A Bidirectional Heteroassociative Memory for Binary and Grey-Level Patterns

Sylvain Chartier, *Member, IEEE,* and Mounir Boukadoum, *Senior Member, IEEE*

***Abstract*—Typical bidirectional associative memories (BAM) use an offline, one-shot learning rule, have poor memory storage capacity, are sensitive to noise, and are subject to spurious steady states during recall. Recent work on BAM has improved network performance in relation to noisy recall and the number of spurious attractors, but at the cost of an increase in BAM complexity. In all cases, the networks can only recall bipolar stimuli and, thus, are of limited use for grey-level pattern recall. In this paper, we introduce a new bidirectional heteroassociative memory model that uses a simple self-convergent iterative learning rule and a new nonlinear output function. As a result, the model can learn online without being subject to overlearning. Our simulation results show that this new model causes fewer spurious attractors when compared to others popular BAM networks, for a comparable performance in terms of tolerance to noise and storage capacity. In addition, the novel output function enables it to learn and recall grey-level patterns in a bidirectional way.**

***Index Terms*—Associative memories, bidirectional associative memories (BAM), learning, neural networks.**

I. Introduction

THE question of how the brain associates different patterns in such a way that, when a given category is presented, another one is recalled has no final answer yet, but with the development of artificial neural networks, particularly heteroassociative memories, tentative explanations are taking form. In the 1970s, Kohonen [1] was among the first investigators to propose a model capable of heteroassociative learning. The model used a linear learning rule and a linear output function. Because of that, when the input patterns were correlated with each other or noisy, the network could not correctly perform the recall task. This weakness led to the development of nonlinear recurrent autoassociative and interpolative memories (e.g., [2] and [3]) whose dynamics, generated by a nonlinear output feedback function, enable them to exhibit stable fixed-point behavior. As a result, if the desired patterns were learned to correspond to given fixed-point attractors, such networks would correctly recall them even under noise degradation. These properties enabled Kosko [4] to adapt the nonlinear feedback of the Hopfield model [5] to a heteroassociative memory, thereby creating a new class of neural network models, the bidirectional associative memory (BAM).

Manuscript received September 23, 2003; revised September 2, 2005.

S. Chartier was with the Département de Psychologie, Université du Québec à Montréal (UQAM), Montréal, QC H3C 3P8, Canada.

M. Boukadoum is with the Département d'Informatique, Université du Québec à Montréal (UQAM), Montréal, QC H3P 3P8, Canada (e-mail: boukadoum.mounir@uqam.ca).

Digital Object Identifier 10.1109/TNN.2005.863420

BAM learning is accomplished with a simple Hebbian rule

$$\mathbf{W} = \mathbf{Y}\mathbf{X}^T. \tag{1}$$

In this expression, $\mathbf{X}$ and $\mathbf{Y}$ are matrices that represent the sets of bipolar vector pairs to be associated, and $\mathbf{W}$ is the weight matrix. Equation (1) forces the use of a one-shot learning process since Hebbian association is strictly additive. A more natural learning process would make (1) incremental but, then, the weight matrix would grow unbounded with the repetition of the input stimuli during learning. This property may be acceptable for orthogonal patterns, however, it leads to disastrous results when the patterns are correlated. In such a case, the weight matrix will be dominated by its first eigenvalue, and this will result in recalling the same pattern whatever the input. A compromise is to use a one-shot learning rule to limit the domination of the first eigenvalue, and to use a recurrent nonlinear output function to allow the network to filter out the different patterns during recall. Kosko's BAM effectively used a signum output function to recall noisy patterns, despite the fact that the weight matrix developed by using (1) is not optimal. Thus, we typically have in a BAM network

$$\mathbf{y}_{[t+1]} = \operatorname{sgn}(\mathbf{W}\mathbf{x}_{[t]}) \tag{2a}$$

and

$$\mathbf{x}_{[t+1]} = \operatorname{sgn}(\mathbf{W}^T\mathbf{y}_{[t]}) \tag{2b}$$

where sgn is the signum function

$$\operatorname{sgn}(z) = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z = 0 \\ -1, & \text{if } z < 0. \end{cases} \tag{3}$$

By using the weight matrix defined by (1) and the output function defined by (2a) and (2b), the network is able to recall Y from X, and by using the weight matrix transpose, the network is able to recall X from Y. These two processes taken together create a recurrent nonlinear dynamic network with the potential to correctly perform binary association.

However, the learning of the BAM network is accomplished offline and the nonlinear output function of (2a) and (2b) is not used during that stage. Moreover, the network is limited to bipolar/binary input patterns and, as such, cannot learn grey-level patterns. Also, the network develops many spurious attractors and has limited storage capacity [6].

One approach to overcome these weaknesses uses a projection matrix based on least mean squared error minimization [1], [6]. This solution increases the storage capacity and recall performance of the network, but its learning rule, based on an inverse matrix principle, is not a local process. Several sophisticated approaches have also been proposed that modify the learning rule or coding procedure, with the result of both increasing storage capacity and performance, and decreasing the number of spurious states (e.g., [7]–[16]). Among them, some use genetic algorithms [8], [9] or Perceptron learning [10], which depart drastically from the simple Hebbian correlation scheme; others use expanded pattern pairs [7], [8], [12], [15], add a time delay [13], [14], or use interconnections among the units inside each layer [16]. There exist also asymmetric BAMs that address the problem of interconnection symmetry, which limits the BAM performance in storage and recall capacities and restricts its use in inference [17]. In all cases, the improvement of BAM performance comes with an increase in network complexity. For example, [18] and [19] add a hidden layer that increases the network dimension in order to facilitate class separation, and they use different learning rules to update the weights of the hidden and output layers. In [20], the networks behave similarly to a feedforward network by not using a correlation matrix and require extended pattern pairs. As a result, those networks lack internal consistency. Also, all the modified networks still use an offline learning algorithm and can only memorize binary or bipolar patterns. This inability to learn real-valued, fixed point attractors limits the models in both cognitive explanations and engineering applications.

An efficient BAM should have the following properties: First, it should learn online since in most real life applications, a network has rarely access to all the stimuli at the same time; and it should iteratively develop weight connections that converge to a stable local optimal solution by incorporating the nonlinear feedback from the output function. Second, the learning should be based solely on correlation without the need to use a hidden layer, a virtual layer, some *a priori* knowledge about the stimuli or the creation of extended pattern pairs. Third, the learning rule should remain as simple as possible and it should be able to associate patterns of different dimensions as well as of the same dimensions. Finally, the output function, in conjunction with the learning rule, should be chosen so that the weight matrix not only develops stable fixed point attractors for bipolar patterns but also for real-valued patterns. In this paper, a neural network that has all these properties is presented. Moreover, simple simulations reveal that the new model develops less spurious attractors than the other BAM studied while maintaining a competitive performance regarding noise degradation and storage capacity on a simple simulation task.

The remainder of the paper is divided as follows. Section II describes the network architecture and presents some theoretical results. Section III shows simulation results about the network's performance in learning and recalling: 1) bipolar correlated prototypes; and 2) real-valued correlated prototypes. Section IV discusses our results and provides the conclusion of this work. An annex follows the references and develops some theoretical derivations.

II. Architecture

Our network architecture is shown in Fig. 1 where $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$ represent the initial vectors-states, $\mathbf{W}$ and $\mathbf{V}$ are the weight matrices, and t is the current iteration number. We see that the network is composed of two Hopfield-like neural networks interconnected in head-to-tail fashion. Taken together, they allow a recurrent flow of information that is processed bidirectionally. The figure shows also that the vectors composing the pairs to be learned need not be of the same dimensions and that, contrary to typical BAM designs, the weight matrix from one side is not necessarily the transpose of that from the other side.

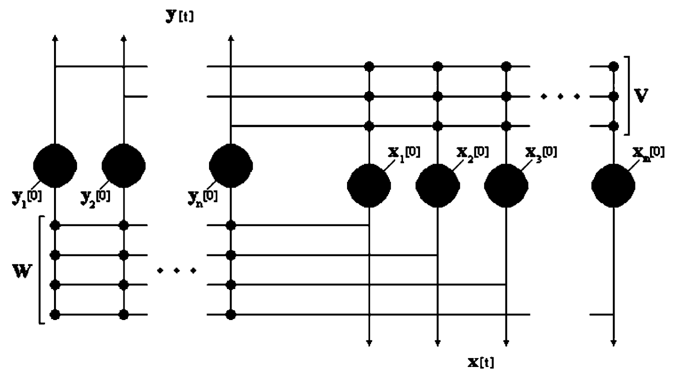


Fig. 1. Architecture of the new heteroassociative model.

A. Output Function

The output function used in our model is based on the classic Verhulst equation [21]. This logistic growth model is described by the following dynamic equation:

$$\frac{dz}{dt} = G(1-z)z \tag{4}$$

where G is a general parameter. Equation (4) has two fixed points [22], $z=0$ and $z=1$. However, only $z=1$ is a stable fixed point and, therefore, (4) has only one attractor. The equation must be modified if we want the existence of two attractors. One way to accomplish this is to change the right term to a cubic map. Then, we obtain

$$\frac{dz}{dt} = G(1-z^2)z. \tag{5}$$

The resulting equation has three fixed points, -1, 0, and 1, of which both the values $z=-1$ and $z=1$ are stable fixed points, thus giving us two attractors.

If we replace the derivative on the left-hand side of (5) by its difference equation approximation, we obtain

$$z_{[t+1]} = \Delta G\left(1 - z_{[t]}^2\right) z_{[t]} + z_{[t]} \tag{6}$$

where Δ is a small constant term; and if we make the changes of variables $\delta = \Delta G$, $\mathbf{y}_{[t+1]} = z_{[t+1]}$, $\mathbf{a}_{[t]} = z_{[t]}$ (or $\mathbf{b}_{[t]} = z_{[t]}$) and rearrange the terms of the previous equation, we obtain the following expressions for the output function:

$$\mathbf{y}_{[t+1]} = (\delta+1)\mathbf{a}_{[t]} - \delta\mathbf{a}_{[t]}^3 \tag{7a}$$

and

$$\mathbf{x}_{[t+1]} = (\delta+1)\mathbf{b}_{[t]} - \delta\mathbf{b}_{[t]}^3. \tag{7b}$$

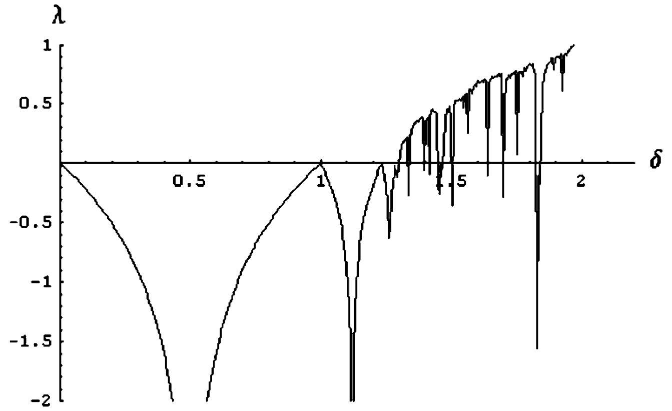


Fig. 2. Lyaponov exponent as a function of the transmission parameter value.

In these equations, $\mathbf{y}_{[t+1]}$ and $\mathbf{x}_{[t+1]}$ represent neural outputs at time $t+1$; $\mathbf{a}_{[t]}$ and $\mathbf{b}_{[t]}$ are the corresponding activation functions at time t ($\mathbf{a}_{[t]} = \mathbf{W}\mathbf{x}_{[t]}$; $\mathbf{b}_{[t]} = \mathbf{V}\mathbf{y}_{[t]}$) and δ is a general output parameter.

The value of δ is crucial to network performance; if it is too high, the network may alternately converge to steady, cyclic or chaotic attractors. By performing a Lyapunov analysis [22], we can find the adequate range of values for δ.

The Lyapunov exponent can be approximated by

$$\lambda \approx \frac{1}{T}\sum_{t=1}^{T} \log\left|\frac{dy_{[t+1]}}{dx_{[t]}}\right| \tag{8}$$

where T is the number of iterations over the network. In our case, the derivative term is obtained from (6), so that λ is given by

$$\lambda \approx \frac{1}{T}\sum_{t=1}^{T} \log|1+\delta-3\delta x_{[t]}^2|. \tag{9}$$

Fig. 2 illustrates the curve of the Lyapunov exponent when $T = 1000$. It shows that the network exhibits a monotonic approach to steady states if the value of δ is between 0 and 0.5. In the case of bipolar inputs, we see from the bifurcation diagram that the steady states are effectively -1 and 1.

Finally, to insure that no output lies outside the interval $[-1, 1]$, a saturating limit was added to bound the output between -1 and 1. The final output function for both network directions is, thus, expressed by the following equations:

$$\forall\, i,\ldots,N, \mathbf{y}_{i[t+1]} = f(\mathbf{a}_{i[t]}) = \begin{cases} 1, & \text{if } \mathbf{a}_{i[t]} > 1 \\ -1, & \text{if } \mathbf{a}_{i[t]} > -1 \\ (\delta+1)\mathbf{a}_{i[t]} - \delta\mathbf{a}_{i[t]}^3, & \text{else} \end{cases} \tag{10a}$$

$$\forall\, i,\ldots,M, \mathbf{x}_{i[t+1]} = f(\mathbf{b}_{i[t]}) = \begin{cases} 1, & \text{if } \mathbf{b}_{i[t]} > 1 \\ -1, & \text{if } \mathbf{b}_{i[t]} > -1 \\ (\delta+1)\mathbf{b}_{i[t]} - \delta\mathbf{b}_{i[t]}^3, & \text{else} \end{cases} \tag{10b}$$

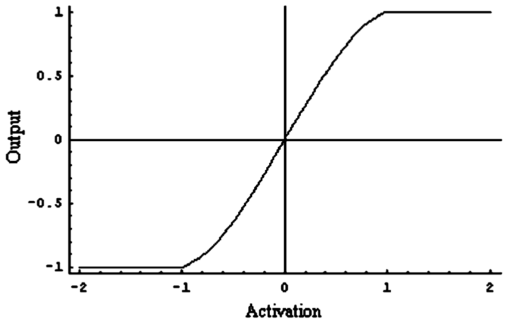


Fig. 3. Curve of the output function for $\delta = 0.4$.

where N and M are the number of units in each layer and i is the index of the respective vector elements during training or recall. Fig. 3 illustrates the shape of the output function for $\delta = 0.4$. It is similar to a sigmoid function but reaches its ± 1 limits in a finite number of iterations (see Discussion section).

A close look at the proposed output function shows two attraction mechanisms. The first one is a hard saturation function that bounds the output at ± 1; the second one is a mechanism that balances the positive $(\delta+1)\mathbf{a}_i$ and negative $-\delta\mathbf{a}_i^3$ parts. Thus, a unit's output remains unchanged if it reaches a value of 1, -1 or if $(\delta+1)\mathbf{a}_i - R = \delta\mathbf{a}_i^3$, where R is a limit with real value (e.g., 0.7). This mechanism enables the network to exhibit real-valued attractor behavior in addition to being a bipolar attractor.

As an example, consider a neuron whose output is 0.5 when given an input of 0.5, and suppose that $\delta = 0.4884$. Then, (10) can be written as

$$0.5 = (0.4884+1)w^*0.5 - 0.4884(w*0.5)^3 \tag{11}$$

and it has three possible solutions: $w = -3.7869$, $w = 0.7$, and $w = 3.0869$. Since the weights are initially set to 0 and their update is additive (17) they will converge to the first positive root, $w = 0.7$. We can then rewrite (10) in terms of that root and δ to obtain

$$x_{[t+1]} = (0.4884+1)0.7x_{[t]} - 0.4884(0.7x[t]^3. \tag{12}$$

When reaching a steady state, we have $x_{[t+1]} = x_{[t]}$ and the equation becomes

$$(0.4884+1)0.7x_{[t]} - 0.4884(0.7x_{[t]}^3) - x_{[t]} = 0 \tag{13}$$

with solutions -0.5, 0, and 0.5. To determine the kind of attraction exerted by each of these fixed points, we need to determine the slope of (13) at each of them. Thus, we need to compute

$$\left.\frac{d(0.4884+1)0.7x_{[t]} - 0.4884\left(0.7x_{[t]}^3\right) - x_{[t]}}{dx_{[t]}}\right|_{x_{[t]} = x^*} \tag{14}$$

where x^* is one of the fixed point. For the fixed points -0.5 and 0.5, the slope is equal to 0.916 239, indicating a stable monotonic approach. On the other hand, at the fixed point 0, the slope is equal to 1.041 88, indicating an unstable fixed point. In short,

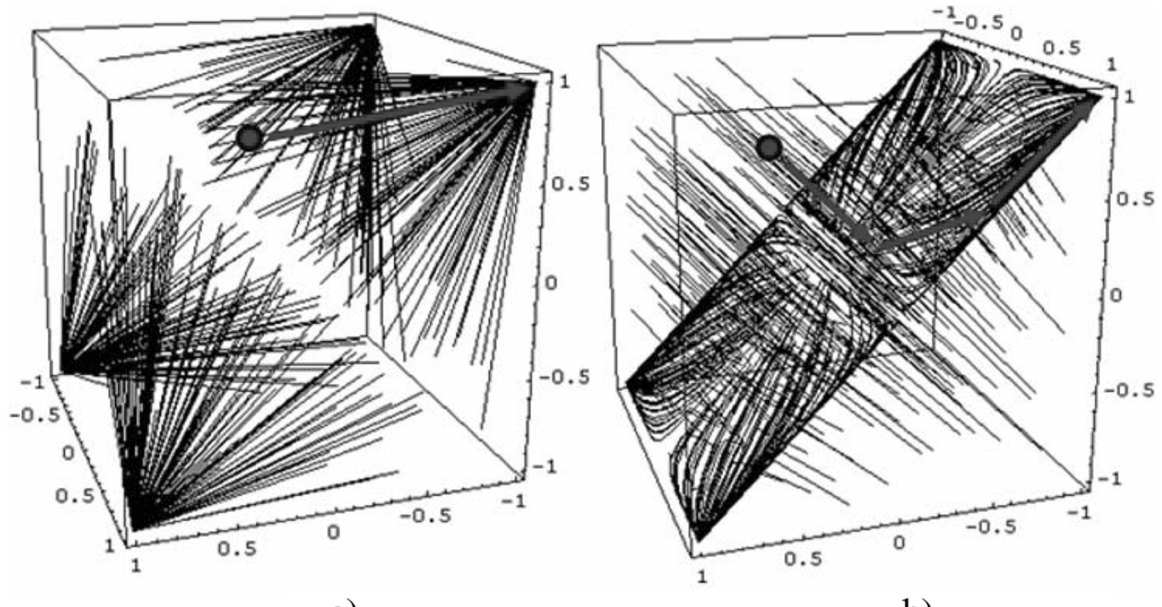


Fig. 4. Recall process example using (a) signum output function and (b) modified cubic map output function. Axes represent the value of a network unit.

if a value belongs to the interval $[-1, 0]$, then, the locally stable attractor is -0.5, and if a value belongs to the interval $[0, 1]$, then, the locally stable attractor is 0.5. On the other hand, the 0 fixed point is a repeller and can only capture a value of 0.

The previous analysis can be repeated for any real value within the definition domain of the proposed output function, showing that the latter may be used to recall any grey-level image whose pixel values belong to the domain.

To compare the effect of this new output function to that of the signum function usually found in the BAM, we computed the trajectories of 400 hundreds random patterns in the context of bipolar learning. The network was composed of three units, which gives a network space of three dimensions. The task was to learn two correlated independent bipolar stimuli; so the stimulus space was of two dimensions. We chose this setup to illustrate the outputs trajectories when the dimension of the network space is greater than that of the stimuli space $(3 > 2)$. Fig. 4 shows the results when the network has learned two such three-dimensional (3-D) bipolar stimuli. In the case of the signum function, the input stimulus converges from the network space to one of the stimuli space corners in one time step. On the other hand, the cubic output function takes several time steps to converge. The first time step projects the given stimulus from the network space to the stimuli space. Then, in the following time steps, the stimulus is progressively pushed toward one of the stimuli space corners. If one more pattern is added to the stimulus bank (three 3-D correlated patterns), the number of attractors becomes equal to six (three for the patterns and three for their complements). In the case of Kosko's BAM, there would have been two additional attractors which are the linear combination of the three stimuli (Fig. 5). Our network did not develop these two spurious attractors. Fig. 5 shows that even if a stimulus is closer to the spurious attractor, it will not be attracted by it.

B. Learning Rule

Most BAM models learn offline and, basically, solve the following linear equation:

$$\mathbf{Y} = \mathbf{W}\mathbf{X}. \tag{15}$$

Therefore, in the best case, the models use the one-shot projection rule as a solution (e.g., [16]). On the other hand, our model tries to find a solution to the following nonlinear constraints:

$$\mathbf{X} = f(\mathbf{V}\mathbf{Y}) \tag{16a}$$
$$\mathbf{Y} = f(\mathbf{W}\mathbf{X}) \tag{16b}$$

where f is the output function defined previously. The form of these constraints and the recurrent nature of the underlying network call for a learning process that is executed online. We used the following learning rule, derived from a Hebbian/anti-Hebbian approach [24]–[27]:

$$\mathbf{W}_{[k+1]} = \mathbf{W}_{[k]} + \eta\left(\mathbf{y}_{[0]}\mathbf{x}_{[0]}^T + \mathbf{y}_{[0]}\mathbf{x}_{[t]}^T - \mathbf{y}_{[t]}\mathbf{x}_{[0]}^T - \mathbf{y}_{[t]}\mathbf{x}_{[t]}^T\right) \tag{17}$$

$$\mathbf{V}_{[k+1]} = \mathbf{V}_{[k]} + \eta\left(\mathbf{x}_{[0]}\mathbf{y}_{[0]}^T + \mathbf{x}_{[0]}\mathbf{y}_{[t]}^T - \mathbf{x}_{[t]}\mathbf{y}_{[0]}^T - \mathbf{x}_{[t]}\mathbf{y}_{[t]}^T\right). \tag{18}$$

In the previous equations, $\mathbf{W}$ and $\mathbf{V}$ represent the weight matrices for both network directions, $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$ are the initial inputs to be associated, η is the learning parameter, and k is the learning trial number. We see that the learning rule includes a feedback from the nonlinear output function via $\mathbf{x}_{[t]}$ and $\mathbf{y}_{[t]}$.

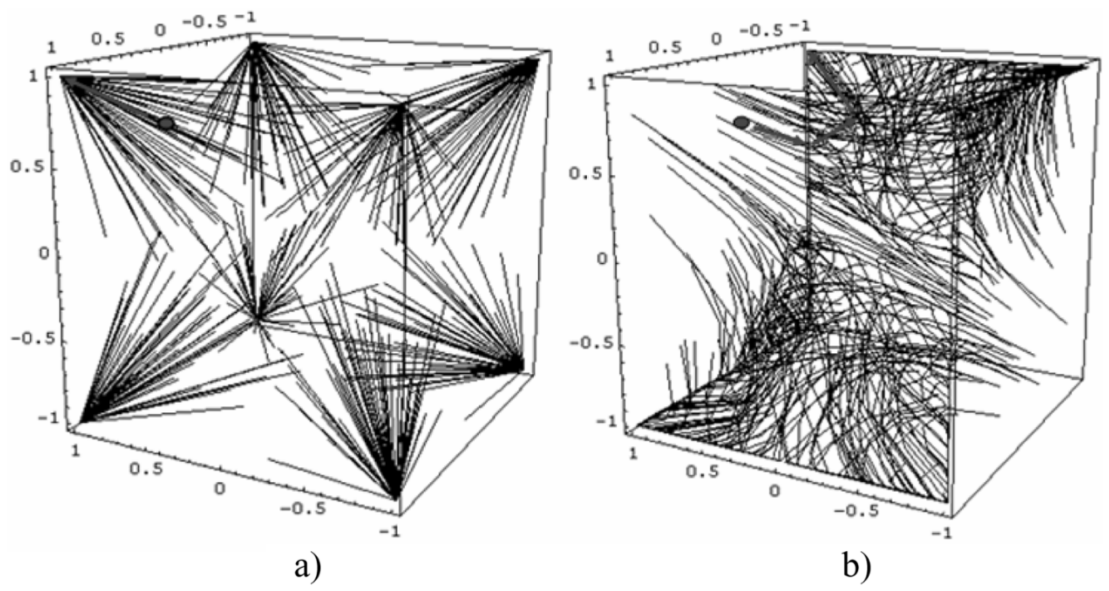


Fig. 5. Recall process example using (a) signum output function and (b) modified cubic map output function (three stimuli and three units).

This enables the network to learn online and contributes to the convergence of the weight connections.

Using binomial identities, we can rewrite the learning rule in compact form as

$$\mathbf{W}_{[k+1]} = \mathbf{W}_{[k]} + \eta(\mathbf{y}_{[0]} - \mathbf{y}_{[t]})(\mathbf{x}_{[0]} - \mathbf{x}_{[t]})^T \tag{19}$$
$$\mathbf{V}_{[k+1]} = \mathbf{V}_{[k]} + \eta(\mathbf{x}_{[0]} - \mathbf{x}_{[t]})(\mathbf{y}_{[0]} - \mathbf{y}_{[t]})^T. \tag{20}$$

Equations (19) and (20) show that the weight matrices will converge only when $\mathbf{y}_{[t]} = \mathbf{y}_{[0]}$ or $\mathbf{x}_{[t]} = \mathbf{x}_{[0]}$. Thus, each weight matrix converges when the feedbacks is the same as the initial inputs or, in other words, when the two are in resonance [28].

We can have another perspective on the learning rule from its error definition by comparing it to that of another neural network. For instance, the Adaline error is defined by the following quadratic function

$$e = (\mathbf{y} - a)(\mathbf{y} - a)^T \tag{21}$$

where $\mathbf{y}$ represents the desired value and $\mathbf{a}$ is the activation function ($\mathbf{a} = \mathbf{W}\mathbf{x}$). The error will be zero only if $\mathbf{y}$ is equal to $\mathbf{a}$. From (21), one can easily specify the Adaline standard update learning rule [29].

The error function of our model is different. If the output function is analyzed under the assumption of linear constraints (i.e., the output function is the same as the activation function instead of being given by (10)), we can show (see Appendix) that the forward and backward errors are expressed by

$$e_x = (\mathbf{x}_{[0]} - \mathbf{x}_{[1]})(\mathbf{x}_{[0]} - \mathbf{x}_{[1]} - \mathbf{x}_{[2]})^T \tag{22a}$$

and

$$e_y = (\mathbf{y}_{[0]} - \mathbf{y}_{[1]})(\mathbf{y}_{[0]} - \mathbf{y}_{[1]} - \mathbf{y}_{[2]})^T. \tag{22b}$$

As these expressions show, the errors are conditioned by the feedback at times one and two prior to the current time, and they become nil when the current values are equal to their respective previous values. This indicates that the learning rule tries to find fixed point attractors as a solution.

The solution the weight matrices converge to can be determined if we make the assumption that the number of iterations before each update of the weight connections takes place is $t = 1$, and that learning is accomplished with the prototype matrices. In this case, we have (see the Appendix)

$$\begin{aligned}\Delta\mathbf{W}_{[k]} &= \eta(\mathbf{Y}_{[0]} - \mathbf{Y}_{[t]})(\mathbf{X}_{[0]} - \mathbf{X}_{[t]})^T = 0\\ \Longrightarrow \mathbf{W} &= f^{-1}\left[\left(\mathbf{Y}_{[0]}\mathbf{X}_{[0]}^T + \mathbf{Y}_{[0]}\mathbf{X}_{[t]}^T\right)(\mathbf{X}_{[0]} + \mathbf{X}_{[t]})\right.\\ &\qquad \left.\times\left((\mathbf{X}_{[0]} + \mathbf{X}_{[t]})^T(\mathbf{X}_{[0]} + \mathbf{X}_{[t]})\right)^{-1}\right]\\ &\quad \times \mathbf{X}_{[0]}^T\left(\mathbf{X}_{[0]}\mathbf{X}_{[0]}^T\right)^{-1}\end{aligned} \tag{23}$$

and

$$\begin{aligned}\Delta\mathbf{V}_{[k]} &= \eta(\mathbf{X}_{[0]} - \mathbf{X}_{[t]})(\mathbf{Y}_{[0]} - \mathbf{Y}_{[t]})^T = 0\\ \Longrightarrow \mathbf{V} &= f^{-1}\left[\left(\mathbf{X}_{[0]}\mathbf{Y}_{[0]}^T + \mathbf{X}_{[0]}\mathbf{Y}_{[t]}^T\right)(\mathbf{Y}_{[0]} + \mathbf{Y}_{[t]})\right.\\ &\qquad \left.\times\left((\mathbf{Y}_{[0]} + \mathbf{Y}_{[t]})^T(\mathbf{Y}_{[0]} + \mathbf{Y}_{[t]})\right)^{-1}\right]\\ &\quad \times \mathbf{Y}_{[0]}^T\left(\mathbf{Y}_{[0]}\mathbf{Y}_{[0]}^T\right)^{-1}.\end{aligned} \tag{24}$$

Equations (23) and (24) show that each of the final weight matrices develops as a function of the nonlinear output function, a cross-correlation matrix and a normalization factor. In addition, each of the weight matrices depends on the value of the other one, thus, making the updates a recurrent nonlinear dynamic process. Because of this, it is likely that the final weight matrix can only be found by iteration.

Finally, the learning rule can be simplified in the case of an autoassociative memory. In this case, the weight matrix is square

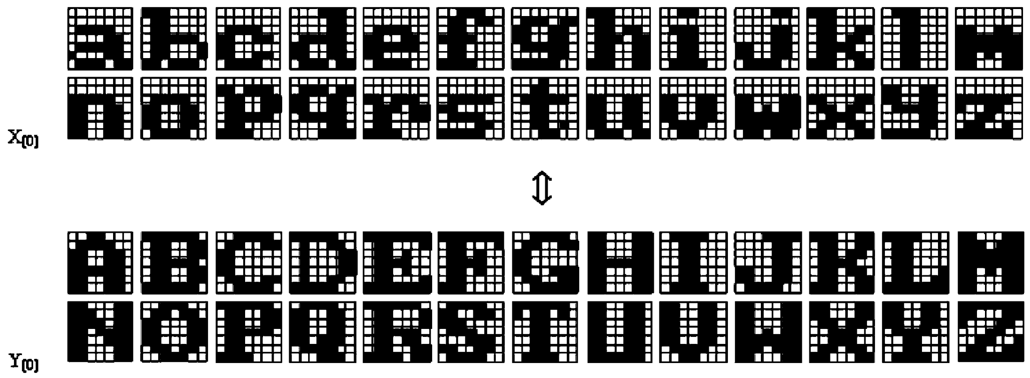


Fig. 6. Bipolar pattern pairs to be associated.

TABLE I
PARAMETERS AND PROPERTIES COMPARISON FOR DIFFERENT BAM MODELS

	KBAM	SBAM	ABAM	GBAM	GABAM	New Model
Weights X→Y Y→X Extra layer/nodes	 **W** $\mathbf{W}^T$ None	 **W** $\mathbf{W}^T$ None	 **A** **B** None	 **A** **B** None	 $\mathbf{W}_f$ $\mathbf{W}_b$ Λ^i	 **W** **V** None
Learning procedure	One-shot Hebbian	Iterative perceptron	One-shot matrix operations	Iterative perceptron	Iterative hebbian and backprop	Iterative matrix operations
Online learning	No	No	No	No	No	Yes
Free parameters Learning Output	 1/N None	 None None	 None None	 None None	 η None	 η δ
Storage capacities	$0.15n$	n	n	$>n$	n	n
Training Pattern pairs expended	No	Yes	No	Yes	No	No

and symmetric, with the effect of canceling the cross terms in (17) and (18), and the general autoassociative learning rule is

$$\begin{aligned}\mathbf{W}_{[k+1]} &= \mathbf{W}_{[k]} \\ &\quad + \eta\left(\mathbf{x}_{[0]}\mathbf{x}_{[0]}^T + \mathbf{x}_{[0]}\mathbf{x}_{[t]}^T - \mathbf{x}_{[t]}\mathbf{x}_{[0]}^T - \mathbf{x}_{[t]}\mathbf{x}_{[t]}^T\right) \\ \Longrightarrow \mathbf{W}_{[k+1]} &= \mathbf{W}_{[k]} + \eta\left(\mathbf{x}_{[0]}\mathbf{x}_{[0]}^T - \mathbf{x}_{[t]}\mathbf{x}_{[t]}^T\right). \qquad (25)\end{aligned}$$

Thus, in the case of an autoassociative network, the learning rule is simply the sum of a Hebbian and an anti-Hebbian term. In this respect, the learning rule described in (17) and (18) may be viewed as a generalization of several autoassociative networks (e.g., [3], [5], [24]–[27]) whose updates also consist of the sum of Hebbian and anti-Hebbian terms. For details about the autoassociative model see [27].

III. SIMULATIONS

In order to study the performance of our model and confirm its distinguishing properties in comparison to other BAM networks, we performed several computer simulations to learn and recall bipolar and real-valued correlated patterns.

A. Simulation 1: Learning and Recall of Bipolar Stimuli

The purpose of this simulation was to compare the performance of the proposed BAM with several BAM networks found in the literature. More precisely, the simulations were made following the ones described in [19].

1) Methodology: The network task was to associate 26 correlated patterns consisting of 7×7 pixels images where a white pixel was given the value -1 and a black pixel the value 1. Fig. 6 illustrates the stimuli used for the simulation. The images were converted to vectors of 49 dimensions before being input to the network. This led to a memory load equal to 53% (26/49) of the 49-dimensional space capacity. Normally, such a high load value cannot be handled by Kosko's BAM.

The output function parameter was set to 0.1 and the learning parameter was set to 0.01. Also, to limit the simulation time, we set the number of output iterations before each weight matrix update to 1.

Learning was carried according to the following procedure:

1) random selection of a pattern pair;
2) computation of $\mathbf{x}_t$ and $\mathbf{y}_t$ according to the output function [(10)];
3) computation of the weight matrix update according to (17) and (18);
4) repetition of steps 1) to 3) until the weight matrix converges (about 2000 learning trials).

After completing the learning phase, the network performance was tested on a noisy recall task. The task was to recall the correct associated stimulus from a noisy input obtained by randomly flipping pixels in the input pattern. The number of pixel flips varied from 0 to 10, thus, corresponding to a noise proportion of 0 to 20%. We compared the network recall performance with the ones reported in [19], which corresponded to the symmetrical BAM (SBAM) [15], asymmetrical BAM (ABAM) [17], general BAM (GBAM) [20] and generalized asymmetrical BAM (GABAM) [19] networks . Table I shows the comparisons between the different models that were considered. There exist newer symmetric models in the literature

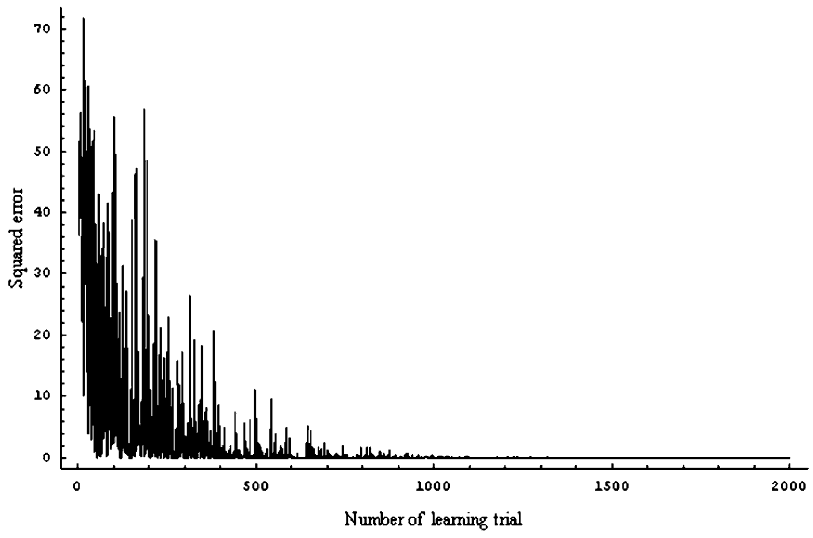


Fig. 7. Squared error as a function of the number of learning trials.

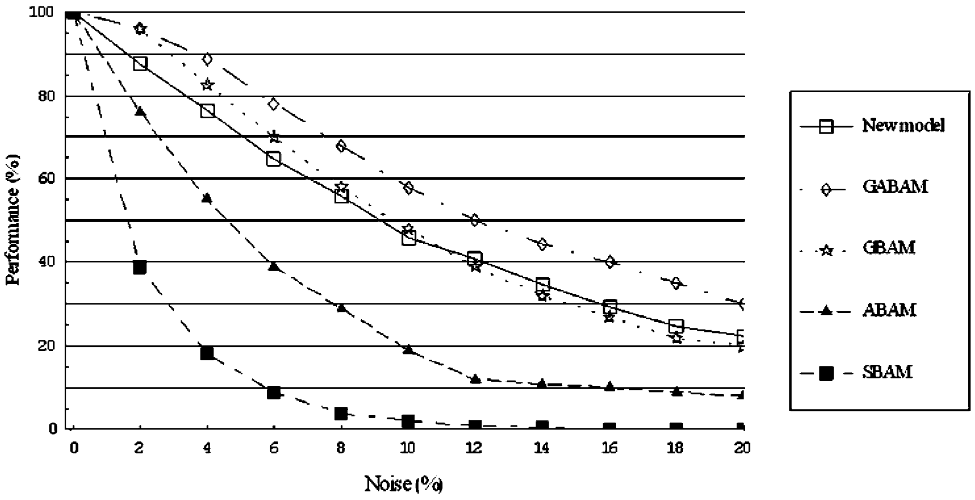


Fig. 8. Performance of the various BAM models for the recall task.

(e.g., [8], [9]), however, their performance compares to that of the SBAM model; we ignored them as a result.

We also determined the proportion of spurious attractors by calculating the number of vectors that stabilize in a spurious state in relation to the total number of vectors. We generated 1000 random vectors in order to conduct that experiment.

Network storage capacity was evaluated by using random 10-dimensional bipolar vectors as done by [19] and [20]. We varied the number of pairs to be learned from 1 to 15 and, in each case, generated 1000 different test sets and computed the resulting network recall accuracy.

Finally, a simulation was added to demonstrate the online learning capacity of the network. A series of 100 patterns pairs was randomly generated, using only the first four characters in the training set for simplicity (instead of the 26 characters used previously). For each pair, the output was calculated based on one output iteration and on 25 output iterations. The last condition is closer to real-life applications where the number of output iterations during the recall task is about the same as that during learning.

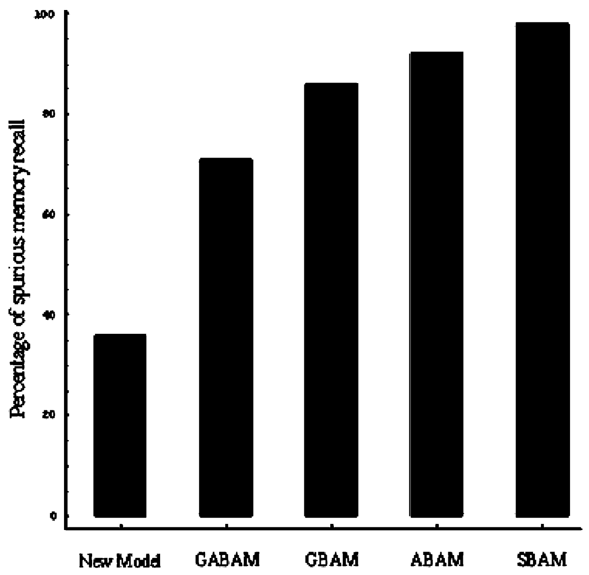


Fig. 9. Percentage of spurious memories during recall.

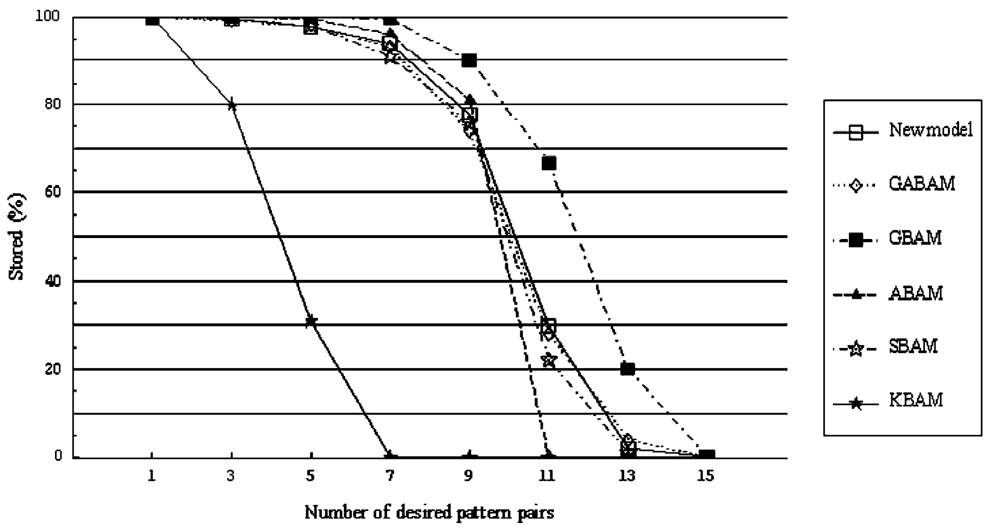


Fig. 10. Storage capacity comparison.

2) Results: Fig. 7 shows that the weight matrices were able to stabilize within a relatively short number of learning trials. After 2000 learning trials, the average squared error was less than 0.0005. Fig. 8 illustrates the network performance on the recall task. It shows that the performance was better than that of SBAM and ABAM; it was comparable to that of GBAM; and it was inferior to that of the best GABAM version composed of a hidden layer of 52 units and using the backpropagation error learning algorithm. The good recall performance is also expressed by the reduced number of spurious attractors: Fig. 9 shows that the network developed only 36% spurious attractors, which is 50% less than the best other network, the GABAM.

The storage capacity results (Fig. 10) show a network storage capacity similar to that of SBAM, ABAM, and GABAM, however, it is inferior to the storage capacity of GBAM. Fig. 10 shows that the storage capacity decreases gradually until the number of stimuli pairs reaches the number of units; then, it falls sharply. For a good recall performance under noise degradation, the number of stimuli pairs to be learned should be about half that threshold.

Finally, Fig. 11 illustrates the network's ability to learn online. There was no need for particular pattern sequences for the network to develop the appropriate stable fixed points. In addition, the network was able to develop the correct stable fixed points regardless of the number of output iterations during the learning phase. On the other hand, the duration of training before convergence appears to be directly proportional to the number of output iterations conducted, thus, imposing a constraint about the recall/learning procedure: The number of output iterations performed during the learning must be equal or greater than the one during recall. For example, if it takes 50 iterations for a given stimulus to convergence, then, the number of iterations performed by (10) must be over 50 during learning.

B. Simulation II: Learning and Recall of Grey- Level Images

In this simulation, the network's task was to associate five grey-level images with five alphabetic letters of a different dimension.

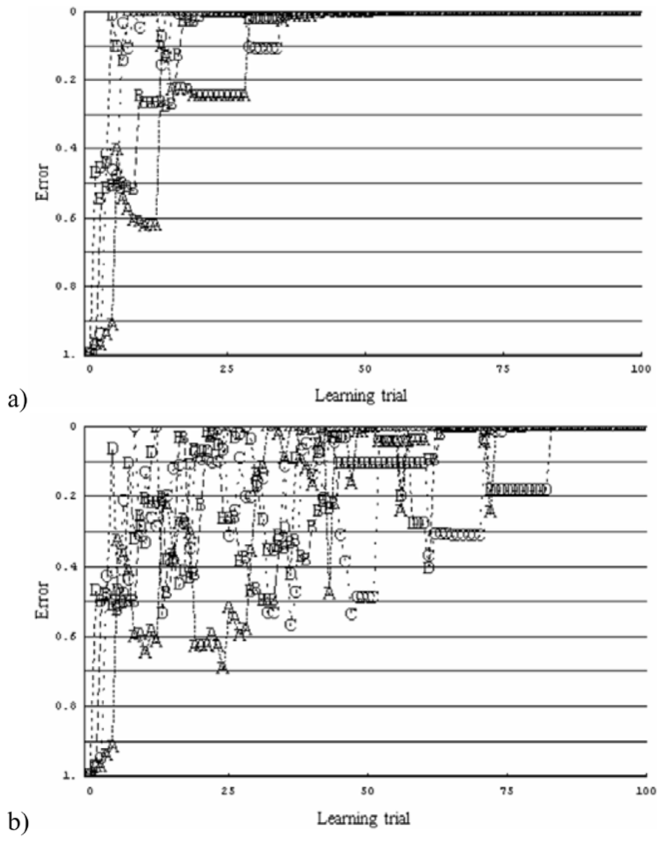


Fig. 11. Online learning of random sequence. (a) One output iteration before learning. (b) 25 output iterations before learning.

1) Methodology: The grey-level images were computer icons of 16×16 pixels and the letters were 7×5 pixels in size. The icons with the corresponding arbitrarily chosen letters are shown in Fig. 12.

Each icon pixel was 8-bit coded with an initial value in the interval $[0, 255]$ before being scaled down to the range $[-1, 1]$. The mid-grey value of 0 was set to 0.1 to avoid the 0 unstable fixed points. The letter patterns were coded with a white pixel having the value -1 and a black pixel the value 1. As a result,

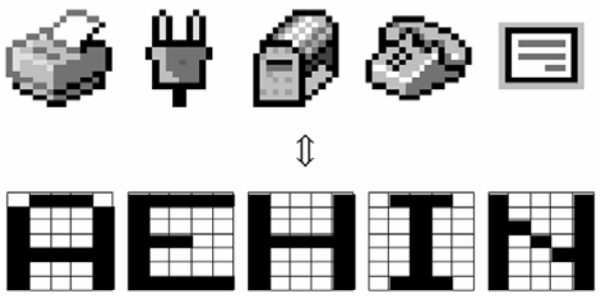


Fig. 12. Pattern pairs for Simulation II.

an associated vector pair was composed of a real-valued vector of 256 dimensions and a bipolar vector of 35 dimensions. The learning methodology was the same as in the previous section, with the same choice of free parameters.

2) Results: Fig. 13 shows the recall accuracy squared error as a function of the number of learning trials. We see that the error is almost zero after 100 learning trials, and it is less that 0.00015 after 200 learning trials. Consequently, the weight matrices were still able to converge when learning grey-level images. The network also exhibited the expected BAM properties of pattern completion and noise tolerance. Fig. 14 shows three different examples that confirm those properties.

IV. Discussion and Conclusion

As our results show, the bidirectional associative memory presented in this paper exhibits good performance even though its learning rule is based solely on local cross-correlation, without any additional optimization constraints. For instance, it only uses two layers of units instead of the three layers found in GABAM [19] and the feedforward BAM [18], and it does not need any extended pattern pairs like GBAM [20] or SBAM [15]. In addition, contrary to other BAM networks, its learning is iterative and accomplished online without overgeneralization or first eigenvalue dominance.

The previous properties are interesting by themselves, however, perhaps the network's most interesting feature is its capacity to learn real-valued patterns. It can learn and recall grey-level images without resorting to any prior bipolar coding or adding inflection points into the output function. This property is a clear advantage over the other BAMs, and it makes the network a more plausible model for cognition since the real-valued network states we introduce can be compared to rate models, instead of just representing the usual presence or absence of an action potential [31]. By using this representation of firing frequencies, the network acquires the ability to model more cognitive processes than the others BAMs.

This property of the network comes from the incorporation of the nonlinear output function of (7) into the learning rule, which not only enables the model to learn online but also increases its performance. Other nonlinear output functions that are bistable (e.g., signum, piecewise linear, etc.,) may increase the performance but they cannot develop real-valued attractors, as reported in [23]. The incorporation of a logistic function may in certain conditions output real values, however, those values must be near -1 and $+1$. Thus, networks that use it may not be able to correctly output the grey-level images shown in Fig. 12, where the pixel values lie anywhere in the range $[-1, 1]$.

Our output function contrasts with the one proposed by Zurada *et al.* [23] who use a normal sigmoid function that is modified by the addition of inflection points. Each inflection point divides the attractor space according to the number of grey-levels needed. However, each time an inflection point is added to the function, the radius of attraction decreases and so does the tolerance to noise. For example, in a bipolar setting, there is only one inflection point, giving a maximum distance of 1. On the other hand, when trying to recall eight levels of grey, we need seven inflection points giving a maximum distance of 0.25. In contrast, the output function expressed in (10) only needs one inflection point at 0 to accomplish its task, regardless of whether we have bipolar or grey-level images. As a result, the function does need not any prior information on the number of shades of grey present in the input in order for the network to operate correctly.

The type of learning rule is less critical, aside from the requirement that it must incorporate the output—not the activation—of the neurons (e.g., [10]). In regards to this output, the weight matrices must be able to self-stabilize in local fashion. The rule must also be incremental and tolerant to overlearning without loss of performance. Thus, the overall performance of the network depends on the interaction of the nonlinear learning rule with the nonlinear output function.

In summary, we introduced a simple new heteroassociative neural network that can learn bipolar and real-valued patterns. The new model offers good performance in comparison to other BAM architectures, without an increase in learning or architectural complexity. We showed that the network is able to stabilize its weights to fixed point attractors in a local fashion. The model is immune to overlearning and develops fewer spurious attractors compared to other BAM. Although more simulations are needed to determine its dynamic behavior with greater precision, the bidirectional associative memory introduced in this paper represents a good alternative to other BAMs and models of cognition.

Further research includes the investigation of the learning of two grey level patterns that share the same quadrant without loss of stability, multistep pattern recognition [30] and nonlinear classification in a multilevel architecture. Further research should also investigate the possibility of modifying the output function to handle nonlinearly separable problems.

Appendix

A. Derivation of the Output Function

A differential equation of the form

$$\frac{dz}{dt} = f(z)$$

can be rewritten as

$$\frac{dz}{dt} = \lim_{\Delta \to 0} \frac{z_{[t+1]} - z_{[t]}}{\Delta}.$$

If we assume that Δ is small but finite, we obtain the following approximation:

$$\frac{z_{[t+1]} - z_{[t]}}{\Delta} = f(z_{[t]}) \Longrightarrow z_{[t+1]} = \Delta f(z_{[t]}) + z_{[t]}.$$

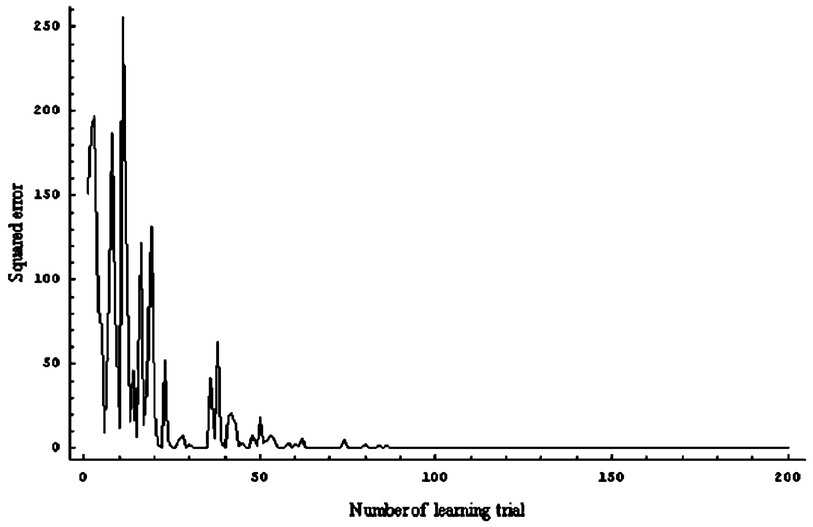


Fig. 13. Squared error as a function of the number of learning trials.

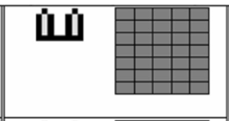
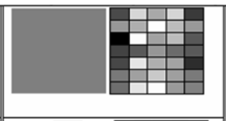
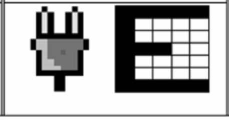
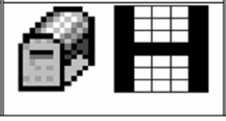
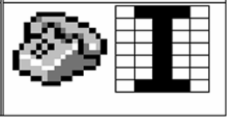
Initial inputs			
Final outputs			

Fig. 14. Network performance for incomplete or noisy inputs.

With

$$f(z_{[t]}) = R\left(1 - z_{[t]}^2\right) z_{[t]}$$

we obtain

$$z_{[t+1]} = \Delta R\left(1 - z_{[t]}^2\right) z_{[t]} + z_{[t]}.$$

Setting $\delta = \Delta R$ and rearranging the terms yields

$$\begin{aligned} z_{[t+1]} &= \delta\left(1 - z_{[t]}^2\right) z_{[t]} + z_{[t]} \\ z_{[t+1]} &= (\delta + 1) z_{[t]} - \delta z_{[t]}^3. \end{aligned}$$

Finally, setting $z_{[t]}$ to the output $(z_{[t]} = \mathbf{a}_{[t]} = \mathbf{W}\mathbf{x}_{[t]})$ and $\mathbf{z}_{[t+1]}$ to the output $(\mathbf{z}_{[t+1]} = \mathbf{y}_{[t+1]})$ leads to

$$\mathbf{y}_{[t+1]} = (\delta + 1)\mathbf{a}_{[t]} - \delta \mathbf{a}_{[t]}^3.$$

B. Error Minimization and Learning Rule

Given the linear output functions

$$\begin{aligned} \mathbf{x}_{[1]} &= \mathbf{V}\mathbf{y}_{[0]} \\ \mathbf{y}_{[1]} &= \mathbf{W}\mathbf{x}_{[0]} \\ \mathbf{y}_{[2]} &= \mathbf{W}\mathbf{x}_{[1]} = \mathbf{W}\mathbf{V}\mathbf{y}_{[0]} \end{aligned}$$

we define the error as

$$\begin{aligned} \mathbf{e} &= (\mathbf{y}_{[0]} - \mathbf{y}_{[1]})(\mathbf{y}_{[0]} - \mathbf{y}_{[1]} - \mathbf{y}_{[2]})^{\mathbf{T}} \\ \mathbf{e} &= (\mathbf{y}_{[0]} - \mathbf{W}\mathbf{x}_{[0]})(\mathbf{y}_{[0]} - \mathbf{W}\mathbf{x}_{[0]} - \mathbf{W}\mathbf{V}\mathbf{y}_{[2]})^{\mathbf{T}} \\ \mathbf{e} &= \mathbf{y}_{[0]}\mathbf{y}_{[0]}^T - \mathbf{y}_{[0]}\mathbf{x}_{[0]}^T\mathbf{W}^T - \mathbf{y}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T\mathbf{W}^T \\ &\quad - \mathbf{W}\mathbf{x}_{[0]}\mathbf{y}_{[0]}^T + \mathbf{W}\mathbf{x}_{[0]}\mathbf{x}_{[0]}^T\mathbf{W}^T + \mathbf{W}\mathbf{x}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T\mathbf{W}^T \\ \mathbf{e} &= \mathbf{y}_{[0]}\mathbf{y}_{[0]}^T - 2\mathbf{y}_{[0]}\mathbf{x}_{[0]}^T\mathbf{W}^T - \mathbf{y}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T\mathbf{W}^T \\ &\quad + \mathbf{W}\mathbf{x}_{[0]}\mathbf{x}_{[0]}^T + \mathbf{W}^T + \mathbf{W}\mathbf{x}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T\mathbf{W}^T. \end{aligned}$$

The gradient of this error with respect to the weight matrix $\mathbf{W}$ is

$$\begin{aligned} \frac{de}{d\mathbf{W}^T} &= -2\mathbf{y}_{[0]}\mathbf{x}_{[0]}^T - \mathbf{y}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T + 2\mathbf{W}\mathbf{x}_{[0]}\mathbf{x}_{[0]}^T \\ &\quad + 2\mathbf{W}\mathbf{x}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T \\ \frac{de}{d\mathbf{W}^T} &= -2(\mathbf{y}_{[0]}\mathbf{x}_{[0]}^T + \frac{1}{2}\mathbf{y}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T + \mathbf{W}\mathbf{x}_{[0]}\mathbf{x}_{[0]}^T \\ &\quad - \mathbf{W}\mathbf{x}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T). \end{aligned}$$

Because we want the weights to update in the opposite direction of the gradient, we multiply this last result by -1. In addition, the weight updates must be small since we do not want the network to oscillate around the solution. We, thus, replace the

$$
\begin{aligned}
&\Delta \mathbf{W} = \eta \left(\mathbf{Y}_{[0]}\mathbf{X}_{[0]}^T + \mathbf{Y}_{[0]}\mathbf{X}^T - f(\mathbf{W}\mathbf{X}_{[0]})\mathbf{X}_{[0]}^T - f(\mathbf{W}\mathbf{X}_{[0]})\mathbf{X}^T \right) = 0 \\
&\Longrightarrow f(\mathbf{W}\mathbf{X}_{[0]})\mathbf{X}_{[0]}^T + f(\mathbf{W}\mathbf{X}_{[0]})\mathbf{X}^T = \mathbf{Y}_{[0]}\mathbf{X}_{[0]}^T + \mathbf{Y}_{[0]}\mathbf{X}^T \\
&\Longrightarrow f(\mathbf{W}\mathbf{X}_{[0]})(\mathbf{X}_{[0]}^T + \mathbf{X}^T) = \mathbf{Y}_{[0]}\mathbf{X}_{[0]}^T + \mathbf{Y}_{[0]}\mathbf{X}^T \\
&\Longrightarrow f(\mathbf{W}\mathbf{X}_{[0]}) = (\mathbf{Y}_{[0]}\mathbf{X}_{[0]}^T + \mathbf{Y}_{[0]}\mathbf{X}^T)(\mathbf{X}_{[0]}^T + \mathbf{T})^T \left((\mathbf{X}_{[0]}^T + \mathbf{X}^T)(\mathbf{X}_{[0]}^T + \mathbf{X}^T)^T \right)^{-1} \\
&\Longrightarrow \mathbf{W}\mathbf{X}_{[0]} = f^{-1}\left[(\mathbf{Y}_{[0]}\mathbf{X}_{[0]}^T + \mathbf{Y}_{[0]}\mathbf{X}^T)(\mathbf{X}_{[0]} + \mathbf{X}) \left((\mathbf{X}_{[0]} + \mathbf{X})^T (\mathbf{X}_{[0]} + \mathbf{X}) \right)^{-1} \right] \\
&\Longrightarrow \mathbf{W} = f^{-1}\left[(\mathbf{Y}_{[0]}\mathbf{X}_{[0]}^T + \mathbf{Y}_{[0]}\mathbf{X}^T)(\mathbf{X}_{[0]} + \mathbf{X}) \left((\mathbf{X}_{[0]} + \mathbf{X})^T (\mathbf{X}_{[0]} + \mathbf{X}) \right)^{-1} \right] \mathbf{X}_{[0]}^T (\mathbf{X}_{[0]}\mathbf{X}_{[0]}^T)^{-1}.
\end{aligned}
$$

value 2 by a general learning parameter, η, and the 1/2 value by another general learning parameter, β. We then obtain

$$
\Delta \mathbf{W} = \eta(\mathbf{y}_{[0]}\mathbf{x}_{[0]}^T + \beta \mathbf{y}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T - \mathbf{W}\mathbf{x}_{[0]}\mathbf{x}_{[0]}^T - \mathbf{W}\mathbf{x}_{[0]}\mathbf{y}_{[0]}^T\mathbf{V}^T).
$$

Finally, if we set $\beta = 1$ and generalize $x_{[1]}$ and $y_{[0]}$ to $x_{[t]}$ and $y_{[t]}$, we end up with the following learning rule:

$$
\Delta \mathbf{W} = \eta \left(\mathbf{y}_{[0]}\mathbf{x}_{[0]}^T + \mathbf{y}_{[0]}\mathbf{x}_{[t]}^T - \mathbf{y}_{[t]}\mathbf{x}_{[0]}^T - \mathbf{y}_{[t]}\mathbf{x}_{[t]}^T \right).
$$

Thus, this learning rule is a specific case of the more general update rule defined previously.

C. Final Solutions for the Weight Update Equations

Given $X_{[0]}$ and $Y_{[0]}$, the stimuli matrices, the values of the output function in each direction are

$$
\mathbf{Y} = f(\mathbf{W}\mathbf{X}_{[0]})
$$

and

$$
\mathbf{X} = f(\mathbf{V}\mathbf{Y}_{[0]}).
$$

The weights update for matrix $\mathbf{W}$ is, then, given by

$$
\Delta \mathbf{W} = \eta \left(\mathbf{Y}_{[0]}\mathbf{X}_{[0]}^T + \mathbf{Y}_{[0]}\mathbf{X}^T - \mathbf{Y}\mathbf{X}_{[0]}^T - \mathbf{Y}\mathbf{X}^T \right).
$$

Setting this equation to 0 and solving it for $\mathbf{W}$ provides the desired expression, as shown at the top of the page.

REFERENCES

[1] T. Kohonen, “Correlation matrix memories,” *IEEE Trans. Comput.* , vol. C-21, pp. 353–359, Dec. 1972.

[2] S.-I. Amari, “Neural theory of association and concept-formation,” *Biol. Cybern.*, vol. 26, pp. 175–185, 1977.

[3] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, “Distinctive features, categorical perception, and probability learning: Some applications of neural model,” *Psychol. Rev.*, vol. 84, pp. 413–451, 1977.

[4] B. Kosko, “Bidirectional associative memories,” *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 1, pp. 49–60, Jan.-Feb. 1988.

[5] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proc. Nat. Acad. Sci.*, vol. 79, pp. 2554–2558, 1982.

[6] L. Personnaz, I. Guyon, and G. Dreyfus, “Information storage and retrieval in spin-glass like neural networks,” *J. Physique Letter*, vol. 46, pp. L359–365, 1985.

[7] Y. F. Wang, J. B. Cruz Jr, and J. H. Mulligan Jr, “Two coding strategies for bidirectional associative memory,” *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 81–92, Mar. 1990.

[8] D. Shen and J. B. Cruz Jr, “Encoding strategy for maximum noise tolerance bidirectional associative memory,” *IEEE Trans. Neural Netw.*, vol. 16, no. Mar., pp. 293–300, 2005.

[9] S. Du, Z. Chen, Z. Yuan, and X. Zhang, “Sensitivity to noise in bidirectional associative memory (BAM),” *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 887–898, Jul. 2005.

[10] C.-S. Leung, “Optimum learning for bidirectional associative memory in the sense of capacity,” *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 5, pp. 791–795, May 1994.

[11] H. Oh and S. C. Kothari, “Adaptation of the relaxation method for learning in bidirectional associative memory,” *IEEE Trans. Neural Netw.*, vol. 5, no. 4, pp. 576–583, Jul. 1994.

[12] T. Wang, X. Zhuang, and X. Xing, “Weighted learning of bidirectional associative memories by global minimization,” *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 1010–1018, Nov. 1992.

[13] S. Arik, “Global asymptotic stability analysis of bidirectional associative memory neural networks with time delays,” *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 580–586, May 2005.

[14] L. Wang and Z. Zou, “Capacity of stable periodic solutions in discrete-time bidirectional associative memory neural networks,” *IEEE Trans. Circuits Syst. II, Exp . Briefs*, vol. 51, no. Jun., pp. 315–319, 2004.

[15] X. Zhuang, Y. Huang, and S. Chen, “Better learning for bidirectional associative memory,” *Neural Netw.*, vol. 6, pp. 1131–1146, 1993.

[16] Z. Wang, “A bidirectional associative memory based on optimal linear associative memory,” *IEEE Trans. Comput.*, vol. 10, pp. 1171–1179, Oct. 1996.

[17] Z. B. Xu, Y. Leung, and X. W. He, “Asymmetric bidirectional associative memories,” *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 10, pp. 1558–1564, Oct. 1994.

[18] Y. Wu and D. A. Pados, “A feedforward bidirectional associative memory,” *IEEE Trans. Neural Netw.*, vol. 11, no. 4, Jul. 2000.

[19] T. Eom, C. Choi, and J. Lee, “Generalized asymmetrical bidirectional associative memory for multiple association,” *Appl. Math. Comput.*, vol. 127, pp. 221–233, 2002.

[20] H. Shi, Y. Zhao, and X. Zhuang, “A general model for bidirectional associative memories,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 28, no. 4, pp. 511–519, Aug. 1998.

[21] Koronovskii, D. I. Trubetskov, and A. E. Khramov, “A discrete population dynamics as a process obeying the nonlinear diffusion equation,” *Doklady Earth Sci.*, vol. 372, no. 4, pp. 755–758, 2000.

[22] D. Kaplan and L. Glass, *Understanding Nonlinear Dynamics*. New-York: Springer-Verlag, 1998.

[23] J. M. Zurada, I. Cloete, and E. van der Poel, “Generalized Hopfield networks for associative memories with multi-valued stable states,” *Neurocomput.*, vol. 13, pp. 135–149, 1996.

[24] J. Storkey and R. Valabregue, “The basins of attraction of a new Hopfield learning rule,” *Neural Netw.*, vol. 12, pp. 869–876, 1999.

[25] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Mach. Learn.*, vol. 3, pp. 9–44, 1988.

[26] J. Bégin and R. Proulx, "Categorization in unsupervised neural networks: The Eidos model," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 147–154, Jan. 1996.

[27] S. Chartier and R. Proulx, "NDRAM: Nonlinear dynamic recurrent associative memory for learning bipolar and nonbipolar correlated patterns," *IEEE Trans. Neural Netw.* , vol. 16, no. 6, pp. 1393–1400, Nov. 2005, to be published.

[28] S. Grossberg, "Nonlinear neural networks: Principle, mechanisms, and Architecture," *Neural Netw.*, vol. 1, pp. 17–61, 1989.

[29] K. Mehrotra, C. K. Mohan, and S. Ranka, *Elements of Artificial Neural Networks*. Cambridge, MA: MIT Press, 2000.

[30] D.-L. Lee, "A discrete sequential bidirectional associative memory for multistep pattern recognition," *Pattern Recognit. Lett.*, vol. 19, pp. 1087–1102, 1998.

[31] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

Sylvain Chartier (M'05) received the B.A. degree, with a minor in psychology, from the Université d'Ottawa, ON, Canada, in 1993, and the B.Sc. and Ph.D. degrees in psychology from the Université du Québec à Montréal, Montréal, QC, Canada, in 1996 and 2004, respectively.

Currently, he is working towards a postdoctoral fellowship at the Institut Philippe Pinel, Montréal, QC, Canada, and at the Université du Québec en Outaouais, QC, Canada, where he is developing neural nets applications for oculomotor data classification. His research interests include associative memory, dynamic system, nonlinear signal processing, and pattern recognition.

Mounir Boukadoum (M'90–SM'05) received the M.E.E. degree in electrical engineering from the Stevens Institute of Technology, Hoboken, NJ, in 1978, and the Ph.D. degree in electrical engineering from the University of Houston, TX, in 1983.

He was an Electronics Instructor at the Houston Community College, TX, for one year before joining the Université du Québec à Montréal (UQAM), Montréal, QC, Canada, in 1984, where he is now Professor of Microelectronics at the Computer Science Department. He was elected Chairperson of Microelectronics Programs, in 1995, and he was re-elected, in 1998 and 2001. He is currently the director of the Microelectronics Prototyping Research Laboratory, UQAM. His research interests focus on the use of neural networks and fuzzy logic models for data processing, pattern recognition, and instrument design, particularly in the area of fluorescence measurements. He is also interested in the biomedical applications of ultrasound.