Identification with Probability One of Stochastic Deterministic Linear Languages

Colin de la Higuera[1] and Jose Oncina[2‡]

[1] EURISE, Université de Saint-Etienne, 23 rue du Docteur Paul Michelon, 42023 Saint-Etienne, France
cdlh@univ-st-etienne.fr,
WWW home page: http://eurise.univ-st-etienne.fr/~cdlh
[2] Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Ap.99. E-03080 Alicante, Spain
oncina@dlsi.ua.es,
WWW home page: http://www.dlsi.es/~oncina

Abstract. Learning context-free grammars is generally considered a very hard task. This is even more the case when learning has to be done from positive examples only. In this context one possibility is to learn stochastic context-free grammars, by making the implicit assumption that the distribution of the examples is given by such an object. Nevertheless this is still a hard task for which no algorithm is known. We use recent results to introduce a proper subclass of linear grammars, called deterministic linear grammars, for which we prove that a small canonical form can be found. This has been a successful condition for a learning algorithm to be possible. We propose an algorithm for this class of grammars and we prove that our algorithm works in polynomial time, and structurally converges to the target in the paradigm of identification in the limit with probability 1. Although this does not ensure that only a polynomial size sample is necessary for learning to be possible, we argue that the criterion means that no added (hidden) bias is present.

1 Introduction

Context-free grammars are known to have a superior modeling capacity than regular grammars or finite state automata. Learning these grammars is also harder but considered an important and challenging task. Yet without external help such as a knowledge of the structure of the strings [Sak92] only clever but limited heuristics have been proposed [LS00,NMW97].

When no positive examples exist, or when the actual problem is that of building a language model, stochastic context-free grammars have been proposed. In a number of applications (computational biology [SBH$^+$94] and speech recognition [WA02] are just two typical examples), it is speculated that success will

‡ The author thanks the Generalitat Valenciana for partial support of this work through project CETIDIB/2002/173.

depend on being able to replace finite state models such as Hidden Markov Models by stochastic context-free grammars. Yet the problem of learning this type of grammar from strings has rarely been addressed. The usual way of dealing with the problem still consists in first learning a structure, and then estimating the probabilities [Bak79].

In the more theoretical setting of learning from both examples and counter-examples classes of grammars that are more general than the regular grammars, but restricted to cases where both determinism and linearity apply have been studied [dlHO02].

On the other hand, learning (deterministic) regular stochastic grammars has received a lot of attention over the past 10 years. A well known algorithm for this task is ALERGIA [CO94], which has been improved by different authors [YLT00,CO99], and applied to different tasks [WA02].

We synthesize in this paper both types of results and propose a novel class of stochastic languages that we call stochastic deterministic linear languages. We prove that each language of the class admits an equivalence relation of finite index, thus leading to a canonical normal form. We propose an algorithm that works in polynomial time with respect to the learning data. It can identify with probability one any language in the class.

In section 2 the necessary definitions are given. We prove in section 3 the existence of a small normal form, and give in section 4 a learning algorithm that can learn grammars in normal form.

2 Definitions

2.1 Languages and grammars

An alphabet Σ is a finite nonempty set of symbols. Σ^* denotes the set of all finite strings over Σ. A language L over Σ is a subset of Σ^*. In the following, unless stated otherwise, symbols are indicated by $a, b, c, \ldots$, strings by $u, v, \ldots$, and the empty string by λ. The length of a string u will be denoted $|u|$.

Let $u, v \in \Sigma^*, u^{-1}v = w$ such that $v = uw$ (undefined if u is not a prefix of v) and $uv^{-1} = w$ such that $u = wv$ (undefined if v is not a suffix of u). Let L be a language and $u \in \Sigma^*$, $u^{-1}L = \{v : uv \in L\}$ and $Lu^{-1} = \{v : vu \in L\}$.

Let L be a language, the *prefix set* is $\mathrm{Pref}(L) = \{x : xy \in L\}$. The *longest common suffix* ($\mathrm{lcs}(L)$) of L is the longest string u such that $(Lu^{-1})u = L$.

A *context-free grammar* G is a quadruple (Σ, V, R, S) where Σ is a finite alphabet (of terminal symbols), V is a finite alphabet (of variables or non-terminals), $R \subset V \times (\Sigma \cup V)^*$ is a finite set of production rules, and $S (\in V)$ is the starting symbol. We will denote $uTv \rightarrow uwv$ when $(T, w) \in R$. $\xrightarrow{*}$ is the reflexive and transitive closure of $\rightarrow$. If there exists $u_0, \ldots, u_k$ such that $u_0 \rightarrow \cdots \rightarrow u_k$ we will write $u_0 \xrightarrow{k} u_k$. We denote by $L_G(T)$ the language $\{w \in \Sigma^* : T \xrightarrow{*} w\}$. Two grammars are equivalent if they generate the same language. A context-free grammar $G = (\Sigma, V, R, S)$ is *linear* if $R \subset V \times (\Sigma^* V \Sigma^* \cup \Sigma^*)$.

2.2 Stochastic languages

A stochastic language L over Σ is defined by a probability density function over Σ^* giving the probability $p(w|L)$ that the string $w \in \Sigma^*$ appears in the language. To be consistent, a necessary condition is that $\sum_{x \in \Sigma^*} p(x|L) = 1$.

When convenient, we are going to represent a stochastic language as a set of pairs: $L = \{(u, p(u|L)) : p(u|L) > 0\}$. Consequently $(u, p_u) \in L \implies p(u|L) > 0$.

Also to avoid unnecessary notations we will allow the empty set $\emptyset$ to be a stochastic language (paired with an arbitrary function).

The probability of any subset $X \subseteq \Sigma^*$ is given by

$$p(X|L) = \sum_{u \in X} p(u|L)$$

Let L be a stochastic language and $u \in \Sigma^*$,

$$\begin{aligned}
\mathrm{Pref}(L) &= \{u : (uv, p) \in L\},\\
\mathrm{Sf}(L) &= \{u : (vu, p) \in L\},\\
uL &= \{(uv, p) : (v, p) \in L\},\\
Lu &= \{(vu, p) : (v, p) \in L\},\\
u^{-1}L &= \{(v, p_v) : (uv, p(u\Sigma^*|L)p_v) \in L\}\\
Lu^{-1} &= \{(v, p_v) : (vu, p_v p(\Sigma^* u|L)) \in L\}.
\end{aligned}$$

Note that the expresions for $u^{-1}L$ and Lu^{-1} are equivalent to $\{(v, p_v) : p_v = p(uv|L)/p(u\Sigma^*|L)\}$ and $\{(v, p_v) : p_v = p(vu|L)/p(u\Sigma^*|L)\}$ respectively but avoiding division by zero problems.

Of course, if u is a common prefix (u common suffix) of L then $p(u\Sigma^*|L) = 1$ ($p(\Sigma^* u|L) = 1$) and $u^{-1}L = \{(v, p_v) : (uv, p_v) \in L\}$ ($Lu^{-1} = \{(v, p_v) : (vu, p_v) \in L\}$).

We denote the *longest common suffix reduction* of a stochastic language L by $L\downarrow = \{(u, p) : z = \mathrm{lcs}(L), (uz, p) \in L\}$, where $\mathrm{lcs}(L) = \mathrm{lcs}\{u : (u, p) \in L\}$.

Note that if L is a stochastic language then $\forall u$ $u^{-1}L$, Lu^{-1} and $L\downarrow$ are also stochastic languages.

A *stochastic deterministic linear* (SDL) grammar, $G = (\Sigma, V, R, S, p)$ consists Σ, V, S as for context-free grammars, a finite set R of derivation rules with either of the structures $X \to aYw$ or $X \to \lambda$; such that $X \to aYw, X \to aZv \in R \Rightarrow Y = Z \wedge w = v$, and a real function $p : R \to]0, 1]$ giving the probability of each derivation.

The probability $p(S \xrightarrow{*} w)$ that the grammar G generates the string w is defined recursively as:

$$p(X \xrightarrow{*} avw) = p(X \to aYw) p(Y \xrightarrow{*} v)$$

where Y is the only variable such that $X \to Yw \in R$ (if such variable does not exist, then $p(X \to aYw) = 0$ is assumed). It can be shown that if $\forall A \in V \sum p(A \to \alpha) = 1$ and G does not contains useless symbols then G

defines a *stochastic deterministic linear language* L_G through the probabilities $p(w|L_G) = p(S \overset{*}{\rightarrow} w)$.

Let X be a variable in the SDL grammar $G = (\Sigma, V, R, S, p)$ then $L_G(X) = \{(u, p_u) : p(X \overset{*}{\rightarrow} u) = p_u\}$.

A non stochastic version of the above definition is studied in [dlHO02]: it corresponds to a very general class of linear grammars that includes for instance grammars for all regular languages, palindrome languages and $\{a^n b^n : n \in \mathbb{N}\}$. In the same paper a more general form of deterministic linear grammars was proposed, equivalent to the form we use to support our grammars here. Extension of these results to general deterministic linear grammars will not be done in this paper.

3 A canonical form for stochastic deterministic linear grammars

For a class of stochastic languages to be identifiable in the limit with probability one a reasonable assumption is that there exists some small canonical form for any language representable in the class. We prove in this section that such is indeed the case for stochastic deterministic linear grammars.

The purpose of this section is to reach a computable normal form for SDL grammars. For this we first define a normal form for these grammars (called *advanced* as the longest common suffixes appear as soon as possible), and then construct such a grammar from any deterministic linear language.

Definition 1 (Advanced form). *A stochastic deterministic linear grammar $G = (\Sigma, V, R, S, p)$ is in* advanced *form if:*

1. $\forall (T, aT'w) \in R, w = \text{lcs}(a^{-1} L_G(T))$;
2. *all non-terminal symbols are accessible:* $\forall T \in V\ \exists u, v \in \Sigma^* : S \overset{*}{\rightarrow} uTv$ *and useful:* $\forall T \in V,\ L_G(T) \neq \emptyset$;
3. $\forall T, T' \in V,\ L_G(T) = L_G(T') \Rightarrow T = T'$.

We build the canonical form from the language so as to ensure uniqueness:

Definition 2 (Common suffix-free language equivalence). *Given a stochastic language L we define recursively the* common suffix-free *languages* $\text{CSF}_L(\cdot)$, *and the associated equivalence relation as follows:*

$$\left.\begin{array}{r}\text{CSF}_L(\lambda) = L \\ \text{CSF}_L(xa) = (a^{-1}\,\text{CSF}_L(x))\downarrow\end{array}\right|\ x \equiv_L y \iff \text{CSF}_L(x) = \text{CSF}_L(y)$$

Proposition 1. *The equivalence relation $\equiv_L$ has a finite index.*

Proof. See the appendix.

Definition 3 (A canonical grammar). *Given any stochastic linear deterministic language L, the canonical grammar associated with L is $G_L = (\Sigma, V, R, S_{\mathrm{CSF}_L(\lambda)}, p)$ where:*

$$
\begin{aligned}
&V = \{S_{\mathrm{CSF}_L(x)} : \mathrm{CSF}_L(x) \neq \emptyset\}\\
&R = \{S_{\mathrm{CSF}_L(x)} \rightarrow a S_{\mathrm{CSF}_L(xa)} \operatorname{lcs}(a^{-1}\,\mathrm{CSF}_L(x)) : \mathrm{CSF}_L(xa) \neq \emptyset\}\\
&\quad \cup \{S_{\mathrm{CSF}_L(x)} \rightarrow \lambda : \lambda \in \mathrm{CSF}_L(x)\}\\
&p(S_{\mathrm{CSF}_L(x)} \rightarrow aYw) = p(a\Sigma^* w |\, \mathrm{CSF}_L(x)) = p(a\Sigma^* |\, \mathrm{CSF}_L(x))\\
&p(S_{\mathrm{CSF}_L(x)} \rightarrow \lambda) = p(\lambda |\, \mathrm{CSF}_L(x))
\end{aligned}
$$

Proposition 1 allows this construction to terminate. The correctness of the construction is a consequence of:

Proposition 2. *Let L be a SDL language and let $G_L = (\Sigma, V, R, S, p)$ be its associated canonical grammar. Then $L = L_{G_L}(S)$.*

Proof. See the appendix.

Theorem 1. *Given a SDL grammar $G = (\Sigma, V_G, R_G, S_G, p_G)$, let $G_L = (\Sigma, V_{G_L}, R_{G_L}, S_{G_L}, p_{G_L})$ be the canonical grammar that generates $L = L_G(S_G)$,*

1. *G_L is advanced*
2. *$|V_{G_L}| \leqslant |V_G| + 1$.*

Proof. We prove that G_L is advanced by showing that conditions 1 to 4 of definition 1 hold. The proof of the second part is a consequence of lemma 5 and proposition 4: both results are given and proved in the appendix: they state that the number of classes of CSF_L and thus the number of variables in the canonical grammar, is bounded by the number of non-terminals in the original grammar.

4 Learning SDL grammars

As SDL languages admit a small canonical form it will be sufficient to have an algorithm that can identify a grammar in this type of canonical form.

We are going to divide the task of learning in two steps:

1. Identify the topology of the grammar, that is type $A \rightarrow aBv$ rules, without the probabilities.
2. Add the $A \rightarrow \lambda$ type rules and assign the probabilities.

The second step can be done by counting the use of the different rules while parsing a sample (maximum likelihood estimation); alternatively, as this does not achieve identification, techniques based on Stern-Brocot trees can be used in a similar way as in [dlHT00]. Hence we are going to concentrate on the first step.

Definition 4. *Let L be a SDL language, and $\leqslant$ a length lexicographic order relation over Σ^*, the* shortest prefix *set of L is* $\mathrm{Sp}_L = \{x \in \mathrm{Pref}(L) : \mathrm{CSF}_L(x) \neq \emptyset \wedge y \equiv_L x \Rightarrow x \leqslant y\}$

Note that, in a canonical grammar, we have a one-to-one relation between strings in Sp and non-terminals of the grammar. We shall thus use the strings in Sp as identifiers for the non terminal symbols. To describe the algorithm we shall imagine that we have access to an unlimited oracle that knows language L and to which we can address the following queries:

$$\begin{array}{rcl} \mathrm{next}_L(x) & = & \{xa \in \mathrm{Pref}(L), a \in \Sigma\} \\ \mathrm{equiv}_L(x,y) & \iff & x \equiv_L y \\ \mathrm{right}_L(xa) & = & \mathrm{lcs}(a^{-1}\,\mathrm{CSF}_L(x)) \end{array}$$

Algorithm 1 visits the prefixes of the language L in length lexicographic order, and constructs the canonical grammar corresponding to definition 3. If a prefix xa is visited and no previous equivalent non terminal has been found (and placed in Sp), this prefix is added to Sp as a new non terminal and the corresponding rule is added to the grammar. If there exists an equivalent non terminal y in Sp then the corresponding rule is added but the strings for which x is a prefix will not be visited (they will not be added to W). When the algorithm finishes, Sp contains all the shortest prefixes of the language.

Algorithm 1 is clearly polynomial in the size of set W, provided the auxiliary functions are polynomial.

A *stochastic sample* S of the stochastic language L is an infinite sequence of strings generated according to the probability distribution $p(w|L)$. We denote with S_n the sequence of the n first strings (not necessarily different) in S, which will be used as input for the algorithm. The number of occurrences in S_n of the string x will be denoted with $c_n(x)$, and for any subset $X \subseteq \Sigma^*$, $c_n(X) = \sum_{x\in X} c_n(x)$. Note that in the context of the algorithm, $\mathrm{next}_L(x)$, $\mathrm{right}_L(xa)$ and $\mathrm{equiv}_L(xa, y)$ are only computed when x and y are in Sp_L. Therefore the size of W is bounded by the number of prefixes of S_n. In order to use algorithm 1 with a sample S_n instead of an oracle with access to the whole language L

Algorithm 1 Computing G using functions next, right and equiv

Require: functions next, right and equiv, language L
Ensure: $L(G) = L$ with $G = (\Sigma, V, R, S_\lambda)$
  $\mathrm{Sp} = \{\lambda\}$; $V = \{S_\lambda\}$
  $W = \mathrm{next}_L(\lambda)$
  while $W \neq \emptyset$ **do**
    $xa = \min_{\leqslant} W$
    $W = W - \{xa\}$
    if $\exists y \in \mathrm{Sp} : \mathrm{equiv}_L(xa, y)$ **then**
      add $S_x \to aS_y\,\mathrm{right}_L(xa)$ to R
    else
      $\mathrm{Sp} = \mathrm{Sp} \cup \{xa\}$; $V = V \cup \{S_{xa}\}$
      $W = W \cup \mathrm{next}_L(xa)$
      add $S_x \to aS_{xa}\,\mathrm{right}_L(xa)$ to R
    end if
  end while

the 3 functions must be implemented as functions of S_n ($\mathrm{next}_{S_n}(\cdot)$, $\mathrm{right}_{S_n}(\cdot)$ and $\mathrm{equiv}_{S_n}(\cdot,\cdot)$) rather than L so that they give the same result as $\mathrm{next}_L(x)$, $\mathrm{right}_L(xa)$ and $\mathrm{equiv}_L(xa, y)$ when $x, y \in \mathrm{Sp}_L$ and n tends to infinity.

In order to simplify notations we introduce:

Definition 5. *Let L be a SDL language, then*

$$\mathrm{tail}_L(x) = \begin{cases} \mathrm{lcs}(x^{-1}L) & \textit{if } x \neq \lambda \\ \lambda & \textit{if } x = \lambda \end{cases} \qquad \forall x : CSF_L(x) \neq \emptyset$$

A slightly different function tail that works over sequences is now introduced. This function will be used to define a function right to work over sequences.

Definition 6. *Let S_n be a finite sequence of strings, then*

$$\mathrm{tail}_{S_n}(x) = \begin{cases} \mathrm{lcs}(x^{-1}S_n) & \textit{if } x \neq \lambda \\ \lambda & \textit{if } x = \lambda \end{cases} \qquad \forall x \in \mathrm{Pref}(S_n)$$

Lemma 1. *Let $G_L = (\Sigma, V, R, S, p)$ be the canonical grammar of a SDL language L, $\forall x : CSF_L(x) \neq \emptyset$,*

$$\mathrm{lcs}(a^{-1}\,\mathrm{CSF}_L(x)) = (\mathrm{tail}_L(xa))(\mathrm{tail}_L(x))^{-1}$$

Proof. The proof is similar to lemma 4(1) of [dlHO02]

Definition 7.

$$\mathrm{next}_{S_n}(x) = \{xa : \exists xay \in S_n\}$$
$$\mathrm{right}_{S_n}(xa) = \mathrm{tail}_{S_n}(xa)\,\mathrm{tail}_{S_n}(x)^{-1}$$

It should be noticed that the above definition ensures that the functions next_{S_n} and right_{S_n} can be computed in time polynomial in the size of S_n. We now prove that the above definition allows functions next_{S_n} and right_{S_n} to converge in the limit, to the intended functions next_L and right_L:

Lemma 2. *Let L be a SDL language, for each sample S_n of L containing a set $D \subseteq \{x : (x, p) \in L\}$ such that:*

1. $\forall x \in \mathrm{Sp}_L\, \forall a \in \Sigma : xa \in \mathrm{Pref}(L) \Rightarrow \exists xaw \in D$.
2. $\forall x \in \mathrm{Sp}_L\, \forall a \in \Sigma : \mathrm{CSF}_L(xa) \neq \emptyset \Rightarrow \mathrm{tail}_D(xa) = \mathrm{tail}_L(xa)$

then $\forall x, y \in \mathrm{Sp}(L)$,

1. $\mathrm{next}_{S_n}(x) = \mathrm{next}_L(x)$
2. $\mathrm{right}_{S_n}(xa) = \mathrm{right}_L(xa)$

Proof. Point 1 is clear by definition and point 2 is a consequence of lemma 1

Lemma 3. *With probability one, $\mathrm{next}_{S_n}(x) = \mathrm{next}_L(x)$ and $\mathrm{right}_{S_n}(xa) = \mathrm{right}_L(xa)$ $\forall x \in \mathrm{Sp}(L)$ except for finitely many values of n.*

Proof. Given a SDL language, there exists (at least one) set D with non null probability. Then with probability 1 any sufficiently large sample contains such a set D. is unique for each SDL language. Then the above lemma yields the result.

In order to evaluate the equivalence relation $\text{equiv}(x, y) \iff x \equiv_L y \iff \text{CSF}_L(x) = \text{CSF}_L(y)$ we have to check if two stochastic languages are equivalent from a finite sample S_n.

To do that, instead of comparing the probabilities of each string of the sample, we are going to compare the probabilities of their prefixes. This strategy (also used in ALERGIA [CO94] and RLIPS [CO99]) allows to distinguish different probabilities faster, as more information is always available about a prefix than about a whole string. It is therefore easy to establish the equivalence between the various definitions:

Proposition 3. *Two stochastic languages L_1 and L_2 are equal iff*

$$p(a\Sigma^*|w^{-1}L_1) = p(a\Sigma^*|w^{-1}L_2)\forall a \in \Sigma, \forall w \in \Sigma^*$$

Proof. $L_1 = L_2 \implies \forall w \in \Sigma^* : p(w|L_1) = p(w|L_2) \implies w^{-1}L_1 = w^{-1}L_2 \implies \forall z \subseteq \Sigma^* : p(z|w^{-1}L_1) = p(z|w^{-1}L_2)$

Conversely $L_1 \neq L_2 \implies \exists w \in \Sigma^* : p(w|L_1) \neq p(w|L_2)$. Let $w = az$, as $p(az|L) = p(a\Sigma^*|L)p(z|a^{-1}L)$ then $p(a\Sigma^*|L_1)p(z|a^{-1}L_1) \neq p(a\Sigma^*|L_2)$ $p(z|a^{-1}L_2)$.

Now we have 2 cases:

1. $p(a\Sigma^*|L_1) \neq p(a\Sigma^*|L_2)$ and the proposition is shown.
2. $p(a\Sigma^*|L_1) = p(a\Sigma^*|L_2)$ then $p(z|a^{-1}L_1) \neq p(z|a^{-1}L_2)$.

This can be applyed recursively unless $w = \lambda$.

In such case we have that $\exists w \in \Sigma^* : p(w|L_1) \neq p(w|L_2) \wedge p(w\Sigma^*|L_1) = p(w\Sigma^*|L_2)$. But since $\sum_{x\in\Sigma^*} p(x|L_i) = 1$, it follows that $\exists a \in \Sigma$ such that $p(wa\Sigma^*|L_1) \neq p(wa\Sigma^*|L_2)$. Thus $p(a\Sigma^*|w^{-1}L_1) \neq p(a\Sigma^*|w^{-1}L_2)$.

As a consequence,

$$x \equiv_L y \iff p(a\Sigma^*|(xz)^{-1}L) = p(a\Sigma^*|(yz)^{-1}L)\forall a \in \Sigma, z \in \Sigma^*$$

If instead of the whole language we have a finite sample S_n we are going to estimate the probabilities counting the appearances of the strings and comparing using a confidence range.

Definition 8. *Let f/n be the obseved frequency of a Bernoulli variable of probability p. We denote by $\epsilon_\alpha(n)$ a fuction such that $p(|\frac{f}{n} - p| < \epsilon_\alpha(n)) > 1 - \alpha$ (the Hoeffding bound is one of such functions).*

Lemma 4. *Let f_1/n_1 and f_2/n_2 two obseved frecuencies of a Bernoulli variable of probability p. Then:*

$$p\left(\left|\frac{f_1}{n_1} - \frac{f_2}{n_2}\right| < \epsilon_\alpha(n_1) + \epsilon_\alpha(n_2)\right) > (1-\alpha)^2$$

Proof. $p(|\frac{f_1}{n_1} - \frac{f_2}{n_2}| < \epsilon_\alpha(n_1) + \epsilon_\alpha(n_2)) < p(|\frac{f_1}{n_1} - p| + |\frac{f_2}{n_2} - p| < \epsilon_\alpha(n_1) + \epsilon_\alpha(n_2)) < p(|\frac{f_1}{n_1} - p| < \epsilon_\alpha(n_1) \wedge |\frac{f_2}{n_2} - p| < \epsilon_\alpha(n_2)) < (1-\alpha)^2$

Definition 9.

$$\begin{aligned}\mathrm{equiv}_{S_n}(x,y) \iff & \forall z \in \Sigma^* : xz \in \mathrm{Pref}(S_n) \wedge yz \in \mathrm{Pref}(S_n), \forall a \in \Sigma \\ & \left|\frac{c_n(xza\Sigma^*)}{c_n(xz\Sigma^*)} - \frac{c_n(yza\Sigma^*)}{c_n(yz\Sigma^*)}\right| < \epsilon_\alpha(c_n(xz\Sigma^*)) + \epsilon_\alpha(c_n(yz\Sigma^*)) \wedge \\ & \left|\frac{c_n(xz)}{c_n(xz\Sigma^*)} - \frac{c_n(yz)}{c_n(yz\Sigma^*)}\right| < \epsilon_\alpha(c_n(xz\Sigma^*)) + \epsilon_\alpha(c_n(yz\Sigma^*))\end{aligned}$$

This does not correspond to an infinite number of tests but only to those for which xz or yz is a prefix in S_n. Each of these tests returns the correct answer with probability greater than $(1-\alpha)^2$. Because the number of checks grows with $|\mathrm{Pref}(L)|$ we will allow the parameter α to depend on n.

Theorem 2. *Let the parameter α_n be such that $\sum_{n=0}^{\infty} n\alpha_n$ is finite. Then, with probability one, $(x \equiv_L y) = \mathrm{equiv}_{S_n}(x,y)$ except for finitely many values of n.*

Proof. In order to compute $\mathrm{equiv}_{S_n}(x,y)$ a maximum of $2|\mathrm{Pref}(S_n)|$ tests are made, each with a confidence above $(1-\alpha_n)^2$. Let A_n be the event that at least one of the equivalence tests fails $((x \equiv_L y) \neq \mathrm{equiv}_{S_n}(x,y)$ when using S_n as a sample. Then $\Pr(A_n) < 4\alpha_n|\mathrm{Pref}(S_n)|$. According to the Borel-Cantelli lemma [Fel68], if $\sum_{n=0}^{\infty} \Pr(A_n) < \infty$ then, with probability one, only finitely many events A_n take place. As the expected size of $\mathrm{Pref}(S_n)$ can not grow faster than linearly with n, it is sufficient that $\sum_{n=1}^{\infty} n\alpha_n < \infty$.

5 Discussion and conclusion

We have described a type of stochastic grammars that correspond to a large class of languages including regular languages, palindrome languages, linear LL(1) languages and other typical linear languages such as $\{a^n b^n, 0 \leqslant n\}$. The existence of a canonical form for any grammar in the class is proved, and an algorithm that can learn stochastic deterministic linear grammars is given. This algorithm works in polynomial time and can identify the structure and the probabilities when these are rational (see [dlHT00] for details).

It is nevertheless easy to construct a grammar for which learning is practically doomed: with high probability, not enough examples will be available to notice that some lethal merge should not take place. A counterexample can be constructed by simulating parity functions with a grammar. So somehow the paradigm we are using of polynomial identification in the limit with probability one seems too weak. But on the other hand it is intriguing to notice that the combination of the two criteria of polynomial runtime and identification in the limit with probability one does not seem to result in a very strong condition: it is for instance unclear if a non effective enumeration algorithm might also meet the required standards.It might even be the case that the entire class of context-free grammars may be identifiable in the limit with probability one by polynomial algorithms.

An open problem for which in our mind an answer would be of real help for further research in the field is that of coming up with a new learning criterion for polynomial distribution learning. This should in a certain may better match the idea of polynomial identification with probability one.

References

[Bak79] J. K. Baker. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Soc. of America*, pages 547–550, 1979.

[CO94] R. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In *Proceedings of ICGI'94*, number 862 in LNAI, pages 139–150. Springer Verlag, 1994.

[CO99] R. C. Carrasco and J. Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. *RAIRO (Theoretical Informatics and Applications)*, 33(1):1–20, 1999.

[dlHO02] C. de la Higuera and J. Oncina. Learning deterministic linear languages. In *Proceedings of COLT 2002*, number 2375 in LNAI, pages 185–200, Berlin, Heidelberg, 2002. Springer-Verlag.

[dlHT00] C. de la Higuera and F. Thollard. Identication in the limit with probability one of stochastic deterministic finite automata. In *Proceedings of ICGI 2000*, volume 1891 of *LNAI*, pages 15–24. Springer-Verlag, 2000.

[Fel68] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1 and 2. John Wiley & Sons, Inc., New York, 3rd edition, 1968.

[LS00] P. Langley and S. Stromsten. Learning context-free grammars with a simplicity bias. In *Proceedings of ECML 2000*, volume 1810 of *LNCS*, pages 220–228. Springer-Verlag, 2000.

[NMW97] C. Nevill-Manning and I. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of A. I.Research*, 7:67–82, 1997.

[Sak92] Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97:23–60, 1992.

[SBH+94] Y. Sakakibara, M. Brown, R. Hughley, I. Mian, K. Sjolander, R. Underwood, and D. Haussler. Stochastic context-free grammars for trna modeling. *Nuclear Acids Res.*, 22:5112–5120, 1994.

[WA02] Y. Wang and A. Acero. Evaluation of spoken language grammar learning in the atis domain. In *Proceedings of ICASSP*, 2002.

[YLT00] M. Young-Lai and F. W. Tompa. Stochastic grammatical inference of text database structure. *Machine Learning*, 40(2):111–137, 2000.

6 Appendix

Propositions from section 3 aim at establishing that a small canonical form exists for each SDL grammar. The following proofs follow the ideas from [dlHO02].

6.1 Proof of proposition 1

In order to prove the propositions we have to establish more definitions.

To define another equivalence relation over Σ^*, when given a stochastic deterministic linear grammar, we first associate in a unique way prefixes of strings in the language with non-terminals:

Definition 10. *Let $G = (\Sigma, V, R, S, p)$ be a SDL grammar. With every string x we associate the unique non terminal $[x]_G = T$ such that $S \xrightarrow{*} xTu$; we extend L_G to be a total function by setting $L_G([x]_G) = \emptyset$ the non terminal T doen not exists.*

We use this definition to give another equivalence relation over Σ^*, when given a SDL grammar:

Definition 11. *Let $G = (\Sigma, V, R, S, p)$ be a SDL grammar. We define the associated* common suffix-free *languages* $\mathrm{CSF}_G(.)$*, and associated equivalence relation as follows:*

$$\mathrm{CSF}_G(\lambda) = L_G(S) \qquad\qquad \mathrm{CSF}_G(xa) = L_G([xa]_G)\downarrow \qquad\Big|\qquad x \equiv_G y \iff \mathrm{CSF}_G(x) = \mathrm{CSF}_G(y)$$

$\equiv_G$ is clearly an equivalence relation, in which all strings x such that $[x]_G$ is undefined are in a unique class. The following lemma establishes that $\equiv_G$ has finite index, when G is a stochastic deterministic linear grammar:

Lemma 5. *If $[x]_G = [y]_G$, $x \neq \lambda$ and $y \neq \lambda \Rightarrow x \equiv_G y$. Hence if G contains n non-terminals, $\equiv_G$ has at most $n + 2$ classes.*

The proof is straightforward. There can be at most two possible extra classes corresponding to λ (when it is alone in its class) and the undefined class

Lemma 6. *Let $G = (\Sigma, V, R, S, p)$ be a SDL grammar. If $X \xrightarrow{*} xYw$ then:*

$$(x^{-1}L(X))\downarrow = L(Y)\downarrow$$

Proof. It is enough to prove $(a^{-1}L(X))\downarrow = L(Y)\downarrow$ if $X \to aYw \in R$, which is clear by double inclusion.

Proposition 4. *Let $G = (\Sigma, V, R, S, p)$ be a SDL grammar, and denote $L = L_G(S)$. $\forall x \in \Sigma^*$, either $\mathrm{CSF}_L(x) = \mathrm{CSF}_G(x)$ or $\mathrm{CSF}_L(x) = \emptyset$*

Proof. By induction on the length of x.

Base: $x = \lambda$, then $\mathrm{CSF}_L(x) = L = \mathrm{CSF}_G(x)$.

Suppose: the proposition is true for all strings of length up to k, so consider string xa of length $k+1$. $\mathrm{CSF}_L(xa) = (a^{-1}\,\mathrm{CSF}_L(x))\downarrow$ (by definition 2).
If $\mathrm{CSF}_L(x) = \emptyset$, $\mathrm{CSF}_L(xa) = \emptyset$. If not ($\mathrm{CSF}_L(x) = \mathrm{CSF}_G(x)$) by induction hypothesis, $\mathrm{CSF}_L(xa) = (a^{-1}\,\mathrm{CSF}_L(x))\downarrow = (a^{-1}\,\mathrm{CSF}_G(x))\downarrow$ and there are two sub-cases:

if $x = \lambda$ $\mathrm{CSF}_G(x) = L_G([x]_G)$, so $\mathrm{CSF}_L(xa) = (a^{-1}L_G([x]_G))\downarrow$

if $x \neq \lambda$ $\mathrm{CSF}_G(x) = L_G([x]_G)\downarrow$, so: $CSF_L(xa) = (a^{-1}(L_G([x]_G)\downarrow))\downarrow$ (by definition 11), $=(a^{-1}(L_G([x]_G)))\downarrow$

In both cases follows: $CSF_L(xa) = (a^{-1}L_G([x]_G))\downarrow = L_G([xa]_G)\downarrow$ (by lemma 6)$=\mathrm{CSF}_G(xa)$.

Corollary 1 (proof of proposition 1). *Let $G = (\Sigma, V, R, S, p)$ be a stochastic deterministic linear grammar. So $\equiv_{L_G(S)}$ has finite index.*

Proof. A consequence of lemma 5 and proposition 4:

6.2 Proof of proposition 2

To avoid extra notations, we will denote (as in definition 10) by $[x]$ the non-terminal corresponding to x in the associated grammar (formally $S_{\mathrm{CSF}_L(x)}$ or $[x]_{G_L}$).

The proof that G_L generates L is established through the following more general result (as the special case where $x = \lambda$):

Proposition 5. $\forall x \in \Sigma^*, L_{G_L}([x]) = \mathrm{CSF}_L(x)$.

Proof. We prove it by double inclusion.

$\forall x \in \Sigma^*,\ \mathrm{CSF}_L(x) \subseteq L_{G_L}([x])$

Proof by induction on the length of all strings in $\mathrm{CSF}_L(x)$.

Base case $|w| = 0 \Rightarrow w = \lambda$. If $(\lambda, p) \in \mathrm{CSF}_L(x)$, by construction of the rules, $[x] \rightarrow \lambda$ and $p([x] \rightarrow \lambda) = p$ so $(\lambda, p) \in L_{G_L}([x])$.

Suppose now (induction hypothesis) that
$\forall x \in \Sigma^*, \forall w \in \Sigma^{\leqslant k}$: $(w,p) \in \mathrm{CSF}_L(x) \Rightarrow (w,p) \in L_{G_L}([x])$.
Let $w = auv$ such that $|w| = k+1$, $(auv, p) \in \mathrm{CSF}_L(x)$ and let $v = \mathrm{lcs}(a^{-1}\,\mathrm{CSF}_L(x))$. As $\mathrm{CSF}_L(xa) = (a^{-1}\,\mathrm{CSF}_L(x))\downarrow$, then $\exists p_u$: $(u, p_u) \in \mathrm{CSF}_L(xa)$ and then $p = p_u p(a\Sigma^* | \,\mathrm{CSF}_L(x))$. As by construction $[x] \rightarrow a[xa]v$ and $p([x] \rightarrow a[xa]v) = p(a\Sigma^*| \,\mathrm{CSF}_L(x))$ and, by hypothesis induction ($|u| \leqslant k$) $(u, p_u) \in L_G([xa])$, then $(auv, p) \in L_G([x])$.

$\forall x \in \Sigma^*,\ L_{G_L}([x]) \subseteq \mathrm{CSF}_L(x)$

Proof by induction on the order (k) of the derivation

$\forall x \in \Sigma^*$, $\forall k \in \mathbb{N}$, $\forall w \in \Sigma^*$, $[x] \xrightarrow{k} w \Rightarrow (w, p([x] \xrightarrow{k} w) \in \mathrm{CSF}_L(x)$.

Base case $[x] \xrightarrow{1} w$. This case is only possible if $w = \lambda$. And, by construction, such a rule is in the grammar because $(\lambda, p(\lambda| \,\mathrm{CSF}_L(x)) \in \mathrm{CSF}_L(x)$

Suppose now (induction hypothesis) that for any $n \leqslant k$:

$\forall x \in \Sigma^*, \forall w \in \Sigma^* : [x] \xrightarrow{n} w \Rightarrow \exists p : (w, p) \in \mathrm{CSF}_L(x)$

Take $w \in \Sigma^*$ such that $[x] \xrightarrow{k+1} w$, then $[x] \to a[xa]v \xrightarrow{k} w = auv$ with $[xa] \xrightarrow{k} u$, and $p = p([x] \to a[xa]v)p_u$ where $p_u = p([xa] \xrightarrow{k} u)$, by induction hypothesis we know that $(u, p_u) \in \mathrm{CSF}_L(xa) = (a^{-1}\,\mathrm{CSF}_L(x)) \downarrow = \{(t, p_t) : (atv, p_a p_t) \in \mathrm{CSF}_L(x), p_a = p(a\Sigma^* | \mathrm{CSF}_L(x)), v = \mathrm{lcs}(a^{-1}\,\mathrm{CSF}_L(x))\}$. As by construction we know that $p([x] \to a[xa]v) = p(a\Sigma^* | \mathrm{CSF}_L(x))$ then $(w, p) = (auv, p([x] \to a[xa]v)p_u) \in \mathrm{CSF}_L(x)$.