Méthodes d'ensemble en Inférence Grammaticale, une approche à base de moindres généralisés

Fabien Torre et Alain Terlutte

INRIA-Mostrare et LIFL-Grappa

CAp 2009, Hammamet

Plan de l'exposé

1. Apprentissage et moindre généralisé
   - Définition d'un moindre généralisé (`MG`)
   - Moindre généralisé correct (`MGC`)
   - Classifieur complet (`AdaBoost-MG`)
2. MG en Inférence Grammaticale
   - Objectifs
   - Langages k-TSS et apprentissage
   - Langages 0-réversibles et apprentissage
3. Expérimentations et bilan
   - Protocole et résultats
   - `volata` : bilan et perspectives

Cadre et notations

- Classification supervisée à deux classes $\mathcal{Y} = \{+1, -1\}$;
- représentation des exemples à l'aide de $\mathcal{E}$;
- représentation des hypothèses à l'aide de $\mathcal{H}$;

$$h \in \mathcal{H} : \mathcal{E} \rightarrow \mathcal{Y}$$

- astuce de représentation [Feigenbaum, 1977] : $\mathcal{E} \subset \mathcal{H}$;
- relation de subsomption $\succeq$: $\mathcal{H} \times \mathcal{H}$ et test de subsomption ;
- structuration de $\mathcal{H}$ par $\succeq$.

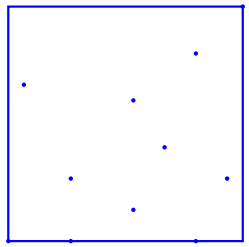
Moindres généralisés

Définition : moindre généralisé

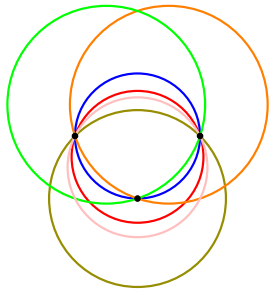
Étant donné un ensemble d'exemples $E \subseteq \mathcal{E}$, une hypothèse $h \in \mathcal{H}$ est dite *moindre généralisée* de E si et seulement si :

- $\forall e \in E : h \succeq e$;
- il n'existe pas h' vérifiant $\forall e \in E : h' \succeq e$ et $h \succ h'$.

- $\mathcal{E} = \mathbb{R}^2$;
- les hypothèses sont des rectangles.



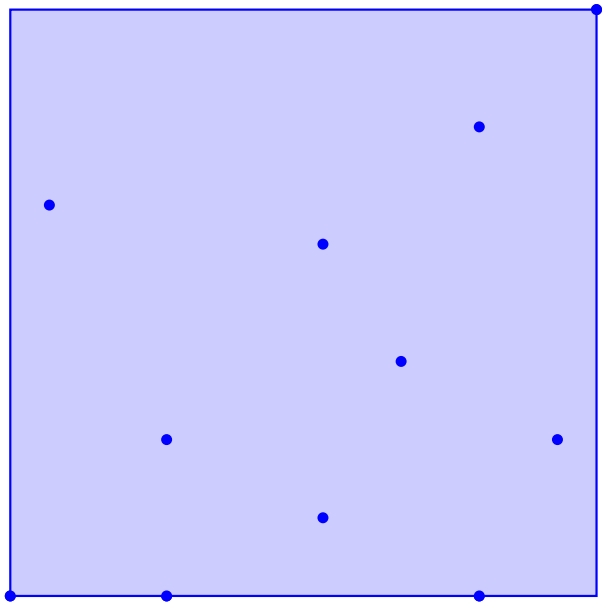
- $\mathcal{E} = \mathbb{R}^2$;
- les hypothèses sont des cercles.



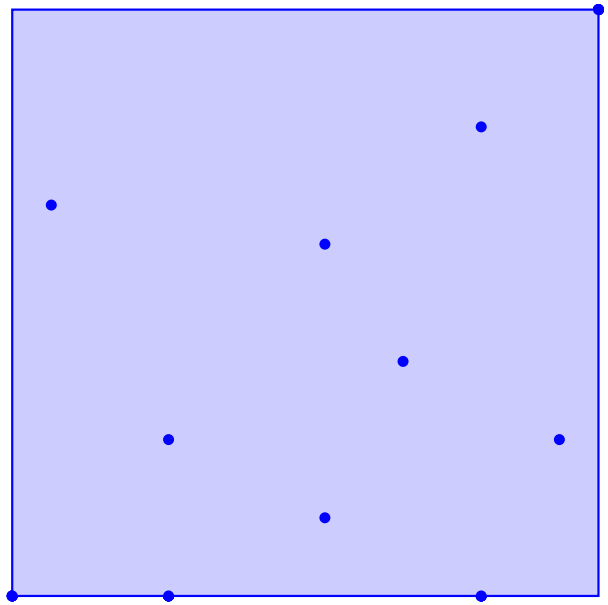
Un exemple et deux profils

$\mathcal{E} = \mathbb{R}^2$, les hypothèses sont des rectangles.

MG $(e_1, e_2, \ldots, e_n \in \mathcal{E})$ returns $h \in \mathcal{H}$



MG (h_{n-1}, e_n) returns $h \in \mathcal{H}$

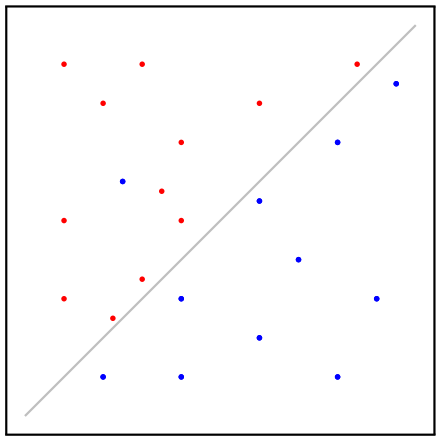


on préfère la version incrémentale.

Caractéristiques des moindres généralisés

La définition n'implique pas l'unicité... mais l'unicité amène :

1. monotonie : inclusion des hypothèses successives ;
2. minimalité / plus petit pas ;
3. parcours sans risque, guidé par les exemples ; ne reste que le biais de langage (le choix de $\mathcal{H}$) ;
4. bonus algorithmique si l'on a un calcul incrémental.



- On sait généraliser une classe. ;
- suite : prendre en compte les autres classes.

MGC [Torre, 1999] : idée du calcul

Objectif : apprendre une règle correcte pour une classe.

une graine et sa classe	exemples de même classe	généralisation maximalement correcte
x_1, y_1	x_5 x_8 ~~x_2~~ $x_{14}\ldots$	$g_1 = \mathtt{MG}(x_1, x_5, x_8, x_{14}, \ldots)$ $g_1 \rightarrow y_1$
x_2, y_2	~~x_{14}~~ x_3 ~~x_{10}~~ $x_{12}\ldots$	$g_2 = \mathtt{MG}(x_2, x_3, x_{12}, \ldots)$ $g_2 \rightarrow y_2$

- instable : dépend de la graine et de l'ordre des exemples ;
- pour un exemple donné, un moindre généralisé correct conclut sur une unique classe (-1 ou $+1$) ou s'abstient (0).

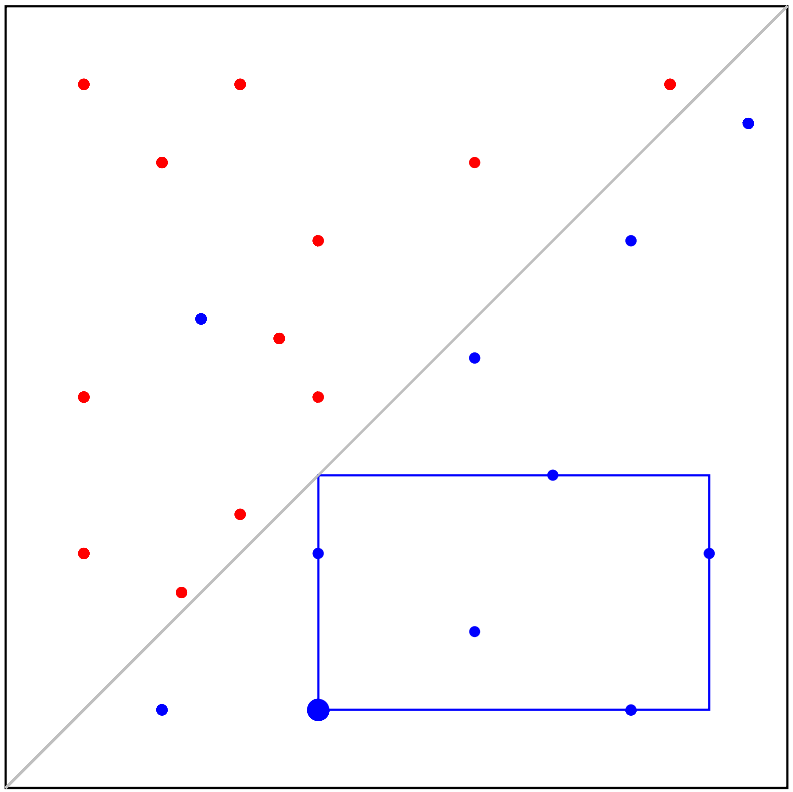
MGC : l'algorithme

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

Sortie : $h \in \mathcal{H}$ une généralisation de E, maximalement correcte par rapport à E et N.

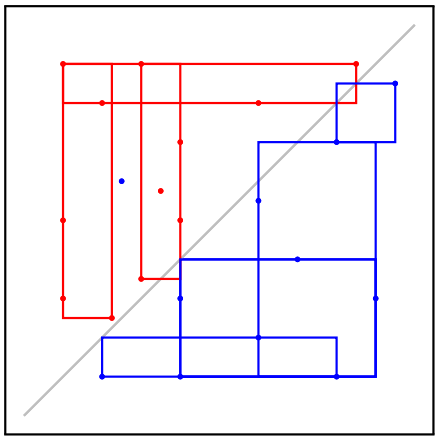
1: $g = e_1$
2: **for** $i = 2$ to n **do**
3: $g' = \mathtt{MG}(g, e_i)$ {généralisation visant à intégrer e_i} {si g' est correcte}
4: **if** $(\forall e \in N : g' \not\succeq e)$ **then**
5: $g = g'$ {g' devient la généralisation courante}
6: **end if**
7: **end for**
8: **return** $h(x) = \text{class}(E)$ si $g \succeq x$, 0 sinon (abstention)

MGC : déroulement



Caractéristiques de MGC

1. MGC est générique (à instancier par $\mathcal{H}$, $\succeq$ et MG) ;
2. une hypothèse apprise par MGC ne se trompe pas sur l'ensemble d'apprentissage, mais elle peut s'abstenir ;
3. MGC est sensible à l'ordre de présentation des positifs, la graine et les premiers sont favorisés ;
4. un positif rejeté lors de MGC ne peut pas revenir ;
5. MGC distingue les concepts conjonctifs des disjonctifs.



- On suppose donné l'algorithme MG ;
- on sait obtenir une unique règle avec MGC. MGC ;
- suite : construire un classifieur complet.

Boosting [Freund, 1995]

Principes :

- Supposer donné A, un apprenant (faible) ;
- jouer sur la distribution des exemples d'apprentissage pour obtenir de A des hypothèses différentes ;
- à l'étape t, A apprend $h_t \in \mathcal{H}$ selon la distribution courante, les poids des exemples mal classés par h_t sont augmentés, les poids des exemples bien classés par h_t sont diminués ;

Algorithmes :

- AdaBoost, version pour hypothèse faible capable d'abstention [Schapire and Singer, 1999] ;
- AdaBoost-MG : version modifiée pour intégrer MGC en tant qu'apprenant de base.

AdaBoost-MG : l'algorithme 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: $N = \{x_i | y_i \neq target\}$
{prise en compte de la distribution}
8: trier P par poids décroissants
9: $h_t =$ MGC (P,N) {appel à l'algorithme MGC }

AdaBoost-MG : l'algorithme II

10: $W_+ = \sum_{i:y_i h_t(x_i)>0} w_i$

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

12: $Z_t = \sum_{i}^{n} [w_i.\exp(-\alpha_t y_i h_t(x_i))]$

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

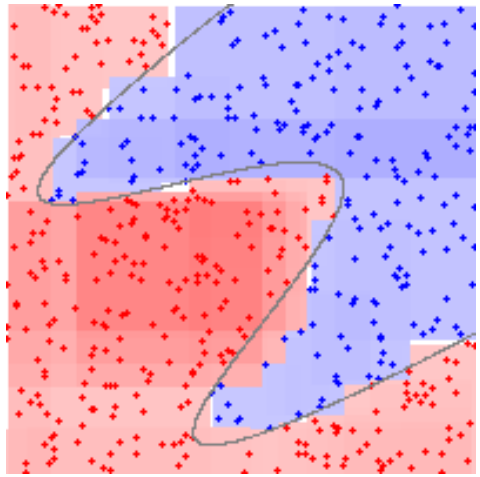
14: $w_i = \frac{w_i.\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

15: **end for**

16: **end for**

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

Bilan AdaBoost-MG



1. On a changé d'espace ($\mathcal{H}^T$), on a gagné en expressivité ;
2. l'erreur empirique s'annule très vite car les hypothèses ne se trompent pas sur l'échantillon d'apprentissage ;
4. un apprenant *parfait* ne peut pas être boosté : un moindre généralisé doit donc s'abstenir, la cible doit être disjonctive...
5. cela suggère de limiter l'expressivité de $\mathcal{H}$; en particulier en enlever les disjonctions ? car la disjonction est au niveau de la théorie.

Application à l'inférence grammaticale

Architecture à trois niveaux, seul le premier est à instancier :

1. langages et opérations associées :
   - les exemples ($\mathcal{E}$) : des mots sur un alphabet Σ ;
   - les hypothèses ($\mathcal{H}$) : les automates d'une classe particulière ;
   - $h \succeq e \Leftrightarrow e \in L(h)$;
   - `MG` : apprentissage par positifs seuls, du plus petit langage de la classe contenant les mots donnés, en version incrémentale ;
2. utilisation des classes, production d'hypothèses correctes (`MGC`) ;
3. apprentissage d'un classifieur complet, par combinaison de règles élémentaires apprises par `MGC` (`AdaBoost-MG`).

k-TSS [García and Vidal, 1990] et k-réversibles [Angluin, 1982].

Langages k-TSS [García and Vidal, 1990]

Définition : langage k testable au sens strict ($k \geq 1$)

Un langage k-TSS, k étant fixé, est défini par :

- I les segments initiaux autorisés u ($|u| = k - 1$) ;
- F les segments finaux autorisés v ($|v| = k - 1$) ;
- T les segments interdits, tous de taille égale à k ;
- W les mots w du langage tels que $|w| < k - 1$) ;

Formellement :

$$L(\Sigma, I, F, T, W) = W \cup (I\Sigma^* \cap \Sigma^* F) \setminus \Sigma^* T \Sigma^*$$

Adaptation de l'algorithme en version incrémentale : `MG-TSSI`.

MG-TSSI : l'algorithme I

Clef : chaque état est identifié par la séquence des $k-1$ dernières lettres lues pour arriver à cet état.

Algorithme MG-TSSI (h, w, k)

Entrées : $h = (Q, \Sigma, q_0, F)$ un k-TSS, w un mot et un entier k
Sortie : h' un k-TSS minimal généralisant h et couvrant w

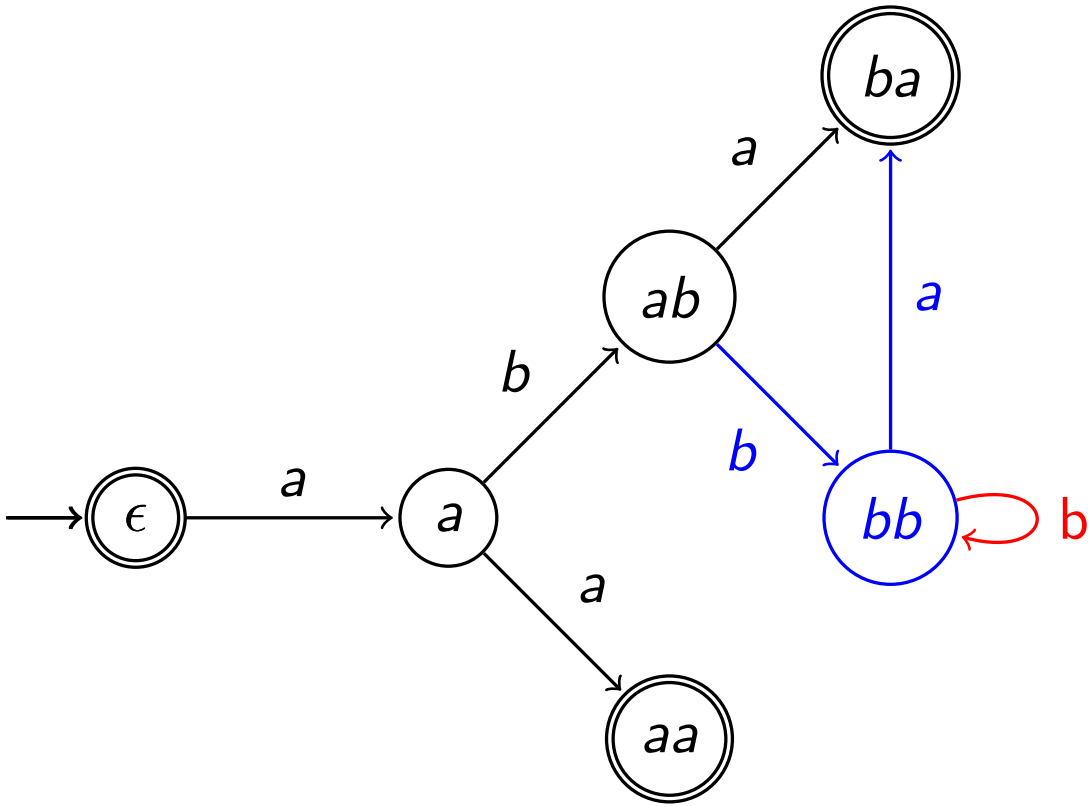
1: $q = q_0$
2: **for** $i = 1$ to $|w|$ **do**
3: $\quad v = q.w_i$ {concat. du mot de q avec la i^e lettre de w}
4: $\quad$ **if** $(|v| = k)$ **then**
5: $\quad\quad v = v_{2,\ldots,|v|}$
6: $\quad$ **end if**

MG-TSSI : l'algorithme II

7: $nq = v$
8: ajouter nq à Q {nq peut déjà être présent dans Q}
9: ajouter (q, w_i, nq) à δ
10: $q = nq$
11: **end for**
12: ajouter q à F
13: **return** $h' = (Q, \Sigma, \delta, q_0, F)$

MG-TSSI : déroulement

1. Échantillon $S = \{\epsilon, aa, aba\}$ et $k = 3$;
2. automate inféré :



3. ajout de *abba* ; ajout de *abbba*.

Automates 0-réversibles [Angluin, 1982]

Définition : 0-réversible

Un automate A est 0-réversible s'il est déterministe dans les deux sens : A est déterministe et son miroir A^R l'est aussi.

L'algorithme ZR de [Angluin, 1982] :

1. construit le PTA de l'échantillon ;
2. le généralise pour le rendre 0-réversible.

Adaptation de l'algorithme en version incrémentale : MG-ZR.

MG-ZR : l'algorithme I

Algorithme MG-ZR (h, w)

Entrées : $h = (Q, \Sigma, q_0, F)$ un automate 0-réversible, w un mot

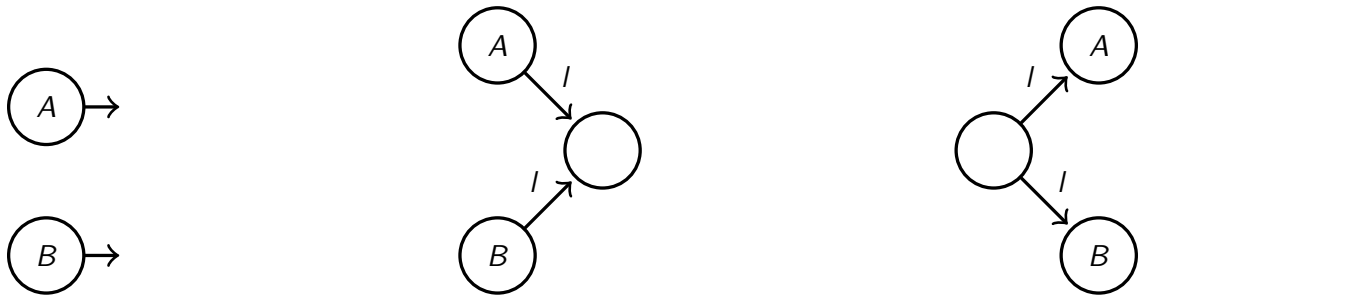
Sortie : h' un 0-réversible minimal généralisant h reconnaissant w

1: $i = 1$; $q = q_0$
{suivi des états existants}
2: **while** $\delta(q, w_i)$ is defined **do**
3: $q = \delta(q, w_i)$; $i = i + 1$
4: **end while**
{création d'une nouvelle branche pour les lettres restantes}
5: **while** $i \leq |w|$ **do**
6: créer un état q' et l'ajouter à Q
7: ajouter (q, w_i, q') à δ ; $i = i + 1$
8: **end while**
9: ajouter q à F

MG-ZR : l'algorithme II

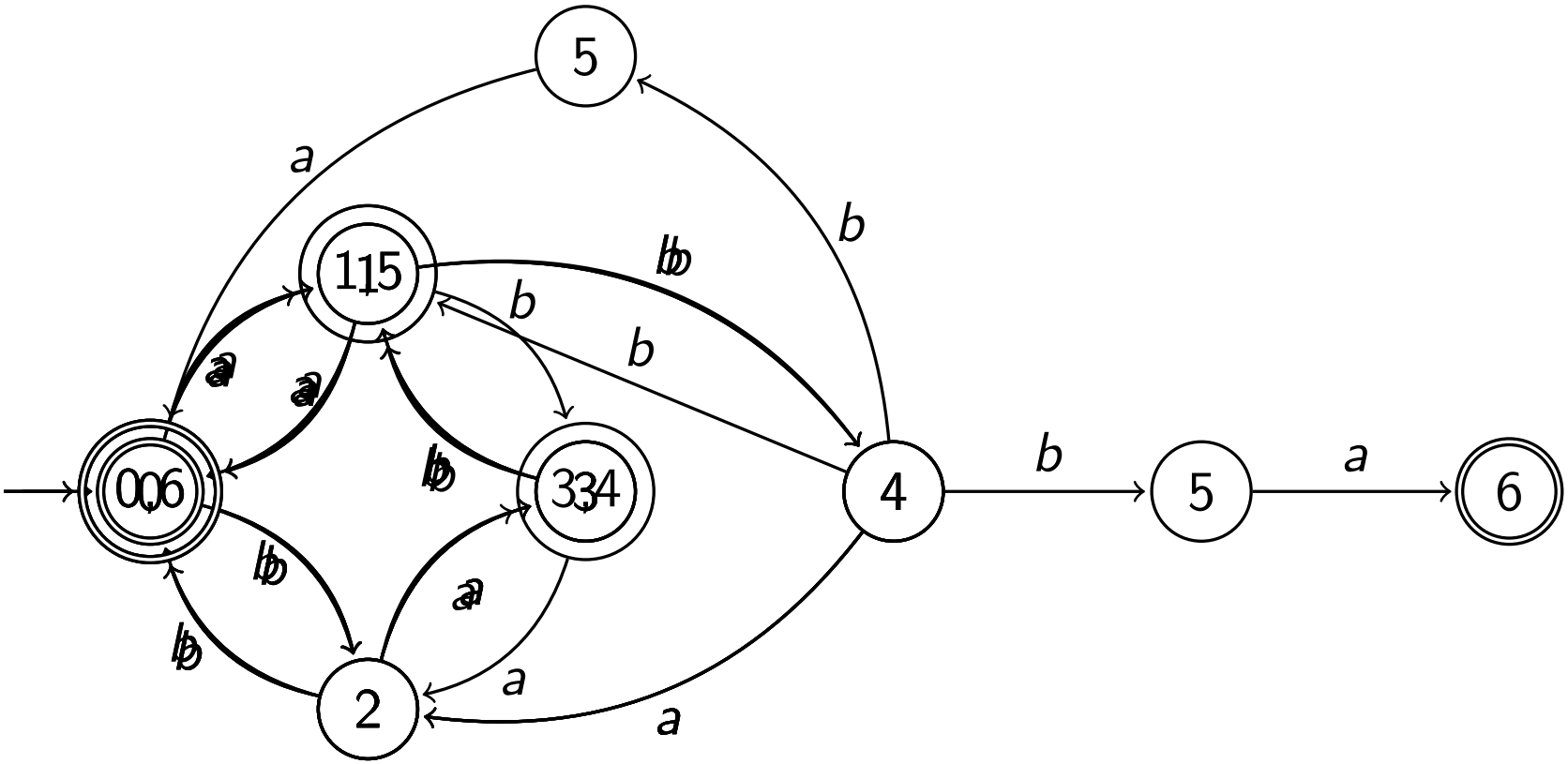
{mise en œuvre des fusions}
10: fusionner les états de F {deux états au plus}
11: **repeat**
12: si $\exists A, B \in Q$ et $\exists l \in \Sigma$ tels que $\delta(A, l) = \delta(B, l)$, fusionner A et B
13: si $\exists A, B, E \in Q$ et $\exists l \in \Sigma$ tels que $\delta(E, l) = \{A, B\}$, fusionner A et B
14: **until** no fusion
15: return $h' = (Q, \Sigma, \delta, q_0, F)$

Situations dans lesquelles les états A et B sont fusionnées par ZR :



MG-ZR : déroulement

$S = \{\epsilon, aa, bb, abab, baba\}$, ajout du mot *abba*.



MG en inférence grammaticale : bilan

- On récupère deux classes de langages : les k-TSS [García and Vidal, 1990] et les k-réversibles [Angluin, 1982] ;
- elles disposent de résultats d'apprenabilité par positifs seuls ;
- les algorithmes sont des calculs de moindres généralisés.

On peut maintenant apprendre à classer des mots en bénéficiant :

- du traitement naturel des disjonctions d'automates (`MGC`) ;
- de méthodes d'ensemble (`AdaBoost-MG`) ;
- du maintien d'une certaine lisibilité de la théorie apprise ;
- d'un gain d'expressivité par rapport aux classes utilisées.

Protocole expérimental

- Combinaisons entre `AdaBoost-MG` et (`MG-TSSI`, `MG-ZR`) ;
- comparées à `Red-Blue` [Lang et al., 1998] ;
- jeux de données (à deux classes) en provenance de l'*UCI repository* [Blake and Merz, 1998] et `soccer` ;
- validation croisée 10 fois, pour les algorithmes stochastiques, chaque exécution est répétée 10 fois ;
- 1 000 itérations pour les méthodes d'ensemble ;
- moyenne des taux de bonne classification en test.

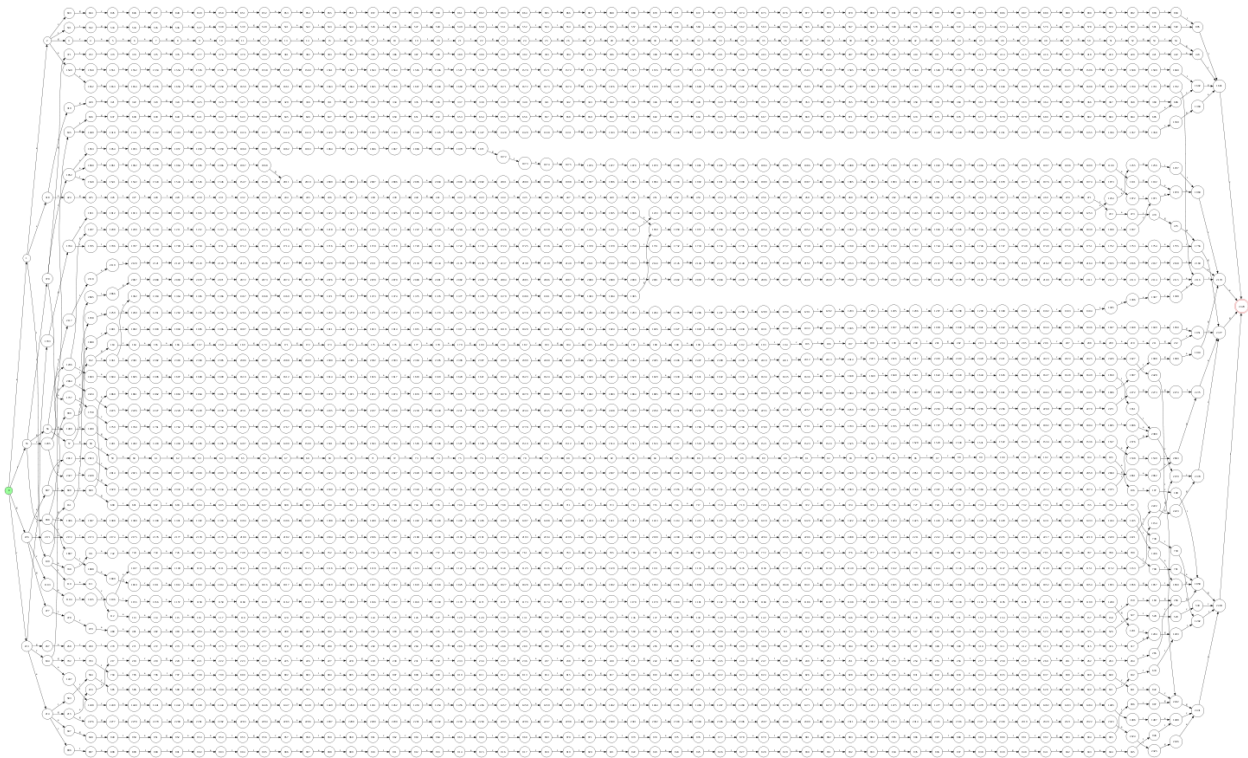
Résultats

	Majorité	Red-Blue	AdaBoost-MG	
			MG-TSSI	MG-ZR
tic-tac-toe	65.34 %	93.89 %	90.27 %	**98.31 %**
badges	71.43 %	61.09 %	**73.47 %**	-
promoters	50.00 %	**63.02 %**	61.51 %	50.00 %
soccer	50.00 %	63.06 %	**68.89 %**	56.81 %
us-first-name	81.62 %	82.83 %	**86.10 %**	83.92 %
splice	50.26 %	54.65 %	**79.79 %**	-

Bilan de ces expérimentations

- `AdaBoost-MG` + `MG-TSSI` obtient les meilleurs résultats ;
- `AdaBoost-MG` présente un comportement déjà observé en attributs-valeurs ;
- résultats décevants des combinaisons de 0-réversibles.

Apprentissage de `MG-ZR` sur la classe + des `promoters`.



volata

Implémentation de notre architecture à trois niveaux :

1. moindres généralisés ;
   - attributs-valeurs : plus petit rectangle ;
   - inférence grammaticale : `MG-TSSI`, `MG-ZR`, `MG-REG` ;
2. validation des généralisations :
   - correction ;
   - précision de Laplace [Clark and Boswell, 1991] ;
3. apprenants complets :
   - `DLG` [Webb and Agar, 1992] ;
   - `GloBo` [Torre, 1999] ;
   - `GloBoost` [Torre, 2005], `Bagging-MG` [Breiman, 1996] et `AdaBoost-MG` [Schapire and Singer, 1999].

`https://fabien-torre.fr/Recherche/Softwares/volata/`

Perspectives

1. Rechercher des langages et calculs de moindres généralisés :
   - limiter les disjonctions des automates : les *monômes réguliers* ;
   - proposer des solutions si moindres généralisés multiples : étude des *boules de mots* [de la Higuera et al., 2008] avec Frédéric Tantini et des *monômes réguliers* ;
   - trouver une classe atteignant les rationnels par union ;
   - passer aux *langages d'arbres et de graphes* ;

2. tester la précision de Laplace comme critère de validation :
   - résistance au bruit ?
   - diversité pour `AdaBoost` ?

Les bonus

- `MG` multiples ;
- accuracy de Laplace ;
- `DLG`, `GloBo`, `GloBoost`, `Bagging-MG` ;
- Fernau et `MG-REG` ;
- toutes les expériences.

MG multiples

Pistes à envisager :

1. trouver un calcul qui ait un maximum de propriétés des MG ;
2. choisir l'un des MG, plus ou moins arbitrairement ;
3. montrer l'unicité de la généralisation entre une hypothèse et un exemple ;
4. briser des liens dans la relation de subsomption pour retrouver un MG unique ;
5. prendre la conjonction des MG ;
6. prendre la disjonction des MG.

retour aux bonus

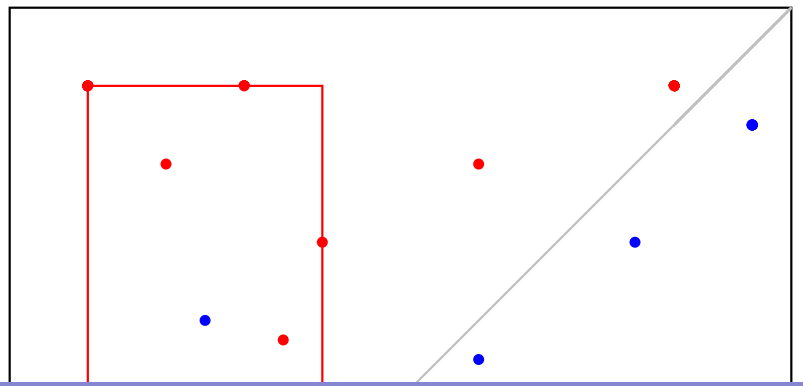
Bruit et précision de Laplace [Clark and Niblett, 1989]

Idée

Autoriser la couverture d'exemples d'autres classes : trouver un compromis entre le nombre total d'exemples couverts par une règle (t) et le nombre d'exemples bien classés par cette règle (b).

Mesures

$$\begin{aligned} \text{Précision} &= \frac{b}{t} = \frac{1}{1} = 100\% = \frac{2}{2} = 100\% = \frac{7}{8} = 87.5\% = \\ \text{Précision de Laplace} &= \frac{b+1}{t+k} = \frac{1+1}{1+2} = 66.67\% = \frac{2+1}{2+2} = 75\% = \frac{7+1}{8+2} = \end{aligned}$$



Données bruitées (2)

Solution pour le bruit

- on réclamait un maintien absolu de la correction ;
- on veut maintenant que la précision de Laplace aille croissante.

$$\text{Précision de Laplace} = \frac{b+1}{t+k}$$

Le critère de validation d'une généralisation dans `MGC` devient :

1: **if** (PrecisionLaplace(h') $\geq$ PrecisionLaplace(h)) **then**
2: $h = h'$
3: **end if**

retour aux bonus

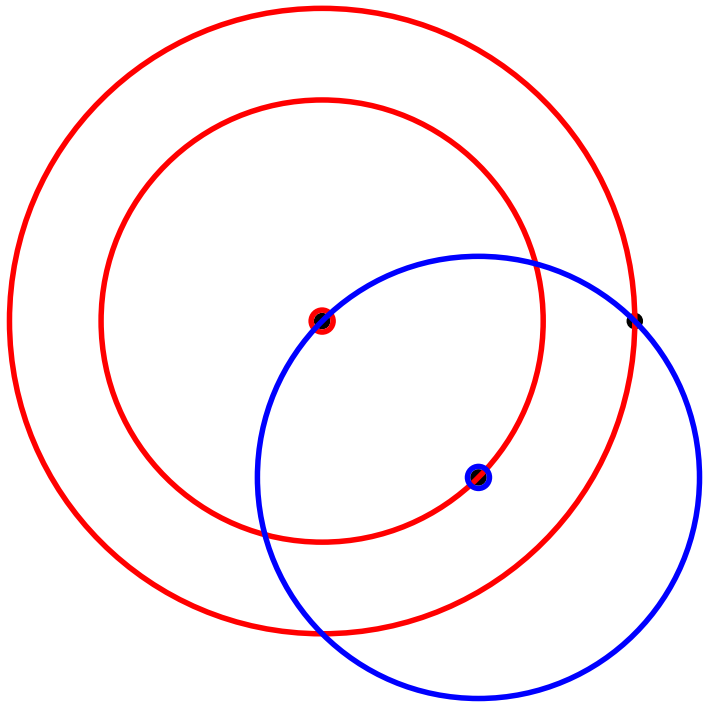
Des hypothèses-cercles

- $\mathcal{E} = \mathbb{R}^2$;
- $d(A, B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$;
- $\mathcal{H} = \{(c, r) | (c \in \mathbb{R}^2) \text{ et } (r \in \mathbb{R}^+)\}$;
- $h \succeq e \Leftrightarrow d(c_h, e) \leq r_h$.

Unicité et calcul du moindre généralisé ?

Cercles : unicité algorithmique

Idée : ...



retour aux bonus

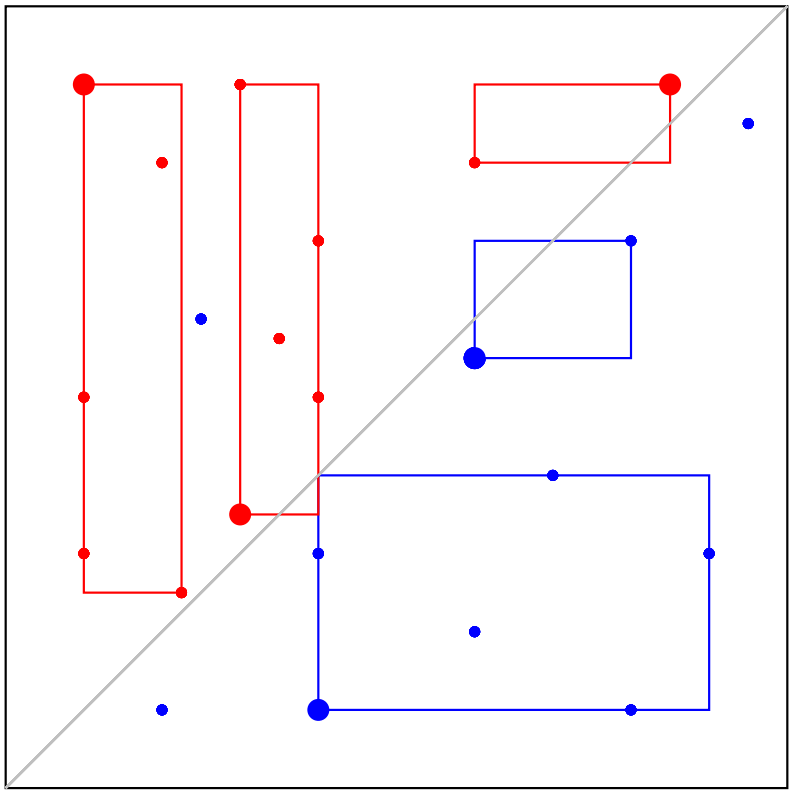
DLG [Webb and Agar, 1992] : l'algorithme

Principe : couverture gloutonne des exemples, on répète le calcul de moindre généralisé correct sur les exemples non couverts jusqu'à couverture complète des exemples.

Entrées : A un ensemble de n exemples (x_i, y_i).
Sortie : H le classifieur final.

1: $O = A$; $t = 0$
2: **while** $(O \neq \emptyset)$ **do**
3: $target =$ classe du premier exemple de O
4: $P = [x_i \in O | y_i = target]$
5: $N = [x_i \in A | y_i \neq target]$
6: $h_t =$ `MGC` (P,N) {appel à l'algorithme `MGC` }
7: $O = O - [x \in O : h_t \succeq x]$
8: $t = t + 1$
9: **end while**
10: **return** $H(x) = \mathrm{ArgMax}_k |\{h_t : h(x) = k\}|$

DLG : déroulement



retour aux bonus

GloBo [Torre, 1999] : principes

Objectif : combattre la dépendance à l'ordre des exemples et chercher la compréhensibilité.

Principes de GloBo

1. calculer plusieurs moindres généralisés en utilisant des exemples différents comme graine et des exemples de la même classe mélangés ;
2. retenir les règles qui permettent une couverture minimale des exemples.

Si chaque exemple sert à un moment de graine, alors au final tout exemple est couvert par au moins une hypothèse.

GloBo : l'algorithme I

Entrées : A un ensemble de n exemples (x_i, y_i).
Sortie : H le classifieur final.

1: $R' = \emptyset$ {un appel à `MGC` par exemple de A}
2: **for** $i = 1$ to n **do**
3: $P = [x_j | y_j = y_i \wedge i \neq j]$
4: $N = [x_j | y_j \neq y_i]$
5: mélanger **P** aléatoirement
6: $h_i =$ `MGC` $(x_i :: P, N)$ {x_i en tête des positifs}
7: ajouter h_i à R'
8: **end for**

GloBo : l'algorithme II

{couverture minimale}

$R = \emptyset$

while $(\exists x_i, \forall h_j \in R, h_j \not\succeq x_i)$ **do**

    $h = \mathrm{ArgMax}_{h_i \in R'} |[x_j : h_i \succeq x_j \wedge \nexists h_k \in R : h_k \succeq x_j]|$

    ajouter h à R

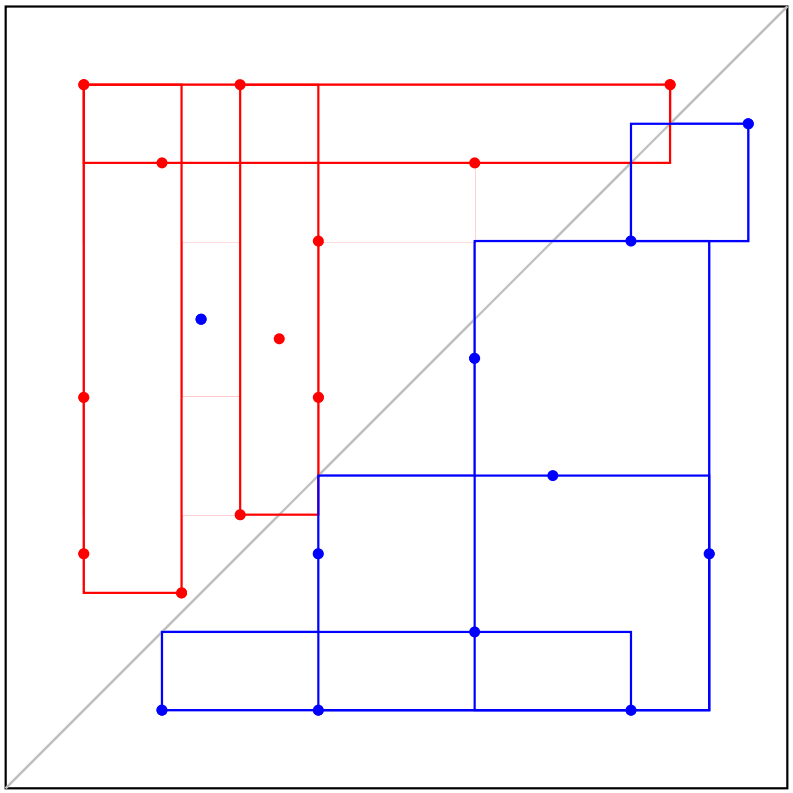
end while

return $H(x) = \mathrm{ArgMax}_k |\{h \in R : h(x) = k\}|$

Couverture minimale : problème NP-complet.

Heuristique quadratique : les règles les plus couvrantes d'abord.

GloBo : déroulement



... puis calcul de la couverture minimale.

... et si l'on gardait toutes les règles produites ?

▸ retour aux bonus

Méthodes aléatoires : principes

Principes

- ne pas toucher à l'échantillon (*bagging*) ;
- ne pas jouer sur la distribution (*boosting*) ;
- mais simplement rendre l'apprenant instable (*randomization*) ;
- puis voter à égalité.

Exemple : les arbres de décision aléatoires de [Dietterich, 2000].

GloBoost [Torre, 2005] : l'algorithme

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: $h_t =$ `MGC` (P,N) {appel à l'algorithme `MGC` }
7: **end for**
8: **return** $H(x) = \text{sign}\left(\sum_{t=1}^{T} h_t(x)\right)$

retour aux bonus

Méthode d'ensemble

Idées générales

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

Éléments constitutifs

Une méthode d'ensemble se définit par :

- $\mathcal{H}$ et l'apprenant faible L ;
- la méthode de production des T hypothèses h_t issues de $\mathcal{H}$;
- la stratégie d'attribution des poids α_t aux h_t.

Sortie

Une méthode d'ensemble fournit un classifieur H :

- $H = [(h_t, \alpha_t)]_{t \in [1,T]} \in (\mathcal{H} \times \mathbb{R}^*)^T$ on a changé d'espace

Bagging [Breiman, 1996] : principes

Points de départ

- 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.

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 E tiré aléatoirement dans l'échantillon complet A ($|E| = |A|$) ;
- poids identique pour chaque h_t.

Bagging-MG : l'algorithme

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: $h_t =$ `MGC` (P,N) {appel à l'algorithme `MGC` }
10: ajouter h_t à H
11: **end for**
12: **return** H

retour aux bonus

[Fernau, 2005] : regroupements, alignement, généralisation

1. Échantillon : $S = \{ababb, aabb, ababa, abc\}$
2. alignement :

$$\begin{array}{cccccc} (a) & (b) & (a) & (bb) & & \\ (aa) & (bb) & & & & \\ (a) & (b) & (a) & (b) & (a) & \\ (a) & (b) & (c) & & & \end{array}$$

3. généralisation

$$\begin{array}{cccc} a^+b^+(a & & & |\epsilon|c) \\ a^+b^+(a & b^+ & & |\epsilon|c) \\ a^+b^+(a & b^+ & (\epsilon|a) & |\epsilon|c) \end{array}$$

4. résultat : $a^+.b^+.(a.b^+.(\epsilon|a)|\epsilon|c)$.

MG-REG

1. Échantillon : $S = \{ababb, aabb, ababa, abc\}$
2. alignement :

$$\begin{array}{cccccc} (a) & (b) & (a) & (bb) & & \\ (aa) & (bb) & & & & \\ (a) & (b) & (a) & (b) & (a) \end{array}$$

3. généralisation : $a^{\{1,2\}} b^{\{1,2\}} a^{\{0,1\}} b^{\{0,2\}} a^{\{0,1\}}$;
4. ajout de abc : $a^{\{1,2\}} b^{\{1,2\}} .^{\{0,1\}} b^{\{0,2\}} a^{\{0,1\}} .$

▸ retour aux bonus

Paramétrages

problem	k-values
tic-tac-toe	1-2-3-4-5
badges	1-2
us-first-name	1-2-3-4
Lyon-Rennes	1-2-3-4
promoters	1-2-3-4
splice	1-2-3

retour aux bonus

problem-plearner	defaults
tic-tac-toe TSSI	positive
tic-tac-toe ZR	positive
tic-tac-toe Fernau	negative
badges TSSI	positive
badges ZR	positive
badges Fernau	negative
us-first-name TSSI	female
us-first-name ZR	female
us-first-name Fernau	female
Lyon-Rennes TSSI	Rennes
Lyon-Rennes ZR	Rennes
Lyon-Rennes Fernau	Lyon
promoters TSSI	positive
promoters ZR	-
promoters Fernau	negative
splice TSSI	EI
splice ZR	EI ?
splice Fernau	IE

Variations sur la taille d'un échantillon

Problème `tic-tac-toe`.

% en apprentissage	90 %	50 %	25 %	10 %
`Red-Blue`	93.89 %	85.07 %	73.04 %	68.29 %
`AB-MG` + `MG-TSSI`	90.27 %	82.33 %	76.29 %	69.57 %
`GloBoost` + `MG-ZR`	98.36 %	98.00 %	93.20 %	66.96 %
`GloBo` + `MG-REG`	99.77 %	99.23 %	93.38 %	72.28 %

▸ retour aux bonus

tic-tac-toe

Références :

Majorité	RPNI	TraxBar	Red-Blue
65.34 %	91.13 %	90.81 %	**93.89 %**

Combinaisons d'automates :

	DLG	GloBo	GloBoost	AdaBoost-MG
MG-TSSI	79.02 %	81.43 %	91.47 %	90.27 %
MG-ZR	96.76 %	95.84 %	98.36 %	98.31 %
MG-REG	95.20 %	99.77 %	98.56 %	**99.61 %**

badges

Références :

Majorité	RPNI	TraxBar	Red-Blue
71.43 %	62.24 %	57.48 %	61.09 %

Combinaisons d'automates :

	DLG	GloBo	GloBoost	AdaBoost-MG
MG-TSSI	73.61 %	72.24 %	72.69 %	73.47 %
MG-ZR	71.43 %	71.43 %	71.43 %	-
MG-REG	98.30 %	99.93 %	95.10 %	**100.0 %**

promoters

Références :

Majorité	RPNI	TraxBar	Red-Blue
50.00 %	- %	56.60 %	**63.02 %**

Combinaisons d'automates :

	DLG	GloBo	GloBoost	AdaBoost-MG
MG-TSSI	53.53 %	60.00 %	61.13 %	61.51 %
MG-ZR	50.00 %	50.00 %	50.00 %	50.00 %
MG-REG	69.81 %	70.85 %	**86.23 %**	85.66 %

soccer

Références :

Majorité	RPNI	TraxBar	Red-Blue
50.00 %	62.50 %	58.33 %	**63.06 %**

Combinaisons d'automates :

	DLG	GloBo	GloBoost	AdaBoost-MG
MG-TSSI	64.31 %	59.17 %	72.08 %	68.89 %
MG-ZR	52.78 %	50.00 %	53.19 %	56.81 %
MG-REG	69.44 %	65.28 %	**73.33 %**	68.33 %

us-first-name

Références :

Majorité	RPNI	TraxBar	Red-Blue
81.62 %	81.42 %	81.37 %	**82.83 %**

Combinaisons d'automates :

	DLG	GloBo	GloBoost	AdaBoost-MG
MG-TSSI	84.19 %	83.83 %	**89.14** %	86.10 %
MG-ZR	81.77 %	81.97 %	83.07 %	83.92 %
MG-REG	87.75 %	88.40 %	88.28 %	87.32 %

splice

Références :

Majorité	RPNI	TraxBar	Red-Blue
50.26 %	- %	**58.33 %**	54.65 %

Combinaisons d'automates :

	DLG	GloBo	GloBoost	AdaBoost-MG
MG-TSSI	67.61 %	73.42 %	78.07 %	79.79 %
MG-ZR	-	-	-	-
MG-REG	85.57 %	89.93 %	93.46 %	**94.91 %**

retour aux bonus

Bibliographie I

retour aux bonus


Angluin, D. (1982).
Inference of reversible languages.
Journal of the ACM, 29(3) :741–765.


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.

Bibliographie II


Clark, P. and Boswell, R. (1991).
Rule induction with CN2 : Some recent improvements.
In *Proc. Fifth European Working Session on Learning*, pages 151–163, Berlin. Springer.


Clark, P. and Niblett, T. (1989).
The cn2 induction algorithm.
Machine Learning, 3(4) :261–283.


de la Higuera, C., Janodet, J.-C., and Tantini, F. (2008).
Learning balls of strings from edit corrections.
Journal of Machine Learning Research, 9 :1823–1852.

Bibliographie III


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


Feigenbaum, E. A. (1977).
The art of artificial intelligence : Themes and case studies of knowledge engineering.
In Reddy, R., editor, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 1014–1029. Morgan Kaufmann.

Bibliographie IV


Fernau, H. (2005).
Algorithms for learning regular expressions.
In Jain, S., Simon, H.-U., and Tomita, E., editors, *Algorithmic Learning Theory, 16th International Conference, ALT 2005, Singapore, October 8-11, 2005, Proceedings*, volume 3734 of *Lecture Notes in Computer Science*, pages 297–311. Springer.


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


García, P. and Vidal, E. (1990).
Inference of k-testable languages in the strict sense and application to syntactic pattern recognition.
IEEE Trans. Pattern Anal. Mach. Intell., 12(9) :920–925.

Bibliographie V


Lang, K. J., Pearlmutter, B. A., and Price, R. A. (1998).
Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm.
In *ICGI '98 : Proceedings of the 4th International Colloquium on Grammatical Inference*, pages 1–12, London, UK. Springer-Verlag.


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

Bibliographie VI


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


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


Webb, G. I. and Agar, J. W. M. (1992).
Inducing diagnostic rules for glomerular disease with the DLG machine learning algorithm.
Artificial Intelligence in Medicine, 4 :419–430.

retour aux bonus