J. Phys. A: Math. Gen. 22 (1989) L711-L717. Printed in the UK

LETTER TO THE EDITOR

# Optimal learning in neural network memories†

L F Abbott and Thomas B Kepler

Physics Department, Brandeis University, Waltham, MA 02254, USA

Received 12 May 1988

**Abstract.** We examine general learning procedures for neural network associative memories and find algorithms which optimise convergence.

A neural network memory uses fixed points of the map

$$S_i(t+1) = \operatorname{sgn}\left(\sum_{j=1}^{N} J_{ij} S_j(t)\right). \tag{1}$$

(where $S_i = \pm 1$ and $J_{ii} = 0$) as memory patterns which attract nearby input patterns providing associative recall. The dynamics (1) takes an initial input $S_i(0)$ and after a sufficient number of iterations maps it to an associated memory pattern $\xi_i$ provided that $\xi_i$ is a fixed point of (1) and that $S_i(0)$ lies within the domain of attraction of this fixed point. Learning in such a network is a process by which a matrix $J_{ij}$ is constructed with the appropriate fixed points and required basins of attraction. Suppose we wish to 'learn' a set of memory patterns $\xi_i^\mu$ with $\mu = 1, 2, \ldots, \alpha N$. Important variables for characterising a fixed point are

$$\gamma_i^\mu = \frac{1}{\|J_i\|} \sum_{j=1}^{N} J_{ij} \xi_i^\mu \xi_j^\mu \tag{2}$$

where the normalisation factor $\|J_i\|$ is

$$\|J_i\| = \left(\sum_{j=1}^{N} J_{ij}^2\right)^{1/2}. \tag{3}$$

In order for $\xi_i^\mu$ to be a stable memory pattern of the network, $\gamma_i^\mu$ must be positive for all $i$. In addition the distribution of $\gamma_i^\mu$ values has a great impact on the size of the basin of attraction [1] associated with $\xi_i^\mu$.

A standard learning problem is to find a matrix $J_{ij}$ satisfying

$$\gamma_i^\mu > \kappa \tag{4}$$

for all $i$ and all $\mu$ with some specified value of $\kappa$. Gardner [2] has computed the range of $\alpha$ and $\kappa$ values for which matrices satisfying (4) exist. The problem remains to find efficient algorithms for constructing such matrices.

† Research supported by Department of Energy Contract AC02-ER0320 and by the US-Israel Binational Science Foundation.

The standard method [2, 4] for finding a matrix satisfying (4) is to start with a random matrix and repeatedly apply the learning rule

$$J_{ij} \rightarrow J_{ij} + \frac{1}{N} \xi_i^\mu \xi_j^\mu (1-\delta_{ij}) \theta(\kappa - \gamma_i^\mu) \tag{5}$$

at each site $i$ and for each memory pattern $\mu$ until (4) is satisfied. It has been proven [2, 4] that this algorithm will converge in a finite number of steps if the desired matrix exists. However, in actual practice the standard algorithm is extremely slow.

There are several reasons for believing that (5) is not a particularly efficient learning algorithm. The step size (that is, the magnitude of the term being added to the original matrix) in (5) is fixed, independent of the normalisation of $J_{ij}$ and of the difference between the actual value of $\gamma_i^\mu$ and the desired value $\kappa$. If the initial matrix has an enormous magnitude $\|J_i\|$ the algorithm prescribes the same step size as if $\|J_i\|$ were tiny. This seems inefficient. In addition, a better strategy might be to adjust the step size so that it is larger if $\gamma_i^\mu \ll \kappa$ and smaller if $\gamma_i^\mu$ is near to $\kappa$. The first issue can be resolved by multiplying the step size by the normalisation factor $\|J_i\|$ but the second is more subtle. Let us therefore consider learning algorithms of the form

$$J_{ij} \rightarrow J_{ij} + \frac{1}{N} \xi_i^\mu \xi_j^\mu (1-\delta_{ij}) f(\gamma_i^\mu) \|J_i\| \theta(\kappa - \gamma_i^\mu) \tag{6}$$

where once again the learning process consists of applying this algorithm at every site and for every memory pattern until (4) is satisfied. We wish to investigate what function $f(\gamma)$ optimises this learning process. The idea of using a variable step size in a related context has been considered before [3]. However, we find that it is the combination of a variable step size and the normalisation correction in the above formula (which is new) which produces the dramatic improvement in learning times which we shall see below.

To study the convergence properties of the algorithm (6) we follow general methods from perceptron proofs [2, 4]. Suppose there exists a matrix $J_{ij}^*$ satisfying

$$\sum_{j=1}^{N} J_{ij}^* \xi_i^\mu \xi_j^\mu > (\kappa + \delta) \|J_i^*\| \tag{7}$$

for all $i$ and $\mu$. Following Gardner [2] we introduce the notation

$$(J \cdot J^*) = \sum_{j=1}^{N} J_{ij} J_{ij}^* \tag{8}$$

and consider the quantity

$$X_i = \frac{(J \cdot J^*)}{\|J_i\| \; \|J_i^*\|} \leqslant 1 \tag{9}$$

which is bounded by the Schwartz inequality. To analyse the convergence properties of the various algorithms characterised by (6) we will calculate how much $X_i$ changes every time the matrix $J_{ij}$ is updated. If this change is greater than a fixed positive number, the algorithm must converge at least by the time all of the $X_i$ have reached the value 1. Furthermore, the larger the change in $X_i$ the shorter will be the maximum possible convergence time.

Let $J_{ij}^m$ be the value of the matrix after $m$ non-trivial applications of the learning rule (6). From (6) we see that after an $(m+1)$th application of the rule occurring at the site $i$, an update using the memory pattern $\xi_i^\mu$ produces

$$(J^{m+1}\cdot J^*)_i=(J^m\cdot J^*)_i+\frac{1}{N}\sum_{j=1}^{N} J_{ij}^*\xi_i^\mu\xi_j^\mu f(\gamma_i^\mu)\|J_i^m\|. \tag{10}$$

Here $\gamma_i^\mu$ corresponds to (2) with $J_{ij}=J_{ij}^m$. Using (7) we find

$$(J^{m+1}\cdot J^*)_i>(J^m\cdot J^*)_i+\frac{1}{N}(\kappa+\delta)\|J_i^*\|f(\gamma_i^\mu)\|J_i^m\|. \tag{11}$$

From the definition (9), we see that the value of $X_i$ after $m+1$ iterations satisfies

$$X_i^{m+1}>\frac{\|J_i^m\|}{\|J_i^{m+1}\|}\left(X_i^m+\frac{1}{N}(\kappa+\delta)f(\gamma_i^\mu)\right). \tag{12}$$

Now from (6)

$$\|J_i^{m+1}\|^2=\|J_i^m\|^2\left\{1+\frac{1}{N}\left[\left(1-\frac{1}{N}\right)f^2(\gamma_i^\mu)+2\gamma_i^\mu f(\gamma_i^\mu)\right]\right\}. \tag{13}$$

Thus, ignoring terms of order $1/N^2$ we have

$$\frac{\|J_i^m\|}{\|J_i^{m+1}\|}=1-\frac{1}{N}\left(\frac{1}{2}f^2(\gamma_i^\mu)+\gamma_i^\mu f(\gamma_i^\mu)\right). \tag{14}$$

Putting this together with (12) gives to order $1/N$,

$$X_i^{m+1}>X_i^m\left[1-\frac{1}{N}\left(\frac{1}{2}f^2(\gamma_i^\mu)+\gamma_i^\mu f(\gamma_i^\mu)\right)\right]+\frac{1}{N}(\kappa+\delta)f(\gamma_i^\mu). \tag{15}$$

Recalling that $X_i\leqslant 1$ we find for $f^2/2+\gamma_i^\mu f>0$

$$X_i^{m+1}>X_i^m+\frac{1}{N}[(\kappa+\delta-\gamma_i^\mu)f(\gamma_i^\mu)-\tfrac{1}{2}f^2(\gamma_i^\mu)]. \tag{16}$$

To treat the case $f^2/2+\gamma_i^\mu f\leqslant 0$ we will assume $X_i^m\geqslant 0$. Since under the conditions we will derive $X_i$ will always grow, $X_i^m$ will be greater than zero if $X_i^0\geqslant 0$ and this is trivial to achieve. If $J_{ij}^0$ makes $X_i^0<0$ then the transformation $J_{ij}^0\to -J_{ij}^0$ will remedy the situation. Thus for $f^2/2+\gamma_i^\mu f\leqslant 0$ with $X_i^m\geqslant 0$ we have

$$X_i^{m+1}>X_i^m+\frac{1}{N}((\kappa+\delta)f(\gamma_i^\mu)). \tag{17}$$

The condition assuring that $X_i$ will grow every time the rule (6) is applied (non-trivially) is then $f>0$ and

$$(\kappa+\delta-\gamma_i^\mu)f(\gamma_i^\mu)-\tfrac{1}{2}f^2(\gamma_i^\mu)>0. \tag{18}$$

In order for $X_i$ to remain smaller than 1, the learning algorithm (6) must converge in a finite number of steps for any positive $f(\gamma)$ satisfying (18).

From the above derivation we see that any function satisfying

$$0<f(\gamma)<2(\kappa+\delta-\gamma) \tag{19}$$

will lead to convergent learning. How do we find the function $f(\gamma)$ within this allowed range which optimises the learning process? One way is to choose $f(\gamma)$ so that the change in $X_i$ is as large as possible for every learning step. This is done by maximising the expression on the left-hand side of (18) with respect to the function $f$ at each point $\gamma_i^\mu$ when $f^2/2+\gamma_i^\mu f>0$ and maximising $f$ itself when $f^2/2+\gamma_i^\mu f\leq 0$ giving the result

$$f(\gamma)=(\kappa+\delta-\gamma)\theta(\kappa+\delta+\gamma)-2\gamma\theta(-\kappa-\delta-\gamma). \tag{20}$$

Despite the presence of the $\theta$ functions we will refer to this as the linear rule. The minimum change in $X_i$ for this functional form occurs at $\gamma_i^\mu=\kappa$ and is bounded by

$$X_i^{m+1}>X_i^m+\frac{\delta^2}{2N}. \tag{21}$$

Thus the algorithm (6) with $f$ given by the linear rule (20) will converge after less than $2N/\delta^2$ applications (assuming we start at $X_i^0=0$).

Since the convergence time in the above approach is dominated by the case $\gamma_i^\mu=\kappa$ where $X_i^{m+1}>X_i^m+\delta^2/2N$ it is not clear that there is much benefit in making $X_i$ change by any larger amount for other $\gamma$ values. A better approach might be to keep the minimum change in $X_i$ fixed at $\delta^2/2N$ with the advantage that in general the function $f$ which determines how quickly the matrix $J_{ij}$ changes will be larger. The maximum possible function $f$ in this case is given by

$$(\kappa+\delta-\gamma_i^\mu)f(\gamma_i^\mu)-\tfrac{1}{2}f^2(\gamma_i^\mu)=\tfrac{1}{2}\delta^2 \tag{22}$$

giving

$$f(\gamma)=\kappa+\delta-\gamma+[(\kappa+\delta-\gamma)^2-\delta^2]^{1/2}. \tag{23}$$

We will refer to this form as the non-linear rule. As we will see this algorithm produces considerably better results than the linear rule and both (20) and (23) are a dramatic improvement over the standard algorithm. In practice, we find that the non-linear rule works best when a small value of $\delta$ is used. In this case the non-linear rule is essentially identical to the upper limit of the inequality (19) so it is equivalent to maximising the step size in $J$ while the linear rule maximises the step size in $X$.

The results of computer simulations using the standard algorithm as well as our linear and non-linear rules are shown in figures 1-4. Figures 1-3 compare the learning times for various values of $\alpha$ and $\kappa$. These figures show the difference between $\kappa$ and the minimum value (worst case) of $\gamma$ for an $N=100$ node network plotted against the number of learning passes. A pass consists of one sweep through all $N$ sites and through all $\alpha N$ memory patterns applying the particular learning algorithm. The algorithms converge of course when $\kappa$ minus the minimum value of $\gamma$ is negative. Because we found it more informative to use a logarithmic scale on figures 1-3 this point is not visible on the plots. However, in all the cases shown the algorithms converged when the curves drawn intersect the bottom of the graphs. All the learning procedures shown in these graphs converged except for the case of the standard algorithm in figure 3 where our patience and that of our computer centre ran out. The convergence times were as follows: for $\alpha=0.25$ and $\kappa=1.44$ the standard algorithm converged in 175 passes, the linear rule in 53 passes and the non-linear rule in 18; for $\alpha=0.75$ and $\kappa=0.42$ the results were standard 325 passes, linear 210 and non-linear 32; and for $\alpha=1.5$ and $\kappa=0.04$ the standard algorithm did *not* converge in 800 passes while the linear and non-linear rules converged in 157 and 53 passes respectively. Throughout we took $\delta=0.01$ which seemed to be a reasonably optimal value. In all cases the linear and non-linear rules are much faster than the standard algorithm. The non-linear rule provides the fastest convergence times but, as indicated by its rapid

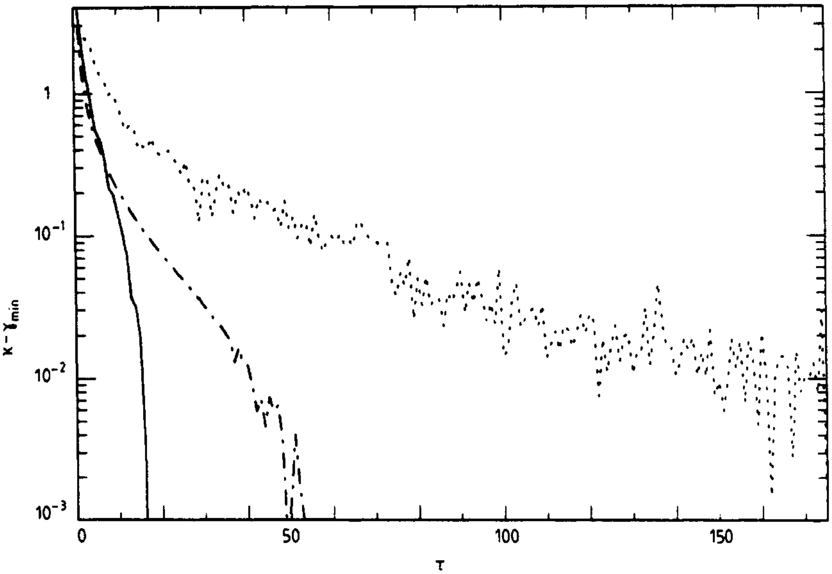

**Figure 1.** The value of $\kappa$ minus the minimum value of $\gamma_i^{\mu}$ for an $N=100$ network as a function of the number of passes through all the network sites and all the memory patterns being learned. Here $\alpha=0.25$, $\kappa=1.44$ and $\delta=0.01$. The full line is the non-linear algorithm, the chain line is the linear rule and the dotted line is the standard algorithm.

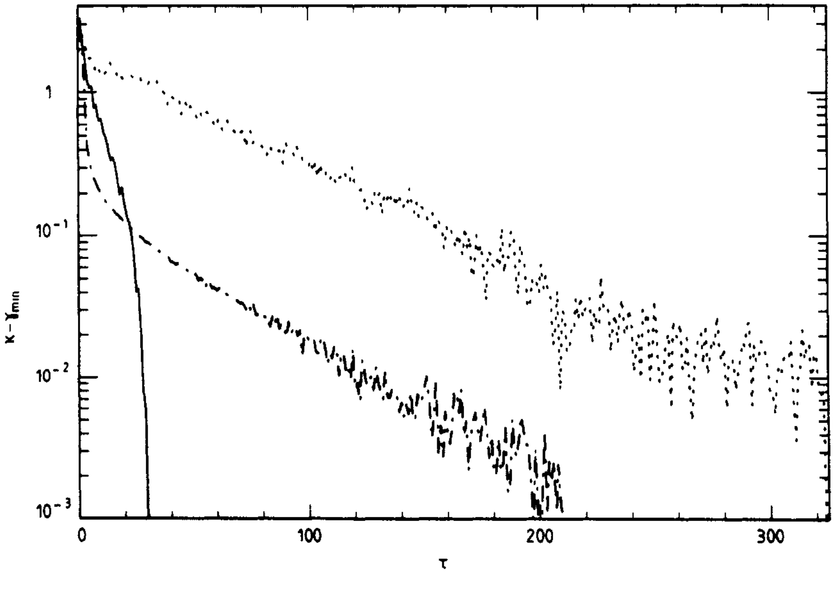

**Figure 2.** Same as figure 1 with $\alpha=0.75$, $\kappa=0.42$ and $\delta=0.01$.

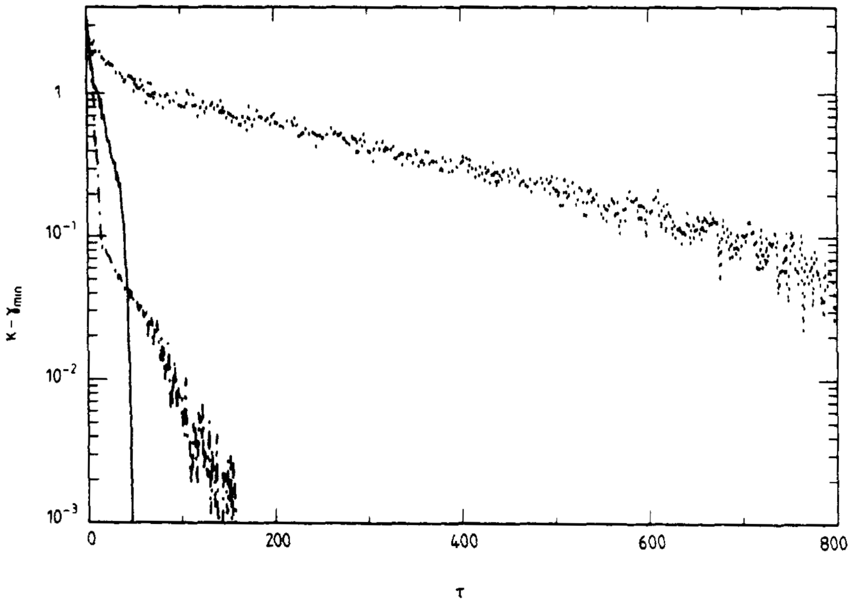

**Figure 3.** Same as figure 1 with $\alpha = 1.5$, $\kappa = 0.04$ and $\delta = 0.01$.

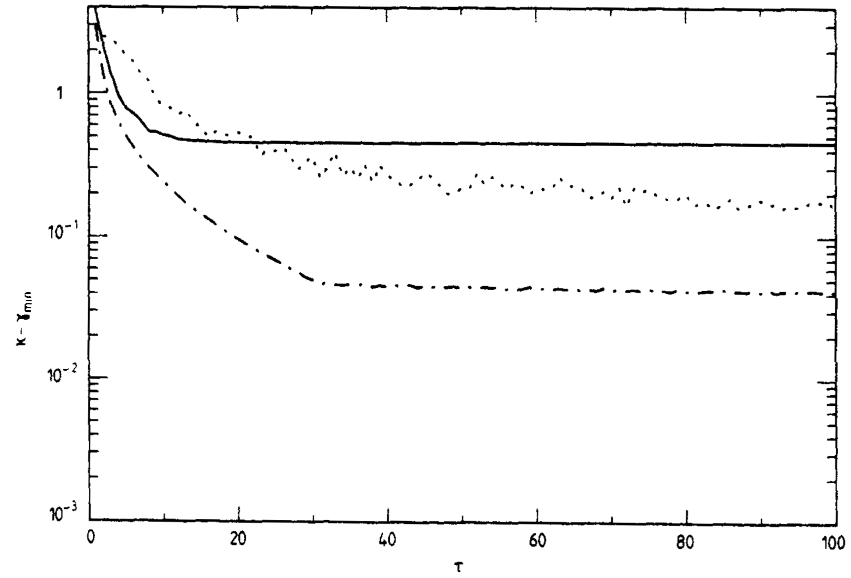

**Figure 4.** Same as figure 1 with $\alpha = 0.25$, $\kappa = 1.46$ and $\delta = 0.01$. In this case none of the algorithms converged because no matrix with the desired properties exists.

initial drop in the figures, the linear rule is the quickest at stabilising the memory patterns.

Figure 4 shows the behaviour of the three algorithms in a situation where they do not converge because $\kappa$ is given too large a value and thus no matrix with the desired properties exists. Here we took $\alpha = 0.25$ and $\kappa = 1.46$. We have verified using the Krauth and Mézard [4] variant of the standard algorithm that no such matrix existed for the memories we used. This value is less than the Gardner [2] bound because of finite-$N$ effects even for $N = 100$. In cases such as this where none of the methods converges the linear rule seems to produce the best possible matrix in the least amount of time while the non-linear rule rapidly stops improving and goes into a steady non-convergent behaviour thus providing the most rapid identification of a larning task that will not converge.

In conclusion, the linear and non-linear algorithms based on (6) (especially the non-linear rule) provide extremely efficient methods for constructing a coupling matrix with desired properties for a neural network memory. They represent a dramatic improvement over previous methods. For simplicity we have worked in the limit of large $N$; however, this is not essential. Equations (20) and (23) can be rederived in a straightforward manner keeping all terms of higher order in $N$ if necessary.

We wish to thank I Yekutieli and H Gutfreund for helpful comments.

## References

[1] Kepler T and Abbott L F 1988 *J. Physique* **49** 1657
Krauth W, Nadal J-P and Mézard M 1988 *J. Phys. A: Math. Gen.* **21** 2995
[2] Gardner E 1988 *J. Phys. A: Math. Gen.* **21** 257
[3] Agmon S 1954 *Can. J. Math.* **6** 382
[4] Minsky M and Papert S 1969 *Perceptrons* (Cambridge, MA: MIT Press)
Gardner E, Stroud N and Wallace D 1989 *J. Phys. A: Math. Gen.* **22** 2019
Poeppel G and Krey U 1987 *Europhys. Lett.* **4** 481
Diederich S and Opper M 1987 *Phys. Rev. Lett.* **58** 949
Krauth W and Mézard M 1987 *J. Phys. A: Math. Gen.* **20** L745