Complex Systems **2** (1988) 641–648

Teaching Feed-Forward Neural Networks by Simulated Annealing

Jonathan Engel
Norman Bridge Laboratory of Physics 161-33, California Institute of Technology, Pasadena, CA 91125, USA

Abstract. Simulated annealing is applied to the problem of teaching feed-forward neural networks with discrete-valued weights. Network performance is optimized by repeated presentation of training data at lower and lower temperatures. Several examples, including the parity and "clump-recognition" problems are treated, scaling with network complexity is discussed, and the viability of mean-field approximations to the annealing process is considered.

1. Introduction

Back propagation [1] and related techniques have focused attention on the prospect of effective learning by feed-forward neural networks with hidden layers. Most current teaching methods suffer from one of several problems, including the tendency to get stuck in local minima, and poor performance in large-scale examples. In addition, gradient-descent methods are applicable only when the weights can assume a continuum of values. The solutions reached by back propagation are sometimes only marginally stable against perturbations, and rounding off weights after or during the procedure can severely affect network performance. If the weights are restricted to a few discrete values, an alternative is required.

At first sight, such a restriction seems counterproductive. Why decrease the flexibility of the network any more than necessary? One answer is that truly tunable analog weights are still a bit beyond the capabilities of current VLSI technology. New techniques will no doubt be developed, but there are other more fundamental reasons to prefer discrete-weight networks. Application of back propagation often results in weights that vary greatly, must be precisely specified, and embody no particular pattern. If the network is to incorporate structured rules underlying the examples it has learned, the weights ought often to assume regular, integer values. Examples of such very structured sets of weights include the "human solution" to the "clump recognition" problem [2] discussed in section 2.2, and the configuration presented in reference [1] that solves the parity problem. The relatively low number of

correct solutions to any problem when weights are restricted to a few integers increases the probability that by finding one the network discovers real rules. Discrete weights can thus in certain instances improve the ability of a network to generalize.

Abandoning the continuum necessarily complicates the teaching problem; combinatorial minization is always harder than gradient descent. New techniques, however, have in certain cases mitigated the problem. Simulated annealing [3], and more recently, a mean-field approximation to it [4,5], are proving successful in a number of optimization tasks. In what follows, we apply the technique to the problem of determining the best set of weights in feed-forward neural networks.

For simplicity, we restrict ourselves here to networks with one hidden layer and a single output. A measure of network performance is given by the "energy"

$$E = \sum_{\alpha} [t^{\alpha} - O^{\alpha}]^2, \tag{1.1}$$

where

$$O^{\alpha} = g(-W_0 + \sum_i W_i V_i^{\alpha}), \tag{1.2}$$

and

$$V_i^{\alpha} = g(-T_{i0} + \sum_j T_{ij} I_j^{\alpha}). \tag{1.3}$$

Here g is a step function with threshold 0 and taking values $\{1, -1\}$, and the W_i and T_{ij} are weights for the output and hidden layers respectively; a subscript 0 indicates a threshold unit. The I_j^{α} are the input values for question α, and t^{α} is the target output value (answer) for that question. Back propagation works by taking derivatives of E with respect to the weights and following a path downhill on the E surface. Since the weights are restricted here to a few discrete values, this procedure is not applicable and finding minima of E, as noted above, becomes a combinatorial problem.

The approach we adopt is to sweep through the weights W_i and T_{ij} one at a time in cyclic order, presenting each with the training data and allowing it the option of changing its value to another (chosen at random) from the allowed set. The weight accepts the change with probability

$$P = \exp[-(E_{\text{new}} - E_{\text{old}})/T] \tag{1.4}$$

if $E_{\text{new}} > E_{\text{old}}$, and with probability $P = 1$ if the change decreases the energy. The parameter T is initially set at some relatively large value, and then decreased according to the prescription

$$T \to \gamma T, \ \gamma < 1, \tag{1.5}$$

after a fixed number of sweeps N_{sw}, during which time the system "equilibrates."

It is far from obvious that this annealing algorithm will prove a useful tool for minimizing E. Equations (1.1–1.3) specify a function very different from the energies associated with spin glasses or the Traveling Salesman Problem, systems to which annealing has been successfully applied. In the next section, we test the procedure on a few simple learning problems.

2. Examples

2.1 Parity

The XOR problem and its generalization to more inputs (the parity problem) have been widely studied as measures of the efficiency of training algorithms. While not "realistic," these problems elucidate some of the issues that arise in the application of annealing.

We consider first a network with two hidden units connected to the inputs, an output unit connected to the hidden layer, and threshold values for each unit. All the weights and thresholds may take the values -1, 1, or 0. We add an extra constraint term to the energy discouraging configurations in which the total input into some unit (for some question α) is exactly equal to the threshold for that unit.

When the temperature T is set to zero from the start, the annealing algorithm reduces to a kind of iterated improvement. Changes in the network are accepted only if they decrease the energy, or leave it unchanged. The XOR problem specified above has enough solutions that the algorithm will find one about half the time starting at $T = 0$. It takes, on the average, 100–300 presentations of the input data to find the minimum. If, on the other hand, annealing is performed, starting at $T_0 \approx 1$, iterating 20 times at each temperature, and cooling with $\gamma = .9$ after each set of iterations, a solution is always found after (on average) about 400 presentations. In this context, therefore, annealing offers no real advantage. Furthermore, by increasing the number of hidden units to four, we can increase the likelihood that the $T = 0$ network finds a solution to 90%. These results persist when we go from the XOR problem with two hidden units to the parity problem with four inputs and four hidden units (we now allow the thresholds to take all integer values between -3 and 3). There, by taking $T_0 = 3.3$ and $\gamma = .95$, the annealing algorithm always finds a solution, typically after about 15,000 presentations. However, the $T = 0$ net finds a solution once every 30 or 40 attempts, so that the work involved is comparable to or less than that needed to anneal. And as before, increasing the number of hidden units makes the $T = 0$ search even faster.

Neither of these points will retain its force when we move to more complicated, interesting, and realistic problems. When the ratio of solutions to total configurations becomes too small, iterated improvement techniques perform badly, and will not solve the problem in a reasonable amount of time. And increasing the number of hidden units is not a good remedy. The more

$n_i = 4$	$n_i = 5$	$n_i = 6$
2,400	1,012,500	$> 2,000,000$

Table 1: Average number of data-set presentations needed to find a zero-energy solution to one-vs.-two clumps problem by iterated improvement.

solutions available to the network, the less likely it is that any particular one will generalize well to data outside the set presented. We want to train the network by finding one of only few solutions, not by altering network structure to allow more.

2.2 Recognizing clumps — scaling behavior

To test our algorithms in more characteristic situations, we present the network with the problem of distinguishing two-or-more clumps of 1's in a binary string (of 1's and -1's) from one-or-fewer [2]. This predicate has order two, no matter the number of input units n_i. In what follows we always take the number of hidden units equal to the number of inputs, allow the weights to take values -1, 0, 1, and let the thresholds assume any integer value between $-(n_i - 1)$ and $n_i + 1$. By examining the performance of the network on the entire 2^{n_i} input strings for n_i equal to four, five, and six, we hope to get a rudimentary idea of how well the performance scales with the number of neurons.

To establish some kind of benchmark, we again discuss the $T = 0$ iterated improvement algorithm. For $n_i = 4$, the technique works very well, but slows considerably for $n_i = 5$. By the time the number of inputs reaches six, we are unable to find a zero-energy solution in many hours of VAX 11/780 time. These results are summarized in table 1, where we present the average number of question presentations needed to find a correct solution.

The full annealing algorithm is slower than iterated improvement for $n_i = 4$, but by $n_i = 5$, it is already a better approach. The average number of presentations for each n_i, with the annealing schedule adjusted so that the ground state is found at least 80% of the time, is shown in table 2. While it is not possible to reliably extrapolate to larger n_i (for $n_i = 7$, running times were already too long to gather statistics), it is clear that for these small values the learning takes substantially longer with each increment in the number of hidden units and corresponding doubling of the number of examples. One factor intrinsic to the algorithm partially accounts for the increase. The number of weights and thresholds in the network is $(n_i + 1)^2$. Since a sweep consists of changing each weight, it takes $(n_i+1)^2$ presentations to expose the entire network to the input data one time. But it is also clear that more sweeps and a slower annealing schedule are required as n_i increases. A better quantification of these tendencies awaits further study.

One interesting feature of the solutions found is that many of the weights allotted to the network are not used, that is they assume the value zero.

	$n_i = 4$	$n_i = 5$	$n_i = 6$
T_0	10	20	33
N_{sw}	40	40	40
γ	0.9	0.95	0.993
presentations	19,000	131,000	652,000

Table 2: Average number of presentations needed to anneal into a zero-energy solution in one-vs.-two clumps problem.

Figure 1 shows two typical solutions for $n_i = 6$. In the second of these, one of the output weights is zero, so that only five hidden units are utilized. While the network shows no inclination to settle into the very low order "human" solution of [2], its tendency to eliminate many allotted connections is encouraging, and could be increased by adding a term to the energy that penalizes large numbers of weights. An algorithm like back propagation that is designed for continuous-valued connections will certainly find solutions [2], but cannot be expected to set weights exactly to zero.

Is it possible that a modification in the procedure could speed convergence? Szu [6] has shown that for continuous functions a sampling less local than the usual one leads to quicker freezing. Our sampling here is extremely local; we change only one weight at a time, and it can therefore be difficult for the network to extract itself from false minima. An approach in which several weights are sometimes changed needs to be tried. This will certainly increase the time needed to simulate the effect of a network change on the energy. A hardware or parallel implementation, however, avoids the problem.

3. Mean field approach

One might hope to speed learning in other ways. In spin glasses [7], certain representative NP-complete combinatorial problems [4], and Boltzmann-machine learning [5], mean-field approximations to the Monte Carlo equilibration have resulted in good solutions while reducing computation time by an order of magnitude. In these calculations, the configurations are not explicitly varied, but rather an average value at any temperature is calculated (self-consistently) for each variable under the assumption that all the others are fixed at *their* average values. The procedure, which is quite similar to the Hopfield-net approach to combinatorial minimization detailed in reference [8], results in the equations

$$\langle X_i \rangle = \frac{\sum_{X_i} X_i \exp[-E(X_i, \langle X \rangle)/T]}{\sum_{X_i} \exp[-E(X_i, \langle X \rangle)/T]} \tag{3.1}$$

where X_i stands for the one of the weights W_i or $T_{i,j}$, $\langle\rangle$ denotes the average value, and the expression $\langle X \rangle$ means that all the weights except X_i are fixed at their average values. These equations are iterated at each temperature until self-consistent solutions for the weights are reached. At very high T,

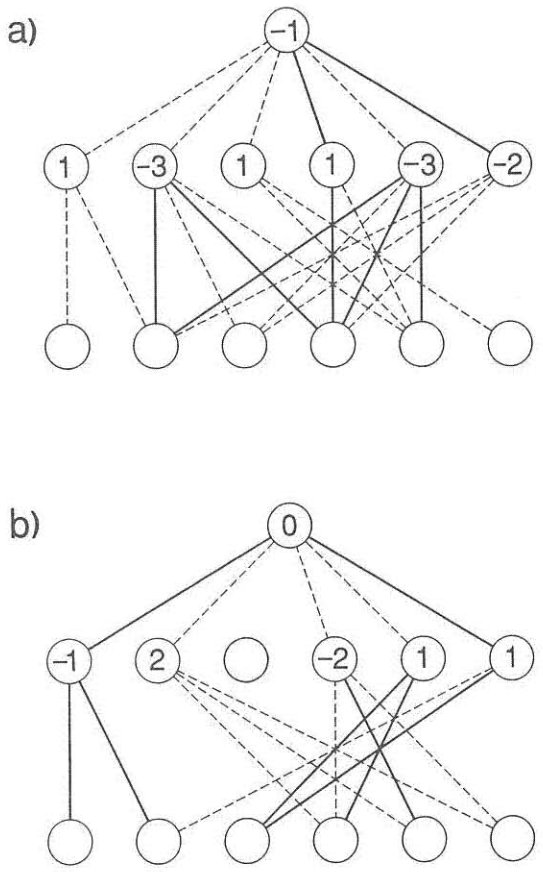

Figure 1: Typical solutions to the clump-recognition problem found by the simulated annealing algorithm. The black lines denote weights of strength $+1$ and the dashed lines weights of strength -1. Threshold values are displayed inside the circles corresponding to hidden and output neurons. Many of the weights are zero (absent), including one of the output weights in (b).

there is only one such solution: all weights equal to zero. By using the self-consistent solutions at a given T as starting points at the next temperature, we hope to reach a zero-energy network at $T = 0$.

Although the algorithm works well for simple tasks like the XOR, it runs into problems when extended to clump recognition. The difficulties arise from the discontinuities in the energy induced by the step function (threshold) g in equations (1.2) and (1.3); steep variations can cause the iteration procedure to fail. At certain temperatures, the self-consistent solutions either disappear or become unstable in our iteration scheme, and the equations never settle down. The problem can be avoided by replacing the step with a relatively shallow-sloped sigmoid function but then, because the weights must take discrete values, no zero-energy solution exists. By changing the output unit from a sigmoid function to a step (to determine right and wrong answers) *after* the teaching, we can obtain solutions in which the network makes 2 or 3 mistakes (in 56 examples) but none in which performance is perfect.

Some of these difficulties can conceivably be overcome. In an article on mean-field theory in spin glasses [9], Ling et al. introduce a more sophisticated iteration procedure that greatly improves the chances of convergence. Their method, however, seems to require the reformulation of our problem in terms of neurons that can take only two values (on or off), and entails a significant amount of matrix diagonalization, making it less practical as the size of the net is increased. Scaling difficulties plague every other teaching algorithm as well though, so while mean-field techniques do not at first sight offer clear improvement over full-scale annealing, they do deserve continued investigation.

4. Conclusions

We have shown that simulated annealing can be used to train discrete-valued weights; questions concerning scaling and generalization, touched on briefly here, remain to be explored in more detail. How well will the procedure perform in truly large-scale problems? The results presented here do not inspire excitement, but the algorithm is to a certain extent parallelizable, has already been implemented in hardware in a different context [10], and can be modified to sample far-away configurations. How well do the discrete-weight nets generalize? Is our intuition about integers and rules really correct? A systematic study of these all these issues is still needed.

References

[1] D. Rummelhart and J. McClelland, *Parallel Distributed Processing; Explorations in the Microstructure of Cognition*, (Cambridge: MIT Press, 1986).

[2] J. Denker, et al., *Complex Systems*, **1** (1987) 877.

[3] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, *Science*, **220** (1983) 671.

[4] G. Bilbro, et al., North Carolina State preprint (1988).

[5] C. Peterson and J.R. Anderson, *Complex Systems*, **1** (1987) 995. .

[6] H. Szu, *Physics Letters*, **122** (1987) 157.

[7] C.M. Soukoulis, G.S. Grest, and K. Levin, *Physcal Review Letters*, **50** (1983) 80.

[8] J.J. Hopfield and D.W. Tank, *Biological Cybernetics*, **52** (1985) 141.

[9] David D. Ling, David R. Bowman, and K. Levin, *Physical Review*, **B28** (1983) 262.

[10] J. Alspector and R.B. Allen, *Advanced Research in VLSI*, ed. Losleber (Cambridge: MIT Press, 1987).