The Complexity of Inducing a Rule from Data

MARK JOHNSON
Brown University

1 Introduction

This paper discusses the problem of identifying the conditioning context of a rule from two perspectives. As motivation it sketches an algorithm initially described in Johnson (1984) for inducing an ordered set of SPE-style phonological rules (Chomsky and Halle 1968) that account for the alternations in a set of surface phonological forms arranged in paradigm format. Then it abstracts away from the details of that algorithm, and uses the tools of computational complexity theory to characterize the computational complexity of two problems that occur as sub-problems of this and presumably most other rule discovery procedures.

The first problem asks if a rule exists that can account for data exhibiting a single alternation, i.e., if there is a rule such that this rule's context is satisfied in all of cases in which the rule must apply, and fails to be satisfied in all of the cases in which the rule does not apply. The second problem is similar to the first, except that in this case the answer is a rule with the minimum number of features. It turns out that the first problem can be answered in deterministic polynomial time, and hence is tractable in the technical sense, while the universal version (Barton, Berwick and Ristad 1987) of the second problem is NP-complete, and hence is intractable in the technical sense (assuming $P \neq NP$). The paper concludes with a discussion of these results and their implications for empiricist and rationalist models of language acquisition.

The discussion in this paper is stated in terms of the identification of the conditioning contexts for SPE-style rules because they are simple and familiar, but the results are far more general. It shows that while the problem of determining if there is some set of features that 'matches' all the members in a set *Pos* of 'positive examples' and does not match any member of another set *Neg* of 'negative examples' is tractable, the probem of finding the *minimal* set of features that distinguish *Pos* from *Neg* is in general computationally intractable.

* I would like to thank Motorola computer corporation for the donation of the 88k Delta computer used in this research.

2 A Phonological Rule Discovery Procedure

Johnson (1984) describes a procedure for discovering an ordered sequence of SPE-style phonological rules that account for certain types of phonological alternations in a data set given to it.[1] Specifically, it induces rules of the format $s_1 \rightarrow s_2 \,/\, C$, where s_1 and s_2 are segments, and C is a feature bundle identifying the contexts in which the rule applies. It does not attempt to generalize these rules to inputs and outputs specified with feature bundles (but this does not appear difficult to do). It also does not posit either insertion or deletion rules.

The procedure's input consists of a table in paradigm-format of surface phonological forms, and another table assigning each phonological segment a set of phonological features. (It's assumed that some other component of the language acquistion device has assembled these tables). Table 1 shows a typical paradigm-format input showing six alternations in Japanese surface forms.[2] Table 2 shows the assignment of features to segments given as input to the procedure.[3]

kak+u	*kai+ta*
tat+u	*tat+ta*
sin+u	*sin+da*
nom+u	*non+da*
tor+u	*tot+ta*
kog+u	*koi+da*
tob+u	*ton+da*

Table 1: A typical paradigm-format input (Japanese)

a	{ + *back*, + *low*, + *segmental*, + *voiced*, - *consonantal*, - *coronal*, - *high*, - *nasal* }
b	{ + *consonantal*, + *segmental*, + *voiced*, - *back*, - *coronal*, - *high*, - *nasal* }
d	{ + *consonantal*, + *coronal*, + *high*, + *segmental*, + *voiced*, - *back*, - *low*, - *nasal* }
g	{ + *back*, + *consonantal*, + *low*, + *segmental*, + *voiced*, - *coronal*, - *high*, - *nasal* }
i	{ + *high*, + *segmental*, + *voiced*, - *back*, - *consonantal*, - *coronal*, - *low*, - *nasal* }
k	{ + *back*, + *consonantal*, + *low*, + *segmental*, - *coronal*, - *high*, - *nasal*, - *voiced* }
m	{ + *consonantal*, + *nasal*, + *segmental*, + *voiced*, - *coronal* }
n	{ + *consonantal*, + *coronal*, + *nasal*, + *segmental*, + *voiced*, - *low* }
o	{ + *back*, + *segmental*, + *voiced*, - *consonantal*, - *coronal*, - *high*, - *low*, - *nasal* }
r	{ + *consonantal*, + *segmental*, + *voiced*, - *back*, - *coronal*, - *nasal* }
s	{ + *consonantal*, + *coronal*, + *segmental*, - *back*, - *low*, - *nasal*, - *voiced* }
t	{ + *consonantal*, + *coronal*, + *high*, + *segmental*, - *back*, - *low*, - *nasal*, - *voiced* }
u	{ + *back*, + *high*, + *segmental*, + *voiced*, - *consonantal*, - *coronal*, - *low*, - *nasal* }

Table 2: A typical segment-feature assignments input

[1] The procedure was first implemented at the University of California in San Diego in Franz Lisp on a VAX 11–750 (computer time provided courtesy of Jeff Elman); this implementation required approximately 1 hour to enumerate the rule systems generating the Japanese data presented here. It has been reimplemented on a Motorola Delta 88k in Prolog; this implementation requires 17 seconds to enumerate the same rule systems generating the same data.

[2] This data was kindly provided by Yuki Kuroda (p.c.).

[3] This feature assignment is for illustrative purposes only. In practice, a larger and more phonologically-realistic set of features is used. The implementation allows a more complex encoding which encodes syllable and morpheme structure as well.

Given these inputs, the procedure returns an underlying form for each row and column of the paradigm format input, and an ordered sequences of rules that when applied to these underlying forms yield the surface input. (In general it finds not just one solution but several, which it simply enumerates). For example, one of the solutions returned by the procedure when given the data in Tables 1 and 2 is displayed in Tables 3 and 4. Each entry in Table 4 actually abbreviates several rules, since any one of the rule contexts listed in each cell correctly conditions that alternation (e.g. the $m \rightarrow n$ rule is consistent with the data if it is conditioned by either a + *consonantal*, a + *coronal* or a + *back* following segment).

Stems:	*kak*	*tat*	*sin*	*nom*	*tor*	*kog*	*tob*
Suffixes:	*u*	*da*					

Table 3: Underlying forms for the data in Table 1 discovered by the procedure

$m \rightarrow n$ /	__ [+ *consonantal*]
	__ [+ *coronal*]
	__ [- *back*]
$b \rightarrow n$ /	__ [+ *consonantal*]
	__ [+ *coronal*]
	__ [- *back*]
$r \rightarrow t$ /	__ [+ *consonantal*]
	__ [+ *coronal*]
	__ [- *back*]
$d \rightarrow t$ /	[- *voiced*] __
$k \rightarrow i$ /	__ [+ *consonantal*]
	__ [+ *coronal*]
	__ [- *back*]
	__ [- *voiced*]
$g \rightarrow i$ /	__ [+ *consonantal*]
	__ [+ *coronal*]
	__ [- *back*]

Table 4: Ordered rules for the data in Table 1 discovered by the procedure

The procedure enumerates the rules in the reverse order in which they would be applied in a derivation. It relies on the fact that the conditioning context of the last rule to apply cannot be 'opaque', since no other rule can have applied to alter its context. The top-level structure of the discovery procedure is shown in Figure 1 (comments appear in parentheses).

If there are no alternating pairs of segments, return the data set and the reversed list of rules, otherwise:

Non-deterministically pick an alternation (say $m \sim n$).

(*Could all* ns *that alternate with* m *be derived by some rule* $m \rightarrow n / C$?)

Let *Pos* be the set of examples in which an *n* that alternates with an *m* appears. (*In Table 1, Pos* = { *#no_+da#* })

Let *Neg* be the set of examples in which an *m* appears on the surface. (*In Table 1, Neg* = { *#no_+u#* })

Determine if there is a rule context *C* such that *C* matches every context in *Pos*, and *C* does *not* match any context in *Neg*. If so, posit rule $m \rightarrow n / C$.

If there is such a context *C*, replace with an *m* every *n* that alternates with an *m* in the data, and repeat.

Figure 1: The top-level structure of the rule discovery procedure

The subprocedure for determining if there is a rule context *C* that matches every context in *Pos* and does not match any context in *Neg* is described and analysed below. At this point, all that is important is that if there is an effective procedure for doing this, then the procedure described in Figure 1 is also effective.[4]

3 Identifying a single rule's context

The discovery procedure in Figure 1 calls an auxiliary procedure to determine a rule context *C* such that *C* matches every example in *Pos* (i.e., the rule context matches all the of the phonological forms in which the rule applies), and *C* does not match any example in *Neg* (i.e., the rule context does not match any context in which the rule does not apply). Two versions of this problem are actually of interest.

- The EXISTENCE problem: does there exist a *C* such that *C* matches every example in *Pos* and none of the examples in *Neg*?
- The MINIMALITY problem: what is the 'simplest' *C* such that *C* matches every example in *Pos* and none of the examples in *Neg* (where 'simplest' is defined with respect to some 'evaluation metric')?

In order to analyse these problems further, it is necessary to specify more precisely what 'rule contexts', 'examples', and 'matching' are. In the rest of this paper, it is assumed that both rule contexts *C* and the examples in the sets *Pos* and

[4] The procedure in Figure 1 is non-deterministic, in that in general there will be several alternations in the data. But because the data set is finite, the number of alternations is also finite, and hence the non-determinism is also finite, and hence can be simulated by a deterministic algorithm in a standard fashion. Further, each (non-deterministic) iteration reduces the number of alternating pairs of segments in the data by one, and since the data set and hence the number of alternating pairs of segments are finite, eventually the procedure must terminate. Thus the procedure is effective.

$$\left\{\begin{array}{llll}
\langle -2,+consonantal\rangle & \langle -1,+back\rangle & \langle +1,+consonantal\rangle & \langle +2,+back\rangle \\
\langle -2,+coronal\rangle & \langle -1,+segmental\rangle & \langle +1,+coronal\rangle & \langle +2,+low\rangle \\
\langle -2,+nasal\rangle & \langle -1,+voiced\rangle & \langle +1,+high\rangle & \langle +2,+segmental\rangle \\
\langle -2,+segmental\rangle & \langle -1,-consonantal\rangle & \langle +1,+segmental\rangle & \langle +2,+voiced\rangle \\
\langle -2,+voiced\rangle & \langle -1,-coronal\rangle & \langle +1,+voiced\rangle & \langle +2,-consonantal\rangle \\
\langle -2,-low\rangle & \langle -1,-high\rangle & \langle +1,-back\rangle & \langle +2,-coronal\rangle \\
 & \langle -1,-low\rangle & \langle +1,-low\rangle & \langle +2,-high\rangle \\
 & \langle -1,-nasal\rangle & \langle +1,-nasal\rangle & \langle +2,-nasal\rangle
\end{array}\right\}$$

Table 5: The set of pairs encoding the example *no_+da*

Neg are sets of features, and a rule context *C* matches an example *E* iff $C \subseteq E$, i.e. if every feature in *C* also appears in *E*.[5] Both rule contexts and the examples in the sets *Pos* and *Neg* are encoded as sets of ordered pairs $\langle p,f \rangle$, where *p* is a non-zero integer indicating the 'offset' or relative position of feature *f* to the segment that the rule changes.

For example, the rule context [+ *voiced*] _ [+ *high*] is encoded as the set of pairs { $\langle -1, +\ voiced\rangle, \langle +1, +\ high\rangle$ }, and the example *no_+da* would be encoded as the set of pairs in Table 5. Since this rule context's features are a subset of the example's features, it matches the example.When rule contexts, examples and matching are defined in this way, the requirements that the rule context *C* must match every example in *Pos* and must not match any example in *Neg* are equivalent to the two following set-theoretic formulae.

1. $\forall P \in Pos, C \subseteq P$
2. $\forall N \in Neg, C \not\subseteq N$

Johnson (1984) describes an algorithm that computes the minimal *C* that simultaneously satisfy these formulae, to which the implementation-oriented reader is referred. The remainder of this paper takes a more abstract approach, and rather than analyse the properties of this or any other particular algorithm, it applies computational complexity theory to characterize the complexity of *any* algorithm that correctly answers the existence and minimality problems for this particular formalization.

4 The complexity of inducing a single rule's context from data

This section shows that the existence problem can be answered in deterministic polynomial time, whereas the minimality problem is NP-complete.

[5] There are two important things to note here. First, other definitions of matching are possible, e.g., based on unification, as in Shieber (1984) and Pollard and Sag (1989). Second, nothing here depends on features being binary valued, so the results presented below generalize to non-binary valued feature systems.

Before presenting the technical details of the proof, this section discusses in very general terms the goals of computational complexity theory. For a technical introduction see Garey and Johnson (1979), and for other linguistic applications of complexity theory see Barton, Berwick and Ristad (1987). Hopcroft and Ullman (1979) is a standard text covering automata theory, the Chomsky hierarchy, and computational complexity theory.

Many linguists will be familiar with the Chomsky hierarchy of classes of languages. This hierarchy classifies a language in terms of the type of machine that can recognize it. For example, every finite state language can be recognized by some finite-state automaton (i.e., a machine with a finite memory capacity), whereas every context-free language can be recognized by some push-down automaton (i.e., a machine with memory organized as a stack), and every context-sensitive language can be recognized by some linear-bounded automaton (i.e., a machine the size of whose memory is linearly related to the length of the string being recognized).

This classification tells us something about the architectural constraints on machines that recognize various classes of languages, but says little about the time or space resources that the machine consumes in recognizing a string.[6] For example, the language $a^n b^n c^n$ is a context-sensitive language but not a context-free language, and hence located fairly 'high' on the Chomsky hierarchy. But clearly it can be recognized in linear time and logarithmic space by checking the linear order constraints and that the numbers of a's, b's and c's are equal. This is a case where the Chomsky hierarchy gives little insight into the inherent computational complexity of a problem.

Computational complexity theory attempts to abstract away from the details of architectures and algorithms, and identify the minimum time and space resources required by *any* algorithm and machine to answer a problem. There is a corresponding hierarchy of problems, ordered by the resources they require. Here only two classes in the hierarchy will be used: P, the class of problems that can be answered by a *deterministic* automaton in time (and hence space) bounded by some polynomial function of the length of its input, and NP, the class of problems that can be answered by a *non-deterministic* automaton in time (and hence space) bounded by some polynomial function of the length of its input.[7]

Currently, it is not known whether NP actually differs from P, i.e., if there is a problem in NP that is not also in P. However, the best known algorithms for solving problems in NP on deterministic machines require exponential time, and moreover there exists a large number of NP-complete problems, i.e., problems to

[6] It does tell us something, however. Because every finite-state language is also a deterministic finite-state language, every finite-state language can be recognized in deterministic linear time. Because every context-sensitive language can be recognized in linear space, it follows that every context-sensitive language is contained in PSPACE. Garey and Johnson (1979) mention that there are context-sensitive grammars which have PSPACE-complete recognition problems.

[7] In the definition of these classes the automata are usually taken to be Turing machines. However the classes of problems that can be answered in deterministic polynomial time and in non-deterministic polynomial time is the same for most uni-processor architectures.

which any problem in NP is translatable in deterministic polynomial time. Thus if P ≠ NP, then the NP-complete problems are not in P.

Given an NP-complete problem Q_1, a proof of the NP-completeness of another problem Q_2 can be established by showing (i) that any instance of Q_1 can be translated in deterministic polynomial time into an instance of Q_2 (i.e., there is a deterministic polynomial reduction of Q_1 to Q_2), and (ii) that Q_2 can be answered by a non-deterministic Turing machine in polynomial time. A large number of problems have already been shown NP-complete; Garey and Johnson (1979) list hundreds of the more important ones.

The classes P and NP are important because problems in P are generally regarded as 'tractible', i.e., amenable to actual implementation, whereas problems in NP are generally regarded as 'intractible', i.e., in general it is not possible to solve these problems exactly on any type of computing machine.

Two problems are now presented that abstractly correspond to the existence and minimality problems for the single rule context identification problem. The examples in the sets *Neg* and *Pos* are formalized here as sets of features in order to avoid reliance on assumptions about the nature of phonological representations. The problems' complexities remain the same if an example is formalized as a string of segments, and a segment-feature assignment table is provided as input to the procedure.

This formalization of the problems relies on there being an unbounded number of features, and that any combination of features can condition any alternation. This corresponds to the UNIVERSAL problem, where a language may have an arbitrary number of 'diacritic' phonetic features that interact in essentially arbitrary fashion. Imposing a finite bound on the number of possible features and on the number of distinct rule contexts that can condition an alternation simplifies these problems enormously, so that both are in P.

For technical reasons explained in Garey and Johnson (1979), the minimality problem is posed in terms of the existence of a context set C of size less than some constant K. (An algorithm that solves this latter problem can be used to determine size of the smallest context set C using binary search, and its witness could be used to identify the corresponding context C).

RULE CONTEXT EXISTENCE
INSTANCE: A finite set F (of feature pairs), and two sets *Pos* and *Neg* of subsets of F.
QUESTION: Does there exists a subset $C \subseteq F$ such that $\forall\, P \in Pos$, $C \subseteq P$, and $\forall\, N \in Neg$, $C \not\subseteq N$?

RULE CONTEXT MINIMALITY
INSTANCE: A finite set F (of feature pairs), two sets *Pos* and *Neg* of subsets of F, and a positive integer $K \leq |F|$.
QUESTION: Does there exists a subset $C \subseteq F$ with $|C| \leq K$ such that $\forall\, P \in Pos$, $C \subseteq P$, and $\forall\, N \in Neg$, $C \not\subseteq N$?

There is an algorithm that answers the existence problem in deterministic polynomial time. To see this, first note that C is bounded above by $C' = \mathbf{I}\, Pos$. Then observe that if any subset C of C' satisfies $\forall N \in Neg, C \not\subseteq N$ then C' does also. Hence the rule context existence problem can be answered in deterministic polynomial time by first computing C' and then checking if $\forall N \in Neg, C' \not\subseteq N$.

The rule context minimality problem is clearly in NP, since a non-deterministic machine can simply 'guess' a set C of size K and then 'check' that it satisfies the constraints involving *Neg* and *Pos*. We show that it is NP-hard (and therefore NP-complete) via a reduction from HITTING SET, a problem already known to be NP-complete (Garey and Johnson 1979, page 222).

> **HITTING SET**
> INSTANCE: A set S of subsets of a finite set U, a positive integer $K \leq |U|$.
> QUESTION: Is there a subset $U' \subseteq U$ with $|U'| \leq K$ such that U' contains at least one element from each subset in S?

Given an instance of HITTING SET, we construct an instance of RULE CONTEXT MINIMALITY as follows. Let $F = U$, $Pos = \{ U \}$, and $Neg = \{ F - W \mid W \in S \}$. Clearly, $\forall N \in Neg, C \not\subseteq N$ just in case C contains at least one element from each subset of S, so the reduction is correct. The size of the rule context minimality problem is at most the square of the size of the hitting set problem, and it can be computed in deterministic polynomial time.

5 Conclusion

This paper has shown that the general problem of inducing even a single rule from a set of positive and negative examples is intractible (assuming P $\neq$ NP). This result seems to bear on the debate between rationalist and empiricist positions on language acquisition. The rationalist holds that the structure of a human language is essentially determined by the nature of the human mind, whereas the empiricist holds that the structure of a language is learned from the examples of it that the language learner is exposed to. There are several possible interpretations of this result.

One might take it as computational evidence for the rationalist position. As remarked above, the reduction relies on the unboundedness of the set of features that might potentially trigger the rule being learnt. If additional 'rationalist' constraints imposed a bound on this set—as a theory of 'universal grammar' might plausibly do—then the reduction would no longer go through, and finding even the minimal rule context that accounts for an alternation could be a computationally tractable problem

However, such a conclusion may be too strong. Although every NP-complete problem must contain 'intractible instances', some instances of an NP-complete problem may be easy, or even trivial, to decide. In fact, it can be the case that the *average* time complexity of an intractible problem grows relatively slowly with

respect to problem size. To make the argument above stronger then, one might try to show that the minimal rule context problem was intractible even in the average case. But this might be hard to do, since it would require assumptions about the nature of the distribution of instances.

An empiricist could also weaken the assumption that the language learner is required to induce the minimal rule context, and claim that all that is necessary is that the language learner induce a rule whose context feature set is *approximately* minimal. Again, some intractible problems become tractable when weakened appropriately in this manner.

Finally, an empiricist might freely admit that the human language faculty in fact can put to use grammars which its grammar learning procedure cannot acquire. Such a grammar would never be learnt, therefore would never be spoken, and hence would never appear as input to a language learner. Thus 'performance constraints' on the grammar learning procedure would impose restrictions on the class of human languages. Functionally these constraints would appear to be similar to those posited by the rationalists, and it might be difficult to distinguish the two positions.

Bibliography

Barton, E., R. Berwick and E. Ristad (1987) *Computational Complexity and Natural Language*. The MIT Press.

Chomsky, N. and M. Halle (1968) *The Sound Pattern of English*. New York: Harper and Row.

Garey, M. and D. Johnson (1979) *Computers and Intractability, A Guide to the Theory of NP-completeness*. W.H. Freeman

Hopcroft, J. and J. Ullman (1979) *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Johnson, M. (1984) "A Discovery Procedure for Certain Phonological Rules", in *The Proceedings of the 10th International Conference on Computational Linguistics*, Stanford, California.

Pollard, C. and I. Sag (1989) *Information-based Syntax and Semantics, Volume 1*. CSLI Lecture Notes, Chicago University Press.

Shieber, S. (1984) *An Introduction to Unification-based Theories of Grammar*. CSLI Lecture Notes Series, Chicago University Press.