Jumpnet: A Multiple-Memory Connectionist Architecture

Jay G. Rueckl
Psychology Department
Harvard University
Cambridge MA 02138
jgr@isr.harvard.edu

Abstract

A jumpnet includes two memory storage systems: a processing network that employs superimpositional storage and a control network that recodes input patterns into minimally overlapping hidden patterns. By creating temporary, input-specific changes in the weights of the processing network, the control network causes the processing network to "jump" to the region of its weight space that is most appropriate for a particular input pattern. Simulation results demonstrate that jumpnets exhibit only moderate levels of interference while retaining the computational advantages of superimpositional memory.

Introduction

Connectionist networks store knowledge in a superimpositional manner: the association between two patterns of activity is encoded across many connections, and each connection plays a role in encoding many associations. Superimpositional memory storage underlies many of the attractive computational properties of connectionist networks, such as the capacity for automatic generalization, prototype extraction, robustness in the face of noisy input patterns, and graceful degradation in performance in response to damage. However, it also renders a network susceptible to interference—the weight changes made to strengthen one association often have the effect of degrading memory for previously learned associations stored by the same weights. While some degree of interference may be tolerable (and is known to occur in humans), it has been suggested that the "catastrophic" levels of interference suffered by connectionist networks raise serious questions about their suitability as either models of human cognition or artificial intelligence devices (McCloskey & Cohen, 1989; Ratcliff, 1990).

Interference and Weight Space. A network's pattern of connectivity can be described as a point in *weight space*— a multidimensional space where each dimension corresponds to one of the connections, and the location along that dimension is determined by the weight of the corresponding connection. Learning corresponds to a movement through weight space in a direction that strengthens a particular association. Note, however, that if the current pattern of connectivity encodes information about previously learned associations, then moving away from that point in weight space may result in the loss of this information (i.e. retroactive interference). The potential for retroactive interference can be reduced by limiting the "stepsize" of the movement through weight space, but only at the expense of slowing the rate at which new associations can be acquired.

This sort of analysis raises an interesting possibility. Suppose that at the time of learning a network took a small step in the direction specified by the learning algorithm, and that at retrieval the network adjusted its weights by making another, temporary change in the same direction. In this case, the network would exhibit tolerable levels of interference and relatively fast learning of new associations while retaining the virtues of superimpositional storage.

The Jumpnet Architecture

Jumpnets are networks designed to do just this. Jumpnets include two interacting subsystems that store associative information

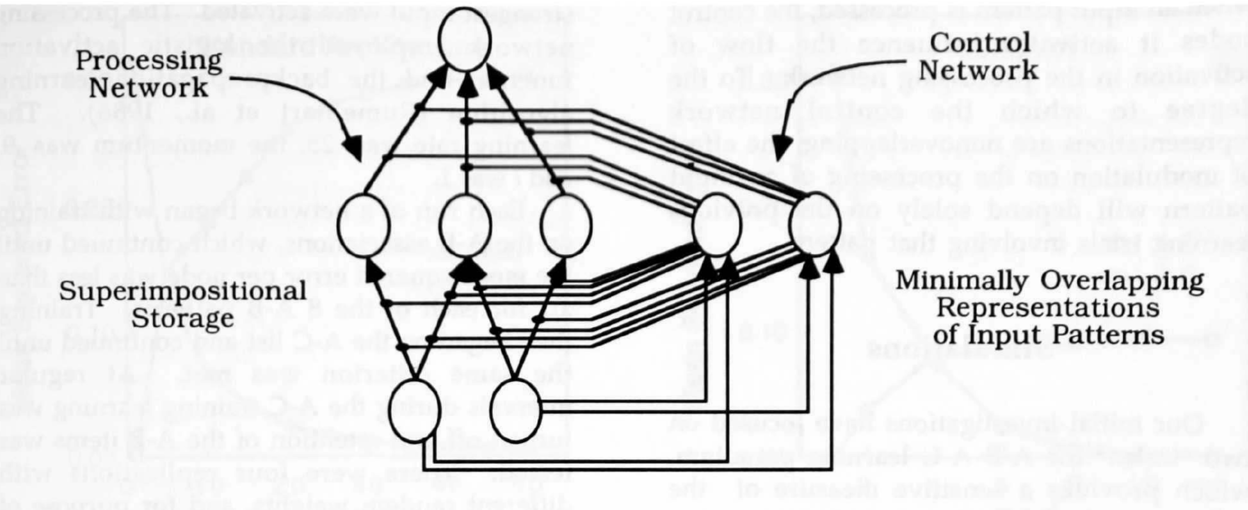

Figure 1. The Jumpnet Architecture.

in complementary ways (see Figure 1). One subsystem (the processing network) uses standard connectionist activation and learning algorithms, and thus stores information in a superimpositional format. The other subsystem (the control network) represents the input in an approximately orthogonal manner, thus minimizing the effects of interference (but also losing the advantages of superimpositional storage). The role of the control system is to create temporary, pattern-specific weight changes in the processing system. In effect, the control system stores information about the weight changes appropriate for an input pattern and causes these changes to be made whenever that input pattern is processed.

The influence of the control system on the processing network occurs through a process called *weight modulation*. In a standard network, the activation of a node is a function of the input that node receives from other nodes: $input_j = \Sigma w_{ij} a_i$. In a jumpnet, the strengths of the weights are modulated by the control units before the input to a node is computed. In particular, the input to a node is given by $input_j = \Sigma W'_{ij} a_i$, where $W'_{ij} = w_{ij} + \Sigma w_{ijk} a_k$, and k indexes the nodes in the control network.

The influence of the control system thus depends on both the activation of the units in the control module and the strength of the modulatory weights. The modulatory connections store part of the weight changes that would otherwise be made directly on the weights in the processing network. Changes in the strengths of these connections are learned according to the equation $\Delta w_{ijk} = l \Delta w_{ij} a_k$, where l controls the rate of learning.

One might note that given this learning rule, the modulatory weights could be susceptible to the same problem with interference that plagues the processing network. However, interference can be minimized by choosing an appropriate representational scheme for the control nodes. In the simulations reported below, the activations of the control nodes were determined by setting a fixed, random weight matrix between the input nodes and the control nodes, and then computing the input to each control node given this matrix and an input pattern. Of the N nodes in the control network, the activations of the M nodes with the largest inputs were set to 1, with the activations of the other nodes set to 0. As N increases, the probability that two input patterns will activate the same control nodes decreases, so that, if M is small, the patterns of activation across the control nodes become approximately orthogonal.

To summarize, the information stored by the modulatory connections is used to create temporary, pattern-specific changes in the functional strength of the connections in the processing network. In effect, the control network allows a jumpnet to make temporary, pattern-specific "jumps" in weight space.

When an input pattern is processed, the control nodes it activates influence the flow of activation in the processing network. To the degree to which the control network representations are nonoverlapping, the effect of modulation on the processing of an input pattern will depend solely on the previous learning trials involving that pattern.

Simulations

Our initial investigations have focused on two tasks: the A-B A-C learning paradigm, which provides a sensitive measure of the influence of retroactive interference on a network's behavior, and the autoencoder task, which provides a measure of a network's ability to perform automatic generalization.

Retroactive Interference: The A-B A-C Task

In the A-B A-C paradigm subjects learn two lists of paired associates. After learning the A-B list the subject is trained on the A-C list, in which the A terms from the first list are paired with new associates. It has often been observed that learning the A-C list interferes with a person's ability to recall the A-B list. McCloskey and Cohen (1989) conducted an extensive series of simulations using this paradigm, and on the basis of their results suggested that the amount of interference exhibited by connectionist networks in this task is of "catastrophic" proportions.

A series of simulations were conducted to investigate the performance of the jumpnet architecture in this task. The parameters of the present simulations were similar to those used by McCloskey and Cohen. The processing network included 20 input units, 30 hidden units, and 10 output units. There were 8 A-B associations and 8 A-C associations. As in the McCloskey and Cohen simulations each input pattern included a 10-unit pattern representing the A item and a 10-unit context pattern that distinguished list A-B from list A-C. There were 48 control nodes, and for each input pattern the 8 control nodes that received the strongest input were activated. The processing network employed the logistic activation function and the backpropagation learning algorithm (Rumelhart et al., 1986). The learning rate was .25, the momentum was .9, and l was 1.

Each run of a network began with training on the A-B associations, which continued until the mean squared error per node was less than .01 for each of the 8 A-B patterns. Training then began on the A-C list and continued until the same criterion was met. At regular intervals during the A-C training learning was turned off and retention of the A-B items was tested. There were four replications with different random weights, and for purpose of comparison the processing subnetwork of each jumpnet was also run in isolation from the control network.

Several aspects of the results are of interest. First, jumpnets learned much faster than standard feedforward networks. Jumpnets learned the A-B associations in an average of 14 epochs and the A-C associations in 9.75 epochs. In contrast, the standard architecture required an average of 28.25 epochs for the the A-B associations and 46.5 epochs for the A-C associations. Note that although the standard architecture exhibited proactive interference (i.e., the A-C associations were more difficult to learn than the A-B associations), the jumpnet architecture did not. Indeed, if anything the jumpnets exhibited a positive transfer effect.

In addition, jumpnets were far less susceptible to retroactive interference. As depicted in Figure 2, the results with the standard feedforward architecture were very similar to those reported by McCloskey and Cohen (1989): Learning the A-C associations quickly eliminated retention of the A-B pairs. In contrast, in addition to learning the A-C lists more quickly, jumpnets also suffered only moderate amounts of retroactive interference.

The results of the A-B A-C simulations indicate that jumpnets can learn reasonably quickly and with tolerable levels of interference. The next set of simulations investigates whether this has been accomplished without the loss of the more positive effects of superimpositional storage.

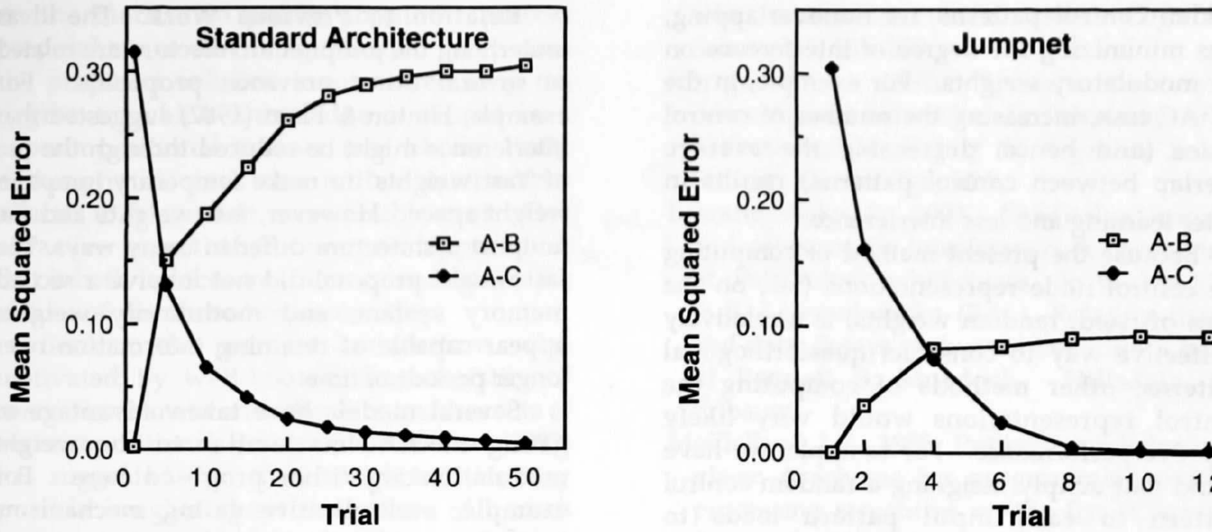

Figure 2. The results from the A-B A-C simulations.

Automatic Generalization: The Autoencoder Task

In the autoencoder task, a network is trained to produce an output pattern identical to the input pattern. Under typical conditions, superimpositional storage allows a network trained on one set of patterns to respond appropriately to patterns that weren't included in the training set (i.e. automatic generalization). In the present simulations jumpnets learned to autoencode a set of 32 randomly generated training items. There were 10 input nodes, 10 hidden nodes, 10 output nodes, and 48 control nodes. The learning rate was .75, the momentum was .9, and *l* was 1. After the training set had been learned, the network was presented with 16 new items in a generalization task. Again, there were four replications with different starting weights, and standard feedforward networks were run as matched controls.

Table 1. The results from the autoencoder task.

		Error on test set	
Network	Trials to learn training set	Before training	After training
Jumpnet	12.5	.170	.043
Feedforward	33.5	.170	.036

As can be seen in Table 1, the jumpnet architecture learned the training set almost three times faster than the standard feedforward architecture. In addition, after training the standard architecture performed only marginally better on the test set than did the jumpnet architecture. Thus, these results reveal that the reduced susceptibility to interference displayed in the A-B A-C task was not purchased with the loss of the virtues of superimpositional storage.

Discussion

Jumpnets include two components: a standard connectionist network that uses superimpositional storage to abstract and make use of the regularities underlying a domain, and a control network that recodes input patterns into nonoverlapping hidden representations and that, via weight modulation, enables the processing network to make temporary jumps in weight space that are well-suited for particular input patterns. The simulation results suggest that in this way the jumpnet architecture avoids much of the cost of superimpositional storage while still benefiting from its virtues.

Of course, there are boundary conditions beyond which jumpnets do not perform as well as in the simulations reported above. Two factors seem to be particularly important. The first of these concerns the representational scheme used by the control network. In general, jumpnets perform better to the degree that their hidden control patterns are nonoverlapping, thus minimizing the degree of interference on the modulatory weights. For example, in the AB AC task, increasing the number of control nodes (and hence, decreasing the average overlap between control patterns) results in faster learning and less interference.[1]

Because the present method of computing the control node representations (i.e., on the basis of fixed, random weights) is a relatively ineffective way to construct quasiorthogonal patterns, other methods of computing the control representations would very likely improve performance. For example, we have found that simply assigning a random control pattern to each input pattern leads to substantially better performance. Although such representations could not be computed by a network without substantial pretraining, these results (together with those of the simulations reported above) indicate that ideally, input patterns should be represented by minimally overlapping control patterns, and to the degree that there is overlap, the similarity of the control patterns should not be correlated with the similarity of the input patterns.

A second important factor concerns the relative influence of the two components on the behavior of the network. As would be expected, increasing the relative contribution of the control network (for example, by increasing the learning rate on the modulatory connections relative to the learning rate on the connections in the processing network) reduces interference, but also diminishes the capacity for generalization.

1. Interestingly, although performance depends both on the number of control nodes and the proportion of these nodes that are activated, as the total number of control nodes increases the effect of the latter factor decreases, provided that the proportion is neither very small (e.g. 1 or 2%) nor very large (30-50%). In the former case, the control representations are generally orthogonal, but if two patterns do activate a common control node the system cannot overcome the lack of redundancy. In the latter case, a given pair of input patterns is likely to overlap a number of common control nodes, resulting in interference on the modulatory weights.

Relation to Previous Work. The ideas underlying the jumpnet architecture are related to several other previous proposals. For example, Hinton & Plaut (1987) suggested that interference might be reduced through the use of "fast weights" to make temporary jumps in weight space. However, fast weights and the jumpnet architecture differ in many ways. The fast-weight proposal did not involve a second memory system, and modulatory weights appear capable of retaining information over longer periods of time.

Several models have taken advantage of gating mechanisms similar to the weight modulation algorithm proposed here. For example, multiplicative gating mechanisms have been used to provide rotation- and translation-invariant letter and word recognition (Hinton; 1981 McClelland, 1985). Of particular relevance here is Sloman and Rumelhart's (1992) proposal to reduce interference through "episodic gating." Although they differ in many of their details (e.g., the episodic gating model does not use hidden nodes in the standard sense, and thus is limited in the class of computations it can compute), the Sloman and Rumelhart model is similar in spirit to the present approach.

Several previous papers have also investigated the computational properties of modular connectionist architectures (Jacobs et al., 1991; Rueckl et al., 1989). For the most part, this work has focused on the advantages of modularity in learning several concurrently acquired tasks. However, Jacobs et al. (1991) demonstrated that a network that uses different subnetworks to compute each of two mappings can learn these mappings sequentially without interference. The Jacobs et al. model differs from the jumpnet approach primarily in that in the former model the component networks compete for the right to respond to a given input, whereas in the latter model one subnetwork modulates the computations performed by the other. In future work it will be interesting to investigate the implications of this distinction.

Finally, a number of authors have suggested that, for one consideration or another, the "catastrophic interference" problem is not as severe as has been claimed (c.f., Hetherington & Seidenberg, 1989; Kortge, 1990; Lewandowsky, 1991; Sloman & Rumelhart, 1992). While many of their points appear to be on the mark, two points about the present proposal are worth noting. First, in many cases the proposed solution involves reducing the degree to which knowledge is stored in a superimpositional manner (see Murre, 1992). Although this tack can reduce interference effects, it does so at considerable cost. One attractive aspect of the jumpnet approach is that it both reduces interference *and* retains the virtues of distributed representation.

Second, the jumpnet architecture was motivated by well-known characteristics of connectionist networks: the costs and benefits of superimpositional storage. McClelland (1992) has suggested that these characteristics may provide insights about the nature of human memory, and particularly about why our memory seems to rely on several distinct systems with rather different properties (see Schacter, 1992). It is an intriguing possibility that, even if the jumpnet approach is wrong in its details, its fundamental characteristics may well shed light on the workings of human memory. With this in mind, we are now examining the behavior of jumpnet systems with with the goal of generating empirical predictions about human behavior.

Acknowledgements

Thanks to Christopher Rasmussen for his help in running the simulations.

References

Hetherington, P. A., & Seidenberg, M. S. 1989. Is there "catastrophic interference" in connectionist networks? In *Proceedings of the 11th Annual Meeting of the Cognitive Science Society*, Ann Arbor, MI.

Hinton, G. 1981. A parallel computation that assigns canonical object-based frames of reference. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence.*

Hinton, G. & Plaut, D. 1987. Using fast weights to deblur old memories. *Proceedings of 9th Annual Meeting of the Cognitive Science Society*, Amherst, MA.

Jacobs, R. A., Jordan, M. I., & Barto, A. G. 1991. Task decomposition through competition in a modular connectionist architecture: The what and where visual tasks. *Cognitive Science, 15*, 219-250.

Kortge, C. A. 1990. Episodic memory in connectionist networks. In *Proceedings of the 12th Annual Meeting of the Cognitive Science Society*,, Cambridge, MA.

Lewandowsky, S. 1991. Gradual unlearning and catastrophic interference: A comparison of distributed architectures. In W. Hockley, & S. Lewandowsky (Eds.), Relating theory and data: Essays on human memory in honor of Bennett B. Murdock. Hillsdale, NJ: Erlbaum.

McClelland, J. L. 1985. Putting knowledge in its place: A scheme for programming parallel processing structures on the fly. *Cognitive Science, 9*, 113-146.

McClelland, J. 1992. Talk given at Harvard University, September 30.

McCloskey, M., & Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower (Ed.), *The psychology of learning and motivation.* (p. 109-165). New York: Academic Press.

Murre, J. 1992. The effects of pattern presentation on interference in backpropagation networks. *Proceedings of the 14th Annual Meeting of the Cognitive Science Society*, Bloomington, IN.

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

Rueckl, J. G., Cave, K. R., & Kosslyn, S. M. 1989. Why are "What" and "Where" processed by two cortical visual systems? A computational investigation. *Journal of Cognitive Neuroscience, 1*, 171-186.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In D. Rumelhart & J. McClelland (Eds.), *Parallel Distributed Processing.* Cambridge, MA: MIT Press.

Schacter, D. L. 1992. Priming and multiple memory systems: Perceptual mechanisms of implicit memory. *Journal of Cognitive Neuroscience, 4*, 244-256.

Sloman, S., & Rumelhart, D. 1992. Reducing interference in distributed memories through episodic gating. In A. F. Healy, S. M. Kosslyn, & R. Shiffrin (Eds.), *Papers in Honor of W. K. Estes.* Hillsdale, NJ: Erlbaum.