Running head: DYNAMICAL INSIGHT INTO STRUCTURE IN CONNECTIONIST MODELS

Tabor, W. (2009). Dynamical Insight into Structure in Connectionist Models. In Spencer, J. P., Thomas, M. S. C., and McClelland, J. L., Toward a Unified Theory of Development. Connectionism and Dynamic Systems Theory Re-considered, (pp. 165-181). Oxford: Oxford University Press.

Dynamical Insight into Structure in Connectionist Models

Whitney Tabor

University of Connecticut

Please address correspondence to: Whitney Tabor

Department of Psychology, U-1020

University of Connecticut

Storrs, CT 06269

860 486 4910 (phone)

860 486 2760 (fax)

whitney.tabor@uconn.edu

Abstract

I discuss a connectionist model, based on Elman's (1990, 1991) Simple Recurrent Network, of the acquisition of complex syntactic structure. While not intended as a detailed model of the process children go through in acquiring natural languages, the model helps clarify concepts that may be useful for understanding the development of complex abilities. It provides evidence that connectionist learning can produce stage-wise development emergently. It is consistent with prior work on connectionist models emphasizing their capability of computing in ways that are not possible within the symbolic paradigm (Siegelmann, 1999). On the other hand, it suggests that one mechanism of the symbolic paradigm (a pushdown automaton) may be identified with an attractor of the learning process. Thus, the model provides a concrete example of what might be called "emergence of new conceptual structure during development" and suggests that we need to use both dynamical systems theory and symbolic computation theory to make sense of it.

Dynamical Insight into Structure in Connectionist Models

Introduction

This paper adds its voice to those that have pointed out that connectionist models are a subclass of dynamical systems. They are a particularly interesting subclass because of their psychological plausibility. Moreover, they have a valuable trait: one can make connectionist models that behave in psychologically plausible ways but which are, at least initially, difficult to understand. The difficulty is connected with the fact that the plausible behaviors are achieved through a learning algorithm, a simple, low-level process that tunes the parameters (or "weights") of the network on the basis of repeated interactions with a body of data. Although the principle of the learning algorithm is well-understood—it says, in essence, at each step, make a small change that improves performance—the pattern of weights that results from training a network on an interesting problem is typically quite complex and difficult to interpret. Such inscrutability might seem like a negative point. But, one can sometimes learn things by striving to see into it. Dynamical systems theory is helpful in this regard: it provides conceptual constructs (e.g., trajectories, basins, attractors, bifurcations) and analytical methods (e.g., phase-space portraits, bifurcation analysis) that can help one understand relationships and causes in these psychologically relevant models. (For introductions to the dynamics terms, see Abraham & Shaw, 1984 and Strogatz, 1994). Dynamical systems theory has been embraced by Ecological Psychology as the best formal means of clarifying the notion that psychological behaviors are emergent phenomena in organism-environment systems (Kugler & Turvey, 1987).

In addition to the connectionist method of psychological modeling and the accompanying analytic framework of dynamical systems theory, it is useful, at this point, to mention a third element: symbolic computation theory. By symbolic computation

theory, I mean theories which describe behavior using discrete symbols and well-formedness rules. Observed behaviors are modeled as allowable symbol patterns under the well-formedness rules. The grammatical theories of natural language proposed by Generative Grammar are examples of this kind of theory (e.g., Chomsky, 1957). The symbolic approach has generally been treated as an enemy by both connectionists and ecological psychologists, including many dynamical-systems oriented ecological psychologists. While I agree with the rejection of symbolic models as a sufficient foundation for the science of organism-environment systems, I argue here that symbolic models are nevertheless particularly relevant to cognition, and in fact, to the relationship between the connectionist and dynamical approaches. In a nutshell, symbolic models correspond to certain idealized behaviors which connectionist models can exhibit and which influence learning the way an attractor can be said to influence the trajectory of a dynamical system, even though realistic connectionist systems rarely, if ever, spend time at these attractors. Symbolic models are thus inadequate as a basis for psychology because they are only a proper subpart of the broader dynamical framework which is needed to characterize cognition. On the other hand, they are an important proper subpart.

I'll make this more concrete by talking about one major domain: development. Several people have suggested that the dynamical notion of a phase transition is particularly relevant to development (Thelen & Smith, 1994; van der Maas & Molenaar, 1992; van Geert, 1998). In the sense intended here, a phase transition is a qualitative change in the behavior of a system that is associated with a monotonic and continuous change of a parameter, called a *control parameter*. Phase transitions seem potentially relevant to modeling developmental stages because they correspond to situations in which behavior changes subtly and quantitatively for a long period of time and then suddenly changes dramatically and qualitatively. Indeed, the researchers mentioned above have pointed out plausible resemblances between developmental data and various mathematical

models of phase transitions. Here, I go along this path and then make an additional suggestion: developmental phase transitions may involve a shift from a phase in which well-formedness rules fail to characterize the structure of behavior to a phase in which these rules constitute a good approximation of the behavior. Thus, the account treats the symbolic models as relevant to characterizing stages, but not sufficient for characterizing all the states people go through in the process of growing up.

What is the evidence for these claims? It takes the form of a connectionist model called a Fractal Learning Neural Network (FLNN). I want to make clear at the outset that I offer no measured developmental data to motivate this model at all. Instead, one might think of this work as an exploration of abstract information-structure. The data that motivate the model are the linguistic observational data that reveal recursive patterning in every natural language. For two reasons, I believe that the results may be of interest to developmentalists: 1. The model is a connectionist learning model and thus has relatives which have made plausible empirical predictions about many domains, including development; 2. The model helps a person think clearly about an appealing alternative to some current theories of development (e.g., maturational explanations for stages, nativist explanations for grammatical complexity). The model's account is appealing because (a) it uses a simple learning mechanism and thus corresponds to a theory that links development and learning, (b) it predicts stage-wise behavior with progressive quantitative changes occurring within the stages, a phenomenon also observed in development, (c) it derives the phase transitions from the learning, thus predicting, rather than stipulating, the stages, (d) it shows highly structured, domain-specific behavior arising from a mechanism which is capable of a wide variety of behaviors, not all of which are highly structured, thus, in some sense, deriving the structure, (e) it generalizes beyond its input in a way that is at least partially consistent with human generalization in the comprehension and production of recursive sentence structures.

I will describe the model and then return to these points.

Description of the Model

Background

Elman (1990, 1991) trained a network, which he dubbed the "Simple Recurrent Network", to process sequences of symbols, including sequences of words. The network (Figure 1) has three layers of units (input, hidden, and output) as well as a context layer, which stores a copy of the hidden unit activations from the previous time step. The connections feed forward from input to hidden to output and from context to hidden. The copying operation associated with the context layer is a kind of recurrence, and it permits the network to encode information about past inputs, allowing it to keep track of the temporal dependencies which are crucial to syntactic processing.

In one experiment, Elman trained his network on a corpus of words approximating the syntactic properties of English. The corpus included singular and plural nouns, as well as center-embedded relative clause structures (1)[1] (articles like "the" and *a* were left out for simplicity).

(1) dogs/dog who girls chase run/runs.

In an example like (1), the network must store the number of the head noun of the subject of the sentence ("dog" or "dogs") during the time that it processes the relative clause ("who girls chase") in order to properly predict the number of the main verb ("run" vs. "runs"). Elman used a single-on-bit ("one-hot") encoding for each word on the input layer and on the output layer. The codes of the words of the corpus were presented in order, one at a time, on the input layer of the network. The network was trained (using the sum-squared error cost function and backpropagation of the error signal through time D. E. Rumelhart, Hinton, & Williams, 1986; Pearlmutter & MacDonald, 1995) on the task

of predicting the next word at each point in time. Because the sequence of sentences in the corpus was approximately random, the network was not able to guess specifically what word was going to appear at any point in time. However, it could pick up on stable syntactic properties of the language. In particular, once the network was fully trained, the activation of each output unit approximated the probability that the word corresponding to that output unit would occur next in the corpus. For example, after having encountered the word "dog" at the beginning of a sentence, output activation was concentrated mostly on the singular verbs and the word "who". While such local co-occurrence tendencies are fairly easy to model (e.g., with Markov processes), some kinds of longer distance dependencies, including those arising in center-embedded structures like (1), are more difficult. Thus, it is significant that the network was also successful at predicting the number on the verbs in relative clause structures like (1).

Since Elman's initial experiments, several projects have successfully used the Simple Recurrent Network and variants on it to learn complex grammatical dependencies in other human-language-like corpora (e.g., Christiansen & Chater, 1999; Rohde & Plaut, 1999; Rohde, 2002). Several other projects have investigated the learning of abstract formal languages, related to human languages, in order to better understand the principles by which the network was succeeding at prediction (Bodén & Wiles, 2002; Rodriguez & Wiles, 1998; Rodriguez, Wiles, & Elman, 1999; Rodriguez, 2001; Wiles & Elman, 1995). Gers, Schmidhuber, and Cummins (2000); Gers and Schmidhuber (2001); Hochreiter and Schmidhuber (1997) propose a different, but related kind of recurrent network called "Long Short Term Memory" (LSTM) which could keep track of very long temporal dependencies (up to 1000s of intervening events).

By an abstract formal language, I mean a set (finite or infinite) of finite-length strings of symbols drawn from a finite symbol alphabet. A foundational idea in the theory of computation is to ask, for a given formal language, what kind of computing device is

needed to decide if a particular string is, or is not, a member of that language (Chomsky, 1956; Hopcroft & Ullman, 1979; Kozen, 1997). Being able to do this for any finite string presented is called *recognizing* the language. The answer is very uninteresting if the set of strings is a finite set: the device can simply be a big list of the strings of the language; when a string is presented for testing, the device simply runs through the list, comparing each list string to the test string and accepts the test string if it finds a match. The situation becomes more interesting when infinite sets of strings are considered.[2] The list approach runs into problems with lengthy searches in this case, but for an important class of examples, one can build compact symbol-manipulating devices that take advantage of the structures of the languages in order to recognize them. These come in some interesting subclasses. The most basic is called the *finite-state machine*: it consists of a computer whose "brain" can only be in a finite number of states. A somewhat more complex type, called a *pushdown automaton*, stores a sequence of symbols in memory and then recovers them in reverse order, retrieving each at just the moment when the environment indicates that it is time to recall it. The string of symbols is called a *stack*. The stack turns out to be a very useful device: it underlies *recursion*, the primary mechanism that gives modern computer programming languages their power; it has been argued to form the basis for the myriad complex patterns that occur in plant growth (Lindenmayer & Prusinkiewicz, 1989); it is related to fractal geometry, which is widely observed in nature (Mandelbrot & Ness, 1968); in one guise or another, it is a primary workhorse of all formalized linguistic theories (Gazdar, 1981). Another name for the class of languages recognized by pushdown automata is *context-free languages*. Context-free languages can be described either by pushdown automata or by finite lists of rules of the form $X \rightarrow \beta$ where β is a finite string of symbols. An example is shown in Table 1.

The projects mentioned above, on neural mechanisms that learn to recognize formal languages on the basis of experience with them, focus on the learning of context-free

languages. Although the networks were successful on one simple case (the language, $a^n b^n)^3$ and the analyses provided clear insight into the method by which the networks were succeeding (see further discussion below), the networks were much less systematically successful with more complex cases (e.g., palindrome languages), which more closely resemble human languages. Meanwhile, work on the encoding of complex symbolic structures in connectionist units independent of the learning question (Moore, 1998; Tabor, 2000) has provided some insight into how it is possible to encode complex syntax in connectionist architectures.

The construction of the Fractal Learning Neural Network (Tabor, 2003) was motivated by the combination of these research strands. It combines the representational results of Moore and Tabor with a hill-climbing algorithm that discovers weight settings embodying complex symbolic processes. I'll describe the architecture of this device, along with some simulation results, and then make some comments on light that this effort may shed on development and on the relationship between dynamical systems and connectionist approaches.

Mechanism and Architecture.

The world of stacks and grammars is a symbolic world where set membership is the natural currency, but quantitative measurement seems to have little meaning. On the other hand, canonical dynamical systems in general, and connectionist networks in particular, live in metric spaces where real-valued distances between states play a central role in the system dynamics. Where might these disparate worlds meet? A plausible answer is *fractal sets*, or recursive structures that lie in metric spaces. A fractal set is a set that exhibits self-similarity over an infinite range of scales (scale-invariance), including arbitrarily small ones. The best known example is the Cantor set, the set that remains when one starts with a line segment, removes the middle third to produce two smaller line segments, removes the middle thirds of each of these, ad infinitum. The sets of points in

the left and right thirds of the original segment are called the left and right *branches* of the fractal, respectively. Levy, Melnik, and Pollack (2000), Melnik, Levy, and Pollack (2000), Moore (1998), and Tabor (2000) show how to use generalizations of the Cantor set to make dynamical systems that recognize context-free languages. The system starts at a pre-specified location on the fractal and jumps to another part of the fractal whenever a symbol of a string is processed; it uses the recursive spatial structure of the fractal to keep track of the recursive symbol embedding of the language.[4]

The FLNN is a learning connectionist network related to Elman's Simple Recurrent Network which is designed to interact with sentences from a context-free language and discover how to move around on appropriate fractal and recognize strings from the language. The FLNN architecture has four layers (Figure 2). The first layer is the input layer, where each unit stands for a word. The second layer has a linear activation function with linear recurrent connections that are gated by the input units. It is in this layer that the fractal arises. Cantor sets and some of their generalizations can be constructed with affine operators on vector spaces (1) (Barnsley, [1988]1993).

$$\vec{z} \leftarrow a\vec{z} + \vec{b} \tag{1}$$

Here, $\vec{z}$ is the vector of activations on the second layer. The gating connections, by which the input units specify the weights of the recurrent self-connections in the hidden layer, allow the input pattern to specify the parameter a, which is set to a value smaller than one for contraction (i.e., moving to a smaller scale on the fractal) and a value greater than one for expansion (i.e., moving to a larger scale on the fractal). The linear connections from the input to the hidden layer allow the network to specify the parameter, $\vec{b}$, which determines which branch of the fractal the system occupies at each point in time. The third layer has Gaussian (radial basis function) units whose means are learned but whose variances are fixed (arbitrarily) at 0.25. Gaussian units are a kind of on-center,

off-surround unit: they activate strongly when the vector of their inputs is in a particular spherical region centered at their mean and of radius specified by their variance. These third-layer units end up being used to detect which branch of the fractal the second layer is on at each point in time[5]. The fourth layer is the output layer. In the output layer, as in the input layer, each unit stands for a word. The output units as a group have the normalized exponential (softmax) activation function because they specify probability distributions over words. In the fractal grammar recognizers of Moore (1998) and Tabor (2000), the set of possible successor symbols at any point in time is determined by the branch of the fractal that the system inhabits at that point in time. In the FLNN, as in Elman's Simple Recurrent Network, "set of possible successor symbols" is translated into a probability distribution over successor symbols with the probabilities approximating the actual occurrence rates.

There is an important question about the number of units in the hidden layers. I chose the number of units to be the minimal number that I knew were necessary to solve the task well (in this case 2 for the recurrent hidden layer, and 3 for the Gaussian layer). This constraint may play a role in the success of generalization touted below. Further testing is needed to find out whether the network generalizes as well when it has an excess of hidden units.

Training.

I trained the network on simple formal languages that are more complex than those studied in prior work that has been successful at fully interpreting network symbol processors' hidden unit representations (Gers et al., 2000; Gers & Schmidhuber, 2001; Rodriguez & Wiles, 1998; Rodriguez et al., 1999; Rodriguez, 2001; Wiles & Elman, 1995). On the other hand, they are not as complex as the corpora used by researchers focused on the details of the resemblance of the network's behavior to natural language (Elman, 1990, 1991; Christiansen & Chater, 1999; Rohde & Plaut, 1999; Rohde, 2002). See Tabor (2003)

for a discussion of this complexity issue.

The training data for the network were generated from the grammar shown in Table 1. In the simulation reported here, the training corpus consisted of one instance of each sentence from the grammar that has 9 or fewer words. Thus the maximum number of center-embeddings was 2.

At the beginning of each sentence, the two recurrent hidden unit activations were set to 0. The words were presented one at a time, in order, on the input layer of the network. The grammar was used to determine output probabilities following each word. In the simulation reported here, these probabilities were used as the targets of training.[6]

The Gaussian Unit means and the Gaussian → Output weights were set to random initial values with mean 0 and variance 0.3. The gated self-connections in the hidden layer were set to initial value 1 (this is the unbiased choice; Tabor, 2000). The recurrent connections between different units (as opposed to the recurrent connections from each hidden unit to itself) were clamped to 0 throughout training and testing. This choice, also, was motivated by the knowledge that these connections would not be needed for the task at hand.

The network was trained by hill-climbing with epoch-sized steps. The first step of hill-climbing was to present every sentence in the corpus to the network and compute the total error at the output layer. Kullback-Leibler (KL) Divergence was used as the error measure.[7] Then, a variety of slight distortions of the current weight setting was examined (in particular, each weight was adjusted slightly, while keeping all the others fixed). For each of these distorted weight settings, the entire corpus was tested and error was again accumulated. If one of these adjustments made a maximal improvement in the error, than it was adopted as the new current weight setting. (Ties for best were broken arbitrarily). This process was repeated until total error dropped below a pre-specified threshold (0.02).

Test Results.

As noted, the network was trained on all sentences of length up to 9 (at most 2 levels of center embedding). The network was tested on all sentences of length 10 to 15 (at most 4 levels of center embedding). The results are shown in Table 2.

Table 2 indicates that while the network's performance on the test corpus was not perfect, it was very nearly so.

Analysis of the weights.

In line with the scheme developed in Tabor (2000), the trained network has discovered how to move around on a fractal set to keep track of the dependencies in the ABC language. An illustration of the learned fractal is shown in panel (e) of Figure 3. Tabor (2003) and Tabor (2005) provide evidence that the learned weight set is organized on the same principles as the ideal fractal discussed in Tabor (2000).

Code.

Matlab code for running these simulations is available at

http://www.sp.uconn.edu/~ps300vc/Projects/FractalLearning/index.html

This code runs slowly but is fairly simple to interpret.

Connectionist networks and dynamical systems theory

Wiles and Elman (1995), Rodriguez and Wiles (1998), Rodriguez et al. (1999), and Rodriguez (2001) analyzed Simple Recurrent Networks trained on simple, context-free languages, related to the case discussed above. They showed that standard dynamical systems constructs (trajectories, attractors, repellers) provided a foundation for understanding how the network successfully generalized from the finite sample it was trained on. For example, Rodriguez et al. (1999) showed that a network trained on the language, $a^n b^n$, employed two quasi-linear dynamical objects: an attractor and a repeller. When the code for a was presented on the input layer, the network's hidden unit dynamics consisted of a single attractor basin with a stable fixed point. At the beginning

of processing, the hidden unit activation vector was relatively far from the attractor. Each successive presentation of an a caused the state to jump a step closer to the fixed point. When the input was switched to b, the result was a new dynamical regime, similar to the a regime, but with a repeller that was located on the other side of hidden unit space. The first presentation of b moved the state to a location close to the repeller. The distance between this point and the repeller was a decreasing function of the number of a's that had been presented. Additional presentations of b caused the network to jump away from the repeller, and head back to the location associated with the beginning of a sentence. The hidden state arrived back at this location when the number of b's matched the number of a's that had been presented. The Hidden $\rightarrow$ Output weights of the network implemented linear separations of the hidden unit space so that when the network was near the periphery of the attractor's basin, it expected the start of a new sentence (only possible next word = a), when it was jumping toward the attractor, it predicted a and b, and when it was in the region of the repeller, it predicted only bs until it arrived back at the start-of-sentence region. Thus, the balancing of the number of a's with the number of b's was accomplished by matching the contraction effect of the attractor with the expanding effect of the repeller.

Rodriguez et al.'s network moves on a simple fractal, the contractive geometric series, to keep track of levels of embeddings. A contractive geometric series is a series of numbers of the form ar^n, where a is a nonzero constant, r is nonzero between -1 and 1, and $n = 0, 1, 2, \ldots$. At the end of training, the FLNN described above moves on a slightly more complex fractal, a two-dimensional Cantor Set, to simultaneously track levels and types of embedding. As in Rodriguez et al.'s case, the system is driven by a collection of attracting and repelling fixed points with carefully coordinated interactions.

Thus, if these cases are typical, they suggest the following relationship between connectionist models, dynamical systems theory, and symbolic processes: certain

connectionist networks with certain weight settings mimic symbolic processes. Moreover, certain dynamical objects are formally related, via the networks, to the symbolic entities (Crutchfield & Young, 1990; Moore, 1998): for example, when the attractor and repeller are in perfect coordination in the symbol prediction network and thus travel around on a Cantor-set like fractal, the network generates the same formal language as a particular pushdown automaton; in fact, for every pushdown automaton (= context-free) language, there exists a fractal traversing network that generates the language (Tabor, 2000). This suggests that dynamical systems theory is in a position to provide insight into the relationship between symbolic and connectionist models. The next section explores the process by which the network approaches such a symbolically organized state, and provides some evidence that the relationship proposed here is not a fluke, enforced by excessive training, but reflects a fundamental relationship that may extend to other cases. The next section also suggests that not all states of the network's development correspond to symbolically describable behaviors.

Phase Portraits and Review of Claims

Figure 3 is a succession of phase portraits[8] of the model over the course of training on the ABC language. The portraits show the points visited by the model when it processes sentences containing up to 7 levels of embedding. Only points visited during the processing of initial strings containing a's and b's are shown. If strings containing c's were shown, the pictures would look similar, but fuzzier, because the model imperfectly approximates the symmetry between contraction and expansion.

Figure 3 illustrates several successive, qualitatively distinct stages of the model's behavior over the course of training. With this illustration in view, I review the claims about "appealing properties" made above and assess their validity:

(a) *The model learns.* The model clearly involves a tuning process which brings it

from domain-oblivious behavior to highly domain-sensitive behavior. Hill-climbing itself is not biologically plausible, for it involves thorough exploration of the environment every time a weight is adjusted. Nevertheless, its success is a prerequisite for the success of currently known biologically plausible algorithms (e.g., Hebbian learning).

(b) *The model passes through qualitatively distinct stages.* Distinct stages can be observed in Figure 3. The dimensional expansions that have occurred between Stage a and Stage b and between Stage b and Stage c, along with the "unfolding" that completes itself between Stage d and Stage e appear to be associated with bifurcations, the mathematical analogues of phase transitions.[9] These bifurcations correspond to changes in the symbolic computational behavior of the system. For example, at Stage a, the model generates the simplest finite state language on a three-symbol alphabet: there is only one state, and any symbol can be produced at any time. Stages b and c approximate progressively more complex finite state devices in that the network becomes increasingly sensitive to differences between the symbols but exhibits no contraction or expansion and hence cannot generate recursive structure. Stage d involves contraction and expansion but the recursion is confused in a complex way which may correspond to super-Turing computation (Siegelmann, 1999). Stage e approximates a pushdown automaton. This apparent correspondence between bifurcations and grammar-changes is one foundation of the claim that there is a link between the dynamical and symbolic computational perspectives—see also Crutchfield and Young (1990); Moore (1998).

The model exhibits "progressive quantitative change occurring during the stages". For example, during the "unfolding" stage illustrated in 3c and 3d, the fractal scale at which the branches first overlap is continuously decreasing. Thus, the depth of embedding that the model can successfully parse is gradually increasing. This is a "progressive quantitative change" because it makes the model's behavior more like the fractal parser it eventually becomes, even though it has not yet crossed the threshold (between 3d and 3e)

of becoming isomorphic to a pushdown automaton.

What is meant by the claim that the model transits from a stage at which well-formedness rules are inadequate for describing its behavior to a stage at which they provide a good approximation? In 3c, the region spanned by the model during the processing of all sentences of the ABC language has infinite area. This is because contraction has not developed yet and the inputs are causing incremental, equal-sized displacements. But the interpretation of the model as an approximation of a pushdown automaton depends on its use of the fixed-width gaussian units in the second hidden layer to classify each grammatical condition it encounters. When the model does not reliably land within the "spotlights" of these gaussians, the basis of the isomorphism with a symbolic device is lost. Viewing the case from the perspective of self-organization, one might say that the model has not yet reached the stage where the constituents of the emergent structure are successfully coordinating.

(c) *The stages derive from the learning.* The qualitatively distinct stages come about through the iterative application of a single, simple process: move to the best proximal weight setting. In this sense, the stages are derived from the learning.

The derivation of the stages from the interaction of the simple, low-level learning process with data is interesting from a developmental standpoint. As noted under (b), the different stages appear to correspond to different classes of mental operations.[10] Under classical maturational accounts, stages are generally stipulated; the causes of their distinct qualities are assumed to be buried in the history of evolution. The current model offers a view of what an explanatory developmental account might look like. We can study the model to find out why the stages occur, whether they will occur in the same order every time, and what types of deviations of the architecture and environmental conditions can stop them. More generally, the network may be a useful formal example to consider when thinking about the question of what it means for an organism to come up with something

new.

In one regard, the current model offers an advance over Catastrophe Theory models (Thom, 1985) which identify traits that phase transitions can be expected to have but do not provide a great deal of insight into what causes the transitions. On the other hand, it is not clear that the bifurcations in the current model exhibit the full range of properties that Catastrophe Theory predicts (e.g., modality, sudden jumps, hysteresis, divergence, critical slowing down, etc.) and for which there is compelling evidence in the developmental literature (van der Maas & Molenaar, 1992; van Geert, 1998). If this discrepancy is present, it may stem, in part, from the fact that the FLNN does not model real-time processing (its smallest grain, the word, is what might be called "sentence-time processing"). In any case, further assessment of the relationship between Catastrophe Theory and the current, fractal learning approach may be fruitful.

(d) *Complex, domain-specific behavior from a general mechanism.* Expert evidence from currently practicing linguists on the types of grammatical patterns present in natural languages finds all of them to be within the symbolic paradigm. I'm skeptical that the sample identified by linguists reflects the true distribution. My skepticism stems from the fact that, until recently, people have not known how to look for anything else. Even within the symbolic paradigm, there is currently no satisfactory model of how a specific grammar is chosen from among the immense number of possibilities the paradigm affords. For these reasons, again recognizing that further empirical justification is needed, I treat it as desirable to adopt a model with very general computational capabilities and to let learning do significant work. This is in contrast to established linguistic treatments of native capacity (e.g., Gibson & Wexler, 1994) which assume that the complex machinery is pre-fabricated by the genes, and only finitely-many discrete parameter settings must be selected in order to achieve adult behavior.

That connectionist learning is a very general induction system is well attested.

Here, we appear to see it building a fairly complex kind of mechanism, a pushdown automaton, to suit the needs of a particular task. The question that needs to be asked is, To what extent was this result contrived by the constraints put on the model? If the observed pushdown stack is the only complex behavior the model is capable of, or if it is one of a small, finite number of possibilities, then the fact that the learning algorithm discovered it is not very revealing—under these conditions, the treatment would be very similar to current nativist accounts. On the other hand, if many behaviors are possible, then the model offers an appealing alternative to current versions of nativism in language learning theory. I know this much at present: close relatives of this model are capable of a great variety of behaviors, including all context-free languages (Tabor, 2000), a crossed serial dependency language (Tabor, 1991), several finite-state languages (Tabor & Terhesiu, 2004), and several chaotic processes (Tabor & Terhesiu, 2004). I do not yet know how many of these behaviors can be learned. Also, the question remains open as to whether these FLNN learning results will scale to more realistic natural language corpora.

Generalization. When trained on shallow embeddings (e.g., down to 2 center embeddings) the model generalizes to greater depths (3+). In fact, its end-of-training behavior closely mirrors the behavior of the infinite-state pushdown automaton that inspired the training data.[11] The fact that the neural model generalizes to this well-known and very useful type of mechanism on the basis of a minute number of examples (4 sentences in the depth 2 corpus for the ABC language) is surprising. Further tests show that when the model is trained on various other small subsets of the infinite state ABC language, it converges on the same, infinite-state behavior. This suggests that the infinite state behavior is a kind of attractor for the learning process. These observations constitute another piece of evidence in support of the view that there is a systematic relationship between the symbolic and dynamical perspectives. They also imply a kind of causal role for symbolic structures in the developmental dynamics of

connectionist models: the attractor steers the learning process.[12] This view is at odds with discussions which treat connectionism as a refutation of the relevance of symbolic models. On the other hand, it is also at odds with symbolic models which want to build in rules natively: on the present view, the recursive structure stems from a very general property of neurons (potential for fractal organization) which is closely related to the nature of physical space, and thus is not an arbitrary property of particular organisms.

But how relevant are these infinite state languages to real, human behavior? This is a point of long-standing controversy. Chomsky (1957) and many other people have argued that we ought to be studying ideal language ability—called "competence"—which is said to include the ability to process embeddings of arbitrary depth. That people cannot manage more than 3 levels of center-embedding easily (e.g., [The essay [that the guinea pig [someone [I knew $_4$] cared for $_3$] shredded $_2$] was about Duchamp $_1$].) is taken to be a performance-limitation: it is supposed that although the mechanism involves a pushdown automaton, the memory buffer for the symbol stack is limited, so we run out of space after only a few embeddings (Kimball, 1973). Others have argued that the distinction between competence and performance is dubious and that connectionist networks like Elman's Simple Recurrent Network (SRN) predict the limited ability of people to process center embedded sentences without needing to adopt the full power of a pushdown automaton (Servan-Schreiber, Cleeremans, & McClelland, 1991) and without needing to make a seemingly untestable distinction between competence and performance (Christiansen & Chater, 1999). The makers of the latter arguments offer connectionist models as examples of systems in which there is no distinction between competence and performance. The present results, however, suggest a different interpretation of at least some connectionist models: an FLNN only exhibits perfect recursive computation if there is perfect balance between contraction and expansion, and if weights and activation are computed with infinite precision. When any noise is added to the values, the system is computationally

equivalent to a finite-state machine (Casey, 1996). This observation suggests a close parallel between Turing machines and FLNNs: the infinite precision is playing the role of the infinite tape (Pollack, 1987). Moreover, the possible attractor status of recursive weight settings suggests that even though real systems only approximate these settings, the idealized settings are relevant theoretical constructs which can help us understand the principles by which the device is succeeding at the task. Put this way, the fractal structure is very much like an element of "competence" in the sense of Chomsky (1957). This line of reasoning casts some doubt on the claim that connectionist models provide a basis for rejection of the competence-performance distinction (but see McClelland, 1993).

However, the argument does not stop here. It should be noted that there are importance differences between the symbolic and FLNN realizations of the competence/performance idea. First, the current account is not a cavalier use of the distinction, in which data that do not match the theory are peremptorily labelled as "performance phenomena" and subsequently ignored. The model provides an explicit interpretation of both competence (a universe of topological structures) and performance (a particular location within that universe). Second, in order to predict that center-embeddings will be hard to process beyond three or four levels, the symbolic approach must bound the length of a stack to three or four elements, a stipulation which is not well motivated. By contrast, there is better motivation for claiming that noise is present in the neural implementation of a fractal computer; also, the combination of fixed variance neural noise with fractal scaling correctly predicts that the growth of difficulty in center-embedding should have a roughly exponential structure (Miller & Isard, 1964). Third, the treatment of recursion as an emergent property rather than an architectural property gives the system a kind of flexibility that seems suited to generating the variety of recursive structures that natural languages exhibit (e.g., center-embedding vs. crossed-serial dependency; Savitch, 1987) and the mutability, over periods of decades and

centuries, of grammatical structures (Hopper & Traugott, 1993). Fourth, the treatment of ideal devices as attractors rather than mechanisms raises the possibility that multiple ideals could simultaneously shape behavior, a prediction born out by some psycholinguistic work (Tabor, Galantucci, & Richardson, 2004) and possibly relevant to creole continua and other cases of language contact. Fifth, while the symbolic paradigm separates the ideal system from the perturbations of it caused by "performance factors" in a stark way, by defining the system as signal and the perturbations as noise, connectionist models claim that there is a continuum between the two. The latter position may be relevant for understanding how new structures arise in language change (Givon, 1971; Tabor, 1994, 1995) and for understanding why grammatical patterns in some languages parallel statistical patterns in others (Bresnan & Nikitina, 2003)—in such cases, noise may nourish metamorphosis, and thus should not be segregated in the theory. In these regards, the FLNN properties are consistent with the claim put forth by (Servan-Schreiber et al., 1991) that recurrent connectionist networks are a new kind of computational device that may naturally be called a "graded state machines". A fractal is aptly described as a "graded recursive structure".

Conclusions

The original motivation for developing FLNNs was to find a way for connectionist networks to effectively learn complex syntactic dependencies and to discern how connectionist treatments differ from symbolic treatments of these behaviors. It is important to clarify how much of this task has been accomplished and how much not, and to draw attention to other elements of language, not treated by this model, which may play an important role in the developmental picture.

First, it is known that connectionist models with these general characteristics (symbol predictors with recurrent hidden layers employing second order weights) are

capable of correctly processing all context-free languages (Moore, 1998; Tabor, 2000). I've only tested learning on two such languages, so it remains an open question how many of these parameterizations are reachable via learning.

Second, it is important to note that human language knowledge contains a good deal of systematic structure beyond that of phrase structure. One major element of this "other" structure goes by the name "pragmatic" (or "semantic") knowledge. Some have argued that pragmatic knowledge plays a central role in bootstrapping syntactic understanding (e.g., Pinker, 1984). Others have argued for the reverse causal relationship (e.g., Fisher, Gleitman, & Gleitman, 1991). I am disinclined to believe that the causality is serial in either direction, but a mutual causality view also attributes a significant role to pragmatic structure. Encouragingly, Elman (1990) found that pragmatic information reflected in the statistical properties of corpora was detected by a Simple Recurrent Network and was encoded via refinement of the syntactic encoding. Servan-Schreiber et al. (1991) found that subtle statistical biases of the type that plausibly stem from pragmatic knowledge can help an SRN learn syntactic structure. Weckerly (1995) provided sentence processing evidence that syntactic and pragmatic information are not independent but are intertwined in a way predicted by the SRN. I have not studied this kind of information structure in the FLNN yet, but it is clearly important, and a good next step will be to see if the fractal analysis can help elucidate the nature of the intertwining.

Final remark: In some dynamical approaches to development as well as in some connectionist work, there is a tendency to view ideal concepts as misleading. For example, Thelen, Schoener, Scheier, and Smith (2001) remark:

> Does this [Dynamical Field Theory] model have anything to say about Piaget's issue: when do infants acquire the object concept? We believe this question is ill-posed and cannot be answered because there is no such thing as an "object concept" in the sense of some causal structure that generates a thought or a

> behavior (Smith, Thelen, Titzer, & McLin, 1999). There is only "knowledge" of objects as embedded in the immediate circumstances and the history of perceiving and acting in similar circumstances. (p. 34)

In keeping with this view, the present work portrays a world in which knowledge is nothing more than experience-tuned action (Elman, 2004), there is considerable flexibility of behavior, and virtually any behavior can occur at any time in an organism's life if strong enough biases encourage it. And yet, according to the view put forth here, there is still value in referring to ideal, or absolute, forms: these are the limiting dynamical systems which exert a force-like influence on the learning process. An example in the present case is the infinite-state grammar which the network converges to in the limit of infinite training and the absence of noise. While it is important to recognize that a realistic network will never behave exactly like this infinite-state grammar, the grammar nevertheless provides insight into the nature of the system that the network "uses" to organize its knowledge of its training data. The current example thus encourages one to look for analogous attractors in other domains in which development appears to cleave to, or consort with, coherent patterns of behavior.

References

Abraham, R. H., & Shaw, C. D. (1984). *Dynamics—the geometry of behavior, books 0 - 4.* P.O. Box 1360, Santa Cruz, CA: Aerial Press, Inc. (Volume 1 of the Visual Mathematics Library.)

Barnsley, M. ([1988]1993). *Fractals everywhere, 2nd ed.* Boston: Academic Press.

Bodén, M., & Wiles, J. (2002). On learning context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, *13*(2), 491–493.

Bresnan, J., & Nikitina, T. (2003). *On the gradience of the dative alternation.* (Manuscript, Department of Linguistics, Stanford University. Available at http://www-lfg.stanford.edu/bresnan/)

Casey, M. (1996). The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, *8*(6).

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113-124. (A corrected version appears in Luce, Bush, and Galanter, eds., 1965 *Readings in Mathematical Psychology, Vol. 2*)

Chomsky, N. (1957). *Syntactic structures.* The Hague: Mouton and Co.

Christiansen, M. H., & Chater, N. (1999). Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, *23*, 157–205.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory.* New York: John Wiley and Sons.

Crutchfield, J. P., & Young, K. (1990). Computation at the onset of chaos. In W. H. Zurek (Ed.), *Complexity, entropy, and the physics of information* (pp. 223–70). Redwood City, California: Addison-Wesley.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179-211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, *7*, 195–225.

Elman, J. L. (2004). An alternative view of the mental lexicon. *Trends in Cognitive Science*, *8*(7), 301–306.

Fisher, C., Gleitman, H., & Gleitman, L. (1991). On the semantic content of subcategorization frames. *Cognitive Psychology*, *23*, 331-392.

Gazdar, G. (1981). On syntactic categories. *Philosophical Transactions (Series B) of the Royal Society*, *295*, 267-83.

Gers, F. A., & Schmidhuber, J. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, *12*(6), 1333–1340.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural Computation*, *12*, 2451–2471.

Gibson, E., & Wexler, K. (1994). Triggers. *Linguistic Inquiry*, *25*(3), 407–454.

Givon, T. (1971). Historical syntax and synchronic morphology: an archaeologist's field trip. In *Proceedings of the 7th regional meeting of the chicago linguistics society* (Vol. 7, pp. 394–415).

Guckenheimer, J., & Holmes, P. (1983). *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*. New York: Springer-Verlag.

Hinton, G. E. (1990). *Connectionist symbol processing*. Cambridge, MA: MIT Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, languages, and computation*. Menlo Park, California: Addison-Wesley.

Hopper, P. J., & Traugott, E. C. (1993). *Grammaticalization*. Cambridge, England: Cambridge University Press.

Jagota, A., Plate, T., Shastri, L., & Sun, R. (1999). Connectionist symbol processing: Dead or alive? *Neural Computing Surveys*, 1-40. (Available at http://www.icsi.berkeley.edu/ jagota/NCS)

Kimball, J. (1973). Seven principles of surface structure parsing in natural language. *Cognition*(2), 15-47.

Kozen, D. C. (1997). *Automata and complexity theory*. New York: Springer.

Kugler, P. N., & Turvey, M. (1987). *Information, natural law, and the self-assembly of rhythmic movement*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Levy, S., Melnik, O., & Pollack, J. (2000). Infinite raam: A principled connectionist basis for grammatical competence. In *Proceedings of the 22nd annual meeting of the cognitive science society* (pp. 298–303). Mahwah, NJ: Lawrence Erlbaum Associates.

Lindenmayer, A., & Prusinkiewicz, P. (1989). Developmental models of multicellular organisms: a computer graphics perspective. In C. G. Langton (Ed.), *Artificial life* (pp. 221–250). Redwood City, CA: Addison-Wesley. (A proceedings volume in the Santa Fe Institute Studies in the Sciences of Complexity)

Mandelbrot, B. B., & Ness, J. W. V. (1968). Fractional brownian motions, fractional noises and applications. *SIAM Review*, *10*(4), 422-437.

McClelland, J. L. (1993). The grain model: A framework for modeling the dynamics of information processing. In D. E. Meyer & S. Kornblum (Eds.), *Attention and performance xiv: Synergies in experimental psychology, artificial intelligence, and cognitive neuroscience* (pp. 655–688). Hillsdale, NJ: Lawrence Erlbaum Associates.

Melnik, O., Levy, S., & Pollack, J. B. (2000). RAAM for infinite context-free languages. In *Proceedings of the international joint conference on artificial neural networks (ijcnn).* IEEE.

Miller, G. A., & Isard, S. (1964). Free recall of self-embedded English sentences. *Information and Control*, *7*(3), 292–303.

Moore, C. (1998). Dynamical recognizers: Real-time language recognition by analog computers. *Theoretical Computer Science*, *201*, 99–136.

Pearlmutter, N. J., & MacDonald, M. A. (1995). Probabilistic constraints and working memory capacity in syntactic ambiguity resolution. *Journal of Memory and Language*, *43*, 521-542.

Pinker, S. (1984). *Language learnability and language development.* Cambridge, MA: Harvard University Press.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623-641.

Plate, T. A. (2003). *Holographic reduced representation: Distributed representation for cognitive structures.* Stanford, CA: CSLI Publications.

Pollack, J. B. (1987). *On connectionist models of natural language processing.* (Ph.D. Thesis, Department of Computer Science, University of Illinois)

Rodriguez, P. (2001). Simple recurrent networks learn context-free and context-sensitive languages by counting. *Neural Computation*, *13*(9).

Rodriguez, P., & Wiles, J. (1998). Recurrent neural networks can learn to implement symbol-sensitive counting. In M. Jordan, M. Kearns, & S. Solla (Eds.), *Advances in neural information processing systems 10* (pp. 87–93). Cambridge, MA: MIT Press.

Rodriguez, P., Wiles, J., & Elman, J. (1999). A recurrent neural network that learns to count. *Connection Science*, *11*(1), 5–40.

Rohde, D. (2002). *A connectionist model of sentence comprehension and production.* (Unpublished PhD Dissertation, Carnegie Mellon University)

Rohde, D., & Plaut, D. (1999). Language acquisition in the absence of explicit negative evidence: How important is starting small? *Journal of Memory and Language*, *72*, 67–109.

Rumelhart, D., Durbin, R., Golden, R., & Chauvin, Y. (1995). Backpropagation: The basic theory. In *Backpropagation: Theory, architectures, and applications.* Lawrence Erlbaum Associates.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal represenations by error propagation. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel distributed processing, v. 1* (pp. 318–362). MIT Press.

Savitch, W. J. (Ed.). (1987). *The formal complexity of natural language.* Norwell, MA: Kluwer.

Servan-Schreiber, D., Cleeremans, A., & McClelland, J. L. (1991). Graded state machines: The representation of temporal contingencies in simple recurrent networks. *Machine Learning*, *7*, 161–193.

Siegelmann, H. T. (1999). *Neural networks and analog computation: Beyond the turing limit.* Boston: Birkhäuser.

Smith, L. B., Thelen, E., Titzer, R., & McLin, D. (1999). Knowing in the context of action: The task dynamics of the a-not-b error. *Psychological Review*, *106*, 235–260.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, *46*. (Special issue on Connectionist symbol processing edited by G. E. Hinton)

Strogatz, S. (1994). *Nonlinear dynamics and chaos.* Reading, MA: Addison-Wesley.

Tabor, W. (1991). Sentence processing and linguistic structure. In S. Kremer & J. Kolen (Eds.), *Field guide to dynamical recurrent networks.* IEEE Press.

Tabor, W. (1994). *Syntactic innovation: A connectionist model.* (Ph.D. dissertation, Stanford University)

Tabor, W. (1995). Lexical change as nonlinear interpolation. In J. D. Moore & J. F. Lehman (Eds.), *Proceedings of the 17th annual cognitive science conference.* Lawrence Erlbaum Associates.

Tabor, W. (2000). Fractal encoding of context-free grammars in connectionist networks. *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks*, *17*(1), 41-56.

Tabor, W. (2003). Learning exponential state growth languages by hill climbing. *IEEE Transactions on Neural Networks*, *14*(2), 444-446.

Tabor, W. (2005). *Fractal learning neural networks.* (Submitted manuscript, University of Connecticut, Department of Psychology: See http://www.sp.uconn.edu/ ps300vc/papers.html)

Tabor, W., Galantucci, B., & Richardson, D. (2004). Effects of merely local syntactic coherence on sentence processing. *Journal of Memory and Language*, *50*(4), 355-370.

Tabor, W., & Terhesiu, D. (2004). *On the relationship between symbolic and neural computation.* (AAAI Technical Report FS-04-03. ISBN 1-57735-214-9)

Thelen, E., Schoener, G., Scheier, C., & Smith, L. B. (2001). The dynamics of embodiment: a field theory of infant perseverative reaching. *Behavioral and Brain Sciences*, *24*, 1–86.

Thelen, E., & Smith, L. B. (1994). *A dynamic systems approach to the development of cognition and action.* Cambridge, MA: MIT Press.

Thom, R. (1985). *Structural stability and morphogenesis.* Reading, MA: Benjamin.

van der Maas, H. L. J., & Molenaar, P. C. M. (1992). Stagewise cognitive development: An application of catastrophe theory. *Psychological Review*, *99*(3), 395–417.

van Geert, P. (1998). A dynamic systems model of basic developmental mechanisms: Piaget, vygotsky, and beyond. *Psychological Review*, *105*(4), 634–677.

Weckerly, J. (1995). *Object relatives viewed through behavioral, electrophysiological, and modeling techniques.* (Ph.D. dissertation, University of California, San Diego)

Wiles, J., & Elman, J. (1995). Landscapes in recurrent networks. In J. D. Moore & J. F. Lehman (Eds.), *Proceedings of the 17th annual cognitive science conference.* Lawrence Erlbaum Associates.

Author Note

In addition to the very helpful interactions at the conference that prompted this book, I had enlightening discussions with James Dixon, James Magnuson, Jay Rueckl, Aaron Schultz, and Dalia Terhesiu.

Footnotes

[1]Example (1) is said to be a sentence with one level of center-embedding. In particular, it consists of a matrix sentence, "dogs run", with a single subordinate sentence, essentially "girls chase dogs", embedded in the middle of it.

[2]It may not seem plausible that real, physical creatures like human beings have infinite knowledge. However, they have something closely related: the ability to handle arbitrary new cases when they arise. One can understand the theory of infinite string languages as a hypothesis about the nature of this open-ended response capability.

[3]$a^n b^n$ is the language consisting of all strings of a's and b's in which some number of a's is followed by the same number of b's: e.g., ab, aabb, aaabbb, etc.

[4]This work is closely related to the large body of work on variable binding with connectionist networks (e.g., Hinton, 1990; Smolensky, 1990; Jagota, Plate, Shastri, & Sun, 1999; Plate, 1995). Both problems are concerned with the challenge of productivity—how to allow a large number of combinations to be stored in a fixed-width vector; one may think of formal language modelling as a subcase of the variable binding problem. In fact, some variable binding proposals use fractal scaling to allow graceful packing of many patterns into one vector (e.g., Plate, 2003). The work of Moore (1998) and Tabor (2000) distinguishes itself by providing a general formalization of the fractal/grammar relationship, although it does not apply the method to variable binding in general.

[5]The analog of a sphere in two dimensions is a disk, and in one dimension, a line segment. Thus, in the case of a simple Cantor set, these Gaussian units would be activating in response to a particular segment of the real line. In the example network given here, there are two hidden units so the Gaussian third layer units are sensitive to disk-shaped regions in the space of the linear second layer units.

[6]The use of probability distributions rather than individual events as training data is

not consistent with what people actually experience. However, this simplification is superceded by a second one—the use of batch error accumulation in the hill-climbing procedure described below—and was used because it is more efficient.

[7]The KL-Divergence of a probability distribution p_2 with respect to another probability distribution p_1 is equal to $p_2 \cdot log(p_2/p_1)$ (Cover & Thomas, 1991). It is a natural way to compare probability distributions and has a motivated use in backpropagation learning when events correspond to single output units, as they do here, (D. Rumelhart, Durbin, Golden, & Chauvin, 1995), but I do not know, at present, how it compares with other error measures when used with hillclimbing.

[8]Here, by "phase portrait", I mean a diagram of the state space at one stage of training that gives enough information that, with knowledge of the system at hand, one can deduce the dynamics at that stage. Standard phase portraits of continuous systems display a sample of their continuous trajectories. Since the FLNNs are discrete (they jump from point to point) and stochastic (there are multiple futures of most points), plotting trajectories would make the figure very busy. See Tabor (2000) for more detailed trajectory portrayals.

[9]I have not yet formalized this claim, so it remains tentative. I am assuming a standard interpretation of bifurcation: a bifurcation has occurred if there does not exist a homeomorphism between the before and after systems (Guckenheimer & Holmes, 1983; Strogatz, 1994). Intuitively, two systems are homeomorphic if one can be put into one-to-one correspondence with the other by stretching and squeezing. For example, changes in dimensionality require bifurcations, so the transitions a $\rightarrow$ b and b $\rightarrow$ c appear to involve bifurcations.

[10]One might be inclined to say that the model progresses monotonically from simplicity at birth (Figure 3a) to complexity at adulthood (Figure 3e). However, if my speculation above about the super-Turing character of the phase illustrated in Figure 3d is

correct, then that phase, which presumably corresponds to some stage of childhood, is arguably more complex than the adult phase (e).

[11]The network only approximates the symbolic device in two senses: (i) after finite training it has not completely converged; (ii) the soft-max units at the output layer take on activations in the open interval, (0, 1), which does not include 0 and 1. Since these output activations are interpreted as probabilities, the network assigns a positive probability to every possible string. The network approximates symbolic behavior by assigning a relatively high probability to transitions that are deemed legal by the corresponding symbolic system.

[12]Thanks to Denis Mareschal for suggesting this approach to thinking about the model.

Table 1

ABC Grammar. Parentheses denote optional constituents, which occur with probability 0.2 in every case. A probability, indicated by a decimal number to its left, is associated with each production. A sentence is generated by starting with the S node, and iteratively replacing symbols according to the rules (and their probabilities) until all the symbols are lowercase. The resulting lowercase string is, by definition, a member of the ABC language. For example, abc, aabcbc, and abcabc are members of the ABC Language, but bc, aabc, and acb are not. The least powerful type of symbolic device that is capable of stepping sequentially through any string of a's, b's and c's and determining whether it belongs to the ABC Language is the pushdown automaton.

1.0 S → A B C 1.0 A → a (S) 1.0 B → b (S) 1.0 C → c (S)

Table 2

Performance on training and test sets. RMSE = Root Mean Squared Error. % Cor. = Percent Correct. N = the number of networks that contributed to the computation of Standard Error (SE). Npoints = the number of words tested per network. All networks were trained by hill climbing.

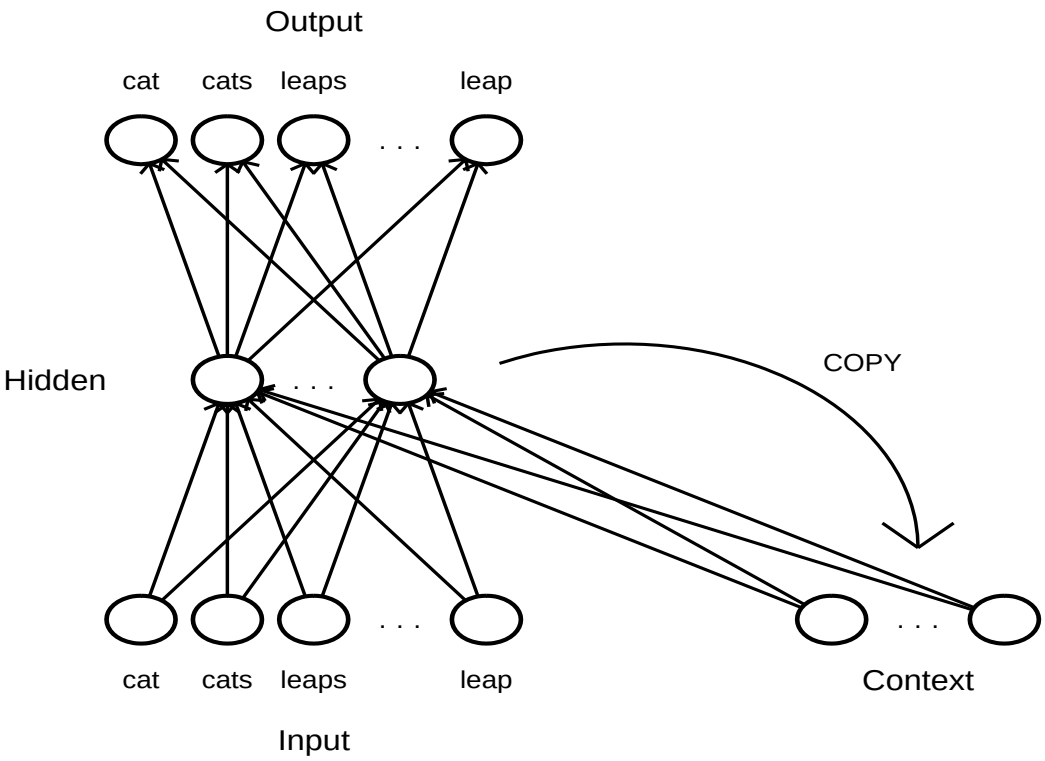
Corpus	RMSE	SE	% Cor.	SE	N	Npoints
Training	0.013	0.001	100.000	0.000	9	129
Test	0.048	0.005	99.424	0.195	9	4755

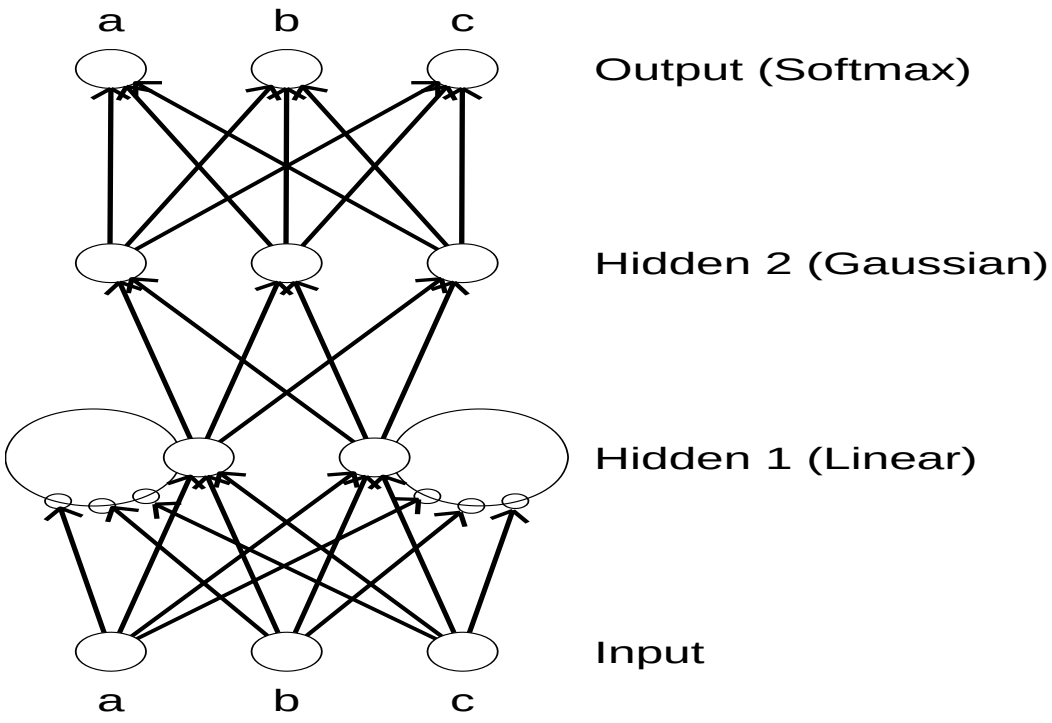
Figure Captions

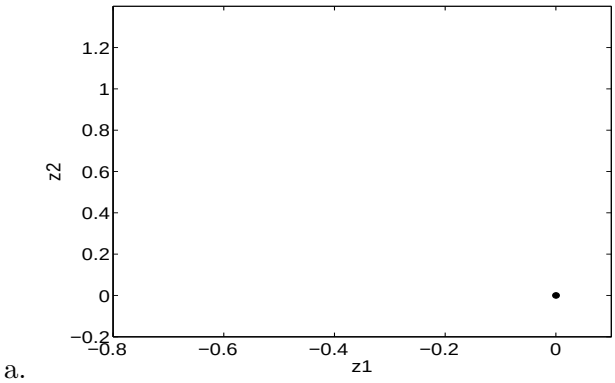
Figure 1. Simple Recurrent Network (Elman, 1990, 1991).

Figure 2. A Fractal Learning Neural Network (FLNN).

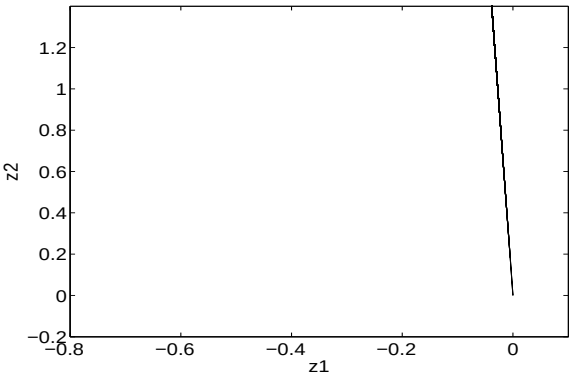
Figure 3. The evolution of the set of visited states in the linear hidden unit layer during processing of grammatical strings over the course of training an FLNN on the language of the grammar in Table 1. z1 and z2 are hidden unit activations. Only states generated by legal sequences of a's and b's are shown—i.e., only states following sequences that increase or maintain the current level of embedding, not any that decrease it. (a) At the beginning of training, the visited states consist of the single point, [0, 0]. (b) At 550 epochs, the visited states lie on an infinite ray with endpoint at [0, 0]. (c) At 1000 epochs, the visited states lie on an infinite lattice (the visited states are at the vertices of triangles marked with horizontal and vertical hatching; the bottommost vertex of the whole lattice is the initial state). There is near-complete overlap between states in which a b (or a) is predicted next (horizontal hatching) and states in which a c (or a) is predicted next (vertical hatching). The network is jumping around on these triangle vertices, going further up in the plane of the figure when a's are encountered, going right at b's and going down at c's. (d) At 10000 epochs, the lattice has become finite and the branches corresponding to b-prediction and c-prediction are spreading apart. (e) At 51174 epochs (the end of training), the branches have fully separated and the set of visited states lies on a totally disconnected fractal in the sense of Barnsley ([1988] 1993). At this stage, the network's behavior is nearly identical to that of the corresponding pushdown automaton. It is not exactly identical because, as mentioned in the text, the expansion associated with the c inputs only approximately inverts the contraction associated with the a and b inputs.


Output
cat
cats
leaps
leap
Hidden
COPY
cat
cats
leaps
leap
Context
Input

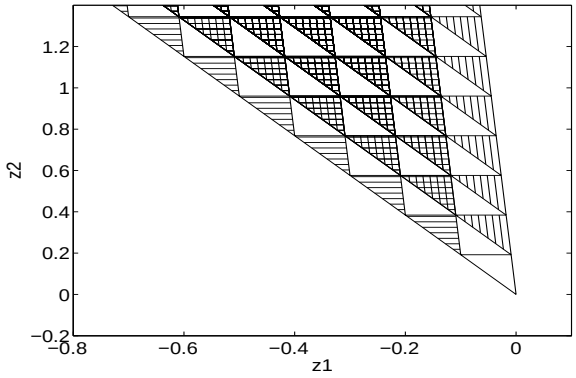

a
b
c
Output (Softmax)
Hidden 2 (Gaussian)
Hidden 1 (Linear)
Input
a
b
c


z2
z1
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.8
−0.6
−0.4
−0.2
0

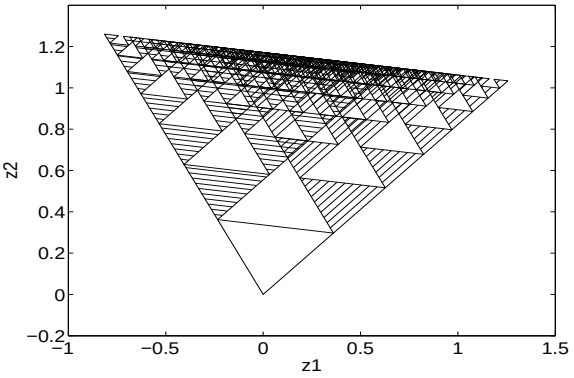
a.


z2
z1
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.8
−0.6
−0.4
−0.2
0

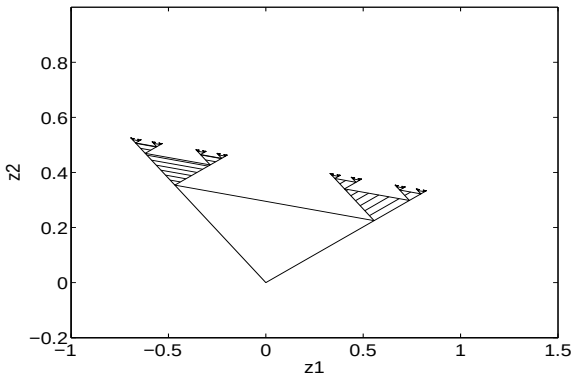
b.


z2
z1
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.8
−0.6
−0.4
−0.2
0

c.


z2
z1
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−1
−0.5
0
0.5
1
1.5

d.


z2
z1
0.8
0.6
0.4
0.2
0
−0.2
−1
−0.5
0
0.5
1
1.5

e.