Adaptation du boosting à l'inférence grammaticale via l'utilisation d'un oracle de confiance ⋆

Jean-Christophe Janodet[1], Richard Nock[2], Marc Sebban[1], Henri-Maxime Suchier[1]

[1] Equipe Universitaire de Recherche en Informatique de Saint-Etienne
{janodet,Marc.Sebban,Henri.Maxime.Suchier}
@univ-st-etienne.fr

[2] Département Scientifique Inter-facultaire, Université Antilles-Guyane
rnock@martinique.univ-ag.fr

Résumé : Cet article présente une adaptation du boosting à l'inférence grammaticale. Notre but est d'améliorer les performances d'un algorithme à base de fusion d'états, en présence de données bruitées. Notre algorithme de boosting utilise une nouvelle règle de mise à jour des poids qui tient compte d'une information supplémentaire fournie par un oracle. Cette information est une évaluation de la confiance en l'étiquette d'un exemple. Nous montrons que la règle de mise à jour des poids conserve les propriétés théoriques du boosting. Quant à l'oracle, il doit être simulé en pratique et nous proposons une construction adaptée à l'inférence grammaticale. Nous décrivons enfin une étude expérimentale sur l'algorithme à base de fusions d'états RPNI⋆, dont les performances sont significativement améliorées.

1 Introduction

L'inférence grammaticale est un sous-domaine de l'apprentissage automatique dont le but est d'apprendre des modèles de langages (ensembles de mots). Parmi ces modèles, les automates finis déterministes (AFD) sont des machines à états qui prennent des mots en entrée et qui les acceptent ou les rejettent. Du point de vue de l'apprentissage automatique, les AFD sont des classifieurs qui séparent l'ensemble de tous les mots en deux classes, la classe dite positive des mots acceptés par l'AFD, et celle dite négative des mots rejetés par l'AFD. Une des raisons qui expliquent les efforts consentis pour apprendre de tels classifieurs tient au fait que de nombreux domaines d'application, comme la reconnaissance de la parole, la reconnaissance de formes ou le traitement des langues naturelles, tirent partie des propriétés structurelles et sémantiques des AFD

⋆ This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

(de la Higuera, 2004). Dans ce papier, nous nous concentrerons uniquement sur les algorithmes d'inférence grammaticale exacte qui utilisent un mécanisme de fusion d'états. Nous utiliserons en particulier RPNI (Oncina & García, 1992), même si nos résultats peuvent s'étendre facilement à d'autres algorithmes, variantes ou extensions de RPNI, comme EDSM (Lang *et al.*, 1998) ou ESMA (Coste, 1999).

RPNI fonctionne à partir de deux ensembles de données, E_+ et E_-, qui contiennent respectivement des mots acceptés et rejetés par l'AFD qu'on doit apprendre. Par exemple, l'AFD de la Figure 1 a été obtenu avec RPNI pour $E_+ = \{b, ab, abb, bab, \lambda\}$ et $E_- = \{aa, ba\}$, où λ désigne le mot vide.

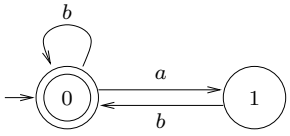

FIG. 1 – Un AFD appris avec RPNI, en l'absence de données bruitées.

RPNI est considéré comme un algorithme d'apprentissage exact au sens où il retourne un AFD qui "colle" aux données : un exemple positif (de E_+) sera nécessairement reconnu par cet AFD, et un exemple négatif (de E_-) sera nécessairement rejeté. D'autre part, si les données d'apprentissage contiennent certains mots, dits *caractéristiques* de l'AFD qui a produit les données (AFD *cible*), alors on peut montrer que RPNI trouve cette cible (Oncina & García, 1992). Ces propriétés ont un intérêt théorique indéniable. Mais dès qu'on tente d'utiliser RPNI sur un problème réel, où les données sont intrinsèquement bruitées, ces propriétés constituent, en fait, son principal handicap. RPNI n'étant pas immunisé contre le sur-apprentissage (au contraire), les données bruitées ont un effet désastreux ; cela se traduit par une augmentation du nombre d'états des AFD produits, et par une chute du taux de succès en généralisation de ces AFD. C'est pourquoi la tolérance au bruit est un problème crucial, aussi bien en inférence grammaticale, pour ce qui concerne les algorithmes à base de fusions d'états, que dans l'ensemble des domaines de l'apprentissage automatique, en général.

Une première approche pour limiter le risque de sur-apprentissage des algorithmes à base de fusions d'états a été présentée dans (Sebban & Janodet, 2003). Les auteurs y proposent une relaxation de la règle de fusion. Ces travaux ont abouti à un nouvel algorithme, RPNI*, dont la règle de fusion, reposant sur la statistique inférentielle, tolère la présence de données bruitées. D'un point de vue expérimental, les améliorations apportées par rapport à RPNI sont très significatives. Mais cette approche reste limitée pour deux raisons. D'abord, les auteurs fournissent des résultats théoriques montrant la difficulté d'inférer des AFD en présence de forts taux de bruit (au delà de 10%). Ensuite, de par sa nature statistique, l'utilisation de RPNI* peut aboutir en l'acceptation à tort d'une fusion. En effet, si RPNI n'accepte pas d'exemple négatif dans un état final, RPNI*, en relâchant la contrainte de fusion, permet à des exemples positifs et négatifs de cohabiter dans un même état. Si l'on espère que ces exemples négatifs sont des données bruitées (et donc faussement négatives), on peut craindre la présence d'exemples réellement

négatifs dans un état final, ce qui peut conduire de nouveau à de sérieux problèmes.

En fait, ces deux problèmes de RPNI* sont à rapprocher de deux questions récurrentes et plus générales en apprentissage automatique :

1. Est-il possible d'optimiser les performances d'un algorithme d'apprentissage donné (dans notre cas un algorithme à base de fusions d'états) ?
2. Est-il possible de contrôler cette optimisation en présence de données bruitées (dans notre cas, éviter les fusions d'états qui feraient cohabiter des exemples positifs et des exemples négatifs dans le même état) ?

Indiscutablement, nous pensons que ces deux questions peuvent être traitées à l'aide du boosting, même si cela n'a jamais été fait en inférence grammaticale, à notre connaissance. Le principe du boosting et de son algorithme ADABOOST (Freund & Schapire, 1996; Freund & Schapire, 1997) (voir Algorithme 1) consiste à apprendre T hypothèses (dites *faibles*) à l'aide d'un algorithme WL sur T distributions de probabilités $\mathbf{w}_t$ d'un échantillon d'apprentissage E, puis à combiner les hypothèses faibles h_t en une seule hypothèse H_t "forte". A chaque itération, la mise à jour favorise (exponentiellement) les exemples mal appris par l'algorithme à l'étape précédente.

for all $x \in E$ **do** $w_0(x) \longleftarrow 1/|E|$;
for $t = 0$ **to** T **do**
  $h_t \longleftarrow \text{WL}(E, \mathbf{w}_t)$;
  $\epsilon_t \longleftarrow \sum_{\{e \in E : y(x) \neq h_t(x)\}} w_t(x)$;
  $c_t \longleftarrow (1/2)\ln((1-\epsilon_t)/\epsilon_t)$;
  $Z_t \longleftarrow \sum_{e \in E} w_t(x)\exp(-c_t y(x) h_t(x))$;
  for all $x \in E$ **do**
    $w_{t+1}(x) \longleftarrow w_t(x)\exp(-c_t y(x) h_t(x))/Z_t$;
return H_T **such that** $H_T(x) = \sum_{t=0}^{T} c_t h_t(x)$;

Algorithm 1: Pseudo-code d'ADABOOST.

Récemment, des travaux ont tenté de limiter les risques de sur-apprentissage du boosting en évaluant l'intérêt d'augmenter le poids d'un exemple mal classé : ADABOOST ayant tendance à augmenter à tort les poids des exemples bruités, certaines approches tentent de contrôler la mise à jour en utilisant des fonctions de repondération plus douces que la fonction exponentielle originale (Friedman *et al.*, 1998; Domingo & Watanabe, 2000; Freund, 2001). Mais l'utilisation de ces nouvelles fonctions peut remettre en question les propriétés théoriques de convergence du boosting. On pourrait aussi penser que des exemples bruités, dont la classe est douteuse, devraient être supprimés de l'échantillon d'apprentissage. Mais une tendance actuelle, initiée par (Blum & Mitchell, 1998), montre que les exemples sans étiquette peuvent aussi apporter une aide significative, pourvu que l'on sache utiliser l'information qu'ils contiennent. En inférence grammaticale, si la classe des mots permet de donner une étiquette aux états, les mots eux-mêmes permettent de deviner la structure de l'AFD (au sens des états et des transitions). Il semble donc essentiel d'intégrer ces mots, à l'étiquette douteuse, dans les processus d'apprentissage et de boosting.

Dans cet article, nous avons choisi de conserver les exemples bruités et l'emploi de la fonction exponentielle du boosting, mais nous supposons avoir accès à un oracle de confiance. Cet oracle, que nous devons simuler en pratique, fournit une estimation sur la confiance dans la classe d'un exemple. Une valeur positive pour un exemple signifie que nous pouvons être certain de sa classe. En revanche, une valeur négative signifie que l'on doit émettre des doutes au sujet de sa classe. Il est crucial de remarquer qu'une confiance négative ne signifie pas que l'exemple appartient à la classe opposée, mais plutôt que l'on ne connaît pas sa classe réelle. De plus, il est nécessaire de supposer que l'oracle fournit des valeurs *réelles* dans $[-1, +1]$ plutôt que des valeurs *entières* dans $\{-1, +1\}$, car cet oracle doit être simulé en pratique et il est préférable de tenir compte des imperfections de cette simulation dès l'étude théorique.

Clairement, l'utilisation de l'oracle de confiance, pour traiter les données bruitées, nécessite une modification de la règle standard de mise à jour des poids. En effet, si ADABOOST est normalement utilisé pour augmenter le poids de tous les exemples mal appris, nous voulons ici augmenter les poids des exemples non bruités uniquement. Si cette stratégie semble utile pour améliorer n'importe quel algorithme d'apprentissage, nous pensons qu'elle est particulièrement adaptée pour repousser les limites de RPNI*.

Cet article est organisé comme suit. Après avoir détaillé le fonctionnement de RPNI* (Section 2), nous proposons un nouveau cadre théorique adapté à l'utilisation d'un oracle. Nous modifions la règle standard de mise à jour d'ADABOOST, et nous prouvons que cette nouvelle stratégie de repondération conserve les propriétés théoriques du boosting. Puis nous proposons dans la Section 4 une stratégie pour simuler de manière pertinente un oracle satisfaisant les propriétés voulues. Cette stratégie est basée sur l'utilisation d'un graphe des k plus proches voisins sur l'ensemble de mots. Dans la Section 5, nous montrons sur différentes bases de données, que notre algorithme de boosting améliore significativement les performances de RPNI*.

2 L'algorithme à base de fusions d'états RPNI*

Le but de cette section est de donner une brève description de RPNI* (voir l'Algorithme 2 et (Sebban & Janodet, 2003) pour plus de détails). On définit un *alphabet* comme un ensemble fini non vide Σ de *lettres*, et un *mot* comme une séquence $w = x_1x_2 \dots x_n$ de *lettres*. Un *automate fini déterministe* (AFD) est un n-uplet $\langle Q, \delta, s_0, F\rangle$ où Q désigne un ensemble d'états, $\delta : Q \times \Sigma \rightarrow Q$ est une fonction de transitions, $s_0 \in Q$ est un état initial et $F \subseteq Q$ est un ensemble d'états finaux. Un état s contient un mot w si et seulement si $\delta(s_0, w) = s$. Un mot est accepté par l'automate si $\delta(s_0, w) \in F$ et rejeté sinon. Par exemple, l'AFD de la Figure 2 est défini par $Q = \{0, 1\}$, $s_0 = 0$, $F = \{0\}$, $\delta(0, a) = 1$ et $\delta(0, b) = \delta(1, b) = 0$. Le mot abb est accepté puisque $\delta(0, abb) = \delta(1, bb) = \delta(0, b) = 0 \in F$. En appliquant le même raisonnement, le mot ba est rejeté car $\delta(0, ba) = 1 \notin F$. De même, le mot aa est rejeté car $\delta(0, aa)$ n'est pas défini.

RPNI* commence par construire le PTA (pour *prefix tree acceptor*) de E_+, défini comme le plus grand AFD reconnaissant l'ensemble des mots de E_+ (voir Figure 2). Ses états sont numérotés selon l'ordre hiérarchique sur les préfixes de E_+. RPNI* parcourt ensuite ces états dans l'ordre. Fusionner deux états consiste à les remplacer par un

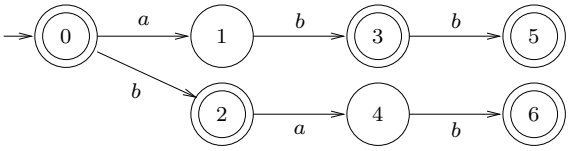

FIG. 2 – Un exemple de PTA.

$A \longleftarrow \text{PTA}(E_+)$;
for $i = 1$ **to** $N - 1$ **do**
  $j \longleftarrow 0$; `continue` $\longleftarrow$ `true`;
  while $(j < i)$ **and** `continue` **do**
    $B \longleftarrow$ `compute_merging`(A, i, j);
    $C \longleftarrow$ `compute_final_states`(B, E_+, E_-);
    if `statistically_acceptable`(C, E_+, E_-) **then**
      $A \longleftarrow C$; `continue` $\longleftarrow$ `false`;
    $j \longleftarrow j + 1$;
return(A);

Algorithm 2: Pseudo-code de RPNI*.

seul dont le numéro est le plus petit des numéros des deux états. Dans RPNI, deux états sont fusionnables si et seulement si, une fois la fusion effectuée, aucun exemple négatif n'est contenu dans un état final. Avec RPNI⋆, cette règle est relâchée pour permettre la présence de mots bruités. Un état est *positif* s'il contient plus d'exemples positifs que d'exemples négatifs, et *négatif* sinon. Un exemple négatif (resp. positif) est *mal classé* s'il est contenu par un état positif (resp. négatif). Enfin, une fusion est *statistiquement acceptable* si la proportion d'exemples mal classés sur l'ensemble de l'AFD après fusion n'est pas significativement plus grande que la proportion d'exemples mal classés sur l'ensemble de l'AFD avant fusion. Cette règle permet d'éviter les phénomènes de sur-apprentissage, dûs à la présence de bruit, et résultant en la construction d'AFD avec un grand nombre d'états et une faible capacité de généralisation.

Pour montrer la pertinence de RPNI⋆, supposons que l'on insère le mot bruité *bbbb* dans l'ensemble E_- de l'exemple décrit dans l'introduction. Si l'automate cible était celui de la Figure 1, ce mot devrait être positif. La partie gauche de la Figure 3 présente l'automate profondément modifié que RPNI a appris sur ce nouvel échantillon bruité. Cet exemple jouet montre bien le fait que RPNI n'est pas immunisé contre le sur-apprentissage. L'automate présenté en partie droite est le résultat obtenu avec RPNI⋆. Nous remarquons que le bruit a, dans ce cas, été géré de manière optimale puisque l'automate appris est l'automate cible lui-même.

Malgré une considérable amélioration des performances de RPNI, on doit espérer que les exemples mal classés sont réellement bruités. Certes, c 'est le cas du mot *bbbb* dans l'exemple précédent. Mais de par sa nature statistique, RPNI⋆ peut toujours faire une mauvaise fusion (cette erreur est mesurable en pratique par le risque de première

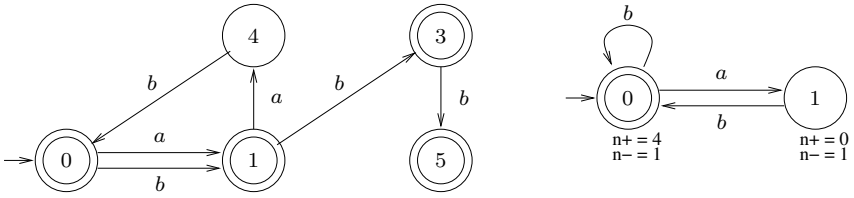

FIG. 3 – AFD inférés par RPNI* : l'AFD de gauche montre l'impact des données bruitées sur RPNI ; celui de droite est obtenu avec RPNI*. Les valeurs n_+ et n_- correspondent respectivement aux nombres d'exemples positifs et négatifs contenus par chacun des états.

espèce du test de comparaison de proportions utilisé dans RPNI*). Même si on peut théoriquement réduire ce risque, il ne peut pas être nul, ce qui peut avoir des effets désastreux sur les performances de l'AFD. Pour améliorer RPNI*, nous supposons ici qu'on peut obtenir une information sur la confiance en l'étiquette d'un exemple, information qui pourrait être utilisée dans une procédure de boosting. En supposant que l'on obtienne cette information par l'intermédiaire d'un oracle, nous serions alors capable de connaître les exemples parmi les mal classés qui méritent réellement d'être appris. En d'autres termes nous serions capable de séparer les exemples mal classés qui devraient bénéficier d'une augmentation de poids de ceux qui devraient être sujet à une baisse de poids. Ceci nécessite cependant la mise en place d'une nouvelle stratégie de repondération. C'est le but de la section suivante.

3 Un oracle de confiance dans le processus de boosting

Soit E un ensemble d'exemples de cardinal $|E| = m$. Chaque exemple x possède une étiquette $y(x)$ pouvant être soit positive ($y(x) = +1$), soit négative ($y(x) = -1$). Nous supposons avoir accès à un *oracle de confiance*, qui associe à tout exemple $x \in E$ une valeur réelle non nulle $q(x) \in [-1, 0[\cup]0, +1]$. La valeur $q(x)$ sera supposée constante au cours du processus d'apprentissage. Une confiance positive pour un exemple x signifie que x appartient avec certitude à la classe qui lui est assignée, tandis qu'une confiance négative signifie que l'on ne peut rien supposer sur l'étiquette de x. Rappelons que l'utilisation de valeurs réelles est justifiée par le fait que l'oracle sera simulé en pratique. Nous partitionnons E en deux sous-ensembles : $E_N = \{x \in E : q(x) < 0\}$ et $E_{\overline{N}} = E \backslash E_N$. Puisque l'étiquette des exemples de E_N n'est pas fiable, nous nous fixons, comme objectif du boosting, la minimisation de l'erreur empirique sur $E_{\overline{N}}$, et nous définissons dans ce sens $\forall T > 0$,

$$\varepsilon(E_{\overline{N}}, H_T) = \frac{1}{|E_{\overline{N}}|} \sum_{x \in E_{\overline{N}}} [\![y(x) \neq H_T(x)]\!], \tag{1}$$

où $[\![\pi]\!]$ désigne la valeur de vérité d'un prédicat π (dans $\{0,1\}$), et

$$H_T(x) = \sum_{t=0}^{T} c_t h_t(x),$$

la combinaison finale d'hypothèses, en reprenant les notations de l'Algorithme 1. Soit $\mathcal{P}_m$ l'espace des vecteurs de probabilité de dimension m et $\mathbf{u} \in \mathcal{P}_m$ le vecteur uniforme. Nous supposons l'existence d'un vecteur $\mathbf{w}_t \in \mathcal{P}_m$ qui contient le poids de chaque exemple $x \in E$ à l'étape t, avec $\mathbf{w_0} = \mathbf{u}$ (la distribution initiale étant donc uniforme, comme dans ADABOOST).

Notre objectif est ici de trouver une nouvelle règle optimale de mise à jour des poids. Nous savons, depuis (Kivinen & Warmuth, 1999), que le boosting standard tel que défini dans (Schapire & Singer, 1998) est un cas particulier de l'optimisation sous contrainte d'une divergence de Breigman. Replaçons ce problème dans notre cadre. La stratégie est de calculer successivement $\mathbf{w}_{t+1}$ à partir de $\mathbf{w}_t$ en minimisant la *divergence d'information*

$$\mathbf{1}.\mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t) = \sum_{x \in E} \mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t)(x),$$

où $\mathbf{i}(.,.)$ est le vecteur de composante

$$\mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t)(x) = \left(\mathbf{w}_{t+1} \ln \frac{\mathbf{w}_{t+1}}{\mathbf{w}_t} - \mathbf{w}_{t+1} + \mathbf{w}_t \right)(x),$$

pour tout $x \in E$. Remarquons que cette divergence d'information est une fonction convexe de $\mathbf{w}_{t+1}$. Nous souhaitons optimiser $\mathbf{1}.\mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t)$ sous deux contraintes. La première est immédiate ; elle exprime le fait que $\mathbf{w}_{t+1}$ est une distribution ($\mathbf{w}_{t+1} \in \mathcal{P}_m$) :

$$\mathbf{1}.\mathbf{w}_{t+1} - 1 \quad = \quad 0. \tag{2}$$

Cette contrainte ne change bien sûr pas dans notre cas. La seconde contrainte intègre une fonction de perte pour chaque exemple de E. Dans le boosting classique, elle exprime le fait que la dernière hypothèse apprise est, en quelque sorte, la plus mauvaise hypothèse sur la nouvelle distribution, puisque ses performances sont en moyenne équivalentes à celles de la *décision aléatoire* : $\sum_{x \in E} (w_{t+1} y h_t)(x) = 0$. Cette contrainte, combinée avec l'*hypothèse d'apprentissage faible* (Kearns & Vazirani, 1994), assure que la nouvelle hypothèse h_{t+1} construite sur la distribution $\mathbf{w}_{t+1}$ est significativement différente de h_t, assurant qu'à chaque étape, l'algorithme apprend quelque chose de nouveau sur E. Dans notre cas, cette contrainte est légèrement différente :

$$\begin{aligned} \mathbf{w}_{t+1}.\mathbf{a}_t &= 0, \\ \forall x \in E_{\overline{N}} : a_t(x) &= q(x)(y h_t)(x), \\ \forall x \in E_N : a_t(x) &= -q(x). \end{aligned} \tag{3}$$

Par comparaison avec la seconde contrainte du boosting, l'égalité (3) entraîne l'égalité suivante : $\sum_{x \in E_{\overline{N}}} (w_{t+1} q y h_t)(x) = \sum_{x \in E_N} (w_{t+1} q)(x)$. Comme $\forall x \in E_N$, $q(x) <$

0, le membre de droite est strictement négatif, ce qui rend la dernière hypothèse construite h_t particulièrement mauvaise sur $E_{\overline{N}}$ avec une distribution dépendant à la fois de $\mathbf{w}_{t+1}$ et de $\mathbf{q}$. Il y a donc clairement des différences avec le boosting classique qui assure une performance moyenne (comparable au choix aléatoire), uniquement avec la distribution $\mathbf{w}_{t+1}$ sur E. La minimisation de la divergence d'information sous les contraintes (2) et (3) est obtenue comme solution ($\forall x \in E$) de :

$$\begin{aligned} & \partial_{\mathbf{w}_{t+1}} \mathbf{i}(\mathbf{w}_{t+1}, \mathbf{w}_t)(x) \\ + \left[b_t \partial_{\mathbf{w}_{t+1}} (\mathbf{1}.\mathbf{w}_{t+1} - 1) + c_t \partial_{\mathbf{w}_{t+1}} (\mathbf{w}_{t+1}.\mathbf{a}_t) \right](x) \quad &= \quad 0 \end{aligned} \tag{4}$$

avec b_t et c_t les multiplicateurs de Lagrange. En résolvant (4) pour $\mathbf{w}_{t+1}$ nous obtenons $\forall x \in E$:

$$\left(\ln \frac{\mathbf{w}_{t+1}}{\mathbf{w}_t} + b_t \mathbf{1} + c_t \mathbf{a}_t \right)(x) = 0$$

$$\Leftrightarrow w_{t+1}(x) = \frac{w_t(x) \exp(-c_t a_t(x))}{\exp(b_t)}.$$

La contrainte (2) permet de déduire :

$$b_t = \ln \sum_{x \in E} w_t(x) \exp(-c_t a_t(x)), \tag{5}$$

Le terme dans le $\ln(.)$ correspond au coefficient de normalisation pour $\mathbf{w}_{t+1}$:

$$Z_t = \sum_{x \in E} w_t(x) \exp(-c_t a_t(x)). \tag{6}$$

Nous verrons par la suite que c_t est en fait positif (Lemme 2). Comme l'oracle donne une confiance positive à tout $x \in E_{\overline{N}}$ ($q(x) > 0$), et comme pour tout exemple,

$$w_{t+1}(x) = \frac{w_t(x) \exp(-c_t q(x)(y h_t)(x))}{Z_t},$$

il apparaît que le poids de tout $x \in E_{\overline{N}}$ augmente entre les distributions $\mathbf{w}_t$ et $\mathbf{w}_{t+1}$ si et seulement si x est mal classé par l'hypothèse construite sur $\mathbf{w}_t$, et diminue sinon. Par contre, les poids des exemples suspects $x \in E_N$ ($q(x) < 0$) seront tous baissés :

$$w_{t+1}(x) = \frac{w_t(x) \exp(c_t q(x))}{Z_t}.$$

Nous verrons dans la section suivante qu'une telle mise à jour améliore les performances de RPNI*. La dernière inconnue c_t est obtenue par la contrainte (3) comme solution de :

$$\sum_{x \in E} (w_t a_t)(x) \exp(-c_t a_t(x)) \quad = \quad 0. \tag{7}$$

Notons que cette équation signifie $(\partial Z_t / \partial c_t) = 0$. Comme Z_t est une fonction convexe de c_t, résoudre (4) revient finalement au problème de la minimisation de Z_t. Pour comprendre pourquoi il est important de résoudre (4), intéressons nous temporairement à l'erreur commise par H_t.

Lemme 1

$$\varepsilon(E_{\overline{N}}, H_T) \quad \leq \quad \frac{m}{|E_{\overline{N}}|}\left(\prod_{t=0}^{T} Z_t\right).$$

Preuve : Comme $\forall x \in E_{\overline{N}}$, $q(x) > 0$, nous avons pour tous ces exemples :

$$[\![H_T(x) \neq y(x)]\!] \leq \exp(-q(x)y(x)\sum_{t=0}^{T} c_t h_t(x)).$$

D'autre part, en déroulant la règle de mise à jour, on obtient :

$$\forall x \in E_{\overline{N}} : w_{T+1}(x) = \frac{w_0(x)\exp(-q(x)y(x)\sum_{t=0}^{T} c_t h_t(x))}{\prod_{t=0}^{T} Z_t}.$$

Par conséquent $[\![H_T(x) \neq y(x)]\!] \leq m w_{T+1}(x)(\prod_{0<t\leq T} Z_t)$. En sommant pour tout $x \in E_{\overline{N}}$, nous obtenons $\varepsilon(E_{\overline{N}}, H_T) \leq (m/|E_{\overline{N}}|)(\prod_{0\leq t\leq T} Z_t)(\sum_{x\in E_{\overline{N}}} w_{T+1}(x))$. Enfin, en utilisant le fait que $\sum_{x\in E_{\overline{N}}} w_{T+1}(x) \leq 1$, nous obtenons le Lemme. □

Comme le terme $(m/|E_{\overline{N}}|)$ est constant pour E et $E_{\overline{N}}$, le Lemme 1 signifie que pour minimiser l'erreur sur $E_{\overline{N}}$, il faut se concentrer sur la minimisation de chaque Z_t. Ceci nous permet de faire d'une pierre deux coups : résoudre (7) nous apporte en même temps la solution de (4) et la minimisation de l'erreur sur $E_{\overline{N}}$ au regard du Lemme 1.

Intéressons-nous maintenant à la solution de (7). Nous commençons par formuler l'hypothèse suivante :

- $\forall t \geq 0$, il existe une constante $\gamma_t > 0$ telle que :

$$\frac{\sum_{x\in E_{\overline{N}}}(w_t q y h_t)(x)}{\sum_{x\in E_{\overline{N}}}(w_t q)(x)} = \gamma_t. \tag{8}$$

En suivant la terminologie classique du boosting, (8) doit être vue comme une hypothèse d'apprentissage faible (*Weak Learning assumption*, WLA). Notons que si h_t correspond au choix aléatoire, alors $\gamma_t = 0$. En d'autres termes, (8) traduit le fait que h_t doit être légèrement plus performante que le choix aléatoire pour pouvoir être considérée comme une hypothèse faible. Notre WLA est différente de celle du boosting classique (Kearns & Vazirani, 1994) pour deux raisons. Premièrement, elle est locale : elle se concentre sur un sous-ensemble de E. Deuxièmement, elle s'appuie sur une distribution qui n'est pas exactement $\mathbf{w}_t$ mais une distribution $\mathbf{w}_{q,t} \in \mathcal{P}_m$ paramétrée par $\mathbf{q}$ comme suit : $\forall x \in E, w_{q,t}(x) = |q(x)|w_t(x)/Q_t$, où $Q_t = \sum_{x\in E} |q(x)|w_t(x)$ est un coefficient de normalisation. Notons que $\mathbf{w}_{q,t} \in \mathcal{P}_m$, car tous les exemples ont une confiance $q(.)$ différente de 0. Nous étendons cette définition à celle plus générale de $W_{q,t}(E') = \sum_{x\in E'} \mathbf{w}_{q,t}(x)$ pour tout $E' \subseteq E$.

Afin de compléter la discussion autour de la contrainte (3), nous définissons $E^{+}_{\overline{N},t} = \{x \in E_{\overline{N}} : (yh_t)(x) = +1\}$ et $E^{-}_{\overline{N},t} = \{x \in E_{\overline{N}} : (yh_t)(x) = -1\}$. La WLA est

équivalente à :

$$W_{q,t}(E^+_{\overline{N},t}) - W_{q,t}(E^-_{\overline{N},t}) = \gamma_t W_{q,t}(E_{\overline{N}}). \quad (9)$$

Nous ne pouvons donc pas choisir $h_{t+1} = h_t$ sous la WLA, car sinon (3) impliquerait que $W_{q,t+1}(E^+_{\overline{N},t}) - W_{q,t+1}(E^-_{\overline{N},t}) < 0$, tandis que (9) impliquerait $W_{q,t+1}(E^+_{\overline{N},t}) - W_{q,t+1}(E^-_{\overline{N},t}) > 0$. La WLA force donc l'algorithme d'apprentissage à retourner une hypothèse h_{t+1} différente de h_t, donc à apprendre quelque chose de nouveau.

Lemme 2
La solution de (7) est unique, strictement positive, et elle vérifie :

$$\frac{1}{2\overline{q}} \ln \frac{1 - W_{q,t}(E^-_{\overline{N},t})}{W_{q,t}(E^-_{\overline{N},t})} \leq c_t \leq \frac{1}{2\underline{q}} \ln \frac{1 - W_{q,t}(E^-_{\overline{N},t})}{W_{q,t}(E^-_{\overline{N},t})},$$

où $\underline{q} = \min_{x \in E} |q(x)|$ *et* $\overline{q} = \max_{x \in E} |q(x)|$.

Preuve : Posons

$$g(c) \quad = \quad \sum_{x \in E} (w_t a_t)(x) \exp(-c a_t(x)). \quad (10)$$

$g'(c) = -\sum_{x \in E} (w_t a_t^2)(x) \exp(-c a_t(x)) < 0$. Donc $g(c)$ est strictement décroissante sur $\mathbb{R}$. Comme $\lim_{c \to +\infty} g(c) = -\infty$ (à la condition qu'il existe $x \in E$ tel que $a_t(x) < 0$) et $\lim_{c \to -\infty} g(c) = +\infty$ (à la condition qu'il existe $x \in E$ tel que $a_t(x) > 0$), l'équation (7) a une solution unique. De plus,

$$\begin{aligned} \frac{g(0)}{Q_t} &= W_{q,t}(E^+_{\overline{N},t}) - W_{q,t}(E^-_{\overline{N},t}) + W_{q,t}(E_N) \\ &= \gamma_t W_{q,t}(E_{\overline{N}}) + W_{q,t}(E_N) \\ &= \gamma_t + (1 - \gamma_t) W_{q,t}(E_N). \end{aligned} \quad (11)$$

Donc $g(0) > 0$, ce qui implique que $c_t > 0$ dans l'équation (7). Enfin,

$$\begin{aligned} \frac{g(c_t)}{Q_t} = 0 = \quad & \textstyle\sum_{x \in E^+_{\overline{N},t} \cup E_N} w_{q,t}(x) \exp\left(-c_t |q(x)|\right) \\ & - \textstyle\sum_{x \in E^-_{\overline{N},t}} w_{q,t}(x) \exp\left(c_t |q(x)|\right). \end{aligned}$$

Comme $\underline{q} \leq |q(x)| \leq \overline{q}$ pour tout $x \in E$, nous obtenons :

$$\begin{aligned} & W_{q,t}(E^+_{\overline{N},t} \cup E_N) e^{-c_t \overline{q}} - W_{q,t}(E^-_{\overline{N},t}) e^{c_t \overline{q}} \leq 0 \quad \text{et} \\ & W_{q,t}(E^+_{\overline{N},t} \cup E_N) e^{-c_t \underline{q}} - W_{q,t}(E^-_{\overline{N},t}) e^{c_t \underline{q}} \geq 0. \end{aligned}$$

En résolvant ces deux inéquations par rapport à c_t, nous obtenons l'inéquation du Lemme. □

Donc $c_t = r \ln((1 - W_{q,t}(E^-_{\overline{N},t}))/W_{q,t}(E^-_{\overline{N},t}))$ est solution de l'équation (7) pour un certain $r \in [1/2\overline{q}, 1/2\underline{q}]$. Cette borne est plus précise que celle donnée dans (Schapire & Singer, 1999) (où $c_t \in \mathbb{R}$). De plus, elle permet une approximation très rapide de c_t par dichotomie. Supposons qu'on veuille approximer c_t par un $\hat{c}_t$ tel que $|\hat{c}_t - c_t|/|c_t| < \epsilon$ pour un $0 < \epsilon \leq 1$. L'approximation dichotomique se fera alors en au plus $\mathcal{O}(\ln(\overline{q}/\underline{q}) + \ln(1/\epsilon))$ étapes. Cette efficacité est d'autant plus appréciable que ce calcul est répété T fois.

Lemme 3
$\forall t \geq 0$,

$$Z_t \leq \sqrt{1 - \left(\frac{\underline{q}\gamma_t}{\overline{q}}\right)^2} < \exp\left(-\frac{1}{2}\left(\frac{\underline{q}\gamma_t}{\overline{q}}\right)^2\right).$$

Preuve : On a l'inégalité suivante, $\forall x \in [-1, 1], \forall \eta \in \mathbb{R}$:

$$\exp(-\eta x) \leq \frac{1+x}{2}\exp(-\eta) + \frac{1-x}{2}\exp(\eta), \tag{12}$$

En effet, la fonction $x \mapsto \exp(-\eta x)$ est convexe et le membre droit de l'inégalité est l'équation de la droite passant par les points de coordonnées $(-1, \exp(\eta))$ et $(1, \exp(-\eta))$. En posant $\eta = c_t\overline{q}$ et $x = a_t(x)/\overline{q}$, nous obtenons $\forall x \in E$:

$$\exp(-c_t a_t(x)) \leq \frac{\overline{q} + a_t(x)}{2\overline{q}}\exp(-c_t\overline{q}) + \frac{\overline{q} - a_t(x)}{2\overline{q}}\exp(c_t\overline{q}). \tag{13}$$

En nommant $\ell(x, c_t)$ le membre droit de l'inéquation (13), nous obtenons

$$Z_t \quad \leq \quad \inf_{c\in\mathbb{R}} \sum_{x\in E} w_t(x)\ell(x, c). \tag{14}$$

Le c minimisant le membre droit de l'inéquation (14) est

$$c \quad = \quad \frac{1}{2\overline{q}} \ln \frac{\overline{q} + \sum_{x\in E}(w_t a_t)(x)}{\overline{q} - \sum_{x\in E}(w_t a_t)(x)},$$

d'où nous déduisons

$$Z_t \quad \leq \quad \sqrt{1 - \left(\frac{g(0)}{\overline{q}}\right)^2}, \tag{15}$$

où $g(.)$ est la fonction définie dans l'équation (10). En utilisant le fait que $g(0) \geq \gamma_t Q_t$ (d'après l'équation (11)) et que $Q_t \geq \underline{q}$, nous obtenons le résultat du Lemme. □

Comme nous l'avons déjà dit, nous supposons que la fonction $q(.)$, donc $\underline{q}$ et $\overline{q}$, sont constants au cours des itérations du boosting. Aussi, sous la WLA, la valeur maximale de Z_t est toujours < 1, ce qui garantit une convergence exponentielle de $\varepsilon(E_{\overline{N}}, H_T)$ vers 0.

4 D'un graphe de voisinage vers un oracle

Nous décrivons dans cette section une approche empirique permettant de simuler un oracle de confiance, qui doit satisfaire au mieux les comportements théoriques décrits précédemment. L'objectif principal est de s'assurer qu'une confiance positive donnée par l'oracle correspond à un exemple dont l'étiquette est correcte. Nous proposons ici une stratégie basée sur un graphe de voisinage, qui permet de distinguer géométriquement les exemples douteux des exemples sûrs.

4.1 Une approche géométrique de la confiance

Nous pensons que les graphes des k plus proches voisins (*k*PPV) (Cover & Hart, 1967) sont particulièrement adaptés pour traiter un tel problème. Rappelons que la règle de classification des *k*PPV attribue la classe d'un exemple x en calculant la classe majoritaire parmi les k exemples de E qui sont les plus proches de x. Cette méthode, au delà du fait que son erreur limite est théoriquement bornée par deux fois l'erreur bayésienne, est très résistante au bruit. En effet, un exemple bruité isolé au milieu d'exemples de la classe opposée a un impact très limité sur la règle de classification. La propriété duale que l'on peut déduire de la remarque précédente, est qu'un exemple bruité peut être plus facilement détecté en analysant son voisinage. Définissons la fonction de marge $m(x)$ d'un exemple x de classe $y(x)$ calculée à l'aide d'un graphe des *k*PPV :

$$m(x) = \frac{N_{y(x)} - N_{\overline{y}(x)}}{N_{y(x)} + N_{\overline{y}(x)}},$$

où $N_{y(x)}$ (resp. $N_{\overline{y}(x)}$) désigne le nombre d'exemples parmi les k plus proches voisins de x étant de la même classe $y(x)$ (resp. étant de la classe opposée $\overline{y}(x)$). Un exemple n'ayant que des voisins de la classe opposée à la sienne (resp. de la même classe) recevra une marge de -1 (resp. 1). Même si $m(x)$ semble représenter d'une certaine manière la valeur de confiance $q(x)$ utilisée dans la section précédente, poser $q(x) = m(x)$ ne serait pas judicieux. En effet, la règle de classification d'un *k*PPV permet "seulement" d'assurer que l'erreur limite sera bornée par deux fois l'erreur bayésienne, et ce n'est pas ce qui nous intéresse ici. Nous cherchons à fournir une mesure pertinente de la confiance des exemples, en nous assurant que les valeurs positives de $q(x)$ correspondent aux x ayant une étiquette correcte, dans $(100 - \epsilon)$% des cas (avec ϵ très petit). Si $m(x)$ est légèrement plus grande que 0, cette condition est loin d'être respectée (bien que la marge soit positive). Pour contourner cet obstacle, nous fixons un paramètre $\beta \in [0, 1[$, et nous calculons la confiance comme suit :

- $q(x) = (m(x) - \beta)/(1 - \beta)$ $\forall x$ satisfaisant $\beta \leq m(x) \leq 1$.
  x aura donc une confiance positive si et seulement si $m(x) > \beta$.
- $q(x) = (m(x) - \beta)/(1 + \beta)$ $\forall x$ satisfaisant $-1 \leq m(x) \leq \beta$,
  et donc x recevra une marge négative si $m(x) < \beta$.

En d'autres termes, les confiances élevées seront assignées uniquement aux exemples ayant une large majorité d'exemples de leur classe dans leur voisinage. En revanche, les exemples peu pertinents, ayant une marge faiblement positive, se verront attribuer une confiance négative. Ce choix est motivé, dans le cadre de l'inférence grammaticale,

par le fait qu'il est préférable de réduire l'impact des données pertinentes, plutôt que d'inférer un AFD à partir de données bruitées.

4.2 Une fonction de distance probabiliste

Avant de calculer les valeurs de confiance, on doit utiliser une distance entre mots pour construire le graphe de voisinage. La distance la plus utilisée pour comparer des mots est probablement la distance d'édition. Cependant nous avons observé dans des expérimentations préliminaires que nous pouvions obtenir de meilleurs résultats en utilisant des distances "orientées données" construites après une transformation de l'espace de représentation. Nous avons décidé ici d'utiliser des bi-grammes comme modèle de langage, afin de décrire les mots sous forme de vecteurs. Les bi-grammes sont un modèle de représentation de connaissances qui ont prouvé leur utilité dans le domaine de la reconnaissance de données textuelles. Ils permettent d'appliquer des méthodes géométriques, où d'autres méthodes mathématiques, qui sont plus appropriées pour des vecteurs que pour des mots. La probabilité d'une lettre dans un mot dépend seulement de la lettre précédente dans ce mot. Etant donné un mot de n lettres, $w = x_1x_2x_3 \dots x_n$, sa probabilité est :

$$p(w) = \prod_{i=1}^{n} p(x_i/x_{i-1}).$$

Les probabilités élémentaires $p(x_i/x_{i-1})$ sont estimées à partir des mots de l'échantillon d'apprentissage. Enfin, pour donner un sens à la probabilité $p(x_1/x_0)$, un caractère particulier <BOW> (pour Beginning Of the Word) est ajouté à l'alphabet, et l'on pose $x_0 =$<BOW>.

Nous avons ici pour but de construire une représentation géométrique en deux dimension des mots. Nous désignons par $P_+(w)$ et $P_-(w)$ les probabilités de w relativement aux sous-classes positives et négatives de E. Dans le but de permettre la comparaison entre deux mots de longueurs différentes, nous normalisons chacune de ces probabilités en utilisant la moyenne géométrique : $P'_+(w) = \sqrt[n+1]{P_+(w)}$ et $P'_-(w) = \sqrt[n+1]{P_-(w)}$. Etant données les valeurs $P'_+(w)$ et $P'_-(w)$, nous pouvons maintenant

- voir w comme un point de coordonnées $P'_+(w)$ et $P'_-(w)$ dans un espace euclidien,
- calculer la distance euclidienne entre deux de ces points,
- construire à partir de cette distance un graphe de voisinage.

5 Expérimentations et résultats

Nos expérimentations ont été guidées par la nécessité de valider nos résultats théoriques, ainsi que d'estimer les performances d'un graphe de voisinage en tant qu'oracle. Pour effectuer cette tâche, nous avons utilisé trois sortes de bases de données. Les premières sont des benchmarks connus du répertoire UCI[1] : AGARICUS et TICTACTOE. Les secondes sont des bases de données réelles. La première, appelée USF, provient des Etats-Unis, et recense les mille prénoms les plus fréquemment donnés aux USA en

[1] http ://www.ics.uci.edu/~mlearn/MLRepository.html

2002. La deuxième, appelée WF, contient tous les prénoms mondiaux commençant par la lettre “a”. Pour ces deux bases de données, le concept à apprendre est le sexe correspondant au prénom. La dernière sorte de bases est constituée de bases artificielles. Nous avons généré sept bases de tailles variables ; les six premières contenant 3000 mots environ, tandis que la dernière, destinée à évaluer l'intérêt de notre méthode sur de grands volumes de données, contient 10000 exemples. Pour chacune de ces sept bases, nous avons étiqueté les mots à l'aide d'expressions régulières représentant des motifs (pattern). Finalement, nous avons artificiellement introduit du bruit dans l'ensemble des onze bases de données, en retournant l'étiquette d'un certain pourcentage de mots, l'estimation des performances en généralisation se faisant, elle, sur un échantillon non bruité.

		5%					10%				
BASE	TAILLE	kNN	RPNI	RPNI*	BOOST	PERF	kNN	RPNI	RPNI*	BOOST	PERF
AGARICUS	5644	0.0%	7.2%	7.2%	4.3%	0.0%	0.0%	14.0%	10.0%	6.9%	0.0%
TICTACTOE	809	9.2%	39.5%	34.5%	28.3%	4.9%	16.0%	41.9%	40.7%	34.5%	10.5%
USF	1871	16.5%	33.6%	31.2%	26.1%	26.4%	16.8%	35.7%	33.0%	31.2%	31.7%
WF	1887	16.4%	29.3%	25.3%	21.4%	20.6%	19.3%	29.8%	31.2%	22.7%	26.1%
BASE1	2540	24.8%	46.4%	41.5%	41.1%	35.6%	27.8%	52.1%	44.8%	43.7%	36.2%
BASE2	1505	9.6%	48.9%	21.9%	19.9%	14.3%	9.6%	39.2%	13.6%	30.5%	12.3%
BASE3	1532	10.4%	43.9%	27.0%	26.7%	13.0%	11.0%	39.0%	39.0%	12.7%	12.7%
BASE4	2969	0.5%	39.7%	2.0%	10.0%	0.0%	1.0%	45.7%	29.2%	23.4%	0.0%
BASE5	2179	0.9%	36.7%	36.7%	22.2%	19.2%	1.3%	45.9%	18.6%	25.0%	16.2%
BASE6	2004	17.7%	42.9%	7.0%	13.0%	2.0%	21.6%	45.9%	42.9%	25.6%	1.2%
BASE7	10000	23.0%	46.1%	39.7%	37.2%	39.7%	25.1%	49.4%	42.7%	43.7%	39.4%
MOYENNE	2994.5	11.7%	37.7 %	24.9%	22.7%	16.0%	13.6%	39.9%	31.4 %	27.3%	16.9%
		15%					20%				
AGARICUS	5644	1.0%	14.0%	10.0%	6.1%	0.0%	2.0%	23.2%	18.2%	11.9%	0.0%
TICTACTOE	809	15.4%	44.0%	40.7%	40.7%	9.8%	17.9%	40.1%	40.1%	38.8%	11.7%
USF	1871	20.5%	34.9%	32.0%	27.7%	27.7%	23.7%	48.8%	32.2%	30.6%	28.8%
WF	1887	20.4%	32.5%	36.2%	24.6%	24.6%	22.2%	35.4%	32.0%	31.7%	22.7%
BASE1	2540	28.2%	47.0%	42.9%	42.3%	38.5%	31.1%	51.2%	45.7%	45.2%	35.2%
BASE2	1505	12.9%	39.8%	39.8%	37.5%	10.6%	16.9%	46.8%	46.8%	45.1%	12.6%
BASE3	1532	12.7%	47.0%	39.4%	35.5%	14.9%	18.5%	49.5%	49.5%	41.6%	20.5%
BASE4	2969	1.5%	44.2%	16.0%	26.9%	0.0%	3.3%	43.4%	35.0%	30.4%	0.0%
BASE5	2179	2.0%	44.5%	43.8%	31.4%	15.1%	2.2%	41.0%	39.4%	34.6%	9.4%
BASE6	2004	24.9%	50.6%	42.9%	33.1%	1.7%	28.9%	46.6%	41.9%	39.2%	1.0%
BASE7	10000	27.1%	48.0%	40.7%	43.2%	40.7%	29.2%	48.0%	44.3%	41.8%	40.5%
MOYENNE	2994.5	15.1%	40.6%	34.9%	31.7%	16.7%	17.8%	43.1%	38.6%	35.5%	16.6%

TAB. 1 – Taux d'erreurs obtenus sur 11 bases avec 5, 10, 15 et 20% de bruit

Dans le Tableau 5, nous présentons les résultats que nous obtenons sur ces bases entre RPNI* et notre approche boostée (après 200 itérations). Cette étude expérimentale a été effectuée pour 5%, 10%, 15% et 20% de bruit, et en utilisant un graphe des kPPV, avec un k déterminé en testant plusieurs valeurs. Deux colonnes supplémentaires ont été ajoutées au tableau. Dans la première (kNN), nous présentons les performances du graphe des kPPV seul, non seulement pour montrer ses performances en tant qu'oracle, mais aussi pour estimer le gain apporté par celui-ci au processus du boosting. Dans la deuxième (PERF), nous présentons les résultats obtenus avec un oracle parfait, *i.e.* un oracle connaissant les exemples non bruités auxquels il attribue une confiance de -1, et ceux qui sont bruités auxquels il attribue une confiance de $+1$. L'objectif de cette colonne est d'établir le taux d'erreur optimal (mais inaccessible en pratique) pour chacune des bases.

Les remarques suivantes peuvent être faites. Tout d'abord, l'utilisation d'un graphe de voisinage pour simuler un oracle est tout à fait justifiée aux vues des résultats obtenus dans la colonne kPPV. Alors que RPNI* améliore déjà significativement RPNI (32.5%

contre 40.3% en moyenne sur tout les niveaux de bruit), notre nouvelle approche est encore plus performante : l'utilisation d'un oracle permet de réduire significativement le taux d'erreur de RPNI* (29.3% contre 32.5% en moyenne), comme un test de Student apparié permet de le vérifier. On peut aussi remarquer que l'oracle simulé permet d'atteindre, au bout de 200 itérations, des performances moins bonnes que celles obtenues avec un oracle parfait. Néanmoins cette dégradation des résultats reste raisonnable (29.3% contre 16.5% en moyenne). Enfin, il est à noter que le comportement observé précédemment reste relativement constant lorsque le niveau de bruit augmente. Pour montrer la convergence exponentielle du taux d'erreur avec les itérations, la Figure 4 présente le comportement de l'algorithme sur quatre bases caractéristiques (pour 5% de bruit). Elle met en évidence un effet rapide du boosting sur les performances (souvent avant la vingtième itération).

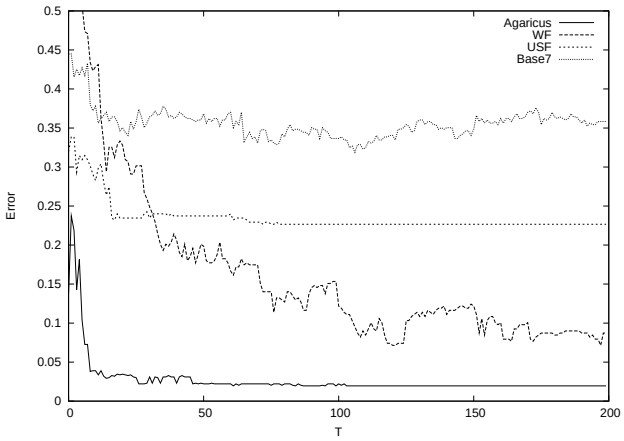

FIG. 4 – L'erreur en généralisation en fonction des itérations.

6 Conclusion

Nous avons introduit dans cet article une nouvelle stratégie de boosting permettant d'améliorer significativement les performances d'un algorithme d'inférence grammaticale en présence de données bruitées. Notre approche est originale, dans la mesure où elle repose sur l'utilisation d'un oracle capable d'estimer une valeur de confiance sur l'étiquette des exemples d'apprentissage. Clairement, nous pensons qu'elle est intéressante pour optimiser *tout* algorithme d'apprentissage, bien que ceci nécessite une étude expérimentale plus poussée. Dans le cadre de l'inférence grammaticale, nous avons proposé une heuristique, basée sur un graphe des *k*PPV, pour simuler efficacement cet oracle. Mais de même, l'utilisation d'autres oracles, basés sur des modèles de Markov cachés par exemple, mériterait de plus amples investigations.

Références

BLUM A. & MITCHELL T. (1998). Combining labeled and unlabeled data with co-training. In *Proc. of the 11 th International Conference on Computational Learning Theory*, p. 92–100.

COSTE F. (1999). *State Merging Inference of Finite State Classifiers*. Rapport interne, Publication interne n°1250.

COVER T. & HART P. (1967). Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory*, **IT13**, 21–27.

DE LA HIGUERA C. (2004). A bibliographic survey on grammatical inference. *Pattern Recognition*. Accepted.

DOMINGO C. & WATANABE O. (2000). Madaboost : a modification of adaboost. In A. PRESS, Ed., *Third Annual Conference on Computational Learning Theory*, p. 180–189.

FREUND Y. (2001). An adaptive version of the boost by majority algorithm. *Machine Learning*, **43**(3), 293–318.

FREUND Y. & SCHAPIRE R. E. (1996). Experiments with a new boosting algorithms. In *Proc. of the 13 th International Conference on Machine Learning*, p. 148–156.

FREUND Y. & SCHAPIRE R. E. (1997). A Decision-Theoretic generalization of on-line learning and an application to Boosting. *Journal of Computer and System Sciences*, **55**, 119–139.

FRIEDMAN J., HASTIE T. & TIBSHIRANI R. (1998). *Additive logistic regression : a statistical view of boosting*. Rapport interne.

KEARNS M. J. & VAZIRANI U. V. (1994). *An Introduction to Computational Learning Theory*. M.I.T. Press.

KIVINEN J. & WARMUTH M. (1999). Boosting as entropy projection. In *Proc. of the 12 th International Conference on Computational Learning Theory*, p. 134–144.

LANG K., PEARLMUTTER B. & PRICE R. (1998). Results of the abbadingo one DFA learning competition. In *4th Int. Coll. on Grammatical Inference*, p. 1–12.

ONCINA J. & GARCÍA P. (1992). *Inferring Regular Languages in Polynomial Update Time*, In *Pattern Recognition and Image Analysis*, volume 1 of *Machine Perception and Artificial Intelligence*, p. 49–61. World Scientific.

SCHAPIRE R. E. & SINGER Y. (1998). Improved boosting algorithms using confidence-rated predictions. In *Proc. of the 11 th International Conference on Computational Learning Theory*, p. 80–91.

SCHAPIRE R. E. & SINGER Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, **37**, 297–336.

SEBBAN M. & JANODET J. (2003). On state merging in grammatical : a statistical approach for dealing with noisy data. In *Proc. of the 20 th International Conference on Machine Learning*, p. 688–695. See also CAp'03.