Unsupervised Spectral Learning of WCFG as Low-rank Matrix Completion

Raphaël Bailly[†] **Xavier Carreras**[†] **Franco M. Luque**[‡] **Ariadna Quattoni**[†]

[†] Universitat Politècnica de Catalunya
Barcelona, 08034
rbailly,carreras,aquattoni@lsi.upc.edu

[‡] Universidad Nacional de Córdoba and CONICET
X500HUA Córdoba, Argentina
francolq@famaf.unc.edu.ar

Abstract

We derive a spectral method for unsupervised learning of Weighted Context Free Grammars. We frame WCFG induction as finding a Hankel matrix that has low rank and is linearly constrained to represent a function computed by inside-outside recursions. The proposed algorithm picks the grammar that agrees with a sample and is the simplest with respect to the nuclear norm of the Hankel matrix.

1 Introduction

Weighted Context Free Grammars (WCFG) define an important class of languages. Their expressivity makes them good candidates for modeling a wide range of natural language phenomena. This expressivity comes at a cost: unsupervised learning of WCFG seems to be a particularly hard task. And while it is a well-studied problem, it is still to a great extent unsolved.

Several methods for unsupervised learning of WCFG have been proposed. Some rely on heuristics that are used to build incrementally an approximation of the unknown grammar (Adriaans et al., 2000; Van Zaanen, 2000; Tu and Honavar, 2008). Other methods are based on maximum likelihood estimation, searching for the grammar that has the largest posterior given the training corpus (Baker, 1979; Lari and Young, 1990; Pereira and Schabes, 1992; Klein and Manning, 2002). Several Bayesian inference approaches have also been proposed (Chen, 1995; Kurihara and Sato, 2006; Liang et al., 2007; Cohen et al., 2010). These approaches perform parameter estimation by exploiting Markov sampling techniques.

Recently, for the related problem of unsupervised dependency parsing, Gormley and Eisner (2013) proposed a new way of framing the max-likelihood estimation. In their formulation the problem is expressed as an integer quadratic program subject to non-linear constraints. They exploit techniques from mathematical programming to solve the resulting optimization.

In spirit, the work by Clark (2001; 2007) is probably the most similar to our approach since both approaches share an algebraic view of the problem. In his case the key idea is to work with an algebraic representation of a WCFG. The problem of recovering the constituents of the grammar is reduced to the problem of identifying its syntactic congruence.

In the last years, multiple spectral learning algorithms have been proposed for a wide range of models (Hsu et al., 2009; Bailly et al., 2009; Bailly et al., 2010; Balle et al., 2011; Luque et al., 2012; Cohen et al., 2012). Since the spectral approach provides a good thinking tool to reason about distributions over Σ^*, the question of whether they can be used for unsupervised learning of WCFG seems natural. Still, while spectral algorithms for unsupervised learning of languages can learn regular languages, tree languages and simple dependency grammars, the frontier to WCFG seems hard to reach.

In fact, the most recent theoretical results on spectral learning of WCFG do not seem to be very encouraging. Recently, Hsu et al. (2012) showed that the problem of recovering the joint distribution over PCFG derivations and their yields is not identifiable.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 624–635,
Seattle, Washington, USA, 18-21 October 2013. ©2013 Association for Computational Linguistics

Although, for some simple grammar subclasses (e.g. independent left and right children), identification in the weaker sense (over the yields of the grammar) implies strong identification (e.g. over joint distribution of yields and derivations). In their paper, they propose a spectral algorithm based on a generalization of the method of moments for these restricted subclasses.

Thus one open direction for spectral research consists on defining subclasses of context free languages that can be learned (in the strong sense) from observations of yields. Yet, an alternative research direction is to consider learnability in the weaker sense. In this paper we take the second road, and focus on the problem of approximating the distribution over yields generated by a WCFG.

Our main contribution is to present a spectral algorithm for unsupervised learning of WCFG. Following ideas from Balle et al. (2012), the algorithm is framed as a convex optimization where we search for a low-rank matrix satisfying two types of constraints: (1) Constraints derived from observable statistics over yields; and (2) Constraints derived from certain recurrence relations satisfied by a WCFG. Our derivations of the learning algorithm illustrate the main ingredients behind the spectral approach to learning functions over Σ^* which are: (1) to exploit the recurrence relations satisfied by the target family of functions and (2) provide algebraic formulations of these relations.

We alert the reader that although we are able to frame the problem as a convex optimization, the number of variables involved is quite large and prohibits a practical implementation of the method on a realistic scenario. The experiments we present should be regarded as examples designed to illustrate the behavior of the method. More research is needed to make the optimization more efficient, and we are optimistic that such improvements can be achieved by exploiting problem-specific properties of the optimization. Regardless of this, ours is a novel way of framing the grammatical inference problem.

The rest of the paper is organized as follows. Section 2 gives preliminaries on WCFG and the type of functions we will learn. Section 3 establishes that spectral methods can learn a WCFG from a Hankel matrix containing statistics about context-free cuts. Section 4 presents the unsupervised algorithm, where we formulate grammar induction as a low-rank optimization. Section 5 presents experiments, and finally we conclude the paper.

Notation Let Σ be an alphabet. We use σ to denote an arbitrary symbol in Σ. The set of all finite strings over Σ is denoted by $\Sigma^\star$, where we write λ for the empty string. We also use the set $\Sigma^+ = \Sigma^\star \setminus \{\lambda\}$.

We use bold letters to represent column vectors $\boldsymbol{v}$ and matrices $\boldsymbol{M}$. We use $\boldsymbol{I}_n$ to denote the n-dimensional identity matrix. We use $\boldsymbol{M}^+$ to denote the *Moore-Penrose pseudoinverse* of some matrix $\boldsymbol{M}$. $\boldsymbol{M} \otimes \boldsymbol{M}'$ is the Kronecker product between matrices $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{M}' \in \mathbb{R}^{p \times q}$ resulting in a matrix in $\mathbb{R}^{mp \times nq}$. The rest of notation will be given as needed.

2 Weighted Context Free Grammars

In this section we define Weighted Context Free Grammars (WCFG). We start with a classic definition and then describe an algebraic form of WCFG that will be used throughout the paper. We also describe the fundamental recursions in WCFG.

2.1 WCFG in Classic Form

A WCFG over Σ is a tuple $\bar{G} = \langle V, R, T, w_\star, w_T, w_R \rangle$ where

- V is the set of non-terminal symbols. We assume that $V = \{1, \ldots, n\}$ for some natural number n, and that $V \cap \Sigma = \emptyset$.
- R is a set of binary rules of the form $i \to j\ k$ where $i, j, k \in V$.
- T is a set of unary rules of the form $i \to \sigma$ where $i \in V$ and $\sigma \in \Sigma$.
- $w_\star : V \to \mathbb{R}$, with $w_\star(i)$ being the weight of starting a derivation with non-terminal i.
- $w_T : V \times \Sigma \to \mathbb{R}$, with $w_T(i \to \sigma)$ being the weight of rule rewriting i into σ.
- $w_R : V \times V \times V \to \mathbb{R}$, with $w_R(i \to j\ k)$ being the weight of rewriting i into $j\ k$.

A WCFG $\bar{G}$ computes a function $g_{\bar{G}} : \Sigma^+ \to \mathbb{R}$ defined as

$$g_{\bar{G}}(x) = \sum_{i \in V} w_\star(i) \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x) \quad , \qquad (1)$$

where we define the *inside* function $\bar{\beta}_{\bar{G}} : V \times \Sigma^+ \rightarrow \mathbb{R}$ recursively:

$$\bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} \sigma) = w_T(i \rightarrow \sigma) \tag{2}$$

$$\bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x) = \sum_{\substack{j,k \in V \\ x_1,x_2 \in \Sigma^+ \\ \text{s.t. } x=x_1x_2}} w_R(i \rightarrow j\ k) \bar{\beta}_{\bar{G}}(j \overset{\star}{\Rightarrow} x_1)\bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} x_2)\ , \tag{3}$$

where in the second case we assume $|x| > 1$. The inside function $\bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x)$ exploits the fundamental inside recursion in WCFG (Baker, 1979; Lari and Young, 1990). We will find useful to define the *outside* function $\bar{\alpha}_{\bar{G}} : \Sigma^\star \times V \times \Sigma^\star \rightarrow \mathbb{R}$ defined recursively as:

$$\bar{\alpha}_{\bar{G}}(\lambda; i; \lambda) = w_\star(i) \tag{4}$$

$$\begin{aligned} \bar{\alpha}_{\bar{G}}(x; i; y) = & \sum_{\substack{j,k \in V \\ x_1 \in \Sigma^\star, x_2 \in \Sigma^+ \\ \text{s.t. } x=x_1x_2}} w_R(j \rightarrow k\ i) \cdot \bar{\alpha}_{\bar{G}}(x_1; j; y) \cdot \bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} x_2) \\ & + \sum_{\substack{j,k \in V \\ y_1 \in \Sigma^+, y_2 \in \Sigma^\star \\ \text{s.t. } y=y_1y_2}} w_R(j \rightarrow i\ k) \cdot \bar{\alpha}_{\bar{G}}(x; j; y_2) \cdot \bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} y_1)\ , \end{aligned} \tag{5}$$

where in the second case we assume that either $x \neq \lambda$ or $y \neq \lambda$.

For $x, z \in \Sigma^\star$ and $y \in \Sigma^+$ we have that

$$\sum_{i \in V} \bar{\alpha}_{\bar{G}}(x; i; z) \cdot \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} y) \tag{6}$$

is the weight that the grammar $\bar{G}$ assigns to a string xyz that has a *cut* or bracketing around y. Technically, it corresponds to the sum of the weights of all derivations that have a constituent spanning y. In particular we have that

$$g_{\bar{G}}(x) = \sum_i \bar{\alpha}_{\bar{G}}(\lambda; i; \lambda) \cdot \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x)\ .$$

If x is a string of length m, and $x_{[t:t']}$ is the substring of x from positions t to t', it also happens that

$$g_{\bar{G}}(x) = \sum_i \bar{\alpha}_{\bar{G}}(x_{[1:t-1]}; i; x_{[t+1:m]}) \cdot \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x_{[t]}))$$

for any t between 1 and m.

In this paper we will make frequent use of inside and outside quantities. Notationally, for outsides the semi-colon between two strings, i.e. $x; z$, will simbolize a cut where we can insert an inside string y.

Finally, we note that Probabilistic Context Free Grammars (PCFG) are a special case of WCFG where: $w_\star(i)$ is the probability to start a derivation with non-terminal i; $w_R(i \rightarrow j\ k)$ is the conditional probability of rewriting nonterminal i into j and k; $w_T(i \rightarrow \sigma)$ is the probability of rewriting i into symbol σ; $\sum_i w_\star(i) = 1$; and for each $i \in V$, $\sum_{j,k} w_R(i \rightarrow j\ k) + \sum_\sigma w_T(i \rightarrow \sigma) = 1$. Under these conditions the function $g_{\bar{G}}$ is a probability distibution over Σ^+.

2.2 WCFG in Algebraic Form

We now define a WCFG in algebraic form. A Weighted Context Free Grammar (WCFG) over Σ with n states is a tuple $G = \langle \boldsymbol{\alpha}_\star, \{\boldsymbol{\beta}_\sigma\}, \boldsymbol{A} \rangle$ with:

- An initial vector $\boldsymbol{\alpha}_\star \in \mathbb{R}^n$.
- Terminal vectors $\boldsymbol{\beta}_\sigma \in \mathbb{R}^n$ for $\sigma \in \Sigma$.
- A bilinear operator $\boldsymbol{A} \in \mathbb{R}^{n \times n^2}$.

A WCFG G computes a function $g_G : \Sigma^\star \rightarrow \mathbb{R}$ defined as

$$g_G(x) = \boldsymbol{\alpha}_\star^\top \beta_G(x) \tag{7}$$

where the *inside* function $\beta_G : \Sigma^+ \rightarrow \mathbb{R}^n$ is

$$\beta_G(\sigma) = \boldsymbol{\beta}_\sigma \tag{8}$$

$$\beta_G(x) = \sum_{\substack{x_1,x_2 \in \Sigma^+ \\ x=x_1x_2}} \boldsymbol{A}(\beta_G(x_1) \otimes \beta_G(x_2)) \tag{9}$$

We will define the *outside* function $\alpha_G : \Sigma^\star \times \Sigma^\star \rightarrow \mathbb{R}^n$ as:

$$\alpha_G(\lambda; \lambda) = \boldsymbol{\alpha}_\star \tag{10}$$

$$\begin{aligned} \alpha_G(x; z)^\top = & \sum_{\substack{x_1 \in \Sigma^\star, x_2 \in \Sigma^+ \\ x=x_1x_2}} \alpha_G(x_1; z)^\top \boldsymbol{A}(\beta_G(x_2) \otimes \boldsymbol{I}_n) \\ & + \sum_{\substack{z_1 \in \Sigma^+, z_2 \in \Sigma^\star \\ z=z_1z_2}} \alpha_G(x; z_2)^\top \boldsymbol{A}(\boldsymbol{I}_n \otimes \beta_G(z_1)) \end{aligned} \tag{11}$$

For $x, z \in \Sigma^\star$ and $y \in \Sigma^+$ we have that

$$\alpha_G(x; z)^\top \beta_G(y) \tag{12}$$

is the weight that the grammar assigns to the string xyz with a cut around y. In particular, $g_G(x) = \alpha_G(\lambda; \lambda)^\top \beta_G(x)$.

Let us make clear that a WCFG is the same device in classic or algebraic forms. If $\bar{G} = \langle V, R, T, w_\star, w_T, w_R \rangle$ and $G = \langle \boldsymbol{\alpha}_\star, \{\boldsymbol{\beta}_\sigma\}, \boldsymbol{A} \rangle$, the mapping is:

$$w_\star(i) = \boldsymbol{\alpha}_\star(i) \tag{13}$$
$$w_T(i \to \sigma) = \boldsymbol{\beta}_\sigma[i] \tag{14}$$
$$w_R(i \to j\ k) = \boldsymbol{A}[i, j, k] \tag{15}$$
$$\bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x) = \beta_G(x)[i] \tag{16}$$
$$\bar{\alpha}_{\bar{G}}(x; i; z) = \alpha_G(x; z)[i] \tag{17}$$

See Section A.1 for a proof of Eq. 16 and 17.

3 WCFG and Hankel Matrices

In this section we describe Hankel matrices for WCFG. These matrices explicitly capture inside-outside recursions employed by WCFG functions, and are key to a derivation of a spectral learning algorithm that learns a grammar G using statistics of a training sample.

Let us define some sets. We say that $\mathcal{I}^1 = \Sigma^+$ is the set of *inside strings*. The set of *composed inside strings* $\mathcal{I}^2$ is the set of elements (x, x'), where $x, x' \in \Sigma^+$. Intuitively (x, x') represents two adjacent spans with an operation, i.e., it keeps the trace of the operation that composes x with x' and yields xx'. We will use the set $\mathcal{I} = \mathcal{I}^1 \cup \mathcal{I}^2$.

The set of *outside contexts* $\mathcal{O}$ is the set containing elements $\langle x; z \rangle$, where $x, z \in \Sigma^\star$. Intuitively, $\langle x; z \rangle$ represents a context where we can insert an inside element y in between x and z, yielding xyz.

Consider a function $f : \mathcal{O} \times \mathcal{I} \to \mathbb{R}$. The Hankel matrix of f is the bi-infinite matrix $\boldsymbol{H}_f \in \mathbb{R}^{\mathcal{O} \times \mathcal{I}}$ such that $\boldsymbol{H}_f(o, i) = f(o, i)$.

In practice we will work with finite sub-blocks of $\boldsymbol{H}_f$. To this end we will employ the notion of basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, where $\{\langle \lambda, \lambda \rangle\} \subseteq \mathcal{P} \subseteq \mathcal{O}$ is a set of outside contexts and $\Sigma \subseteq \mathcal{S} \subseteq \mathcal{I}^1$ is a set of inside strings. We will use $p = |\mathcal{P}|$ and $s = |\mathcal{S}|$. Furthermore, we define the *inside completion* of $\mathcal{S}$ as the set $\mathcal{S}^\dagger = \{(x, x') \mid x, x' \in \mathcal{S}\}$. Note that $\mathcal{S}^\dagger \subseteq \mathcal{I}^2$. We say that $\mathcal{B}^\dagger = (\mathcal{P}, \mathcal{S}^\dagger)$ is the inside completion of $\mathcal{B}$.

The sub-block of $\boldsymbol{H}_f$ defined by $\mathcal{B}$ is the $p \times s$ matrix $\boldsymbol{H}_\mathcal{B} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ with $\boldsymbol{H}_\mathcal{B}(o, i) = \boldsymbol{H}_f(o, i) = f(o, i)$. In addition to $\boldsymbol{H}_\mathcal{B}$, we are interested in these additional finite vectors and matrices:

- $\boldsymbol{h}_\star \in \mathbb{R}^\mathcal{S}$ is the s-dimensional vector with coordinates $\boldsymbol{h}_\star(x) = f(\langle \lambda, \lambda \rangle, x)$.
- $\boldsymbol{h}_\sigma \in \mathbb{R}^\mathcal{P}$ is the p-dimensional vector with coordinates $\boldsymbol{h}_\sigma(o) = f(o, \sigma)$.
- $\boldsymbol{H}_A \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}^\dagger}$ with $\boldsymbol{H}_A(o, (x_1, x_2)) = f(o, (x_1, x_2))$.

3.1 Hankel Factorizations

If f is computed by a WCFG G, then $\boldsymbol{H}_f$ has rank n factorization. To see this, consider the following matrices. First a matrix $\boldsymbol{S} \in \mathbb{R}^{n \times \mathcal{I}^1}$ of *inside* vectors for all strings, with column x taking value $\boldsymbol{S}_x = \beta_G(x)$. Then a matrix $\boldsymbol{P} \in \mathbb{R}^{\mathcal{O} \times n}$ of outside vectors for all contexts, with row $\langle x; z \rangle$ taking value $\boldsymbol{P}_{\langle x; z \rangle} = \alpha_G(x; z)$. It is easy to see that $\boldsymbol{H}_f = \boldsymbol{P}\boldsymbol{S}$, since $\boldsymbol{H}_f(\langle x; z \rangle, y) = \boldsymbol{P}_{\langle x; z \rangle} \boldsymbol{S}_y = \alpha_G(x; z)^\top \beta_G(y)$. Therefore $\boldsymbol{H}_f$ has rank n.

The same happens for sub-blocks. If $\boldsymbol{H}_\mathcal{B}$ is the sub-block associated with basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, then the sub-blocks $\boldsymbol{P}_\mathcal{B} \in \mathbb{R}^{\mathcal{P} \times n}$ and $\boldsymbol{S}_\mathcal{B} \in \mathbb{R}^{n \times \mathcal{S}}$ of $\boldsymbol{P}$ and $\boldsymbol{S}$ also accomplish that $\boldsymbol{H}_\mathcal{B} = \boldsymbol{P}_\mathcal{B} \boldsymbol{S}_B$. It also happens that

$$\boldsymbol{h}_\star^\top = \boldsymbol{\alpha}_\star^\top \boldsymbol{S}_\mathcal{B} \tag{18}$$
$$\boldsymbol{h}_\sigma = \boldsymbol{P}_\mathcal{B} \boldsymbol{\beta}_\sigma \tag{19}$$
$$\boldsymbol{H}_A = \boldsymbol{P}_\mathcal{B} \boldsymbol{A} (\boldsymbol{S}_\mathcal{B} \otimes \boldsymbol{S}_\mathcal{B}) \quad . \tag{20}$$

We say that a basis $\mathcal{B}$ is complete for f if $\mathrm{rank}(\boldsymbol{H}_\mathcal{B}) = \mathrm{rank}(\boldsymbol{H}_f)$. The following is a key result for spectral methods.

Lemma 1. *Let $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ be a complete basis of dimension n for a function f and let $\boldsymbol{H}_\mathcal{B} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ be the Hankel sub-block of f for $\mathcal{B}$. Let $\boldsymbol{h}_\star$, $\boldsymbol{h}_\sigma$ and $\boldsymbol{H}_A$ be the additional matrices for $\mathcal{B}$. If $\boldsymbol{H}_\mathcal{B} = \boldsymbol{P}\boldsymbol{S}$ is a rank n factorization, then the WCFG $G = \langle \boldsymbol{\alpha}_\star, \{\boldsymbol{\beta}_\sigma\}, \boldsymbol{A} \rangle$ with*

$$\boldsymbol{\alpha}_\star^\top = \boldsymbol{h}_\star^\top \boldsymbol{S}^+ \tag{21}$$
$$\boldsymbol{\beta}_\sigma = \boldsymbol{P}^+ \boldsymbol{h}_\sigma \tag{22}$$
$$\boldsymbol{A} = \boldsymbol{P}^+ \boldsymbol{H}_A (\boldsymbol{S} \otimes \boldsymbol{S})^+ \tag{23}$$

computes f.

See proof in Section A.2.

3.2 Supervised Spectral Learning of WCFG

The spectral learning method directly exploits Lemma 1. In a nutshell, the spectral method is:

1. Choose a complete basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ and a dimension n.
2. Use training data to compute estimates of the necessary Hankel matrices: $\boldsymbol{H}_{\mathcal{B}}$, $\boldsymbol{h}_{\star}$, $\boldsymbol{h}_{\sigma}$, $\boldsymbol{H}_{A}$.
3. Compute the SVD of $\boldsymbol{H}_{\mathcal{B}}$, $\boldsymbol{H}_{\mathcal{B}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^{\top}$.
4. Create a truncated rank n factorization of $\boldsymbol{H}_{\mathcal{B}}$ as $\boldsymbol{P}_n\boldsymbol{S}_n$, having $\boldsymbol{P}_n = \boldsymbol{U}_n\boldsymbol{\Lambda}_n$ and $\boldsymbol{S}_n = \boldsymbol{V}_n^{\top}$, where we only consider the top n singular values/vectors of $\boldsymbol{\Lambda}, \boldsymbol{U}, \boldsymbol{V}$.
5. Use Lemma 1 to compute G, using $\boldsymbol{P}_n$ and $\boldsymbol{S}_n$.

Because of Lemma 1, if $\mathcal{B}$ is complete and we have access to the true $\boldsymbol{H}_{\mathcal{B}}$, $\boldsymbol{h}_{\star}$, $\boldsymbol{h}_{\sigma}$, $\boldsymbol{H}_{A}$ of a WCFG target function g^*, then the algorithm will compute a G that exactly computes g^*. In practice, we only have access to empirical estimates of the Hankel matrices. In this case, there exist PAC-style sample complexity bounds that state that g_G will be a close approximation to g^* (Hsu et al., 2009; Bailly et al., 2009; Bailly et al., 2010).

The parameters of the algorithm are the basis and the dimension of the grammar n. One typically employs some validation strategy using held-out data. Empirically, the performance of these methods has been shown to be good, and similar to that of EM (Luque et al., 2012; Cohen et al., 2013). It is also important to mention that in the case that the target g^* is a probability distribution, the function g_G will be close to g^*, but it will only define a distribution *in the limit*: in practice it will not sum to one, and for some inputs it might return negative values. This is a practical difficulty of spectral methods, for example to apply evaluation metrics like perplexity which are only defined for distributions.

4 Unsupervised Learning of WCFG

In the previous section we have exposed that if we have access to estimates of a Hankel matrix of a WCFG G, we can recover G. However, the statistics in the Hankel require access to strings that have information about context-free cuts. We will assume that we only have access to statistics about plain strings of a distribution, i.e. $p(x)$, which we call *observations*. In this scenario, one natural idea is to search for a Hankel matrix that agrees with the observations. The method we present in this section frames this problem as a low-rank matrix optimization problem. We first characterize the space of solutions to our problem, i.e. Hankel matrices associated with WCFG that agree with observable statistics. Then we present the method.

4.1 Characterization of a WCFG Hankel

In this section we describe valid WCFG Hankel matrices using linear constraints.

We first describe an inside-outside basis that is an extension of the one in the previous section. Inside elements are the same, namely $\mathcal{I} = \mathcal{I}^1 \cup \mathcal{I}^2$, where $\mathcal{I}^1$ are strings (x) and $\mathcal{I}^2$ are composed strings (x, x'). The set of *outside contexts* $\mathcal{O}^1$ is the set containing elements $\langle x; z\rangle$, defined as before. The set of *composed outside contexts* has elements $\langle x, x'; z\rangle$, and $\langle x; z', z\rangle$, where $x, z \in \Sigma^\star$ and $x', z' \in \Sigma^+$. These outside contexts keep an operation open in one of the sides. For example, if we consider $\langle x; z', z\rangle$ and insert a string y, we obtain $x(y, z')z$, where we use (y, z') to explicitly denote a composed inside string. We will use $\mathcal{O} = \mathcal{O}^1 \cup \mathcal{O}^2$.

In this section, we will assume that $\mathcal{I}$ and $\mathcal{O}$ are finite and closed. By closed, we mean that:

- $(x) \in \mathcal{I} \Rightarrow (x_1, x_2) \in \mathcal{I}$ for $x = x_1x_2$
- $(x_1, x_2) \in \mathcal{I} \Rightarrow x_1 \in \mathcal{I}, x_2 \in \mathcal{I}$
- $\langle x; z\rangle \in \mathcal{O} \Rightarrow \langle x_1, x_2; z\rangle \in \mathcal{O}$ for $x = x_1x_2$
- $\langle x; z\rangle \in \mathcal{O} \Rightarrow \langle x; z_1, z_2\rangle \in \mathcal{O}$ for $z = z_1z_2$
- $\langle x_1, x_2; z\rangle \in \mathcal{O} \Rightarrow (x_2) \in \mathcal{I}$
- $\langle x; z_1, z_2\rangle \in \mathcal{O} \Rightarrow (z_1) \in \mathcal{I}$

We will consider a Hankel matrix $\boldsymbol{H} \in \mathbb{R}^{\mathcal{O}\times\mathcal{I}}$. Some entries of this matrix will correspond to *observable* quantities. Specifically, for any string $x \in \mathcal{I}^1$ for which we know $p(x)$ we can define the following *observable constraint*:

$$p(x) = \boldsymbol{H}(\langle\lambda; \lambda\rangle, (x)) \qquad (24)$$

The rest of entries of $\boldsymbol{H}$ correspond to a string with an inside-outside cut, and these are not observable. Our method will infer the values of these entries. The following constraints will ensure that the matrix $\boldsymbol{H}$ is a well defined Hankel matrix for WCFG:

- Hankel constraints: $\forall \langle x; z\rangle \in \mathcal{O}, (y_1, y_2) \in \mathcal{I}$

$$\begin{aligned} \boldsymbol{H}(\langle x; z\rangle, (y_1, y_2)) &= \boldsymbol{H}(\langle x, y_1; z\rangle, (y_2)) \\ &= \boldsymbol{H}(\langle x; y_2, z\rangle, (y_1)) \end{aligned} \quad (25)$$

- Inside constraints: $\forall\, o \in \mathcal{O}, (x) \in \mathcal{I}$

$$\boldsymbol{H}(o, (x)) = \sum_{x = x_1 x_2} \boldsymbol{H}(o, (x_1, x_2)) \quad (26)$$

- Outside constraints: $\forall \langle x; z\rangle \in \mathcal{O}, i \in \mathcal{I}$

$$\begin{aligned} \boldsymbol{H}(\langle x; z\rangle, i) &= \sum_{x = x_1 x_2} \boldsymbol{H}(\langle x_1, x_2; z\rangle, i) \\ &+ \sum_{z = z_1 z_2} \boldsymbol{H}(\langle x; z_1, z_2\rangle, i) \end{aligned} \quad (27)$$

Constraint (25) states that composition operations that result in the same structure should have the same value. Constraints (26) and (27) ensure that the values in the Hankel follow the inside-outside recursions that define the computations of a WCFG function. The following lemma formalizes this concept. Let $\boldsymbol{H}_\varepsilon$ be the sub-block of $\boldsymbol{H}$ restricted to $\mathcal{O}^1 \times \mathcal{I}^1$, i.e. without compositions.

Lemma 2. *If $\boldsymbol{H}$ satisfies constraints (25),(26) and (27), and if $\operatorname{rank}(\boldsymbol{H}) = \operatorname{rank}(\boldsymbol{H}_\varepsilon)$ then there exists a WCFG that generates $\boldsymbol{H}_\varepsilon$.*

See proof in Section A.3.

4.2 Convex Optimization

We now present the core optimization program behind our method. Let $\mathbf{vec}(\boldsymbol{H})$ be a vector in $\mathbb{R}^{|\mathcal{O}|\cdot|\mathcal{I}|}$ corresponding to all coefficients of $\boldsymbol{H}$ in column vector form. Let $\boldsymbol{O}$ be a matrix such that $\boldsymbol{O} \cdot \mathbf{vec}(\boldsymbol{H}) = \boldsymbol{z}$ represents the observation constraints. For example, if i-th row of $\boldsymbol{O}$ corresponds to the Hankel coefficient $\boldsymbol{H}(\langle\lambda; \lambda\rangle, (x))$ then $\boldsymbol{z}(i) = p(x)$. Let $\boldsymbol{K}$ be a matrix such that $\boldsymbol{K} \cdot \mathbf{vec}(\boldsymbol{H}) = \mathbf{0}$ represents the constraints (25), (26) and (27).

The optimization problem is:

$$\begin{aligned} \underset{\boldsymbol{H}}{\text{minimize}} \quad & \operatorname{rank}(\boldsymbol{H}) \\ \text{subject to} \quad & \|\boldsymbol{O} \cdot \mathbf{vec}(\boldsymbol{H}) - \boldsymbol{z}\|_2 \le \mu \\ & \boldsymbol{K} \cdot \mathbf{vec}(\boldsymbol{H}) = \mathbf{0} \\ & \|\boldsymbol{H}\|_2 \le 1. \end{aligned} \quad (28)$$

Intuitively, we look for $\boldsymbol{H}$ that agrees with the observable statistics and satisfies the inside-outside constraints. μ is a parameter of the method that controls the degree of error in fitting the observables $\boldsymbol{z}$. The $\|\boldsymbol{H}\|_2 \le 1$ is satisfied by any Hankel matrix derived from a true distribution, and is used to avoid incoherent solutions.

The above optimization problem, however, is computationally hard because of the rank objective. We employ a common relaxation of the rank objective, based on the nuclear norm as in (Balle et al., 2012). The optimization is:

$$\begin{aligned} \underset{\boldsymbol{H}}{\text{minimize}} \quad & \|\boldsymbol{H}\|_* \\ \text{subject to} \quad & \|\boldsymbol{O} \cdot \mathbf{vec}(\boldsymbol{H}) - \boldsymbol{z}\|_2 \le \mu \\ & \boldsymbol{K} \cdot \mathbf{vec}(\boldsymbol{H}) = \mathbf{0} \\ & \|\boldsymbol{H}\|_2 \le 1. \end{aligned} \quad (29)$$

To optimize (29) we employ a projected gradient strategy, similar to the FISTA scheme proposed by Beck and Teboulle (2009). The method alternates between separate projections for the observable constraints, the ℓ_2 norm, the inside-outside constraints, and the nuclear norm. Of these, the latter two are the most expensive.

Elsewhere, we develop theoretical properties of the optimization (28) applied to finite-state transductions (Bailly et al., 2013). One can prove that there is theoretical identifiability of the rank and the parameters of an FST distribution, using a rank minimization formulation. However, this problem is NP-hard, and it remains open whether there exists a polynomial method with identifiability results. These results should generalize to WCFG.

5 Experiments

In this section we describe some experiments with the learning algorithms for WCFG. Our goal is to verify that the algorithms can learn some basic context-free languages, and to study the possibility of using them on real data.

5.1 Synthetic Experiments

We performed experiments on synthetic data, obtained by choosing a PCFG with random parameters ($\in [0, 1]$), with a normalization step in order to get a probability distribution. We built the Hankel matrix from the inside basis $\{(x)\}_{x\in\Sigma}$ and outside basis

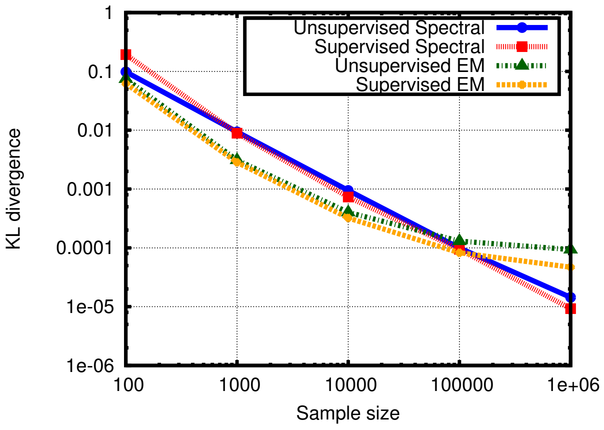

Figure 1: KL divergence for spectral and EM methods, unsupervised and supervised, for different sizes of learning sample, on log-log scales. Results are averages over 50 random target PCFG with 2 states and 2 symbols.

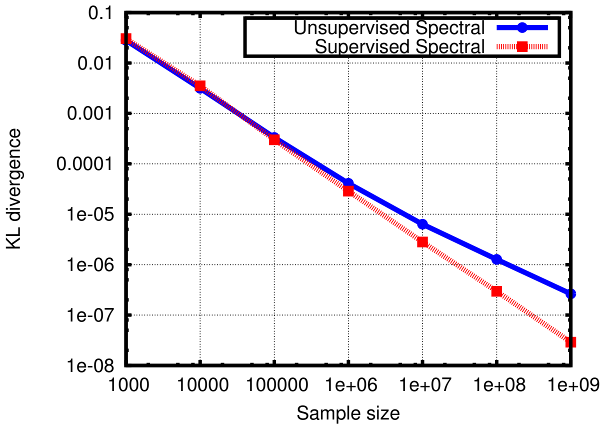

Figure 2: KL divergence for unsupervised and supervised spectral methods, for different sizes of learning sample, on log-log scales. Results are averages over 50 random target PCFG with 3 states and 6 symbols.

$\{\langle\lambda;\lambda\rangle\} \cup \{\langle x;\lambda\rangle, \langle\lambda;x\rangle\}_{x\in\Sigma}$. The composed insides for the operator matrix are thus $\{(x,y)\}_{x,y\in\Sigma}$. The matrix in the optimizer has the following structure

$$H = \begin{array}{c} \\ \langle\lambda;\lambda\rangle \\ \langle x;\lambda\rangle \\ \langle\lambda;x\rangle \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} (y) & \cdots & (y,z) \end{array} \\ \begin{pmatrix} (\lambda;y;\lambda) & \cdots & (\lambda;y,z;\lambda) \\ (x;y;\lambda) & \cdots & (x;y,z;\lambda) \\ (\lambda;y;x) & \cdots & (\lambda;y,z;x) \\ \dots & \dots & \dots \end{pmatrix} \end{array}$$

The constraints we use are:

$$\begin{aligned} K = &\{\boldsymbol{H}((x;y;\lambda)) = \boldsymbol{H}((\lambda;x;y))\}_{x,y\in\Sigma} \cup \\ &\{\boldsymbol{H}((\lambda;x;y)) = \boldsymbol{H}((\lambda;x,y;\lambda))\}_{x,y\in\Sigma} \cup \\ &\{\boldsymbol{H}((x;y;\lambda)) = \boldsymbol{H}((\lambda;x,y;\lambda))\}_{x,y\in\Sigma} \end{aligned}$$

and

$$\begin{aligned} O = &\{\boldsymbol{H}((\lambda;x;\lambda)) = p_S(x)\}_{x\in\Sigma} \cup \\ &\{\boldsymbol{H}((\lambda;x;y)) = p_S(xy)\}_{x,y\in\Sigma} \cup \\ &\{\boldsymbol{H}((x;y,z;\lambda)) + \boldsymbol{H}((\lambda;x,y;z)) = p_S(xyz)\}_{x,y,z\in\Sigma} \end{aligned}$$

We use p_S to denote the empirical distribution. Those are simplified versions of the Hankel, inside, outside and observation constraints. The set O is built from the following remarks: (1) $(xy) = (x,y)$ and (2) $(xyz) = (xy,z) + (x,yz)$. The method uses statistics for sequences up to length 3.

The algorithm we use for the unsupervised spectral method is a simplified version: we use alternatively a hard projection on the constraints (by projecting iteratively on each constraint), and a thresholding-shrinkage operation for the target dimension. We use the same trick as FISTA for the update. We finally use the regular spectral method on this matrix to get our model.

We compare this method with an unsupervised EM, and also with supervised versions of spectral method and EM. We compare the accuracy of the different models in terms of KL-divergence for sequences up to length 10. We run 50 optimization steps for the unsupervised spectral method, and 200 iterations for the EM methods. Figure 1 shows the results, corresponding the the geometric mean over 50 experiments on random targets of 2 symbols and 2 states.

For sample size greater than 10^5, the unsupervised spectral method seems to provide better solutions than both EM and supervised EM. The solution, in terms of KL-divergence, is comparable to the one obtained with the supervised spectral method. The computation time of unsupervised spectral method is almost constant w.r.t. the sample size, around $1.67s$, while computation time of unsupervised EM (resp. supervised EM) is $6.10^3 s$ (resp. $2.10^4 s$) for sample size 10^6.

Figure 2 presents learnings curve for random targets with 3 states and 6 symbols. One can see that, for big sample sizes (10^9), the unsupervised spectral method is losing accuracy compared to the supervised method. This is due to a lack of information,

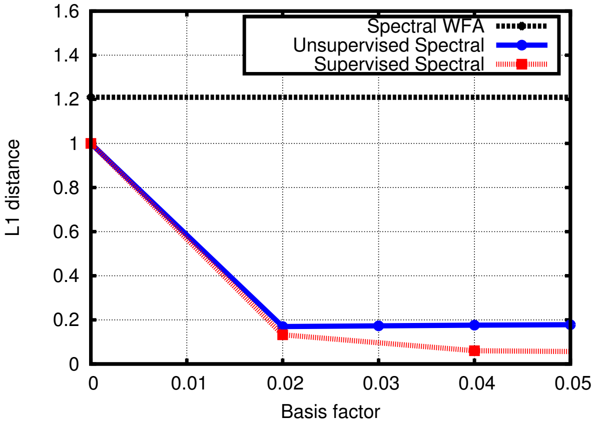

Figure 3: Learning errors for different models in terms of the size of the basis.

and could be overcome by considering a greater basis (e.g. inside sequences up to length 2 or 3).

5.2 Dyck Languages

We now present experiments using the following PCFG:

$$S \rightarrow \; S\, S\ (0.2) \mid a\, S\, b\ (0.4) \mid a\, b\ (0.4)$$

This PCFG generates a probabilistic version of the well-known Dyck language or balanced parenthesis language, an archetypical context-free language.

We do experiments with the following models and algorithms:

- WFA: a Weighted Finite Automata learned using spectral methods as described in (Luque et al., 2012). Parameters: number of states and size of basis.

- Supervised Spectral: a WCFG learned from structured strings using the algorithm of section 3.2. We choose as basis the most frequent insides and outsides observed in the training data. The size of the basis is determined by a parameter f called the basis factor, that determines the proportion of total insides and outsides that will be in the basis.

- Unsupervised Spectral: a WCFG learned from strings using the algorithm of Section 4. The basis is like in the supervised case, but since context-free cuts in the strings are not observed, all possible inside and outsides of the sample (i.e. all possible substrings and contexts) are considered.

basis	size of $\boldsymbol{H}$	obs.	i/o ctr.
1 × 11	39 × 159	34	162
6 × 14	1,163 × 764	146	6,360
12 × 18	4,462 × 2,239	322	25,374
18 × 22	9,124 × 4,149	479	52,524
24 × 26	15,755 × 6,858	657	89,718
30 × 29	19,801 × 8,545	769	112,374
36 × 34	27,989 × 11,682	916	156,690
42 × 37	3,638 × 15,026	1,035	200,346
48 × 41	45,192 × 18,235	1,157	244,398
54 × 45	53,741 × 21,196	1,281	284,466
60 × 48	60,844 × 23,890	1,382	318,354

Table 1: Problem sizes for the WSJ10 training corpus.

basis / n	5	10	15	20
1 × 11	$1.265\,10^{-3}$			
6 × 14	$7.06\,10^{-4}$	$6.92\,10^{-4}$		
12 × 18	$7.30\,10^{-4}$	$6.28\,10^{-4}$	$6.01\,10^{-4}$	
18 × 22	$7.31\,10^{-4}$	$6.29\,10^{-4}$	$5.84\,10^{-4}$	$5.59\,10^{-4}$
24 × 26	$7.35\,10^{-4}$	$6.39\,10^{-4}$	$5.88\,10^{-4}$	$5.31\,10^{-4}$
30 × 29	$7.34\,10^{-4}$	$6.41\,10^{-4}$	$5.86\,10^{-4}$	$5.30\,10^{-4}$

Table 2: Experiments with the unsupervised spectral method on the WSJ10 corpus. Results are in terms of expected L_1 on the training set, for different basis and numbers of states.

We generate a training set by sampling 4,000 strings from the target PCFG and counting the relative frequency of each. For the supervised model, we generate strings paired with their context-free derivation. To measure the quality of the learned models, we use the L_1 distance to the target distribution over a fixed set of strings $\Sigma^{\leq n}$, for $n = 7$.[1]

Figure 3 shows the results for the different models and for different basis sizes (in terms of the basis factor f). Here we can clearly see that the WCFG models, even the unsupervised one, outperform the WFA in reproducing the target distribution.

5.3 Natural Language Experiments

Now we present some preliminar tests using natural language data. For these tests, we used the WSJ10 subset of the Penn Treebank, as Klein and Manning (2002). This dataset consists of the sentences of length ≤ 10 after filtering punctuation and currency. We removed lexical items and mapped the POS tags

[1] Given two functions f_1 and f_2 over strings, the L_1 distance is the sum of the absolute difference over all strings in a set: $\sum_x |f_1(x) - f_2(x)|$.

to the Universal Part-of-Speech Tagset (Petrov et al., 2012), reducing the alphabet to a set of 11 symbols.

Table 1 shows the size of the problem for different basis sizes. As described in the previous subsection for the unsupervised case, we obtain the basis by taking the most frequent observed substrings and contexts. We then compute all yields that can be generated with this basis, and close the basis to include all possible insides and outsides with operations completions, such that we create a Hankel as described in Section 4.1. Table 1 shows, for each base, the size of $\boldsymbol{H}$ we induce, the number of observable constraints (i.e. sentences we train from), and the number of inside-outside constraints.

With the current implementation of the optimizer we were only able to run the unsupervised learning for small basis sizes. Table 2 shows the expected L_1 on training data. For a fixed basis, as we increase the number of states we see that the error decreases, showing that the method is inducing a Hankel matrix that explains the observable statistics.

6 Conclusions

We have presented a novel approach for unsupervised learning of WCFG. Our method combines ingredients of spectral learning with low-rank convex optimization methods.

Our method optimizes over a matrix that, even if it grows polynomially with respect to the size of training, results in a large problem. To scale the method to learn languages of the complexity of natural languages we would need to identify optimization algorithms specially suited for this problem.

A Proofs

A.1 Proof of Inside-Outside Eq. 16 and 17

For the inside function, the base case is trivial. By induction:

$$\begin{aligned}
\beta_G(x)[i] &= \sum_{x=x_1x_2} \boldsymbol{A}(\beta_G(x_1) \otimes \beta_G(x_2))[i] \\
&= \sum_{\substack{j,k\in V\\ x=x_1x_2}} \boldsymbol{A}[i,j,k] \cdot \beta_G(x_1)[j] \cdot \beta_G(x_2)[k] \\
&= \sum_{\substack{j,k\in V\\ x=x_1x_2}} w_R(i \rightarrow j\ k) \cdot \bar{\beta}_{\bar{G}}(j \overset{\star}{\Rightarrow} x_1) \cdot \bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} x_2) \\
&= \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x)
\end{aligned}$$

For the outside function, let $\boldsymbol{e}_i$ be an n-dimensional vector with coordinate i to 1 and the rest to 0. We reformulate the mapping as:

$$\alpha_G(x;z)^\top \boldsymbol{e}_i = \bar{\alpha}_{\bar{G}}(x;i;z) \tag{30}$$

The base case is trivial by definitions. We use the property of Kronecker products that $(\boldsymbol{v} \otimes \boldsymbol{I}_n)\boldsymbol{v}' = (\boldsymbol{v} \otimes \boldsymbol{v}')$ and $(\boldsymbol{I}_n \otimes \boldsymbol{v})\boldsymbol{v}' = (\boldsymbol{v}' \otimes \boldsymbol{v})$ for $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{R}^n$. We first look at one of the terms of $\alpha_G(x;z)^\top \boldsymbol{e}_i$:

$$\begin{aligned}
&\alpha_G(x_1;z)^\top \boldsymbol{A}(\beta_G(x_2) \otimes \boldsymbol{I}_n)\boldsymbol{e}_i \\
&= \alpha_G(x_1;z)^\top \boldsymbol{A}(\beta_G(x_2) \otimes \boldsymbol{e}_i) \\
&= \sum_{j,k\in V} (\alpha_G(x_1;z)^\top \boldsymbol{e}_j) \cdot \boldsymbol{A}[j,k,i] \cdot \beta_G(x_2)[k] \\
&= \sum_{j,k\in V} \bar{\alpha}_{\bar{G}}(x_1;j;z) \cdot w_R(j \rightarrow k\ i) \cdot \bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} x_2)
\end{aligned}$$

Applying the distributive property in $\alpha_G(x;z)^\top \boldsymbol{e}_i$ it is easy to see that all terms are mapped to the corresponding term in $\bar{\alpha}_{\bar{G}}(x;i;z)$.

A.2 Proof of Lemma 1

Let $G' = \langle \boldsymbol{\alpha}'_\star, \{\boldsymbol{\beta}'_\sigma\}, \boldsymbol{A}' \rangle$ be a WCFG for f that induces a rank factorization $\boldsymbol{H} = \boldsymbol{P}'\boldsymbol{S}'$. We first show that there exists an invertible matrix $\boldsymbol{M}$ that changes the basis of the operators of G into those of G'. Define $\boldsymbol{M} = \boldsymbol{S}'\boldsymbol{S}^+$ and note that $\boldsymbol{P}^+\boldsymbol{P}'\boldsymbol{S}'\boldsymbol{S}^+ = \boldsymbol{P}^+\boldsymbol{H}\boldsymbol{S}^+ = \boldsymbol{I}$ implies that $\boldsymbol{M}$ is invertible with $\boldsymbol{M}^{-1} = \boldsymbol{P}^+\boldsymbol{P}'$. We now check that the operators of G correspond to the operators of G' under this change of basis. First we see that

$$\begin{aligned}
\boldsymbol{A} &= \boldsymbol{P}^+\boldsymbol{H}_A(\boldsymbol{S} \otimes \boldsymbol{S})^+ \\
&= \boldsymbol{P}^+\boldsymbol{P}'\boldsymbol{A}'(\boldsymbol{S}' \otimes \boldsymbol{S}')(\boldsymbol{S} \otimes \boldsymbol{S})^+ \\
&= \boldsymbol{M}^{-1}\boldsymbol{A}'(\boldsymbol{S}'\boldsymbol{S}^+ \otimes \boldsymbol{S}'\boldsymbol{S}^+) \\
&= \boldsymbol{M}^{-1}\boldsymbol{A}'(\boldsymbol{M} \otimes \boldsymbol{M}) \quad .
\end{aligned}$$

Now, since $\boldsymbol{h}_\star = \boldsymbol{\alpha}'^\top_\star \boldsymbol{S}'$ and $\boldsymbol{h}_\sigma = \boldsymbol{P}'\boldsymbol{\beta}'_\sigma$, it follows that $\boldsymbol{\alpha}^\top_\star = \boldsymbol{\alpha}'_\star{}^\top \boldsymbol{M}$ and $\boldsymbol{\beta}_\sigma = \boldsymbol{M}^{-1}\boldsymbol{\beta}'_\sigma$.

Finally we check that G and G' compute the same function, namely $f(o,i) = \alpha_G(o)^\top\beta_G(i) = \alpha_{G'}(o)^\top\beta_{G'}(i)$. We first see that $\beta_G(x) =$

$\boldsymbol{M}^{-1}\beta_{G'}(x)$:

$$\beta_G(\sigma) = \boldsymbol{\beta}_\sigma = \boldsymbol{M}^{-1}\boldsymbol{\beta}'_\sigma \tag{31}$$

$$\begin{aligned}\beta_G(x) &= \sum_{x=x_1x_2} \boldsymbol{A}(\beta_G(x_1) \otimes \beta_G(x_2)) \qquad (32)\\ &= \sum_{x=x_1x_2} \boldsymbol{M}^{-1}\boldsymbol{A}'(\boldsymbol{M} \otimes \boldsymbol{M})(\beta_G(x_1) \otimes \beta_G(x_2))\\ &= \boldsymbol{M}^{-1} \sum_{x=x_1x_2} \boldsymbol{A}'(\boldsymbol{M}\beta_G(x_1) \otimes \boldsymbol{M}\beta_G(x_2))\\ &= \boldsymbol{M}^{-1} \sum_{x=x_1x_2} \boldsymbol{A}'(\beta_{G'}(x_1) \otimes \beta_{G'}(x_2))\end{aligned}$$

It can also be shown that $\alpha_G(x;z)^\top = \alpha_{G'}(x;z)^\top \boldsymbol{M}$. One must see that in any term:

$$\begin{aligned}&\alpha_G(x_1;z)^\top \boldsymbol{A}(\beta_G(x_2) \otimes \boldsymbol{I}_n) \qquad (33)\\ &= \alpha_G(x_1;z)^\top \boldsymbol{M}^{-1}\boldsymbol{A}'(\boldsymbol{M} \otimes \boldsymbol{M})(\beta_G(x_2) \otimes \boldsymbol{I}_n)\\ &= \alpha_{G'}(x_1;z)^\top \boldsymbol{A}'(\boldsymbol{M}\beta_G(x_2) \otimes \boldsymbol{M}\boldsymbol{I}_n)\\ &= \alpha_{G'}(x_1;z)^\top \boldsymbol{A}'(\beta_{G'}(x_2) \otimes \boldsymbol{I}_n)\boldsymbol{M}\end{aligned}$$

and the relation follows. Finally:

$$\begin{aligned}&\alpha_G(x;z)^\top \beta_G(y) \qquad (34)\\ &= \alpha_{G'}(x;z)^\top \boldsymbol{M}\boldsymbol{M}^{-1}\beta_{G'}(y)\\ &= \alpha_{G'}(x;z)^\top \beta_{G'}(y)\end{aligned}$$

□

A.3 Proof of Lemma 2

We will use the following sub-blocks of $\boldsymbol{H}$:

- $\boldsymbol{H}_\varepsilon$ is the sub-block restricted to $\mathcal{O}^1 \times \mathcal{I}^1$, i.e. without compositions.
- $\boldsymbol{H}_A$ is the sub-block restricted to $\mathcal{O}^1 \times \mathcal{I}^2$, i.e. inside compositions.
- $\boldsymbol{H}'_A$ is the sub-block restricted to $\mathcal{O}^2 \times \mathcal{I}^1$, i.e. outside compositions.
- $\boldsymbol{h}_\star^\top \in \mathbb{R}^{\mathcal{I}^1}$ is the row of $\boldsymbol{H}_\varepsilon$ for $\langle\lambda;\lambda\rangle$.
- $\boldsymbol{h}_{(x)} \in \mathbb{R}^{\mathcal{O}^1}$ is the column of $\boldsymbol{H}_\varepsilon$ for (x).
- $\boldsymbol{h}_{(x_1,x_2)} \in \mathbb{R}^{\mathcal{O}^1}$ is the column of $\boldsymbol{H}_A$ for (x_1,x_2).
- $\boldsymbol{h}'_{\langle x;z\rangle} \in \mathbb{R}^{\mathcal{I}^1}$ is the row of $\boldsymbol{h}_\varepsilon$ for $\langle x;z\rangle$.
- $\boldsymbol{h}'_{\langle x_1,x_2;z\rangle}$ and $\boldsymbol{h}'_{\langle x;z_1,z_2\rangle}$ be the rows in $\mathbb{R}^{\mathcal{I}^1}$ of $\boldsymbol{h'}_A$ for $\langle x_1,x_2;z\rangle$ and $\langle x;z_1,z_2\rangle$).

One supposes that $\mathrm{rank}(\boldsymbol{H}_\varepsilon) = \mathrm{rank}(\boldsymbol{H})$. We define G as

$$\alpha_\star^\top = \boldsymbol{h}_\star^\top \boldsymbol{H}_\varepsilon^+, \beta_a = \boldsymbol{h}_{(a)}, \boldsymbol{A} = \boldsymbol{H}_A(\boldsymbol{H}_\varepsilon^+ \otimes \boldsymbol{H}_\varepsilon^+)$$

Lemma 3. *One has that* $\beta_G(x) = \boldsymbol{h}_{(x)}$*, and* $\beta_G(x_1,x_2) = \boldsymbol{h}_{(x_1,x_2)}$.

Proof. By induction. For sequences of size 1, one has $\beta_G(x) = \beta_x = \boldsymbol{h}_{(x)}$. For the recursive case, let $\boldsymbol{e}_{(x)}$ be a vector in $\mathbb{R}^{\mathcal{I}^1}$ with 1 in the coordinate of (x) in $\boldsymbol{H}_\varepsilon$. Let $\boldsymbol{e}_{(x,y)}$ be a vector in $\mathbb{R}^{\mathcal{I}^2}$ with 1 in the coordinate of (x,y) in $\boldsymbol{H}_A$. For $\beta_G(x,y)$, one has $\boldsymbol{H}_\varepsilon^+\beta_G(x) = \boldsymbol{e}_{(x)}$, and $\boldsymbol{H}_\varepsilon^+\beta_G(y) = \boldsymbol{e}_{(y)}$, thus $\boldsymbol{H}_\varepsilon^+\beta_G(x) \otimes \boldsymbol{H}_\varepsilon^+\beta_G(y) = \boldsymbol{e}_{(x,y)}$ and $\boldsymbol{H}_A(\boldsymbol{H}_\varepsilon^+\beta_G(x) \otimes \boldsymbol{H}_\varepsilon^+\beta_G(y)) = \boldsymbol{h}_{(x,y)}$. Finally, one has that $\beta_G(x) = \sum_{x=x_1x_2} \beta_G(x_1,x_2) = \sum_{x=x_1x_2} \boldsymbol{h}_{(x_1,x_2)} = \boldsymbol{h}_{(x_1x_2x_3)}$ by the equation (26). □

One has a symmetric result for outside vectors. We define G' as

$$\alpha_\star^\top = \boldsymbol{h}_\star^\top, \beta_a = \boldsymbol{H}_\varepsilon^+\boldsymbol{h}_{(a)}, \boldsymbol{A} = \boldsymbol{H}_\varepsilon^+\boldsymbol{H}_A$$

Lemma 4. *One has that* $\alpha_{G'}(\langle x;z\rangle)^\top = \boldsymbol{h}'_{\langle x;z\rangle}$, $\alpha_{G'}(\langle x_1,x_2;z\rangle)^\top = \boldsymbol{h}'_{\langle x_1,x_2;z\rangle}$ *and* $\alpha_{G'}(\langle x;z_1,z_2\rangle)^\top = \boldsymbol{h}'_{\langle x;z_1,z_2\rangle}$.

Proof. (Sketch) Equation (31) is used in the same way than (27) before. Equation (25) is used to ensure a link between $\boldsymbol{H}'_A$ and $\boldsymbol{H}_A$. □

Let g be the mapping computed by G and G'. One has that $g(o,i) = \alpha_{G'}(o)^\top\beta_{G'}(i) = \alpha_G(o)^\top\beta_G(i) = \alpha_{G'}(o)^\top\boldsymbol{H}_\varepsilon^+\beta_G(i) = \boldsymbol{H}_\varepsilon(o,i)$.

Acknowledgements We are grateful to Borja Balle and the anonymous reviewers for providing us with helpful comments. This work was supported by a Google Research Award, and by projects XLike (FP7-288342), ERA-Net CHISTERA VISEN, TACARDI (TIN2012-38523-C02-02), BASMATI (TIN2011-27479-C04-03), SGR-GPLN (2009-SGR-1082) and SGR-LARCA (2009-SGR-1428). Xavier Carreras was supported by the Ramón y Cajal program of the Spanish Government (RYC-2008-02223). Franco M. Luque was supported by the National University of Córdoba and by a Postdoctoral fellowship of CONICET, Argentinian Ministry of Science, Technology and Productive Innovation.

References

Pieter Adriaans, Marten Trautwein, and Marco Vervoort. 2000. Towards high speed grammar induction on large text corpora. In *SOFSEM 2000: Theory and Practice of Informatics*, pages 173–186. Springer.

Raphaël Bailly, Franois Denis, and Liva Ralaivola. 2009. Grammatical inference as a principal component analysis problem. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 33–40, Montreal, June. Omnipress.

Raphaël Bailly, Amaury Habrard, and François Denis. 2010. A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of the 21st International Conference Algorithmic Learning Theory*, Lecture Notes in Computer Science, pages 74–88. Springer.

Raphaël Bailly, Xavier Carreras, and Ariadna Quattoni. 2013. Unsupervised spectral learning of finite-state transducers. In *Advances in Neural Information Processing Systems 26*.

James K. Baker. 1979. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf, editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

Borja Balle, Ariadna Quattoni, and Xavier Carreras. 2011. A spectral learning algorithm for finite state transducers. In *Proceedings of ECML PKDD*, pages 156–171.

Borja Balle, Ariadna Quattoni, and Xavier Carreras. 2012. Local loss optimization in operator models: A new insight into spectral learning. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-2012)*, ICML '12, pages 1879–1886, New York, NY, USA, July. Omnipress.

Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, March.

Stanley F Chen. 1995. Bayesian grammar induction for language modeling. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 228–235. Association for Computational Linguistics.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*, page 13. Association for Computational Linguistics.

Alexander Clark. 2007. Learning deterministic context free grammars: The omphalos competition. *Machine Learning*, 66(1):93–110.

Shay B. Cohen, David M. Blei, and Noah A. Smith. 2010. Variational inference for adaptor grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 564–572, Los Angeles, California, June. Association for Computational Linguistics.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable pcfgs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 223–231, Jeju Island, Korea, July. Association for Computational Linguistics.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable pcfgs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 148–157, Atlanta, Georgia, June. Association for Computational Linguistics.

Matthew Gormley and Jason Eisner. 2013. Nonconvex global optimization for latent-variable models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sofia, Bulgaria, August. 11 pages.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. 2009. A spectral algorithm for learning hidden markov models. In *Proceedings of the Annual Conference on Computational Learning Theory (COLT)*.

Daniel Hsu, Sham Kakade, and Percy Liang. 2012. Identifiability and unmixing of latent parse trees. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1520–1528.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Kenichi Kurihara and Taisuke Sato. 2006. Variational bayesian grammar induction for natural language. In *Grammatical Inference: Algorithms and Applications*, pages 84–96. Springer.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Percy Liang, Slav Petrov, Michael I Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical dirichlet processes. In *EMNLP-CoNLL*, pages 688–697.

Franco M. Luque, Ariadna Quattoni, Borja Balle, and Xavier Carreras. 2012. Spectral learning for non-deterministic dependency parsing. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 409–419, Avignon, France, April. Association for Computational Linguistics.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Newark, Delaware, USA, June. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of LREC*, May.

Kewei Tu and Vasant Honavar. 2008. Unsupervised learning of probabilistic context-free grammar using iterative biclustering. In *Grammatical Inference: Algorithms and Applications*, pages 224–237. Springer.

Menno Van Zaanen. 2000. Abl: Alignment-based learning. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 961–967. Association for Computational Linguistics.