Sparse Representations for Fast, One-Shot Learning

Kenneth Yip Gerald Jay Sussman
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

November 26, 1997

Abstract

Humans rapidly and reliably learn many kinds of regularities and generalizations. We propose a novel model of fast learning that exploits the properties of sparse representations and the constraints imposed by a plausible hardware mechanism. To demonstrate our approach we describe a computational model of acquisition in the domain of morphophonology. We encapsulate phonological information as bidirectional boolean constraint relations operating on the classical linguistic representations of speech sounds in term of distinctive features. The performance model is described as a hardware mechanism that incrementally enforces the constraints. Phonological behavior arises from the action of this mechanism. Constraints are induced from a corpus of common English nouns and verbs. The induction algorithm compiles the corpus into increasingly sophisticated constraints. The algorithm yields one-shot learning from a few examples. Our model has been implemented as a computer program. The program exhibits phonological behavior similar to that of young children. As a bonus the constraints that are acquired can be interpreted as classical linguistic rules.

Keywords:

morphophonology, sparse representation, fast learning, rule induction, language learning

Contact Information:

`yip@ai.mit.edu`, 617-253-8581, NE43-433, 545 Technology Square, MIT, Cambridge, MA 02139.

1 Introduction

The ability to learn is a hallmark of intelligence. Humans rapidly and reliably learn many kinds of regularities and generalizations. Any learning theory must explain the search and representation biases that make fast and robust learning possible. We propose a model of incremental one-shot learning that exploits the properties of sparse representations and the constraints imposed by a plausible hardware mechanism.

Our particular system design is consistent with what you would expect of computer engineers. We think naturally in terms of shift registers, bidirectional constraint elements, bit vectors, and greedy learning algorithms. We envision these mechanisms to be part of a set of compatible components (as those in a TTL data book) that can be mixed and matched to construct a variety of learning models.

The performance module is implemented as a network of constraint elements and several data registers. The constraint elements attempt to fill in missing information in the data registers. This mechanism allows information to be used multidirectionally; there are no distinguished input or output ports. There are two benefits of organizing the performance module around constraints. Missing data can often be filled in, conferring a degree of noise immunity. Also, there is no need for additional mechanisms to maintain correlations between separate representations that mediate input and output.

The learning module incrementally builds the performance module by abstracting regularities from the data. Both the raw data and the generalizations are represented as vectors in a high-dimensional feature space. These vectors are implemented as the constraint elements of the performance module.

A key to fast learning is sparseness of the high-dimensional feature space. This allows simple hyperplanes to separate clusters of examples. As a consequence, a parsimonious description (such as a minimum length description) covering a large region of the generalization space can be induced from a few examples. This sparseness property makes the acquisition of regularities highly effective.

We demonstrate our model in the domain of morphophonology—the connection between the structure of words and their pronunciation. We attack this problem for two reasons. First, the problem is relevant to the foundation of cognitive science, as evidenced by the controversy between the supporters of symbolic AI and connectionist AI.[1] Second, learning phonological regularities is an example of a class of induction problems which presents special challenges to a learner who must form valid generalizations on the basis of a few positive examples and no explicit corrections for wrong behavior.

[1] See [16, 13, 12, 14, 6].

Sparseness of the representation is partially a consequence of the fact that phonemes are not atomic but are encoded as combinations of elementary *distinctive features*. These features can be thought of as abstract muscular controls of the speech organs. The phonemes that are actually used in human languages are only a few of the possible phonemes that can be constructed from the distinctive features.

2 Human learning

Almost every child learns how to speak and to understand his native language. At an appropriate stage of development a child learns vocabulary with amazing speed: typically a child learns many new words, and their correct usage, each day. The learning is efficient, in that a child does not need to hear the same words repeated over and over again or to be corrected very often. Thus learning language must be easy, but we do not have effective theories that explain the phenomenon.

The mystery deepens when we notice that children learn many new words without ever hearing them. In a classic experiment by Berko [2], a number of English-speaking children were shown representations of a fanciful being called a "wug." When asked to say something about a situation with more than one of these beings, the children correctly pluralized the novel word to make "wugz" (not "wugs"). In another experiment [9], Marcus et. al. showed that young children who first use an irregular verb properly (such as "came") would later err on the same verb (by supplementing "came" with "comed") before they use the verb correctly again. Even more striking is that children who have mastered the English pluralization rules can produce the correct plural for a new word ending with a sound not present in English such as *ch* in the name of the German composer *Bach*. If the name is pluralized, they add the unvoiced *s*.

Thus children reliably exhibit behavior that indicates that they have made generalizations that linguists describe with rules. Moreover, these generalizations are not simply stated in terms of an arbitrary list of speech sounds, but in terms of significant features shared by classes of speech sounds.

Although much research has been done in this problem, it is fair to say that no previous learning theory can account for the phonological behavior observed in children, in a way that is consistent with the regularities that linguists have isolated.

3 Our approach

We focus on the acquisition of inflectional morphophonology (such as pluralization and verbal inflection) where developmental data are abundant. In and of itself inflectional morphophonology is not particularly significant. The reason we study this problem so intensely is that it is a simple case of the regularities that are found in most natural languages.

We present a theory of how to make and use phonological generalizations. Our theory explains how the generalizations can be learned from a few randomly chosen examples. For example, after seeing a dozen common nouns and their plurals, our mechanism incorporates constraints that capture English pluralization rules: (1) Nouns ending in one of the "hissing" sounds ([s], [z], [sh], [ch], [zh] and [j]) are pluralized by adding an additional syllable [I.z] to the root word, (2) Nouns ending in a voiced phoneme (other than the hissing sounds) are pluralized by adding a [z] sound, and (3) Nouns ending in a voiceless consonant (other than the hissing sounds) are pluralized by adding a [s] sound.

Our theory of acquisition differs significantly from those based on statistics (such as [16, 8]). It is incremental, greedy, and fast. It has almost no parameters to adjust. Our theory makes falsifiable claims about the learning of phonological constraints: (1) that learning requires very few examples—tens of examples in a few steps as opposed to thousands of examples trained in thousands of epochs [7], (2) that the same target constraints are learned independent of the presentation order of the corpus, (3) that effective learning is nearly insensitive to the token frequency,[2] and (4) that learning is more effective as more constraints are acquired.

We do not attack the problem of how an acoustic waveform is processed. We start with an abstraction from linguistics (as developed by Roman Jakobson, Nikolai Trubetzkoy, Morris Halle, and Noam Chomsky) [3]: Speech sounds (phonemes) are not atomic but are encoded as combinations of more primitive structures, the distinctive features. The distinctive features refer to gestures that the speech organs (such as tongue, lips, and vocal cords) execute during the speaking process.[3] The feature system of Chomsky and Halle uses 14 binary-valued distinctive features. Each phoneme is uniquely characterized by its values on the distinctive features. The distinctive-feature representation is extremely sparse: English uses only 40 or so phonemes out of the thousands possible feature combinations, and no human language uses many

[2]Clahsen et. al. showed that the German *-s* plural acts like a regular plural even though it applies to a tiny fraction of the nouns [4].

[3]For example, the voicing feature refers to the state of the vocal cords. If a phoneme (e.g., [z]) is pronounced with vibration of the vocal cords, the phoneme is said to be [+voice]. On the contrary, an unvoiced phoneme (e.g., [s]) is said to be [−voice]. The plus indicates the presence of voicing, while the minus indicates its absence.

more than 100 phonemes.

The representation of speech sounds as a sequence of discrete features is a crude approximation to what physically takes place during speech. We make two idealizations. First, the distinctive features are discretized to binary values. Second, the distinctive features are assumed to change synchronously. Although these idealizations are not true—the distinctive features are really analog signals and the durations of the signals need not be aligned perfectly—they are reasonable first approximations for building a mechanistic model to understand how phonological regularities might be acquired[4].

Our use of vectors of distinctive features to represent the phonemes does not imply that we believe that the recognition of speech from the acoustic waveform passes through an intermediate stage where the features are recognized and then the phonemes are assembled from them. Perhaps other mechanisms[5] are used to obtain the phonemic representation from the acoustic waveform, and the distinctive feature bit representation is a result of this process, not a stage in it.

4 A Mechanistic Performance Model

Our performance model is envisioned as a hardware mechanism, limiting the range of behavior that can be developed. A mechanism that exhibits human-like phonological behavior gives us an upper limit on the complexity necessary to produce that behavior. By restricting ourselves to a simple mechanism, limited in the kinds of parts that we may postulate and in the ways they may be connected, we construct a robust theory. Our aim is to show that phonological behavior is a natural consequence of the organization of the hardware.

The mechanism consists of data registers and constraint elements. The data registers hold the state of the computation as linguistic events are processed. (See Figure 1.) The linguistic information is described in terms of boolean features (bits). The constraint elements embody phonological knowledge relating sound and meaning patterns. For example, the plural constraints distinguish the variants of the plural morphemes ([z], [s], and [I.z]) conditioned by the last phoneme of a common noun.

The constraint elements implement boolean relations among the values of the

[4]Modern phonology postulates more elaborate representation devices such as multiple tiers and metrical grids. See [5]. These devices describe phonological phenomena that we do not address.

[5]For example, the phonemes may be extracted from the acoustic waveform using statistical techniques on other features, such as cepstrum coefficients [15]. We understand that current techniques cannot reliably extract such information from noisy continuous speech. Our only concern here is that the resulting phonemes are represented in terms of some set of distinctive features similar to the SPE set (developed in *The Sound Pattern of English* [3]).

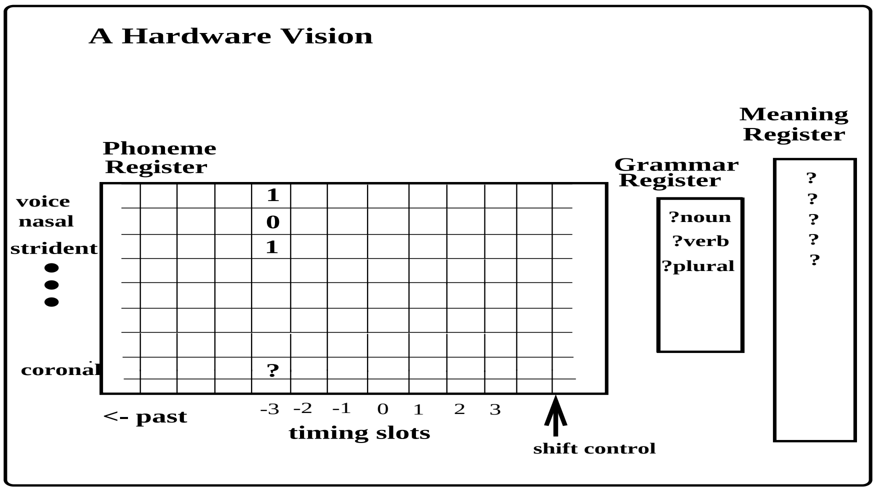

Figure 1: The hardware mechanism consists of three data registers. In the Phoneme Shift Register, each vertical stripe represents a time slot. There are slots for future phonemes (positive time labels) as well as past phonemes (negative time labels). Each horizontal stripe represents a distinctive feature bit. For example, if the phoneme in time slot -3 is known to be a voiced non-nasal then in the column labeled -3 the voice entry would be 1 and the nasal entry would be 0. If the phoneme is known to be strident, the strident entry would be 1. If a feature is unknown, as in the coronal bit, we leave a question mark. The Grammar Register contains bits describing the grammatical status of the phoneme sequence. The Meaning Register contains meaning bits.

features in the registers.[6] If there is sufficient match between the features in the registers and those enforced by a particular constraint element, that element becomes active. An active element fills in details by assigning values that satisfy the constraint to bits in the registers that were previously unspecified. If the information in the registers is inconsistent with the relation enforced by an active constraint element, this conflict is noted.

A linguistic event might be the hearing or speaking of a word. An event is described by three types of information: sound, grammar, and meaning. The sound pattern of a word, represented as a sequence of discrete phonemes, is stored in a shift register called the *phoneme register*. Each time slot of the phoneme register holds a vector of 14 binary-valued distinctive features representing a particular phoneme. (See Figure 2.) As the speech sound is heard, the phoneme sequence is shifted. The grammatical information of the word (such as its part of speech, number, and gender)

[6]We do not yet have a complete hardware model for constraint elements. We are imaging that each constraint element has thousands of ports that connect to all the slots of the shift register and the feature bundles within each slot.

is stored as a vector of grammatical features in the *grammar register*. The *meaning register* contains a set of bits that uniquely identify the meaning of the word. Our learning theory does not depend on the assignment of the meaning bits.

	[ae]	[p]	[l]	[z]	"apples"
syllabic	1	0	0	0	
consonantal	0	1	1	1	
sonorant	1	0	1	0	
high	0	0	0	0	
back	0	0	0	0	
low	1	0	0	0	
round	0	0	0	0	
tense	0	1	0	1	
anterior	0	1	1	1	
coronal	0	0	1	1	
voice	1	0	1	1	
continuant	1	0	1	1	
nasal	0	0	0	0	
strident	0	0	0	1	
TIME $\cdots$	-3	-2	-1	0	1 $\cdots$

Figure 2: The sound pattern of a word is represented by a sequence of phonemes. For example, the word "apples" consists of four phonemes. Each phoneme is uniquely characterized by its values on the 14 distinctive features. Phonetic symbols are enclosed in square brackets. For example, [ae] is the symbol for the low [+low], front [−back], lax [−tense] vowel in "apples." Each feature vector is indexed by a time instant. The column labeled by time $= 0$ corresponds to the most recently heard phoneme. Phonemes with negative time indices are already heard.

The "bits" in the registers have four possible states {0, 1, ?, *}. The bits can be set by an external linguistic event or by constraint relations. If the value of a bit is currently unknown it contains an unknown symbol (?). If a bit is asserted to be both 1 and 0 because of a disagreement among the constraints it participates in, it is in the conflict state, which we denote by (*).

The constraint relation enforced by a constraint element is represented by a bit vector. We refer to these bit vectors as *classifiers*. A classifier is a finite string over the three-symbol alphabet $0, 1, -$. A "1" (or "0") typically represents the presence (or absence) of a characteristic. A "−" means don't care, i.e., the bit's actual value does not matter.

There are two types of classifiers: *rote-classifier* and *rule-classifier*. (See Figure 3.)

The rote-classifiers capture specific correlations among the bit patterns in the data registers. For example, a rote-classifier for "apple" enforces a certain constraint among the phonemes [ae.p.l] in the phoneme register, the [+noun,−verb,−plural...] features in the grammar register, and the bits in the meaning register. Rule-classifiers capture the regularities among rote-classifiers; they can be interpreted as general phonological constraints. Rule-classifiers are the basis for predicting responses to novel words. If the prediction is correct, there is no need for rote-learning the particular correlations in question.

5 Phonological Behavior From Competing Constraint Elements

The basic execution cycle of the performance model consists of three steps implementing a constraint propagation process:

1. Activate the most excited constraint element.
2. Enforce bit patterns in the data registers according to the relation the constraint element represents.
3. Deactivate previously excited constraint elements that no longer match the register contents.

The cycle is repeated until the data registers reach a quiescent state.

A constraint element is excited if its excitation strength exceeds a certain threshold. The excitation strength is measured by the Hamming distance between the classifier of the constraint element and the bit patterns in the data registers. Multiple competing constraint elements can be excited at any instant. When an excited constraint element is activated, it gains exclusive control over the data registers, preventing other constraint elements from writing over the register contents. As the register contents change, an activated constraint element might be deactivated and relinquish its control.[7]

The constraint propagation process is not committed to using any particular classifier in a predetermined way. A classifier may use partial semantic information to enforce constraints on the phoneme register. It may also use partial phonological information to infer semantic information. The propagation process can be freely intermixed with the addition of new constraints and modification of the existing ones.

[7]The hardware model assumes the constraint propagation step is fast compared to the rate of incoming phonemes.

Classifiers: Bit Vector Representation of Constraint Relations

rote-classifier-apple

```
Phonemes: [ae.p.l]
Grammar:  [+noun -verb -plural ...]
Meaning:  [+red +round +edible +fruit ...  ]
```

rule-classifier-voiced-plural

```
Phonemes:        [dc.dc.dc.[+voice].z]
Source-Grammar:  [+noun -plural]
Target-Grammar:  [+noun +plural]
Control:         [shift
                        direction    :  left
                        start loc    :  0
                        unit         :  1
                        fill symbol  :  ?]
                 [unlock
                        phoneme slot 0:  no
                        phoneme slot -1:  no
                        phoneme slot -2:  no]
```

Figure 3: Two types of classifiers. We use symbolic notations to simplify the description of a classifier. The notation [ae.p.l] refers to the 42 bits (3×14) representing the distinctive features of the phonemes enclosed in the square brackets. The symbol "dc" for a phoneme abbreviates a string of 14 don't-care bits. The notation [+noun] indicates that the noun bit in the classifier is on. *Top*: The rote-classifier "apple" correlates the sound pattern [ae.p.l] with the grammar and meaning bits of the word. *Bottom:* The rule-classifier "voiced-plural" relates a singular (i.e. [−plural]) common noun ending in a voiced phoneme with its plural form. If the bit pattern in the grammar register matches the source-grammar component of a rule-classifier, the constraint element described by the rule-classifier produces a plural form by shifting the phoneme register left one slot (as specified in the control component of the rule-classifier description) and filling the unknowns in the terminal slot with a [z] phoneme. If the pattern in the grammar register matches the target-grammar component, the constraint element produces a singular form by reversing the control actions. The unlock privilege grants (or refuses) write access by the classifier to the last 3 phoneme slots.

To illustrate how competing constraint elements can cooperate to enforce phonological and semantic constraints, we examine a simple situation. Assume that at some time the meaning identifier describing a red, round, edible fruit appears in the meaning register. These bits might have been set by a vision module that has recognized an apple or a picture of an apple, or perhaps by an olfactory module that has recognized the smell of an apple. We also assume that for some reason the plural bit of the grammar register is set, perhaps because there are two apples.

Suppose also that at this point the performance model has two classifiers: a rote-classifier for the apple constraint, which captures the correlation between the phoneme sequence [ae.p.l], the [+red +round +edible +fruit] meaning, and the [+noun −verb −plural ...] grammar, and a rule-classifier for the voiced-plural rule, which captures the phonological rule that the plural of a noun ending in a voiced phoneme is formed by appending a [z] phoneme to the noun.

The situation at the initial time is depicted in Figure 4. The initial situation triggers a sequence of classifier actions to fill the slots of the phoneme register with the sound sequence corresponding to the plural of "apple." The content of the meaning register is sufficient to activate the constraint element described by the rote-classifier for apple. The apple constraint then attempts to set as many unknown bits as it can. It asserts the bits describing the phoneme sequence into the phoneme register. This encounters no resistance because all of those bits were initially unknown. The apple constraint also sets some grammar bits. The noun bit is turned on and the verb bit is turned off. However, a conflict arises over the setting of the plural bit. The picture of two apples forced the plural bit on, but the apple constraint is trying to assert a singular. Figure 4(b) shows the contents of the registers at this point.

All the phoneme bits from the apple constraint are now in the phoneme register. The fact that there is a noun under consideration (+noun in the grammar register), that there is a conflict over the plural bit, and that the terminal [l] phoneme is [+voice] is a sufficient trigger to activate the constraint represented by the voiced-plural classifier. It sends a shift left signal to the phoneme register, moving the phonemes ae.p.l to less recent positions, and locking the determined phonemes so that they cannot change. The most recent phoneme slot is filled with unknowns, which are certainly allowed to change. The apple constraint now becomes less excited because the values it would like in the phoneme register are all in conflict with the ones that are there. The voiced-plural constraint now fills the unknowns in the current phoneme slot with the phoneme [z]. See Figure 4(c).

As the apple classifier is deactivated, it drops its attempt to set the plural bit to 0. The noun, the verb, and the plural bits retain their last values. The plural bit is still in conflict, but it will put up no resistance if another constraint tries to turn it on. In particular, the excited voiced-plural rule-classifier restores the plural bit to 1. At this point the system reaches a quiescent state (Figure 4(d)) with a consistent representation of the plural noun pronounced [ae.p.l.z] in the phoneme register.

Constraint elements infer meaning from sound as well as sound from meaning. For example, using the same two classifiers as before, the performance model can fill in the grammatical and semantic details as the sound pattern of "apples" is shifted into the phoneme register. The same mechanism of constraint elements and shift registers is effective for both production and comprehension of a word.

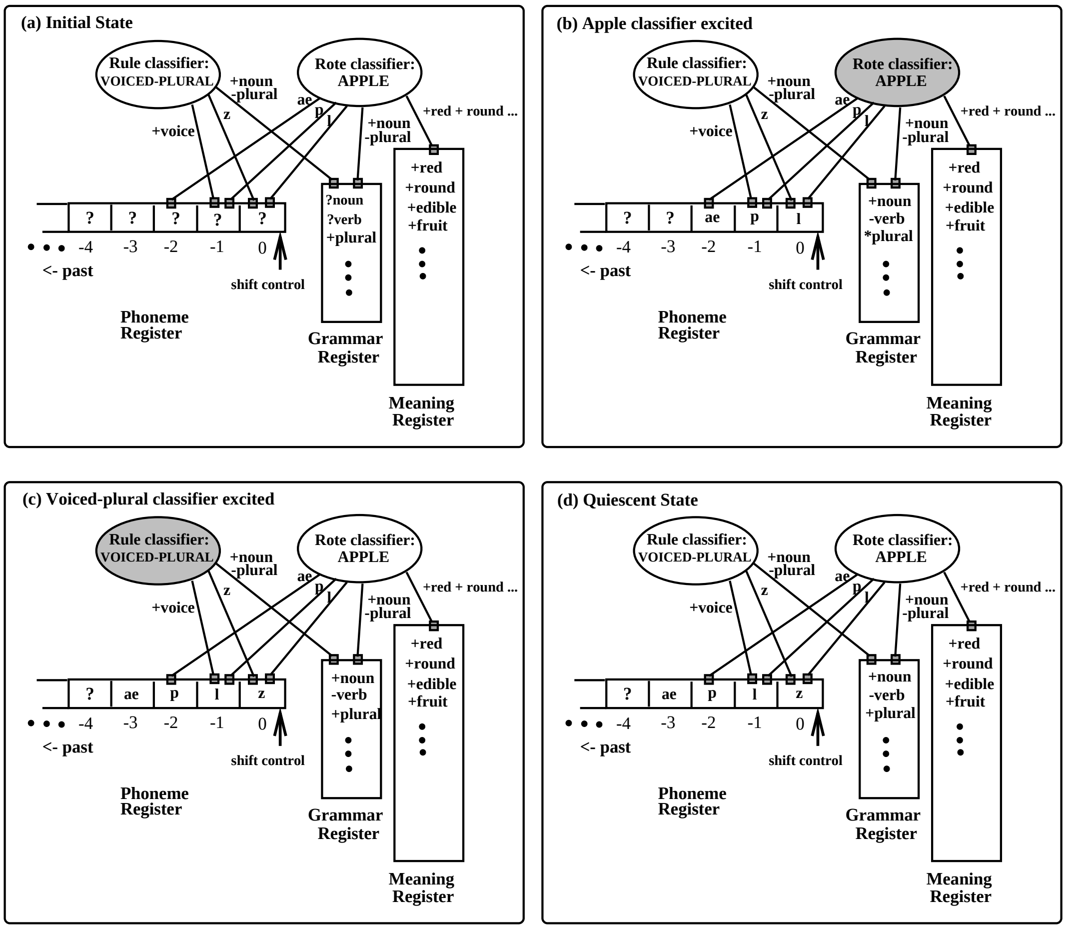

Figure 4: Generating sound from meaning. (a) *Initial state*: The performance model has two classifiers: a rote-classifier for the apple constraint and a rule-classifier for the voiced-plural constraint. An event fills the meaning register with features describing an apple, and the grammar register with the [+plural] feature. (b) *Apple constraint excited:* The apple constraint fires and writes the sound sequence [ae.p.l] into the phoneme register. Some unknown grammatical bits (such as ?noun and ?verb) are also filled by the apple classifier. Note that a conflict arises over the assignment of the plural bit. (c) *Voiced-plural constraint excited:* The voiced-plural constraint sends a shift left signal to the phoneme register, and fills the unknown terminal slot with the [z] phoneme. The voiced-plural constraint also restores the conflict plural bit to 1. The apple constraint is deactivated. (d) *Quiescent state:* The system reaches a consistent state with the pronunciation of "apples," [ae.p.l.z], in the phoneme register. No new constraints are excited.

6 Learning Classifiers

In a full system, there are many classifiers. How are the classifiers learned?

Initially the learner has no classifiers. The learner must acquire the classifiers from example words presented to it. We assume a sequence of words is presented to the learner. Each presentation of a word triggers an event that fills the data registers with the appropriate sound, grammar, and meaning information. If the learner can fill in the details of the word without error, then it proceeds to the next word. The learner might fail either because there are not any applicable classifiers or because competing classifiers fight to assert inconsistent values in the registers. In the first failure situation, a new rote-classifier is created to remember the bit patterns in the data registers. In the second failure situation, the fighting classifiers are incrementally refined.

Let us consider a simple example to illustrate the basic operations of the learning procedure. Suppose that to begin with the learner has no classifiers and is presented four noun pairs and one verb pair in random order: cat/cats `[k.ae.t.s]`, dog/dogs `[d.).g.z]`, duck/ducks `[d.^.k.s]`, gun/guns `[g.^.n.z]`, and go/went `[w.`ϵ`.n.t]`. A rote-classifier is created for each of the words.

The learning algorithm first finds correlations among pairs of rote-classifiers that have the same meaning bits. One element of each pair of correlated classifiers is labeled *positive* and the other *negative*.[8] The criterion for this labeling is arbitrary, but consistently applied across all pairs. The pairs are grouped into sets. Each set is defined by a grammar type and a shift specification that maximally aligns the positive and negative element of a pair. The learning algorithm then attempts to produce a summary description of the pairs in each set.

An example of how the algorithm works is given in Figure 5. The rote-classifiers "cat" and "cats" are correlated because they share the same meaning bits. There are 10 such correlations. These 10 rote-classifier pairs are divided into two sets: the first one is related to changes in the plural bit, and the second to changes in the past-tense bit. The algorithm looks for a summary description of the phoneme bit pattern that covers all the rote-classifiers with the [+plural] feature (the positive example) and avoids all the ones with [−plural] feature (the negative examples). A description is said to *cover* an example if the example is consistent with all the conditions in the description.

Starting with the phoneme bits of a rote-classifier as the initial description, the generalization algorithm performs a specific-to-general search in the space of possible descriptions.[9] For example, an initial description, the seed, might be the phoneme bits for "cats." The seed is a bit vector of 56 bits (14 bits for each of the 4 phonemes [k.ae.t.s]), which can be thought of as a logical conjunction of boolean features:

---

[8]These negative examples are not externally provided "near misses" to inhibit incorrect behavior. Here the positive and negative labels are arbitrary distinctions.

[9]Our generalization algorithm differs from the version space algorithm [10] in two respects. First, our algorithm does not maintain all the most general and most specific generalizations consistent with the current set of examples. Second, our algorithm handles disjunctive generalizations and noise.

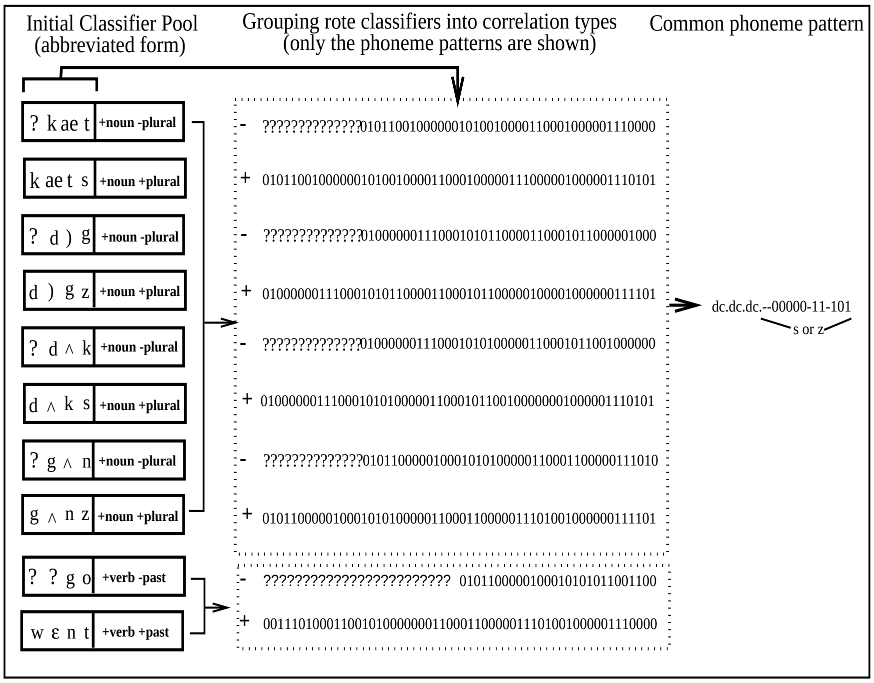

Figure 5: Learning of classifiers consists of two steps: (1) Grouping of rote-classifiers into correlation types (middle column), and (2) Generalizing rote-classifiers belonging to the same correlation type (right column). In this example, the initial classifier pool has 10 rote-classifiers (shown in an abbreviated form in the left column). The rote-classifiers are divided into two correlation types. The first correlation type (relating a common noun and its plural) has 8 rote-classifiers. Each rote-classifier is labeled with a plus to indicate that it is the plural form or a minus to indicate the singular form. The learner finds a rule-classifier whose sound pattern (right column) covers all positive examples and avoids the negative ones. The second correlation type (relating a verb stem and its past) has two rote-classifiers. The learner will not generalize this group of classifiers until more such examples have been accumulated.

```
01011001000000101001000011000100000011100000010000001110101
<------ k ---><----- ae  --><----- t ----><----- s ---->
```

The generalization space of possible phoneme bits for a classifier is $O(3^n)$, where n is the number of phoneme bits. (See Figure 6.) For example the generalization space for classifiers with four phonemes contains $O(3^{56})$ instances. To explore this huge space, the generalization process relies on three search biases:

1. Whenever possible it revises the current best classifiers instead of starting from scratch,

2. It prefers classifiers that contain the most recently heard phonemes, and

3. It is non-aggressive: the search terminates on the first few classifiers found to

cover a given set of correlations without deliberately looking for the minimal classifiers (i.e., those with the largest number of don't cares).

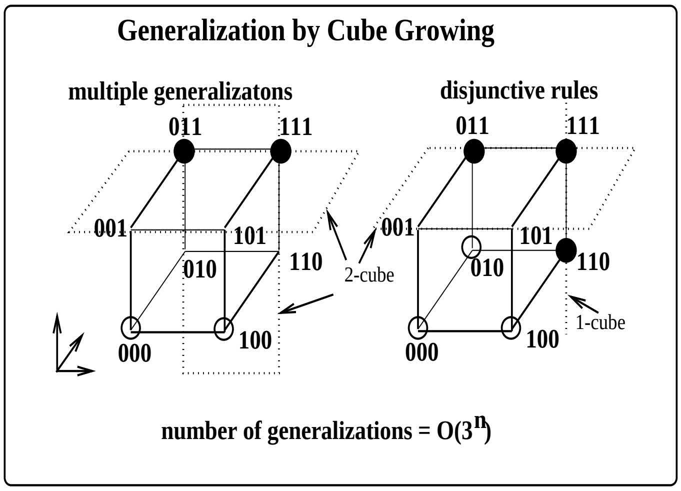

Figure 6: Classifier generalization as cube growing. A relation with n boolean variables defines an n-dimensional instance space with 3^n possible instances. The positive examples (solid dots) and negative examples (circles) occupy the vertices of an n-dimensional cube. Generalization can be thought of as finding a collection of m-cubes ($0 \leq m \leq n$) covering the positive ones without overlapping the negative ones. A 0-cube is a point, 1-cube is a line, and so on. There may be multiple m-cubes that cover the same positive examples (as shown by the two 2-cubes in the left diagram). It may also require more than one m-cube to cover the positive examples (as shown by the 1-cube and 2-cube in the right diagram). The generalization algorithm uses a beam search with inductive biases to find disjunctive generalizations.

The generalization procedure is a beam search with a simple goodness function. The best k candidate generalizations are retained for further generalizations. The goodness of a cube is equal to the sum of Pc and Nc, where Pc is the number of positive examples the cube covers, and Nc is the number of negative examples it does not cover. To break ties in the goodness score, the search prefers larger cubes with higher Pc. See Figure 7 for details.

At each iteration the algorithm generates new candidate generalizations by raising the phoneme bits (i.e. changing 0's and 1's to don't cares), one or two bits at a time. The phonemes are ordered by recency. The bits of the least recently heard phoneme are raised first. The search terminates when either all positive examples are covered or a negative example is covered.[10]

[10]The generalization algorithm has a dual, the specialization algorithm, which refines overly-general rule-classifiers to avoid negative examples.

Procedure GENERALIZE
Input: PSET is a set of positive examples.
NSET is a set of negative examples.
Output: GSET is a set of generalizations.
Representation:
All examples and generalizations are represented as bit vectors. Each bit vector consists of multiple slots. Each slot is an unordered set of distinctive features.
Algorithm:
The generalization space is the set of 3^n bit vectors where n is the length of the bit vector. GENERALIZE starts from a seed, a randomly chosen element of PSET, and finds generalizations by a beam search with width k. The beam search aims to minimize a cost function.
Let G be a candidate generalization. The beam search is defined by the following functions:

1. Cost Function = Pc + Nc

   where Pc is the number of elements in PSET not covered by G, and Nc is the number of elements in NSET covered by G. Ties are broken by preferring generalizations with more don't cares.

2. Successor Function

   Let a *definite slot* be a slot with at least one bit that is not a don't care. Define the successors of G to be the set of all bit vectors obtained by setting one or two bits of the leftmost definite slot of G to don't cares.

3. Termination Function

   Let GSET be the set of the current k generalizations being considered. The search terminates if, for all G $\in$ GSET, G satisfies one of the three conditions:

   (a) G covers the entire PSET.

   (b) G overlaps NSET.

   (c) All the successors of G overlaps NSET.

Figure 7: Algorithmic description of the generalization procedure. The beam width k is set to 2 for all the examples and experiments described in this paper.

The search (with beam width $k = 2$) eventually produces a description G that covers all four positive examples and avoids all four negative examples. The description says that all positive examples end with either the [s] or [z] phoneme.[11]

```
G: [dc.dc.dc.{s,z}]
```

The next step in the summarization process is to verify the covering description. The description G is overly general because applying it to the negative examples gives not only the correct plural forms (such as [k.ae.t.s]) but also incorrect ones (such as

[11] The symbol "dc" abbreviates 14 don't-care bits.

*[k.ae.t.z]). The incorrect ones are treated as near misses (i.e., negative examples that are slightly different from the positive ones). The learning algorithm assumes a general uniqueness heuristics: there is only one way to satisfy the requirements. Since [k.ae.t.s] is the known positive example, the system-generated [k.ae.t.z] must be incorrect. Near misses greatly speed up the discovery of correct generalizations. [12]

The generalization algorithm is re-invoked with the addition of these new negative examples:

```
Seed     : [k.ae.t.s]
Positives: [k.ae.t.s] [d.).g.z] [d.^.k.s] [g.^.n.z]
Negatives: *[k.ae.t.z] *[d.).g.s] *[d.^.k.z]
           *[g.^.n.s] [k.ae.t] [d.).g] [d.^.k] [g.^.n]
```

This time the search results in a disjunction of three generalizations G1, G2, and G3:

```
G1:  [dc.dc.[-voice].s]
G2:  [dc.dc.[+voice,-strident].z]
G3:  [dc.dc.[+voice,-continuant].z]
```

The generalization G1 covers two positive examples: "cats" and "ducks." G1 describes a correlation between the penultimate voiceless phoneme and a terminal [s] phoneme. The generalizations G2 and G3 overlap in their coverings. They both cover the remaining two positive examples: "dogs" and "guns." G2 says that a terminal [z] phoneme is preceded by a phoneme that has the [+voice] and [−strident] features.[13] G3 correlates a terminal [z] phoneme with a preceding voiced non-continuant. [14] The three generalizations are verified as before. However, this time the generalizations are consistent: there are not any new exceptions or near misses. Note that after seeing only 4 positive examples, the learner is able to acquire constraints on the plural formation that closely resemble those found in linguistics texts[1]. These rule-classifiers are now available for constraint propagation, and are subject to further refinement when new examples appear.

[12] Winston [17] emphasized the usefulness of near misses in his ARCH learning program. In our program, the near misses are not supplied by a teacher or given in the input. They are generated internally.

[13] The strident feature refers to noisy fricatives and affricates. In English there are eight stridents: [s,z,f,v,ch,j,sh,zh].

[14] A phoneme is a non-continuant or a stop if the passage of air through the month is stopped completely for a brief period. [b,d,g,p,t,k] and the nasals [m,n] are examples of stops.

7 Experimental Results

The corpus used to develop the learning program comes from the CELEX lexical databases of English (version 2.5) obtained from the Linguistic Data Consortium. We select 250 words (50 common nouns and 200 verbs) that first-graders might know. The corpus includes most of the regular and irregular verbs used in the psycholinguistic experiments of Marcus et. al. [9] on English tenses.

The data record for each word in the corpus consists of five pieces of information: (1) word identifier, (2) word spelling, (3) a unique meaning identifier (e.g., "cat" and "cats" have the same meaning id, but "cat" and "dog" do not), (4) its pronunciation as a sequence of phonemes, (5) its grammatical status (16 grammatical bits indicating whether the word is a noun or verb, singular or plural, present or past, etc.). The spelling information is not used by the learner; it is only for the human experimenter to read.

word id	spelling	meaning id	phonetic symbols	grammar
12789	cat	6601	k.ae.t.	Noun Sing ...
12956	cats	6601	k.ae.t.s.	Noun Plu ...
25815	dog	13185	d.).g.	Noun Sing ...
25869	dogs	13185	d.).g.z.	Noun Plu ...

The data records are pre-processed to produce bit vector inputs for the performance model and learner. The output of the performance model and learner is a set of bit vectors that typically have a straightforward symbolic interpretation.

To test the performance of the program on past-tense inflection, we use the same dataset that MacWhinney [8], Ling [6], and Mooney and Califf [11] used. The dataset contains approximately 1400 stem/past-tense pairs. We randomly choose 500 verb pairs as the test set. The test set is disjoint from the training set. The program is trained on progressively larger samples from the training set. Starting from a training sample of 10 examples, we measure the predictive accuracy of the program on the test set at 10-example intervals. The mean and standard deviation of the accuracy are calculated over 5 trials. During each trial, the choice and order of the training examples presented to the program are randomized.

In all the experiments below, we use the same parameter settings for the beam search width ($k = 2$) in the generalization algorithm and the excitation threshold for classifiers. The results are not sensitive to the particular parameter settings.

Experiment 1: Learning regular past-tense

The first experiment tests the performance of the program on past-tense inflection using regular verbs only. Although the task is simplistic, it does allow quantitative comparisons with previous work. Figure 8 presents the learning curves from four programs: K&G (our program), FOIDL (a program based on inductive logic programming) [11], SPA (a program based on decision trees) [6], and M&L (a program based on artificial neural network) [8].

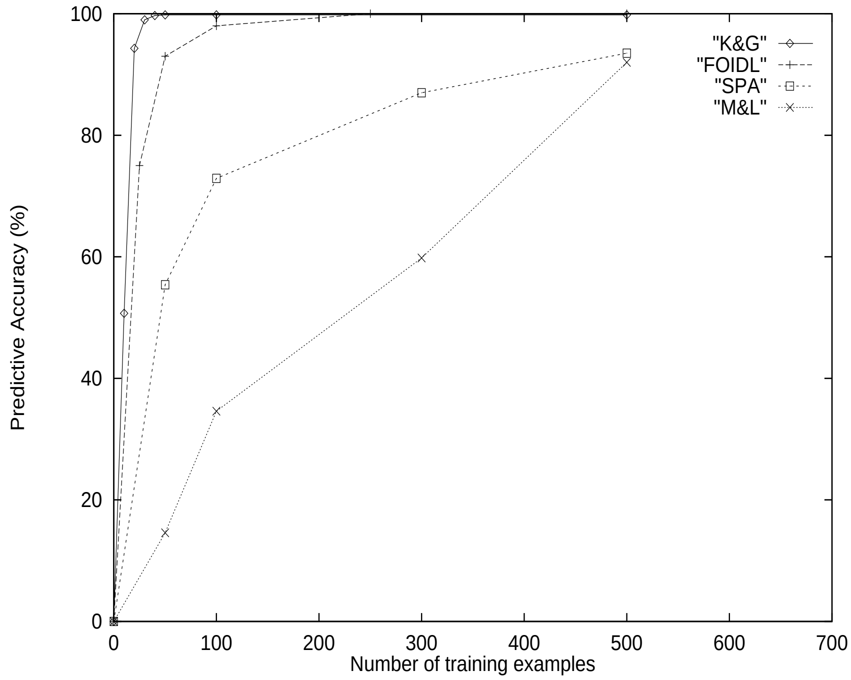

Figure 8: Comparison of learning curves for learning regular past tenses.

The graph shows that K&G gives the best accuracy result. FOIDL is the next best performer. K&G has the steepest learning curve. It reaches 90+% performance with 20 examples.

Figure 9 shows a closer look at the performance of K&G and FOIDL. The standard deviation of K&G's learning curve is also plotted. It shows that the accuracy results vary by less than 1% over different trials after 30 examples.

The learning curve of K&G saturates at the 99.8% asymptote. K&G gives wrong prediction on one test example "depict" [d.I.p.I.k.t] even though it has acquired the correct "add-Id" classifier. The reason is that the last three phonemes of "depict" match the phonemes of the word "picked" [p.I.k.t]. The rote-classifier corresponding to "picked" is excited and controls the phoneme register. It prevents the more general "add-Id" rule-classifier from firing. Since "picked" is already a past-tense inflection,

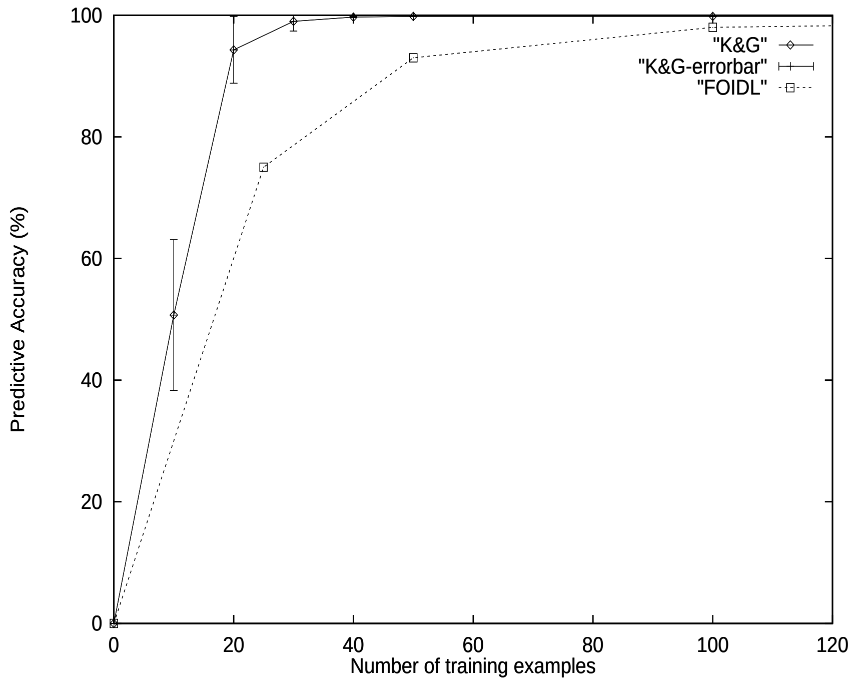

Figure 9: A closer look at the learning curves of K&G and FOIDL.

the program returns "depict" unchanged as its past tense.

While the performance model of K&G does not achieve 100% on the regular verbs, it makes correct predictions on irregular verbs which would otherwise be missed. For example, "become" is correctly pluralized because its last 3 phonemes match those of the word "come."

It is instructive to examine the internal representations of generalizations actually acquired by K&G. The following classifiers are typical:

```
1. [dc.dc.[+voice,+sonorant].d]
2. [dc.dc.[+voice,-coronal].d]
3. [dc.dc.[-low,-round,-tense,+continuant].d]
4. [dc.dc.[-voice,+strident].t]
5. [dc.dc.[-voice,-coronal,-continuant].t]
6. [dc.{d,t}.I.d]
```

Rule-classifiers 1, 2, and 3 together cover all the verb stems that end in a voiced phoneme other than [d]. These rule-classifiers overlap in the examples they cover. Rule-classifier 1 covers the majority of these cases (all the vowels, nasals, liquids, and glides). Rule-classifier 2 covers the voiced non-coronal stops ([b] and [g]) as well as some of the cases covered by rule-classifier 1, while rule-classifier 3 covers the voiced stridents.

Similarly, rule-classifiers 4 and 5 cover verb stems that end in an unvoiced phoneme other than [t]. Rule-classifier 4 covers stems ending in [k] or [p], while rule-classifier 5 covers the unvoiced stridents.

Experiment 2: Learning past tenses using all verbs

The second experiment compares the accuracy when both regular and irregular verbs are used in the training/test sets. Figure 10 shows that the performance of K&G and FOIDL are comparable up to 25 examples. Thereafter K&G is more accurate and reaches a higher asymptote: 95% versus 90%. SPA has a 76% testing accuracy and M&L 57% after training with 500 examples.

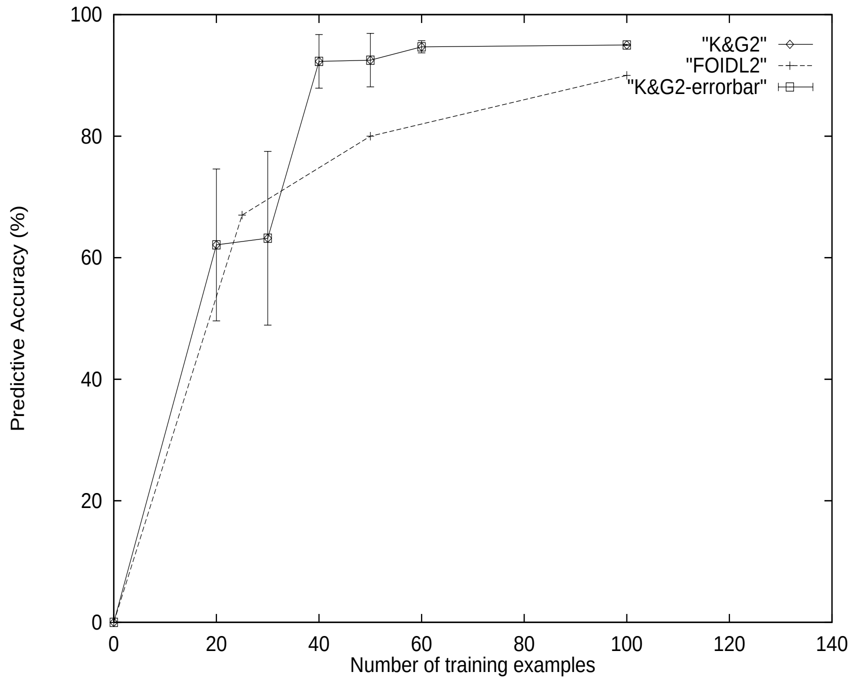

Figure 10: Comparison of learning curves for learning all past tenses.

The better performance of K&G can be attributed to its ability to form limited generalizations with irregular verbs (e.g., the "come/become" example mentioned in previous section). When we examine its internal representations, we find generalizations such as:

```
[dc.dc.ae.ng]        rang, sang
[dc.dc.a.t]          forgot, got, shot
[dc.E.n.t]           bent, lent, meant, spent
[dc.dc.{r,l}.u]      blew, drew, grew
```

```
[dc.dc.).t]          bought, brought, caught, taught
[dc.dc.o.z]          chose, froze, rose
```

Since irregular verb forms are in general idiosyncratic and not productive (such as go/went), we expect they fall into many sub-classes. The results confirm our expectation. The learner is able to find the more common patterns (such as blew/drew/grew and bought/caught/taught). The results suggest that most irregulars are just learned by rote and the learner makes limited generalizations about these forms.

Experiment 3: Learning regular past-tense in the presence of noise

This experiment tests the accuracy of K&G when the training set is noisy. The issue of noise is critical for a generalizer like K&G that induces strong generalizations from very few examples. One might think that small amount of noise would have a large (negative) effect on its performance.

To test K&G's noise tolerance, we randomly mutate the suffix of a few past-tense examples in the training set. For example, a [d] suffix must be changed to a [t] or [I.d]. The same test set of 500 regular verbs from experiment 1 is used.

Figure 11 shows that the performance of K&G degrades by 6% at the 4% noise level. The variance of the accuracy is large compared to that shown in Figure 9.[15]

Although the averaged learning curve is useful is showing noise-tolerance, it does not give much insight into how the program is able to isolate the exceptions. To understand why the program works, we have to look at a learning curve for an individual trial. See Figure 12.

Typically, the accuracy drops or stays flat as the program hits a noise example. As more examples come in, it becomes less costly to ignore the exception. The program finds generalizations that cover most of the positives and isolates the exception into a separate class. Since noise examples do not readily repeat (because the suffix is randomly mutated), it is unlikely that the exceptions cluster. This is how self-repairing occurs.

Experiment 4: Learning plurals

This experiment tests the performance of the learner on the pluralization task. There is no standard test set for pluralization. We run tests similar to the past-tense experiments on 200 randomly selected common nouns from the CELEX lexical databases.

[15] It would be nice to have comparison data on noise tolerance of other systems.

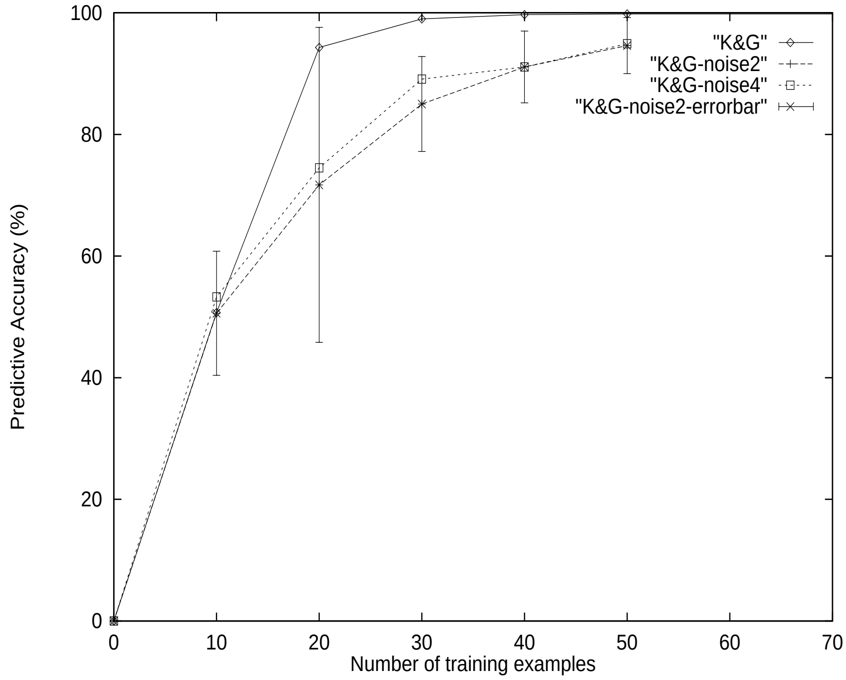

Figure 11: Performance on regular past-tense in the presence of 2% and 4% noise.

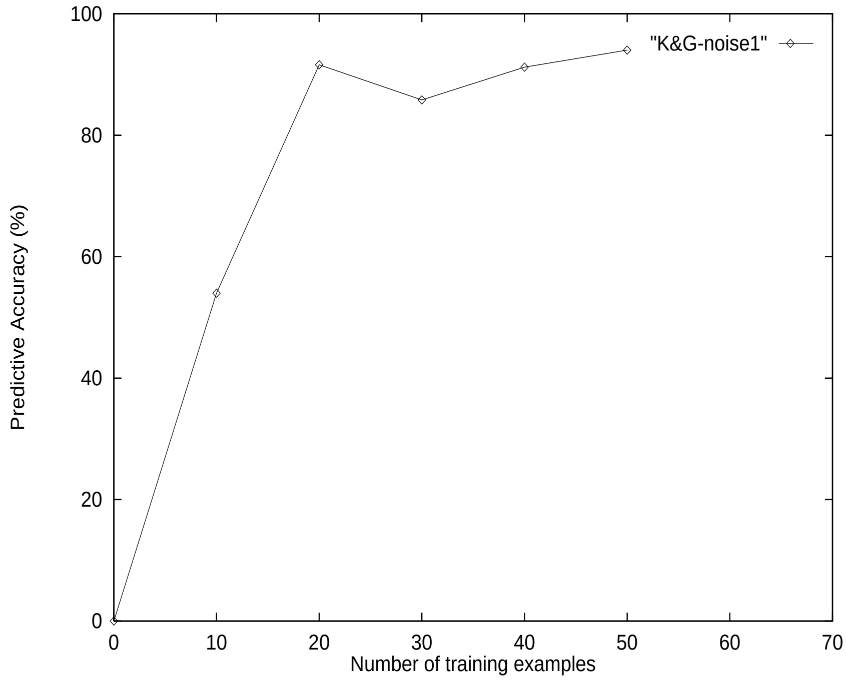

Figure 12: A learning curve for an individual trial. When the program hits a noise example, its accuracy drops. The accuracy starts to rise again as the program encounters more (good) examples.

We obtain a similar steep learning curve with accuracy hitting 99+% after 20 examples.

The formation of English plurals is unusually regular. There are very few irregular plural nouns. This property of English might lead one to propose learning mechanisms that exploit the statistics of regular plurals by training on a large number of examples so that any new test noun is sufficiently similar to a known one to produce the closest matched plural ending.

But there is evidence that the statistical property may not be essential to the acquisition of regular rules. For example, Marcus et. al. [9] and Clahsen [4] showed that the German -s plural behaves like a regular rule despite the fact that the rule applies to fewer than 30 common nouns. This observation raises the question of how a child can acquire regular rules from very few examples. The experiment will show that our learner can acquire generalizations that closely resemble those described in linguistics texts after seeing on the order of 10 examples.

It is more instructive to look at the generalizations actually acquired:

```
1. [dc.dc.[+voice,-strident].z]
2. [dc.dc.{y,e,I,v}.z]
3. [dc.dc.[-voice,-strident].s]
4. [dc.dc.[-voice,-coronal].s]
5. [dc.[+coronal,+strident].I.z]
```

Rule-classifiers like 1 and 3 are acquired after the presentation of as few as 4 or 5 examples. This surprises us considering that even a strong generalizer like FOIDL just memorizes what it sees for such few examples.[16]

Notice that we can almost read off the standard English pluralization rules from these classifiers. There are, however, two differences. First, the standard English pluralization rules are typically ordered (see, for example, [1]):

a. If the noun ends in a phoneme containing the features [+strident, +coronal] (i.e., one of the sounds [s], [z], [sh], [zh], [ch], [j]), the plural affix is [I z].
Otherwise,
b. If the noun ends in a [+voice] phoneme, the affix is [z].
c. If the noun ends in a [−voice] phoneme, the affix is [s].

In our system, the classifiers are activated in parallel, with the most excited ones gaining control over the data registers.

The second difference is that the unvoiced-plural rule c is represented by a disjunction of two rule-classifiers, 3 and 4, in our system. Rule-classifier 3 covers nouns

[16] Personal communication from Cynthia Ann Thomps.

ending in consonants [t], [k], [p], or [th]. Rule-classifier 4 covers nouns ending in the strident [f] or the non-coronal[17] stops [k] and [p]. Similarly, the voiced-plural rule b is split into rule-classifiers 1 and 2.

The learner also exhibits intermediate behaviors similar to those of young children [2]. After rule-classifier 1 and rule-classifier 3 are acquired, the performance program produces plurals like *foot[s] and *man[z]. Upon presentation of the nonce word "wug," it gives wug[z]. For nonce words ending in a strident like "tass" or "gutch," it gives the unaltered singular forms as plurals.

There is however a remaining mystery regarding the "add-[I.z]" rule. Berko in her study of children's learning of English morphology made the following observation. While the first-graders can apply the "add-[z]" and "add-[s]" pluralization rules productively to new words, they fail to apply the "add-[I.z]" rule to nonce words like "tass" or "gutch." When asked to produce the plural of a nonce word ending in [s] or [ch], they either repeat the word in its singular form or fail to respond. In no cases do they give wrong answers like tass[z], tass[s], or gutch[z], and only in few cases do they respond with gutch[s]. The children fail to use the [I.z] rule productively despite the fact that they can recognize and use real words like "glass" and "glasses" correctly.

Although the intermediate result of experiment 1 is consistent with Berko's interpretation of the developmental data, the result depends on the higher density of English plurals ending in non-stridents. Contrary to Berko's interpretation, our theory predicts that the learner would have no difficulty in acquiring the add-[I.z] rule before the add-[s] or add-[z] rules if it were given the plurals ending in stridents first.[18]

[17] The coronal feature refers to phonemes articulated by raising the tongue toward the alveolar ridge and the hard palate.

[18] There is some evidence to support this prediction. It is possible that Berko's observation on the add-[I.z] rule for plural formation is not entirely robust because the same Berko subjects perform quite well in adding [I.z] to form the third person singular verbs and possessives. Of course, we cannot at this stage rule out Berko's interpretation. It might be the case that the plural formation involves mechanisms more complicated than the addition of the [s] or [z] or [I.z] ending. For instance, Pinker and Prince [13] suggests that, instead of the three rules for adding plural endings, one might have a combination of different morphological and phonological rules that produce the same pronunciation. In their account, there is a morphological rule that adds [z] to a stem. The phonetic content of the "stem + [z]" is then modified by two competing phonological rules. The first rule, devoicing, changes the terminal [z] to [s] under certain contexts. The second rule, vowel insertion, inserts the vowel [I] between two word-final adjacent consonants that sound too similar. According to this account, the difficulty in producing plurals like "tasses" may be correlated with the child's additional effort in acquiring the vowel insertion rule.

Experiment 5: Learning plurals in the presence of noise

In this experiment, we examine the behavior of the learner when the input contains error. The learner is given the same training set from experiment 5 plus an additional incorrect plural form cat[z].

The incorrect form does not affect the acquisition of the correct phonological constraints. The learner is able to isolate the exception, cat[z], by a single-classifier:

```
6.  [dc.[-tense,-strident],t,z]
```
[19]

The same five rule-classifiers are acquired as in the previous experiment.

Experiment 6: Learning plural and past-tense rules together

In this experiment, we use 25 plural pairs and 25 past-tense pairs. The 50 examples are randomized. In addition to the rule-classifiers similar to those shown in previous experiments, K&G finds generalizations that we do not expect: two higher-order correlations relating the plural and past tense rule-classifiers. The two new higher-order rule-classifier enforce the constraint that the voicing bits of the ending phoneme of the stem and the affix must match:

```
[dc.dc.[-voice].[-voice]]
[dc.dc.[+voice].[+voice]]
```

These rule-classifiers can be interpreted as the *voicing assimilation rule* described in linguistics texts (such as [1]). Voicing assimilation captures cross-categorical generalizations governing the formation of not only plural nouns and past-tense verbs, but also third-person singular verbs, possessive nouns, and several other morphological categories.

Linguists explain complicated phonological processes in terms of the interactions of nearly independent and widely applicable rules. Our learning theory gives a plausible mechanism to produce this kind of compact, elegant phonological rules.

8 Evaluation

The K&G program achieves a human-level performance on the past-tense inflection and pluralization tasks. It shows not only nice noise tolerance, but also intermediate behavior consistent with what is observed about little children.

[19] The tense feature refers to phonemes produced with considerable muscular effort and a long duration.

Two factors contribute to the program's generalization power. First, the distinctive feature representation exploits a property of language: that very few of the possible phonemes are actually used in all human languages. Since the high dimensional distinctive feature space is sparsely populated, one can imagine relatively simple hyperplanes will cleanly separate classes of examples.

Second, the search in the generalization space has two important biases: small descriptions and local correlations. Preferring generalizations with more don't cares (or equivalently few number of bits) is a weak form of minimal length description. It works because the generalization space is sparse. Generalizing a rote-classifier sequentially from left to right embodies a locality assumption; it rules out arbitrarily long distance correlations. The locality bias dramatically reduces the number of candidate generalizations to be considered.

For learning to be effective, these two factors have to work together. A learner looking for small descriptions would not fare well in a dense space because there are no simple separating hyperplanes (as in the XOR problem). Similarly a learner looking for local correlations would not be able to learn regularities that depend on arbitrarily long distance relationship.[20]

We should point out that our analysis of why our program works depends on rather general concepts of high-dimensional space, sparseness, minimal description, and locality. We would not be surprised that mechanisms such as shift registers, bit vectors, and cube-covering biases will be generally effective in the extraction of local space-time correlations.

9 Philosophical Discussion

Over the past few years there has been a heated debate between advocates of "Connectionism" and advocates of more traditional "Symbolic Artificial Intelligence." We believe that contemplation of our mechanism for acquiring and using phonological knowledge can shed considerable light on this question.The essence here is in understanding the relationship between the signals in the neural circuits of the brain and the symbols that they are said to represent.

Consider first an ordinary computer. Are there symbols in the computer? No, there are transistors in the computer, and capacitors, and wires interconnecting them, etc. It is a connectionist system. There are voltages on the nodes and currents in the wires. We as programmers interpret the patterns of voltages as representations of our symbols and symbolic expressions. We impose patterns we call programs that cause

[20] It is entirely plausible that some form of hierarchical structures has to evolve in order to allow for certain kind of non-local regularities to become local in a more abstract representation.

the patterns of data voltages to evolve in a way that we interpret as the manipulation of symbolic expressions that we intend. Thus the symbols and symbolic expressions are a compact and useful way of describing the behavior of the connectionist system. We as engineers arrange for our connectionist system to exhibit behavior that we can usefully describe as the manipulation of our symbols.

In much the same way, auditory signals are analog trajectories through a low-dimensional space—a time-series of acoustic pressure. By signal processing these are transformed into trajectories in a high-dimensional space that linguists abstract, approximate, and describe in terms of phonemes and their distinctive features. This high-dimensional space is very sparsely populated by linguistic utterances. Because of the sparsity of this space, we can easily interpret configurations in this space as discrete symbolic expressions and interpret behaviors in this space as symbolic manipulations.

It may be the case that the linguistic representation is necessarily sparse because that is the key to making a simple, efficient, one-shot learning algorithm. Thus sparseness of the representation, and the attendant possibility of symbolic description, is just a consequence of the fact that human language is learnable and understandable by mechanisms that are evolvable and implementable in realistic biological systems. In fact, we believe this model of learning is applicable to problem areas outside phonology.

So in the case of phonology at least, the Connectionist/Symbolic distinction is a matter of level of detail. Everything is implemented in terms of neurons or transistors, depending on whether we are building neural circuits or hardware. However, because the representation of linguistic information is sparse, we can think of the data as bits and the mechanisms as shift registers and boolean constraints. If we were dealing with the details of muscle control we would probably have a much denser representation and then we would want to think in terms of approximations of multivariate functions. But when it is possible to abstract symbols we obtain a tremendous advantage. We get the power to express descriptions of mechanisms in a compact form that is convenient for communication to other scientists, or as part of an engineering design.

So what of signals and symbols? There are signals in the brain, and when possible, there are symbols in the mind.

10 Conclusion

We have demonstrated a performance module that can be implemented by simple physical hardware (or perhaps neural mechanisms?) with a small variety of parts, and a learning module that has been successful for learning a portion of English morphophonology. Our performance module uses constraint elements to fill in information in multiple ways. The same constraint element can enforce inflections or remove the inflections to recover the sound corresponding to the word stem. The learning module yields almost one-shot learning, similar to that observed in children: It takes only a few carelessly chosen examples to learn the important rules; there is no unreasonable repetition of the data; and there is no requirement to zealously correct erroneous behavior. The mechanism tolerates noise and exceptions. It learns higher-order constraints as it knows more. Furthermore, the intermediate states of learning produce errors that are just like the errors produced by children as they are learning phonology.

Fast learning occurs because the learning module exploits the sparseness of the generalization space, and has built-in biases to construct crucial negative examples and to seek local minimum description.

What have we learned about making effective language learners? We believe that human languages and the learning mechanisms have to co-evolve in such a way that they bootstrap each other. The features of language that tend to survive are those that are easily learnable and generalizable. For example, sparse local space-time correlations tend to be easily learnable by generalizers that have locality and small description biases. On the other hand, if the generalizer evolves a new form of representation (e.g., a hierarchical structure), new features of the languages become easily learnable and hence can survive. There is an intimate relation between the properties of the languages and the mechanisms that learn them.

Acknowledgments

We thank Morris Halle for teaching us elementary phonology and helping us to get started in this research. We thank Tom Knight for showing us that an electrical implementation of constraints is feasible. We thank Julie Sussman for numerous criticisms and suggestions to improve the presentation of the material. We thank Patrick Winston for his thoughtful critique and ideas for extending this work to elucidate questions on the general principles of language and learning. We thank Hal Abelson, Jon Allen, Rodney Brooks, Mike Maxwell, Steve Pinker, Elisha Sacks, Peter Szolovits, and Feng Zhao for comments on the manuscript. We thank Ben Kuipers, Cynthia Ann Thomps, and Raymond Mooney for their questions and suggestions.

References

[1] Adrian Akmajian, Richard Demers, and Robert Harnish. *Linguistics: An Introduction to Language and Communication.* MIT Press, 3rd edition, 1990.

[2] J. Berko. The child's learning of English morphology. *Word*, 14, 1958.

[3] Noam Chomsky and Morris Halle. *The Sound Pattern of English.* Harper and Row, 1968.

[4] Harold Clahsen, Monika Rothweiler, and Andreas Woest. Regular and irregular inflection of German noun plurals. *Cognition*, 45(3), 1992.

[5] Michael Kenstowicz. *Phonology in Generative Grammar*. Blackwell Publishers, 1994.

[6] Charles Ling and Marin Marinov. Answering the connectionist challenge: A symbolic model of learning the past tenses of English verbs. *Cognition*, 49(3), 1993.

[7] Brian MacWhinney. Connections and symbols: closing the gap. *Cognition*, 49, 1993.

[8] Brian MacWhinney and Jared Leinbach. Implementations are not conceptualizations: Revising the verb learning model. *Cognition*, 40, 1991.

[9] Gary Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, T. John Rosen, and Fei Xu. *Overregularization in Language Acquisition*, volume 57. Monographs of the Society for research in child development, 1992.

[10] Tom Mitchell. Generalization as search. *Artificial Intelligence Journal*, 18, 1982.

[11] Raymond Mooney and Mary Elaine Califf. Induction of first-order decision lists: results on learning the past tense of English verbs. *JAIR*, 3, 1995.

[12] Steven Pinker. Rules of language. *Science*, 253, 1991.

[13] Steven Pinker and Alan Prince. On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28, 1988.

[14] Sandeep Prasada and Steven Pinker. Generalization of regular and irregular morphological patterns. *Cognition*, 45(3), 1992.

[15] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[16] D Rumelhart and J.L. McClelland. On learning the past tenses of English verbs. In *Parallel Distributed Processing: Exploration in the microstructure of cognition.* MIT Press, 1986.

[17] Patrick Winston. Learning structural descriptions from examples. In *The Psychology of Computer Vision.* McGraw-Hill, 1975.