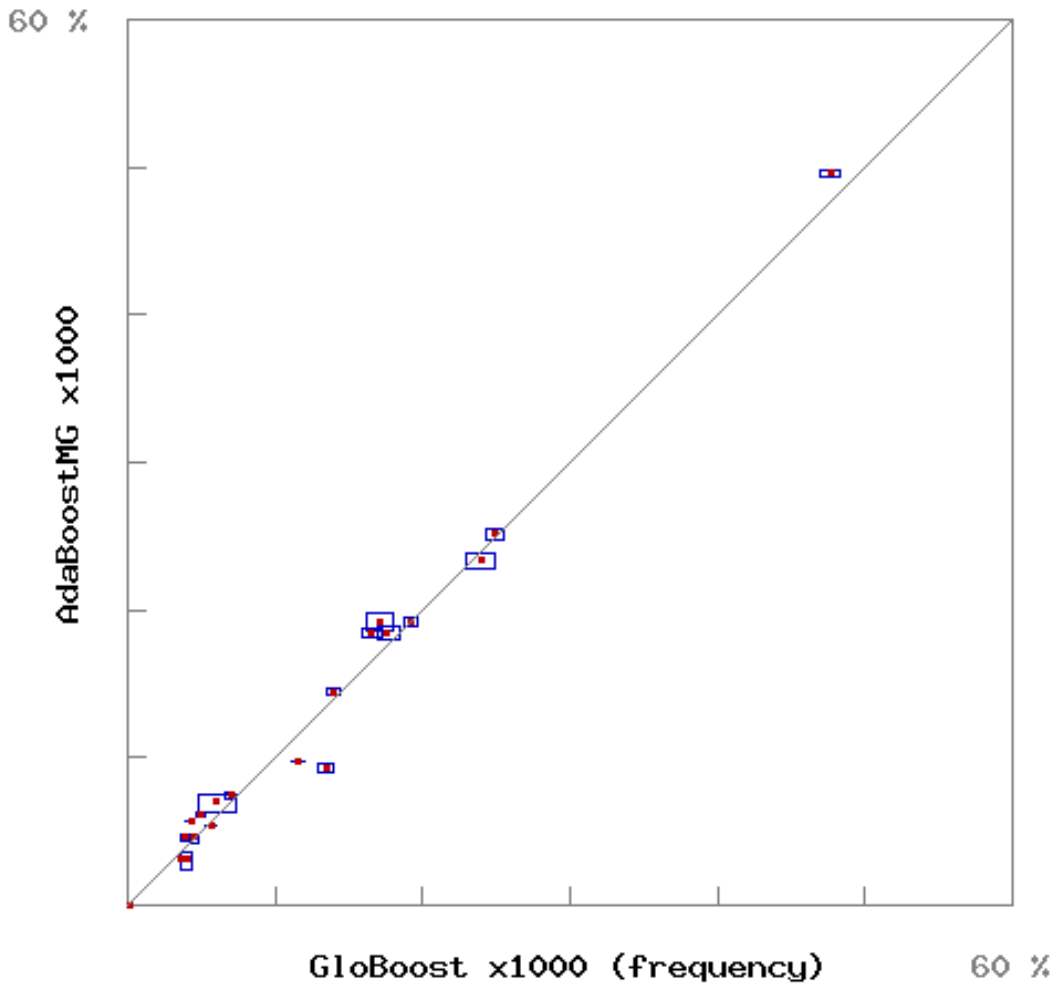
GloBoost vs GloBo



GloBoost vs AdaBoost-stumps



GloBoost vs AdaBoost-MG



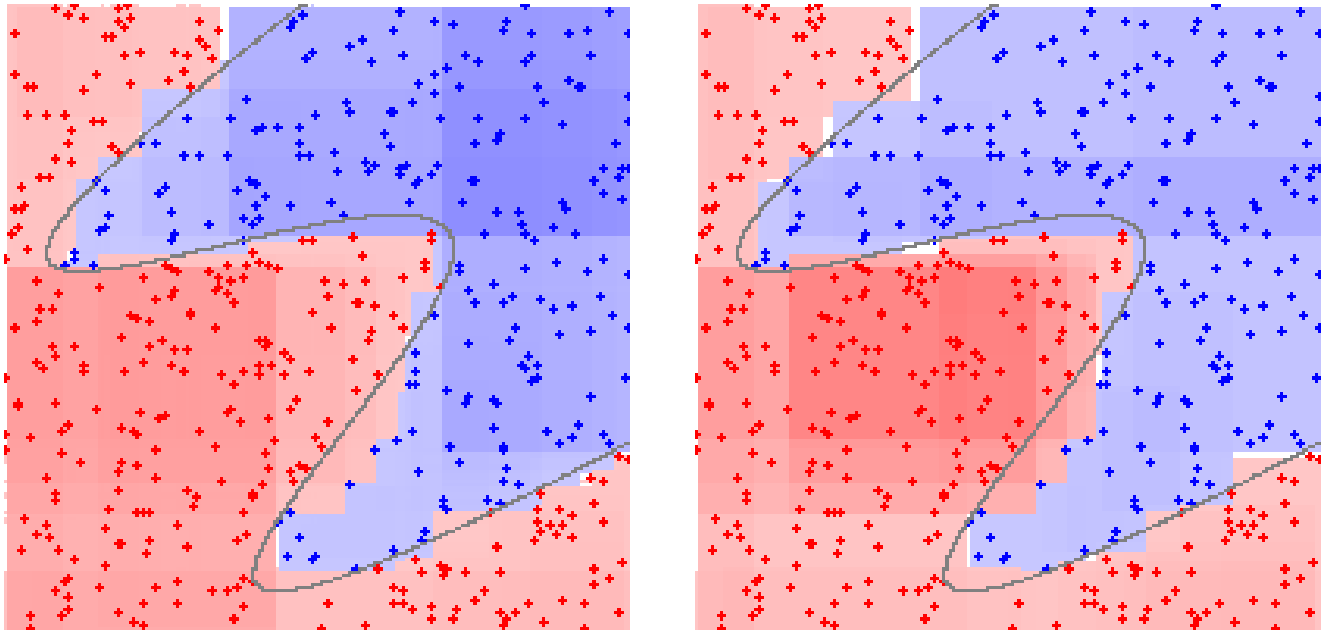
Et une animation en prime...

La création des hypothèses faibles par GloBoost et AdaBoost :

- qui atteint la couverture en premier ?
- insistance portée dans quelles zones ?
- approximation d'une séparatrice sinueuse par des rectangles ?

voir animations

Après 1 000 tours de *boosting* :



Constats

De bons résultats empiriques et pourtant :

- le rasoir d'Occam semble contredit ;
- les hypothèses du cadre PAC ne sont pas vérifiées.

Éléments d'explication :

- biais, variance et erreur empirique ;
- les marges ;
- la diversité.

Explication biais/variance

- Limitation de l'erreur de biais par sortie de $\mathcal{H}$ pour un langage plus expressif : $(\mathcal{H}, \mathbb{R}_+^*)^T$;
- limitation de l'erreur de variance en évitant le choix d'une hypothèse $h \in \mathcal{H}$ particulière.

Faut-il privilégier un $\mathcal{H}$ riche ou pauvre ?

Minimisation de l'erreur empirique

Borne sur l'erreur en généralisation :

$$\text{erreur}(H) \leq e_A(H) + \sqrt{\frac{T.VC(\mathcal{H})}{n}}$$

- *Annuler l'erreur empirique est facile pour les MGC* ;
- on peut s'attendre à une sur-spécialisation si T devient trop grand... compromis à nouveau ;
- (mauvaise) idée : arrêter dès que l'erreur empirique est nulle ;
- on observe que l'erreur en généralisation continue de baisser après que l'erreur empirique se soit stabilisée (à zéro par exemple).

Explication par les marges

Définition : Marge

La marge d'un exemple x est définie par : $m(x) = \frac{\sum_{t=1}^{T} \alpha_t . h_t(x)}{\sum_{t=1}^{T} \alpha_t}$

Avec une probabilité $1 - \delta$, on a pour tout θ :

$$\text{erreur}(H) \leq \Pr(y.m(x) \leq \theta) + \mathcal{O}(\sqrt{\frac{\ln(VC(\mathcal{H}))}{n.\theta^2}})$$

- observation : AdaBoost fait grandir les marges des exemples d'apprentissage, après que l'erreur empirique soit à zéro...
- une interprétation géométrique.

Maximisation des marges

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

Autres travaux liés :

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

Ces méthodes font moins bien que AdaBoost...

Explication par la diversité

Observation de [Dietterich, 2000a] : AdaBoost produit des hypothèses bien plus diverses que les autres méthodes d'ensemble...

...voir graphiques...

- méthode encourageant explicitement la diversité [Melville and Mooney, 2004] ;
- mesures et discussion de la diversité [Kuncheva and Whitaker, 2003] ;
- encourager à faire des erreurs ?
- pour les moindres généralisés par exemple ?

Bilan et suite

Bilan

- méthodes spécifiques à l'attributs-valeurs : C4.5 et moindres généralisés en hyper-rectangles.
- méthodes génériques : MGC, MGLaplace, DLG, GloBo bagging, *boosting*, randomization.

La suite

- séquences et graphes ;
- des méthodes spécifiques ;
- tentative de réutilisation des méthodes génériques.

Bibliographie I


Blake, C. and Merz, C. (1998).
UCI repository of machine learning databases
[http ://archive.ics.uci.edu/ml/].


Breiman, L. (1996).
Bagging predictors.
Machine Learning, 24(2) :123–140.


Breiman, L. (2001).
Random forests.
Machine Learning, 45(1) :5–32.

Bibliographie II


Dietterich, T. G. (2000a).
Ensemble methods in machine learning.
In Kittler, J. and Roli, F., editors, *First International Workshop on Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag.


Dietterich, T. G. (2000b).
An experimental comparison of three methods for constructing ensembles of decision trees : Bagging, boosting, and randomization.
Machine Learning, 40(2) :139–158.


Freund, Y. (1995).
Boosting a weak learning algorithm by majority.
Information and Computation, 121(2) :256–285.

Bibliographie III


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


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

Bibliographie IV


Kuncheva, L. I. and Whitaker, C. J. (2003).
Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy.
Machine Learning, 51(2) :181–207.


Melville, P. and Mooney, R. J. (2004).
Creating diversity in ensembles using artificial data.
Information Fusion : Special Issue on Diversity in Multiclassifier Systems.


Quinlan, R. (2004).
Data mining tools see5 and c5.0.

Bibliographie V


Rätsch, G. and Warmuth, M. K. (2003).
Efficient margin maximizing with boosting.
submitted to Journal of Machine Learning Research (JMLR).


Schapire, R. E. (1990).
The strength of weak learnability.
Machine Learning, 5 :197–227.


Schapire, R. E. and Singer, Y. (1999).
Improved boosting algorithms using confidence-rated predictions.
Machine Learning, 37(3) :297–336.

Bibliographie VI


Schapire, R. E. and Singer, Y. (2000).
Boostexter : A boosting-based system for text categorization.
Machine Learning, 39(2/3) :135–168.


Torre, F. (2005).
Globoost : Combinaisons de moindres généralisés.
Revue d'Intelligence Artificielle, 19(4-5) :769–797.