Apprentissage de langages à partir de ressources bornées : le cas des AFD et des boules de mots

Colin de la Higuera, Jean-Christophe Janodet et Frédéric Tantini

Laboratoire Hubert Curien, Université Jean Monnet
18 rue du Professeur Benoît Lauras, 42000 Saint-Étienne, France
{cdlh,janodet,frederic.tantini}@univ-st-etienne.fr

Résumé : Il existe un grand nombre de paradigmes permettant d'étudier l'apprenabilité de classes de langages : identification à la limite, apprentissage à partir de requêtes, apprentissage probablement approximativement correct. La comparaison entre ces cadres est difficile. Pour en montrer toute la richesse, nous nous concentrons sur deux classes : les AFD et les boules de mots (au sens de la distance d'édition).
Mots-clés : Identification à la limite, apprentissage actif, apprentissage PAC, boules de mots, AFD, inférence grammaticale.

1 Introduction

L'étude des propriétés des algorithmes d'apprentissage, et particulièrement de ceux d'apprentissage de langages, peut être ou bien empirique (basée sur des expérimentations avec des ensembles de données), ou bien théoriques. Dans ce second cas, il s'agit d'étudier la capacité de l'algorithme à retrouver (exactement ou approximativement) un langage cible. Il s'agit également de mesurer les ressources (temps, données) nécessaires à cette tâche. Différents paradigmes ont été proposés pour rendre compte de cette notion de convergence avec ressources bornées, mais aucun ne s'est réellement imposé, et il existe bien peu de comparaisons entre ces définitions.

Dans cet article nous visitons une grande partie des critères d'*apprentissage polynomial* à travers deux classes de langages très différentes : les langages réguliers (représentés par les automates finis déterministes) et les boules de mots pour la distance d'édition (Levenshtein, 1965). Ces dernières sont apprenables à partir de données bruitées (Tantini *et al.*, 2006), et de requêtes de correction (Becerra-Bonache *et al.*, 2007).

Lorsque l'on souhaite prouver qu'une classe de langages est apprenable, il existe trois paradigmes standards. Le premier, l'identification à la limite (Gold, 1967), mime le processus cognitif d'un enfant qui apprendrait sa langue maternelle en écoutant les phrases qui circulent dans son environnement. Plus formellement, l'apprenant reçoit des informations à propos du langage cible qui arrivent sans discontinuer, et il émet

sans cesse des hypothèses. On dit qu'il y a convergence s'il arrive un moment où le processus est stationnaire et que l'hypothèse est correcte.

Le second, l'apprentissage à partir de requêtes (Angluin, 1987b), aussi appelé apprentissage actif en inférence grammaticale, s'inspire plutôt d'un jeu de devinettes où l'apprenant (élève) pose des questions à un oracle (professeur) à propos du langage cible. Le jeu s'arrête lorsque l'apprenant trouve la cible. Bien entendu, les résultats d'apprentissage dépendent fortement du type de questions que peut poser l'apprenant.

Les deux paradigmes précédents sont au moins aussi intéressants pour les résultats positifs que pour les résultats négatifs qu'ils permettent d'établir. Dans le cadre de l'apprentissage actif par exemple, si un apprenant ne peut pas deviner un langage cible en choisissant ses exemples, comment pourrait-il réussir à apprendre à partir d'exemples qui lui sont imposés par le contexte ?

Le troisième paradigme, l'apprentissage PAC (Valiant, 1984) (Probablement Approximativement Correct), est beaucoup plus pragmatique. Il formalise une situation où on cherche à construire automatiquement un modèle prédictif à partir de données. Dans ce cadre, on suppose qu'il existe une distribution $\mathcal{D}$ sur les mots du langage cible, qui est utilisée pour obtenir des exemples d'apprentissage et de test. Deux paramètres sont fixés : le premier, ϵ, est lié à l'erreur du modèle, c'est-à-dire la probabilité qu'un mot tiré selon $\mathcal{D}$ soit mal classé : le second, δ, est lié à la probabilité que les exemples tirés selon $\mathcal{D}$ ne soient pas représentatifs du langage cible.

Ces trois cadres sont habituellement difficiles à comparer, en particulier lorsque des contraintes d'efficacité entrent en jeu. Quelques exceptions sont à noter comme (Pitt, 1989; Angluin, 1990; Kearns & Valiant, 1989; Haussler *et al.*, 1991; de la Higuera, 1997; Parekh & Honavar, 2000). Si l'approche habituelle est d'introduire un paradigme d'apprentissage et d'étudier différentes classes de langages pour ce paradigme, nous choisissons ici de fixer les classes de langages et de faire une étude transversale de leur apprenabilité selon les paradigmes.

Concernant les AFD, nous complétons la liste des résultats qui pour la plupart sont connus. Concernant les boules de mots, nos résultats sont généralement négatifs : l'identification à la limite à partir d'exemples et de contre-exemples est impossible pour la plupart des définitions, ainsi que l'apprentissage à partir de requêtes d'équivalence et d'appartenance. L'apprentissage PAC est lui aussi impossible en temps polynomial, sauf si $\mathcal{RP} = \mathcal{NP}$. D'un autre côté, les erreurs sont généralement dues aux contre-exemples. Ainsi, on montre qu'il est parfois plus facile d'apprendre à partir d'exemples positifs seulement que d'exemples et de contre-exemples.

La section 2 rassemble un certain nombre de définitions préliminaires. Dans les sections 3, 4 et 5, nous nous concentrons sur les *bonnes boules* et sur les automates, et nous présentons les résultats concernant respectivement l'apprentissage PAC, l'apprentissage actif et l'identification à la limite. Nous envisageons ensuite le cas général des boules en section 6, avant de conclure en section 7.

2 Définitions

Un *alphabet* Σ est un ensemble fini non vide de symboles appelés *lettres*. On supposera par la suite que $|\Sigma| \geq 2$. Un *mot* $w = a_1 \cdots a_n$ est une séquence finie de lettres. On

écrira λ le mot vide et $|w|$ la longueur de w. On dit que u est un *sous-mot* de v, dénoté $u \preceq v$, si $u = a_1 \dots a_n$ et il existe $u_0, \dots, u_n \in \Sigma^\star$ tels que $v = u_0 a_1 u_1 \dots a_n u_n$. On note $lcs(u, v)$ l'ensemble des plus longs sous-mots communs à u et v.

Soit $\Sigma^\star$ l'ensemble des mots de Σ. Un *langage* est un sous-ensemble $L \subseteq \Sigma^\star$. Soit $\mathbb{N}$ l'ensemble des entiers positifs. Pour tout $k \in \mathbb{N}$, soient $\Sigma^{\leq k} = \{w \in \Sigma^\star : |w| \leq k\}$ et $\Sigma^{>k} = \{w \in \Sigma^\star : |w| > k\}$. On pose $A \oplus B = (A \setminus B) \cup (B \setminus A)$.

L'inférence grammaticale a pour but d'apprendre les langages d'une classe $\boldsymbol{\mathcal{L}}$ représentée par les grammaires d'une classe $\boldsymbol{\mathcal{G}}$. $\boldsymbol{\mathcal{L}}$ et $\boldsymbol{\mathcal{G}}$ sont reliées par une fonction de *nommage* $\mathbb{L} : \boldsymbol{\mathcal{G}} \to \boldsymbol{\mathcal{L}}$ qui est totale ($\forall G \in \boldsymbol{\mathcal{G}}, \mathbb{L}(G) \in \boldsymbol{\mathcal{L}}$) et surjective ($\forall L \in \boldsymbol{\mathcal{L}}, \exists G \in \boldsymbol{\mathcal{G}}$ *tel que* $\mathbb{L}(G) = L$). Pour n'importe quels mot $w \in \Sigma^\star$ et langage $L \in \boldsymbol{\mathcal{L}}$, nous écrirons $L \models w$ si $w \in L$. Concernant les grammaires, elles peuvent être vues comme un ensemble d'informations permettant à un analyseur (*parser*) de reconnaître des mots. Pour n'importe quels mot $w \in \Sigma^\star$ et grammaire $G \in \boldsymbol{\mathcal{G}}$, nous écrirons $G \vdash w$ si l'analyseur reconnaît w. Syntaxe et sémantique sont liées : $G \vdash w \iff \mathbb{L}(G) \models w$.

Dans la suite, nous considérerons essentiellement les paradigmes d'apprentissage sujets à des contraintes d'efficacité. Pour les définir, nous utiliserons $\|G\|$ pour noter la taille de la grammaire G (par exemple, le nombre d'états dans le cas des AFD). De plus, étant donné un ensemble de mots X, nous noterons $\|X\|$ la somme des longueurs des mots de X. La notation $|\cdot|$ sera utilisée pour la cardinalité des ensembles.

La *distance d'édition* $d(w, w')$ est le plus petit nombre d'*opérations d'édition* nécessaire pour transformer w en w' (Levenshtein, 1965; Crochemore *et al.*, 2001). Une opération est soit (1) une *suppression* : $w = uav$ et $w' = uv$, soit (2) une *insertion* $w = uv$ et $w' = uav$, soit (3) une *substitution* : $w = uav$ et $w' = ubv$, où $u, v \in \Sigma^\star$, $a, b \in \Sigma$ et $a \neq b$. Par exemple, $d(abaa, aab) = 2$ puisque $a\underline{b}aa \to aa\underline{a} \to aab$ et la réécriture de $abaa$ en aab ne peut se faire en moins de 2 étapes. $d(w, w')$ peut être calculé en temps $\mathcal{O}(|w| \cdot |w'|)$ par programmation dynamique (Wagner & Fisher, 1974).

La distance d'édition est une *métrique*. Il est donc naturel d'introduire les boules de $\Sigma^\star$. La *boule de centre* $o \in \Sigma^\star$ *et de rayon* $r \in \mathbb{N}$, dénotée par $B_r(o)$, est l'ensemble des mots à distance au plus r de o : $B_r(o) = \{w \in \Sigma^\star : d(o, w) \leq r\}$. Par exemple, si $\Sigma = \{a, b\}$, $B_1(ba) = \{a,b,aa,ba,bb,aba,baa,bab,bba\}$ et $B_r(\lambda) = \Sigma^{\leq r}$ pour tout $r \in \mathbb{N}$. On notera $\boldsymbol{\mathcal{BALL}}(\Sigma)$ la famille de toutes les boules.

Du point de vue de l'apprentissage, on représentera la boule $B_r(o)$ par la paire (o, r) qui servira de grammaire. En effet, sa taille est $|o| + \log r$ (ce qui correspond au nombre de bits nécessaire pour encoder la représentation[1]). De plus, l'analyseur qui permet de décider si $w \in B_r(o)$ est simple : il (1) calcule $d(o, w)$ et (2) vérifie que cette distance est $\leq r$, ce qui est réalisable en temps $\mathcal{O}(|o| \cdot |w| + \log r)$. Enfin, comme $|\Sigma| \geq 2$, on peut montrer que (o, r) est une représentation unique, et donc *canonique*, de $B_r(o)$ (Becerra-Bonache *et al.*, 2007). Du coup, nous noterons aussi $\boldsymbol{\mathcal{BALL}}(\Sigma)$ la classe des grammaires associées aux boules.

Une *bonne boule* est une boule dont le rayon est au plus égal à la longueur du centre. L'avantage d'utiliser les bonnes boules est qu'il existe une relation polynomiale entre la taille du centre et la taille des plus longs mots de la boule. On notera $\boldsymbol{\mathcal{GB}}(\Sigma)$ la classe de toutes les bonnes boules (et des grammaires correspondantes).

[1] Notons que $|o| + r$ *n'est pas* une mesure correcte de taille : cela correspondrait à un encodage en base 1 du rayon r, ce qui n'est pas considéré comme raisonnable (Garey & Johnson, 1979).

Un *automate fini déterministe* est un quintuplet $A = \langle \Sigma, Q, q_0, F, \delta \rangle$ tel que Q est un ensemble fini d'états, $q_0 \in Q$ est un état dit initial, $\delta : Q \times \Sigma \to Q$ est une fonction de transition et $F \subseteq Q$ est un ensemble d'états dits finaux. On étend δ à $\Sigma^\star$. On définit le langage reconnu par A en posant $\mathbb{L}(A) = \{w \in \Sigma^\star : \delta(q_0, w) \in F\}$. La taille d'un AFD désigne son nombre d'états. On note $\mathcal{AFD}(\Sigma)$ la classe de tous les AFD sur un alphabet Σ donné.

Si l'inférence des AFD est un problème étudié depuis 40 ans (Gold, 1967; Pitt, 1989; Angluin, 1987a), apprendre des boules de mots est un problème récent (Becerra-Bonache *et al.*, 2007) motivé par la question d'apprendre des classes de langages en milieu bruité. Une question préliminaire se pose cependant : si toute boule codable en n bits était représentable par un AFD à $p(n)$ états (où $p()$ est un polynôme fixé), il suffirait de réduire l'apprenabilité d'une classe à celle de l'autre. Cependant, ce changement de représentation n'existe pas, et l'on ignore encore si toute (bonne) boule est représentable ou non par un automate de taille polynomiale. Cela implique donc que l'étude séparée des deux classes (boules et AFD) est utile.

3 Résultats de PAC apprenabilité

Introduit par (Valiant, 1984), le paradigme PAC a été largement utilisé en apprentissage automatique. Intuitivement, il vise à construire, avec une grande confiance, de bonnes approximations d'un concept cible inconnu.

Définition 1 (Hypothèse ϵ-bonne)
Soient G la grammaire cible et H une grammaire hypothèse. Soient $\mathcal{D}$ une distribution sur $\Sigma^\star$ et $\epsilon > 0$. On dit que H est une hypothèse ϵ-bonne *par rapport à G si $Pr_{\mathcal{D}}(x \in \mathbb{L}(G) \oplus \mathbb{L}(H)) < \epsilon$.*

L'apprenabilité PAC des grammaires à partir de mots (de longueur variable) a toujours été compliquée à définir (Warmuth, 1989; Kearns & Valiant, 1989; Kearns & Vazirani, 1994). En effet, dans la définition standard de l'apprentissage PAC, l'algorithme peut demander à un oracle des exemples tirés aléatoirement selon la distribution $\mathcal{D}$. Or dans le cas des mots, il existe toujours un risque de tirer un mot trop long pour être simplement lu en temps polynomial. Pour éviter ce problème, on tire des exemples à partir d'une distribution restreinte à des mots plus petits qu'une certaine valeur donnée par le lemme suivant :

Lemme 1
Soit $\mathcal{D}$ une distribution sur $\Sigma^\star$. Alors $\forall \epsilon, \delta > 0$, avec probabilité $> 1 - \delta$, si on tire un échantillon X d'au moins $\frac{1}{\epsilon} \ln \frac{1}{\delta}$ mots suivant $\mathcal{D}$, la probabilité qu'un nouveau mot x soit plus long que n'importe quel autre mot de X est $< \epsilon$. Formellement, si on définit $\mu_X = \max\{|y| : y \in X\}$, alors $Pr_{\mathcal{D}}(|x| > \mu_X) < \epsilon$.

Preuve : Soit ℓ le plus petit entier tel que $Pr(\Sigma^{>\ell}) < \epsilon$. Une condition suffisante pour que $Pr_{\mathcal{D}}(|x| > \mu_X) < \epsilon$ est que l'on prenne un échantillon X suffisamment grand pour être pratiquement sûr (avec probabilité $> 1 - \delta$) d'avoir au moins un mot plus long que ℓ. Clairement, la probabilité que n mots tirés selon $\mathcal{D}$ soient tous de longueur

$< \ell$ est $\leq (1-\epsilon)^n$. Donc la probabilité qu'il y ait au moins un mot de longueur $\geq \ell$ parmi n mots est $> 1-(1-\epsilon)^n$. Pour construire X, on veut donc $1-(1-\epsilon)^n > 1-\delta$, c'est-à-dire $(1-\epsilon)^n < \delta$. Comme $(1-\epsilon)^n \leq e^{-n\epsilon}$, il suffit que $n \geq \frac{1}{\epsilon}\ln\frac{1}{\delta}$ pour obtenir une valeur convenable de μ_X. □

On demande maintenant à un algorithme d'apprendre une grammaire étant donnés les paramètres d'erreur ϵ et de confiance δ. En outre, on doit lui fournir une borne n sur la taille de la grammaire cible et une borne m sur la longueur des exemples d'apprentissage (calculée grâce au lemme 1). L'algorithme peut demander des exemples à un oracle. On notera EX() la requête d'un exemple ou d'un contre-exemple et POS-EX() la requête d'un exemple positif uniquement. Si, en outre, on souhaite un mot de longueur $\leq m$, on utilisera EX(m) et POS-EX(m). Formellement, l'oracle retournera alors un mot tiré selon $\mathcal{D}$, $\mathcal{D}(\mathbb{L}(G))$, $\mathcal{D}(\Sigma^{\leq m})$ ou $\mathcal{D}(\mathbb{L}(G)\cap\Sigma^{\leq m})$ selon la requête, où $\mathcal{D}(L)$ est la restriction de $\mathcal{D}$ aux mots de L : $Pr_{\mathcal{D}(L)}(x) = Pr_{\mathcal{D}}(x)/Pr_{\mathcal{D}}(L)$ si $x \in L$, 0 sinon. $Pr_{\mathcal{D}(L)}(x)$ n'est pas défini si $L = \emptyset$.

Définition 2 (Polynomialement PAC apprenable)
Soit $\mathcal{G}$ une classe de grammaires. $\mathcal{G}$ est PAC *apprenable s'il existe un algorithme $\mathfrak{A}$ tel que $\forall\epsilon,\delta > 0$, pour toute distribution $\mathcal{D}$ sur $\Sigma^\star$, $\forall n \in \mathbb{N}$, $\forall G \in \mathcal{G}$ de taille $\leq n$, si $\mathfrak{A}$ a accès à* EX*(), ϵ, δ, n et m, alors avec probabilité $> 1-\delta$, $\mathfrak{A}$ retourne une hypothèse ϵ-bonne par rapport à G. Si $\mathfrak{A}$ s'exécute en temps polynomial en $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $|\Sigma|$, m et n, on dit que $\mathcal{G}$ est* polynomialement PAC apprenable.

Notons que pour pouvoir traiter la longueur non bornée des exemples on peut utiliser $\epsilon' = \frac{\epsilon}{2}$ et une fraction de δ pour calculer m et accepter de faire une erreur d'au plus ϵ' sur tous les mots de longueur supérieure à m, et utiliser ainsi EX(m) au lieu de EX().

Les techniques typiques prouvant la non apprenabilité PAC reposent souvent sur des hypothèses de complexité (Pitt & Valiant, 1988). Rappelons ici que $\mathcal{RP}$ (‘*Randomised Polynomial Time*’) est la classe de complexité des problèmes de décisions pour lesquels une machine de Turing probabiliste existe et (1) elle fonctionne en temps polynomial en la taille des données d'entrée, (2) sur une instance négative, elle retourne toujours NON et (3) sur une instance positive, elle retourne OUI avec une probabilité $> \frac{1}{2}$ (sinon, elle retourne NON). Un tel algorithme est randomisé puisqu'il a le droit de tirer une pièce à pile ou face pendant son exécution. L'algorithme ne fait des erreurs que dans le cas positif, et il est important de noter que puisque cette erreur est $< \frac{1}{2}$, en répétant l'exécution de l'algorithme autant de fois que nécessaire, on peut la rendre aussi petite que l'on veut. Nous utiliserons l'hypothèse communément admise que $\mathcal{RP} \neq \mathcal{NP}$. Pour une étude plus approfondie, voir par exemple (Papadimitriou, 1994).

Nous allons montrer que les bonnes boules ne sont pas polynomialement PAC apprenables. La preuve suit les lignes classiques de ce genre de résultats : nous montrons d'abord que le problème de consistance associé est $\mathcal{NP}$-difficile, par réduction à un problème $\mathcal{NP}$-complet (*plus long sous-mot commun*). Alors, s'il existait un algorithme polynomial PAC-apprenant les boules, cet algorithme nous fournirait une preuve que ce problème $\mathcal{NP}$-complet est dans $\mathcal{RP}$.

Lemme 2
Les problèmes suivants sont $\mathcal{NP}$-complets :

1. ***Plus long sous-mot commun*** *(*PLSC*) : étant donnés n mots $x_1 \dots x_n$ et un entier k, existe-t-il un mot w de longueur k et sous-mot de chaque x_i ?*
2. ***Plus long sous-mot communs à des mots d'une longueur donnée*** *(*PLSCMLD*) : étant donnés n mots $x_1 \dots x_n$ de longueurs $2k$, existe-t-il un mot w de longueur k sous-mot de chaque x_i ?*
3. ***Boule consistante*** *(*BC*) : étant donnés deux ensembles X_+ et X_- de mots sur un alphabet Σ, existe-t-il une bonne boule contenant X_+ et aucun mot de X_- ?*

Preuve : (1) Voir (Maier, 1977; Garey & Johnson, 1979). (2) Voir (de la Higuera & Casacuberta, 2000), problème LCS0, page 42. (3) Nous utilisons une réduction du problème PLSCMLD. Dans X_+, nous mettons λ et les mots de longueurs $2k$. Nous construisons X_- en prenant chaque mot de longueur $2k$ de X_+, et en leur insérant chaque lettre possible, une fois seulement (ainsi on construit au plus $n(2k+1)|\Sigma|$ mots de taille $2k+1$). Une boule qui contient X_+ et aucun mot de X_- a nécessairement un centre de longueur k et un rayon de k (puisque nous nous concentrons sur les bonnes boules). Le centre est alors un sous-mot commun à tous les mots de longueurs $2k$. Inversement, si une boule est construite avec un sous-mot de longueur k comme centre, cette boule est de rayon k, contient aussi λ, et à cause du rayon, ne contient pas d'éléments de X_-. Finalement, le problème est dans $\mathcal{NP}$, puisqu'étant donné un centre u, il est facile de vérifier si $\max_{x \in X_+} d(u,x) < \min_{x \in X_-} d(u,x)$. □

Théorème 1
$\mathcal{GB}(\Sigma)$ n'est pas polynomialement PAC *apprenable.*

Preuve : Supposons que $\mathcal{GB}(\Sigma)$ soit polynomialement PAC apprenable avec $\mathfrak{A}$ et prenons l'instance $\langle X_+, X_- \rangle$ du problème BC. On pose $h = |X_+| + |X_-|$ et on définit sur $\Sigma^\star$ la distribution $Pr(x) = \frac{1}{h}$ si $x \in X_+ \cup X_-$, 0 sinon. Soient $\epsilon = \frac{1}{h+1}$, $\delta < \frac{1}{2}$, $m = n = \max\{|w| : w \in X_+\}$. Soit $B_r(o)$ la boule retournée par une exécution de $\mathfrak{A}(\epsilon, \delta, m, n)$. On teste si $X_+ \subseteq B_r(o)$ et $X_- \cap B_r(o) = \emptyset$. S'il n'existe pas de boule consistante, alors $B_r(o)$ est nécessairement inconsistante avec les données, donc le test ci-dessus est faux. S'il existe une boule consistante, alors $B_r(o)$ est une hypothèse ϵ-bonne, avec $\epsilon < \frac{1}{h}$. Donc, avec probabilité au moins $1 - \delta > \frac{1}{2}$, il n'y a pas d'erreur du tout et le test sera vrai. Clairement, cette procédure s'exécute en temps polynomial en $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $|\Sigma|$, m et n. Ainsi, si les bonnes boules étaient PAC apprenables, il existerait un algorithme randomisé pour le problème BC, qui est $\mathcal{NP}$-complet par le lemme 2. □

L'apprentissage PAC des AFD a, lui, fait l'objet de plusieurs études (Pitt, 1989). C'est un problème difficile (Pitt & Warmuth, 1993) : un algorithme efficace pour les apprendre permettrait la résolution de problèmes $\mathcal{NP}$-difficiles comme le décodage de clés RSA ou la factorisation des entiers de Blum :

Théorème 2 (Kearns & Vazirani (1994))
$\mathcal{AFD}(\Sigma)$ n'est pas polynomialement PAC *apprenable.*

Dans certains cas, il est également intéressant de PAC apprendre à partir de données positives seulement. Dans ce contexte, durant la phase d'apprentissage, les exemples sont tirés suivant POS-EX() tandis que durant la phase de test, l'échantillonnage se fait suivant EX(). Une fois encore, on peut utiliser POS-EX(m), où m est obtenu grâce au lemme 1 et à un coût supplémentaire. Il est facile de voir que si dans une classe il y a deux langages L_1 et L_2 tels que $L_1 \cap L_2$ n'est pas vide, alors la classe n'est pas polynomialement PAC apprenable à partir d'exemples positifs seulement :

Lemme 3
Si $\mathcal{L}$ contient deux langages L_1 et L_2 tels que $L_1 \cap L_2 \neq \emptyset$, $L_1 \not\subset L_2$ et $L_2 \not\subset L_1$, alors $\mathcal{L}$ n'est pas polynomialement PAC *apprenable à partir d'exemples positifs seulement.*

Preuve : Soient $w_1 \in L_1 - L_2$, $w_2 \in L_2 - L_1$ et $w_3 \in L_1 \cap L_2$. Ces trois mots existent par définition. Considérons maintenant la distribution $\mathcal{D}_1$ où $\Pr_{\mathcal{D}_1}(w_1) = \Pr_{\mathcal{D}_1}(w_3) = \frac{1}{2}$ et la distribution $\mathcal{D}_2$ où $\Pr_{\mathcal{D}_2}(w_2) = \Pr_{\mathcal{D}_2}(w_3) = \frac{1}{2}$. Il est facile de voir que lors de l'apprentissage de L_1 à partir de $\mathcal{D}_2$ ou de L_2 à partir de $\mathcal{D}_1$, les exemples tirés ne seront que des répétitions du mot w_3. Mais lors du test, l'erreur sera forcément $\geq \frac{1}{2}$. □

Théorème 3
(1) $\mathcal{GB}(\Sigma)$ et (2) $\mathcal{AFD}(\Sigma)$ ne sont pas polynomialement PAC *apprenable à partir d'exemples positifs seulement.*

Preuve : (1) Soient $L_1 = B_1(a)$ et $L_2 = B_1(b)$. On pose $w_1 = aa$, $w_2 = bb$ et $w_3 = ab$. Le résultat découle alors du lemme 3. (2) On considère les AFD reconnaissant L_1 et L_2 et les mêmes trois exemples. □

4 Résultats en apprentissage actif

Apprendre à partir de requêtes met en jeu un *apprenant* capable d'interroger un *oracle* en utilisant des requêtes d'un ensemble donné (Angluin, 1987b). Le but de l'apprenant est d'identifier la représentation d'un langage inconnu. L'oracle connaît le langage cible L et répond correctement aux requêtes. Nous utiliserons ici trois types de requêtes :

- les requêtes d'appartenance (MQ) : l'apprenant soumet un mot w à l'oracle qui répond OUI si $w \in L$, NON sinon.
- les requêtes d'équivalence (EQ) : l'apprenant soumet (la représentation d') un langage K à l'oracle qui répond OUI si $K = L$, et sinon retourne un mot $K \oplus L$.
- les requêtes de correction basées sur la distance d'édition (CQ_{EDIT}) : l'apprenant soumet un mot w à l'oracle qui répond OUI si $w \in L$, et retourne une correction $z \in L$ à distance d'édition minimum de w sinon.

On appelle REQ la classe des requêtes. Par exemple, si l'apprenant a seulement le droit de faire des requêtes d'appartenance, on aura $\text{REQ} = \{\text{MQ}\}$.

Définition 3
Une classe $\mathcal{G}$ est polynomialement identifiable à partir de REQ *s'il existe un algorithme $\mathfrak{A}$ capable d'identifier chaque $G \in \mathcal{G}$ et tel que, à chaque appel de requêtes, le nombre*

total de requêtes ainsi que le temps utilisé jusqu'à cet appel par $\mathfrak{A}$ *est polynomial en* $\|G\|$ *et en la taille de l'information présentée par l'oracle jusqu'à ce point.*

Dans le cas des bonnes boules, nous avons établi :

Théorème 4 (Becerra-Bonache *et al.* (2007))
(1) $\mathcal{GB}(\Sigma)$ *n'est pas polynomialement identifiable à partir de* MQ *et* EQ. *(2)* $\mathcal{GB}(\Sigma)$ *est polynomialement identifiable à partir de* CQ_{EDIT}.

Notons cependant que si l'on donne à l'apprenant un mot de la bonne boule, il peut alors utiliser un nombre polynomial de MQ seulement.

Dans le cas des AFD, on a :

Théorème 5
(1) $\mathcal{AFD}(\Sigma)$ *est polynomialement identifiable à partir de* MQ *et* EQ *(Angluin, 1987a) mais ne l'est pas à partir (2) de* MQ *uniquement (Angluin, 1987b), (3) de* EQ *uniquement (Angluin, 1990), ni (4) de* CQ_{EDIT}.

Preuve : (4) Soient $\mathcal{A}_w$ l'AFD reconnaissant $\Sigma^\star \setminus \{w\}$, $n \in \mathbb{N}$ et $\mathcal{AFD}_{\leq n} = \{\mathcal{A}_w : w \in \Sigma^{\leq n}\}$. Suivant (Angluin, 1990), nous décrivons un adversaire qui maintient un ensemble X des AFD possibles. Au début, $X = \mathcal{AFD}_{\leq n}$. À chaque requête de correction w, l'adversaire répond OUI, éliminant de ce fait un unique AFD : $\mathcal{A}_w$ et n'apportant aucune autre information sur les autres AFD. Comme il y a $\Omega(|\Sigma|^n)$ AFD dans $\mathcal{AFD}_{\leq n}$, les identifier nécessite $\Omega(|\Sigma|^n)$ requêtes. □

5 Résultats d'identification à la limite

Dans le paradigme standard d'identification à la limite de Gold, un apprenant reçoit une séquence infinie d'informations (présentation) qui devrait l'aider à trouver la représentation $G \in \mathcal{G}$ d'un langage cible inconnu $L \in \mathcal{L}$. L'ensemble des présentations admissibles est noté **Pres**, chaque présentation étant une fonction $\mathbb{N} \to X$ où X est un ensemble. Étant donné $\boldsymbol{f} \in$ **Pres**, on notera $\boldsymbol{f}_m$ les $m+1$ premiers éléments de $\boldsymbol{f}$, et $\boldsymbol{f}(n)$ son n-ième élément. Par la suite, on utilisera deux sortes de présentations :

- **Pres** $=$ TEXTE : tous les mots de L sont présentés : $\boldsymbol{f}(\mathbb{N}) = \mathbb{L}(G)$;
- **Pres** $=$ INFORMATEUR : la présentation est composée de paires étiquetées (w, l) où $(w \in L \Rightarrow l = +)$ et $(w \notin L \Rightarrow l = -)$: $\boldsymbol{f}(\mathbb{N}) = \mathbb{L}(G) \times \{+\} \cup \overline{\mathbb{L}(G)} \times \{-\}$;

On notera **Pres** $=$ PRESENTATION lorsqu'un résultat concernera indifféremment les TEXTE et les INFORMATEUR.

Définition 4
On dit que $\mathcal{G}$ *est identifiable à la limite à partir de* **Pres** *s'il existe un algorithme* $\mathfrak{A}$ *tel que pour tout* $G \in \mathcal{G}$ *et pour toute présentation* $\boldsymbol{f}$ *de* $\mathbb{L}(G)$*, il existe un rang* n *tel que pour tout* $m \geq n$, $\mathfrak{A}(\boldsymbol{f}_m) = \mathfrak{A}(\boldsymbol{f}_n)$ *et* $\mathbb{L}(\mathfrak{A}(\boldsymbol{f}_m)) = \mathbb{L}(G)$.

On peut obtenir un grand nombre de résultats d'apprenabilité grâce à cette définition. Cependant, l'absence de contraintes peut conduire à des algorithmes inutilisables. C'est

pourquoi plusieurs auteurs ont essayé de définir une identification à la limite *polynomiale*, en introduisant différents critères d'efficacité et en les combinant.

5.1 Critères d'identification polynomiale

Premièrement, il est raisonnable de penser que la polynomialité doit concerner la quantité de temps dont l'algorithme dispose pour apprendre :

Définition 5 (Temps de mise à jour polynomial)
On dit qu'un algorithme $\mathfrak{A}$ *a un* temps de mise à jour polynomial *s'il existe un polynôme* $p()$ *tel que, pour chaque présentation* $\boldsymbol{f}$ *et chaque entier* n*, construire* $\mathfrak{A}(\boldsymbol{f}_n)$ *nécessite un temps en* $\mathcal{O}(p(\|\boldsymbol{f}_n\|))$.

Cependant, il est connu que le temps de mise à jour polynomial n'est pas suffisant (Pitt, 1989). En effet, un apprenant pourrait recevoir un nombre exponentiel d'exemples sans rien faire d'autre qu'attendre, puis utiliser la grande quantité de temps qu'il a accumulé pour résoudre un problème $\mathcal{NP}$-difficile...

Deuxièmement, la polynomialité doit aussi concerner la quantité minimum de données qu'un algorithme doit recevoir pour apprendre :

Définition 6 (Ensemble caractéristique polynomial)
Une classe de grammaires $\boldsymbol{\mathcal{G}}$ *admet un* ensemble caractéristique polynomial *s'il existe un algorithme* $\mathfrak{A}$ *et un polynôme* $p()$ *tels que pour toute grammaire* $G \in \boldsymbol{\mathcal{G}}$*, il existe* $\text{Cs} \subseteq X$ *avec (1)* $\|\text{Cs}\| \leq p(\|G\|)$*, (2)* $\mathbb{L}(\mathfrak{A}(\text{Cs})) = \mathbb{L}(G)$ *et (3) pour tout* $\boldsymbol{f} \in \textbf{\textit{Pres}}$*, pour tout* $n \geq 0$*, si* $\text{Cs} \subseteq \boldsymbol{f}_n$ *alors* $\mathfrak{A}(\boldsymbol{f}_n) = \mathfrak{A}(\text{Cs})$*. Un tel ensemble* Cs *est appelé* ensemble caractéristique *de* G *pour* $\mathfrak{A}$*, et si* $\mathfrak{A}$ *existe, on dit que* $\boldsymbol{\mathcal{G}}$ est identifiable à la limite en temps Cs polynomial.

Notons que si une classe de grammaires admet seulement des ensembles caractéristiques dont la taille est exponentielle, alors aucun algorithme ne pourra converger sans recevoir une quantité déraisonnable de données ! L'existence d'un ensemble caractéristique polynomial est donc nécessaire mais pas suffisant. Des définitions plus contraignantes d'ensembles caractéristiques polynomiaux existent (de la Higuera, 1997).

Troisièmement, la polynomialité peut concerner le nombre d'erreurs implicites de prédiction (Pitt, 1989) que commet l'algorithme durant son apprentissage :

Définition 7 (Erreur implicite de prédiction)
Étant donnés un algorithme d'apprentissage $\mathfrak{A}$ *et une présentation* $\boldsymbol{f}$*, on dit que* $\mathfrak{A}$ *fait* une erreur implicite de prédiction *(Implicit Prediction Error,* IPE*)* au temps n *si* $\mathfrak{A}(\boldsymbol{f}_{n-1}) \not\vdash \boldsymbol{f}(n)$*. On dit que* $\mathfrak{A}$ *est* consistant *s'il change d'avis chaque fois qu'une erreur de prédiction est détectée avec le nouvel élément.*

Définition 8 (Identification IPE polynomiale)
Un algorithme $\mathfrak{A}$ *identifie une classe* $\boldsymbol{\mathcal{G}}$ *à la limite en temps* IPE *polynomial si (1)* $\mathfrak{A}$ *identifie* $\boldsymbol{\mathcal{G}}$ *à la limite, (2)* $\mathfrak{A}$ *a un temps de mise à jour polynomial et (3)* $\mathfrak{A}$ *fait un nombre polynomial d'erreurs implicites de prédiction : Soit* $\#\text{IPE}(\boldsymbol{f}) = |\{k \in \mathbb{N} : \mathfrak{A}(\boldsymbol{f}_k) \not\vdash \boldsymbol{f}(k+1)\}|$*; il existe un polynôme* $p()$ *tel que, pour chaque grammaire* G *et chaque présentation* $\boldsymbol{f}$ *de* $\mathbb{L}(G)$*,* $\#\text{IPE}(\boldsymbol{f}) \leq p(\|G\|)$.

Notons que la première condition n'implique pas les deux autres.

Quatrièmement, on peut borner le nombre de changement d'avis (*Mind Changes*, MC) plutôt que le nombre d'IPE (Angluin & Smith, 1983) :

Définition 9 (Changement d'avis)
Étant donnés un algorithme $\mathfrak{A}$ *et une présentation* $\boldsymbol{f}$*, on dit que* $\mathfrak{A}$ change d'avis au temps n *si* $\mathfrak{A}(\boldsymbol{f}_n) \neq \mathfrak{A}(\boldsymbol{f}_{n-1})$*. On dit que* $\mathfrak{A}$ *est* conservatif *s'il ne change jamais d'avis lorsque son hypothèse courante est consistante avec le nouvel élément présenté.*

Définition 10 (Identification MC polynomiale)
Un algorithme $\mathfrak{A}$ *identifie une classe* $\boldsymbol{\mathcal{G}}$ *à la limite en temps* MC *polynomial si (1)* $\mathfrak{A}$ *identifie* $\boldsymbol{\mathcal{G}}$ *à la limite, (2)* $\mathfrak{A}$ *a un temps de mise à jour polynomial et (3)* $\mathfrak{A}$ *fait un nombre polynomial de changement d'avis : Soit* $\#\text{MC}(\boldsymbol{f}) = |\{k \in \mathbb{N} : \mathfrak{A}(\boldsymbol{f}_k) \neq \mathfrak{A}(\boldsymbol{f}_{k+1})\}|$ *; il existe un polynôme* $p()$ *tel que, pour chaque grammaire* G *et chaque présentation* $\boldsymbol{f}$ *de* $\mathbb{L}(G)$*,* $\#\text{MC}(\boldsymbol{f}) \leq p(\|G\|)$*.*

Concernant les deux dernières notions, on peut noter que si un algorithme $\mathfrak{A}$ est consistant alors $\#\text{IPE}(\boldsymbol{f}) \leq \#\text{MC}(\boldsymbol{f})$ pour chaque présentation $\boldsymbol{f}$. De la même façon, si $\mathfrak{A}$ est conservatif alors $\#\text{MC}(\boldsymbol{f}) \leq \#\text{IPE}(\boldsymbol{f})$. Aussi, on en déduit le lemme suivant :

Lemme 4
Si $\mathfrak{A}$ *identifie la classe* $\boldsymbol{\mathcal{G}}$ *à la limite en temps* MC *polynomial et qu'il est consistant, alors* $\mathfrak{A}$ *identifie* $\boldsymbol{\mathcal{G}}$ *en temps* IPE *polynomial. De même, si* $\mathfrak{A}$ *identifie* $\boldsymbol{\mathcal{G}}$ *en temps* IPE *polynomial et qu'il est conservatif, alors* $\mathfrak{A}$ *identifie* $\boldsymbol{\mathcal{G}}$ *en temps* MC *polynomial.*

5.2 Identification à partir de TEXTE

L'objectif de cette section est de montrer que :

Théorème 6
$\boldsymbol{\mathcal{GB}}(\Sigma)$ *est identifiable à la limite à partir de* TEXTE *(1) en temps* MC *polynomial, (2) en temps* IPE *polynomial et (3) en temps* CS *polynomial.*

Remarquons que comme les AFD reconnaissent une classe *superfinie* de langages (*i.e*, contenant tous les langages finis et au moins un langage infini), on ne peut pas les identifier à la limite à partir d'exemples positifs seulement :

Théorème 7 (Gold (1967))
$\boldsymbol{\mathcal{AFD}}(\Sigma)$ *n'est pas identifiable à la limite à partir de* TEXTE*.*

Pour montrer le théorème 6, nous introduisons la notion de *témoin de minimalité* qui nous servira aussi pour le théorème 9.

Définition 11 (Témoin de minimalité)
Un témoin de minimalité pour $G \in \boldsymbol{\mathcal{G}}$ *est un ensemble* X *compatible avec* G *tel que* $\forall G' \in \boldsymbol{\mathcal{G}}$*, si* X *est aussi compatible avec* G'*, alors* $\|G'\| > \|G\|$ *ou bien* $G' = G$*. Il peut exister 0, 1 ou plusieurs témoins pour* G *; on dira que* G admet *un témoin s'il en existe au moins un.*

Étant donné un ensemble $X = \{x_1, \ldots x_n\}$ de mots (ensemble d'apprentissage), on note X^{max} et X^{min} les mots de longueur maximum et minimum dans X.

Dans le cas des boules, X est un témoin pour $B_r(o)$ s'il contient les mots u, v, w tels que (1) $u, v \in X^{max}$, (2) $w \in X^{min}$, (3) $|u| - |w| = 2r$, (4) $lcs(u, v) = \{o\}$, (5) $|o| = |u| - r$ et (6) $X \subseteq B_r(o)$. En effet, on peut alors montrer que $B_r(o)$ est l'unique plus petite boule contenant X. Plus précisément, si $X \subseteq B_{r'}(o')$ alors soit $r' > r$, soit ($r' = r$ et $o' = o$). Cette propriété, assez technique, repose uniquement sur les contraintes (fortes) imposées par les points précédents, et les propriétés élémentaires de la distance d'édition.

D'autre part, il existe un algorithme polynomial (en $\|X\|$) pour vérifier si X est le témoin de minimalité pour une boule, et qui calcule cette boule $B_r(o)$. Il suffit d'extraire X^{max} et X^{min}, de vérifier si la différence des longueurs entre ces deux ensembles est paire, ce qui permet de trouver r ; ensuite, on cherche $u, v \in X^{max}$ admettant un plus petit sous-mot commun unique, ce qui permet de trouver o. C'est un point délicat car le cardinal de $lcs(u, v)$ peut être $> 1.442^n$ (Greenberg, 2003) (où $n = |u| = |v|$). Néanmoins, une structure de données comme le LCS-graphe (Greenberg, 2002) permet de conclure en temps polynomial ; enfin, on teste les conditions (5) et (6).

Algorithm 1: Identification des bonnes boules à partir de texte.

```
Data: Un texte f = {x_1, x_2, ...}
read(x_1) ; c ← x_1 ; output (x_1, 0);
while true do
    read(x_i);
    if f_i est un témoin de minimalité pour B_r(o) then
        output (o, r) (* boule valide *)
    else
        if c ∉ X^max then c ← un mot de X^max;
        output (c, |c|) (* boule poubelle *)
    end
end
```

On peut maintenant envisager l'algorithme 1. Cet algorithme identifie bien $\mathcal{GB}(\Sigma)$ à la limite car si $B_r(o)$ dénote la boule cible, alors l'algorithme verra un jour les mots $u = a^r o, v = b^r o$ et w de longueur $|o| - r$ qui répondent aux exigences du témoin de minimalité pour $B_r(o)$. De plus, il commet un nombre de MC polynomial. En effet, il ne change d'hypothèse en faveur d'une boule valide que si celle-ci à un rayon plus grand que la boule valide précédente, ce qui arrive au plus r fois. De plus, il ne change d'hypothèse en faveur d'une boule poubelle que lorsqu'il abandonne une boule valide ou une boule poubelle ne contenant manifestement pas tous les exemples, ce qui arrive au plus $r + 2r$ fois. Le nombre total de MC est donc $\leq 4r$, d'où le point (1).

Concernant le point (2), on peut noter de l'Algo. 1 est consistant (c'est l'intérêt d'utiliser des boules poubelles). Aussi, le point (2) découle du point (1) et du lemme 4. Enfin, n'importe quel témoin de minimalité de la boule cible est la base d'un ensemble caractéristique qui fait converger l'Algo. 1 vers la cible, d'où le point (3).

5.3 Identification à partir d'INFORMATEUR

Théorème 8
*(1) $\mathcal{GB}(\Sigma)$ n'est pas identifiable à la limite à partir d'*INFORMATEUR *en temps* IPE *polynomial mais elle l'est (2) en temps* MC *polynomial et (3) en temps* CS *polynomial.*

Preuve : (1) Preuve similaire à celle de (Pitt, 1989) pour les AFD : si $\mathcal{GB}(\Sigma)$ était identifiable en temps IPE polynomial à partir d'INFORMATEUR, alors $\mathcal{GB}(\Sigma)$ serait polynomialement identifiable à partir de EQ, ce qui contredirait le théorème 4. (2) Comme les hypothèses n'ont pas besoin d'être consistantes avec les données, on peut utiliser l'algorithme identifiant $\mathcal{GB}(\Sigma)$ à partir de TEXTE en ignorant les exemples négatifs. (3) Mêmes ensembles caractéristiques que pour le théorème 6 (3). □

Concernant les AFD, on obtient :

Théorème 9
(1) $\mathcal{AFD}(\Sigma)$ n'est pas identifiable à la limite en temps IPE *polynomial à partir d'*INFORMATEUR *(Pitt, 1989) mais elle l'est (2) en temps* MC *polynomial et (3) en temps* CS *polynomial (Gold, 1978; Oncina & García, 1992).*

Pour montrer le deuxième point, nous réutilisons la notion de témoin de minimalité. On dit que $X = \langle X_+, X_- \rangle$ est un témoin pour l'AFD $A = \langle \Sigma, Q, q_0, F, \delta \rangle$ si (1) tous les états et toutes les transitions sont exercés par au moins une chaîne de X, (2) il existe un mot de X pour chaque état indiquant si celui-ci est final ou non, (3) pour chaque lettre a, pour chaque transition $\delta(q, a) = q' \neq q''$, il existe des mots w, w', f tels que w (*resp.* w') soit le plus petit mot (de $\Sigma^\star$, au sens de l'ordre hiérarchique) tel que $\delta(q_0, w) = q$ (*resp.* $\delta(q_0, w') = q'$) et $waf \in X_+, w'af \in X_-$ ou $waf \in X_-, w'af \in X_+$ rendant la fusion de q et q'' impossible.

Si X est un témoin pour A, alors il n'existe pas d'autres AFD de taille inférieure ou égale à $\|A\|$ compatible avec cet ensemble. En effet, toute transition absente est rendue impossible par le témoin et chaque état est rendu nécessaire par les couples de mots incompatibles deux à deux. De plus, X est un ensemble caractéristique de A pour l'algorithme RPNI (Oncina & García, 1992; Dupont & Miclet, 1998). Aussi, A est *l'unique* AFD de taille minimale compatible avec X.

On définit maintenant un algorithme identifiant $\mathcal{AFD}(\Sigma)$ en temps MC polynomial. Soit $\mathfrak{A}$ l'algorithme qui comme première hypothèse émet l'hypothèse vide. $\mathfrak{A}$ change ensuite d'avis uniquement si $\boldsymbol{f}_i$ est un témoin de minimalité pour l'AFD A_i retourné par RPNI sur $\boldsymbol{f}_i$. Clairement, cette vérification se fait en temps polynomial en $\|\boldsymbol{f}_i\|$. Si $\boldsymbol{f}_i$ n'est pas un témoin pour A_i, $\mathfrak{A}$ retourne l'hypothèse précédente (qui n'est pas forcément consistante).

Chaque fois que $\mathfrak{A}$ change d'avis, le nombre d'états de l'AFD retourné augmente strictement (grâce au témoin de minimalité). Comme il est borné par le nombre d'états de l'AFD cible, le nombre de changement d'avis est polynomial en la taille de l'AFD cible. Enfin, il arrivera forcément un moment où le témoin de minimalité de l'AFD cible apparaîtra, et alors $\mathfrak{A}$ convergera. $\mathfrak{A}$ identifie donc bien $\mathcal{AFD}(\Sigma)$ à la limite en temps MC polynomial.

6 Le cas des mauvaises boules

Dans cette section dont la justification est essentiellement technique, nous étudions l'apprenabilité des boules générales. Rappelons qu'une boule $B_r(o)$ est bonne lorsque $r \leq |o|$. Aussi, lorsqu'une boule n'est pas bonne, il est possible que $r \gg 2^{|o|}$. Mais alors, les plus longs mots de $B_r(o)$, de taille $|o| + r$, qui délimitent la frontière extérieure de $B_r(o)$, ne sont plus liés polynomialement à la taille des boules ($|o| + \log r$). Nous nous trouvons donc dans une situation similaire à celles des *automates non déterministes*, où il faut parfois considérer des mots de taille exponentielle pour distinguer deux états (de la Higuera, 1997; Denis *et al.*, 2004). Dans cette section, nous nous intéressons donc aux limites des paradigmes d'apprentissage, quand un algorithme ne peut plus échapper aux données de longueur exponentielle.

Sans surprise, la plupart des résultats sont négatifs :

Théorème 10
La classe $\mathcal{BALL}(\Sigma)$ n'est pas (1) polynomialement PAC *apprenable à partir de* PRÉSENTATION*, (2) polynomialement identifiable à partir de* MQ *et* EQ*, (3) polynomialement identifiable à partir de* CQ_{EDIT}*, (4) identifiable à la limite en temps* MC *polynomial à partir de* TEXTE*, (5) identifiable à la limite en temps* IPE *polynomial à partir de* PRÉSENTATION*, (6) identifiable à la limite en temps* CS *polynomial à partir de* PRÉSENTATION*.*

Preuve : (1) et (2) viennent des résultats sur $\mathcal{GB}(\Sigma)$, de même que (5) dans le cas des INFORMATEUR. (3) On ne peut pas apprendre $B_n(\lambda)$ sans faire une requête hors de la boule, ce qui impliquerait un mot de longueur exponentielle (en $\log(n)$). (6) Les ensembles caractéristiques de $B_n(\lambda)$ et $B_{n+1}(\lambda)$ ne sont pas différents sur les mots de longueur polynomiale (en $\log n$).

Concernant (4), supposons que nous ayons un apprenant $\mathfrak{A}$ et un polynôme $p()$ tels que $\forall G \in \mathcal{G}, \forall \boldsymbol{f} \in \textbf{Pres}, \#\text{MC}(\boldsymbol{f}) \leq p(\|G\|)$. Soit n un entier suffisamment grand, et considérons la sous-classe cible $B_k(\lambda)$ avec $k \leq n$. Pour chaque cible, on construit une présentation $\boldsymbol{f}^k$ en utilisant $\mathfrak{A}$ de façon interactive. À chaque étape i, $\mathfrak{A}$ produit une hypothèse H_i, et nous devons calculer un nouveau mot $\boldsymbol{f}^k(i+1)$. Pour cela, si $i = 0$ on retourne λ. Sinon, il y a deux cas. (i) Si $H_{i-1} = B_k(\lambda)$, alors on retourne $\min(B_k(\lambda) - \boldsymbol{f}^k_{i-1})$ si $B_k(\lambda) - \boldsymbol{f}^k_{i-1} \neq \emptyset$, et λ sinon. (ii) Si $H_{i-1} \neq B_k(\lambda)$, alors on retourne a^{j+1} si $H_{i-1} = B_j(\lambda)$ avec $j = \max\{|u| : u \in \boldsymbol{f}^k_{i-1}\}$, et λ sinon.

Chaque présentation $\boldsymbol{f}^k$ est alors une présentation (TEXTE) correcte de la cible $B_k(\lambda)$, *i.e.*, $\boldsymbol{f}^k(\mathbb{N}) = B_k(\lambda)$. Posons $m(k) = \min\{i \in \mathbb{N} : \boldsymbol{f}^k(i) = a^k\}$. Pour chaque k, $\boldsymbol{f}^k$ et $\boldsymbol{f}^{k+1}$ coïncident sur les mêmes $m(k)$ valeurs initiales. Alors $\boldsymbol{f}^n$ peut être réécrit en : $\lambda, \ldots, \lambda, a, \ldots, a^j, \ldots, a^n, \ldots$ avec : $\forall 0 < j \leq n, \forall i \in \{m(j-1), .., m(j) - 1\}, \boldsymbol{f}^n(i) = \boldsymbol{f}^j(m(j-1)) = \boldsymbol{f}^j(i) = a^{j-1}$, et $\mathfrak{A}$ change d'avis juste avant de recevoir le nouvel exemple a^i. Cela prouve que pour tout polynôme $p()$, $\exists n \in \mathbb{N}$ tel que $\#\text{MC}(\boldsymbol{f}^n) > p(\log n)$. De la même façon, on peut montrer que $\#\text{IPE}(\boldsymbol{f}^n) > p(\log n)$. □

On peut malgré tout montrer le résultat positif suivant :

TAB. 1 – Vue générale des différents théorèmes présentés. † signale les théorèmes prouvés dans le papier et ‡ indique les corollaires.

Critères	$\mathcal{GB}(\Sigma)$		$\mathcal{AFD}(\Sigma)$		$\mathcal{BALL}(\Sigma)$	
PAC INFORM.	NON†	Theo. 1	NON	Theo. 2	NON‡	Theo. 10 (1)
PAC TEXTE	NON†	Theo. 3 (1)	NON†	Theo. 3 (2)	NON‡	Theo. 10 (1)
IPE INFORM.	NON†	Theo. 8 (1)	NON	Theo. 9 (1)	NON‡	Theo. 10 (5)
IPE TEXTE	OUI†	Theo. 6 (2)	NON	Theo. 7	NON†	Theo. 10 (5)
MC INFORM.	OUI†	Theo. 8 (2)	OUI†	Theo. 9 (2)	OUI†	Theo. 11
MC TEXTE	OUI†	Theo. 6 (1)	NON	Theo. 7	NON†	Theo. 10 (4)
CS INFORM.	OUI†	Theo. 8 (3)	OUI	Theo. 9 (3)	NON†	Theo. 10 (6)
CS TEXTE	OUI†	Theo. 6 (3)	NON	Theo. 7	NON†	Theo. 10 (6)
MQ (ou EQ)	NON	Theo. 4 (1)	NON	Theo. 5 (2,3)	NON‡	Theo. 10 (2)
MQ et EQ	NON	Theo. 4 (1)	OUI	Theo. 5 (1)	NON‡	Theo. 10 (2)
CQ_{EDIT}	OUI	Theo. 4 (2)	NON	Theo. 5 (4)	NON‡	Theo. 10 (3)

Théorème 11
La classe $\mathcal{BALL}(\Sigma)$ est identifiable en temps MC *polynomial à partir d'* INFORMATEUR.

Preuve : Nous montrons qu'il existe un apprenant qui vérifie les données jusqu'à être sûr qu'il n'existe qu'une boule consistante avec les données et fait donc un unique changement d'avis. Soient $B_r(o)$ la boule cible et $\langle X_+, X_- \rangle$ des exemples tels qu'il existe un mot u pour lequel (1) $a^k u, b^k u \in X_+$, (2) tous les sur-mots de $a^k u$ et de $b^k u$ de longueur $|u| + 1 + k$ sont dans X_- et (3) si $u \neq \lambda$, pour chaque sous-mot v de u de longueur $|u| - 1$, il existe un sur-mot de v de longueur $|u| + k$ dans X_-. Notons que (1) étant donné une boule $B_r(o)$, ces exemples existent toujours ; (2) vérifier si un tel mot est dans X est en $\mathcal{O}(\|X\|)$; (3) toutes les opérations d'édition dans un chemin minimal pour transformer o en $a^k u$ et $b^k u$ sont des insertions : sinon en changeant l'opération qui ne serait pas une insertion par une insertion, on peut construire un mot w tel que $d(o, w) = d(o, a^k u) \wedge w \in X_-$; Ainsi, $o \preceq u$; (4) puisque pour chaque sous-mot w de u il existe un sur-mot de longueur $|u| + k$ dans X_-, aucun sous-mot propre de u n'est le centre. On en déduit que $u = o$ et $k = r$. Évidemment, les conditions requises seront vraies à un certain moment de la présentation. □

7 Conclusion

Nous résumons l'ensemble de nos résultats dans le tableau 1.

Dans un paysage plutôt confus, nous avons fait l'étude systématique de deux classes de langages dont les définitions reposent sur des principes très différents. Clairement, le propos de l'article n'était pas de montrer qu'un paradigme est équivalent ou meilleur qu'un autre : comparer deux classes ne suffit pas.

En revanche, nous avons montré qu'un certain nombre de croyances ne tenaient pas. Par exemple, il est faux de penser que l'identification en temps MC polynomial im-

plique celle en temps IPE polynomial. Il est également faux de penser qu'il est plus simple d'apprendre avec des exemples positifs et négatifs, qu'avec des exemples positifs seulement. Au delà des raisons techniques propres aux paradigmes d'apprentissage, c'est d'abord parce que le complémentaire d'une boule n'est pas une boule (de même que le complémentaire d'un langage hors-contexte n'est pas hors-contexte).

Concernant le paradigme PAC, nous n'obtenons aucun résultat positif pour les boules, résultats déjà connus pour les AFD. Cette difficulté est en fait liée aux distributions utilisées. Sous certaines restrictions, on peut obtenir des résultats positifs. Ainsi, $\mathcal{AFD}(\Sigma)$ est polynomialement PAC apprenable à partir d'*exemples simples* (Denis, 2001).

Pour finir, il est remarquable de noter que les boules de mots ne sont pas apprenables avec un nombre polynomial de MQ et EQ, alors que les AFD le sont. Or comme ce sont des langages finis, les boules sont reconnaissables avec des AFD. Ainsi, une sous classe d'une classe apprenable pourrait ne pas être apprenable (!) À celà, il y a deux explications possibles. D'une part, on peut penser que l'AFD minimal reconnaissant une boule est de taille exponentielle en la taille de la boule ; cette question divise les auteurs. . . D'autre part, il est possible qu'à partir d'un AFD reconnaissant une boule, extraire le centre et le rayon de cette dernière ne puisse pas se faire en temps polynomial en la taille de l'AFD. Cette question divise également les auteurs. . .

Remerciements : Les auteurs remercient les reviewers pour les problèmes qu'ils ont relevés dans la version soumise, et la justesse de leurs questions.

Références

ANGLUIN D. (1987a). Learning regular sets from queries and counterexamples. *Information and Control*, **39**, 337–350.

ANGLUIN D. (1987b). Queries and concept learning. *Machine Learning Journal*, **2**, 319–342.

ANGLUIN D. (1990). Negative results for equivalence queries. *Machine Learning Journal*, **5**, 121–150.

ANGLUIN D. & SMITH C. (1983). Inductive inference : theory and methods. *ACM computing surveys*, **15**(3), 237–269.

BECERRA-BONACHE L., DE LA HIGUERA C., JANODET J.-C. & TANTINI F. (2007). Learning balls of strings with correction queries. In *European Conference on Machine Learning (ECML'07)*, p. 18–29. (also in CAp'07).

CROCHEMORE M., HANCART C. & LECROQ T. (2001). *Algorithmique du Texte*. Vuibert.

DE LA HIGUERA C. (1997). Characteristic sets for polynomial grammatical inference. *Machine Learning Journal*, **27**, 125–138.

DE LA HIGUERA C. & CASACUBERTA F. (2000). Topology of strings : Median string is NP-complete. *Theoretical Computer Science*, **230**, 39–48.

DENIS F. (2001). Learning regular languages from simple positive examples. *Machine Learning Journal*, **44**(1), 37–66.

DENIS F., LEMAY A. & TERLUTTE A. (2004). Learning regular languages using rfsas. *Theoritical Computer Science*, **313**(2), 267–294.

DUPONT P. & MICLET L. (1998). *Inférence grammaticale régulière : fondements théoriques et principaux algorithmes*. Rapport interne, IRISA.

GAREY M. R. & JOHNSON D. S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman.

GOLD E. M. (1967). Language identification in the limit. *Information and Control*, **10**(5), 447–474.

GOLD E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, **37**, 302–320.

GREENBERG R. I. (2002). *Fast and Simple Computation of All Longest Common Subsequences*. Rapport interne, Loyola University. http ://arXiv.org/abs/cs.DS/0211001.

GREENBERG R. I. (2003). *Bounds on the Number of Longest Common Subsequences*. Rapport interne, Loyola University. http ://arXiv.org/abs/cs/0301030v2.

HAUSSLER D., KEARNS M. J., LITTLESTONE N. & WARMUTH M. K. (1991). Equivalence of models for polynomial learnability. *Information and Computation*, **95**(2), 129–161.

KEARNS M. & VALIANT L. (1989). Cryptographic limitations on learning boolean formulae and finite automata. In *21st ACM Symposium on Theory of Computing*, p. 433–444.

KEARNS M. & VAZIRANI U. (1994). *An Introduction to Computational Learning Theory*. MIT press.

LEVENSHTEIN V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, **163**(4), 845–848.

MAIER D. (1977). The complexity of some problems on subsequences and supersequences. *Journal of the ACM*, **25**, 322–336.

ONCINA J. & GARCÍA P. (1992). Identifying regular languages in polynomial time. In H. BUNKE, Ed., *Advances in Structural and Syntactic Pattern Recognition*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, p. 99–108. World Scientific.

PAPADIMITRIOU C. M. (1994). *Computational Complexity*. New York : Addison–Wesley.

PAREKH R. J. & HONAVAR V. (2000). On the relationship between models for learning in helpful environments. In *Proc. of Grammatical Inference : Algorithms and Applications (ICGI'00)*, p. 207–220 : LNAI 1891.

PITT L. (1989). Inductive inference, DFA's, and computational complexity. In *Analogical and Inductive Inference*, number 397 in LNAI, p. 18–44. Springer-Verlag.

PITT L. & VALIANT L. G. (1988). Computational limitations on learning from examples. *Journal of the ACM*, **35**(4), 965–984.

PITT L. & WARMUTH M. (1993). The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM*, **40**(1), 95–142.

TANTINI F., DE LA HIGUERA C. & JANODET J.-C. (2006). Identification in the limit of systematic-noisy languages. In *Proc. of Grammatical Inference : Algorithms and Applications (ICGI'06)*, p. 19–31 : LNAI 4201. (also in CAp'06).

VALIANT L. G. (1984). A theory of the learnable. *Communications of the ACM*, **27**(11), 1134–1142.

WAGNER R. & FISHER M. (1974). The string-to-string correction problem. *Journal of the ACM*, **21**, 168–178.

WARMUTH M. (1989). Towards representation independence in pac-learning. In *Proc. International Workshop on Analogical and Inductive Inference (AII'89)*, p. 78–103 : LNAI 397.