Complex Systems **1** (1987) 367–372

Scaling relationships in back-propagation learning: dependence on training set size

Gerald Tesauro
Center for Complex Systems Research and Department of Physics,
University of Illinois at Urbana-Champaign,
508 South Sixth Street, Champaign, IL 61820, USA

Abstract. We study the amount of time needed to learn a fixed training set in the "back-propagation" procedure for learning in multi-layer neural network models. The task chosen was 32-bit parity, a high-order function for which memorization of specific input-output pairs is necessary. For small training sets, the learning time is consistent with a $\frac{4}{3}$-power law dependence on the number of patterns in the training set. For larger training sets, the learning time diverges at a critical training set size which appears to be related to the storage capacity of the network.

There is now widespread interest in devising adaptive learning procedures for massively-parallel networks of neuron-like computing elements [1,2,10,11,12,17]. A prime example is the "back-propagation" learning procedure [8,9,6] for multi-layer feed-forward networks. In this procedure, a set of patterns is presented to the input layer of the network, and the network's output is computed by feed-forward propagation of the input signal. An error function is then obtained by calculating the mean squared difference between the network's output and the desired output for each pattern. The connection strengths, or weights, of the network are then modified so as to minimize the error function according to a gradient-descent rule. This algorithm has displayed impressive performance for small-scale problems, and it is now of great interest to determine how it will scale to larger, more difficult problems.

The question of scaling can be approached from several different directions. Typically one would ask how some measure of the network's performance (such as the training time, or the fraction of patterns classified correctly) scales with a parameter describing the network, the task, or the learning algorithm. Basic parameters describing the network include: the number of input units n, the number of hidden units h, and the number of layers l. Related parameters include: the total number of possible input patterns (2^n for binary inputs), and the storage capacity of the network,

i.e., the number of modifiable weights and thresholds. (For a standard 3-layer network this is approximately nh.) The major parameter describing the computational task is the order of the computation k (as defined by Minsky and Papert [7]). Parameters describing the learning algorithm include: the learning rate ϵ, the "momentum" coefficient α, and the size of initial random weights r. An example of a scaling relationship would be a statement of how the training time T depends on the order of computation k. (There is reason to believe that the dependence should be exponential, i.e., $T \sim c^k$ [18,19,20,3,16].)

In this brief note, we consider the situation in which the network is trained to compute a Boolean function of a fixed set of input patterns. The size of the training set S is taken to be much less than the total number of possible input patterns. This is expected to be representative of the general situation for large problems. In contrast, most previous studies of small problems used all possible inputs in training the network. This is feasible only for networks with a small number of input units; for large problems it quickly becomes intractable to generate all 2^n input patterns for a problem of size n.

Given a fixed training set of size S, a natural question to ask is how the training time T scales with S, keeping everything else fixed. One would expect two fundamentally different kinds of scaling behavior, depending on whether or not the network is able to "generalize". By generalization we mean that the network abstracts the general principle of the computation, and is able to correctly classify new inputs without having seen them previously in the training phase. Generalization will be possible if the network has enough weights to solve the problem, and if the training set contains enough exemplars to determine the general rule.

If generalization is possible, then one would expect that the learning of a particular example in the training set is assisted by the presence of the other examples. In other words, the required number of presentations of each example should *decrease* as the training set size increases, and multiplying by S, one concludes that the total training time T should increase at a *slower* than linear rate. (Such sublinear scaling has in fact been seen for the problem of text-to-speech conversion [14], a problem in which substantial generalization is possible.) In the extreme case of perfect generalization, one would expect that T would remain *constant* as S increases, because the network would already be able to correctly classify new exemplars due to generalization. (This type of scaling behavior has also been seen for the problem of detecting symmetries in random-dot patterns [13].) On the other hand, if generalization is not possible, then the required number of presentations of each example should remain constant (or possibly increase due to conflicts) as S increases. This implies that the training time should increase at least linearly with the number of patterns to be memorized.

The study reported here conforms to the situation in which generalization is (effectively) not possible. The computational task was 32-bit parity, i.e., the input is a string of 32 binary digits, and the desired output is a 1

if there are an odd number of 1's in the input string and 0 otherwise. It is suggested that the minimal number of distinct training patterns necessary to obtain correct generalization increases exponentially with the order of the computation being trained. Since $k = 32$ for 32-bit parity, this would imply that generalization would take place only when the training set contains some substantial fraction of all possible inputs (i.e. $S \sim 2^{32}$). For training sets much smaller than this, generalization would not be possible, and the network could only memorize the patterns in the training set.

The network used in this study was a standard three-layer feed-forward network, with 32 input units and one output unit. The hidden layer contained either 8, 16, or 32 hidden units, which were fully connected to the input and output layers. The training set contained between 10 and 1000 random 32-bit integers. The back-propagation algorithm of [14] was used to modify the weights, with parameters $\epsilon = 1.0$, $\alpha = 0.9$, $r = 0.5$, and a margin of 0.1. The weights were updated after every pattern. Continuous cycling through the data base proceeded until the network reached 80% performance on the training set, at which time the training was stopped and the training time recorded.

Numerical results are plotted in log-log form in figure 1. Reported training times typically represent averages over 3-5 individual runs, and the errors in these measurements are expected to be around 10%. One can see in figure 1 that for small values of S, the log-log plot appears to be roughly linear in form, indicating a power-law $T \sim S^{\gamma}$. For the network with 32 hidden units, no significant devation from this law was observed up to 1000 training patterns. However, for 16 hidden units, the final data point at $S = 1000$ deviated from the power-law relationship, and for 8 hidden units, the deviation appears earlier, at $S = 500$. (For $S = 1000$ the 8-hidden unit net was unable to learn the training set to the 80% performance level.)

Standard fits applied to the apparently linear regions in figure 1 yield $\gamma = 1.33 \pm 0.02$ for 32 hidden units, $\gamma = 1.34 \pm 0.03$ for 16 hidden units, and $\gamma = 1.34 \pm 0.03$ for 8 hidden units.

The data presented is consistent with the following interpretation. There is a critical training set size S_c which is proportional to the storage capacity of the network. For values of $S \ll S_c$, the network has no trouble memorizing the individual patterns, and the training time follows a well-behaved $\frac{4}{3}$-power law dependence on S. However, as S approaches S_c, the limitations of the finite storage capacity of the network cause the training time to diverge. (For $S > S_c$ the training time is infinite, i.e., the network is not able to learn the training set.) This proposed scaling relationship is summarized in the following formula:

$$T \sim AS^{4/3} + B(S_c - S)^{-q}$$

where $S_c \sim nh$ and q is undetermined from the data.

In summary, there appears to be a simple scaling relationship for learning 32-bit parity with a fixed training set. Such scaling relationships may be elucidated by a more fundamental mathematical analysis. No fundamental

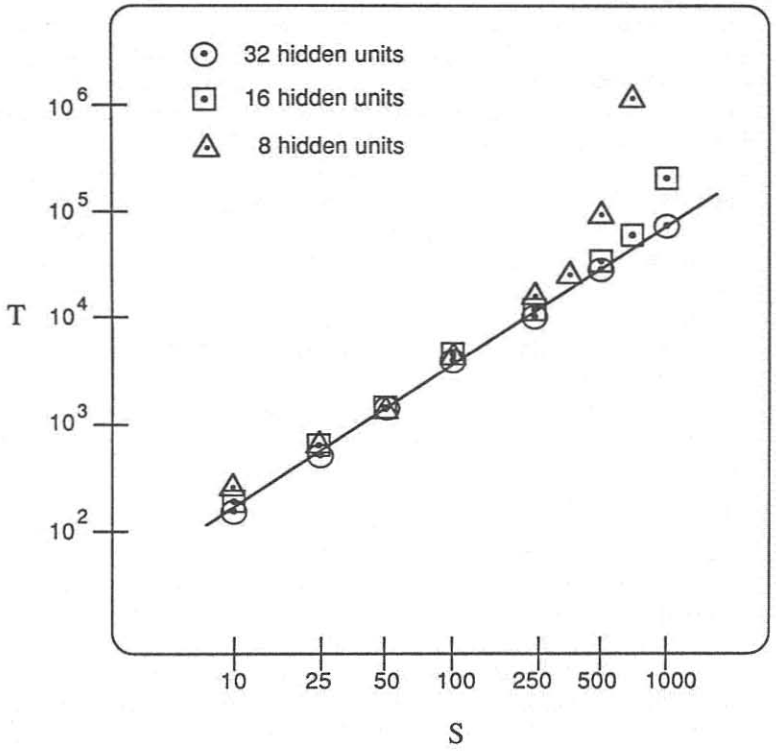

Figure 1: The number T of presentations needed for networks of various sizes to give 80% correct results for the 32-bit parity computation, as a function of the number of instances S used in training them. The straight line represents a best fit to the results for the 32 hidden unit case, and suggests a relation $T \sim S^{4/3}$.

explanation is yet known for the peculiar value of the scaling exponent γ. Qualitative arguments might explain an integer or half-integer exponent; however, a $\frac{4}{3}$ exponent defies simple explanation. It would be of great interest to determine the extent of "universality" of this particular exponent. This might be approached for this particular problem by varying other parameters such as n, ϵ, and α. It might also be approached by studying other computational problems such as memorization of random outputs. (For learning of a data base of expert backgammon moves, the training time appears to scale roughly quadratically with the size of the data base [15].) Finally, a comparison with other learning algorithms is clearly worthwhile. In particular, a similar scaling study for the Boltzmann machine learning algorithm [4,5,1] would provide a great deal of insight into the issue of whether the Boltzmann algorithm or the back-propagation algorithm will scale more favorably in large difficult problem domains.

Acknowledgements

The author thanks T. Sejnowski for providing the back-propagation simulator code. Useful discussions with S. Omohundro and T. Sejnowski are also acknowledged. This work was supported in part by the National Center for Supercomputing Applications.

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] A. G. Barto and P. Anandan, "Pattern-recognizing stochastic learning automata," *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-15** (1985) 360–375.

[3] S. E. Hampson and D. J. Volper, "Disjunctive models of boolean category learning," *Biological Cybernetics* (1987), in press.

[4] G. E. Hinton and T. J. Sejnowski, "Optimal perceptual inference," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Washington, D. C., 1983) 448–453.

[5] G. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 2: Psychological and Biological Models*, edited by J. L. McClelland & D. E. Rumelhart, (MIT Press, 1986).

[6] Y. Le Cun, "A learning procedure for asymmetric network," *Proceedings of Cognitiva (Paris)*, **85** (1985) 599–604.

[7] M. Minsky and S. Papert, *Perceptrons*, (MIT Press, 1969).

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in D. E. Rumelhart and J. L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, (MIT Press, 1986) 318–362.

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

[10] D. E. Rumelhart and J. L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, (MIT Press, 1986).

[11] D. E. Rumelhart and J. L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 2: Psychological and Biological Models*, (MIT Press, 1986).

[12] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," *Cognitive Science* **9** (1985) 75–112.

[13] T. J. Sejnowski, P. K. Kienker and G. E. Hinton, "Learning symmetry groups with hidden units: beyond the perceptron," in D. Farmer et al. (eds.), *Evolution, Games and Learning: Models for Adaptation in Machines and Nature* (North-Holland, 1986) 260–275.

[14] T. J. Sejnowski and C. R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text," *Complex Systems* **1** (1987) 145–168.

[15] G. Tesauro and T. J. Sejnowski, "A parallel network that learns to play backgammon," in preparation (1987).

[16] G. Tesauro and R. Janssens, "Scaling relationships in back-propagation learning: dependence on predicate order," in preparation (1987).

[17] G. Toulouse, S. Dehaene, and J. P. Changeux, "Spin glass model of learning by selection," *Proceedings of the National Academy of Sciences USA*, **83** (1986) 1695–1698.

[18] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, **27** (1984) 1134–1142.

[19] L. G. Valiant, "Learning disjunctions of conjunctions," *Proceedings of the International Joint Conference on Artificial Intelligence*, (1985) 560–566.

[20] D. J. Volper and S. E. Hampson, "Connectionist models of boolean category representation," *Biological Cybernetics*, **54** (1986) 393–406.