Which Learning Algorithms Can Generalize Identity Effects to Novel Inputs?

(1 March 2016)

A recurring theme in phonology is that of *identity effects*: in which the grammaticality of a form hinges on whether two of its segments are identical or not. Identity effects have proven to be difficult or impossible for certain types of learning algorithms to acquire from realistic datasets. This contrasts with human behaviour, in which subjects easily generalize identity effects from training data to segments not present in the data. In this work we provide a framework for rigorously establishing which phonological learning algorithms will fail at generalizing identity effects to novel stimuli. We develop a theory of *symmetry* for linguistic forms, training sets of data, and learning algorithms, and show that if an algorithm has a particular symmetry that is shared by the data it is trained on, it cannot generalize to novel inputs in a human-like way. We demonstrate these results computationally using a multilayer feedforward neural network.

1. Introduction

Suppose a subject is asked to learn an artificial language in which all words have the form $C_1V_1C_2V_2$, where C_1, C_2 are consonants and V_1, V_2 are the vowel /a/. Subjects are told that /kaka/, /baba/, /haha/, /mama/, and /rara/ are all examples of valid words in the language but that /kaha/, /rama/, /kata/, /maba/, /hara/ are not valid words. Now suppose that the learner is asked whether /nana/ and /naza/ could be valid words in the language. Presumably they will say that /nana/ could be a valid word in the language whereas /naza/ could not be. The obvious feature that all the valid words have in common is that their two consonants are identical. This feature is not shared by the invalid words. Following the existing terminology (e.g. Benua (1995); Walker (2000)), we say in this case that the human learners are able to learn an identity effect. (It is likely that the human learners would respond the same way with just the positive examples of the grammatical words and not the negative examples.)

There have been many tests of the psychological reality of identity effects in the phonological domain (e.g. Berent et al. (2002); Gallagher (2013)). In artificial language learning tasks, human subjects are sensitive to identity relations between segments, and are able to generalize them to novel inputs.

Surprisingly, given how obvious the above pattern is to human learners, many computer models of learning are not able to learn identity-based rules like those implicit in the data above, without being presented with nearly all possible inputs. These computational learners may give the same rating to both the forms /nana/ and /naza/, since neither of them have any similarity to the training words in a manner that is deemed relevant by the algorithms. Important classes of such algorithms include basic connectionist algorithms Rumelhart & McClelland (1988) and the "Plain" (Baseline version) of the UCLA Phonotactic Learner (Hayes & Wilson (2008)). There are ways to modify these algorithms to perform better on such tasks, for example, by introducing copying (Colavin et al. (2010)), special representations of identical segments in the input (Gallagher (2013)), or weight sharing across connections as is done in convolutional neural networks (LeCun & Bengio (1995)).

There are compelling informal arguments given for why the basic versions of these algorithms cannot learn identity-based rules, as given by Marcus (2003) and Berent (2012). We survey these

arguments in Section 2. Our goal is to provide alternative arguments to those of these authors with similar consequences for cognitive modelling, but expressed in a rigorous mathematical framework. We provide clearly defined conditions under which algorithms will not be able to generalize identity effects to novel stimuli. These conditions apply not just to the now-standard neural network models of Rumelhart & McClelland (1988), but also to many of the new architectures proposed under the Deep Learning framework (LeCun et al. (2015)).

In Section 3 we formally define *learning algorithms*, *symmetries* of sets of words, and what it means for an algorithm to be *invariant* under a symmetry. Briefly, an algorithm is any procedure that takes a set of training data and a novel input word and then outputs a score indicating how well-formed the word is. A symmetry is an operation that takes a word (well-formed or not) as input and gives another word (well-formed or not) as output. An example would be an operation that reverses the order of all the segments in a word. An algorithm is invariant under a symmetry if the score it gives to a word based on some training data is identical to what it would give if the symmetry were first applied to all the words in the training set and to the input word. In our main result we show that if an algorithm is invariant under some symmetry, and the training data is invariant under the same symmetry, then the algorithm cannot learn a grammar that fails to be invariant under that symmetry.

As our major application, in Section 4 we demonstrate a symmetry that identity-based rules are not invariant under, and then show that a wide class of algorithms are invariant under it. This means that such algorithms cannot learn identity-based grammars with invariant training data, in contrast to human performance on analogous tasks. In Section 5 we show that the analyses in Section 3 and 4 extend to the case of algorithms that use randomness in their training or in their generation of scores. In Section 6 we then numerically demonstrate how feed-forward neural networks suffer from these limitations, independent of the number of hidden layers in the network. We conclude in Section 7 we extend our analysis to phonological alternations, demonstrating how our technique applies to the experimental data of Gallagher (2013).

2. Limitations of Connectionist Models of Learning

Connectionism is a style of modelling in cognitive science and linguistics in which models consist primarily of nodes and connections. Every node has a variable associated with it called an activation. Nodes are connected to each other via connections, which have weights. The activation of a node is determined by the activation of all the other nodes that are connected to it, weighted by the connection weights. If an artificial neural network, as it is called, is used to compute an input-output map, a subset of the nodes are called input nodes, and are clamped to values corresponding to the input. Another set of nodes are called the output nodes, which encode the output, and finally there are the so-called hidden nodes which play an intermediary role. Such an architecture is also known as a multilayer perceptron.

One of the most important works analyzing the uses of connectionist model in cognitive science is Marcus's *The Algebraic Mind* (Marcus (2003)). The book contains a wide-ranging survey of connectionist modelling and its tension with the mind's ability to perform symbolic manipulations. The book takes a nuanced view of the issues but can read as taking a largely negative view of the use of connectionist neural networks as the central tool for cognitive modelling. The problem is mainly the incompatibility of artificial neural networks with symbolic processing. He outlines five things that are required of the mind, and must be captured by any complete model of cognition: the system must be able to (i) distinguish between variables and instances of the variables, (ii) represent relationships between variables (iii) bind variables to their instances (iv) peform operations on variables (v) extract relationships between variables from instances. Marcus gives reasons why popular connectionist architectures and their associated training methods are incompatible with these desiderata.

More specifically, Marcus introduces the concepts of *input space*, the set of all possible inputs, the *training set*, the data provided to the learner, and *training space*, portions of the input space corresponding to inputs whose features appear inside the training set (Marcus (1998)). He claims "Many-nodes-per-variables multilayer perceptrons that are trained by back-propagation can generalize one-to-one mappings within the training space, but assuming that the inputs are binary ... they cannot generalize one-to-one mappings outside the training space." ((Marcus 2003, p. 46)). Besides the results of various computer experiments, he offers several arguments for why this is the case. These arguments apply equally well to other cognitive models besides the specific ones he cites. An important concept for his discussion is that of "universally quantified one-to-one mappings" or UQOTOMs. Such a mapping is universally quantified, meaning that it must apply to all inputs, and one-to-one, meaning that each output is only mapped to by one input. Linguistic examples of UQOTOMs include taking the regular past-tense form of a verb or reduplication of a syllable. A problem for a wide class of connectionist models is that the models do not naturally encode UQOTOMs: rather, particular special choices of weights are required to do so, and these weights are not naturally generated by standard learning algorithms (such as backpropagation). Such algorithms can only learn UQOTOMs if trained on a larger space of training data than would be required by a human learner. Furthermore, such models and training regimes have the property of *training independence*, meaning that if an input or output node's activation does not vary during training, the relations evident from the activation of other nodes cannot be generalized to it.

These conclusions are amplified and applied in particular to phonology in *The Phonological Mind* (Berent (2012)). An important example is that of stems in Hebrew, where the last two consonants in a stem may be identical (e.g. /simem/) but the first two consonants may not (e.g. /sisem/). Berent surveys a series of experimental studies demonstrating that this restriction is a psychologically real part of Hebrew grammar, and furthermore it is extended by speakers to segments that do not even occur in Hebrew (e.g. Berent et al. (2002)). Berent argues that such behaviour requires the ability of the learner to perform across-the-board generalizations, and are possible only with "computational devices that operate on variables" ((Berent 2012, p. 84)), rejecting non-algebraic accounts of phonology including connectionism.

We agree for the most part with the conclusions of Marcus and Berent: most connectionist architectures as currently used are ill-suited to modelling the symbolic manipulation and algebraic relations necessary for phonological behaviour. Our goal here is to provide rigorous proof of the inability of some models and training regimes to perform these tasks. In this paper, we focus on a single simple task: recognizing identity between segments. Our hope is that by isolating clearly defined conditions under which algorithms cannot generalize identity relations outside the training set, we will streamline the analysis and creation of algorithms that can overcome these limitations.

3. Formal Definitions

We consider a set of words W which contains all possible inputs to an algorithm. Importantly, W consists of all words whether they are grammatical or not. In what follows we will consider words to be strings of segments, but individual words can be anything, such as strings of feature bundles or gestural scores. To fix ideas, in what follows we will often consider a particular example of a set of words: we let $\widetilde{W}$ be the set of all two-segment words, where we take the segments from some finite segment inventory.

We define the training data D to be a collection of word-rating pairs $\langle w, r\rangle$ where w is a word in W and r is a number interpreted as a *rating* of how "good" w is. For example, using the word

set $\widetilde{W}$, a dataset D might consist of

$$\langle /\text{cc}/, 1\rangle,\ \langle /\text{aa}/, 1\rangle,\ \langle /\text{ee}/, 1\rangle,\ \langle /\text{ga}/, 0\rangle,\ \langle /\text{eh}/, 0\rangle,\ \langle /\text{ra}/, 0\rangle. \tag{3.1}$$

This dataset says that /cc/, /aa/, and /ee/ have rating 1 (and thus are grammatical words) and that /ga/, /eh/, and /ra/ have rating 0 (and thus are ungrammatical words). Alternatively, in a training task where only good words are given to the learner, D might consist only of good words paired with the rating 1. But there are other possibilities: words could be paired with a rating given by their prevalence in a corpus, for example.

To formally define a learning algorithm, consider what a learning algorithm such as the UCLA Phonotactic Learner (Hayes & Wilson (2008)) does. First, a collection of data D is input to the algorithm and used to choose a set of parameters p in a model. We can formalize this as $p = \mathscr{A}(D)$. Once we have p, given any new input w the algorithm outputs a *score*, which we can formalize as $f(p,w)$. Typically, the computation of p from D is computationally intensive whereas once we have p, the score $f(p,w)$ is cheap to evaluate. This matches our experience of human behaviour where learning a language occurs over long periods of time, whereas judgements of the well-formedness of novel words are readily produced by adult speakers.

Here we will abstract away issues of parameter setting and computational effort and just view an algorithm as a map that takes a set of training data D and an input w and outputs a rating. We consider the map $\mathscr{L}$ given by

$$\mathscr{L}(D,w) = f(\mathscr{A}(D),w).$$

The interpretation is that the score $\mathscr{L}(D,w)$ is what you would get if you used the data D to train the algorithm and then used the resulting computational model to evaluate the word w.

We note that we make a distinction between *ratings* which are part of the training data D, and *scores* which are what the algorithm outputs for words w given data D. We make this distinction because the two quantities might be quite different things. For example, for the UCLA Phonotactic learner, input words are coupled with ratings that are type frequencies (whole numbers) whereas the output scores for input words are positive numbers with a larger number being interpreted as a less well-formed word.

In general, a symmetry is an operation that leaves an object unchanged when the operation is applied to it. Accordingly, we define an operation σ on W to be a symmetry if when it is applied to all the elements in W we get all the elements in W back. More formally, a symmetry σ is a bijective map from the set of words W to itself; in other words, a map such that $\sigma(w)$ is in W for all w in W, and for all v in W there is an u in W such that $\sigma(u) = v$. As an example of a symmetry, let $\tilde{\sigma}$ be the map from $\widetilde{W}$ to itself given by

$$\tilde{\sigma}(/\mathbf{xy}/) = /\mathbf{yx}/, \tag{3.2}$$

for any segments $/\mathbf{x}/$, $/\mathbf{y}/$. Thus the symmetry $\tilde{\sigma}$ reverses the order of segments in two-segment words.

We introduce symmetries in order to analyze algorithms; we are not claiming that they have any psychological or linguistic reality. Indeed, as far as we know, all maps that are naturally occurring phonological processes are not symmetries. For one thing, most phonological maps satisfy $\sigma(\sigma(w)) = \sigma(w)$ for all w (also known as idempotency (Magri (2015))). But this can only happen with a symmetry if $\sigma(w) = w$ for all w, meaning that σ does nothing.

Likewise, once you have a symmetry, you can study the objects that are invariant under the symmetry, meaning that they are unchanged when the symmetry acts upon them. What it means for a symmetry to act upon an object depends on the object in question. In what follows we will define invariance for words w, training sets D, and learning algorithms $\mathscr{L}$ in turn.

First, we say a word w is invariant under a symmetry σ if $\sigma(w) = w$. Continuing with our running example, the word /bb/ is invariant under the symmetry $\tilde{\sigma}$ whereas the word /ab/ is not.

TABLE 1. Segments in a Hypothetical Language

segments	sonority
a o	5
w j	4
m n	3
v z	2
b d	1

To apply a symmetry to a set of training data D, we say that σ just acts on each word in every word-rating pair in the data set, but does not change the rating of that word. So if the word-rating pair $\langle w,r\rangle$ is in D, then the pair $\langle \sigma(w),r\rangle$ is in $\sigma(D)$. For example, if we applied $\tilde{\sigma}$ (as defined in (3.2)) to the dataset in (3.1) we would get the dataset

$$\langle /\text{cc}/,1\rangle,\ \langle /\text{aa}/,1\rangle,\ \langle /\text{ee}/,1\rangle,\ \langle /\text{ag}/,0\rangle,\ \langle /\text{he}/,0\rangle,\ \langle /\text{ra}/,0\rangle.$$

We say that a dataset D is invariant under a symmetry σ if $\sigma(D)$ has precisely the same word-rating pairs as D. The simplest way for data D to be invariant under a symmetry σ is if each word in each word-rating pair in D is invariant under σ. But there are other ways. For example, the symmetry $\tilde{\sigma}$ leaves the data

$$\langle /\text{bb}/,1\rangle,\quad \langle /\text{gg}/,2\rangle,\quad \langle /\text{ee}/,0\rangle$$

invariant because the words /bb/, /gg/, /ee/ are all invariant under $\tilde{\sigma}$. On the other hand the data

$$\langle /\text{bg}/,1\rangle,\quad \langle /\text{gb}/,1\rangle,\quad \langle /\text{ea}/,2\rangle,\quad \langle /\text{ae}/,2\rangle$$

is also invariant under $\tilde{\sigma}$, but in this case the individual words are not invariant, it is just that w and $\sigma(w)$ always have the same rating in this data set.

Finally, we say an algorithm $\mathscr{L}$ is invariant under σ if $\mathscr{L}(\sigma(D),\sigma(w)) = \mathscr{L}(D,w)$ for all D and w. In words, the rating that the algorithm gives to w when trained on D is the same that the algorithm gives to $\sigma(w)$ when trained on $\sigma(D)$.

Our main result is a simple consequence of these definitions.

THEOREM 1. *If algorithm $\mathscr{L}$ and training data D are invariant under symmetry σ then*

$$\mathscr{L}(D,w) = \mathscr{L}(D,\sigma(w)),$$

for all w in W. In other words, the algorithm $\mathscr{L}$ gives the same rating to w and $\sigma(w)$ when trained on D.

Proof. For all w in W we have

$$\mathscr{L}(D,w) = \mathscr{L}(\sigma(D),\sigma(w)) = \mathscr{L}(D,\sigma(w))$$

where the first equality follows from the invariance of $\mathscr{L}$ under σ, and the second inequality follows from the invariance of D under σ. □

Example: Consider a language with 10 segments, each segment having a sonority value between 1 and 5 according to Table 1. Words in the language consist of only two segments. Suppose that all words in the language have increasing or constant sonority. So, /ba/, /mo/, /zw/, /bd/ could all be words in the language, but /ad/, /an/, and /wv/ could not be. Consider the segment

reversing symmetry $\tilde{\sigma}$ given in (3.2). If you apply $\tilde{\sigma}$ to an ungrammatical word (e.g. /ab/) you get a grammatical word (/ba/). If you apply $\tilde{\sigma}$ to a grammatical word with increasing sonority you get an ungrammatical word. Words with two segments of the same sonority give you back another word with segments of the same sonority.

Now suppose you have a learning algorithm $\mathscr{L}$ that is invariant under $\tilde{\sigma}$. This means that if you take a data set D, train the algorithm on it, and then use the algorithm to evaluate word w, you will get the same result if you train the algorithm on $\tilde{\sigma}(D)$ (in which all the words are reversed) and then use the algorithm to evaluate $\tilde{\sigma}(w)$, which is just the reversal of w.

Suppose we give the algorithm data D that is invariant under $\tilde{\sigma}$. For simplicity we assume that D consists only of grammatical words each assigned the rating 1. In this case, the only way D can be invariant under $\tilde{\sigma}$ is if all the words in D have constant sonority, and for every such word **xy** in D, **yx** is also in D. Can the algorithm correctly learn the generalization that words in the language must have increasing or level sonority from this data set?

Theorem 1 shows that it cannot, as follows. According to the theorem, $\mathscr{L}(D,w) = \mathscr{L}(D,\tilde{\sigma}(w))$. All we need to do is let w equal a word of increasing sonority, such as /ba/, to see that the algorithm with training data D gives the same score to /ba/ and /ab/. Since the first is grammatical and the second is ungrammatical, the algorithm clearly has not learned the correct rule governing grammaticality in the language. This is pretty commonsensical: one way to think of it is that there is nothing in the algorithm or the training data to make the algorithm prefer /ab/ to /ba/, since both the algorithm and the training data are invariant under $\tilde{\sigma}$, and $/\mathrm{ba}/ = \tilde{\sigma}(/\mathrm{ab}/)$. Of course, this is not necessarily a defect of the algorithm $\mathscr{L}$; if some words with increasing or decreasing sonority were included in D, then D would not be invariant under $\tilde{\sigma}$, and $\mathscr{L}$ could learn the grammar.

In the next section we will give a less straightforward example, allowing us to formalize the idea of identity effects for learning algorithms.

4. Identity Effects

We now use the above result to show that certain algorithms cannot learn identity effects unless trained on words containing nearly all segments in the inventory. That is, such algorithms cannot extend the identity-based rules to words containing segments that it has not explicitly been trained on, in contrast to human learners (Berent et al. (2002)).

We return to the example at the beginning of the paper: $\widetilde{W}$ is the set of all words consisting of two segments. For typographic ease, we will suppose the segment inventory consists of the 26 lower case letter of the English alphabet. We stipulate that grammatical words are those consisting of two identical segments and all other words are ungrammatical. Suppose we want the algorithm to learn this grammar, but train it on data omitting any words containing either of the segments /y/ and /z/. What algorithms will not be able to learn the correct grammar under these conditions?

Define the symmetry σ of W by the following:

$$\sigma(/\mathbf{x}_1\mathrm{y}/) = /\mathbf{x}_1\mathrm{z}/, \quad \sigma(/\mathbf{x}_1 z/) = /\mathbf{x}_1\mathrm{y}/, \quad \sigma(/\mathbf{x}_1\mathbf{x}_2/) = /\mathbf{x}_1\mathbf{x}_2/,$$

for all segments $/\mathbf{x}_1/, /\mathbf{x}_2/$, with $/\mathbf{x}_2/$ not equal to /y/ or /z/. In other words, if the second segment is /y/, σ changes it to /z/, if the second segment is /z/, σ changes it to /y/, and if the second segment is neither, then the word is unchanged.

Now suppose our training data D contains no words with either /y/ or /z/ as the second segment. Then D is invariant under σ. Theorem 1 shows that if the algorithm $\mathscr{L}$ is also invariant under σ then it will give the same rating to w and $\sigma(w)$ for any word W when trained on D. In that case we would have that it gives the same rating to the words /yy/ and /yz/, showing that it cannot

learn the grammar with an identity effect. In general, our result applies to any situation where two or more segments are not included in the training set.

Below we provide an example of an algorithm invariant under this symmetry. But in general we informally argue that any algorithm that does not in some way explicitly check for identity between the segments, or somehow enforce a similar treatment of those two segments in processing, cannot correctly learn that /yy/ is a more well formed word than /yz/, if it is never given words with a second letter /y/ or /z/ as training data.

5. Randomized Algorithms

Many algorithms for learning use randomness at some point in their operation. It may either be in the computation that takes the input data to the parameters p (for example, by which order the input words are used in training) or in the map from the parameters and a new input word to a word score s. In the former case $p = \mathscr{A}(D)$ is a random function of D; in the latter $s = f(p,w)$ is a random function of p and w. In either case, this leads to $\mathscr{L}(D,w)$ being random for any fixed D and w.

Under these conditions, it is unlikely that invariance of the form described above will hold. Instead we now define invariance of $\mathscr{L}$ under σ to be

$$\mathbb{E}\mathscr{L}(\sigma(D),\sigma(w)) = \mathbb{E}\mathscr{L}(D,w),$$

where $\mathbb{E}$ denotes expectation. (If X is a random variable, $\mathbb{E}X$ is approximately what we would get if we took the average of a large number of samples of X.)

We now get the analogous result to the non-deterministic case.

THEOREM 2. *If random algorithm $\mathscr{L}$ and training data D are invariant under symmetry σ then*

$$\mathbb{E}\mathscr{L}(D,w) = \mathbb{E}\mathscr{L}(D,\sigma(w)),$$

for all w in W. In other words, the algorithm $\mathscr{L}$ gives on average the same rating to w and $\sigma(w)$ when trained on D.

Proof. We have

$$\mathbb{E}\mathscr{L}(D,w) = \mathbb{E}\mathscr{L}(\sigma(D),\sigma(w)) = \mathbb{E}\mathscr{L}(D,\sigma(w))$$

where the first equality follows from the invariance of $\mathscr{L}$ under σ, and the second inequality follows from the invariance of D under σ. □

6. Computational Experiments

We demonstrate the consequences of our theorems in a computational experiment where we use a deep neural network to learn the grammar described in our introduction. The networks are trained using data in which two-segment words with two identical segments are grammatical, and two-segment words with two different segments are ungrammatical. The network is then asked to assess novel words containing segments it has not seen in the training set. Randomness enters into the training of these networks in various places and so Theorem 2 is the relevant result in this case. Consequently, we do not compare individual trainings of the network on the novel stimuli. Rather, for each novel stimulus we train the network numerous times and take the average score over all the trainings. It is these scores that are compared between stimuli.

6.1. *Task and Dataset*

Before discussing the neural network learners that were tested, we describe the dataset and task that was required of them. As before, our set of words W consisted of all two-segment words with

segments running from /a/ to /z/. The training set consisted of the 24 words /aa/, /bb/, …, /xx/ paired with rating 1, along with 48 randomly generated words with mismatched segments, two for each segment, each paired with rating 0. In neither set of training words (grammatical and ungrammatical) do the segments /y/ and /z/ appear.

To assess the ability of the learner to generalize to novel inputs, we tested it on the words

$$/\mathrm{yy}/,\ /\mathrm{zz}/,\ /\mathbf{x}\mathrm{y}/,\ /\mathrm{yz}/,\ /\mathbf{x}\mathrm{z}/,\ /\mathrm{zy}/,$$

where $/\mathbf{x}/$ were randomly selected from all the segments other than /y/ and /z/.

We perform our experiment using two different ways of encoding the segments /a/ through /z/ in the neural network: localist and distributed. In both of these representations, each segment is represented by a length-26 string of bits, where each bit is a 0 or a 1. In the localist representation the *k*th segment in the alphabetic ordering is represented by a string of all 0s except for a 1 in the *k*th place. To show the effects we document are not particular to this kind of representation, we also consider a distributed representation of segments. In the distributed representation we randomly generate a 26-bit representation of each of the 26 segments. Each bit in each representation is selected to be 0 or 1 with equal probability.

For each learner, the experiment was independently repeated N times with different random seeds. Since the distributed encodings were randomly generated, the exact encoding of each segment is almost certainly different between two repetitions of a given run.

6.2. *Neural Network Learners*

We tested our theoretical finding on the most popular model in the machine learning literature today: the multilayer artificial neural network. The words were fed into the neural network by simply concatenating the two 26-bit codes of their letters. We experimented with many different architectures, ranging from one to three hidden layers, and from 256 to 1024 units per hidden layer. We used tanh nonlinearities for all hidden units. We trained the models via backpropagation using an iterative quasi-Newton gradient descent algorithm called the limited memory Broyden-Fletcher-Goldfarb-Shanno method (L-BFGS), with a maximum of 100 iterations. Both the neural network and its optimization are implemented in `torch` (Collobert et al. (2002)), a state-of-art deep neural network package.

6.3. *Results*

We present results for the case of each hidden layer having 256 units, as the results are similar for more units per hidden layer. In Figure 1, for the localist encoding, we plot the average score output by the neural network for each of the test words above, for 1, 2, and 3 hidden layers. In addition, the averaged training scores are reported in the top two bars of each panel. While these show that all the architectures were able to learn from the dataset, the six bars below them show how each and every one of them failed to generalize to words containing the new letters. In Figure 2, we show that this remains true in the case of distributed representations.

7. Alternations

Up until now we have considered the problem of learning a map which takes words w to scores which are numbers. We considered symmetries that just acted on the input words w. In the phonological domain this corresponds to the problem of phonotactics: learning how to judge how well-formed a word is. However, another important computational problem in phonology is that of alternations. A typical example would be going from the singular form of a English noun to the plural form. In this case both input and outputs are words, and instead of symmetries applying just to inputs, as we have done until now, we can apply symmetries to both inputs and outputs.

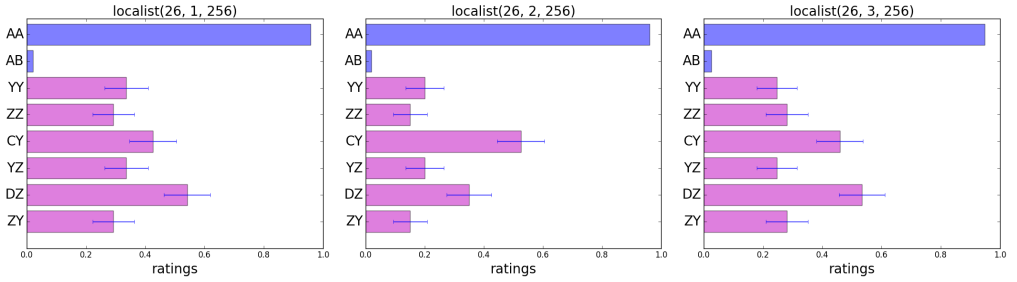

FIGURE 1. Scores for various words for the network with localist encoding for 1, 2, and 3 hidden layers.

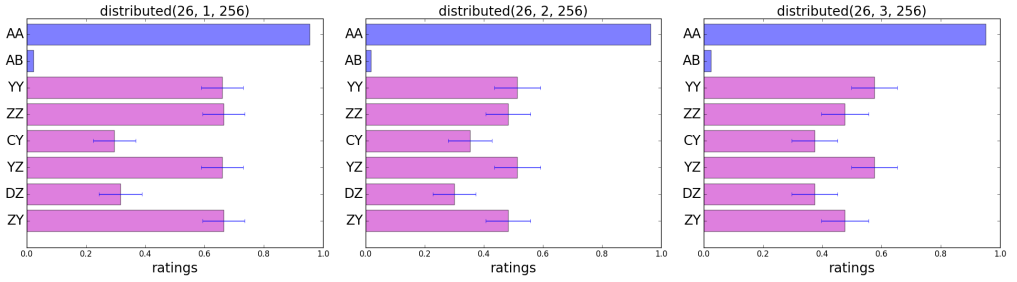

FIGURE 2. Scores for various words for the network with distributed encoding for 1, 2, and 3 hidden layers.

Just as in phonotactic judgements, identity effects play a role in alternations. Gallagher (2013) investigates whether subjects learn an identity-based rule in an artificial language learning task faster than a rule based on an arbitrary collection of segments. In Experiment 1 of her paper, results show that subjects learn the identity-based rule faster than a similar rule based on an arbitrary collection of segments. Using an extension of the framework we have developed so far, we illustrate a class of algorithms that will invariably perform equally well on the two rules, in contrast to human behaviour.

In Gallagher (2013) the subjects are required to learn an alternation (in other words, an input-output map) from examples under two conditions. They are given inputs of the form $C_1V_1C_2V_2$, in which C_1, C_2 are consonants and V_1, V_2 are vowels. In the IDENTITY condition, if C_1 and C_2 are distinct voiced consonants then the output is obtained from the input by devoicing the second consonant (e.g. gabu $\rightarrow$ gapu). On the other hand, if C_1 and C_2 are identical, then the output is the same as the input (e.g. bibo $\rightarrow$ bibo). In the SEGMENTAL condition whether there is an alternation or not depends on the root having an arbitrary combination of voiced consonants. The precise alternations are given in Table 2. In both conditions the vowels do not undergo change and do not condition change so they are omitted from the table.

The experimental result is that subjects learn the pattern faster in the IDENTITY case then the SEGMENTAL case. However, the "plain" version of the UCLA phonological learner (Hayes and Wilson 2008) does not learn the IDENTITY pattern faster. We want to find conditions on an algorithm under which the two patterns are learned equally fast.

For this particular experimental task, we specify a symmetry σ (as defined in Section 3) on the set of all two-consonant strings. We define σ by saying it takes a second /b/ in a word and replaces it with a /d/, it takes a second /d/ to a /g/, it takes a second /g/ to a /b/, a second /p/ to a /t/, a second /t/ to a /k/, and a second /k/ to a /p/. Thus we can express σ as

$$\sigma(\mathbf{x}b) = \mathbf{x}d, \quad \sigma(\mathbf{x}d) = \mathbf{x}g, \quad \sigma(\mathbf{x}g) = \mathbf{x}b,$$
$$\sigma(\mathbf{x}p) = \mathbf{x}t, \quad \sigma(\mathbf{x}t) = \mathbf{x}k, \quad \sigma(\mathbf{x}k) = \mathbf{x}p.$$

TABLE 2. The alternations to be learned by subjects in Gallagher's Experiment 1 (Gallagher (2013)).

IDENTITY		SEGMENTAL	
Input	Output	Input	Output
bd	bt	bg	bk
bg	bk	bb	bp
db	dp	dd	dt
dg	dk	db	dp
gb	gp	gd	gt
gd	gt	gg	gk
bb	bb	bd	bd
dd	dd	dg	dg
gg	gg	gb	gb

where **x** is any consonant and $\sigma(w) = w$ for all other words. We note that σ is a symmetry as defined in Section 3, since no two inputs get mapped to the same output. The mapping σ provides a connection between the IDENTITY and the SEGMENTAL condition in Gallagher's Experiment 1. In Table 2, any consonant pair under the IDENTITY heading gets mapped to the corresponding consonant pair under the SEGMENTAL heading.

We can view the goal of the subject's task as learning a mapping π from the set of all two-consonant strings to itself. We define π_I to be the mapping for the identity condition and π_S to be the mapping for the segmental condition. Our observation about Table 2 amounts to the fact that $\sigma(\pi_I(w)) = \pi_S(\sigma(w))$ for all w, as can be checked for each input in the left-most column of the table.

We formalize the concept of a phonological learning algorithm exactly as in Section 3, except that now D consists of a set of input-output pairs, where both input and output are words. Applying a symmetry σ to D consists of applying σ to both the input and output words of each pair in D. Now $\mathscr{L}(D,w)$ is an output word as well, what the algorithm concludes is the output corresponding to input w when trained on the data D. We say that $\mathscr{L}$ is invariant under the symmetry σ if $\mathscr{L}(\sigma(D),\sigma(w)) = \sigma(\mathscr{L}(D,w))$ for all w in W.

As in the case of phonotactics, algorithms for learning alternations may involve randomness, either in going from the training data D to the parameters in a model, or in going from the parameters to an output. In either case, $\mathscr{L}(D,w)$ is a random output word for fixed D and w. Accordingly, in this case we define symmetry of $\mathscr{L}$ under σ as being

$$\Pr[\mathscr{L}(\sigma(D),\sigma(w)) = v] = \Pr[\sigma(\mathscr{L}(D,w)) = v]$$

for all v and w in W, where we have used Pr to denote the probability of an event. We define the accuracy of a learning algorithm $\mathscr{L}$ in learning map π from data D to be the average probability of the algorithm being correct over all words in W:

$$\frac{1}{|W|}\sum_{w\in W}\Pr[\mathscr{L}(D,w) = \pi(w)]$$

where $|W|$ is the number of words in W. The accuracy is a number between 0 and 1.

Our claim is that any algorithm that is invariant to the symmetry σ defined above will learn π_I

from the IDENTITY condition data with the same accuracy as it learns π_S from the SEGMENTAL condition. This is implied by the following theorem.

THEOREM 3. *Let σ be a symmetry of the set of words W. Let π_1 and π_2 be mappings from W to itself such that*

$$\sigma(\pi_1(w)) = \pi_2(\sigma(w))$$

for all w in W. Suppose $\mathscr{L}$ is a learning algorithm invariant under σ. Then the accuracy of $\mathscr{L}$ learning π_1 from D is the same as the accuracy of $\mathscr{L}$ learning π_2 from $\sigma(D)$.

Proof.

$$\begin{aligned}\Pr(\mathscr{L}(\sigma D, z) = \pi_2(z)) &= \Pr(\mathscr{L}(\sigma D, \sigma w) = \pi_2(\sigma(w))) \\ &= \Pr(\sigma\mathscr{L}(D, w) = \sigma(\pi_1(w))) \\ &= \Pr(\mathscr{L}(D, w) = \pi_1(w))\end{aligned}$$

where we have defined $z = \sigma w$, used the hypothesis that $\mathscr{L}$ is invariant under σ and the connection between π_1 and π_2, and used the fact that σ is a symmetry. Summing over all words w in W (which is equivalent to summing over all z in W since σ is a symmetry) and then dividing by $|W|$ gives the result. □

REFERENCES

Benua, L. (1995). Identity effects in morphological truncation. *University of Massachusetts occasional papers in linguistics*, *18*, 77–136.

Berent, I. (2012). *The phonological mind*. Cambridge University Press.

Berent, I., Marcus, G. F., Shimron, J., & Gafos, A. I. (2002). The scope of linguistic generalizations: evidence from Hebrew word formation. *Cognition*, *83*(2), 113 - 139.

Colavin, R., Levy, R., & Rose, S. (2010). Modeling OCP-place in Amharic with the Maximum Entropy phonotactic learner. In *Proceedings volume of the 46th meeting of the Chicago Linguistics Society.*

Collobert, R., Bengio, S., & Mariéthoz, J. (2002). Torch: a modular machine learning software library. *Technical Report IDIAP-RR 02-46*.

Gallagher, G. (2013, 8). Learning the identity effect as an artificial language: bias and generalisation. *Phonology*, *30*, 253–295. doi:

Hayes, B., & Wilson, C. (2008, 2016/01/09). A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, *39*(3), 379–440.

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Magri, G. (2015). Idempotency and chain shifts. In K.-m. Kim (Ed.), *Proceedings of WCCFL 33: the 33rd annual West Coast Conference in Formal Linguistics.* Cascadilla Proceedings Project.

Marcus, G. F. (1998). Rethinking eliminative connectionism. *Cognitive psychology*, *37*(3), 243–282.

Marcus, G. F. (2003). *The algebraic mind: Integrating connectionism and cognitive science*. MIT press.

Rumelhart, D. E., & McClelland, J. L. (1988). *Parallel distributed processing* (Vol. 1). IEEE.

Walker, R. (2000). Long-distance consonantal identity effects. In *Proceedings of WCCFL* (Vol. 19, pp. 532–545).