Two Algorithms for Learning the Parameters of Stochastic Context-Free Grammars

Brent Heeringa
Department of Computer Science
University of Massachusetts, Amherst
140 Governor's Drive
Amherst, MA 01003
heeringa@cs.umass.edu

Tim Oates
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250
oates@eecs.umbc.edu

Abstract

Stochastic context-free grammars (SCFGs) are often used to represent the syntax of natural languages. Most algorithms for learning them require storage and repeated processing of a sentence corpus. The memory and computational demands of such algorithms are ill-suited for embedded agents such as a mobile robot. Two algorithms are presented that incrementally learn the parameters of stochastic context-free grammars as sentences are observed. Both algorithms require a fixed amount of space regardless of the number of sentence observations. Despite using less information than the inside-outside algorithm, the algorithms perform almost as well.

Introduction

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 the agent to retain every sentence it hears. Second, they are batch algorithms that make repeated passes over the data, often requiring significant computation in each pass. These shortcomings are addressed through two online algorithms called SPAN[1] and PRESPAN[2] that learn the parameters of SCFGs using only summary statistics in combination with repeated sampling techniques.

SCFGs contain both structure (i.e. rules) and parameters (i.e. rule probabilities). One approach to learning SCFGs from data is to start with a grammar containing all possible rules that can be created from some alphabet of terminals and non-terminals. Typically the size of the right-hand-side of each rule is bound by a small constant (e.g. 2). Then an algorithm for learning parameters is applied and allowed to "prune" rules by setting their expansion probabilities to zero (Lari & Young, 1990). PRESPAN and SPAN operate in this paradigm by assuming a fixed structure and modifying the parameters.

Given a SCFG to be learned, both algorithms have access to the structure of the grammar and a set of sentences generated by the grammar. The correct parameters are unknown. PRESPAN and SPAN begin by parsing the sentence corpus using a chart parser. Note that the parse of an individual sentence does not depend on the parameters; it only depends on the structure. However, the distribution of sentences parsed does depend on the parameters of the grammar used to generate them. Both algorithms associate with each rule a histogram that records the number of times the rule is used in parses of the individual sentences.

PRESPAN and SPAN make an initial guess at the values of the parameters by setting them randomly. They then generate a corpus of sentences with these parameters and parse them, resulting 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 parameter estimates and the target parameters. PRESPAN modifies its parameter estimates so the sum total difference between the histograms is minimized. In contrast, SPAN modifies its estimates so the difference between individual histograms is minimized. Empirical results show that this procedure yields parameters that are close to those found by the inside-outside algorithm.

Stochastic Context-Free Grammars

Stochastic context-free grammars [3] are the natural extension of Context-Free Grammars to the probabilistic domain (Sipser, 1997; Charniak, 1993). Said differently, they are context-free grammars with probabilities associated with each rule. Formally, a SCFG is a four-tuple $M = \langle V, \Sigma, R, S \rangle$ where

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 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 rule and w as the right-hand side (RHS), or expansion, of the rule. Additionally, each rule r has an

[1] SPAN stands for Sample Parse Adjust Normalize

[2] PRESPAN is so named because it is the predecessor of SPAN, so it literally means pre-SPAN

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

associated probability $p(r)$ such that the probabilities of rules 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.

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, one faces the additional problem of learning rule probabilities (parameters) from the corpus. Given a set of sentence observations $O = \{o_0 \ldots o_{n-1}\}$, the goal is to discover the grammar that generated O. Typically, this problem is framed in terms of a search in grammar space where the objective function is the likelihood of the data given the grammar. While the problem of incrementally learning the structure of SCFGs is interesting in its own right, the main focus here is on learning parameters. For a thorough overview of existing techniques for learning structure, see (Stolcke, 1994; Chen, 1996; Nevill-Manning & Witten, 1997).

Learning Parameters

The inside-outside algorithm (Lari & Young, 1990; Lari & Young, 1991) is the standard method for estimating parameters in SCFGs. The algorithm uses the general-purpose expectation-maximization (EM) procedure. Almost all parameter learning is done batch style using some version of the inside-outside algorithm. For example, in learning parameters, (Chen, 1995) initially estimates the rule 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.

To use EM 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., the chart of a chart parse). Because we are interested in language acquisition in embedded agents over long periods of time, the prospect of memorizing and repeatedly processing 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 incorporate new sentences in between iterations, it still uses the entire sentence corpus for estimation.

The Algorithms

PRESPAN and SPAN address some of the concerns given in the previous section. Both are unsupervised, incremental algorithms for finding parameter estimates in stochastic context-free grammars.

PRESPAN and SPAN are incremental in two ways. First, in the classical sense, at every iteration they use the previously learned parameter estimation as a stepping stone to the new one. Second, both algorithms naturally allow new data to contribute to learning without restarting the entire process.

Table 1: A grammar that generates palindromes

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	
X	$\rightarrow$	z	

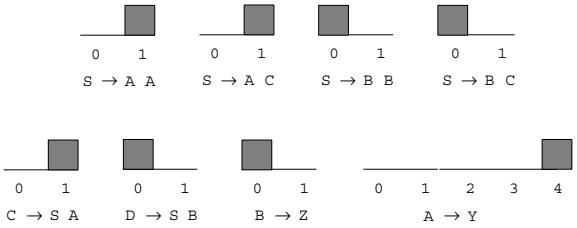

Figure 1: The palindrome grammar rule histograms after only one parse of the sentence *y y y y*. Because only one sentence has been parse, the mass of the distribution is concentrated in a single bin.

PRESPAN and SPAN use only a statistical summary of the observation data for learning. Both store the summary information in histograms. SPAN and PRESPAN also record histogram information about their current parameter estimates. So the addition of new sentences typically does not increase the memory requirements. Furthermore, the histograms play a crucial role in learning. If the parameter estimates are accurate, the histograms of the observed data should resemble the histograms of the current parameterization. When the histograms do not resemble each other, the difference is used to guide the learning process.

A Description of PRESPAN

Let $T = \langle V, \Sigma, R, S \rangle$ be a SCFG and let $O = \{o_0 \ldots o_{n-1}\}$ be a set of n sentences generated stochastically from T. Let $M = \langle V, \Sigma, R', S \rangle$ be a SCFG that is the same as T except the rule probabilities in R' have been assigned at random (subject to the constraint that the sum of the probabilities for all rules with the same left-hand side is one). T is called the target grammar and M the learning grammar. The goal is to use a statistical summary of O to obtain parameters for M that are as close to the unknown parameters of T as possible.

Using M and a standard chart parsing algorithm (e.g., Charniak, 1993 or Allen, 1995) one can parse a sentence and count how many times a particular rule was used in deriving that sentence. Let each rule r in a grammar have two associated histograms called H_r^O and H_r^L. H_r^O is constructed by parsing each sentence in O and recording the number of times rule r appears in the parse tree of the sentence. His-

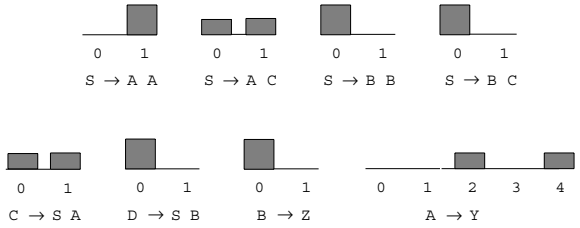

Figure 2: The palindrome grammar rule histograms after parsing *y y y y* and *y y*. Notice that the mass of rules $S \rightarrow A\,C$, and $C \rightarrow S\,A$ are now evenly distributed between 0 and 1. Similarly the mass of rule $S \rightarrow Y$ is evenly distributed between 2 and 4.

tograms 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 identical in nature to H_r^O but is used during the learning process, so it's a learning histogram. Like the observation histograms, PRESPAN uses each learning histogram to record the number of times each rule occurs in single sentence parses during the learning process. The difference is that the corpus of sentences parsed to fill H_r^L is generated stochastically from M using its current parameters.

For example, suppose PRESPAN is provided with the palindrome-generating structure given in Table 1 and encounters the sentence $y\ y\ y\ y$. Chart parsing the sentence reveals that rule $A \rightarrow Y$ has frequency 4, rules $S \rightarrow A\,A$, $S \rightarrow A\,C$ and $S \rightarrow A$ have frequency 1, and the remaining non-terminals have frequency 0. Figure 1 depicts graphically the histograms for each rule after parsing the sentence. In parsing the sentence $y\,y$, the rule $S \rightarrow A\,A$ is used once, $A \rightarrow Y$ is used twice and the other rules are not used. Figure 2 shows how the histograms in Figure 1 change after additionally parsing $y\,y$.

After every sentence parse, PRESPAN updates the observation histograms and discards the sentence along with its parse. It is left with only a statistical summary of the corpus. As a result, one cannot reconstruct the observation corpus or any single sentence within it. From this point forward the observation histograms are updated only when new data is encountered.

PRESPAN now begins the iterative process. First, it randomly generates a small sentence corpus of prespecified constant size s from its learning grammar M. Each sentence in the sample is parsed using a chart parser. Using the chart, PRESPAN records summary statistics exactly as it did for the observation corpus except the statistics for each rule r are added to the learning histograms instead of the observation histograms. After discarding the sentences, the learning histograms are normalized to some fixed size h. Without normalization, the information provided by the new statistics would have decreasing impact on the histograms' distributions. This is because the bin counts typically increase linearly while the sample size remains constant. Future work will examine the role of the normalization factor, however, for this work it is kept fixed throughout the duration of the algorithm.

For each rule r PRESPAN now has two distributions: H_r^O, based on the corpus generated from T, and H_r^L based on the corpus generated from M. Comparing H_r^L to H_r^O seems a natural predictor of the likelihood of the observation corpus given PRESPAN's learning grammar. *Relative entropy* (also known as the Kullback-Leibler distance) is commonly used to compare two distributions p and q (Cover & Thomas, 1991). 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, the relative entropies are summed over the rules.

$$\tau = \sum_r \sum_x p_r(x) \log \frac{p_r(x)}{q_r(x)} \quad (1)$$

If τ decreases between iterations, then the likelihood of M is increasing so PRESPAN increases the probabilities of the rules used in generating the sample corpus. When s is large, the algorithm only increases a small subset of the rules used to generate the sample.[4] Likewise, if τ increases between iterations, PRESPAN decreases the rule probabilities.

PRESPAN uses a multiplicative update function. Suppose rule r was selected for an update at time t. If $p_t(r)$ is the probability of r at time t and τ decreased between iterations, then $p_{t+1}(r) = 1.01 * p_t(r)$. Once the probability updates are performed PRESPAN starts another iteration beginning with the generation of a small sentence corpus from the learning grammar. The algorithm stops iterating when the relative entropy falls below a threshold, or some prespecified number of iterations has completed.

A Description of SPAN

SPAN differs from PRESPAN in the selection of rules to update, the criteria for updates, and the update rule itself. Recall that PRESPAN uses τ (see Equation 1), the sum of relative entropy calculations for each rule, as a measure of progress or deterioration of grammar updates. Since τ is an aggregate value, an unsuccessful change in probability for one rule could overshadow a successful change of another rule. Furthermore, the update rule does not differentiate between small and large successes and failures.

SPAN addresses these concerns by examining local changes in relative entropy and using those values to make rule specific changes. SPAN calculates the relative entropy for rule r at time t and compares it with the relative entropy at time $t-1$. If the relative entropy decreases it means SPAN updated the rule probability favorably, if it increases, the distributions have become more dissimilar so the probability

[4] Using only the rules fired during the generation of the last sentence seems to works well.

should move in the opposite direction. This is best explained by examining SPAN's update rule:

$$\begin{aligned} P_{t+1}(r) = P_t(r) \quad &+ \quad \alpha \quad * \quad sgn(P_t(r) - P_{t-1}(r)) \\ &\quad\quad * \quad sgn(\Delta RE_t) \\ &\quad\quad * \quad f(\Delta RE_t) \\ &+ \quad \beta \quad * \quad (P_t(r) - P_{t-1}(r)) \end{aligned} \tag{2}$$

The update rule is based on the steepest descent method (Bertsekas & Tsitsiklis, 1996). Here, sgn is the "sign" function that returns -1 if its argument is negative, 0 if its argument is zero and +1 if its argument is positive. The first sign function determines the direction of the previous update. That is, it determines whether, in the last time step, SPAN increased or decreased the probability. The second sign function determines if the relative entropy has increased or decreased. If it has decreased, then the difference is positive, if it increased, the difference is negative. Together these sign functions determine the direction of the step. The function $f(\Delta RE_t)$ returns the magnitude of the step. This, intuitively, is an estimate of the gradient since the magnitude of the change in relative entropy is reflective of the slope. The α parameter is a step-size. Finally the $\beta * (P_t(r) - P_{t-1}(r))$ expression is a momentum term.

Once the probability updates are performed for each rule, another iteration starts beginning with the generation of a small sentence corpus from the learning grammar. Like PRESPAN, the algorithm stops iterating when the relative entropy falls below a threshold, or some prespecified number of iterations has completed.

SPAN is a more focused learning algorithm than PRESPAN. This is because all rules are individually updated based on local changes instead of stochastically selected and updated based on global changes. The algorithmic changes speed up learning by, at times, two orders of magnitude. While these benefits drastically increase learning time, they do not necessarily result in more accurate grammars. Evidence and explanation of this is given in the Experiments section.

Algorithm Analysis

In this section both the time and space requirements of the algorithms are analyzed. Comparing the results with the time and space requirements of the inside-outside algorithm shows that SPAN and PRESPAN are asymptotically equivalent in time but nearly constant (as opposed to linear with inside-outside) in space.

The inside-outside algorithm runs in $O(\mathcal{L}^3|V|^3)$ time where $\mathcal{L}$ is the length of sentence corpus and $|V|$ is the number of non-terminal symbols (Stolcke, 1994). The complexity arises directly from the chart parsing routines used to estimate probabilities. Note that the number of iterations used by the inside-outside algorithm is dominated by the computational complexity of chart parsing.

Both SPAN and PRESPAN chart parse the observation corpus once but repeatedly chart parse the fixed size samples they generates during the learning process. Taken as a whole, this iterative process typically dominates the single parse of the observations sentences, so the computational complexity is $O(\mathcal{J}^3|V|^3)$ where $\mathcal{J}$ is the length of of the maximum sample of any iteration.

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	
$ROLLS$	$\rightarrow$	$rolls$	
$BOUNCES$	$\rightarrow$	$bounces$	
$CIRCLE$	$\rightarrow$	$circle$	
$SQURE$	$\rightarrow$	$square$	
$TRIANGLE$	$\rightarrow$	$triangle$	
$ABOVE$	$\rightarrow$	$above$	
$BELOW$	$\rightarrow$	$below$	

Every iteration of the inside-outside algorithm requires the complete sentence corpus. Using the algorithm in the context of embedded agents, where the sentence corpus increases continuously with time, means a corresponding continuous increase in memory. With SPAN and PRESPAN, the memory requirements remain effectively constant.

While the algorithms continually update their learning histograms through the learning process, the number of bins increases only when a sentence parse contains an occurrence count larger than any encountered previously. The sample is representative of the grammar parameters and structure, so typically after a few iterations, the number of bins becomes stable. This means that when new sentences are encountered there is typically no increase in the amount of space required.

Experiments

The previous section described two online algorithms for learning the parameters of SCFGs given summary statistics computed from a corpus of sentences. The remaining question is whether the quality of the learned grammar is sacrificed because a statistical summary of the information is used rather than the complete sentence corpus. This section presents the results of experiments that compare the grammars learned with PRESPAN and SPAN with those learned by the inside-outside algorithm.

The following sections provide experimental results for both the PRESPAN and SPAN algorithms.

Experiments with PRESPAN

Let M^T be the target grammar whose parameters are to be learned. Let M^L be a grammar that has the same structure

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	

as M^T but with rule probabilities initialized uniformly random and normalized so the sum of the probabilities of the rules with the same left-hand side is 1.0. Let O^T be a set of sentences generated stochastically from M^T. The performance of the algorithm is compared by running it on M^L and O^T and computing the log likelihood of O^T given the final grammar.

Because the algorithm learns parameters for a fixed structure, a number of different target grammars are used in experimentation; each with the same structure but different rule probabilities. The goal is to determine whether any regions of parameter space were significantly better for one algorithm over the other. This is accomplished by stochastically sampling from this space. Note that a new corpus is generated for each new set of parameters as they influence which sentences are generated.

The grammar shown in Table 2 (Stolcke, 1994) 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 PRESPAN 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. Suppose the means of both empirical distributions are equal. With this assumption as the null hypothesis, a two-tailed t-test results in $p = 0.95$. This means that if one rejects 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. One 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, a more powerful test is needed.

Since both PRESPAN and the inside-outside algorithm were run on the same problems, a paired sample t-test can be applied. This test is more powerful than the standard t-test. Suppose again the the means of the two distributions are equal. Using this as the null hypothesis and performing the paired sample t-test yields $p < 0.01$. 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.

The same experiments were conducted with the ambiguous grammar shown in Table 3. The grammar is ambiguous because, for example, $z\ z$ can be generated by $S \rightarrow A \rightarrow C\,C \rightarrow zz$ 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 PRESPAN 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.

Experiments with SPAN

The same experiments were performed using SPAN. That is, the grammar in Table 2 was used 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 SPAN with $h = s = 100$ (histogram size and learning corpus size respectively) were $\mu = -4266.44$ and $\sigma = 650.57$. These values for the inside-outside algorithm were $\mu = -3987.58$ and $\sigma = 608.59$. Recall that equivalent performance would be a significant accomplishment because the online algorithm has access to much less information about the data. Assuming both the means of the distributions are equal and using this as the null hypothesis of a two-tailed t-test results in $p = 0.03$.

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 $S \rightarrow A \rightarrow C\,C \rightarrow z, z$ 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 $\mu = -2025.93$ and $\sigma = 589.78$. These values for the inside-outside algorithm were $\mu = -1838.41$ and $\sigma = 523.46$. The t-test returned a p value of 0.33.

Inside-outside performed significantly better on the unambiguous grammar but there was not a significant difference on the ambiguous grammar. Given the fact that SPAN has access to far less information than the inside-outside algorithm, this is not a trivial accomplishment. One conjecture is that SPAN never actually converges to a stable set of parameters but walks around whatever local optimum it finds in parameter space. This is suggested by the observation that for any given training set the log likelihood for

the inside-outside algorithm is always higher than that for SPAN. Comparison of the parameters learned shows that SPAN is moving in the direction of the correct parameters but that it never actually converges on them.

Discussion

It was noted earlier that SPAN learns more quickly than PRESPAN but the Experiments section shows this improvement may come at a cost. One reason for this may lie in the the sentence samples produced from the learning grammar during each iteration. Recall that SPAN learns by generating a sentence sample using its current parameter estimates. Then this sample is parsed and the distribution is compared to the distribution of the sentences generated from the target grammar. Each sentence sample reflects the current parameter estimates, but also has some amount of error. This error may be more pronounced in SPAN because at each iteration, every rule is updated. This update is a direct function of statistics computed from the sample, so the sample error may overshadow actual improvement or deterioration in parameter updates from the last iteration.

Overcoming the sample error problem in SPAN might be accomplished by incorporating global views of progress, not unlike those used in PRESPAN. In fact, a synergy of the two algorithms may be an appropriate next step in this research.

Another interesting prospect for future parameter-learning research is based on rule orderings. Remember that parameter changes in rules *closer* to the start-symbol of a grammar have more effect on the overall distribution of sentences than changes to parameters farther away. One idea is to take the grammar, transform it into a graph so that each unique left-hand side symbol is a vertex and each individual right-hand-side symbol is a weighted arc. Using the start-symbol vertex as the root node and assuming each arc has weight 1.0, one can assign a rank to each vertex by finding the weight of the shortest path from the root to all the other vertices. This ordering may provide a convenient way to iteratively learn the rule probabilities. One can imagine concentrating only on learning the parameters of the rules with rank 1, then fixing those parameters and working on rules with rank 2, and so forth. When the final rank is reached, the process would start again from the beginning. Clearly self-referential rules may pose some difficulty, but the ideas have yet to be fully examined.

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 learning in embedded agents. PRESPAN and SPAN are two incremental algorithms for learning parameters in stochastic context-free grammars using only summary statistics of the observed data. Both algorithms require a fixed amount of space regardless of the number of sentences they processes. Despite using much less information than the inside-outside algorithm, PRESPAN and SPAN perform almost as well.

Acknowledgements Paul R. Cohen provided helpful comments and guidance when revising and editing the paper. Additional thanks to Gary Warren King and Clayton Morrison for their excellent suggestions.

Mark Johnson supplied a clean implementation of the inside-outside algorithm.

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.

References

Allen, J. (1995). *Natural language understanding*. The Benjamins/Cummings Publishing Company, Inc. 2 edition.

Bertsekas, D. P., & Tsitsiklis, J. N. (Eds.). (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.

Charniak, E. (1993). *Statistical language learning*. Cambridge: The MIT Press.

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.

Chen, S. F. (1996). *Building probabilistic models for natural language*. Doctoral dissertation, The Division of Applied Sciences, Harvard University.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. John Wiley and Sons, Inc.

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.

Lari, K., & Young, S. J. (1991). Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language, 5*, 237–257.

Nevill-Manning, C., & Witten, I. (1997). Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research, 7*, 67–82.

Sipser, M. (1997). *Introduction to the theory of computation*. Boston: PWS Publishing Company.

Stolcke, A. (1994). *Bayesian learning of probabilistic language models*. Doctoral dissertation, Division of Computer Science, University of California, Berkeley.

Stolcke, A., & Omohundro, S. (1994). Inducing probabilistic grammars by bayesian model merging. *Grammatical Inference and Applications* (pp. 106–118). Berlin, Heidelberg: Springer.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: an introduction*. Cambridge: The MIT Press.