Multiassociative Memory

John F. Kolen
Jordan B. Pollack

The Laboratory for AI Research
Department of Computer and Information Science
The Ohio State University
Columbus, OH, 43210
Telephone:(614)292-7402
kolen-j@cis.ohio-state.edu
pollack@cis.ohio-state.edu

Abstract

This paper discusses the problem of how to implement many-to-many, or multi-associative, mappings within connectionist models. Traditional symbolic approaches wield explicit representation of all alternatives via stored links, or implicitly through enumerative algorithms. Classical pattern association models ignore the issue of generating multiple outputs for a single input pattern, and while recent research on recurrent networks is promising, the field has not clearly focused upon multi-associativity as a goal. In this paper, we define multiassociative memory MM, and several possible variants, and discuss its utility in general cognitive modeling. We extend sequential cascaded networks (Pollack 1987, 1990a) to fit the task, and perform several initial experiments which demonstrate the feasibility of the concept.

This paper appears in *The Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society.* August 7-10, 1991.

Multiassociative Memory[1]

John F. Kolen
Jordan B. Pollack

The Laboratory for AI Research
Department of Computer and Information Science
The Ohio State University
Columbus, OH, 43210
Telephone:(614)292-7402
kolen-j@cis.ohio-state.edu
pollack@cis.ohio-state.edu

Abstract

This paper discusses the problem of how to implement many-to-many, or multi-associative, mappings within connectionist models. Traditional symbolic approaches wield explicit representation of all alternatives via stored links, or implicitly through enumerative algorithms. Classical pattern association models ignore the issue of generating multiple outputs for a single input pattern, and while recent research on recurrent networks is promising, the field has not clearly focused upon multi-associativity as a goal. In this paper, we define multiassociative memory MM, and several possible variants, and discuss its utility in general cognitive modeling. We extend sequential cascaded networks (Pollack 1987, 1990a) to fit the task, and perform several initial experiments which demonstrate the feasibility of the concept.

Introduction

In associative memory models, a function is applied to a pattern which transforms it to another pattern. This transformation, or association from a domain set of noisy or partial patterns to a range set of correct patterns is often offered as a model of memory retrieval or recall. Several connectionist models of associative memory exist, such as linear associators (Kohonen, 1972;Anderson, 1972), Hopfield networks (Hopfield, 1982), and feed-forward networks (Rumelhart, Hinton & Williams, 1986), each assuming a many-to-one mapping from domain to range Figure 1a. For noisy or partial memory retrieval, or for perceptual categorization, this assumption might prove valid, however, many other tasks exist, each indescribable within associative frameworks. Consider associating a word with its possible lexical categories or meanings, a chess position to possible next moves, or a category to prototypical members; each input can have a multitude of possible outputs. Even categorization itself is known to vary over time in individual subjects (McCloskey and Glucksberg, 1978).

In an attempt to address the issue of multiple output responses, this paper discusses the nature of *multiassociative* memories capable of responding with one output from a set of possible outputs to an input. The theoretical basis of this class of models is the many to many mapping. Standard associative models use many to one mappings, where several input patterns can elicit the same output pattern. Any system with an internal state, such as a recurrent network, can be viewed as a many to many mapping, and this is our initial choice of model (Figure 1b).

This paper begins by presenting formal definitions of mappings, establishing the relationship between many to many mappings and multiassociative memories, and identifying several problems in cognitive science which exhibit many to many mapping properties. The discussion will then turn to the relative computational and psychological merits of two opposing implementation strategies, namely explicit storage and enumeration. After establishing the practical superiority of enumeration, this strategy will guide the implementation of a multiassociative memory system using a sequential cascaded network, a high-order recurrent connectionist architecture. Reflection on the implementation process and its results suggests possible difficulties with the recurrent network, indicating directions for further work.

Many to Many Mappings

Associative memories take input patterns, process them, and return output patterns (Kohonen, 1984). One way of describing this operation is as a mapping from input patterns of the domain set to the generated output patterns the range set. Mappings come in four varieties based on the relations between domain and range: one to one, many to one, one to many, and many to many. A simple example of a many to many mapping is (A-> {B, C}, B-> {C, E}, C->{E, F}, D->{E, F)} with domain set {A, B, C, D} and range set {B, C, E, F}.

To capture such an association, a multiassociative memory system must be correct and complete. Thus the multiassociative memory acquisition problem is:

> Given a many to many mapping from a set of input patterns to output patterns, find a system that can associate each domain element to at least one element from its range set during any memory access *(correctness)* and also eventually generate all correct range elements within a finite number of memory accesses (*completeness)*.

It is quite trivial to implement this concept exercising conventional symbolic means, employing either explicit storage or enumeration of the associations. One just explic-

1. This work supported by office of Naval Research grant number N00014-89-J1200.

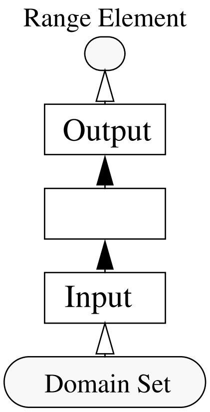

Figure 1a:
Associative Memory
Many to One

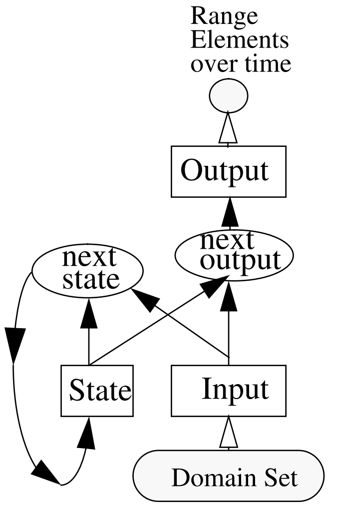

Figure 1b:
Multiassociative Memory
State Machine Conceptualization

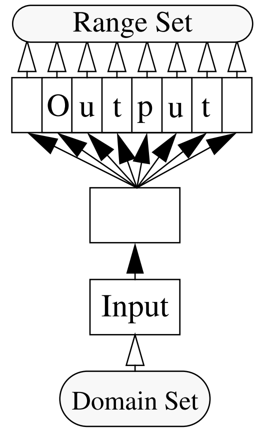

Figure 1c:
Multiassociative Memory
Power Set Approach

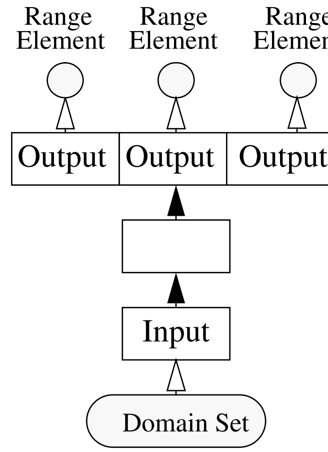

Figure 1d:
Multiassociative Memory
Field Approach

itly stores associations as a linked-list in random-access memory, and retrieve either all the outputs, or keep a counter and get one at a time. This "explicit storage" method does not generalize to very large systems, like chess moves, where it is impractical to store all associations in a memory. Secondly, it is too powerful to be a good cognitive model in that it has no distinction between "recognition" and "recall". Consider the daunting task of recalling the names of all fifty states, as opposed to the trivial problem of recognizing "New Mexico" as one of them.

Connectionist models also have explicit storage solutions. Through the manipulation of output patterns, the *power set* approach involves allocation of one output node for each element in the range (Figure 1c). Thus, any possible subset of the range can be captured by this representation. This localist approach has been used to encode possible moves in a tic-tac-toe game (McClelland and Rumelhart, 1986) and the spreading of activation in a semantic network (Collins and Loftus, 1975). Unconstrained representational power, however, is not without its costs. Since individual elements of a range of n units can be adequately represented with a binary pattern of length $lg(n)$, using n units is exponentially more expensive than a sequential retrieval model.

A possible circumvention to this exponential explosion is to limit the size of the subsets of the range by allocating a fixed number of output *fields* (slots, registers, or buffers) to hold the representations of the elements. Figure 1d illustrates a system capable of outputting at most three range elements for a given domain element. But determining the needed resources *a priori* may be difficult, or in principle impossible: What's the maximum number of possible moves from any given chess board configuration, or the maximum number of words in a sentence? Overestimation wastes resources and underestimation might fail at a critical point.

Enumerative approaches bypass this storage problem by not explicitly operating on the possible responses themselves, rather they calculate the desired responses sequentially. Since only one range element has to be stored at a time, memory usage will be lower. Such a system must have an "internal state" (or counter) that can be used to keep track of the enumeration algorithm, and suitably powerful computational means to compute the output set.

Connectionist Implementations

A connectionist implementation of Multiassociative Memory (MM) using enumeration would have an underlying state machine, which can be implemented as a recurrent network. Many such networks exist (*e. g.* Jordan, 1987; Elman, 1988; Pollack, 1987) and already exhibit many to many mapping ability. When the input remains stable long enough to establish an periodic attractor (a fixed point or limit cycle) in the output, each element of the attractor corresponds to one of the correct associations to the input pattern. Such a "temporal MM" accesses the range set in a particular order for each input, whereas an "atemporal MM" just guarantees access to the proper range subsets through an aperiodic attractor (aka strange attractor).

To illustrate temporal/atemporal distinction, consider an implementation of a many to many mapping using a recursive auto-associative memory (RAAM) (Pollack, 1990b). A RAAM consists of two functions, an encoder and a decoder. The encoder takes two input patterns and combines them into a single output pattern. The decoder takes one input pattern and generates the two patterns the encoder used to build it. Since all patterns have the same size, the strategic application of the encoder and decoder can construct representations of recursive data structures such as trees and stacks. A MM can be built around a RAAM by assigning non-terminal symbols to the domain elements which eventually decode into elements in the image (see Figure 2). A counter, or random number generator, could be used as the internal state mechanism specifying a unique decoding paths, *i. e.* whether the left or right decoding would be re-

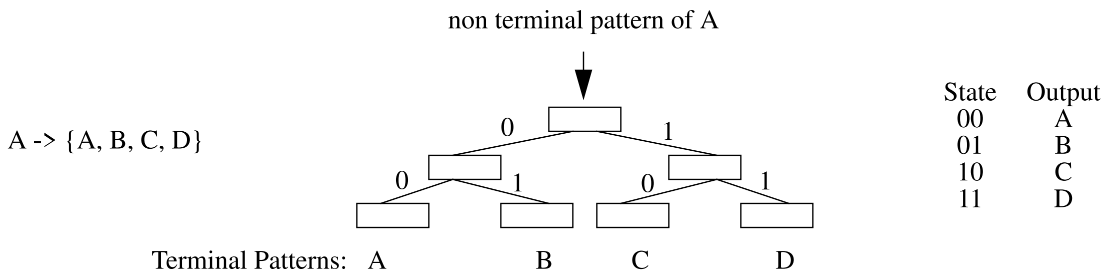

Figure 2: Using a RAAM for Multiassociative Memory

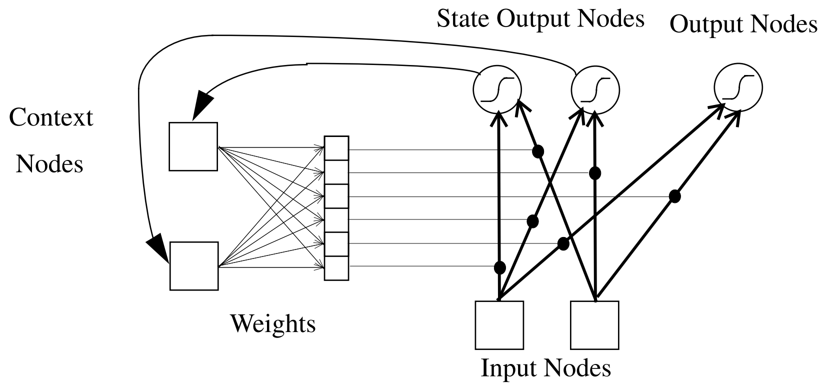

Figure 3: A Sequential Cascaded Network SCN

tained for further expansion. The counter implementation introduces temporal relationships between outputs since it would periodically visit every decoding path, while the random number generator ensures little or no correlation between two successive outputs by randomly varying the paths.

A drawback of this implementation is that tree structure and state organization must be specified *a priori*. A more interesting approach involves the evolution of state organization through learning. Learning, for a MM problem, has two constraints. First, no "reset" information can be provided to the network between changing inputs. Second, only a small number of examples will be available to the learning procedure. The system must be able to generalize its internal state machine to have acceptable behavior over finite but unbounded sequences of inputs. Sequential cascaded networks have exhibited such inductive capabilities, as demonstrated in (Pollack, in press), and therefore been selected for this initial study.

The sequential cascaded network (SCN) is a higher order (sigma-pi) recurrent network. The weights are stored in a three-d array (Context Network) that is multiplied by the current state yielding a weight matrix. This matrix, called the Function Network, is multiplied by the input vector resulting in a net input vector. A sigmoid function is applied to elements of this vector, yielding the next state and output vectors. SCN's were originally used in a formal language decision task, in which the network observed a sequence of symbols from a two symbol alphabet (0 and 1) and calculated a single output value plus a recurrent state vector. The desired output value was 1 if the input string seen so far was contained in the language, and 0 otherwise. Figure 3 contains a schematic representation of a SCN.

This paper extends the sequential usage of the cascaded network by utilizing an output vector longer than a single element. Increasing the output width from a single value necessitates a slight modification to the learning rules described in (Pollack, 1990). More output nodes imply each will contribute error to the context node error. The new error function derivative with respect to the weights feeding into the state output nodes is

$$\frac{\partial E}{\partial w_{ijk}} = \left(\sum_{a \in A} (z_a(n) - d_a)\, g(z_a(n)) \left(\sum_l y_l(n) w_{alk} \right) \right) g(z_i(n-1)) y_j(n-1) z_k(n-2)$$

and the error derivative for the weights leading to the output units is

$$\frac{\partial E}{\partial w_{ajk}} = (z_a(n) - d_a)\, g(z_a(n)) y_j(n) z_k(n-1).$$

In both equations, $z_i(n)$ is the output of unit i at time n, $y_i(n)$ is the i^{th} input at time n, $g(x)$ is the derivative of the nonlinear squashing function in terms of the activation of the unit, d_a is the desired output for output unit a from the set A of output nodes, and E is the standard sum of squared error function. The simulations below use baseline back

Input Sequences:	AAAA	BBBB	CCCC	DDDD
Training Sequences:	BCBC	DEDE	EFEF	EFEF

Figure 4a: Sample MM training set

Input
Sequence: AA**AABB**BBCC**CCDD**DDAA**AA

Output: BCBCBCDEDEDEEFEFEFEFEFEFBCBCBC
** stands for 16 more input cycles

Figure 4b: Output From SCN as a MM

propagation (gradient descent with momentum) to modify the weights in the network.

The training paradigm used in this paper attempts to exploit the inductive ability of this system by limiting the number of exemplar sequences. One way to develop an SCN training set for a particular mapping is to train the network sequence through the associated range elements while the input to the SCN remains constant with the input vector for the domain element. For instance, the association A -> {B, C} give rise to the sequences

Input: AAAAAAAAAAAA...

Output: BCBCBCBCBCBC...

Since SCN's have shown induction of similarly unbounded sequences from short initial sequences, only two such cycles for each domain element are presented to the network during training. Figure 4a shows the training sequences used to capture a small many to many mapping. This training set, along with domain and range element pattern vectors, was presented to the SCN learning program and after 10 passes through the training set (weight update occurred after presentation of all sequences) the network demonstrated that it could reproduce the training set within tolerance. Tolerance was one half the difference of the maximum and minimum values. Table 1 describes network parameters used for this experiment and the next. In this experiments, the input and output symbols were converted to fixed width random binary patterns. At this point the network captures only the explicit sequences in the training set. To generalize, the learning program must **overtrain** the network. Overtraining forces the network to go through phase changes, drastic alterations in its behavior, as it tries to separate the outputs more and more. Pollack (In press) demonstrates the characteristic shift in behavior with a network which learns successively larger finite subsets of the parity language (even number of 1's in the input string) until it suddenly acquires the parity language for any length input string. In this experiment, the learning procedure stopped only when 10 consecutive overtraining passes occurred.[2] Figure 4b illustrates the response of the SCN to the string $A^{20}B^{20}C^{20}D^{20}A^{20}$.

2. During an overtraining pass weights are modified, even though the all of the outputs are with tolerance of their desired values.

	Letters	Figures
Learning rate	0.1	0.1
Momentum rate	0.3	0.4
Overtraining	10	20
High Symbol Value	1.0	1.0 & see fig. 4
Low Symbol Value	-1.0	-1.0 & see fig. 4
Squashing Function	tanh	tanh
Error Tolerance	1.0	0.125
State Width	3	4
Pattern Width	3	2
Initial State Value	0.9	0.25

Table 1: Training Parameters

The next experiment demonstrates the possibility of storing visual images as many to many mappings. The domain of this mapping is the set {TRIANGLE, SQUARE}, represented as binary patterns, {(0,1), (1,0)}. The range set of each of these symbols is the vertices of that object as shown in Figures 5a and 5b. A similar training method was employed and Figures 5c, 5d, and 5e illustrate the results of presenting the input string $\text{TRIANGLE}^{20}\text{SQUARE}^{20}\text{TRIANGLE}^{20}$.

Discussion and Conclusion

In both studies, the network created for itself a periodic output attractor for each input pattern utilizing hidden state information (which will be explored in a longer paper). We observed different organizations of internal state in different training runs due to back propagation's sensitivity to the initial weights (Kolen & Pollack, 1990). In most cases, we also observed that quickly switching between symbols can lead to several cycle delays of the correctness condition as the output dances around the target response. We believe this parallels the "tip of the tongue" phenomena and suggests that enumerative multiassociative memories could provide a useful foundation for modeling this psychological phenomena and others. Nature's solution to the scaling issue, which affects all connectionist models, may turn out to be tied to the distinction between recall and recognition. As has been pointed out in the past, inverting a categorization model or an associative memory is a difficult problem, because it requires a many-to-many mapping, not supplied by the standard architectures (Williams, 1986). The inversion of a multiassociative memory would remain a multiassociative memory.

The main difficulty with our initial MM model, and with most recurrent back propagation systems is the reliance on global state information and synchronous update of that information (Grossberg, 1987). As nature teaches us, it is much cheaper to build asynchronous and local parallel processing, which unfortunately can be quite unstable. The multiassociative task is well-suited for an asynchronous and local treatment. Following Hanson 1990, we have been experimenting with a model in which each weight in a system is a dynamical system, loosely coupled to nearby connections, asynchronously changing over time. Learning does not constrain the network to taking on particular values, but of meeting the necessary functionality of the network in terms of an atemporally multiassociative memory.

By focusing on the many-to-many mapping as a cogni-

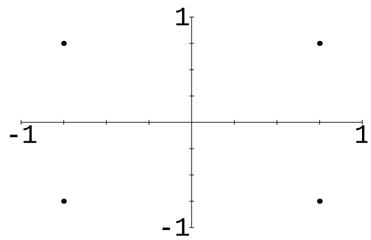

Figure 5a: SQUARE training data

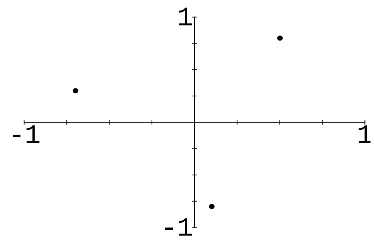

Figure 5b: TRIANGLE training data

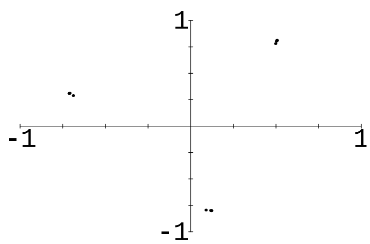

Figure 5c: cycles 1-20

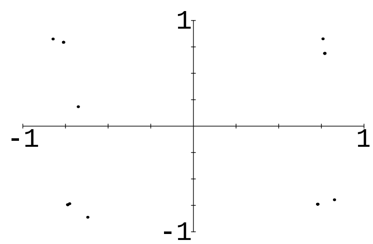

Figure 5d: cycles 21-40

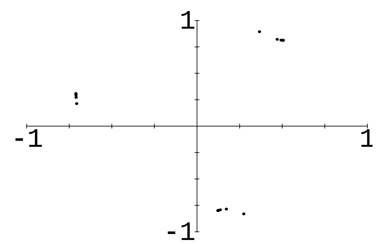

Figure 5e: cycles 41-60

Figure 5: Training Sets and Output displays for the {TRIANGLE, SQUARE} problem.

tive task, implemented in an enumerative rather than explicit storage memory, we see a new general-purpose connectionist model beyond the categorization and associative memory of feed-forward and relaxation models, and the prediction and sequence generation of recurrent networks.

Acknowledgments

Discussions with members of the LAIR connectionist group have been invaluable. Comments from Mary Jo Carnot helped round out the paper and make it more accessible.

References

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

Collins, A. M. and Loftus, E. F. 1975. A spreading activation theory of semantic processing. *Psychological Review*, 82:240–247.

Elman, J. L. 1988. Finding structure in time. Technical Report CRL 8801, University of California, San Diego.

Grossberg, S. 1987. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11:23–63.

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

Jordan, M. I. 1987. Supervised learning and systems with excess degrees of freedom. Technical Report COINS Technical Report 88-27, Massachusetts Institute of Technology, Boston.

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

Kohonen, T. 1984. *Self-Organization and Associative Memory*. Springer-Verlag.

Kolen, J. and Pollack, J. 1990 Back-Propagation is sensitive to Initial Conditions. *Complex Systems* 4, 269-280.

McClelland J., & Rumelhart, D. 1986 *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MIT Press, II, 48-53.

McCloskey, M. and Glucksberg, S. 1978. Natural categories: Well defined or fuzzy sets? *Memory and Cognition*, 6:462–472.

Pollack, J. B.1987. Cascaded back propagation on dynamic connectionist networks. In *Proceedings of the Fourth Annual Cognitive Science Conference*. Seattle, WA, 391-404.

Pollack, J. B. 1990. Language acquisition via strange automata. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*. Cambridge, MA, 678-685.

Pollack, J. B. 1990. Recursive autoassociative memories. *Artificial Intelligence*, 46, 1, 77-105.

Pollack, J. B. In press The acquisition of dynamical recognizers *Machine Learning,* 1991.

Rumelhart D. Hinton, G., and Williams R. 1985. Learning representations by back-propagating errors. *Nature*, 323, 533-536.

Williams, R. 1986 Inverting a connectionist network mapping by back-propagation. *Proc. Eight Annual Conf. of the Cognitive Science Society,* Amherst. 859-865.