Inferring Synaptic Weight Dynamics with a Circuit Simulator

Undergraduate Honours Thesis
CPSC 449

Ryan Fayyazi

Department of Computer Science
The University of British Columbia
April 2020

Acknowledgements

I would like to thank Andrew Warrington for his mentorship and invaluable contributions throughout the development of this project. I would also like to thank Frank Wood and Catherine Rankin for their encouragement and consultation.

Abstract

The modification of synaptic weights is thought to be a fundamental mechanism underlying learning. Therefore, a primary objective of neuroscience is to discover the algorithms which govern synaptic updates to produce learning in different behaviours. The mathematical formalization of synaptic update rules has been of particular interest to computational neuroscientists, and the the current canon of formalized rules has been successful in modeling various neural phenomena. However, hand-deriving them from in vitro experimental observations is slow, and proposed rules cannot be tested directly in freely learning organisms. This thesis contributes the synaptic update rule finder (SURF), a framework for automatically inferring the update rule and naive synaptic weights controlling the circuit underlying a given plastic behaviour, from a time series of neural activity observations from that circuit over the course of learning in a real organism. The test case chosen for this thesis is habituation of the tap-withdrawal response in *Caenorhabditis elegans*.

Inferring Synaptic Weight Dynamics with a Circuit Simulator

1 Introduction

One of the primary objectives of neuroscience is to discover the algorithms with which nervous systems process information in order to produce behaviour. In both biological and artificial neural agents, behaviour is the emergent result of a response cascade through a network of interconnected, lower-level computational units - such as nodes in a policy network or neurons in a brain - triggered by some input. In biological neural networks, the connection between a pair neurons is called a synapse. Synapses are the sites of electrochemical communication between neurons, and can be categorized into two broad types: chemical synapses and electrical synapses. Electrical synapses are composed of channels which directly link the intracellular spaces of a pair of neurons, allowing bidirectional passive diffusion of electrical currents between them [1]. A Chemical synapse is a unidirectional junction at which the axon terminal of an active presynaptic neuron releases neurotransmitter molecules onto the dendrite of a postsynaptic neuron. The neurotransmitters bind specific receptors on the dendrite membrane, triggering various molecular processes in the postsynaptic neuron. Principle among these processes is the opening of ion channels, which in turn causes depolarization or hyperpolarization of the postsynaptic neuron's membrane, propagating the signal forward [2]. When two neurons are connected via a chemical synapse, the magnitude of the influence exerted by the upstream neuron on the activity of the downstream neuron is determined by a number of factors, such as the number of ion channels and receptors on the postsynaptic dendrite, and the amount of neurotransmitter released from the upstream axon terminal. This magnitude is often abstracted to a scalar value, called the synaptic weight. Similarly, the weight of an electrical synapse is determined by quantities such as the number and conductance of the channels that comprise it.

In 1890, William James proposed the notion that experience-driven changes in a behaviour are mediated by physical changes in the neural pathways responsible for it [3]. Shortly thereafter, Eugenio Tanzi and his student Ernesto Lugaro introduced the idea that repetitive activation of a neural pathway might increase the efficiency of transmission (later termed weight) across articulations between neurons (later called synapses) within the pathway, and claimed that this might underlie the phenomenon described by James [3]. This line of thinking culminated in Donald Hebb's landmark book *The Organization of Behaviour* [4], which brought synaptic plasticity, or modification of synaptic weight, to the forefront of learning research. In this book, Hebb presents three main postulates [4]:

1. When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.
2. Groups of neurons that tend to fire together form a cell assembly whose activity can persist after the triggering event and serves to represent it.

3. Behaviour patterns are built as cell assemblies become interconnected through experience, and they are executed as activity from one assembly propagates through the others in a particular phase sequence.

Today, most learning theories incorporate the idea that changes in synaptic weight are a fundamental mechanism by which a network's behaviour is modified [5, 6]. The first experimental confirmation of Hebb's postulates came in the form of long-term potentiation (LTP) at synapses in the rabbit hippocampus [7]. LTP describes the facilitation of synaptic transmission induced by coincident presynaptic and postsynaptic depolarization. Long-term depression (LTD), which requires presynaptic activity without a coincident postsynaptic response, has been observed as well [8, 9]. Potentiation and depression of inhibitory synapses has also been demonstrated, but relies on slightly different mechanisms [10]. These results demonstrate that experience-driven weight modulation is an important capability of synapses throughout the brain. It is now believed that LTP and LTD play critical roles in producing the neural assemblies that Hebb proposed, and that these processes play a central role in learning [11, 6].

Starting with Hebb, significant effort has been spent on mathematically formalizing synaptic update rules [12], typically as differential equations which take as arguments the presynaptic and postsynaptic membrane potentials, and the current synaptic weight. The current canon of formalized rules has been successful at modeling various neural phenomena, such as hippocampal associative plasticity [13, 9], visual receptive field formation [13, 14], and spike-timing dependent plasticity [15]. However, hand-deriving them one at a time is a slow process, and restricts the complexity that the rules can feasibly achieve. This process is further complicated by the fact that weight updates are typically small [16] and involve a specific, limited subset of synapse, and cannot yet be observed reliably in vivo. Therefore, a different approach is needed to fully characterize the algorithms which dictate how adaptive changes in neural behaviour are accomplished [17, 18]

This thesis contributes the synaptic update rule finder (SURF), a framework for automatically inferring the update rule and naive synaptic weights controlling the circuit underlying a given plastic behaviour, from a time series of observations of the activity of the circuit's neurons over the course of learning in a real organism. The test case chosen for this thesis is habituation of the tap-withdrawal response in *Caenorhabditis elegans*, a species of roundworm.

2 Background

2.1 Formalized Synaptic Update Rules

This section provides an overview of the current canon of synaptic update rules. Hebb's postulates are usually formalized as the following synaptic update rule [19]:

$$\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u}, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^n$ is the weight vector for n incoming synapses, $v \in \mathbb{R}$ is the postsynaptic activity (e.g. firing rate), $\mathbf{u} \in \mathbb{R}^n$ is the activity vector for the presynaptic neurons, and $\tau_w \in \mathbb{R}$ is a rate constant. Hebb's rule results in unbounded synaptic weight growth and cannot explain synaptic depression, but has led to a number of other equations with more biologically valid properties. The following is an incomplete list of alternative synaptic update rules which adhere to Hebbian principles:

Presynaptic Threshold Rule [19]:

$$\tau_w \frac{d\mathbf{w}}{dt} = v(\mathbf{u} - \boldsymbol{\theta}_\mathbf{u}), \tag{2}$$

where $\boldsymbol{\theta}_\mathbf{u} \in \mathbb{R}^n$ is a vector of presynaptic thresholds.

Postsynaptic Threshold Rule [19]:

$$\tau_w \frac{d\mathbf{w}}{dt} = (v - \theta_v)\mathbf{u}, \tag{3}$$

where $\theta_v \in \mathbb{R}$ is a postsynaptic threshold.

Covariance Rule [19]:

$$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{C} \cdot \mathbf{w}, \tag{4}$$

with input covariance matrix $\mathbf{C} = \overline{\mathbf{u} \cdot \mathbf{u}^T} - \overline{\mathbf{u}} \cdot \overline{\mathbf{u}}^T$, where the bar denotes the mean of the matrix (i.e. average $\mathbf{u}$ over the course of training).

Oja's Rule [20]:

$$\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u} - \alpha v^2 \mathbf{w}, \tag{5}$$

with $\alpha \in \mathbb{R}$.

Bienenstock-Cooper-Munro Rule [13]:

$$\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u}(v - \theta_v) \qquad \tau_{\theta_v} \frac{d\theta_v}{dt} = v^2 - \theta_v, \tag{6}$$

where sliding threshold $\theta_v \in \mathbb{R}$ induces competition between synapses.

Spike-Timing Based Rule [19]:

$$\tau_w \frac{d\mathbf{w}}{dt} = \int_0^\infty (H(\tau)v(t)\mathbf{u}(t-\tau) + H(-\tau)v(t-\tau)\mathbf{u}(t))\mathrm{d}\tau, \tag{7}$$

where $H(\tau)$ determines the rate of synaptic modification that occurs due to postsynaptic activity separated in time from presynaptic activity by an interval τ.

It is interesting to note that artificial neural networks also modify their behaviour through the modification of weights, a process often controlled by learning rules which strongly resemble the biological ones listed above. For example, gradient descent with error back-propagation updates weights according to the differential equation

$$\frac{dW_{ij}^{(l)}}{dt} = -\eta^t \nabla_{W_{ij}^{(l)}} C((\mathbf{W})^t) = -\eta^t \frac{\partial (a_i^{(l)})^t}{\partial W_{ij}^{(l)}} \sum_{k=1}^{D^{(l+1)}} \frac{\partial}{\partial a_k^{(l+1)}} C((\mathbf{W})^t) \frac{\partial (a_k^{(l+1)})^t}{\partial a_j^{(l+1)}}. \tag{8}$$

Here, η is the step size; $W_{ji}^{(l)}$ is the weight of the edge between node j in layer $l-1$ and node i in layer l; (a_i^l) is the activation of node i in layer l; and $C(\cdot)$ is a cost function (e.g. mean squared error). Indexing with t indicates the values of the variable being indexed at optimization step t.

2.2 C. elegans & Habituation

The test case for SURF is the habituation of the tap-withdrawal response (TWR) in *Caenorhabditis elegans*, a 1mm long roundworm. *C. elegans* has a fully-mapped genome [21, 22], cell lineage [23] and connectome, which is comprised of 302 neurons and is invariant across individuals [24]. These features, along with its short life cycle and ease of propagation [25], make *C. elegans* an ideal model organism for neuroscience research. *C. elegans* was chosen to be the test subject for SURF for three reasons. Firstly, its small, invariant and fully mapped connectome allows simulation of the worm's nervous system. Secondly, *C. elegans* can learn, with the circuitry underlying some types of learning consisting of only a few neurons. Thirdly, the neural correlates of some plastic behaviours have been characterized.

Remarkably, *C. elegans* demonstrates both non-associative [26] and associative [27] learning [28]. The first evidence for learning in this organism was the habituation of the TWR, an avoidance behaviour wherein the worm rapidly moves backwards, changes direction, and

moves forward again in response to a global mechanosensory stimulus. In experiments, this stimulus is typically a tap to the side of the petri dish that the worm is swimming in. Habituation, a type of non-associative learning, is often considered the simplest form of learning. Non-associative learning describes learning phenomena wherein alterations in elicited behavior occur as a result of repeated presentations of a stimulus [6]. Habituation denotes the case when repeated stimulation results in a decrease in responsiveness, with respect to one or more parameters of a response. In the case of the TWR, repeated global mechanosensory stimulation causes a decrease in reversal speed, probability and/or magnitude, depending on the specific stimulus and presentation schedule [29]. Habituation is distinguished from other forms of motor response inhibition (e.g. adaptation, fatigue) by dishabituation - the rapid restoration of the inhibited response by a change in the eliciting stimulus. Once stimulus presentation is ceased, the response recovers over time, a process termed spontaneous recovery. If stimulus presentation is resumed after spontaneous recover, the habituation that ensues is more rapid than the first time. The habituation rate can also be increased by decreasing the strength of the stimulus or increasing the frequency of presentation [29, 6].

Although a large amount work has been done on habituation in *C. elegans*, the molecular mechanisms responsible remain poorly understood. This is largely due to the fact that different molecular processes seem to underlie habituation in different stimulus contexts [29, 30]. However, there is evidence that synaptic weight modification underlies habituation. For example, long-term habituation, induced by blocks of repeated stimulation over relatively long periods of time, is caused by down-regulation of glutamate receptors on the interneurons of the tap-withdrawal circuit [29]. When glutamate receptors are activated, they cause depolarization in the postsynaptic neuron, which may then propagate the signal onward. Decreasing glutamate receptor expression at a synapse therefore represents a decrease in synaptic weight.

A nine-neuron tap withdrawal circuit mediates the TWR, and is composed of four mechanosensory neurons (ALM, AVM, PLM, PVD) and five interneurons (AVA, AVB, AVD, PVC, DVA) [31, 28]. The connectivity of this circuit is known, as are the relative weights of the synapses in naive individuals [24, 31, 32]. Furthermore, the mechanosensory-interneurons synapses are thought to be the site of the plasticity underlying TWR habituation [28], rather than downstream synapses which are involved in reverse locomotion in general. This is based on the fact that TWR habituation does not have any impact on spontaneous reversal behaviour or reversals evoked by aversive thermal stimulation [33].

Finally, the neural correlates of TWR-relevant locomotive parameters are known. Let

$$\delta = \int_{t_{start}}^{t_{end}} V_{AVA} - V_{AVB} \, \mathrm{d}t, \tag{9}$$

where t_{start} and t_{end} are the start and end times of a given stimulation, and V_{AVA} and V_{AVB} are the membrane potential (mV) traces of interneurons AVA and AVB in the stimulated worm. Then δ is directly proportional to the worm's reversal distance in response to the stimulation [32, 34].

3 Framework

The latent dynamics of a neural circuit simulator during learning can be modeled approximately as a discrete-time Markov process $\{X_t\}_{0 \leq t \leq T}$, with initial state $X_0 \sim \mu(x_0)$ and transition density

$$(X_t | X_{t-1} = x_{t-1}) \sim f(x_t | x_{t-1}). \tag{10}$$

Observations of neural activity during learning $\{Y_t\}_{0 \leq t \leq T}$ are independent given $\{X_t\}_{0 \leq t \leq T}$, and are distributed according to the emission density

$$(Y_t | X_t = x_t) \sim g(y_t | x_t). \tag{11}$$

This formulation defines a hidden Markov model (HMM) of the circuit's activity [35, 36]. The contents of X_t depend on the specifics of the simulator, which, equipped with a synaptic update rule, also implicitly defines the transition density. For SURF to be applicable to a simulator, X_t must include a $N_n \times N_n$ synaptic weight matrix state $\mathbf{w}_t$ and a membrane potential state $\mathbf{v}_t$ of size N_n, where $N_n \in \mathbb{Z}^+$ is the number of neurons in the simulated circuit. X_t must also contain a synaptic update rule R parameterized by a constant vector $\boldsymbol{\theta} \in \mathbb{R}^{N_R}$, where $N_R \in \mathbb{Z}^+$ is the number of constant parameters in R. Finally, with an eye towards the experiment in section 3, assume that X_t includes an intracellular calcium concentration state $\mathbf{c}_t$ of size N_n. The simulator's biophysical parameters $\boldsymbol{\phi}$ could be included in X_t, but since they are constant throughout this thesis, they are left out. For notational simplicity, let $\mathbf{y}$ denote $\{Y_t\}_{0 \leq t \leq T}$, let $\mathbf{x}$ denote $\{X_t\}_{0 \leq t \leq T}$, and let $\mathbf{z}_t = \{\mathbf{v}_t, \mathbf{c}_t\}$ denote the electrochemical latent states. The dependencies between these states are summarized by the graphical model in Figure 1. According to this model, the distribution over the circuit's synaptic dynamics can be formalized as the following posterior (derivation in Section 6.1):

$$p(\mathbf{w}, R | \mathbf{y}) \ \propto \int_{\mathbf{z}} p(\mathbf{y} | \mathbf{z}, \mathbf{w}, R) p(\mathbf{z}, \mathbf{w} | R) p(R) \mathrm{d}\mathbf{z}. \tag{12}$$

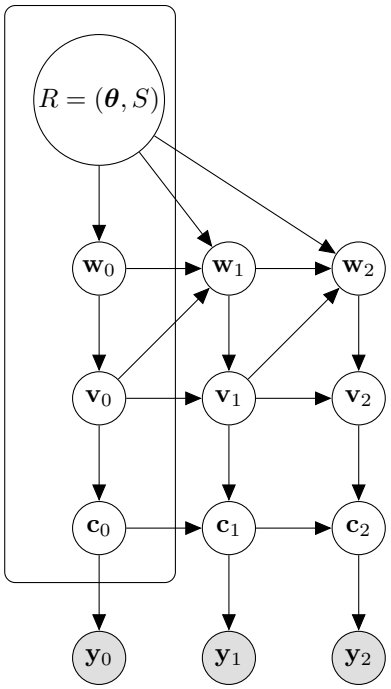

Figure 1: The hidden Markov model for a neural circuit simulator. The box indicates the hidden states which comprise $\mathbf{x}_0$

As depicted in Figure 1, most measures of neuron activity are independent of $\mathbf{w}$ and R given $\mathbf{z}$. For example, calcium fluorescence and extracellular electrical activity are determined by intracellular calcium concentration and membrane potential, respectively. Therefore, $p(\mathbf{y}|\mathbf{z}, \mathbf{w}, R) = p(\mathbf{y}|\mathbf{z})$. Moreover, by the conditional independence of the observations,

$$p(\mathbf{y}|\mathbf{z}) \ = \prod_{t=0}^{T} p(\mathbf{y}_t|\mathbf{z}_t), \tag{13}$$

where $p(\mathbf{y}_t|\mathbf{z}_t)$ is determined by the chosen activity measurement method. This is typically an easily-sampled density, such as a Gaussian or Poisson. The distribution $p(\mathbf{z}, \mathbf{w}|R)$ can be written as

$$p(\mathbf{z}, \mathbf{w}|R) = p(\mathbf{z}_0, \mathbf{w}_0|R) \prod_{t=1}^{T} p(\mathbf{z}_t, \mathbf{w}_t|\mathbf{z}_{t-1}, \mathbf{w}_{t-1}, R), \tag{14}$$

where $p(\mathbf{z}_t, \mathbf{w}_t|\mathbf{z}_{t-1}, \mathbf{w}_{t-1}, R)$ is defined implicitly by the simulator and update rule, both of which may be either stochastic or deterministic. In order to sample from this distribution, the simulator (equipped with R) is simply iterated forward one step. The distribution $p(\mathbf{z}_0, \mathbf{w}_0|R)$ must be specified by the experimenter, and equals $p(\mathbf{z}_0, \mathbf{w}_0)$ if the initialization

of $\mathbf{z}_0, \mathbf{w}_0$ is independent of R. Finally, the prior $p(R)$ is defined implicitly by the method used to sample the discrete structure and continuous parameters of R. For example, SURF employs a recursive random rule generator with maximum recursion depth $d > 0$, where the probability of generating a rule with computational graph depth x is inversely proportional to x if $3 \leq x \leq d+2$ and zero otherwise. The generator is described in depth in Section 4.4, and the code is presented in Section 6.6.

Thus, the integral in (3) can be reformulated as

$$\int_{\mathbf{z}} p(\mathbf{y}_0|\mathbf{z}_0)p(\mathbf{z}_0, \mathbf{w}_0|R)p(R) \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{z}_t)p(\mathbf{z}_t, \mathbf{w}_t|\mathbf{z}_{t-1}, \mathbf{w}_{t-1}, R)\mathrm{d}\mathbf{z}. \tag{15}$$

Since each distribution in this product can be sampled from, it is feasible to perform inference to estimate the target posterior $p(\mathbf{w}, R|\mathbf{y})$ using techniques such as Metropolis-Hastings or sequential Monte Carlo estimation. Unfortunately, these methods are computationally expensive, especially in complex simulators. However, if the simulator and update rule are deterministic, $p(\mathbf{z}_t, \mathbf{w}_t|\mathbf{z}_{t-1}, \mathbf{w}_{t-1}, R)$ becomes a Dirac delta function, resulting in a low-entropy posterior $p(\mathbf{w}, R|\mathbf{y})$. In this case, maximum a posteriori (MAP) estimation is a reasonable, tractable approximation of the true posterior inference problem. Under the deterministic assumptions, the trace $\mathbf{x}_{1:T}$ is pre-determined given initial state $\mathbf{x}_0$, so $\mathbf{w}_0$ can be estimated instead of $\mathbf{w}$. Therefore, in this context, SURF attempts MAP estimation of $\mathbf{w}_0$ and R given $\mathbf{y}$, resulting in point estimates

$$\mathbf{w}_0^*, R^* = \underset{\mathbf{w}_0, R}{\operatorname{argmax}}\ p(\mathbf{w}_0, R|\mathbf{y}) = \underset{\mathbf{w}_0, R}{\operatorname{argmin}} -\log(p(\mathbf{w}_0, R|\mathbf{y})). \tag{16}$$

(16) defines an optimization problem, with the negative log posterior taken as the objective function to be minimized. Through the derivation outlined in Section 6.2, the objective function can be reformulated as

$$-\log(p(\mathbf{w}_0, R|\mathbf{y})) \propto -\log(\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{w}_0, R)}[p(\mathbf{y}|\mathbf{z})]) - \log(p(\mathbf{w}_0|R)) - \log(p(R)). \tag{17}$$

To optimize the right hand side of (17), each of the three additive terms must be evaluable for a given $\mathbf{y}$, $\mathbf{w}_0$ and R. The expectation is difficult to compute, but can be approximated using Monte Carlo (MC) integration if $p(\mathbf{z}|\mathbf{w}_0, R)$ can be sampled from. This distribution can be sampled by randomly initializing the states in $\mathbf{x}_0$, iterating the simulator equipped with the given R and $\mathbf{w}_0$ forward for $T+1$ observations, and extracting the trace $\mathbf{z}$. Consider N samples $\{(\mathbf{z}^{(i)}\}_{1\leq i\leq N} \sim p(\mathbf{z}|\mathbf{w}_0, R)$. Then by the law of large numbers,

$$-\log(\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{w}_0, R)}[p(\mathbf{y}|\mathbf{z})]) \ = \lim_{N\to\infty}\left[-\log\left(\frac{1}{N}\sum_{i=1}^{N}\prod_{t=0}^{T} p(\mathbf{y}_t|\mathbf{z}_t^{(i)})\right)\right]. \tag{18}$$

The right hand side of (18) can be approximated with MC integration. The values of

$p(\mathbf{w}_0|R)$ and $p(R)$ depend on the distributions used to initialize $\mathbf{w}_0$ and R respectively, which can be chosen for ease of evaluation.

For each neuron, R maps some combination of the neuron's activity, its upstream synaptic weights, and the activity of its input neurons, to a weight update vector to be added to the upstream synaptic weights. This mapping has a particular compositional structure determined by the relations that comprise it, as well as parameters $\boldsymbol{\theta}$. Therefore, optimizing R entails a joint search over the discrete structure space and continuous parameter space. This kind of joint optimization is an inherently difficult problem [37], and is made more so by the fact that the structure space is theoretically infinite. As a first tractable step, SURF finds the best rule out of a finite set $\{R_1, \ldots, R_k\}$ of k candidate rules, with associated parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$ and compositional structures $S_1, \ldots, S_k$. Note that $\boldsymbol{\theta}_k$ denotes the parameter vector of R_k, rather than the kth entry of $\boldsymbol{\theta}$. To accomplish this, SURF solves the following continuous optimization problem for each R_k:

$$\mathbf{w}_0^*, \boldsymbol{\theta}_k^* = \underset{\mathbf{w}_0, \boldsymbol{\theta}_k}{\operatorname{argmin}} - \log(p(\mathbf{w}_0, \boldsymbol{\theta}_k|\mathbf{y})) \tag{19}$$

$$= \underset{\mathbf{w}_0, \boldsymbol{\theta}_k}{\operatorname{argmin}} - \log(\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{w}_0, \boldsymbol{\theta}_k, S_k)}[p(\mathbf{y}|\mathbf{z})]) - \log(p(\mathbf{w}_0|\boldsymbol{\theta}_k, S_k) - \log(p(\boldsymbol{\theta}_k|S_k)) \tag{20}$$

and returns the R_k with the lowest $-\log(p(\mathbf{w}_0^*, \boldsymbol{\theta}_k^*|\mathbf{y}))$ (i.e. R^*), along with its $\mathbf{w}_0^*$ and $\boldsymbol{\theta}_k^*$. The conversion of the objective in (17) to the one in (20) is shown in Section 6.3. $p(\boldsymbol{\theta}_k|S_k)$ is the density from which the parameters of R_k are initialized before optimization, and can be chosen for easy evaluation (e.g. Gaussian). The continuous optimization can be performed with standard techniques such as gradient descent.

The full SURF procedure, in the context of a finite set of deterministic candidate rules and a deterministic simulator, is summarized by Algorithm 1 in Section 6.4. The framework requires an observation trace $\mathbf{y}$, a neural circuit simulator, a random rule generator $G(d)$, an emission density $p(\mathbf{y}_t|\mathbf{z}_t)$ and methods for sampling $\mathbf{x}_0$ and evaluating $p(\mathbf{w}|\boldsymbol{\theta}, S)$ and $p(\boldsymbol{\theta}|S)$. Each of these components will now be described.

4 Experiment

4.1 Neural Circuit Simulator

To build the neural circuit simulator, the nine-neuron tap-withdrawal circuit was dissected from a deterministic full-connectome simulator called *simple-C-elegans* (SCE) [38]. SCE is based on single-compartment biophysical models presented by Wicks et al. [32] and Kunert et al. [39], wherein forward simulation of the membrane potential is accomplished by numerically integrating over time a set of ordinary differential equations which describe the relevant dynamics. The membrane potential $v_i \in \mathbb{R}$ of a given neuron i is governed by the equations:

$$c_T \frac{dv_i}{dt} = I_i^{(s)} - I_i^{(c)} - I_i^{(e)} - g_m(v_i - V_L) \tag{21}$$

$$I_i^{(e)} = \sum_{j=1}^{N_n} g^{(e)} W_{ji}^{(e)} (v_j - v_i) \tag{22}$$

$$I_i^{(c)} = \sum_{j=1}^{N_n} g^{(c)} W_{ji}^{(c)} s_j (v_i - E_j) \tag{23}$$

$$s_i = \left(1 + \exp\left\{K \frac{v_i - V_i^{Eq}}{V_{Range}}\right\}\right)^{-1} \tag{24}$$

$$V_i^{Eq} = (A^{-1} b)_i \tag{25}$$

$$A_{jk} = \begin{cases} \frac{1}{g_m} g^{(e)} W_{kj}^{(e)} & j \neq k \\ 1 + \frac{1}{g_m} \sum_{k=1}^{N_n} g^{(e)} W_{kj}^{(e)} + g^{(c)} W_{ji}^{(c)} \;/\; 2 & j = k \end{cases} \tag{26}$$

$$b_j = V_L + \frac{1}{g_m} \sum_{k=1}^{N_n} E_j g^{(c)} W_{kj}^{(c)}. \tag{27}$$

Here, $v \in \mathbb{R}^{N_n}$ is a vector of the neurons' membrane potentials (in mV), and $W_{ij}^{(c)}, W_{ij}^{(e)} \in \mathbb{R}$ are the total number of chemical and electrical synapses (i.e. synaptic weight) from neuron i to neuron j, at a given point in time. $I_i^{(s)}, I_i^{(c)}, I_i^{(e)} \in \mathbb{R}$ are the total currents flowing into neuron i from external stimuli (e.g. injected current), chemical synapses and electrical synapses, respectively (in amperes). V_i^{Eq} is the equilibrium membrane potential of neuron i, and E_i is its reversal potential. g_m is the membrane leakage conductance, and $g^{(c)}$ and $g^{(e)}$ are the maximum conductivity of chemical and electrical synapses (in Siemens). $g^{(c)}$ is modulated by the synaptic activity variable s_i. V_L is the leakage potential, and V_{Range} is the

presynaptic potential range over which synapses are activated. $\{E_i\}_{1\leq i\leq N_n}, V_L, g_m, g^{(c)}, g^{(e)}$ and V_{Range} are included in the simulator's biophysical constant state $\boldsymbol{\phi}$, and are set to experimentally determined values [32]. The virtual length of the simulation (arbitrary units), number of iterations per observation and dt are all set by the experimenter. These values determine the simulator resolution, with the total number of iterations given by the virtual length divided by dt.

Single-compartment models like this one assume that the physiological states and properties of a neuron are identical at every point in that neuron. This assumption prevents the simulator from capturing fine-grain dendrite dynamics which may be relevant to learning, but is necessary to make the generation of a large number of samples from $p(\mathbf{z}|\mathbf{w}_0, \boldsymbol{\theta}_k, S_k)$ tractable.

As the above differential equations indicate, SCE distinguishes between chemical and electrical synapses. Therefore, this simulator defines a pair of synaptic weight state traces, $\mathbf{w}^{(c)}, \mathbf{w}^{(e)} \in \mathbf{x}$, which represent as separate matrices the number of chemical and electrical synapses between each pair of neurons in the circuit. Out of the box, the synaptic weights in SCE are static. In order to accommodate the different functionalities of chemical and electrical synapses, the simulator was modified with two update rules $R^{(c)}$ and $R^{(e)}$, so that the chemical and electrical synaptic weights can be modulated independently.

In order to connect the calcium fluorescence observations $\mathbf{y}$ (outlined in Section 4.2) to the hidden membrane potential trace $\mathbf{v}$, the simulator is augmented with an additional set of differential equations which model intracellular calcium dynamic as a low-pass filter of membrane potential [40]. The intracellular calcium concentration $c_i \in \mathbb{R}$ of neuron i with membrane potential v_i is determined by the equations:

$$\frac{dc_i}{dt} = -\kappa_{Ca} I_i^{(Ca)} - \frac{c_i - c^{Base}}{\tau_{Ca}} \tag{28}$$

$$I_i^{(\mathrm{Ca})} = g_{Ca} s_\infty (v_i - E_{Ca}) \tag{29}$$

$$s_\infty(v) = \left(1 + \exp\left\{\frac{-(v - v^H)}{\rho}\right\}\right). \tag{30}$$

Here, $I_i^{(\mathrm{Ca})}$ denotes the calcium current across the membrane of neuron i and g_{Ca} and E_{Ca} are the maximum conductivity and reversal potential of the calcium channels, respectively. Time constant τ_{Ca} controls the rate at which intracellular calcium concentration returns to baseline c^{Base} and κ_{Ca} converts the calcium current to an intracellular calcium concentration. The steady-state activation of the voltage-gated calcium channels, s_∞, is parameterized by half-activation voltage v^H and slope factor ρ. $E_{Ca}, g_{Ca}, \tau_{Ca}, \kappa_{Ca}, c^{Base}, v^H$ and ρ are included in $\boldsymbol{\phi}$, and set to experimentally determined values [40].

Finally, the habituation-inducing stimulus (i.e. tap) train is simulated through the phasic injection of a depolarizing current into the mechanosensory neurons mediating response to tap (PLM, ALM, AVM) [31], as outlined by Wicks et al. [32]. With a virtual simulation

length of l and dt set to 0.001, the stimulus train contains $1000l$ stimuli.

The contents of the simulator define the states included in $\mathbf{x}$. At each time step t, $\mathbf{x}_t^{(i)}$, denotes: the chemical and electrical synaptic weight states $\mathbf{w}_t^{(c,i)}, \mathbf{w}_t^{(e,i)} \in \mathbb{R}^{N_n \times N_n}$; the membrane potential state $\mathbf{v}_t^{(i)} \in \mathbb{R}^{N_n}$; the intracellular calcium state $\mathbf{c}_t^{(i)} \in \mathbb{R}^{N_n}$; and the constant update rules $R^{(c,i)} = (\boldsymbol{\theta}^{(c,i)}, S^{(c,i)}), R^{(e,i)} = (\boldsymbol{\theta}^{(e,i)}, S^{(e)})$. N_n denotes the number of neurons in the simulated circuit (9 in this case).

4.2 Observation Trace and Emission Density

Each entry in an observation $\mathbf{y}_t$ corresponds to a noisy measurement of the activity of one of neurons in the circuit of interest, at time t during learning. There are many ways to measure individual neural activity in vivo. For example, extracellular electrical recording, performed with implanted electrodes, measures electrical activity within a small radius around the electrode tip [41]. Multi-electrode arrays are often used to triangulate the source of the electrical activity, allowing the separation of signals from different neurons (spike sorting). In order to avoid electrical pollution from sources outside the nearby neurons, a signal is usually recorded only if it overcomes some user-set threshold. However, this threshold prevents the reliable detection of membrane depolarization below the action potential threshold, and complicates the detection of hyperpolarization. Since many of *C. elegans*' neurons are thought to be graded rather than spiking [42, 43], extracellular electrical recording is insufficient for characterizing computationally-relevant neural dynamics in this organism.

Another common method for measuring neural activity is the imaging of intracellular calcium using fluorescent calcium indicators. These indicators may be injected in the form of fluorescent dyes into a brain region, where they then enter cells indiscriminately, or expressed in a specific class of cells in the form of a transgenic protein. In either case, the indicator fluoresces when it binds calcium, and the magnitude of this fluorescence is proportional to the amount of intracellular calcium present in the neuron [44]. Calcium participates in a vast number of cellular pathways [45], and is especially prominent in molecular processes in the nervous system, such as neurotransmitter release [46] and long-term potentiation [47]. Membrane depolarization at the axon terminal of a neuron causes voltage-gated calcium ion channels to open, resulting in a passive influx of extracellular calcium. Calcium triggers neurotransmitter exocytosis into the synaptic cleft, propagating the signal to the postsynaptic neuron [48]. Therefore, intracellular calcium concentration can be used as an indirect measure of the neuron activity in vivo. However, the calcium signal from an individual neuron is weak, and even slight animal motion causes imaging artifacts, so calcium imaging has traditionally been used to measure the aggregate activity of a group of neurons in a stationary animal. Recently, advances in fluorescence imaging and tracking software have facilitated whole-brain (~ 100 neurons) calcium imaging in free-moving *C. elegans*, and led to the correlation of the activity of individual neurons with locomotive behaviour [49, 50]. This opens the door to the analyses of neural dynamics during learning proposed by this thesis, so calcium fluorescence is chosen as the measure of neuron activity represented by $\mathbf{y}$ in the following experiments.

Unfortunately, no calcium fluorescent data are currently available for the neurons of the

tap-withdrawal circuit during habituation of the TWR. Therefore, SURF is demonstrated on synthetic calcium observations $\mathbf{y} \sim p(\mathbf{y}|\mathbf{z})$, where the intracellular calcium concentration state $\mathbf{c}$ is included in $\mathbf{z}$. The trace $\mathbf{c}$ necessary for sampling $p(\mathbf{y}|\mathbf{z})$ is generated by initializing $\mathbf{x}_0$ and iterating the simulator for $T+1$ observations. $\mathbf{w}_0^{(c)}$ and $\mathbf{w}_0^{(e)}$ are set to reflect the experimentally determined number of synaptic connections in a naive *C. elegans* [24, 31, 32]. $\mathbf{v}_0$ and $\mathbf{c}_0$ are sampled randomly according to the procedure outlined in Section 4.3. $R^{(c)}$ and $R^{(e)}$ are assigned Hebb's rule (see Section 2.1), corresponding to the synapse weight updates:

$$\mathbf{w}^{(c)} \leftarrow \mathbf{w}^{(c)} + \tau_w^{(c)}(\mathbf{v}\mathbf{v}^T \odot C^{(c)}) \tag{31}$$

$$\mathbf{w}^{(e)} \leftarrow \mathbf{w}^{(e)} + \tau_w^{(e)}(\mathbf{v}\mathbf{v}^T \odot C^{(e)}) \tag{32}$$

where the rate constants $\tau_w^{(c)}$ and $\tau_w^{(e)}$ are set to 0.001, $C^{(c)}$ and $C^{(e)}$ are the binary chemical and electrical connectivity matrices respectively [24], and $\odot$ denotes Hadamard multiplication. Hebb's rule is chosen because it produces a membrane potential trace $\mathbf{v}$ which corresponds to decreasing reversal magnitude, in response to a habituation-inducing stimulus train (Figure 2). Negative values correspond to forward locomotion, and may be interpreted as sensitization of the forward response. Alternatively, this could model 'below-zero' habituation, a phenomenon wherein continuing habituation past a response of zero magnitude causes spontaneous recovery to be slower. Testing SURF on synthetic data confers the ability to directly compare the optimized values $\mathbf{w}_0^{(c)*}$ and $\mathbf{w}_0^{(e)*}$ against ground truth. This approach also allows the computation of the loss resulting from the "true" values of $\mathbf{w}_0^{(c)}$, $\mathbf{w}_0^{(e)}$ and R, providing a baseline against which the performance of estimated values can be judged.

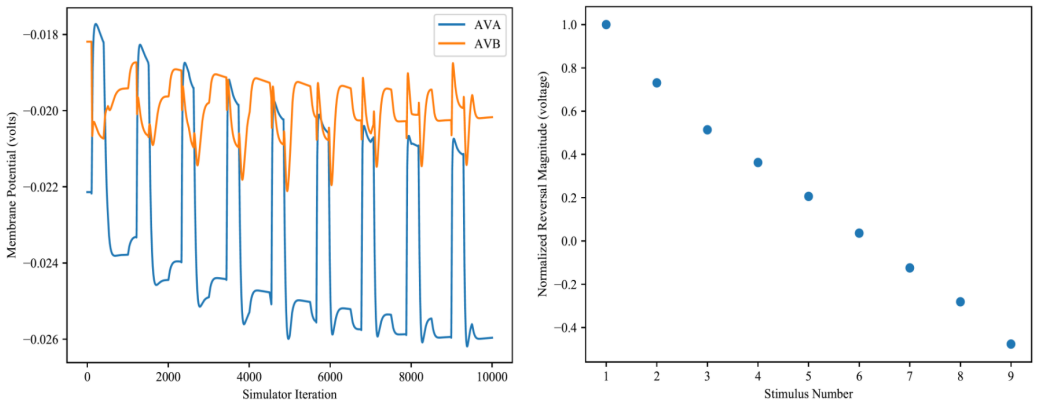

Figure 2: Left: Membrane potential traces for neurons AVA and AVB, with chemical and electrical synaptic weights updated by Hebb's rule. Right: Reversal magnitudes during the nine simulated stimulations, according to (9). Magnitudes were all divided by the maximum magnitude.

Given the generated $\mathbf{z}$, the observations are independent, so $\mathbf{y}$ can be obtained by sampling the emission density $p(\mathbf{y}_t|\mathbf{z}_t)$. The emission density can be modeled with a saturating Hill-type function [40] with zero-mean Gaussian noise [51]:

$$p(\mathbf{y}_t|\mathbf{z}_t) = \ p(\mathbf{y}_t|\mathbf{c}_t) \ = \kappa_F(\mathbf{c}_t \oslash (\mathbf{c}_t + K_d)) + d_F + \mathcal{N}(0, \sigma_F), \tag{33}$$

Here, each element of $\mathbf{y}_t \in \mathbb{R}^{N_n}$ denotes the fluorescence of one of the N_n neurons in the circuit and the corresponding element of the random vector $\mathbf{c}_t \in \mathbb{R}^{N_n}$ denotes the intracellular calcium for that neuron. κ_F and d_F are scale and offset parameters for the fluorescence trace, K_d is a measure of the fluorescent indicator's affinity to the calcium (i.e. dissociation constant), σ_F is the standard deviation of the Gaussian noise, and $\oslash$ denotes Hadamard division. κ_F, d_F and K_d, included in $\boldsymbol{\phi}$, are set to experimentally determined values [40].

When generating the synthetic observation trace, each $\mathbf{y}_t$ is obtained by evaluating the right hand side of (33) using a sample from $\mathcal{N}(0, \sigma_F = 0.001)$ and the value of $\mathbf{c}_t$ generated above. The same density is used for computing $p(\mathbf{y}_t|\mathbf{z}_t^{(i)})$ when evaluating the objective function during optimization. This probability is obtained using the following procedure:

$$\mu_F = \kappa_F(\mathbf{c}_t^{(i)} \oslash (\mathbf{c}_t^{(i)} + K_d)) + d_F \tag{34}$$

$$p(\mathbf{y}_t|\mathbf{z}_t^{(i)}) = \ \frac{1}{\sigma_F\sqrt{2\pi}} \exp\left\{\frac{-1}{2}\left(\frac{\mathbf{y}_t - \mu_F}{\sigma_F}\right)^2\right\} \tag{35}$$

where σ_F is again set to 0.001. Note that the mean of this Gaussian is generated by sampling from $p(\mathbf{y}|\mathbf{c}^{(i)})$ with $\sigma_F = 0$.

4.3 Hidden Trace Sampling

The evaluation of the SURF objective requires samples from $p(\mathbf{z}|\mathbf{w}_0, \boldsymbol{\theta}, S)$. Since the simulator and update rules are deterministic, variability is induced in the samples $\{\mathbf{z}^{(i)}\}_{1\leq i\leq N}$ by randomly initializing the states $\{\mathbf{v}_0^{(i)}\}_{1\leq i\leq N}$ and $\{\mathbf{c}_0^{(i)}\}_{1\leq i\leq N}$, and iterating the simulator T_{sim} times for each initialization. T_{sim} is the virtual length of the simulation (in seconds) divided by dt, both of which are set by the experimenter. $\boldsymbol{\phi}$ is not randomly initialized, but rather set to experimentally determined values [32, 39, 40] so that the simulator is able to integrate. Inference of these parameters is itself an interesting problem [52], but will not be addressed in this thesis. The habituation-inducing stimulus train used for the simulations is outlined by Wicks et al. [32].

Each $\mathbf{v}_0$ is sampled from $\mathcal{N}(\mu = -0.025, \sigma = 0.003)$. In order to prevent an extremely low emission probability $p(\mathbf{y}_0|\mathbf{c}_0)$, $\{\mathbf{c}_0^{(i)}\}_{1\leq i\leq N}$ are sampled using the importance sampling procedure outlined in Section 6.5.

$\mathbf{w}_0^{(c)}, \mathbf{w}_0^{(e)}, \boldsymbol{\theta}^{(c)}, \boldsymbol{\theta}^{(e)}, S^{(c)}$ and $S^{(e)}$ need to be initialized before the first optimization iteration, and are assigned by the optimization procedure for all subsequent optimization steps. On all optimization steps, all hidden trace samples are initialized with the same values for these states. Each synapse is assigned a weight sampled from a Rayleigh distribution with a scale parameter equal to the experimentally determined number of synapses of that type between the corresponding pair of neurons [24, 31, 32]. This initialization scheme is used in order to promote and expedite convergence to the correct weights, but should be replaced with a synapse-agnostic distribution in future work. The independent sampling of the weight matrices, and the fact that they are controlled by independent update rules, means that

$$-\log(p(\mathbf{w}_0|\boldsymbol{\theta}, S)) = -\log(p(\mathbf{w}_0^{(c)}|\boldsymbol{\theta}^{(c)}, S^{(c)})) - \log(p(\mathbf{w}_0^{(e)}|\boldsymbol{\theta}^{(e)}, S^{(e)})) \tag{36}$$

$$-\log(p(\boldsymbol{\theta}|S) = -\log(p(\boldsymbol{\theta}^{(c)}|S^{(c)}) - \log(p(\boldsymbol{\theta}^{(e)}|S^{(e)}) \tag{37}$$

Finally, the structure S and parameters $\boldsymbol{\theta}$ of each rule are sampled using the random rule generator $G(d)$, outlined in Sections 4.4 and 6.6. Each element of $\boldsymbol{\theta}$ is sampled independently from $\mathcal{N}(1, 0.1)$ during the generation of the update rule. $p(\mathbf{w}_0^{(c)}|\boldsymbol{\theta}^{(c)}, S^{(c)})$, $p(\mathbf{w}_0^{(e)}|\boldsymbol{\theta}^{(e)}, S^{(e)}), p(\boldsymbol{\theta}^{(c)}|S^{(c)})$, and $p(\boldsymbol{\theta}^{(e)}|S^{(e)})$ can be computed with these initialization distributions when evaluating the SURF objective function.

4.4 Synaptic Rule Generator

Although the experimenter has the option of hand-programming candidate synaptic update rules, SURF includes a random rule generator $G(d)$ which automates this process. $G(d)$ generates a random synaptic update rule $R^{(c)}$ or $R^{(e)}$ in the form of a string corresponding to a valid compositional pytorch function, and attaches the corresponding initialized parameters $\boldsymbol{\theta}^{(c)}$ or $\boldsymbol{\theta}^{(e)}$ to the computational graph for backpropagation. A chemical synapse update rule is generated recursively from the following context-free grammar (CFG). Generating a rule for electrical synapses is analogous.

`P` → `torch.add(P, P)` | `torch.mul(P, P)` | `torch.exp(P)` | `torch.matmul(P, P)` | `torch.unsqueeze(P, 1)` | `torch.unsqueeze(P, -1)` | `v_bar` | `W_c` | `Variable(torch.tensor([z]), requires_grad=True)`

This CFG can be easily expanded with additional relations for a more comprehensive search of the full structure space of programs. For example, conditional branching could be included to switch between discrete processes depending on neural state. If the CFG is converted to a probabilistic language, sampling and Bayesian operations, which may be prevalent throughout the brain's algorithms [53, 54], could be incorporated as well. `Variable(torch.tensor(z), requires_grad=True)` represents a new constant rule parameter, and becomes an entry in $\boldsymbol{\theta}^{(c)}$. If this is randomly chosen to be the next phrase `P` added, it is initialized with a sample from $\mathcal{N}(1, 0.1)$. On a given optimization step, the $\boldsymbol{\theta}^{(c)}$ is the same for all $\{\mathbf{x}_t^{(i)}\}_{1 \leq i \leq N, 0 \leq t \leq T}$. `W_c` and `v_bar` are variables in the code. The former represents $\mathbf{w}_t^{(c)}$ and the latter represents $\bar{\mathbf{v}}_t \in \mathbb{R}^{N_n}$ where each element $\bar{\mathbf{v}}_{t,i} = \mathbf{v}_{t,i} - V_i^{Eq}$.

Once a rule is generated in the form of a string, a lambda function is used to evaluate it:

```
eval('lambda self, v_bar, W_c, C_c:  {0}'.format(rule.string))
```

where `C_c` is the experimentally determined binary connectivity matrix for chemical synapses [24]. Generating an $R^{(c)}$ always begins with the relation `torch.mul(C_c, torch.mul{}{})` so that only existing synapses are updated. The empty argument is assigned a randomly chosen relation with a probability of 0.5, or a randomly chosen state or new constant, each with probability 0.25. The arguments of all added relations are assigned recursively in the same manner. If a recursion depth of d is reached (stopping criterion), each argument of the current relation is assigned a state or new constant, which also constitute base cases. Once a rule has been generated, its executability is tested in order to avoid invalid programs, such as those involving matrix multiplications with mismatched dimensions. If a rule cannot be executed, a new one is generated. As one of the principle contributions of this thesis, the code for $G(d)$ is presented in section 6.6.

4.5 Results

SURF successfully optimized the parameters of $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$ of a set of automatically generated candidate learning rules $R_1, \ldots, R_k$, improving their ability to recreate the synthetic observations and resulting in the losses decreasing to approach that achieved by the true learning rule. SURF was also able to estimate the initial synaptic weights $\mathbf{w}_0^{(c)}$ and $\mathbf{w}_0^{(e)}$, when the weight update dynamics were governed by a rule with the same compositional structure as the one used to generate the synthetic observations. The pair of rule structures which achieved the lowest loss was the one which matched the update rules used to generate the synthetic observations. This pair of candidate structures also resulted in the best qualitative reconstruction of the membrane potential and intracellular calcium concentration traces underlying the synthetic observations

The following results were produced by the following candidate pairs of randomly generated rule structures:

Candidate update rule pair A:

$$\mathbf{w}^{(c)} \leftarrow \mathbf{w}^{(c)} + C^{(c)} \odot e^{\boldsymbol{\theta}_1^{(c)}} \mathbf{w}^{(c)} \mathbf{v} \tag{38}$$

$$\mathbf{w}^{(e)} \leftarrow \mathbf{w}^{(e)} + C^{(e)} \odot \left(e^{\boldsymbol{\theta}_1^{(e)}} \mathbf{v} + \boldsymbol{\theta}_2^{(e)} \right) \tag{39}$$

Candidate update rule pair B:

$$\mathbf{w}^{(c)} \leftarrow \mathbf{w}^{(c)} + C^{(c)} \odot \exp\left\{ \boldsymbol{\theta}_1^{(c)} + \mathbf{v}^T \mathbf{v} \right\} \tag{40}$$

$$\mathbf{w}^{(e)} \leftarrow \mathbf{w}^{(e)} + C^{(e)} \odot (\boldsymbol{\theta}_1^{(e)} \mathbf{v}\mathbf{v}^T + e^{\boldsymbol{\theta}_2^{(e)}}) \tag{41}$$

The set of candidate rule pairs also included the hand-coded version of Hebb's rule with

parameters to be optimized (denoted candidate update rule pair C):

$$\mathbf{w}^{(c)} \leftarrow \mathbf{w}^{(c)} + C^{(c)} \odot \boldsymbol{\theta}_1^{(c)} \mathbf{v}\mathbf{v}^T \tag{42}$$

$$\mathbf{w}^{(e)} \leftarrow \mathbf{w}^{(e)} + C^{(e)} \odot \boldsymbol{\theta}_1^{(e)} \mathbf{v}\mathbf{v}^T \tag{43}$$

Here, $\boldsymbol{\theta}_i^{(c)}$ and $\boldsymbol{\theta}_i^{(e)}$ denote the ith parameter of the corresponding chemical and electrical synapse update rules, respectively. For the following experiments, the observation trace was produced with virtual stimulation time set to 4.0 and dt set to 0.001, resulting in 4000 simulator iterations, 160 observations, and 8 simulated tap stimuli.

As the loss curves in Figure 3 indicate, SURF was able to optimize the rule parameters and initial synaptic weights for all pairs of candidate rule structures, resulting in a rapid drops towards the true loss. Although not clear from Figure 3, the lowest loss is achieved by the optimized 'true' pair of candidate rule structures (C). The losses stabilized near their final values around optimization step 1200.

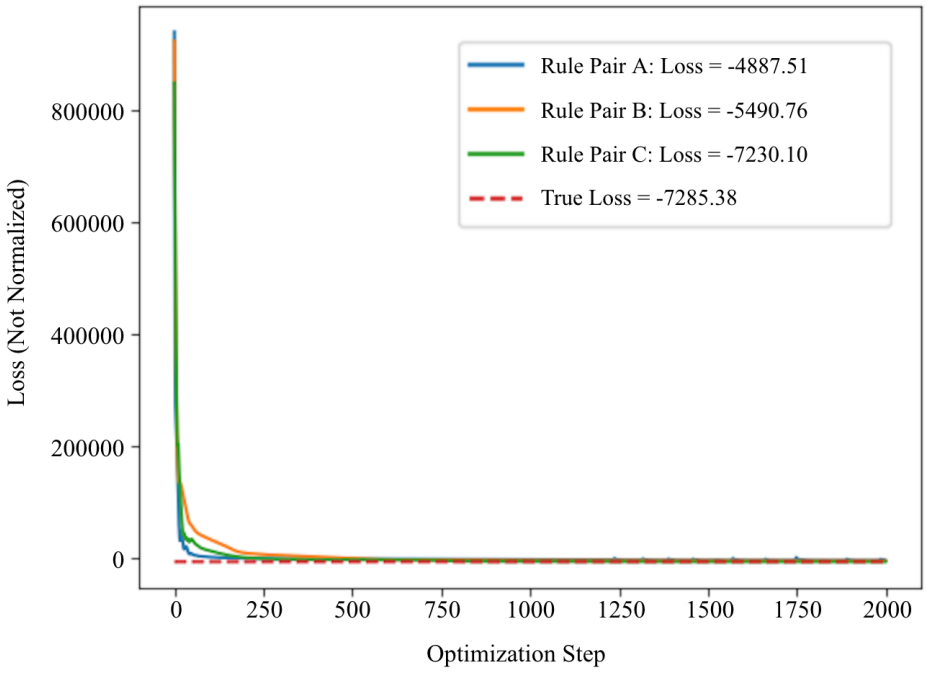

Figure 3: Loss curves for the optimization of candidate learning rule pairs A, B and C. The legend contains the final loss for each pair of candidate rules. The dashed red line represents the loss obtained with the initializations used for generating the synthetic observations.

Based on these results, it seems that all three pairs of candidate structures were sufficient to reproduce the observations to a reasonable degree. Predictably, the second-best pair of candidate rule structures was B, which includes an equivalent electrical synapse update rule to the true one, if $\boldsymbol{\theta}_2^{(e)}$ is set to zero. However, the weight update dynamics defined by

candidate rule pair C were qualitatively superior to A and B at reproducing the hidden cellular dynamics which underlie the synthetic observations. Special attention should be paid to the membrane voltages traces (Figure 4) of neurons AVA and AVB, as these neurons determine the reversal behaviour of the worm during each of the 8 simulated taps. Figure 4 clearly demonstrates that the simulator is best able to produce membrane potentials corresponding to the desired behavioural changes when equipped with candidate rules C and the corresponding $\mathbf{w}_0^{(c)*}, \mathbf{w}_0^{(e)*}, \boldsymbol{\theta}^{(c)*}$ and $\boldsymbol{\theta}^{(c)*}$. This is evident both in the increasing distance between the membrane potentials of AVA and AVB as stimulation proceeds, and the membrane potential dynamics of AVB within each of the eight tap stimulations. This further supports the idea that SURF returns the update rule and initial synaptic weights which best explain the observed learning trace.

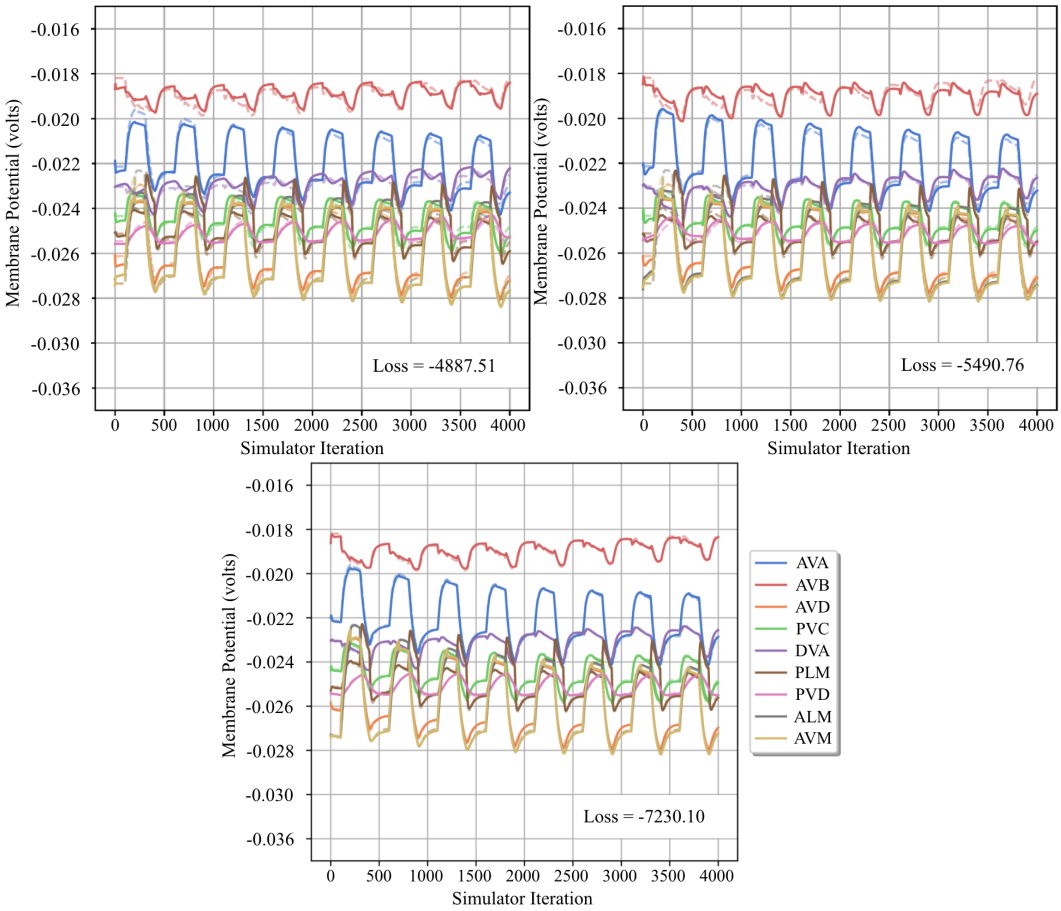

Figure 4: Membrane potential traces for all neurons in the tap-withdrawal circuit, simulated with the optimized update rule parameters and initial synaptic weights for each pair of candidate rule structures. The top left panel corresponds to candidate structure pair A, top right to pair B, and bottom to pair C. Dashed lines indicate the ground truth, produced by the simulator when equipped with the initial weights and update rule used to produce the synthetic observations.

The optimization of some of the entries of $\mathbf{w}_0^{(c)}$ and $\mathbf{w}_0^{(c)}$ are presented in Figure 5. For the majority of these entries, the true value was only retrieved when the simulator was equipped with the true update rule structures (C).

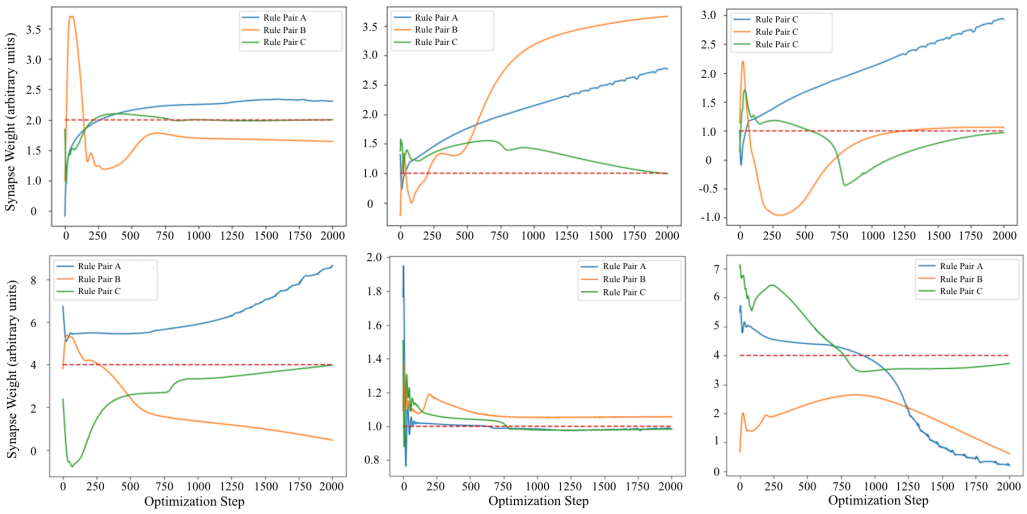

Figure 5: Optimization of initial synaptic weights. The top row depicts three electrical synapses, and the bottom row depicts three chemical synapses. The initial weights used for generating the synthetic observations are indicated by the red dashed line.

5 Discussion

This thesis presents SURF, a framework for estimating the naive synaptic weight matrix and update rule which result in synaptic weights dynamics that correspond to learning of a particular behaviour in a real organism. This is accomplished by automatically and randomly generating compositional structures for a set of candidate update rule parameterized by $\boldsymbol{\theta}$, and using maximum a posteriori estimation to maximize $p(\mathbf{w}_0, \boldsymbol{\theta}|\mathbf{y})$. $\mathbf{y}$ is a time series of noisy measurements of neuron activity during learning in the behaviour of interest. The learning behaviour on which SURF is tested is tap-withdrawal response habituation in *C. elegans*, with the synthetically generated calcium fluorescence of tap-withdrawal circuit neurons, resulting from Hebbian synaptic updates, used as the observations. Calcium fluorescence data from the tap withdrawal circuit during habituation are not currently available, so SURF was tested on synthetically generated observations with handset initialization of the synaptic weight matrix and update rule. The results of these experiments are promising. SURF was able to identify the rule structure used to generate the observations from a set of candidate rules. When the simulator was equipped with this rule structure, SURF was also able to estimate with reasonable accuracy the true initial synaptic weights. This suggests that theoretically, if SURF were applied on all possible update rule structures, it would identify one which produces simulator dynamics matching the given observations.

The next test for SURF is to replace the synthetic observations with more biologically representative ones. This is especially important because the synthetic observations used in this thesis do not capture the identifying features of habituation, such as more rapid habituation for weaker stimuli and after spontaneous recovery. There are several ways to obtain real observations from *C. elegans* as it undergoes TWR habituation, apart from collecting the necessary calcium fluorescence data. Recently, a framework has been submitted for inferring membrane potential and intracellular calcium concentration for a set of unobserved neurons given calcium fluorescence measurements from a different set of neurons in *C. elegans* [52]. If the calcium fluorescence of a large subset of neurons was measured during habituation in a real organism, this framework would allow the estimation of intracellular calcium in the neurons of the tap-withdrawal circuit, which could then be converted to calcium fluorescence and used as observations to test SURF. Although this type of data is available for free-moving *C. elegans* [49, 50], it has not yet been obtained for worms undergoing TWR habituation. However, there is a wealth of video recording data of worm behaviour over the course of TWR habituation. A new framework is currently under development by the author of this thesis which infers the membrane voltage and intracellular calcium concentration of all motor neurons from body shape observations extracted from video recordings of *C. elegans* during locomotion. The ultimate goal is to expand the latent space to include more of the worm's connectome, and since the tap-withdrawal circuit is directly connected to the motor circuitry, it may be possible to infer the intracellular calcium concentration of these neurons accurately. If this is the case, these values can be converted to fluorescence, and used as real observation with which SURF can be tested. A third option is to use feature-based inference to estimate the update rule and naive synaptic weights from quantitative measurements of reversal parameters over the course of TWR habituation. This is possible because features such as reversal magnitude and probability can be extracted from membrane potential traces, using equations such as (9). The inference can be conducted using pre-existing methods, such as approximate Bayesian computation, which compares

features extracted from simulated data against observed features [55, 56]. Another interesting experiment is the introduction of randomness into the transition density, through either a stochastic simulator or synaptic update rule, or both. As biologically-valid simulators are developed for more learning circuits in different organisms, the array of scenarios on which SURF can be tested will expand.

Another important area for future work is the development of a strategy for joint optimization over the continuous space of rule parameters and naive synaptic weights, and the discrete space of rule structures. One potential avenue for solving this problem is to treat the task as an infinitely many-armed bandit, with a finite budget of optimization steps [57, 58]. With this approach, the goal is to explore the potential of different update rule structures by spending optimization steps, then picking the most promising rule structure and fully optimizing its parameters. At each step, this algorithm could choose to further optimize a previously considered rule structure to get a better sense of its value (i.e. exploit) or to sample a new rule structure, which may perform better than any of the previously considered ones (i.e. explore).

Finally, there is reason to believe that SURF cannot possibly capture the algorithms by which synaptic strengths are modified during TWR habituation in *C. elegans*. For example, it has recently been suggested that some aspects of TWR habituation are controlled by long-range neuropeptide signals from neurons which do not participate in the synapse being strengthened [30]. This would suggest that Hebbian principles are not well-suited to describe the algorithm by which synaptic weights are modified in TWR habituation. If this is true, an entirely new construction of SURF is necessary, where update rules take arguments relating to the activity of distant neurons. This also means that the simulator to which SURF is applied in this thesis needs to be augmented to model the effects of long-range neuropeptide signals on membrane potentials.

Nonetheless, the results presented in this thesis are significant for several reasons. It is demonstrated that a neural simulator with static synaptic weights can be augmented with Hebbian synaptic update dynamics, while maintaining stable and predictable simulator outputs. It has also been shown that the compositional structure of such a rule has a significant impact on the neural activity dynamics it can model. This suggests that there may be synaptic update rules governing synaptic plasticity in real organisms which cannot be adequately modelled with the current canon of learning rules, but that still adhere to the Hebbian idea that synaptic weights are modulated based on pre- and postsynaptic activity. Furthermore, SURF is able to estimate the correct initial synaptic weights, and to assign the lowest loss to the pair of candidate rules structures which matched the true rules. Overall, SURF represents an interesting first step in the attempt to infer the synaptic weight dynamics which underlie behavioural learning in real organisms.

6 Appendix

6.1 Generative Process

$$
\begin{aligned}
p(\mathbf{w}, R|\mathbf{y}) &= \int_{\mathbf{z}} p(\mathbf{w}, \mathbf{z}, R|\mathbf{y}) \mathrm{d}\mathbf{z} && \text{by the law of total probability} \\
&= \int_{\mathbf{z}} \frac{p(\mathbf{y}|\mathbf{w}, \mathbf{z}, R) p(\mathbf{w}, \mathbf{z}, R)}{p(\mathbf{y})} \mathrm{d}\mathbf{z} && \text{by Bayes' theorem} \\
&\propto \int_{\mathbf{z}} p(\mathbf{y}|\mathbf{w}, \mathbf{z}, R) p(\mathbf{w}, \mathbf{z}, R) \mathrm{d}\mathbf{z} && \text{w.r.t. } \mathbf{w}, R \\
&= \int_{\mathbf{z}} p(\mathbf{y}|\mathbf{w}, \mathbf{z}, R) p(\mathbf{z}, \mathbf{w}|R) p(R) \mathrm{d}\mathbf{z} && \text{by Bayes' theorem}
\end{aligned}
$$

6.2 Objective Function Derivation

$-\log(p(\mathbf{w}_0, R|\mathbf{y}))$

$$
\begin{aligned}
&\propto -\log\left(\int_{\mathbf{z}} p(\mathbf{y}|\mathbf{z}, \mathbf{w}_0, R) p(\mathbf{z}, \mathbf{w}_0|R) p(R) \mathrm{d}\mathbf{z}\right) && \text{by Section 5.1} \\
&= -\log\left(\int_{\mathbf{z}} p(\mathbf{y}|\mathbf{z}, \mathbf{w}_0, R) p(\mathbf{z}|\mathbf{w}_0, R) p(\mathbf{w}_0|R) p(R) \mathrm{d}\mathbf{z}\right) && \text{by Bayes' theorem} \\
&= -\log(p(\mathbf{w}_0|R) p(R) \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{w}_0, R)}[p(\mathbf{y}|\mathbf{z}, \mathbf{w}_0, R)]) && \text{by defn. of expectation} \\
&= -\log(p(\mathbf{w}_0|R) p(R) \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{w}_0, R)}[p(\mathbf{y}|\mathbf{z})]) && \text{by the HMM} \\
&= -\log(\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{w}_0, R)}[p(\mathbf{y}|\mathbf{z})]) - \log(p(\mathbf{w}_0|R)) - \log(p(R)) && \text{by log product rule}
\end{aligned}
$$

6.3 Objective Function Simplification

$$
\begin{aligned}
-\log(p(\mathbf{w}_0, R|\mathbf{y})) \;&\propto -\log(\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{w}_0,R)}[p(\mathbf{y}|\mathbf{z})]) - \log(p(\mathbf{w}_0|R)) - \log(p(R)) \\
&\text{Let } R = (\text{structure } S, \text{parameters } \boldsymbol{\theta}) \\
\Rightarrow -\log(p(\mathbf{w}_0, \boldsymbol{\theta}|\mathbf{y})) \;&\propto -\log(\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{w}_0,\boldsymbol{\theta},S)}[p(\mathbf{y}|\mathbf{z})]) - \log(p(\mathbf{w}_0|\boldsymbol{\theta}, S) \\
&\quad - \log(p(\boldsymbol{\theta}|S)) - \log(p(S)) \\
&\propto -\log(\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{w}_0,\boldsymbol{\theta},S)}[p(\mathbf{y}|\mathbf{z})]) - \log(p(\mathbf{w}_0|\boldsymbol{\theta}, S) \\
&\quad - \log(p(\boldsymbol{\theta}|S)) \quad \text{w.r.t. } \mathbf{w}_0, \boldsymbol{\theta}
\end{aligned}
$$

6.4 SURF Algorithm

Algorithm 1 Synaptic Update Rule Finder

procedure SURF($d, k, n, N \in \mathbb{Z}^+$)
  $\mathcal{L} \leftarrow \mathbf{0}_{k\times 1}$
  for $i \leftarrow 1$ **to** k **do**
    $(\boldsymbol{\theta}_i, S_i) \leftarrow G(d)$ ▷ Randomly initialize rule structure and parameters
    $\mathbf{w}_{0,i} \sim p(\mathbf{w}_0|\boldsymbol{\theta}_i, S_i)$ ▷ Randomly initialize initial weights
    for $t \leftarrow 1$ **to** n **do** ▷ Descend gradient n times, with variable step size η
      for $j \leftarrow 1$ **to** N **do** ▷ Generate N samples for MC integration
        $\mathbf{z}^{(j)} \sim p(\mathbf{z}|\mathbf{w}_{0,i}, \boldsymbol{\theta}_i, S_i)$
      end for
      $\mathcal{L}_i \leftarrow -\log\left(\frac{1}{N}\sum_{j=1}^{N}\prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{z}_t^{(j)})\right) - \log(p(\mathbf{w}_{0,i}|\boldsymbol{\theta}_i, S_i)) - \log(p(\boldsymbol{\theta}_i|S_i))$
      $[\mathbf{w}_{0,i}, \boldsymbol{\theta}_i] \leftarrow [\mathbf{w}_{0,i}, \boldsymbol{\theta}_i] - (\eta)^t \nabla_{\mathbf{w}_{i,0}, \boldsymbol{\theta}_i}(\mathcal{L})$
    end for
  end for
  $j \leftarrow \text{argmin}_{1\le i\le k}\, \mathcal{L}_i$
  return $(\mathbf{w}_{0,j}, \boldsymbol{\theta}_j, S_j)$
end procedure

6.5 Initial Calcium Importance Sampling

Algorithm 2 Initial Calcium Importance Sampler

procedure InitializeCa($N, N_n \in \mathbb{Z}^+$)
  $\mathbf{c}^0 \leftarrow \mathbf{0}_{N \times N_n}$
  for $n \leftarrow 1$ **to** N_n **do**
    $\mathbf{s} \sim \mathcal{N}_{10N}(0, \mathbb{I}_{10N}(5 \cdot 10^{-10}))$
    $\mathbf{s} \leftarrow \text{abs}(\mathbf{s})$ ▷ Element-wise absolute value
    $\mathbf{s} \leftarrow \kappa_F(\mathbf{s} \oslash (\mathbf{s} + K_d) + d_F$
    $\mathbf{p} \leftarrow p(\mathbf{s}) \sim \mathcal{N}_{10N}(\mathbf{y_0}, \mathbb{I}_{10N}(\sigma_F))$ ▷ σ_F equals 0.001
    $\mathbf{p} \leftarrow \exp(\mathbf{p} - \max(\mathbf{p}))$
    $\mathbf{p} \leftarrow \mathbf{p} \,/\, \|\mathbf{p}\|_1$
    $\mathbf{u} \sim U_N(0, 1)$
    $\mathbf{u} \leftarrow (\mathbf{u} + [0 : N-1]) \,/\, N$
    $\mathbf{i} \leftarrow \text{digitize}(\mathbf{u}, \text{cumsum}(\mathbf{p}))$ ▷ Using numpy functions
    $\mathbf{c}^0_{:,n} \leftarrow \mathbf{s}[\mathbf{i}]$ ▷ Python-style array indexing
  end for
  return $\mathbf{c}^0$
end procedure

6.6 Recursive Random Rule Generation Algorithm

```
1
2  class Operation:
3      def __init__(self, string, args):
4          self.string = string
5          self.args = args
6          self.curr_arg = 1
7          self.n_args = len(args)
8
9  class Rule:
10     def __init__(self, d):
11         self.string = ''
12         self.curr_d = 0
13         self.max_d = d
14         self.params = {}
15         self.ops_stack = []
16         self.operations = build_operations()
17         self.states = build_states()
18         build_rule(self)
19
20 def build_rule(rule): # entry into recursion
21     rand_operation = Operation('torch.mul(C_c, torch.mul({}, {}))', ('param',
       'prim', )) # guarantees that rule's computational graph depth > 2
22     rule.ops_stack.insert(0, rand_operation)
23     args = [build_arg(rule, arg) for arg in rand_operation.args]
24     rule.string = rand_operation.string.format(*args)
25     return
26
27 def build_arg(rule, arg): # recursive function
28     if rule.curr_depth == rule.max_depth
29         if arg == 'param':
30             return get_parameter(rule)
31         elif arg == 'state':
32             return get_state(rule)
33         elif random.choice([0, 1]):
34             return get_state(rule)
35         else:
36             return get_parameter(rule)
37
38     else:
39         if arg == 'param':
40             return get_parameter(rule)
41         elif arg == 'state':
42             return get_state(rule)
43         elif random.choice([0, 1]): # p(operation) = 0.5
44             rand_operation = get_operation(rule)
45             args = [build_arg(rule_str, arg) for arg in rand_operation.args]
46             return rand_operation.string.format(*args)
47         else:
48             if random.choice([0, 1]): # p(state) = p(param) = 0.25
49                 return get_state(rule_str)
50             else:
51                 return get_parameter(rule_str)
52
53 def get_operation(rule):
54     rand_operation = random.choice(rule.operations)
55     rule.curr_depth += 1
56     rule.ops_stack.insert(0, rand_operation)
57     return rand_operation
```

```
58
59
60 def get_state(rule):
61     if rule.ops_stack[0].curr_arg == rule.ops_stack[0].n_args:
62         rule.curr_depth -= 1
63         del rule.ops_stack[0]
64     else:
65         rule.ops_stack[0].curr_arg += 1
66     return random.choice(rule.states)
67
68
69 def get_parameter(rule):
70     if rule.ops_stack[0].curr_arg == rule.ops_stack[0].n_args:
71         rule.curr_depth -= 1
72         del rule.ops_stack[0]
73     else:
74         rule.ops_stack[0].curr_arg += 1
75     return 'torch.exp({})'.format(get_fresh_param(rule))
76
77
78 def get_fresh_param(rule):
79     # sample new theta entries from Gaussian with mean=1 sd=0.1
80     param = 'self.parameters.{}{}'.format(rule.param_string, str(rule.
      param_idx))
81     rule.param_idx += 1
82     x = 1.0 + np.random.normal(0, 0.1)
83     rule.parameters[param] = Variable(torch.tensor([x]), requires_grad=True)
84     return param
85
86
87 def build_operations(): # new relations can be added here
88     mul = Operation('torch.mul({}, {})', ('prim', 'prim', ))
89     add = Operation('torch.add({}, {})', ('prim', 'prim', ))
90     exp = Operation('torch.exp({})', ('prim', ))
91     us1 = Operation('torch.unsqueeze({}, 1)', ('prim', ))
92     us2 = Operation('torch.unsqueeze({}, -1)', ('prim', ))
93     return [mul, add, exp, us1, us2]
94
95 def build_states():
96     return ['v', 'W_c']
97
98 if __name__ == "__main__":
99     rule = Rule(d) # d = user specified max recursion depth
```

References

[1] Pepe Alcamí and Alberto E. Pereda. Beyond plasticity: the dynamic impact of electrical synapses on neural circuits. *Nature Reviews Neuroscience*, 20:253–271, 2019.

[2] Eric R. Kandel. *Principles of Neural Science*. The McGraw-Hill Companies, Inc., 5 edition, 2013.

[3] Giovanni Berlucchi and Henry A. Buchtel. Neuronal plasticity: historical roots and evolution of meaning. *Experimental Brain Research*, 192:307–319, 2009.

[4] Donald O. Hebb. *The Organization of Behaviour: A Neuropsychological Theory*. Wiley, New York, New York, 1949.

[5] P. Read Montague and Terrence J. Sejnowski. The predictive brain: Temporal coincidence and temporal order in synaptic learning mechanisms. *Learning & Memory*, 1:1–33, 1994.

[6] Michael Domjan. *The Principles of Learning and Behaviour*. Cengage Learning, Stamford, Connecticut, 2015.

[7] Timothy V. P. Bliss and Terje Lømo. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology*, 232:331–356, 1973.

[8] John Lisman. A mechanism for the Hebb and the anti-Hebb processes underlying learning and memory. 86:9574–9578.

[9] Serena M. Dudek and Mark F. Bear. Homosynaptic long-term depression in area cal of hippocampus and effects of n-methyl-d-aspartate receptor blockade. *Proceedings of the National Academy of Sciences of the United States of America*, 89:4363–4367, 1992.

[10] Pablo E. Castillo, Chiayu Q. Chiu, and Reed C. Carroll. Long-term synaptic plasticity at inhibitory synapses. *Current Opinion in Neurobiology*, 21:328–338, 2011.

[11] Robert C. Malenka and Mark F. Bear. Ltp and ltd:an embarrassment of riches. *Neuron*, 44:5–21, 2004.

[12] Mark F. Bear, Leon N. Cooper, and Ford F. Ebner. A physiological basis for a theory of synapse modification. *Science*, 237:42–48, 1987.

[13] Elie Bienenstock, Leon N. Cooper, and Paul Munro. Theory of the development of the neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2:32–48, 1982.

[14] Eugene E. Clothiaux, Mark F. Bear, and Leon N. Cooper. Synaptic plasticity in visual cortex: Comparison of theory with experiment. *Journal of Neurophysiology*, 66:1785–1804, 1991.

[15] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18:10464–10472, 1998.

[16] Pierre Yger, Marcel Stimberg, and Romaine Brette. Fast learning with weak synaptic plasticity. *Journal of Neuroscience*, 35:13351–13362, 2015.

[17] Thomas C. Südhof and Robert C. Malenka. Understanding synapses: Past, present, and future. *Neuron*, 60:469–476, 2008.

[18] Stephen D. Glasgow, Ryan McPhedrain, Jeanne F. Madranges, Timothy E. Kennedy, and Edward S. Ruthazer. Approaches and limitations in the investigation of synaptic transmission and plasticity. *Frontiers in Synaptic Neuroscience*, 11:20, 2019.

[19] Peter Dayan and Larry F. Abbott. *Theoretical Neuroscience: Computational and mathematical modeling of neural systems*. The MIT Press, Cambridge, Massachusetts, 2001.

[20] Erkki Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[21] Jonathan Hodgkin, H. Robert Horvitz, Barbara R. Jasny, and Judith Kimble. C. elegans: Sequence to biology. *Science*, 282:2011.

[22] Jun Yoshimura, Kazuki Ichikawa, Massa J. Shoura, Karen L. Artiles, Idan Gabdank, Lamia Wahba, Cheryl L. Smith, Mark L. Edgley, Ann E. Rougvie, Andrew Z. Fire, Shinichi Morishita, and Erich M. Schwarz. Recompleting the Caenorhabditis elegans genome. *Genome Research*, 29:1009–1022, 2019.

[23] Sydney Brenner. The genetics of Caenorhabditis elegans. *Genetics*, 77:71–94, 1974.

[24] John Graham White, E. Southgate, J. N. Thomson, and Sydney Brenner. The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 314:1–340, 1986.

[25] Eduardo J. Izquierdo and Randall D. Beer. Connecting a connectome to behavior: An ensemble of neuroanatomical models of C. elegans klinotaxis. *PLoS Computational Biology*, 9:e1002890, 2013.

[26] Catherine H. Rankin, Christine D. O. Beck, and Catherine M. Chiba. Caenorhabditis elegans: a new model system for the study of learning and memory. *Behavioural Brain Research*, 37:89–92, 1990.

[27] Hisayuki Amano and Ichiro N. Maruyama. Aversive olfactory learning and associative long-term memory in Caenorhabditis elegans. *Learning & Memory*, 18:654–665, 2011.

[28] Evan L. Ardiel and Catharine H. Rankin. An elegant mind: Learning and memory in Caenorhabditis elegans. *Learning & Memory*, 17:191–201, 2010.

[29] Jacqueline K. Rose and Catharine H. Rankin. Analyses of habituation in Caenorhabditis elegans. *Learning & Memory*, 8:63–69, 2001.

[30] Troy A. McDiarmid, Alex J. Yu, and Catharine H. Rankin. Habituation is more than learning to ignore: Multiple mechanisms serve to facilitate shifts in behavioral strategy. *BioEssays*, 41:10.1002, 2019.

[31] Stephen R. Wicks and Catherine H. Rankin. Integration of mechanosensory stimuli in Caenorhabditis elegans. *Journal of Neuroscience*, 15:2434–2444, 1995.

[32] Stephen R. Wicks, Chris J. Roehrig, and Catherine H. Rankin. A dynamic network simulation of the nematode tap withdrawal circuit: Predictions concerning synaptic function using behavioral criteria. *Journal of Neuroscience*, 16(12):4017–4031, 1996.

[33] Stephen R. Wicks and Catherine H. Rankin. The effects of tap withdrawal response habituation on other withdrawal behaviors: The localization of habituation in the nematode Caenorhabditis elegans. *Behavioural Neuroscience*, 111:342–353, 1997.

[34] Taizo Kawano, Michelle D. Po, Shangbang Gao, George Leung, William S. Ryu, and Mei Zhen. An imbalancing act: Gap junctions reduce the backward motor circuit activity to bias C. elegans for forward locomotion. *Neuron*, 72(4):572–586, 2011.

[35] Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later, April 2008.

[36] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*, chapter Appendix A. 3 edition, 2019. Online preprint from https://web.stanford.edu/ jurafsky/slp3/.

[37] Christopher Zach. Dual decomposition for joint discrete-continuous optimization. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, 2013.

[38] Adam Marbelstone. simple-C-elegans. https://github.com/adammarblestone/simple-C-elegans, 2016.

[39] James Kunert, Eli Shlizerman, and J. Nathan Kutz. Low-dimensional functionality of complex network dynamics: Neuro-sensory integration in the caenorhabditis elegans connectome. *Physical Review E*, 89(5):052805, 2014.

[40] Vahid Rahmati, Knut Kirmse, Dimitrije Marković, Knut Holthoff, and Stefan J. Kiebel. Inferring neuronal dynamics from calcium imaging data using biophysical models and bayesian inference. *PLoS Computational Biology*, 12:e1004736, 2016.

[41] György Buzsáki, Costas A. Anastassiou, and Christof Koch. The origin of extracellular fields and currents - EEG, ECoG, LFP and spikes. *Nature Reviews Neuroscience*, 13:407–420, 2012.

[42] Shawn R. Lockery and Miriam B. Goodman. The quest for action potentials in C. elegans neurons hits a plateau. *Nature Neuroscience*, 12:377–378, 2009.

[43] Qiang Lui, Gunther Hollopeter, and Erik M. Jorgensen. Graded synaptic transmission at the Caenorhabditis elegans neuromuscular junction. *Proceedings of the National Academy of Sciences of the United States of America*, 106:10823–10828, 2009.

[44] Werner Göbel and Fritjof Helmchen. In vivo calcium imaging of neural network function. *Physiology*, 22:359–365, 2007.

[45] Michael J. Berridge and Martin D. Bootman. The versatility and universality of calcium signalling. *Nature Reviews Molecular Cell Biology*, 1:11–21, 2000.

[46] Bernard Katz and Ricardo Miledi. Ionic requirements of synaptic transmitter release. *Nature*, 215:651, 1967.

[47] Robert C. Malenka. The role of postsynaptic calcium in the induction of long-term potentiation. *Molecular Neurobiology*, 5:289–295, 1992.

[48] Thomas C. Südhof. Calcium control of neurotransmitter release. *Cold Springs Harbour Perspectives in Biology*, 4:a011353, 2012.

[49] Saul Kato, Harris S. Kaplan, Tina Schrödel, Susanne Skora, Theodore H. Lindsay, Eviatar Yemini, Shawn Lockery, and Manuel Zimmer. Global brain dynamics embed the motor command sequence of Caenorhabditis elegans. *Cell*, 16:656–669, 2015.

[50] Jeffrey P. Nguyen, Frederick B. Shipley, Ashley N. Linder, George S. Plummer, Mochi Liu, Sagar U. Setru, Joshua W. Shaevitz, and Andrew M. Leifer. Whole-brain calcium imaging with cellular resolution in freely behaving Caenorhabditis elegans. *Proceedings of the National Academy of Sciences of the United States of America*, 113:E1074–E1081, 2016.

[51] Yuriy Mishchencko, Joshua T. Vogelstein, and Liam Paninski. A Bayesian approach for inferring neuronal connectivity from calcium fluorescent imaging data. *The Annals of Applied Statistics*, 5:1229–1261, 2011.

[52] Andrew Warrington, Arthur Spencer, and Frank Wood. The virtual patch clamp: Imputing C. elegans membrane potentials from calcium imaging. *arXiv preprint arXiv:1907.11075*, 2019.

[53] Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Computational Biology*, 7:e1002211, 2011.

[54] Timothy R. Darlington, Jeffrey M. Beck, and Stephen G. Lisberger. Neural implementation of bayesian inference in a sensorimotor behavior. 21:1442–1451, 2018.

[55] Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation. *Journal of the Royal Statistical Society, Series B, Statistical Methodology*, 73:419–474, 2012.

[56] Mikael Sunnaåer, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate Bayesian Computation. *PLoS Computational Biology*, 9:e1002803, 2013.

[57] Yizao Wang, Jean-Yves Audibert, and Rémi Munos. Algorithms for infinitely many-armed bandits. *Advances in Neural Information Processing Systems 21*, 2008.

[58] Haifang Li and Yingce Xia. Infinitely many-armed bandits with budget constraints. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.