High Performance Associative Memory and Weight Dilution

N.Davey, R.G.Adams, S.P.Hunt

Department of Computer Science,
University of Hertfordshire,
College Lane, Hatfield, AL10 9AB. United Kingdom
N.Davey@herts.ac.uk, R.G.Adams@herts.ac.uk, S.P.Hunt@herts.ac.uk

Abstract *The consequences of diluting the weights of the standard Hopfield architecture associative memory model, trained using perceptron like learning rules, is examined. A proportion of the weights of the network are removed; this can be done in a symmetric and asymmetric way and both methods are investigated. This paper reports experimental investigations into the consequences of dilution in terms of: capacity, training times and size of basins of attraction. It is concluded that these networks maintain a reasonable performance at fairly high dilution rates.*

Key-Words Associative Memory, Hopfield Networks, Weight Dilution, Capacity, Basins of Attraction, Perceptron Learning.

1 Introduction

Neural networks designed to function as associative memories are usually based around the standard Hopfield architecture. It has been known for some time [1] that a variety of local learning rules can produce models with much better performance than the original Hebbian learning, proposed by Hopfield. These learning rules either find an approximation to the projection weight matrix or employ perceptron style learning.

Diluted networks are versions of their fully connected progenitors, but differ in that only a fraction of the neural connections are maintained. Of course the human brain represents such a style of connectivity: each neuron is connected to roughly 10,000 other neurons, but the total number of neurons is of the order of 10 million times greater. Networks may be diluted by post training pruning, in which it is hoped that the most efficacious connections are kept [2,3]; it has even been suggested that a network with random fixed weights can be trained by the systematic removal of a fraction of the weights [10]. Alternatively a proportion of the weights may be randomly removed before training, and this is the approach taken here. For one shot learning schemes, such as simple Hebbian learning, the two approaches are equivalent and it is known [11] that capacity drops linearly with the fraction of synapses removed. In this paper we examine the effect of pre-training synapse removal on associative memory networks, trained using perceptron style local learning.

2 Models Examined

In each experiment we take a network of N units which we train with a set of N–ary, bipolar (+1/-1) training vectors, $\{\xi^p\}$. The N by N weight matrix is denoted by $\mathbf{W}$, and the state (output) of the i'th unit is denoted by S_i

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:

$$h_i = \sum_{j \neq i} w_{ij} S_j$$

where w_{ij} is the weight on the connection from unit j to unit i. The *next* state of a unit is derived from its local field and its *current* state:

$$S_i' = \begin{cases} 1 & \text{if } h_i > \theta_i \\ -1 & \text{if } h_i < \theta_i \\ S_i & \text{if } h_i = \theta_i \end{cases}$$

where the threshold, θ_i, is normally taken as zero. Unit states may be updated synchronously or asynchronously. Here we use asynchronous, random order updates. These network dynamics and a symmetric weight matrix guarantee simple point attractors in the network's state space.

A training vector, ξ, will be a stable state of the network if the *aligned local fields*, $h_i\xi_i$ are non-negative for all i (assuming all θ_i are zero). Each training vector that is a stable state is known as a *fundamental memory* of the trained network. The *capacity* of a network is the maximum number of fundamental memories it can store. The *loading*, α, on a network is calculated by dividing the number of vectors in the training set by the number of units in the network, N.

2.1 Learning Rules

In the late 1980s it was demonstrated that perceptron like learning could be used in associative memory networks to produce much higher capacity than the basic model. In fact, as Gardner [9] showed, a Hopfield type network of N units may store up to $2N$ uncorrelated patterns (a loading, α, of 2), with this figure 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.

The training patterns will be stable if T is non-negative (see section 2) and, for ease of training, a value of 1 (or even 0) may be taken. However, by raising T we may improve the attractor performance of the network [1]. Some care must be taken though. Consider a network in which all training patterns are stable ($h_i\xi_i \geq T$ for all patterns and units): any uniform, upward scaling of the weight matrix will increase the aligned local fields, but will obviously not improve the attractor performance. Optimal attractor performance is achieved when the threshold is maximised with respect to the size of the weights, so the relevant characterization 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

$$\kappa = \min_{p,i}(\gamma_i^p).$$

1.1.1 Local Learning (LL)

Diederich and Opper's [8] 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 $\xi^p$
  For each unit, i, in turn
    Calculate $h_i^p \xi_i^p$.
    If this is less than T then change the weights
    on connections into 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 [7], at which point the trained patterns are guaranteed to be stable. We refer to this as the LL (local learning) rule.

As shown by Abbott [1], this rule 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.

1.1.2 Symmetric Local Learning

The Hopfield network has a symmetric weight matrix. Such weight matrices have the desirable property of implying point attractors with asynchronous updating and cycles of length at most 2 with synchronous updates. As the symmetry is broken, more complex dynamics become progressively more likely. On the other hand Krauth, Nadal and Mezard [9] showed that, under certain circumstances, decreasing the symmetry of the weight matrix should improve attractor performance. Gardner [8] 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. Gardner also showed that such algorithms would find a symmetric weight matrix, if one existed, for a particular training set. The SLL (symmetric local 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 $\xi^p$
 For each unit, i, in turn
  Calculate $h_i^p \xi_i^p$
  If this is less than T then change the weights on
  connections into and out of unit i according to:
```

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

1.2 Dilution

In the scheme adopted here a fraction of the weights of the network are set to a constant value of zero (effectively removed from playing any part in the network dynamics). This may be done in such a fashion that the symmetry of the connection matrix is maintained, that is if w_{ij} is removed then so is w_{ji}, or alternatively in a completely random way. We use both approaches. If symmetry is maintained in dilution, subsequent training uses symmetric local learning, otherwise normal perceptron style learning is used. The dilution rate, *d*, is the proportion of weights that are removed prior to training.

3 Analysing Performance

For an associative memory model to be effective, the training patterns should not only be stable states of the network, but should also act as attractors in the network's state space.

As stated above, the perceptron type learning rules will store a set of training vectors in the network when the aligned local fields of those vectors have all been driven to be non-negative. Moreover, the larger these aligned local fields become, the better the attractor performance should be. Therefore we examine the performance of our networks by varying both the loading, α, and the training threshold, T.

We also consider the effect of correlations in the training patterns. An uncorrelated training set is one in which the patterns are completely random. Correlation can be increased by varying the probability that a given bit in a training pattern is +1 (or –1). We refer to the probability of any bit being +1 in each the training vector as the *bias*, b, on the training set. So: *i,p* • *prob* (ξ_i^p = +1) = b, Thus, a bias of 0.5 corresponds to an uncorrelated training set and a bias of 1 corresponds to a completely correlated one, as does a bias of 0.

To measure the capacity of diluted networks trained using repeated Hebbian learning is not straightforward. The approach taken here is to perform an incremental search, with gradually increasing α. At each increment in α, 10 random sets of training vectors, each containing αN patterns, are formed and the network is required to separately learn all ten training sets. We define the capacity as the largest value of αN for which this is achieved.

We use, *R*, the normalized mean radius of the basins of attraction [5], as a measure of attractor performance. It is defined as

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

where m_0 is the minimum overlap an initial state must have with a fundamental memory for the network to converge on that fundamental memory, and m_1 is the largest overlap of the initial state with the rest of the fundamental memories. The angled braces denote an average over sets of training patterns. Details of the algorithm used can be found in [5].

The training time of the local learning rules is reported as the number of epochs (complete presentations of the training set) required for convergence.

Finally, 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 [9] 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.

4 Results

4.1 Capacity

As described in the previous section, to find the capacity of the diluted networks we search for the point at which the learning rule fails to converge, when presented with ten sets of patterns at the given loading. We investigate 100 unit networks with dilution rates varying from 0 to 0.9 in increments of 0.1. The symmetric learning rule, SLL, is used here, with T=1. All training sets are unbiased (b = 0.5).

The results (Figure 1) show a similar pattern to that reported for one-shot Hebbian learning [11]: a roughly linear decrease in capacity with increasing dilution. Interestingly the capacity of this form of network with 80% of the connections removed is roughly equivalent to a fully connected standard Hopfield network. Note also that for all dilutions up to 0.6, at least 30 patterns are learnable by the network.

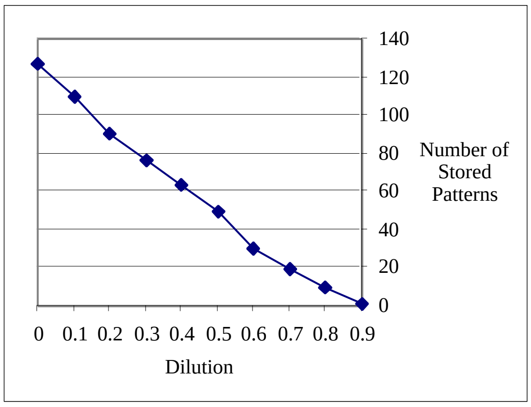

Figure 1: Capacity of diluted networks (N = 100) trained with the SLL rule.

In the next two sections we examine the performance of diluted networks in more detail. In each experiment 100 unit networks are trained with 30 random patterns. Each training run is repeated 50 times with a different training set each time and the results are averages over these 50 runs.

4.2 Effect of Varying Training Threshold

In this section we present the results of varying the learning threshold, T, for networks trained using both the LL and SLL rules. In all cases dilution of 0.4 is considered, since networks with this many connections have a capacity well in excess of the 30 patterns we wish to store, as shown in Figure 1.

4.2.1 Attractor Performance

Table 1 shows how the attractor performance changes for the network with d = 0.4, as the learning threshold is increased. As a base case the undiluted version of the network is also given. Consider first the results for the non-symmetric networks (LL). It is immediately apparent that the effect of dilution is to lower the κ value and correspondingly lower the R values. Increasing the learning threshold from 1 to 10 does improve the R value slightly, but the R value does not approach that of the undiluted network. Increasing the learning threshold further (from 10 to 100) does not appear to bring benefit.

The symmetric networks (SLL) show a similar pattern. However their performance is inferior to the non-symmetric versions.

Network	**T**	κ	**R**
LL (d = 0)	1	0.83	0.56
LL (d = 0.4)	1	0.55	0.23
LL (d = 0.4)	10	0.68	0.26
LL (d = 0.4)	100	0.67	0.23
SLL (d = 0)	1	0.80	0.55
SLL (d = 0.4)	1	0.53	0.10
SLL (d = 0.4)	10	0.62	0.11
SLL (d = 0.4)	100	0.63	0.11

Table 1: Attractor performance of diluted networks, under a loading of 0.3 (N = 100). Training sets are unbiased (b = 0.5) and results are averages over 50 runs.

4.2.2 Training Times

Table 2 shows how the training time varies as T is increased. Again the undiluted networks are shown for comparison. The symmetric and non-symmetric versions take a similar number of epochs to train. With the threshold, T, at 1, the effect of dilution is to significantly increase the training time, when compared with the undiluted networks. Moreover, as T is increased the training time increases in a roughly linear way. This pattern is also seen in undiluted networks [4].

Network	T	Epochs
LL	1	10.32
LL (d = 0.4)	1	27.63
LL (d = 0.4)	10	184.47
LL (d = 0.4)	100	1941.84
SLL	1	8.26
SLL (d = 0.4)	1	27.11
SLL (d = 0.4)	10	195.53
SLL (d = 0.4)	100	1881.84

Table 2: Training times for diluted networks under a loading of 0.3 (N = 100). Training sets are unbiased and results are averages over 50 runs.

4.2.3 Symmetry

Finally the symmetry of the asymmetric networks is examined. If a network is randomly diluted at a rate of d = 0.4, but the remaining weights are symmetric, we would expect σ (our measure of symmetry) to be roughly 0.6. As can be seen from Table 3, the weight matrix for the undiluted network is very nearly symmetric, but σ is significantly less than 0.6 for each of the diluted networks. The inference we draw from this is that the learning rule is introducing greater asymmetry to the weight matrix in order to cope with the asymmetric dilution. This degree of asymmetry is likely to be a problem for these types of networks, as symmetry is necessary for prohibiting non-point attractors. So when the heavily diluted LL networks are run they often reach multi-point orbits, which are difficult to identify.

Dilution	T	σ
0.4	1	0.49
0.4	10	0.49
0.4	100	0.48
0.0	1	0.96

Table 3: Symmetry of weight matrices in networks trained with LL. Averages over 50 runs.

4.3 Effect of Varying Dilution and Bias

In this section we examine how the attractor performance and training times change as the dilution rate is varied. The SLL rule is used here due to the difficulty of measuring R for highly diluted networks trained with the LL rule, in which the dynamics are increasingly complicated (see above). Training sets which are unbiased (b = 0.5) and correlated (b = 0.7) are used.

4.3.1 Attractor Performance

It can be seen that the R values decrease with increasing dilution and that the networks perform better with correlated patterns, regardless of the amount of dilution. Once again, this also holds for undiluted networks [5].

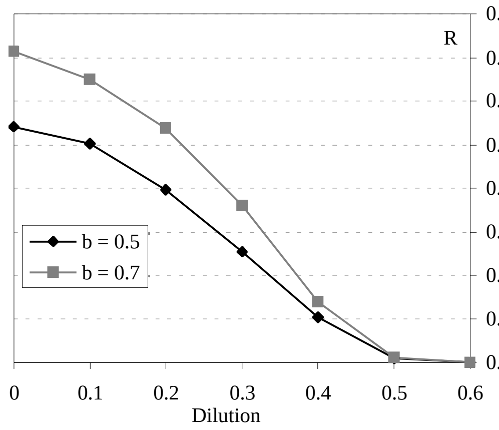

Figure 2: Attractor performance of networks trained using SLL, under a loading of 0.3 (N = 100) and varying dilution. Patterns are either unbiased (b = 0.5) or correlated (b = 0.7). Results are averages over 50 runs.

4.3.2 Training Times

Finally the effect of increasing dilution on training times is given. Increasing dilution increases the training time, the bias of the training patterns does not have a significant effect.

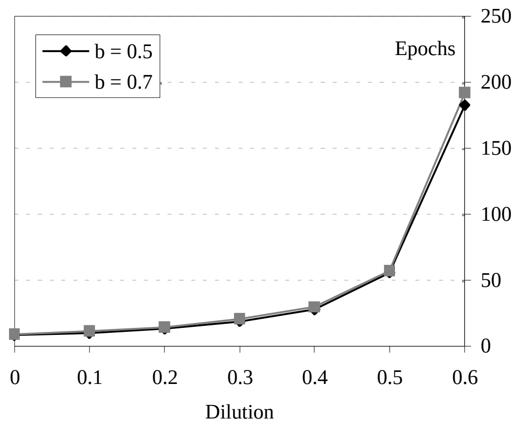

Figure 3: Training times for networks trained using SLL, under a loading of 0.3 (N = 100) and varying dilution. Patterns are either unbiased (b = 0.5) or correlated (b = 0.7). Results are averages over 50 runs.

5 Conclusions

Dilution of the weights in a high capacity associative neural network is interesting from both the neurophysiological perspective and from an engineering point of view, in which the number of connections can be viewed as a resource to be minimised. There are at least two ways in which pre training dilution can be undertaken in such networks, either maintaining symmetry or not. In the latter case the asymmetry of the remaining weights causes problems with the network dynamics, as discussed in section 4.2.3.

The capacity of the SLL networks is shown to decrease linearily with the rate of dilution, a similar pattern to that of networks trained with one-shot Hebbian learning. However the SLL network maintains a relatively high capacity for diltution rates up to 80%. The attractor performance of the diluted networks is poorer than the undiluted counterparts, and although increasing the learning threshold does improve performance it is not possible to recover to the level attained by fully connected networks, and training times are significantly increased.

The presence of correlated training patterns is not a problem for these networks, indeed the attractor performance is actually better for biased patterns, as shown in section 4.3.1.

An interesting question that it has not been possible to explore here is whether a symmetric dilution policy together with the asymmetric learning rule would bring benefit. The low symmetry of the LL networks suggests that this is a possibility worthy of exploration.

References:

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

[2] Barbato,D.M.L. and O.Kinouchi, (2000) Optimal pruning in neural networks, *Physical Review E* **62**(6), 8387-8394

[3] Chechik,G., I.Meilijson and E.Ruppin (1998) Synaptic Pruning in Development: A Computational Account, *Neural Computation* **10**, 1759-1777

[4] Davey,N. and R. Adams (2001). High Performance Associative Memory Models and Sign Constraints *Proceedings of Neural Networks and Applications (NNA 2001)*, 416-420, 2001

[5] Davey,N. and S.P.Hunt (2000). A Comparative Analysis of High Performance Associative Memory Models. *Proceedings of 2nd International ICSC Symposium on Neural Computation (NC 2000)* 55-61

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

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

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

[9] 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.

[10] López,B. and W. Kinzel (1997) Learning by dilution in a neural network, *J.Phys. A* **30** 7753-7764

[11] Sompolinsky,H. (1986), Neural Networks with nonlinear synapses and a static noise, *Physics Review* A **34**, L519-L523.