ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/290636470

The N-2-N Encoder: A Matter of Representation

Chapter · January 1993

DOI: 10.1007/978-1-4471-2063-6_152

CITATIONS
9

READS
408

3 authors, including:

Steven Phillips
National Institute of Advanced Industrial Science and Technology
115 PUBLICATIONS **2,111** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

EEG functional connectivity : modeling and inference View project

All content following this page was uploaded by Steven Phillips on 04 October 2016.

The user has requested enhancement of the downloaded file.

The N-2-N Encoder: A Matter of Representation

Paul Bakker, Steven Phillips†, Janet Wiles

Depts. of Computer Science & Psychology
†Dept. of Computer Science
The University of Queensland, QLD 4072 Australia
email: *bakker@cs.uq.oz.au*

Abstract Kruglyak [4] demonstrated that weights exist to implement the N-2-N encoder for all finite N, but it is known that Backpropagation (BP) is unusually poor at learning such solutions for N > 8 ([6]). We show that the learning problem lies not with BP, but with the pattern representation typically used in the encoder task. With an appropriate representation, we demonstrate that BP can learn encoders in approximately linear time with N as large as 100. This underlines yet again the crucial importance of pattern representation in neural network learning.

1 Introduction

The N-M-N encoder is a simple task that has been used to study the training properties of the Backpropagation learning algorithm [1, 2, 7]. A three-layer network consisting of N input and output units, and M hidden units, must learn to compress a set of N input patterns into M hidden units, and reconstruct them on the output units. If M is less than log_2 N, the encoder is termed a *tight* encoder [2].

The most popular encoder in the current literature is the N-2-N. This is the tightest encoder possible (for N > 2), and is known to present BP with a formidable learning task [5]. Kruglyak [4] proved geometrically that weights exist to implement the N-2-N encoder for *any* finite N. It was noted, however, that the existence of these weights does not imply that BP can learn them; he reported that he was only aware of encoders as large as 8-2-8 being learned successfully by a "modified" version of BP. The N-2-N was hence proposed as a challenging task for testing new learning algorithms [4].

The N-2-N is a good test of a learning algorithm because it requires that all input patterns be compressed into 2-dimensional hidden unit space. The efficient use of hidden unit space is vital in neural net learning tasks, where the number of hidden units is usually minimized to improve generalization.

2 Solving the N-2-N encoder task

Kruglyak [4] plotted a theoretical solution of the N-2-N encoder; in Figure 1 we show the hidden unit space of an *actual* solution[1] to the 19-2-19 encoder, similar to that found by Lister [5]. The activation range of hidden unit 1 is on the abscissa, and that of hidden unit 2 is on the ordinate. The 19 data points in Figure 1 correspond to the internal representations of the input patterns, and the line segments represent the 19 hyperplanes determined by the weights to each output unit. With respect to a particular output unit, the solution requires all hyperplanes to be arranged so that each point is on the *low side* of all but one of the hyperplanes; this corresponds to an output response of 0 for all patterns except one. When this constraint is extended to all output units, the solution requires the points to be evenly distributed to form an approximate circle with each hyperplane uniquely partitioning off one point [4].

The difficulty of the task is due to the critical positioning of the points and hyperplanes. As N increases, the triangular region which isolates a single point becomes smaller. Consequently, finding a solution requires greater precision and hence smaller steps across the error surface. Empirical results

[1] This 19-2-19 encoder was learned by standard BP in 38.3 million pattern presentations

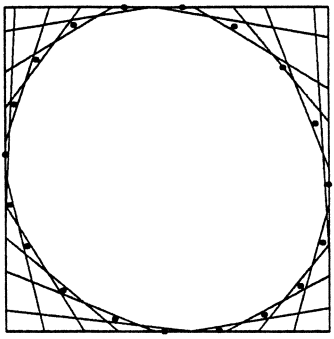

Figure 1: Solution of the 19-2-19 encoder with local encoding on the target vectors.

bear this out: Lister's [6] convergence figures demonstrate that learning time grows exponentially with N.

3 Would a Different Representation Help?

Representation is a crucial factor in neural network learning [3]. In studying the learning dynamics of the N-2-N problem, we noticed that the input and output representation traditionally used greatly handicaps Backpropagation. The standard encoding is *local*, with one unit on and all others off. For example, in the 6-2-6 encoder, the input (and target) set is:

100000 010000 001000 000100 000010 000001

Note that for every output unit, the target is 0 for (N-1) patterns, and 1 for just one pattern. When training commences, we have observed that BP always sets all output values to near-zero values in the first few epochs. This is because a null output vector is a very close approximation of *all* of the target vectors shown above [6]. This initial action, though understandable, is actually a step *away* from the final solution. It leads to all weights in the output layer becoming negative and saturates the output units, making further learning very slow (the *flat-spot* effect; [2]).

Obviously, this problem is directly related to the preponderance of zeroes in the local pattern encoding. This prompted us to speculate that the difficulty that BP has in learning the N-2-N task may not be due so much to the demands of the task itself, but due to the output representation typically used.

We decided to experiment with different representations to test this hypothesis. The N-2-N task remains logically the same (information must be compressed into a bottleneck and reconstructed), but we used a pattern encoding that has equal (or near equal) quantities of 1's and 0's in the target patterns. This will be referred to as N/2 *block* encoding.

For example, in the 6-2-6 encoder, the target vectors become:

111000 011100 001110 000111 100011 110001

It was expected that the use of block encoding on the target vectors would stop the counterproductive behaviour of Backpropagation in the initial stages of training, and would lead to faster convergence on N-2-N encoders, as well as the ability to learn larger encoders.

4 Simulations and Results

Standard Backpropagation [7] was used to learn N-2-N encoders for increasing N. For each N and each target pattern encoding (local and block), ten trials were run using a set of matched starting weights. Starting weights were matched to control for their influence on convergence times. A

learning rate of 0.01, epoch update and a momentum factor of 0.9 were used in all runs; a training run was considered to have converged if all outputs were within 0.5 of their target values.

Table 1: Median number of pattern presentations required to achieve convergence.

	Median number of pattern presentations	
Encoder	Local Encoding	Block Encoding
4-2-4	5416	2208
9-2-9	743058	4077
14-2-14	8950718	4844
19-2-19	35583922	6973
50-2-50	—	12600
100-2-100	—	30300

With a learning rate of 0.01 and a local encoding scheme on the target vectors, BP was only able to learn up to N = 19. When the block representation was used, however, BP was able to learn all encoders faster, and was able to learn the 100-2-100 encoder consistently in just a few hundred epochs.

5 Analysis and Discussion

The use of N/2 block encoding on the target vectors resulted in significantly accelerated learning and the ability to learn much larger encoders than in the local case. This effect can be explained both in terms of learning dynamics, and in the organization of hyperplanes required to implement the solution.

Firstly, the equalization of the number of 1's and 0's presented to the output units as targets changed the learning behaviour of BP; instead of all output activations moving to 0 and saturating in the first few epochs, we now find all output units moving to an intermediate activation value of 0.5. Although significant, this does not completely explain the magnitude of the speedup reported. In examining the hidden unit space of the solution (see Figure 2), we found that the use of N/2 block encoding fundamentally changes the arrangement of hyperplanes required to solve the encoder. Instead of each hyperplane being required to carve off one unique data point, it now only has to carve off N/2 points; that is, in each case, half of the output units must be "on", and the other half "off". This requires far less precision than in the local encoding case, as each point can now be isolated within a larger region of hidden unit activation space.

In fact, with a block encoding, learning time (number of pattern presentations) grew approximately linearly with the size of the encoder (N). This is a dramatic improvement over the local encoding case where, with the same learning algorithm (BP) and parameters, learning time grew exponentially with N.

6 Conclusion

Because of its simplicity, the N-2-N encoder/decoder is a useful and interesting problem for characterizing the learning properties of Backpropagation. We found that BP learned much faster when a desired response was distributed over half the output units (block encoding), than when confined to a single output unit (local encoding), which is the more common practice. This reiterates the important point that learnability is determined not only by the search technique, but also by the representation of the problem [3]. We agree with Lister's [6] conclusions that the poor performance of BP on the encoder task is due to the error surface being ill-conditioned for gradient descent. The block representation used here creates an error surface more conducive to gradient descent search.

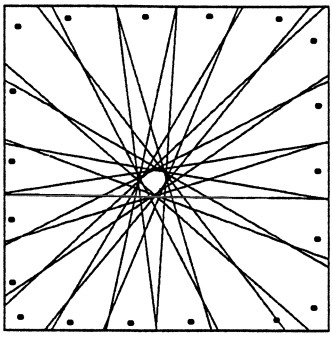

Figure2: Solution of the 19-2-19 encoder with block encoding on the target vectors.

In further work, we are trying to establish the limits of block encoding. We have been able to consistently solve 500-2-500 encoders, and are currently attempting the 1000-2-1000. We anticipate that, subject to the precision of the simulator, any size encoder can be tractably learned.

Acknowledgements

The authors would like to thank Ray Lister for insightful discussions on the N-2-N problem. We would also like to thank Anthony Bloesch for his help, and the rest of the Connectionists group at The University of Queensland for encouragement and discussions.

References

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

[2] Fahlman, S.E. (1989). Faster-learning variations on back-propagation: An empirical study. In D. Touretzky, G. Hinton & T. Sejnowski (Eds.), *Proceedings of the 1988 Connectionist Models Summer School* (pp. 38-51). San Mateo, CA: Morgan Kaufmann.

[3] Hertz, J., Krogh, K., & Palmer, R.G. (1991). *Introduction to the theory of neural computation* (p. 144). Redwood City, CA: Addison Wesley.

[4] Kruglyak, L. (1990). How to solve the N bit encoder problem with just two hidden units. *Neural Computation*, *2*(4), 399-401.

[5] Lister, R. (1992). *On making the right moves: Neural networks, gradient descent, and simulated annealing.* Unpublished doctoral dissertation, The University of Sydney, Australia.

[6] Lister, R. (1993). Visualizing weight dynamics in the N-2-N encoder. In *IEEE International Conference on Neural Networks* (pp. 684-689). IEEE.

[7] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (eds.) *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1. Cambridge, MA: The MIT Press.

View publication stats