1-1995

Adaptive resonance associative map

Ah-hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Pergamon

Neural Networks, Vol. 8, No. 3, pp. 437–446, 1995

0893-6080/95 $9.50 + .00

0893-6080(94)00092-1

CONTRIBUTED ARTICLE

Adaptive Resonance Associative Map

AH-HWEE TAN

National University of Singapore

(*Received* 11 *May* 1994; *revised and accepted* 21 *September* 1994)

Abstract—*This article introduces a neural architecture termed Adaptive Resonance Associative Map (ARAM) that extends unsupervised Adaptive Resonance Theory (ART) systems for rapid, yet stable, heteroassociative learning. ARAM can be visualized as two overlapping ART networks sharing a single category field. Although ARAM is simpler in architecture than another class of supervised ART models known as ARTMAP, it produces classification performance equivalent to that of ARTMAP. As ARAM network structure and operations are symmetrical, associative recall can be performed in both directions. With maximal vigilance settings, ARAM encodes pattern pairs explicitly as cognitive chunks and thus guarantees perfect storage and recall of an arbitrary number of arbitrary pattern pairs. Simulations on an iris plant and a sonar return recognition problems compare ARAM classification performance with that of counterpropagation network, K-nearest neighbor system, and back-propagation network. Associative recall experiments on two pattern sets show that, besides the advantages of fast learning, guaranteed perfect storage, and full memory capacity, ARAM produces a stronger noise immunity than Bidirectional Associative Memory (BAM).*

Keywords—Self-organization, Neural network architecture, Associative memory, Heteroassociative recall, Supervised learning.

1. INTRODUCTION

This article introduces a neural architecture termed Adaptive Resonance Associative Map (ARAM) that performs rapid, yet stable, heteroassociative learning in a real-time environment. Whereas a similar supervised Adaptive Resonance Theory (ART) system, ARTMAP (Carpenter et al., 1992), consists of two ART modules interconnected by an inter-ART associative map field, ARAM can be visualized as two overlapping ART modules sharing a single category field. The category field F_2 receives bottom-up activities from the two feature fields F_1^a and F_1^b. Thus, an F_2 category node learns to encode a complete pattern pair. By synchronizing the unsupervised categorization of two pattern sets, ARAM learns supervised mapping between the pattern sets. Code stabilization is ensured by restricting encoding to states where resonances are reached in both the ART_a and ART_b modules. Due to the code stabilization mechanism, fast learning in a real-time environment is feasible. As the network structure and operations are symmetrical, associative recall can be performed in both directions.

ARAM performs two slightly different memory tasks, namely pattern classification and heteroassociative recall. Pattern classification involves the learning of many-to-one mappings from a set of patterns to pattern classes. Although ARAM has a simpler architecture than ARTMAP, it exhibits the same dynamics as ARTMAP under certain parameter settings (Tan, 1992). Simulations on a well-known iris plant data set compare the ARAM performance with that of counterpropagation network. Experiments on a sonar return recognition problem show that ARAM produces better generalization than back-propagation network at the cost of creating more category nodes.

Heteroassociative recall, which is a general form of pattern classification, involves the learning of associative mappings between two sets of possibly distributed patterns. With maximal vigilance settings, ARAM dynamically allocates a category node for encoding each distinct pattern pair. It thus guarantees perfect storage and recall of an *arbitrary* number of *arbitrary* pattern pairs, a property that is highly desirable for Bidirec-

Acknowledgements: This article is based on a chapter of a doctoral dissertation submitted to Boston University. I wish to thank my dissertation committee members: Gail A. Carpenter, Stephen Grossberg, and Michael Cohen. Thanks also go to the two anonymous referees for giving many useful comments to a previous version of the manuscript. I also thank Loo-Nin Teow for his help in refining the article.

Request for reprints should be sent to Ah-Hwee Tan, Institute of Systems Science, National University of Singapore, Kent Ridge, Singapore 0511.

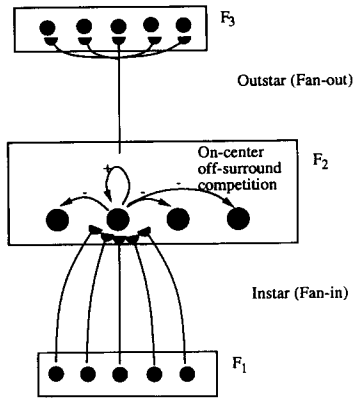

FIGURE 1. Computational map (Grossberg, 1976a, b; Carpenter, 1989). F_1 and F_2 form a competitive learning module. F_2 and F_3 form an outstar learning system.

tional Associative Memory (BAM) models (Kosko, 1987, 1988, 1992; Lin & Chang, 1993; Wang, Cruz, & Mulligan, 1990, 1991; Zhuang, Huang, & Chen, 1993). Although ARAM uses high vigilance for pattern storage, the vigilance parameter values can be set as low as possible during recall to allow maximal generalization. Compared with BAM, ARAM is more effective for the storage and recall of small sets of pattern pairs with arbitrarily high correlation within each pattern set. ARAM is evaluated against BAM in two pattern storage and recall tasks. Besides the advantages of fast learning, guaranteed perfect storage, and full memory capacity, simulation results indicate that ARAM produces a much stronger noise immunity than BAM using three different learning rules.

The remaining sections of this paper are organized as follows. Section 2 motivates the design of ARAM architecture. Section 3 discusses in more details the ARAM properties and design principles. Section 4 presents an analog ARAM model called fuzzy ARAM that utilizes fuzzy ART operations. Section 5 reports experimental results of fuzzy ARAM in pattern classification and heteroassociative recall. Concluding remarks and future research directions are given in the final section.

2. FROM COMPUTATIONAL MAP TO ARAM

The design of ARAM architecture is motivated by that of computational map (Grossberg, 1976a, b, 1987; Carpenter, 1989). The core of a computational map is an instar–outstar system (Figure 1). The input field F_1 and the category field F_2 form a competitive learning system that comprises an instar adaptive filter ($F_1 \rightarrow F_2$) and a shunting competitive network (F_2). The category field F_2 and the output field F_3 form a fan-out adaptive filter that performs outstar learning of output patterns. A variant of computational map is counterpropagation network (Hecht–Nielsen, 1987, 1988), which, in its full form, is a bidirectional system. Computational map has generated much interest because it provides a powerful mechanism to associate pattern sets of different dimensions and probability distributions. However, computational map suffers from a code instability problem. It is the analysis of the instability problem that leads to the introduction of Adaptive Resonance Theory (ART) (Carpenter & Grossberg, 1987a,b).

One way of stabilizing codes in computational map is to replace the instar network by an ART module to form an ART–outstar system, which self-organizes input pattern categories and learns output patterns through outstar sampling (Grossberg, 1987). However, the outstar network may also suffer from the same instability problem, given the simplest reason that the input patterns coded into a category may be associated with very different output patterns. By using the same category node to sample diverse output patterns, fast learning is not possible as the $F_2 \rightarrow F_3$ weights will oscillate. Slow learning will make the weights more stable, but the resulting template can hardly be representative of many dissimilar patterns. The above consideration points to a need for another matching mechanism at the output field. It is thus natural to extend the ART–outstar system by replacing the outstar network by another (inverted) ART module. The resultant system is an Adaptive Resonance Associative Map (ARAM) that can be visualized as two ART modules sharing a single category field (Figure 2).

An interesting analogy can be drawn between ARAM and counterpropagation network (CPN) (Figure 3). Without the top-down priming and reset mechanism, ARAM reduces to a compact counterpropagation network. F_1^a of ARAM subsumes layer 1 and layer 4 of CPN whereas F_1^b of ARAM subsumes layer 2 and layer 5 of CPN. The category field F_2 of ARAM corresponds to layer 3 in CPN. However, whereas the

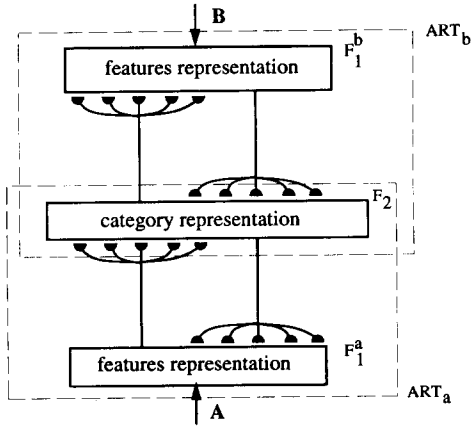

FIGURE 2. A schematic Adaptive Resonance Associative Map (ARAM). It consists of two ART modules sharing a single category field F_2.

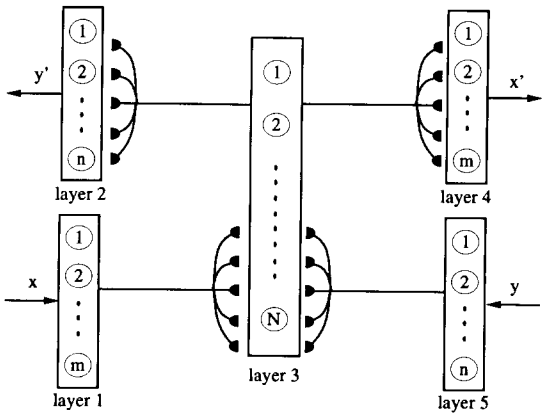

FIGURE 3. Counterpropagation network (Hecht-Nielsen, 1987, 1988).

number of layer 3 nodes in CPN is fixed, the category nodes of ARAM are allotted one by one as novel patterns are encountered. The strengths of ARAM over CPN include fast learning, self-stabilizing, and need-based allocation of category nodes.

3. ARAM DESIGN PRINCIPLES

The design principles and properties of ARAM are outlined below.

(A) Heteroassociative versus autoassociative memory. As category learning systems, ART networks are designed for autoassociative recall. ARAM extends ART to learn heteroassociative mappings between pattern sets of generally different dimensions and probability distributions.

(B) Recognition categories for encoding pattern pairs. ARAM learns pattern associations across two different fields by using a single category node to encode a pair of associated patterns. New category nodes are recruited automatically when novel patterns are encountered. As many category nodes as required can be created until the system capacity is fully utilized.

(C) Fast stable learning. ARAM learns a pair of patterns only if the matches between the patterns and the selected weight templates satisfy the vigilance criteria in their respective modules. Moreover, in fuzzy ARAM (Section 4), weight templates can only decrease but not increase. With the code stabilizing mechanism, fast and stable learning in a real-time environment is made feasible.

(D) Continuous learning and performance. ARAM learns as it performs. A single set of system equations is used for both the learning and performance phases. The model exhibits different functional behaviors in response to different pattern presentation paradigms (Figure 4).

(E) Associative recall in both directions. The network structure and operations of ARAM are symmetrical. No distinction is drawn between the input and output patterns. Presenting part of an encoded pattern pair results in the readout of the complete pattern pair (Figure 4).

(F) Many-to-one and one-to-many mappings. Real-world applications sometimes require both many-to-one and one-to-many mappings. For example, in a medical diagnostic system, many different sets of symptoms can map to a disease and a set of symptoms may appear for more than one disease. Essentially, learning many-to-one and one-to-many mappings involves the task of encoding the following set of pattern pairs:

$$(\mathbf{A}_1, \mathbf{B}), (\mathbf{A}_2, \mathbf{B}), \ldots, (\mathbf{A}_m, \mathbf{B}),$$

$$(\mathbf{A}, \mathbf{B}_1), (\mathbf{A}, \mathbf{B}_1), \ldots, (\mathbf{A}, \mathbf{B}_n).$$

By the pattern pair encoding scheme (Property B), ARAM is able to encode the above set of pattern associations and thus allows both many-to-one and one-to-many mappings.

(G) Maximal generalization under external demands. Depending on the constraints and demands of the problem domain, the vigilance parameters can be set as low as possible to allow maximal generalization. Applications that require high accuracy and/or concern critical consequences can be assigned a higher vigilance level. Otherwise, the vigilance level can be relaxed. The use of two separate vigilance parameters allows the system to respond differently to constraints imposed on two pattern populations.

4. FUZZY ARAM

In an ARAM network (Figure 5), the unit for recruiting an F_2 category node is a complete pattern pair. Given a pair of patterns, the category field F_2 selects a winner that receives the largest overall input from the feature

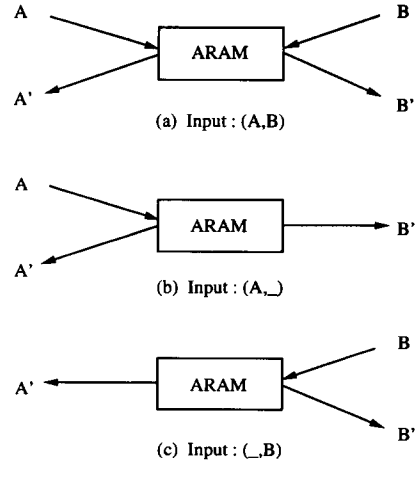

FIGURE 4. Operational dynamics of ARAM. The model learns pattern associations as pattern pairs are presented. When an incomplete or noisy pattern pair (A, B) is presented, the model recovers the complete pattern pair (A', B').

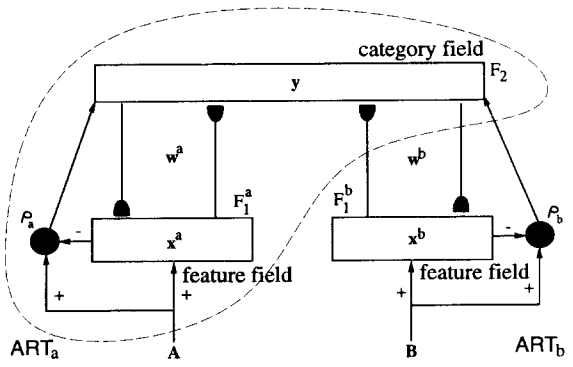

FIGURE 5. Adaptive Resonance Associative Map architecture.

fields F_1^a and F_1^b. The winning node selected in F_2 then triggers a top-down priming on F_1^a and F_1^b, monitored by separate reset mechanisms. Code stabilization is ensured by restricting encoding to states where resonances are reached in both modules.

The ART modules used in ARAM can be ART 1 (Carpenter & Grossberg, 1987a), which categorizes binary patterns, or analog ART modules such as ART 2 (Carpenter & Grossberg, 1987b), ART 2-A (Carpenter, Grossberg, & Rosen, 1991a), and fuzzy ART (Carpenter, Grossberg, & Rosen, 1991b), which categorize both binary and analog patterns. The fuzzy ARAM model, which is composed of two overlapping fuzzy ART modules (Figure 5), is described below.

Input vectors. Normalization of fuzzy ART inputs prevents category proliferation. The F_1^a and F_1^b input vectors are normalized by complement coding that preserves amplitude information. Complement coding represents both the on-response and the off-response to an input vector $\mathbf{a}$. The complement coded F_1^a input vector $\mathbf{A}$ is a $2M$-dimensional vector

$$\mathbf{A} = (\mathbf{a}, \mathbf{a}^c) \equiv (a_1, \ldots, a_M, a_1^c, \ldots, a_M^c) \quad (1)$$

where $a_i^c \equiv 1 - a_i$. Similarly, the complement coded F_1^b input vector $\mathbf{B}$ is a $2N$-dimensional vector

$$\mathbf{B} = (\mathbf{b}, \mathbf{b}^c) \equiv (b_1, \ldots, b_N, b_1^c, \ldots, b_N^c) \quad (2)$$

where $b_i^c \equiv 1 - b_i$.

Activity vectors. Let $\mathbf{x}^a$ and $\mathbf{x}^b$ denote the F_1^a and F_1^b activity vectors, respectively. Let $\mathbf{y}$ denote the F_2 activity vector.

Weight vectors. Each F_2 category node j is associated with two adaptive weight templates $\mathbf{w}_j^a$ and $\mathbf{w}_j^b$. Initially, all category nodes are uncommitted and all weights equal ones. After a category node is selected for encoding, it becomes *committed.*

Parameters. Fuzzy ARAM dynamics are determined by the choice parameters $\alpha_a > 0$ and $\alpha_b > 0$; the learning rates $\beta_a \in [0, 1]$ and $\beta_b \in [0, 1]$; the vigilance parameters $\rho_a \in [0, 1]$ and $\rho_b \in [0, 1]$; and a contribution parameter $\gamma \in [0, 1]$.

Category choice. Given the F_1^a and F_1^b input vectors $\mathbf{A}$ and $\mathbf{B}$, for each F_2 node j, the choice function T_j is defined by

$$T_j = \gamma \frac{|\mathbf{A} \wedge \mathbf{w}_j^a|}{\alpha_a + |\mathbf{w}_j^a|} + (1 - \gamma) \frac{|\mathbf{B} \wedge \mathbf{w}_j^b|}{\alpha_b + |\mathbf{w}_j^b|}, \quad (3)$$

where the fuzzy AND operation $\wedge$ is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i), \quad (4)$$

and where the norm $|\cdot|$ is defined by

$$|\mathbf{p}| \equiv \sum_i p_i \quad (5)$$

for vectors $\mathbf{p}$ and $\mathbf{q}$.

The system is said to make a choice when at most one F_2 node can become active. The choice is indexed at J where

$$T_J = \max\{T_j : \text{for all } F_2 \text{ node } j\}. \quad (6)$$

When a category choice is made at node J, $y_J = 1$; and $y_j = 0$ for all $j \neq J$. In a choice system, the F_1^a and F_1^b activity vectors $\mathbf{x}^a$ and $\mathbf{x}^b$ obey the equations

$$\mathbf{x}^a = \begin{cases} \mathbf{A} & \text{if } F_2 \text{ is inactive} \\ \mathbf{A} \wedge \mathbf{w}_J^a & \text{if the } J\text{th } F_2 \text{ node is chosen} \end{cases} \quad (7)$$

and

$$\mathbf{x}^b = \begin{cases} \mathbf{B} & \text{if } F_2 \text{ is inactive} \\ \mathbf{B} \wedge \mathbf{w}_J^b & \text{if the } J\text{th } F_2 \text{ node is chosen} \end{cases} \quad (8)$$

respectively.

Resonance or reset. Resonance occurs if the *match functions,* m_J^a and m_J^b, meet the vigilance criteria in their respective modules:

$$m_J^a = \frac{|\mathbf{A} \wedge \mathbf{w}_J^a|}{|\mathbf{A}|} \geq \rho_a \quad \text{and} \quad m_J^b = \frac{|\mathbf{B} \wedge \mathbf{w}_J^b|}{|\mathbf{B}|} \geq \rho_b. \quad (9)$$

Learning then ensues, as defined below. If any of the vigilance constraints is violated, mismatch reset occurs in which the value of the choice function T_J is set to 0 for the duration of the input presentation. The search process repeats to select another new index J until resonance is achieved.

Learning. Once the search ends, the weight vectors $\mathbf{w}_J^a$ and $\mathbf{w}_J^b$ are updated according to the equations

$$\mathbf{w}_J^{a(\text{new})} = (1 - \beta_a)\mathbf{w}_J^{a(\text{old})} + \beta_a(\mathbf{A} \wedge \mathbf{w}_J^{a(\text{old})}) \quad (10)$$

and

$$\mathbf{w}_J^{b(\text{new})} = (1 - \beta_b)\mathbf{w}_J^{b(\text{old})} + \beta_b(\mathbf{B} \wedge \mathbf{w}_J^{b(\text{old})}), \quad (11)$$

respectively. For efficient coding of noisy input sets, it is useful to set $\beta_a = \beta_b = 1$ when J is an uncommitted node, and then take $\beta_a < 1$ and $\beta_b < 1$ after the category node is committed. *Fast learning* corresponds to setting $\beta_a = \beta_b = 1$ for committed nodes.

Match tracking. Match tracking rule as used in the ARTMAP search and prediction process is useful in

maximizing code compression. At the start of each input presentation, the vigilance parameter ρ_a equals a baseline vigilance $\overline{\rho_a}$. If a reset occurs in the category field F_2, ρ_a is increased until it is slightly larger than the match function m_J^a. The search process then selects another F_2 node J under the revised vigilance criterion. With the match tracking rule and setting the contribution parameter $\gamma = 1$, ARAM emulates the search and test dynamics of ARTMAP.

5. EXPERIMENTAL RESULTS

5.1. Pattern Classification

5.1.1. *Iris Plant Classification.* Fuzzy ARAM is first evaluated on an iris plant data set (Fisher, 1936) obtained from the UCI machine learning data base directory (Murphy & Aha, 1992). The iris plant data set consists of three classes: setosa, versicolour, and virginica, of 50 instances each, with four numeric attributes: sepal length, sepal width, petal length, and petal width. This well-known and relatively simple data set is used to study the behavior of fuzzy ARAM under different parameter settings.

For each simulation, a 75-case training set and a 75-case test set are selected randomly. The training set is presented repeatedly until no reset occurs. The simulation results averaged over 100 runs are summarized in Table 1. Increasing $\overline{\rho_a}$ improves the predictive accuracy at the cost of requiring more training epochs and category nodes. Increasing α_a, on the other hand, provides better accuracy with only marginal increase in the number of category nodes. In all simulations, there is no misclassification for class 1 plant. The last two rows in Table 1 illustrate the performance of ARTMAP configurations. With the ARTMAP match tracking process, the number of category nodes is significantly reduced while the system maintains roughly the same level of performance. However, as match tracking introduces a series of search, test, and reset cycles, the learning time is slightly longer.

Simulations are also conducted to compare the performance of counterpropagation network (Hecht-Nielsen, 1987, 1988) with that of ARAM. A forward version of counterpropagation network (CPN) consisting of only layer 1, 2, and 3 (Figure 3) developed inhouse is used. Layer 3 of the CPN is also known as the Kohonen layer as it performs Kohonen's self-organization. Layer 2 is the output layer that performs Grossberg's outstar learning. The main parameters of CPN are the number of units in the Kohonen layer and the learning rates $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ of the Kohonen and Grossberg layers, respectively. Whereas β is fixed during learning, α decreases with time, as determined by the equation

$$\alpha = \frac{\alpha_0}{1 + 0.1t} \tag{12}$$

where $\alpha_0 \in [0, 1]$ is the initial learning rate and t is the number of training iterations.

Using 10 Kohonen units, empirical experiments are first conducted with different learning rates α_0 and β. The best test accuracy is obtained with $\alpha_0 = 1.0$ and $\beta = 0.1$. The parameter values are then used in the subsequent simulations in which the number of Kohonen units varies from 10 to 50. As CPN does not guarantee convergence on the training data, learning in all simulations is stopped after 300 iterations at which no or very little weight changes occur.

Comparing performance, the best accuracy obtained by CPN is still slightly inferior to that of ARAM (Table 2). The optimal number of Kohonen units of CPN for the iris plant problem seems to be around 10–20. It is about the same as that of ARAM without match tracking, but is more than that of ARAM with ARTMAP configuration. Increasing the number of Kohonen units beyond 20 results in degradation of test set generalization. This could be due to the overfitting effect on the training set.

5.1.2. *Sonar Return Recognition.* The sonar return data set (Gorman & Sejnowski, 1988) contains 208 in-

TABLE 1
Performance of Fuzzy ARAM in Classifying Iris Plants

Parameters					Resource Utilized		Predictive Accuracy	
α_a	β_a	γ	$\overline{\rho_a}$	MT	No. Epochs	No. Nodes	Correct (%)	SD
0.01	0.9	0.5	0.0	No	1–2	3	93.5	2.1
0.01	0.9	0.5	0.7	No	3–5	11–34	94.2	2.3
0.01	0.9	0.5	0.9	No	2–5	31–53	94.7	2.3
0.1	0.9	0.5	0.7	No	2–5	10–29	94.5	2.3
0.2	0.9	0.5	0.7	No	2–5	9–31	94.9	1.9
0.1	0.9	1.0	0.0	Yes	2–6	3–10	94.3	2.3
0.2	0.9	1.0	0.0	Yes	2–7	3–11	94.7	2.1

MT indicates whether match tracking is employed. SD stands for standard deviation. The ART_b parameter values are $\alpha_b = 0.01$, $\beta_b = 1$, and $\rho_b = 1$.

TABLE 2
Performance of Counterpropagation Network in Classifying Iris Plants

Parameters		Resource Utilized		Predictive Accuracy	
α_0	β	No. Epochs	No. Nodes	Correct (%)	SD
0.6	0.1	300	10	93.1	2.8
0.8	0.1	300	10	93.5	2.7
1.0	0.1	300	10	94.2	2.4
1.0	0.1	300	20	94.3	2.9
1.0	0.1	300	30	93.5	2.4
1.0	0.1	300	50	93.5	2.5

stances with 60 real-valued features, of which 97 instances are returns from roughly cylindrical rocks and 111 instances are returns from metal cylinders. This is a relatively difficult domain as the number of training examples is small and the data contain noises. In Gorman and Sejnowski's aspect angle-dependent experiments, the data set was divided into a 104-element training set and a 104-element test set, with balanced representation in each aspect angle. After learning the training set, perceptron classifies only 73% of the test set patterns correctly (Table 3). Back-propagation network with 12 hidden units obtains a test set accuracy of 90.4%. Increasing the number of hidden units to 24, however, degrades the performance.

For comparison of performance, the same training and test sets are used here. The K-nearest neighbor (KNN) system that stores all training patterns, is also evaluated on the sonar return data set. KNN performs best with $K = 1$, producing a test set performance of 91.6%, that is, slightly better than that of back-propagation network. Each ARAM simulation is repeated for 20 runs. In each run, the training patterns are presented in a random order. Fuzzy ARAM with ARTMAP configuration ($\gamma = 1$ and match tracking) performs best with slow learning ($\beta = 0.1$). The same level of accuracy as KNN is obtained with only 22–42 category nodes. The number of learning iterations ranges from 8 to 34, about 10 times less than that of back-propagation networks. By raising ART_a vigilance $\overline{\rho_a}$ to 0.9, fuzzy ARAM with fast learning ($\beta_a = 1.0$) converges in merely two iterations. Also, a better prediction rate is obtained at 92.9%. The number of category nodes, however, increases to around 70, but is still smaller than that of KNN. Using the *voting strategy* (as used in fuzzy ARTMAP), an ARAM is trained in several simulation runs using different orderings of the training set. For each test case, predictions made in multiple runs are averaged to produce a final prediction. Voting across five simulations improves the accuracy to 94.2%.

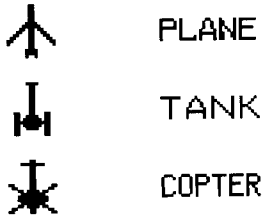

FIGURE 6. The three pattern pairs devised by Wang et al. (1990). The dimensions of A-space that represents pictures and B-space that represents written names are 16 × 18 and 35 × 8, respectively.

5.2. Associative Recall

5.2.1. *Three-Pattern Storage Experiment.* Zhuang, Huang, and Chen (1993) used a sample pattern set originally devised by Wang, Cruz, and Mulligan (1990) to evaluate two new learning rules for BAM, known as the Bidirectional Perceptron Stability Learning (BPSL) rule and the Bidirectional Perceptron Hamming-Stability Learning (BPHSL) rule. The sample pattern set contains a picture and a written name, each of a plane, a tank, and a helicopter (Figure 6). The pictures are each represented using a space of 288 (16 × 18) binary bits, hereafter called A-space. The words are each represented using a space of 280 (35 × 8) binary bits, hereafter called B-space. Based on this pattern set, Zhuang et al. compared their BPSL and BPHSL rules with the Kosko's formula of computing

TABLE 3
Performance of Perceptron, Back-Propagation Network, KNN, and Fuzzy ARAM (α_a, β_a, γ, $\overline{\rho_a}$, Match-Track) in Classifying Sonar Returns (Aspect Angle-Dependent Case)

Model	No. Epochs	No. Hidden Nodes	Accuracy (%)	SD
Perceptron	300	0	73.1	4.8
Back-propagation network	300	12	90.4	1.8
Back-propagation network	300	24	89.2	1.4
KNN ($K = 1$)	1	104	91.6	0.0
Fuzzy ARAM (12, 0.1, 1.0, 0.0, y)	8–34	22–42	91.6	2.7
Fuzzy ARAM (12, 1.0, 0.5, 0.9, y)	2	68–72	92.9	0.9
Fuzzy ARAM (12, 1.0, 0.5, 0.9, y)	(voting across five simulations)		94.2	0.9

The ARAM ART_b parameter values are $\alpha_b = 0.01$, $\beta_b = 1$, and $\rho_b = 1$.

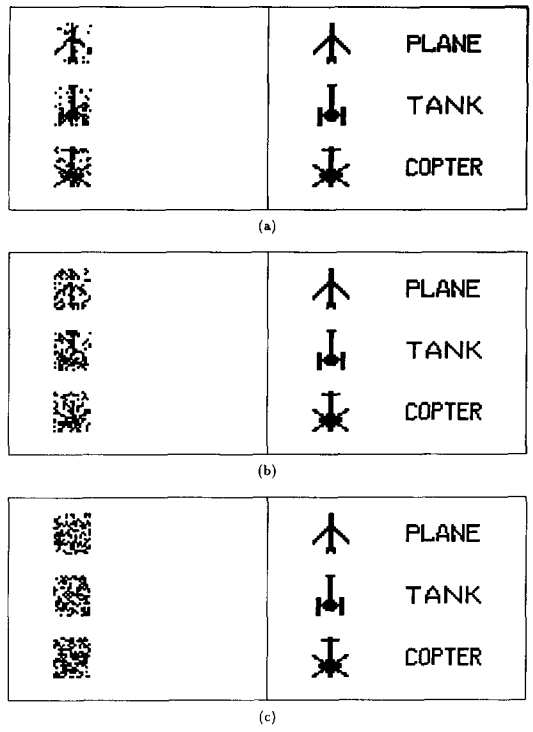

FIGURE 7. ARAM recall from A-space patterns in the three-pattern set with noise intensities of (a) 10%, (b) 25%, and (c) 40%. The left-hand columns contain the initial noisy patterns. The right-hand columns contain the final recovered patterns.

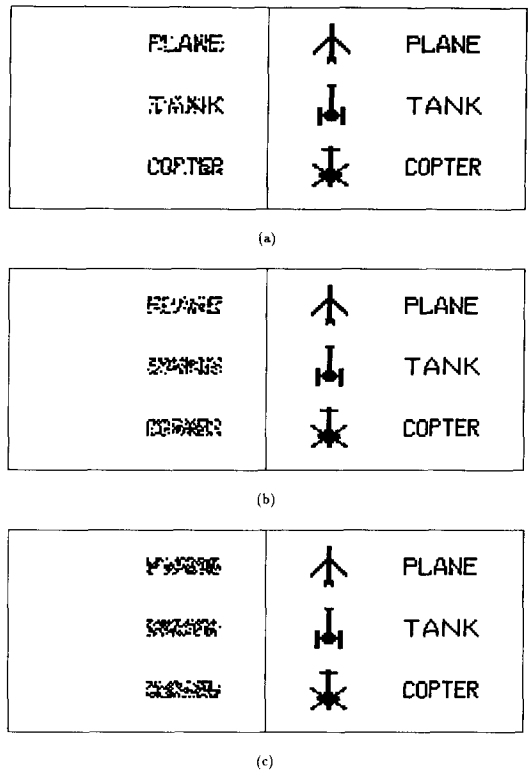

FIGURE 8. ARAM recall from B-space patterns in the three-pattern set with noise intensities of (a) 10%, (b) 25%, and (c) 40%. The left-hand columns contain the initial noisy patterns. The right-hand columns contain the final recovered patterns.

BAM weight matrix (Kosko, 1987, 1988, 1992), hereafter called the Kosko's rule. Whereas both the BPSL and BPHSL rules store all the three pattern pairs stably, one of the three pattern pairs could not be stored using the Kosko's rule. In noise immunity tests, noisy patterns are generated by negating each bit of the original patterns with a probability measured by a noise intensity. BAM networks trained using the three different learning rules were tasked to recall the original patterns from the A-space and B-space noisy patterns. Simulation results of Zhuang et al. indicate that the BPSL and BPHSL rules, which perform equally well in this problem, produce much better recall accuracy than the Kosko's rule.

In ARAM associative recall experiments, the input patterns are not complement coded. Fast learning is used with $\beta_a = \beta_b = 1$. γ is fixed at 0.5. Using $\rho_a = \rho_b = 1$ during encoding, the three pattern pairs are stable after two pattern presentations, in contrast to the many learning iterations required by the BPSL and BPHSL rules. During recall, the vigilance parameters ρ_a and ρ_b are each set to 0. Using a fixed noise intensity of 25%, empirical simulations are conducted with different choice parameter values α_a and α_b. For each pair of the choice parameter values, 100 recall simulations are performed. The simulation results indicate that the ARAM performance is highly immune to the choice parameter values. In fact, 100% recall accuracy is consistently obtained with α_a and α_b ranging from 0.001 to 400 due to the simplicity of the problem. The choice parameters are thus arbitrarily set to 0.001 in the subsequent simulations in which the noise intensity is varied from 0% to 50%.

For each noise intensity, ARAM simulation is repeated for 1000 times. Figure 7 illustrates ARAM recall from A-space patterns with noise intensities of 10%, 25%, and 40%. Figure 8 illustrates ARAM recall from B-space patterns with noise intensities of 10%, 25%,

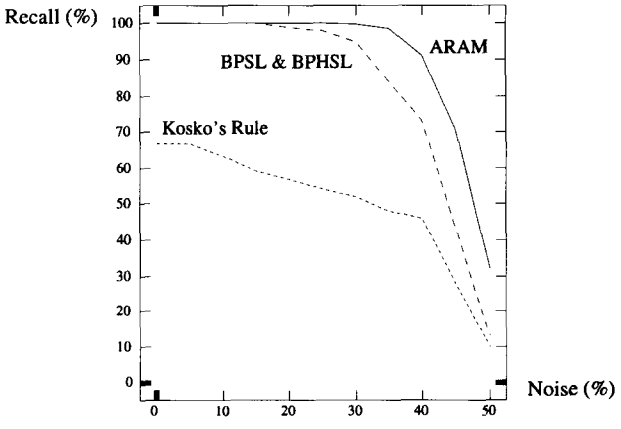

FIGURE 9. Noise immunity of ARAM in A-space for the three-pattern set comparing with BAM using the Kosko's, BPSL, and BPHSL rules.

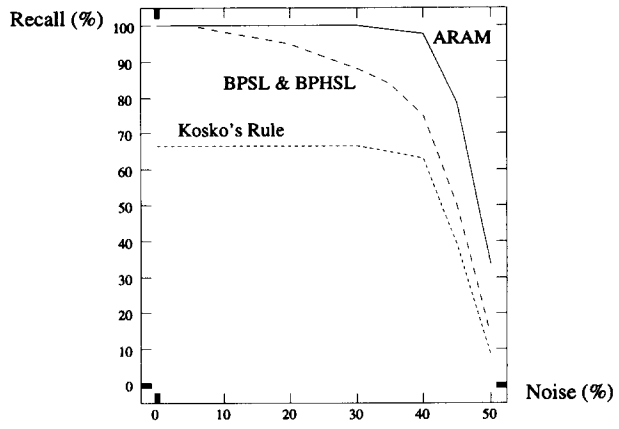

FIGURE 10. Noise immunity of ARAM in B-space for the three-pattern set comparing with BAM using the Kosko's, BPSL, and BPHSL rules.

and 40%. Figures 9 and 10 compare the ARAM noise immunity performance with BAM using the Kosko's, BPSL, and BPHSL rules in A-space and B-space, respectively. Clearly, ARAM has a much stronger noise immunity over all the three learning rules of BAM. In fact, ARAM recall accuracy is maintained at well above 90%, even with a noise intensity as high as 40%.

5.2.2. *Extended Pattern Storage Experiment.* Note that the first memory task requires storage of only three pattern pairs but uses huge pattern dimensions. Zhuang et al. (1993) devised a more challenging problem in which the number of bits was reduced to 168 (12 × 14) in A-space and 55 (11 × 5) in B-space. Moreover, seven more pattern pairs were added to compose a total of 10 desired pattern pairs (Figure 11). The reduction in the number of bits and the increase in the number of desired patterns both make it more difficult to distinguish among the stored patterns. Zhuang et al. reported that the Kosko's rule failed to store any one of the desired pattern pairs as a stable state. The BPSL and BPHSL rules, on the other hand, stored all the 10 desired pattern pairs. Additionally, the BPHSL rule showed a stronger noise immunity than the BPSL rule.

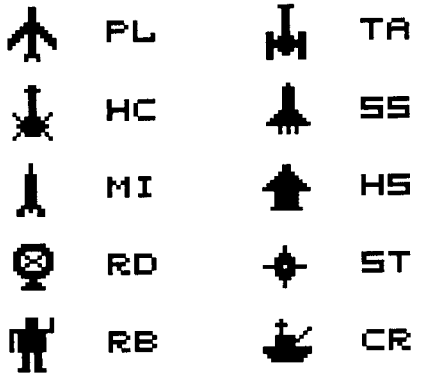

FIGURE 11. The extended pattern set that consists of 10 pattern pairs. The dimensions of A-space representing pictures and B-space representing written codes are 12 × 14 and 11 × 5, respectively.

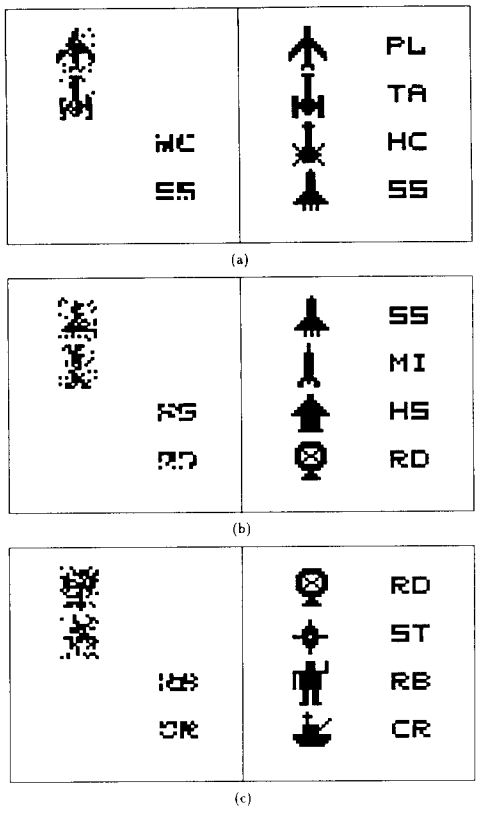

FIGURE 12. ARAM recall of patterns in the extended pattern set with noise intensities of (a) 10%, (b) 20%, and (c) 25%. The left-hand columns contain the initial noisy patterns. The right-hand columns contain the final recovered patterns. In each table, two recall tasks each are illustrated for A-space and B-space noisy patterns.

In ARAM simulations, the 10 pattern pairs are stable after two training iterations. Using a fixed noise intensity of 25%, simulations are again conducted using different choice parameter values α_a and α_b. For each pair

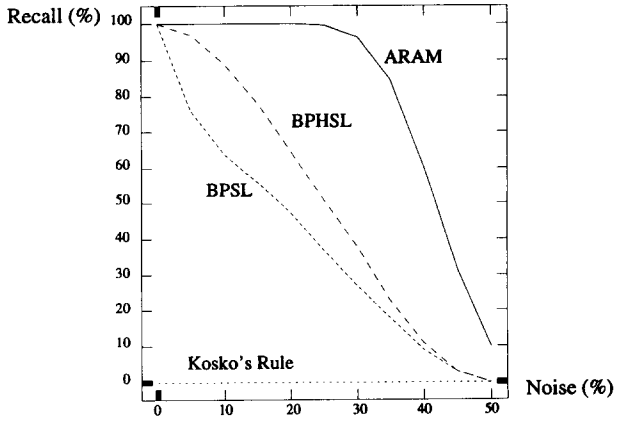

FIGURE 13. Noise immunity of ARAM in A-space for the extended pattern set comparing with BAM using the Kosko's, BPSL, and BPHSL rules.

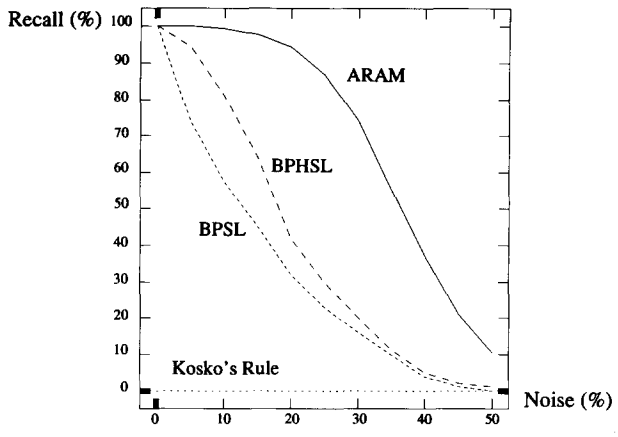

FIGURE 14. Noise immunity of ARAM in B-space for the extended pattern set comparing with BAM using the Kosko's, BPSL, and BPHSL rules.

of the choice parameter values, 100 recall simulations were performed. It is found that with small choice parameter values, a pattern can be easily confused with its *subset* pattern. For example, with a few bits off, a noisy HS A-space pattern can be misclassified as a ST A-space pattern. Increasing choice parameter values gives an advantage to patterns that have larger norms (number of positive bits) and thus makes the patterns more distinguishable. The simulations find that α_a = 25 and α_b = 15 work best for this problem and are thus used in the subsequent simulations.

For each noise intensity ranging from 0% to 50%, 1000 ARAM recall experiments are conducted. Figure 12 illustrates ARAM recall from noisy A-space and B-space patterns. The noise immunity test results of ARAM compared with BAM trained using the Kosko's, BPSL, and BPHSL rules in A-space and B-space are summarized in Figures 13 and 14, respectively. Again, ARAM consistently outperforms all the three learning rules of BAM across the entire range of noise intensities.

5.2.3. *Comparing ARAM With BAM.* ARAM adopts an approach fundamentally different from BAM to learning heteroassociative mappings. Whereas ARAM automatically guarantees storage and recall of an arbitrary number of arbitrary pattern pairs using maximal vigilance settings, BAM utilizes perceptron learning rules to ensure that the desired patterns are stably stored.

ARAM and BAM are comparable in network size. Let M and N denote the dimensions of A-space and B-space respectively, and S denote the number of desired pattern pairs. The weight matrix of BAM contains $M*N$ weight values and ARAM uses a total of $S*(M + N)$ weights. If S is small, ARAM is more efficient than BAM. For the two pattern storage tasks evaluated, ARAM utilizes a much smaller number of weights than BAM (Table 4). If S is large, BAM is more economical. However, as the largest possible S in BAM is typically smaller than M or N (explained below), ARAM uses at most twice the number of weights of BAM in the worst case.

The number of patterns that can be successfully stored in BAM depends highly on the pattern dimensions M and N. Given a fixed network size, the percentage of successful storage in BAM drops dramatically when S becomes greater than M or N. When a larger number of patterns needs to be stored, one has to switch to another BAM model with larger space dimensions and retrain all the patterns. In ARAM, the number of patterns stored does not affect the accuracy of storage and recall. By expanding network architecture dynamically, ARAM ensures perfect storage of an arbitrary number of patterns and has full memory capacity. Moreover, as strong contrast enhancement in the category field always results in a choice, ARAM does not have the spurious memory problem of BAM.

6. CONCLUSIONS AND EXTENSIONS

A neural network architecture termed ARAM has been described. As a direct generalization of ART systems, ARAM inherits ART properties including self-organizing, self-stabilizing, fast yet stable learning, and does not distinguish between learning and performance phases. In addition, ARAM performs supervised learning and bidirectional associative recall.

Whereas BAM model is limited to learning heteroassociative mappings between two sets of patterns, ARAM architecture can be readily generalized to K-way ARAM that learns pattern associations across multiple pattern channels (Tan, 1994). Whereas ARAM consists of two input representation fields sharing a category field, K-way ARAM comprises K input representation fields and a category field. Two-way ARAM is essentially ARAM. With K = 1, the system reduces

TABLE 4
Comparison of BAM and ARAM System Size in Terms of the Number of Weights

Pattern Set	A-Space Dimension (M)	B-Space Dimension (N)	No. Patterns Encoded (S)	BAM Size	ARAM Size
3-Pattern set	288	280	3	80640	1704
10-Pattern set	168	55	10	9240	2230

to ART. K-way ARAM has been used as a building block of a higher-level cognitive architecture termed Concept Hierarchy Memory Model (CHMM) that is developed for conceptual knowledge representation and common sense reasoning (Soon & Tan, 1993a, b; Tan, 1994). Based on K-way ARAM that supports fast and stable associative learning, CHMM provides a systematic way for creating new concepts and organizing a concept hierarchy.

REFERENCES

Carpenter, G. A. (1989). Neural network models for pattern recognition and associative memory. *Neural Networks,* **2**(4), 243–257.

Carpenter, G. A., & Grossberg, S. (1987a). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing,* **37,** 54–115.

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

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., & Rosen, D. B. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks,* **3,** 698–713.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991a). ART 2-A: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks,* **4,** 493–504.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991b). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks,* **4,** 759–771.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annual Eugenics,* **7,** 179–188. Also in *Contributions to mathematical statistics.* John Wiley, NY, 1950.

Gorman, R. P., & Sejnowski, T. J. (1988). Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks,* **1,** 75–89.

Grossberg, S. (1976a). Adaptive pattern recognition and universal recoding, I: Parallel development and coding of neural feature detectors. *Biological Cybernetics,* **23,** 121–134.

Grossberg, S. (1976b). Adaptive pattern recognition and universal recoding, II: Feedback, expectation, olfaction, and illusion. *Biological Cybernetics,* **23,** 187–202.

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

Hecht-Nielsen, R. (1987). Counterpropagation networks. *Applied Optics,* **26,** 4979–4984.

Hecht-Nielsen, R. (1988). Applications of counterpropagation networks. *Neural Networks,* **1,** 131–139.

Kosko, B. (1987). Adaptive bidirectional associative memories. *Applied Optics,* **26**(23), 4947–4960.

Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on System, Man, and Cybernetics,* **18,** 49–60.

Kosko, B. (1992). *Neural networks and fuzzy systems.* Englewood Cliffs, NJ: Prentice-Hall.

Lin, J. K., & Chang, J. Y. (1993). The perceptron training rule for bidirectional associative memory. In *Proceedings, World congress on neural networks, Portland, OR* (Vol. II, pp. 249–255). Hillsdale, NJ: Lawrence Erlbaum Associates.

Murphy, P. M., & Aha, D. W. (1992). UCI repository of machine learning databases [machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science.

Soon, H. S., & Tan, A. H. (1993a). Concept hierarchy networks for inheritance systems: Concept formation, property inheritance and conflict resolution. In *Proceedings, 15th Conference of Cognitive Science Society,* Boulder, CO, pp. 941–946.

Soon, H. S., & Tan, A. H. (1993b). A memory model for concept hierarchy representation and commonsense reasoning. In *Proceedings, World Congress on Neural Networks, Portland, OR* (Vol. II, pp. 206–209). Hillsdale, NJ: Lawrence Erlbaum Associates.

Tan, A. H. (1992). Adaptive resonance associative map: A hierarchical ART system for fast stable associative learning. In *Proceedings, IJCNN-92 Baltimore, MD* (Vol. I, pp. 860–865). Piscataway, NJ: IEEE Service Center.

Tan, A. H. (1994). *Synthesizing neural network and symbolic knowledge processing.* Doctoral dissertation, Department of Cognitive and Neural Systems, Boston University.

Wang, Y. F., Cruz, J. B. J., & Mulligan, J. H. J. (1990). Two coding strategies for bidirectional associative memory. *IEEE Transactions on Neural Networks,* **1,** 81–92.

Wang, Y. F., Cruz, J. B. J., & Mulligan, J. H. J. (1991). Guaranteed recall of all training pairs for bidirectional associative memory. *IEEE Transactions on Neural Networks,* **2,** 559–567.

Zhuang, X. H., Huang, Y., & Chen, S. S. (1993). Better learning for bidirectional associative memory. *Neural Networks,* **6,** 1131–1146.