Visualizing Weight Dynamics in the N-2-N Encoder

Raymond Lister

Dept. of Electrical Engineering, University of Queensland, QLD 4072, Australia

Abstract— Kruglyak proved that sets of weights exist so that Multi-Layer Perceptrons can solve arbitrarily large N–2–N encoder problems. We extend Kruglyak's static geometric construction to give a way of visualizing weights dynamics during learning. This visualization provides insight as to why Back Propagation has difficulty in finding suitable N–2–N encoder weights for N>8. We believe this new insight has general consequences, relating to the danger of utilizing intermediate activity values in hidden units, and difficulties with finding solutions for tightly constrained (but not necessarily large) problems.

I. INTRODUCTION

The N–M–N encoder is a standard benchmark learning problem. It was first used to demonstrate a learning algorithm for the Boltzmann Machine [1], and first applied to Multi-Layer Perceptrons [2], when Rumelhart, Hinton, and Williams introduced their version of the Back Propagation (BP) algorithm.

The "obvious" N–M–N solution (for humans) is where the M hidden units only adopt near discrete values. Such solutions are only possible for $M \geq \log_2 N$ encoders. Kruglyak showed that sets of weights exist so that analog Multi-Layer Perceptrons can utilize intermediate levels of hidden unit activity to solve arbitrarily large N–2–N encoder problems [3]. Furthermore, Rumelhart *et al.* demonstrated that BP can in principle learn weights that give intermediate levels of activity. They commented: *"Our experience is that ... whenever the learning problem calls for it, ... [hidden units] ... can learn to take on graded values"* (page 339). They also note: *"Relatively small learning rates make units employing intermediate values easier to obtain"* (page 338), but they did not explore the quantitative relationship.

N–2–N encoders with N>4 are a useful vehicle for studying a learning algorithm's ability to utilize graded values of activity in the hidden units. However, earlier workers have concentrated on encoders that do not require graded values. For instance, most of Fahlman's *Quickprop* experiments [4] were performed on 10–5–10 encoders. He did experiment with an 8–2–8 encoder, but in all other experiments $M \geq \log_2 N$, despite experimenting with encoders as large as 256–8–256. In the light of Rumelhart *et al*'s comments, and Kruglyak's proof, Fahlman's designation of N–$\log_2 N$–N encoders as *"tight"* seems arbitrary.

We carried out sets of 100 runs on N–2–N problems of various sizes, using a learning rate $\eta = 0.5$, and momentum $\alpha = 0.9$. The results are summarized in Table I. BP consistently solved problems with $N \leq 8$. However, twenty percent of 9-2-9 runs failed, in that they consumed 1×10^6 epochs before being terminated. Each of the 80 successful N=9 runs required less than one tenth of the number of epochs allotted to each failed run. Only three of 100 runs on 10–2–10 encoders were successful. Although one of these three successful runs required almost half a million epochs, this is still less than half the number of epochs allotted to the 97 failed runs.

TABLE I

RESULTS FROM BP RUNS ON VARIOUS N–2–N PROBLEMS, WITH $\eta = 0.5$ AND $\alpha = 0.9$.

Problem Size	No. Successes	Number of Epochs: Fastest Success	Median Success	Slowest Success
4-2-4	100/100	33	67	238
5-2-5	100/100	60	147	520
6-2-6	100/100	123	521	1559
7-2-7	100/100	647	1672	4103
8-2-8	100/100	1739	5448	21291
9-2-9	80/100	4972	19792	85778
10-2-10	3/100	31186	48864	453947

Why should the BP-trained N–2–N encoder exhibit such a sudden change of behavior for N>8? The remainder of this paper describes why, and explores some of the consequences for other learning problems.

A. Kruglyak's Construction

We shall denote the input units as I1 ... IN, the output units as O1 ... ON, and the hidden units as H1 and H2. The activity of any of these units is represented in lower case: e.g. i_j, o_j, and h_g respectively. Bias weights are implemented by connections from a *true unit* T, that always has activity equal to one. The weight of a connection from unit X to unit Y is $w_{X,Y}$. Throughout this paper we use the standard *sigmoid* activation function, ranging from zero to one.

Kruglyak's proof is based on a geometric construction on the $h_1 h_2$ plane. The activity of the two hidden units define a point within the unit square of the $h_1 h_2$ plane. Since only one input unit is on in each training pattern of the encoder problem, it is a simple matter to determine weights that place such a point anywhere in the $h_1 h_2$ unit square. Let h_g^j be the desired value of hidden unit Hg when $i_j = 1$. Given

0-7803-0999-5/93/$03.00 ©1993 IEEE

any value for $w_{T,Hg}$, the necessary value for $w_{Ij,Hg}$ is:

$$w_{Ij,Hg} = -w_{T,Hg} - \log_e(1/h_g^j - 1) \quad \cdots (1)$$

Weights into an output unit Oj are represented as a set of parallel lines (*activity contours*), where each line describes the set of points (h_1,h_2) for which the unit's activity is constant: $o_j = a$. The slope of these contours, m_j, and their h_2 intercepts, $b_{j,a}$ are:

$$m_j = \frac{-w_{H1,Oj}}{w_{H2,Oj}} \quad \cdots (2)$$

$$b_{j,a} = \frac{-w_{T,Oj}}{w_{H2,Oj}} - \frac{\log_e(1/a - 1)}{w_{H2,Oj}} \quad \cdots (3)$$

We shall refer to the region of the h_1h_2 plane where $o_j > 0.5$ as the *high side* of the 0.5 contour of Oj, and the other region as the *low side*.

Kruglyak showed that it is always possible to position N 0.5 contours in the unit square of the h_1h_2 plane so that there exist N regions in the square that are on the high side of exactly one 0.5 contour, for arbitrarily large N. A generalized form of Kruglyak's construction is given in Fig. 1. The line crossing the lower left quadrant of the unit square is the 0.5 contour for unit O1. The other lines in the unit square are 0.5 contours for other Oj. The low side of all these 0.5 contours is the side containing the center of the unit square. The point marked "I1" is a suitable hidden unit pattern for when $i_1 = 1$. The other points marked "I2"..."I6" are the correspondingly suitable points for other input units. (Henceforth, we shall use the notation "Oj" to refer to both an output unit, and its corresponding 0.5 contour. Similarly, we shall use "Ij" to refer to an input unit and the hidden unit pattern when $i_j = 1$. Context resolves ambiguity. In fact, having now defined the geometry, we shall largely dispense with the network itself, and talk almost exclusively of the geometry.) Table II gives suitable weights for some 0.5 contours in Fig. 1. The scaling factors k_j may take on any real positive value. Equation (1) can be used to derive suitable values for the points marked "Ij" in Fig. 1. BP tends to learn weights so that the N Ij patterns are evenly distributed about the square, which requires approximately $N/2$ levels of activity in each hidden unit.

II. WEIGHT DYNAMICS

We have used Kruglyak's representation to study how weights change during learning. The hidden unit patterns Ij move (usually slowly) around the unit square. The contours change in three ways. As the ratios in equation (2) and the first term of (3) change, the contours undergo *contour rotation* and *contour translation* respectively. The third form of movement is *contour compression*. When $a = 0.5$,

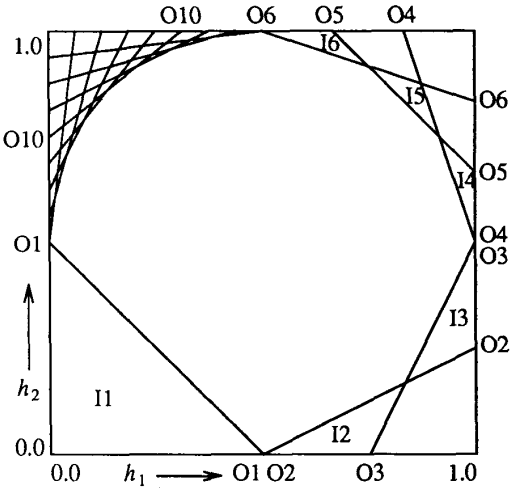

Fig. 1. Kruglyak's geometric proof.

TABLE II
SOME WEIGHTS ASSOCIATED WITH FIG. 1.

$W_{x,y}$	O1	O2	O3	O6	O10
T	$1/2k_1$	$-1/2k_2$	$-3/2k_3$	$-7/2k_6$	$-15/4k_{10}$
H1	$-1k_1$	$1k_2$	$2k_3$	$1k_6$	$-4k_{10}$
H2	$-1k_1$	$-2k_2$	$-1k_3$	$3k_6$	$5k_{10}$
m_j	-1	1/2	2	-1/3	4/5
$b_{j,0.5}$	1/2	-1/4	-3/2	7/6	12/16

the second term in (3) equals zero. The perpendicular distance from the $o_j = 0.5$ contour to the $o_j = a$ contour is:

$$p_{j,a} = \frac{\log_e(1/a - 1)}{\sqrt{(w_{H1,Oj})^2 + (w_{H2,Oj})^2}} \quad \cdots (4)$$

Thus, if we increased the k_j scaling factors in Table II, the 0.5 contours in Fig. 1 would not move, but their corresponding non-0.5 contours would compress around each of them. We refer to the non-0.5 contours as the *contour field*. At each point in the h_1h_2 unit square, we refer to the rate of change of an output unit's activity, in the direction perpendicular to its contours and increasing activity, as the *activity gradient* or *field strength*.

The net "force" on a hidden unit pattern Ij has N components, one for each contour field. Each Ij is pushed in the direction of increasing field strength for its corresponding output unit, and decreasing field strength for all other output units. The magnitude of each component is related to the field strength. Thus each component is strongest when the Ij is near the 0.5 contour for that output unit. The component is particularly strong when the weights into the output unit are large, and as a result the non-0.5 contours are tightly bunched around the 0.5 contour.

Conversely, the component is particularly weak when the weights into the output unit are large, but the hidden unit pattern is well removed from the 0.5 contour.

Likewise, there are N components to the net "force" on the contour field for an output unit Oj, one for each hidden unit pattern. The component due to Ij acts to move the Oj activity field so as to increase o_j at the point occupied by Ij. The remaining components act to decrease o_j at the points occupied by Ix≠j. The magnitude of each of these components is proportional to the field strength at the point occupied by each hidden unit pattern. The combination of these N "forces" results in some combination of contour translation, rotation, and compression on the Oj activity field.

III. A 9-2-9 BACK PROPAGATION RUN THAT FAILS

The initial weights for all BP runs described here were chosen uniformly from the interval [–0.3,0.3], giving an initial network where the hidden and output units always have activity near 0.5. Initially then all hidden unit patterns Ij lie near the center of the unit square, the 0.5 contours are at random orientations, and their associated non-0.5 contours are relatively uncompressed. Within the first few iterations, however, all 0.5 contours rotate to have a negative slope and translate to a position where they pass beneath the unit square. This is because each output unit is being trained to be off in all but one training case, and being off in all cases is a good early approximation. As this happens, the *overall measure of error*, E, (as defined in [2], p.323) drops quickly from its initial value above 10, to around 4.5. The upper graph in Fig. 2 plots E against epoch for the remainder of the run. The lower graph plots the number of correct training patterns against epoch.

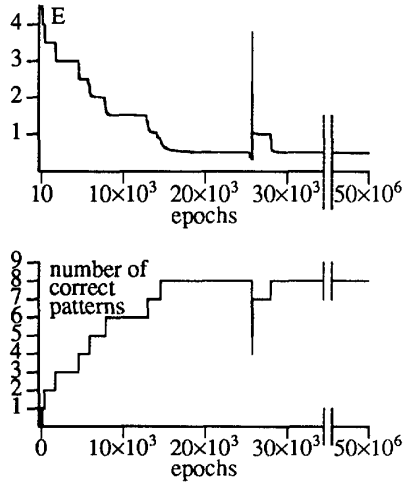

Fig. 2. The behavior of a failed 9-2-9 run.

From epoch 10 till around epoch 15×10^3, the plot of E has a terraced structure. That is, E stays relatively constant for long periods, interspersed with periods in which it drops quickly. These sharp drops occur each time a 0.5 contour moves into the unit square and to a position where its corresponding Ij pattern is on its high side (i.e. each time a training pattern is learnt) The first 0.5 contour to do so (O4) translates through the bottom left corner of the unit square. The next three 0.5 contours to enter (O5, O3, and O7, in that order) rotate and enter via the other three corners. Fig. 3 shows the weight geometry at iteration 5000, which is a few hundred epochs after O7 became the fourth 0.5 contour to enter the square, via the top right corner. The dashed lines show non 0.5 contours 0.1, 0.2, 0.3 ... 0.9. Hidden unit patterns I4, I5, I3, and I7 have all taken up positions on the high side of their respective 0.5 contours, near the corners of the square. The remaining five 0.5 contours still lie to the lower left of the square.

Notice that, in Fig. 3, hidden unit patterns I1, I2, I6, I8, and I9 are crushed together, near the center of the square. This is because they are all pushed towards the region of lower field strength of O3, O4, O5, and O7. This

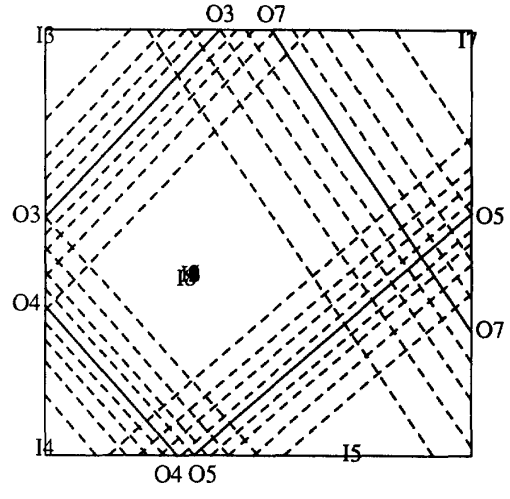

Fig. 3. The 9-2-9 run at epoch 5000.

TABLE III
WEIGHTS ASSOCIATED WITH FIG. 3.

W	O3	O4	O5	O7
T	-5.51	3.66	-3.75	-9.08
H1	-10.49	-11.78	10.58	7.58
H2	9.81	-10.39	-12.08	5.09
m_j	1.07	-1.13	0.88	-1.49
b_j	0.56	0.35	-0.31	1.78
$p_{j,01}$	0.15	0.14	0.14	0.24

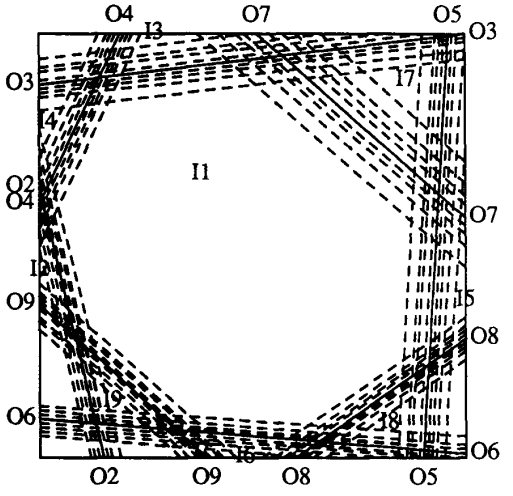

Fig. 4. The 9-2-9 run at epoch 20000.

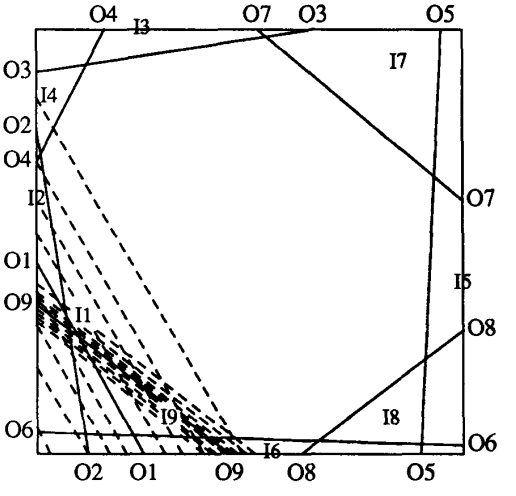

Fig. 5. The 9-2-9 run at epoch 25740.

TABLE IV
WEIGHTS ASSOCIATED WITH FIG. 4.

W	H1	H2	O1	O9
T	-0.33	-0.05	-7.31	13.64
I1	-0.17	0.75		
I9	-1.24	-1.88		
H1			-3.45	-34.75
H2			-3.81	-38.20
m_j			-0.91	-0.91
b_j			-1.92	0.36
$p_{j,0.1}$			0.43	0.04

TABLE V
WEIGHTS ASSOCIATED WITH FIG. 5.

W	H1	H2	O1	O9
T	-0.35	-0.11	2.54	16.13
I1	-1.79	-0.65		
I9	-0.46	-2.32		
H1			-9.97	-36.10
H2			-5.66	-45.67
m_{Oj}			-1.76	-0.79
b_{Oj}			0.45	0.35
$p_{j,0.1}$			0.19	0.04

crushing effect is a major impediment to the acquisition of the remaining training patterns. The respective 0.5 contours of these Ij cannot move towards them until there is some degree of spatial separation between these patterns. These Ij do eventually move apart, as contour compression proceeds around the O4, O5, O3, and O7 contours, but the process is slow, and this accounts for the terraced nature of the plot of E.

As more 0.5 contours enter the square, the incumbent 0.5 contours move to accommodate them. The eighth training pattern is learnt after about 15000 epochs. Fig. 4 shows the weight geometry after 20000 epochs. Note how the incumbent contours of Fig. 3 (O3, O4, O5, and O7) have rotated, translated, and compressed to accommodate the later 0.5 contours.

It is at this point of the run, after the acquisition of the eighth training pattern, that the run's behavior changes. From around epoch 15000, there follows a long period in which E improves very slowly. At around 25×10^3 epochs, there is a small dip in E, and then it increases very quickly, accompanied by a sudden loss of most training patterns. Fig. 5 shows the weight geometry just before the spike in E. The final 0.5 contour, marked O1, is trying to enter the square, but the eight 0.5 contours already in the square cannot move easily to accommodate O1. The eight 0.5 contours, and their respective Ij, have "locked" together, so that one of them can only move if the others also move.

The O1 contour left "hovering" just inside the square has a destabilizing effect on all the Ij, and most training patterns are lost within 10 iterations (see Fig. 6). The destabilization is due indirectly to the disparity between the degree of compression associated with O1 and all the other 0.5 contours. The low degree of compression of the O1 field allows it to attract the I1 pattern towards the O1 0.5 contour, from a large distance across the square. However, when the I1 pattern gets near to the O1 0.5 contour, it steps into regions of very high field strength associated with O9 and O2, and is then hurled a considerable distance back across the square ... only to be attracted by O1 again, and so on. The O1 field also affects the other Ij, and I9 in

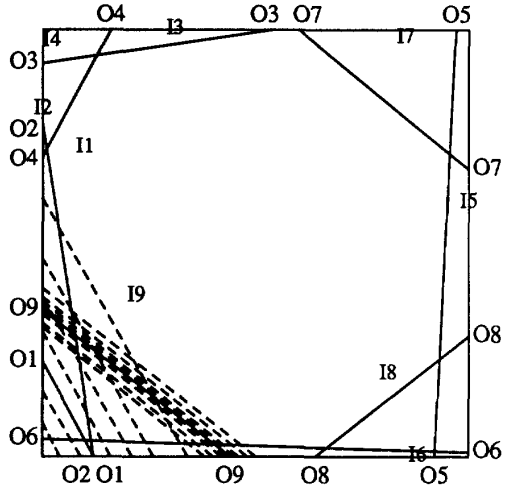

Fig. 6. The 9-2-9 run at epoch 25750.

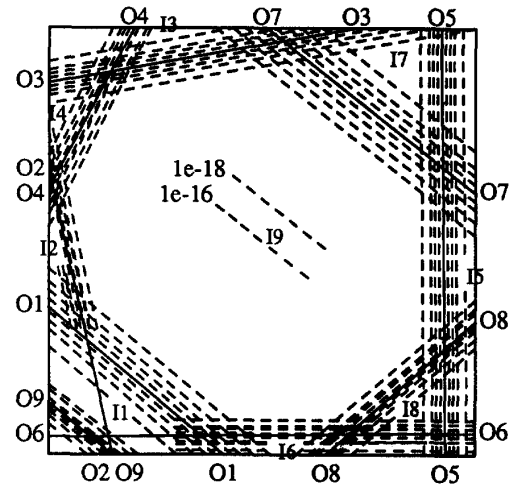

Fig. 7. The 9-2-9 run at epoch 30000.

TABLE VI
WEIGHTS ASSOCIATED WITH FIG. 6.

W	H1	H2	O1	O9
T	-0.20	1.05	1.29	15.81
I1	-1.99	-0.12		
I9	-1.05	-1.57		
H1			-10.32	-36.05
H2			-5.85	-45.87
m_{oj}			-1.76	-0.79
b_{oj}			0.22	0.34
$p_{j,0.1}$			0.19	0.04

TABLE VII
WEIGHTS ASSOCIATED WITH FIG. 7.

W	H1	H2	O1	O9
T	-0.21	-0.08	8.35	5.56
I1	-1.38	-2.21		
I9	0.32	0.09		
H1			-20.42	-37.49
H2			-24.39	-47.60
m_j			-0.84	-0.79
b_j			0.34	0.12
$p_{j,0.1}$			0.07	0.04

particular. The O1 field pushes I9 back towards the region of high field strength associated with O9, where I9 is then thrown some distance, towards the regions of field strength associated with O2 and O6 ... from where its thrown back towards the O9 0.5 contour. By this *pin ball effect*, the I9 hidden unit pattern can "pick up speed" to the point where it can jump, in consecutive epochs from the region where $o_9 > 0.9$ to the region where $o_9 < 0.1$.

The movement of hidden unit patterns I1 and I9 are further exacerbated by *hidden unit bias coupling*. When two or more Ij are moving in the same direction, they combine to have a large effect on $w_{T,H1}$ and $w_{T,H2}$, which causes all Ij to "jump" some distance across the unit square. Ij located away from the corners of the square (i.e. hidden unit patterns with intermediate activity values) are particularly susceptible to hidden unit bias coupling, because in these cases $w_{T,Hg}$ approximately equals $w_{Ij,Hg}$. Note in Fig. 6 that it is not just I1 and I9 that have moved to positions either on the low side of their respective 0.5 contours, or positions on the high side of the wrong 0.5 contour. Most Ij have moved to an incorrect position, due to this coupling effect; even those well away from the lower left corner and the direct destabilizing effect of O1.

Fig. 7 shows the weight geometry after 30000 epochs. The Ij have moved back to correct positions, except I9. The training patterns associated with I1/O1 has been learnt, but at the expense of the pattern associated with I9/O9. Hidden unit patterns I1, I2, and I6 then act to push O9 away from I9. The extra annotated contours near I9 indicate the weakness of "forces" between I9 and O9. This run was terminated after 50×10^6 epochs, without any apparent improvement.

IV. VARIATION OF THE LEARNING PARAMETERS.

As Rumelhart *et al.* indicated (non-geometrically), the destabilizing effect of an incoming contour can be decreased by reducing the amount the Ij can move i.e. by lowering η and α. However, our experiments with lower learning parameters indicate that each halving of the learning rate only increased the size of problem that could be solved reliably by a near constant increment. But each

halving also roughly doubled the median epoch length of the successful runs at a given N. Thus, as N grows, the run time required for reliable convergence would appear to increase exponentially. With $\eta = 2^{-5}$ and $\alpha = 0.9$, 19-2-19 problems could be solved reliably, when given a maximum of 2×10^6 epochs, but the median length of successful runs was approximately 6.3×10^5 epochs.

Momentum is a major cause of the instability that leads to loss of training patterns. With $\eta = 0.5$, $\alpha = 0$, and failed runs terminated after 4×10^6 epochs, 15-2-15 problems were solved reliably, but the median length successful run was about 2.3×10^6 epochs.

V. QUICKPROP

We repeated this study with Quickprop [4]. Apart from our use of the [0,1] sigmoid activation function, we did not alter the default parameters in the code made publicly available via FTP by Fahlman. Any run that consumed 1×10^6 was terminated, and deemed to be a failure. The results are summarized in Table VIII.

TABLE VIII
RESULTS FOR QUICKPROP.

Problem Size	No. Successes	Number of Epochs		
		Fastest Success	Median Success	Slowest Success
4-2-4	100/100	8	16	42
8-2-8	100/100	39	96	180
9-2-9	100/100	70	178	350
10-2-10	100/100	99	345	543
11-2-11	100/100	250	768	1697
12-2-12	50/100	515	1056	55962
13-2-13	16/100	1103	40934	118781

Quickprop was able to reliably learn larger problems than BP could with $\eta = 2^{-1}$ and $\alpha = 0.9$, but Quickprop could not reliably learn larger problems solved routinely by BP with $\eta = 2^{-5}$ and $\alpha = 0.9$. We are somewhat surprised by this, as Quickprop adjusts its step size according to its estimate of the second derivatives. Perhaps the behavior is due to one of Quickprop's design assumptions: *"... the change in the slope of the error surface, as seen by each weight, is not affected by all the other weights"* [4] (p. 46). Our failed Quickprop runs did not exhibit the "catastrophic disassembly" that BP runs exhibit. This is presumably because Quickprop is more sensitive to sudden changes in the slope of the error surface, and only makes small changes to the weights in those regions. Unfortunately, it appears this can also make the algorithm very slow.

VI. CONCLUSION

The poor performance of BP and Quickprop on the N-2-N problem, for large N, is due to the error surface being ill-conditioned for gradient descent. When sigmoidal hidden units utilize intermediate activity values, small perturbations of the weights can lead to sudden loss of many training patterns. In these ill-conditioned regions of the weight space, the curvature of the error surface changes very rapidly, leading BP to "overstep" and lose many training patterns already learnt. Reducing the learning rate and momentum does increase the size of problem that can be solved reliably but, as N increases, the run time required for reliable convergence appears to increase exponentially.

Quickprop, which utilizes second derivative information, proved much faster than BP on smaller encoders, but became extremely slow around N=12. More sophisticated gradient descent algorithms, such as conjugate gradient techniques, will probably increase the size of problem that can be solved reliably, but we suspect that such techniques will still exhibit poor scaling behavior at some N.

The N–M–N encoder was first introduced to demonstrate the Boltzmann Machine (BM) [1]. It is well known that the BM struggles on encoder problems as small as 4–2–4, but it should be noted that for such a problem the discrete-state BM is operating at the limits of its representational capabilities, since weights do not exist for the BM to solve N–2–N encoders with N>4. The analog Multi-Layer Perceptron has no such hard representational limit. Perhaps the apparently inferior performance of the BM learning algorithm is not due to any fundamental superiority of the Back Propagation learning algorithm, but results from the greater representational redundancy of Multi-Layer Perceptrons.

ACKNOWLEDGMENT

Thanks to Jim Stone for his comments on a draft of this paper.

REFERENCES

[1] D. Ackley, G. Hinton, and T. Sejnowski, "A Learning Algorithm for Boltzmann Machines", *Cognitive Science* vol. 9, pp 147-169, 1985.

[2] D. Rumelhart, G. Hinton, and R. Williams, "Learning Internal Representations by Error Propagation", in *Parallel Distributed Processing*, vol. 1, D. Rumelhart and J. McClelland, Eds. MIT Press, 1986, pp 318-362.

[3] L. Kruglyak, "How to Solve the N Bit Encoder Problem with Just Two Hidden Units", *Neural Computation*, vol. 2, no. 4 (Winter), pp 399-401, 1990.

[4] S. Fahlman, "Faster-Learning Variations on Back-Propagation: An Empirical Study", in *Proceedings of the 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton and T. Sejnowski, Eds. Morgan Kaufmann, 1989, pp 38-51.