A Chaotic Bidirectional Associative Memory

Sylvain Chartier\textsuperscript{1,2} & Mounir Boukadoum\textsuperscript{3}

\textsuperscript{1}Université du Québec en Outaouais, \textsuperscript{2}Institut Philippe Pinel de Montréal
\textsuperscript{3}Université du Québec à Montréal
chartier.sylvain@courrier.uqam.ca
boukadoum.mounir@uqam.ca

Abstract

A new BAM model is presented that uses a chaotic output function operating in chaotic mode during recall. Our results show that the model develops well-defined attractor basins, with the result that our chaotic BAM is more tolerant to noise than a regular fixed point BAM. This is concluded from simulations that showed the superior performance of the new model when compared to the original BAM architecture or when using a linear technique such as the pseudo-inverse.

1. Introduction

Over the past years, nonlinear dynamics and chaotic behavior have become an essential tool for understanding low and high-level brain mechanisms [1]. Initially, chaos in the brain was thought to be simply a factor of disorder. Today, it is viewed more and more as an essential component of normal brain function [2].

Since Freeman’s paper on chaotic behavior in the olfactory system [3] many models have been proposed to explain or exploit chaos in neural systems. They include microscopic models that use simple artificial neurons to mimic the chaotic behavior found in real neurons [4] and macroscopic models that respond with flexibility to a large range of behavior. Research on the latter has mainly dealt with recurrent associative memories [5]. Few models exist that use chaos in bidirectional associative memories [6, 7] and they rely on either complex architectures or high-dimensional systems.

Recently, we introduced a BAM model, with a nonlinear dynamic output function, that uses simpler principles while having the same general properties [8, 9]. The model builds on the standard BAM architecture. Its learning rule uses simple time-delayed Hebbian correlation matrices and its output function derives from the classic Verhulst’s finite-difference equation. By using the output function in its non chaotic region, we were able to generate stable fixed-point attractors for both binary and gray level stimuli. In this paper, we investigate the use of our BAM model with its output function set to operate in the chaotic region of its bifurcation diagram.

In the balance of this article, we start by describing our chaotic model, and then show experiments to explore its behavior after the onset of chaos. The article concludes with a discussion and conclusion.

2. Model description

The topology and learning rule of the model have been described elsewhere [9] and are summarized below. As for the output function, it is a modified version of the one described in the same reference and, therefore, it will be described in more details.

2.1. Topology

Figure 1 shows the topology of our BAM model. It shows to Hopfield-like neural network interconnected in a way that allows the recurrent flow of information in a bidirectional fashion.

In Figure 1, \(x_{00}\) and \(y_{00}\) are the initial vectors-states, \(W\) and \(V\) are the weight matrices in each direction, and \(t\) specifies the current iteration number. Notice that the vectors composing the pairs to be learned need not be of the same dimensions and, contrary to typical BAM designs, weight matrices \(W\) and \(V\) need not be the transposes of one another.

![Network topology](image)

Figure 1. Network topology

2.2. Learning rule

The learning rule is derived from a Hebbian/anti-Hebbian approach and is expressed as:

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

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

(1b)

where \( \eta \) 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 \( x[t] \) and \( y[t] \). This enables the network to learn online and contributes to the convergence of the weight connections.

Using binomial identities, the learning rule can also be written as:

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

(2a)

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

(2b)

The previous equations show that the weight matrices converge only when \( y[0] = y[0] \) and \( x[0] = x[0] \), i.e. when the feedbacks are the same as the initial inputs.

### 2.3. Output function

Recently, we introduced a BAM model, with a nonlinear dynamic output function, that uses simpler The original output function of the network was derived from the classic logistic growth equation and was defined as follows [9]:

\[ \forall i, a_i \begin{cases} 1, \text{ if } a_i > \frac{1 + \delta}{\delta} \\ -1, \text{ if } a_i < -\delta \\ \frac{(\delta + 1)a_i - \delta a_i^3}{\delta}, \text{ Else} \end{cases} \]  

(3a)

\[ \forall i, b_i \begin{cases} 1, \text{ if } b_i > \frac{1 + \delta}{\delta} \\ -1, \text{ if } b_i < -\delta \\ \frac{(\delta + 1)b_i - \delta b_i^3}{\delta}, \text{ Else} \end{cases} \]  

(3b)

In the previous equations, \( \delta \) is a parameter that dictates the dynamic behavior of the outputs, \( a_i \) and \( b_i \) are the usual activation function \( a_i = Wx_i \) and \( b_i = Vy_i \).

Figure 2 illustrates the shape of the output function for \( \delta = 0.4 \) and Figure 3 provides its bifurcation diagram as a function of \( \delta \) in the case of binary inputs. We see that Equations 1a and 1b leads to stable binary attractors for values of \( \delta < 1 \). We have also shown that the output function leads to stable attractors for gray-level inputs [9].

In this work, we defined the output function as follows:

\[ \forall i, a_i \begin{cases} 1, \text{ if } a_i > \frac{1 + \delta}{\delta} \\ -1, \text{ if } a_i < -\delta \end{cases} \]  

(4a)

\[ \forall i, b_i \begin{cases} 1, \text{ if } b_i > \frac{1 + \delta}{\delta} \\ -1, \text{ if } b_i < -\delta \end{cases} \]  

(4b)

The new bounds were chosen so that the cubic map underlying the output function covers all the quadrants of a hypothetical hypercube with no overlapping of values. Figure 4 illustrates the new shape of the output function for \( \delta = 0.4 \) and Figure 5 provides its bifurcation diagram as a function of \( \delta \) in the case of binary inputs.
3. Simulations

We performed simulation to investigate the effect of operating the output function in its chaotic mode during recall. To that end, we set the value of $\delta$ to 0.1 during training and 1.5 during recall. Figure 5 clearly shows that those values correspond to fixed-point and chaotic behavior of the output function. Two simulations were performed, one aimed at investigating general network properties and the other aimed at investigating BAM performance.

3.1. 2-Dimensional bipolar patterns

The simulation consisted of learning two 2-dimensional bipolar stimuli: [1, 1] and [-1, 1]. The learning parameters were $\eta = 0.01$ and $\delta = 0.1$, and the number of learning trials was set to 500. During recall the value of $\delta$ was changed to 1.5 to operate within chaos. Figure 6 shows the obtained phase diagrams for each of the input vectors’ components when using positive inputs. They show the behavior of a quadratic map as one would expect.

![Phase diagrams](image)

Figure 6. Phase diagrams

Figure 7 shows the output variations within the basins of attraction for different random input. Contrary to what would be expected from a regular BAM network, the attractors are not the corners of a square but regions clearly delimited within quadrants. Thus, although the network behavior is chaotic, no boundary crossing occurs while recalling an input vector.

![Output variations](image)

Figure 7. Output variations within the basin of attraction.

3.2. 49-Dimensional bipolar patterns

The simulation consisted of associating two groups of 4 and 10 pairs of letters, respectively. The letters were encoded in 7x7 pixel matrices, for a memory load of 4/49 for the first group and 10/49 for the second. The letters to be associated are shown in Figure 8. The learning parameters were $\eta = 0.01$ and $\delta = 0.1$, and the number of learning trials was set to 2000. As before, the value of $\delta$ was changed to 1.5 during recall to operate within chaos. The input patterns were corrupted by randomly flipping the pixels from 2 to 31 % (1 to 15 pixels). We computed the network’s output in each case and compared it with the original pattern.

![Bipolar pattern pairs](image)

Figure 8. Bipolar pattern pairs to be associated.

The network performance (proportion of accurate recall) was tested on different percentage of pixels flipped. An example of different percentage of pixels flipped is given at the Figure 9.

![Percentage of pixels flipped](image)
To allow for comparisons, the chaotic BAM network was compared with the original Kosko BAM (KBAM) network [10], and using the well known pseudo-inverse technique \( W = X(X'X)^{-1}Y' \) [11]. All the models use the signum function for their output function except the chaotic model. Figures 10 and 11 show the clear superiority of the chaotic model over the others, in particular the KBAM.

Finally, Figure 12 illustrated a recall example of the letter “d” with a percentage of pixels flipped of 16\% (8/49). It is shown that although the network never output the same value twice, nevertheless it is always constrained to its quadrant (Figure 13).

4. Discussion and conclusion

As shown by the preceding results, the new BAM can produce chaotic behavior that remains within bound and separate basins of attractions. Thus, it is possible to restrain chaos to specific quadrants in network space while preserving the BAM associative properties. Also, the network performs better than Kosko’s BAM with Hebbian learning and has a similar performance to using the pseudo-inverse matrix for low percentage of pixels flipped.

Our results agree with the idea that chaos may play an important role in brain dynamics, even if not necessarily present at all stages of the cognitive processes. Further studies should investigate how the learning function can be modified to enable learning from chaotic inputs.

10. References


