Sparse Representations for Fast, One-Shot Learning

Kenneth Yip and **Gerald Jay Sussman**
Artificial Intelligence Laboratory
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

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.

Introduction[1]

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 buffers, bidirectional constraints, symbolic differences, and greedy learning algorithms. As you will see, each of those particular concepts came to play an important role in our processing and learning system and in our ultimate conclusions.

We demonstrate our learning model in the domain of morphophonology—the connection between the structure of words and their pronunciation. We attack this problem partly because it is relevant to the foundation of cognitive science as evidenced in the controversy between the supporters of symbolic AI and connectionist AI.[2] Here a key to fast learning is that the phonemes that are actually used in language are only a few of the possible phonemes. In each language only a few of the possible combinations of phonemes may appear in words. We find that it is the sparseness of these spaces that makes the acquisition of regularities effective.

The phonemes are equivalence classes of speech sounds that are distinguished by speakers of a particular language. A phoneme is a representation of a range of continuous signals as a discrete symbol. Although one can "morph" one speech sound into another by a continuous process, the speaker will usually perceive it as a distinct phoneme: the phonemes are the psychoacoustic equivalent of digital values in a system implemented with electrical voltages and currents.

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 (Berko 1958), 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 (Marcus *et al.* 1992), Marcus et. al.

[1] Copyright ©1997, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

[2] See (Rumelhart & McClelland 1986; Pinker & Prince 1988; Pinker 1991; Prasada & Pinker 1992; Ling & Marinov 1993).

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. Thus children reliably exhibit behavior that indicates that they have made generalizations that linguists describe with rules.

If children do have knowledge of the generalizations, what is the form of such knowledge?

Our approach

We focus on the acquisition of inflectional morphology where developmental data are abundant. We present a theory of how to make and use phonological generalizations. Our theory explains how the generalizations can be learned from a few carelessly 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 (Rumelhart & McClelland 1986; MacWhinney & Leinbach 1991)). It is a theory of *learning* – not training. It is incremental, greedy, and fast. It has almost no parameters to adjust. It 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 (MacWhinney 1993), (2) that the same target constraints are learned independent of the presentation order of the corpus, (3) that learning is insensitive to token frequency,[3] and (4) that learning is more effective as more constraints are acquired. These claims are contrary to those made by the statistical learning theories.

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) (Chomsky & Halle 1968): Speech sounds (phonemes) are not atomic but are encoded as combinations of more primitive structures, called *distinctive features*. The distinctive features refer to gestures that the speech organs (such as tongue, lips, and vocal cords) execute during the speaking process.[4] 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 more than 100 phonemes.

The representation of a speech sound as a sequence of discrete phonemes is a crude approximation to what physically takes place during speech. We make two idealizations. First, the distinctive features are discretized to be 0 or 1. 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[5].

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 (such as hidden markov models) 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.

A Mechanistic Performance Model

Our performance model is envisioned as a hardware mechanism. The choice of mechanism limits the range of behavior that can be developed. Thus 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.

The constraint elements enforce boolean relations among the values of the features in the registers.[6] If

[3] See the psycholinguistic evidence presented in (Clahsen, Rothweiler, & Woest 1992).

[4] 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.

[5] Modern phonology postulates more elaborate representation devices such as multiple tiers and metrical grids. See (Kenstowicz 1994). These devices describe phonological phenomena that we do not address.

[6] We do not yet have a complete hardware model for

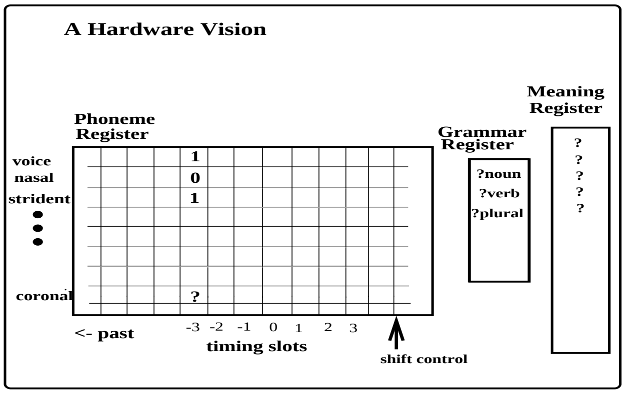

Figure 1: The hardware mechanism consists of three data registers.

there is sufficient match between the features in the registers and those enforced by a constraint element, the element fills in the details in the partially assigned values in the registers. If the information in the registers is inconsistent with the relation enforced by a 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. 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) 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 details of the meaning bits.

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*. The rote-classifiers capture specific correlations among the bit patterns in the data registers. 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.

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.

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.

The same mechanism of constraint elements and shift registers is effective for both production and comprehension of a word.

Learning Classifiers

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

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

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

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. The correlation between two rote-classifiers is determined by the difference in their bit patterns. For example, the pair of rote-classifiers "cat" and "cats" differs in the plural bit and the alignment of the phoneme bits. The 10 rote-classifiers are divided into two groups: the first one is related to changes in the plural bit, and the second to changes in the past-tense bit.

The learning algorithm then attempts to summarize the rote-classifiers in each correlation group. For example, it looks for a general 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. 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:

```
0101100100000010100100001100010000001110000001000001110101
←—      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 2.) 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).

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.

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.

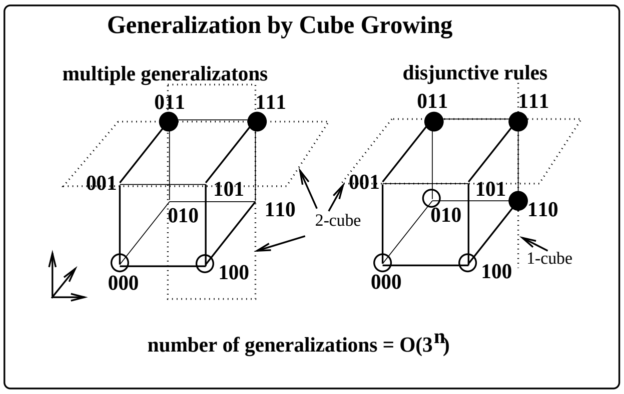

Figure 2: Classifier generalization as cube growing. A relation with n boolean variables defines an n-dimensional instance space with 3^n possible instances. The positive examples (solid dots) and negative examples (circles) occupy the vertices of an n-dimensional cube. Generalization can be thought 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 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.[8]

```
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 *[k.ae.t.z]). The incorrect ones are treated as near misses (i.e., negative examples that are slightly different from the positive ones). Basically 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.

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

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

```
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.[9] G3 correlates a terminal [z] phoneme with a preceding voiced non-continuant. 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(Akmajian, Demers, & Harnish 1990). These rule-classifiers are now available for constraint propagation, and are subject to further refinement when new examples appear.

Experimental Results

The corpus consists of 250 words.[10] The words are common nouns (about 50) and verbs (about 200) that first-graders might know. The nouns are the singular and plural forms of common animals and everyday objects. (e.g., cat, cats, dog, dogs, cup, cups, nose, noses, man, men) The corpus includes most of the regular and irregular verbs used in the psycholinguistic experiments of Marcus et. al. (Marcus *et al.* 1992) on English tenses.

Consistent with the observation that a human learner receives little explicit correction, the corpus contains only positive examples. However, the lack of external negative evidence does not rule out the possibility that the learner can generate internal negative examples when testing hypotheses. These internal negative examples, as we have seen, play a significant role in the rapid learning of classifiers.

The data record for each word in the corpus has 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 human to read.

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 bit vectors that typically have a straightforward symbolic interpretation.

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 plurals

The objective of this experiment is to determine what pluralization rules are acquired by our learner given a sample of common nouns and their plurals. 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. (Marcus *et al.* 1992) and Clahsen (Clahsen, Rothweiler, & Woest 1992) 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.

The input of this experiment consists of 22 noun-plural pairs. The particular number and choices of words are not very important as long as there are some examples of singular nouns ending in different phonemes. We pick a few examples for each type of plural formation:

[s]	[z]	[I.z]	semi-regular or irregular
cake(s) cat(s) chief(s) cup(s) fruit(s) month(s)	bottle(s) boy(s) dog(s) girl(s) gun(s)	box(es) bush(es) church(es) dish(es) glass(es) horse(s) nose(s)	house(s) leaf/leaves man/men foot/feet

The 22 pairs are fed to the learner sequentially in a random order once. We have experimented with several other random collections of plural pairs from the

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

[10] Initially we planned to use a corpus of several thousand most frequent words. But it soon became apparent that the learner can do extremely well even with a few dozen words.

corpus: 20, 30, 40, and 50 pairs. The learner rapidly settles down on the correct rule-classifiers after seeing a dozen or so regular plural pairs. Further examples may add exceptions but will not interfere with the correctly learned classifiers. The results presented here are typical. The final set of rule-classifiers acquired is not sensitive to either the order of presentation or the particular choice of examples.

After the presentation of all 22 pairs, the learner has acquired five rule-classifiers and four exceptions. The phoneme bits of the classifiers are as follows:

```
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]
```

Notice that we can almost read off the standard English pluralization rules from these classifiers.

The learner also exhibits intermediate behaviors similar to those of young children (Berko 1958). 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.

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.

Experiment 2: 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 22 noun-pairs from experiment 1 and an incorrect plural form cat[z].

The incorrect form is found not to affect the acquisition of the correct phonological constraints. The learner acquires the same 5 rule-classifiers as in experiment 1. An additional rule-classifier is created to account for the incorrect cat[z]:

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

Experiment 3: Learning regular past-tense

The input consists of 21 verbs and their past-tense forms. The stem-past pairs are presented sequentially in a random order once. After the presentation of all the verb pairs, the learner has acquired six rule-classifiers and three exceptions (the irregulars). The phoneme bits of the classifiers are as follows:

```
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]
```

The experiment shows that even though word stems ending in [d] or [t] do not form a natural class, the learner can still acquire correct past-tense rule-classifiers in the form of disjunctive rules.

Experiment 4: Learning plural and past-tense rules together

In this experiment, the 22 noun pairs used in experiment 1 and the 21 verb pairs used in experiment 3 are mixed together and presented to the learner in a random sequential order. In addition to the 11 rule-classifiers obtained in the previous experiments, the generalization algorithm produces 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 (Akmajian, Demers, & Harnish 1990)). 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.

Experiment 5: Learning irregular past-tense

The input consists of 55 common irregular verbs (such as eat, blow, buy, go) and their past forms. The learner acquires six rule-classifiers that cover 19 of the 55 input verbs.

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 also suggest that most irregulars are just learned by rote and the learner makes few generalizations about these forms.

Discussion/Conclusion

We have demonstrated that a performance model that can be implemented by simple physical hardware (or perhaps neural mechanisms?) with a few variety of parts and a learning algorithm has been successful for learning a portion of English morphophonology. Our mechanism 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.

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 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.

References

Akmajian, A.; Demers, R.; and Harnish, R. 1990. *Linguistics: An Introduction to Language and Communication*. MIT Press, 3rd edition.

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

Chomsky, N., and Halle, M. 1968. *The Sound Pattern of English*. Harper and Row.

Clahsen, H.; Rothweiler, M.; and Woest, A. 1992. Regular and irregular inflection of German noun plurals. *Cognition* 45(3).

Kenstowicz, M. 1994. *Phonology in Generative Grammar*. Blackwell Publishers.

Ling, C., and Marinov, M. 1993. Answering the connectionist challenge: A symbolic model of learning the past tenses of English verbs. *Cognition* 49(3).

MacWhinney, B., and Leinbach, J. 1991. Implementations are not conceptualizations: Revising the verb learning model. *Cognition* 40.

MacWhinney. 1993. Connections and symbols: closing the gap. *Cognition* 49.

Marcus, G.; Pinker, S.; Ullman, M.; Hollander, M.; Rosen, T. J.; and Xu, F. 1992. *Overregularization in Language Acquisition*, volume 57. Monographs of the Society for research in child development.

Pinker, S., and Prince, A. 1988. On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition* 28.

Pinker, S. 1991. Rules of language. *Science* 253.

Prasada, S., and Pinker, S. 1992. Generalization of regular and irregular morphological patterns. *Cognition* 45(3).

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