Memory and Pattern Recognition in the Abstract Neuron

Analysis of the Simple Hebbian, Oja, and BCM Models in a biological and computational context

Hugh Easton

Middlebury College
Spring 2023

Memory and Pattern Recognition in the Abstract Neuron

Analysis of the Simple Hebbian, Oja, and BCM Models in a biological and computational context

Hugh Easton

Abstract

Memory recollection and the ability to reconstruct signals from partial input is one of the most important capabilities and deepest mysteries of the biological neuron. A suite of models have been designed to understand and imitate this power of the brain, based on the hypothesis that learning occurs through the changing of synaptic connections in the neuron. Here, we pursue the analysis of Hebbian-based models (the Simple Hebbian Model, Oja's Rule, and the BCM Model), all based on the notion that if a neuron persistently takes part in the firing of another neuron, their connection will become stronger. We analyze the utility of each of these rules in both a biological and computational context. Through stability analysis and simulation, we show the instability and shortcomings of the Hebbian Rule, the stability and ability to recognize patterns of Oja's Rule, and the BCM Model's unique ability to both reconstruct and recollect patterns.

Contents

List of Figures

Chapter 1

Introduction

During a famous passage of the novel *Rememberance of Lost Time* by Marcel Proust, the narrator tastes a Madeleine Cake dipped in tea and suddenly vividly recalls his childhood home. This theme comes up several more times throughout the semi-autobiographical work's 4,215 pages. At one point, the sensory feeling of walking on the uneven cobblestones of a street floods the protagonist, Marcel, with memories of his past. You might be familiar with this kind of "involuntary memory" (a term coined by Proust) [3]. Anything from the smell of an old school to the cracks in a familiar city sidewalk can send our mind racing through memories we did not even realize we had access to. It is a concept that has fascinated everyone from philosophers to neuroscientists since ancient times. Freud thought these memories were the gateway to repressed feelings about ourselves, and today, AI researchers attempt to replicate associative memory's power in capturing the meaning of words and images [1] [18]. The desire to understand and recreate this effect has gradually morphed into a theory of the power of the brain: associative memory, and its main ingredient, **synaptic plasticity** [15].

Before diving into the connection between Proust's "involuntary memory" and synaptic plasticity, it is necessary to understand, at a basic level, how the brain works. Our brains are formed by networks of interconnected biological cells called **neurons**. Figure 1.1 illustrates this biological informational unit. At a simple level, one neuron takes inputs

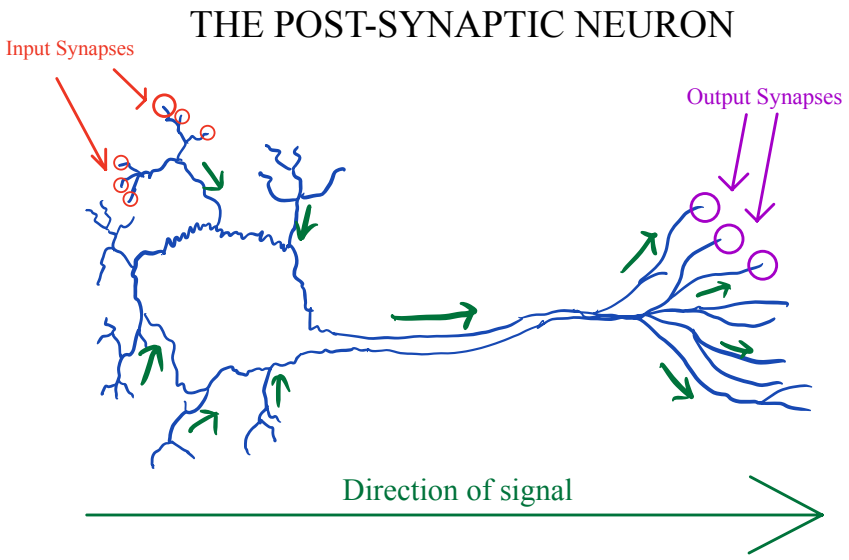

Figure 1.1: A sketch of a simplified post-synaptic neuron. The neuron both receives and outputs a message through specialized connections called *synapses*.

from other neurons, and based on those inputs, it fires into the next neurons it is connected to. These signals travel between neurons at specialized sites known as **synapses**. The neuron firing into another is called a **pre-synaptic neuron**, and the neuron receiving that input is the **post-synaptic neuron**. A basic way of looking at informational processing on this small scale, is the idea that the neuron weights all these inputs or synapses differently, and the output of the neuron is the sum of all of those weights multiplied by their respective synaptic inputs. Learning occurs in the brain via change in the weights of the inputs from different synapses, i.e. different pre-synaptic neurons. These changing weights are called **synaptic weights**, or synaptic strengths, and the process of their change is known as synaptic plasticity [15].

In 1949, Psychologist Donald O. Hebb proposed a law explaining how synaptic plasticity worked towards the associative memory that interests Proust, Freud and AI researchers. He suggested that if one neuron "persistently takes part" in the firing of another, the synaptic weight between the two becomes stronger. This rule is otherwise referred to as the Hebbian Rule or informally, "what fires together wires together" [15]. Let us look at Hebb's Law in the framework of Marcel's Madeleine Cake. In Marcel's childhood he tastes

the cake every Sunday morning. Every Sunday morning, an input pattern representing this taste persistently causes activity in a section of Marcel's brain. At the same time, his neurons are sending messages to register his childhood kitchen, who he is with and even his plans for the day. This happens every Sunday, and each time the neuronal networks holding the information of his surroundings and the sensory input of the cake both fire. According to Hebb's Law, as these seemingly separate networks continue to fire in sync, they become more and more connected. When Marcel tastes the cake twenty years later, this sensory input is received once again. In turn, because this signal is strongly connected to all of these other neurons representing so much of his childhood, they begin firing again, and Marcel starts remembering. The ability to recreate entire sections of memory from one input pattern is fascinating, and not only understanding the process, but being able to recreate it helps to detail the nature of human memory, and the ability to imitate how we learn.

A simple way to understand and mimic the function of a neuron is used in **Artificial Neural Networks (ANNs)**. The fundamental unit of an ANN is an **abstract neuron**. While the neurons used in ANNs are a bit more complex, for our purposes the abstract neuron takes in a vector of n inputs at time t, $\mathbf{x}(t) \in \mathbb{R}^n$, and generates a scalar output, $y(t)$, as the weighted sum,

$$y(t) = \mathbf{x}(t) \cdot \mathbf{w}(t) = \mathbf{x}^\top(t)\mathbf{w}(t) = \mathbf{w}^\top(t)\mathbf{x}(t) = \sum_{i=1}^{n} w_i(t)x_i(t), \tag{1.1}$$

where $\mathbf{w}(t) \in \mathbb{R}^n$ are our synaptic weights from earlier [9]. The neuron in Equation (1.1) is represented visually in Figure 1.2. While this version of a neuron is not a perfect representation of how neurons in our brain actually learn and fire, it is a useful abstraction. We seek to show how the abstraction can exhibit similar learning and memory behavior to a biological neuron, as well as the desired behavior of a simple **unsupervised learner** (a learner who does not need labels to learn patterns).

Our goal is to demonstrate the utility of the abstract neuron (presented in Equation (1.1) and Figure 1.1) in both the biological and computational context of unsupervised

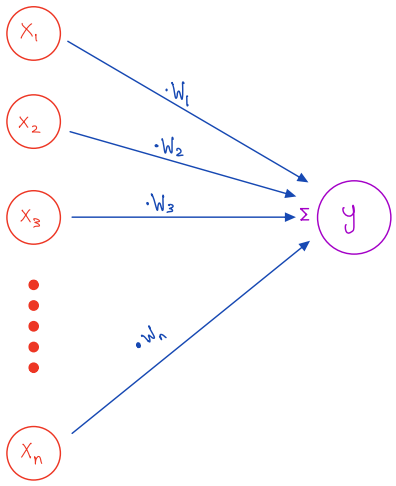

Figure 1.2: A visual representation of a very simple abstract neuron of the kind represented in Equation (1.1). The diagram is colored with the same scheme as the biological neuron in Figure 1.1 according to the functionality of each piece.

learning. In particular, to find the simplest mathematical rule for synaptic learning that provides pattern recognition and recollection for the abstract neuron.

Since Hebb's claim in 1949, many models have tried to emulate the ability of synaptic plasticity to demonstrate learning. Here we analyze three: The Simple Hebbian Model, Oja's Rule, and the BCM Rule, to show the learning capability of the Abstract Neuron. First in Section 2, we will introduce the concept of Simple Hebbian Learning, which will be the motivation for the remainder of our models. We then show the Simple Hebbian Rule is unstable and therefore unsuitable for our purposes. The instability of the rule is the motivation for our next rule in Section 3: Oja's Rule, which fixes the Simple Hebbian instability by normalizing the update synaptic weights. Oja's Rule shows our desired stability as well as the ability to recognize patterns. We demonstrate these features through more stability analysis and simulation. In Section 4, we discuss the BCM rule of learning, which incorporates more biological realism into the learning rule, and has the added ability to recall patterns. Through simulations of the BCM rule, we compare it to Oja's Rule, especially in its ability to remember. Through these steps, we demonstrate that simple math models of plasticity for the basic abstract neuron can learn and recall patterned

input demonstrating the key features of associative memory.

Chapter 2

Simple Hebbian Learning

In his book *The Organization of Behaviour*, Donald O. Hebb characterized learning as follows,

> "When an axon of a cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth or metabolic change takes place in both cells such that A's efficiency, as one of the cells firing B, is increased." [8]

In the most simple terms, this Hebbian Rule is represented over discrete time as

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta y(t)\mathbf{x}(t), \tag{2.1}$$

with synaptic weights, $\mathbf{w}$, output y and input $\mathbf{x}$ as functions of time. The essence of the Hebbian Rule is captured in Equation (2.1) because if there is a high output from our post-synaptic neuron, $y(t)$, at some input with a strong signal from synapse i, at the next time step the corresponding synaptic weight, w_i will be much larger. Hence, a neuron's connections to pre-synaptic neurons correlated with high output become stronger. The learning rate is $0 < \eta \ll 1$, which is chosen to be small in accordance with how weight changes in our brains actually occur. The actual synaptic strengthening that occurs each time a neuron fires is a small fraction of the changes that happen in our brain. This

makes sense, since neurons can fire up to 16 times per second, and drastically changing connections in our brain would present a problem for our daily function [15]. We can also adjust our learning rate to hold this same rule true over continuous time as

$$\frac{d\mathbf{w}}{dt} = y\mathbf{x}, \tag{2.2}$$

to the same effect. The process of moving from discrete to continuous time is shown in Appendix A.1.

2.1 Instability of Simple Hebbian Learning

To investigate the appropriateness of the Hebbian Rule in a biological and computational context, we analyze its stability. As we discuss later, stability in weights is vital for an Abstract Neuron to be neurologically accurate and mathematically useful.

First, we suppose that there *is* a stable fixed point in the space of the synaptic weights; call it $\mathbf{w}^*$. Due to the stochastic nature of the input vector $\mathbf{x}$ at that stable fixed point, we would expect the weights to fluctuate around $\mathbf{w}^*$ over time, but stay relatively close by. In other words, *on average* over time, the weights would be equal to $\mathbf{w}^*$. We denote taking an average over time of a vector $\mathbf{x}$, as $\langle \mathbf{x} \rangle$, so it should be true that,

$$\left\langle \left. \frac{d\mathbf{w}}{dt} \right|_{\mathbf{w}^*} \right\rangle = 0.$$

Therefore, to begin our stability analysis, we take the average over time of the learning rule in Equation (2.2). In order to justify looking at change in weights as an average over time, we assume that the change in synaptic weights happens slower than the changes in input. Recall that according to the abstract neuron in Equation (1.1) and Figure 1.2, the output is represented as a weighted sum of the input, $y = \mathbf{w}^\top \mathbf{x}$. Therefore we can write the time-averaged continuous Hebbian Model as

$$\left\langle \frac{d\mathbf{w}}{dt} \right\rangle = \langle (\mathbf{w}^\top \mathbf{x})\mathbf{x} \rangle. \tag{2.3}$$

Now, before we take that time average, we will simplify our notation further. To do this, first imagine how this learning update in Equation (2.3) applies to an individual weight, $\mathbf{w}_i$:

$$\begin{aligned}\left\langle \frac{dw_i}{dt} \right\rangle &= \left\langle \left(\sum_j w_j x_j \right) x_i \right\rangle \\ &= \left\langle \sum_j w_j x_j x_i \right\rangle.\end{aligned}$$

We define the **correlation matrix**, $\mathbf{C} = \langle \mathbf{x}\mathbf{x}^\top \rangle$, as symmetric with an entry $\mathbf{C}_{ij} = \langle x_i x_j \rangle$ as the product of the time-averaged inputs of synapses i and j. Therefore, the time-averaged change in all weights can be represented

$$\left\langle \frac{d\mathbf{w}}{dt} \right\rangle = \mathbf{C}\mathbf{w}.$$

Returning to our fixed point, $\mathbf{w}^*$, its average change in time must be 0 in order to be a fixed point, and therefore,

$$\mathbf{0} = \mathbf{C}\mathbf{w}^*. \tag{2.4}$$

Now, in order to be a fixed point, $\mathbf{w}^*$ acts as an eigenvector of $\mathbf{C}$ with corresponding eigenvalue zero; this will be important for our stability analysis to come.

To continue, let's observe some things about the correlation matrix $\mathbf{C}$. $\mathbf{C}$ is positive semi-definite. By definition, a matrix is positive semi-definite if and only if for any non-zero vector $\mathbf{a}$, $\mathbf{a}^\top \mathbf{C}\mathbf{a} \geq 0$. This is shown for $\mathbf{C}$ by the following,

$$\begin{aligned}\mathbf{a}^\top \mathbf{C}\mathbf{a} &= \mathbf{a}^\top \langle \mathbf{x}^\top \mathbf{x} \rangle \mathbf{a} \\ &= \langle \mathbf{a}^\top \mathbf{x}^\top \mathbf{x}\mathbf{a} \rangle \\ &= \langle (\mathbf{x}\mathbf{a})^\top (\mathbf{x}\mathbf{a}) \rangle \\ &= \langle (\mathbf{x}\mathbf{a})^2 \rangle \geq 0.\end{aligned}$$

Most importantly, being positive semi-definite means that $\mathbf{C}$ solely has non-negative eigen-

values. Returning to Equation (2.4), imagine we are at the fixed point $\mathbf{w}^*$ which is an eigenvector with eigenvalue zero. Because we are dealing with stochastic input, and $\mathbf{C}$ must have some positive eigenvalues, if there is any fluctuation in the input with a component along a different eigenvector than $\mathbf{w}^*$, then the behavior of $\mathbf{w}$ will grow exponentially along that more dominant eigenvector, since $\mathbf{C}$ must have some positive eigenvalues. Eventually the weights would move in the direction of the eigenvector with the largest eigenvalue, but they will still increase in size unboundedly. Therefore there are only unstable fixed points for the Hebbian Learning Rule [9].

2.2 Simple Hebbian Learning in a Simulated Environment

Unbounded growth for synaptic weights makes little sense in the context of the human brain. The instability of the Hebbian Rule means that unless somehow the synaptic weights are initialized at zero and refuse inputs from pre-synaptic neurons, they will increase towards infinity. Since an actual brain has finite resources, it cannot afford to increase the strengths of these connections to such an extreme.

We simulate the Simple Hebbian Model on three kinds of input to test whether it describes adequate behavior of the abstract neuron. In Figure 2.1**b)** the neuron is trained on fully random input, $\mathbf{x} \in \mathbb{R}^4$, $\forall \mathbf{x}_i \sim \text{uniform}(0, 1)$. This represents a similar input to what an optical sensory neuron would receive with eyes closed. An ideal neuron on this input would not react or increase its output, since it should not recognize a pattern, as experimental evidence shows [6]. When the neuron receives a pattern repeatedly at each time step, for example when our eyes remain looking at the same scene for a period of time, the neuron should increment the highest weights of that input and therefore settle to

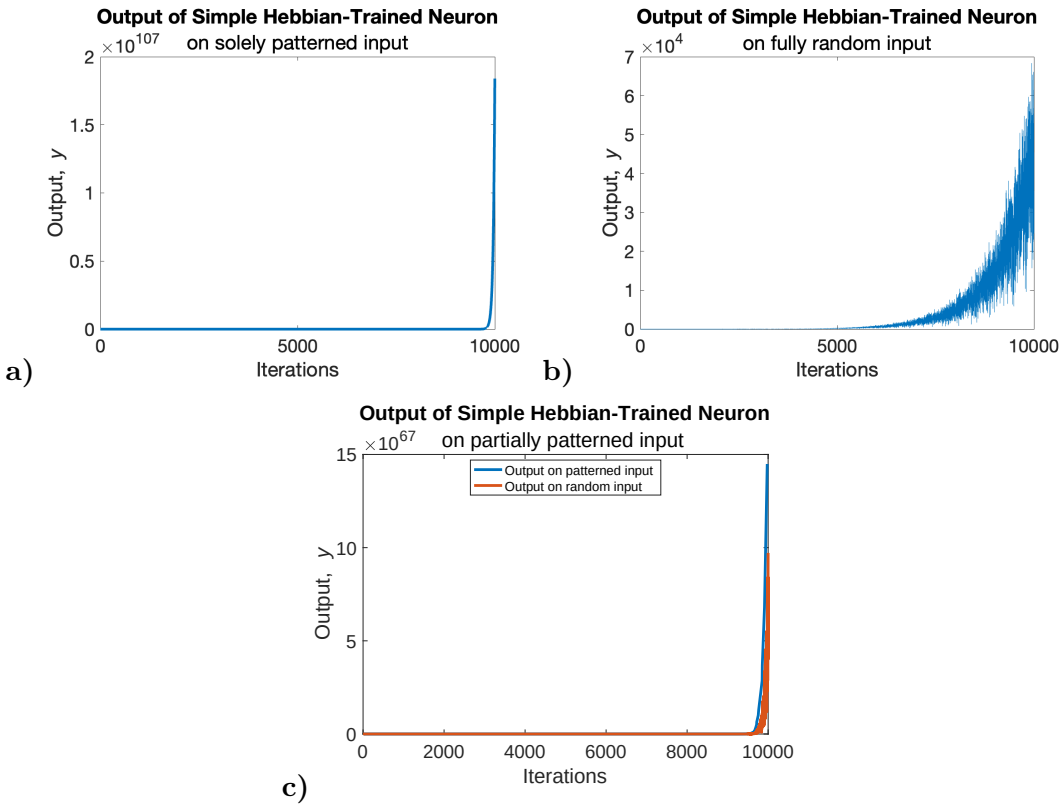

Figure 2.1: The outputs of the Simple Hebbian Model in simulation. The weights of the neuron were trained using the rule in Equation (2.1), with a learning rate of $\eta = 10^{-}3$ for 10,000 iterations. In Plot **a)**, the rule was trained on only one specific pattern of input, then in **b)** on uniform random input, and in Plot **c)**, lastly with a probability of 95% random input and 5% occurring intermittently mixed. In all cases, the Simple Hebbian-trained weights exhibit instability.

a higher output than it provides on random input. However, as when $\mathbf{x}$ is fully patterned,

$$\mathbf{x} = \begin{bmatrix} 5 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}, \tag{2.5}$$

the simulated neuron acts the same and in Figure 2.1**a)**, $y, \mathbf{w} \to \infty$. The neuron does show promise, however, in that it does not grow as fast during fully random input as it does when receiving the same pattern over and over.

To test the Simple Hebbian Rule's ability to decipher between these two inputs (patterned and random), The Hebbian-Trained neuron is simulated around 95% of the time receiving random inputs, and 5% of the time receiving the pattern in Equation (2.5). This is akin to the sensory neurons that fire with the input of a particular scent; our noses do not receive particles encoding that scent 100% of the time. This well-mixed pattern scenario is more accurate to how actual patterned signals appear on a single-neuron scale [17]. High-response to this type of mixed-in pattern would also be beneficial from an unsupervised learning point of view. Distinguishing pattern from noise is important in a wide array of machine learning methods, including Generative Adversarial Networks (GANs) which train by distinguishing real images from noise [5]. As evident in Figure 2.1, the Simple Hebbian neuron's output for both the patterned and random inputs grows unboundedly. While visually, we can decipher between the outputs when the neuron receives the pattern and the noise, to be of any use computationally, this would require some sort of moving window to determine when the threshold output for a pattern had been met. As time goes on through more iterations, we can imagine that the difference between successive patterned input spikes will be orders of magnitude from each other, thereby rendering the training rule difficult to use in a computational setting. From a biological perspective, the output of the neuron on partially patterned input does not make sense. For example, when we smell a cake baking in the oven, it does not generate magnitudes more of recog-

nition in our minds as it did the last time we smelled a baking cake. Therefore, a Simple Hebbian-trained neuron has no usable memory or pattern reconstruction in a biological or computational context. In order to justify the validity of the abstract neuron in Equation (1.1), a stable learning rule for neurons still must be found to exhibit pattern recognition and recollection.

Chapter 3

Oja's Rule

The problem with the Hebbian Rule in Equations (2.1) and (2.2) is that it provides us with no stable fixed weights. The result is an unbounded strengthening of all synapses, inconsistent with the finite resources of biology. There must exist a way to keep the synaptic weights constrained in order to find a model accurate to the way neurons actually train. Here we introduce Oja's Rule which uses normalization to stabilize the weights, and show that this modification gives the Oja-trained neuron the ability to recognize patterns.

We seek to keep the synaptic weights stable in an intuitive way by normalizing the Hebbian Rule. We start with the discretized version of the learning update in Equation (2.1). Following Oja *et al.* [13], we then divide each updated weight vector by the Euclidean Norm of itself (its elements squared, summed and their total square-rooted),

$$\mathbf{w}(t+1) = \frac{\mathbf{w}(t) + \eta y(t)\mathbf{x}(t)}{\|\mathbf{w}(t) + \eta y(t)\mathbf{x}(t)\|}. \tag{3.1}$$

Component-wise, this equation can be written,

$$w_i(t+1) = \frac{w_i(t) + \eta y(t) x_i(t)}{\sqrt{\sum_j \left[(w_j(t) + \eta y(t) x_j(t))^2\right]}}, \tag{3.2}$$

which we can imagine as an update applying to each i^{th} synapse. To show boundedness of each weight call $\hat{w}_i(t) = w_i(t) + \eta y(t) x_i(t)$. Since all weights are positive, it must be

true that

$$\hat{w}_i(t+1)^2 \leq \sum_j (\hat{w}_j(t+1))^2$$
$$\hat{w}_i(t+1) \leq \sqrt{\sum_j (\hat{w}_j(t+1))^2}$$
$$w_i(t+1) = \frac{\hat{w}_i(t+1)}{\sqrt{\sum_j (\hat{w}_j(t+1))^2}} \leq 1.$$

Therefore, the weights in this rule are bounded by 1. Because of its boundedness, we expect Oja's Rule to converge to a particular set of weights given any particular input scheme over time [13].

3.1 Oja's Rule with Penalization and in Continuous Space

Assuming synaptic modification is slow compared to the statistical variations in the input over time, we can find a continuous version of Oja's Rule whose asymptotic paths are the same as the rule in Equation (3.2).

Taking the normalized version of Oja's Rule into continuous time proves difficult; plus, normalization is also inconsistent with biological processes. For example, in the biological neuron of Figure 1.1, input signals are sent into the neuron at different stages along the neuron. The neuron therefore never has the ability to weight a particular input according to the sum of all inputs. Therefore, an equivalent form not involving normalization should be found. To find a more realistic version of Oja's Rule, we expand it as a power series and translate it to a continuous time framework.

Note that in Equation (3.2), during the update, we essentially have two separate values for a weight at the next time step, $w_i(t+1)$. We have the Simple Hebbian weight $\hat{w}_i(t+1)$, call it,

$$\hat{w}_i(t+1) = w_i(t) + \eta y(t) x_i(t),$$

and the Oja Rule version of that weight,

$$w_i(t+1) = \frac{\hat{w}_i(t+1)}{\sqrt{\sum_{j=1}^{n}(\hat{w}_j(t+1))^2}}, \tag{3.3}$$

as defined in Equation (3.2). Then we'll perform a small notation change. We define the Euclidean Norm function as

$$\begin{aligned} N(\hat{w}_1(t+1), ..., \hat{w}_n(t+1)) &= \sqrt{\sum_{j=1}^{n}(\hat{w}_j(t+1))^2} \\ &= \sqrt{\sum_{j=1}^{n}(w_j(t) + \eta y(t) x_j(t))^2}. \end{aligned}$$

Also note that for our normalization, it's true that for any constant a, $aN(w_i) = N(aw_i)$. At any time, t, if we apply the normalization function to each Oja-defined weight, $w_i(t)$,

$$\begin{aligned} N(w_1(t), w_2(t), ..., w_n(t)) &= N\left(\frac{\hat{w}_1(t)}{N(\hat{w}_1(t), ...\hat{w}_n(t))}, ..., \frac{\hat{w}_n(t)}{N(\hat{w}_1(t), ...\hat{w}_n(t))}\right) \\ &= \frac{1}{N(\hat{w}_1(t), ...\hat{w}_n(t))} N(\hat{w}_1(t), ...\hat{w}_n(t)) \\ &= 1. \end{aligned}$$

We will use this property of the Oja-defined weights later on. Now, we apply the normalization to the denominator of the discrete Oja Rule as defined in Equation (3.3),

$$N(\hat{w}_1(t+1), ...\hat{w}_n(t+1)) = N\left(w_1(t) + \eta y(t) x_1(t), ..., w_n(t) + \eta y(t) x_n(t)\right) \tag{3.4}$$

The normalization function still makes for some sloppy analysis, to get rid of the N, we expand the expression on the right side of Equation (3.4) as a Taylor Series with respect to the learning rate η about the point $\eta = 0$, so that it becomes

$$N(\hat{w}_1(t+1), ...\hat{w}_n(t+1)) = N(w_1(t), ..., w_n(t)) + \eta \frac{\partial N}{\partial \eta}\bigg|_{\eta=0} + \mathrm{O}(\eta^2). \tag{3.5}$$

We know from above that $N(w_1(t), ..., w_n(t)) = 1$, and since the learning rate η is very small we can neglect the terms of higher order in η. Then the expression in (3.5) becomes,

$$\begin{aligned}
N(\hat{w}_1(t+1), ...\hat{w}_n(t+1)) &= 1 + \frac{\partial}{\partial \eta}\left[\sqrt{\sum_{i=1}^{n}(w_i(t)^2 + 2\eta w_i(t)x_i(t)y(t) + \eta^2 x_i(t)^2 y(t)^2}\right]_{\eta=0} \\
&= 1 + \left[\frac{1}{2}\left(\sum_{i=1}^{n}(w_i(t))^2 + 2\eta w_i(t)x_i(t)y(t) + \eta^2 (x_i(t))^2 (y(t))^2\right)^{-\frac{1}{2}}\right. \\
&\quad \left.\cdot \left(\sum_{i=1}^{n} 2w_i(t)x_i(t)y(t) + 2\eta (x_i(t))^2 (y(t))^2\right)\right]_{\eta=0} \\
&= 1 + \frac{1}{2}\left(\sum_{i=1}^{n}(w_i(t))^2\right)^{-\frac{1}{2}}\left(\sum_{i=1}^{n} 2w_i(t)x_i(t)y(t)\right) \\
&= 1 + N(w_1(t), ..., w_n(t)) \cdot \frac{1}{2} \cdot \left(\sum_{i=1}^{n} 2w_i(t)x_i(t)y(t)\right) \\
&= 1 + 1 \cdot \left(\sum_{i=1}^{n} w_i(t)x_i(t)y(t)\right) \\
&= 1 + \left(\sum_{i=1}^{n} w_i(t)x_i(t)\right) y(t) \\
N(\hat{w}_1(t+1), ...\hat{w}_n(t+1)) &= 1 + (y(t))^2,
\end{aligned}$$

since according to the abstract neuron in Equation (1.1), our output $y(t) = \sum_{i=1}^{n} w_i(t)x_i(t)$. Now, recall our definition of the next weight in time, $w_i(t+1)$,

$$\begin{aligned}
w_i(t+1) &= \frac{\hat{w}_i(t+1)}{N(\hat{w}_1(t+1), ...\hat{w}_n(t+1))} \\
&= \frac{w_i(t) + \eta x_i(t)y(t)}{1 + (y(t))^2}.
\end{aligned}$$

Again, we can express this component-wise form of the learning rule as a Taylor Series Expansion about $\eta = 0$. We calculate the partial of $w_i(t+1)$ with respect to η, at $\eta = 0$

as,

$$\begin{aligned}\frac{\partial w_i(t+1))}{\partial \eta} &= \frac{\partial}{\partial \eta}\left[\frac{w_i(t)+\eta x_i(t)y(t)}{1+(y(t))^2}\right]_{\eta=0} \\ &= \left[-(w_i(t)+\eta x_i(t)y(t))(1+\eta(y(t))^2)(y(t))^2 + \frac{x_i y(t)}{1+\eta(y(t))^2}\right]_{\eta=0} \\ &= -w_i(t)(y(t))^2 + x_i y(t).\end{aligned}$$

Substitute this into the Taylor Series, and the result is,

$$w_i(t+1) = w_i(t) + \eta y(t)x_i(t) - \eta y(t)^2 w_i(t), \tag{3.6}$$

which is our discrete, component-wise form of Oja's Rule. We can neglect the terms of second order or higher since our learning rate η, is chosen to be very small [13].

For ease of analysis, we take the continuous version of Oja's Rule which asymptotically approaches the behavior of the discrete version [13]. We perform the steps outlined in Appendix A.1, similarly to the Hebbian Rule to get the continuous version of Oja's Rule,

$$\frac{d\mathbf{w}}{dt} = \underbrace{y\mathbf{x}}_{\text{Hebbian Term}} - \overbrace{y^2\mathbf{w}}^{\text{Penalization Term}}. \tag{3.7}$$

Each of the terms present in Equation (3.7) tell us something about how our mathematical neuron is designed for memory. First, there is the basic term from Hebb's rule that "what fires together, wires together," $\mathbf{x}y$. This term ensures the associative learning of our rule, if a synapse has a strong input when the neuron has a high output, the corresponding synaptic connection will be rewarded. The other key aspect, the "penalization term," awards higher weights less on high outputs. This term should keep the stability of our synaptic connections, so the weights cannot get stronger unboundedly. The task left is to prove that this penalization works to keep Oja-trained weights stable.

3.2 Stability of Oja's Rule

To determine the stability of the fixed points of Oja's Rule, we follow similar steps to our stability analysis of Simple Hebbian Learning. First, assuming that the change in inputs happens at a slower rate than the change in synaptic weights, we average the update rule time such that Equation (3.7) becomes,

$$\left\langle \frac{d\mathbf{w}}{dt} \right\rangle = \langle \mathbf{x}y - y^2\mathbf{w} \rangle,$$

which component-wise is,

$$\left\langle \frac{dw_i}{dt} \right\rangle = \left\langle \sum_j w_j x_j x_i - \left[\sum_{j,k} w_j x_j w_k x_k \right] w_i \right\rangle.$$

Now, remember we defined the correlation matrix, $\mathbf{C}$, such that an entry $\mathbf{C}_{ij}$ was equal to the time-averaged product of the inputs x_i and x_j. Therefore, provided the weights $\mathbf{w}(t)$ and the inputs $\mathbf{x}(t)$ are statistically independent, we can take an average over time, and the expression becomes,

$$\left\langle \frac{d\mathbf{w}}{dt} \right\rangle = \mathbf{C}\mathbf{w} - \left[\mathbf{w}^\top \mathbf{C}\mathbf{w} \right] \mathbf{w}. \tag{3.8}$$

To find equilibrium points for the weight vector, we set the update rule equal to zero, and rearrange:

$$\mathbf{C}\mathbf{w} = \left[\mathbf{w}^\top \mathbf{C}\mathbf{w} \right] \mathbf{w}. \tag{3.9}$$

Therefore, a set of weights at equilibrium must be some eigenvector of $\mathbf{C}$, with eigenvalue $\lambda = \mathbf{w}^\top \mathbf{C}\mathbf{w}$. This claim is verified below. We now must show that the eigenvector corresponding to the maximum eigenvalue is the only stable fixed point.

First, consider the normalized eigenvector (and fixed point in weights) $\mathbf{v}_1$ of $\mathbf{C}$, with eigenvalue λ_1, such that $\mathbf{C}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$. We choose a set of weights, $\mathbf{w}$, close to this

eigenvector,

$$\mathbf{w} = \mathbf{v}_1 + \boldsymbol{\epsilon}.$$

The change in this arbitrarily close weight over time can be represented by

$$\frac{d\mathbf{w}}{dt} = \frac{d\mathbf{v}_1}{dt} + \frac{d\boldsymbol{\epsilon}}{dt},$$
$$\frac{d\mathbf{w}}{dt} = \frac{d\boldsymbol{\epsilon}}{dt},$$

since $\frac{d\mathbf{v}_1}{dt} = 0$. According to our averaged learning rule in Equation (3.8),

$$\begin{aligned}
\left\langle \frac{d\mathbf{w}}{dt} \right\rangle = \left\langle \frac{d\boldsymbol{\epsilon}}{dt} \right\rangle &= \mathbf{C}(\mathbf{v}_1 + \boldsymbol{\epsilon}) - \left[(\mathbf{v}_1 + \boldsymbol{\epsilon})^\top \mathbf{C}(\mathbf{v}_1 + \boldsymbol{\epsilon})\right](\mathbf{v}_1 + \boldsymbol{\epsilon}), \\
&= \mathbf{C}\mathbf{v}_1 + \mathbf{C}\boldsymbol{\epsilon} - (\mathbf{v}_1^\top \mathbf{C}\mathbf{v}_1 + \boldsymbol{\epsilon}^\top \mathbf{C}\mathbf{v}_1 + \mathbf{v}_1^\top \mathbf{C}\boldsymbol{\epsilon} + \boldsymbol{\epsilon}^\top \mathbf{C}\boldsymbol{\epsilon})(\mathbf{v}_1 + \boldsymbol{\epsilon}), \\
&= \lambda_1 \mathbf{v}_1 + \mathbf{C}\boldsymbol{\epsilon} - (\mathbf{v}_1^\top \mathbf{C}\mathbf{v}_1)\mathbf{v}_1 - (\boldsymbol{\epsilon}^\top \mathbf{C}\mathbf{v}_1)\mathbf{v}_1 - (\mathbf{v}_1^\top \mathbf{C}\boldsymbol{\epsilon})\mathbf{v}_1 \\
&\quad - (\mathbf{v}_1^\top \mathbf{C}\mathbf{v}_1)\boldsymbol{\epsilon} + \mathrm{O}(\boldsymbol{\epsilon}^2), \\
&= \lambda_1 \mathbf{v}_1 + \mathbf{C}\boldsymbol{\epsilon} - \lambda_1(\mathbf{v}_1^\top \mathbf{v}_1)\mathbf{v}_1 - \lambda_1(\boldsymbol{\epsilon}^\top \mathbf{v}_1)\mathbf{v}_1 - (\mathbf{v}_1^\top \mathbf{C}\boldsymbol{\epsilon})\mathbf{v}_1 \\
&\quad - \lambda_1(\mathbf{v}_1^\top \mathbf{v}_1)\boldsymbol{\epsilon} + \mathrm{O}(\boldsymbol{\epsilon}^2).
\end{aligned}$$

Since $\mathbf{v}_1$ is normal ($|\mathbf{v}_1| = \mathbf{v}_1^\top \mathbf{v}_1 = 1$) and because $\mathbf{C}$ is symmetric, we can write this as

$$\begin{aligned}
\left\langle \frac{d\boldsymbol{\epsilon}}{dt} \right\rangle &= \lambda_1 \mathbf{v}_1 + \mathbf{C}\boldsymbol{\epsilon} - \lambda_1 \mathbf{v}_1 - \lambda_1(\boldsymbol{\epsilon}^\top \mathbf{v}_1)\mathbf{v}_1 - (\boldsymbol{\epsilon}^\top \mathbf{C}\mathbf{v}_1)\mathbf{v}_1 \\
&\quad - \lambda_1 \boldsymbol{\epsilon} + \mathrm{O}(\boldsymbol{\epsilon}^2), \\
&= \mathbf{C}\boldsymbol{\epsilon} - 2\lambda_1(\boldsymbol{\epsilon}^\top \mathbf{v}_1)\mathbf{v}_1 - \lambda_1 \boldsymbol{\epsilon} + \mathrm{O}(\boldsymbol{\epsilon}^2).
\end{aligned}$$

$\left\langle \frac{d\boldsymbol{\epsilon}}{dt} \right\rangle$ is a vector of the averaged rate of change in each of the components of the perturbed weights. We can safely ignore terms of higher order of $\boldsymbol{\epsilon}$, since it is chosen such that each component is small. Now, we take the vector projection of $\left\langle \frac{d\boldsymbol{\epsilon}}{dt} \right\rangle$ onto another normalized eigenvector, $\mathbf{v}_2$, to find the component of the average change of the perturbation over time

in the direction of another fixed point. This projection is,

$$\begin{aligned}\mathbf{v}_2^\top \left\langle \frac{d\boldsymbol{\epsilon}}{dt} \right\rangle &= \mathbf{v}_2^\top \mathbf{C}\boldsymbol{\epsilon} - 2\lambda_1(\boldsymbol{\epsilon}^\top \mathbf{v}_1)\mathbf{v}_2^\top \mathbf{v}_1 - \lambda_1 \boldsymbol{\epsilon}\mathbf{v}_2^\top \\ &= \lambda_2 \mathbf{v}_2^\top \boldsymbol{\epsilon} - 2\lambda_1(\boldsymbol{\epsilon}^\top \mathbf{v}_1)\mathbf{v}_2^\top \mathbf{v}_1 - \lambda_1 2\lambda_1(\boldsymbol{\epsilon}^\top \mathbf{v}_1) - \lambda_1 \mathbf{v}_2^\top \boldsymbol{\epsilon} \\ &= (\lambda_2 - \lambda_1)\mathbf{v}_2^\top,\end{aligned}$$

keeping in mind at the last step that the distinct eigenvectors are orthogonal since $\mathbf{C}$ is symmetric. The expression $\mathbf{v}_2^\top \langle \frac{d\boldsymbol{\epsilon}}{dt} \rangle = (\lambda_2 - \lambda_1)\mathbf{v}_2^\top$, is only negative if λ_1 is a larger eigenvalue than any other λ_2. Therefore, since we showed our only fixed points are eigenvectors of $\mathbf{C}$, the distance between the current weight value and a fixed point will shrink *only if the fixed point is the maximal eigenvector*. The only stable fixed point of Oja's rule is at the dominant eigenvector. In the long-term, under training in Oja's Rule, we expect the weights, $\mathbf{w}$, to head towards the dominant eigenvector of $\mathbf{C}$ [9].

Now, a stable fixed point in the weights of Oja's rule does not necessarily prove their convergence to the dominant eigenvector. However, we have also shown the boundedness of the discrete version of this rule in our earlier introduction. Therefore, it is reasonable to assume that the system of weights is not unstable, and engages in some kind of long-term behavior towards the fixed point. A fuller proof of convergence of the stochastic differential equation has been made by Oja *et al.* [14]. Through simulation we show that even when the necessary assumptions are not met, the weights will still converge towards the dominant eigenvector.

3.3 An Oja-Trained Neuron in a Simulated Environment

When we simulate the synaptic growth of the Oja-Trained Neuron, we are interested in a few aspects of the neuron as both a biological representation and a computational tool. As described before, a good learning tool should demonstrate stability, an ability to distinguish patterns, and to recall those patterns. We simulate the neuron in random and patterned environments to see that it demonstrates these desired aspects.

We use simulations to test for the stability of Oja's Rule, first by running a simulation feeding the neuron only identical input. In the Simple Hebbian simulation, this provoked the neuron into higher output quicker than other forms of input, so if anything should show instability in Oja's rule, this should be it. While the results of this simulation hold true for any positive-valued patterned input, the input we choose to feed to our neuron takes the same form,

$$\mathbf{x} = \begin{bmatrix} 5 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix} \tag{3.10}$$

A correlation matrix of only this input would therefore look like,

$$\mathbf{C} = \mathbf{x}\mathbf{x}^\top = \begin{bmatrix} 25 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.01 & 0.01 & 0.01 \\ 0.5 & 0.01 & 0.01 & 0.01 \\ 0.5 & 0.01 & 0.01 & 0.01 \end{bmatrix},$$

with a dominant eigenvalue of $\lambda_1 = 25.03$ and a corresponding eigenvector of

$$\mathbf{v}_1 = \begin{bmatrix} 0.994 \\ 0.02 \\ 0.02 \\ 0.02 \end{bmatrix}.$$

Using the correlation matrix for this patterned input and setting the synaptic weights

equal to this eigenvector verifies the claim in Equation (4.14) that the eigenvalue

$$\lambda = \mathbf{w}^\top \mathbf{C}\mathbf{w} = \mathbf{v}_1^\top \mathbf{C}\mathbf{v}_1,$$

$$25.03 = \begin{bmatrix} 0.994 & 0.02 & 0.02 & 0.02 \end{bmatrix} \begin{bmatrix} 25 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.01 & 0.01 & 0.01 \\ 0.5 & 0.01 & 0.01 & 0.01 \\ 0.5 & 0.01 & 0.01 & 0.01 \end{bmatrix} \begin{bmatrix} 0.994 \\ 0.02 \\ 0.02 \\ 0.02 \end{bmatrix},$$

$$25.03 = \begin{bmatrix} 24.88 & 0.4976 & 0.4976 & 0.4976 \end{bmatrix} \begin{bmatrix} 0.994 \\ 0.02 \\ 0.02 \\ 0.02 \end{bmatrix} \approx 25.03.$$

The results of simulating learning on the pattern in Equation (3.10) can be seen in Figure 3.1. The weights converge to the dominant eigenvector of the correlation matrix therefore demonstrating the desired stability of this learning rule even with fully correlated inputs. A similar stability is found on uniformly random inputs, as shown in Figure 3.2. For uniformly random inputs, the expected value, $\mathbb{E}\left[\text{unif}(0,1)\right] = 0.5$, meaning,

$$\langle \mathbf{x} \rangle = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix},$$

and the corresponding correlation matrix $\mathbf{C}$ is

$$\mathbf{C} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}.$$

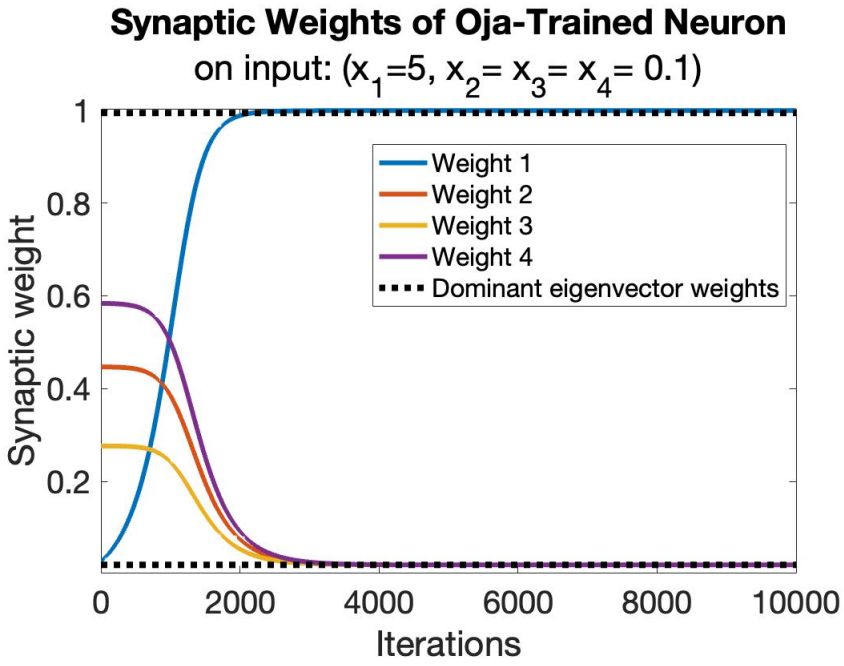

Figure 3.1: Oja's rule is simulated only receiving one specific pattern, Equation (3.10), as input. The simulation was run for 10^4 iterations and a learning rate of $\eta = 10^{-4}$, the synaptic weights converge to the dominant eigenvector of the correlation matrix.

The dominant eigenvector, corresponding to an eigenvalue of $\lambda = 1$, is

$$\mathbf{v} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}.$$

In Figure 3.2, it is shown that the synaptic weights simulated under random input do in fact converge to this fixed point. So, the Oja-trained neurons demonstrate stability on both random and patterned inputs, but it remains to be shown if they can differentiate between the two.

To test if utilizing Oja's rule on real, randomly-generated inputs will give it the ability to learn the difference between patterned and noisy inputs, we perform an identical simulation to the Simple Hebbian Model, mixing 5% patterned with 95% random inputs. Since the patterned input in Equation (3.10) has a larger magnitude than is possible for

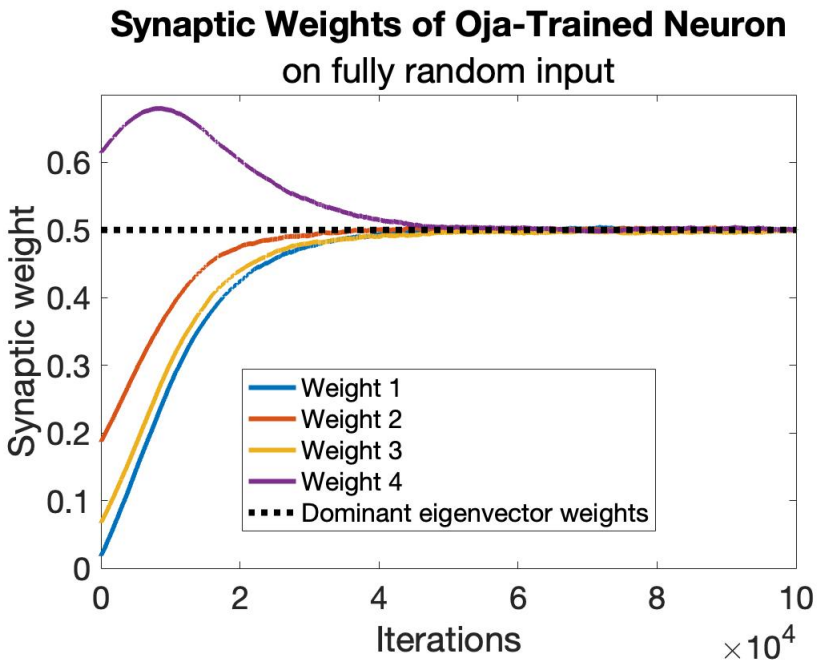

Figure 3.2: Oja's rule is simulated using input randomly chosen from a uniform distribution, for 10^5 iterations. Parameters are otherwise identical to the simulation shown in Figure 3.1.

the random input, Oja's rule should learn to maximize output on this pattern. The results can be seen in Figure 3.3. The neuron is still able to find the most important (largest magnitude) synaptic input x_1 in this pattern and weigh it the heaviest. Any partial input that is strong in the input x_1, will trigger a strong response from this neuron. This is the input reconstruction we see in the ability to visualize a hamburger upon just smelling it, or how Marcel can recall his childhood from the taste of a cake. It remains to be shown, however, that the Oja-Trained Neuron can retain this connection for enough time.

To test whether the Oja Neuron can retain the connections it has made to receive patterns for a long enough time, as real associative memory does in humans, we train the neuron solely on patterned input, then a period of noise for an interval, before giving the neuron that same pattern again. We use the input pattern in Equation (3.10) for our simulation once more. The simulation is run for 10,000 iterations, with a learning rate of $\eta = 10^{-3}$. For the first 1,000 iterations, the neuron is exposed to only the patterned input such that the weights stabilize at the dominant eigenvector. Then, for the next

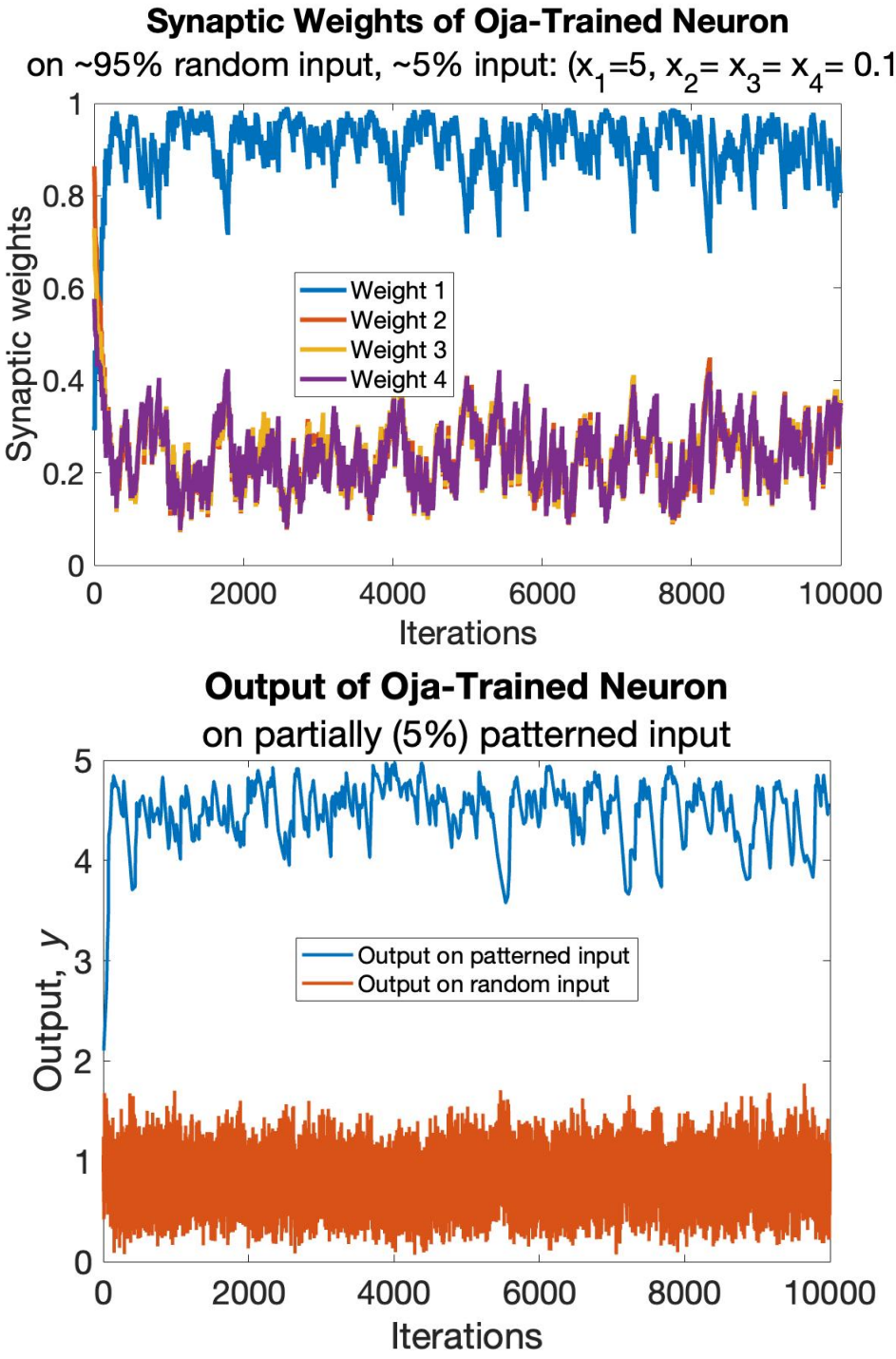

Figure 3.3: The Oja-Trained Neuron's weights and output on a well-mixed input of 95% noise and 5% pattern with a magnitude higher than the noise.The learning rate is $\eta = 10^{-3}$. We can tell the neuron is learning this pattern because the weights begin to favor the synapse that receives the largest input in the pattern, x_1.

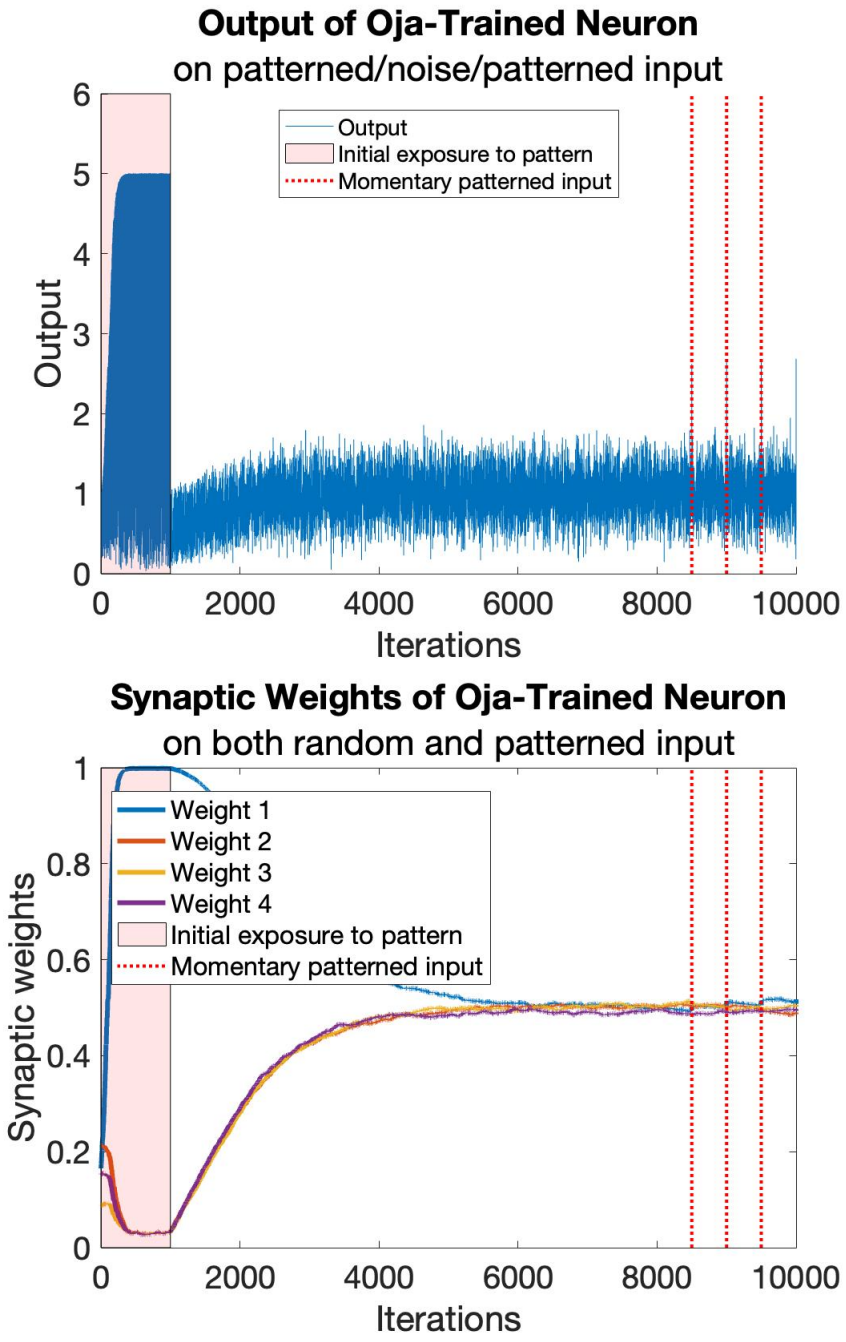

Figure 3.4: The neuron is trained on 1,000 iterations of patterned input after which it is exposed to only noise. After 7,500 more iterations, the neuron is given the pattern four more times to test its memory. The learning rate is $\eta = 10^{-3}$. We see the Oja-Trained Neuron forgets this pattern after enough time.

7,500 iterations, the neuron is exposed to only noise, after which the pattern is shown to it 4 separate instances. The results are presented in Figure 3.4. Over enough time, we can see this Oja model forget the large weight that corresponds to the largest synaptic input. Therefore, provided a large enough learning rate or a large enough time-scale with no patterned input, the Oja-trained weights would almost surely forget the input. Recall that in Proust's story involving associative memory, the Madeleine Cake's sensory pattern triggered memories many years earlier from the protagonist's childhood. The Oja Rule provides no such ability in the synaptic plasticity it creates. To mimic the effect of human recollection, a more robust model is needed, this is pursued in the following section.

Chapter 4

The Bienenstock-Cooper-Munro (BCM) Model

The BCM Model was created in 1982 as an expansion of Hebb's Law designed to explain the **selectivity** of neurons in **sensory areas** of the brain. These are areas that deal with patterned inputs in taste, smell or feel, like the memories from Proust's novel [1]. Selectivity is a measurement of the difference between a neuron's maximum and mean outputs, defined as

$$\text{Sel}_{\mathbf{x}}(\mathbf{w}) = 1 - \frac{\mathbb{E}\left[\mathbf{x}^\top \mathbf{w}\right]}{\sup(\mathbf{x}^\top \mathbf{w})}. \tag{4.1}$$

In Equation (4.1), $\mathbb{E}\left[\mathbf{x}^\top \mathbf{w}\right]$ is the expected value of the output of the neuron, and $\sup(\mathbf{x}^\top \mathbf{w})$ is the essential supremum of the output of the neuron which we can think of as the maximum output of the neuron. A neuron that has a similar mean output and maximum output makes it difficult to differentiate between normal noise in output, and a strong output; according to Equation (4.1), the result would also be a low selectivity. If maximum output is much greater than expected output, the fraction $\frac{\mathbb{E}\left[\mathbf{x}^\top \mathbf{w}\right]}{\sup(\mathbf{x}^\top \mathbf{w})}$ would end up small, and therefore selectivity would be large. BCM sought to *maximize* the selectivity of the abstract neuron from Equation (1.1) [4].

The BCM model chooses to address the problem of instability in the Simple Hebbian Model differently than the preceding models. While Oja's Rule uses normalization and a

"forgetting" term to penalize high weights, it also results in an inability to differentiate between two inputs similar in magnitude. Note that in all the above simulations, the effect of differentiating noise from patterns is accomplished by ensuring patterned input has a larger magnitude than noise. Bienenstock, Cooper and Munro chose a more biologically-motivated approach. The BCM Model uses temporal competition between input patterns to determine if weights are increased or decreased. The general idea behind this is to introduce some threshold, θ, such that when the output, $y(t)$, goes below that threshold, the corresponding synaptic weights are penalized. If the output is above that threshold, the weights are increased. The BCM rule proposed by Bienenstock *et al.* in 1982 takes the general form

$$\frac{d\mathbf{w}}{dt} = \phi(y(t), \theta)\mathbf{x}(t) - \epsilon\mathbf{w}(t) \tag{4.2}$$

$$\phi(y(t), \theta) = \begin{cases} -\eta_w, \text{ if } y(t) < \theta, \\ \eta_w, \text{ if } y(t) > \theta, \\ 0, \text{ if } y(t) = \theta, \end{cases} \tag{4.3}$$

where $0 < \eta_w \ll 1$, is the learning rate for the synaptic weights [4]. In Equations (4.2) and (4.3), the direction of the weights' change is based on whether the output is above or below some **modification threshold**, θ. The term $-\epsilon\mathbf{w}(t)$ implies uniform decay, although that term is not weighted as highly as in Oja's rule and is multiplied by a small constant, $0 < \epsilon \ll 1$. Later formulations of the rule (including the version used here) drop the uniform decay term altogether [6] [10] [7] [11]. Now, ϕ, and θ must be found to ensure biological realism and utility in computation, and to keep the BCM rule stable.

An early choice of the threshold function was to set it as a constant value, but this resulted in unstable weights, just as in the Simple Hebbian Model. To show this, imagine we choose some fixed threshold, θ_{fixed}, and any fixed point in weights $\mathbf{w}^*$. We should see

that over time, the average change in weight, along this point, $\left\langle \frac{d\mathbf{w}}{dt} \right\rangle = 0$. Therefore,

$$0 = \left\langle \left. \frac{d\mathbf{w}}{dt} \right|_{\mathbf{w}^*} \right\rangle = \langle \phi(y(t), \theta_{\text{fixed}}) \mathbf{x}(t) \rangle.,$$
$$= \phi(\bar{y}, \theta_{\text{fixed}}) \bar{\mathbf{x}},$$

where $\bar{y}, \bar{\mathbf{x}}$ denotes an overall average over time. Again, we assume the synaptic weights change on a much slower time scale than the input signals, and that the inputs are uncorrelated. In order for $\mathbf{w}^*$ to be a fixed point, the input to the neuron must always be $\mathbf{x} = \mathbf{0}$ (which would make no sense for a functional neuron), or

$$\bar{y} = \mathbf{w}^* \bar{\mathbf{x}}^\top = \theta_{\text{fixed}}.$$

Imagine we perturb some small distance from the fixed point $\mathbf{w} = \mathbf{w}^* + \boldsymbol{\delta}$, then

$$\bar{y} = (\mathbf{w}^* + \boldsymbol{\delta}) \bar{\mathbf{x}}^\top \neq \theta_{\text{fixed}}.$$

If the perturbation $\boldsymbol{\delta} > \mathbf{0}$, then $\phi > 0$, and the weights are driven down further. If $\boldsymbol{\delta} < \mathbf{0}$, then $\phi < 0$, and the weights are incremented. In both cases, since the input is positive, this perturbation changes $\left\langle \frac{d\mathbf{w}}{dt} \right\rangle$ such that the weights are driven further away from $\mathbf{w}^*$. Therefore a fixed threshold results in unstable fixed points. A moving threshold is proposed to help fix this instability problem.

The original BCM Model proposed a choice of ϕ using the time-averaged output of the neuron, $\bar{y}$,

$$\phi(y(t), \theta) = y(y - \theta), \tag{4.4}$$
$$\theta = (\bar{y})^p, \tag{4.5}$$

which provides stability for choice of $p > 1$ [4]. The choice of the threshold as a moving-time average also results in weights of low-selectivity being *unstable*. To show this, imagine

a set of weights with zero selectivity where the average output equals the maximum output. Because the threshold for weight change is the average output, on a higher-than-average output, the weights are incremented, causing more higher than average outputs and more synaptic strengthening. On a lower-than-average output, the synaptic weights fall and the outputs begin to to be lower than average (causing more weight weakening). Logically, this should also result in stable fixed points being of high selectivity [4]. A moving-time average threshold is also consistent with monocular eye-deprivation experimental results; for example, when researchers deprived a mouse of sight in one eye, the neurons receiving sensory information from the non-covered eye doubled in firing rate [6].

Since the original paper, other learning rules and choices of θ-modification schemes and $\phi(y(t), \theta)$ have been devised for specific applications such as one for receptive fields in visual environments [11], or using a spatial average instead of a time average [10]. We instead take the commonly-used approach of replacing the temporal average with a first-order low-pass filter [7] [16] [17]. This changes the rule stated in Equations (4.2) and (4.3) to

$$\frac{d\mathbf{w}}{dt} = \eta_w y \mathbf{x}(y - \theta), \tag{4.6}$$

$$\frac{d\theta}{dt} = \eta_\theta (y^2 - \theta), \tag{4.7}$$

with rates of change for the weights and threshold, $0 < \eta_w < \eta_\theta$, since θ should change at a rate faster than the output. Equation (4.3) works as simply a moving approximation of the squared average output, $\bar{y}^2$. At each increment of time, the update moves the threshold η_θ of the way along the direction from θ to $\bar{y}^2$, thus this rule works as a kind of a approximate moving average to the original threshold proposed by Bienenstock *et al.* [4].

4.1 Stability of the BCM Model

As established in previous chapters, for biological and computational competence, the BCM Model must exhibit stability. For this stability analysis, we pursue a different technique than for the earlier models. Instead of focusing on the change in *weight* of our neuron over time, we choose to follow the change in *output* of the abstract neuron over time, as done by Udeigwe *et al.* [17]. We choose to analyze the output because, provided the input remains relatively stable over time, if the output $y = \mathbf{w} \cdot \mathbf{x}$ remains stable, so must the weights.

We start by assuming a fixed set of only two possible normalized, noncollinear inputs that this neuron receives, $\mathbf{X} = \{\mathbf{x}_1 = (x_{11}, x_{12}),\ \mathbf{x}_2 = (x_{21}, x_{12})\}$, and their corresponding outputs, $y_1 = \mathbf{w}^\top \mathbf{x}_1,\ y_2 = \mathbf{w}^\top \mathbf{x}_2$. The neuron receives the input $\mathbf{x}_1$ with probability ρ, and $\mathbf{x}_2$ with probability $1 - \rho$. The rate of change of the synaptic weights can therefore be written as

$$\frac{d\mathbf{w}}{dt} = \rho x_{1i} y_1 (y_1 - \theta) + (1 - \rho) x_{2i} y_2 (y_2 - \theta), \quad i \in \{1, 2\},$$

from Equations (4.6) and (4.7). Recalling that $y = \mathbf{w}^\top \mathbf{x}$, changes in output are written,

$$\frac{dy_1}{dt} = x_{11}\frac{d\mathbf{w}_1}{dt} + x_{12}\frac{d\mathbf{w}_2}{dt},$$
$$\frac{dy_2}{dt} = x_{21}\frac{d\mathbf{w}_1}{dt} + x_{22}\frac{d\mathbf{w}_2}{dt}.$$

We are interested in the changes in the outputs themselves, which using the above equations, we can express as,

$$\frac{dy_1}{dt} = \eta_w \left[\rho \mathbf{x}_1 \mathbf{x}_1^\top y_1 (y_1 - \theta) + (1 - \rho)\mathbf{x}_1 \mathbf{x}_2^\top y_2 (y_2 - \theta)\right], \tag{4.8}$$
$$\frac{dy_2}{dt} = \eta_w \left[\rho \mathbf{x}_1 \mathbf{x}_2^\top y_1 (y_1 - \theta) + (1 - \rho)\mathbf{x}_2 \mathbf{x}_2^\top y_2 (y_2 - \theta)\right], \tag{4.9}$$
$$\frac{d\theta}{dt} = \eta_\theta \left[\rho y_1^2 + (1 - \rho) y_2^2 - \theta\right]. \tag{4.10}$$

First, we find the threshold parameter, θ, at equilibrium. From Equation (4.19), we can say that at equilibrium

$$\theta = \rho y_1^2 + (1-\rho) y_2^2. \tag{4.11}$$

Note, since the two inputs are not collinear and their probability $\rho \in (0,1)$, the Equations (4.8) and (4.9) for $\frac{dy_1}{dt}$, and $\frac{dy_2}{dt}$ are zero if and only if for one choice of $j \in \{1,2\}$, $y_j(y_j - \theta) = 0$. Also assuming that the inputs $\mathbf{x}_1, \mathbf{x}_2 \neq \mathbf{0}$ and $\rho \neq 1$, we substitute the value for θ in Equation (4.11) into the change in output from Equations (4.8) and (4.9), and we find,

$$0 = y_1\big(y_1 - \big(\rho {y_1}^2 + (1-\rho){y_2}^2\big)\big), \tag{4.12}$$

$$0 = y_2\big(y_2 - \big(\rho {y_1}^2 + (1-\rho){y_2}^2\big)\big). \tag{4.13}$$

This shows us the fixed points of the change in output are when

$$(y_1, y_2, \theta) = \left\{(0,0,0), \left(\frac{1}{\rho}, 0, \frac{1}{\rho}\right), \left(0, \frac{1}{1-\rho}, \frac{1}{1-\rho}\right), (1,1,1)\right\}. \tag{4.14}$$

Before finding the stability of these fixed points, to make analysis a little easier for us, we will rescale time in terms of one of the learning rates, η_w. We will define the rescaled time dimension as,

$$\hat{t} = \frac{t}{\eta_w}, \tag{4.15}$$

and therefore,

$$\frac{d}{d\hat{t}} = \frac{1}{\eta_w} \cdot \frac{d}{dt}. \tag{4.16}$$

We will also scale the inputs such that $\mathbf{x}_1 \mathbf{x}_1^\top = 1$. Therefore the magnitude of the other input $\mathbf{x}_2$ is simply scaled according to the magnitude of the first without loss of generality.

The rescaled time derivatives in Equations (4.8) through (4.19) are then,

$$\frac{dy_1}{d\hat{t}} = \rho y_1(y_1-\theta) + (1-\rho)\mathbf{x}_1\mathbf{x}_2^\top y_2(y_2-\theta), \tag{4.17}$$

$$\frac{dy_2}{d\hat{t}} = \rho \mathbf{x}_1\mathbf{x}_2^\top y_1(y_1-\theta) + (1-\rho)\mathbf{x}_2\mathbf{x}_2^\top y_2(y_2-\theta), \tag{4.18}$$

$$\frac{d\theta}{d\hat{t}} = \frac{\eta_\theta}{\eta_w}\left[\rho y_1^2 + (1-\rho)y_2^2 - \theta\right]. \tag{4.19}$$

Using these differential equations rescaled with respect to time, we calculate the Jacobian matrix (a linearization about the fixed points) by taking the derivatives with respect to each independent variable ($y_1, y_2,$ and θ) in Equations (4.17) through (4.19). To simplify notation, from here on we call $a = \mathbf{x}_2\mathbf{x}_2^\top, b = \mathbf{x}_1\mathbf{x}_2^\top, c = \frac{\rho}{1-\rho}, \eta = \frac{\eta_w}{\eta_\theta}$. The Jacobian of this system of differential equations is expressed as,

$$J(y_1,y_2,\theta) = \begin{bmatrix} 2\rho y_1 - \rho\theta & 2(1-\rho)by_2 - (1-\rho)b\theta & -\rho y_1 - (1-\rho)by_2 \\ 2\rho b y_1 - \rho b\theta & 2(1-\rho)ay_2 - (1-\rho)a\theta & -\rho b y_1 - (1-\rho)ay_2 \\ \frac{2}{\eta}\rho y_1 & \frac{2}{\eta}(1-\rho)y_2 & -\frac{1}{\eta} \end{bmatrix}. \tag{4.20}$$

To find the stability of the fixed points, in particular to show that the points $(\frac{1}{\rho}, 0, \frac{1}{\rho})$ and $(0, \frac{1}{1-\rho}, \frac{1}{1-\rho})$ are stable, we find the eigenvalues of the Jacobian at these points.

First, we demonstrate the stability of the point $(y_1, y_2, \theta) = (\frac{1}{\rho}, 0, \frac{1}{\rho})$, which means we must find the eigenvalues of

$$J\left(\frac{1}{\rho}, 0, \frac{1}{\rho}\right) = \begin{bmatrix} 1 & -\frac{b}{c} & -1 \\ b & -\frac{a}{c} & -b \\ \frac{2}{\eta} & 0 & -\frac{1}{\eta} \end{bmatrix},$$

which we can find by solving,

$$
\begin{aligned}
0 &= \begin{vmatrix} 1-\lambda & -\frac{b}{c} & -1 \\ b & -\frac{a}{c}-\lambda & -b \\ \frac{2}{\eta} & 0 & -\frac{1}{\eta}-\lambda \end{vmatrix} \\
&= (1-\lambda)\left(\frac{a}{c\eta}+\frac{a}{c}\lambda+\frac{1}{\eta}\lambda+\lambda^2\right)+\frac{b}{c}\left(-\frac{b}{\eta}-b\lambda_{\frac{2b}{\eta}}\right)-\frac{2a}{\eta c}-\frac{2}{\eta}\lambda \\
&= \frac{a}{c\eta}+\frac{a}{c}\lambda+\frac{1}{\eta}\lambda+\lambda^2+\frac{a}{c\eta}\lambda-\frac{a}{c}\lambda^2-\frac{1}{\eta}\lambda^2-\lambda^3-\frac{b^2}{c\eta}-\frac{b^2}{c}\lambda+\frac{2b^2}{c\eta}-\frac{2a}{\eta c}-\frac{2}{\eta}\lambda \\
0 &= \lambda^3+\left(\frac{a}{c}+\frac{1}{\eta}-1\right)\lambda^2+\left(\frac{b^2-a}{c}+\frac{1}{\eta}+\frac{a}{c\eta}\right)\lambda+\frac{a-b^2}{c\eta}.
\end{aligned}
$$

We define coefficients $A_2 = \frac{a}{c}+\frac{1}{\eta}-1$, $A_1 = \frac{b^2-a}{c}+\frac{1}{\eta}+\frac{a}{c\eta}$, and $A_0 = \frac{a-b^2}{c\eta}$ as the coefficients of the eigenvalues λ of second and first degree and the constants. According to the Routh-Hurwitz criterion, if all these coefficients are positive and if $A_1A_2 - A_0 > 0$, then the roots of this polynomial are negative [2]. Therefore, if these conditions is met, all eigenvalues are negative and the fixed point is stable. We can already say that $A_2 > 0$, since $a, c > 0$ and $\eta < 1$. We may also say that $A_0 > 0$ since via the Cauchy-Schwarz inequality, $\mathbf{x}_2\mathbf{x}_2^\top > (\mathbf{x}_1\mathbf{x}_2^\top)^2$, therefore $a > b^2$. Lastly we can show $A_1 > 0$ by using the facts that $c \in (0, \infty)$ and $\eta \ll 1$,

$$
\begin{aligned}
A_1 &= \frac{b^2-a}{c}+\frac{1}{\eta}+\frac{a}{c\eta} \\
cA_1 &= b^2-a+\frac{c+a}{\eta} \\
cA_1 &= \frac{\eta b^2-\eta a+c+a}{\eta} \\
cA_1 &= \frac{\eta b^2-(1-\eta)a+c}{\eta} \\
0 &< cA_1 \\
0 &< A_1.
\end{aligned}
$$

So the only condition to meet for this fixed point to be stable is

$$
\begin{aligned}
0 &< \left(\frac{b^2-a}{c}+\frac{1}{\eta}+\frac{a}{c\eta}\right)\left(\frac{a}{c}+\frac{1}{\eta}-1\right)-\frac{a-b^2}{c\eta} \\
0 &< \frac{ab^2-a^2}{c}+\frac{a}{c\eta}+\frac{a^2}{c^2\eta}+\frac{b^2-a^2}{c\eta}+\frac{1}{\eta^2}+\frac{a}{c\eta^2}+\frac{a-b^2}{c}-\frac{1}{\eta}-\frac{a}{c\eta}+\frac{b^2-a}{c\eta} \\
0 &< \frac{ab^2-a^2}{c}\eta^2+\frac{a}{c}\eta+\frac{a^2}{c^2}\eta+\frac{b^2-a^2}{c}\eta+1+\frac{a}{c}+\frac{a-b^2}{c}\eta^2-\eta-\frac{a}{c}\eta+\frac{b^2-a}{c}\eta \\
0 &< \left(\frac{(a-b^2)(1-a)}{c}\right)\eta^2+\left(\frac{a^2}{c^2}+\frac{b^2-a}{c}-1\right)\eta+1+\frac{a}{c}.
\end{aligned}
$$

Therefore, there is a stable fixed point at $(\frac{1}{\rho},0,\frac{1}{\rho})$ if the learning rates, inputs and their probabilities meet the conditions

$$
0 < \left(\frac{(a-b^2)(1-a)}{c}\right)\eta^2+\left(\frac{a^2}{c^2}+\frac{b^2-a}{c}-1\right)\eta+1+\frac{a}{c}, \tag{4.21}
$$

with $a=\mathbf{x}_2\mathbf{x}_2^\top, b=\mathbf{x}_1\mathbf{x}_2^\top, c=\frac{\rho}{1-\rho}$. We can follow the same process for the fixed point at $(y_1,y_2,\theta)=(0,\frac{1}{1-\rho},\frac{1}{1-\rho})$. The Jacobian evaluated at that point is

$$
J(0,\frac{1}{1-\rho},\frac{1}{1-\rho})=\begin{bmatrix}-c & b & -b\\ -cb & a & -a\\ 0 & \frac{2}{\eta} & -\frac{1}{\eta}\end{bmatrix},
$$

which means to find the eigenvalues and therefore the stability, we must evaluate

$$
\begin{aligned}
0 &= \begin{vmatrix}-c-\lambda & b & -b\\ -cb & a-\lambda & -a\\ 0 & \frac{2}{\eta} & -\frac{1}{\eta}-\lambda\end{vmatrix} \\
&= (-c-\lambda)\left(-\frac{a}{\eta}+\frac{1}{\eta}\lambda-a\lambda+\lambda^2+\frac{2a}{\eta}\right)-b\left(\frac{cb}{\eta}+cb\lambda\right)-b\left(\frac{-2cb}{\eta}\right) \\
&= \frac{-ac}{\eta}-\frac{c}{\eta}\lambda+ac\lambda-c\lambda^2-\frac{a}{\eta}\lambda-\frac{1}{\eta}\lambda^2+a\lambda^2-\lambda^3+\frac{cb^2}{\eta}-cb^2\lambda \\
0 &= \lambda^3+\left(c+\frac{1}{\eta}-a\right)\lambda^2+\left(\frac{a+c}{\eta}-ac+cb^2\right)\lambda+\frac{c(a-b^2)}{\eta}.
\end{aligned}
$$

We will again use the Routh-Hurwitz criterion to find the constraints for stability. Again, via the Cauchy-Shwartz Inequality it is true that $A_0 = \frac{c(a-b^2)}{\eta} > 0$, and if we set the learning rate such that $\frac{1}{\eta} > a$ then,

$$\begin{aligned} A_1 &= \frac{a+c}{\eta} - ac + cb^2 \\ &= \frac{(1-\eta a)c + (1+b^2\eta)c}{\eta} > 0, \end{aligned}$$

and, $A_2 = c + \frac{1}{\eta} - a > 0$. Via the Routh-Hurwitz criterion, we can say the fixed point $(0, \frac{1}{1-\rho}, \frac{1}{1-\rho})$ is stable if:

$$0 > \left(c + \frac{1}{\eta} - a\right)\left(\frac{a+c}{\eta} - ac + cb^2\right) - \frac{c(a-b^2)}{\eta}$$

$$0 > \frac{ac+c^2}{\eta} + \frac{a+c}{\eta^2} + \frac{-a^2-ac}{\eta} - ac^2 + \frac{-ac}{\eta} + a^2c + c^2b^2 + \frac{cb^2}{\eta} - acb^2 - \frac{ac-cb^2}{\eta}$$

$$0 > ac\eta + c^2\eta + a + c - a^2\eta - ac\eta - ac^2\eta^2 - ac\eta + a^2c\eta^2 + c^2b^2\eta^2 + \eta cb^2 - ab^2c\eta^2 - ac\eta + cb^2\eta$$

$$0 > c\big(a-b^2\big)(a-c)\eta^2 + \big(2c\big(b^2-a\big) + c^2 - a^2\big)\eta + a + c.$$

Thus the fixed point $(0, \frac{1}{1-\rho}, \frac{1}{1-\rho})$ is stable if

$$c\big(a-b^2\big)(a-c)\eta^2 + \big(2c\big(b^2-a\big) + c^2 - a^2\big)\eta + a + c > 0, \tag{4.22}$$

and the learning rate $\frac{1}{\eta} > a$. Thus we prove the stability of the BCM Learning Rule for some rigorous constraints on inputs. In the following section we demonstrate the convergence to these fixed points under these strict conditions, and that the BCM Rule demonstrates stability even if these constraints are not fully met.

4.2 Simulating BCM Learning

Due to its constraints for stability stated in Equations (4.21) and (4.22), to properly demonstrate the convergence of the BCM rule, a different, carefully designed strategy is performed than for Simple Hebbian and Oja Rules. Following this contrived strategy, the

BCM Rule will be more directly compared to the simulations conducted on the BCM and Oja Rules.

We start by following Udeigwe *et al.* [17] and Albesa-González *et al.* [16] in demonstrating the *selectivity* of the BCM rule. For the purpose of simulation, the BCM rule is shown in discrete time. Following the time-difference definition of the rule cited by Trappenberg [15], we take the derivatives in Equations (4.6) and (4.7) as discrete time differences and use the update rules:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta_w y(t)\mathbf{x}(t)(y(t) - \theta(t)), \tag{4.23}$$

$$\theta(t+1) = \eta_\theta(y(t)^2 - \theta(t)). \tag{4.24}$$

The BCM rule is unique because it gives the abstract neuron the ability to select one input pattern over another despite no difference in magnitude between the two. To illustrate this, we simulate the neuron by alternating inputs every other iteration between $\mathbf{x}_1 = (\cos(\phi), \sin(\phi))$ and $\mathbf{x}_2 = (\sin(\phi), \cos(\phi))$. This simulation satisfies every assumption made in our stability analysis in the previous section: we have two normalized input patterns and select an input with probability $\rho = 0.5$, the two inputs themselves have no correlation, by being opposite each other, and to satisfy the stability requirements from our analysis, we choose the parameters $\eta_w = 0.25, \eta_\theta = 0.425, \phi = 0.4$. Initial choices for the threshold and both weights are chosen randomly between 0 and 1. The resulting simulation is in Figure 4.1. In this simulation, the neuron demonstrates the maximization of selectivity in the BCM rule when the neuron's outputs converge to the fixed points, $y_1 = \theta = 2, y_2 = 0$, showing the neuron's selection of input $\mathbf{x}_1$. This selectivity is important in how a neuron can be "assigned" or naturally become aligned to a particular input, for example a neuron aligning to the signal from the taste of a Madeleine Cake.

After having demonstrated the desired convergence proved in our stability analysis, we simulate the BCM rule on a similar input scheme to the Simple Hebbian and Oja Rule for a more rigorous comparison between the models. In particular, we are interested if the BCM weighting scheme can both select between two incoming patterned signals and

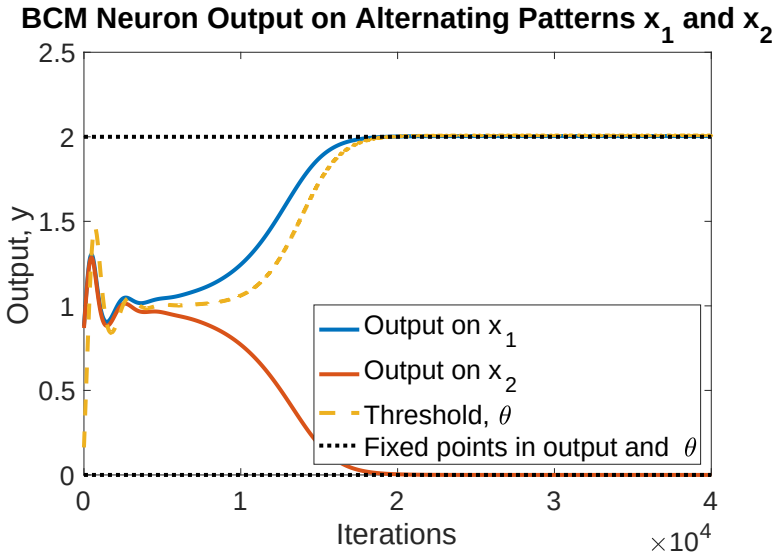

Figure 4.1: The abstract neuron's output over 40,000 iterations when trained on the BCM rule for two alternating inputs, $\mathbf{x}_1 = (\cos(0.4), \sin(0.4))$ and $\mathbf{x}_2 = (\sin(0.4), \cos(0.4))$. The BCM rule is run with $\eta = \frac{\eta_w}{\eta_\theta} = 0.001$ and $\eta_\theta = 0.425$. The neuron takes around 20,000 iterations to reach stable weights at which point it has reached a large selectivity between the outputs, and remains there for the duration of its stimulation.

differentiate between mixed patterns and noise. We alter the input from Figures 2.1 and 3.3 to accommodate our assumption of normalized inputs to the abstract neuron. We first normalize the "patterned" input in Equations (2.5) and (3.10), as,

$$\mathbf{x} = \begin{bmatrix} 0.943 \\ 0.019 \\ 0.019 \\ 0.019 \end{bmatrix}, \tag{4.25}$$

which holds the same proportion of input coming from each synapse as the patterns used in simulating the Simple Hebbian and the Oja Neurons. The random input the neuron receives is normalized in a similar way: each input is drawn from the uniform distribution between 0 and 1, then each input is divided by the sum of all inputs before being fed to the neuron. The simulations in Figure 4.2 show that the BCM neuron maintains the ability to recognize a repeated signal from random noise under most circumstances. In

simulation 4.2**a)**, the patterned input occurs 5% of the time, yet the neuron is still able to differentiate between the pattern in Equation (4.25) and noise. The fixed point for the patterned input of this rule should be $y_{\text{pattern}} = 20$, but the output is never that large, since weights are consistently being decremented in unpredictable ways due to the low-magnitude random output. This simulation is the best visualisation of how the threshold works in a neuron, learning to become the boundary between rewarded, patterned input and noise. In simulations 4.2**b)** and 4.2**c)**, a similar outcome for the outputs is seen for 50% and 95% patterned input, and less unpredictable input brings the patterned input closer to its fixed point. This result is consistent with the findings of our contrived alternating input scenario from earlier.

All simulations in Figure 4.2 provide a fairly accurate picture of how biological neurons learn to only respond to given patterns; however, it should be noted that this kind of learning did not always occur in simulation. In order for the BCM neuron to prefer the normalized pattern over the normalized random noise, the pattern needs to provoke higher output than the noise at the start of the simulation. Therefore, the threshold starts below the output for the pattern and above that for the noise. This occurs naturally most of the time in all but the 5% patterned input simulation, and can be accomplished, by artificially weighting $\mathbf{w}_1$ higher than the other weights (since it corresponds to the highest synaptic input in the patterned case), or by ensuring the neuron receives mostly patterns at the start of its learning. Either way, the neuron will always select either the pattern or the noise as its preferred input. The necessity of the neuron to receive one input more often at the beginning of its learning period, is reminiscent of how a new memory may form. Constant exposure to one form of input is typically how we learn to make associations.

To test if the BCM Neuron is able to keep its learned associations and therefore perform better than Oja's Rule, we perform a test of its recall ability. We use a simulation similar to the test of the Oja-trained neuron's recollection. First, we expose the neuron to the pattern from Equation (4.25) for a period of time long enough so that the output reaches its fixed point. Since the neuron is constantly exposed to that one pattern we can say $\rho = 1$,

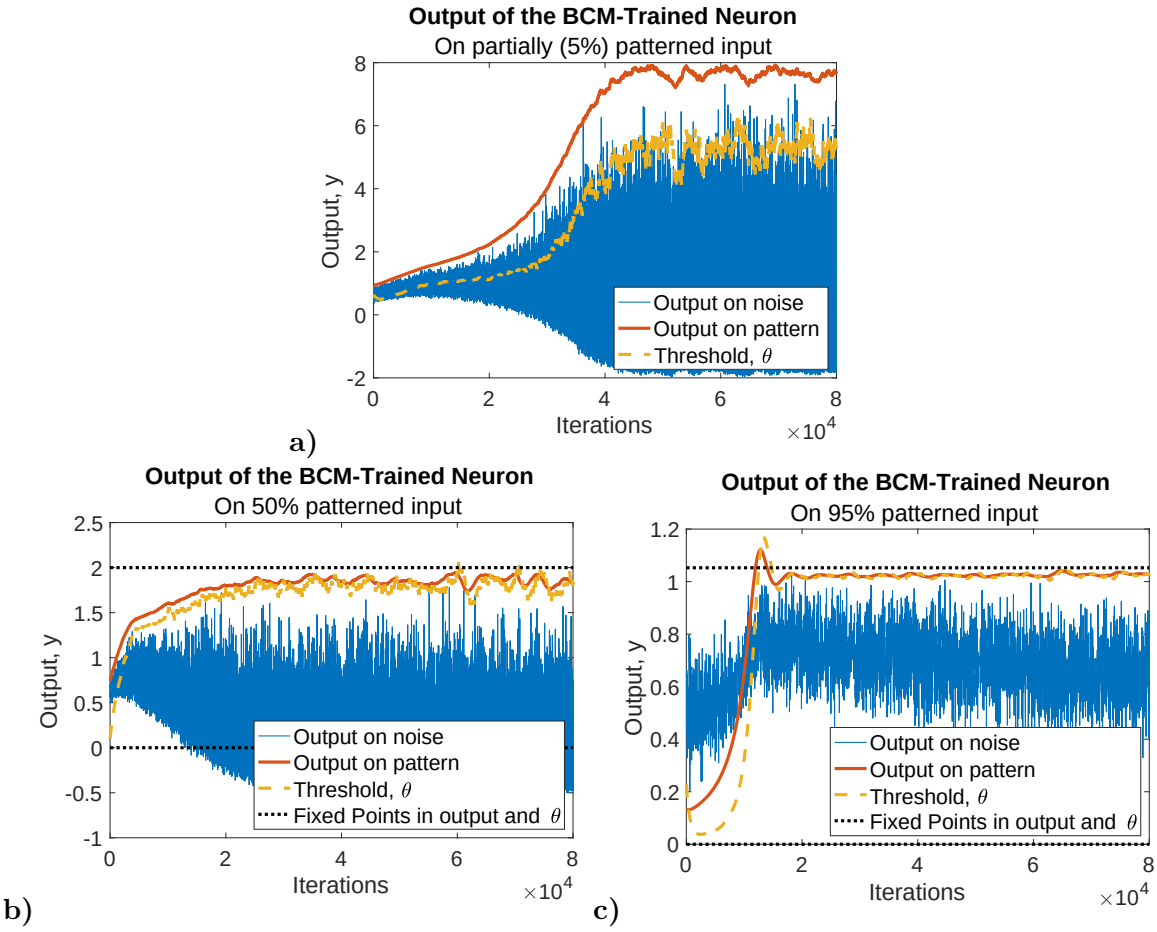

Figure 4.2: The BCM Neuron is trained on both patterned and random input for 10,000 iterations and a learning rate of $\eta = 10^{-3}$. The simulations demonstrate that not only will output on the patterned input remain stable, but, provided the output on the initial pattern is higher than the output on the noise, the neuron will consistently favor the patterned input with one synapse favored over random noise.

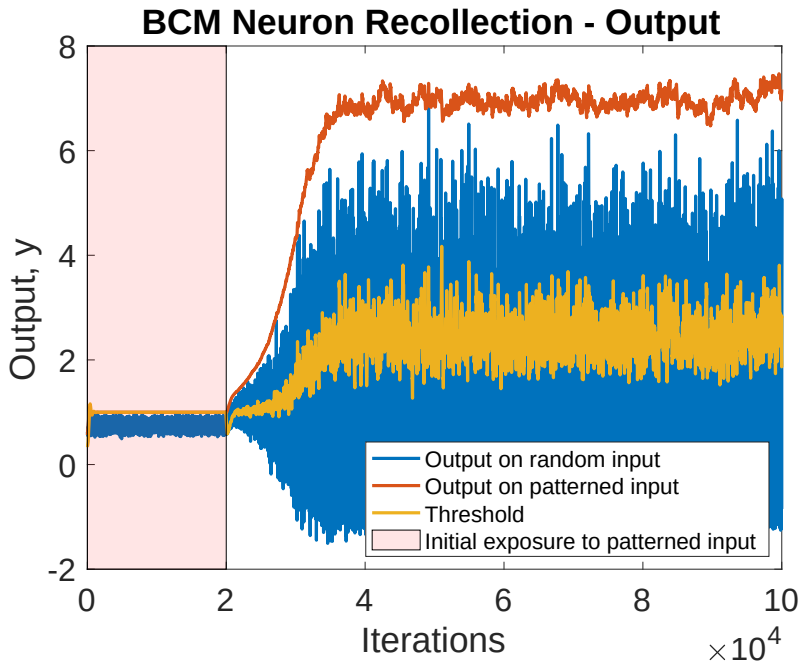

Figure 4.3: The BCM-Trained Neuron is simulated for 20,000 iterations on solely the patterned input from Equation (4.25). After selecting this pattern, the neuron is fed normalized random noise for the rest of the simulation. The neuron is trained with $\eta_w = 0.01$ and $\eta_\theta = 0.017$. Shown are the neuron's outputs for the pattern and for random noise over time. Note that the outputs are shown for both types of inputs for the entire duration of the simulation, even if our neuron is not updating its weights according to that input. Even when not receiving the patterned input, the neuron will still prefer that pattern over random noise for an indefinite period of time.

and therefore the stable fixed point reached is $y = 1, \theta = 1$. After the neuron has selected this pattern and the threshold and output are equal to each other, the neuron is then solely exposed to normalized random noise for the remainder of the simulation. Since the noise is not similar to the neuron's preferred pattern, the neuron's output decreases. The initial decrease in the neuron's output means that the threshold (as a moving time-average) will drop as well as the weights themselves. Now, the neuron only produces output higher than the threshold when the random input happens to weight input $\mathbf{x}_1$ strongly (since this was the input synapse it learned as the strongest indicator of our "pattern"). Therefore, that synaptic weight ($\mathbf{w}_1$) will be rewarded the strongest, and correspondingly the neuron will react highly to input resembling the initial pattern it learned. The simulation is shown in Figure 4.3. We can also see how this would apply in the inverse to a different pattern indicated by one particular input being small, or for particular inputs of a given ratio. For example, random noise high in a dimension of the pattern with a weak input would still be below the threshold and keep the corresponding synaptic weight low.

In Figure 4.3, we see that this neuron has the ability to remember. Unlike the Oja's Rule simulation (Figure 3.4), the weight of the first synapse is not steadily degraded over time despite the neuron's constant exposure to noise. We have also seen that the BCM neuron exhibits stability and pattern recognition, forming all our desired qualities of both a biological and computational neuron: the ability to select between patterns in a stable fashion, and to remember which pattern it selected. This justifies the BCM Rule in both representing how neurons learn, and using the incredible capability of a simple weighted sum, the abstract neuron, for memory and pattern recognition.

Chapter 5

Conclusion

Fueled by a fascination with associative memory, this thesis explored the biological accuracy and computational utility of an abstract neuron in pattern recognition and memory through stability analysis and simulation. We pursued three models of synaptic weight plasticity, all derived from the Hebbian theory of unsupervised learning in neurons that "what fires together, wires together". We demonstrated that the simplest form of this rule provides no computational use as it is unstable. Then we introduced normalization to the Simple Hebbian Rule to find Oja's Rule: a learning rule in which synaptic weights converge to a stable point. Simulation showed that Oja's Rule resulted in pattern recognition, even when interspersed by random noise. The ability to differentiate noise from patterns is an impressive ability for such a simple algorithm, and a powerful tool in many machine learning applications. When exposed to only noise for an extended period of time, however, the Oja-trained neuron lost its association with this pattern. For further biological realism, we followed BCM's theory to devise a model to maximize selectivity in the neuron. When analyzed, the resulting model demonstrated stability of output on two discrete inputs for some constraints. When simulated, the BCM model showed stability and selectivity between patterns, even when inputs did not follow those requirements. Most importantly, this rule demonstrated a better ability to recall than Oja's Rule. In total, these models demonstrated the utility of a computational neuron as a way of both

understanding and using the capabilities of the human brain.

Today, from a computational perspective, the gold standard in recreating our brains is the modern Artificial Neural Network (ANN). The ANN is composed by abstract neurons like those analyzed here, with the addition of a bias to the weighted sum ($y = \mathbf{w}\mathbf{x} + b$), and an activation function in between layers of neurons (representing the choice to "fire" of a real neuron). These networks have shown their ability to create images from text prompts, and answer questions in a way that approaches the power of the human brain. But their thirst for data and computation to learn is expensive, and in some cases very inefficient [12]. We propose that turning back to their origins, and incorporating the biology reflected in the Hebbian-inspired learning rules could help with both efficiency and accuracy in these models.

As a whole, the Hebbian-based rules analyzed in this paper can remain stable in weight while differentiating patterns from noise and use associative memory to recollect those patterns. Modern Artificial Intelligence has tried to imitate both of these abilities in complex Neural Networks tangentially-related to real biology. For example, highly complex Diffusion Models are trained to differentiate added noise from realistic images in order to generate those images from noise. This is a skill for which the BCM and Oja rules show talent. Furthermore, the denoising capability of the BCM Rule has been shown in earlier papers [11] [6]. The Diffusion Model as well as several Large Language Models also use the notion of associativity to "encode" meanings of words and images into what they call "latent space". These models use a highly nonlinear and complicated function called Long Short-Term Memory, which (put reductively) encode the meanings of words by counting how often words appear together [18]. The Hebbian Inspired rules perform this task in a more biologically-realistic way. Oja's Rule demonstrates associativity, as shown above, with stable weights corresponding to an input pattern, and the BCM Rule shows that it can hold onto this associativity for a long time even when extensively exposed to non-relevant input. Perhaps connected BCM-trained neurons each tailored to specific words or phrases, could form a simpler Hebbian associativity like those that may actually be in

our brains. Instability is also a problem that modern Neural Networks suffer from, with the common problem of some synaptic weights exploding or vanishing to zero. Hebbian-inspired learning rules open the door to a wide array of applications in today's Machine Learning field. Most importantly, the usefulness of the models shown here tells us that we can always benefit from connecting AI back to its human roots, and remembering our greatest inspiration: ourselves.

Appendix A

Appendix

A.1 Taking learning rules from discrete to continuous time

Biologically, time is not as simple as presented in discrete versions of the learning rules in Equations (2.1) or (3.6). Therefore to keep a connection between the learning of the abstract neuron and the biological cell it is based on, a continuous form of the learning rule should be found. We will first demonstrate this process for the Hebbian Rule.

First, we rearrange the discrete Hebbian Rule in Equation (2.1) as

$$\mathbf{w}(t+1) - \mathbf{w}(t) = \eta y(t)\mathbf{x}(t). \tag{A.1}$$

Call the change in weights from one time step to the next, $\Delta\mathbf{w} = \mathbf{w}(t+1) - \mathbf{w}(t)$. If we take the limit as $\Delta t \to 0$ of $\frac{\Delta w}{\Delta t}$, we will have an equation for the instantaneous change in weights over time. Taking this limit, we get the expression

$$\lim_{\Delta t \to 0} \frac{\Delta \mathbf{w}}{\Delta t} = \lim_{\Delta t \to 0} \frac{\eta y \mathbf{x}}{\Delta t}. \tag{A.2}$$

Now, if we choose η such that it decays to 0 at the same rate as Δt, we can say

$$\frac{d\mathbf{w}}{dt} = y\mathbf{x}, \tag{A.3}$$

which is the continuous version of The Simple Hebbian Rule. This is true if, for example, we choose a decay rate for η proportional to $\frac{1}{t}$, then $\frac{\eta}{\Delta t} = 1$.

The same steps can be performed on the discrete version of Oja's Rule in Equation (3.6), except the limit taken in that case is:

$$\lim_{\Delta t \to 0} \frac{w_i(t+1) - w_i(t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{\eta(y(t)x_i(t) - y(t)^2 w_i(t))}{\Delta t}, \tag{A.4}$$

and if we assign η in that same way such that it decays at the rate of Δt,

$$\frac{\partial \mathbf{w}}{\partial t} = y\mathbf{x} - y^2\mathbf{w}. \tag{A.5}$$

Equation (3.7) is the corresponding continuous form of that rule. Note, however, while a decaying learning rate is commonly used in machine learning applications, it makes little sense in a biological context, as humans learn over their entire lifetimes. However it can be shown that even when η does not necessarily decay to 0, but remains small, the continuous learning in (3.7) is a good approximation of the discrete learning in System (3.6) [13].

A.2 Code

Code for simulating all learning rules under different input schemes was written in MATLAB. The corresponding live scripts are available in a public repository at:
https://github.com/heaston2000/AbstractNeuronMemory.

A.3 Interactive Visualization

You can experiment with the Oja and BCM Rules (including feeding them visual input) in a visualization found at: https://heaston2000.github.io/abstractneuronviz.

Bibliography

[1] Involuntary memory, March 2023. Page Version ID: 1147034031.

[2] Routh–Hurwitz stability criterion, April 2023. Page Version ID: 1147902477.

[3] *In Search of Lost Time*, April 2023. Page Version ID: 1148752309.

[4] E. L. Bienenstock, L. N. Cooper, and P. W. Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 2:32–48, January 1982.

[5] Jason Brownlee. A Gentle Introduction to Generative Adversarial Networks (GANs) - MachineLearningMastery.com, June 2019.

[6] Leon N Cooper and Mark F. Bear. The BCM theory of synapse modification at 30: interaction of theory with experiment. *Nature Reviews Neuroscience*, 13(11):798–810, November 2012.

[7] Peter Dayan and L.F. Abbott. *Theoretical Neuroscience, Computational and Mathematical Modeling of Neural Systems*. The MIT Press, Cambridge, Massachusetts, 2005.

[8] Donald Hebb. *The Organization of Behaviour*. John Wiley & SONS, Inc. , Chapman & HALL, Limited, New York, NY; London, UK, 1st edition, 1949.

[9] John Hertz. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Redwood City, California, 1991.

[10] Nathan Intrator and Leon N. Cooper. Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions. *Neural Networks*, 5(1):3–17, 1992.

[11] C C Law and L N Cooper. Formation of receptive fields in realistic visual environments according to the Bienenstock, Cooper, and Munro (BCM) theory. *Proceedings of the National Academy of Sciences of the United States of America*, 91(16):7797–7801, August 1994.

[12] Cade Metz. What's the Future for A.I.? *The New York Times*, March 2023.

[13] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

[14] Erkki Oja and Juha Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106(1):69–84, 1985.

[15] Trappenberg T. *Fundamentals of Computational Neuroscience*. OUP, 2002.

[16] Albesa-González *et al*. Weight dependence in bcm leads to adjustable synaptic competition. *Journal of Computational Neuroscience*, 50:431–444, June 2022.

[17] Lawrence C. Udeigwe, Paul W. Munro, and G. Bard Ermentrout. Emergent Dynamical Properties of the BCM Learning Rule. *The Journal of Mathematical Neuroscience*, 7(1):2, February 2017.

[18] Teal Wittier. Deep learning, January 2023.