# A Hebbian learning algorithm for training a neural circuit to perform context-dependent associations of stimuli *

Henghui Zhu †, Ioannis Ch. Paschalidis ‡, and Michael E. Hasselmo §

***Abstract*— We propose a biologically plausible learning algorithm to train a neural circuit model to perform context-dependent associations of stimuli with correct responses. The specific cognitive task we consider requires the ability to learn a context-dependent association rule and generalize beyond what has been seen during training. We analyze the learning algorithm using a Markov chain framework and establish its convergence. Using numerical simulation, we validate the performance of the learning algorithm and the generalization ability of the neural circuit model.**

## I. Introduction

Behavioral data indicates that humans and animals are able to learn new rules and make different decisions based on different contexts. This requires the ability to generalize from what is learned, i.e., interpreting previously unseen sensory input to make a correct decision based on a previously learned rule [1]. It is believed that the neural circuit cortical structures, especially the prefrontal cortex, play a very important role in learning new rules [2], [3]. This requires some form of symbolic processing for the neural circuit to flexibly apply learned rules to new input.

There are many ways to modeling how a neural circuit flexibly links an unseen sensory input to the correct response. We focus on a flexible gating mechanism between different cortical working memory buffers by the basal ganglia [4] and the flexible routing in prefrontal cortex [2]. We consider a new neural circuit model inspired by the one described in [5]. The model uses interacting populations of neurons to gate neurons on the synaptic spread of activity between other neurons. Gating can be mediated by the interaction of activity from different populations of input neurons on the dendrites of a set of output neurons. For example, the activity of a first set of gating neurons may cause synaptic currents in the dendritic tree of the output neurons that are adjacent to the synaptic currents of a second set of input neurons. If the second set of input neurons activates voltage-sensitive N-Methyl-D-Aspartate synaptic currents [6], [7], these voltage-sensitive properties mean that the voltage change caused by the gating neurons will increase the current caused by the second set of input neurons, thereby gating the activity of the output neurons. Alternatively, axo-axonic inhibitory interneurons can also regulate spiking output [8].

We are interested in performing a context-dependent association task presented in [9]. Such a task was also considered in our earlier work [10], [11], which explored the use of approximate dynamic programming methods using deep neural network approximation architectures. Here, we prove properties of a learning algorithm previously proposed in [5] to optimize parameters of the neural circuit model. The learning algorithm is based on Hebbian learning [12], hence, it is biologically plausible because experimental data supports Hebbian modification in neural circuits. We establish convergence of the algorithm in a parameterized neural circuit model which can perform the context association task correctly. To that end, we use machinery from finite-state Markov chains. Finally, we use simulation to evaluate the convergence speed of the learning algorithm and the generalization ability of the optimized neural circuit model.

The remainder of the paper is organized as follows. Section II presents the context association learning task, the neural circuit model, and our new, biologically plausible learning algorithm. In Section III, we establish our main result on the convergence of the learning algorithm using Markov chain analysis. In Section IV, we conduct a simulation analysis of the performance of the learning algorithm and the generalization ability of the neural circuit model. Conclusions can be found in Section V.

## II. Notation and Problem Setup

We will use boldfaced lowercase letters to denote vectors, boldfaced uppercase letters to denote matrices, ordinary lowercase letters to denote scalars, and calligraphic capital letters to denote sets. For a matrix $\mathbf{A}$, we will denote by $A(i,j)$ its element in the $i$th row and $j$th column. All vectors are column vectors. For space saving reasons, we will write $\mathbf{x} = (x_1, \ldots, x_{\dim(\mathbf{x})})$ to denote the column vector $\mathbf{x}$, where $\dim(\mathbf{x})$ is the dimension of $\mathbf{x}$. We use prime to denote the transpose of a matrix or a vector, $\mathbf{0}$ for the matrix or vector of all zeroes, and $\mathbf{e}$ for the vector of all ones. We will also denote by $\lfloor x \rfloor$ the maximum integer that is smaller than or equal to $x$.

### A. Learning task

Fig. 1 shows the diagram of the learning task. There are four contexts in this task, corresponding to the four quadrants

* Research partially supported by the Office of Naval Research under MURI grant N00014-16-1-2832, by the NSF under grants DMS-1664644, CNS-1645681, CCF-1527292, and IIS-1237022, by the ARO under grant W911NF-12-1-0390, by the Boston University Digital Health Initiative, and by the Center for Information and Systems Eng. at Boston University.

† Henghui Zhu is with Center for Information and Systems Eng., Boston University, e-mail: henghuiz@bu.edu.

‡ Ioannis Ch. Paschalidis is with the Department of Electrical and Computer Eng., the Division of Systems Eng., and the Dept. of Biomedical Eng., Boston University, 8 St. Mary's St., Boston, MA 02215, e-mail: yannisp@bu.edu, http://sites.bu.edu/paschalidis/.

§ Michael E. Hasselmo is with the Center for Systems Neuroscience, and the Dept. of Psychological and Brain Sciences, Boston University, e-mail: hasselmo@bu.edu, www.bu.edu/hasselmo.

TABLE I: The encoding of different stimuli and contexts in the task of Figure 1.

| Stimuli and contexts | | Encoding |
|---|---|---|
| Stimulus | A | (1, 0, 0, 0, 0, 0, 0 ,0) |
| | B | (0, 1, 0, 0, 0, 0, 0 ,0) |
| | C | (0, 0, 1, 0, 0, 0, 0 ,0) |
| | D | (0, 0, 0, 1, 0, 0, 0 ,0) |
| Context | 1 | (0, 0, 0, 0, 1, 0, 0 ,0) |
| | 2 | (0, 0, 0, 0, 0, 1, 0 ,0) |
| | 3 | (0, 0, 0, 0, 0, 0, 1 ,0) |
| | 4 | (0, 0, 0, 0, 0, 0, 0 ,1) |

and denoted by numbers 1, 2, 3, and 4. Four input stimuli, denoted by letters A, B, C, and D, are associated with two responses, $X$ and $Y$, using different rules for different contexts. Specifically, the association rule between stimulus and response is the same in the 1st (upper left) and 4th quadrant (lower right). The same symmetry exists between the 2nd and 3rd quadrants.

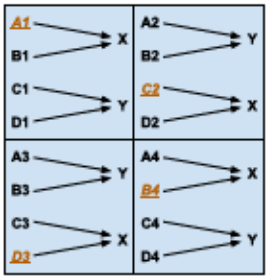



Fig. 1: Mapping between individual stimuli (A, B, C, D) and the spatial context (quadrants 1, 2, 3, 4) onto correct responses $X$ or $Y$, providing 16 input-response pairs. The underlined (red) input-response pairs are not seen during training but presented during the generalization test in Section IV.

We will use the term *input* to refer to a stimulus-context pair. Every stimulus and context is encoded as an 8-dimensional binary vector according to Table I. Notice that the first four bits of the vector correspond to the stimulus and the last four bits to the context. An input is represented as a sequence of two such vectors. For example, consider the stimulus-context pair B3. Stimulus B is encoded as $(0,1,0,0,0,0,0,0)$ and context 3 is encoded as $(0,0,0,0,0,0,1,0)$. Hence, the input is represented as a sequence $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$ where $\mathbf{s}_1 = (0,1,0,0,0,0,0,0)$ and $\mathbf{s}_2 = (0,0,0,0,0,0,1,0)$.

### B. Neural circuit model

Inspired by the work in [5], we consider a neural circuit model to perform the context association task we introduced. The model consists of a gating mechanism based on the nonlinear effects between synaptic inputs on adjacent parts of the dendritic tree that are due to voltage-sensitive conductances such as the N-Methyl-D-Aspartate (NMDA) current [6], [7]. These effects make it possible to determine the influence of spiking hidden neurons to the membrane potential of their adjacent neurons. Depending on which hidden neurons are spiking, different weight (gating) matrices get activated.

The input to the neural circuit model is the stimulus-context pair $\mathbf{s}$ we described earlier, represented as a sequence $(\mathbf{s}_1, \mathbf{s}_2)$ of two 8-dimensional vectors. The model has $n = 8$ neurons that receive such input sequences in two time instances. In addition, there are $m \geq 2$ hidden neurons and $m$ weight matrices connecting the input neurons and the hidden neurons; these weight matrices are denoted by $\mathbf{W}_i \in \mathbb{R}^{m \times n}$, $i = 1, \ldots, m$.

For each input $\mathbf{s}$, the model makes a decision in two time steps denoted by $t = 1, 2$. $t = 0$ will correspond to the initial condition. After a decision is made, the model gets initialized again and $t$ is set to zero. We assume that the stimulus $\mathbf{s}_1$ is the input at time $t = 1$ and the context $\mathbf{s}_2$ at time $t = 2$. Denote by $\mathbf{a}_t = (a_{t,1}, \ldots, a_{t,m}) \in \{0,1\}^m$ the indicator vector associated with the activation of hidden neurons at time $t$; specifically, $a_{t,i} = 1$ if neuron $i$ spikes at time $t$ and is zero otherwise. At each time instance, there is only one hidden neuron spiking. We denote by $i_t$ the activated neuron at time $t$, i.e., $i_t = \arg\max_i a_{t,i}$.

The decision-making rule of the neural circuit model is described as follows. We assume neuron 1 is spiking initially, i.e., $a_{0,1} = 1$. At each time instance $t$, the pre-activation function $\mathbf{f}_t = (f_{t,1}, \ldots, f_{t,m})$ of the hidden neurons is defined as

$$\mathbf{f}_t = \mathbf{W}_{i_t \ 1} \mathbf{s}_t + \boldsymbol{\omega}_t, \tag{1}$$

$$a_{t,i} = \begin{cases} 1, & \text{if } i = \arg\max_i f_{t,i}, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $\boldsymbol{\omega}_t \in \mathbb{R}^m$ is an environment noise with each element independent of the others and uniformly distributed in $[0, 1)$. (Ties in (2) are broken arbitrarily.) We assume $\boldsymbol{\omega}_t$ are independent for different times $t$. Notice that the activated hidden neuron at the previous time instance determines the weight matrix to be used. The spiking neuron at $t$ is the neuron that has the maximum pre-activation value.

At the last time instance for each input ($t = 2$), the last two hidden neurons determine the selection of either response $X$ or response $Y$. Specifically, if $a_{2,m-1} = 1$, then response $X$ is selected. If $a_{2,m} = 1$, then response $Y$ is selected. If the activated neuron is not one of the last two, then no response is provided.

We provide an example to illustrate the operation of the neural circuit model. Suppose there are two hidden neurons ($m = 2$) and the two corresponding weight matrices are:

$$\mathbf{W}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix},$$
$$\mathbf{W}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

Suppose the input is the stimulus-context pair C1 and the noise $\boldsymbol{\omega}_t = (0,0)$. The input at the first time instance is $\mathbf{s}_1 = (0,0,1,0,0,0,0,0)$. Then, the pre-activation function becomes $\mathbf{f}_1 = \mathbf{W}_1 \mathbf{s}_1 = (0,1)$. Therefore, $\mathbf{a}_1 = (0,1)$ and the second hidden neuron spikes at time $t = 1$. The input for

$t = 2$ is $\mathbf{s}_2 = (0,0,0,0,1,0,0,0)$ and $\mathbf{f}_2 = \mathbf{W}_2\mathbf{s}_2 = (0,1)$. Hence, $\mathbf{a}_2 = (0,1)$ and response $Y$ is selected since it is the second hidden neuron spiking at the final time instance $t = 2$. It can be verified that the weight matrices in (3) will always yield the correct response for every stimulus-context pair input according to the rule of Fig. 1.

### C. Learning Algorithm

To train the weight matrices, we devise a learning algorithm for the neural circuit based on the Hebbian rule for plasticity of synaptic connections. The weight matrices $\mathbf{W}_i$ are updated according to a combination of Long-Term Potentiation (LTP) and Long-Term Depression (LTD), modulated by appropriate gating and depending on whether the output response matches the correct one.

The LTP rule is mostly based on the basic Hebbian rule in [12], in which simultaneous pre- and post-synaptic activity increases synaptic strength. Recall that $i_t$ is the index of the spiking hidden neuron at time $t$, $t = 0, 1, 2$. According to our assumption, $i_0 = 1$. The LTP term is:

$$\Delta \mathbf{W}_{i_t\ 1,\mathrm{LTP}} = \mathbf{a}_t \mathbf{s}'_t. \tag{4}$$

Long-Term Depression (LTD) provides an activity-dependent reduction in the efficacy of neuronal synapses to serve as a regularization of the learning process. The LTD term is:

$$\Delta \mathbf{W}_{i_t\ 1,\mathrm{LTD}} = -\mathbf{e}\mathbf{s}'_t. \tag{5}$$

Finally, we consider the following update rule. Given an input signal, if the response of the neural circuit model coincides with the correct response, we update the weight matrices using the LTP rule. Otherwise, when either the response is incorrect or there is no response produced, we update the weight matrices using the LTD rule. We project the elements of the weight matrices $\mathbf{W}_i$ to $[0,1]$ after every update. Algorithm 1 specifies the steps of the learning algorithm, where $\alpha \in (0,1)$ is a stepsize.

**Algorithm 1** Hebbian Learning Algorithm for the neural circuit model.

*Initialization:* Initialize $\mathbf{W}_i = \mathbf{0}$, $i = 1, \ldots, m$.
**repeat**
  Sample a state $\mathbf{s}$. Set $\mathbf{a}_0 = (1, 0, \ldots, 0)$.
  **for** $t = 1, 2$ **do**
    Find the spiking neuron at time $t$ using (1) and (2).
    Set $\Delta \mathbf{W}_{i_t\ 1,LTP}$, $\Delta \mathbf{W}_{i_t\ 1,LTD}$ according to (4) and (5).
  **end for**
  Select response $X$ or $Y$ according the spiking neuron at $t = 2$.
  **if** the response is correct **then**
    Perform an LTP update for the weight matrices

$$\mathbf{W}_{i_t\ 1} \leftarrow \mathbf{W}_{i_t\ 1} + \alpha \Delta \mathbf{W}_{i_t\ 1,LTP},\ t = 1, 2.$$

  **else**
    Perform an LTD update for the weight matrices

$$\mathbf{W}_{i_t\ 1} \leftarrow \mathbf{W}_{i_t\ 1} + \alpha \Delta \mathbf{W}_{i_t\ 1,LTD},\ t = 1, 2.$$

  **end if**
  Project all elements of $\mathbf{W}_i$ to $[0,1]$, $\forall i = 1, \ldots, m$.
**until** some convergence criterion on $\mathbf{W}_i$ is satisfied.

## III. Main Results

In this section, we establish the convergence of Algorithm 1 for the context association task we have introduced. To that end, we will consider a discrete-time Markov chain whose "state" is characterized by the weight matrices. We will use $x_t$ to denote the state of a (generic) Markov chain at time $t$. We make the following assumption about the inputs provided to the algorithm.

**Assumption 1** *Each input (stimulus-context pair) is sampled in Algorithm 1 uniformly.*

In fact, it suffices to sample any input with a constant positive probability at each time; we assume a uniform sampling distribution for simplicity.

**Definition 2 ([13])** *A state $i$ in a Markov chain is called* closed, *if $P(x_{t+1} = i | x_t = i) = 1$. We say a state $i$ is* accessible *from a state $j$, if there exists some $T > 0$, such that $P(x_{t+T} = i | x_t = j) > 0$.*

The following lemma follows from standard Markov chain theory. We omit the proof.

**Lemma 3** *Consider a Markov chain with a finite state space $\mathcal{X}$. Let $\mathcal{X}_c$ be the set that contains all its closed states. Suppose that for all states $i \in \mathcal{X}$, there exists some $j(i) \in \mathcal{X}_c$ such that $j(i)$ is accessible from $i$. Then, for every initial distribution of the Markov chain, a stationary distribution exists and is such that all elements corresponding to non-closed states are zero.*

Let us now denote by $\mathbf{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_m\}$ the collection of weight matrices for the neural circuit and by $\mathcal{W}$ the set of all possible weight matrices generated by Algorithm 1. We consider a Markov chain with states $\mathbf{W}$ and state space $\mathcal{W}$. The state space is finite since, as a result of the updates in Algorithm 1, each element of a weight matrix takes values in $\{0, \alpha, 2\alpha, \ldots, \lfloor \frac{1}{\alpha} \rfloor \alpha, 1, 1-\alpha, \ldots 1 - \lfloor \frac{1}{\alpha} \rfloor \alpha\}$. The zero state is defined as the collection of zero weight matrices and notice that it corresponds to the initial condition of Algorithm 1. The Markov chain we have defined is homogeneous due to Assumption 1.

Consider now Algorithm 1 and the updates it performs when the input is some $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$. We will denote a *decision path* by the 5-tuple $(i_0, j_1, i_1, j_2, i_2)$, where $i_0 = 1$, $i_1$ and $i_2$ are the activated neurons at times 0, 1, and 2, respectively, and $j_1 = \arg\max_k \mathbf{s}_{1,k}$, $j_2 = \arg\max_k \mathbf{s}_{2,k}$ are the indices of the non-zero entries in the stimulus $\mathbf{s}_1$ and the context $\mathbf{s}_2$, respectively. For example, for the stimulus-context pair $C2$, $j_1 = 3$ and $j_2 = 6$. Notice that an LTP update under a decision path $(i_0, j_1, i_1, j_2, i_2)$ implies an increase of $W_{i_0}(i_1, j_1)$ and $W_{i_1}(i_2, j_2)$. Similarly, an LTD

update under the same decision path implies a decrease of all the elements in the $j_1$th column of $\mathbf{W}_{i_0}$ and all the elements in the $j_2$th column of $\mathbf{W}_{i_1}$. We will call the first four columns of each $\mathbf{W}_i$ the *stimulus columns* and the last four columns the *context columns*. From the above discussion, it is seen that the stimulus columns of $\mathbf{W}_i$, $i = 2, 3, \ldots$ are not updated by Algorithm 1, since $i_0$ is always 1. Hence, given the initialization, these columns maintain all elements equal to zero.

**Definition 4** *We call a state* $\mathbf{W}$ sub-optimal*, if for all inputs and all possible values of the noise, the forward propagation* (1) *under* $\mathbf{W}$ *yields the correct response. We say a state* $\mathbf{W}$ *is* optimal*, if it is sub-optimal and invariant under all possible LTP updates.*

**Lemma 5** *For any sub-optimal state* $\mathbf{W}$*, there is an optimal state* $\mathbf{W}^*$ *that is accessible from* $\mathbf{W}$.

*Proof:* An LTP update only reinforces the matrix elements in a decision path. Since a sub-optimal state only produces the correct response for all inputs, it can be seen that the state after an LTP update remains sub-optimal. Further, the LTP update is monotonic with respect to the elements of all weight matrices. Hence, if we keep sampling uniformly all possible inputs at a sub-optimal state, the LTP updates will lead to saturation (a value equal to 1) all nonzero elements of the weight matrices, which will result in an invariant, hence, optimal state. ■

Next, we argue that an optimal state exists for all the circuits with $m \geq 2$. For $m = 2$ we have provided an optimal state in Equation (3) and we can easily extend that construction for $m > 2$. It is possible that there exist multiple optimal states. In any case, however, an optimal state is accessible from the zero state, since we can always provide the appropriate inputs to reinforce the desired matrix elements.

We note that, based on the initialization assumptions in Algorithm 1, hidden neuron 1 always spikes at time $t = 0$. This implies that the stimulus columns of $\mathbf{W}_1$ determine which neuron will spike after the stimulus is presented.

**Corollary 6** *If all elements in the* $i$*th column of* $\mathbf{W}_1$ *are smaller than* 1*, for* $i \leq 4$*, then all hidden neurons could spike after the* $i$*th stimulus is presented. Similarly, if all the elements in the* $i$*th column of* $\mathbf{W}_1$ *are all equal to* 1*, for* $i \leq 4$*, then all hidden neurons could spike after the* $i$*th stimulus is presented.*

This is a simple observation due to (1) and the noise term $\boldsymbol{\omega}_t$; we omit the proof.

**Lemma 7** *The zero state is accessible from state* $\mathbf{W}$ *if for all* $i = 1, \ldots, m$*, all elements in the stimulus columns of the matrix* $\mathbf{W}_1$ *are either all smaller* 1 *or all equal to* 1.

*Proof:* First, by Corollary 6, all hidden neurons have the possibility to spike after any stimulus is presented. Moreover, given a stimulus, all responses are possible depending on the context. It follows that it is possible to find an adversarial input that leads to an incorrect response.

As a result, given a weight matrix $\mathbf{W}_1$ with elements in the stimulus columns smaller than 1 or all equal to 1, it is possible to find a sequence of adversarial inputs that eventually drive to zero every nonzero element in the stimulus columns of $\mathbf{W}_1$. Extending this argument, it is possible to drive to zero all remaining nonzero elements in the weight matrices included in $\mathbf{W}$. Specifically, for a context $j_2$ and hidden neuron 1, it is possible to find a stimulus with index $j_1$ and associated decision path $(1, j_1, 1, j_2, i_2)$ that leads to an incorrect response. We can first drive to zero the context columns of $\mathbf{W}_1$ using LTD. Then, for a stimulus $j_1$, since the last four column of $W_1$ are all zeros, we can pick an incorrect response $i_2$ in the decision path of the form $(1, j_1, 1, 5, i_2)$. This will drive to zero the stimulus columns of $\mathbf{W}_1$. Finally, to drive to zero any $j_2$th column in matrix $\mathbf{W}_i$, where $j_2 > 4$ and $i > 1$, we can pick a stimulus $j_1$ that leads to an incorrect decision path of the form $(1, j_1, i, j_2, i_2)$. ■

**Lemma 8** *Suppose* $\mathbf{W}$ *is invariant under LTD given all possible examples. If there is a zero column in the stimulus columns of* $\mathbf{W}_1$*, then* $\mathbf{W}$ *is the zero state.*

*Proof:* Suppose a non-zero $\mathbf{W}$ is invariant under LTD given all possible inputs, and, without loss of generality, the first column of $\mathbf{W}_1$ is zero. This implies that all hidden neurons, $i = 1, \ldots, m$, can spike after stimulus $A$ is presented. Also, recall that the stimulus columns of $\mathbf{W}_i$, for $i = 2, 3, \ldots$, are zero since they are not updated by Algorithm 1. Therefore, the fact that $\mathbf{W}$ is not zero implies the following two possibilities.

1) The context columns of $\mathbf{W}_i$ are all zero for all $i$. This indicates there is some non-zero element in the context columns of $\mathbf{W}_1$. This is impossible since there is some decision path resulting in an LTD update which will decrease this non-zero element.
2) There is some non-zero element in the context columns of $\mathbf{W}_i$. Given the invariance under LTD, the neural circuit will produce the correct response for every input. In this case, since all hidden neurons, $i = 1, \ldots, m$, may spike after stimulus $A$ is presented, $\mathbf{W}_i$ should map context 1 to response $X$ for all $i$, according the rule in Figure 1. This also implies that each of the context columns of $\mathbf{W}_i$ leads to a unique response. Consider now the input $C1$. No matter which hidden neuron, say neuron $k$, is spiking at time $t = 1$, the neural circuit will produce response $X$, which is not correct. Hence, according to Algorithm 1, an LTD update will be performed and matrix $\mathbf{W}_k$ will be changed. This contradicts the invariance of $\mathbf{W}$.

■

**Corollary 9** *Suppose a non-zero* $\mathbf{W}$ *is invariant under LTD given all possible inputs. It is impossible to have a column*

*in the stimulus columns of* $\mathbf{W}_1$ *whose elements are either all smaller than* 1 *or all equal to* 1.

By using Lemma 6, the proof is similar to the proof of Lemma 10 and is therefore omitted.

**Lemma 10** *Suppose* $\mathbf{W}$ *is invariant under LTD given all possible examples. If* $\mathbf{W}$ *is not sub-optimal, then* $\mathbf{W}$ *is the zero state.*

*Proof:* By Lemma 8, if $\mathbf{W}$ is not sub-optimal or the zero state, then all stimulus columns of $\mathbf{W}_1$ have all their elements being non-zero. Since $\mathbf{W}$ is not sub-optimal, then there exists some decision path, say $(1, 1, j_1, i_2, j_2)$, which produces an incorrect response. According to Algorithm 1, an incorrect response will lead to an LTD update. Since $\mathbf{W}$ is invariant under LTD, according to Corollary 9 the first column of $\mathbf{W}_1$ should not lead to an uncertain choice of the hidden neuron to be activated. This implies that $W_1(1, j_1) > 0$, which is not invariant under an LTD update. This contradicts the assumption that $\mathbf{W}$ is invariant under LTD. ■

**Lemma 11** *For every* $\mathbf{W} \in \mathcal{W}$*, there exists an optimal solution* $\mathbf{W}^*$ *such that* $\mathbf{W}^*$ *is accessible from* $\mathbf{W}$.

*Proof:* If $\mathbf{W}$ is sub-optimal, the result holds by Lemma 5. Now consider a $\mathbf{W}$ which is not sub-optimal. This implies that the neural circuit model can produce an incorrect response. Then, an LTD update is performed according to Algorithm 1. After such an LTD update, some elements of the matrices in $\mathbf{W}$ are decreased. Consider the set $\mathcal{U}_{\mathbf{W}}$, which contains all states invariant under LTD we can reach by starting from $\mathbf{W}$ and providing as many adversarial inputs as necessary (with all resulting in LTD updates). According to Lemma 10, states in $\mathcal{U}_{\mathbf{W}}$ can only be sub-optimal or the zero state. If the zero state is in $\mathcal{U}_{\mathbf{W}}$, then it is accessible from $\mathbf{W}$. Therefore, any optimal state $\mathbf{W}^*$ is accessible from $\mathbf{W}$ since optimal states are accessible from the zero state. If $\mathcal{U}_{\mathbf{W}}$ contains only sub-optimal states, the result is also true since there exist some optimal state accessible from a sub-optimal state. ■

Finally, we state our main convergence result for Algorithm 1.

**Theorem 12** *The neural circuit model converges to an optimal state under Algorithm 1 and Assumption 1.*

*Proof:* Consider the Markov chain on $\mathcal{W}$ induced by Algorithm 1. Optimal states are closed. By Lemma 11, for any state $\tilde{\mathbf{W}}$, there exists some optimal state $\mathbf{W}^*$ such that $\mathbf{W}^*$ is accessible from $\tilde{\mathbf{W}}$. By Lemma 3, and given the zero initial state, Algorithm 1 converges to some optimal state. ■

**Remark 13** *The convergence result can be easily extended to the case where the environment noise* $\boldsymbol{\omega}_t$ *has elements taking values in* $[0, U_{\text{noise}}]$*, where* $U_{\text{noise}} < 1$*. Further, the result can be extended to the case where the initial state does not consist of all-zero weight matrices. In such a case, optimal states may contain some weight matrices with elements in* $(0, 1)$.

## IV. Simulation

In this section, we validate the convergence of Algorithm 1 and demonstrate the generalization ability of the neural circuit model.

In our first experiment, we let the number of hidden neurons to be $m = 2$ and set the stepsize to $\alpha = 0.1$. We run the learning algorithm for many iterations and after each iteration we evaluate the accuracy of the neural circuit model over all inputs. Fig. 2 plots the accuracy, measured by the fraction of correct responses to all possible inputs, while we vary the number of iterations. We observe convergence to an optimal state. Note that 100% accuracy does not necessarily imply we have reached an optimal state. However, based on our results, we are guaranteed we will reach an optimal state.

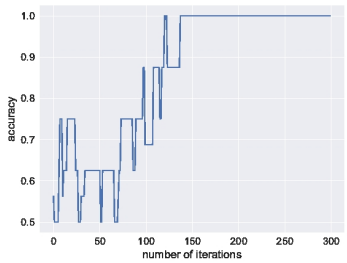
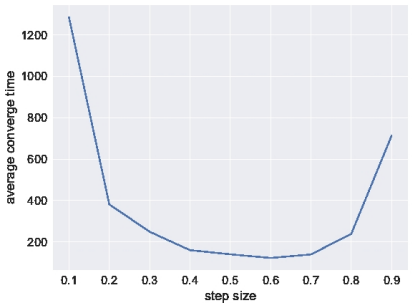



Fig. 2: Accuracy of the neural circuit model trained by Algorithm 1.

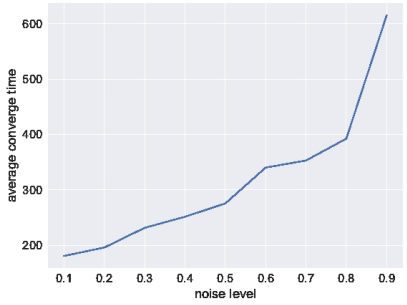
Next, we evaluate the performance of the learning algorithm with respect to the two hyperparameters: the stepsize $\alpha$ and the strength of the noise $U_{\text{noise}}$ defined in Remark 13. We set the number of hidden neurons to $m = 2$. To evaluate the performance with respect to the stepsize $\alpha$, we use noise terms uniformly distributed in $[0, 1)$. For each value of $\alpha$, we run the learning algorithm 100 times and compute the average number of iterations required to reach an optimal state. The results are shown in Fig. 3a. It can be seen that there exists a stepsize value (close to 0.7) which minimizes the average number of iterations. Having a too small, or too large stepsize, can lead to slower convergence either due to small stepsize, or due to oscillations.

To evaluate the performance with respect to the noise strength $U_{\text{noise}}$, we fix the number of hidden neurons to $m = 2$ and the stepsize to $\alpha = 0.1$. For each value of $U_{\text{noise}}$, we run the learning algorithm 100 times and compute the average number of iterations required to reach an optimal state. Shown in the Fig. 3b is a plot of the average number of iterations as a function of $U_{\text{noise}}$. Not surprisingly, the number of iterations increases as the noise strength increases.

Finally, we evaluate the generalization ability of the neural circuit model. In this test, we hide four stimulus-context pairs




(a) Average number of iterations with respect to the stepsize $\alpha$.




(b) Average number of iterations with respect to the noise strength $U_{\text{noise}}$.

Fig. 3: Average number of iterations required to converge to an optimal state with respect to the stepsize $\alpha$ and the noise strength $U_{\text{noise}}$.

during the training (A1, C2, D3, and B4, shown in red in Fig. 1). Notice that we hide one input from each context, and as a result, the neural circuit has to be able to learn the context-dependent rule and generalize beyond what it has seen during training. We evaluate the generalization ability of the neural circuit model with respect to the number of the hidden neurons $m$. We fix the stepsize to $\alpha = 0.3$ and the noise strength to $U_{\text{noise}} = 0.7$ during the training. Also, we set the maximum number of iterations to $10,000$. For each value of $m$, we again run the learning algorithm 100 times and compute the average accuracy under all the stimulus-context pairs, including the hidden ones. The result is shown in Fig. 4. We observe that when the number of the hidden neurons is large, the neural circuit model is more complex and hence does not generalize as effectively across additional stimuli, resulting in generalization errors for the simple task of Fig. 1. Naturally, however, if one increases the complexity of the task (e.g., by increasing the number of stimuli and contexts) more than 2 hidden neurons will be needed to learn effectively.

## V. Conclusion

In this paper, we proposed a biologically plausible learning algorithm to train a neural circuit model to perform a context association task. The algorithm uses the principle of Hebbian learning which is believed to be consistent with the way the human brain learns. In contrast, gradient-based methods


Fig. 4: Accuracy of the neural circuit with respect to the number of the hidden neurons in the generalization test.

for reinforcement learning have less support from biological data. We established the convergence of the algorithm using Markov chain techniques. Using numerical simulation, we validated the performance of the learning algorithm and demonstrated the generalization ability of the neural circuit model.

## References

[1] J. A. Fodor and Z. W. Pylyshyn, "Connectionism and cognitive architecture: A critical analysis," *Cognition*, vol. 28, no. 1-2, pp. 3–71, 1988.

[2] E. K. Miller and J. D. Cohen, "An integrative theory of prefrontal cortex function," *Annual review of neuroscience*, vol. 24, no. 1, pp. 167–202, 2001.

[3] J. D. Wallis, K. C. Anderson, and E. K. Miller, "Single neurons in prefrontal cortex encode abstract rules," *Nature*, vol. 411, no. 6840, p. 953, 2001.

[4] T. Kriete, D. C. Noelle, J. D. Cohen, and R. C. O'Reilly, "Indirection and symbol-like processing in the prefrontal cortex and basal ganglia," *Proceedings of the National Academy of Sciences*, p. 201303547, 2013.

[5] M. E. Hasselmo and C. E. Stern, "A network model of behavioural performance in a rule learning task," *Phil. Trans. R. Soc. B*, vol. 373, no. 1744, p. 20170275, 2018.

[6] P. Poirazi, T. Brannon, and B. W. Mel, "Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell," *Neuron*, vol. 37, no. 6, pp. 977–987, 2003.

[7] Y. Katz, W. L. Kath, N. Spruston, and M. E. Hasselmo, "Coincidence detection of place and temporal context in a network model of spiking hippocampal neurons," *PLoS Comput Biol*, vol. 3, no. 12, p. e234, 2007.

[8] V. Cutsuridis and M. Hasselmo, "Gabaergic contributions to gating, timing, and phase precession of hippocampal neuronal activity during theta oscillations," *Hippocampus*, vol. 22, no. 7, pp. 1597–1621, 2012.

[9] F. Raudies, E. A. Zilli, and M. E. Hasselmo, "Deep belief networks learn context dependent behavior," *PLoS ONE*, vol. 9, no. 3, 2014.

[10] H. Zhu, M. Hasselmo, and I. C. Paschalidis, "Feature extraction in Q-Learning using neural networks," in *Proceedings of the 56th IEEE Conference on Decision and Control*, Melbourne, Australia, December 12–16 2017, pp. 3330–3335.

[11] H. Zhu, I. C. Paschalidis, and M. E. Hasselmo, "Neural circuits for learning context-dependent associations of stimuli," *Neural Networks*, vol. 107, pp. 48–60, November 2018.

[12] P. Dayan and L. F. Abbott, *Theoretical neuroscience*. Cambridge, MA: MIT Press, 2001, vol. 10.

[13] P. Brémaud, *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer Science & Business Media, 2013, vol. 31.