Language Acquisition via Strange Automata.

Jordan B. Pollack

Laboratory for AI Research &
Computer & Information Science Department
The Ohio State University
2036 Neil Avenue
Columbus, OH 43210
(614) 292-4890
pollack@cis.ohio-state.edu

ABSTRACT

Sequential Cascaded Networks are recurrent higher order connectionist networks which are used, like finite state automata, to recognize languages. Such networks may be viewed as discrete dynamical systems (Dynamical Recognizers) whose states are points inside a multi-dimensional hypercube, whose transitions are defined not by a list of rules, but by a parameterized non-linear function, and whose acceptance decision is defined by a threshold applied to one dimension. Learning proceeds by the adaptation of weight parameters under error-driven feedback from performance on a teacher-supplied set of exemplars. The weights give rise to a landscape where input tokens cause transitions between attractive points or regions, and induction in this framework corresponds to the clustering, splitting and joining of these regions. Usually, the resulting landscape settles into a finite set of attractive regions, and is isomorphic to a classical finite-state automaton. Occasionally, however, the landscape contains a "Strange Attractor" (e.g fig 3g), to which there is no direct analogy in *finite* automata theory.

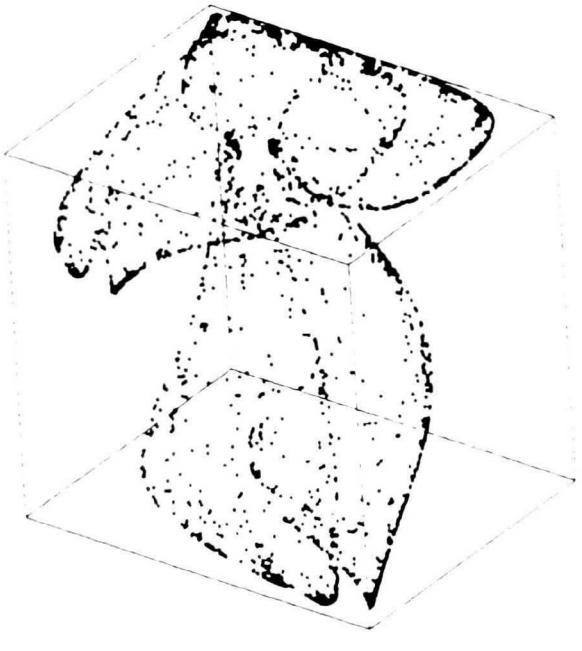

Figure 3g: *Infinite states of a "strange" automaton?*

1. Introduction & Background

Recently, J. Feldman (personal communication) posed the language acquisition problem, as a challenge to connectionist networks. In its most general form, it can be stated quite simply:

Given a language, specified by example, find a machine which can recognize (or generate) that language.

The problem is long-standing and has many specialized variants, especially driven by the goals of of various disciplines. On the one hand mathematical and computational theorists might be concerned with the basic questions and definitions of learning, or with optimal algorithms (Angluin, 1982; Feldman, 1972; Gold, 1967; Rivest & Schapire, 1987). On another hand, linguists may be concerned with how the question of learnability discriminates among grammatical frameworks and specifies necessarily innate properties of mind. On the third hand, psychologists might be concerned, in detail, with how a computational model actually matches up to the empirical data on child language acquisition. Rather than attempting to survey these areas, I point to the excellent theoretical review by (Angluin & Smith, 1983) and the books by (Wexler & Culicover, 1980) and by (MacWhinney, 1987) covering the linguistic and psychological approaches.

In this paper I expose a recurrent high-order back-propagation network to both positive and negative examples of boolean strings, and report that although the network does not find the minimal-description finite state automata for the languages (which is intractable), it does induction in a novel and interesting fashion, and searches through a hypothesis space which, theoretically, is not constrained to machines of finite state. This interpretation is dependent on an analogy among automata, neural networks, and non-linear dynamical systems. The pairwise sub-analogies are, of course, longstanding, as (McCullogh & Pitts, 1943) connected automata to neural nets, (Ashby, 1960) studied nets as dynamical systems, and (Wolfram, 1984) treated (cellular) automata as dynamical systems.

Although we usually think of the transitions among states in a finite-state automata as being fully specified by a table, a transition function can also be specified as a mathematical function of the current state and the input. For example, to get a machine to recognize boolean strings of odd parity, one merely has to specify that the next state is the exclusive-or of the current state and the input. Generalizing from a multilayer networks' ability to perform exclusive-or to the various constructive and existence proofs of the functional/interpolative power of such networks (Hornik et al., To Appear; Lapedes & Farber, 1988; Lippman, 1987), it is pretty obvious that recurrent neural networks can work just like finite state automata, where the transition table is folded up into some moderately complex boolean function of the previous state and current input.

From a different point of view, a recurrent network with a state evolving across k units can be considered a k-dimensional discrete-time dynamical system, with a precise initial condition, $z_k(0)$ and a state space in a bounded subspace of $\mathbf{R}^k$ (i.e., "in-a-box" (Anderson et al., 1977)). The input string, $y_j(t)$, is merely considered "noise" from the environment which may or may not affect the systems evolution, and the governing function, F, is parameterized by weights, W:

$$z_k(t+1) = F_W(z_k(t), y_j(t))$$

If we view one of the dimensions of this system, say z_a as an "acceptance" dimension, we can define the language accepted by such a *Dynamical Recognizer* as all strings of input tokens evolved from the precise initial state for which the accepting dimension of the state is above a certain threshold.

The first question to ask is how can such a dynamical system be constructed, or taught, to accept a particular language? The weights in the network, individually, do not correspond directly to graph transitions or to phrase structure rules. The second question to ask is what sort of generative power can be achieved by such systems?

2. The Model

To begin to answer the question of learning, I now present and elaborate upon my earlier work on Cascaded Networks (Pollack, 1987), which were used in a recurrent fashion to learn parity, depth-limited parenthesis balancing, and to map between word sequences and proposition representations (Pollack, To Appear). A Cascaded Network is a well-controlled higher-order connectionist architecture to which the back-propagation technique of weight adjustment (Rumelhart et al., 1986) can be applied. Basically, it consists of two subnetworks: The *function network* is a standard feed-forward network, with or without hidden layers. However, the weights are dynamically computed by the linear *context network*, whose outputs are mapped in a 1:1 fashion to the weights of the function net. Thus the input pattern to the context network is used to "multiplex" the the function computed, which can result in simpler learning tasks. For example, the famous Exclusive-or function of two inputs can be decomposed into two simpler functions of one input which are selected by the other:

$$XOR(y) = \begin{cases} y & \text{if } x = 0 \\ \neg y & \text{if } x = 1 \end{cases}$$

A simple 1-1 feedforward network (with 2 weights) can implement either of these functions. Identity is $g(3y - 1.5)$, and inversion is $g(1.5 - 3y)$, where $g(z) = 1/1+e^{-z}$, the usual sigmoidal squashing function. The essential idea of cascading is that these solutions in weight space can be a parameterized vector function of x.

Back-propagation is quite straightforward on a cascaded network. After determining the error terms for the weights of the function network, these are used as the error terms for the output units of the context network.

When the outputs of the function network are used as inputs to context network, a system can be built which learns to produce specific outputs for variable-length sequences of inputs. Because of the multiplicative connections, each input is, in effect, processed by a different function.

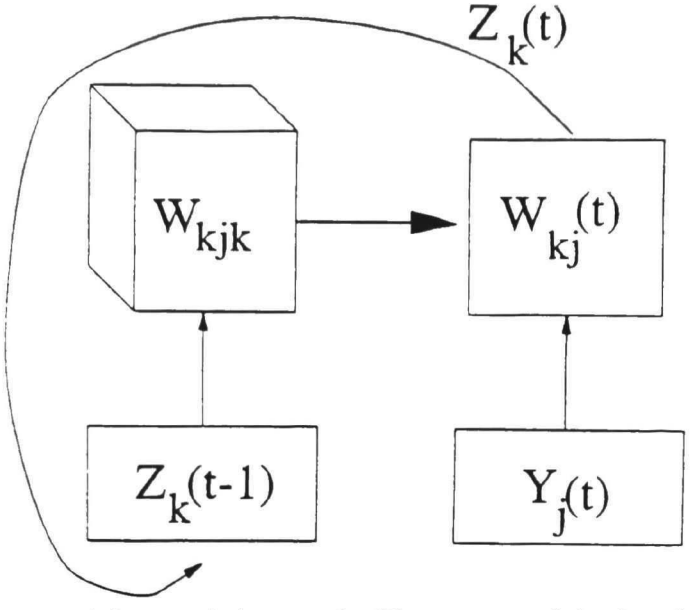

Figure 1. *A sequential cascaded network. The outputs of the function network are used as the next inputs to the context network, yielding a system whose function varies over time.*

Figure 1 shows a block diagram of a simple sequential cascaded network. Given an initial context, $z_k(0)$, and a sequence of inputs, $y_j(t)$, $t = 1...n$, the network computes a sequence of state vectors, $z_k(t)$, $t = 1...n$ by dynamically changing the set of weights, $w_{kj}(t)$:

$$w_{kj}(t) = w_{kjk}\, z_k(t-1)$$

$$z_k(t) = g(w_{kj}(t)\, y_j(t))$$

In previous work, I assumed that the teacher can supply a consistent and generalizable desired state for each member of a large set of strings. Unfortunately, this severely overconstrains the model. In learning a two-state machine like parity, this doesn't matter, as the 1-bit state fully determines the output. Such a teacher would be too powerful in the case of a higher-dimensional system, where we may know what the desired output of a system is but we *don't know* what its internal recurrent state should be.,

Jordan (1986) showed how recurrent back-propagation networks could be trained with "don't care" conditions. If there is no specific preference for the value of an output unit for a particular training example, simply consider the error term for that unit to be 0. This will work, *as long as that same unit receives feedback from other examples*. When the dont-cares line up, the weights to those unit will never change.

The first reaction, fully unrolling a recurrent network by maintaining vector histories (Rumelhart et al., 1986) has not lead to spectacular results (Mozer, 1988), the reason being that very tall networks with equivalence constraints between interdependent layers are unstable. My solution to this dilemma involves a *backspace*, unrolling the loop only once: After propagating the errors determined on only a subset of the weights by the known accept bit, d:

$$\frac{\partial E}{\partial z_a(n)} = (z_a(n) - d)\, z_a(n)\, (1 - z_a(n))$$

$$\frac{\partial E}{\partial w_{aj}(n)} = \frac{\partial E}{\partial z_a(n)}\, y_j(n)$$

$$\frac{\partial E}{\partial w_{ajk}} = \frac{\partial E}{\partial w_{aj}(n)} z_k(n-1)$$

The error on the remainder of the weights (e.g. $i = \{1 \cdots k \mid i \neq a\}$) is calculated by recycling the error on the accept plane with the network reset to its penultimate state:

$$\frac{\partial E}{\partial z_i(n-1)} = \sum_j \frac{\partial E}{\partial w_{ajk}} \frac{\partial}{\partial w_{aj}(n)}$$

$$\frac{\partial E}{\partial w_{ij}(n-1)} = \frac{\partial E}{\partial z_i(n-1)} y_j(n-1)$$

$$\frac{\partial E}{\partial w_{ijk}} = \frac{\partial E}{\partial w_{ij}(n-1)} z_k(n-2)$$

This is done, in batch (epoch) style, for a set of examples of varying lengths.

3. Experiments

Connectionist learning algorithms are very sensitive to the statistical properties of the set of exemplars which make up the learning environment. This has lead some psychological researchers to include the learning environment in the experimental parameters to manipulate (Plunkett & Marchman, 1989). Otherwise, it may not be clear if the results of a connectionist learning architecture are due to itself or due to skill or luck with setting up a collection of testcases. Therefore, I chose to work with test cases from the literature.

Tomita (1982) performed beautiful experiments in inducing finite automata from positive and negative examples. He used a genetically inspired hill-climbing procedure, which manipulated 9-state machines by randomly adding, deleting or moving transitions, or inverting the acceptability of a state, and accepting mutations based on their ability to improve the machine. Tomita ran his system on 7 cases and their complements. Each case was defined by two small sets of boolean strings, accepted by and rejected by the regular languages listed below.

1	1*
2	(1 0)*
3	no odd zero strings after odd 1 strings
4	no triples of zeros
5	pairwise, an even sum of 01's and 10's.
6	number of 1's - number of 0's = 3n
7	0*1*0*1*

For uniformity, I ran all 7 cases on a sequential cascaded network of a 1-input 4-output function network (with bias, 8 weights to set) and a 3-input 8-output context network with bias. The total of 32 weights is essentially arranged as a 4 by 2 by 4 array. Only three of the output dimensions were fed back to the context network, along with a set of biases, and the 4th output unit was used as the acceptance dimension. The standard back-propagation learning rate was set to 0.3 and the momentum to 0.7. All 32 weights were reset to random numbers between ±0.5 for each run. Termination was when all accepted strings returned output bits above 0.8 and rejected strings below 0.2. I changed initial conditions during the period of experimentation, and used an initial state of (.2 .2 .2) for cases 1, 3, and 4, and (.5 .5 .5) for the rest.

3.1. Results

Of Tomita's 7 cases, all but cases #2 and #6 converged without a problem in several hundred epochs. Case 2 would not converge, and kept treating negative case 110101010 as correct; I had to modify the training set (by added reject strings 110 and 11010) in order to overcome this problem. Case 6 took several restarts and thousands of cycles to converge, cause unknown. Presentation of the complete experimental data is in a longer report (Pollack, 1990).

However, none of the minimal-description regular languages were induced by the network. Even for the first language 1*, the network did not create an inescapable error state, so a 0 followed by a long string of 1's would be accepted by the network. If the network is not inducing the smallest consistent FSA, what is it doing?

4. Analysis

In my attempts at understanding the resultant networks, the first approach was to analyze their corresponding finite-state automata. The procedure was very simple. I ran the network as a generator, subjecting it to all possible boolean strings as input, and collecting first, the set of strings for which the acceptance dimension was past threshold, and second, the set of states

(points in 3-space) visited by the machine.

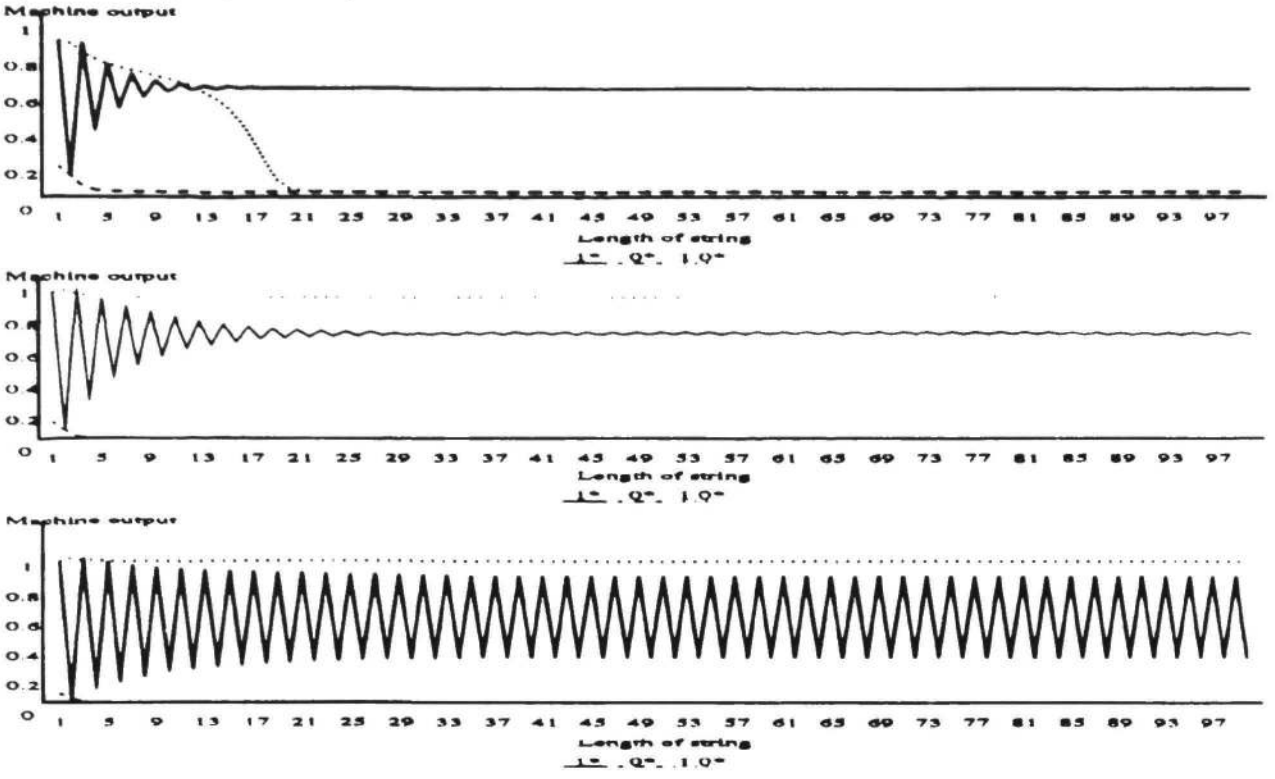

Figure 2. *Three stages in the adaptation of a network learning parity. (a) the test cases are separated, but there is a limit point for 1* at about 0.6. (b) after further training, the even and odd sequences are slightly separated. (c) after a little more training, the oscillating cycle is pronounced.*

Collecting the strings indicated one potential problem with the approach. After training the system can "fuzz out" for longer inputs than the ones given in the test cases. This can be examined for any particular recognizer. We simply observe the limit behavior on the accepting dimension for very long strings. For parity, since the string 1* requires an oscillation of states, we can examine the acceptance dimension as a function of the length. Figure 2 shows three stages in the adaptation of a network for parity. At first, despite success at separating a small training set, a single attractor exists in the limit, so that long strings are indistinguishable. After a little further training, the even and odd strings are separated, and after still further training, the separation is enough to set a threshold easily.

What initially appeared as a bug turns out to indicate a very interesting form of induction. Under feedback pressure to adapt, a slight change in weights leads to a point attractor being "bifurcated" into two. The result, in terms of performance, is significant! Before the split the network only worked correctly on short finite strings; afterwards, it worked on infinite strings.

4.1. Visualizing the Machines

Based upon preliminary studies of the parity example, my initial hypothesis was that a set of clusters would be found, organized in some geometric fashion to be harnessed by the way input causes the state to jump around. Thus, after collecting the state information, it seemed that this would cluster into dense regions which would correspond to states in a FSA. I wrote a diagnostic program to explore this space automatically, by taking an unexplored state and combining it with both 0 and 1 inputs. To remove floating-point fuzz, it had a parameter ε and threw out new states which were within ε euclidean distance from any state already known. Unfortunately, some of the machines seemed to grow exponentially in size as ε was lowered!

One reason for this seems to be that many "ravine" shaped clusters rather than point clusters are developed. Because the states are "in a box" of low dimension, we can view these machines graphically to gain some understanding of how the state space is being arranged. Graphs of the states visited by all possible inputs up to length 10, for the 7 test cases are shown in figure 3. Each figure contains 2048 points, but often they overlap.

The lack of closure under ε can now be seen as a completely different sort of attractor, making my earlier mapping attempt reminiscent of Mandelbrot's (1977, p. 25) essay about measuring the coastline of Britain. The variability in these structures certainly deserves further study, especially with regards to what types of landscapes are possible with different sized networks and alternative activation functions. The images (a) and (d) are what were expected, clumps of points which closely map to states of equivalent FSA's. Images (b) and (e) have simple ravines, which bleed into each other at their ends, probably indicating that longer strings will fuzz out.

Images (c), (f), and (g), are complex and quite unexpected, and will be further discussed below.

5. Related Work

The architecture and learning paradigm I used is closely related to recurrent archtecture devised by (Elman, 1988) and explored by others. Both networks rely on extending Michael Jordan's networks in a direction which separates visible output states from hidden recurrent states, without making the unstable "back-propagation through time" assumption. Besides our choice of language data to model, the two main differences are that

(1) They use a "predictive" paradigm, where error feedback is provided at every time step in the computation, and I used a "classification" paradigm, feeding back only at the end of the given examples.[1]

(2) They use a single layer (quasi-linear) recurrence between states, whereas I use a higher-order (quadratic) recurrence. It is certainly plausible that this quadratic nature allows more "radical" non-linearities to blossom.

Besides continued analysis, scaling the network up beyond binary symbol alphabets, immediate followup work involves comparing and contrasting our respective models with the other two possible models, a higher-order network trained on prediction, and a quasi-linear model trained on classification.

6. Discussion and Conclusion

The state spaces of the dynamical recognizers for Tomita cases 3, 6, and 7, are interesting, because, theoretically, they may be *infinite* state machines, where the states are not arbitrary or random, requiring an infinite table of transitions, but are constrained in a powerful way by some mathematical principle. I believe that it is closely related to related to Barnsley's work on iterated systems, where affine "shrinking" transformations direct an infinite stream of random points onto a underlying fractal or strange attractor.[2] In the recurrent network case, the "shrinking" is accomplished via the sigmoidal function, and the stream of random points are all possible input strings.

Certainly, the link between work in complex dynamical systems and neural networks is well-established both on the neurobiological level (Skarda & Freeman, 1987) and on the mathematical level (Derrida & Meir, 1988; Huberman & Hogg, 1987; Kurten, 1987). It is time that this link be further developed, especially as it applies to the question of the adequacy of connectionist, and other "emergent" approaches to high-level cognitive faculties, such as language (Pollack, 1989). The big question is whether any of the information structures which can be generated by complex dynamical systems can be at all correlated with the structures arising in natural language. Along these lines, (Crutchfield & Young, 1989) have analyzed the computation underlying period-doubling in chaotic dynamical systems and has found power equivalent to indexed context-free grammars.

In conclusion, I have by no means proven that a recurrent dynamical system can act as an efficient recognizer and generator for non-regular languages, though it does seem obvious.[3] But since Dynamical Recognizers are not organized as a PDA's or Turing Machines, it is not clear where the range of languages learnable by these systems would fit inside the Chomsky Hierarchy.

---

[1] Negative information is not crucial to the classification paradigm. Some distinction must be made among strings in order that the state space doesn't collapse into a point. The accept/reject bit is just the smallest such distinction.

[2] My use of the term "attractor" is more related to a stable pattern emerging from deterministic chaos (Lorenz, 1963) than to the traditional use (as energy minima) in optimization models (Ackley et al., 1985; Hopfield & Tank, 1985).

[3] Assuming rational numbers for states, a recurrent multiplicative relationship would be enough to start counting, which is necessary for beginning to handle context-free embeddings, of the sort $a^n b^n$; e.g. consider separate boolean inputs for a and b, and a recurrence $z(t+1) = .5a(t)z(t) + 2b(t)z(t)$. Assuming irrationals in the recurrence relationship, as physicists inadvertently do, and an ideal transcendental sigmoid, the "competance" languages may not even be computable.

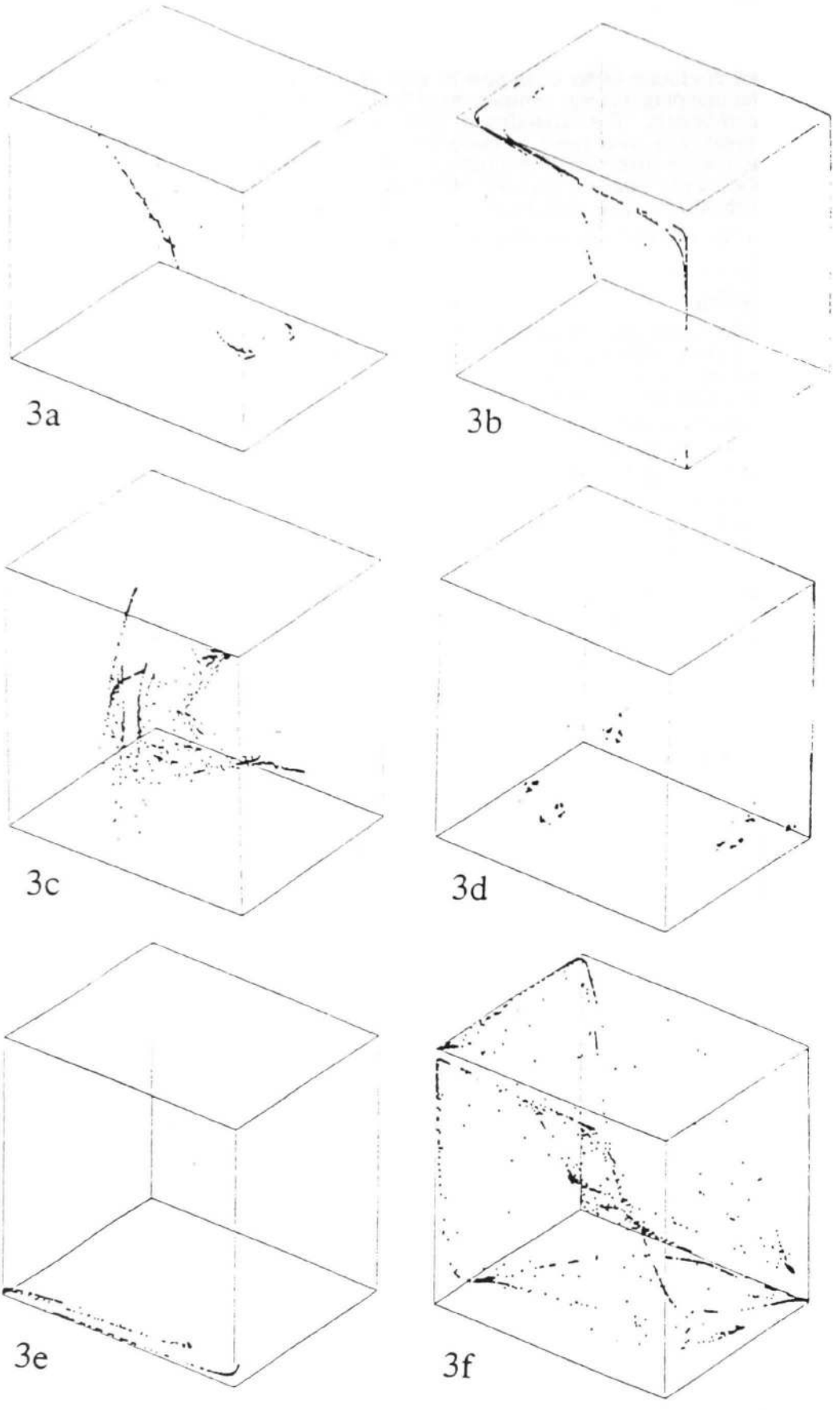

Figure 3. *Images of the attractors for six of the seven Tomita testcases. The points visited by all boolean input strings up to length ten are plotted. The seventh was viewed earlier.*

Nevertheless, we can consider the implications for language (and language acquisition) of a family of automata which smoothly evolve between finite and infinite state machines without massively duplicated transition tables: It will give rise to an induction method which will apply without a priori specification of the grammatical framework of a language in question. Generative capacity is neither natively assumed nor directly manipulated, but is an emergent property of the (fractal) geometry of a bounded non-linear system which arises in response to a specific learning task and is only revealed through performance.

This work is funded by Office of Naval Research Grant N00014-89-J-1200.

7. References

Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann Machines. *Cognitive Science, 9*, 147-169.

Anderson, J. A., Silverstein, J. W., Ritz, S. A. & Jones, R. S. (1977). Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model. *Psychological Review, 84*, 413-451.

Angluin, D. (1982). *Journal of the Association for Computing Machinery, 29*, 741-765.

Angluin, D. & Smith, C. H. (1983). Inductive Inference: Theory and Methods. *Computing Surveys, 15*, 237-269.

Ashby, W. R. (1960). *Design for a Brain: The origin of adaptive behaviour (Second Edition)*. New York: John Wiley & Sons.

Crutchfield, J. P & Young, K. (1989). Computation at the Onset of Chaos. In W. Zurek, (Ed.), *Complexity, Entropy and the Physics of INformation*. Reading, MA: Addison-Wesley.

Derrida, B. & Meir, R. (1988). Chaotic behavior of a layered neural network. *Phys. Rev. A, 38*.

Elman, J. L. (1988). Finding Structure in Time. Report 8801, San Diego: Center for Research in Language, UCSD.

Feldman, J. A. (1972). Some Decidability Results in grammatical Inference. *Information & Control, 20*, 244-462.

Gold, E. M. (1967). Language Identification in the Limit. *Information & Control, 10*, 447-474.

Hopfield, J. J. & Tank, D. W. (1985). 'Neural' computation of decisions in optimization problems. *Biological Cybernetics, 52*, 141-152.

Hornik, K., Stinchcombe, M. & White, H. (To Appear). Multi-layer Feedforward Networks are Universal Approximators. In *Neural Networks*. .

Huberman, B. A. & Hogg, T. (1987). Phase Transitions in Artificial Intelligence Systems. *Artificial Intelligence, 33*, 155-172.

Kurten, K. E. (1987). Phase transitions in quasirandom neural networks. In *Institute of Electrical and Electronics Engineers First International Conference on Neural Networks*. San Diego, II-197-20.

Lapedes, A. S. & Farber, R. M. (1988). How Neural Nets Work. LAUR-88-418: Los Alamos.

Lippman, R. P. (1987). An introduction to computing with neural networks. *Institute of Electrical and Electronics Engineers ASSP Magazine, April*, 4-22.

Lorenz, E. N. (1963). Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences, 20*, 130-141.

MacWhinney, B. (1987). In *Mechanisms of Language Acquisition*. Hillsdale: Lawrence Erlbaum Associates.

Mandelbrot, B. (1982). *The Fractal Geometry of Nature*. San Francisco: Freeman.

McCullogh, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*, 115-133.

Mozer, M. (1988). A focused Back-propagation Algorithm for Temporal Pattern Recognition. CRG-Technical Report-88-3: University of Toronto.

Plunkett, K. & Marchman, V. (1989). Pattern Association in a Back-propagation Network: Implications for Child Language Acquisition. Technical Report 8902, San Diego: UCSD Center for Research in Languge.

Pollack, J. B. (1987). Cascaded Back Propagation on Dynamic Connectionist Networks. In *Proceedings of the Ninth Conference of the Cognitive Science Society*. Seattle, 391-404.

Pollack, J. B. (1989). Implications of Recursive Distributed Representations. In D. Touretzky, (Ed.), *Advances in Neural Information Processing Systems*. Los Gatos, CA: Morgan Kaufman.

Pollack, J. B. (1990). The Induction of Dynamical Recognizers. Tech Report 90-JP-Automata, Columbus, OH 43210: LAIR, Ohio State University.

Pollack, J. B. (To Appear). Recursive Distributed Representation. In *Artificial Intelligence*. .

Rivest, R. L. & Schapire, R. E. (1987). A new approach to unsupervised learning in deterministic environments. In *Proceedings of the Fourth International Workshop on Machine Learning*. Irvine, 364-475.

Rumelhart, D. E., Hinton, G. & Williams, R. (1986). Learning Internal Representations through Error Propagation. In D. E. Rumelhart, J. L. McClelland & the PDP research Group, (Eds.), *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*, Vol. 1. Cambridge: MIT Press.

Skarda, C. A. & Freeman, W. J. (1987). How brains make chaos. *Brain & Behavioral Science, 10*.

Tomita, M. (1982). Dynamic construction of finite-state automata from examples using hill-climbing. In *Proceedings of the Fourth Annual Cognitive Science Conference*. Ann Arbor, MI, 105-108.

Wexler, K. & Culicover, P. W. (1980). *Formal Principles of Language Acquisition*. Cambridge: MIT Press.

Wolfram, S. (1984). Universality and Complexity in Cellular Automata. *Physica, 10D*, 1-35.