A Comparative Analysis of High Performance Associative Memory Models

N.Davey, S.P.Hunt
Department of Computer Science,
University of Hertfordshire
Hatfield, Herts. AL10 9AB. United Kingdom
E-mail: {N.Davey, S.P.Hunt}@herts.ac.uk
WWW: http://www.cs.herts.ac.uk/~nngroup/

Abstract

Three variants of the Hopfield network are examined, each of which is trained using a different iterative approximation of the pseudo-inverse rule. All three variants are known to have significantly higher memory capacity than the standard Hopfield model. The performance measure employed in this study is the average size of attractor basins. The three models are tested with both biased and unbiased random data and under different memory loads. We find that a model employing an iterative local learning regime, based upon the least mean squares learning rule, gives the best attractor performance, albeit at the expense of increased training times.

1 Introduction

Associative memories based upon the original Hopfield [1982] model suffer from low capacity, and have been demonstrated to have poor attractor performance [e.g. Storkey & Valabregue, 1999; Davey & Hunt, 1999].

If capacity were the only consideration in choosing an associative memory model, then it would be tempting to adopt Hopfield's architecture and train it using the pseudo-inverse rule [Kanter & Sompolinsky, 1987], to give a memory with a maximum capacity of $N - 1$ patterns (where N is the number of nodes in the network). However, there are other considerations:

- whether or not the learning rule is local;
- the ability to add patterns to a memory incrementally;
- size and shape of attractor basins;
- the network's ability to correctly recall memories when trained on sets of correlated patterns;
- the time it takes to train the network;
- the time it takes the network to converge during recall.

A learning rule is local if the amount by which a connection weight, w_{ij}, is to be updated at some time t, may be calculated entirely from the information available to the processing elements at either end of that connection at time t. Kanter and Sompolinsky's [1987] original learning rule involves the calculation of a pseudo-inverse, so it is not local; nor does it allow for the incremental addition of patterns to a memory.

In this paper, we investigate the performance of three high capacity variations of the basic Hopfield model of associative memory. All three models employ learning rules based around approximations to the pseudo-inverse rule, and so have high capacity. Two out of the three rules are completely local; all three allow the incremental addition of patterns to a memory; and all three guarantee the correct recall of trained patterns.

The first two models, iterative local learning and iterative local learning with adjustable thresholds (which we will refer to as LL and Adj LL respectively), have already been demonstrated experimentally to have high capacity and good attractor performance [Davey & Hunt, 1999]. The third model, iterative local learning with equal fields (referred to as LL-Equal), is a refinement of the LL model which was not examined in the earlier work, but which we have also confirmed has high memory capacity.

The investigation presented here is focussed on the size of attractor basins, the performance of networks trained on highly correlated patterns, and the time taken to train the networks under investigation.

2 Models Examined

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

All the high capacity models studied here are modifications to the standard Hopfield network. The net input, or local field, of unit i, 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. The weight matrix is calculated using one-shot Hebbian learning:

$$w_{ij} = \frac{1}{N} \sum_{p=1}^{\Pi} \xi_i^p \xi_j^p$$

2.1 Iterative Local Learning (LL)

In this model we employ a learning rule proposed by Diederich and Opper [1987]. The Hebbian learning rule is modified to create a simple iterative scheme that will converge to a weight matrix in which all the training patterns are guaranteed to be stable. Learning here is iterative and may take several epochs.

For training patterns to be stable, a sufficient condition is that:

$$\mathbf{W}\xi^p = \xi^p \text{ for all } p \text{ (assuming that all } \theta_i = 0),$$

or, equivalently, for $\mathbf{W}\Xi = \Xi$ where Ξ is the matrix whose columns are the ξ^p.

One solution to this is given by the projection learning rule [Personnaz et al, 1986] which gives $\mathbf{W} = \Xi\,\Xi^I$ where Ξ^I is the pseudo-inverse of Ξ. Fortunately there is an iterative, and local, learning rule which converges to the appropriate weight matrix. This is described by the following algorithm :

Begin with a zero weight matrix

Repeat until all patterns are stable

Set the initial state of network to one of the ξ^p

For each unit, i, in turn

Calculate the net input to unit i, and hence its next state.

If i wishes to change state, update its incoming weights according to:

$$\Delta w_{ij} = \frac{S_i^p S_j^p}{N-1}$$

This algorithm will find a suitable weight matrix if one exists [Diederich and Opper, 1987], at which point the trained patterns are guaranteed to be stable. The maximum useful capacity of such a network is $N-1$ linearly independent patterns.

1.2 Iterative Local Learning with Adjustable Thresholds (Adj LL)

In the normal Hopfield network, all thresholds are set to zero. In the Adjustable Threshold Network [Schultz, 1995] an attempt is made to use thresholds to maximise network performance once a weight matrix is in place.

The motivation for this method can be seen from a consideration of a unit, i, in a trained network, whose local field for each of four training vectors is: -3, -1, 5 and 7. If i has a zero threshold, its state will be –1 for the first two patterns and +1 for the latter two. Schultz proposes moving the threshold for each unit so that it gives maximum separation between positive and negative local fields; in this case that would mean putting the threshold half way between –1 and +5, giving θ_i = +2. In general, the threshold for a unit, i, is set half way between the positive and negative fields with the smallest magnitudes, as given by:

$$\theta_i = \frac{h_i^+ + h_i^-}{2}, \quad \text{where} \quad \begin{aligned} h_i^+ &= \min\left\{h_i^p \mid h_i^p \geq 0\right\} \\ h_i^- &= \max\left\{h_i^p \mid h_i^p < 0\right\} \end{aligned}$$

If threshold adjustments are performed in this manner, training patterns that were stable before adjustment should remain stable after adjustment. Thus, variable thresholds should be usable with any learning rule.

The weight matrix for an adjustable threshold iterative local learning (Adj LL) network is determined in the manner set out in section 2.1, and the thresholds are then adjusted as described above. Adjusting thresholds in this way should not change the stability of the learned patterns, so the maximum capacity of this model should also be N-1 linearly independent patterns.

1.3 Iterative Local Learning with Equal Fields (LL-Equal)

Diederich and Opper [1987] propose a variant of their iterative learning rule, in which the network is trained so that the local field of each unit, for each trained pattern, is +1 or –1 as appropriate. In contrast to the first learning rule, all weights are updated on every presentation according to:

$$\Delta w_{ij} = \frac{\left(1 - h_i^p S_i^p\right) S_i^p S_j^p}{N}$$

This continues until the summative error, given by:

$$\sum_{i,p} \left| 1 - h_i^p S_i^p \right|$$

is sufficiently low (we take 0.1 as our convergence threshold). Diederich and Opper [1987] demonstrate that this learning rule will converge for up to N linearly independent training patterns. Adjusting the thresholds of such a network is pointless: when all local fields are +1 or –1 the threshold of 0 is already correctly placed.

3 Measurement of Performance

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

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

We also investigated the effect on attractor performance of increasing the degree to which the patterns in the training set were correlated. In order to do this, we fixed the size of the training set at 30, and trained the networks on sets of random patterns with different degrees of bias, b, where:

$$i,p \bullet prob\,(\xi_i^p = +1) = b, \text{ and}$$

$$b \quad \{0.1, 0.2, 0.3, 0.4, 0.5\}$$

Since we employ bipolar (±1) training vectors, we would expect a training set with bias b to give the same results as a training set with bias 1–b; thus, those training sets which were generated with b = 0.5 may be said to be *unbiased*, and those with b = 0.1 may be said to be *highly biased*. Furthermore, the smaller we make the value of b, the more highly correlated the (random) patterns in the training set become.

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. Thus, an important 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,

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

,

can act as a measure of the quality of a particular associative memory. It is also common for R to 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 of 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:

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

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:

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

,

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 the members of the training set (low average *m*). 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 be undertaken when these patterns are themselves stable.

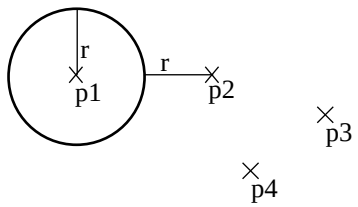

Figure 1 The closest pattern in the training set to p1 is p2, at a distance of 2r. Optimal performance occurs when all vectors within the hypersphere centred on p1 and radius r, are attracted to p1. If all patterns stored in a network exhibit this performance, its normalised average basin of attraction,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

4 Results

4.1 Attractor Basins and Loading

The first set of results, shown in Figure 2 (over page), illustrate how the value of *R* decreases as the loading of the networks increases. After a period of near perfect performance under low loading, all three models show a roughly linear fall-off. For unbiased patterns, the performance difference between the different networks is small. For example, when each network was trained with sets of 50 unbiased patterns, the value of *R* was as shown in Table 1

***R* Values**		
LL	*Adj LL*	*LL-Equal*
0.192	0.196	0.208

Table 1 Comparison of attractor basin size for sets of 50 unbiased patterns (averaged over 50 training sets in each case)

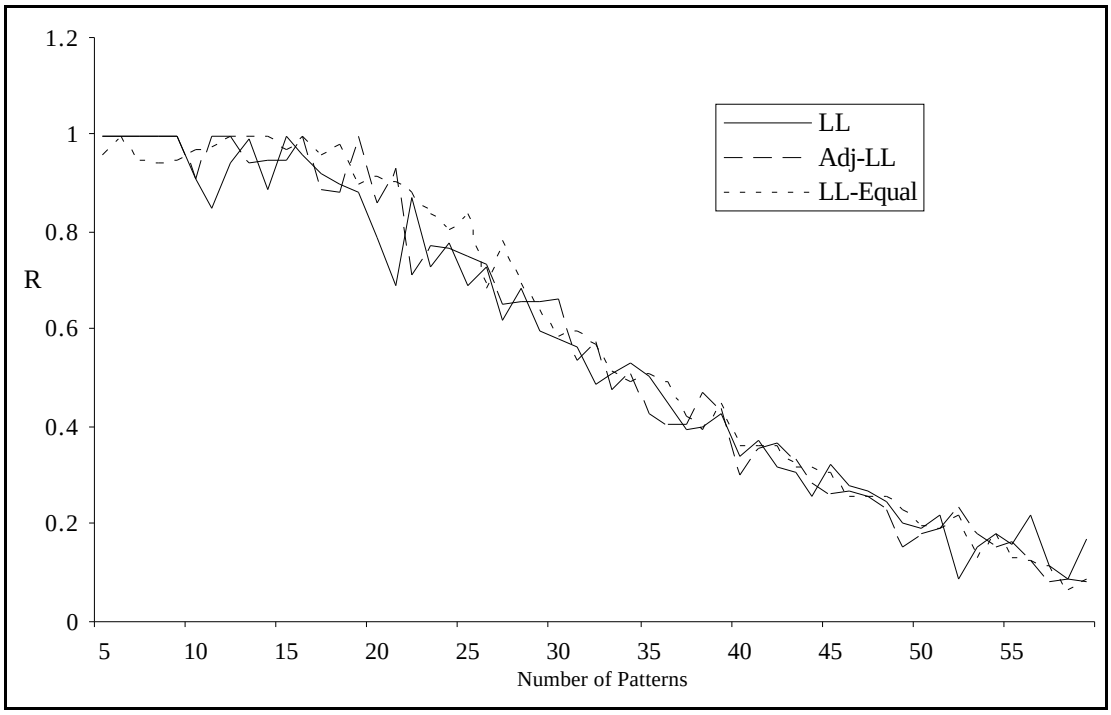

Figure 2. 100 unit networks and random $prob(\xi_i^p = +1) = 0.5$ patterns. Results are the mean of 2 runs, at each loading for each network. The vertical axis shows the normalised average basin of attraction, *R*

4.2 Comparative Performance

The next set of results, shown in Table 2 and Figure 3 illustrate how the *R* values for the three network models vary as the patterns in the training set become more correlated. Somewhat surprisingly, as the extent to which patterns are correlated increases (i.e. the bias *decreases*), the *R* values increase in all three models. However, for the highly correlated patterns (bias = 0.1), both LL and Adj LL models show a marked fall-off in performance. The LL-Equal model also shows some loss, but to a much lesser extent. As the training patterns become more correlated the performance difference between LL-Equal models and the other two variants becomes clearer and clearer.

Bias	*LL*	*Adj LL*	*LL-Equal*
0.5	0.558	0.576	0.615
0.4	0.613	0.59	0.644
0.3	0.706	0.697	0.736
0.2	0.869	0.850	0.930
0.1	0.399	0.408	0.796

Table 2 *R* values calculated for the three high capacity models of 100 units, with 50 training sets of 30 patterns, for each bias.

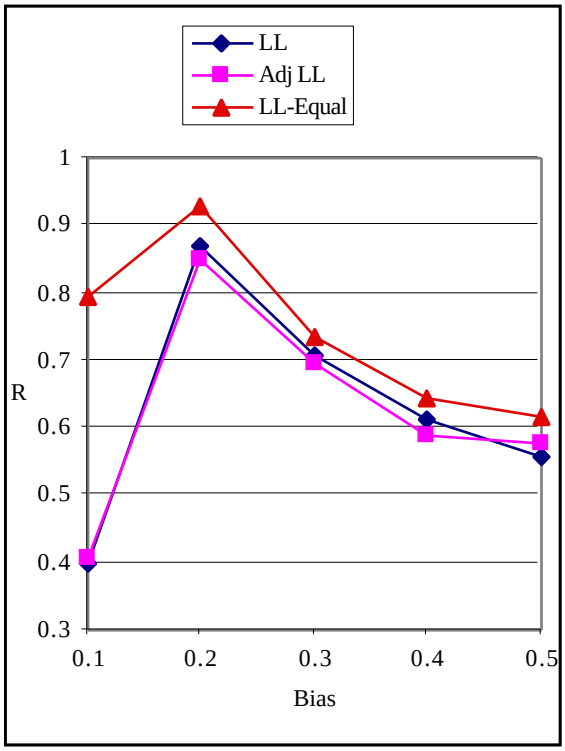

Figure 3 Attractor performance of training sets with different levels of pattern bias.

1.3 Effects of Sample Size (Γ)

Since *R* is measured statistically, it is important to investigate the effects of the sample size in the calculation of *R*. Using training sets of 30 patterns biased at 0.3, *R* was measured, firstly using the standard value of Γ = 50, and then with a requirement of 100 sampled states being correctly attracted. As is expected, the value of R computed for all three models is lowered. However, the differential performance advantage of LL-Equal is increased by this more detailed analysis. Further experiments are underway to investigate the asymptotic behaviour of R as Γ is increased.

Γ	*LL*	*Adj LL*	*LL-Equal*
50	0.706	0.697	0.736
100	0.474	0.489	0.547

Table 3 *R* values calculated for the three high capacity models of 100 units, with 50 training sets of 30 patterns with bias 0.3, for different sample sizes, Γ.

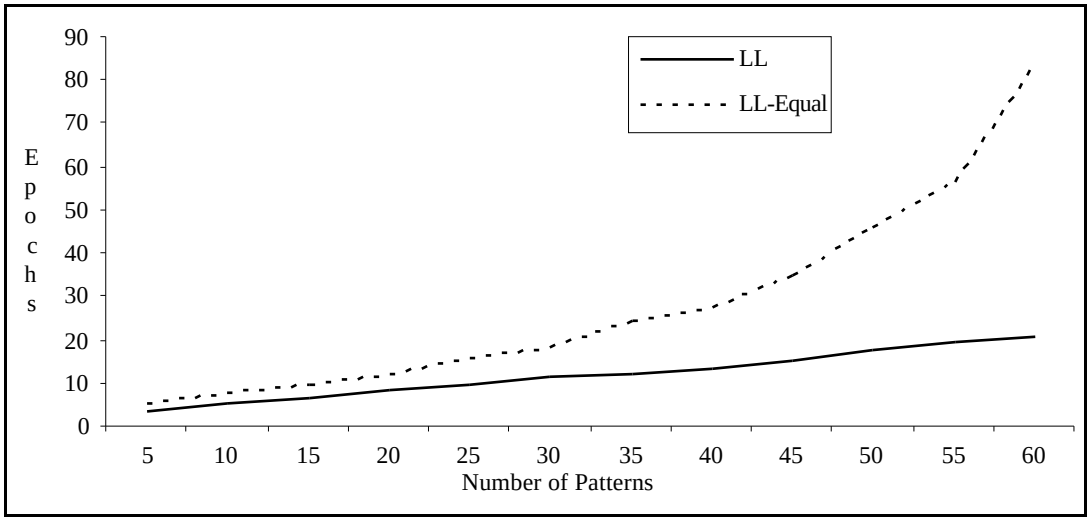

Figure 4 The number of training epochs required by the two iterative training methods. For both networks the reported figures are averages over five data sets.

1.4 Training Times

As already mentioned, the local learning rules are incremental. Figure 4 shows the number of epochs required to reach a weight matrix that gives stability for all patterns in the training set. Results are averaged over five runs at each increment in Π.

It is apparent that the standard iterative local learning rule gives a linear increase in the required number of training epochs as the number of training patterns increases. The second form of iterative local learning, however, shows an exponential increase in training time as the size of the training set is increased.

5 Discussion

We have performed a systematic analysis of attractor performance of the three models, and found that all of them perform well. Up to N-1 patterns are stable and will act as attractors at all loading levels. Adjusting the thresholds of an iterative local learning network does not appear to bring benefit. This, coupled with the fact that it is inherently non-local (in that it depends upon information gathered over many time steps), leads us to conclude that threshold adjustment is not a worthwhile technique for improving the performance of Hopfield-type associative memories. However, the LL-Equal model shows an improvement in performance over the straightforward LL model. Interestingly, the differences in attractor performance are most marked when the models are required to deal with sets of highly correlated training vectors.

It was apparent from our earlier work [Davey & Hunt, 1999] that the LL model had higher capacity and better attractor performance than both the original Hopfield model and the model proposed by Storkey and Valabregue [1997, 1999]. What is clear from this series of experiments is that, whilst the attractor performance of the LL model is very good, the LL-Equal model is considerably better, the only drawback to adopting this model being the increase in training time.

Investigation of the learning rules employed in the LL and LL-Equal models gives a clue as to the reasons for the improvement in performance. The LL rule is analogous to the Perceptron Convergence Procedure [see Haykin, 1994, p.109], whilst the LL-Equal rule is analogous to the Least Mean Square algorithm, or delta rule [see Haykin, 1994, p.127]. Thus, in the LL case, the decision boundary of each processing element is moved until it produces an acceptable set of classifications, whereas, in the LL-Equal case, a form of *error correction* is employed to place the decision boundary of each processing element in an 'ideal' position. It is therefore not entirely surprising that the LL-Equal model has superior attractor performance.

As is well known, error-correction learning rules can be costly in terms of training time, particularly when the error target is very small. This is borne out by the analysis of training times here. The LL model achieves an approximation to the pseudo-inverse weight matrix in linear time.

The addition of a requirement to drive all fields to unity appears to result in learning times that rise exponentially with increases in training set size. It remains to be seen how great an impediment this will be to the adoption of this rule for the training of associative memories to perform real-world tasks.

6 References

[1] Davey, N. and S.P.Hunt (1999)
The Capacity and Attractor Basins of Associative Memory Models.
Proceedings of the 5th International Work Conference on Artificial and Natural Neural Networks, IWANN 99 (J. Mira and J.V. Sánchez-Andrés, eds.) Vol 1, pp 330-339, Berlin: Springer-Verlag.

[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] Haykin, S. (1994)
Neural Networks: A Comprehensive Foundation
New York, NY: Macmillan

[4] Hopfield, J.J. (1982)
Neural networks and physical systems with emergent collective computational abilities.
*Proceedings of the National Academy of Sciences of the USA, **79**, 2554-2558*

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

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

[7] Schultz,A. (1995)
Five Variations of Hopfield Associative Memory Network.
Journal of Artificial Neural Networks **2**(3), 285-294

[8] Storkey,A., and R. Valabregue (1997)
Hopfield Learning Rule with High Capacity Storage of Time-Correlated Patterns.
Electronics Letters **33**(21), 1803-1804

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