Using pseudo-recurrent connectionist networks to solve the problem of sequential learning

(One page abstract in: *Proceedings of the 19th Annual Cognitive Science Society Conference*, (1997) NJ:LEA. 921)

Robert M. French
Psychology Department, B32
University of Liège
4000 Liège, Belgium
rfrench@ulg.ac.be

Abstract

A major problem facing connectionist models of memory is how to make them simultaneously sensitive to, but not disrupted by, new input. This paper describes one solution to this problem. The resulting connectionist architecture is capable of sequential learning and exhibits gradual forgetting where standard connectionist architectures may forget catastrophically. The proposed architecture relies on separating previously learned representations from those that are currently being learned. Crucially, a method is described in which *approximations* of the previously learned data, called *pseudopatterns*, will be extracted from the network and mixed in with the new patterns to be learned, thereby alleviating sudden forgetting caused by new learning and allowing the network to forget gracefully.

Introduction

One of the most important problems facing connectionist models of memory — in fact, facing *any* model of memory — is how to make them simultaneously sensitive to, but not disrupted by, new input. This problem is often referred to as the "sensitivity-stability" dilemma (D. O. Hebb, 1949) or the "stability-plasticity" problem (Carpenter & Grossberg, 1987). It is particularly relevant for connectionist networks, especially since they can be afflicted by a particularly severe manifestation of the sensitivity-stability problem known as catastrophic interference. Catastrophic interference is the tendency of neural networks to abruptly and completely forget previously learned information in the presence of new input (McCloskey & Cohen, 1989; Ratcliff, 1990).

Because of the sensitivity-stability problem, *all* of the patterns given to a connectionist network must be learned "concurrently". In other words, the entire set of patterns to be learned must be presented over and over, each time adjusting the weights of the network by small increments until the network gradually finds a weight-space solution for the entire set of patterns. This a far cry from how humans learn a series of patterns, however. Much of human learning tends to be *sequential*. A particular pattern is learned, then another, and another, and so on. While some of the earlier patterns may be seen again, this is not necessary for them to be retained in memory. As new patterns are learned, forgetting of old, unrepeated patterns occurs gradually as a function of time.

In this paper, a connectionist architecture will be presented that is capable of effective sequential learning, exhibiting gradual forgetting where a standard backpropagation architecture forgets catastrophically. The proposed architecture relies on separating the previously learned representations from those that are currently being learned. Crucially, a method is described in which an *approximation* of the previously learned data (not the original patterns themselves, which the network will not see again) will be extracted from the network and will be mixed in with the new patterns to be learned.

The need to separate old and new representations

Most approaches to reducing catastrophic interference in traditional connectionist architectures have relied on reducing the overlap of representations either by orthogonal recoding of the input patterns (Kortge, 1990; Lewandowsky and Goebel, 1991; Lewandowsky and Shu-Chen, 1993), or, alternately, by orthogonalizing the network's hidden layer representations (French, 1992, 1994; Murre, 1992; Krushke, 1993; McRae & Hetherington (1993)). A thorough discussion of these techniques and an analysis of the underlying causes of catastrophic interference can be found in Sharkey & Sharkey (1996). Pushing the logic of reducing representational overlap to its ultimate conclusion, McClelland, McNaughton, and O'Reilly (1995) have argued that radical separation of representations might have been the approach arrived at by evolution in the development of the hippocampus and the neocortex. They justify the brain's bi-modal architecture as follows:

> "The sequential acquisition of new data is incompatible with the gradual discovery of structure and can lead to *catastrophic interference* with what has previously been learned. In light of these observations, we suggest that the neocortex may be optimized for the gradual discovery of the shared structure of events and experiences, and that the hippocampal system is there to provide a mechanism for rapid acquisition of new information without

interference with previously discovered regularities. After this initial acquisition, the hippocampal system serves as a teacher to the neocortex..."

But this stills leaves unanswered a crucial question—namely: *How* does the neocortex store new information, whether it comes from the hippocampus or elsewhere, without disrupting information already stored there? McClelland, McNaughton, and O'Reilly make a distinction between *focused* and *interleaved* learning. In the former case, new information is simply presented to the system, perhaps a number of times, but without interleaving it with old, previously acquired knowledge. In the latter type of learning, the new knowledge is interleaved with the rest of the database of already acquired knowledge. They show that significant disruption of previously acquired information (very much like catastrophic interference) occurs in focused learning.

Their solution involves the very gradual incorporation of the new information into the neocortical structure (i.e., long-term memory). Hippocampal representations very gradually train the neocortex. The problem is that no matter how slowly the hippocampal information is passed to the neocortex, radical forgetting of the old information may result, *unless a way is found to interleave the already stored neocortical patterns* with the new patterns being learned. This interleaving cannot always use "the rest of the [original] database" of previously learned patterns because many of these patterns will no longer be explicitly available for re-presentation to the network. Once we have learned about penguins and wish to learn about, say, mice or tractors, we do not have to continue to be shown penguins so as not to forget them. Unfortunately, this *is* necessary for standard backpropagation networks to prevent forgetting of previously learned patterns. When learning new information, the previously learned information must once again be explicitly presented to the network, otherwise it may be completely forgotten.

The architecture proposed in this paper will provide a way to automatically refresh the network without recourse to the original patterns. Instead of the original patterns, internally-produced *approximation*s of these patterns, called pseudopatterns (Robins, 1995), will be used and interleaved with the new patterns to be learned. The architecture proposed will argue for two functionally distinct areas of long-term memory: one, an "early-processing area" (or "buffer-storage area") in which new information will be initially processed and a second, a "final-storage area," in which information will be consolidated. This model of long-term memory will suggest a natural means of consolidation of information in long-term memory that supports the neurobiologically motivated conclusions of Robins (1996).

Training in standard error-driven connectionist networks

The ideal way, of course, to solve the stability-sensitivity problem would be to store all previously learned patterns out of harm's way until new input was presented to the system. At that point, all of the previously learned patterns would be taken out of storage, so to speak, and would be mixed with the new patterns. The system would then learn the mixed set of old and new patterns. After the augmented set of patterns had been learned by the network, it would be put in storage, awaiting the next time new information was presented to the network. There would be no forgetting, catastrophic or otherwise, in this ideal world and new input would have no deleterious effect on the network's ability to generalize, categorize or discriminate.

Unfortunately, this way of learning new data is rarely possible in the real world except in the most artificial situations. It is in essence impossible to explicitly store all, or even a reasonable fraction of previous training exemplars for future learning. I will suggest that *internal approximations* of the original patterns are generated in long-term memory and it is these approximations that, in the absence of the real pattern-in-the-environment, serve to continually reinforce the long-term memory traces of the original patterns. The use of pseudopatterns to improve performance of connectionist networks on catastrophic interference was first proposed by Robins (1995) and their plausibility has been further explored in Frean & Robins (1996) and Robins (1996).

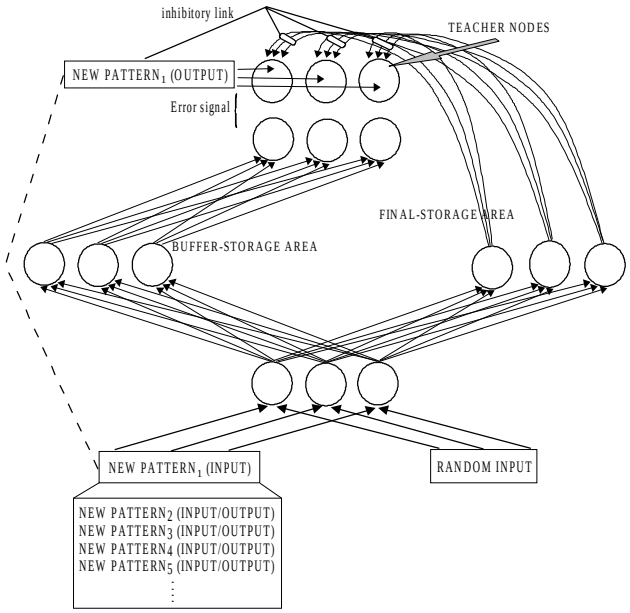

Figure 1. The pseudo-recurrent network architecture

The "pseudo-recurrent" architecture

The architecture discussed in this paper consists of a feedforward backpropagation network that is divided into two parts, one used to help train the other (Figure 1). We will call the left-hand side of the network the "buffer memory" and the right-hand side the "final-storage memory." The system works as follows. When a new pattern from the environment, consisting of an input and an associated output (the "teacher"), is presented to the system, this will cause activation to spread through both sides of the network. The fact that the new pattern is "real" (i.e., from the environment) means that its "teacher" will inhibit the output coming from final storage. The new pattern will be learned in the "buffer memory" by the standard backpropagation algorithm.

At the same time, a number of "pseudopatterns" will be generated by the final storage memory. In other words, random input will be fed into the network. Activation will spread through both the buffer and final-storage areas of the network. Crucially, *the output of this random input sent through the final-storage part of the network will be used as a teacher for the buffer part of the network*. In other words, the buffer network will then learn a series of pseudopatterns produced in the final-storage area that reflect the patterns previously stored in final-storage. So, rather than interleaving the real, originally learned patterns with the new input coming to the buffer memory, we do the next best thing — namely, we interleave pseudopatterns that are *approximations* of the previously stored patterns. Once the new pattern and the pseudopatterns are learned in the buffer area, the weights from the buffer network are copied to the corresponding weights in the final-storage network.

I have called the architecture "pseudo-recurrent" not only because of the recurrent nature of the training of the buffer memory by the final-storage memory, but also as a means of acknowledging the all-important mechanism of information transfer from final-storage to buffer storage — namely, pseudopatterns.

A specific example may help to clarify the learning mechanism. Suppose that there are 20 patterns, P_1, P_2, ... P_{20}. Each of these patterns, P_i consists of an input and an output ("teacher") association (I_i, T_i). The system will be required to *sequentially* learn all 20 patterns, In other words, each individual pattern must be learned to criterion before the system can begin to learn the subsequent pattern. To learn pattern P_1, its input I_1 is presented to the network. Activation flows through both parts of the network but the output from the final-storage part is prevented from reaching the teacher nodes by the "real" teacher T_1. The buffer network then adjusts its weights with the standard backpropagation algorithm using as the error signal the difference between T_1 and the output of the buffer network O_i. Internally created pseudopatterns from the final-storage memory are now generated and will be learned by the buffer memory. This is done by presenting random input to the network, which causes activation to spread through both the buffer and the final-storage memories. For "pseudo"-input, unlike for "real" input, there is no "real" teacher to inhibit the arrival of final-storage activation to the teacher nodes (i.e., the third layer of the final-storage network). The activation on the teacher nodes is thus produced by the spreading of activation by the pseudo-input through the final-storage memory. These teacher nodes then serve to train the buffer memory.

It is instructive to examine in detail the operation of one of the pseudopatterns. A random input pattern, i_1, is presented to the input nodes of the system. This input produces an output, o_1, at the output layer of the buffer memory and also produces an output, t_1, on the teacher nodes of the final-storage memory. This input-output pair (i_1, t_1) defines a pseudopattern, ψ_1, that reflects the contents of the final-storage memory. The difference between t_1 and o_1 determines the error signal for changing the weights in the buffer memory. The other random inputs, i_2, i_3, . . . i_n, are successively presented to the system and the resulting pseudopatterns, ψ_2, ψ_3, . . . ψ_n will also be learned by the buffer memory. Once the weight changes have been made for the first epoch consisting of {P_1, ψ_1, ψ_2, . . . ψ_n}, the buffer memory cycles through this set of patterns again and again until it has learned them all to criterion. By learning the pattern P_1 the buffer memory is learning the new information presented to it; by learning the pseudopatterns $\psi_{1, \ldots,} \psi_n$, the buffer memory is learning an approximation of the information previously stored in final storage. Obviously, the more pseudopatterns that are generated, the more accurately they will reflect the contents of final storage. (However, with too many pseudopatterns the buffer memory may fail to converge. The type and number of pseudopatterns to use is an area of on-going investigation.) Once learning in the buffer network has converged for P_1, ψ_1, ψ_2, . . . ψ_n, the buffer weights then replace the final-storage weights. In other words, the buffer memory *becomes* the final storage memory and the network is ready to learn the next pattern, P_2.

Although a fairly obvious point, it is worth noting that as the number of pseudopatterns per new item increases, there is a corresponding decrease in the amount of previously learned information lost. This will be clearly seen in the experiments reported below.

Databases used for the experiments

A number of experiments were run to test this architecture. Two "real-world" database from the University of California at Irvine repository of machine learning databases (Murphy & Aha, 1992) were used to test this system. The first of these UCI databases was the "mushroom" database and the second database, also used

to study catastrophic interference in French (1992, 1994), was the 1984 U.S. Congressional Voting Records database.

Tests of the pseudo-recurrent network

In the experiments described below will show that if a network can continually train itself (pseudo-recurrence) on an approximation of previously acquired information, that the problem of severe interference is drastically reduced and the system will become stable in the presence of new information. We then show how this type of system is able to do true sequential learning, gradually, rather than suddenly, forgetting old information. Finally, we mention how the system can be extended so that the requirement of an equal number of hidden units for the buffer and final-storage memories can be removed. This extension involves using pseudopatterns to transfer information between both parts of the memory.

Experiments

Experiment 1: Performance of a pseudo-recurrent network on the UCI mushroom database

For this experiment, we will consider data from the UCI mushroom database (Murphy & Aha, 1992). From this database we selected two "quasi-contradictory" sets of mushrooms. Quasi-contradictory pairs of mushrooms were are those that differ on no more than 3 (of 22) attributes with one member of each pair being poisonous, the other edible. There were approximately 70 such pairs of closely paired edible/poisonous mushrooms in the UCI database. We picked 20 mushrooms for the original training set and 20 "quasi-contradictory" partners for the second set to be learned. The original set contained 17 edible mushrooms and 3 poisonous mushrooms. The second set of new mushrooms that closely resembled those in the first set (the "contradictory data") contained 17 poisonous and 3 edible mushrooms.

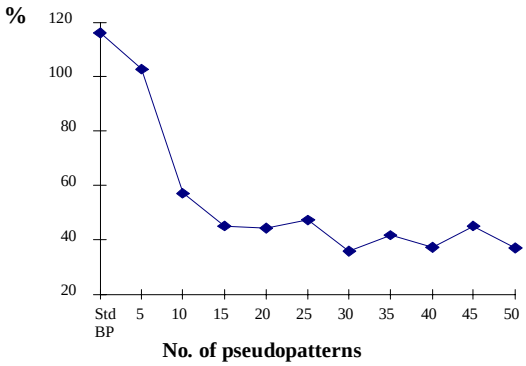

Figure 2. Percentage indicating the number of epochs required to relearn the original 20 patterns compared to the number of epochs initially required to learn these patterns after the pseudo-recurrent network had learned the second set of 20 "contradictory" patterns.

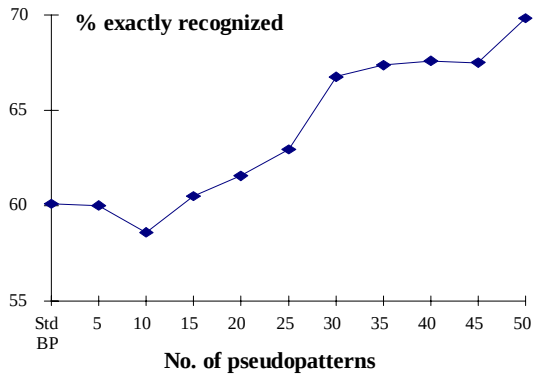

Figure 3. The increase in the proportion of items in the originally learned set exactly recognized by the network after learning the set of contradictory items.

For this experiment, each mushroom attribute was coded in a binary manner and added to the input vector describing the mushroom. The system performed a yes/no (poisonous/edible) classification task. The architecture of the pseudo-recurrent network had 57 input nodes, 10 buffer hidden units, 10 final-storage hidden units, one buffer output unit and one teacher unit (see Figure 1 for the design of the overall architecture). The learning rate was set at 0.2, momentum at 0.9, and the convergence criterion at 0.2.

The large number (57) of input nodes means that the information transfer using pseudopatterns will necessarily be approximate, since the probability of actually reproducing any of the actual patterns previously stored in final storage will be, for all intents and purposes, zero.

Figures 2 and 3 show significant improvements on depth-of-forgetting (as measured by the time required to relearned the original data) and exact recognition of items previously learned. In other words, the pseudo-recurrent network succeeds in maintaining the memory trace of the previously learned patterns while learning the new patterns. This should mean that true sequential learning of patterns, characterized by gradual forgetting of old information, should be possible with this type of network. Experiment 2 is designed to demonstrate the feasibility of sequential learning with a pseudo-recurrent network

Experiment 2: Sequential learning

Arguably the most significant contribution of the pseudo-recurrent network is its ability to learn data in a serial manner without experiencing severe interference (Hetherington, 1991). In this experiment, the patterns were taken randomly from the 1984 Congressional Voting Records database at the UCI repository. Twenty members of Congress (10 Republicans, 10 Democrats, defined by

their voting record on 16 issues) were chosen randomly from the database and were presented *sequentially* to the network. In other words, a new pattern was presented only after the previous one had been learned to criterion. The order of presentation was randomized over 50 runs of the program.

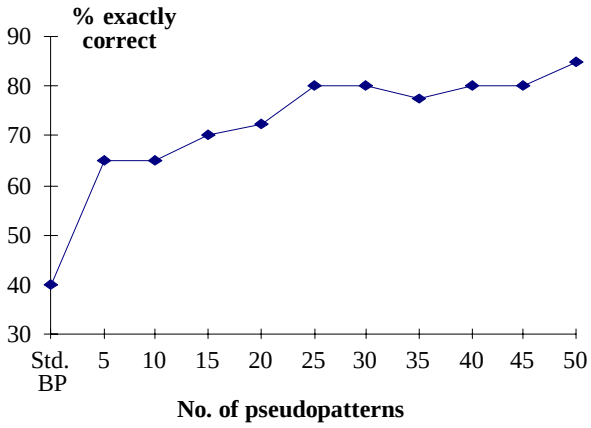

Figure 4. Percentage of all data exactly recalled by the network after serial presentation of all 20 patterns (median data).

After the twenty patterns were sequentially learned by the network, a test was made of the percentage of these items were exactly recognized (i.e., actual output within the criterion of 0.2 of the desired output). Figure 4 shows that with standard backpropagation this figure is around 40% whereas it improves rapidly to 65% with the addition to each new pattern to be learned with only 5 pseudopatterns from final storage. With 50 pseudopatterns added to each new item to be learned, exact recognition of all of patterns climbs to 85%.

After the network learned the twenty items sequentially, each item was individually tested to see how well the network remembered that item. The hypothesis was that the items that were learned first would show the greatest amount of error and that this curve would become more gradual as the number of pseudopatterns increased. Figure 5 shows that forgetting in the case of the pseudopatterns is indeed more gradual. As can be seen in the figure, the standard BP network is, on average, significantly above the 0.2 convergence criterion for *all* of the previously learned items, whereas the pseudo-recurrent network is at or below criterion for the last eight items learned (items 13-20) and within 0.05 of the criterion for items 7-12. Clearly, forgetting is taking place far more gradually in the pseudo-recurrent network than in the backpropagation network, where *none* of the 19 previously learned items are below criterion after the 20th item has been learned (Figure. 5).

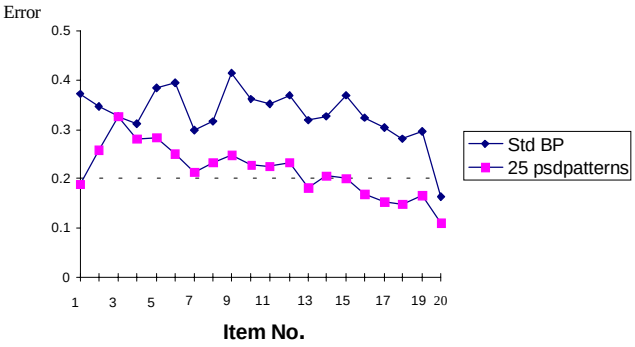

Figure 5. Amount of error for each of the 20 items learned sequentially after the final item has been learned to criterion (in this case, 0.2).

The most important advantage of the pseudo-recurrent network is its ability to learn patterns sequentially. This experiment shows that the forgetting curves for this type of network are considerably more gradual than with standard backpropagation. This experiment also points out the importance of interleaving approximations of the already-learned patterns with the new patterns to be learned.

Using pseudopatterns to transfer information in both directions between the buffer memory and final storage

Does this pseudo-recurrent architecture always require identical numbers of hidden nodes in the buffer memory and in the final-storage memory? As presented in this paper, the answer is yes. But it is not difficult to modify the model so that information is transferred both from final storage to the buffer memory as well as from the buffer memory to final storage by means of pseudopatterns. The advantage of directly copying the contents of the buffer memory to final storage is, of course, that there is no loss of information. The disadvantage is that the number of buffer and final-storage hidden units must be identical and that the idea of copying weights directly from the buffer memory to the final-storage memory lacks psychological plausibility.

Instead of copying the weights from the buffer storage to the final storage, the contents of the buffer memory are transferred to final storage by means of pseudopatterns. In other words, once the new data (along with the pseudopatterns from final storage) have been learned to criterion in the buffer memory, the buffer memory generates a number of pseudopatterns that are then learned by the final-storage memory. The output nodes of the buffer memory then become the teacher nodes for the final-storage memory.

This work has been done and is reported in French (1997).

General Conclusions

This paper proposes a "pseudo-recurrent" connectionist model of long-term memory in which a "buffer" memory and a permanent, "final-storage" memory interact. To ensure stability of previously learned information and sensitivity to new information this system new information to be learned is mixed with an approximation of previously learned information kept in the final-storage part of the network. The medium of information exchange from the final-storage memory to the buffer memory is pseudopatterns generated by the final-storage memory and learned by the buffer memory when it is learning new information. This pseudo-recurrent network has numerous advantages, the most significant being that it provides an effective, plausible means of doing sequential learning.

Acknowledgments

This work was supported by Belgian FNRS Grant No. D.4516.93 and PAI Grant No. P4/19.

References

Carpenter, G. and Grossberg, S. (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics* (26)23. 4919-4930.

Frean, M. and Robins, A. (1996) Transfer and Consolidation in Neural Networks: Further Exploration of the Pseudorehearsal Mechanism. (under review).

French, R. M. (1992) Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, **4**, 365-377.

French, R. M. (1994) Dynamically constraining connectionist networks to produce orthogonal, distributed representations to reduce catastrophic interference. In *Proceedings of the 16th Annual Cognitive Science Society Conference*. Hillsdale, NJ: Lawrence Erlbaum, 335-340.

French, R. M. (1997) Pseudo-recurrent connectionist networks: An approach to the "sensitivity–stability" dilemma. *Connection Science*, *9*(4).

Hebb, D. O. (1949). *Organization of Behavior*. New York, N. Y.: Wiley & Sons.

Hetherington, P. A. (1991) The sequential learning problem in connectionist networks. Unpublished Master's Thesis, Dept. of Psychology, McGill University, Montreal.

Kortge, Chris A., (1990). "Episodic Memory in Connectionist Networks", Proceedings of the 12th Annual Conference of the Cognitive Science Society, pp. 764Ð771, Hillsdale, NJ: Erlbaum.

Kruschke, J. K. (1993) Human Category Learning: Implications for Backpropagation Models. *Connection Science*, Vol. 5, No. 1, 1993.

Lewandowsky, S. & Goebel, R. (1991) Gradual unlearning, catastrophic interference, and generalization in distributed memory models. University of Oklahoma Psychology Department Technical Report, presented at the 1991 Mathematical Psychology Conference, Indiana University, Bloomington, IN

Lewandowsky, S. & Shu-Chen Li (1993) Catastrophic Interference in Neural Networks: Causes, Solutions, and Data. In *New Perspectives on interference and inhibition in cognition* F.N. Dempster & C. Brainerd(eds.). New York, NY: Academic Press.

McClelland, J., McNaughton, B., & O'Reilly, R. (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, 419–457.

McCloskey, M. & Cohen, N. J. (1989). "Catastrophic interference in connectionist networks: The sequential learning problem" The Psychology of Learning and Motivation, 24, 109-165.

McRae, K. & Hetherington, P. (1993) Catastrophic interference is eliminated in pretrained networks. In *Proceedings of the 15h Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum. 723-728.

Murphy, P. & Aha, D. (1992). UCI repository of machine learning databases. Maintained at the Dept. of Information and Computer Science, U. C. Irvine, CA.

Murre, J. (1992) The effects of pattern presentation on interference in backpropagation networks. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum. 54-59.

Ratcliff, R. (1990) Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions, 97, 285-308.

Robins, A. (1995) Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, **7**(2), 123–146.

Robins. A. (1996) Consolidation in Neural Networks and in the Sleeping Brain. (under review).

Sharkey, N. & Sharkey, A., (1995) An analysis of catastrophic interference. *Connection Science*, **7**(3-4), 301–329.