Improving Associative Memory Capacity: One-Shot Learning in Multilayer Hopfield Networks

Arnold Liwanag (LIWANAG@HYPATIA.MCMASTER.CA)
Suzanna Becker (BECKER@HYPATIA.MCMASTER.CA)
Department of Psychology
McMaster University
1280 Main St. West
Hamilton Ont. Canada L8S 4K1

Abstract

Our brains have an extraordinarily large capacity to store and recognize complex patterns after only one or a very few exposures to each item. Existing computational learning algorithms fall short of accounting for these properties of human memory; they either require a great many learning iterations, or they can do one-shot learning but suffer from very poor capacity. In this paper, we explore one approach to improving the capacity of simple Hebbian pattern associators: adding hidden units. We propose a deterministic algorithm for choosing good target states for the hidden layer. In assessing performance of the model, we argue that it is critical to examine both increased *stability* and increased *basin size* of the attractor around each stored pattern. Our algorithm achieves both, thereby improving the network's capacity to recall noisy patterns. Further, the hidden layer helps to cushion the network from interference effects as the memory is overloaded. Another technique, almost as effective, is to "soft-clamp" the input layer during retrieval. Finally, we discuss other approaches to improving memory capacity, as well the relation between our model and extant models of the hippocampal system.

Introduction

Human episodic memory (Tulving, 1972) is a puzzling phenomenon to many modellers. Our brains have an extraordinarily large capacity to store and recognize patterns such as pictures (Standing, 1973), for example. This remarkable capacity must be reconciled with the fact that we can learn a complex item such as a picture or an association between a pair of unrelated words after only a single exposure (one-shot learning). How could we model such a system? Existing computational learning algorithms appear to be inadequate. Connectionist learning procedures can be grouped into two broad types: 1) those that have high capacity but require many learning iterations, and 2) those that have low capacity but can do one-shot learning. Models of the first type are able to extract statistical regularities or hidden variables gradually from the input; these include back-propagation networks, Boltzmann machines and competitive learning networks (for a review of these and other connectionist learning procedures, see Hinton, 1989). Models of the second type rapidly memorize the input without recoding it into hidden features; these include linear pattern associators (e.g., Anderson, 1972; Kohonen, 1972), Hopfield networks (Hopfield, 1982) and convolutional memory models (e.g., Murdock, 1982; Eich, 1982; Humphries et al., 1989).

Various attempts have been made to improve the capacity of single-layer associative memory networks (see the Discussion section). The approach taken here is to devise a principled way to train a memorizing device with *hidden units*. Our starting point is the Hopfield network. We retain the critical features of a Hopfield network: symmetric connections, symmetric Hebbian learning,[1] and Hopfield's activation dynamics for networks with real-valued units (Hopfield, 1984). With these features, the network will always settle into an *attractor state*.[2] Hopfield networks are popular among theoreticians because of their ease of analysis: they are now well understood with respect to their storage capacity and convergence properties. They are also appealing to many cognitive modellers because of their apparent similarity to human episodic memory: they can recall patterns after only a single exposure using a Hebbian learning rule, and they are capable of retrieval from partial or noisy cues (pattern completion). However, their capacity is extremely low. The number of patterns recalled nearly correctly as a proportion of the number of units is about 0.15. Further, as the memory is loaded beyond this point, performance deteriorates catastrophically. These are certainly not typical characteristics of human memory.

The Model

The solution explored here is to add hidden units to a Hopfield network, so that the network can encode hidden or latent variables and thereby improve its capacity. The problem is then how to adapt the weights to the hidden units. An obvious idea is to add a hidden layer with random initial weights, allow the hidden layer units to settle to "random states", and then simply apply one-shot Hebbian learning as in the usual Hopfield network. If the hidden layer states are truly random – or nearly so, this should improve the capacity because it is equivalent to increasing the input dimensionality, while making the input patterns more nearly orthogonal. However, we have found that this leads to even worse capacity. This is likely because 1) the hidden layer states are not really random, and 2) the Hebb-rule encourages them to encode features

[1] In symmetric Hebbian learning, the weight change on the connection from unit A to unit B, and also on that from unit B to A, is proportional to the product of the two units' activations

[2] An attractor state has the desirable property that it is stable; i.e., applying the state update rule to any unit leaves it's activation unchanged. Moreover, if the network is close to an attractor state, further state updates will tend to drive it into that attractor. This enables the network to perform error-correcting pattern retrieval.

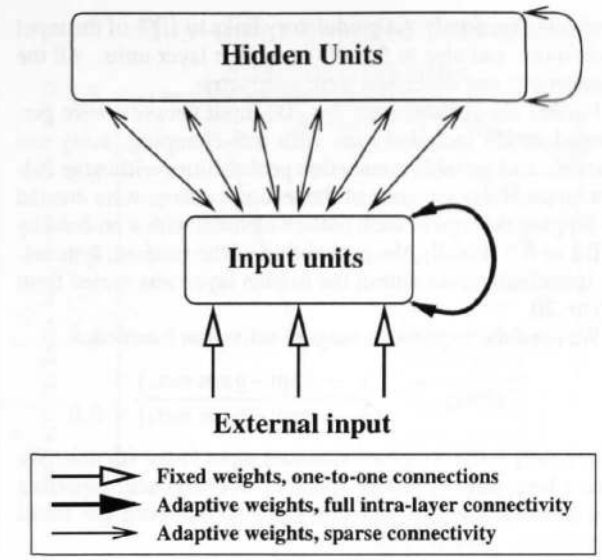

Figure 1: The network architecture. External input connections implement "soft-clamping" (see text); all other connections are symmetric. Input-to-hidden connections are involved in modulating the plasticity in the hidden layer.

common to many patterns, thereby leading to more incorrect retrievals (i.e., spurious attractors or "blend states").

The idea behind our approach is that the hidden layer should learn to correct for coding errors at the input layer. This should help to create stable attractors out of the input patterns. Further, the attractors should be *widened* so that the network is tolerant to noise in the input. We now describe our network architecture and training procedure. A key feature of our approach is a deterministic scheme for choosing the target states of the hidden layer by *minimizing input frustration* so that performance improves dramatically. Another critical feature of our model is *sparse hidden layer connectivity*. We first describe the network architecture (shown in Figure 1), and then the procedure for hidden state initialization and learning (summarized in Table 1).

Sparse input-to-hidden-layer connectivity: Each hidden unit should be responsible for the cleanup of only one or a few input units' activities. The hidden layer is therefore randomly and sparsely connected to the input layer, and these connections initially have small uniform, positive weights.

Sparse hidden-to-hidden-layer connectivity: If each hidden unit were fully connected to every other hidden unit, the learned associations within the hidden layer would simply mirror those of the input layer. Very little improvement in capacity would be expected in this case. On the other hand, if each hidden unit is connected to a small number of other randomly chosen hidden units, it should learn to predict the input from what is approximately a random feature of the input.

Hidden unit state initialization: Each time an input pattern is presented to the network, the hidden unit states are first initialized to zero, and then synchronously updated for a single time step. Thus, their initial states depend only upon the input layer activations, and not on other hidden unit activations. (Subsequent to learning, when the memory performance of the network is evaluated, the entire network's states would be updated at this point until they stabilize.) Next, their states are thresholded at zero to produce binary states of 1 or -1. These states are then refined as described below to produce the final *hidden layer target states*.

Frustration minimization: Hopfield's state update equations allow the network to settle to a stable state (an attractor), by minimizing the following energy function:

$$E = -0.5 \sum_i \sum_j state_i \; state_j \; weight_{ij}$$

If an input pattern is presented to the network but that state is unstable, the network will move away from that state and settle into the wrong attractor. We therefore want the network to build attractor states out of the input patterns. Thus, our goal for the hidden layer is to form representations that compensate for instabilities in the input layer, and correct for potential coding errors. This motivates the following scheme for adjusting the initial hidden layer states to produce the final hidden layer target states: We minimize the input layer *frustration*. An input unit is in a "frustrated" state when its net input is of opposite sign to its external input. In this case, the input unit sends a modulatory signal to its hidden units, alerting each hidden unit that it is frustrated. Each hidden unit can then evaluate whether it is contributing to the input layer's frustration, and reverse its own state if appropriate. Our algorithm for selecting hidden layer target states follows the gradient of the energy for frustrated input units. This energy measure corresponds to the energy equation defined above, with i indexing over frustrated input units and j indexing over all units.

Weight updates: Once the states of the hidden units have been determined, the one-shot, symmetric Hebbian learning rule is then applied. This minimizes the energy in the entire network to make an attractor out of the current global network state (including both input and hidden layer units).

Table 1. The Learning Phase

1. Initialize modulatory connection weights to 1.
2. Initialize all other connection weights to 0.
3. For each of the N input patterns,
  3.1 Clamp input layer states to external inputs.
  3.2 Set hidden layer states to zero.
  3.3 Do 1 synchronous update of hidden layer states.
  3.4 Threshold hidden layer states to get -1,1 states.
  3.5 Improve hidden states to get final target states:
    3.5.1 For each input unit i,
    Compute total input, net_i, from all layers.
    If frustrated (i.e. $state_i * net_i < 0$),
      For each hidden unit j that input i projects to,
        Send a modulatory signal, m_{ij}:
        $m_{ij} = state_i * weight_{ij}$
    3.5.2 For each hidden unit j,
      If hidden unit is frustrating the input
      (i.e. $state_j * \sum_i m_{ij} < 0$),
        Reverse this hidden unit's state.
  3.6 Apply one Hebbian learning step to all weights:
    $\Delta weight_{ij} = 1/N \; state_i \; state_j$

Measuring the trained network's performance

Soft-clamping the input layer: Once the network's weights have been trained, it can then be tested on its ability to recall or recognize *test patterns*, presented as external input to the network. Geoff Hinton (personal communication) suggested to us that one reason the Hopfield network recalls so few patterns correctly may be that the input is only made available for a single time step. As the input units' states are updated, they are free to forget completely their initial states. Thus, the final states of the units after settling may be very far from the external input pattern. A quick fix for this problem is to provide the external input to each unit as a constant input source while the input unit states are updating. This scheme been used elsewhere (e.g. the BSB model, Anderson, 1995, Chapter 15) although, to our knowledge, its effect on capacity has not been reported previously. It is sometimes referred to as *soft-clamping*. To implement this idea, we provided each input unit with an extra incoming link with a positive connection weight (see Figure 1).

Measures of Capacity: Two capacity measures are commonly used in the literature: 1) The *absolute capacity* is the proportion of patterns which can be recalled exactly, and 2) the *relative capacity* is the proportion of the training patterns which can be recalled nearly correctly. We used the latter, with a 98% correctness criterion. We tested the network on both noise-free patterns (identical to the training patterns) and noisy versions of the training patterns. The former case is analagous to a recognition memory test, while the latter is analagous to cued recall. Table 2 summarizes the procedure for testing the network. In the next section, we summarize the simulations we have performed with this model.

Table 2. The Test Phase

For each input pattern,

1. *Initialize input layer states and external (soft-clamped) inputs according to the input pattern.*
2. *Initialize hidden layer states as in the learning phase (steps 3.2 and 3.3 in Table 1).*
3. *Allow the entire network to settle to equilibrium.*
4. *Retrieval is counted as correct if at least 98% of the input units are in correct states.*

Simulations

Procedure

All simulations were run on run on Silicon Graphics Indy workstations. The training patterns were random vectors of binary numbers that had an equal probability of being 1 or -1. Five different random training sets were created for each input layer size and training set size. Hence, simulations with a given architecture consisted of running five independent trials with each of the input training set sizes.

Our initial simulations were run using networks with 50, 100, and 150 input layer units, without soft-clamping. We varied the number of hidden units from 0 to 500 at intervals of 100 units. For each of these conditions, the connectivity scheme was as follows. The input layer was fully interconnected (without self-connections). The hidden units were connected randomly via modulatory links to 10% of the input layer units, and also to 5% of the hidden layer units. All the connections just described were symmetric.

Further simulations with the 100-input network were performed which included runs with soft-clamping, noisy test patterns, and variable connection probabilities within the hidden layer. Noisy versions of the testing patterns were created by flipping the sign of each pattern element with a probability of 0.1 or 0.2. Finally, the probability of the random, symmetric interconnections within the hidden layer was varied from .10 to .20.

We used the hyperbolic tangent activation function:

$$state_i = \frac{1 - \exp(-gain\ net_i)}{1 + \exp(-gain\ net_i)}$$

where net_i is the weighted summed input to the ith unit. We used a large gain of 50.0 to speed up convergence. Lowering the value of the gain did not seem to influence the recall capabilities much if at all.

Results and Discussion

Results with noise-free patterns

Some of the results for the 100-input network with noise-free data are presented in Figure 2. The bottom curve shows the performance of networks with no hidden units and no soft-clamping. The next-lowest curve shows the benefits of soft-clamping with no hidden units. The remaining curves show results for networks with soft-clamping and varying numbers of hidden units. Although this figure suggests that the hidden layer is beneficial, it also illustrates that testing with noise-free patterns can be rather deceptive. In fact, we could get perfect recall for any pattern set size (i.e., infinite capacity) simply by setting the soft-clamping weights to be sufficiently large. In doing so, input units would then ignore the states of other units in the network, and simply copy the states of their external input lines. However, these networks would exhibit no tolerance to noise. In other words, the soft-clamping helps to make stable attractors of the input patterns, but it does not help to make wide attractor basins. In fact, it may lead to more spurious attractors because it helps to make stable attractors of every possible state. For the remaining simulations reported here, networks were tested with noisy patterns.

Results with noisy patterns

Figure 3 shows the results of 100-input networks tested with noisy input patterns.

Effects of training set size: As mentioned earlier, the simple Hopfield network with no hidden units and no soft-clamping has a relative capacity of about .15N, where N is the number of units. If such a network is overloaded with a number of patterns exceeding its capacity, its performance rapidly deteriorates toward zero. Moreover, the capacity is a great deal worse when measured with noisy test patterns. These effects can be seen in the bottom curves of Figures 2 and 3 respectively. With soft-clamping and hidden units, the network performs much better in the presence of noise. With increasing training set sizes, the network performance

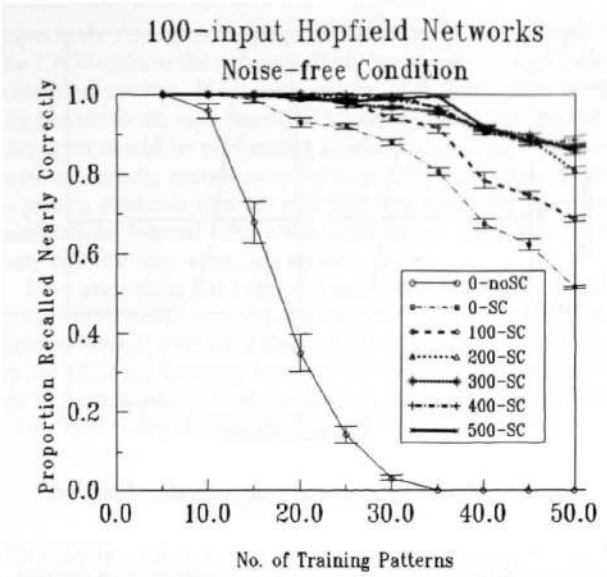

Figure 2: Plot of capacity versus training set size, averaged across five different runs using different random training sets (with standard error bars), for networks with 100 input units tested on noise-free patterns. Each curve shows the results for a network with a different number of hidden units (see legend). All networks had soft-clamping except the bottom one, labelled "0-noSC".

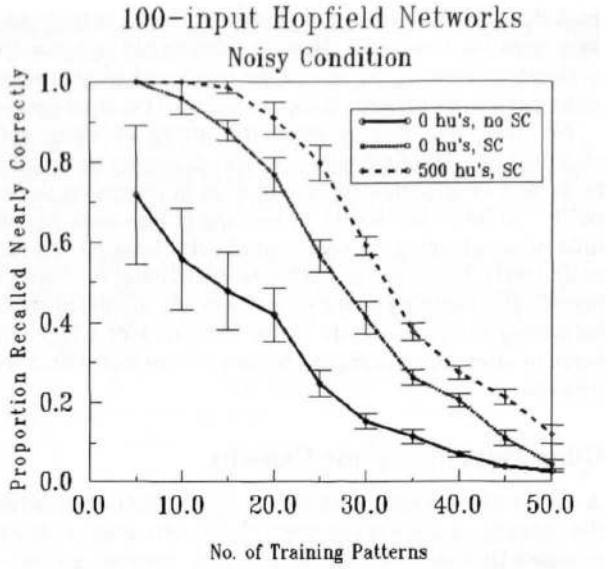

Figure 3: Plot of capacity versus training set size, averaged across five runs (with standard error bars), for networks with 100 input units tested on noisy patterns. Results for three different networks are plotted: no soft-clamping and no hidden units (bottom curve), soft-clamping and no hidden units (midde curve), and soft-clamping and 500 hidden units (top curve).

still deteriorates toward zero. Using an incremental learning rule with weight decay may help to prevent this deterioration, by allowing gradual replacement of old memories with new ones. This should allow the capacity to remain relatively constant (in absolute numbers of patterns recalled, as opposed to proportion of training set size) as the training set size grows.

Although our model does not yield the sort of capacity increases one would see with a more powerful learning procedure such as back-propagation, in the league of Hopfield-style networks with one-shot learning, it performs quite impressively.

Effects of hidden layer size and connectivity: Varying the size of the hidden layer from 0 to 500 units produced smoothly increasing gains in capacity for all input sizes and training set sizes tested. Only the performance curve for the 500-hidden-unit network is shown in Figure 3. With respect to the connectivity of the hidden layer, more significant gains were achieved when the probability of the interconnections among the hidden units was increased from 0.1 to 0.2 (not shown in Figure). This has the effect of widening the basins of attraction for the input patterns, since more units are involved in the representation. However, there is a limited benefit in continuing to increase the connectivity in the hidden layer. In fact, if the hidden layer is too strongly interconnected, performance deteriorates – presumably because the hidden layer features become correlated and may then lead to spurious attractors.

Effect of soft-clamping versus hidden units: By adding hidden units and soft-clamping, we see more than a five-fold improvement in capacity over the simple Hopfield network without hidden units or soft-clamping. This estimate is derived from Figure 3 by equating performance at the .7 recall level as a proportion of the training set size. This is the maximum achieved by the simple Hopfield network. At this point, the original Hopfield network can store about 3.5 patterns in a training set of 5 patterns, while the network with 500 hidden units and soft-clamping can store about 19 of 27 patterns. Note that the network with 500 hidden units and 0.1 connection probability has only a four-fold increase in the number of connections over a simple Hopfield network with 100 units. To achieve this same improvement in a simple fully connected Hopfield network would require increasing the network size four- to five-fold, hence, adding 16 to 25 times the number of connections. Additionally, one would need to increase the number of bits of information in the input pattern, whereas in our networks with hidden units we have not added any new information to the input.

Hopfield networks with soft-clamping and hidden units outperformed those with soft-clamping and no hidden units by factors of 2.0, 1.6 and 1.4 when measured at the 1.0, 0.9 and 0.8 recall levels respectively (see Figure 3). Thus, the soft-clamping accounts for most of the performance improvement, but the hidden layer enhances peroformance significantly beyond this.

General Discussion

We began by stating that one way to improve the capacity of an associative memory model would be to create random codes in the hidden layer. Our model uses random sparse connectivity, rather than random initial weights, to achieve a degree of randomness in the hidden layer features. More importantly, however, our model uses a principled, *deterministic* scheme for improving upon the hidden codes, by minimizing the amount of frustration in the input layer. One might

think that simply assigning random initial states to the hidden units would be better still. However, this would be equivalent to simply increasing the size of the input, and would require adding *new external information* to augment the input pattern.

We have also demonstrated the utility of using soft-clamping in a Hopfield network: soft-clamping turns out to be at least as important as hidden units in improving the capacity. At low noise levels, in fact, the hidden units provide little or no advantage beyond that of soft-clamping. At high noise levels, however, the hidden layer becomes increasingly beneficial in cleaning up the input states. As discussed above, increasing the connectivity in the hidden layer widens the basin of attraction, leading to better performance with noisy patterns.

Other Ways to Improve Capacity

A variety of other schemes have been proposed for improving the capacity in associative memory models, while retaining one-shot Hebbian learning. We describe two here: the use of sparse activations, and non-monotonic activation functions.

Sparse activations

Several investigators have demonstrated the utility of sparse activations in improving memory capacity (Amit et al., 1987; Tsodyks, 1988; Tsodyks & Feigel'man, 1988). That is, rather than using equal bit probabilities in generating the input patterns, only a small number of the input units should be on. An intuitive way to understand the reason for this is to think of the probability of any pair of units being on together. With random patterns, the pairwise probabilities are just equal to the product of the individual bit probabilities. Thus, with sparse patterns, any pair of units has a much lower chance of being on together. In this case, the connections encode associations which occur relatively infrequently; thus, each association may only represent a single memory episode. Large sparsely connected networks therefore seem to be well-suited as models of episodic memory. This analysis applies equally well to the hidden layer codes in our network. Thus, we would expect a comparable improvement in capacity for our multi-layer network if we used sparse codes at both the input and hidden layers.

Alternative activation functions

Morita has studied a version of the Hopfield network which uses a non-monotonic activation function (Morita, 1993; Yoshizawa et al., 1993) of the form shown in Figure 4. Morita's network achieves approximately a three-fold improvement in capacity over Hopfield's dynamics. Interestingly, when the Morita network is presented with a noisy pattern that it fails to recognize, rather than settling to an incorrect attractor, Morita claims that it instead wanders chaotically through its state space. Recent estimates of the fractal dimension of Morita network dynamics by Thomas Trappenberg (personal communication) confirm that they do indeed appear to be chaotic.

Why would chaotic behavior be desirable in a memory model? Steve Joordens (personal communication) has suggested that one could build a recognition memory network

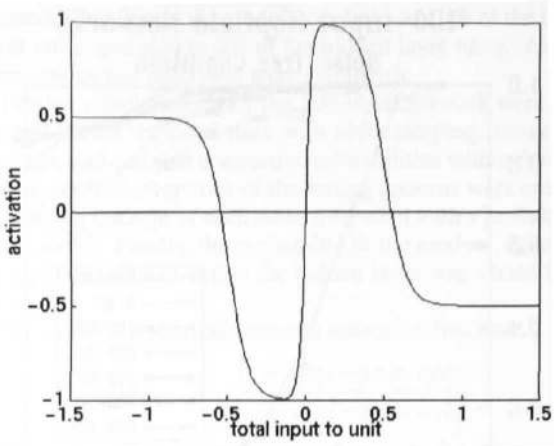

Figure 4: Morita's non-monotonic activation function.

based on Morita's dynamics that can discriminate a familiar stimulus, which brings the network to an attractor state, from a novel stimulus, which does not. This suggests two intriguing possibilities for memory models: 1) The distinction between attractor states and chaotic dynamics could be used to signal the memory system as to when it should encode something new, rather than encoding every item that is presented to the network. 2) If the chaotic path through state space remains sufficiently far from any stored patterns, it may be used to select pseudo-random target states for the hidden layer when a new code is to be learned. In other words, the Morita dynamics could be used to advantage in our model to help select good hidden layer codes for novel patterns. We are currently investigating the utility of combining both sparse codes and Morita dynamics with the learning procedure described here.

Relation to Hippocampal Function

It is widely believed by both neuroscientists and memory researchers that the seat of the episodic or explicit memory system is the hippocampal area (including surrounding cortical structures). Early attempts to model hippocampal function assumed that this brain region behaved like a simple Hebbian pattern associator (Marr, 1971; McNaughton & Nadel, 1990). Two more recent models have emphasized the need for a random recoding of the hippocampal inputs from the dentate gyrus into a sparse code in the CA3 region of the hippocampus (McClelland et al., 1995; Levy & Wu, 1993; Levy & Wu, 1996; Levy, 1997). The model proposed here is consistent with that interpretation, i.e., the dentate gyrus acts as the input layer and CA3 acts as the hidden layer. The mossy fiber projections from the dentate to CA3 could play the role of sending plasticity-gating modulatory signals. This could be accomplished via a frequency code (e.g., the hippocampal theta rhythm), or by modulating the gain of the hidden units' activation functions in combination with a plasticity threshold.

Our model would need to be elaborated to account for the detailed circuitry of the hippocampal structures. For example, our model has symmetric feedback connections from the hidden to the input layer, whereas there do not appear to be direct back-projections from CA3 to the dentate gyrus. Instead, the

input to the dentate is re-integrated with the code produced in the CA3 region at the subsequent processing stages (CA1 and entorhinal cortex). If we removed the back-projections from the hidden to the input layer, and used sparse codes, the hidden layer would be performing a rather interesting function: it would directly encode only the *surprising* parts of the input – pattern elements that are not expected given the pairwise associations learned within the input layer. This may be a very efficient way to encode novel episodes.

Levy and others have stressed the role of the hippocampal system in forming *temporal associations* in sequence learning. Several investigators have shown that adding a temporal delay in the Hebbian learning rule can accomplish this in simple associative memory models (e.g. Amari, 1972; Griniasty et al., 1993; Levy & Wu, 1996; Levy, 1997).

Conclusions

We have presented a novel algorithm for training Hopfield networks with hidden units which both deepens and widens the attractor basins around the training patterns. The model is of interest as a potential account of human recognition memory and hippocampal function.

Acknowledgements

We thank Steve Joordens, Thomas Trappenberg, Geoff Hinton and the members of the Connectionist Research Group at the University of Toronto for valuable feedback on this work. All simulations were performed using software developed with the Xerion Neural Network Simulator from Hinton's lab at the University of Toronto. This work was supported by research grants to the second author from the Natural Sciences and Engineering Research Council of Canada and the James S. McDonnell Foundation's McDonnell-Pew Program in Cognitive Neuroscience.

References

Amari, S. I. (1972). Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers, C-21*:1197–206.

Amit, D., Gutfreund, H., & Sompolinsky, H. (1987). Information storage in neural networks with low levels of activity. *Physical Review A, 35*:2293–2303.

Anderson, J. A. (1972). A simple neural network generating an interactive memory. *Mathematical Biosciences, 14*:197–220.

Anderson, J. A. (1995). *An introduction to neural networks*. Cambridge, MA: The MIT Press.

Eich, J. M. (1982). A composite holographic associative recall model. *Psychological Review, 89*:627–661.

Griniasty, M., Tsodyks, M. V., & Amit, D. J. (1993). Conversion of temporal correlations between stimuli to spatial correlations between attractors. *Neural Computation, 5*:1–17.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence*, 40:185–234.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences U.S.A., 79*:2554–2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences U.S.A., 81*:3088–3092.

Humphries, M., Bain, J., & Pike, R. (1989). Different ways to cue a coherent memory system: A theory for episodic, semantic, and procedural tasks. *Psychological Review, 96*:208–233.

Kohonen, T. (1972). Correlation matrix memories. *IEEE Transactions on computers, C-21*:353–359.

Levy, W. B. (1997). A sequence predicting CA3 is a flexible associator that learns and uses context to solve hippocampal-like tasks. to appear in *Hippocampus*.

Levy, W. B. & Wu, X. (1993). Predicting complex behavior in sparse asymmetric networks. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in Neural Information Processing Systems 5* (pp. 556–563). Morgan Kaufmann.

Levy, W. B. & Wu, X. (1996). The relationship of local context codes to sequence length memory capacity. *Network: Computation in Neural Systems, 7*:371–384.

Marr, A. (1971). Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society of London, 262 (Series B)*:23–81.

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review, 102*(3):419–457.

McNaughton, B. & Nadel, L. (1990). Hebb-Marr networks and the neurobiological representation of action in space. In *Neuroscience and connectionist theory*. Lawrence Erlbaum.

Morita, M. (1993). Associtive memory with nonmonotone dynamics. *Neural Networks, 6*:115–126.

Murdock, B. B. (1982). A theory for the storage and retrieval of item and associative information. *Psychological Review, 89*(6):316–338.

Standing, L. (1973). Learning 10,000 pictures. *Quarterly Journal of Experimental Psychology, 25*:207–222.

Tsodyks, M. V. (1988). Associative memory in asymmetric diluted network with low level of activity. *Europhysics Letters, 7*(3):203–208.

Tsodyks, M. V. & Feigel'man, M. V. (1988). The enhanced storage capacity in neural networks with low activity level. *Europhysics Letters, 6*:101–105.

Tulving, E. (1972). Episodic and semantic memory. In E. Tulving & W. Donaldson (Eds.), *Organization of memory* (pp. 381–403). New York: Academic Press.

Yoshizawa, S., Morita, M., & Amari, S. (1993). Capacity of associtive memory using a nonmonotonic neuron model. *Neural Networks, 6*:167–176.