Optimal Dilution and High Capacity Associative Memories

Stephen Hunt, Neil Davey and Ray Frank
University of Hertfordshire – School of Computer Science
College Lane, Hatfield, Herts. AL10 9AB – United Kingdom

Abstract: *Associative memories with recurrent connectivity can be built from networks of perceptrons and trained using the perceptron learning rule. Recent work has shown how a perceptron can be diluted, so that incoming connections are removed to produce an optimal perceptron. Such a perceptron classifies its training set with optimal margin and minimal connectivity. Here this technique is applied to perceptrons in associative memory networks. The main result shows that, by using these dilution methods, effective associative memories can be built with very sparse connectivity.*

Keywords: Associative memory, Hopfield network, optimal dilution.

1. Introduction

It has long been known that the performance of an associative memory based on the Hopfield model can be significantly improved by substituting perceptron learning for one-shot Hebbian learning [1, 2]. Interest in models of this type has been renewed recently, particularly where the connectivity is sparse: so called *dilute* models [3-5]. Alongside this, there has been some interest in finding optimal dilution strategies for feed-forward networks of perceptrons [6-8]. In this paper we examine whether some of the suggestions for optimal dilution of perceptrons can also be effectively applied to associative memories. Our main result shows that, by using these dilution methods, effective associative memories can be built with very sparse connectivity.

2. The Associative Memory Model

In this work we use a sparsely-connected high-performance variant of the canonical Hopfield model. We start with a set of fully-interconnected threshold logic units (each unit receives inputs from every other unit, but no self-connections are allowed). We convert this to a sparsely-connected network by removing a proportion of the connections with no attempt to maintain symmetry of connectivity.

The net input, or *local field*, of a unit, is given by:

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

where S_j ($= \pm 1$) is the current state of unit j and w_{ij} is the weight on the connection from unit j to unit i (zero if no connection exists).

The standard update rule is used for calculating the next state of each unit in the network:

$$S_i' = \Theta(h_i)$$

where Θ is the Heaviside function. Units are updated *asynchronously in random order*.

Networks are trained using a variant of the perceptron convergence procedure:

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

$$\forall j \neq i \quad w'_{ij} = w_{ij} + \frac{\xi_i^p \xi_j^p}{K}$$

where $\boldsymbol{\xi}^p$ denotes one of the training vectors, K the number of incoming connections to each unit and T is the learning threshold which here has the value of 10, a value that was established to work well in [9]. All weights on removed connections are fixed at zero throughout. Whilst such a trained network will not satisfy the normal requirements for simple dynamics, in practice these models perform well [10].

3. Performance Measures

We are interested in how well these networks perform as associative memories. The capacity of such networks is determined by the number of incoming connections (K) that each processing element receives. For random pattern sets a threshold logic unit trained by the perceptron convergence procedure can learn up to $2K$ patterns [11]. Assuming roughly regular connectivity graphs (as is the case here) the capacity will be determined by the level of dilution and not the specific pattern of connections, and hence is not subject to empirical investigation.

We have employed two metrics to measure the potential pattern correction performance of these networks. The first is based on the set of *normalised stability measures* for a trained network [9]. The normalised stability measure, γ_i^p of a specific unit i with respect to a specific training pattern $\boldsymbol{\xi}^p$ is given by:

$$\gamma_i^p = \frac{h_i \xi_i^p}{|\mathbf{W}_i|},$$

where $\mathbf{W}_i$ is the incoming weight vector to unit i. The minimum of all the γ_i^p (denoted κ) gives a measure of the likely attractor performance:

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

This metric gives a good indication of performance in networks with random connectivity [10]. The size of the attractor basins of the trained patterns can also be estimated using R, *the normalised mean radius of basins of attraction*, 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 network's fundamental memories. The angled braces denote a double average over sets of training patterns (10 used in each case) and initial states. Details of the algorithm used can be found in [9]. A value of $R = 1$ implies perfect performance and a value of $R = 0$ implies no pattern correction.

4. Optimal Dilution of Perceptrons

Given a set of bipolar input vectors that can be divided into two linearly separable subsets the perceptron convergence procedure can be guaranteed to find a separating hyperplane. However, for a given classification task some of the inputs to the perceptron may be redundant. The optimally diluted perceptron has only the minimal number of incoming connections needed to perform the classification [6].

The first suggestion of a practical method for finding such a diluted perceptron was made in [7]. Their method was to first compute the covariance weights (obtained by one-shot Hebbian learning), and then remove the smallest of these computed weights. The resulting network was then retrained using the standard perceptron rule. Another suggestion [8] was to first learn the

classification with a fully connected perceptron, then to remove the smallest weights and finally to retrain the remaining weights.

As described in Section 2, a recurrent associative memory can be built from threshold logic units (which act as classifiers) trained using the perceptron convergence procedure. So, in this study we have examined how effective the two approaches to dilution suggested in [6] and [7] are when applied to associative memories constructed from perceptrons.

5. Dilution Strategies Employed in this Study

The four methods we use to choose a sparse connectivity matrix are as follows:

A) *Random dilution*. As a base case we examine networks with a random selection of connections removed prior to training.

B) *Covariance dilution*. The covariance weights are calculated by one-shot Hebbian learning and those with smallest magnitude are then removed.

C) *Perceptron dilution*. The network is fully trained using perceptron learning and the connections with smallest magnitude are then removed. The remaining connections are left unchanged.

D) *Perceptron dilution with zeroed weights*. As C) but the weights on those connections remaining after dilution are all set to zero.

In all four cases the network is (re)trained after dilution using perceptron learning as described above. The full dilution process therefore involves a first training phase followed by connection removal and finally a second training phase.

The motivation for investigating both method C and method D is that the weights that are formed by the first phase of training may not be appropriate as starting points for the second phase of training, given the revised pattern of connectivity.

6. Training sets

Throughout this work we employ training sets made up of pseudo-random training vectors. 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 training vector as the *bias*, b, on the training set. So: $\forall i,p \bullet prob\,(\xi_i^p = +1) = b$, 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).

We are interested in behaviour at below the saturation point of these networks, so we have confined ourselves to training sets containing 10 or 20 patterns. We have investigated the performance of networks trained with both completely random (bias=0.5) and heavily biased (bias=0.9) data, in order to see whether statistical bias in the data to be stored in an associative memory is likely to affect which dilution strategy is most likely to give optimal results.

7. Results

In this study we used 100 unit networks. In each case connections were chosen for removal according to one of the four dilution strategies laid out in Section 5. Dilution rates varied between 0 and 0.8, so at the highest rate of dilution 80% of the network connections were removed. Figure 1 shows how κ and R vary with dilution in networks trained with 10 patterns, whilst Figure 2 shows equivalent results for networks trained with 20 patterns. All results are averages of 10 runs at each dilution rate.

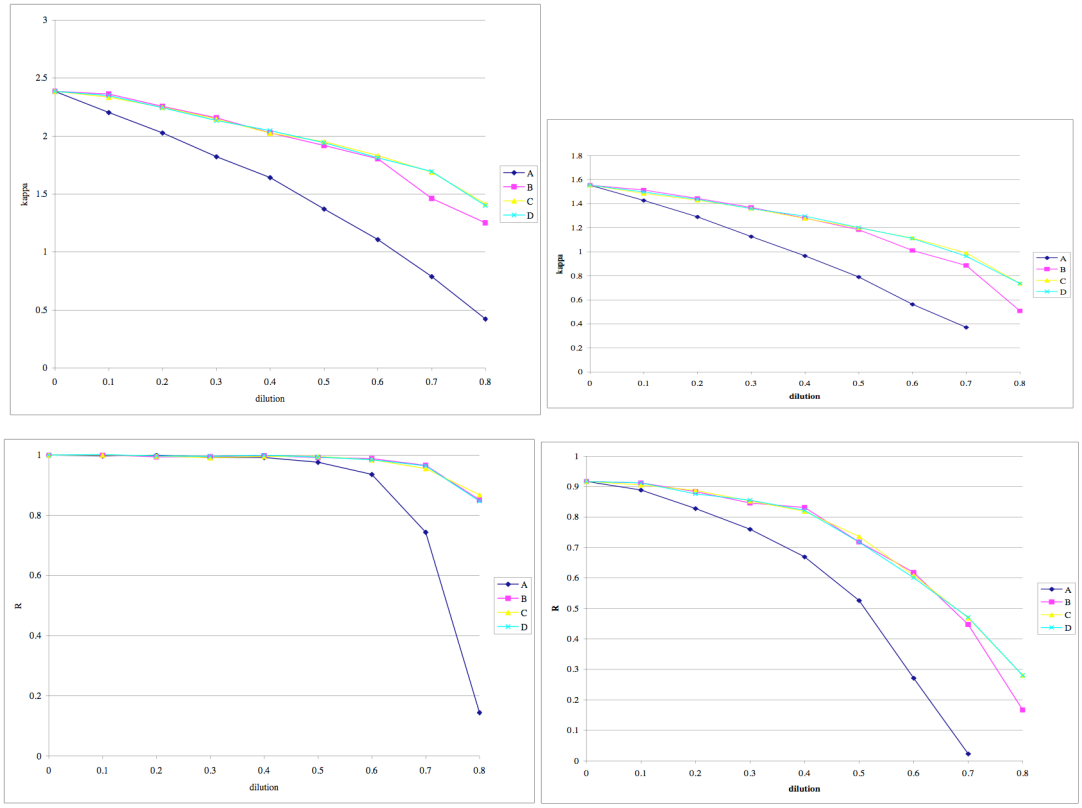

Figure 1: Effect of dilution on κ (top) and R (bottom) in networks trained with random training sets (bias=0.5). Left: loading = 0.1 (10 patterns). Right: loading = 0.2 (20 patterns).

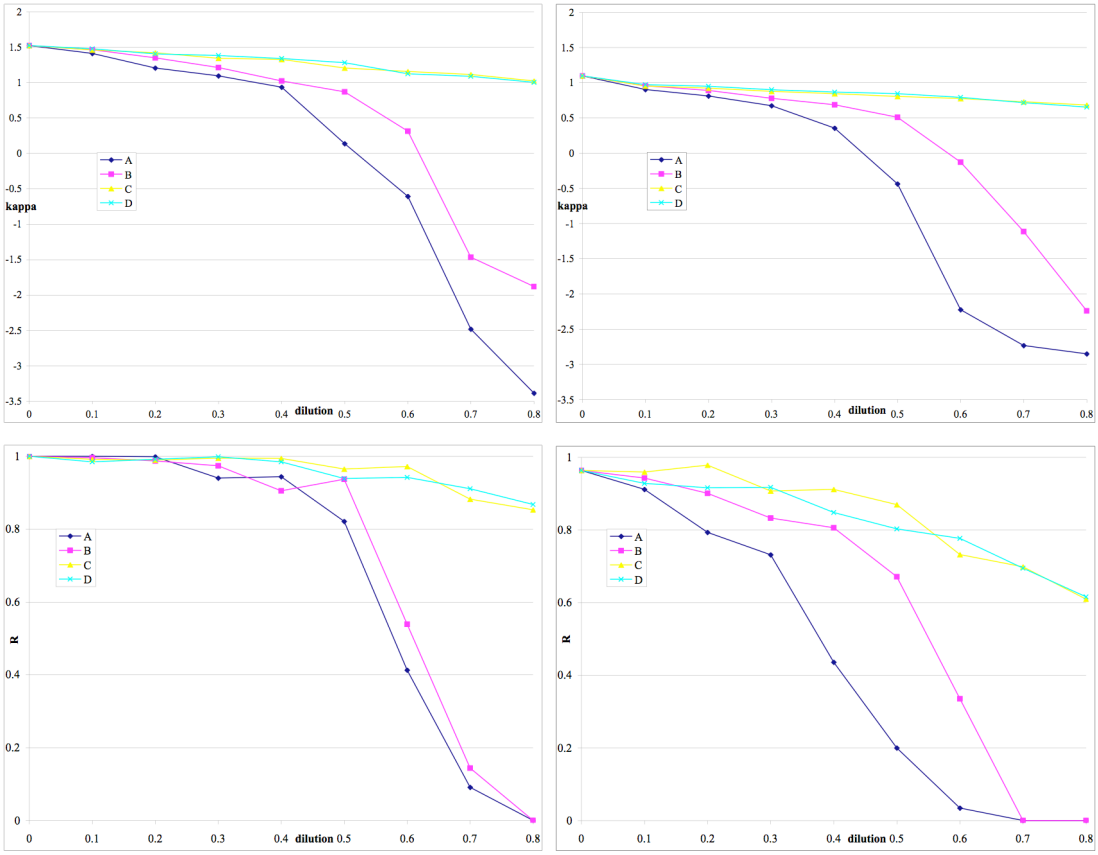

Figure 2: Effect of dilution on κ (top) and R (bottom) in networks trained with biased training sets (bias=0.9). Left: loading = 0.1 (10 patterns). Right: loading = 0.2 (20 patterns).

From the perspective of recall performance it is clear that there is little to choose between the three principled dilution methods (B,C and D) when the training data are unbiased and, as expected, all of them do better than the random method. However, when we switch to biased training data the picture changes, and method B (covariance dilution) performs significantly worse than either C or D. Both κ and R values show the same pattern. It can also be seen that there is no appreciable difference in the performance of networks diluted using methods C and D.

Whilst pattern correction performance in networks trained with 20 patterns is worse than it is in networks trained with 10 patterns across the board, the pattern of results is similar for both loading levels.

Alongside pattern correction performance we also considered the total training time for each method. Figure 3 shows how the four methods compared for networks trained with 20 patterns. It is clear that networks subjected to the random dilution strategy (A) require considerably more training than those where dilution is performed using one of the three principled methods. With unbiased data, only at very high dilution rates does the simple covariance dilution strategy (method B) give rise to a network that requires substantially more training time than either of the other two principled methods. Once again, the picture changes when biased data are used (so method B takes longer than C or D to train the network *and* the resulting network performs worse than either C or D). By comparison there is little to choose between the other two methods, except that networks appear to take longer to re-train under method D than method C for both unbiased and biased data at all levels of dilution.

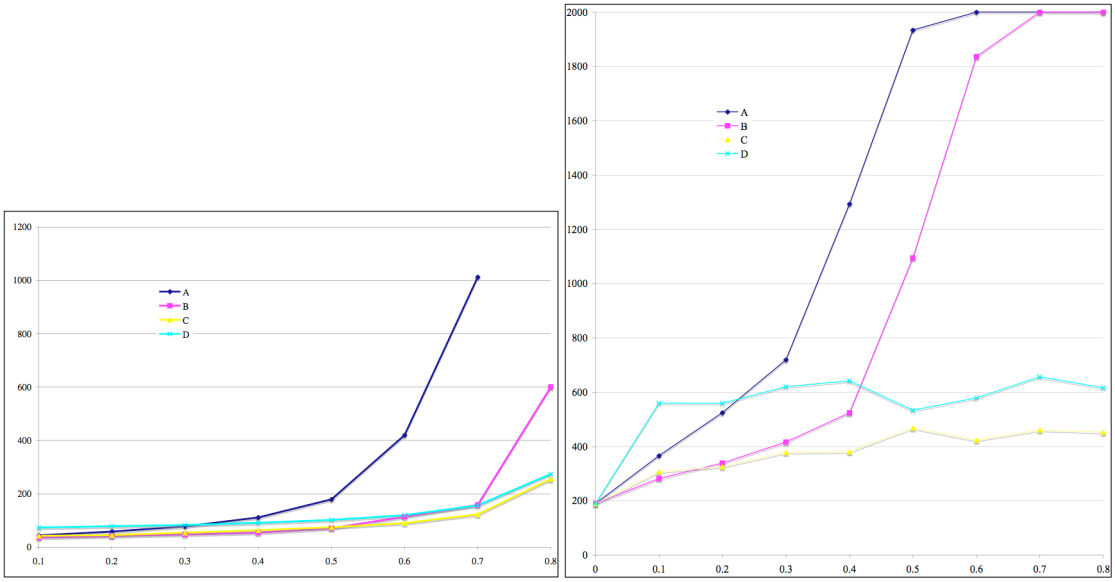

Figure 3: Number of passes through the training set required to store 20 patterns.
Left: training set bias = 0.5. Right: training set bias = 0.9.

8. Discussion

The results of the investigation reported here can be summarized as follows:

- A smallest-weight removal policy is much better than a random removal policy.
- The simplest principled dilution method (covariance dilution) is effective when the patterns to be stored are unbiased, but poor when the training set is highly biased.
- Perceptron dilution with zeroed weights is inferior to simple perceptron dilution: the zeroing of weights adds an extra step to the process, and may significantly increase re-training times, without improving the performance of the network.

Taken together these points suggest a way forward for those in search of an appropriate connection topology for a parsimonious associative memory.

A good method to find a near-optimal connection topology is to use the perceptron convergence procedure to train the network, then remove those connections that have weights of the smallest magnitude, then re-train the resulting network. At the loadings used here a large proportion of the weights can be removed before the performance of the network becomes significantly degraded, even where the data are highly biased. In fact networks trained with biased data may safely be diluted to a greater degree than those trained with unbiased data. Whilst this result is, on the face of it, counter-intuitive, it is very probably a direct result of the fact that a perceptron-trained associative memory has a higher maximum capacity when biased data sets are used [9].

It should also be noted that, whilst the covariance dilution strategy initially showed some promise with unbiased data, it could actually prove disastrous for real-world data sets (which tend to be biased). Not only is the pattern correction performance (as represented by R) poor even at modest dilution levels, but κ is negative for heavily diluted networks, indicating that many of the training patterns are not even stable states [9]: a fundamental requirement of any associative memory.

An interesting refinement of the method suggested by the results presented here would be to perform the initial training of the network using the perceptron convergence procedure with a lower training threshold (e.g. T=1, or even T=0) than we wish to use for the final training phase. The lower training threshold would facilitate faster convergence in the initial training phase, the purpose of which is to initialize connection weights so that the smallest may be selected for deletion from the network. We will be trying this in the near future.

9. References

[1] Diederich, S., et al., Learning by error corrections in spin glass models of neural networks, in *Computer simulation in brain science*, R.M.J. Cotterill, Editor. 1988, Cambridge University Press: Cambridge, UK. p. 232-239.

[2] Forrest, B.M., Content-addressability and learning in neural networks. *Journal of Physics A*, 1988. 21: p. 245-255.

[3] McGraw, P. and M. Menzinger, Topology and computational performance of attractor neural networks. *Physical Review E*, 2003. 68: p. 047102.

[4] Torres, J.J., et al., Influence of topology on the performance of a neural network. *Neurocomputing*, 2004. 58-60: p. 229-234.

[5] Davey, N., B. Christianson, and R. Adams. High Capacity Associative Memories and Small World Networks. in *Proceedings of the IJCNN*, 2004, Budapest.

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

[7] Garces, R., P. Kuhlmann, and H. Eissfeller, In search of an optimal dilution algorithm for feedforward networks. *Journal of Physics A: Mathematical and General*, 1992. 25(23): p. L1335.

[8] Kuhlmann, P., R. Garces, and H. Eissfeller, A dilution algorithm for neural networks. *Journal of Physics A: Mathematical and General*, 1992. 25(9): p. L593.

[9] Davey, N., S.P. Hunt, and R.G. Adams, High capacity recurrent associative memories. *Neurocomputing*, 2004. 62: p. 459-491.

[10] Davey, N. and R. Adams, High Capacity Associative Memories and Connection Constraints. *Connection Science*, 2004. 16(1): p. 47-66.

[11] Cover, T.M., Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 1965. EC-14: p. 326-334