Complex Systems **5** (1991) 359–370

Chaos-based Learning

Paul F. M. J. Verschure*
Institute for Informatics, University of Zurich,
Winterthurerstraße 190, CH-8057 Zurich, Switzerland

Abstract. It is demonstrated that the chaotic properties of neural networks (in this case networks defined by the generalized delta procedure) can be used to improve their learning performance. By adaptively varying the learning-rate parameter an annealing mechanism can be introduced that is founded in chaos. The proposed mechanism, chaos-based learning, provides faster convergence than standard back-propagation and also seems to provide a computationally less intensive alternative to other back-propagation accelerating techniques by using adaptive step-size control.

Introduction

In the study of neural networks most attention is directed to their performance given stable behavior. The chaotic properties of neural networks, however, have become well established [5, 6, 13, 14, 17]. In our own work we have concentrated on the chaotic behavior of standard neural networks [19, 20]. We have shown that for these network types the learning parameter η could be used to introduce chaotic behavior independent of network size.

From an engineering point of view chaos in neural networks seems inefficient. This implies that it should be suppressed. I would like to illustrate, however, that chaos can provide dynamical systems like neural networks with an inherent fast-search mechanism. As an example the chaotic properties of error back-propagation networks will be used to improve their performance. The generalized delta procedure (GDP) will be extended with an annealing-like mechanism that is controlled by the learning-rate parameter η. In this learning scheme, chaos-based learning (CBL), annealing will take place from a chaotic phase to a stable phase.

Before elaborating on CBL, the chaotic properties of GDP will be described briefly.

*Electronic mail address: verschur@ifi.unizh.ch

The generalized delta procedure

GDP [8, 12, 15, 21] describes the interaction of multilayered neural networks and is a connectionist implementation of the classical optimization method of gradient descent. GDP networks can be trained to learn to associate a set of input patterns ξ^μ ($\mu = 1, \ldots, P$) with a set of output patterns that are a transform of these input patterns $f(\xi^\mu)$. This mapping is established by minimizing the averaged error $Q^\mu(\mathbf{w})$ between the expected output, $f(\xi^\mu)$, and the output calculated by the network, $g(\xi^\mu, \mathbf{w})$, which is a function of the input pattern presented, ξ^μ, and the weights and biases in the system, $\mathbf{w}$. The error function $Q(\mathbf{w})$ can be expressed as:

$$Q(\mathbf{w}) = \frac{1}{P} \sum_{\mu} [f(\xi^\mu) - g(\xi^\mu, \mathbf{w})]^2 \tag{1}$$

To implement gradient descent the weights are updated according to

$$\Delta \mathbf{w} = -\eta \frac{\partial Q^\mu}{\partial \mathbf{w}} \tag{2}$$

in which η determines the learning rate.

The error surface on which gradient descent takes place is defined by (1) and has as many dimensions as there are weights and biases in the system. The system is supposed to make steps along the error surface until Q is minimal and a representation of the input-output mapping has been found. The trajectory made is determined by the mapping to be learned, the presentation strategy, and the step sizes. These step sizes are determined by Q, $\mathbf{w}$, and the learning rate parameter η (equation (2)).

The error surface defined above has three general properties. First, it can have very little slope, which implies that Q will only decrease after a considerable number of steps. Second, the error surface of GDP contains local minima that can trap the system in a non-optimal solution. Third, the error surface contains many global minima.

In every simulation experiment, however, one cannot perform strict gradient descent, which would require infinitesimally small step sizes. For every finite step size the dynamics in weight space can at the most be an approximation of gradient descent; the accuracy of this depends critically on the curvature of the error surface. Therefore, for finite step sizes, local minima in which the system will *not* be trapped can be postulated.

Chaos

In Verschure et al. [20] the chaotic properties of GDP are demonstrated by means of bifurcation diagrams, pseudo-phase plots, Lyapunov exponents, and power spectra. Here, as an illustration of these chaotic properties, a small network consisting of 2 input units, 3 hidden units, and 1 output unit will be submitted to the classical exclusive-or task for increasing values of η. As a measure of the behavior of the system the error function $-Q$ is used.

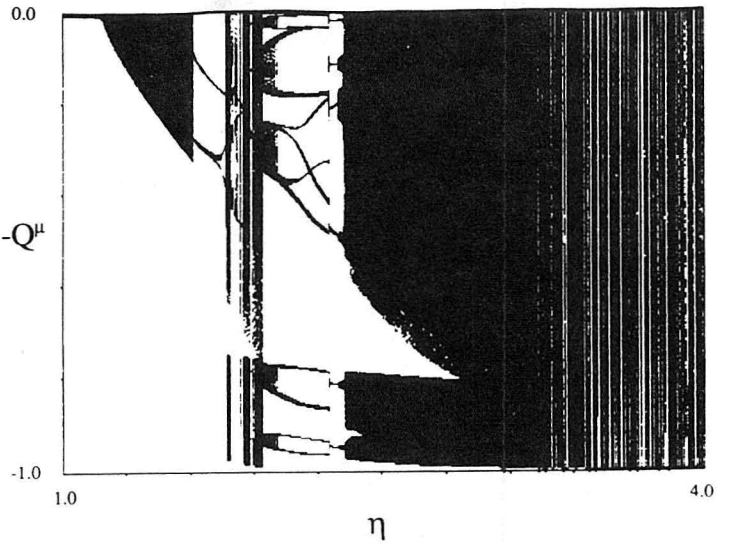

Figure 1: Bifurcation diagram of GDP performing the exclusive-or. $\eta = 1.0$ to 4.0.

In figure 1 the development of $-Q$ over the range $\eta = 1.0$ to 4.0 is shown. For every value of η, 3000 learning cycles were performed in which the four patterns constituting the exclusive-or were presented. Before a learning sequence commenced, the weights and biases were initialized at the same values, which were chosen randomly. The values of $-Q$ are only depicted from cycle 1000 to 3000 to assure that only stationary behavior is shown. In this simulation the patterns are presented periodically and the weights are updated after every pattern presentation.

For η in the range 1.0 to 1.25, the development of $-Q$ is stable; the system has converged to a stable solution. After $\eta = 1.25$ the system ends up in a chaotic regime. Between $\eta = 1.6$ and $\eta = 2.3$ some bifurcations are clearly visible.

The control parameter η

The speed of convergence of GDP is strongly dependent on the properties of the error surface along which gradient descent takes place. When the error surface is practically flat or the direction of the negative gradient vector—which directs the update of the parameters—does not point toward the global minimum of the surface, the rate of convergence will be slow. On the other hand, when the error surface is strongly curved the step size can become too large, which can lead to an overshoot. To find an optimal adjustment of the step size, adaptive tuning of the learning rate parameter η dependent on the properties of the error surface seems appropriate [3, 18].

A number of heuristics for adapting η have already been proposed (see [2] for an overview). In defining these heuristics, one is confronted with a dilemma. High values of η lead to a quick approximation of convergence and seem appropriate when the error surface is relatively flat. Unfortunately, high values of η also imply that the system will probably not settle into the detected minimum, but will overshoot. Low values of η imply that the system could take a very long time to reach a minimum. Also, in this case there is no guarantee the system will remain at this solution. In choosing an optimal step size, we seem to be confronted with a speed-stability trade-off.

In dealing with the speed-stability trade-off two strategies seem appropriate. The first tries to specify the relation between the properties of the task domain and optimal initial conditions. A second approach would rely on equipping the system with internal mechanisms that would allow it to adapt to task demands. The present proposal is an example of the second strategy.

Chaos-based learning

In CBL learning takes place in three phases. In the first or stabilizing phase a semi-stable state is established using a predetermined value of η. In the second or chaos detecting phase the system searches for the chaotic domain of the system by adaptively increasing η. When the error function becomes very variable chaos is inferred, and the third or cooling down phase is initiated. In this last phase, the system cools down from a chaotic domain to a stable domain with η as the control parameter. In all phases learning takes place according to GDP after every pattern presentation.

Phase 1: Stabilizing

During a number of learning cycles, the pattern mapping is learned with a predetermined value of η until the error function becomes stable. In the first few training cycles the error function can go through rapid changes. After these rapid changes the dynamics mostly settle into a relatively stable state, which is related to a low slope area of the error function mentioned earlier. Since chaos is expressed in strongly varying values of the error function Q, the system could erroneously classify these early changes as chaotic behavior. Therefore, after every presentation cycle c (in which all P pattern combinations are presented) the ratio of the error function at c and $c-1$ is determined. If this ratio does not significantly differ from 1, stability is reached and the next phase is entered.

Phase 2: Chaos detection

Learning proceeds by adding a predetermined step size $\Delta\eta$ ($\Delta\eta < \eta$) to the initial value of η. For a predetermined number R of presentation cycles, the learning task is performed with this new value of η. The subsequent series of R presentation cycles with increasing values of η are called *search cycles.*

Within every search cycle, η is held constant. Between subsequent search cycles, η increases.

Within every search cycle the average of the error function Q is determined after every completed presentation cycle c. After the last presentation cycle of a search cycle ($c = R$) the ratios between the last value of Q and all earlier values of Q within the search cycle are determined. If all the ratios significantly differ from 1.0, and Q is not steadily decreasing, chaos is inferred and cooling down can begin. If one or more of these ratios, however, does not differ significantly from 1.0, η is increased and the next search cycle is initiated.

The increments of η in this chaos detection phase are dependent on the variability of Q in the search cycle. After one search cycle is terminated the standard deviation σ_Q of Q over the previous R presentation cycles is determined:

$$\sigma_Q = \sqrt{\frac{\sum_{c=1}^{R}(Q_c - \overline{Q})^2}{R}} \tag{3}$$

The new value of the step size $\Delta\eta$ is determined by the product of the ratio of the value of the standard deviation of Q for the previous search cycle, the present search cycle, and the difference between the associated values of η. When we index the search cycles with s, this product can be expressed as

$$\Delta\eta = \left(\frac{\sigma_Q^{s-1}}{\sigma_Q^{s}}\right)(\eta^s - \eta^{s-1}) \tag{4}$$

This definition of $\Delta\eta$ assures that, when the variance of Q is low, which probably indicates a flat error surface, the step sizes will increase markedly. In case variance is high, the step sizes are decreased to allow a fine-tuned chaos detection.

Phase 3: Cooling down

The previous chaos detection phase increases η until chaotic behavior is inferred. This last value of η constitutes the upper limit of the control parameter η in the cooling down phase. The step size $\Delta\eta$ in this phase is the average value of all earlier step sizes. This average step size also defines the lower bound of η in cooling down.

Changes in η are determined using a Metropolis-type algorithm [11]. The probability P_c of accepting a given value of η after presentation cycle c given the performance of the system, expressed in Q_c, is determined by $P_c = 1 - Q_c$. If P_c exceeds a randomly chosen criterion that is between 0 and 1, η is decreased by $P_c\Delta\eta$. Otherwise, η increases by $(1 - P_c)\Delta\eta$.

The upper and lower bounds of η function as reflecting barriers. If the upper bound is reached, η is set at $0.75(\eta - \Delta\eta)$. If the lower bound is reached, η is set at $0.5(\eta - \Delta\eta)$. The constants 0.75 and 0.5 are determined on the basis of simulation studies. The lower reflecting bound counteracts

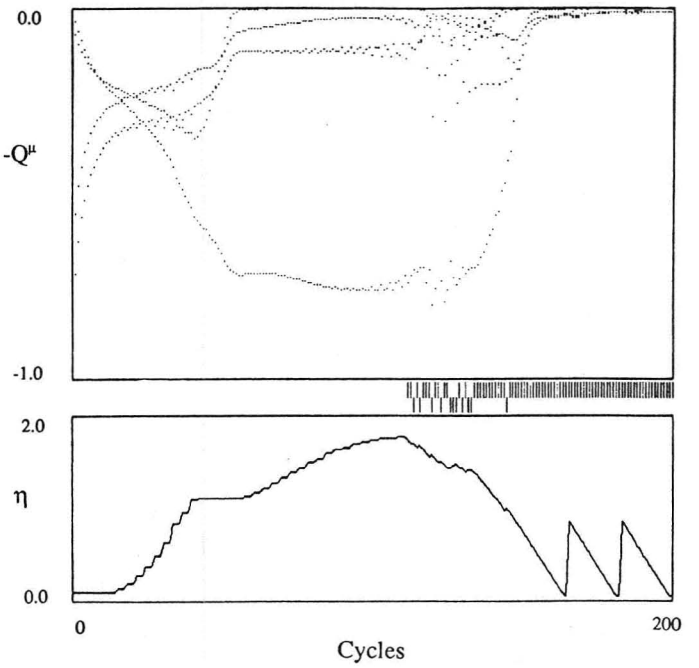

Figure 2: (a) CBL performing the exclusive-or: the first 200 cycles. The upper panel displays the development of Q. The lower panel represents the development of η. The vertical lines between the two panels represent the annealing schedule. η and $\Delta\eta$ were initialized at 0.1 and 0.001. respectively. Chaos was detected at $\eta = 1.7749$. The step size and the lower bound in the cooling down phase were set at 0.0508. Final $Q = 0.00409566$. After 1000 cycles Q had decreased to 0.00048217.

stages of slow convergence. If only η decreases then there is a large probability that η will take on a minimal value quickly. while convergence is not reached or approached. In this situation larger values of η would be more appropriate. The upper reflecting barrier forestalls that η exceeds the chaotic domain found earlier. which would make the system jump out of the basin of attraction of the minimum it is approaching.

Simulation results

As an illustration of the properties of CBL. a 2-3-1 network was submitted to the classical exclusive-or task with the same initial values used to obtain figure 1. In these simulations the patterns are again presented periodically.

In figures 2(a) and 2(b) the results of the experiment are shown for CBL and GDP. respectively. The upper panel of figure 2(a) displays the develop-

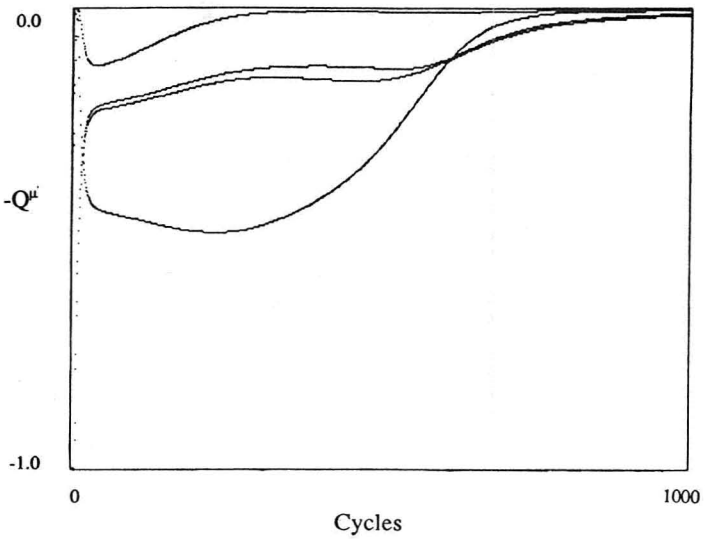

Figure 2: (b) GDP learning the exclusive-or: the first 1000 cycles. $\eta = 0.1$. Final Q after 1000 cycles is 0.0057026.

ment of $-Q^{\mu}$ during 200 presentation cycles. The lower panel of the figure represents the development of η. The vertical lines between the two panels represent the cooling down schedule. When the line points upward, P_c exceeded the randomly chosen criterion and η was lowered. When the line points downward, η was increased. The point along the axis representing the cycles where the annealing sequence commences indicates the start of the cooling down phase.

In this experiment, η was initialized at 0.1 with a first step size of 0.001. Chaos was detected after 112 cycles at $\eta = 1.7749$. The cooling down step size and lower bound was set at 0.0508. The final value of Q (averaged over the four patterns) after 200 cycles was 0.00409566. After 1000 cycles Q had further decreased to 0.00048217.

The relation between the cost function Q and η in determining the step sizes (equation (4)) in the chaos detection phase is clearly demonstrated. If Q stabilizes, $\Delta\eta$ rapidly increases in search of chaos; otherwise $\Delta\eta$ decreases in order to be able to come to a precise determination of the chaotic level.

Figure 2(b), which is constructed in the same way as the upper panel of figure 2(a), displays the behavior of GDP given identical initial conditions and network architecture. GDP gave a final Q of 0.0057026 after 1000 cycles for $\eta = 0.1$. When comparing figures 2(a) and 2(b) it is evident that CBL adjusts η very rapidly in such a way as to counteract the slow convergence phase that is expressed in the behavior of GDP.

To demonstrate that CBL has properly detected the chaotic domain of the dynamics, the pseudo-phase plot related to the time series generated with

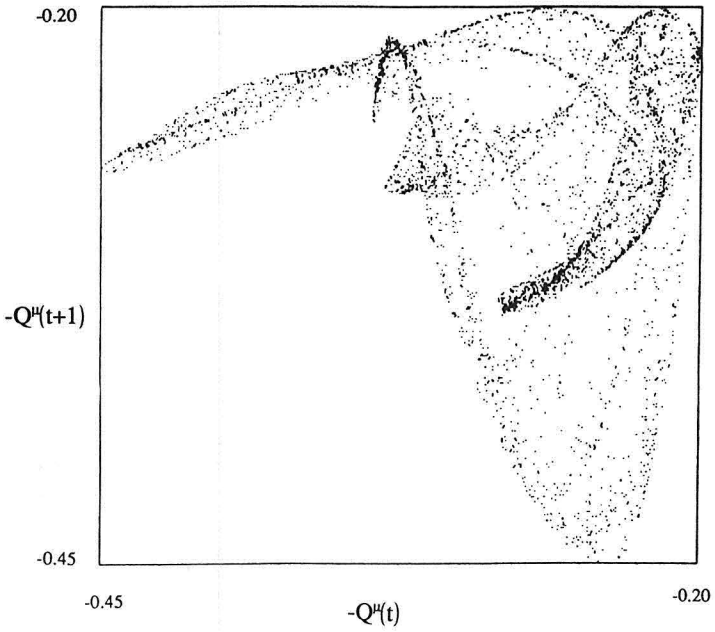

Figure 3: Pseudo-phase plot of the time series generated with the chaotic level of η detected by CBL. $\eta = 1.7749$. The Lyapunov exponent of this time series was 0.30587, indicating chaos.

$\eta = 1.7749$—the chaotic level of η in the previous experiment—is depicted in figure 3. The Lyapunov exponent [9, 22] related to this time series has the value 0.30587, clearly indicating chaos.

As an illustration of the independence of the performance of CBL on network size, the parity problem in a 10-3-1 network was also tested. The results are shown in figure 4. η and $\Delta\eta$ are initialized at 0.1 and 0.01, respectively. Chaos is detected after 50 cycles at $\eta = 0.9921$, and the cooling down step size was set at 0.0686. The final value of Q after 200 cycles was 0.00116492. After 1000 cycles Q had further decreased to 0.00008613.

In this cooling down phase another attractive property of the lower reflecting barrier was demonstrated. One of the patterns seems to slow down convergence. This was counteracted by the constant perturbation introduced by the reflecting lower barrier. GDP gave a final Q of 0.0886442 after 200 cycles and of 0.0003913 after 1000 cycles. GDP had terminated the slow convergence phase after 450 cycles.

In the simulations, differences in cooling down schedules could result in considerable differences in learning performance. Some decreasing/increasing sequences of η gave rise to quicker convergence than others. The exact relation between performance and the cooling down schedule still has to be

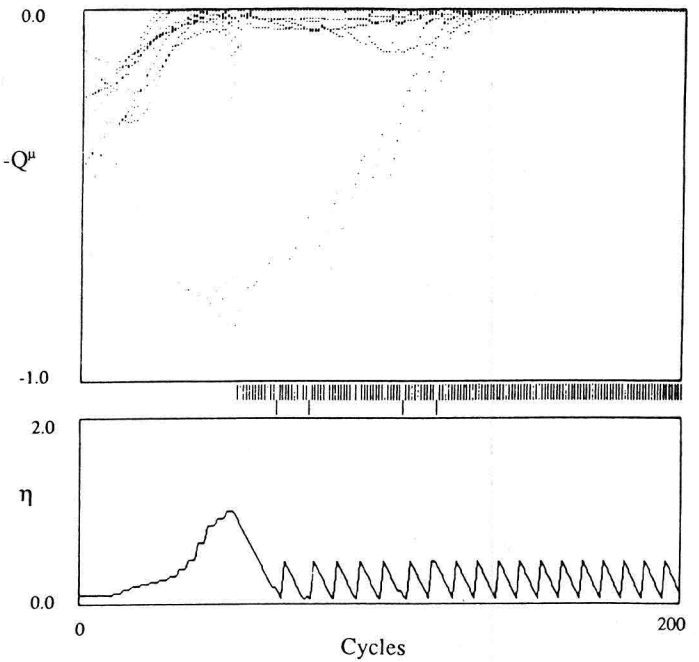

Figure 4: CBL performing the parity problem in a 10-3-1 network. η and $\Delta\eta$ were initialized at 0.1 and 0.01. respectively. Chaos was detected at $\eta = 0.9921$. The step size and the lower bound in the cooling down phase were set at 0.0686. Final $Q = 0.00116492$. After 1000 cycles Q had decreased to 0.00008613.

assessed. These results. however. suggest that a more deterministic method than the Metropolis algorithm could prove useful in speeding up convergence in annealing-like methods.

CBL, GDP acceleration techniques, and simulated annealing

Routines that adaptively vary η in order to speed up convergence mostly avoid instability (limit cycles or chaos) by increasing η in a linear way while decrements are exponential [2]. Another solution is to lock the system into stable behavior by rejecting the weights. which are related to instability [16]. CBL. however. indicates that the chaotic properties of networks can be used to speed up convergence.

Although standard GDP acceleration techniques. which adaptively vary the step sizes. provide faster convergence than standard GDP. they also imply extra computational overhead since they rely on an individual learning rate parameter for every weight in the system. CBL. however. suggests that

convergence can also be accelerated by varying only one global learning-rate parameter.

Like the simulated annealing method [4], implemented in the Boltzmann machine [1], CBL prescribes that the system cools down from an unstable state to a stable state. In CBL, η has a function similar to the computational temperature T used in the Boltzmann machine. In simulated annealing methods, T determines the amount of random state changes in the dynamics. This property can lift the dynamics out of local minima and enables the system to jump over energy barriers. A system is set off at a high temperature, which leads to a coarse or semi-global search for energy minima. Lowering the temperature makes the system respond to smaller energy differences and introduces local-search.

In the Boltzmann machine, T works on the level of the activity updating process and indirectly influences the weight changing process, which is optimized. Furthermore, T introduces random behavior into the network, which is completely independent of the initial conditions and the task the model has to perform.

η, as used in CBL, gives rise to chaotic behavior, which is completely dependent on the task at hand and the initial conditions, and works directly on the level at which optimization takes place. η allows the system to cross energy barriers by means of chaotically overshooting minima in the error surface. Varying η will not, like computational temperature, introduce dynamics-independent random search, but dynamics-dependent search founded in chaos.

Conclusion

CBL is based on a collective chaotic property of neural networks that is driven by an increased interaction between the computational elements. The function of this chaotic movement is to allow the system to perform coarse search in weight space and to provide a criterium in estimating optimal step sizes as a solution to the speed-stability trade-off.

How the properties of CBL bear upon our understanding of chaos observed in biological processes remains to be seen. CBL, however, provides an additional suggestion about the relation between chaos and computation in dynamical systems, which is an alternative to the hypothesis put forward by Langton [7]. Langton assumes that the fundamental conditions necessary for computation in complex systems have to be found at the edge of chaos, in the vicinity of a phase transition. CBL suggests that the optimal condition for some computational processes might be found by entering the chaotic domain and returning to a stable position.

CBL confirms the suggestion of Mantica and Sloan [10] that chaotic optimization could prove superior to other optimization schemes, and is a demonstration that chaos is not only an interesting property of dynamical systems, but can also be a useful one.

Acknowledgments

The author is very indebted to Ton Coolen, Yong Yao, and Walter Freeman for their helpful and critical comments.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, **9** (1985) 147–169.

[2] R. A. Jacobs, "Increasing Rates of Convergence through Learning Rate Adaptation," *Neural Networks*, **1**(4) (1988) 295–307.

[3] H. Kesten, "Accelerated Stochastic Approximation," *Annals of Mathematical Statistics*, **29** (1958) 41–59.

[4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, **220** (1983) 671–680.

[5] J. F. Kolen and J. B. Pollack, "Backpropagation is Sensitive to Initial Conditions," *Complex Systems*, **4** (1990) 269–280.

[6] K. E. Kürten and J. W. Clark, "Exemplification of Chaotic Activity in Nonlinear Neural Networks Obeying a Deterministic Dynamics in Continuous Time," in *Computer Simulation in Brain Science*, edited by R. M. J. Cotterill (Cambridge, Cambridge University Press, 1988).

[7] C. G. Langton, "Computation at the Edge of Chaos: Phase Transitions and Emergent Computation," *Physica D*, **42** (1990) 12–37.

[8] Y. Le Cun, "Learning Processes in an Asymmetric Threshold Network," in *Disordered Systems and Biological Organization*, edited by E. Bienenstock, F. Fogelman Souli, and G. Weisbuch (Berlin, Springer Verlag, 1986).

[9] A. J. Lichtenberg and M. A. Lieberman, *Regular and Stochastic Motion* (Berlin: Springer Verlag, 1983).

[10] G. Mantica and A. Sloan, "Chaotic Optimization and the Construction of Fractals: Solution of the Inverse Problem," *Complex Systems*, **3** (1989) 37–62.

[11] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of State Calculations for Fast Computing Machines," *Journal of Chemical Physics*, **6** (1953) 1087.

[12] D. Parker, "Learning Logic," Technical Report TR-87, Center for Computational Research in Economics and Management Science (Cambridge, MA, MIT, 1985).

[13] G. Radons, H. G. Schuster, and D. Werner, "Drift and Diffusion in Backpropagation Networks," pages 261–264 in *Parallel Processing in Neural Networks*, edited by R. Eckmiller, G. Hartman, and G. Hauske (Amsterdam, North-Holland, 1990).

[14] S. Renals and R. Rohwer, "A Study of Network Dynamics," *Journal of Statistical Physics*, **58** (1990) 825–848.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back Propagating Errors," *Nature*, **323** (1986) 533–536.

[16] F. M. Silva and L. B. Almeida, "Acceleration Techniques for the Backpropagation Algorithm," in *Lecture Notes in Computer Science: Neural Networks, Proceedings of Eurasip Workshop 1990*, edited by L. B. Almeida and C. J. Wellekens (Berlin, Springer Verlag, 1990).

[17] H. Sompolinsky, A. Cristiani, and H. J. Sommers, "Chaos in Random Neural Networks," *Physical Review Letters*, **61** (1988) 259–262.

[18] R. S. Sutton, "Two Problems with Back-Propagation and Other Steepest-Descent Learning Procedures for Networks," *Proceedings of the Eight Annual Conference of the Cognitive Science Society* (1986) 823–831.

[19] H. L. J. Van der Maas, P. F. M. J. Verschure, and P. C. M. Molenaar, "A Note on Chaotic Behavior in Simple Neural Networks," *Neural Networks*, **3**(1) (1990) 119–122.

[20] P. F. M. J. Verschure, H. L. J. Van der Maas, and P. C. M. Molenaar, "Chaos in Multilayered Networks" (manuscript).

[21] P. J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences" (Doctoral dissertation, Harvard University, 1974).

[22] A. Wolff, J. B. Swift, H. L. Swinney, and J. A. Vasano, "Determining Lyapunov Exponents from a Time Series," *Physica D*, **16** (1985) 285–317.