High Performance Associative Memory Models and Symmetric Connections

N. Davey, R.G. Adams, S. P. Hunt
Department of Computer Science,
University of Hertfordshire
Hatfield, Herts. AL10 9AB. United Kingdom
Email: {N.Davey, R.G.Adams, S.P.Hunt}@herts.ac.uk

Abstract

Two existing high capacity training rules for the standard Hopfield architecture associative memory are examined. Both rules, based on the perceptron learning rule produce asymmetric weight matrices, for which the simple dynamics (only point attractors) of a symmetric network can no longer be guaranteed. This paper examines the consequences of imposing a symmetry constraint in learning. The mean size of attractor basins of trained patterns and the mean time for learning convergence are analysed for the networks that arise from these learning rules, in both the asymmetric and symmetric instantiations. It is concluded that a symmetry constraint does not have any adverse affect on performance but that it does offer benefits in learning time and in network dynamics.

1 Introduction

Due to the low capacity of the original Hopfield associative memory model, several other, more recent, models, have been developed that have a higher performance [Diederich and Opper 1987, Krauth and Mezard 1987, Gardner 1988]. The models examined here all use variants of the perceptron learning procedure that are known to produce networks with high capacity. However, unlike the original Hopfield prescription, the weight matrices so formed are normally non-symmetric. This effects the dynamics of the network, so that the normal asynchronous state update is no longer guaranteed to converge to a point attractor. In this paper we investigate the implications, for attractor performance and training times, of reimposing the symmetry constraint on the weights of such networks.

2 Models Examined

In this section we take a set, Π, of N-ary, bipolar (+1/-1) training vectors, $\{\xi^p\}$. The N by N weight matrix is denoted by **W**, and the state of the i'th unit is denoted by S_i. The loading of the network is defined as:

$$\alpha = \frac{|\Pi|}{N} .$$

All the high capacity models studied here are modifications to the standard Hopfield network. The net input, or *local field*, of a unit, is given by: $h_i = \sum_{j \neq i} w_{ij} S_j$

The next state of the unit is then given by:

$$S_i(t+1) = \begin{cases} 1 & \text{if } h_i > \theta_i \\ -1 & \text{if } h_i < \theta_i \\ S_i(t) & \text{if } h_i = \theta_i \end{cases}$$

where the threshold, θ_i , is normally taken as zero. The update can be synchronous or asynchronous. Here we use asynchronous, random order updates. These network dynamics and a symmetric weight matrix guarantee simple point attractors in the phase space.

If the *aligned local fields*, $h_i\xi_i$, are all non-negative then the pattern ξ, will be stable in the network.

2.1 Perceptron Style Learning

In the late 1980s it was demonstrated that perceptron like learning could be applied to associative memory networks to produce much higher capacity. In fact as Gardner [1988] showed a Hopfield type network of N units could store up to 2N uncorrelated patterns, with this optimal capacity increasing for correlated patterns. Learning rules of this type are designed to drive the aligned local fields of patterns in the training set over a threshold value, T. As shown above, a necessary and sufficient condition for the training patterns to be learnt is that T is non-negative, and often, for ease of training, a value of 1 (or even 0) is taken. Nevertheless increasing T may improve the attractor performance of the network [Abbott 1990]. Some care must be taken though, since if we consider a network in which all the training patterns are stable, that is $h_i\xi_i > T$ for all patterns and units, i, then any uniform, upward scaling

of the weight matrix will increase the aligned local fields, but will obviously not increase the attractor performance. In fact the optimal attractor performance in achieved when the threshold is maximised with respect to the size of the weights. For this reason the relevant characterisation is the *normalised stability measure*, defined as: $\gamma_i = \frac{h_i \xi_i}{|W_i|}$ where W_i is the incoming weight vector to unit i. The minimum of all the γ_i therefore gives a measure of the likely attractor performance and we take $\kappa = \min_{p,i}(\gamma_i^p)$.

2.1.1 Local Learning (LL)

Diederich and Opper's [1987] local learning rule is an iterative learning rule in which the local fields for each training pattern are driven to the correct side of +T or –T as appropriate. This is equivalent to the condition that:

$$i, p \cdot h_i^p \xi_i^p \geq T$$

So the learning rule is given by:

Begin with a zero weight matrix
Repeat until all local fields are correct
  Set the state of network to one of the ξ^p
  For each unit, i, in turn
    Calculate $h_i^p \xi_i^p$.
    If this is less than T then change the weights to unit i according to:

$$\Delta w_{ij} = \frac{\xi_i^p \xi_j^p}{N}$$

This is the perceptron learning rule with a fixed margin of T and a learning rate of $\frac{1}{N}$. The process will converge on a suitable weight matrix if one exists [Diederich and Opper, 1987], at which point the trained patterns are guaranteed to be stable. Normally a symmetric weight matrix will not be the result, so that cycles can arise in the network dynamics. These are actually quite rare, but nonetheless present problems.

As shown by Abbott [1988], this leads to a network in which $\kappa \geq \frac{T}{2T+1}\kappa_{\max}$, where $\kappa_{\max}$ is the optimal value of κ. From this it is apparent that increasing T will in turn increase the lower bound of κ, and this may give better attractor performance.

2.1.2 Krauth and Mezard Local Learning (KM)

A modification to this learning rule proposed by Krauth and Mezard [1988] can be shown to produce a κ value that tends towards $\kappa_{\max}$ as T increases. In this version the patterns are not presented to the network in an arbitrary order. Instead the pattern that has the smallest aligned local field is chosen as the one for next presentation.

Begin with a zero weight matrix
Repeat until all local fields are correct
  For each unit, i, in turn
    Select the pattern, ξ^p *with lowest aligned local field at this unit and update the incoming weights according to:*

$$\Delta w_{ij} = \frac{\xi_i^p \xi_j^p}{N}$$

2.2 Symmetry of the Weights

The original Hopfield network has a symmetric weight matrix and such weights have the desirable property of implying point atttractors, with asynchronous updating and cycles of at most length 2 with synchronous updates. As the symmetry is broken, more complex dynamics become progressively more likely. On the other hand Krauth, Nadal and Mezard [1988] showed that, under certain circumstances, decreasing the symmetry of the weight matrix should improve attractor performance. Another factor, not pursued in this paper, is that a symmetric weight matrix can represent less information than an asymmetric one, so it is no surprise that the storage capacity of a fully symmetric network is less than an asymmetric one [Gardner 1989].

Gardner [1988] pointed out that an iterative perceptron like training rule could be made to produce symmetric weights by simply updating both w_{ij} and w_{ji} when either changes. She also showed that such algorithms would find a symmetric weight matrix, if one existed, for a particular training set. To investigate the implications of having a symmetry constraint we compare asymmetric and symmetric versions of both the Diederich and Opper local learning method (LL) and the Krauth and Mezard optimal version (KM).

The *Symmetric Local Learning rule* (SLL) is therefore:

Begin with a zero weight matrix
Repeat until all local fields are correct
  Set the state of network to one of the ξ^p
  For each unit, i, in turn
    Calculate $h_i^p \xi_i^p$.
    If this is less than T then change the weights between unit i and all other units, j, according to:

$$\Delta w_{ij} = \Delta w_{ji} = \frac{\xi_i^p \xi_j^p}{N}$$

We donate as *SKM* the corresponding symmetrical version of KM.

3 Analysing Performance

As a consequence of the training rules used, all the associative memory models discussed in this paper perform with perfect fidelity with respect to the stability of trained patterns. Nonetheless there are two other important features of associative memory that should be examined. These are the training time, and attractor performance – the size of the attractor basins of the trained patterns.

The relationship between attractor basin size and memory loading is presented first. Each of the models was trained with sets of random training patterns, in which:

$$i,p \bullet prob\,(\xi_i^p = +1) = 0.5,$$

3.1 Basins of Attraction

For useful pattern association to occur, the patterns stored in an associative memory of the Hopfield type must act as attractors. The ideal behaviour of such an associative memory would be such that a given initial state would relax to the nearest trained pattern. If the trained patterns are not the only attractors then such behaviour cannot be guaranteed. As is well known, Hopfield type networks necessarily contain many attractors not in the training set, such as inverses and mixture patterns. So the key question to ask of a Hopfield type network is: what is the mean size of the basins of attraction of the trained patterns?

Since the attractor basins cannot be expected to be Hamming hyperspherical [Storkey & Valabregue, 1999], it is usual to take the minimum Hamming radius:

$$R(p) = \inf\left\{\left\langle q, p\right\rangle : q \quad Basin(p)\right\}$$

The mean radius of attraction,

$$R = \frac{1}{|\Pi|}\sum_{p\,\Pi} R(p),$$

can act as a measure of the quality of a particular associative memory. It is also common for R to be normalised with respect to the size of the network, so that it lies between zero and one:

$$R' = \frac{1}{N} R$$

For very small networks it is possible to exhaustively explore the state space (see, for example [Personnaz et al, 1986]), in order to calculate R exactly, but for more realistic sizes the nature of the attractors can only be explored statistically. A sample (of size Γ) of states at a fixed distance, r, from a trained pattern, p, is made, and if all them relax to p, it is concluded that $R(p)$ is at least as big as r. An incremental search over increasing values of r provides an estimate for $R(p)$. Clearly, the larger the value of Γ the higher the quality of the estimate.

For finite size associative memories another factor needs to be considered. Any random starting point may have a relatively high overlap with many of the trained patterns, as well as the intended attractor and, to compensate, we follow the method of Kanter and Sompolinsky [1987] in the calculation of R, as described below. A series of initial states is chosen. In each case, a fixed fraction, m, of the state is identical to the corresponding part of one of the stored patterns, ξ^p, and the rest of the state is random. If the value of m is high then the network will relax to ξ^p for every one of those initial states. The value of m is reduced until a value, m_0, is reached at which one or more initial states do not relax to the desired state. Averaging m_0 over different stored patterns yields:

$$R = 1 - \left\langle\left\langle m_0 \right\rangle\right\rangle$$

As is pointed out in [Kanter & Sompolinsky, 1987], the initial states used in this calculation may overlap one of the other stored patterns more closely than ξ^p, and to compensate for this the definition of R is modified to:

$$R = \left\langle\left\langle \frac{1 - m_0}{1 - m_1} \right\rangle\right\rangle$$

where m_1 is the largest overlap with the rest of the stored patterns.

In our implementation, the search for the value m_0 is undertaken from low m to high m. A fixed number, Γ, of random starting points are chosen, each of which has low overlap with every member of the training set. If, as is likely, the start state does not relax to the closest training pattern in one or more of the Γ cases, the value of m is increased (by 0.01), and the search is repeated. This continues until all random start states relax to the closest stored pattern. This procedure is performed for six different sets of stored patterns for each network type: three sets of unbiased random patterns, and three sets of biased random patterns. Unless otherwise stated Γ has value 50.

The perfect attractor network has R = 1, which means that it is possible to move away from any stored pattern, and stay within its basin of attraction up to the point at which another stored pattern becomes nearer (see Figure

1). Note that the calculation of average attractor basin size for the trained patterns can only sensibly be undertaken when these patterns are themselves stable.

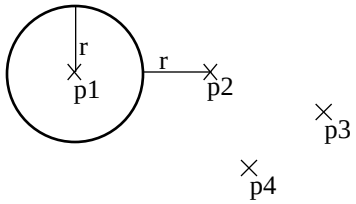

Figure 1: The closest pattern in the training set to p1 is p2, at a distance of 2r. Optimal performance occurs when all vectors within the hypersphere centred on p1 and radius r, are attracted to p1. If all patterns stored in a network exhibit this performance, its normalised average basin of attraction, *R*, is 1

4 Results

4.1 Attractor Basins and Loading

For all four models the size of the attractor basins decreases with loading in a roughly linear way. Figure 2 shows the results for *Symmetric Local Learning*, with a network of 100 units, unbiased patterns and increasing loading. The graphs for the other learning rules are similar. Differences in performance can only be seen by detailed examination of the results for particular loadings, as is seen in the next section.

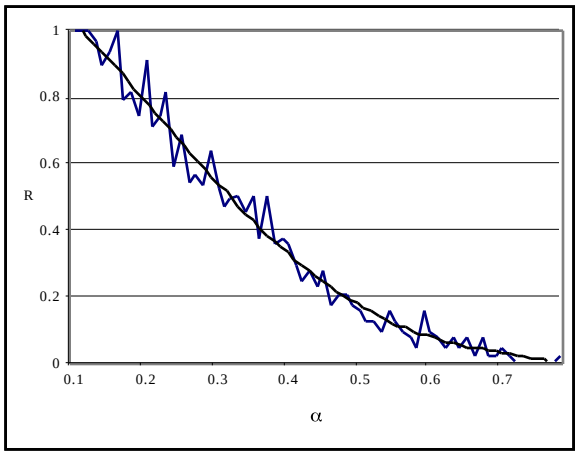

Figure 2: The basin of attraction size for the Symmetric Local Learning Network, with 100 units and 30 unbiased patterns. The dark line is a trend line, showing the fall of R with α. Graphs for the other networks show a similar pattern.

4.2 Comparative Performance

The next set of results, shown in Table 1, compares Local Learning (*LL*) with the symmetric version, *SLL*. In all cases the loading of the (100 node) network is α = 0.3, the patterns are unbiased and the results are averages over fifty runs. At this loading the theoretical value of κ_{max} is 1.27

	T	κ	R	Training Epochs
LL	1	0.84	0.57	7.7
LL	10	1.14	0.64	54.8
LL	100	1.18	0.63	500.6
SLL	1	0.80	0.54	11.6
SLL	10	1.14	0.65	35.6
SLL	100	1.18	0.65	307.8

Table 1: The comparative performance of local learning and its symmetric counterpart, under a loading of 0.3 (30 patterns in a 100 node network). The patterns are unbiased and the results averages over 50 runs.

It can be seen that the imposition of symmetry does not affect the attractor performance (R) of the network. Moreover the increase in T raises the value of κ but, interestingly, this does not improve attractor performance, in the change from T = 10 to T = 100, in either case. The actual value of κ obtained is much higher than the theoretical lower bound, which for this learning rule, at this loading is: 0.42 for T =1, 0.60 for T = 10 and 0.63 for T = 100.

The training time (epoch count) is increasing linearly with T, which is in accordance with the theoretical upper bound on training time [Abbott 1988]. However it is apparent that the convergence of SLL, at the higher values of T, is significantly faster than the non-symmetric version.

The results for the Krauth and Mezard rule, shown in Table 2, again with α = 0.3, unbiased patterns and the results averaged over fifty runs show a similar pattern to LL. The imposition of symmetry does not make much difference to R, with KM being marginally better than SKM. A comparison of Tables 1 and 2 shows the R values for LL to be similar to those for KM, although as in accordance with the theoretical result, the κ values are higher for KM, getting close to the theoretical maximum (1.27) with the highest threshold. The results do not contain the training epoch count as the algorithm does not take place in a simple epoch by epoch fashion.

	T	κ	R
KM	1	0.87	0.57
KM	10	1.19	0.66
KM	100	1.23	0.64
SKM	1	0.87	0.56
SKM	10	1.19	0.61
SKM	100	1.23	0.62

Table 2: The comparative performance of Krauth /Mezard local learning and its symmetric counterpart, under a loading of 0.3 (30 patterns in a 100 node network). The patterns are unbiased and the results averages over 50 runs.

4.2.1 Symmetry

It is interesting to look at the degree of symmetry in the weight matrices produced by the asymmetric versions of the learning rules. To this end the symmetry measure of Krauth, Nadal and Mezard [1988] was applied to the resulting weight matrices. It is defined as:

$$\sigma = \frac{\sum_{i,j} w_{ij} w_{ji}}{\sum_{i,j} w_{ij}^2} .$$

For a symmetric matrix this takes the value +1. For an anti-symmetric matrix it takes the value –1 and for a random set of weights it will be roughly zero. The results, in Table 3, show that the weight matrices produced for all thresholds are highly symmetric with the symmetry increasing with the threshold.

T	σ - LL	σ - KM
1	0.961	0.968
10	0.983	0.991
100	0.983	0.991

Table 3: Symmetry of LL and KM with alpha = 0.3, and unbiased patterns. Averages over 50 runs

5 Discussion

Symmetry of the weights in an associative neural network is a mixed blessing. Desirable from the perspective of dynamics, but with potentially damaging implications for the attractor performance. However as shown above, for both forms of learning rule, the addition of a symmetry constraint did not have an adverse affect on the attractor basins. The reason for this is probably that the matrices that result from unconstrained learning are already highly symmetric and become more so with a larger learning threshold, which is itself an interesting result.

It is also apparent that, for the LL rule, imposing a symmetry constraint during learning had a helpful affect on convergence – almost halving the training epochs required. In symmetric local learning, for each epoch, each weight is changed twice, so if both these changes are constructive in moving the weight towards its final value, the learning may, at best, be twice as efficient. If the LL weight matrix was actually strongly asymmetric, then it is improbable that the SLL double weight change would be constructive, and the observed halving of training epochs would not have occurred.

In conclusion, the best versions of these high capacity Hopfield networks are those with strictly symmetric weights, since they have simple dynamics (only point attractors in the phase space), and learn faster.

6 References

[1] Abbott, L. F. (1990). Learning in Neural Network Memories. *Network: Comp. Neural Sys*. 1:105-122

[2] Diederich, S. and M.Opper (1987). Learning of Correlated Patterns in Spin-Glass Networks by Local Learning Rules. *Physical Review Letters* **58**, 949-952

[3] Gardner, E. (1988). The space of interactions in neural network models, *J. Phys*. A **21**, 257-270;

[4] Gardener, E., H. Gutfreund and I. Yekutieli (1989). The Phase Space of Interactions in Neural Networks with definite Symmetry, *J. Phys.* A **22**.

[5] Kanter,I. and H. Sompolinsky (1987). Associative Recall of Memory Without Errors. *Physical Review* A **35**, 380-392

[6] Krauth, W., J.-P. Nadal and M. Mezard (1988), The role of stability and symmetry in the dynamics of neural networks, *J.Phys.* A21, 2995-3011.

[7] Krauth, W. and Mezard, M. (1987), Learning algorithms with optimal stability for neural networks, *J.Phys*. A **20** L745-L752.

[8] Personnaz,L., I.Guyon and G.Dreyfus (1986). Collective Computational Properties of Neural Networks: New Learning Mechanisms. *Physical Review* A **34**, 4217-4228

[9] Storkey,A., and R. Valabregue (1999) The basins of attraction of a new Hopfield learning rule. *Neural Networks* **12**(6), 869 – 876.