Bidirectional Associative Memories, Self-Organizing Maps and $k$-Winners-Take-All: Uniting Feature Extraction and Topological Principles

Sylvain Chartier, Gyslain Giguère, Dominic Langlois and Rana Sioufi

Abstract—In this paper, we introduce a network combining $k$-Winners-Take-All and Self-Organizing Map principles within a Feature Extracting Bidirectional Associative Memory. When compared with its "strictly winner-take-all" version, the modified model shows increased performance for clustering, by producing a better weight distribution and a lower dispersion level (higher density) for each given category. Moreover, because the model is recurrent, it is able to develop prototype representations strictly from exemplar encounters. Finally, just like any recurrent associative memory, the model keeps its reconstructive memory and noise filtering properties.

I. INTRODUCTION

In everyday life, humans are continually required to organize and reorganize perceptual patterns in categories on the basis of new incoming information, in order to act upon the identity and properties of the encountered stimuli. While it is generally agreed that the cognitive system uses some type of feature system to achieve these operations, the idea of a perceptual feature system (e.g. [1]), in which properties would be: 1) defined in a strict iconic fashion, and 2) self-created in major part by associative bottom-up processes (such as those found in connectionist models), is quite recent. Representation-wise, cognitive scientists have argued endlessly over the exclusive use of either generic abstractions (such as prototypes) or very specific perceptual stimulations (such as complete exemplars) to achieve proper category learning and classification. Seemingly, the use of one of these representations is likely to be closely linked to specific environmental demands and system goals [2].

Overall, category formation in a perceptual framework is often seen as a process akin to classic clustering techniques, which involve partitioning stimulus spaces in a number of finite sets, or clusters. In cognitive modeling, Principal Component Analysis (PCA) and Independent Component Analysis (ICA) neural networks have been shown to achieve clustering (see [3] for a review). Recently, [4] have shown that their Feature-Extracting Bidirectional Associative Memory (FEBAM) can lead to the unification of PCA/ICA networks and Bidirectional Heteroassociative Memories (BHMs) under a common framework. More precisely, FEBAM, which is based on a Bidirectional Associative Memory (BAM) framework, has been shown to use the equivalent of a nonlinear PCA learning rule.

Of course, clustering in neural nets is not limited to PCA networks. In fact, competitive networks (e.g. [5],[6]) constitute local, dynamic versions of clustering algorithms. In these models, each output unit represents a specific cluster. When taking decisions, the association between an exemplar and its determined cluster unit in the output layer is strengthened. In winner-take-all (WTA) networks [5],[6], exemplars may only be associated with one cluster (i.e. only one output unit at a time can be activated).

An example of one such hard competitive framework is that of the Adaptive Resonance Theory (ART: [5]). ART networks possess the advantage of being able to deal effectively with the exemplars/prototype scheme, while solving the stability/plasticity dilemma. These unsupervised models achieve the desired behavior through the use of a novelty detector (using vigilance). Various degrees of generalization can be achieved by this procedure: low vigilance parameter values lead to the creation of broad categories, while high values lead to narrow categories, with the network ultimately performing exemplar learning.

Another example of this framework is the self-organizing feature map (SOFM: [6]), which, in addition, uses a topological representation of inputs and outputs. Although SOFMs only consider one active output unit at a time, the learning algorithm also allows for physically close neighboring units to update their connection weights. In a SOFM, an exemplar may thus, for instance, be geometrically positioned between two clusters, and possess various degrees of membership. Recently, we have proposed a Bidirectional Associative Memory (BAM) model that was modified to take into account the properties of SOFMs [7].

An extension of the WTA principle selects the $k$ largest outputs from the total $n$ outputs [8]. This $k$-winners-take-all ($k$WTA) rule is thus a more general case of the WTA principle, within which exemplars may be associated with many clusters at differing degrees. This procedure provides a more distributed classification. It was shown in [7] that the modified model can create exemplar and/or prototype
In the original SOFM, the output forms an array (grid) where the neurons are geometrically arranged. The choice of a topological neighborhood $h$ is based on a Gaussian function:

$$i(x) = \arg \min_j \| x - w_j \|, \quad j = 1, ..., n$$

This allows for the detection of the winning unit, and of the topological neighborhood center's position for SOFMs. This is equivalent to minimizing the Euclidian distance between the input vector $x$ and the weight vector $w_j$. If $i(x)$ identifies the closest match for $x$, then the winning unit is determined by the categorical representations, according to the number of $k$ winning units allowed.

In this study, FEBAM will be further enhanced by using $k$WTA and SOFM properties simultaneously. This modification will enable the network to increase its clustering capacity; hence, higher network performance and readability should follow. In addition, this $k$WTA-SOFM version of FEBAM will allow for sparse coding, a distributed representation principle supported by neuropsychological findings [9]. Previous propositions based on FEBAM will now become special cases of this more general model. Therefore, the goal of the study is to propose a general model based on BAMs that shows properties similar to those found in other network types, such as competitive behavior in SOFMs.

### II. THEORETICAL BACKGROUND

#### A. SOFMs and Other General Competitive Models

Competitive models are based on a linear output function defined by:

$$y = Wx$$

where $x$ is the original input vector, $W$ is the weight matrix, and $y$ is the output. Weight connection reinforcement follows an Hebbian learning principle with forgetting term $g(y_j) w_j$, where $g(y_j)$ must satisfy the following requirement:

$$g(y_j) = 0 \text{ for } y_j = 0$$

Therefore, the learning function can be expressed by

$$w_j(p+1) = w_j(p) + \eta(p) y_j x - g(y_j) w_j(p)$$

where $w_j$ represents neuron $j$'s weight vector, $\eta$ represents a learning parameter, and $p$ is the learning trial number. The specific choice of function $g(y_j)$ can lead to two different unsupervised model rules. Equation 3 will lead to Kohonen’s [6] competitive learning rule if

$$g(y_j) = \eta y_j$$

and to a PCA learning rule [11] if

$$g(y_j) = y_j^2 \text{ and } \eta(p) = \eta.$$  

Hence, both PCA and competitive neural networks constitute special cases of Hebbian learning using a forgetting term.

In a WTA perspective, the largest output must be selected.

Using a forgetting term prevents the model’s weight values from growing indefinitely, which makes the model “explode”, such as with strict Hebbian learning [10].

This allows for the detection of the winning unit, and of the topological neighborhood center’s position for SOFMs. This is equivalent to minimizing the Euclidian distance between the input vector $x$ and the weight vector $w_j$. If $i(x)$ identifies the closest match for $x$, then the winning unit is determined by

$$i(x) = \arg \min_j \| x - w_j \|, \quad j = 1, ..., n$$

In the original SOFM, the output forms an array (grid) where the neurons are geometrically arranged. The choice of a topological neighborhood $h$ is based on a Gaussian function:

$$h_{j,i(x)}(p) = \exp \left( \frac{-d^2_{j,i(x)}}{2\sigma^2(p)} \right), \quad d^2_{j,i(x)} = \| r_j - r_i \|^2$$

where $r_j$ defines the position of neuron $j$ in the output array, $r_i$ defines the position of winning neuron $i$, and $\sigma$ represents neighborhood size. Therefore, the connection between input $x$ and winning unit $i(x)$ will be maximally reinforced, and the level of strengthening for the remaining connections will be lower as the distance from the winning unit $d_{j,i(x)}$ increases. The size of the topological neighborhood and the value of the learning parameter on trial $p$ are also required to decrease with time. This is done by using exponential decay:

$$\sigma(p) = \sigma_0 \exp \left( \frac{-p}{\tau_1} \right) \text{ and } \eta(p) = \eta_0 \exp \left( \frac{-p}{\tau_2} \right)$$

where $\tau_1$ and $\tau_2$ represent distinct time constants, and $\sigma_0$ and $\eta_0$ respectively represent the initial values for the neighborhood size parameter $\sigma$ and learning parameter $\eta$ [24].

An extension of the WTA principle selects the $k$ largest outputs from the total $n$ outputs [8]. This $k$-winners-take-all ($k$WTA) rule is thus a more general case of the WTA principle, within which exemplars may be associated with many clusters at differing degrees. This procedure provides a more distributed classification.

### III. FEBAM: THE FEATURE-EXTRACTING BIDIRECTIONAL ASSOCIATIVE MEMORY

Like any artificial neural network, the Feature-Extracting Bidirectional Associative Memory (FEBAM) can be entirely described by its architecture, as well as its output and learning functions.

#### A. Architecture

FEBAM’s original architecture is illustrated in Figure 1. This architecture is nearly identical to that of the...
Bidirectional Heteroassociative Memory (BHM) model proposed by [12]. It consists of two Hopfield-like neural networks interconnected in head-to-toe fashion [13]. When connected, these networks allow a recurrent flow of information that is processed bidirectionally. As shown in Figure 1, the \( W \) layer returns information to the \( V \) layer and vice versa, in a kind of “top-down/bottom-up” process fashion. As in a standard BAM, both layers serve as a teacher for the other layer, and the connections are explicitly depicted in the model (as opposed to Multi-layer Perceptrons). However, to enable a BAM to perform feature extraction, one set of those explicit connections must be removed.

**Bidirectional Heteroassociative Memory (BHM)** model proposed by [12]. It consists of two Hopfield-like neural networks interconnected in head-to-toe fashion [13]. When connected, these networks allow a recurrent flow of information that is processed bidirectionally. As shown in Figure 1, the \( W \) layer returns information to the \( V \) layer and vice versa, in a kind of “top-down/bottom-up” process fashion. As in a standard BAM, both layers serve as a teacher for the other layer, and the connections are explicitly depicted in the model (as opposed to Multi-layer Perceptrons). However, to enable a BAM to perform feature extraction, one set of those explicit connections must be removed.

\[ \forall i, ..., N, y_i(t+1) = \begin{cases} 1, & \text{If } W_{xi}(t) > 1 \\ -1, & \text{If } W_{xi}(t) < -1 \end{cases} \] 

(9)

and

\[ \forall i, ..., M, x_i(t+1) = \begin{cases} 1, & \text{If } V_{yi}(t) > 1 \\ -1, & \text{If } V_{yi}(t) < -1 \end{cases} \] 

\( (\delta + 1)W_{xi}(t) - \delta(W_{xi})_i(t), \text{ Else} \) 

\( (\delta + 1)W_{yi}(t) - \delta(W_{yi})_i(t), \text{ Else} \) 

where \( N \) and \( M \) are the number of units in each layer, \( i \) is the index of the respective vector element, \( y(t+1) \) and \( x(t+1) \) represent network outputs at time \( t + 1 \), and \( \delta \) is a general positive output parameter that should be fixed at a value lower than 0.5 to assure fixed-point behavior [14],[15]. Figure 3 illustrates the shape of the output function when \( \delta = 0.4 \). This output function possesses the advantage of exhibiting continuous-valued (gray-level) attractor behavior (for a detailed example see [12]). Such properties contrast with networks using a standard nonlinear output function, which can only exhibit bipolar attractor behavior (e.g. [16]).

**C. Learning function**

Learning is based on time-difference Hebbian association [10, 12, 14, 17-19], and is formally expressed by the following equations:

\[ W(p+1) = W(k) + \eta(y(0) - y(t))(x(0) + x(t))^T \] 

(11)

and

\[ V(p+1) = V(k) + \eta(x(0) - x(t))(y(0) + y(t))^T \] 

(12)

where \( \eta \) represents a learning parameter, \( y(0) \) and \( x(0) \), the initial patterns at \( t = 0 \), \( y(t) \) and \( x(t) \), the state vectors after \( t \) iterations through the network, and \( p \) the learning trial. The learning rule is thus very simple, and can be shown to constitute a generalization of Hebbian/anti-Hebbian correlation in its autoassociative memory version ([12], [14]). For weight convergence to occur, \( \eta \) must be set according to the following condition [17]:

\[ \eta < \frac{1}{2(1-2\delta)\text{Max}[N,M]}, \delta \neq \frac{1}{2} \] 

(13)

Equations 11 and 12 show that the weights can only
converge when “feedback” is identical to the initial inputs (that is, \( y(t) = y(0) \) and \( x(t) = x(0) \)). The function therefore correlates directly with network outputs, instead of activations. As a result, the learning rule is dynamically linked to the network’s output (unlike most BAMs).

To emulate WTA and \( k \)WTA processes, the perceptual feature extraction (or compression) (\( y \)) layer’s architecture must be further constrained by the addition of inhibitory connections from each “compression” unit to all units from that same layer. The \( W \) connection matrix will be used to memorize the mappings between the input and the associated constrained representation, while the \( V \) connection matrix will be used to extract constrained versions of the input patterns.

As more and more \( y \) units become active (i.e. as we turn to a \( kWTA \)-type setting), the network evolves towards a more distributed kind of representation. It can be seen here that the original FEBAM constitutes a special case of a \( kWTA \) process, where all compression units can be used. Therefore, inhibitory connection values are all set at zero. If the network’s architecture is further constrained using a rectangular 2D topology, the network can behave like a SOFM (Figure 4). A SOFM not only categorizes the input data, it also recognizes which input patterns are close to each other in stimulus space. In order to extend FEBAM into a SOFM, the original topological neighborhood function of the SOFM was extended into a “Mexican-hat” function (Figure 5).

\[
\sigma(p) = \sigma_0 \exp \left( -\frac{p}{\tau} \right)
\]

(15)

where \( \tau \) represents a time constant, and \( \sigma_0 \) represents the initial value for the neighborhood size \( \sigma \).

**IV. SIMULATIONS**

**A. Simulation 1: A case study**

A first simulation was conducted in order to demonstrate the network’s ability to cluster exemplars into categories. For this simulation, artificial pixel-based stimuli were created. Stimuli examples are shown in Figure 6.

---

**Category 1**

![Image of Category 1 patterns]

**Category 2**

![Image of Category 2 patterns]

**Category 3**

![Image of Category 3 patterns]

**Category 4**

![Image of Category 4 patterns]

---

1) **Methodology:** Four category prototypes were produced by generating bipolar-valued vectors, for which the value of each vector position (or “feature”) followed a random discrete uniform distribution. The presence of a feature (black pixel) was represented by a value of +1, and the absence of a feature (white pixel) by a value of -1. Each prototype vector comprised 36 features. 5 exemplars were generated using each prototype, for a total of 20 items. Each exemplar was created by randomly “flipping” the value of between one to four features. The average intra-category correlation was 0.72 and the average inter-category correlation was 0.08.

Learning followed the following procedure:

0. Random weight initializations;
1. Random selection of a given exemplar \((x(0))\);
2. Computation of the activation according to Equation 9.
3. Selection of \( k \) winners.
4. Computation of the initial output (y(0)) according to Equation 9 and Equation 14.
5. Computation of the reconstructed input (x(1)) according to Equation 10.
6. Computation of the compressed output (y(1)) according to Equation 9.
7. Weight updates according to Equations 11 and 12.
8. Repetition of 1 to 7 until the error (‖y(0) − y(1)‖) is sufficiently low (< 0.1).

For the simulation, the learning (η) and transmission (δ) parameters were respectively set to 0.01 and 0.1. Those values respect the requirements described in [12, 17]. The number of y units was set to 100 (10x10 grid), and the initial value of σ was accordingly set to 10 (√100, [24]). Finally, the number of winning units was set to 15, a value that allows for good performance (see Simulation 2).

2) Results: Figure 7 shows the error in function of the number of learning trials. In this particular case, after 236 learning trials, error was lower than 0.1, and therefore learning stopped.

The main difference between the present model and SOFMs is the way weight connections encode the information. In SOFMs, the weight connections represent the input set (Figure 9b), while in FEBAM, the weights are not directly interpretable (Figure 10). Prototypes can be extracted if the initial inputs (Figure 6; x(0)) are iterated until convergence (x(c)); their invariant states are illustrated in Figure 11. As can be seen, each separate category exemplar reaches the same attractor. The network develops four attractors, one for each category. The recurrent nature of the model allows for the development of invariant states. In this case, since the variability within each category is quite high and since the number of output units (100) is low, the network will extract the prototype from the exemplars. However, if one is to store specific exemplars, then the number of output units must be increased\(^2\).

\[^2\] Of course, there are some other ways to encode the desired number of exemplars in a more efficient way. For example a vigilance parameter could be used [5].
Results: As shown by Figure 14a, the distance between weights (dispersion) decreases as the number of winners increases. The graph shows a decrease in variability as the number of winning units increases from 1 to 10, after which it stabilizes. In other words, the density of the cluster increases with the number of winning units. Figure 14b shows the maximum proportion of weights used to encode a given category. The graph shows that the minimum proportion will be close to 25% (theoretical optimality) at around 10 winners and more. With a minimum of 10 winners, the four categories will be encoded using the same proportion of weights.

From the previous results, if 15 units are allowed to win at the same time, the network should, on average, show correct performance. Results indicate that the network will correctly classify a novel pattern 91% of the time. Moreover, in this case, only 15 units are activated at the same time, and increasing that number will not help generalization performance. In other words, a sparse representation seems to achieve better performance than a strict local representation. Moreover, in this case, a distributed representation, like that used in the original FEBAM, constitutes a waste of resources, since the kWTA-SOFM version of the network can achieve the same performance with only approximately 15% of the output units.
C. Simulation 3: Alphanumeric letters

Simulation 1 was replicated, but this time using stimuli with no predetermined categorical (or cluster) membership. The patterns used for the simulations are shown in Figure 15. Each pattern consisted of a 7 x 7 pixel matrix representing a letter of the alphabet. White and black pixels were respectively assigned corresponding values of -1 and +1. Correlations (in absolute values) between the patterns varied from 0.02 to 0.84. Since the network must discriminate between a larger number of patterns than in Simulations 1 and 2 (10 vs. 4), the output grid was increased to 400 units (20x20). The learning parameter $\eta$ was set to 0.001 and the number of winning units was increased to 20 (5% of the output units’ resources). Learning was conducted using the procedure described in the previous section. However, in this case, the minimum error was set to 0.005. This criterion is essential if proper reconstruction in the x layer is desired.

![Fig. 16. Error in function of the number of learning trials](image)

Fig. 16. Error in function of the number of learning trials

Although the “H” pattern seems to be encoded into different regions, Figure 18 indicates that only the upper left region is used to encode that particular pattern. In the y layer, it is clear that only a small portion of the 2-D map is used at one time.

![Fig. 18. y layer output in function of a given input pattern](image)

25
20
15
10
5
0
0
500
1000
1500
2000
2500
N
b
o
f
l
e
a
r
n
i
n
t
r
i
a
l
tr
i
a
l
s

Fig. 17. 2-D topology for weight connections. Each letter represents the best matched pattern for each weight.

1) Results (SOFM properties): Figure 16 shows that the weights converged after approximately 1500 learning trials. Figure 17 shows the best matched pattern for each weight in the V matrix (this is equivalent to a WTA setting). Each pattern is well-separated into a closed, well-delimited region.

![Fig. 17. 2-D topology for weight connections. Each letter represents the best matched pattern for each weight.](image)

2) Results (Autoassociative properties): Since the model is recurrent, it should also preserve its autoassociative memory properties. Figure 19 shows the reconstructed patterns, or outputs of the x layer, which are invariant states in the network. Overall, the reconstructed patterns closely match the original ones (Figure 15); there are some slight differences in the lower right region of letters E and G.

Because the model is recurrent, it can filter noise and complete patterns just like any recurrent associative memory (ex. [20]). Figure 20 illustrates such properties through a noisy recall task, where different examples of patterns are contaminated with various types of noise (random pixels flipped, addition of normally distributed random noise, specific portion of a given pattern removed).

![Fig. 20. Noisy recall of different patterns after 20 cycles through the network. The first pattern is the letter “A” for which 6 pixels have been randomly flipped. For the second pattern, a random noise vector X~N (0, 0.5) has been added to the letter “F”. The third pattern is the letter “B”, for which the upper portion pixels have been set to zero. The fourth pattern is the letter “J” for which the right portion pixels have been set to zero.](image)

V. DISCUSSION

FEBAM is able to reproduce properties of SOFMs while keeping the properties of a recurrent associative memory. Being able to introduce a topological mapping and k-winner-take-all properties simultaneously into a BAM-like architecture represents a scientific achievement by itself. The
recurrent architecture allows for the simultaneous development of prototypes in one layer, and of clusters in the other layer. This new property adds to the many other BHM/FEBAM framework properties, such as aperiodic recall, many-to-one association, multi-step pattern recognition, blind source separation, perceptual feature extraction, input compression and signal separation ([4], [12], [17], [21], [22]). In short, the network replicates various behaviors shown by recurrent associative memories, principal component analysis and self-organizing maps while keeping the same learning and output functions, as well as its general bidirectional heteroassociative architecture. This result constitutes a step toward unifying various classes within a general architecture.

Parameters are constrained in function of the dimensionality of the inputs (learning parameter, \(\eta\)), the analytic results (transmission parameter, \(\delta\)), the size of the topology (neighborhood size of the “Mexican-hat”, \(\sigma\)) and its decreases (time constant, \(\tau\)). However, this time-dependent function limits the model’s ability to adapt to new patterns that are not part of the original set of patterns. This problem is not unique to this model but to the majority of SOFMs [5].

One solution would be the inclusion of a vigilance procedure [5], similar to that of distributed associative memories [23]. Further studies should also explore how basins of attraction can be optimized in function of the number of categories to be developed, and within-category correlations. In addition, quantitative comparison should be made between the model and its BAM version.

Also, since, in a neuropsychological framework, the presence of local connections is more probable than that of distal connections, specific further investigations should focus on the interaction between \(kWTA\) and SOFMs within the FEBAM framework, concentrating more specifically on the role of the topological neighborhood, the number of active units and the encoding representation. Finally, further studies will also be focused on the extent to which this framework can reproduce other kinds of human learning behaviors.

VI. REFERENCES


