BACKPROPAGATION IN PERCEPTRONS WITH FEEDBACK

Luís B. Almeida
R. Alves Redol, 9-2
P-1000 Lisboa
Portugal

ABSTRACT

Backpropagation has shown to be an efficient learning rule for graded perceptrons. However, as initially introduced, it was limited to feedforward structures. Extension of backpropagation to systems with feedback was done by this author, in [4]. In this paper, this extension is presented, and the error propagation circuit is interpreted as the transpose of the linearized perceptron network. The error propagation network is shown to always be stable during training, and a sufficient condition for the stability of the perceptron network is derived. Finally, potentially useful relationships with Hopfield networks and Boltzmann machines are discussed.

1. INTRODUCTION

Backpropagation has been independently introduced by several authors (including at least Parker, Le Cun, and Rummelhart, Hinton and Williams), as a learning rule for feedforward multilayer graded perceptron networks [1]. Its power is by now well demonstrated (see [2] for an example). It is based on the minimization of the squared error of the actual output, relative to a desired output, this minimization being performed through a gradient descent technique. Its extension to a special class of perceptrons with feedback was made in [1], followiong a suggestion by Minsky and Papert [3]. This class of perceptrons is characterized by the (implicit) assumption of the existence of a sample-and-hold operation at the output of each unit, all sample-and-holds being triggered synchronously. Under this assumption, the perceptron with feedback can be "unfolded" in time, into an equivalent feedforward one, and can therefore be trained using backpropagation. An important limitation of backpropagation in this context, however, is

R. Eckmiller et al. (eds.), *Neural Computers*

that it demands the existence of an essentially unlimited amount of memory in each unit.

In this paper, we will be concerned with a different class of feedback perceptrons: they will be assumed not to have any sample-and-hold; instead, for each input pattern, the outputs of the units will change continuously in time until a stable state is reached. The outputs of the perceptron are observed only in the stable state, and are then compared to the desired outputs. Training, i.e., weight update, is performed with the system in the stable state. The input-output mapping to be learned by the perceptron is assumed to be combinatorial, i.e., the desired outputs depend only on present inputs, not on past ones.

The extension of backpropagation to this class of perceptrons was first made by this author, in [4]. Here, we will review its derivation, and we will briefly discuss the problem of stability. We will then proceed to discuss the relationships between feedback perceptrons and Hopfield networks and Boltzmann machines.

2. BACKPROPAGATION IN FEEDBACK PERCEPTRONS

Consider a graded perceptron network, and designate by x_k the external inputs ($k = 1,...,K$), by y_i the outputs of the units ($i = 1,...,N$), by s_i the result of the sum performed at the input of unit i, and by o_p the external outputs ($p \in O$, where O is the set of units producing external outputs). The static equations of the perceptron network are

$$s_i = \sum_{n=1}^{N} a_{ni}\, y_n + \sum_{k=1}^{K} b_{ki}\, x_k + c_i \qquad i = 1,...,N \qquad (1)$$

$$y_i = S_i(s_i) \qquad i = 1,...,N \qquad (2)$$

$$o_p = y_p \qquad p \in O \qquad (3)$$

where a_{ni} and b_{ki} are weights, c_i is a bias term, and S_i is the nonlinear function in unit i (usually a sigmoid). In a feedforward perceptron, the units can be numbered in such a way that the array $[a_{ni}]$ is lower triangular, with zeros in the main diagonal. Note that in the nomenclature used in this paper, we do not consider external inputs as units.

Equations (1-3) are the equations of the equilibrium states of the network, for a given input pattern (or vector) $\mathbf{x} = [x_k]$. If we linearize the network around an equilibrium state, we obtain the network

$$s'_i = \sum_{n=1}^{N} a_{ni}\, y'_n + \sum_{k=1}^{K} b_{ki}\, x'_k \qquad i = 1,\ldots,N \qquad (4)$$

$$y'_i = D_i(s_i)\, s'_i \qquad i = 1,\ldots,N \qquad (5)$$

$$o'_p = y'_p \qquad p \in O \qquad (6)$$

where the primes denote the variables of the linearized system, and D_i is the derivative of S_i. Note that, in terms of the linearized network, $D_i(s_i)$ is just a constant coefficient. Transposing [5] this network, we obtain (using double-primed variables for the transposed network)

$$y''_i = \begin{cases} \sum_{n=1}^{N} a_{in}\, s''_n + o''_i & \text{if } i \in O \\ \sum_{n=1}^{N} a_{in}\, s''_n & \text{if } i \notin O \end{cases} \qquad (7)$$

$$s''_i = D_i(s_i)\, y''_i \qquad i = 1,\ldots,N \qquad (8)$$

$$x''_k = \sum_{n=i}^{N} b_{kn}\, s''_n \qquad k = 1,\ldots,K \qquad (9)$$

It is easy to see that, in the case of a feedforward perceptron, this is also a feedforward network, though it propagates in the reverse direction. In fact, it is exactly the backward error propagation network, as one can check by comparing these equations with those given in [1]. We shall now show that this fact extends to feedback perceptrons: the transpose of the linearized perceptron network is always the adequate network for error propagation. For this proof, let us first define the error at output p

$$e_p = o_p - d_p \qquad p \in O \qquad (10)$$

and the total quadratic error

$$E = \sum_{p \in O} e_p^2 \qquad (11)$$

Taking the partial derivative relative to weight a_{qr}

$$\dot{E} = \sum_{p \epsilon O} \frac{\partial E}{\partial o_p} \dot{o}_p = 2 \sum_{p \epsilon O} e_p \dot{o}_p \qquad (12)$$

where the dots denote derivatives relative to a_{qr}. If we differentiate equations (1-3) relative to a_{qr}, we obtain

$$\dot{s}_i = \begin{cases} \sum_{n=1}^{N} a_{ni} \dot{y}_i + y_q & \text{if } i = r \\ \sum_{n=1}^{N} a_{ni} \dot{y}_i & \text{if } i \neq r \end{cases} \qquad (13)$$

$$\dot{y}_i = D_i(s_i) \dot{s}_i \qquad (14)$$

$$\dot{o}_p = \dot{y}_p \qquad (15)$$

These equations are those of the linearized perceptron network (eqs. 4-6) with a single input y_q applied to node s'_r with a unit weight. Since that network is linear, we can write

$$\dot{o}_p = y_q t'_{rp} \qquad p \epsilon O, \quad q,r = 1,...,N \qquad (16)$$

where t'_{rp} is the transfer ratio from node s'_r to the output o_p. But, from the transposition theorem [5], that transfer ratio is equal to the transfer ratio from o''_p to s''_n in the transposed network. Therefore,

$$\dot{o}_p = y_q t''_{pr} \qquad p \epsilon O, \quad q,r = 1,...,N \qquad (17)$$

which we can replace in equation (12), obtaining

$$\dot{E} = 2 y_q \sum_{p \epsilon O} e_p t''_{pr} \qquad q,r = 1,...,N \qquad (18)$$

and the sum in the right hand side is the value that will be obtained at node s''_r, in the transposed network, if we apply the errors e_p at the outputs o''_p:

$$\frac{\partial E}{\partial a_{qr}} = 2 y_q s''_r \qquad q,r = 1,...,N \qquad (19)$$

which is just the result we wanted to obtain: the update of weight a_{qr} is proportional to the output of unit q multiplied by the result of propagating the error(s) through the transposed linearized network. Similar derivations can also be made for the weights b_{ki} and the bias terms c_i. The only assumption is that the functions S_i must be differentiable. Figure 1 shows an example of a feedback perceptron and of the corresponding error propagation network. For example, we have for this figure $\partial E/\partial c = 2\ y_2\ v_1$ and $\partial E/\partial h = 2\ y_1\ v_2$.

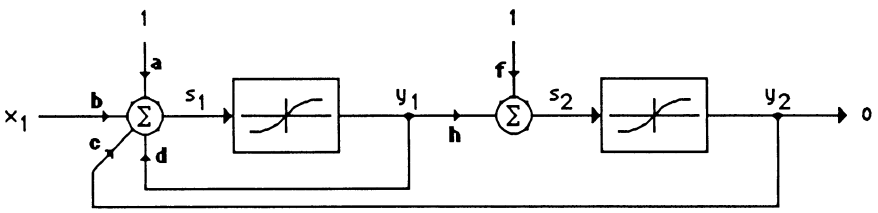

Figure 1a - A simple perceptron with feedback. Bold characters indicate weights, light characters indicate network variables.

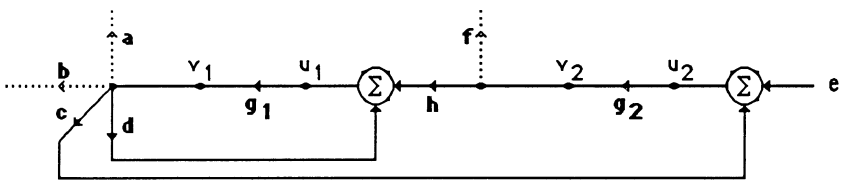

Figure 1b - The error propagation network corresponding to figure 1a. In this figure, $g_i=D_i(s_i)$. Dotted parts belong to the transposed linearized network, but are not needed for error propagation.

3. STABILITY

In feedback perceptrons, the error propagation network also has feedback, and thus we have to study its stability, since an unstable error propagation network would be useless. However, it is a well known result from the theory of dynamical systems that the stability of a nonlinear system at an equilibrium state is equivalent to the stability of the system obtained through linearization around that state (except in very infrequent marginal situations [6]). Furthermore, the stability of

a linear system is equivalent to the stability of its transpose. Therefore, since we have assumed that training is performed with the perceptron network at a stable state, the backward error propagation network will also be stable. Note, however, that the error propagation network must be the transpose of the linearized perceptron not only in static, but also in dynamical terms: the dynamical properties of the two networks must be matched.

Another issue is the problem of whether the perceptron network itself is stable, so that it can be used, and trained, as described above. The stability of the perceptron does not depend only on its static equations (1–3), but also on the dynamical behavior of its units. Figure 2 depicts dynamical behaviors that are commonly assumed for neural network units. The uppermost circuit comes from considerations on the dynamical behavior of actual neurons [7], while the lower one corresponds to a plausible dynamical behavior of electronic implementations.

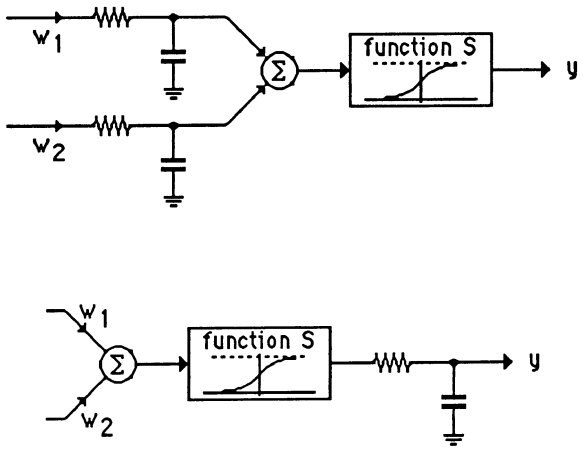

Figure 2 - Two circuits corresponding to dynamical behaviors often assumed for perceptron units. Resistor and capacitor values may differ from unit to unit, or from branch to branch.

Networks with both kinds of units have been shown to be stable if the weights are symmetrical, i.e., if $a_{in} = a_{ni}$, for all i,n, and the functions S_i are monotonically increasing and bounded. The proof was given in [7] for

networks of the upper kind, and in [4] for those of the lower kind. Actually, the sufficient condition for stability obtained in [4] is somewhat broader than weight symmetry: it is that there exist positive coefficients μ_i such that

$$\mu_i \, a_{ni} = \mu_n \, a_{in} \qquad i,n = 1,...,N \qquad (20)$$

and the proof of this condition would be easy to extend to the networks of the upper kind.

These proofs are based on the use of a so-called "energy function"

$$W = -\frac{1}{2} \sum_{n=1}^{N} \sum_{i=1}^{N} \mu_i \, a_{ni} \, y_n \, y_i - \sum_{k=1}^{K} \sum_{i=1}^{N} \mu_i \, b_{ki} \, x_k \, y_i -$$

$$- \sum_{i=1}^{N} \mu_i \, c_i \, y_i + \sum_{i=1}^{N} \mu_i \, U_i(y_i) \qquad (21)$$

where U_i is a primitive of S_i^{-1} (the inverse of S_i). What is actually done, in both cases, is to show that this energy function always decreases in time, with the dynamical behavior of the perceptron network, and therefore that it cannot oscillate, and must stop at some stable point, corresponding to a local minimum of W.

It should be noted, however, that experimental tests performed by the author have led him to suggest that unstable situations are encountered only very infrequently, even when the condition (20) is not enforced. On the other hand, it is easy to see that that condition is still too restrictive: feedforward perceptrons are always stable, but they do not obey this condition, in general.

A third problem concerning stability is the possibility of there being multiple stable states for the same input pattern. An intuitive reasoning was given in [4], suggesting that this probably is not a serious problem, and this seems to be confirmed by the tests performed so far.

4. EXPERIMENTAL RESULTS

A number of experimental tests were performed on feedback perceptrons, some of which are described in [4] They will not be given

here for lack of space). These experiments included pattern completion, a kind of problem for which feedforward perceptrons seem to be quite unsuited. Though still few in number, those tests apparently point to some conclusions:

- Feedback perceptrons seem to have advantages over feedforward ones in some situations (including pattern completion), but in other cases their advantage may be only marginal.

- Unstable situations are encountered only very infrequently, when no measures are taken to ensure stability.

- Weight symmetry, as a sufficient condition for stability, does not seem to strongly impair the performance of feedback perceptrons.

5. FEEDBACK PERCEPTRONS AND HOPFIELD NETWORKS

As was emphasized above, training of feedback perceptrons by backpropagation is done in the stable states, and therefore back-propagation can be viewed as a means to "move" the stable states toward desired positions (if the network contains hidden units, i.e. units that do not directly produce outputs, backpropagation will move the stable states toward desired subspaces of the state space).

If we impose weight symmetry, force the "self-feedback" weights a_{ii} to be zero and do not allow any hidden units, feedback perceptrons become formally equivalent to Hopfield networks with graded neurons [7]. Therefore, backpropagation can be viewed as a learning rule for graded Hopfield networks (Hopfield's learning rule is for networks of binary units [8]). Graded networks are the natural choice for representing patterns with analog valued features.

On the other hand, backpropagation can be used in networks with hidden units, thus eliminating one of the basic limitations of Hopfield networks, allowing them to express more complex dependencies among pattern features. Backpropagation also does not require the weights to be symmetrical, though stability cannot be guaranteed if condition (20) is not satisfied.

Finally, backpropagation does not impose any limitation on the patterns to be stored, and therefore eliminates the restriction that they should be approximately orthogonal to each other [8].

6. FEEDBACK PERCEPTRONS AND BOLTZMANN MACHINES

Let us again consider a feedback perceptron with symmetrical weights and null "self-feedback" weights. If we let the sigmoids S_i approach step functions, the energy W given in equation (21) approaches, in the limit, the energy function of Boltzmann machines [1,4]. Therefore, a feedback perceptron with steep sigmoids has approximately the same energy minima as a Boltzmann machine with the same weights. Backpropagation can thus, very probably, be used to train Boltzmann machines, i.e., to adapt their weights in such a way that they have a minimum of the energy function at the desired location, for each input pattern. In this context, the following comments may be appropriate:

- Backpropagation, if used for the initial training of a Boltzmann machine, may be faster than the standard Boltzmann machine training, due to its deterministic character.

- Initial training by backpropagation may need to be refined by standard Boltzmann machine training, mainly because the energy function of the graded perceptron network is not exactly equal to the one of the Boltzmann machine (though it may be as close as desired).

- Backpropagation does not guarantee the existence of a global minimum at the desired location, since it treats all minima equally. However, it tends to move all local minima toward that location, and thus it probably will end up yielding an energy function with a global minimum at the desired site, and with fewer local minima. If so, the resulting Boltzmann machine can use a faster cooling schedule, and will therefore run faster.

7. CONCLUSIONS

The backpropagation learning rule extends to nonfeedforward perceptrons in a very natural way, as was first shown in [4]. The error

propagation network can be viewed as the transpose of the linearized perceptron network, and is always stable when training is performed.

The stability of the feedback perceptron network can be guaranteed by means of a condition on the weights, which does not seem to significantly restrict its capabilities. On the other hand, if this condition is not imposed, unstable situations seem to arise only very infrequently.

Close relationships exist between feedback perceptrons, Hopfield networks and Boltzmann machines. Backpropagation can presumably be used for the training of graded Hopfield networks (with hidden units if desired), and may also exhibit some advantages if used for the training of Boltzmann machines.

REFERENCES

1. D. Rumelhart, J. McClelland and the PDP Research Group, eds., "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", Cambridge, MA: MIT Press 1986.
2. T. Sejnowski and C. Rosenberg, NETtalk: A Parallel Network that Learns to Read Aloud", technical report JHU/EECS-86-01, The Johns Hopkins University, 1986.
3. M. Minsky and S. Papert, "Perceptrons", Cambridge, MA: MIT Press, 1969.
4. L. Almeida, "A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment", Proceedings of the 1987 IEEE First Annual International Conference on Neural Networks, S. Diego, CA, June 1987.
5. A. Oppenheim and R. Schafer, "Digital Signal Processing", Englewood Cliffs, NJ: Prentice-Hall, 1975.
6. J. Willems, "Stability Theory of Dynamical Systems, London: Thomas Nelson and Sons Ltd., 1970.
7. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons", Proc. Nat. Acad. Sci. USA, vol. 81, pp. 3088-3092, May 1984.
8. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities, Proc. Nat. Acad. Sci. USA, vol. 79, pp. 141-152, 1985.