ARTICLE

An Interactive Simulation Program for Exploring Computational Models of Auto-Associative Memory

Christian G. Fink
Neuroscience Program and Physics Department, Ohio Wesleyan University, Delaware, OH 43015.

While neuroscience students typically learn about activity-dependent plasticity early in their education, they often struggle to conceptually connect modification at the synaptic scale with network-level neuronal dynamics, not to mention with their own everyday experience of recalling a memory. We have developed an interactive simulation program (based on the Hopfield model of auto-associative memory) that enables the user to visualize the connections generated by any pattern of neural activity, as well as to simulate the network dynamics resulting from such connectivity. An accompanying set of student exercises introduces the concepts of pattern completion, pattern separation, and sparse versus distributed neural representations. Results from a conceptual assessment administered before and after students worked through these exercises indicate that the simulation program is a useful pedagogical tool for illustrating fundamental concepts of computational models of memory.

Key words: memory, computational neuroscience, neural networks, Hebbian plasticity, Hopfield networks

Computational neuroscience has made many valuable contributions to our understanding of the brain (Trappenberg, 2009), with perhaps the most powerful being the elucidation of the biophysical mechanisms underpinning memory. Donald Hebb's notion that "neurons that fire together, wire together" (Hebb, 1949; Lowel, 1992) was shown to be a robust theoretical mechanism for memory storage by John Hopfield over thirty years ago (Hopfield, 1982). While most neuroscience students are familiar with Hebb's adage, it is often conceptually difficult to connect modification of individual synapses with the everyday experience of recalling a memory. How exactly does the alteration of millions of synapses underpin our ability to recall what we ate for breakfast this morning? What is the neural correlate of remembering an event? And what happens on a synaptic level when we *mis*remember an event? Answering such questions from a computational perspective requires an abstract mapping between our common-sense notion of a "memory" and its associated "neural activation pattern."

The family of computational models derived from this perspective is well described in the textbook *Tutorial on Neural Systems Modeling* (Anastasio, 2009), which has been used in a recently-described introductory course in computational neuroscience (Fink, 2016). Anastasio's treatment is mathematically approachable, using just algebra to explore computational models of memory. It does, however, involve computer programming (programs are freely downloadable at http://sites.sinauer.com/anastasio/, which pushes the exploration beyond the reach of many students in a typical Introduction to Neuroscience course.

We have therefore developed an interactive simulation program for students in introductory neuroscience courses to explore computational models of the three stages of episodic memory (encoding, storage, and retrieval). We have also developed a series of in-class exercises (inspired by the Anastasio text) to guide student exploration. These exercises focus on developing conceptual understanding by visualizing the connections formed between neurons in the network model, and they require neither programming experience nor mathematical analysis. This simulation program therefore complements other paradigms for teaching about computational models of memory that involve pencil and paper calculations (Crisp et al., 2015).

To assess the impact of the simulation program and in-class exercises on conceptual understanding of these models, a set of five conceptual questions was answered by students at Ohio Wesleyan University both before and after working through the exercises. Results showed dramatic improvement in student understanding of pattern separation, pattern completion, and both Hebbian and Hopfield synaptic learning rules.

MATERIALS AND METHODS

Simulation Program: The simulation program and in-classes exercises may be downloaded at http://chrisfink.xyz/teaching/teaching_resources.html. The download includes an executable file that may be run on 64-bit Windows computers by simply double-clicking on `memory_app.exe`. The simulation program may also be run on Linux or Mac operating systems, though it is not quite as straightforward. The user must first have Python 3 installed (including the `tkinter`, `numpy`, `time`, and `random` packages). Then they may use the terminal to `cd` to the directory that contains the downloaded program and type `python memory_app_source_code.py`. Either way, running the program opens the window shown in Figure 1.

There are three panes, one for each of the three stages of memory: encoding, storage, and retrieval. In the Encoding pane, the user may define up to five "input patterns" by clicking on a square to represent the activation of a particular neuron (in this model, neurons are either

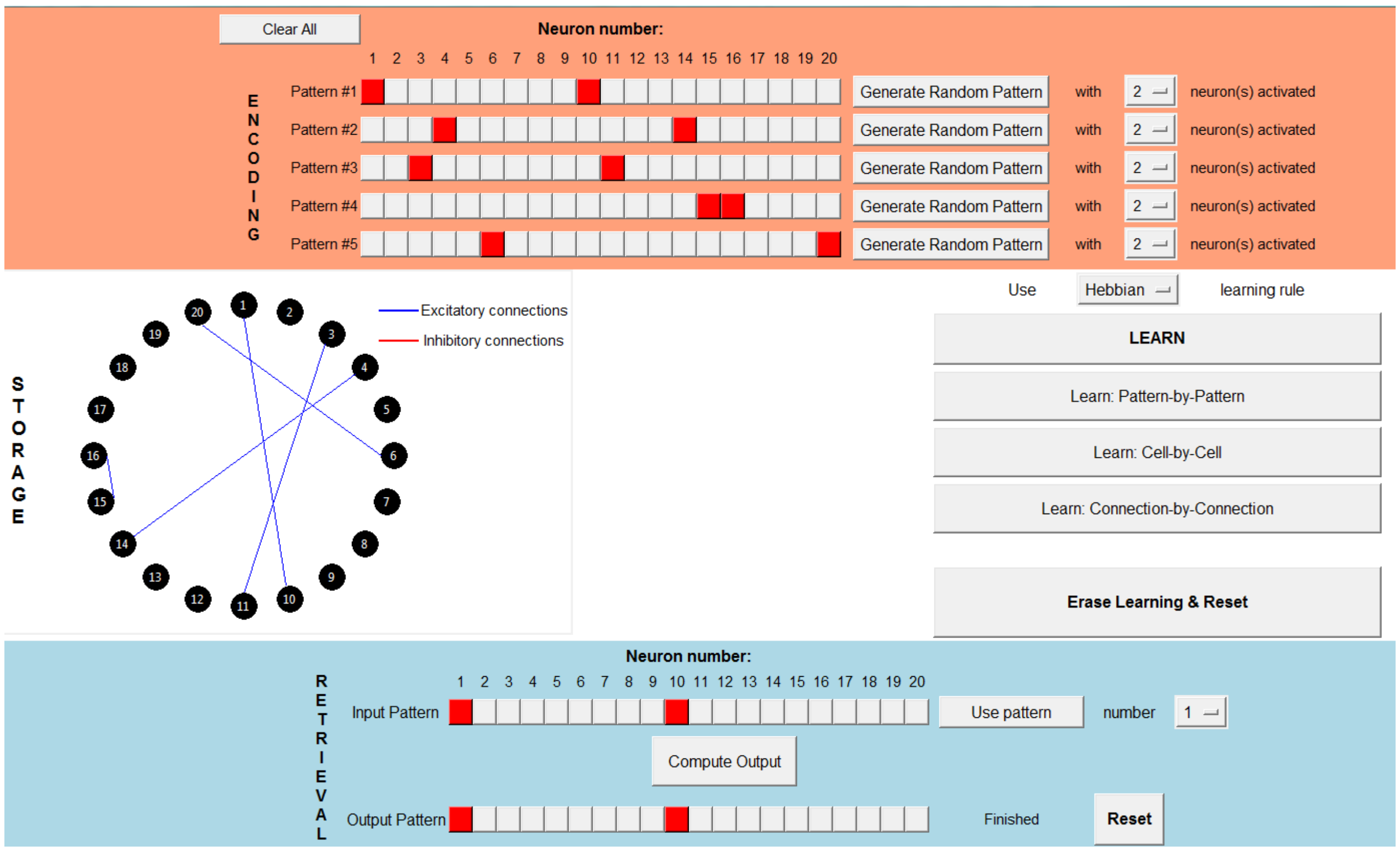

Figure 1. Example window from the memory simulation program. The three stages of episodic memory are represented in three different panes. The user starts by defining up to five different neural activation patterns in the Encoding stage. In the Storage stage, the user chooses either the "Hebbian" or "Hopfield" synaptic plasticity rule, which prescribes how neural activity influences the modification of neuronal connections. In the Retrieval stage the user may test the accuracy of memory storage by starting with an input pattern similar to one of the five input patterns, then pressing the "Compute Output" button to observe how the network activity evolves over time, eventually reaching a steady state. In the example above, Pattern #1 is successfully recalled, facilitated by the excitatory connection between neurons 1 and 10, as well as by the absence of connections between those two neurons and the rest of the network.

fully active, i.e., spiking at their maximal rate, or completely silent). Alternatively, rather than explicitly assigning each individual neural activation, the user may have the program generate a random pattern by specifying the number of neurons to activate and clicking the "Generate Random Pattern" button. Either way, it should be emphasized to students that each input pattern encodes a response to an external stimulus, and these input patterns may be stored as memories and later recalled. Storage is achieved by associating elements of patterns with one another by modifying neural connections (hence why this model of memory is termed "auto-associative").

In the Storage pane, the user must first select which activity-dependent "learning rule" to apply. The Hebbian learning rule applies Hebb's adage, "neuron that fire together wire together," with the understanding that only excitatory connections are possible. In the example shown in Fig. 1, neurons 1 and 10 fire together in Pattern #1, so an excitatory connection forms between neurons 1 and 10. Neurons 4 and 14 fire together in Pattern #2, so an excitatory connection forms between neurons 4 and 14. And so on. The more patterns there are in which two given neurons activate simultaneously, the stronger the excitatory connection between those two neurons becomes.

The other learning rule--known as the Hopfield learning rule, after its originator--is more complex. Both excitatory and inhibitory connections are possible, and in general connections form between virtually all possible pairs of neurons. This learning rule essentially specifies that "neurons that do the same thing have their connections become *more excitatory*, and neurons that do the opposite thing have their connections become *more inhibitory*." By "do the same thing" we mean that two neurons either *both activate* or *both remain silent* within an input pattern. By "more excitatory" we mean that if an excitatory connection already exists between two neurons, and those neurons do the same thing on the next input pattern, then that connection grows stronger; but if an inhibitory connection exists between two neurons, and those neurons do the same thing on the next input pattern, then that connection grows weaker.

Pressing the "LEARN" button in the Storage pane will generate the connections due to all 20 neurons and from all five input patterns all at once. Yet, this will often generate an overwhelming number of connections (especially for the Hopfield learning rule), thus obscuring the relationship between neural *activity* and neural *connectivity* (which is one of the primary points of this simulation program). For this reason, the user may generate connections either one pattern at a time, one cell at a time, or even one connection at a time, using the three buttons beneath the main "LEARN" button.

Once all connections have been generated, the user may move on to the Retrieval pane to test how accurately the input patterns have been stored. The user may define any input pattern to start, though they will most likely be interested in starting with an input pattern very similar to one of the patterns defined in the Encoding pane (modeling, for example, an image similar but not identical to one "seen before"). To do so they may press the "Use pattern" button, which will copy the input pattern specified in the dropdown menu. The user may then alter this pattern by simply clicking on boxes. To test whether a memory is correctly retrieved, the user then clicks the "Compute Output" button.

This runs a simulation which evolves network activity forward in time, according to the connections between neurons displayed in the Storage pane. At a given point in time, the total input to a particular neuron is computed by adding up the connection strengths (which are positive for excitatory connections and negative for inhibitory connections) of all other activated neurons that connect to the neuron in question. If the total input is greater than 0, then the neuron in question is activated; if less than 0, then the neuron is silent. Intuitively, when the network receives an input "hint," in the form of an incomplete pattern, the neurons that have positive connections will force each other to activate, and the neurons that have negative connections will force each other to inactivate, and these interactions will continue until none of the neurons can change their state. At this point we say the network has reached "steady state." If the steady state activation pattern matches one of the five patterns in the Encoding pane, then we say that particular memory has been retrieved.

The mathematical rules governing the evolution of network activity are intentionally hidden from the user so that they may instead focus on conceptual understanding: excitatory connections cause neural activity to spread, while inhibitory connections suppress spreading.

In-class exercises: There are five in-class exercises that accompany the simulation program. It is recommended that students work through these exercises in pairs, and that the instructor lead the class in discussing their responses to each exercise before moving on to the next one. The exercises are as follows:

(1) Simulating encoding, storage, and retrieval. Students are exposed to the idea of an abstract mapping between a particular sensory stimulus and a corresponding pattern of neural activation in the brain. This is what is meant by "neural encoding." A simplistic but concrete example might be that each different input pattern in the Encoding pane is the neural response to a different image. It is very important that students understand the concept of "neural encoding" before moving on to the remaining exercises.

(2) Pattern completion. Students work through an exercise illustrating how the "neurons that fire together wire together" rule results in pattern completion, since a partial activation of one input pattern will spread to the inactivated neurons due to the excitatory connections that have formed between them. Here the opportunity for students to actually *see* the network connections in the Storage pane is key to their making the conceptual connection between activity-dependent plasticity and memory retrieval.

(3) Pattern separation. Students test the accuracy of memory retrieval when overlapping sets of neurons are activated in two different input patterns. They find that when the Hebbian learning rule is used, memory retrieval fails, since the final pattern is a combination of the two input patterns.

(4) Hopfield learning rule. The failure observed in the previous exercise motivates the use of a different learning rule—a rule that allows for the formation of inhibitory as well as excitatory connections. Students first deduce the learning rule itself, then observe that the addition of inhibitory connections prevents the uncontrolled spread of neural activation, thus resulting in accurate memory retrieval. At the end of this exercise, the instructor should tell students that the Hopfield learning has never actually been observed in the brain, while the Hebbian learning rule has…even though the exercise they just completed appeared to show the superiority of the Hopfield learning rule! The final exercise resolves this conundrum.

(5) Sparse versus distributed representations. Students explore the accuracy of memory retrieval for both learning rules under two different circumstances: 2 neurons activated per input pattern (a "sparse" representation), and 10 neurons activated per input pattern (a "distributed" representation). They find that the Hebbian learning rule is more accurate for sparse representations, while the Hopfield learning rule is more accurate for distributed representations. Students are informed that the hippocampus (the area of the brain modeled by these simulations) typically employs a sparse representation, thus explaining why the Hopfield learning rule is not utilized. The instructor should help students brainstorm reasons that the brain might have evolved to prefer sparse representations to distributed representations. (Fewer activated neurons implies lower energy consumption.)

RESULTS

To assess student learning from the simulation program and in-class exercises, fourteen students in Ohio Wesleyan's Introduction to Neuroscience course answered five multiple-choice conceptual questions both before and after working through the in-class exercises (see Supplementary Material). Students completed the questions anonymously, but used coded identifiers so that pre-post performance changes could be tracked. The questions probed students' understanding of pattern separation, pattern completion, neural connectivity generated by both the Hebbian and Hopfield learning rules, and neural encoding. Figure 2 shows that student scores increased significantly after working through the in-class exercises in conjunction with the memory app, suggesting enhanced conceptual understanding of core concepts related to computational models of memory.

Figure 3 shows the question-by-question breakdown of student performance. Performance increased on all conceptual questions, with the largest gains observed on the questions related to pattern separation and pattern completion. Performance on the question related to neural encoding increased only slightly, as students demonstrated a surprisingly good grasp of this concept even before working through the memory app.

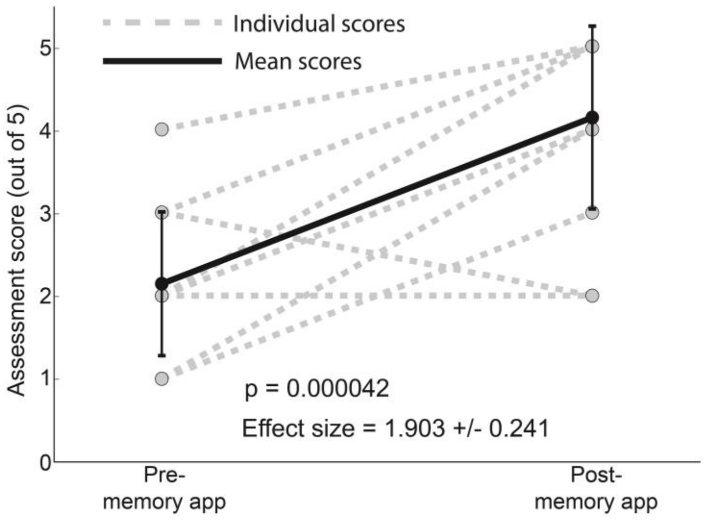

Figure 2. Student assessment scores (out of 5) before and after working through exercises related to the memory app. Dashed lines depict changes in scores for individual students, while solid line portrays the change in mean score (which increased from 2.14 before the memory app to 4.14 afterward). p-value calculated from paired-sample t-test. Effect size calculated using Cohen's d with pooled standard deviation. N=14 students (some lines are on top of one another), and error bars indicate standard deviation.

DISCUSSION

Overall, these results suggest that the interactive simulation program described in this paper is a useful tool for teaching computational models of memory. It is especially useful for developing facility with fundamental computational concepts, such as neural encoding, pattern completion, and pattern separation, without requiring students to perform any mathematical calculations. This work therefore complements the pencil-and-paper neural network exercise developed by Crisp et al. (2015), as well as the pedagogical exercises developed by May (2010) which explore how neural networks are capable of implementing fundamental logical functions.

It is recommended that students work through the accompanying exercises in pairs, so that they can solve problems together. In general, students seemed to enjoy the interactive nature of the simulation program, and it is important to give them an opportunity to ask their own questions and then freely explore the program to discover answers. One of the strengths of the program is that it is a very general model of the three stages of memory, in principle allowing for exploration of many other questions and concepts.

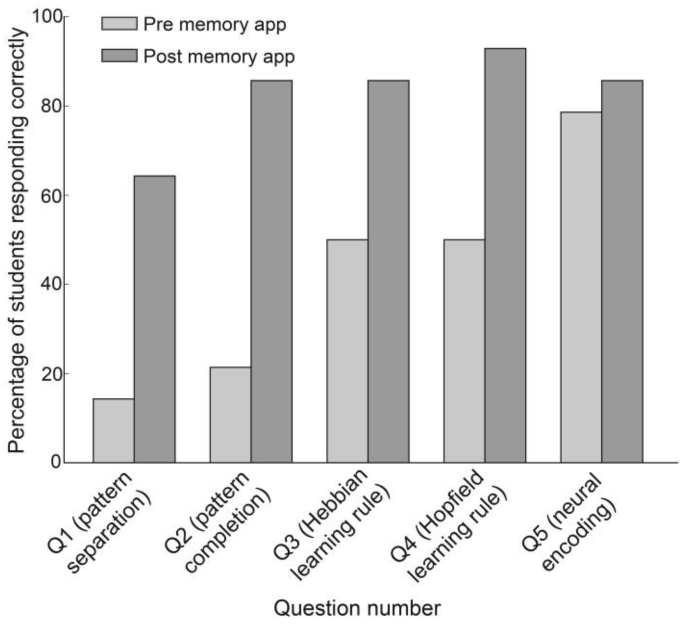

Figure 3. Student performance on each question of the pre/post conceptual assessment. Percentages reflect number of students answering each question correctly out of 14 total students in the class.

ACKNOWLEDGEMENTS

Much thanks to Tom Anastasio for his helpful comments on both the paper and the simulation program.

REFERENCES

Anastasio TJ (2009) Tutorial on neural systems modeling. Sinauer Associates, Incorporated.

Crisp KM, Sutter EN, Westerberg JA (2015) Pencil-and-paper neural networks: an undergraduate laboratory exercise in computational neuroscience. J Undergrad Neurosci Edu 14:A13-A22.

Fink CG (2016) An algebra-based introductory computational neuroscience course with lab. J Undergrad Neurosci Edu 15:A117-A221.

Hebb DO (1949) The organization of behavior: a neuropsychological theory. New York, New York: Wiley & Sons.

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. PNAS 79:2554-2558.

Lowel S, Singer W (1992) Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. Science 255:209.

May CJ (2010) Demonstrations of neural network computations involving students. J Undergrad Neurosci Edu 8:A116-A121.

Trappenberg T (2009) Fundamentals of computational neuroscience. Oxford, England: Oxford University Press

Received February 28, 2017; revised April 26, 2017; accepted May 05, 2017.

This work was supported by NIH Grant No. R01-NS094399.

Address correspondence to: Dr. Christian G. Fink, 61 S. Sandusky St., Ohio Wesleyan University, Delaware, OH 43015. Email: cgfink@owu.edu