Current Issues in Mathematical Linguistics
C. Martín-Vide (Editor)
© 1994 Elsevier Science B.V. All rights reserved.

Cartesian Computation and Linguistics in a Current Context

Robert C. Berwick[a] *

[a]Department of Computer Science and Electrical Engineering and
Brain and Cognitive Sciences
Room 838, MIT AI Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Mathematics plays a distinct role in the formal analysis of human language. On the one hand, given the essentially biological character of language, we should not expect mathematical formalisms to play a greater role here than in the normal sciences such as biology or physics: that is, formalisms should make explicit otherwise implicit and unclear implications of linguistic theories. On the other hand, for whatever reason—perhaps because of its close relationship to abstract string manipulation—linguistic science has been susceptible to a kind of "overexpectation" effect: it seems as though mathematics should have more to say about natural language than it has, partially due to the history of generative grammar. The end result is mixed: we often find partial mathematical analyses that, in the end, do not have much relevance for the study of language from the mathematical point of view and can lead to paradoxes on the computational end. In this paper we give a sharp example of this sort of paradox that arise from the study of language as a modular, Cartesian system: we examine the mathematics of how Cartesian systems like these might be computed, that is, learned.

1. INTRODUCTION: MATHEMATICAL MODELING AND LANGUAGE

Mathematics and language have had a long and some would say shaky marriage. In the current era, the mathematics-language connection begins of course with the notion of generative grammar developed by Chomsky, and the classical demonstration that finite-state languages are descriptively inadequate for English (presumably other) natural languages. These mathematical results had a large impact on the field at their time: they put a lower bound on the complexity of natural language, showing that simple memoryless concatenative systems could not begin to describe the complexity of natural languages.

Through the intervening three and half decades, mathematics has now and again played a role in the construction of linguistic models, but usually a tangential role. Rather, mathematical models have been used to give added plausibility to existing theories. From time to time, there have been efforts to use mathematical or computational techniques to argue for one or another approach to linguistic theory. Thus we find in the

*Kind acknowledgement is made to Partha Niyogi for his original insight into the Markovian formulation of triggers, to Sandiway Fong for discussion of modular parsing, and to NSF grant 9217041-ASC and ARPA under the High Performance Computing

presentations of Generalized Phrase Structure Grammar (GPSG) [1] the pro-
at eliminating transformational rules is a step forward because we would have
ginnings of an explanation" for efficient parsability. However, that possibility
hort-lived. Over the next few years, using the tools of modern computational
ity theory, results accumulated demonstrating that to all intents and purposes,
PSG and transformational grammar are equally difficult to parse, in the com-
ial complexity theory sense; both are NP-hard. Thus by the 1985 publication
ll book on GPSG [2] we find that any argument that theory is "better" based
utational efficiency has been explicitly denied. The same fate occurred with
Functional Grammar (LFG): in the late 1970s LFG was touted as being more
to parse, in the formal sense, hence superior to transformational grammar; by
1980s, computational complexity results showed that LFG too was NP-hard,
ussion of its superiority in this formal sense was abandoned.
esults demonstrate at least that formal computational methods can in fact make
lications of even a supposedly formalized theory more clearcut. For example,
book by Gazdar, Klein, Pullum, and Sag takes great pains to be as precise
r as possible, using set definitions and the like. However, because the GPSG
is build out of several very complicated components, e.g., the "Foot Feature
ion," that interact with each other, the computational import of this system is
eans clear. For a discussion that attempts to remedy some of these definitional
e [3]; indeed, as Ristad points out, many of the GSPG definitions interact with
er in such a way as to be circular. Thus, a textually formal presentation is not
ily even an easy path to salvation.
lition, these formal results of course do not settle the question of how hard it
se sentences in practice. Here there has been substantial progress in last few
ch that the gap between parsing sentences using, say, GPSG and using current
mational theory has narrowed: I would estimate that there is no more than an
magnitude difference in practical parsing times—10 seconds per sentence at
a TG-based system that maps to some level of logical form vs. 1 second or
GPSG system. The remaining difference is due mostly to our better grip on
s needed to build efficient parsers for classical context-free grammars. As these
built up for the TG designs, the gap has begun to narrow. Thus, some of the
l differences are based on the usual "lamplight" effect and not on any inherent
: we tend to look where the light is, at systems where ready-made techniques
parsing.
ll though, Cartesian computation for parsing sentences is now a practical possi-
y "Cartesian" I simply mean an account that attempts to be language universal,
struction specific, and highly modular. Many theories of language, including the
cent versions of GPSG, fall under that rubric, and in fact the models described
ould equally well be applied to general GPSG principles. The intended con-
with linguistic systems that are language specific—for example, sets of highly
ar if-then rules, or context-free rules, as in the classical ATN systems of the
mentioned just above, while there has been substantial progress in parsing sen-
sing such Cartesian models, there remains the important question of Cartesian
ation in the domain of learnability. We turn to this question in the next section.

2. HOW MATHEMATICS CAN HELP: CARTESIAN COMPUTATION AND CARTESIAN LEARNING

A key question for the "Cartesian" view of language is learning. If there is in effect just one universal grammar, then the linguists tell us that individual languages are acquired by a process of "parameter setting". The idea is that there are a small number of (binary-valued) parameters that account for the variation from language to language, and acquiring say, Japanese instead of English means to set those parameter values on the basis of (positive example) input evidence, plus learn a dictionary for that language. Learning proceeds piece by piece, switch setting by switch setting.

But what is the Cartesian computation corresponding to this process? Until recently, there have been few concrete suggestions as to how the parameter settings might actually get computed. The learning models we have for language are based on older notions of rule-by-rule acquisition and generalization. Linguists by and large have assumed the process is "easy": there is somehow a special piece of data the child hears, a trigger, that sets a relevant parameter one way or another, and that is the end of the story. For instance, one standard cited example is the "Head-Complement" parameter: the order of verb and its objects, preposition and its objects, and so forth. English is Head-first: we have verbs before their objects, prepositions before their objects. Japanese is Head-final; verbs appear at the end of phrases. The idea is that a child in an English-speaking environment will hear something like *drink the milk* and fix the order as Head-Complement once and for all.

Obviously, this is far from a complete computational model. Further, the standard methods developed in the computational learning literature over the last 10 years all assume learning over an infinite family of possibilities, whereas the parametric view assumes just a finite number of hypotheses—2^n, where n = the number of (binary-valued) parameters. Computer science has instead used the "PAC" probably approximately correct model of learning or else the models of language identification learning originally proposed by Gold [4]. What is needed is some way to bring the powerful techniques of modern computational learning theory into this finite realm. (Of course, this is not to say that such procedures are banished in the parametric view, since the parametric framework is intended to account for what is called "core" grammar—the chief construction types of a language—another ill-defined notion—rather than more "peripheral" constructs. A discussion of this possible distinction would lead us too far afield here.)

How can we bridge this gap to get a model for Cartesian computation in this domain? Recently, Gibson and Wexler ([5], GW) have attempted to formalize the notion of language learning in a (finite) space whose grammars are characterized by a finite number of parameters or 1-dimensional Boolean-valued arrays, n long. A *grammar* in this space is simply a particular n-length array of 0's and 1's; hence there are 2^n possible grammars (languages). One of Gibson and Wexler's aims is to establish that under some simple hill-climbing learning regimes, namely, single-step gradient ascent, some such natural, finite, spaces are unlearnable, in the sense that positive-only examples lead to *local maxima*—incorrect hypotheses from which a learner can never escape. More broadly, they wish to show that learnability in such spaces is still an interesting problem, in that

; a substantive learning theory concerning feasibility, convergence time, and the at must be addressed beyond traditional linguistic theory and that might even between otherwise adequate linguistic theories.

is section we show that one can considerably extend their results by adopting a mathematical model for Cartesian, or parameter-based learning. The sequel is l to exploring this model and demonstrating the pitfalls in avoiding formalization particular instance.

ıs first outline the GW model The hypothetical Gibson-Wexler parameter learner by certain central constraints, and uses an learning method that they call the ing Learning Algorithm (TLA). We also recall their definition of a *local trig-* ıor notational changes aside): given values for all parameters but one, a *local* for value v of parameters p_i, $p_i(v)$, is a sentence s from the target grammar G_T at s is grammatical iff $p_i(v) = v$. Then they state their TLA as follows:

nitialize] Step 1. Start at some random point in the (finite) space of possible arameter settings, specifying a single hypothesized grammar with its resulting xtension as a language;

ʾrocess input sentence] Step 2. Receive a positive example sentence s_i at time (examples drawn from the language of a single target grammar, $L(G_t)$), from a niform distribution on the language (we shall be able to relax this distributional onstraint later on);

ˌearnability on error detection] Step 3. If the current grammar parses (generates) , then go to Step 2; otherwise, continue.

ingle-step gradient-ascent] Select a single parameter at random, uniformly with robability $1/n$, to flip from its current setting, and change it (0 mapped to 1, 1 › 0) *iff that change allows the current sentence to be analyzed*; otherwise go to tep 2;

ourse, this algorithm never halts in the usual sense. GW aim to show under onditions this algorithm converges "in the limit"—that is, after some number ₑps, where n is unknown, the correct target parameter settings will be selected ver be changed. Their central claim is stated as their Theorem 1 (p. 7 in their ript). Note that the notion of "trigger" does not enter into the statement of the : the constraints the TLA employs, but only into their theorem.

em 1 *As long as the probability is always greater than a lower bound b ($b > 0$) e learner will 1) encounter a local trigger for some incorrectly-set parameter P, then reset P accordingly to the target value, it turns out that the target grammar ays be learned using the Triggering Learning Algorithm.*

ι the standpoint of learning theory, however, GW leave open several questions n be addressed by a more precise formalization of this model in terms of Markov (a possible formalization suggested but left unpursued in a footnote of GW). We ture the hypothesis space, of size 2^n, as a set of points, each corresponding to

one particular array of parameter settings. Call each point a *hypothesis state* or simply *state* of this space. We arbitrarily place the (single) target grammar at the center of this space. Since by the TLA the learner is restricted to moving at most 1 binary value in a single step, the theoretically possible transitions between states can be drawn as (directed) lines connecting parameter arrays (hypotheses) that differ by at most 1 binary digit (a 0 or a 1 in some corresponding position in their arrays). Recall that this is the so-called *Hamming distance* for binary arrays. We may further place *weights* on these transitions, corresponding to the nonzero b's mentioned in the theorem above; these are actually the transition probabilities between hypothesis states (we omit arcs between states that have 0 transition probabilities). Given a distribution over $L(G)$, we can further carry out the calculation of the actual b's themselves. Thus, we can picture the TLA learning space as a directed, labeled graph V with n vertices.

More precisely, we can make the following remarks about the TLA system GW describe.

Remark. The TLA system is *memoryless*, that is, given a sequence s of sentences up to time s_i, the selection of hypothesis h depends only on sentence s_i, and not (directly) on previous sentences, i.e.,

$$p\{h(s_i) \leq x_i | x(t), t \leq t_{i-1}\} = P\{x(t_i) \leq x_i | x(t_{n-1})\}$$

In other words, the TLA system is a classical *discrete stochastic process*, in particular, a discrete *Markov process* or Markov chain. We can now use the theory of Markov chains to describe TLA parameter spaces[6]. For example, as is well known, we can convert the graphical representation of an n-dimensional Markov chain to an $n \times n$ matrix, where each matrix entry (i, j) represents the transition probability from state i to state j. A single step of the Markov process is computed via the matrix multiplication $M \times M$; n steps is given by M^n.

However, as mentioned, not all these transitions will be possible in general. By assumption, only differences in surface strings can force the learner from one hypothesis state to another. For instance, if state i corresponds to a grammar that generates a language that is a proper subset of another grammar hypothesis j, there can never be a transition (nonzero b) from j to i, and there must be one from i to j. Further, by assumption and the TLA, it is clear that once we reach the target grammar there is nothing that can move the learner from this state, since all remaining positive evidence will not cause the learner to change its hypothesis. Thus, there must be a loop from the target state to itself, with some positive label b', and no exit arcs. In the Markov chain literature, this is known as an *Absorbing State* (AS). There is one other kind of Absorbing State: any state that exits only to another AS. Finally, if a state corresponds to a grammar that generates some sentences of the target there is always a loop from any state to itself, that has some nonzero probability. Clearly, can conclude at once the following learnability result, which is at once more transparent and accurate than the one given in GW:

Theorem 2 *Given a Markov chain C corresponding to a GW TLA learner.* $\exists$ exactly *1 AS (corresponding to the target grammar/language) iff C is learnable.*

⇐. By assumption, C is learnable. Now assume for sake of contradiction
ɜre is not exactly one AS. Then there must be either 0 AS or > 1 AS. In the
ːe, by the definition of an absorbing state, there is no hypothesis in which the
will remain forever. Therefore C is not learnable, a contradiction. In the second
thout loss of generality, assume there are exactly two absorbing states, the first
ɜponding to the target parameter setting, and the second S' corresponding to
her setting. By the definition of an absorbing state, in the limit C will with some
probability enter S', and never exit S'. Then C is not learnable, a contradiction.
ur assumption that there is not exactly 1 AS must be false
.ssume that there exists exactly 1 AS i in the Markov chain M. Then, by the
ɪn of an absorbing state, after some number of steps n, no matter what the
state, M will end up in state i, corresponding to the target grammar. ∎
that this approach avoids the pitfalls of proceeding without a formalization, as
ɔf given in GW (pp. 7-8 in manuscript) contains a small, but crucial, flaw:

That is, if the learner never goes through the same state twice, then she
bound to end up in the target state at some point, because the parameter
ace is finite in size. Thus the probability of avoiding the target state forever
equivalent to the probability of cycling forever through some ordered set
states (a cycle).

We can divide the parameter space into a finite set of minimal cycles,
ɪere each minimal cycle contains no cycles as a subpart. *Because the
rameter space is finite, the set of minimal cycles in the parameter space
also finite.* [our emph. rcb] For each miuimal cycle, we can now calculate
e probability that the learner remains in that cycle forever...

ɪubtle error is introduced in the second sentence of the second paragraph, in
nition of a cycle. While it is true that the parameter space is finite, it is not
ily the case that "the set of minimal cycles in the parameter space is finite."
ɜider the decimal expansion of any irrational number, say the square root of 2
2136.... This will never have any periodic sequence, yet specifies a countably
sequence of states that a 10-state Markov chain could run through, without
ɪeating a sequence. The number of such sequences is infinite, not finite, even
the Markov structure itself is finite. Thus the proof as given is incorrect. One
way to formulate the proof is by resorting to an explicit Markov formulation,
ɜsted but not executed in GW's footnote 9, and as we established above. The
simple: use formalization as a tool when it helps.

ɜ.

r the GW 3-parameter system, which they consider as the most "complex"
they study, in fact, as analyzable only by computer simulation. Its binary
ɜers are: (1) Spec(ifier) first (0) or last (1); (2) Comp(lement) first (0) or last
Verb Second (V2) does not exist (0) or does exist (1). By *Specifier* GW follow
ɪdard linguistic convention of whether there is part of a phrase that "specifies"
ɾase, roughly, like *the old* in *the old book*; by *Complement* GW roughly mean a
arguments, like *an ice-cream* in *John ate an ice-cream* or *with envy* in *green
vy*. There are also 7 possible "words" in this language: S, V, O, O1, O2,

Adv, and Aux, corresponding to Subject, Verb, Object, Direct Object, Indirect Object, Adverb, and Adjective, and 12 possible surface strings. Note that the "surface strings" of these languages are actually *phrases* such as Subject, Verb, and Object. Figure (3) of GW summarizes the possible binary parameter settings in this system. For instance, parameter setting (5) corresponds to the array [0 1 0]= Specifier first, Comp last, and −V2, which works out to the possible basic English surface phrase order of Subject–Verb–Object (SVO). As shown in GW's figure (3), the other possible arrangements of surface strings corresponding to this parameter setting include SV; SV O1 O2 (two objects, as in *give John an ice-cream*); S Aux V (as in *John is a nice guy*; S Aux V O; S Aux V O1 O2; Adv S V (where Adv is an Adverb, like *quickly*; Adv S V O; Adv S V O1 O2; Adv S Aux V; Adv S Aux V O; and Adv S Aux V O1 O2.

Suppose SOV (setting #5=[0 1 0]) is the target grammar (language). With the GW 3-parameter system, there are $2^3 = 8$ possible hypotheses, so we can draw this as an 8-point Markov configuration space, as shown in the figure above. Each labeled circle is a Markov state, a possible array of parameter settings or grammar, hence extensionally specifies a possible target language. Each state is exactly 1 binary digit away from its possible transition neighbors. Each directed arc between the points is a possible (nonzero) transition from state i to state j; we shall show how to compute this immediately below. We assume that the target grammar, a double circle, lies at the center. This corresponds to the (English) SOV language. Surrounding the bulls-eye target are the 3 other parameter arrays that differ from [0 1 0] by one binary digit each; we picture these as a ring 1 Hamming bit away from the target: [0, 1, 1], corresponding to GW's parameter setting #6 in their figure 3 (Spec-first, Comp-final, +V2, basic order SVO+V2); [0 0 0], corresponding to GW's setting #7 (Spec-first, Comp-first, −V2), basic order SOV; and [1 1 0], GW's setting #1 (Spec-final, Comp-final, $-V2$], basic order VOS.

Around this inner ring lie the 3 parameter setting possibilities, all 2 binary digits away from the target: [0 0 1], [1 0 0], [and [1 1 1] (grammars #8, 3, and 2 in GW figure 3). Finally, 3 binary digits away on a third ring lies hypothesis grammar 4, [1 0 1].

It is easy to see from inspection of the figure and without resorting to computer simulation, that there are exactly 3 absorbing states in this Markov chain, that is, states that have no exit arcs or lead directly to states with no exit arcs. One AS is the target grammar (by definition). The other two are what GW dub "local maxima," states 2 and 4, that lead only to sinks. These two states correspond to the local maxima at the head of GW's figure 4. Hence this system is *not* learnable. The computation of the local maxima, again a seemingly computationally difficult chore in the GW simulation as they note, also becomes rather more elegant in the Markov formulation.

How then do we construct this Markov diagram? We first arrange the 8 states in Hamming distance, with the target in the middle. The intuition for connecting two "adjacent" states 1 binary digit apart (i, j) is as follows: if there is some sentence (string) in the target language that is in hypothesis j but not i, then there is some positive probability of making a transition from i to j. If there are no such target strings, there is no positive evidence to move the learner from i to j. More precisely, there are exactly four possibilities for this set difference between i and j: (i) the two sets are identical, in which case the set difference is the empty set, and then there is no path

5. E. Gibson and K. Wexler, Triggers. Linguistic Inquiry, 1993, to appear.
6. D. Isaacson and J. Masden, Markov Chains, John Wiley, New York, 1976.
7. P. Niyogi and R. Berwick, The Mathematics of Triggers. Journal of Machine Learning (1993) in preparation.
8. N. Chomsky and M. Shutzenberger, The Algebraic Theory of Context-free Languages. Computer Programming and Formal Systems, North Holland, Amsterdam, 1963, 53–77.