Apprentissage spectral non négatif de systèmes séquentiels linéaires

Hadrien Glaude[1,2], Cyrille Enderli[1], et Olivier Pietquin[2,3]

[1]Thales Systèmes Aéroportés
[2]Univ. Lille, CRIStAL, UMR 9189, 59650, équipe SequeL, Villeneuve d'Ascq, France
[3]Institut Universitaire de France

30 avril 2015

Résumé

La méthode des moments (MoM) est devenue récemment une alternative intéressante aux approches itératives standards comme Expectation Maximization (EM) pour apprendre des modèles à variables latentes. Les algorithmes issues de la MoM viennent avec des garanties de convergence vers l'optimum sous la forme de bornes de concentration en échantillons finis. Toutefois, avec du temps de calcul et en utilisant heuristiques pour éviter les optima locaux, les approches itératives obtiennent souvent de meilleures performances. Nous pensons que cet écart de performance est en partie dû au fait que les algorithmes basés sur la MoM peuvent apprendre des modèles générant des probabilités négatives. En limitant l'espace de recherche, nous proposons un algorithme spectral non-négatif (NNSpectral) en évitant de fait l'apprentissage de probabilités négatives. NNSpectral est comparé à d'autres algorithmes basés sur la MoM et EM sur des problèmes artificiels du défi PAutomaC. Non seulement, NNSpectral surpasse les autres algorithmes basés sur la MoM, mais aussi, d'obtenir des résultats très compétitifs par rapport à EM.

1 Introduction

Traditionnellement, les distributions de probabilité complexes sur des données structurées sont apprises grâce à des modèles génératifs avec variables latentes comme les Modèles de Markov Cachés (HMM) ou les Processus Décisionnels Markoviens Partiellement Observables (POMDP). Une approche largement utilisée pour ajuster un modèle aux observations recueillies est de maximiser la vraisemblance. C'est équivalent à minimiser la divergence *kl* entre la distribution apprise et l'empirique. Cependant, pour la plupart des modèles probabilistes, la divergence *kl* est non convexe et peut être difficile à minimiser. A cause de ces difficultés, les algorithmes standards, tels que EM et la descente de gradient, sont conçus pour être des procédures itératives qui convergent vers des minima locaux. En plus d'être sujets à se coincer dans un optimum local, ces algorithmes sont coûteux en calcul, au point où l'obtention de bonnes solutions pour les grands modèles devient intraitable.

Une récente alternative consiste à concevoir des algorithmes d'apprentissage en exploitant la MoM. Le succès de cette méthode repose sur le fait que les moments d'ordre inférieur d'une distribution sont généralement faciles à estimer. Ensuite, en exprimant les paramètres de la distribution recherchée en fonction des moments, on peut retrouver par des opérations algébriques une approximation de la distribution cible. La MoM a plusieurs avantages par rapport aux méthodes itératives. Tout d'abord, elle peut fournir des algorithmes d'apprentissage plus rapides. En effet, les moments peuvent être estimés en temps linéaire avec le nombre d'échantillons. Ensuite, les paramètres de la distribution peuvent être calculés en un temps polynomial en la dimension des moments mais indépendant de la taille de l'ensemble d'apprentissage. Enfin, ces algorithmes sont généralement construits pour minimiser une distance euclidienne avec des contraintes convexes [BQC12]. Cette convexité conduit généralement à des algorithmes avec des garanties théoriques en échantillons finis.

Un autre avantage de la MoM est la grande variété de modèles qui peuvent être appris [AGH+12, BHD10]. Dans un article récent, [TJar] ont ainsi montré que de nombreux modèles font partie du cadre commun des Systèmes Linéaires Séquentiels (SS) qui sont équivalents aux Automates à Multiplicité (MA). Les SS

linéaires représentent des fonctions réelles sur des séquences de symboles $f_{\mathcal{M}} : \Sigma^\star \to \mathbb{R}$, où $\Sigma^\star$ est l'ensemble des mots de taille finis fait à partir des symboles de Σ. En particulier, ces fonctions peuvent être utilisées pour représenter des distributions de probabilité, où $f_{\mathcal{M}}(x) = p_{\mathcal{M}}(x) \in [0, 1]$ tels que les langages rationnels stochastiques. Dans cet article, pour des raisons de clarté, nous nous concentrons sur l'apprentissage des langages rationnels stochastiques. Néanmoins, notre travail s'étend à d'autres modèles, comme expliqué dans la Section 4.

Au-delà de tous les aspects attrayants de la MoM, un problème bien connu est celui des probabilités négatives. Celles-ci proviennent du fait que la plupart des algorithmes basés sur la MoM ne contraignent pas la fonction apprise à être une distribution. Dans certaines applications nécessitant de véritables probabilités, c'est problématique. Par exemple, planifier avec des représentations à états prédictifs (PSR) approchées et qui ne renvoient pas des probabilités est un problème ouvert [LYJ14]. Heureusement, pour d'autres applications, une approximation d'une distribution de probabilité est suffisante. Par exemple, pour calculer un maximum a posteriori (MAP), ne pas avoir de distribution n'induit pas d'erreur tant que le maximum peut toujours être correctement identifié. Cependant, nous allons montrer expérimentalement que contraindre la fonction apprise à sortir des valeurs positives aide le processus d'apprentissage et fournit de meilleurs résultats, même pour calculer le MAP. Un second problème est l'écart de performance observé avec des algorithmes itératifs. Bien qu'ils aient généralement besoin d'être relancés plusieurs fois, avec suffisamment de temps pour explorer l'espace des paramètres, ils finissent par produire de bon résultats. Récemment, une comparaison empirique [BHP14] a montré que les algorithmes basés sur la MoM ont de plus faibles performances que EM.

Dans cet article, nous proposons un nouvel algorithme basé sur la MoM, appelé apprentissage spectral non-négatif (NNSpectral). Inspiré par les résultats théoriques sur les MAs définis sur des semi-anneaux commutatifs positifs, NNSpectral utilise la factorisation non-négative de matrices (NMF) et les moindres carrés non-négatifs (NNLS). Cela permet de contraindre la fonction apprise à être non-négative. Finalement, de véritables probabilités peuvent être récupérées après renormalisation. Une évaluation sur les douze mêmes problèmes synthétiques du défi PAutomaC utilisés dans [BHP14] permet une comparaison objective de NNSpectral à trois grands algorithmes basés sur la MoM et à EM. Non seulement, NNSpectral surpasse les algorithmes basés sur la MoM, mais c'est aussi la première fois à notre connaissance qu'un algorithme basé sur la MoM obtient des résultats compétitifs par rapport à EM.

2 Fondements

2.1 Définitions

Notre objectif est d'apprendre, à partir de séquences de symboles, la distribution qui les a générées. Nous nous limitons à l'étude des distributions qui peuvent être représentées par différents modèles génératifs comme les HMMs et les POMDPs. Ces modèles sont compris dans la théorie des MAs et des langages rationnels. Plus de précisions sont données dans la Section 4. Les preuves et plus de détails sur les notions présentées dans cette partie peuvent être trouvés dans [DE08, TJar].

Soit Σ un ensemble de symboles, aussi appelé un alphabet. On note $\Sigma^\star$, l'ensemble de tous les mots finis sur les symboles de Σ, y compris le mot vide noté ε. L'ensemble des mots de longueur k est noté Σ^k. Soit u et $v \in \Sigma^\star$, uv est la concaténation des deux mots et $u\Sigma^\star$ est l'ensemble des mots finis ayant comme préfixe u. Dans la suite, K est un semi-anneau commutatif. Plus particulièrement, on prendra $K = \mathbb{R}$ ou $K = \mathbb{R}^+$.

Définition (Série formelle)**.** *Une série formelle est une application f de $\Sigma^\star$ dans K.*

L'ensemble des séries formelles est un semi-module [Gut07] sur le semi-anneau K. Cette propriété est utile pour travailler avec la représentation en matrice de Hankel dans la section suivante. Pour un ensemble de mots $\mathcal{S}$, nous notons $f(\mathcal{S}) = \sum_{u \in \mathcal{S}} f(u)$. Certaines séries formelles peut être représentées par des modèles compacts, appelés MA.

Définition (Automate à multiplicité)**.** *Soit K un semi-ring, un automate à multiplicité définit sur K (K-MA) est un structure $\langle \Sigma, Q, \{A_\sigma\}_{\sigma \in \Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$, où Σ est un alphabet et Q est un ensemble fini d'états. On note $n = |Q|$ le nombre d'états. Les matrices $A_\sigma \in K^{n \times n}$ contiennent les poids de transition et les vecteurs $\boldsymbol{\alpha}_\infty \in K^n$ and $\boldsymbol{\alpha}_0 \in K^n$ contiennent respectivement les poids finaux et terminaux.*

Un K-MA $\mathcal{M}$ définit un langage rationnel,

$$r_{\mathcal{M}}(u) = r_{\mathcal{M}}(\sigma_1 \dots \sigma_k) = \boldsymbol{\alpha}_0^\top A_u \boldsymbol{\alpha}_0 = \boldsymbol{\alpha}_0^\top A_{\sigma_1} \dots A_{\sigma_k} \boldsymbol{\alpha}_\infty.$$

Une fonction f est dîte *réalisée* par un K-MA $\mathcal{M}$, si $r_{\mathcal{M}} = f$.

Définition (Langage rationnel). *Une série formelle r est rationnelle sur K si et seulement si elle est réalisée par un K-MA.*

Deux K-MA réalisant le même langage rationnel sont *équivalents*. Un K-MA est minimal s'il n'existe pas de K-MA équivalent ayant strictement moins d'états. Il existe plusieurs K-MA minimaux équivalents et peuvent se retrouver par changement de base. Plus précisément, soit un K-MA $\mathcal{M} = \langle \Sigma, Q, \{A_\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$ de dimension n et $R \in K^{n\times n}$ une matrice inversible, alors

$$\mathcal{M}' = \left\langle \Sigma, Q, \left\{R^{-1}A_\sigma R\right\}_{\sigma\in\Sigma}, R^\top \boldsymbol{\alpha}_0, R^{-1}\boldsymbol{\alpha}_\infty \right\rangle,$$

$\mathcal{M}'$ est équivalent à $\mathcal{M}$ car $r_{\mathcal{M}'} = r_{\mathcal{M}}$.

On suppose que les données d'apprentissage sont générés par des langages rationnels stochastiques.

Définition (Langage rationnel stochastique). *Un langage rationnel stochastique p est un langage rationnel à valeur dans $\mathbb{R}^+$ et tel que $\sum_{u\in\Sigma^\star} p(u) = 1$.*

Les langages rationnels stochastiques représentent des distributions sur les mots de $\Sigma^\star$. De plus, les langages rationnels stochastiques peuvent être réalisés par une sous classe des $\mathbb{R}$-MAs décrite par les automate à multiplicité stochastique.

Définition (Automate à Multiplicité Stochastique). *Un Automate à Multiplicité Stochastique (SMA) $\mathcal{M}$ est un $\mathbb{R}$-MA tel que $p_\mathcal{M} : \Sigma^\star \to [0,1]$ et $p_\mathcal{M}(\Sigma^\star) = 1$. Ainsi, $p_\mathcal{M}$ est un langage rationnel stochastique.*

Ainsi, pour apprendre un modèle génératif compact à partir de mots générés par un langage stochastique rationnel, on va chercher le MA minimal dont la série est proche de celle empirique au sens des matrices de Hankel, présentées dans la prochaine partie.

2.2 Matrice de Hankel

Pour toute série formelle f sur $\Sigma^\star$ à valeur dans K, on peut définir $H_f \in K^{\Sigma^\star\times\Sigma^\star}$ la matrice bi-infinie, appelé matrice de Hankel, dont les lignes et les colonnes sont indexés par des mots, vérifiant $H_f[u,v] = f(uv)$.

$$H_f = \begin{array}{c} \\ \varepsilon \\ a \\ b \\ aa \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} \varepsilon & a & b & aa & \dots \end{array} \\ \begin{pmatrix} f(\varepsilon) & f(a) & f(b) & f(aa) & \dots \\ f(a) & f(aa) & f(ab) & f(aaa) & \dots \\ f(b) & f(ba) & f(bb) & f(baa) & \dots \\ f(aa) & f(aaa) & f(aab) & f(aaaa) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array}$$

Si p est un langage stochastique, alors H_p contient des probabilités et peut être estimée par comptages. Plus de détails sur les matrices définies sur des semi-anneaux peuvent être trouvés dans [Gut07]. Lorsque la série formelle peut être déduite du contexte, nous notons H sa matrice de Hankel. Soit pour tous $\sigma \in \Sigma$, on pose $H_\sigma \in K^{\Sigma^\star\times\Sigma^\star}$ et $\mathbf{h}_S \in K^{\Sigma^\star}$, $\mathbf{h}_P \in K^{\Sigma^\star}$ tels que $H_\sigma(u,v) = f(u\sigma v)$, $\mathbf{h}_S(u) = \mathbf{h}_P(u) = f(u)$. Dans le cas d'une matrice de Hankel bi-infinie, ces vecteurs et ces matrices peuvent être extraits à partir de H. Cette représentation de la série réside dans le cœur de tous les algorithmes d'apprentissage basés sur la MoM à cause du théorème fondamental suivant. Notez que la preuve originale du théorème suppose que K est un corps mais reste vrai quand K est un semi-anneau commutatif, car rangs, déterminants et inverses sont bien définis dans un semi-module.

Théorème 1 ([CP71]). *Soit une série formelle r réalisée par un K-MA à n états, alors* $\operatorname{rank}(H_r) \leq n$. *Réciproquement, si une matrice de Hankel H_r issue d'une série formelle r à un rang fini égal à n sur le semi-anneau K, alors r est rationnelle sur K et peut être réalisée par un K-MA minimal avec exactement n états.*

La preuve du Théorème 1 montre aussi comment construire un K-MA depuis H. En effet, en partant d'un K-MA à n états, on a $H[u,v] = (\boldsymbol{\alpha}_0^\top A_u)(A_v\boldsymbol{\alpha}_\infty)$. Posons $P \in K^{\Sigma^\star\times n}$ et $S \in K^{n\times\Sigma^\star}$ des matrices définies par,

$$P = ((\boldsymbol{\alpha}_0^\top A_u)^\top)_{u\in\Sigma^\star}^\top,$$
$$S = (A_v\boldsymbol{\alpha}_\infty)_{v\in\Sigma^\star},$$

alors $H = PS$. Notons que de cette égalité, on a clairement que $\operatorname{rank}(H) \leq n$. De plus, on a que,

$$H_\sigma = PA_\sigma S, \quad \mathbf{h}_S^\top = \boldsymbol{\alpha}_0^\top S, \quad \mathbf{h}_P = P\boldsymbol{\alpha}_\infty.$$

Ainsi, d'une matrice de Hankel de rang n on peut retrouver les paramètres d'un K-MA par,

$$A_\sigma = P^\dagger H_\sigma S^\dagger, \quad \boldsymbol{\alpha}_0^\top = \mathbf{h}_S^\top S^\dagger, \quad \boldsymbol{\alpha}_\infty = P^\dagger \mathbf{h}_P, \quad (1)$$

où $X^\dagger$ est la pseudo inverse de X.

Heureusement, on a pas besoin de calculer la matrice de Hankel bi-infinie H pour retrouver un K-MA réalisant la série associée à H. En effet, étant donné une base de préfixes et de suffixes $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, on note $H_\mathcal{B}$ le sous-bloc correspondant de H. On dit qu'une base est *complète* si H et $H_\mathcal{B}$ ont le même rang. Ainsi, soit $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ une base complète, en définissant P sur $\mathcal{P}$, S sur $\mathcal{S}$, H et H_σ sur $\mathcal{B}$, $\mathbf{h}_P$ sur $\mathcal{P}$, et $\mathbf{h}_S$ sur $\mathcal{S}$, on peut retrouver un K-MA en utilisant (1) [Bal13].

2.3 Algorithme Spectral

Cette section présente l'algorithme Spectral basé sur la MoM pour apprendre un langage p rationnel sur $\mathbb{R}$ et stochastique à partir d'un ensemble de séquences de symboles. La factorisation de la matrice de Hankel et l'invariance de tout changement de base a motivé l'algorithme Spectral, qui utilise la décomposition en valeurs singulières (SVD) tronquée pour récupérer une approximation de rang faible de $\hat{H}$ et sa factorisation $\hat{H} = PS$, où $P = UD$ et $S = V^\top$. Le choix de rang dans l'algorithme est une question importante, comme expliqué dans [KRS14, GPE14], mais ne sera pas discuté ici.

Algorithme Spectral pour les $\mathbb{R}$-MA

Entrées Un alphabet Σ, un ensemble de trajectoires d'apprentissage, un rang estimé n

Sortie Un $\mathbb{R}$-MA $\mathcal{M} = \langle \Sigma, \{1..n\}, \{A_\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$

1: Choisir une base de préfixes $\mathcal{P} \subset \Sigma^\star$ et de suffixes $\mathcal{S} \subset \Sigma^\star$
2: Estimer $\hat{H}$, $\mathbf{h}_S$, $\mathbf{h}_P$ et $\hat{H}_\sigma$ à partir des trajectoires d'apprentissage.
3: $U, D, V \leftarrow \mathcal{SVD}_n(\hat{H})$
4: Pour tous $\sigma \in \Sigma$ faire $A_\sigma \leftarrow D^{-1}U^\top \hat{H}_\sigma V = (\hat{H}V)^\dagger \hat{H}_\sigma V$
5: $\boldsymbol{\alpha}_0^\top \leftarrow \hat{\mathbf{h}}_S^\top V$
6: $\boldsymbol{\alpha}_\infty \leftarrow D^{-1}U^\top \hat{\mathbf{h}}_P = (\hat{H}V)^\dagger \hat{\mathbf{h}}_P$

On remarque que si $\varepsilon \in \mathcal{P}$ alors la colonne de $\hat{H}$ correspondant à l'historique vide vaut $\hat{\mathbf{h}}_P$. Sans nuire à la généralité, on va supposer qu'il s'agit de la première colonne. Dans ce cas $\boldsymbol{\alpha}_0^\top$ est la première ligne de P, que l'on note $P[0,:]$. De même, si $\varepsilon \in \mathcal{S}$ alors, sans nuire à la généralité, $\hat{\mathbf{h}}_S^\top$ est la première ligne de $\hat{H}$ et $\boldsymbol{\alpha}_\infty$ est la première colonne de S, noté $S[:,0]$. Un des principaux inconvénients de l' algorithme Spectral est qu'il produit un $\mathbb{R}$ -MA $\mathcal{M}$ sans aucune garantie que le langage rationnel réalisé par $\mathcal{M}$ soit stochastique. Ainsi, $r_\mathcal{M}$ peut produire probabilités négatives. Lorsque de vraies probabilités sont nécessaires, il faut compter sur des heuristiques comme par exemple tronquer la sortie à $[0, +\infty]$ et renormaliser. Ce genre d'heuristique peut introduire des erreurs dans les prévisions.

3 Algorithme NNSpectral

3.1 Apprendre un $\mathbb{R}^+$-MA

L'algorithme NNSpectral suit la même démarche induite par le théorème 1, mais appliquée aux MA définis sur le semi-anneau commutatif $\mathbb{R}^+$ au lieu de $\mathbb{R}$. Étant donné que le théorème 1 reste vrai pour les semi-anneaux commutatif, alors il existe une factorisation de faible rang de la matrice de Hankel représentant un $\mathbb{R}^+$-MA avec des facteurs non négatifs. Trouver une telle décomposition est un problème bien connu, appelé Factorisation Non-négative de Matrices (NMF), qui a reçu beaucoup d'intérêt au cours de la dernière décennie. Ces algorithmes et leurs propriétés seront décrites dans la section suivante. A partir de la factorisation, on peut retrouver les paramètres du $\mathbb{R}^+$-MA, en utilisant (1). Cependant, comme pour l'algorithme Spectral, lors de l'apprentissage $\hat{H}_\sigma$, $\hat{\mathbf{h}}_S$, $\hat{\mathbf{h}}_P$ sont bruités et les équations (1) ne sont plus exactes. Les résoudre approximativement au sens des moindres carrées, comme précédemment, peut faire apparaître des poids négatifs. A la place, nous proposons de récupérer des poids non négatifs en résolvant un problème de moindres carrés non négatifs (NNLS). Comme $\boldsymbol{\alpha}_0$ et $\boldsymbol{\alpha}_\infty$ peuvent être directement extrais de la factorisation, comme dans l'algorithme Spectral, nous choisissons d'inclure ε dans la base de préfixes et suffixes au lieu de résoudre deux problèmes de NNLS supplémentaires. Finalement, on a bien que l' algorithme NNSpectral retourne un $\mathbb{R}^+$-MA réalisant un langage rationnel sur $\mathbb{R}^+$.

Algorithme NNSpectral pour les $\mathbb{R}^+$-MA

Entrées An alphabet Σ, a training set, a targeted rank n

Sortie A $\mathbb{R}^+$-MA $\mathcal{M} = \langle \Sigma, \{1..n\}, \{A_\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$

1: Choose a set of prefixes $\mathcal{P} \subset \Sigma^\star$ and suffixes $\mathcal{S} \subset \Sigma^\star$ both containing ε
2: Estimate $\hat{H}$, $\mathbf{h}_S$, $\mathbf{h}_P$ and $\hat{H}_\sigma$ from the training set.
3: $P, S \leftarrow \operatorname{argmin}_{\substack{P\in\mathbb{R}^{|\mathcal{P}|\times n} \\ S\in\mathbb{R}^{n\times|\mathcal{S}|} \\ P\geq 0, S\geq 0}} \|\hat{H} - PS\|_F$
4: For all $\sigma \in \Sigma$ do $B_\sigma \leftarrow \operatorname{argmin}_{X\geq 0} \|PX - \hat{H}_\sigma\|_F$
5: For all $\sigma \in \Sigma$ do $A_\sigma \leftarrow \operatorname{argmin}_{X\geq 0} \|XS - B_\sigma\|_F$
6: $\boldsymbol{\alpha}_0^\top \leftarrow P[0,:]$
7: $\boldsymbol{\alpha}_\infty \leftarrow S[:,0]$

3.2 Factorisation Non-Négative

La NMF est devenu un outil largement utilisé pour la réduire la dimension des attributs non-négatifs. Il cherche à décomposer une matrice de non-négative $X \in \mathbb{R}^{n\times m}$ tel que $X \approx WH$ où $W \in \mathbb{R}^{n\times r}$, $W \geq 0$ et $H \in \mathbb{R}^{r\times m}$, $H \geq 0$ (la positivité est évaluée par composante). A la différence de la SVD, la NMF est connu pour extraire des attributs parcimonieux et significatifs. Ces propriétés ont prouvé leur utilité dans le traitement d'image pour l'extraction d'attribut du

visage [LS99] et dans l'exploration de texte pour la classification des documents [XLG03]. Pour la NMF, plusieurs fonctions de perte pour ont été envisagées. Nous nous intéressons au problème suivant,

$$\min_{W\in\mathbb{R}^{n\times r},H\in\mathbb{R}^{r\times m}} \|X-WH\|_F \text{ such that } W\geq 0, H\geq 0,$$

où la norme de Frobenius $\|X\|_F = \sqrt{\sum_{i,j} x_{ij}^2}$ est appropriée quand X contient un bruit Gaussien. Malheureusement, la NMF est un problème NP-dur en général [Vav09] et la décomposition non unique en fait un problème mal posé. Ainsi, en pratique, on utilise des heuristiques. Souvent, ces heuristiques ont seulement la garantie de converger vers un point stationnaire. La plupart d'entre eux s'exécute en $\mathcal{O}(nmr)$ pas de temps. Le lecteur peut se référer à [Gil14] pour un comparatif de ces méthodes. Dans nos expérience, les moindres carrés non-négatifs alterné avec projection du gradient de [Lin07] ont montré être un bon compromis entre qualité de la solution et performance. Cet algorithme optimise de façon alternée W et H en résolvant à chaque étape un problème de NNLS.

3.3 Moindres Carrés Non-Négatifs

Les moindres carrés non négatifs (NNLS) sont une version contrainte du problème des moindres carrés où les coefficients ne sont pas autorisés à devenir négatif.

$$\operatorname*{argmin}_{\mathbf{X}} \|\mathbf{AX}-\mathbf{B}\|_F \text{ such that } X\geq 0$$

Étant équivalent à un problème de programmation quadratique, le problème est convexe et les contraintes forment un ensemble convexe, ce qui garantie la convergence en temps polynomial vers un point stationnaire. De plus, si A est une matrice de rang plein, le problème est fortement convexe et le seul point fixe est l'optimum. Comme dans les sous-problèmes de la NMF, nous utilisons un algorithme de gradient projeté pour trouver la solution.

3.4 Comparaison à Spectral

Nous avons montré que l'algorithme Spectral renvoie un $\mathbb{R}$-MA alors que l'algorithme NNSpectral retourne un $\mathbb{R}^+$-MA. Par définition, ces modèles ne sont pas stochastiques. Plus précisément, les SMA sont inclus dans $\mathbb{R}$-MA mais pas dans les $\mathbb{R}^+$-MA. Dans cette partie, on précise ces relations.

Dans [DE08], l'auteur montre que vérifier si un $\mathbb{R}$-MA est stochastique est un problème indécidable et donc que les $\mathbb{R}$-MA sont généralisent strictement les SMA. En fait, deux conditions sont nécessaires pour qu'un langage rationnel r soit stochastique. Tout d'abord, $r_{\mathcal{M}}(\Sigma^\star) = \sum_k r_{\mathcal{M}}(\Sigma^k)$ doit converger vers 1. Soit $A = \sum_{\sigma\in\Sigma} A_\sigma$, si son rayon spectral $\rho(A) < 1$, alors la somme converge vers $\boldsymbol{\alpha}_0^\top (I-A)^{-1}\boldsymbol{\alpha}_\infty$. Cette condition peut être vérifié en temps polynomial. Un MA vérifiant cette condition est dit *convergent*. La deuxième condition est la non-négativité de r. Malheureusement, cette condition est indécidable par réduction au problème d'arrêt. Une façon de contourner cette limitation est de restreindre le langage à être rationnel sur $\mathbb{R}^+$. C'est ce que l'algorithme NNSpectral fait. Mais cela a un prix : la non-négativité des poids est plus contraignante que de considérer les $\mathbb{R}$-MA dont la série est positive. En effet, il y a des SMA, et ainsi des $\mathbb{R}$-MA, réalisant des langages rationnels qui ne peuvent pas être réalisés par les $\mathbb{R}^+$-MA avec un nombre fini d'états. Donc, les $\mathbb{R}^+$-MA ne contiennent pas la classe des SMA. Cependant, leur intersection n'est pas vide et contient les $\mathbb{R}^+$-MA stochastique. Toujours dans [DE08] l'auteur montre que les Automates Probabilistes Finis (PFA) et les $\mathbb{R}^+$-MA stochastique définissent la même classe de langage. En fait, tout $\mathbb{R}^+$-MA stochastique peuvent être convertis en temps polynomial en un PFA équivalent par renormalisation.

Définition (Automate Probabiliste Fini)**.** *Un Automate Probabiliste Fini (PFA)* $\mathcal{M} = \langle \Sigma, Q, \{A_\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty\rangle$ *est un SMA avec des poids non-négatifs vérifiant* $\mathbf{1}^\top\boldsymbol{\alpha}_0 = 1$, $\boldsymbol{\alpha}_\infty + \sum_{\sigma\in\Sigma} A_\sigma \mathbf{1} = \mathbf{1}$.

Les poids des PFA sont dans $[0,1]$ et peuvent être considérés comme des probabilités sur les transitions, les états initiaux et les états terminaux. Parce que les $\mathbb{R}^+$-MA assure la non négativité de r et parce que vérifier si un $\mathbb{R}^+$-MA est convergent peut être fait en temps polynomial, alors décider si un $\mathbb{R}^+$-MA est stochastique peut être fait en temps polynomial. Finalement, on peut normaliser le langage rationnel réalisé par un $\mathbb{R}^+$-MA convergent pour obtenir un langage stochastique rationnel sur $\mathbb{R}^+$. Ainsi, un PFA équivalent peut être construit à partir de ce langage en temps polynomial. Malheureusement, l'algorithme NNSpectral ne garantit pas la convergence de la série réalisée par le modèle retourné. Cependant, il est encore possible d'obtenir un PFA approximativement équivalent en supposant la convergence comme l'a fait [GDH14]. Dans leur document, ils donnent une procédure pour transformer un $\mathbb{R}$-MA renvoyée par l'algorithme Spectral en un PFA approximativement équivalent avec un nombre accru d'états dans le but d'initialiser un algorithme EM. Ils procèdent en deux

étapes. Tout d'abord, ils recherchent une représentation équivalent sur une base de préfixes où la somme des poids négatifs est minimisée. Ensuite, les valeurs absolues des coefficients de pondération sont prises et le modèle est normalisé. Cela a permis d'améliorer les résultats par rapport aux algorithmes EM et Spectral dans la plupart des cas. Un manque d'amélioration a été signalé lorsque la somme des poids négatifs après la première étape était encore élevée. Ainsi, suivre une procédure équivalente après l'algorithme NNSpectral pourrait mener à des performances accrues, mais est hors de la portée de ce document.

4 Travaux Similaires

Dans la littérature, l'algorithme Spectral a de nombreuses variantes. Ici, nous détaillons seulement deux autres algorithmes basés sur la MoM qui diffèrent sensiblement de l'algorithme Spectral. Ces trois algorithmes seront utilisées pour la comparaison avec l'algorithme NNSpectral dans les expériences. Le premier, appelé CO [BQC12], reformule le problème de l'apprentissage en une tâche de régression de faible rang. Ainsi, il attaque en même temps le problème de factorisation et celui de régression linéaire pour retrouver les paramètres du $\mathbb{R}$-MA. Le problème de régression de faible rang cherche à minimiser le rang de la matrice des coefficients de régression en contraignant en norme de Frobenius les résidus à être faible. Ce problème est NP-dur, mais l'auteur relâche la contrainte sur le rang en utilisant la norme nucléaire. Ainsi, leur méthode récupère les poids d'un $\mathbb{R}$-MA en résolvant un problème d'optimisation convexe impliquant la norme des résidus et une régularisation en norme nucléaire. Comme pour l'algorithme Spectral, la méthode de CO est consistante et vient avec une analyse en échantillons finis, cependant le $\mathbb{R}$-MA retourné n'est garantie d'être stochastique. Contrairement à l'algorithme spectrale, la dimension des paramètres du modèle appris sont celles de la base utilisée pour estimer la matrice de Hankel. Cela rend le calcul de l'inférence plus coûteux.

L'autre méthode, décrite dans [AGH+12], utilise des tenseurs et est appelée Tensor. Elle permet d'apprendre des $\mathbb{R}$-MA sous une forme particulière appelée Automates Factorisés à Multiplicité.

Définition (Automate Factorisé à Multiplicité)**.** *Un Automate Factorisé à Multiplicité (FMA) est une structure* $\mathcal{M} = \langle \Sigma, Q, T, \{O_\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$*, où* Σ *est un alphabet,* Q *un ensemble de* n *états,* $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \in \mathbb{R}^n$ *sont des vecteurs contenant les poids initiaux et finaux,* $T \in \mathbb{R}^{n\times n}$ *sont des matrices contenant les poids de transitions et* $O_\sigma \in \mathbb{R}^{n\times n}$ *sont des matrices diagonales contenant les poids d'observations.*

Un FMA $\mathcal{M} = \langle \Sigma, Q, T, \{O_\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$ peut être convertit en un $\mathbb{R}$-MA $\mathcal{M}' = \langle \Sigma, Q, \{A_\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \rangle$ en prenant $A_\sigma = O_\sigma T$. L'inverse est aussi vrai en prenant $T = I$, où I est l'identité et $A_\sigma = O_\sigma$. En ajoutant des contraintes donnant une interprétation probabiliste aux poids, cette forme particulière se réduit à celle des HMMs, où la matrice des probabilités d'observations contient en colonne les poids diagonaux des matrices O_σ et la matrice des probabilités de transitions est donné par T (cf. [AGH+12]). L'avantage d'apprendre une forme factorisée réside dans la possibilité ensuite de projeter les poids appris, qui ne sont pas nécessairement des probabilités, sur un simplexe de façon à retrouver un HMM. Ce qui peut servir ensuite à initialiser un algorithme EM. Malheureusement, expérimentalement, cette projection dégrade fortement les performances [BHP14].

Bien que l'algorithme Tensor ait été originalement conçu pour apprendre des HMMs, celui-ci s'étend naturellement à l'apprentissage de $\mathbb{R}$-MA en autorisant des bases de préfixes et de suffixes de longueur supérieure à 1. Rappelons que pour un K-MA, nous avons $H = PS$ et $H_\sigma = PA_\sigma S$. Une propriété importante des FMA est que nous pouvons écrire $H_\sigma = PO_\sigma TS$. Comme les matrices d'observations sont diagonales, les matrices de Hankel H_σ admettent une diagonalisation commune. L'algorithme Tensor utilise cette propriété en empilant ces matrices le long de la dimension Σ dans un tenseur avec une structure particulière. Après symétrisation et blanchiment du bruit, ce tenseur peut être décomposé en une somme de tenseurs orthogonaux super-symétriques de rang 1. De cette somme les paramètres du FMA peuvent être récupérées, si certaines hypothèses sur le rang de matrices d'observation sont vérifiés et si $|\Sigma| \geq n$.

D'autres travaux ont étudié les relations entre les différents modèles englobés par les SSs linéaires. Un langage rationnel stochastique est une distributions sur l'ensemble des mots de longueur finie et peuvent être réalisé par un SMA. Les SMA généralisent les PFA qui imposent la positivité des poids. D'autre part, un processus stochastique définit des distributions sur des ensembles de mots de même longueur. Ces processus peuvent être réalisé par les modèles à opérateurs observables (OOMs) qui généralise de la même façon les HMMs. Langages rationnels stochastiques et processus stochastiques sont en fait assez similaire. Plus de détails sur l'équivalence entre OOMs et SMA peut être

trouvés dans [TJar]. Dans [DDE05], l'auteur expliquent comment HMM et PFA sont liés. De plus, les SSs linéaire englobe les systèmes à entrées-sorties, comme les transducteurs stochastique à états finis, en utilisant un alphabet $\Sigma = \Sigma_I \times \Sigma_O$ composé de paires de symboles d'entrées et de sorties. Ces modèles génératifs ont été utilisés en reconnaissance de la parole [MPR08] ou en traduction automatique [dGIB+10]. Ces systèmes à entrées-sorties peuvent aussi décrire des processus contrôlés où les entrées sont des actions et les sorties sont des observations. On parle alors de PSRs ou de façon équivalent de OOMs à entrées et sorties (IO-OOM) qui généralisent les POMDPs.

Enfin, on mentionne deux autres travaux plus confidentiels qui se sont aussi intéressés aux problèmes des probabilités négatives en proposant aussi d'apprendre des modèles plus contraints, appelé Norm-OOM [ZJ10] et Automate Pondéré Fini Quadratique (QWA) [Bai11]. L'algorithme proposé pour apprendre un Norm-OOM est basé sur une procédure itérative qui maximise la vraisemblance comme EM. Celui pour apprendre les QWA effectue une SVD comme l'algorithme Spectral. Malheureusement, les QWA et les norm-HMM ne sont pas capable de modéliser tous les HMMs. C'est pourquoi nous ne les avons pas utilisé en comparaison dans les expériences.

5 Expériences

5.1 Critère d'Évaluation

La qualité d'un modèle peut être mesurée par la qualité de la distribution de probabilité qu'il réalise. L'objectif est alors d'apprendre un automate réalisant une série p proche de la distribution, notée $p_\star$, qui a généré l'ensemble d'apprentissage, noté $\mathcal{T}$. La qualité de p est mesurée par la perplexité qui correspond au nombre de bits moyen nécessaire pour représenter un mot en utilisant le code optimal donné par $p_\star$.

$$\text{Perplexité}(\mathcal{M}) = 2^{-\sum_{u\in\mathcal{T}} p_\star(u)\log(p_\mathcal{M}(u))}$$

La qualité d'un modèle peut aussi être évaluée par sa capacité à prédire le prochain symbole dans un mot. Le critère que l'on utilise est alors le taux d'erreurs de symboles, communément appelé WER quand les symboles sont des mots du langage naturel. Pour chaque mot, le symbole prédit à chaque pas de temps est le plus probable étant donné les symboles précédents.

5.2 Implémentation

Dans cette partie, nous discutons des détails pratiques de l'algorithme NNSpectral. Premièrement, il existe de nombreuses approches pour choisir une base de préfixes et de suffixes en espérant quelle soit complète. Par exemple, on peut prendre tous les préfixes et suffixes qui apparaissent dans l'ensemble d'entraînement. Lorsque l'ensemble d'entraînement est trop grand, on peut limiter la taille de la base en limitant la longueur maximale des préfixes et suffixes. Cependant, utiliser des préfixes et des suffixes courts peut conduire à une perte d'information sur la dynamique en milieu de séquences. Une autre possibilité est de prendre seulement les préfixes et les suffixes les plus fréquents, car leur probabilité d'occurrence sera estimée plus précisément. Mais cela peut conduire à l'utilisation de préfixes et de suffixes trop courts car beaucoup plus fréquents que les longs. Dans ce cas, il peut être préférable de travailler avec d'autres séries dérivées de p et qui peut être mieux estimées puis de retrouver p. On donne deux exemples de séries dérivées : la série basée sur les préfixes et définie par $p^{prefixe}(u) = p(u\Sigma^\star) = \sum_{v\in\Sigma^\star} p(uv)$ et la série basée sur les sous-chaîne $p^{sous}(u) = \sum_{w,v\in\Sigma^\star} p(wuv)$. En effet, dans [Bal13], l'auteur montre que lorsque p est réalisé par un SMA, alors $p^{prefixe}$ et p^{sous} le sont. De plus, il fournit une conversion explicite qui préserve le nombre d'états entre les SMA qui réalisent les séries dérivés et la série originale. Par conséquent, pour l'apprentissage, on peut travailler avec l'une de ces trois représentations. Toutefois, l'exactitude des conversions suppose la convergence de la série, ce qui est pas toujours le cas pour le modèle appris. Ensuite, en suivant les conseils dans [CSC+13], nous avons normalisé la variance des probabilités estimées en multipliant les lignes et les colonnes de $\hat{H}$ par un facteur $c_u = \sqrt{|\mathcal{S}|/(\#(u)+5)}$, où $\#(u)$ est le nombre d'occurrences de u dans l'ensemble de la formation. Cela améliore un peu les performances. Enfin, les hyperparamètres optimaux tel que le rang ont été trouvé à l'aide d'une grille de recherche. Pour chaque problème, les données d'entrainement sont divisés en deux ensembles. Trois-quarts sert à apprendre des modèles avec différents hyper-paramètres. Le quart restant sert à sélectionner les hyper-paramètres optimaux. Puis, toutes les données sont utilisés pour réapprendre un modèle avec le bon jeu de paramètre.

5.3 Compétition PAutomaC

La compétition PAutomaC [VEdlH12], proposée dans la conférence ICGI 2012, propose de résoudre un ensemble de 48 problèmes dont le but est d'apprendre un modèle génératif de séquences de symboles, celles-ci ayant été générées par différent types d'automates. Ces modèles peuvent être de trois types : PFA, HMM et PDFA. Un Automate Probabiliste Déterministe Fini (PDFA) est un PFA avec des transitions déterministes. Nous avons sélectionné les douze mêmes problèmes que dans [BHP14], c'est à dire quatre modèles de chaque type. Une description détaillée de chaque problème peut être trouvée dans [VEdlH12]. La Tableau 1 donne la perplexité et le WER du modèle appris par l'algorithme NNSpectral en utilisant les chaînes où les sous chaînes pour estimer la matrice de Hankel. Les Tableaux 2 et 3 comparent les meilleurs résultats de NNSpectral à ceux de EM et au meilleur des autres algorithmes basés sur la MoM : CO en utilisant des chaînes, Tensor en utilisant des chaînes et spectrale en utilisant des chaînes et des sous chaînes. Pour chaque problème, le meilleur algorithme est indiqué par un score en gras. Le meilleur score entre NNSpectral et les autres algorithmes basés sur la MoM sont soulignés. Le score du vrai modèle est aussi donné. Pour la perplexité, NNSpectral surpasse les autres algorithmes basés sur la MoM sur les douze problèmes et fait mieux que EM pour sept problèmes. Pour les cinq autres problèmes, NNSpectral réalise des performances très proches de EM et de celles du vrai modèle. Pour la métrique WER, NNSpectral bat les autres algorithmes basés sur la MoM sur dix problèmes et est le meilleur de tous sur sept problèmes. La performance de NNSpectral est moins marquée pour le WER car, le MAP peut être bien évalué sans avoir de vraies probabilités. Nous avons observé sur la tableau 2 que, pour tous les problèmes, NNSpectral produit des perplexité proches de celles optimales, alors que, sur les cinq problèmes (1, 45, 27, 42, 43) EM produit des solutions aberrantes.

6 Conclusion

Dans cet article, nous avons proposé l'algorithme NNSpectral inspiré par les développements théoriques des MA définies sur les semi-anneaux commutatifs. L'algorithme NNSpectral fonctionne en contraignant l'espace de recherche pour faire assurer la non-négativité du langage rationnelle. Comme d'autres algorithmes basés sur la MoM, l'algorithme NNSpectral est en mesure de traiter des problèmes de tailles importantes beaucoup plus rapidement que EM. Malheureusement, la consistance est perdue en raison de la NMF qui est résolue par heuristique. Expérimentalement, cela ne semble pas être un problème majeur car NNSpectral surpasse les autres algorithmes basés la MoM et se mesure favorablement à l'algorithme EM, qui produit parfois des solutions aberrantes. Ainsi, algorithme NNSpectral est une bonne alternative à l'algorithme EM avec un coût de calcul beaucoup plus faible. Dans les travaux futurs, nous aimerions explorer davantage les liens établis dans cette article avec la NMF, pour la quelle de nombreuses variantes avec d'intéressantes propriétés ont été étudiées. Par exemple, la NMF peut être étendue aux tenseurs pour proposer une version non-négative de l'algorithme Tensor ou travailler avec noyaux pour gérer des espaces d'actions et d'observations continus.

Références

[AGH+12] Anima Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *arXiv preprint arXiv :1210.7559*, 2012.

[Bai11] Raphael Bailly. Quadratic weighted automata : Spectral algorithm and likelihood maximization. *Journal of Machine Learning Research*, 20 :147–162, 2011.

[Bal13] Borja Balle. *Learning finite-state machines : statistical and algorithmic aspects*. PhD thesis, Universitat Politècnica de Catalunya, 2013.

[BHD10] Raphaël Bailly, Amaury Habrard, and François Denis. A spectral approach for probabilistic grammatical inference on trees. In *Proc of ALT-10*, pages 74–88. Springer, 2010.

[BHP14] Borja Balle, William Hamilton, and Joelle Pineau. Methods of moments for learning stochastic languages : Unified presentation and empirical comparison. In *Proc. of ICML-14*, pages 1386–1394, 2014.

[BQC12] Borja Balle, Ariadna Quattoni, and Xavier Carreras. Local loss optimization in operator models : A new insight into spectral learning. In *Proc. of ICML-12*, 2012.

[BSG11] Byron Boots, Sajid M Siddiqi, and Geoffrey J Gordon. Closing the learning-planning loop with predictive state repre-

sentations. *The International Journal of Robotics Research*, 30(7) :954–966, 2011.

[CP71] Jack W. Carlyle and Azaria Paz. Realizations by stochastic finite automata. *Journal of Computer and System Sciences*, 5(1) :26 – 40, 1971.

[CSC+13] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle H Ungar. Experiments with spectral learning of latent-variable pcfgs. In *Proc of HLT-NAACL-13*, pages 148–157, 2013.

[DDE05] Pierre Dupont, François Denis, and Yann Esposito. Links between probabilistic automata and hidden markov models : probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38(9) :1349 – 1371, 2005. Grammatical Inference.

[DE08] François Denis and Yann Esposito. On rational stochastic languages. *Fundamenta Informaticae*, 86(1) :41–77, 2008.

[dGIB+10] Adrià de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R Banga, and William Byrne. Hierarchical phrase-based translation with weighted finite-state transducers and shallow-n grammars. *Computational linguistics*, 36(3) :505–533, 2010.

[GDH14] Mattias Gybels, François Denis, and Amaury Habrard. Some improvements of the spectral learning approach for probabilistic grammatical inference. In *Proc. of ICGI-12*, volume 34, pages 64–78, 2014.

[Gil14] Nicolas Gillis. The Why and How of Nonnegative Matrix Factorization. *ArXiv e-prints*, January 2014.

[GPE14] Hadrien Glaude, Olivier Pietquin, and Cyrille Enderli. Subspace identification for predictive state representation by nuclear norm minimization. In *Proc. of ADPRL-14*, 2014.

[Gut07] A. E. Guterman. Rank and determinant functions for matrices over semirings. In Nicholas Young and Yemon Choi, editors, *Surveys in Contemporary Mathematics*, pages 1–33. Cambridge University Press, 2007. Cambridge Books Online.

[KRS14] Alex Kulesza, N Raj Rao, and Satinder Singh. Low-rank spectral learning. In *Proc. of AISTATS-14*, pages 522–530, 2014.

[Lin07] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10) :2756–2779, 2007.

[LS99] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755) :788–791, 1999.

[LYJ14] Yunlong Liu, Zijiang Yang, and Guoli Ji. Solving partially observable problems with inaccurate psr models. *Information Sciences*, 283 :142–152, 2014.

[MPR08] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer, 2008.

[TJar] Michael Thon and Herbert Jaeger. Links between multiplicity automata, observable operator models and predictive state representations—a unified learning framework. *Journal of Machine Learning Research*, to appear.

[Vav09] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3) :1364–1377, 2009.

[VEdlH12] Sicco Verwer, Rémi Eyraud, and Colin de la Higuera. Results of the pautomac probabilistic automaton learning competition. *Journal of Machine Learning Research - Proc. Track*, 21 :243–248, 2012.

[WK08] Alex Wang and Vikram Krishnamurthy. Signal interpretation of multifunction radars : Modeling and statistical signal processing with stochastic context free grammar. *IEEE Transactions on Signal Processing*, 56(3) :1106–1119, 2008.

[XLG03] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proc. of ACM SIGIR-03*, pages 267–273. ACM, 2003.

[ZJ10] MingJie Zhao and Herbert Jaeger. Norm-observable operator models. *Neural Computation*, 22(7) :1927–1959, July 2010.

	Perplexité		WER	
ID	chaînes	sous-chaînes	chaînes	sous-chaînes
HMM 1	**30.54(64)**	65.48(5)	80.61(18)	**72.69(30)**
14	**117.22(8)**	116.98(11)	70.30(10)	**68.76(7)**
33	**32.21(14)**	32.42(5)	74.69(4)	**74.27(6)**
45	**24.08(2)**	25.61(2)	78.26(2)	**78.24(2)**
PDFA 6	107.31(35)	**76.99(28)**	59.16(34)	**47.05(21)**
7	$\sim 10^{14}$(54)	**51.26(12)**	96.78(39)	**48.41(42)**
27	**43.81(46)**	1432.12(60)	82.81(14)	**73.87(20)**
42	**16.12(20)**	16.39(8)	66.18(13)	**56.63(8)**
PFA 29	118.53(47)	**25.24(35)**	70.32(20)	**47.62(36)**
39	10.18(20)	**10.00(6)**	62.49(16)	**59.42(19)**
43	**32.85(7)**	35.08(2)	77.29(11)	**76.77(25)**
46	**12.28(44)**	54.74(38)	90.53(6)	**78.02(20)**

TABLE 1 – Performance de NNSpectral sur douze problèmes de PAutomaC. La taille des modèles est données entre parenthèses.

ID	NNSpectral	EM	Meilleur des MoM	Vrai modèle
HMM 1	**30.54**	500.10	44.77	29.90(63)
14	116.98	**116.84**	128.53	116.79(15)
33	32.21	**32.14**	49.22	31.87(13)
45	**24.08**	107.75	31.87	24.04(14)
PDFA 6	76.99	**67.32**	95.12	66.98(19)
7	**51.26**	51.27	62.74	51.22(12)
27	**43.81**	94.40	102.85	42.43(19)
42	**16.12**	168.52	23.91	16.00(6)
PFA 29	25.24	**25.09**	34.57	24.03(36)
39	**10.00**	10.43	11.24	10.00(6)
43	**32.85**	461.23	36.61	32.64(67)
46	12.28	**12.02**	25.28	11.98(19)

TABLE 2 – Comparaison avec d'autres algorithmes pour la perplexité. La taille des vrais modèles sont données entre parenthèses.

ID	NNSpectral	EM	Meilleur des MoM	Vrai modèle
HMM 1	72.7	75.7	**71.3**	68.8(63)
14	**68.8**	68.8	70.0	68.4(15)
33	**74.3**	74.3	76.7	74.1(13)
45	78.24	**78.1**	80.1	78.1(14)
PDFA 6	**47.1**	47.4	50.2	46.9(19)
7	48.41	**48.1**	50.6	48.3(12)
27	**73.9**	83.0	75.5	73.0(19)
42	**56.6**	58.1	61.4	56.6(6)
PFA 29	47.6	49.2	**47.3**	47.2(36)
39	**59.4**	63.3	62.0	59.3(6)
43	**76.8**	77.4	78.0	77.1(67)
46	78.0	**77.5**	79.4	77.3(19)

TABLE 3 – Comparaison avec d'autres algorithmes pour le WER. La taille des vrais modèles sont données entre parenthèses.