Learning Context Free Grammars in the Limit Aided by the Sample Distribution

Yoav Seginer

ILLC, Universiteit van Amsterdam
yseginer@science.uva.nl

Abstract. We present an algorithm for learning context free grammars from positive structural examples (unlabeled parse trees). The algorithm receives a parameter in the form of a finite set of structures and the class of languages learnable by the algorithm depends on this parameter. Every context free language belongs to many such learnable classes. A second part of the algorithm is then used to determine this parameter (based on the language sample). By Gold's theorem, without introducing additional assumptions, there is no way to ensure that, for every language, the parameter chosen by the learner will make the language learnable. However, we show that determining the parameter based on the sample distribution is often reasonable, given some weak assumptions on this distribution. Among other things, repeated learning, where one learner learns the language the previous learner converged to, is guaranteed to produce a learnable language after a finite number of steps. This set of limit languages then forms a natural class of learnable languages.

1 Introduction

The problem of learning context free grammars (in the limit) from structural examples has attracted much attention ([6],[7],[2],[4]). A structural example for a sentence consists of the parse tree structure of that sentence without the non-terminal labels. Also known as a tree language, the parse trees of a context free language can be characterized by a tree automaton ([6],[7]). This allows methods for learning regular languages to be extended to the learning of context free languages from structural examples (see [6],[7],[2]).

When Gold [3] first presented his paradigm of learning in the limit, he also showed that within this paradigm the class of context free grammars is not learnable from positive examples. While the use of structural examples does seem to make the learning problem easier, it is still not enough to get around Gold's theorem and (just as with the class of regular languages) the class of context free languages is not learnable from structural examples. For some, this is sufficient reason to reject Gold's learning paradigm altogether. Others, who still want to apply this paradigm to the learning of context free grammars, may resort to two methods. The first is to supply the learner with some additional information, the other, to restrict the class of languages which has to be learned to a subclass of the context free languages.

Sakakibara presented solutions of both types. In [6], the learner is allowed to use structural equivalence and membership queries, thus gaining additional information about the language, while in [7] only positive examples are used but the algorithm is restricted to the class of reversible context free grammars. The two approaches can also be combined, of course. Fernau [2] extended [7] to learning the δ-distinguishable tree languages. Every choice of a distinguishing function δ results in a learnable class. For any language, when the learning algorithm is equipped with an appropriate distinguishing function, it is guaranteed to learn the language correctly from positive examples. Clearly, by Gold's theorem, to guarantee a correct choice of the distinguishing function δ for every language, the learner must make use of some information beyond the positive examples for the language.

The algorithm we present in this paper also learns context free grammars from structural examples. The learning algorithm, which is very simple and natural, uses a finite set of structures, the *context set*. Similarly to Fernau's distinguishing functions, every context set makes a subclass of the context free languages learnable and together these subclasses cover the whole class of context free grammars. Another property shared with Fernau's algorithm is that even when a language is outside the class of languages learnable by the algorithm (for a given context set), the algorithm converges to a language containing (over-generalizing) the original language. The resulting language does not depend on the order in which the language is presented to the learner (compare this with the approximation property in [1]). Beyond these similarities, the learning algorithm itself is of a completely different nature, as the ability to distinguish trees, which is given as an oracle (the distinguishing function) to Fernau's algorithm, does not exist for our algorithm (until the learning process actually converges). The workings of our algorithm can in some ways be compared with the family of tail algorithms presented in [5], even though these algorithms are not learning algorithms in the sense of Gold (as convergence in the limit is not guaranteed).

In addition to presenting the basic algorithm (section 2) and proving its convergence (section 3) we present (in section 5) a method for choosing the context set to be used, based on the sample being presented to the learner. It relies on the fact that even when the learning process is confined to receiving positive examples, the learner has access not only to the positive examples themselves, but also to the order and frequencies in which they appear. Whether this contains any information about the language being learned depends on the way in which the examples are generated. Usually, when learning within Gold's paradigm of identification in the limit, the only assumption one makes about the process generating the examples is that every sentence will eventually be generated by it. It has often been observed that this is a very strong requirement (even in Gold's original paper some alternatives are examined).

The method presented here constructs the context set out of the most frequent structures in (some initial segment of) the language sample. Such frequent structures are clearly evident in natural languages and it has already been observed by linguists [8] that these structures may play an important role in language acquisition by children. Whether this procedure produces a good context set for the language to be learned depends on the settings in which we wish to apply the algorithm and is usually an empirical question. There are, however, several properties of the learning algorithm which make this method a good candidate. We discuss this in section 5.

A basic property shown in sections 4 and 5 is that there is a natural process which leads to learnable languages and if a language becomes unlearnable (for whatever reason) there is a way of returning to a learnable language (even if a somewhat different one). Beginning with any language, repeated learning (that is, every learner learning the language which the previous learner converged to) is bound to arrive (after a finite number of steps) at a language learnable by all subsequent learners. The class of these limit languages is then a very natural learnable class.

2 The Learning Algorithm

The learning algorithm receives *trees* as input and produces a set of rules (the grammar) as output. It learns, therefore, from structural examples. The algorithm creates grammar rules by looking at subtrees of the sample trees and the context trees in which they appear. We begin by defining all these structures.

A *tree* T is either a single terminal $\sigma \in \Sigma$ or a tuple $\langle T_1, \ldots, T_l \rangle$ $(1 \leq l)$ where $T_1, \ldots, T_l$ are trees. We write $\mathcal{T}(\Sigma)$ for the set of all trees over the terminal set Σ. A *context* is a tree in which a single leaf can be substituted by a tree to create a tree. We assume that $* \notin \Sigma$ ($*$ indicates the leaf in the context where a tree can be substituted). A context c is either the symbol $*$ or $c = \langle T_1, \ldots, T_{i-1}, c', T_{i+1}, \ldots, T_l \rangle$ $(1 \leq l,\ 1 \leq i)$ where c' is a context and $T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_l \in \mathcal{T}(\Sigma)$. We write $\mathcal{CT}(\Sigma)$ for the set of all contexts over the terminal set Σ.

Given a context c and a tree T we write $c(T)$ for the tree created by substituting T for the (unique) leaf labeled $*$ in c. Given a tree T and a node ν in this tree, we can always write $T = c(S)$ where S is the subtree rooted at the node ν (when ν is the root of the tree T, $c = *$). A tree S is a *subtree* of tree T if there exists a context c such that $T = c(S)$. We define $\mathcal{S}(T) = \{S \in \mathcal{T}(\Sigma) : \exists c \in \mathcal{CT}(\Sigma) \text{ s.t. } T = c(S)\}$, the set of subtrees of T. Given a set of trees X (e.g. a language L), the set of subtrees of X is $\mathcal{S}(X) = \bigcup_{T \in X} \mathcal{S}(T)$.

Central to the operation of the algorithm is a *finite* context set $\mathcal{C} \subseteq \mathcal{CT}(\Sigma)$. This context set $\mathcal{C}$ is allowed to change a finite number of times as the language is being presented to the learner, so we have a finite sequence of context sets $\mathcal{C}_1, \ldots, \mathcal{C}_n$. However, in what follows we will assume that the last context set, $\mathcal{C}_n$, is used throughout the presentation of the language sample. This is a legitimate assumption, since, in the worst case, each time the context set changes, the algorithm can rerun all previous learning steps using the updated context set. For the algorithm presented here it will become clear that much less is needed. The way in which the context set sequence $\mathcal{C}_1, \ldots, \mathcal{C}_n$ is determined will be discussed later, in section 5. Therefore, from now on we will assume a finite context set $\mathcal{C}$ which is fixed at the beginning of the learning process. It is also required that $* \in \mathcal{C}$.

To construct grammar rules, the algorithm uses the partially ordered set $\mathcal{O} = (2^{\mathcal{C}} \cup \Sigma, \leq)$ with the subset ordering on the elements of $2^{\mathcal{C}}$ (the elements in Σ are neither comparable with elements in $2^{\mathcal{C}}$ nor with each other). Every grammar rule is then of the form $(y|x_1, \ldots, x_l)_d$ where $y \in 2^{\mathcal{C}}$, $x_1, \ldots, x_l \in \mathcal{O}$, $1 \leq l$ and $0 \leq d$. The natural number d is the *level* of the rule.

Let $\mathcal{R}$ be a set of such rules. We define the function $\mathrm{ctx}^{\mathcal{R}}(T)$ which maps a tree T into an element in $\mathcal{O}$ based on the rules in the rule set $\mathcal{R}$. If $T = \sigma \in \Sigma$ then $\mathrm{ctx}^{\mathcal{R}}(T) = \sigma$. Otherwise $T = \langle T_1, \ldots, T_l \rangle$ and $\mathrm{ctx}^{\mathcal{R}}(T) = \bigcup \{y : \exists (y|x_1, \ldots, x_l)_k \in \mathcal{R} \ s.t. \ k \leq \mathrm{depth}(T), x_i \leq \mathrm{ctx}(T_i), 1 \leq i \leq l\}$ where $\mathrm{depth}(T)$ is defined recursively as $\mathrm{depth}(T) = 1 + \max_{i=1,\ldots,l} \mathrm{depth}(T_i)$ and $\mathrm{depth}(\sigma) = 0$ (for a terminal $\sigma \in \Sigma$). The language $L(\mathcal{R})$ generated by a rule set $\mathcal{R}$ is defined to be $L(\mathcal{R}) = \{T \in \mathcal{T}(\Sigma) \ : \ * \in \mathrm{ctx}^{\mathcal{R}}(T)\}$.

The algorithm maintains a set of pre-rules, $\mathcal{P}$. A pre-rule is of the form $(y|T_1, \ldots, T_l)$ where $y \in 2^{\mathcal{C}}$ and $T_1, \ldots, T_l$ are trees. A projection $\pi^{\mathcal{R}}$ from pre-rules to rules, based on the rule-set $\mathcal{R}$, is defined. If the pre-rule P is $(y|T_1, \ldots, T_l)$ then $\pi^{\mathcal{R}}(P) = (y|\mathrm{ctx}^{\mathcal{R}}(T_1), \ldots, \mathrm{ctx}^{\mathcal{R}}(T_l))_{\mathrm{depth}(\langle T_1, \ldots, T_l \rangle)}$. The number $\mathrm{depth}(\langle T_1, \ldots, T_l \rangle)$ is the *depth* of the pre-rule P, which is equal to the level of the rule $\pi^{\mathcal{R}}(P)$.

The algorithm is initialized with an empty set of pre-rules $\mathcal{P}$ and may change this pre-rule set at every step. For every pre-rule set $\mathcal{P}$, the rule set hypothesized by the algorithm is the unique rule set $\mathcal{R}$ such that $\mathcal{R} = \pi^{\mathcal{R}}(\mathcal{P}) = \left\{\pi^{\mathcal{R}}(P) \ : \ P \in \mathcal{P}\right\}$. This rule set is easy to calculate from $\mathcal{P}$ because of the level assigned to each rule. When computing $\pi^{\mathcal{R}}(P)$ for a a pre-rule of depth d, the function $\mathrm{ctx}^{\mathcal{R}}$ makes use only of rules of a level strictly smaller than d. Therefore, the computation can advance in a straightforward manner through $\mathcal{P}$, in increasing order of depth.

We now define an ordering of the rules by $(y|x_1, \ldots, x_l)_d \leq (z|w_1, \ldots, w_m)_k$ iff $l = m$, $x_i \leq w_i$ for $i = 1, \ldots, l$, $y \geq z$ and $d \leq k$. After having calculated the rule set $\mathcal{R}$ associated with a pre-rule set $\mathcal{P}$, the algorithm removes from $\mathcal{P}$ any pre-rule $P \in \mathcal{P}$ such that the rule $\pi^{\mathcal{R}}(P)$ is not minimal in $\mathcal{R}$. Removing pre-rules from $\mathcal{P}$ entails the removal of rules in $\mathcal{R}$. However, it is easy to see that, because of the way the rule ordering is defined, this does not change the function $\mathrm{ctx}^{\mathcal{R}}$ and there is therefore no need to recalculate the rule set.

When presented a sample tree S, the algorithm traverses all its nodes (except for the leaves) in increasing order of depth (the depth of a node being the depth of the subtree rooted at it). For each such node ν, the tree S can be written as $S = c(T)$, where T is the subtree rooted at ν and c is an appropriate context. Let $T = \langle T_1, \ldots, T_l \rangle$ (ν is not a leaf) and let $\mathcal{P}$ and $\mathcal{R}$ be the pre-rule set and corresponding rule set as calculated by the algorithm up to this point. The algorithm then performs the following operations:

1. Check whether there is a rule $R \in \mathcal{R}$ such that $R \leq \pi^{\mathcal{R}}(\mathcal{C} \cap \{c\}|T_1, \ldots, T_l)$. If there is such a rule in $\mathcal{R}$ then go on to the next node.

2. Otherwise, if there exists already a pre-rule $(y|T_1, \ldots, T_l)$ in $\mathcal{P}$ then this pre-rule is replaced by the pre-rule $(y \cup (\mathcal{C} \cap \{c\})|T_1, \ldots, T_l)$. Otherwise, the pre-rule $(\mathcal{C} \cap \{c\}|T_1, \ldots, T_l)$ is added (as is) to $\mathcal{P}$.
3. After the pre-rule set is updated, the corresponding rule set is calculated and all non-minimal rules (and corresponding pre-rules) are removed.

3 Convergence of the Algorithm

Given any finite context set $\mathcal{C}$ and any context free language L, the algorithm will be shown to converge to a language $L^{\mathcal{C}}$, which is independent of the specific language sample which is presented to the algorithm and such that $L \subseteq L^{\mathcal{C}}$. Not only the language but also the grammar (rule set) hypothesized by the algorithm will be shown to converge. However, the exact grammar which the algorithm converges to may depend on the order in which the examples are presented to it.

From now on we assume that the context set $\mathcal{C}$ has been fixed. We begin by constructing a rule set $\mathcal{R}^L$ which will be shown to generate the language $L^{\mathcal{C}}$ to which the algorithm converges. The algorithm need not necessarily converge to the same rule set, but will be shown to converge to an equivalent one. We define the set $\mathcal{P}^L = \{(\{c \in \mathcal{C} \, : \, c(T) \in L\}|T_1, \ldots, T_l) \, : \, T = \langle T_1, \ldots, T_l \rangle \in \mathcal{S}(L)\}$ of pre-rules and then define $\hat{\mathcal{R}}^L$ to be the unique rule set such that $\hat{\mathcal{R}}^L = \pi^{\hat{\mathcal{R}}^L}(\mathcal{P}^L)$. Finally, we define $\mathcal{R}^L$ to be the set of minimal elements in $\hat{\mathcal{R}}^L$. Because the language L is generated by a finite context free grammar, there is a bound on the degree of nodes of the trees in L. From this, together with the fact that $\mathcal{C}$ is finite, it follows that the rule set $\mathcal{R}^L$ is finite.

Lemma 1. *For any finite context set $\mathcal{C}$ and context free language L, $L \subseteq L(\mathcal{R}^L)$.*

Proof. It follows from the definition of the rule ordering that $\mathcal{R}^L$ and $\hat{\mathcal{R}}^L$ generate the same language. By the construction of $\hat{\mathcal{R}}^L$, for every tree T, $\{c \in \mathcal{C} \, : \, c(T) \in L\} \subseteq \mathrm{ctx}^{\hat{\mathcal{R}}^L}(T)$. This proves the lemma, since $*(T) = T \in L$ implies $* \in \mathrm{ctx}^{\hat{\mathcal{R}}^L}(T)$ and then, by definition, $T \in L(\hat{\mathcal{R}}^L) = L(\mathcal{R}^L)$. $\square$

We write ctx^L for the function $\mathrm{ctx}^{\mathcal{R}^L}$. For every rule $R = (y|x_1, \ldots, x_l)_d \in \mathcal{R}^L$ and for every $c \in y$, we define the representative class $CL(R, c) = \{S \in L \, : \, S = c(\langle T_1, \ldots, T_l \rangle) \in L, \mathrm{ctx}^L(T_i) = x_i, 1 \leq i \leq l, \mathrm{depth}(\langle T_1, \ldots, T_l \rangle) = d\}$. There are finitely many such classes. We call a sample $\bar{S} = \{S^i\}_{i=1}^{\infty}$ *class fat*, if it contains infinitely many trees from $CL(R, c)$ for every such class. To fulfill this condition, trees from each class may be repeated.

The only condition we impose on the language sample is that it be class fat. Since the algorithm can simply remember all examples presented to it and repeat the calculations on them as needed, the class fat sample requirement can be replaced, at the expense of increased memory and time resources, by the requirement that at least one tree from each class appear in the sample. This last condition is even weaker than the standard assumption that all sentences in the language appear in the sample. We prefer to assume the repetition of the necessary examples because the repetition seems reasonable in many settings and reduces the resources required by the algorithm.

We will usually assume some fixed language sample $\bar{S} = \{S^i\}_{i=1}^{\infty}$. Relative to this sample, we wish to examine the pre-rule and rule sets generated by the algorithm at each step. We therefore write $\mathcal{P}^n$ and $\mathcal{R}^n$ for the pre-rule set and the corresponding rule set as generated by the algorithm at the end of the n'th step (one algorithm step is applied for every non-leaf node of a sample tree and so several steps may be needed for the processing of a single sample tree).

The rule set will be shown to converge level by level, beginning with the lowest level rules. We therefore write $\mathcal{R}_k^n$ for the rules of depth at most k in $\mathcal{R}^n$ and similarly $\mathcal{R}_k^L$ for rules of level at most k in $\mathcal{R}^L$. We also write $\mathrm{ctx}_k^L = \mathrm{ctx}^{\mathcal{R}_k^L}$, $\mathrm{ctx}^n = \mathrm{ctx}^{\mathcal{R}^n}$ and $\mathrm{ctx}_k^n = \mathrm{ctx}^{\mathcal{R}_k^n}$. The first lemma shows convergence of the rule set for any given level.

Lemma 2. *For any context free language L, any finite context set $\mathcal{C}$, any class fat sample $\bar{S}$ of L and for any $0 \leq k$ there is a number N_k such that for every $N_k \leq n$, $\mathcal{R}_k^{N_k} = \mathcal{R}_k^n$ and for every tree T, $\mathrm{ctx}_k^L(T) = \mathrm{ctx}_k^n(T)$.*

Proof. By induction on k. For $k = 0$, the claim is immediate from the definitions. We now assume the claim for k and prove it for $k+1$. Let $N_k < n$ and let $R = (y|x_1, \ldots, x_l)_d \in \mathcal{R}_{k+1}^n$. There exists some $S = \langle S_1, \ldots, S_l \rangle \in \mathcal{S}(L)$ of depth $d \leq k+1$ such that $x_i = \mathrm{ctx}^n(S_i)$ $(1 \leq i \leq l)$ and $y \subseteq \{c \in \mathcal{C} \ : \ c(S) \in L\}$. Since $\mathrm{depth}(S) \leq k+1$ and by the induction hypothesis, $x_i = \mathrm{ctx}^n(S_i) = \mathrm{ctx}_k^n(S_i) = \mathrm{ctx}_k^L(S_i) = \mathrm{ctx}^L(S_i)$ $(1 \leq i \leq l)$. Therefore, $(\{c \in \mathcal{C} \ : \ c(S) \in L\}|x_1, \ldots, x_l)_d \in \hat{\mathcal{R}}^L$ and it follows that there is a rule $R' \in \mathcal{R}^L$ such that $R' \leq R$. As this is true for any rule $R \in \mathcal{R}_{k+1}^n$, $\mathrm{ctx}_{k+1}^n(T) \leq \mathrm{ctx}_{k+1}^L(T)$ for any tree T.

We next show that for every tree T there is an $n(T)$ such that for any $n(T) \leq n$, $\mathrm{ctx}_{k+1}^n(T) = \mathrm{ctx}_{k+1}^L(T)$. The proof is by induction on the depth of the tree T. For $T = \sigma \in \Sigma$ (tree of depth 0) the claim is immediate from the definitions. We assume the claim for d and prove it for a tree $T = \langle T_1, \ldots, T_l \rangle$ of depth $d+1$. It is immediate from the way the algorithm works and the induction hypothesis on k that if $N_k \leq n < m$ and $R \in \mathcal{R}_{k+1}^n$ then there is a rule $R' \in \mathcal{R}_{k+1}^m$ such that $R' \leq R$. Therefore, for every tree T, the sequence $\{\mathrm{ctx}_{k+1}^n(T)\}_{N_k \leq n}$ is monotonically increasing (in $\mathcal{O}$). It remains to show that if $c \in \mathrm{ctx}_{k+1}^L(T)$ then there is an $N_k \leq n$ such that $c \in \mathrm{ctx}_{k+1}^n(T)$. By the class fat sample assumption, since $c \in \mathrm{ctx}_{k+1}^L(T)$, there must be a tree $S = \langle S_1, \ldots, S_l \rangle \in \mathcal{S}(L)$ such that $c(S) \in L$, $\mathrm{depth}(S) \leq \min(k+1, d+1)$, $\mathrm{ctx}^L(S_i) \leq \mathrm{ctx}_{k+1}^L(T_i)$ $(1 \leq i \leq l)$ and there is some $N_k, N(T_1), \ldots, N(T_l) < n$ such that at step n, the algorithm tries to add the pre-rule $(\{c\}|S_1, \ldots, S_l)$. Therefore, after this step, there is $R \in \mathcal{R}_{k+1}^n$ such that $R \leq (\{c\}|\mathrm{ctx}^n(S_1), \ldots, \mathrm{ctx}^n(S_l))_{\mathrm{depth}(S)}$. By what has been shown in the previous paragraph, the assumptions on S and the induction hypothesis on d, $\mathrm{ctx}^n(S_i) = \mathrm{ctx}_{k+1}^n(S_i) \leq \mathrm{ctx}_{k+1}^L(S_i) = \mathrm{ctx}^L(S_i) \leq \mathrm{ctx}_{k+1}^L(T_i) = \mathrm{ctx}_{k+1}^n(T_i)$ and therefore the rule R applies in calculating $\mathrm{ctx}_{k+1}^n(T)$ and $c \in \mathrm{ctx}_{k+1}^n(T)$, which completes the proof of the claim.

Since there is a finite number of trees of depth at most $k+1$, there is an N such that for every $N \leq n$ and every tree T of depth at most $k+1$, $\mathrm{ctx}^n_{k+1}(T) = \mathrm{ctx}^L_{k+1}(T)$. For step $N < n$, if the algorithm tries to add a pre-rule $(y|S_1, \ldots, S_l)$ for $S = \langle S_1, \ldots, S_l \rangle$ of depth at most $k+1$ then, since $\{c \in \mathcal{C} \ : \ c(S) \in L\} \subseteq \mathrm{ctx}^L(S) = \mathrm{ctx}^L_{k+1}(S)$, $y \subseteq \mathrm{ctx}^L_{k+1}(S) = \mathrm{ctx}^{n-1}_{k+1}(S)$. Since also $|y| \leq 1$, there must be a rule $R \in \mathcal{R}^{n-1}_{k+1}$ such that $R \leq \pi^{\mathcal{R}^{n-1}}((y|S_1, \ldots, S_l))$ and the pre-rule is not added. It follows that $\mathcal{R}^n_{k+1} = \mathcal{R}^N_{k+1}$ for any $N \leq n$. Since for every tree T there is an $N(T)$ such that, for $N(T) \leq n$, $\mathrm{ctx}^n_{k+1}(T) = \mathrm{ctx}^L_{k+1}(T)$ and since the rules of level at most $k+1$ do not change after step N, it follows that $N(T) \leq N$ for every tree T. Therefore, we can take $N_{k+1} = N$. $\square$

Based on the convergence of levels we can now prove the convergence of the whole rule set.

Theorem 1. *For any context free language L, any finite context set $\mathcal{C}$ and any class fat sample $\bar{S}$ of L, the algorithm converges to a rule set $\mathcal{R}$ such that $L \subseteq L(\mathcal{R}^L) = L(\mathcal{R})$.*

Proof. That $L \subseteq L(\mathcal{R}^L)$ is a restatement of Lemma 1. We therefore show that $L(\mathcal{R}^L) = L(\mathcal{R})$. Let k be the level of the highest level rule in $\mathcal{R}^L$. Using Lemma 2 we take $N = N_k$. Let $N \leq n$ and assume that at step $n+1$, the algorithm attempts to add the pre-rule $P = (y|S_1, \ldots, S_l)$. Writing $S = \langle S_1, \ldots, S_l \rangle$ we have, by definition, that $y \subseteq \{c \in \mathcal{C} \ : \ c(S) \in L\}$ and (as in the proof of Lemma 1) $y \subseteq \mathrm{ctx}^L(S)$. By the choice of k and Lemma 2, $y \subseteq \mathrm{ctx}^L(S) = \mathrm{ctx}^L_k(S) = \mathrm{ctx}^n_k(S) \subseteq \mathrm{ctx}^n(S)$. Therefore, since $|y|$ is either 0 or 1, there is a rule $R = (z|x_1, \ldots, x_l)_d \in \mathcal{R}^n$ such that $y \subseteq z$ and $x_i \leq \mathrm{ctx}^n(S_i)$ $(1 \leq i \leq l)$ and $d \leq \mathrm{depth}(S)$. In other words, $R \leq \pi^{\mathcal{R}^n}(P)$ and the algorithm does not add the pre-rule. Therefore, the pre-rule set does not change. This shows that the rules set converges at step N_k. To complete the proof, it remains to prove that for any tree T, $\mathrm{ctx}^{N_k}(T) = \mathrm{ctx}^L(T)$. This follows immediately from Lemma 2, because there are finitely many rules in $\mathcal{R}^{N_k}$ and applying Lemma 2 for the level of the highest level rule in $\mathcal{R}^{N_k}$ gives the required equality (notice that $\mathcal{R}^{N_k}$ may contain rules of level higher than k). $\square$

Our next step is to prove that for any context free language L there is a context set $\mathcal{C}$ such that $L^{\mathcal{C}} = L$. We will actually show that for every finite set of languages we can find a context set which makes them all learnable. From the proof it will immediately be clear that for every language L there are many different context sets which guarantee convergence to L. As a result, an algorithm for finding the context set $\mathcal{C}$ enjoys a considerable amount of flexibility.

Theorem 2. *For every finite set $\mathcal{L}$ of context free languages, there is a finite set of contexts $\mathcal{C}^{\mathcal{L}}$ such that for any finite set of contexts $\mathcal{C}^{\mathcal{L}} \subseteq \mathcal{C}$ and any $L \in \mathcal{L}$, $L^{\mathcal{C}} = L$.*

Proof. It is enough to prove the theorem for $|\mathcal{L}| = 1$ since then we can take $\mathcal{C}^{\mathcal{L}} = \bigcup_{L \in \mathcal{L}} \mathcal{C}^{\{L\}}$. Fix a context free language L. For every tree T, we define the set $C(T) = \{c \in \mathcal{CT}(\Sigma) \ : \ c(T) \in L\}$. Since the language L is generated by a finite number of rules, the sub-trees $\mathcal{S}(L)$ of L can be assigned only finitely many different types by the context free grammar which generates L. Since two trees of the same type appear in the same contexts in the language L, there are only finitely many different sets $C(T)$. Let $C_1, \ldots, C_h$ be these different sets. The set $\mathcal{C}^{\{L\}}$ is constructed by taking, for every $i \neq j$, a context $c \in C_i \setminus C_j$ (if such a context exists). We also add $*$ to $\mathcal{C}^{\{L\}}$ (if it is not already there). From now on we fix some set $\mathcal{C}$ as the context set for the algorithm, such that $\mathcal{C}^{\{L\}} \subseteq \mathcal{C}$.

We will show that for every tree T of depth at least 1, $\mathrm{ctx}^L(T) = \mathcal{C} \cap C(T)$. This proves the theorem because trees of depth zero cannot be in any language and for trees of depth at least 1, $T \in L^{\mathcal{C}} \iff T \in L(\mathcal{R}^L) \iff * \in \mathrm{ctx}^L(T) \iff * \in C(T) \iff T \in L$ and therefore $L^{\mathcal{C}} = L$.

If $T = \langle T_1, \ldots, T_l \rangle \in \mathcal{S}(L)$ then, since $\{c \in \mathcal{C} \ : \ c(T) \in L\} = \mathcal{C} \cap C(T)$, there is a rule $(\mathcal{C} \cap C(T) | \mathrm{ctx}^L(T_1), \ldots, \mathrm{ctx}^L(T_l))_{\mathrm{depth}(T)} \in \hat{\mathcal{R}}^L$ and it follows that $\mathcal{C} \cap C(T) \subseteq \mathrm{ctx}^L(T)$. If $T \notin \mathcal{S}(L)$ then $\mathcal{C} \cap C(T) \subseteq \mathrm{ctx}^L(T)$ trivially, since $C(T) = \emptyset$.

It remains to show that $\mathrm{ctx}^L(T) \subseteq \mathcal{C} \cap C(T)$. Since, by definition, $\mathrm{ctx}^L(T) \subseteq \mathcal{C}$, we show (by induction on the depth of T) that $\mathrm{ctx}^L(T) \subseteq C(T)$. The claim is immediate for a tree of depth 1, since in this case $T = \langle \sigma_1, \ldots, \sigma_l \rangle$ where $\sigma_1, \ldots, \sigma_l \in \Sigma$ and there can be at most one rule in $\hat{\mathcal{R}}^L$ which matches this tree, namely, $(\mathcal{C} \cap C(T) | \sigma_1, \ldots, \sigma_l)_1$. We now assume the claim for trees of depth k and prove it for trees of depth $k+1$.

Let $T = \langle T_1, \ldots, T_l \rangle$ be a tree of depth $k+1$ and let $c \in \mathrm{ctx}^L(T)$. There must be a rule $(y | x_1, \ldots, x_l)_d \in \mathcal{R}^L$ such that $c \in y$, $x_i \leq \mathrm{ctx}^L(T_i)$ $(1 \leq i \leq l)$ and $d \leq k+1$. Therefore, there is a tree $S = \langle S_1, \ldots, S_l \rangle$ of depth at most $k+1$ such that $\mathrm{ctx}^L(S_i) = x_i$ for $i = 1, \ldots, l$ and $y = \mathcal{C} \cap C(S)$. For any $1 \leq i \leq l$, if $\mathrm{depth}(S_i) = 0$ then $x_i = S_i = \sigma_i \in \Sigma$ and therefore also $\mathrm{ctx}^L(T_i) = \sigma_i$ which implies that $T_i = S_i$. In the same way, if $\mathrm{depth}(T_i) = 0$ then $T_i = S_i$. If $0 < \mathrm{depth}(S_i)$ (and therefore also $0 < \mathrm{depth}(T_i)$) it follows, using the induction hypothesis, that $\mathcal{C} \cap C(S_i) = \mathrm{ctx}^L(S_i) \subseteq \mathrm{ctx}^L(T_i) = \mathcal{C} \cap C(T_i)$. By the construction of $\mathcal{C}$, this means that $C(S_i) \subseteq C(T_i)$ or, in other words, that T_i can appear in L in any context in which S_i appears. This is, of course, true also for S_i with $\mathrm{depth}(S_i) = 0$ since then $S_i = T_i$. Since $c(\langle S_1, \ldots, S_l \rangle) \in L$ this implies that $c(\langle T_1, \ldots, S_l \rangle) \in L$ and repeating this argument we conclude that $c(T) = c(\langle T_1, \ldots, T_l \rangle) \in L$. Therefore, $c \in C(T)$. □

The grammars produced by the algorithm are not context free grammars. However, Theorem 2 shows that for any context free grammar, the algorithm can generate a grammar which is strongly equivalent to it (that is, generates the same trees). It is also not too difficult to check that for any rule set $\mathcal{R}$, $L(\mathcal{R})$ is a context free language. We omit the proof of this here.

4 Stabilizing to a Learnable Language

As we saw in the previous section, the learning algorithm always converges, but may over-generalize and converge to a language which contains the original language. In this section we show that repeated learning (that is, each learner learning the language to which the previous learner converged) is bound (after a finite number of steps) to arrive at a language learnable by subsequent learners. It is not even necessary for every learner in the sequence to use the same context set $\mathcal{C}$ and the context set can be some random function of the language being learned (as will be discussed in detail later on).

To prove this, we need some notation for the cycles of learning. First, we assume that we have a sequence of finite context sets $\bar{\mathcal{C}} = \{\mathcal{C}_n\}_{n=0}^{\infty}$. We then define recursively $L^{\bar{\mathcal{C}}(n)} = \left(L^{\bar{\mathcal{C}}(n-1)}\right)^{\mathcal{C}_{n-1}}$ where $L^{\bar{\mathcal{C}}(0)} = L$. When there is no risk of confusion, we omit the $\bar{\mathcal{C}}$ superscript and simply write $L^{(n)}$. The language $L^{(n)}$ is the n'th *generation* language of L.

Since L will usually be fixed, we (usually) omit it from the notation and write $\mathcal{R}^{(n)}$ for $\mathcal{R}^{L^{(n)}}$ and $\mathrm{ctx}^{(n)}(T)$ for $\mathrm{ctx}^{L^{(n)}}(T)$. Note that $\mathcal{R}^{(n)}$ is the rule set learned (by the algorithm) from $L^{(n)}$ and that this rule set generates the language $L^{(n+1)}$. So as to be able to compare context sets in different generations, we define $\mathcal{C}_\infty = \bigcup_{0 \le n} \mathcal{C}_n$, which then allows us to define $C^{(n)}(T) = \{c \in \mathcal{C}_\infty \ : \ c(T) \in L^{(n)}\}$.

Lemma 3. *For any context free language L, any context set sequence $\bar{\mathcal{C}}$, any $0 \le n$ and any parse tree T with $1 \le \mathrm{depth}(T)$, $C^{(n)}(T) \cap \mathcal{C}_n \subseteq \mathrm{ctx}^{(n)}(T) \subseteq C^{(n+1)}(T) \cap \mathcal{C}_n$.*

Proof. The left inclusion is easy, since for $T = \langle T_1, \ldots, T_l \rangle \in \mathcal{S}(L^{(n)})$, the rule $(C^{(n)}(T) \cap \mathcal{C}_n | \mathrm{ctx}^{(n)}(T_1), \ldots, \mathrm{ctx}^{(n)}(T_l))_{\mathrm{depth}(T)}$ is in $\hat{\mathcal{R}}^{L^{(n)}}$.

We now prove the right inclusion. Let $c \in \mathrm{ctx}^{(n)}(T)$. We show that $c(T) \in L^{(n+1)}$. Let $T = \langle T_1, \ldots, T_l \rangle$. Since $c \in \mathrm{ctx}^{(n)}(T)$, there is a tree $S = \langle S_1, \ldots, S_l \rangle$ such that $c(S) \in L^{(n)}$, $\mathrm{ctx}^{(n)}(S_i) \le \mathrm{ctx}^{(n)}(T_i)$ $(1 \le i \le l)$ and $\mathrm{depth}(S) \le \mathrm{depth}(T)$. This means that any rule which applies to the calculation of $\mathrm{ctx}^{(n)}(S)$ also applies to the calculation of $\mathrm{ctx}^{(n)}(T)$ and therefore $\mathrm{ctx}^{(n)}(S) \subseteq \mathrm{ctx}^{(n)}(T)$. Together with $\mathrm{depth}(S) \le \mathrm{depth}(T)$ this implies that T can appear in $L^{(n+1)}$ in any context in which S can. Since $c(S) \in L^{(n)} \subseteq L^{(n+1)}$, we can conclude that $c(T) \in L^{(n+1)}$. $\square$

Our main assumption from now on is that $\mathcal{C}_\infty$ is finite. In other words, while the context sets used by the algorithm may vary from generation to generation, they are all bounded by a common finite set.

We now define a language L^∞ to which the language sequence $\left\{L^{(n)}\right\}_{0 \le n}$ will be shown to converge. For every tree T with $1 \le \mathrm{depth}(T)$, let $C^\infty(T) = \bigcup_{0 \le n} C^{(n)}(T)$ and for $T = \sigma \in \Sigma$ (tree of depth 0) let $C^\infty(T) = \sigma$. Let $\mathcal{S}^\infty(L) = \mathcal{S}\left(\bigcup_{0 \le n} L^{(n)}\right)$, that is, the set of all subtrees in all generations. The language L^∞ is defined to be the language generated by the rule set $\hat{\mathcal{R}}^\infty(L) = \left\{(C^\infty(T) | C^\infty(T_1), \ldots, C^\infty(T_l))_{\mathrm{depth}(T)} \ : \ T = \langle T_1, \ldots, T_l \rangle \in \mathcal{S}^\infty(L)\right\}$. We take $\mathcal{R}^\infty(L)$ to be the set of minimal rules in $\hat{\mathcal{R}}^\infty(L)$. Clearly, $L^\infty = L(\mathcal{R}^\infty(L))$. We write $\mathrm{ctx}^\infty(T)$ for $\mathrm{ctx}^{\mathcal{R}^\infty(L)}(T)$.

Theorem 3. *For any context free language L and any context set sequence $\bar{\mathcal{C}}$ such that $\mathcal{C}_\infty$ is finite, there exists an N such that for every $N \leq n$, $L^\infty = L^{(n)}$.*

Proof. Let $T \in L^{(n)}$. Then, by Lemma 3, $* \in \text{ctx}^{(n-1)}(T) \subseteq C^{(n)}(T) \cap \mathcal{C}_{n-1} \subseteq C^\infty(T)$. Therefore $* \in C^\infty(T)$. It is easy to check by induction that $C^\infty(T) \subseteq \text{ctx}^\infty(T)$ and therefore $T \in L^\infty$. This shows that $L^{(n)} \subseteq L^\infty$ for all n.

To complete the proof, we show that there exists an N such that for every $N < n$, $L^\infty \subseteq L^{(n)}$. Since $\{L^{(n)}\}_{0 \leq n}$ is an increasing sequence of languages, $\{C^{(n)}(T)\}_{0 \leq n}$ is an increasing sequence of sets. From this, together with the assumption that $\mathcal{C}_\infty$ is finite, it follows that for every tree T with $1 \leq \text{depth}(T)$ there is a number $N(T)$ such that for every $N(T) \leq n$, $C^{(n)}(T) = C^\infty(T)$. Since for every d the number of trees of depth no greater than d is finite, there is a number $N(d)$ such that for every tree $T \in \mathcal{S}^\infty(L)$ with $\text{depth}(T) \leq d$ and for every $N(d) \leq n$, $T \in \mathcal{S}(L^{(n)})$ and if $1 \leq \text{depth}(T)$ then also $C^{(n)}(T) = C^\infty(T)$. We prove the claim by taking $N = N(k) + 1$ where k is the maximal level of any rule in $\mathcal{R}^\infty(L)$.

Fix some $N \leq n$. We prove the claim by constructing a rule set $\mathcal{R}$ such that $L^\infty \subseteq L(\mathcal{R}) \subseteq L^{(n)}$. Let $\phi(x)$ be defined by $\phi(\sigma) = \sigma$ for $\sigma \in \Sigma$ and $\phi(y) = y \cap \mathcal{C}_{n-1}$ for $y \subseteq \mathcal{CT}(\Sigma)$. For a rule $R = (y|x_1, \ldots, x_l)_d$ we can then define $\phi(R) = (\phi(y)|\phi(x_1), \ldots, \phi(x_l))_d$. The rule set $\mathcal{R}$ is then defined by $\mathcal{R} = \{\phi(R) \ : \ R \in \mathcal{R}^\infty(L)\}$.

Let $R \in \mathcal{R}$, then there is a rule $R' \in \mathcal{R}^\infty(L)$ such that $R = \phi(R')$. By the choice of k, $R' = (C^\infty(S)|C^\infty(S_1), \ldots, C^\infty(S_l))$ for some $S = \langle S_1, \ldots, S_l \rangle \in \mathcal{S}^\infty(L)$ of depth at most k. By the choice of N, $S \in \mathcal{S}(L^{(n-1)})$ and $\{c \in \mathcal{C}_{n-1} \ : \ c(S) \in L^{(n-1)}\} = \phi(C^\infty(S))$. By Lemma 3 together with the choice of N, $\text{ctx}^{(n-1)}(S_i) = \phi(C^\infty(S_i))$ $(1 \leq i \leq l)$. Therefore, $R \in \hat{\mathcal{R}}^{L^{(n-1)}}$. This shows that $L(\mathcal{R}) \subseteq L(\mathcal{R}^{(n-1)}) = L^{(n)}$.

We now show that for every tree T, $\phi(\text{ctx}^\infty(T)) \leq \text{ctx}^{\mathcal{R}}(T)$. This proves that $L^\infty \subseteq L(\mathcal{R})$ because $T \in L^\infty \iff * \in \text{ctx}^\infty(T) \implies * \in \text{ctx}^{\mathcal{R}}(T) \iff T \in L(\mathcal{R})$. We prove the claim by induction on the depth of T. For $T = \sigma \in \Sigma$ (depth 0) this is immediate from the definitions. We assume now the claim for d and let $T = \langle T_1, \ldots, T_l \rangle$ be a tree of depth $d+1$. Let $c \in \phi(\text{ctx}^\infty(T))$. There must be a rule $R = (y|x_1, \ldots, x_l)_j \in \mathcal{R}^\infty(L)$ $(j \leq d+1)$ such that $c \in y$ and $x_i \leq \text{ctx}^\infty(T_i)$ $(1 \leq i \leq l)$. Since $\phi(R) \in \mathcal{R}$ and by the induction hypothesis $\phi(x_i) \leq \phi(\text{ctx}^\infty(T_i)) \leq \text{ctx}^{\mathcal{R}}(T_i)$ $(1 \leq i \leq l)$, it follows that the rule $\phi(R)$ is used in calculating $\text{ctx}^{\mathcal{R}}(T)$ and therefore $c \in \phi(y) \subseteq \text{ctx}^{\mathcal{R}}(T)$. □

We wish to apply this theorem in the following setting. For every context free language L there is a random variable $X(L)$ which selects a context set for L (that is, has values in $2^{\mathcal{CT}(\Sigma)}$). We assume that there exists a finite context set $\mathcal{C}_{\max}$ such that for every L, $P(X(L) \subseteq \mathcal{C}_{\max}) = 1$. In this case, beginning with any context free language L over Σ, we can generate the random sequence of languages $\{L^{\bar{\mathcal{C}}(n)}\}_{0 \leq n}$ where $\bar{\mathcal{C}}$ is defined by $\mathcal{C}_n = X(L^{(n)})$. With probability 1, Theorem 3 is applicable to the sequence generated in this way and we get that the language sequence converges to a language L^∞, that is, there is an N such that for every $N \leq n$, $L^{(n)} = L^\infty$. This means that L^∞ is learnable using any $\mathcal{C}_n$ with $N \leq n$. Every such $\mathcal{C}_n$ was generated by the random variable $X(L^{(n)}) = X(L^\infty)$. Moreover, since there are infinitely many such n, it follows that, with probability 1, for any $\mathcal{C}$ such that $P(X(L^\infty) = \mathcal{C}) > 0$, $\mathcal{C} = \mathcal{C}_n$ for some $N \leq n$. Therefore, with probability 1, L^∞ is learnable by any context to which $X(L^\infty)$ gives a non-zero probability. This means that if the learner generates the context set used by the algorithm based on the random variable X, the language sequence eventually converges to a language which is learnable by this learner with probability 1. In the next section we will see an example of such a learning strategy.

5 Choosing the Context Set

One natural way for the learner to choose the context set to be used with the learning algorithm is to take those contexts which appear most frequently in the language sample. Since for every language there are many different context sets which make it learnable, the algorithm is not too sensitive to moderate (unavoidable) changes in the frequencies and to the exact definition of "most frequent structures".

One reason for choosing the most frequent structures is that the more frequent the contexts in the context set are, the sooner will the algorithm see all the examples it needs to see in order to converge (since the interesting examples are those in which contexts from the chosen context set appear). We can imagine that some learners may prefer quick convergence even at the expense of inaccuracy of learning (over-generalization). Another reason for taking the most frequent contexts is that it is reasonable to assume that in most settings, the most frequent contexts depend not only on the language structure (grammar) but also on language usage. The influence of language usage on the chosen context set is then described by a random variable $X(L)$. This random variable depends, indeed, on the language being used. However, we assume that even if the language changes (over the generations) some common properties of language usage remain fixed. For example, since the size of a context is proportional to the number of words in the sentence in which it appears and since this directly influences the semantics of the sentence, it seems reasonable to assume that there is a bound on the size of the most frequent contexts, regardless of the specific grammar being used. In this way we satisfy the condition $P(X(L) \subseteq \mathcal{C}_{\max}) = 1$ of the previous section for all languages L. As we saw before, this condition guarantees that, with probability 1, the language sequence will eventually converge to a language which is learnable with probability 1.

It is possible to come up with different variants of "the most frequent contexts" approach outlined here. The discussion above would apply to any of them. We here present just one, very simple method which is based on the frequencies of contexts in a predetermined initial segment of the sample. The algorithm selects the k most frequent contexts in the first n examples or, alternatively, can take all those contexts which appeared at least k times in the first n examples.

This method is certain to construct a finite context set and to stabilize to a constant set after a finite number of steps.

6 Conclusion

The algorithm presented in this paper is extremely simple and natural. It essentially amounts to reading off rules from examples and discarding rules when they become redundant (non-minimal). This simplicity allows us to distinguish those factors essential to the correct functioning of the algorithm from those details which may be more freely modified. What is essential to the working of the algorithm is the use of a finite set of structures (the context set), the use of a grammar based on a partial order and the use of rules already inferred by the algorithm to infer additional rules (through the *ctx* function). Since inferred rules may initially be incorrect, the rules are layered to ensure correct convergence. Other details of the algorithm can be easily modified without changing the basic results. For example, the use of full contexts can be replaced by sub-structures of those contexts, sub-structures of the parse trees (equipped with an appropriate ordering) or even by information which is external to the sentence structure itself (such as semantic information). All one has to ensure is that a large enough variety of structures is available for the algorithm to be able to distinguish trees of different types (as in Theorem 2). Different choices of such structures can be made, without any significant effect on the analysis given here. In addition, different methods for determining the finite structure set (the context set given as a parameter) can be devised. While all these different variants have the same basic convergence properties, they may differ greatly in their rate of convergence, extent of over-generalization, computational complexity and applicability to actual learning situations.

References

1. H. Fernau, Learning Tree Languages from Text. *Technical report WSI-2001-19*, Universität Tübingen (Germany), Wilhelm-Schickard-Institut für Informatik, 2001.
2. H. Fernau, Learning Tree Languages from Text. In J. Kivinen and R. H. Sloan (eds.) *Proceedings of COLT 2002*, no. 2375 of LNAI, pp. 153-168. Springer, 2002.
3. E. M. Gold, Language Identification in the Limit. *Inform. and Control* 10, pp. 447-474, 1967.
4. M. Kanazawa, Learnable Classes of Categorial Grammars. *Studies in Logic Language and Information*, CSLI Publications, 1998.
5. T. Knuutila and M. Steinby, The Inference of Tree Languages from Finite Samples: an Algebraic Approach. *Theo. Comp. Sci.* 129, pp. 337-367, 1994.
6. Y. Sakakibara, Learning Context-Free Grammars from Structural Data in Polynomial Time. *Theo. Comp. Sci.* 76, pp. 223-242, 1990.
7. Y. Sakakibara, Efficient Learning of Context-Free Grammars from Positive Structural Examples. *Inform. Comput.* 97, pp. 23-60, 1992.
8. J. van Kampen, Bootstraps at Two for Lexicon and Discourse. In *Proceedings of ELA (Early Lexicon Acquisition) 2001.*