Training with noise and the storage of correlated patterns in a neural network model

1989 J. Phys. A: Math. Gen. 22 2019

(http://iopscience.iop.org/0305-4470/22/12/007)

J. Phys. A: Math. Gen. **22** (1989) 2019-2030. Printed in the UK

Training with noise and the storage of correlated patterns in a neural network model

E J Gardner N Stroud and D J Wallace

Physics Department, University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK

Received 22 February 1989, in final form 27 March 1989

Abstract. Local iterative learning algorithms for the interactions between Ising spins in neural network models are discussed. They converge to solutions with basins of attraction whose shape is determined by the noise in the training data, provided such solutions exist. The training is applied both to the storage of random patterns and to a model for the storage of correlated words. The existence of correlations increases the storage capacity of a given network beyond that for random patterns. The model can be modified to store cycles of patterns and in particular is applied to the storage of continuous items of English text.

1. Introduction

Networks of N Ising spins with pairwise interactions between the N sites can be interpreted as neural network models. In this analogy, the Ising spin variable represents a (grossly simplified) neuron which can exist in only two states, *firing* and *not firing*, and the exchange constant between two spins represents the synaptic efficacy or connection strength between the corresponding 'neurons', determining the effective field (potential) on one due to the state of the other (McCulloch and Pitts 1943, Hebb 1949). Such a network can be used to store a prescribed set of N-bit patterns, in the sense that, in the absence of thermal noise, each of the prescribed patterns is self-replicating. This information is stored with content-addressable memory: it is possible in principle to recover one of the prescribed patterns from a noisy initial version of that pattern.

In the context of such Ising-like models, the dynamics for the recovery of this information is defined by

$$S_i(t+1) = \operatorname{sgn} h_i(t) \qquad \text{where} \qquad h_i(t) = \sum_{j=1}^{n} T_{ij} S_j(t). \tag{1}$$

Here $S_i(t) = \pm 1$ is the Ising spin at site $i (= 1, \ldots N)$ at time t, and T_{ij} is the interaction strength of the bond from site j to site i. The updating of the spins depicted by equation (1) can be done either in series (Hopfield 1982) or in parallel (Little 1974). The interaction strengths T_{ij} should be constructed if possible so that the attractors of the dynamics (1) are the P prescribed patterns $\xi_i^\mu = \pm 1$, $i = 1, \ldots N$, $\mu = 1, \ldots P$, which one wants to store; a weaker constraint is that the attractors should be highly correlated with the ξ_i^μ. This construction should be designed to maximise both the number of patterns which can be stored and the number of wrong bits which can be tolerated in the noisy state in order to be in the domain of attraction of the pattern.

0305-4470/89/122019+12$02.50

Analytic results have been obtained using particularly simple ansatzes for the couplings. The Hopfield model can store up to $0.14N$ random patterns (Amit *et al* 1985a, b, 1987, Gardner 1986, Bruce *et al* 1987) while the pseudo-inverse (Kohonen 1984, Personnaz *et al* 1985, Kanter and Sompolinsky 1987) can store up to N linearly independent patterns.

Although these models have considerable interest, their storage capacities are disappointing. Firstly, the potential capacity of pairwise connected networks can be much larger; the maximum number of random patterns (Cover 1965, Venkatesh 1986a, b, Baldi and Venkatesh 1987) which can be stored is $2N$, and this increases if the patterns are correlated (Gardner 1987a, 1988). Secondly, at this level of storage, the model defined by (1) is computationally wasteful; noisy pattern recognition for $P < N$ can be achieved by a direct comparison of the noisy input pattern with each of the nominated patterns.

The purpose of this paper is twofold. First, in § 2, we review an iterative local training algorithm which increases the storage capacity to its maximum possible, in terms of both the number of stored patterns and their content-addressability. We shall prove that provided solutions for the T_{ij} exist, i.e. that it is possible in principle to store the prescribed patterns with a specified content-addressability, then the algorithm converges to one such solution. This algorithm (more precisely, family of algorithms) is an extension of Perceptron learning (Rosenblatt 1959, Minsky and Papert 1969) in two senses: to recurrent networks and to training with noisy initial vectors. The former aspect ensures that it is also akin to the error-correcting back-propagation algorithm (Werbos 1974, Parker 1985, Rumelhart *et al* 1986), for which however no convergence theorem exists as yet. The latter aspect ensures that the content-addressability is adapted to the noise statistics in the data. Preliminary results for the storage of random prescribed patterns have been described earlier (Wallace 1985, 1986, Bruce *et al* 1986). Second, in § 3 we consider a model for the storage of words. In particulaar the model can be used to store continuous pieces of english text. In §§ 4 and 5, numerical results on the application of the algorithms to the storage of random patterns and to word storage respectively will be given. Conclusions are in § 6.

2. The training algorithms

The strategy of the algorithm is as follows. Let S_i^μ be any vector which differs from a particular input pattern ξ_i^μ on a fraction of bits less than or equal to f. We then define an error mask, which depends on the particular noisy pattern, by

$$\varepsilon_i^\mu = \frac{1}{2}\left[1 - \operatorname{sgn}\left(\xi_i^\mu \sum_j T_{ij} S_j^\mu\right)\right] \tag{2}$$

for each site i; it takes the value 0 (1) according to whether site i of pattern μ is (is not) correctly retrieved. The matrix T_{ij} is then updated according to the rule

$$\Delta T_{ij} = \varepsilon_i^\mu \xi_i^\mu S_j^\mu \qquad i \neq j \text{ only} \tag{3}$$

which increases the quantities $\xi_i^\mu h_i(S^\mu)$ only on sites which are in error at time t.

The symmetrical version changes T_{ij} by $\Delta T_{ij} + \Delta T_{ji}$, where ΔT_{ij} is given by (3).

In contrast to the back-propagation algorithms used in models with hidden units (Werbos 1976, Parker 1985, Rumelhart *et al* 1986), a convergence theorem exists for these algorithms. Specifically, provided there exists a solution for the couplings T_{ij}

which will recover each pattern μ in one iteration using equation (1) from any initial S^μ within a Hamming distance Nf of ξ^μ, repeated iteration of (3) will converge to some such solution.

The proof of convergence for the first algorithm (3) parallels the Perceptron convergence theorem (Rosenblatt 1959, Minsky and Papert 1969): we define the scalar product of two interaction matrices T and U at the site i by

$$(T \cdot U)_i = \sum_j T_{ij} U_{ij} \tag{4}$$

and the norm of T at the site i by

$$\| T \|_i = \sqrt{(T \cdot T)_i} \tag{5}$$

and we will assume that a solution T^*_{ij} exists such that

$$\xi^\mu_i (T^* \cdot S^\mu)_i > \delta \| T^* \|_i \tag{6}$$

for some positive number δ and for each noisy initial vector S^μ. The proof of convergence follows by showing that the quantity

$$\frac{(T^{(n)} \cdot T^*)_i}{\| T^{(n)} \|_i \| T^* \|_i} \tag{7}$$

where $T^{(n)}$ is the interaction matrix at the nth application of (3) at the site i, will eventually increase above 1 if the algorithm does not converge. However this quantity is bounded above by 1, by the Schwartz inequality; and the algorithm must therefore converge after a finite number of steps. Specifically, consider a pattern with $\varepsilon^\mu_i > 0$. Since from equation (3)

$$\begin{aligned} \Delta((T^* \cdot T^{(n)})_i) &= \varepsilon^\mu_i \xi^\mu_i (T^* \cdot S^\mu)_i \\ &> \delta \| T^* \|_i \varepsilon^\mu_i \end{aligned}$$

and

$$\begin{aligned} \Delta(\| T^{(n)} \|^2) &= 2\varepsilon^\mu_i \xi^\mu_i (T^{(n)} \cdot S^{(\mu)})_i + N\varepsilon^\mu_i \\ &< N\varepsilon^\mu_i \end{aligned}$$

we have that

$$\frac{(T^* \cdot T^{(n)})_i}{\| T^{(n)} \|_i \| T^* \|_i} > \delta \sqrt{\frac{n}{N}}$$

where n is the number of iterations with $\varepsilon^\mu_i = 1$, so the algorithm must converge when n becomes sufficiently large.

The asymmetric algorithm can be done either in series or in parallel in the sites i since the theorem holds at each site i. The proof of convergence for the symmetric algorithm ($T_{ij} = T_{ji}$) is similar but the scalar product defined in equation (4) is replaced by

$$T \cdot U = \sum_{i,j} T_{ij} U_{ij}$$

and the theorem applies only if the updating of the sites is done in parallel. As given, the proof assumes the patterns are presented in serial. It extends trivially to the case in which the patterns are divided into groups, those within each group being presented

in parallel and the groups sequentially. The above inequalities are replaced by $>\delta\|T\|_1\Sigma_\mu\varepsilon_i^\mu$ and $<Ng\Sigma_\mu\varepsilon_i^\mu$, where g is the number of patterns in a group.

It is also straightforward to show that for a given stored nominated pattern, if all noisy versions with exactly fN bits flipped iterate into it in one step, then so do all patterns with fewer spins flipped. Hence, for the isotropic basins of attraction which we study, we need train only on the 'rim' of the basin of attraction and this is what we do in practice. For the proof, suppose that T is a row of the weight matrix and $S(fN)$ is a certain nominated state, S, with fN spins flipped. Suppose further (without loss of generality) that the desired output of the element corresponding to the row with weights T is positive. By assumption, $T\cdot S(fN)>0$ (we have a solution for fN spins flipped). Now consider one of the noisy states with $fN-1$ spins flipped. We are required to prove that $T\cdot S(fN-1)>0$. Flipping the ath (correct) spin creates one of the $S(fN)$, so

$$\begin{aligned} T\cdot S(fN-1) &= T\cdot S(fN)-2T_aS_a(fN) \\ &> 2T_aS_a \end{aligned}$$

where S_a is the ath component of the nominated state itself. This must be true for any choice of the component a from the set of correct spins in $S(fN)$ since we assume $T\cdot S(fN)>0$ for all $S(fN)$. However, at least one of those components gives the correct sign for the output since $T\cdot S(fN)+T\cdot S>0$, so $T\cdot S(fN-1)>0$, as required.

There are many generalisations of these algorithms and here we note two. First, they may obviously be extended to solutions which converge to the pattern in more than one iteration. For example the asymmetric one becomes for two iterations $\Delta T_{ij}=\varepsilon_i^\mu\xi_i^\mu S_j'^\mu$ where S'^μ is the configuration S^μ after one iteration of equation (1) and ε_i is 1 provided the bit i is wrong after two iterations using equation (1). Second, they can be generalised to situations in which the noise is not isotropic, and can therefore be used to construct asymmetric basins of attraction.

Finally in this formal section, we note that, as for perceptrons, the dynamical range of the T_{ij} is also self-limiting for those algorithms for which convergence can be proved. Thus, even if one training task cannot be accomplished, because for example no solution exists for the prescribed patterns, the values of the T_{ij} will not have increased unboundedly. Clear evidence of this desirable property is found in the numerical results of § 4.

3. Word storage model

The storage of representations of English words is an example of a wide class of common problems. Consider words of a given number, L, of letters. For a lower-case dictionary, the number of such words is in principle exponential in L: 26^L. However, in reality the correlations between letters reduce the number of recognisably 'English' words to much less than 26^L. Other examples are the sets of phonetically spelled words, or of syntactically allowed sentences in English (or any other language) composed of appropriate syntactic elements.

In order to implement such problems, we represent each letter by some n-bit pattern. In principle, these could be random-bit patterns, or n-pixel representations of the letters, or (modulo the crucial dynamic time-warp problem) a spectrographic representation of the phonetic sound. Words of L letters are then represented by patterns of $N=nL$ bits, which one wishes to store on an N-mode network. This structure has

some resemblance to the word storage reported by Personnaz *et al* (1986), but differs in its aims and training strategy.

In practice, we have used random 64-bit patterns to represent characters, and have studied the storage of four-letter words on 256 nodes, and eight-letter words on 512 nodes (the storage of words of up to eight letters is readily achieved by introducing a dummy character to make the number of letters up to eight in every word). The simulations were performed on the ICL Distributed Array Processor (DAP), a 64×64 grid of bit-serial processing elements. The full parallelism of the machine is exploited by training simultaneously on successive groups of 16 words (8 for the 512-node model), achieving some 25 million operations (conditional ADDs) per second. Further details are provided by Stroud and Wallace (1987).

Some preliminary comments may be helpful to indicate why the fully connected net without hidden units should be able to tackle this problem. We note that the prototype spurious states (Amit *et al* 1987) generated by the Hopfield prescription are mixtures of three or more of the prescribed patterns. Words can be viewed in the same way; for example, SAT is a $(\frac{5}{6}, \frac{5}{6}, \frac{5}{6})$ mixture of SIT, CAT and SAD (assuming each letter is a random bit pattern). To the extent that the language is determined by its pair correlations, the specific mixture states are useful generalisations and the storage capacity of the network should be larger than for random words. Higher spin correlations could also be introduced to remove undesired words based on pair correlations between letters.

We have extended the code to handle continuous text, by stepping a 'window' along it to pick out successive eight-letter segments. In the training mode, one would typically start with an input pattern corresponding to the first eight letters of the text, with some noise level. This is iterated once, the error mask (2) calculated, and hence the change in weights (3). The existing pattern on the net slides along to introduce a new 64-bit pattern, corresponding to the ninth letter in the text with some noise level. The 512-pixel pattern is iterated once to find the error mask (2), and so on. The same approach is adopted in the recall mode, the 64-bit pattern on the left providing the single output letter at each step of the window; this output has been influenced by a 15-letter wide segment of the text, including seven letters before and seven after. The number of iterations of the net between each step can be a rather crucial parameter. If only one iteration is allowed, this may be insufficient to remove all of the pixel errors. If a large number of iterations is allowed, the incoming noisy pattern may corrupt the existing seven-letter pattern to such an extent that the net 'crashes', producing completely spurious output. The net can also be used in a pattern-generating mode, using a cue of the first seven letters alone to generate the remaining text, to the capacity of the net.

4. Numerical results for random patterns

Numerical results for random patterns will first be discussed.

The training schedule consisted of up to 20 cycles with zero noise—in order to learn the patterns—followed by a multiple of 16 cycles with noise of a given number of spin flips, finishing off with a further session of up to 20 cycles with zero noise to ensure that the patterns themselves remain learnt. Each cycle consists of one entire sweep through all sites and all patterns for one noisy initial vector per pattern, using the symmetric algorithm in parallel on the sites and in series on groups of 16 (4-letter)

or 8 (8-letter) patterns. The percentage of patterns which are exactly recovered after complete iteration starting with noisy initial vectors with the same amount of noise is plotted in figure 1 against the number of training cycles, for 64 patterns on 256 nodes and for noise levels of 25, 31.25 and 37.5% (32, 40 and 48 spin flips). This shows that the system learns on 25% noise but fails to learn on 31.25% noise after 150 cycles. Clearly this is a substantial improvement on the Hopfield model, which fails to store more than $0.14N$ patterns.

An important question is the way in which the number of learning cycles scales with the system size. Since the number of states at a given fractional Hamming distance from a pattern scales exponentially with system size N, this suggests that the algorithms might also scale in this way. However we find that the algorithms scale polynomially with N. In figure 2, the fraction of wrong bits (after one iteration) is plotted against the number of learning cycles for two different sizes $N = 256$, $N = 512$ for noise levels 12.5, 25, 37.5% (16, 32, 48 spin flips for $N = 256$), for $P = 0.25N$, and shows that if a given fraction of wrong bits is allowed, then the number of cycles required does not depend on the system size. This means that if the number of patterns P is of order N, the computational complexity scales as N^2, i.e. linearly with the information stored.

Typically we find that although the mean number of iterations to stability decreases with learning, it has not decreased to one iteration after 150 cycles even for situations in which all patterns have been learnt. The mean iteration time was calculated only for initial states which moved towards the pattern. For example, for 64 patterns on 256 nodes, iteration times reduced from 1.2 to 1.07 on 12.5% noise after 150 cycles, while for 6.25% noise times reduced from 1.2 to 1.02.

The boundedness of the T_{ij} was demonstrated. Results for the symmetric parallel algorithm of § 2 are compared with results using an algorithm which modifies the T_{ij}:

$$\Delta T_{ij} = \varepsilon_i^\mu \xi_i^\mu \xi_j^\mu \tag{8}$$

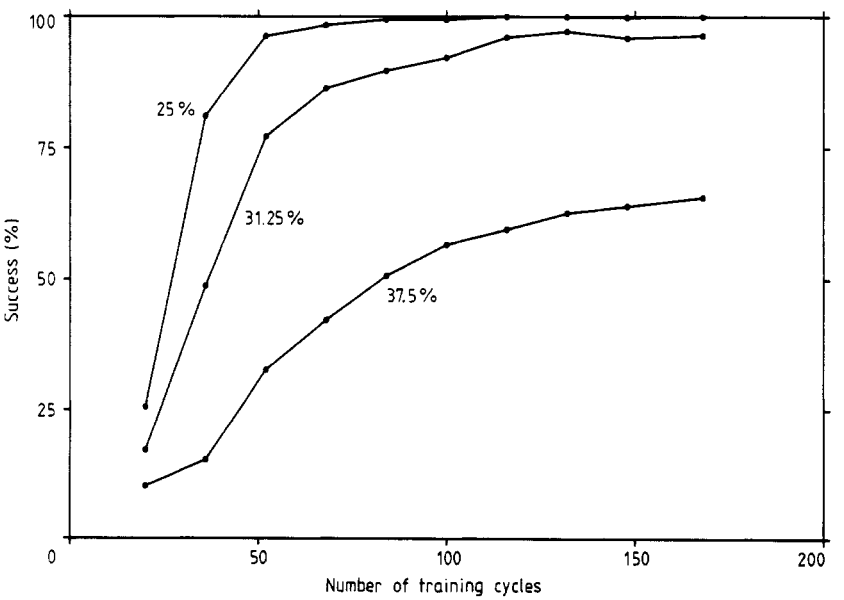

Figure 1. Percentage success after complete iteration against number of learning cycles for 64 random patterns on 256 nodes trained on 25, 31.25 and 37.5% noise.

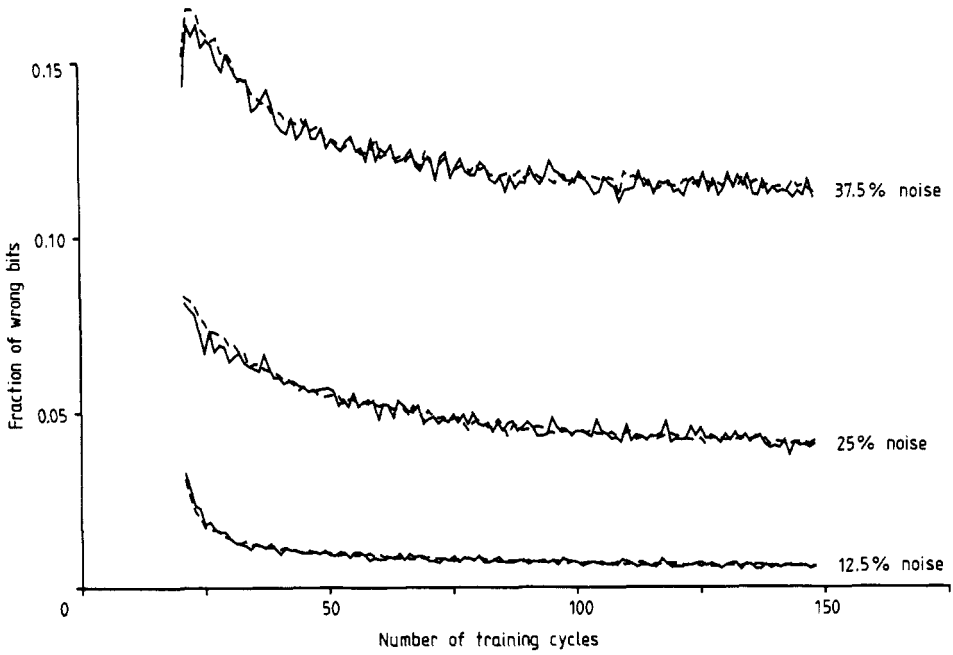

Figure 2. Fraction of wrong bits after one iteration against number of learning cycles for 64 random patterns on 256 nodes (——) and 128 random patterns on 512 nodes (– – –), trained on 12.5, 25 and 37.5% noise.

for which no convergence theorem can be proved. The result, for example, for 128 patterns on 512 nodes and 37.5% noise is that the T_{ij} remain bounded at an RMS value of 25 after 150 learning cycles for the algorithm of § 2, whereas the RMS value of T_{ij} is 40 after 150 cycles and continues to increase, for algorithm (8).

5. Numerical results on word storage

In figure 3 the training schedule is the same as that described in § 4, figures 1 and 2, for random patterns. The percentage of letters which are exactly recovered after complete iteration starting from noisy initial vectors with the same amount of noise is plotted against the number of training cycles, and is compared for three dictionaries, on 25, 31.25, 37.5, 43.75, 50% noise, for 64 words on 256 nodes and averaged over five runs. Figure 3(a) shows the results for 64 random patterns, 3(c) is for a dictionary of 64 real words, and 3(b) is for the same dictionary but with the letters jumbled at random. The learning in 3(b) is faster than 3(a) due to correlations coming from different words containing letters which are the same, and 3(c) is faster than 3(b) due to the pair correlation which occurs in real words.

Of course there may be many other spurious states in which the sub-patterns cannot be identified with letters. Further, if the number of different letters or characters is less than 64, all possible words still span only a subspace of all possible states. Clearly, one possible storage solution is a separable net, in which all possible letters are stored on each 64-bit subnet, and all the connections *between* the letters of a word are set to

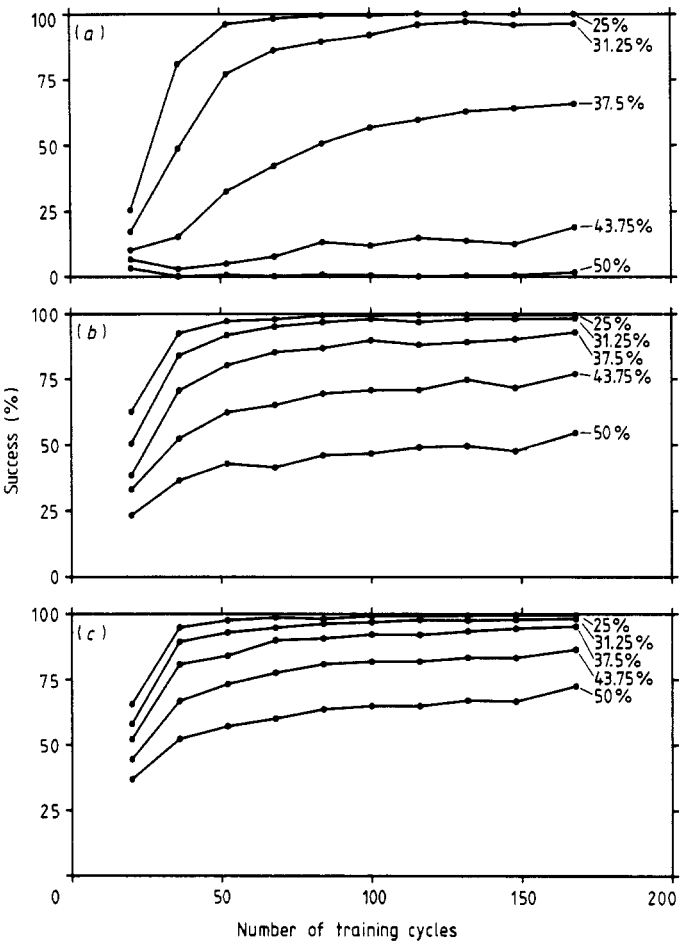

Figure 3. Percentage success to a letter after complete iteration against number of learning cycles on 25, 31.25, 37.5, 43.75 and 50% noise, for 64 words on 256 nodes, for (*a*) random patterns (from figure 1), (*b*) jumbled words, (*c*) real words.

Table 1. Results for the percentage success to a letter obtained after training on a dictionary of 64 real words (D1) and one of the same dictionary with the words jumbled (R1) are tested on two dictionaries of real words (D1), (D2) and the corresponding jumbled word dictionaries (R1), (R2) after training for 150 cycles on (*a*) 12.5, (*b*) 25 and (*c*) 37.5% noise.

(*a*)	D1	R1
D1	100%	37%
D2	56%	40%
R1	37%	100%
R2	38%	35%

(*b*)	D1	R1
D1	100%	36%
D2	58%	41%
R1	36%	99%
R2	37%	30%

(*c*)	D1	R1
D1	95%	28%
D2	48%	30%
R1	31%	92%
R2	28%	22%

zero. Such a net would store all words, but would have no information about the characteristics of English words, beyond possibly the letter frequency. A key question will therefore be the relative strengths of the interactions *within* and *between* letters. We show that contextual information is being used in table 1. Four dictionaries of 256 words are used. The first two, D1 and D2, are different dictionaries of real words, and the second two are jumbled versions of D1 and D2, R1 and R2 respectively. One 'real' dictionary (D1) and one 'jumbled' one (R1) are trained into a 256-node network with 12.5, 25 and 37.5% noise, for 150 cycles. All four dictionaries are then tested against the resulting networks. The results show firstly that percentage success is significantly greater for the dictionary used in training, implying that a large fraction of spurious words have not been learnt. Secondly the results for a real-word dictionary

Table 2. Results for percentage success to a letter for 150 learning cycles on 12.5% noise for 256 words on 256 nodes for dictionaries (D1), (D2), (R1) and (R2), for (*a*) 0, (*b*) 25 and (*c*) 50% dilution of bonds within each letter.

(*a*)	D1	R1
D1	100%	91%
D2	94%	90%
R1	70%	100%
R2	69%	94%

(*b*)	D1	R1
D1	96%	64%
D2	84%	63%
R1	51%	91%
R2	51%	72%

(*c*)	D1	R1
D1	76%	26%
D2	59%	25%
R1	27%	50%
R2	26%	32%

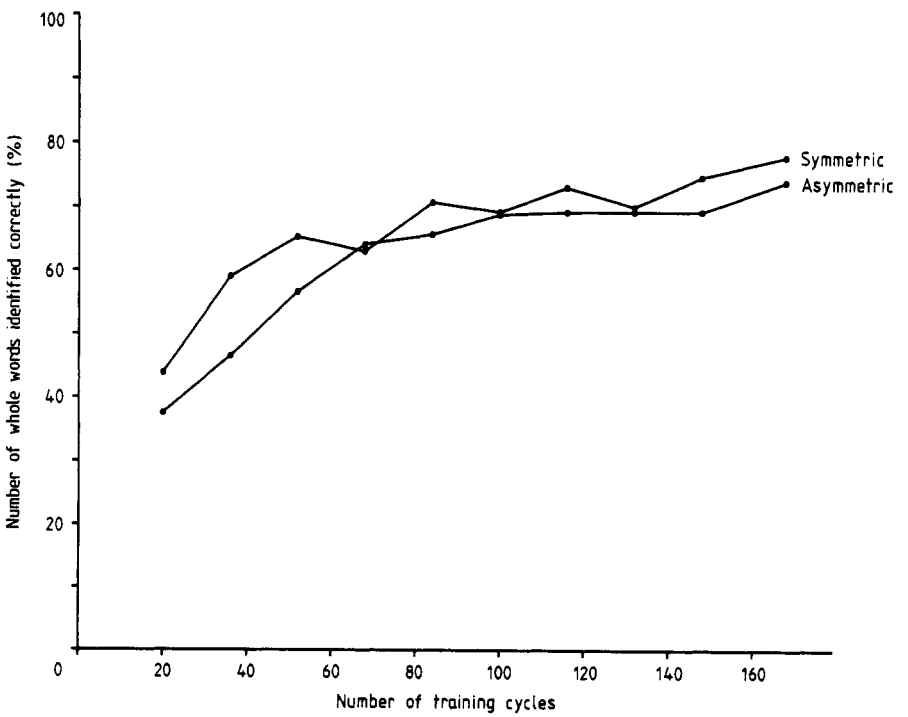

Figure 4. Percentage success to a letter after complete iteration against number of learning cycles for 64 words on 256 nodes, on 25% noise, for symmetric and asymmetric algorithms.

tested on a network trained with the other real-word dictionary are significantly better than the results on the jumbled word dictionaries, showing that contextual information (i.e. from the pair correlations between letters) is being used by the network. We also tried to enhance the effect of contextual information over single letter information by diluting connections within letters. Results with 0, 25 and 50% dilution are shown in table 2 for the same four dictionaries as above, for 256 words on 256 nodes with 12.5% noise. Although storage capacity is reduced as the dilution is increased, the fraction of spurious words is also reduced at high dilution, but with the second set of real words D2 still favoured compared with R1 and R2.

The symmetric algorithms improve only slightly on the asymmetric algorithms. This is shown in figure 4 for 64 words on 256 nodes with 25% noise, where the percentage success to a letter is plotted as a function of the number of learning cycles.

In figure 5 we show the output obtained using the continuous text model of § 3, with 25% noise. The output illustrates how the training first stores common letters and common words, and then more general features, as one might expect.

```
Continuous Text
***************
****************************************  Returning from DAP:
                                          TIME = 564  (9.40 mins)    NFIN =15
Training success:
  45.50%  58.75%  63.25%  66.75%  78.00%  84.75%  88.50%  90.75%

Session  1:  Final states of (fixed) random start:
___t**_~***e_~_**_t~*_****_t*e_the_***e__**e_*_*e_**e***_the*e_**_*_**e_*_**_***
**e_*e_the****~**E~_~*_*e_~**E~*~**E*_~_***e*_**t*e_***_**__*_**_**__t**_*******
_**e__***__***_**i_*_******_**_**e__**e__***_***_**___t*e_****_*h**_**e*_*****_t
**_*e_**_**_*_**e*_**_**_***e_**___**e_~e**_t**_**e**_**e_*_~*_**_*_*~*_*****_**
__~e_t*e*__**_**e*e__***_*e***e*_***_***_*_******_t**_*e_**e*_**_**_**e_*_**_**_**_

Session  2:  Final states of (fixed) random start:
___the_***d**s_o*_t*e_**nd_*re_t*e_****s_**ich_*e_**e_**_***c*__o**_**ems_o*_**o
**ed*e_*h********o*_*e_i*_e**a*e_*_***e**e*ed___th*_or*a****tio*_*n*_the_she**es
_*i**_*or__i**o*t*nt_to*i**_is_o**_*_*er_*i**_s****___t**_****_t*_*_**es_o*_**_t
he_*e*ch_m**t_a*s*_*e_*o*_**ered___***_*e_e_the_*tems_**ich_~*_****_*he_****e_*a
*_*e_*e*e*_to_****e*_an*_re*sse***e*_and_*_**e***__a*_*e_*a*e_o*_theor_*nte****_

Session  4:  Final states of (fixed) random start:
___the_***d*rs_o*_t*e_mind_are_t*e_*e**s_w*ic*_**_**e_*o_**ic*_*o*n_*tem*_**_**o
**e_*e_w*ich_cannot_*e_im*e*ia*e**_**_e**e*ed___t*e_or*a****t*o*_and_t*e_s*e**es
_*i**_*or*_im**r*ant_to*ic*_i*_***_*a*e*_*i****s_*n___t**_*ork_t*at_****_**_*t_t
he_*ench_must_a*s*_*e_~**_**e*ed___*or_here_the_i*ems_**ich_*o_int*_the_st***_**
*_*e_t**en_t*_**e*e*_an*_re*ssem*le*_and_a_**et*h_*a*_*e_ma*e_o*_thei*_intern**_

Session  6:  Final states of (fixed) random start:
___the_ladders_**_the_mind_are_t*e_***es_which_we_**e_to_*r*c*_*o*n_items_of_kno
wled*e_which_cann*t_be_imme*iatel*_re*embered___**e_or*anisation_and_the_she**es
_*il*_*orm_importa*t_to*i*s_in_our_later_dis**ssion___the_*or*_that_*oes_on_a*_t
he_*enc*_must_a*so_be_*on*idered___*or_here_the_*tems_which_**_into_*he***ore_**
y_be_ta**n_to_*ie*e*_and_reassem*led_an*_a_s*et*h_may_*e_*ade_of_their_internal_

Session  8:  Final states of (fixed) random start:
___the_ladders_of_the_mind_are_the_***es_which_*e_*se_to_trac*_down_items_of_kno
wledge_which_cannot_be_immediatel*_r*mem*ered___the_or*anisation_and_the_shel*es
_will_form_importa*t_to*ic**i*_o*r_later_dis*ussion___the_*or*_that_goes_on_at_t
he_*ench_must_also_*e***n***ered___*or_here_the_items_wh*ch_*o_into_the_*tore_ma
y_be_taken_to_*ieces_and_reassembled_and_a_s*etch_may_*e_made_o*_their_interna*_
```

Figure 5. Output from continuous text model trained on 25% noise, at various stages during the training.

6. Conclusions

We have discussed how the perceptron convergence theorem can be extended to the Hopfield net and generalisations thereof, to realise the storage capacity to its theoretical maximum. By training with noise we can create arbitrarily shaped neighbourhoods of content-addressability, again to the capacity of the net; 'real' data create the required neighbourhoods.

The algorithm (3) and its symmetrical equivalent can also be generalised to cycles of patterns (Sompolinsky and Kanter 1986), ultrametric hierarchies (Parga and Virasoro 1986, Cortes *et al* 1987, Gutfreund 1988), and multiconnected models (Gardner 1987b). Finally, we note that there is also another set of algorithms (Gardner 1987a, Diederich and Opper 1987, Forrest 1988, Krauth and Mézard 1987) which train on the prescribed patterns only (without noise). In Gardner (1987a), Forrest (1988) and Krauth and Mézard (1987), the algorithm obtains a solution T^* such that

$$\xi_i^\mu \sum_j T_{ij}^* \xi_j^\mu > C \| T^* \|_i \tag{9}$$

for each pattern μ and for each i, where C is a constant, and where the mask ε_i^μ is defined as

$$\varepsilon_i^\mu = \frac{1}{2}\left[1 - \operatorname{sgn}\left(\xi_i^\mu \sum_j T_{ij} \xi_j^\mu - C(\| T \|_i)\right)\right]. \tag{10}$$

This algorithm also converges provided solutions exist. These algorithms should have shorter convergence times since they train only on the patterns, and increasing the value of C to some point will enhance content addressability. Analytic results (Gardner 1987a, Gardner and Derrida 1988) on the existence of solutions with the property (10) can also be obtained for both random and correlated patterns.

Acknowledgments

This work was supported in part by the Science and Engineering Research Council under grant NG/15908 and by the Ministry of Defence, under contract 2087/27. EG thanks the University of Edinburgh for the award of a Dewar Fellowship. It is a pleasure to thank D G Bounds, J S Bridle and R K Moore for stimulating discussions, and M Wong for comments on the original manuscript.

References

Amit D J, Gutfreund H and Sompolinsky H 1985a *Phys. Rev. Lett.* **55** 1530
—— 1985b *Phys. Rev.* A **32** 1007
—— 1987 *Ann. Phys., NY* **173** 30
Baldi P and Venkatash S 1987 *Phys. Rev. Lett.* **58** 913
Bruce A D, Canning A, Forrest B, Gardner E and Wallace D J 1986 *Proc. Conf. on Neural Networks for Computing, Snowbird, Utah*, 1986 (*AIP Conference Series* **151**) ed J S Denker (New York: AIP) p 65
Bruce A D, Gardner E and Wallace D J 1987 *J. Phys. A: Math. Gen.* **20** 2909
Cortes C, Krogh A and Hertz J A 1987 *J. Phys. A: Math. Gen.* **20** 4449
Cover T M 1965 *IEEE Trans.* **EC-14** 326
Diederich S and Opper M 1987 *Phys. Rev. Lett.* **58** 949
Forrest B M 1988 *J. Phys. A: Math. Gen.* **21** 245

Gardner E 1986 *J. Phys. A: Math. Gen.* **19** L1047
—— 1987a *Europhys. Lett.* **4** 481
—— 1987b *J. Phys. A: Math. Gen.* **20** 3453
—— 1988 *J. Phys. A: Math. Gen.* **21** 257
Gardner E and Derrida B 1988 *J. Phys. A: Math. Gen.* **21** 271
Gutfreund H 1988 *Phys. Rev.* A **37** 570
Hebb D O 1949 *The Organisation of Behaviour* (New York: Wiley)
Hopfield J J 1982 *Proc. Natl Acad. Sci. USA* **79** 2554
Kanter I and Sompolinsky H 1987 *Phys. Rev.* A **35** 380
Kohonen T 1984 *Self Organisation and Associative Memory* (Berlin: Springer)
Krauth W and Mézard M 1987 *J. Phys. A: Math. Gen.* **20** L745
Little W A 1974 *Math. Biosci.* **19** 101
McCulloch W S and Pitts W A 1943 *Bull. Math. Biophys.* **5** 115
Minsky M L and Papert S 1969 *Perceptrons* (Cambridge, MA: MIT Press)
Parga N and Virasoro M 1986 *J. Physique* **48** 1857
Parker D B 1985 *Learning Logic* MIT Technical Report TR-47
Personnaz L, Guyon I and Dreyfus G 1985 *J. Physique Lett.* **46** L359
—— 1986 *Phys. Rev.* A **34** 4217
Rosenblatt F 1959 *Principles of Neurodynamics* (New York: Spartan)
Rumelhart D E, Hinton G E and Williams R J 1986 *Parallel Distributed Processing: Explorations in the Microstructure of Cognition Vol 1: Foundations* ed D E Rumelhart and J L McClelland (Cambridge MA: Bedford Books/MIT Press)
Sompolinsky H and Kanter I 1986 *Phys. Rev. Lett.* **57** 2861
Stroud N and Wallace D J 1987 *The DAP implementation of Algorithms for word storage in networks without hidden units* Edinburgh University Report 87/390
Venkatesh S 1986a *Proc. Conf. on Neural Networks for Computing, Snowbird, Utah*, 1986 (*AIP Conference Series* **151**) ed J S Denker (New York: AIP) pp 440–5
—— 1986b *Ph D Thesis* California Institute of Technology, unpublished
Wallace D J 1985 *Advances in Lattice Gauge Theory* ed D W Duke and J F Owens (Singapore: World Scientific) pp 326
—— 1986 *Lattice Gauge Theory—A Challenge to Large Scale Computing* B Bunk, K H Müttor and K Schilling (New York: Plenum) pp 313
Werbos P J 1974 *Ph D Thesis* Harvard University, unpublished