Complex Systems 4 (1990) 269–280

Backpropagation is Sensitive to Initial Conditions

John F. Kolen
Jordan B. Pollack
Laboratory for Artificial Intelligence Research,
Computer and Information Science Department,
The Ohio State University, Columbus, OH 43210, USA

Abstract. This paper explores the effect of initial weight selection on feed-forward networks learning simple functions with the backpropagation technique. We first demonstrate, through the use of Monte Carlo techniques, that the magnitude of the initial condition vector (in weight space) is a very significant parameter in convergence time variability. In order to further understand this result, additional deterministic experiments were performed. The results of these experiments demonstrate the extreme sensitivity of backpropagation to initial weight configuration.

Introduction

Backpropagation [17] is the network training method of choice for many neural network projects, for good reason. Like other weak methods, it is simple to implement, faster than many other "general" approaches, well-tested by the field, and easy to mold (with domain knowledge encoded in the learning environment) into very specific and efficient algorithms.

Rumelhart et al. made a confident statement: for many tasks, "the network rarely gets stuck in poor local minima that are significantly worse than the global minima." [17, p. 536] According to them, initial weights of exactly 0 cannot be used, since symmetries in the environment are not sufficient to break symmetries in initial weights. Since their paper was published, the convention in the field has been to choose initial weights with a uniform distribution between plus and minus ρ, usually set to 0.5 or less.

The convergence claim was based solely upon their empirical experience with the backpropagation technique. Since then, Minsky and Papert [14] have argued that there exists no proof of convergence for the technique, and several researchers [3, 9, 10] have found that the convergence time must be related to the difficulty of the problem; otherwise an unsolved computer science question ($P \stackrel{?}{=} NP$) would finally be answered. We do not wish to make claims about convergence of the technique in the limit (with vanishing

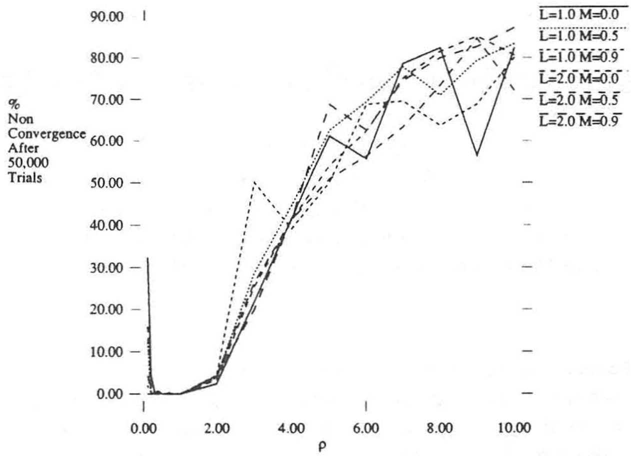

Figure 1: Percentage T-convergence vs. initial weight range.

step-size), or the relationship between task and performance, but wish to talk about a pervasive behavior of the technique which has gone unnoticed for several years: the sensitivity of backpropagation to initial conditions.

The Monte-Carlo experiment

Initially, we performed empirical studies to determine the effect of learning rate, momentum rate, and the range of initial weights on t-convergence [11]. We use the term *t-convergence* to refer to whether or not a network, starting at a precise initial configuration, could learn to separate the input patterns according to a boolean function (correct outputs above or below 0.5) within t epochs. The experiment consisted of training a 2-2-1 network on exclusive-or while varying three independent variables in 114 combinations: learning rate, η, equal to 1.0 or 2.0; momentum rate, α, equal to 0.0, 0.5, or 0.9; and initial weight range, ρ, equal to 0.1–0.9 in 0.1 increments, and 1.0–10.0 in 1.0 increments. Each combination of parameters was used to initialize and train a number of networks.[1] Figure 1 plots the percentage of t-convergent (where $t = 50,000$ epochs of 4 presentations) initial conditions for the 2-2-1 network trained on the exclusive-or problem. From the figure we thus conclude the choice of $\rho \leq 0.5$ is more than a convenient symmetry-breaking default; it is quite necessary to obtain low levels of nonconvergent behavior.

[1] Numbers ranged from 8 to 8355, depending on availability of computational resources. Those data points calculated with small samples were usually surrounded by data points with larger samples.

Scenes from exclusive-or

Why do networks exhibit the behavior illustrated in figure 1? While some might argue that very high initial weights (i.e., $\rho > 10$) lead to very long convergence times since the derivative of the semilinear sigmoid function is effectively zero for large weights, this does not explain the fact that when ρ is between 2 and 4, the non-t-convergence rate varies from 5 to 50 percent.

Thus, we decided to utilize a more deterministic approach for eliciting the structure of initial conditions giving rise to t-convergence. Unfortunately, most networks have many weights, and thus many dimensions in initial-condition space. We can, however, examine two-dimensional slices through the space in great detail. A slice is specified by an origin and two orthogonal directions (the X and Y axes). In the figures below, we vary the initial weights regularly throughout the plane formed by the axes (with the origin in the lower left-hand comer) and collect the results of running backpropagation to a particular time limit for each initial condition. The map is displayed with grey-level linearly related to time of convergence: black meaning not t-convergent and white representing the fastest convergence time in the picture. Figure 2 is a schematic representation of the networks used in this and the following experiment. The numbers on the links and in the nodes will be used for identification purposes. Figures 3 through 11 show several interesting "slices" of the the initial condition space for 2-2-1 networks trained on exclusive-or. Each slice is compactly identified by its nine-dimensional weight vector and associated learning/momentum rates. For instance, the vector $(-3+2+7-4X+5-2-6Y)$ describes a network with an initial weight of -0.3 between the left hidden unit and the left input unit. Likewise, "+5" in the sixth position represents an initial bias of 0.5 to the right hidden unit. The letters "X" and "Y" indicate that the corresponding weight is varied along the X- or Y-axis from -10.0 to $+10.0$ in steps of 0.1. All the figures in this paper contain the results of 40,000 runs of backpropagation (i.e., 200 pixels by 200 pixels) for up to 200 epochs (where an epoch consists of 4 training examples).

Figures 12 and 13 present a closer look at the sensitivity of backpropagation to initial conditions. These figures zoom into a complex region of figure 11; the captions list the location of the origin and step size used to generate each picture.

Sensitivity behavior can also be demonstrated with even simpler functions. Take the case of a 2-2-1 network learning the "or" function. Figure 14 shows the effect of learning "or" on networks $(+5+5-1X+5-1Y+3-1)$ and varying weights 4 (X-axis) and 7 (Y-axis) from -20.0 to 20.0 in steps of 0.2. Figure 15 shows the same region, except that it partitions the display according to equivalent solution networks after t-convergence (200 epoch limit), rather than the time to convergence. Two networks are considered equivalent[2] if their weights have the same sign. Since there are 9 weights,

[2]For rendering purposes only. It is extremely difficult to know precisely the equivalence classes of solutions, so we approximated.

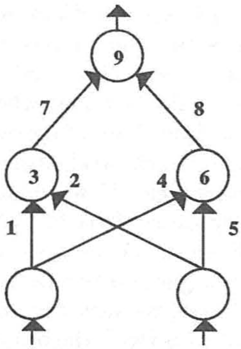

Figure 2. Schematic network.

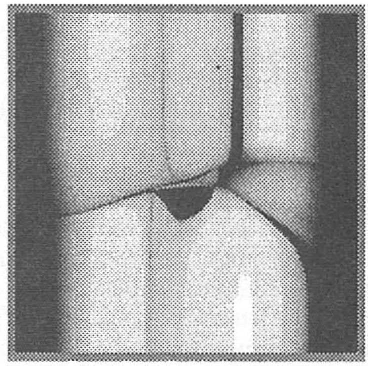

Figure 3. $(-5-3+3+6Y-1-6+7X)\eta = 3.25$ $\alpha = 0.40$.

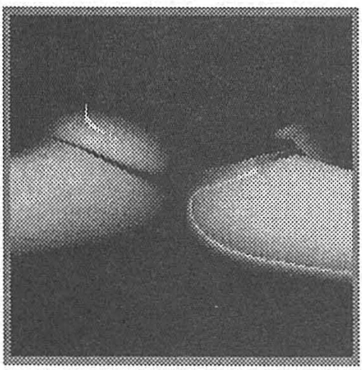

Figure 4. $(+4-7+6+0-3Y+1X+1)$ $\eta = 2.75$ $\alpha = 0.0$.

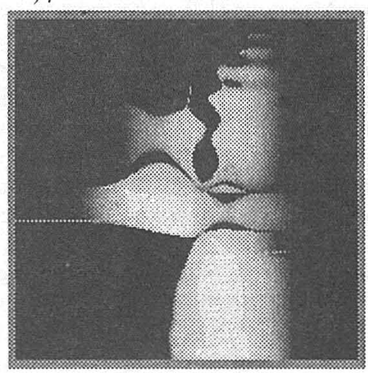

Figure 5. $(-5+5+1-6+3XY+8+3)$ $\eta = 2.75$ $\alpha = 0.80$.

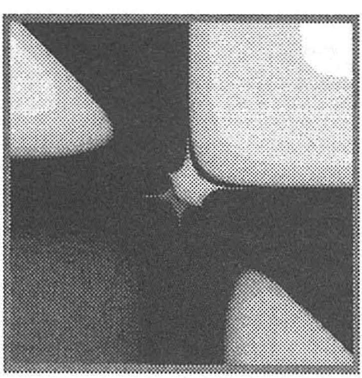

Figure 6. $(Y, X-3+6+8+3+1+7-3)$ $\eta = 3.25$ $\alpha = 0.00$.

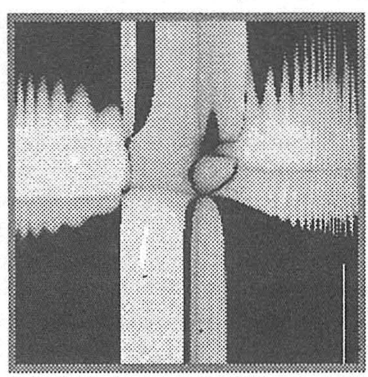

Figure 7. $(Y+3-9-2+6+7-3X+7)$ $\eta = 3.25$ $\alpha = 0.60$.

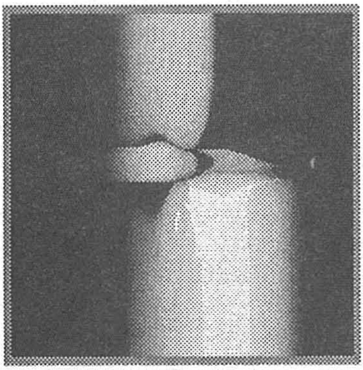

Figure 8. $(-6-4XY-6-6+9-4-9)$ $\eta = 3.00$ $\alpha = 0.50$.

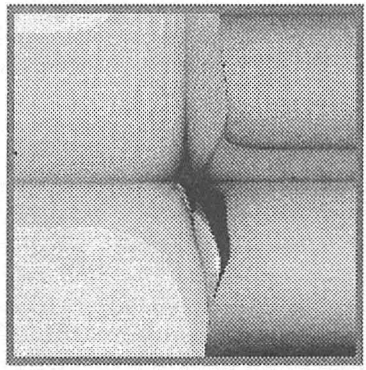

Figure 9. $(-2+1+9-1X-3+8Y-4)$ $\eta = 2.75$ $\alpha = 0.20$.

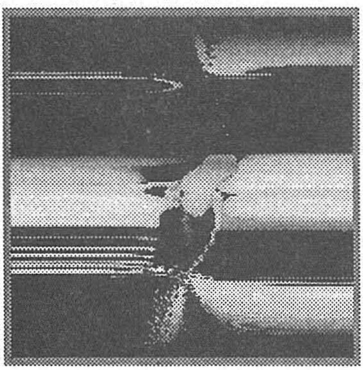

Figure 10. $(+1+8-3-6X-1+1+8Y)$ $\eta = 3.50$ $\alpha = 0.90$.

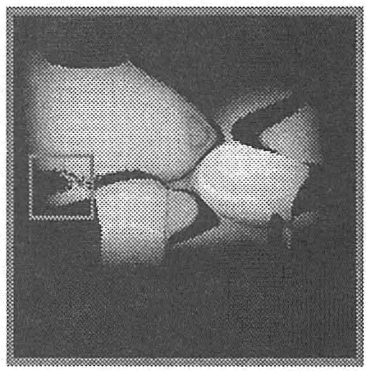

Figure 11. $(+7+4-9-9-5Y-3+9X)$ $\eta = 3.00$ $\alpha = 0.70$.

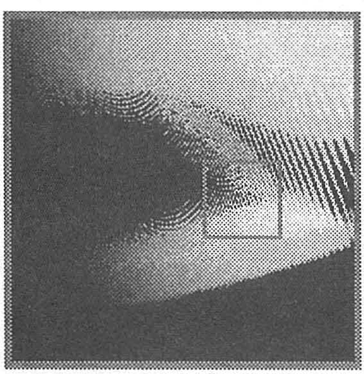

Figure 12. $(-9.0, -1.8)$ step 0.018.

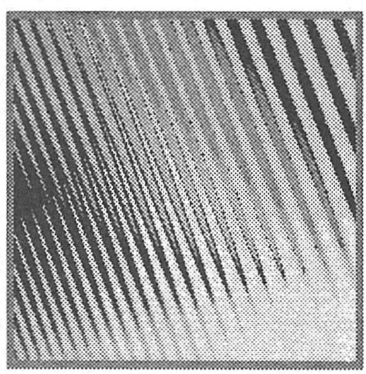

Figure 13. $(-6.966, -0.500)$ step 0.004.

there are 512 (2^9) possible network equivalence classes. Figures 16 through 25 show successive zooms into the central swirl identified by the XY coordinate of the lower-left corner and pixel step size. After 200 iterations, the resulting networks could be partitioned into 37 (both convergent and nonconvergent) classes. Obviously, the smooth behavior of the t-convergence plots can be deceiving, since two initial conditions, arbitrarily alike, can obtain quite different final network configuration.

Note the triangles appearing in figures 19, 21, 23 and the mosaic in figure 25 corresponding to the area that did not converge in 200 iterations in figure 24. The triangular boundaries are similar to fractal structures generated under iterated function systems [2]; in this case, the iterated function is the backpropagation learning method. We propose that these fractal-like boundaries arise in backpropagation due to the existence of multiple solutions (attractors), the nonzero learning parameters, and the nonlinear deterministic nature of the gradient descent approach. When more than one hidden unit is utilized, or when an environment has internal symmetry or is very underconstrained, there will be multiple attractors corresponding to the large number of hidden-unit permutations that form equivalence classes of functionality. As the number of solutions available to the gradient descent method increases, the more complicated the nonlocal interactions between them. This explains the puzzling result that several researchers have noted, that as more hidden units are added, instead of speeding up, backpropagation slows down (e.g., [13]). Rather than a hill-climbing metaphor with local peaks to get stuck on, we should instead think of a many-body metaphor: The existence of many bodies does not imply that a particle will take a simple path to land on one. From this view, we see that Rumelhart et al.'s claim of backpropagation usually converging is due to a very tight focus inside the "eye of the storm."

Could learning and momentum rates also be involved in the storm? Such a question prompted another study, this time focused on the interaction of learning and momentum rates. Rather than alter the initial weights of a set of networks, we varied the learning rate along the X axis and momentum rate along the Y axis. Figures 26, 27, and 28 were produced by training a 3-3-1 network on 3-bit parity until t-convergence (250 epoch limit). Table 1 lists the initial weights of the networks trained in figures 26 through 31. Examination of the fuzzy area in figure 26 shows how small changes in learning and/or momentum rate can drastically affect t-convergence (figures 30 and 31).

Discussion

Chaotic behavior has been carefully circumvented by many neural network researchers (e.g., through the choice of symmetric weights by Hopfield [5]), but has been reported in increasing frequency over the past few years [1, 4, 6, 12, 16, 18]. Connectionists, who use neural models for cognitive modeling, disregard these reports of extreme nonlinear behavior in spite of common knowledge that nonlinearity is what enables network models to perform non-

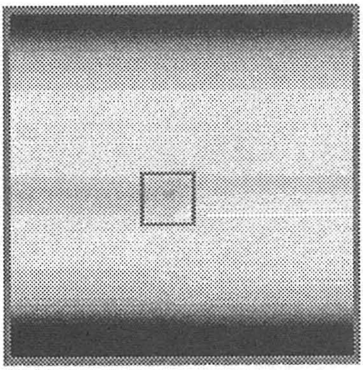

Figure 14. (−20.00000, −20.00000) step 0.200000.

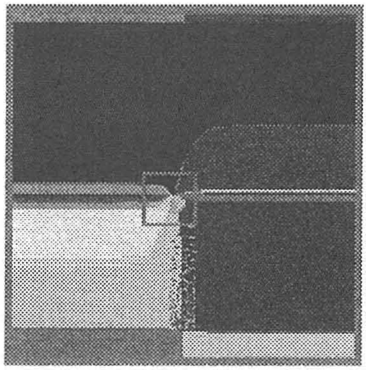

Figure 15. Solution networks.

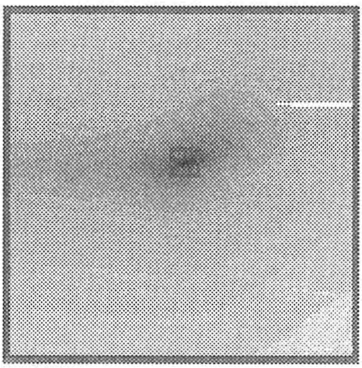

Figure 16. (−4.500000, −4.500000) step 0.30000.

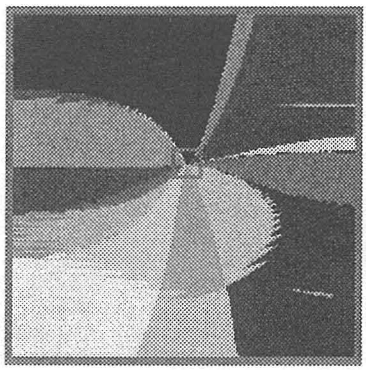

Figure 17. Solution networks.

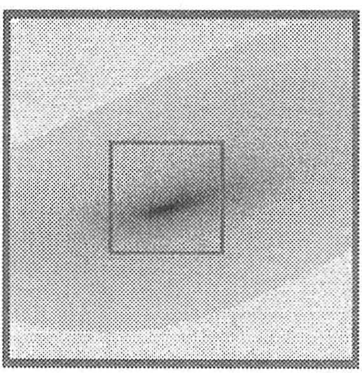

Figure 18. (−1.680000, −1.350000) step 0.002400.

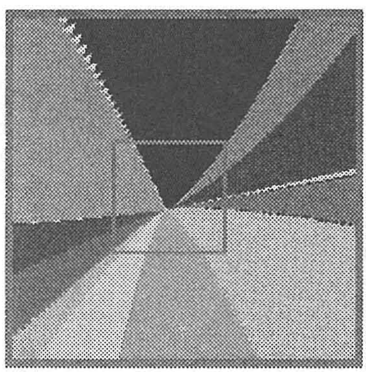

Figure 19. Solution networks.

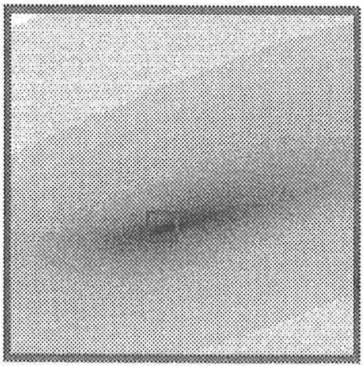

Figure 20. (−1.536000, −1.197000) step 0.000780.

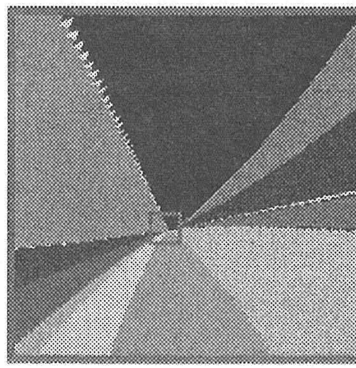

Figure 21. Solution networks.

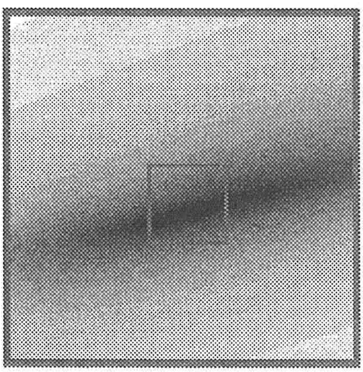

Figure 22. (−1.472820, −1.145520) step 0.000070.

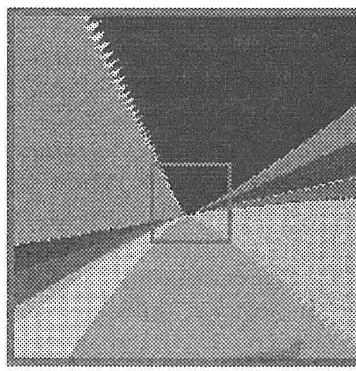

Figure 23. Solution networks.

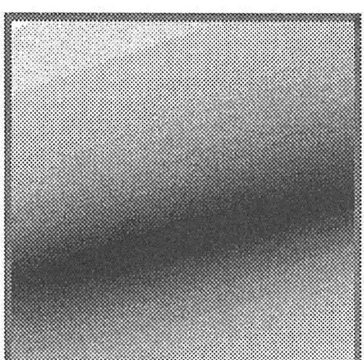

Figure 24. (−1.467150, −1.140760) step 0.000016.

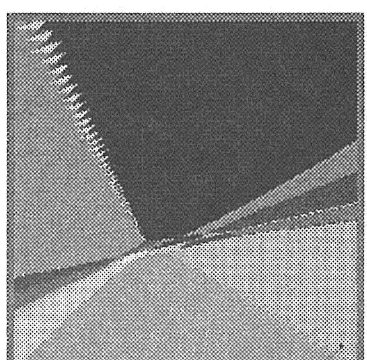

Figure 25. Solution networks.

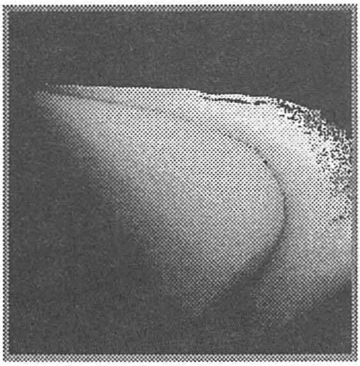

Figure 26. $\eta = (0.0, 4.0)$ $\alpha = (0.0, 1.25)$.

Figure 27. $\eta = (0.0, 4.0)$ $\alpha = (0.0, 1.25)$.

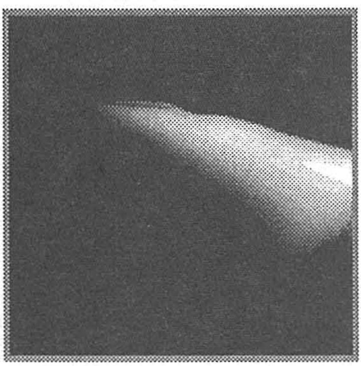

Figure 28. $\eta = (0.0, 4.0)$ $\alpha = (0.0, 1.25)$.

Figure 29. $\eta = (3.456, 3.504)$ $\alpha = (0.835, 0.840)$.

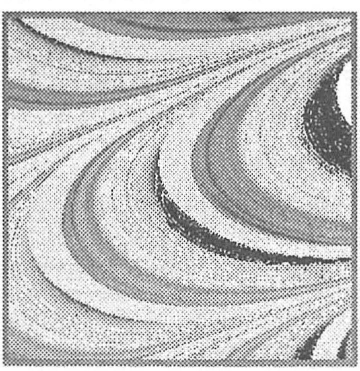

Figure 30. $\eta = (3.84, 3.936)$ $\alpha = (0.59, 0.62)$.

	Figures 26, 29, 30	Figure 27	Figure 28
Weight 1	−0.34959000	−0.34959000	−0.34959000
Weight 2	0.00560000	0.00560000	0.00560000
Weight 3	−0.26338813	0.39881098	0.65060705
Weight 4	0.75501968	−0.16718577	0.75501968
Weight 5	0.47040862	−0.28598450	0.91281711
Weight 6	−0.18438011	−0.18438011	−0.19279729
Weight 7	0.46700363	−0.06778983	0.56181073
Weight 8	−0.48619500	0.66061292	0.20220653
Weight 9	0.62821201	−0.39539510	0.11201949
Weight 10	0.90039973	0.55021922	0.67401200
Weight 11	0.48940201	0.35141364	−0.54978875
Weight 12	−0.70239312	−0.17438740	−0.69839197
Weight 13	−0.95838741	−0.07619988	−0.19659844
Weight 14	0.46940394	0.88460041	0.89221204
Weight 15	−0.73719884	0.67141031	−0.56879740
Weight 16	0.96140103	−0.10578894	0.20201484

Table 1: Network weights for figures 26 through 30.

trivial computations in the first place. All work to date has noticed various forms of chaos in network dynamics, but not in learning dynamics. Even if backpropagation is shown to be nonchaotic in the limit, this still does not preclude the existence of fractal boundaries between attractor basins since other nonchaotic nonlinear systems produce such boundaries (i.e., forced pendulums with two attractors [7]).

What does this mean to the backpropagation community? From an engineering applications standpoint, where only the solution matters, nothing at all. When an optimal set of weights for a particular problem is discovered, it can be reproduced through digital means. From a scientific standpoint, however, this sensitivity to initial conditions demands that neural network *learning* results must be specially treated to guarantee replicability. When theoretical claims are made (from experience) regarding the power of an adaptive network to model some phenomena, or when claims are made regarding the similarity between psychological data and network performance, *the initial conditions for the network need to be precisely specified or filed in a public scientific database.*

What about the future of backpropagation? We remain neutral on the issue of its ultimate convergence, but our result points to a few directions for improved methods. Since the slowdown occurs as a result of global influences of multiple solutions, an algorithm for first factoring the symmetry out of both network and training environment (e.g., domain knowledge) may be helpful. Furthermore, it may also turn out that search methods that harness "strange attractors" ergodically guaranteed to come arbitrarily close to some subset of solutions might work better than methods based on strict gradient descent. Finally, we view this result as strong impetus to discover how to exploit the information-creative aspects of nonlinear dynamical systems for future models of cognition [15].

Acknowledgments

This work was supported by Office of Naval Research grant number N00014-89-J1200. Substantial free use of over 200 Sun workstations was generously provided by our department.

References

[1] K.L. Babcock and R.M. Westervelt, "Dynamics of simple electronic neural networks," *Physica*, **28D** (1987) 305–316.

[2] M. Barnsley, *Fractals Everywhere* (Academic Press, San Diego, 1988).

[3] A. Blum and R. Rivest, "Training a 3-node neural network is NP-complete," *Proceedings of IEEE Conference on Neural Information Processing Systems* (Denver, Colorado, 1988).

[4] M.Y. Choi and B.A. Huberman, "Dynamic behavior of nonlinear networks," *Physical Review A*, **28** (1983) 1204–1206.

[5] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings US National Academy of Science* **79** (1982) 2554–2558.

[6] B. Derrida and R. Meir, "Chaotic behavior of a layered neural network," *Physical Review A*, **38** (1988) 3116–3119.

[7] D. D'Humieres, M.R. Beasley, B.A. Huberman, and A. Libchaber, "Chaotic states and routes to chaos in the forced pendulum," *Physical Review A*, **26** (1982) 3483–3496.

[8] B.A. Huberman and T. Hogg, "Phase transitions and artificial intelligence," *Artificial Intelligence*, **33** (1987) 155–172.

[9] S. Judd, "Learning in networks is hard," *Journal of Complexity*, 4 (1988) 177–192.

[10] J. Kolen, "Faster learning through a probabilistic approximation algorithm," *Proceedings of the Second IEEE International Conference on Neural Networks* (San Diego, 1988) I:449–454.

[11] J. Kolen and A. Goel, "Learning in parallel distributed processing networks: Computational complexity and information content," *IEEE Transactions on Systems, Man, and Cybernetics*, in press.

[12] K.E. Kurten and J.W. Clark, "Chaos in neural networks," *Physics Letters*, **114A** (1986) 413–418.

[13] R.P. Lippman and B. Gold, "Neural classifiers useful for speech recognition," in *1st International Conference on Neural Networks* (IEEE, 1987) IV:417–426.

[14] M.L. Minsky and S.A. Papert, *Perceptrons* (MIT Press, 1988).

[15] J.B. Pollack, "Implications of recursive auto associative memories," in D. Touretzky (ed.), *Advances in Neural Information Processing Systems* (Morgan Kaufman, San Mateo, 1989) 527–536.

[16] U. Riedal, R. Kuhn, and J.L. van Hemmen, "Temporal sequences and chaos in neural nets," *Physical Review A*, **38** (1988) 1105–1108.

[17] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning representation by back-propagating errors," *Nature*, **323** (1986) 533–536.

[18] H. Sompolinsky, A. Crisanti, H.J. Sommers, "Chaos in random neural networks," *Physical Review Letters*, **61** (1988) 259–262.