Evolutionary Grammatical Inference

Ernesto Rodrigues
Federal University of Technology, Brazil

Heitor Silvério Lopes
Federal University of Technology, Brazil

INTRODUCTION

Grammatical Inference (also known as grammar induction) is the problem of learning a grammar for a language from a set of examples. In a broad sense, some data is presented to the learner that should return a grammar capable of explaining to some extent the input data. The grammar inferred from data can then be used to classify unseen data or provide some suitable model for it.

The classical formalization of **Grammatical Inference** (GI) is known as Language Identification in the Limit (Gold, 1967). Here, there are a finite set S_+ of strings known to belong to the language L (the positive examples) and another finite set S_- of strings not belonging to L (the negative examples). The language L is said to be identifiable in the limit if there exists a procedure to find a grammar G such that $S_+ \subseteq L(G)$, $S_- \not\subset L(G)$ and, in the limit, for sufficiently large S_+ and S_-, $L = L(G)$. The disjoint sets S_+ and S_- are given to provide clues for the inference of the production rules P of the unknown grammar G used to generate the language L.

Grammatical inference include such diverse fields as speech and natural language processing, gene analysis, pattern recognition, image processing, sequence prediction, information retrieval, cryptography, and many more. An excellent source for a state-of-the art overview of the subject is provided in (de la Higuera, 2005).

Traditionally, most work in GI has been focused on the inference of regular grammars trying to induce finite-state automata, which can be efficiently learned. For context free languages some recent approaches have shown limited success (Starckie, Costie & Zaanen, 2004), because the search space of possible grammars is infinite. Basically, the parenthesis and palindrome languages are common test cases for the effectiveness of grammatical inference methods. Both languages are context-free. The parenthesis language is deterministic but the palindrome language is nondeterministic (de la Higuera, 2005).

The use of evolutionary methods for context-free grammatical inference are not new, but only a few attempts have been successful.

Wyard (1991) used Genetic Algorithm (GA) to infer grammars for the language of correctly balanced and nested parentheses with success, but fails on the language of sentences containing the same number of *a*'s and *b*'s ($a^n b^n$ language). In another attempt (Wyard, 1994), he obtained positive results on the inference of two classes of context-free grammars: the class of *n*-symbol palindromes with $2 \leq n \leq 4$ and a class of small natural language grammars.

Sen and Janakiraman (1992) applied a GA using a pushdown automata to the inference and successfully learned the $a^n b^n$ language and the parentheses balancing problem. But their approach does not scale well.

Huijsen (1994) applied GA to infer context-free grammars for the parentheses balancing problem, the language of equal numbers of *a*'s and *b*'s and the even-length 2-symbol palindromes. Huijsen uses a "markerbased" encoding scheme with has the main advantage of allowing variable length chromosomes. The inference of regular grammars was successful but the inference of context-free grammars failed.

Those results obtained in earlier attempts using GA to context-free grammatical inference were limited. The first attempt to use Genetic Programming (GP) for grammatical inference used a pushdown automata (Dunay, 1994) and successfully learned the parenthesis language, but failed for the $a^n b^n$ language.

Korkmaz and Ucoluk (2001) also presented a GP approach using a prototype theory, which provides a way to recognize similarity between the grammars in the population. With this representation, it is possible to recognize the so-called building blocks but the results are preliminary.

Javed and his colleagues (2004) proposed a Genetic Programming (GP) approach with grammar-specific heuristic operators with non-random construction of the initial grammar population. Their approach succeeded in inducing small context-free grammars.

More recently, Rodrigues and Lopes (2006) proposed a hybrid GP approach that uses a confusion matrix to compute the fitness. They also proposed a local search mechanism that uses information obtained from the sentence parsing to generate a set of useful productions. The system was used for the parenthesis and palindromes languages with success.

BACKGROUND

A formal language is usually defined as follows. Given a finite alphabet $\sum$ of symbols, we define the set of all strings (including the empty string ε) over $\sum$ as $\sum^*$. Thus, we want to learn a language $L \subset \sum^*$. The alphabet $\sum$ could be a set of characters or a set of words. The most common way to define a language is based on grammars which gives rules for combining symbols and to produce the all sentences of a language.

A grammar is defined by a quadruple $G = (N, \sum, P, S)$, where N is an alphabet of nonterminal symbols, $\sum$ is an alphabet of terminal symbols such that $N \cap \sum = \phi$, P is a finite set of production rules of the form $\alpha \rightarrow \beta$ for $\alpha, \beta \in (N \cup \sum)^*$ where * represents the set of symbols that can be formed by taking any number of them, possibly with repetitions. S is a special nonterminal symbol called the start symbol.

The language $L(G)$ produced from grammar G is the set of all strings consisting only of terminal symbols that can be derived from the start symbol S by the application of production rules. The process of deriving strings by applying productions requires the definition of a new relation symbol $\Rightarrow$. Let $\alpha X \beta$ be a string of terminals and nonterminals, where X is a nonterminal. That is, α and β are strings in $(N \cup \Sigma)^*$, and $X \in N$. If $X \rightarrow \varphi$ is a production of G, we can say $\alpha X \beta \Rightarrow \alpha \varphi \beta$. It is important to say that one derivation step can replace any nonterminal anywhere in the string. We may extend the $\Rightarrow$ relationship to represent one or many derivation steps. We use a * to denote more steps. Therefore, we formally define the language $L(G)$ produced from grammar G as $L(G) = \{ w \mid w \in \sum^*, S \Rightarrow^* w \}$.

More details about formal languages and grammars can be found in textbooks such as Hopcroft et al (2001).

The Chomsky Hierarchy

Grammars are classified according to the form of the production rules used. They are commonly grouped into a hierarchy of four classes, known as the **Chomsky hierarchy** (Chomsky, 1957).

- *Recursively enumerable languages:* a grammar is unrestricted, and its productions may replace any number of grammar symbols by any other number of grammar symbols. The productions are of the form $\alpha \rightarrow \beta$ with $\alpha, \beta \in (N \cup \sum)^*$.
- *Context-sensitive languages:* they have grammars with productions that replace a single nonterminal by a string of symbols, whenever the nonterminal occurs in a specific *context*, i.e., has certain left and right neighbors. These productions are of the form $\alpha A \gamma \rightarrow \alpha \beta \gamma$, with $A \in N$ and $\alpha, \beta, \gamma \in (N \cup \sum)^*$. A is replaced by β if it occurs between α and γ .
- *Context-free languages:* in this type, grammars have productions that replace a single nonterminal by a string of symbols, regardless of this nonterminal's context. The productions are of the form $A \rightarrow \alpha$ for $A \in N$ and $\alpha \in (N \cup \sum)^*$; thus A has no context.
- *Regular languages:* they have grammars in which a production may only replace a single nonterminal by another nonterminal and a terminal. The productions are of the form $A \rightarrow B\alpha$ or $A \rightarrow \alpha B$ for $A, B \in N$ and $\alpha \in \sum^*$.

It is sometimes useful to write a grammar in a particular form. The most commonly used in grammatical inference is the Chomsky Normal Form. A CFG G is in Chomsky Normal Form (CNF) if all production rules are of the form $A \rightarrow BC$ or $A \rightarrow \alpha$ for $A, B, C \in N$ and $\alpha \in \sum$.

The Cocke-Younger-Kasami Algorithm

To determine whether a string can be generated by a given context-free grammar in CNF, the Cocke-Younger-Kasami (CYK) algorithm can be used. This