Incrementally Learning Parameters of Stochastic Context-Free Grammars using Summary Statistics and Repeated Sampling

Brent Heeringa[1] and Tim Oates[2]

[1] Experimental Knowledge Systems Laboratory,
Computer Science Department,
University of Massachusetts, Amherst,
140 Governor's Drive,
Amherst, MA 01003 USA,
heeringa@cs.umass.edu
[2] Artificial Intelligence Lab,
Massachusetts Institute of Technology,
545 Technology Square,
Cambridge, MA 02139 USA,
oates@ai.mit.edu

Abstract. We are interested in how robots might learn language from exposure to utterances and sensory information about the physical contexts in which they occur. Although stochastic context free grammars are often used to represent the syntactic structure of natural languages, most algorithms for learning them from data require repeated processing of a corpus of sentences. The memory and computational demands of such algorithms are ill-suited for embedded agents. We present an online algorithm that uses summary statistics computed from sentences in combination with repeated sampling techniques to learn the parameters of stochastic context free grammars. Empirical results demonstrate that the algorithm performs just as well as the Inside-Outside algorithm despite the fact that it has access to less information about the training data.

1 Introduction

How might an embedded agent, such as a mobile robot, acquire a natural language from utterances describing its environment and activities paired with sensory information about the physical contexts in which they occurred? In other work we investigated the problem of identifying words in the speech signal and learning their denotations [8]. In this paper we describe the next step in this line of research - learning syntax of multiple word utterances.

Although natural languages are not entirely context free, stochastic context free grammars (SCFGs) are an effective representation for capturing much of their structure. However, for embedded agents, most algorithms for learning SCFGs from data have two shortcomings. First, they need access to a corpus

of complete sentences, requiring that the agent retain every sentence it hears. Second, they are batch algorithms that make repeated passes over the data, often requiring significant computation in each pass.

To address these shortcomings we present an online algorithm for learning the parameters of SCFGs using only summary statistics in combination in with repeated sampling techniques. SCFGs contain both structure (i.e. productions) and parameters (i.e. production probabilities). Many approaches to learning SCFGs from data alternate between improving the structure of the grammar given the parameters and then improving the parameters given the structure. Our algorithm assumes a fixed structure and attempts to find parameters that maximize the likelihood of the data. Future work will address online learning of grammar structure from summary statistics.

Given a SCFG to be learned, our algorithm has access to the grammar structure and a set of sentences generated by the grammar, but the parameters are hidden. The structure is first transformed so that each right-hand side is unique and the resulting productions are given random initial probabilities. The corpus is then parsed via a chart parser. Note that the parse does not depend on the probabilities, it only depends on the structure. Associated with each production is a histogram of the number of times it occurred in the parses of individual sentences in the corpus. The corpus is then discarded and only the histograms are retained.

The learner then repeatedly generates a corpus of sample sentences from its grammar, a process that does depend on the parameters, and parses them. This results in a second set of histograms. The degree to which the two sets of histograms differ is a measure of the difference between the current parameters of the learner's grammar and the target parameters. As the learner generates sentences and observes changes in its histograms, production probabilities are updated: production probabilities are increased when the histograms of the generated sentences are similar to the original histograms and decreased when they're dissimilar. Empirical results show that this procedure yields parameters that are comparable to those found by the Inside-Outside algorithm.

The remainder of the paper is organized as follows. In sections 2 and 3 we formally present stochastic context-free grammars and discuss previous research into learning their structure and parameters. Section 4 presents the algorithm and gives an analysis of its running time. Section 5 explains some experimental results and section 6 discusses future work. We conclude the paper in section 7.

2 Stochastic Context Free Grammars

Stochastic Context-Free Grammars [1] (SCFGs) are the natural extension of Context-Free Grammars to the probabilistic domain [9, 1]. Said differently, they are context-free grammars with probabilities associated with each production. Formally, a SCFG is a four-tuple $M = \langle V, \Sigma, R, S \rangle$ where

[1] Stochastic Context-Free Grammars are often called Probabilistic Context-Free Grammars.

1. V is a finite set of non-terminals
2. Σ is a finite set, disjoint from V, of terminals
3. R is a finite set of productions (often called rules or rewrite rules) of the form $A \rightarrow w$ where A belongs to V, and w is a finite string composed of elements from V and Σ. We refer to A as the left-hand side (LHS) of the production and w as the right-hand side (RHS), or expansion, of the production. Additionally, each production r has an associated probability $p(r)$ such that the probabilities of productions with the same left-hand side sum to 1.
4. S is the start symbol.

Grammars can either be ambiguous or unambiguous. Ambiguous grammars can generate the same string in multiple ways. Unambiguous grammars cannot.

3 Learning Stochastic-Context Free Grammars

Learning context-free grammars is the problem of inducing a context-free structure (or model) from a corpus of sentences (i.e., data). When the grammars are stochastic, we have the additional problem of learning production probabilities (parameters) from the corpus. More formally, given a set of sentence observations $O = \{o_0 \ldots o_{n-1}\}$, the problem is to learn a parameterized model M that best fits the observations.

Typically the problem is framed in terms of search where the objective function is the likelihood of the data given the grammar. One usually starts with a sentence corpus consisting of only positive training examples and a grammar that can generate each of the sentences in the corpus. Positive training examples are sentences that provide positive evidence of the structure and parameters in a model whereas negative training examples illustrate what we don't want in our model.

While we have interest in learning the structure of stochastic context-free grammars, our main focus here is on learning parameters. For a thorough overview of techniques for learning structure, see [10] and [3].

3.1 Learning Parameters

The Inside-Outside algorithm [6, 7] is the standard method for estimating parameters in SCFGs. The algorithm uses the general-purpose expectation-maximization (EM) procedure to iterate between calculating expectations of the parameters and then maximizing the likelihood given both the expectations and the sentence corpus. Almost all parameter learning is done batch style using some derivation of the Inside-Outside algorithm.

For example, in learning parameters, [2] initially estimates the production probabilities using the most probable parse of the sentences given the grammar (the Viterbi parse) and then uses a post-pass procedure that incorporates the Inside-Outside algorithm. He transforms the grammar into Chomsky Normal Form, runs Inside-Outside and then transforms it back.

In using EM to iterate between parameter estimation and maximization, the entire sentence corpus must be stored. While this storage may not be in the form of actual sentences, it is always in some representation that easily allows the reconstruction of the original corpus (e.g., a chart parse). Because we are interested in language acquisition in embedded agents and are motivated by developmental psychology, the prospect of memorizing entire sentence corpora is unpalatable.

This motivation also carries a desire to easily adjust our parameters when new sentences are encountered. That is, we want to learn production probabilities incrementally. While the Inside-Outside algorithm can be updated with the new sentences during the iterations, it still uses the entire sentence corpus for estimation. Hence, adding a new sentence is no different than simply starting from scratch with the new sentence included in the original corpus.

4 The Algorithm

To address some of the concerns given in the previous section, we have developed an unsupervised, incremental algorithm for finding parameter estimates in stochastic context-free grammars using repeated sampling from the estimation grammar and without retention of the sentence corpus.

The algorithm is incremental in two ways. First, in the classical sense, at every iteration it uses the previously learned grammar parameterization as a stepping stone to the new one. Second, it naturally allows new data to contribute to learning without restarting the entire process. In fact, because we store only histograms, the addition of new sentences typically does not increase the memory requirements.

4.1 Grammar Transformation

For our algorithm to work, we must first guarantee some properties of the grammar. All context-free grammars can be reduced to Chomsky Normal Form (CNF) [9] where each rule r in R is of the form:

$$A \to BC \text{ or } A \to z$$

where A, B, and C are non-terminals and z is a terminal. This means that each expansion is either a bigram (in the case of the non-terminal productions) or a unigram (in the case of the terminal productions). We derive a different normal form from CNF, called Unique Normal Form (UNF), which guarantees all rules involving only non-terminals are unique in their expansion and that all terminal rules have probability 1.0.

First, we replace all terminal rules $A \to z$ with two new rules $A \to D$ and $D \to z$ where D is a new, unique non-terminal. Additionally, we let $p(A \to D) = p(A \to z)$ and $p(D \to z) = 1.0$. This process pushes the previous probability of a terminal rule into a rule involving only non-terminals.

Table 1. A grammar that generates palindromes

$$
\begin{array}{l}
S \rightarrow A\ A \\
S \rightarrow B\ B \\
S \rightarrow A\ C \\
S \rightarrow B\ D \\
C \rightarrow S\ A \\
D \rightarrow S\ B \\
A \rightarrow Y \\
B \rightarrow Z \\
Y \rightarrow y \\
Z \rightarrow z
\end{array}
$$

Second, we ensure that no two productions have the same right-hand side. Suppose we have an indexed list of the CNF non-terminal rules $\{r_0 \ldots r_{n-1}\}$. When we first encounter a rule $r_j = A \rightarrow XY$ where the right-hand side of r_j is equivalent to the right-hand side of some earlier rule $r_i = B \rightarrow XY$, we replace r_j, and all rules with matching expansions that follow (r_k with $k > j$), with $A \rightarrow XZ$, where Z is a new non-terminal. Next, we add a new rule $Z \rightarrow Y$ with associated probability 1.0 to the end of the list. Inductively repeating this process with the subsequence $\{r_j \ldots r_{n-1}\}$ guarantees all rules in our final list (including the rules added at the end) have unique right-hand sides.

Note, this transformation does not keep the grammar in CNF, but does allow us to identify all productions involving only non-terminals by their expansion and guarantees the expansion will be either a bigram or unigram. Because our algorithm uses only summary statistics of the sentence corpus, we need to capture as much information about the data as possible, so UNF is both required and appropriate.

4.2 Algorithm Description

Suppose we have n sentence observations, $O = \{o_0 \ldots o_{n-1}\}$, and a SCFG, $M = \langle V, \Sigma, R, S \rangle$, in UNF with random parameter values that meet the constraints given in Section 2. Suppose further that M can generate each observation in our corpus O.

Let each rule in our grammar have two associated histograms called H_r^O and H_r^L. H_r^O is constructed by parsing the sentences in O and recording the number of times rule r was used when parsing individual sentences. Histograms constructed in this way are called observation histograms. The indices of the histogram range from 0 to k where k is the maximum number of times a rule was used in a particular sentence parse. In many cases, k remains small, and more importantly, when a sentence parse does not increase k, the storage requirements remain unchanged.

Each H_r^L is a histogram used during the learning process, so we call it a learning histogram. Like H_r^O, it records the number of times rule r occurred in

a single sentence parse. Unlike H_r^O however, the algorithm continually updates H_r^L during the learning process.

Using M and a standard chart parsing algorithm (e.g., [1]) we parse each observation o from the sentence corpus. Chart parsing a sentence allows us to count how many times a particular production (and because of our grammar form, a particular expansion) was used in deriving that sentence. For example, suppose we have the palindrome-generating grammar given in table 1 and the sentence $y\ y\ y\ y$. Chart parsing the sentence reveals that unigram Y has frequency 4 because y's can only be derived from the rule $A \rightarrow Y$. Additionally, we know that bigrams $A\,A$, $A\,C$ and $S\,A$ (the expansion of the rules $S \rightarrow A\,A$, $S \rightarrow A\,C$, and $C \rightarrow S\,A$) have frequency 1, and the remaining non-terminal expansions have frequency 0.

After every sentence parse, we update the observation histograms and discard the sentence along with its parse. We are left with only a statistical summary of the corpus. As a result, we cannot reconstruct the observation corpus or any single sentence within it. Since we have less information, estimating parameters (the production probabilities) would appear more difficult.

Beginning the iterative process, we randomly generate a small sentence corpus from the learning grammar O^L of size s, chart parse each sentence, and update each rule's learning histogram H_r^L appropriately. After discarding the sentences, we rescale the histograms to some fixed size h. If we don't rescale, the information provided in each additional sentence parse would have a decreasing impact on the histograms' distributions. This, in turn, decreases the learning rate. In some cases, decreasing the learning rate is required for convergence to a set of parameters [12]. However, in our experience, it was conveniently effective to keep h fixed throughout the duration of the algorithm.

So, for each rule r we now have two distributions: H_r^O, reflecting the observed corpus, and H_r^L reflecting the overall corpora generated from M. Comparing H_r^L to H_r^O seems a natural predictor of the likelihood of our observations given the grammar. *Relative entropy* (also known as the Kullback-Leibler distance) is commonly used to compare two distributions p and q [5]. It is defined as:

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

Because two distributions are associated with each rule r, we sum the relative entropies over the rules.

$$T = \sum_r \sum_x p_r(x) \log \frac{p_r(x)}{q_r(x)}$$

If T decreases between iterations, then the likelihood of M is increasing, so we increase the probabilities of productions used in parses of sentences generated in our small corpus. When s is large, we only increase a small subset of the productions used in the parse.[2] Likewise, if T increases between iterations, we

[2] We found, through experimentation, that using only the rules fired during generation of the last sentence works well.

Table 2. A grammar generating simple English phrases

S	$\rightarrow NP$	VP
NP	$\rightarrow Det$	N
VP	$\rightarrow Vt$	NP
VP	$\rightarrow Vc$	PP
VP	$\rightarrow Vi$	
PP	$\rightarrow P$	NP
Det	$\rightarrow A$	
Det	$\rightarrow THE$	
Vt	$\rightarrow TOUCHES$	
Vt	$\rightarrow COVERS$	
Vc	$\rightarrow IS$	
Vi	$\rightarrow ROLLS$	
Vi	$\rightarrow BOUNCES$	
N	$\rightarrow CIRCLE$	
N	$\rightarrow SQUARE$	
N	$\rightarrow TRIANGLE$	
P	$\rightarrow ABOVE$	
P	$\rightarrow BELOW$	
A	$\rightarrow a$	
THE	$\rightarrow the$	
$TOUCHES$	$\rightarrow touches$	
IS	$\rightarrow is$	
$ROLLS$	$\rightarrow rolls$	
$BOUNCES$	$\rightarrow bounces$	
$CIRCLE$	$\rightarrow circle$	
$SQURE$	$\rightarrow square$	
$TRIANGLE$	$\rightarrow triangle$	
$ABOVE$	$\rightarrow above$	
$BELOW$	$\rightarrow below$	

decrease the production probabilities. For our purposes, multiplicative update rules are used. For example, if $p_t(r)$ is the probability of r at time t and T decreased between iterations, then $p_{t+1}(r) = 0.01 * p_t(r)$. Once the probability updates are performed we start another iteration beginning with the generation of a small sentence corpus from our learning grammar. The algorithm stops iterating when the relative entropy falls below a threshold, or some prespecified number of iterations has completed.

5 Experiments

The previous section described an online algorithm for learning the parameters of SCFGs given summary statistics computed from a corpus of sentences. The question that remains is whether anything is sacrificed in terms of the quality of

Table 3. An ambiguous grammar

$S \rightarrow A$
$S \rightarrow B\ A$
$A \rightarrow C$
$A \rightarrow C\ C$
$B \rightarrow S$
$B \rightarrow D$
$B \rightarrow E$
$B \rightarrow F$
$C \rightarrow z$
$D \rightarrow y$
$E \rightarrow x$
$F \rightarrow w$

the learned grammar by using these statistics rather than the complete corpus. This section presents the results of experiments that compare the grammars learned by our algorithm with those learned by the Inside-Outside algorithm.

Let M^T be the target grammar whose parameters are to be learned. Let M^L be a grammar that has the same structure as M^T but whose production probabilities have been assigned random values so as to obey the constraint that the sum of these probabilities for all productions with the same left-hand side must be one. Let O^T be a set of sentences generated stochastically from M^T. The performance of the two algorithms can be compared by running them both on M^L and O^T and computing the log likelihood of O^T given the final grammar.

Because we are interested in learning parameters for a fixed structure, the above procedure involved using a number of different target grammars, each with the same structure but different production probabilities. The goal was to determine whether there were any regions of parameter space for which one algorithm was significantly better. We did this by stochastically sampling from this space. Note that a new corpus was generated for each new set of parameters as they influence which sentences are generated.

The grammar shown in table 2 [10] was used in this manner with 50 different target parameter settings and 500 sentences in O^T for each setting. The mean and standard deviation of the log likelihoods for the online algorithm with $h = s = 100$ (histogram size and learning corpus size respectively) were $\mu = -962.58$ and $\sigma = 241.25$. These values for the Inside-Outside algorithm were $\mu = -959.83$ and $\sigma = 240.85$. Recall that equivalent performance would be a significant accomplishment because the online algorithm has access to much less information about the data. We began by assuming the means of both empirical distributions were equal. With our assumption as the null hypothesis, we ran a two-tailed t-test resulting in $p = 0.95$. This means that if we reject the null hypothesis, the probability of making an error is 0.95.

Unfortunately, the above result does not sanction the conclusion that the two distributions are the same. We can, however, look at the power of the test in this case. If the test's power is high then it is likely that a true difference in the means would be detected. If the power is low then it is unlikely that the test would detect a real difference. The power of a test depends on a number of factors, including the sample size, the standard deviation, the significance level of the test, and the actual difference between the means. Given a sample size of 50, a standard deviation of 240.05, a significance level of 0.05, and an actual delta of 174.79, the power of the t-test is 0.95. That is, with probability 0.95 the t-test will detect a difference in means of at least 174.79 at the given significance level. Because the mean of the two distributions is minute, we need a more powerful test.

Since we ran both algorithms on the same problems, we can also use a paired sample t-test, which is more powerful than the standard t-test. We again set the null hypothesis to our earlier assumption that the means of the two distributions are equal. Running the paired sample t-test yielded $p < 0.01$. That is, the probability of making an error in rejecting the null hypothesis is less than 0.01. Closer inspection of the data reveals why this is the case. Inside-Outside performed better than the online algorithm on each of the 50 grammars. However, as is evident from the means and standard deviation, the absolute difference in each case was quite small. We know that problem variance is minimized when using the paired sample t-test [4]. So, because the unpaired t-test reported no significance and the paired t-test did report significance, we can conclude that the variance in log-likelihood is due almost entirely to problem instance rather than choice of algorithm. So, as a purely practical matter, the performance of the algorithms is virtually identical.

The same experiment was conducted with the ambiguous grammar shown in table 3. The grammar is ambiguous, for example, because $z\ z$ could be generated by $S \rightarrow A \rightarrow CC \rightarrow z, z$ or $S \rightarrow BA$ with $B \rightarrow S \rightarrow C \rightarrow z$ and $A \rightarrow C \rightarrow z$. The mean and standard deviation of the log likelihoods for the online algorithm were $\mu = -1983.15$ and $\sigma = 250.95$. These values for the Inside-Outside algorithm were $\mu = -1979.37$ and $\sigma = 250.57$. The standard t-test returned a p value of 0.94 and the paired sample t-test was significant at the 0.01 level. Again, inside-outside performed better on every one of the 50 grammars, but the differences were very small.

5.1 Some Analysis

The Inside-Outside algorithm runs in $O(n^3)$ time in the size of the input [6]. This is because Inside-Outside uses dynamic programming to compute sentence probabilities. Because we also compute sentence probabilities in the guise of chart-parsing, our algorithm runs in $O(n^3)$ time in the size of the input.

Every iteration of the Inside-Outside algorithm however, requires the complete sentence corpus. So, using the Inside-Outside algorithm in the context of embedded agents, where the sentence corpus increases continuously with time,

means a corresponding continuous increase in memory. With our algorithm, the memory requirements remain effectively constant.

One drawback of our approach is that learning good parameter estimations requires many more iterations than our Inside-Outside counterpart. For example, when learning grammars in tables 2 and 3 we required 500 to 1000 iterations whereas the Inside-Outside algorithm only needed 3-5 iterations.

6 Discussion

Our algorithm can loosely be set into an actor-critic framework [12]. In such methods, the actor performs action selection and the critic criticizes the actor's selections. Hence, the learning grammar M is the actor and the sets of histograms $\{H_r^O \mid r \in R\}$ and $\{H_r^L \mid r \in R\}$ are the critics because M generates the sentence corpus for each iteration (action selection) and the histograms critique the selection. [12] note that actor-critic methods may be "more appealing in psychological or biological models" because it is easier to "improve domain-specific constraints".

Because our work is motivated by developmental psychology and the unsupervised acquisition of language by embedded agents, we are interested in learning structural models of SCFG's. Remember that one appeal of our algorithm is the ability to add new sentence data naturally and incrementally during the learning process. But if we encounter a new sentence that our grammar cannot parse, it is not clear how we should handle the situation. One idea is to add a completely new set of productions so the sentence can be parsed, and then use structural learning frameworks such as *model merging* [11] or *move selection* [2] to generalize the grammar.

7 Conclusion

Most parameter learning algorithms for stochastic context-free grammars retain the entire sentence corpus throughout the learning process. Incorporating a complete memory of sentence corpora seems ill-suited for language acquisition in embedded agents. We have introduced an incremental algorithm for learning parameters in stochastic context-free grammars using only summary statistics of the observed data. Empirical results demonstrate that the algorithm performs as well as the Inside-Outside algorithm despite using only a statistical summary of the data.

Acknowledgements

This research is supported by DARPA/USASMDC under contract number DASG60-99-C-0074. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should

not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA/USASMDC or the U.S. Government.

Many thanks to Paul R. Cohen for his helpful comments and guidance when revising and editing the paper. Additional thanks to Gary Warren King and Clayton Morrison for their excellent suggestions. We would also like to acknowledge Mark Johnson for the use of his Inside-Outside algorithm implementation.

References

1. Charniak, E. (1993). *Statistical language learning.* Cambridge: The MIT Press.
2. Chen, S. F. (1995). *Bayesian grammar induction for language modeling* (Technical Report TR-01-95). Center for Research in Computing Technology, Harvard University, Cambridge, MA.
3. Chen, S. F. (1996). *Building probabilistic models for natural language.* Doctoral dissertation, The Division of Applied Sciences, Harvard University.
4. Cohen, P. R. (1995). *Empirical methods for artificial intelligence.* The MIT Press.
5. Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory.* John Wiley and Sons, Inc.
6. Lari, K., & Young, S. J. (1990). The estimation of stochastic context-free grammars using the inside-outiside algorithm. *Computer Speech and Language, 4,* 35–56.
7. Lari, K., & Young, S. J. (1991). Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language, 5,* 237–257.
8. Oates, T. (2001). *Grounding knowledge in sensors: Unsupervised learning for language and planning.* Doctoral dissertation, The University of Massachusetts, Amherst.
9. Sipser, M. (1997). *Introduction to the theory of computation.* Boston: PWS Publishing Company.
10. Stolcke, A. (1994). *Bayesian learning of probabilistic language models.* Doctoral dissertation, Division of Computer Science, University of California, Berkeley.
11. Stolcke, A., & Omohundro, S. (1994). Inducing probabilistic grammars by bayesian model merging. *Grammatical Inference and Applications* (pp. 106–118). Berlin, Heidelberg: Springer.
12. Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: an introduction.* Cambridge: The MIT Press.