Biological Softmax: Demonstrated in Modern Hopfield Networks

Mallory Snow (m5snow@uwaterloo.ca)
Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, N2L 3G1 CANADA

Jeff Orchard (jorchard@uwaterloo.ca)
Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, N2L 3G1 CANADA

Abstract

Modern Hopfield networks (HNs) exhibit properties of a Content Addressable Memory (CAM) that can store and retrieve a large number of memories. They also provide a basis for modelling associative memory in humans. However, the implementation of these networks is often not biologically plausible as they assume the strengths of synaptic connections are symmetric, and utilize functions that rely on many-body synapses. More biologically realistic versions of Modern HNs have been proposed, although these implementations often still utilize the softmax function. Computing the softmax for a single node requires the knowledge of all other neurons, and thus still poses a degree of biological implausibility. We present a Modern HN that uses a version of softmax that can be computed in a more bio-realistic way, and hence moves us closer to a model of memory that is biologically sound. We also show that our proposed network can learn the connection weights using a local learning rule, derived from gradient descent on the energy function. Finally, we verify that our proposed biological network behaves like a Modern HN and explore some of its other interesting properties.

Keywords: Neural networks, Hopfield networks, associative memory, content addressable memory, biological plausibility, attractor networks.

Introduction

A Content Addressable Memory (CAM) is a system that can retrieve a memory (e.g. a pattern) given sufficient partial or noisy information (part of the memory). Hopfield Networks (HNs) are neural networks that act as CAMs by learning patterns and retrieving them when given a partial or perturbed pattern (Hopfield, 1982). Human memory has some similar properties, often referred to as *cued-recall* whereby a full memory can be elicited by partial information (Tulving & Pearlstone, 1966; Earhard, 1967).

HNs are based on an energy function, E, which combines the cost of neurons being active in the network, and the inconsistency/conflict between connected neurons. This consistency is measured by an interaction function, F. Traditional HNs use $F(x) = x^2$ as this interaction function. A HN with d nodes (neurons) can recall approximately $0.138d$ different patterns; this is called its *storage capacity*. A lot of work has been put into increasing the storage capacity of HNs.

More recently, Krotov and Hopfield presented Modern HNs, or Dense Associative Memory (DAM) models, which are HNs that use polynomials as the interaction function in the energy (Krotov & Hopfield, 2016). They showed that using $F(x) = x^n$ increased the storage capacity to d^{n-1} for a network with d nodes. They later showed that these modern HNs are much more robust to adversarial input (Krotov & Hopfield, 2018). Other interaction functions have been proposed, but many implementations of HNs are non-biological. First of all, the connection weights in a HN are assumed to be symmetric; the strengths of two physically different synapses – say from neuron μ to neuron i, and from i to μ – are equal. Secondly, many of these modern HNs have many-body synapses, where the input to a neuron depends on nonlinear combinations of other neurons. For example, if one uses a quartic interaction function, $F(x) = x^4$, the inputs to each neuron will depend on the products of triplets of neurons. It is unclear how such many-body synapses can be implemented in a biological neural network.

Krotov and Hopfield proposed a model of large associative memory that moves closer to being biologically plausible by addressing the many-body synapse issue (Krotov & Hopfield, 2021). Their approach is to couple a set of N_h "memory" or "hidden" neurons (with input currents h_μ) to the N_v feature neurons (with input currents v_i). The full network of $N_v + N_h$ neurons forms a bipartite graph, which behaves like a classifier; each target pattern in the feature nodes corresponds to a one-hot setting of the hidden nodes, and *vice versa*. In these networks, ξ is a connection-weight matrix in which $\xi_{\mu i}$ represents the strength of the synapse from feature neuron i to the memory neuron μ. Again, these connections are assumed to be symmetric, so $\xi_{\mu i} = \xi_{i\mu}$. There are no synaptic connections among the hidden neurons, or feature neurons. The outputs of the feature neurons and memory neurons are denoted by g_i and f_μ respectively, which are (nonlinear) functions of their corresponding input currents. A small example of the network discussed above can be seen in Figure 1.

However, even in some of Krotov and Hopfield's neurobiological networks, the outputs of the feature and

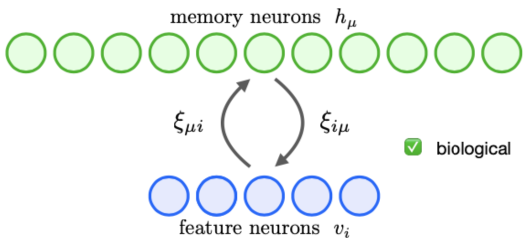


Figure 1: A small example of the biological HN with $N_v = 5$ feature neurons and $N_h = 11$ memory (hidden) neurons and symmetric synaptic connections between them ($\xi_{\mu i} = \xi_{i\mu}$). Image taken from (Krotov & Hopfield, 2021).

memory neurons (g_i and f_μ) still involve "contrastive normalization". This means the output of a neuron is normalized with respect to the currents of all other neurons in the layer (Krotov & Hopfield, 2021). Specifically, the model we focus on in this paper, the *LSE network* (corresponding to *Model B* in (Krotov & Hopfield, 2021)), involves activation functions of the form $g_i = v_i$, and

$$f_\mu = \mathcal{S}_\mu(h) = [\text{Softmax}(h)]_\mu = \frac{e^{h_\mu}}{\sum_m e^{h_m}} . \tag{1}$$

Computing the softmax for a single hidden node requires knowledge of other hidden neurons, and still poses a degree of biological implausibility.

In this paper, we extend the LSE network presented by Krotov and Hopfield to incorporate a more biological implementation of the softmax function, and thus move even closer to a biologically plausible model of associative memory. In addition, we show that this biological LSE network can *learn* the connection weights using a local learning rule, derived from gradient descent on the energy function. We then explore and analyze some interesting behaviours of these networks.

Mathematical Framework: LSE Network

First, we present the LSE Hopfield network in continuous time as presented in (Krotov & Hopfield, 2020). Recall that the network is made up of N_v feature neurons (with input currents v_i), and N_h hidden neurons (with input currents h_μ). All feature neurons are connected to all hidden neurons by symmetric connection weights, so $\xi_{\mu i} = \xi_{i\mu}$. The outputs of the hidden and feature neurons are denoted f_μ and g_i, respectively, such that $f_\mu = \frac{\partial L_h}{\partial h_\mu}$ and $g_i = \frac{\partial L_v}{\partial v_i}$, for some Lagrange functions L_h and L_v. The time constants for the feature and memory neurons are denoted by τ_v and τ_h respectively. With all of this in mind, the model can be written as a dynamical system,

$$\tau_v \frac{dv_i}{dt} = \sum_{\mu=1}^{N_h} \xi_{i\mu} f_\mu - \kappa v_i + I_i \ , \tag{2}$$

$$\tau_h \frac{dh_\mu}{dt} = \sum_{\mu=1}^{N_v} \xi_{\mu i} g_i - h_\mu \ , \tag{3}$$

where κ controls the decay term for the feature neurons, and I_i denotes the input current to feature neuron i. The energy function for this model is thus the sum of three terms: one term for the feature neurons, one term for the hidden neurons, and one term for the interaction between the two groups of neurons. Hence, $E(t)$ can we written

$$\left[\sum_{i=1}^{N_v} (v_i - I_i) g_i - L_v\right] + \left[\sum_{\mu=1}^{N_h} h_\mu f_\mu - L_h\right] - \sum_{\mu,i} f_\mu \xi_{\mu i} g_i \ .$$

In this LSE network, the Lagrange functions are $L_h = \log\left(\sum_\mu e^{h_\mu}\right) = \text{LSE}(h)$, and $L_v = \frac{1}{2}\sum_i v_i^2$, yielding the energy function,

$$E(t) = \sum_{i=1}^{N_v}\left(\frac{1}{2}v_i^2 - v_i I_i\right) + \sum_{\mu=1}^{N_h}\Big(h_\mu \mathcal{S}_\mu(h) - \text{LSE}(h_\mu)\Big) - \sum_{\mu,i} \mathcal{S}_\mu(h)\xi_{\mu i} v_i . \tag{4}$$

As the theory of Hopfield networks specifies, the dynamics of the nodes in the network perform gradient descent on the energy function, so are governed by,

$$\tau_v \frac{dv_i}{dt} = (1-\beta)\sum_{\mu=1}^{N_h} \xi_{i\mu} \mathcal{S}_\mu(h) - v_i + \beta I_i, \tag{5}$$

$$\tau_h \frac{dh_\mu}{dt} = \sum_{\mu=1}^{N_v} \xi_{\mu i} v_i - h_\mu, \tag{6}$$

where $f_\mu = \mathcal{S}_\mu(h)$ is the softmax, which is the gradient of $\text{LSE}(h)$, and $g_i = v_i$ is the gradient of L_v. The introduction of the weighting factor β in (5) allows the input to be turned on or off. The input to the feature neurons, I, can be 'turned off' by setting $\beta = 0$. That input can be 'turned on' by setting β to be non-zero.

In (5), we see the biologically problematic use of softmax, $\mathcal{S}_\mu(h)$. The output of hidden node μ depends on **all** other hidden nodes.

Biological Softmax

We introduce a neural network that computes the softmax function in a more biologically plausible way. This subnetwork is then combined with Krotov and Hopfield's model to enhance its biological plausibility. Recall that we denote the softmax function by $\mathcal{S}_\mu(h)$,

$$S_\mu(h) = \frac{e^{h_\mu}}{\sum_j^{N_h} e^{h_j}} \quad \text{for } h \in \mathbb{R}^{N_h}.$$

Notice that computing a single element (i.e. $\mathcal{S}_\mu(h)$) requires the input from all N_h elements of h. To be biologically plausible, each neuron's output must depend only on its own input. Thus, softmax cannot be computed by each neuron by itself, but must be implemented by a *network* of neurons.

First, let us consider $\log(\mathcal{S}_\mu(h))$:

$$\begin{aligned}\log(\mathcal{S}_\mu(h)) &= \log\left(\frac{e^{h_\mu}}{\sum_j^{N_h} e^{h_j}}\right)\\ &= \log(e^{h_\mu}) - \log\left(\sum_j^{N_h} e^{h_j}\right)\\ &= h_\mu - \mathrm{LSE}(h).\end{aligned}$$

Suppose we have a node (or a collection of neurons), denoted b, that stores $\sum_j^{N_h} e^{h_j}$. Then ,

$$\log(\mathcal{S}_\mu(h)) = h_\mu - \log(b)\ .$$

Next, if we define $r_\mu = h_\mu - \log(b)$, then we get $\log(\mathcal{S}_\mu(h)) = r_\mu$. Taking the exponential function of both sides yields,

$$\mathcal{S}_\mu(h) = e^{r_\mu}\ .$$

This is similar to the way Bogacz and Gurney use logarithms to do normalization of probabilities (2007). We can model this biological softmax network as a system of differential equations,

$$\begin{aligned}\tau_s \frac{db}{dt} &= \sum_{j=1}^{N_h} e^{h_j} - b\\ \tau_s \frac{dr_\mu}{dt} &= h_\mu - \log(b) - r_\mu\\ \tau_s \frac{df_\mu}{dt} &= e^{r_\mu} - f_\mu\,,\end{aligned}$$

where we have introduced a population of output nodes, denoted f, and τ_s is the time constant for computing softmax. It is straightforward to show that, for fixed h, the equilibrium solution of this dynamical system yields $f_\mu = \mathcal{S}_\mu(h)$, the softmax function (consistent with (1)). This biological softmax network is shown in Figure 2.

Biological LSE Network

Finally, we combine the LSE and the biological softmax networks to arrive at the biological LSE network, replacing the softmax function, $\mathcal{S}_\mu(h)$ in (5) with e^{r_μ}. This results in the following set of coupled differential equations,

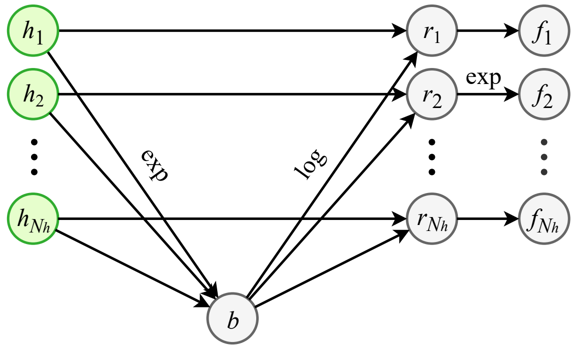


Figure 2: Biological softmax subnetwork.

$$\begin{aligned}\tau_s \frac{db}{dt} &= \sum_{j=1}^{N_h} e^{h_j} - b\\ \tau_s \frac{dr_\mu}{dt} &= h_\mu - \log(b) - r_\mu\\ \tau_v \frac{dv_i}{dt} &= (1-\beta)\sum_{\mu=1}^{N_h} \xi_{i\mu} e^{r_\mu} - v_i + \beta I_i\\ \tau_h \frac{dh_\mu}{dt} &= \sum_{\mu=1}^{N_v} \xi_{\mu i} v_i - h_\mu\ .\end{aligned}$$

It is important that the time constant for computing the softmax function, τ_s, is smaller than τ_v and τ_h; we need the softmax network to produce its output quickly so that the sofmax function appears instantaneous relative to the rest of the network.

Notice that we avoid the need to model the f nodes by projecting directly from the r nodes to the v nodes. Figure 3 shows the network before and after the addition of the biological softmax subnetwork.

Learning Weights

Consider how we might change ξ to learn a set of target patterns. In (4), we see that ξ appears only in the last term of the energy function. Thus, we get

$$\frac{\partial E}{\partial \xi_{\mu i}} = -\mathcal{S}_\mu(h) v_i\ .$$

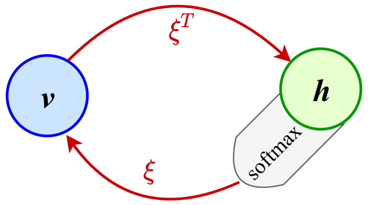
Collecting all the partial derivatives into a matrix $\nabla_\xi E$, we get

$$\nabla_\xi E = -\mathcal{S}(h) \otimes v\ ,$$

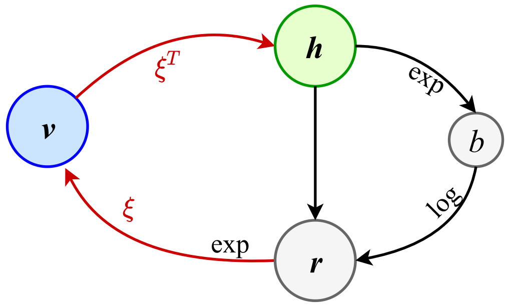
where $\otimes$ is the outer-product. Gradient descent on ξ then leads to

$$\tau_\xi \frac{d\xi}{dt} = \mathcal{S}(h) \otimes v, \qquad (7)$$

where τ_ξ is a time constant that determines the rate at which ξ is learned.



(a) LSE Hopfield network using artificial softmax



(b) LSE Hopfield network using biological softmax

Figure 3: Krotov and Hopfield's network (a) before, and (b) after the addition of the biological softmax subnetwork. Each node represents a population of neurons. Specifically, the $\boldsymbol{v}$ node represents a population of N_v neurons, while $\boldsymbol{h}$ and $\boldsymbol{r}$ each represent a population of N_h neurons. The connection weight matrix is ξ.

To discourage large connection weights, we add a penalty term, $\|\xi\|_F^2$, to the energy function in (4), based on the Frobenius norm of the weight matrix. Gradient descent on that term (with respect to ξ) introduces a weight-decay term,

$$\tau_\xi \frac{d\xi}{dt} = e^r \otimes v - \xi \tag{7}$$

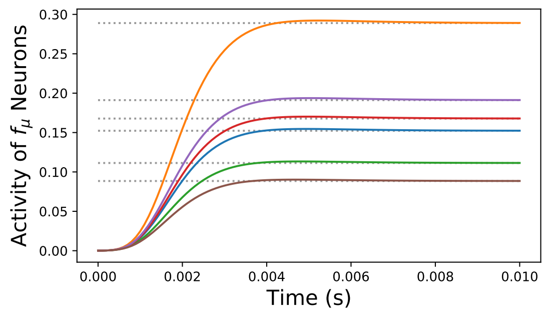
where we have replaced $\mathcal{S}(h)$ with e^r.

Thus, if we clamp the feature neurons to their values for one of the target patterns, and set the hidden neurons to the corresponding one-hot state, the biological LSE network is able to *learn* the connection weights through (7), instead of having them prescribed *a priori*. The other set of connection weights, ξ^T, are updated using the same rule. It should be noted, however, that the r nodes are not local to those connections, so ξ^T essentially constitutes weight copying.

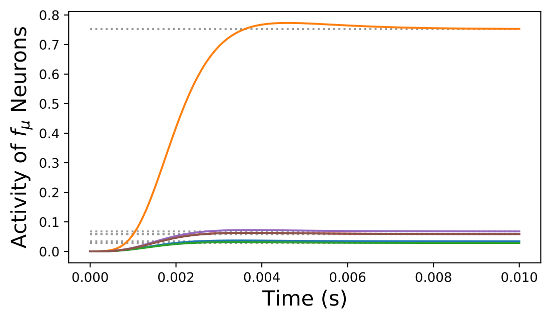
Behaviours of biological LSE Hopfield network

Softmax Network

To show that the softmax subnetwork does, indeed, compute the softmax of h, we ran some experiments in which we held h constant and let the rest of the network (shown in Fig. 2) run to equilibrium. We used 6 hidden nodes,



(a) Random h



(b) One element of h is larger than the others

Figure 4: Examples of output of the softmax subnetwork. The plots show the evolution of the f_μ nodes, as shown in Fig. 2. Here, the solid lines represent the activities of the f_μ nodes, with each colour representing a different node. The horizontal dotted lines show the true values of softmax(h).

and $\tau_s = 1$ ms. Two example runs are shown in Figure 4. It is clear that the f_μ nodes approach the true softmax values, and hence the network successfully computes the softmax of h.

Content Addressable Memory

In this section we look at how the biological LSE network acts as a CAM. Let us start by defining a dataset of patterns, X, which contains N_h patterns, each with N_v bits. Ordinarily, the weight matrix ξ is constructed using the target patterns themselves (i.e. $\xi = X$). However, we *learn* the weights using the biological LSE network. The network acts as a CAM since the feature nodes converge to a *target* pattern, $X[k]$, given a noisy version of that pattern. This behaviour is illustrated in the following experiment.

Convergence to Closest Target: Here we illustrate that the biological LSE network exhibits the same behaviours as a normal HN. This is done by providing our network with some random input pattern, I, and showing that the network converges to the pattern in its memory

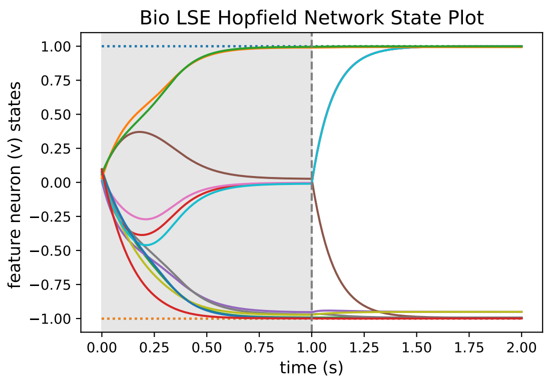


Figure 5: State plot showing the dynamics of the feature neurons when given a random input pattern. Here, the solid lines represent the activities of the feature neurons (v_i), with each colour representing a different neuron. The gray shaded region shows when the input was on.

that is the closest to I. Let X be a dataset of $N_h = 20$ patterns, each with $N_v = 12$ bits (1s and -1s). The input, I, is a random binary pattern. Suppose we compute $\mathcal{H}(I)$, the hamming distance between I and each of the target patterns, and get

$$\mathcal{H}(I) = [6,6,7,9,5,5,5,7,\mathbf{4},8,7,5,6,7,9,5,8,5,7,8].$$

The target pattern closest to the input differs from it by 4 bits. Hence, we would like the network to converge to this closest pattern. This is indeed what happens, as seen in Figure 5. For the first second, while the input is on (shaded grey in the plot), the feature nodes settle to equilibrium values. That equilibrium state does not match any of the target patterns. But once the input is turned off (after 1 second), the feature nodes converge to states corresponding to the closest pattern. It is interesting to note that when the input is on, the 4 **in**correct bits converge to the midpoint between what they are in I and the closest pattern (0 in this case since the patterns are made up of 1s and -1s).

To demonstrate that this behaviour occurs consistently, we ran this experiment 100 times. In each run, a new dataset of target patterns was created, and the network learned these weights by the process described above. Then, with those weights fixed, we initialized all neurons to random values between 0 and 0.1. The input, I, was set to a random binary pattern and held 'on' for 1s, and then turned off. Once the input was off, we continued simulating the network for an additional second to allow convergence. The run is considered a 'success' if the network converged to the pattern with the smallest hamming distance to I. Out of 100 runs, the network successfully converged to the closest pattern 94% of the time.

This experiment demonstrates one of the fundamental properties of HNs; when given a noisy version of one of the patterns, the network corrects the perturbed bits and converges to the closest pattern. This is similar to how humans can read the word "STRABWERRY" with ease since our brain automatically corrects it to the word "STRAWBERRY", which is stored in our memory. Thus, we can conclude that the biological LSE network behaves as a CAM.

Multistability: In this experiment, we illustrate the multistability of the biological LSE network. First, we set the network state to one of its target patterns: its feature nodes contain the target pattern, the hidden nodes are set to the corresponding one-hot state, and the input is turned off. Let us denote this initial target pattern as k_1. Then, for a small period of time, from 0.25s-0.5s, the input is turned on. The input pattern, I, is set to a **different** target pattern, which we will denote k_2. During that brief period of time, pattern k_2 is fed into the feature nodes, after which the input is turned off again, from 0.5s-2s. Figure 6 shows the network starting in pattern k_1. Once the input is turned on, the network starts to shift toward the state of pattern k_2. Even after the input is turned back off, the network continues converging and stabilizes at pattern k_2's state. One way to think of this is that the network is initially "thinking about" the first memory (pattern k_1), then, prompted by some reminder (the pulse of input), the network shifts to the second memory (pattern k_2).

Again, we want to ensure that this is a consistent behaviour for the biological LSE network. To do so, this experiment was run 100 times. In each run, a new dataset of target patterns was created, and the network learned those new weights. Then the network was initialized with target k_1, and then provided with a 0.25s-long input pulse corresponding to a different target, k_2. The network successfully converged to k_2 in 99% of the runs, demonstrating the mulistability of the biological LSE network.

Conclusions

In this paper we present a more biologically plausible modern HN. Modern HNs have received attention for their ability to store a large number of memories and then recall a memory in its entirety prompted by a subset or perturbed version of it. This is notable since human memory can operate in a very similar manner, with memories often being elicited by cues of partial or noisy information. Nevertheless, these networks are biologically unrealistic for a number of reasons, the most prominent being the assumption of symmetric connection weights, and the use of functions that require many-body synapses. Although more biological Modern HNs

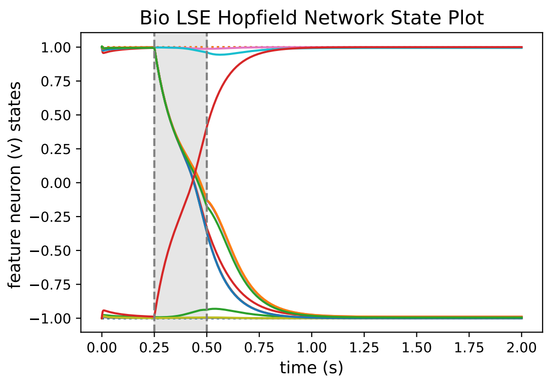


Figure 6: State plot showing the dynamics of the feature neurons when starting in the state of k_1, then switching to the state corresponding to target k_2 after a pulse of input. The solid lines represent the activities of the feature neurons, with each colour representing a different neuron. The gray shaded region shows when the input was on.

have been proposed, some still involve "contrastive normalization", meaning the output of a single neuron is normalized with respect to all other neurons in that layer. In Krotov and Hopfield's LSE network that we focus on here, the output of the hidden neurons is normalized according to the softmax function, which requires the knowledge of other neurons.

We implemented a neural network that computes the softmax function in a more biologically sound manner. Introducing this as a subnetwork to the LSE network overcomes the nonbiological aspects still posed from the traditional softmax function. Our experiments showed that this softmax network dependably computes the softmax, and the resulting biological LSE Hopfield network behaves like a high-capacity Content Addressable Memory.

Future Work

Even though our goal was to demonstrate a biologically plausible implementation of modern Hopfield networks, there are still aspects of the resulting network that are biologically problematic. For example, we learn the connection weights, ξ, using a local learning rule. However, a separate, symmetric set of connection weights, ξ^T, are copied from the learned weights. This weight copying is similar in nature to using a symmetric weight matrix in the original formulation of Hopfield networks, where convergence to a set of steady-state patterns was observed even when ξ was not symmetric (Hopfield, 1982). Also, some work has shown that association networks can still learn even when one set of weights is random (Lillicrap, Cownden, Tweed, & Akerman, 2016). Still, strategies for learning ξ^T are an interesting area for further development.

Another aspect of the softmax network that might be biologically questionable is the use of exp and log as activation functions. While these functions might seem unorthodox in the neural-network community, there is no reason in principle why a neuron could not perform those transformations, so long as the input currents fall within a reasonable range. One possible issue that might arise is if b is small, causing $\log(b)$ to be large and negative. The fact that the log function has a vertical asymptote at $b = 0$ suggests that our model is not valid if all the h_μ values are large and negative.

One last factor that is biologically suspicious is the fact that each target pattern corresponds to the activation of a single hidden neuron. Such one-hot representations are not robust, and not observed in real biology. It is interesting to consider if one could instead learn a combinatorial hidden representation. Of course, such a method would preclude the need to write a paper on a biological implementation of softmax. But a strategy similar to ours might be used to enforce some other attractor dynamics. We leave this as an area for future work.

Acknowledgements

This research was supported by funding from the Natural Sciences and Engineering Research Council (NSERC) of Canada. We also gratefully acknowledge the support of NVIDIA Corporation for the donation of a Titan Xp GPU used for this research.

References

Bogacz, R., & Gurney, K. (2007). The basal ganglia and cortex implement optimal decision making between alternative actions. *Neural computation*, *19*(2), 442–477.

Earhard, M. (1967). Cued recall and free recall as a function of the number of items per cue. *Journal of Verbal Learning and Verbal Behavior*, *6*(2), 257–263.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554–2558.

Krotov, D., & Hopfield, J. (2018). Dense associative memory is robust to adversarial inputs. *Neural computation*, *30*(12), 3151–3167.

Krotov, D., & Hopfield, J. (2020). Large associative memory problem in neurobiology and machine learning. *arXiv preprint arXiv:2008.06996*.

Krotov, D., & Hopfield, J. (2021). Large Associative Memory Problem in Neurobiology and Machine Learning. In *ICLR* (pp. 1–12).

Krotov, D., & Hopfield, J. J. (2016). Dense associative memory for pattern recognition. *Advances in neural information processing systems*, *29*, 1172–1180.

Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2016). Random feedback weights support learning in deep neural networks. *Nature Communications*, *7*, 13276 (2016).

Tulving, E., & Pearlstone, Z. (1966). Availability versus accessibility of information in memory for words. *Journal of Verbal Learning and Verbal Behavior*, *5*(4), 381–391.