Dynamical Assessment of Symbolic Processes with Backprop Nets

Whitney Tabor

University of Connecticut
Department of Psychology
406 Babbidge Road Storrs, CT 06269 USA
tabor@uconnvm.uconn.edu

In Proceedings of the
INNS-IEEE International Joint Conference on Artificial Neural Networks (IJCNN),
Washington, D.C., 2001.

Abstract

Simple Recurrent Networks (SRNs—[1]) were trained to predict the outputs of various probabilistic symbolic processes. Two of the symbolic processes made critical use of a push-down stack, and two were finite state Markov processes. The memory-intensive (stack) processes, in contrast to the Markov processes, pushed the largest Lyapunov exponents toward zero, although they never reached zero. The growth in the Lyapunov exponents was conditioned by the memory-intensiveness of the task, not by the growth rate of the states. The results indicate a link between the traditional use of stack-memories to create complex computation and dynamical treatments of complexity based on trajectory divergence.

1 Introduction

Backpropagation networks have been successfully used for symbolic sequence learning in a number of domains [2] [1] [3] [4] [5]. Some of these models have succeeded in learning approximations to infinite state languages ([6] [7] [8] [9]). A different set of studies has provided insight into what representations can be used to encode complex symbolic languages in artificial neural networks (ANNs) when the weights are hardcoded [10] [11] [12]. But there is a lack of understanding of how learning drives (if it does) the formation of complex representations, and indeed of the nature of the representations that are learned when the task is complex. A natural approach to answering this question is to seek links between symbolic conceptions of complexity and dynamical conceptions, which have received much attention recently in nonlinear systems theory (e.g. [13]). With this goal in mind, I describe several simulation experiments in which neural networks were trained on the outputs of symbolic processes and then analyzed by measurement of Lyapunov exponents.

Lyapunov exponents ([14] [15] [16]) provide a measure of the average rates of exponential growth and contraction of the space surrounding trajectories in a basin of a dynamical system. [10] and [12] found methods of hardwiring neural networks to process context free languages (CFLs) using *fractal grammars.* Fractal Grammars for CFLs maintain a perfect balance between exponential contraction and growth in order to keep track of arbitrary depths of embedding in a bounded metric space. Thus one might expect a Lyapunov exponent of 0 in a neural network trained on a CFL.

This paper seeks 0-valued Lyapunov exponents in SRNs trained on languages involving various kinds of structure. Since neural net symbol generators like the SRN have a stochastic component, it is necessary to specify how the standard definition of Lyapunov exponents can be extended to the relevant kind of stochasticity. Here, I am working in the domain laid out by [17], who has established Lypanov exponents for stochastic systems. But since, I do not fully understand Kifer's arguments for lack of familiarity with all the mathematical technology he employs, I outline a justification of a special case which makes my proposal easier to understand.

Section 2 discusses Lyapunov exponents for SRNs. Section 3 describes four simulation experiments examining the relationship between symbolic stack memory and state divergence. Section 4 concludes.

2 Lyapunov Exponents for SRNs

Lyapunov exponents measure the average rate of divergence of trajectories near an attractor in a dynamical system. Let

$$\vec{x}_{t+1} = f(\vec{x}_t) \tag{1}$$

be a discrete, deterministic dynamical system with n-dimensional state, $\vec{x}$. Following [14], define the Lyapunov Exponents, λ_i for $i = 1, \ldots, n$, of the trajectory starting at $\vec{x}$ as the logarithms of the eigenvalues of the matrix

$$OSL(\vec{x}) = \lim_{t\to\infty} ((Df_t)^T(Df_t))^{\frac{1}{2t}}(\vec{x}) \tag{2}$$

where T denotes transpose, $Df(\vec{x})$ is the Jacobian of f at $\vec{x}$, and

$$Df_t(\vec{x}) = D[f_t \circ f_{t-1} \circ \ldots \circ f(\vec{x})]$$

$$= Df(\vec{x}_t) \cdot Df(\vec{x}_{t-1}) \cdot \ldots \cdot Df(\vec{x}) \tag{3}$$

For $\vec{x}$ in the basin of a single attractor, the values of the eigenvalues are essentially independent of the choice of $\vec{x}$ so we may speak of the Lyapunov exponents of the attractor. From another perspective, the ith eigenvalue measures the average rate of growth of the ith principle axis of infinitismal ellipses surrounding points on the attractor [18]. The sum of the Liapunov exponents indicates the global stability of the system: the sum must be negative for the system to have Lyapunov stability. If all the exponents are negative, then the system is a limit cycle and visits only finitely many points. If at least one exponent is positive (and the sum is negative), then the system is chaotic. The case in which the greatest Lyapunov exponent is 0 in a discrete system is a special case which can yield complex behavior (famously for the logistic map at the "edge of chaos"—[13]).

Here, I explore the use of Simple Recurrent Networks (SRNs—[2] [1]), a class of neural networks that can be trained to predict symbolic symbol sequences. The SRNs I considered had the form

$$\vec{o}_t = normexp(\mathbf{W}_{ho} \cdot \vec{h}_t) \tag{4}$$

$$\vec{h}_t = sigmoid(\mathbf{W}_{ih} \cdot \vec{i}_t + \mathbf{W}_{hh} \cdot \vec{h}_{t-1}) \tag{5}$$

where *sigmoid* refers to the standard sigmoid function, *normexp* refers to the normalized exponential (softmax) function, $\vec{i}$ refers to the input layer, $\vec{h}$ to the hidden layer, $\vec{o}$ to the output layer, $\mathbf{W}_{ab}$ is the weight matrix connecting layer a to layer b, and subscript t is a time index. The networks were trained using backpropagation [19, 20] assuming a multinomial cost function ($E_D = \log \Pi_{j\in Outputs} o_j^{t_j}$, where t_j is the target for unit i). The recurrent weights are trained using discrete BPTT, but as in Elman's studies, the gradient was truncated after a single timestep (called BPTT(1) by [21]).

The network's outputs define a probability distribution. Thus, the network can function as an autonomous stochastic dynamical system by sampling the output distribution at time t to pick an input for time $t + 1$.

Let $\vec{x}$ be an initial point for the network and let S be an infinite sequence of inputs to h generated under h starting from $\vec{x}$. Consider the contingent value of the Oseledec matrix given by

Def. $OSL(\vec{x}, S) = \lim_{t\to\infty}((Df_t)^T(Df_t))^{\frac{1}{2t}}(\vec{x})$ via the processing of the recurrently generated input sequence S.

This will be a useful definition if the eigenvalues of $OSL(\vec{x}, S)$ are reliably independent of the choice of $\vec{x}$ and of the randomness in the process that creates S. For initial states, $\vec{x}$ and $\vec{y}$, consider the case where the trajectories issuing from $\vec{x}$ (called t_x) and from $\vec{y}$ (called t_y) visit finitely many points ergodically. If these trajectories coincide on one of the ergodic points, then they have converged to the same probabilistic finite state process and we say the distance between them is zero in the limit. By the theory of Markov processes, the transition probabilities into and out of each state converge to stable values. Thus for each ergodic state, s and history length k there will be a number M, such that after M transitions have been made, the distribution of length k prehistories of s in t_x will be arbitrarily close to the distribution of length k prehistories of s in t_y. Consider an iterative QR decomposition of t_x and t_y, each into an orthogonal matrix, Q_m multiplied by a series of upper triangular matrices, R ([22]):

$$t_x = Df_M Df_{M-1} \dots Df_1$$

$$= Q_M R_M R_{M-1} \dots R_1 \quad (6)$$

and correspondingly for t_y. The jth Liapunov Exponent is given by

$$E_j = \frac{1}{m} \sum_{i=1}^{M} ln|R_i(j,j)| \quad (7)$$

Since the sequences of Dfs have nearly identical distributions over length k prehistories of s in t_x and t_y, substantial differences in the $R_{j,j}$ at s can only be due to events further away from s than k. But these have negligible influence on average as k grows large. Therefore, the sums converge to the same values in t_x and t_y.

A similar argument can be made when the trajectories ergodically visit a countable infinity of states with convergent probabilities. See [17] for a more general treatment.

These arguments indicate that the definition of Lyapunov exponents given above is robust in the sense that if we can find basins of points which lead to the same ergodic trajectory, then the Lyapunov exponents will be associated with the entire basin. In particular, two trajectories issuing from the same point will have the same Lyaponov exponents.

Consider the context-free language defined by the rules in Table 1.

[12] defines *Dynamical Automata* which recognize context free languages by deploying states on a fractal. For example DA 0, a dynamical automaton for recognizing the language of Grammar 0 is specified by the Input Map shown in Table 2. The essence of the DA is a two-element vector, **z**, corresponding to a position on the Sierpinski triangle ([23]). The DA functions as follows: when **z** is in the subset of the plane specified in the "Compartment" column, the possible inputs are those shown in the "Input" column. Given a compartment and a legal input for that compartment, the change in **z** that results from reading the input is shown in the "State Change" column. If we specify that the DA must start with **z** = (1/2, 1/2), make state changes according to the rules in Table 2 as symbols are read from an input string, and return to **z** = (1/2, 1/2) (the Final Region) when the last symbol is read, then the computer functions as a recognizer for the language of Grammar 0. To see this intuitively, note that any subsequence of the form "a b c d" invokes the identity map on **z**. Thus DA 0 is equivalent to the nested finite-state machine version of Grammar 0.

Table 1: Grammar 0.

Rule 1a.	S $\rightarrow$ A B C D
Rule 1b.	S $\rightarrow \epsilon$
Rule 2a.	A $\rightarrow$ a S
Rule 2b.	A $\rightarrow$ a
Rule 3a.	B $\rightarrow$ b S
Rule 3b.	B $\rightarrow$ b
Rule 4a.	C $\rightarrow$ c S
Rule 4b.	C $\rightarrow$ c
Rule 5a.	D $\rightarrow$ d S
Rule 5b.	D $\rightarrow$ d

Table 2: Dynamical Automaton 0.

Compartment	Input	State Change
$z1 > 1/2$ and $z2 < 1/2$	b	**z** ← **z** - (1/2, 0)
$z1 < 1/2$ and $z2 < 1/2$	c	**z** ← **z** + (0, 1/2)
$z1 < 1/2$ and $z2 > 1/2$	d	**z** ← 2(**z** - (0, 1/2))
Any	a	**z** ← 1/2 **z** + (1/2, 0)

Using the definition above, we can derive the Lyapunov exponents for an example like DA 0. If DA 0 starts at $\vec{x} = (0,0)$, then every trajectory produces a string with an equal number of a's, b's, c's, and d's. The Jacobian for the b and c transitions is I, the identity matrix. For the a transitions, it is diag(1/2, 1/2) and for the d transitions, it is diag(2, 2). Therefore, the Oseledec matrix is I in the limit and both Lyapunov exponents are 0.

The next section uses Lyapunov analysis to ask whether backpropagation networks for CFLs are employing a similar mechanism.

3 Finite-state vs. context free languages

I hypothesized the following for Simple Recurrent Networks (SRNs) trained using backpropagation:

> Training on finite state languages will produce limit cycle dynamics, hence negative Lyapunov exponents.
>
> Training on context free languages will produce "edge of chaos" dynamics, hence maximum Lyapunov exponent 0.

I tested these hypotheses on four cases: (1) a deterministic Markov language with only three states, (2) a context free language with an exponential relationship between state frequency and state frequency rank (3) a context free language with a hyperbolic (Zipf's Law) relationship between state frequency and state frequency rank, (4) a probabilistic Markov language with a Zipf's Law distribution (Table 3). By a "Markov language" I mean a language in which the current state-transition probabilities are a function of the most recently-read symbol.

Each network had 4 hidden units and thus 16 corners of the hidden unit hypercube. In general, backpropagation networks with sigmoidal hidden units tend to drive their hidden unit representations to the corners of this hypercube. Each network was trained to the point where it had learned 16 or fewer states well (a network was taken to have learned a state well if the predicted transition probability distribution out of its approximation to that state was closer to the correct distribution than to any distinct target distribution). Thus, each network had ample room to create spherical state deployments and was not forced to induce variation in contractivity by the structure of the representation space.

A priori, one might expect the Zipf's Law frequency-rank relationship (of Languages 3 and 4) to spur fractal organization of the network's state space and thus lead to dynamics with exponential separation of trajectories. Thus Language 4 (Zipf's Law, Markov) serves as a control to test whether state frequency distribution or stack-memory-intensiveness has more influence on the structure of the learned trajectories.

Lyapunov exponents were calculated using the Householder QR-based (HQRB) method described in [22].

Table 3: Symbol generating grammars. A constituent in parentheses is generated in half the instances and absent in the other half. Two constituents with a slash (/) between split the instances half and half as well.

Grammar No.	Definition
(1)	**S $\rightarrow$ a b c**
(2)	**S $\rightarrow$ S1 p** **S1 $\rightarrow$ a (S1) b**
(3)	**S1 $\rightarrow$ S11/S12 p** **S11 $\rightarrow$ a (S1) b** **S12 $\rightarrow$ x (S1) y**
(4)	**S $\rightarrow$ a P1/P2** **P1 $\rightarrow$ (a1 P11)/(a1 P12)** **P2 $\rightarrow$ (a2 P21)/(a2 P22)** **P11 $\rightarrow$ (a11 P111)/(a11 P112)** **P12 $\rightarrow$ (a12 P121)/(a12 P122)** **P21 $\rightarrow$ (a21 P211)/(a21 P212)** **P22 $\rightarrow$ (a22 P211)/(a22 P212)** **P111 $\rightarrow$ a111 a111e** **P112 $\rightarrow$ a112 a112e** **P121 $\rightarrow$ a121 a121e** **P122 $\rightarrow$ a122 a122e** **P211 $\rightarrow$ a211 a211e** **P212 $\rightarrow$ a212 a212e** **P221 $\rightarrow$ a221 a221e** **P222 $\rightarrow$ a222 a222e**

Table 4: Lyapunov exponents for the grammar-trained networks and an untrained network ("None") with weights randomly drawn from a uniform distribution with mean 0 and radius 0. Mean maximal exponent estimates and their standard deviations are shown for 10 tests each from the same initial point are shown.

Grammar	Zipf's?	Stack?	Max. Exp	S.D.
(1)	No	No	-2.113	0.001
(2)	No	Yes	-0.156	0.008
(3)	Yes	Yes	-0.235	0.003
(4)	Yes	No	-2.108	0.035
None	—	—	-1.718	0.0004

3.1 Results

As shown in Table 3, maximal Lyapunov exponents close to 0 seem to be uniquely associated with those languages whose generation depends on a stack memory. Moreover, within those languages that were generated by stack memories, the maximal Lyapunov exponent increased in value over the course of training, approaching 0 approximately, though the curves were not very smooth.

4 Conclusions

There may be at least a partial correspondence between the Chomsky hierarchy of formal languages (finite state, context free, context sensitive, Turing language) and the range of dynamical regimes identified by nonlinear systems theory. Finite state languages may correspond to limit cycles and context free languages may correspond to edge-of-chaos phenomena with Lyapunov exponents $= 0$. The fact that a Turning machine can be built with three stacks suggests that all recursively enumerable computations may lie at the boundary of chaos.

Although there seem to be parallels between symbolic and dynamical characterizations of complexity, the relationships between different languages are portrayed differently within the two regimes. For example, within the same Dyanmical Automaton, the context free languages may appear densely intermixed with the non-context free languages and bear metric relations to them ([12]).

The growth in Lyapunov exponents observed here during the training of networks for processing stack-based languages suggests a bifurcation route to the development of complex memory. Mapping out this bifurcation route may assist with some of the challenging problems facing the learning of structured data by neural networks.

5 Acknowledgments

Thanks to Bruno Galantucci for helpful discussions. Thanks to Chaopeng Zhou for help running the simulations.

References

[1] J. L. Elman, Machine Learning **7**, 195 (1991).

[2] J. L. Elman, Cognitive Science **14**, 179 (1990).

[3] D. Servan-Schreiber, A. Cleeremans, and J. L. McClelland, Machine Learning **7**, 161 (1991).

[4] J. B. Pollack, Machine Learning **7**, 227 (1991).

[5] T. A. Plate, IEEE Transactions on Neural Networks **6**, 623 (1995).

[6] M. C. Mozer and S. Das, A connectionist symbol manipulator that discovers the structure of context-free languages, in *Advances in Neural Information Processing Systems 5*, edited by S. J. Hanson, J. D. Cowan, and C. L. Giles, pages 863–70, Morgan Kaufmann, San Mateo, CA, 1993.

[7] P. Rodriguez, J. Wiles, and J. Elman, Connection Science **11**, 5 (1999).

[8] G. Z. Sun, H. H. Chen, C. L. Giles, Y. C. Lee, and D. Chen, Connectionist pushdown automata that learn context-free grammars, in *Proceedings of the International Joint Conference on Neural Networks*, edited by M. Caudill, pages 577–580, Lawrence Earlbaum, Hillsdale, NJ, 1990.

[9] S. Levy, O. Melnik, and J. Pollack, Infinite raam: A principled conectionist basis for grammatical competence, in *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*, pages 298–303, Lawrence Erlbaum Associates, Mahwah, NJ, 2000.

[10] C. Moore, Theoretical Computer Science **201**, 99 (1998).

[11] M. Steijvers and P. Grünwald, A recurrent network that performs a context-sensitive prediction task, in *Proceedings of the 18th Annual Cognitive Science Conference*, Lawrence Erlbaum Associates, 1996.

[12] W. Tabor, Expert Systems: The International Journal of Knowledge Engineering and Neural Networks **17**, 41 (2000).

[13] J. P. Crutchfield, Physica D **75**, 11 (1994), In the special issue on the Proceedings of the Oji International Seminar, *Complex Systems—from Complex Dynamics to Artificial Reality.*

[14] V. I. Oseledec, Trudy Mosk. Mat. Obsc. **19**, 197 (1968).

[15] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer-Verlag, New York, 1983.

[16] H. D. I. Abarbanel, *Analysis of Observed Chaotic Data*, Springer-Verlag, New York, 1996.

[17] Y. Kifer, *Ergodic Theoory of Random Transformations*, Birkhäuser, Boston, 1986.

[18] A. Wolf, J. Swift, H. Swinney, and J. Vastano, Physica D **16**, 285 (1985).

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning internal represenations by error propagation, in *Parallel Distributed Processing, v. 1*, edited by D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, pages 318–362, MIT Press, 1986.

[20] D. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, Backpropagation: The basic theory, in *Backpropagation: Theory, Architectures, and Applications*, Lawrence Erlbaum Associates, 1995.

[21] R. J. Williams and J. Peng, Neural Computation **2**, 490 (1990).

[22] H. F. von Bremen, F. E. Udwadia, and W. Proskurowski, Physica D **101**, 1 (1997).

[23] M. Barnsley, *Fractals Everywhere, 2nd ed.*, Academic Press, Boston, [1988]1993.