# Using a Bidirectional Associative Memory and Feature Extraction to model Nonlinear Exploitation Problems

**Kinsey Church (kchur026@uottawa.ca)**
School of Psychology, 136 Jean-Jacques Lussier, Vanier Hall
Ottawa, ON, K1N 6N5, CAN

**Matt Ross (mross094@uottawa.ca)**
School of Psychology, 136 Jean-Jacques Lussier, Vanier Hall
Ottawa, ON, K1N 6N5, CAN

**Sylvain Chartier (sylvain.chartier@uottawa.ca)**
School of Psychology, 136 Jean-Jacques Lussier, Vanier Hall
Ottawa, ON, K1N 6N5, CAN

**Abstract**

Each day we are faced with a decision of maximizing our resources by using our current knowledge to learn new things. Should we go to the new restaurant that just opened around the corner or stick to an old, reliable favourite? This is known as the exploration-exploitation dilemma and it is at the heart of reinforcement learning. The present study looks at the exploitation half of this problem and aims to implement it in a biologically plausible recurrent associative memory model. In the framework of Artificial Neural Networks, exploitation is observed when the network can iterate through many learned responses and stabilize on the correct one to solve a given task. This is a process akin to being able to switch from a cyclic to a point attractor. More precisely, Bidirectional Associative Memory (BAM) is used to accomplish such tasks where the context dictates which attractor the network should converge to. For simple independent tasks, the BAM is sufficient. However, for overlapping tasks, the task becomes nonlinearly separable. Therefore, the BAM needs an extra unsupervised layer to extract unique features from the inputs. These features combined with input are then sent to BAM where it can learn the different attractors adequately. This network was able to stabilize on the correct responses of tasks that involved time series of varying lengths, overlap, and levels of correlation; the variability one would expect from the real world.

**Keywords:** exploitation/exploration; recurrent associative memory; one-to-many associations.

## Introduction

In order to increase their chances of survival, many organisms have evolved various complex mechanisms to solve problems, make decisions, and learn about the environment around them. These mechanisms come with several different trade-offs, where increasing proficiency in one area may result in a lack of ability in another area. One of these problems in cognition is the exploration-exploitation dilemma (Hills, Todd, Lazer, Redish, & Couzin, 2015). This is when an organism must choose whether it wants to employ something it has already learned (exploitation) or explore its environment to learn new things (exploration). For example, when a young child is first learning how to open different doors. They may start by trying the few ways they already know to open a door (exploitation), such as pulling and pushing. If these behaviours fail, however, they will have to start looking for new solutions (exploring). Understanding exactly how animals and humans switch between these two processes and how each of them works is crucial to understanding how we adapt, learn, behave, and survive (Hills et al., 2015). This trade-off is one of many phenomena affected by Reinforcement Learning (RL).

Previous studies have proposed several methods to solving this problem from an "optimal solution" perspective. For example, Hwang, Chiou and Wu (2003) created an Artificial Neural Network (ANN; an evaluation predictor) that solves the exploration-exploitation problem that was able to both explore and exploit to find an optimal solution to their task (balancing a pendulum). They used the mean of a normal distribution to represent the exploitation function, which may be a bit too basic to capture the exploitation process. This study and many others focus too much on obtaining a desired state rather than the underlying process of how exploitation (and exploration) take place or attempt to find a biologically plausible solution (Cohen, McClure, & Yu, 2007; Tilahun, 2019).

Another promising study was conducted by Lew, Rey, and Zanutto (2013). It outlines a biologically plausible model of switching between exploration and exploitation depending on changes in the environment and rewards from the dopaminergic system. Their model is a computational framework of how exploration and exploitation are selected and relates this process back to the reward system in the human brain, focusing on the selection only. Many studies have a similar focus and are more about the overall model rather than how each process works (Gershman & Niv, 2015; Hallquist & Dombrovski, 2019).

An important consideration overseen by these studies is the context in which the decision takes place. Empirical studies have shown how the environment is an important factor in predicting how an animal will behave and how it will exploit its resources (Naruse et al., 2018). Some studies have shown how the decisions humans and animals make influence their ecosystem and vice versa, from small-scale interactions

among individual agents to large-scale adaptive systems, like population dynamics (Monk et al., 2018). The context in which an animal is found clearly plays a role on how it chooses to exploit and explore.

The present study looks at the exploitation half of the dilemma using environmental context as a cue. For instance, an animal faces the exploration-exploitation dilemma while foraging for food (Hills et al., 2015). Is it more efficient to explore the area for new sources of food or stick to a learned set of locations where food has been found in the past?

The most well-known example of exploitation has been in Q-Learning (Watkins & Dayan, 1992). In this implementation, the best outcome is simply the one that has the maximum possible reward; max(Q). This works best when the reinforcement is non-binary. However, in the situation of multiple possible outcomes, the agent should be able to test different possible behaviours until the correct one has been found. For example, if the animal chooses to exploit their already-known resources, they cycle through them until the problem is solved (they find food). Once it is solved, they will stop trying different behaviours and focus on the successful one.

In ANNs, the problem can be framed as how the network can generate a different output (behaviour) when the input (context) remains the same? In ANNs, if a given input is sent to the network, it will always give the same associated output. However, if that output is not satisfying, the network must give a different output even though it is still the same input that is presented. One way to solve this problem is to use a cycle attractor. This process can be modeled by the network generating various different outputs from the same input to represent the different learned behaviours. For example, input 1 generates output 1, which generates output 2, all the way until output *n*. Output *n*, the final output, is then associated with the output 1, creating a loop of learned patterns. This would be like a squirrel having a series of berry bushes where it knows food can be found and going from bush to bush until it finds some.

This can be easily implemented in a Bidirectional Associative Memory (BAM) as a multistep time series (Chartier & Boukadoum, 2006). However, a problem arises when the network must stabilize on a specific output *i*. Therefore, each of the patterns (output) must also be stored as point attractors in the network. This represents a one-to-many situation where for the same output *i* the network must generate output *i*+1 (next pattern; cyclic attractor) or output *i* (same pattern; point attractor). See Fig. 1 for a diagram of cyclic and point attractors.

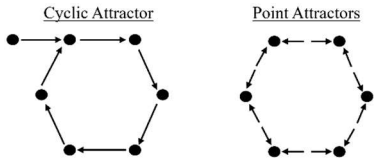


Figure 1: Illustration of cyclic and point attractors.

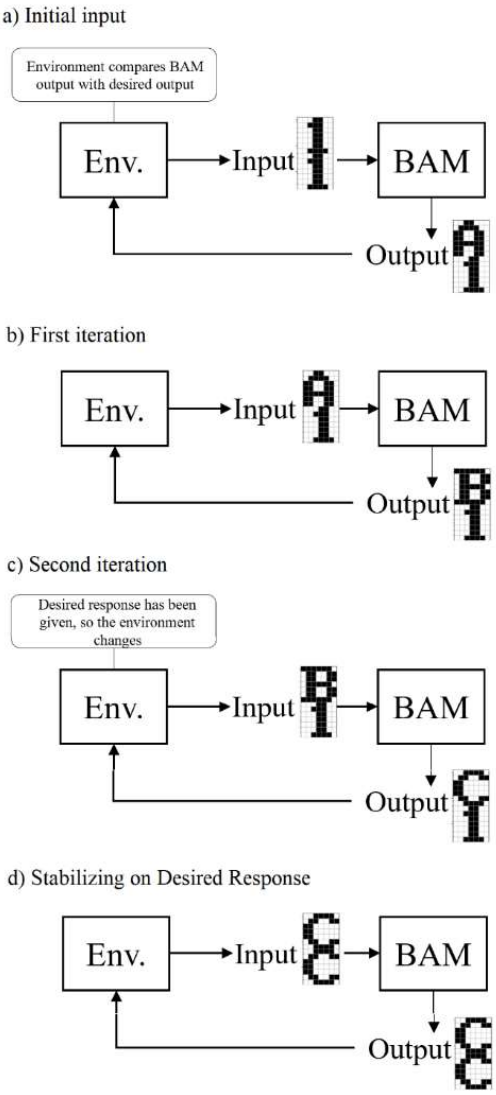


Figure 2: Flowchart illustrating the switching from a cyclic attractor to a fixed-point attractor using the environment.

The solution may reside in the inclusion of context that can dictate which state the network should be in and therefore give distinctive features. From the feedback given by the environment, the network should be able to switch between a cyclic (repeating series of outputs) and a point attractor (single output). However, instead of using context from the time series itself (Elman, 1990), the context has to be extracted from the reading of the environment; a process proposed in (Jordan, 1997) and successfully implemented in BAM (Rolon-Mérette, Rolon-Mérette, & Chartier, 2019). By having a unique context as the initial input of the time series, it can also be used to distinguish multiple cyclic attractors (different time series).

The BAM can be used to store various attractors including point, cyclic, and chaotic attractors (Chartier, Boukadoum, & Amiri, 2009), making it a well-suited candidate to perform exploitation. Fig. 2 shows an example of both cyclic

attractors (a to c) and fixed-point attractors (d) in the model. When the BAM is giving the next output, it is recalling associations in a cycle. In this case, it recalls 'A' from '1' and proceeds in order ('B', 'C', etc.). However, when it needs to stabilize, it switches to a fixed-point attractor. In this example, the environment gives 'C' associated with itself as a response, which is different from 'C' in the '1' context, causing the BAM to switch from a cyclic attractor to a fixed-point attractor. In order to focus exclusively on the exploitation phase, it is assumed that some level of exploration has already taken place. In other words, that series of behaviours have already been learned.

Putting Fig. 2 in the context of the foraging example, '1' could represent hunger, and the other patterns represent learned behaviours, such as searching for food in different locations; 'A', 'B', 'C', etc. When food is located, the environment lets the animal know to stop cycling through its learned behaviours (checking different locations) and make the decision to stay there.

The remainder of this paper is divided into two separate sets of simulations. Simulation I introduces the problem of modeling exploitation and switching from iteration to stabilizing on a given response using the environmental context (one-to-many association). Simulation II solves more complex and diverse situations involving nonlinearly separable associations. An overall discussion is provided at the end.

## Simulation I- Independent Time Series

### Methods

To model exploitation, time series of patterns were used to represent different possible output behaviours. Each series was preceded by a unique contextual input. The network task consisted of associating multiple time series of different lengths and various levels of overlap. Moreover, the level of correlation between patterns varies vastly to encapsulate the variability found in the real world.

**Stimuli** Patterns of behaviours were represented by a series of letters and the context by a number. Each pattern was placed on a 7x7 grid where a black pixel had a value of +1 and a white pixel a value of -1. Following the procedure in Rolon-Mérette et al. (2019) for each pattern of the time series, the associated context pattern was appended. These contexts allow the network to differentiate between different time series and the desired type of attractor (cyclic vs. point). The various time series are illustrated in Fig. 3, with grey boxes to highlight the repeated and overlapping patterns. Series 1 is a single time series of patterns associated with the context of number '1' and each of the patterns are also associated with themselves. It represents a basic, independent time series with no overlap and is the least complex series. Series 2 increases the difficulty by having two independent time series of various lengths. Series 3 includes three

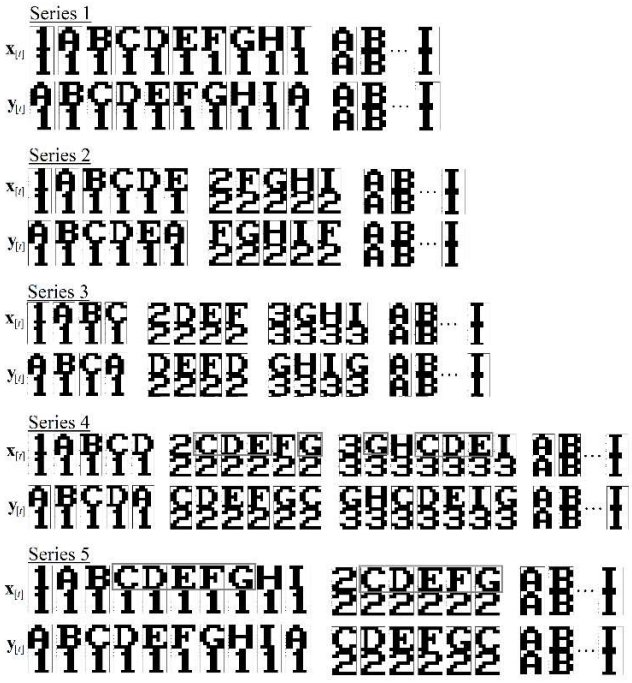


Figure 3: Stimuli for Simulations I and II.

independent time series of various lengths as well. Series 4 and 5 represent situations closer to reality, where the difficulty is further increased. In both examples, a given pattern is associated not only to two settings (the cyclic and the point attractor) but to four (series 4) or three (series 5) settings. Series 4 shows a setting where patterns can belong to various solutions (overlapping), while series 5 shows a series that is a subtype of the other. In both cases, the problem is no longer linearly solvable.

**Architecture** The architecture for the modified BAM (Chartier et al., 2006) is made of two Hopfield-like neural networks interconnected in a head-to-tail fashion to create a bidirectional flow of information (see Fig. 4). The initial vector-states are represented by $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$, the weight matrices by **W** and **V**, the dimensionality by $m$ and $n$, and $t$ represents the time (current iteration number).

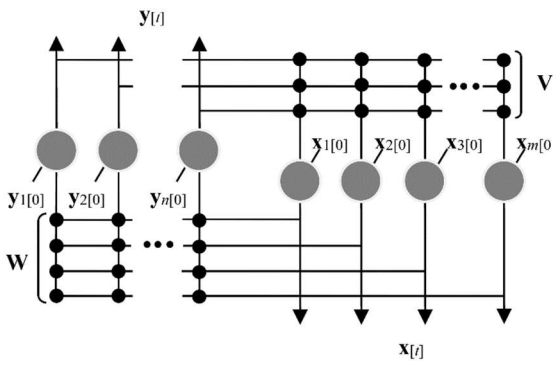


Figure 4: Architecture of the BAM.

**Output Function** The transmission is composed of the activation function that is then fed through the output function. The activation is obtained the usual way:

$$\mathbf{a}_{[t]} = \mathbf{W}\mathbf{x}_{[t]} \tag{1a}$$

$$\mathbf{b}_{[t]} = \mathbf{V}\mathbf{x}_{[t]} \tag{1b}$$

Once this linear activation is obtained it goes through the nonlinear output function defined as (2a) and (2b):

$$\forall i,\ldots,n,\ \mathbf{y}_{i[t+1]} = f\left(\mathbf{a}_{i[t]}\right) = \begin{cases} 1, & \text{if } \mathbf{a}_{i[t]} > 1 \\ -1, & \text{if } \mathbf{a}_{i[t]} < -1 \\ (\delta+1)\mathbf{a}_{i[t]} - \delta\mathbf{a}_{i[t]}^3, & \text{else} \end{cases} \tag{2a}$$

$$\forall i,\ldots,m,\ \mathbf{y}_{i[t+1]} = f\left(\mathbf{b}_{i[t]}\right) = \begin{cases} 1, & \text{if } \mathbf{b}_{i[t]} > 1 \\ -1, & \text{if } \mathbf{b}_{i[t]} < -1 \\ (\delta+1)\mathbf{b}_{i[t]} - \delta\mathbf{b}_{i[t]}^3, & \text{else} \end{cases} \tag{2b}$$

where $n$ and $m$ are the number of units in each layer of the network, $i$ is the index unit, $\delta$ is the transmission parameter, and **a** and **b** are the activations. The output is a cubic function with saturating limits at ±1.

**Learning Function** The connection weights are modified following a Hebbian/anti-Hebbian rule (Bégin & Proulx, 1996; Storkey & Valabregue, 1999) and can be represented as:

$$\mathbf{W}_{[k+1]} = \mathbf{W}_{[k]} + \eta(\mathbf{y}_{[0]} - \mathbf{y}_{[t]})(\mathbf{x}_{[0]} + \mathbf{x}_{[t]})^{\mathrm{T}} \tag{3a}$$

$$\mathbf{V}_{[k+1]} = \mathbf{V}_{[k]} + \eta(\mathbf{y}_{[0]} - \mathbf{y}_{[t]})(\mathbf{x}_{[0]} + \mathbf{x}_{[t]})^{\mathrm{T}} \tag{3b}$$

where **W** and **V** represent the two sets of weight connections, $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$ the initial inputs, $\eta$ the learning parameter, and $k$ the current learning trial. The $\eta$ is calculated using the following inequality function to guarantee that the learning will converge (Chartier et al., 2006):

$$\eta < \frac{1}{2(1-2\delta)\mathrm{Max}[m,n]}, \delta \neq \frac{1}{2} \tag{4}$$

**Parameters** To ensure all associations are stable, delta ($\delta$) was set to 0.2, number of iterations ($t$) to 1 and learning rate ($\eta$) was set to 0.008 (equation 4). The minimum Mean Squared Error (MSE) was $10^{-8}$ and the maximum amount of learning trials was set to 1000 (in case of non-convergence). The weights **W** and **V** were initialized at 0.

**Learning Procedure**

1. Selection of a series (Fig.3).
2. Selection of an associated pair as the initial input for $\mathbf{x}_{[0]}$ and $\mathbf{y}_{[0]}$.
3. Computation of the outputs $\mathbf{x}_{[1]}$ and $\mathbf{y}_{[1]}$ using equations 1 and 2.
4. Weights update (**W** and **V**) according to equation 3.
5. This was repeated with each of the subsequent patterns until the MSE for all associations reached $10^{-8}$ or the number of trials reached 1000.

**Exploitation Procedure**

1. Selection of a "desired response" for the network to stabilize on for a given time series ("initial context"). This is the response stored in the environment for comparison.
2. First pattern in the time series is used as the input for BAM ($\mathbf{x}_{[0]}$).
3. Environment compares output ($\mathbf{y}_{[t]}$) with the "desired response" for each iteration. If it does not match, the network uses this pattern as the new input ($\mathbf{x}_{[t]}=\mathbf{y}_{[t]}$) and generates the next one ($\mathbf{y}_{[t+1]}$). If it matches (BAM has found the correct pattern), the environment sends only this pattern back to the BAM. This can be seen as the environment changing when the solution to a task is found (for example, when you are trying to open a door and it opens vs. remaining closed), providing enough feedback to stabilize on this correct response.
4. The BAM recalls the pattern associated with itself and stabilizes on it instead of continuing to iterate through the series.

## Results

The BAM was able to learn and recall the first three series. It was also able to stabilize its output on any desired pattern with perfect accuracy. For example, in Fig. 5 we see one trial in series 1, where 'F' was the correct response in time series '1'. The initial input was '1' and it was given to the BAM, which recalled 'A'. This output was compared to the desired response in the environment, and since it was not a match, the output became the new input and was sent back to the BAM. This continued until the correct response 'F' was given, where the environment changed because the task had been solved and sent 'F' both as input and context, to the BAM.

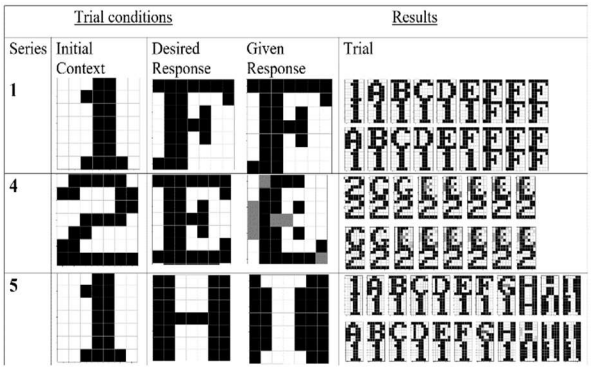


Figure 5: Results of example trials from Simulation I.

The network was then able to give the same output, therefore causing it to stabilize on the correct response.

However, when the BAM was faced with more complex tasks (overlaps or subtype), it could no longer learn the associations properly and stabilize on the desired response. For example, when encountering a nonlinear task as seen in series 4 and 5, the results are noisy, and the BAM often fails to learn all the time series properly. It also cannot stabilize on the correct response. This is due to the nature of the task that necessitates a nonlinear classification, a capacity lacking in a single layer BAM (Chartier & Boukadoum, 2006). Therefore, the next simulation solves this problem by adding an unsupervised layer to the BAM (Rolon-Mérette, Rolon-Mérette, & Chartier, 2018).

## Simulation II- Overlapping Time Series

### Methods

As indicated by the results from Simulation 1, a BAM alone using contexts is not sufficient to solve problems of higher complexity, meaning it cannot solve real-world problems. Previous works have shown that a Feature Extraction Bidirectional Associative Memory (FEBAM) can be used in combination with the BAM in order to solve nonlinearly separable tasks (Tremblay, Myers-Stewart, Morissette, & Chartier, 2013). Therefore, the architecture of the network has to be modified accordingly. The new flowchart is shown in Fig. 6.

**Stimuli** All series from Fig. 3 were used with the addition of one additional complex series (Fig. 7). This new series includes a mix of all possible combinations of multiple, overlapping and subtype time series; representing the kind of variability found in nature. It encompasses all possible combinations and arrangements of stimuli. Fig. 8 illustrates these combinations, using a Venn diagram for clarity.

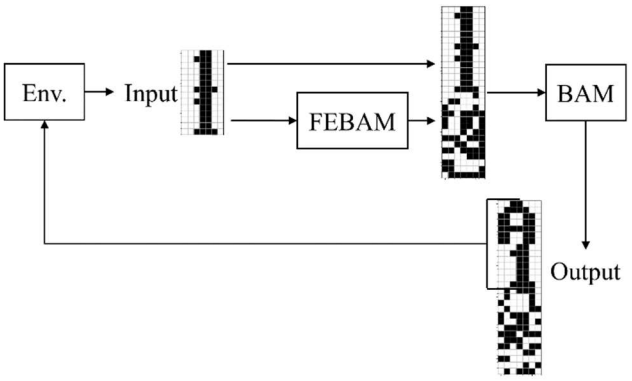


Figure 6: Flowchart for Simulation II, showing unique representation from FEBAM being appended to stimuli.


Figure 7: Added stimuli for Simulation II.

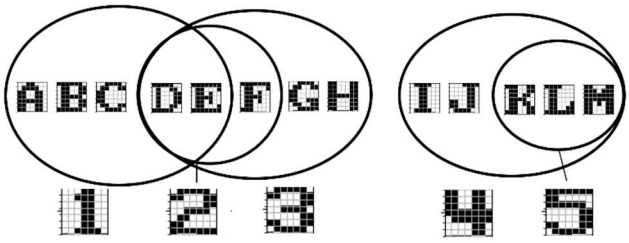


Figure 8: Venn diagram illustrating which time series are independent, subtype, or overlap in series 6.

**Architecture** The FEBAM is the unsupervised version of the BAM. The only difference between the two models is the absence of a set of external connections; the $\mathbf{y}_{[0]}$ inputs (see Fig. 4). This means there is only one explicit connection, $\mathbf{x}_{[0]}$, making the learning unsupervised.

**Output Function and Learning Function** The transmission is obtained the same way as the BAM. Regarding the learning, because the $\mathbf{y}_{[0]}$ connections are no longer available, they are obtained by iterating once through the network as illustrated in Fig. 9. The weights of the FEBAM are then updated exactly as the BAM (equations 3a, 3b).

**Parameters** For both BAM and FEBAM, the transmission parameter ($\delta$) was set to 0.2 to ensure the fixed points would be stable. The learning parameter ($\eta$) was set to 0.004. The maximum amount of learning trials was set to 1000, and the minimum required MSE was $10^{-8}$. The weights of the FEBAM were randomly initialized between -2 and 2 (Tremblay et al., 2013).

**Learning Procedure**

1. Selection of a series (Fig. 2 and 6).
2. Selection of an associated pair as the initial input for the FEBAM.
3. The output of the FEBAM is then appended to the initial input and sent to the BAM (Fig. 6).
4. The BAM outputs the next associated patterns.
5. Weights update for both the FEBAM and BAM.
6. This was repeated with each of the subsequent patterns until the MSE for all associations reached $10^{-8}$ or the number of trials reached 1000.

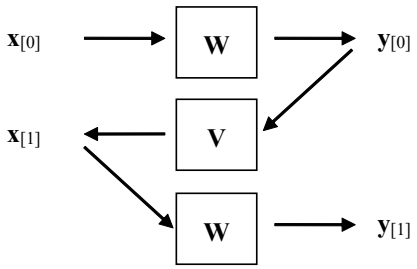


Figure 9: Output iterative process used for the FEBAM learning.

**Exploitation Procedure**

1. Selection of a "desired response" for the network to stabilize on for a given time series ("initial context"). This is the response stored in the environment for comparison.
2. First pattern in the time series is an input for FEBAM.
3. The output of the FEBAM is then appended to the initial pattern and sent to the BAM.
4. Environment compares output of the BAM (**y**[*t*]) with "desired response" for each iteration *t*. If it does not match, the network uses this pattern as the new input (**x**[*t*]=**y**[*t*]) and generates the next one (**y**[*t*+1]). If it matches (BAM has correctly solved the task), the environment sends only this pattern back to the FEBAM (then to the BAM).
5. The BAM recalls the pattern associated with itself and stabilizes instead of continuing to iterate through the series.

## Results

As shown in Fig. 10, the FEBAM-BAM model was able to learn and perfectly recall both time series 4 and 5 with no problems, which the BAM alone was not able to do. It could also stabilize on any given response without issue. In the example shown for series 4 (Fig. 10), the desired response for that trial was 'E' in the '2' time series. The first input for that time series, '2', was given as the initial input and sent through the FEBAM, where the unique signature for that pattern was generated and appended to it. This new stimulus was then entered as input for the BAM, which gave 'C' as an output. That output was compared to the desired response in the environment and was sent back to the FEBAM until the desired response, 'E' was given. This desired response 'E' was then sent, associated with itself, as the new input for FEBAM, causing the BAM to stabilize on the correct answer. Most importantly, the model was able to successfully stabilize on any given response in series 6, which contained all possible combinations of interaction and correlation between stimuli: subtype, independent time series, and overlap. This model was successfully able to iterate through all series and stabilize on a given response.

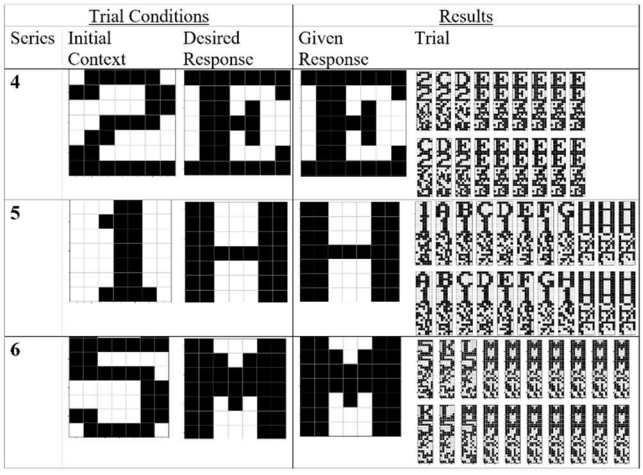


Figure 10: Results of example trials from Simulation II.

## Discussion and Conclusion

To further our understanding of Reinforcement Learning and the exploration-exploitation trade-off, the exploitation phase was implemented using a BAM. Here it was assumed that a given set of patterns were already encoded for various contexts. The network's job was then to iterate through a given series and stabilize on the desired response using feedback from the environment. Various levels of complexity were tested ranging from a single time series all the way to the same variability found in a natural setting. When the complexity was too difficult for a single BAM (nonlinearly separable cases), a FEBAM was included in order to generate unique representations for each pattern. This combination of networks was sufficient to solve any type of situation.

Furthermore, by adding layers of FEBAM-BAM, the point attractors could become the context of a new time series and allow chains of time series; something akin to chunking (Gobet et al., 2001).

One limitation of this study is that it represents exploitation under very specific circumstances. For example, if one is trying to solve a problem with a series of learned responses, they will learn which of the responses are helpful and which are not. Therefore, the order of the patterns should reflect the probability of success, a phenomenon well-captured by standard Q-Learning. Changing the order of some items without retraining the whole dataset is a challenging avenue in a distributed associative memory and should be addressed in future studies. This could be implemented using an additional BAM to store "correct" responses in function of the desired new order with a novel correlated context generated by the network itself or by switching to more interesting attractors such as aperiodic ones (Tsuda, 2001).

The current network could also be added to a model of exploration to study the exploration-exploitation trade-off. Finally, temporal aspects should be taken into consideration, allowing for the timing of when actions should be accomplished, easing the transition from numerical simulation to real-time neurorobotic implementation.

In conclusion, by using a BAM, we were able to model the exploitation phase of the exploration-exploitation dilemma,

where the subject iterates through different learned responses and can stabilize on the correct response based on feedback from the environment. By adding a FEBAM to generate a unique representation for each learned stimulus, we were able to model this with all of the complexity of a real-world setting, including different lengths of stimuli, different levels of correlation, and nonlinear problems. Being able to model exploitation is a crucial part of understanding our own cognition and how we learn from the dynamic world around us.

## References


Bégin, J., & Proulx, R. (1996). Categorization in unsupervised neural networks: The Eidos model. *IEEE Transactions on Neural Networks*, *7*(1), 147–154. https://doi.org/10.1109/72.478399

Chartier, S., & Boukadoum, M. (2006). A Sequential Dynamic Heteroassociative Memory for Multistep Pattern Recognition and One-to-Many Association. *IEEE Transactions on Neural Networks*, *17*(1), 59–68. https://doi.org/10.1109/TNN.2005.860855

Chartier, S., Boukadoum, M., & Amiri, M. (2009). BAM learning of nonlinearly separable tasks by using an asymmetrical output function and reinforcement learning. *IEEE Transactions on Neural Networks*, *20*(8), 1281–1292. https://doi.org/10.1109/TNN.2009.2023120

Cohen, J. D., McClure, S. M., & Yu, A. J. (2007). Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *362*(1481), 933–942.

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, *14*(2), 179–211.

Gershman, S. J., & Niv, Y. (2015). Novelty and Inductive Generalization in Human Reinforcement Learning. *Topics in Cognitive Science*, *7*(3), 391–415.

Gobet, F., Lane, P. C. R., Croker, S., Cheng, P. C. H., Jones, G., Oliver, I., & Pine, J. M. (2001, June 1). Chunking mechanisms in human learning. *Trends in Cognitive Sciences*, *5*, 236–243. https://doi.org/10.1016/S1364-6613(00)01662-4

Hallquist, M. N., & Dombrovski, A. Y. (2019). Selective maintenance of value information helps resolve the exploration/exploitation dilemma. *Cognition*, *183*, 226–243. https://doi.org/10.1016/j.cognition.2018.11.004

Hills, T. T., Todd, P. M., Lazer, D., Redish, A. D., & Couzin, I. D. (2015). Exploration versus exploitation in space, mind, and society. *Trends in Cognitive Sciences*, *19*(1), 46–54. https://doi.org/10.1016/J.TICS.2014.10.004

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. In *Advances in Psychology*, *121*, 471–495. https://doi.org/10.1016/S0166-4115(97)80111-2

Lew, S., Rey, H. G., & Zanutto, B. S. (2013). Neuronal mechanisms underlying exploration-exploitation strategies in operant learning. *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 1–6.

Monk, C. T., Barbier, M., Romanczuk, P., Watson, J. R., Alós, J., Nakayama, S., … Arlinghaus, R. (2018). How ecology shapes exploitation: a framework to predict the behavioural response of human and animal foragers along exploration-exploitation trade-offs. *Ecology Letters*, *21*(6), 779–793. https://doi.org/10.1111/ele.12949

Naruse, M., Yamamoto, E., Nakao, T., Akimoto, T., Saigo, H., Okamura, K., … Hori, H. (2018). Local reservoir model for choice-based learning. *PLoS ONE*, *13*(10).

Rolon-Merette, D., Rolon-Merette, T., & Chartier, S. (2018). Distinguishing Highly Correlated Patterns using a Context Based Aproach in Bidirectional Associative Memory. *Proceedings of the International Joint Conference on Neural Networks*, *July 2018*, *1-8.*

Rolon-Mérette, D., Rolon-Mérette, T., & Chartier, S. (2019). Learning and Recalling Arbitrary Lists of Overlapping Exemplars in a Recurrent Artificial Neural Network. In *Proceedings of the International Conference on Cognitive Modelling*, 186-191.

Storkey, A. J., & Valabregue, R. (1999). The basins of attraction of a new Hopfield learning rule. *Neural Networks*, *12*(6), 869–876. https://doi.org/10.1016/S0893-6080(99)00038-6

Tilahun, S. L. (2019). Balancing the Degree of Exploration and Exploitation of Swarm Intelligence Using Parallel Computing. *International Journal on Artificial Intelligence Tools*, *28*(3). https://doi.org/10.1142/S0218213019500143

Tremblay, C., Myers-Stewart, K., Morissette, L., & Chartier, S. (2013). Bidirectional Associative Memory and Learning of Nonlinearly Separable Tasks. In *Proceedings of the International Conference on Cognitive Modeling,* 420–425.

Tsuda, I. (2001). Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems. *Behavioral and Brain Sciences*, *24*(5), 793–810.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-Learning. *Machine Learning, 8*(3-4). 279-292.