Comments on
A Computational Learning Model
for Metrical Phonology

Kenneth Church
Bell Laboratories
1989

1. Introduction

Before I begin my remarks, let me say that I am basically sympathetic with the spirit of Dresher's argument, but I feel duty bound after having agreed to write these comments, to take a more critical perspective. These remarks are intended to point out some of the difficulties that one may run into when trying to formulate a computational theory of learning.

Sometimes it is hard to draw a line between *learning* and *training*. The statistical literature uses the term *training* to refer to simple methods such as linear regression for fitting curves (models) to data. The term *learning* has been used in a number of ways. There are some uses of the term that overlap with statistical training, but there are also (possibly) quite different uses in formal computer science, Artificial Intelligence and Linguistic Theory. I'll mostly concentrate on trying to distinguish the statistical notions of training from the use of learning that is found in the Artificial Intelligence literature.

As a computer scientist with a background in Artificial Intelligence, I know that computers are much better at turn-the-crank tasks like parameter fitting than tasks requiring human-like behavior such as learning. And consequently, when I see a computer performing a task that looks like learning, I am very tempted to say that the computer is just fitting data, a task that doesn't require much intelligence (or whatever it is that distinguishes learning from training).[1] Basically, the difference between crank-turning and intelligence is that we know how to write crank-turning programs, but it can be more difficult to write ''intelligent'' programs. I might go so far as to distinguish the two kinds of tasks by whether or not they can be solved by a computer. Indeed, in AI, it has been said (partly in jest) that once we know how to solve a problem with a computer, we no longer think of the problem as requiring ''intelligence''.

By this criterion, Dresher's argument is self-defeating. In showing that a computer can solve the acquisition problem, Dresher is showing that language acquisition is not a learning problem, but rather a crank-turning problem that can be solved by simple statistical training methods. Of course, I am grossly overstating the case, but I think some readers might share at least some form of the concern. I might take a less extreme position and allow some computer programs to be considered intelligent, but only those that are sufficiently ''interesting.'' Thus, I would want to say that linear regression and Hidden Markov Modeling do not count as learning because they are too well understood, but maybe neural networks and Dresher's program are more mystical and less well understood, and so they would count as learning.

I will attempt to argue that Dresher's program is actually more statistical than mystical. That is, I will try to show that Dresher's program is equivalent in a certain sense to a trivial program that is simply fitting data not unlike linear regression. In other words, I will attempt to show that Dresher's program doesn't actually learn anything very interesting by showing that an utterly uninteresting program can ''learn'' the same facts. I am basically trying to reduce Dresher's program to a trivial one, and thereby show that Dresher's

1. I'm not sure what the property is that distinguishes learning from training. In these remarks, I use the term ''intelligence'' for this purpose, but there do seem to be some forms of learning, such as learning motor control commands, which do not seem to require ''intelligence''.

program is also trivial (in a certain sense). This kind of reduction is a standard form of argumentation in computer science. Of course, it puts me in the ironic position of trying to construct a completely uninteresting program that can ‘‘learn’’ phonology. Normally, as a computer scientist I try to write interesting programs, but right now, I find myself trying to write an utterly uninteresting one.

2. An Uninteresting ‘‘Learning’’ Program

Let me define the learning task (as Dresher does), to take a long sequence of words from language X as input, and to output a description of stress assignment for language X.

1. Input: a long sequence of words from language X
2. Output: a description of stress assignment for language X

Let me represent a *word* as a sequence of syllable weights (L or H) and stresses (0, 2). For example,[2]

- *record* (noun) → L2H0
- *record* (verb) → L0H2

The point is that once you have adopted this representation, learning is easy, since most of the hard problems have been encoded into the representation. This representation provides a lot of ‘‘inside information’’ (stipulations), so that it sidesteps a number of hard questions, including:

1. Where does the concept of syllable weight come from?
2. How do you learn the underlying representation from surface forms?

Of course, since Dresher’s program assumes the same representation, I’m sidesteping the same questions that he is when I adopt this representation.

Given this representation, it is possible to learn the stress assignment facts with a trivial one-line program. Suppose the input to the learner is simply a sequence of *words* from language X

```
L2H0
L0H2
 ...
```

and the output is a description of the set of all possible *words* in language X. Since the set of possible *words* (weight/stress strings) is finite (in practice) and very small ($approx$ 1000), it is possible (and practical) to describe the set by simply enumerating its members.

The following ‘‘trivial’’ Unix® program will ‘‘learn’’ the facts of the language very quickly:

```
sort -u
```

Surely, this program is uninteresting. Anything this program can do is probably not worth doing. If this

2. It is straightforward to extend the representation in obvious ways to incorporate other facts that should be included, such as extrametricality.

program can learn the same facts that Dresher's program can ''learn,'' then whatever Dresher's program is doing, it shouldn't be called learning.

Not only does my program ''learn'' the same facts that Dresher's program does, but, in addition, my program has a number of engineering advantages over his program:

1. There are fewer lines of code,
2. it is easier to replicate, and
3. it makes fewer assumptions (no trees, grids, etc) so that has a better chance to stand the test of time.

The main problem with my program is that it is too good; it can learn too much. According to Dresher (section 1.3), my program can learn 4.3 billion stress systems, and yet only a few hundred have ever been observed. Dresher argues that his program is superior because it can learn only 432 stress systems, which is much closer to what we find in the languages of the world.

From a computational point of view, we needn't be very concerned with this overgeneration. Of course, it indicates that we are missing some generalizations, but there is no pressing need to capture them (under the terms of the learning task contract as defined above). The ''trivial learner'' is a perfectly practical solution to the learning task as defined above. If it ain't broke, don't fix it.

3. An Uninteresting Fix

Of course, it is possible to fix the overgeneration without making my program interesting. Recall, that I am in the unusual position of trying to avoid writing an interesting program.

So far, we have been assuming that a language could be any enumeration of *words*. The overgeneration problem can be solved ''trivially'' by stipulating a representation of languages. Recall, that ''representation'' is a form of ''inside information'' or stipulation.

Let a *language* be an 11 bit number where

P1:	The word-tree is strong on the [Left/Right]
P2:	Feet are [Binary/Unbounded]
P3:	Feet are build from the [Left/Right]
P4:	Feet are strong on the [Left/Right]
P5:	Feet are quantity sensitive (QS) [Yes/No]
P6:	Feet are QS to the [Rime/Nucleus]
P7:	A strong branch of a foot must itself branch [No/Yes]
P8A:	There is an extrametrical syllable [No/Yes]
P8:	It is extrametrical on the [Left/Right]
P9:	A weak foot is defooted in clash [No/Yes]
P10:	Feet are noniterative [No/Yes]

With this representation (stipulation), there are 432 possible *languages*, according to Dresher's calculation.

Now, we can ''fix'' our trivial learning model by making sure that the resulting language is one of the possible 432 *languages*. One could accomplish this trivially by filtering the results at the end:

```
sort -u | convert_to_language
```

where *convert_to_language* returns an 11 bit number if the input is one of the possible *languages*, or

signals an error if it isn't.

Now, the trivial model no longer overgenerates. Thus, it is possible to get the job done in a completely uninteresting way. The ''learning'' program doesn't telling us anything about how cues interact. Therefore, learnability considerations need not tell us anything ''interesting''.

In general, it is hard to make strong computational arguments when the inputs and outputs are finite (and very small). If there were an infinite number of inputs and outputs (or at least a large number of them), then one could show that exhaustive uninteresting approaches such as the ones that I have been proposing would not work. But I think that it is pretty clear that stress systems are fairly small. One can talk about the possibility that a word can be arbitrarily long, but, as a practical matter, it is not a bad approximation to assume that the number of output stress sequences is small and finite, (especially if we limit our attention to morphemes rather than words).

4. The Trivial Model is not Crazy

For English, for example, I have found that a table with 1024 input weight sequences and only 51 possible output stress sequences is adequate for predicting stress patterns in a large coverage Text-to-Speech system (Church, 1985). This table assumes that there are no more than eight syllables in a morpheme. As far as I know, there are no exceptions in English. Another mechanism was used for stressing over morpheme boundaries since the rules are somewhat different in that case. I will not go into that here since Dresher doesn't discuss the effects of morpheme boundaries.

Weight	Extrametricality	
	0 (Verb)	1 (Noun)
H	1	1
L	1	1
HH	31	10
HL	10	10
LH	01	10
LL	10	10
HHH	103	310
HHL	310	310
HLH	103	100
HLL	310	100
LHH	103	010
LHL	010	010
LLH	103	100
LLL	010	100
etc.		

The table approach has some real computational advantages for a practical real-time system. Table lookup is obviously very fast. In addition, the storage requirements are quite modest (in fact, probably less than Dresher's program as a practical matter). The table also turned out to allow underspecified input. In the Text-to-Speech application, it is very difficult to determine underlying weight from surface spelling. It was convenient to allow variables in the input description of syllable weights.

In easy cases, it is possible to determine the weight directly for the orthography. For example, the weight of *torment* must be "HH" because both syllables are closed (even after stripping off the final consonant). Thus, the stress of *torment* is either "31" or "13" stress depending on whether is has 0 or 1 extrametrical final syllables:[3]

(stress-from-weights "HH" 0) → ("31") ; verb
(stress-from-weights "HH" 1) → ("13") ; noun

However, most cases are not this easy. Consider a word like *record* where the first syllable might be light if the first vowel is reduced or it might be heavy if the vowel is underlyingly long or if the first syllable includes the /k/. It seems like it is impossible to say anything in a case like this. The weight, it appears is either "LH" or "HH". Even with this ambiguity, there are only three distinct stress assignments: 01, 31, and 13.

(stress-from-weights "LH" 0) → ("01")
(stress-from-weights "HH" 0) → ("31")
(stress-from-weights "LH" 1) → ("13")
(stress-from-weights "HH" 1) → ("13")

In fact, it is possible now to use the stress to further constrain the weight. Note that if the first syllable of *record* is light it must also be unstressed and if it is heavy it also must be stressed. Thus, the third line above is inconsistent.

I implemented this additional constraint by assigning *record* a pseudo-weight of "=H", where the "=" sign is a variable which indicates that the weight assignment is constrained to be the same as the stress assignment (either heavy & stressed or not heavy & not stressed). I can now determine the possible stress assignments of the pseudo-weight "=H" by filling in the "=" constraint with all possible bindings (H or L) and testing the results to make sure the constraint is met.

(stress-from-weights "LH" 0) → ("01")
(stress-from-weights "HH" 0) → ("31")
(stress-from-weights "LH" 1) → ("13") ; No Good
(stress-from-weights "HH" 1) → ("13")

Of the four logical inputs, the = constraint excludes the third case which would assign the first syllable a stress but not a heavy weight. Thus, there are only three possible input/output relations meeting all of the constraints:[4]

Weight	Extrametrical Syllables	Stress
LH	0 (verb)	01
HH	0 (verb)	31
HH	1 (noun)	13

Thus, compiling the stress assignment rules into a table has three computational advantages. In addition to improved time and space performance, the table provides a set of constraints that can be used to reason backwards. In this case, we couldn't tell what the underlying syllable weight was from the orthography because we didn't know if the first syllable contained an schwa or a tense vowel. However, we found that we could make use of the output stress assignment in order to infer the input syllable weight. Thus, we

3. Actually, in practice, the weight determination is complicated by the possibility that *-ment* and *-ent* might be suffixes. Note, for example, that the adjective *dórmant* does not stress like the verb *torménт* because the adjectival suffix *-ant* is extrametrical.

4. The noun should probably have the stress 10 rather than the stress 13. I assume that an extrametrical syllable has 3 stress if it is heavy, and 0 stress if it is light. The stress of the extrametrical syllable is very difficult to predict, as discussed in (Ross, 1972).

showed that the table could be used in the backward direction to infer the input.

5. Conclusion

Dresher ultimately rejects the table-based model for two reasons:

1. It overgenerates the space of possible languages by a factor of 4.3 billion / 432, and
2. it places an upper limit on the size of a word.

In an earlier version of the paper, he also objected to the table-based model as ‘‘uninteresting,’’ but this third objection seems to have been dropped in the revision, presumably because my program is intended to be uninteresting. After all, as mentioned in the introduction, it is my aim to write an uninteresting program that gets the job done. If I succeed, then I have demonstrated that the job is too easy and the intelligence is not in the learning task but somewhere else.[5]

I am not too concerned with either of Dresher’s remaining criticisms. It was suggested above that morphemes are limited in length, at least in English. Even in those languages with very long morphemes, I would suspect that one could modify the representation in some way (using run-length coding or something similar) so that the table would be small and finite. Obviously, I am using the representation as ‘‘inside information,’’ but Dresher also depends on coding a fair amount of smarts (stipulation) into the representation.

As for overgeneration, it isn’t clear to me why the language learner should be concerned with the number of possible languages. The learner does not need to know *why* stress works the way that it does. All that he needs is an adequate description of one language so that he can process well-formed strings in that language. The learner might be more of a tradesman than a scholar. The learner needs to know how to turn-the-crank, but he does not need to know the best way to do so, or why turning-the-crank is the right thing to do. An adequate description is sufficient. Explanation is not required. Nor is optimality.

Of course, if you take this view of learnability, then it is unlikely that learnability arguments will tell us very much about language. It is logically possible that learnability will tell us nothing. For example, it is possible that there are only a small and finite set of language, say 432. Perhaps all 432 languages are innately stored in the child’s head so that when he hears his first word he could compute some simple function that told him which of the 432 languages to use. The code need not have anything to do with the structure of language. It could, for example, depend on some biological or astronomical clock at the time when the child hears his first word.[6] While we are all sympathetic with the learnability approach, I have yet to see a convincing argument that demonstrates that the structure of language necessarily follow from learnability considerations.

---

5. I might suggest that too much has been encoded into the input representation. It was my experience when I was working on letter-to-sound rules for speech synthesis that stress assignment was the easy part. It was much harder to decide which syllables were heavy and which were light. Obviously, orthography obscures the issues. But, in addition, there are difficulties from morphology and elsewhere.

6. This argument was inspired by a casual comment by Chomsky (personal communication). I remember him arguing that learnability considerations need not tell us about language. Suppose, for example, that there were only two possible languages (as opposed to the 432 that we have in this case). It is logically possible, he argued, that the child could select the right one on the basis of something that had nothing to do with the internal structure of either one. And thus, the child could, in principle, learn a language without telling us anything about the structure of language.

6. References

Church, K., "Stress Assignment in Letter to Sound Rules for Speech Synthesis," Association for Computation Linguistics, June 1985.

Ross, J., *A Reanalysis of English Word Stress (Part I)*, in Brame, M., (ed), *Contributions to Generative Phonology*, University of Texas Press, Austin, Texas, 1972.