Learning in large-scale spiking neural networks

by

Trevor Bekolay

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2011

Abstract

Learning is central to the exploration of intelligence. Psychology and machine learning provide high-level explanations of how rational agents learn. Neuroscience provides low-level descriptions of how the brain changes as a result of learning. This thesis attempts to bridge the gap between these two levels of description by solving problems using machine learning ideas, implemented in biologically plausible spiking neural networks with experimentally supported learning rules.

We present three novel neural models that contribute to the understanding of how the brain might solve the three main problems posed by machine learning: supervised learning, in which the rational agent has a fine-grained feedback signal, reinforcement learning, in which the agent gets sparse feedback, and unsupervised learning, in which the agents has no explicit environmental feedback.

In supervised learning, we argue that previous models of supervised learning in spiking neural networks solve a problem that is less general than the supervised learning problem posed by machine learning. We use an existing learning rule to solve the general supervised learning problem with a spiking neural network. We show that the learning rule can be mapped onto the well-known backpropagation rule used in artificial neural networks.

In reinforcement learning, we augment an existing model of the basal ganglia to implement a simple actor-critic model that has a direct mapping to brain areas. The model is used to recreate behavioural and neural results from an experimental study of rats performing a simple reinforcement learning task.

In unsupervised learning, we show that the BCM rule, a common learning rule used in unsupervised learning with rate-based neurons, can be adapted to a spiking neural network. We recreate the effects of STDP, a learning rule with strict time dependencies, using BCM, which does not explicitly remember the times of previous spikes. The simulations suggest that BCM is a more general rule than STDP.

Finally, we propose a novel learning rule that can be used in all three of these simulations. The existence of such a rule suggests that the three types of learning examined separately in machine learning may not be implemented with separate processes in the brain.

Acknowledgements

I would like to thank Xuan Choo, Travis DeWolf, Bruce Bobier, Daniel Rasmussen, Terry Stewart, and Charlie Tang for their help getting acquainted with the University of Waterloo, the Center for Theoretical Neuroscience, and the NEF. Extra thanks to Terry and Xuan for continuing help with NEF models and Nengo programming, and for providing some of the figures in this thesis.

Some comments on early drafts were provided by Lucy Spardy and Travis DeWolf, to whom I am grateful. Gratitude also goes to my readers, Matthijs van der Meer and Jeff Orchard, for their comments and fruitful discussion.

Most of all, I am truly indebted to my supervisor Chris Eliasmith, whose insights are responsible for any good ideas that may have sneaked into this thesis.

Dedication

For my parents, Cathy and David, who would be proud of me if my only publication was on their refrigerator.

Table of Contents

List of Figures

Chapter 1

Introduction

Learning is central to the exploration of intelligence. Many biological and artificial systems begin with a small set of abilities, and as time goes on, develop new abilities and refine existing abilities through learning and other forms of adaptation.

Psychology and machine learning provide high-level explanations of how systems learn. Neuroscience provides low-level descriptions of how the brain changes as a result of learning. The overarching goal of this thesis is to bridge the gap between high-level explanations of system-level learning, and low-level descriptions of the biological consequences of learning.

At the intersection of the two levels of description is the field of theoretical neuroscience. From this field have come dozens of spiking neuron models: flexible computational devices mimicking the biological neurons that make up brains. As computational power has increased, the ability to connect very large numbers of spiking neurons together in computer models has opened the way for models that can explain behaviour while still permitting analysis at the level of spikes.

The spiking level is essential for theoretical models to maintain *biological plausibility*. By modelling neural systems in a biologically plausible manner, theoreticians hope to make predictions that can be explored through neuroscientific and behavioural experiments.

This thesis begins by exploring how neurobiological systems change as a result of learning. It then gives background on large-scale spiking neural networks, which is the primary tool used to investigate specific learning problems faced by systems attempting to act rationally. We use this tool under neurobiological constraints to model supervised, reinforcement, and unsupervised learning, the three main learning problems identified by machine learning.

1.1 Thesis organization

Chapter 2, synaptic plasticity, provides some experimental neuroscientific background on learning. Learning in the brain is theorized to be implemented by changes in the strength of connection between neurons. The connection between two neurons is called a synapse, and the ability for its strength to change over time is called plasticity. This chapter defines these terms further through a brief introduction to neuroscience, and then reviews the primary experimental protocols that can induce synaptic plasticity.

Chapter 3, large-scale neural modelling, provides the theoretical neuroscientific background that is applied to the problem of learning. Spiking neuron models are presented, focusing on the phenomenological leaky integrate-and-fire model. A method of combining large numbers of single neuron models called the Neural Engineering Framework (NEF) is reviewed, as it is necessary to understand the models presented in subsequent chapters.

Chapter 4, supervised learning, poses the supervised learning problem, in which a system learns by explicitly being taught the solution. As will be the model for the next two chapters, the supervised learning problem is explicitly stated, and previous models attempting to solve the problem are discussed. Models from machine learning and theoretical neuroscience are presented. We then propose our own solution to the general supervised learning problem through a model created with the NEF.

Chapter 5, reinforcement learning, follows chapter 4's template, but examines the reinforcement learning problem, in which a system learns through sparse feedback, but not explicit teaching. Previous models that solve the reinforcement learning problem are summarized. In this chapter, we also briefly present relevant psychological and neuroscientific literature. We propose a model that solves a portion of the reinforcement learning problem, and discuss possible solutions to portions of the problem left unsolved in this thesis.

Chapter 6, unsupervised learning, investigates the final machine learning problem, in which a system learns without any external feedback or teaching. While previous models are discussed, the bulk of this chapter is dedicated to mathematical descriptions of the low-level plasticity experiments discussed in chapter 2. Implementations of some of these descriptions give insight into how the brain learns in the absence of feedback.

Finally, **chapter 7, discussion and conclusions,** concludes the thesis and revisits the goals that immediately follow this section. Future research directions based on this work are reviewed.

1.2 Thesis goals

The motivation of this thesis is to bridge the gap between high-level descriptions of learning and low-level neural realizations of learning. A complete story for how low-level connection strength changes bring about all of human behaviour is overly ambitious, so we make the following the goals of this thesis.

- To provide a comprehensive review of synaptic plasticity literature, identifying learning rules that evidence suggests are implemented in the brain.
- To show that supervised learning has not been adequately solved by previous biologically plausible models.
- To solve the supervised learning problem with a biologically plausible model.
- To recreate behavioural and neural results from a reinforcement learning task with a single biologically plausible model.
- To show that an unsupervised learning rule without strict time-dependence built in (the BCM rule) can exhibit the temporal effects of a time-dependent rule (the STDP rule).

Chapter 2

Synaptic plasticity

In order to understand the constraints involved in brain modelling, we begin this chapter with an introduction to the relevant neurobiology. This introduction is a simplification of the true complexity of neurobiology, but serves as a good introduction to the phenomenological level of the models presented in this thesis.

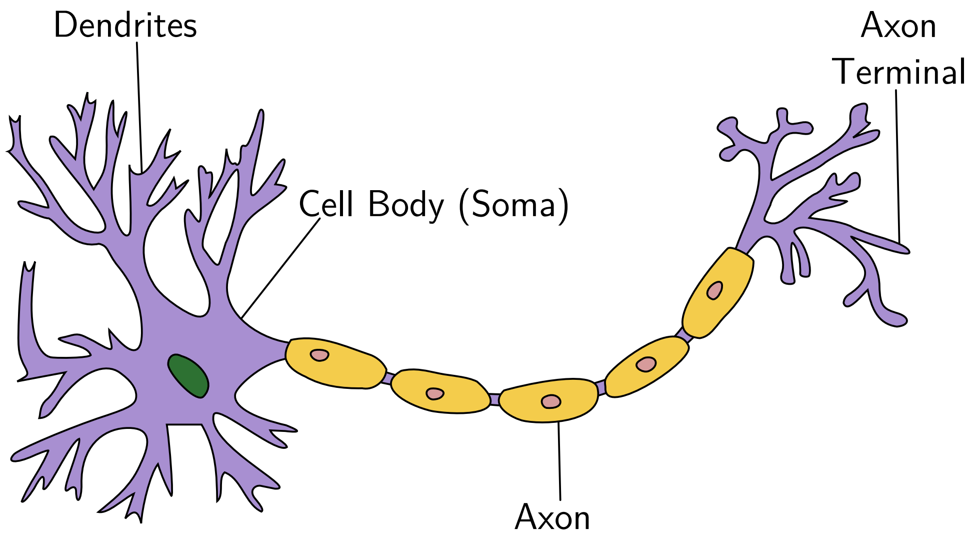


Figure 2.1: Illustration of the main parts of a neuron.

The brain is primarily made up of neurons. In general, a neuron can be thought of as a very simple computational device; it takes in input from other neurons through dendrites, performs a non-linear transformation on the weighted sum of the dendritic input, and outputs electrical signals through its axon. Those electrical signals are in the form of action potentials, or spikes, which are short bursts of current flowing through the axon. Axons branch off, and connect with the dendrites of many other neurons (see figure 2.1). The connection point between the axon terminal of one neuron and the dendrite of another neuron is called a synapse (see figure 2.2). We refer to the neuron whose output is being transmitted through its axon as the *presynaptic* neuron, and the neuron receiving input through its dendrite as the *postsynaptic* neuron.

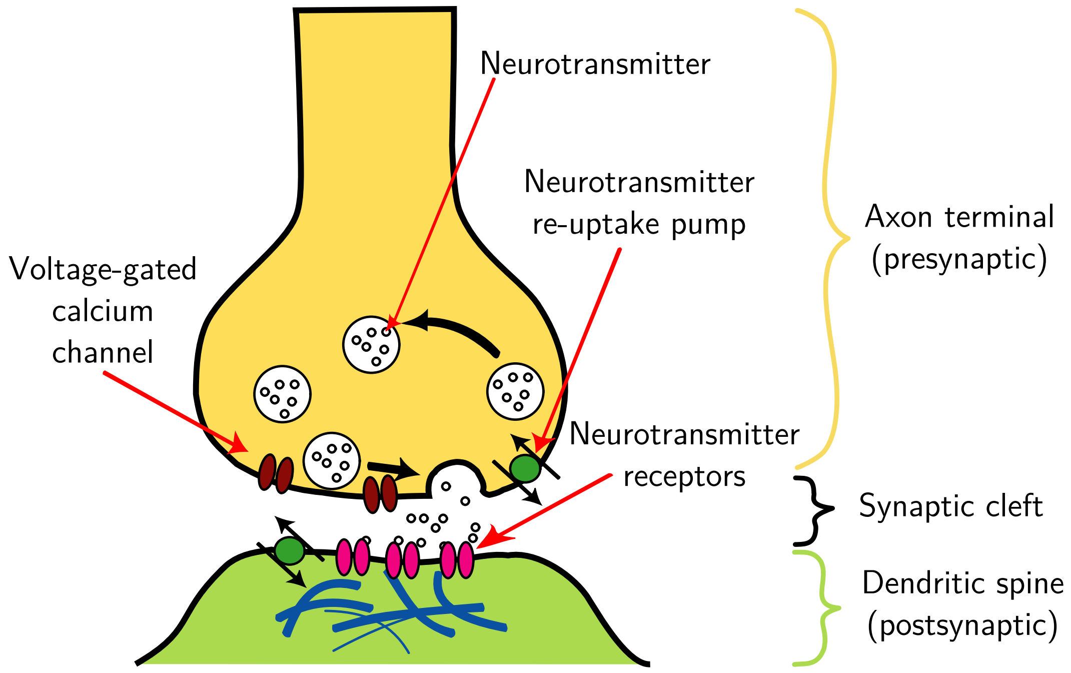

Neurotransmitter
Neurotransmitter re-uptake pump
Voltage-gated calcium channel
Neurotransmitter receptors
Axon terminal (presynaptic)
Synaptic cleft
Dendritic spine (postsynaptic)

Figure 2.2: Illustration of the main parts of a synapse.

The vast majority of synapses in the brain are chemical synapses, meaning that information is transmitted by tiny vesicles of molecules called *neurotransmitters* that are emitted from the presynaptic neuron, travel across the synaptic cleft, and bind to *receptors* in the postsynaptic neuron designed to accept a specific type of neurotransmitter. When the presynaptic neuron spikes, vesicles of neurotransmitter are released, and when the postsynaptic neuron's receptors receive neurotransmitter, a certain amount of current is imparted in the postsynaptic cell.

Synapses are highly heterogeneous. Presynaptically and postsynaptically, the number and type of neurotransmitters and receptors vary by neuron type, and by specific cells within each neuron type. Because of this, a spike in a presynaptic neuron can impart almost no current in the postsynaptic neuron, or it can impart an amount of current sufficient to make the postsynaptic cell spike immediately. The amount of current imparted to the postsynaptic cell when a presynaptic cell spikes is known as the *strength* of that synapse. Another interpretation of synaptic strength is the amount of influence that the presynaptic neuron has on the postsynaptic neuron. Yet another interpretation is that synaptic strength represents the efficiency through which the synapse carries information from the presynaptic neuron to the postsynaptic neuron.

Learning in the brain is thought to lie primarily in synaptic strength changes over time. The ability for the brain to change synaptic strengths over time is known as *synaptic plasticity*. Experimental neuroscientists interested in learning investigate the exact conditions under which synaptic strength changes occur. There are a number of leading theories, some of which have been expressed mathematically in the form of *learning rules*.

The remainder of this chapter examines the history of the exploration of synaptic strength changes in the brain, paying particular attention to theories that are still widely accepted. Theoretical models based on these experimental explorations follow in subsequent chapters.

2.1 Hebbian learning

One of the earliest hypotheses on when synaptic strength changes might occur came from Donald Hebb, who stated in his *The Organization of Behavior* [84],

> "When an axon of cell A is near enough to excite 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."

The simplest mathematical way of phrasing this notion is

$$\Delta\omega_{ij} = \kappa a_i a_j, \tag{2.1}$$

where

- ω_{ij} is the synaptic strength of the synapse between the axon of presynaptic neuron i and the dendrite of postsynaptic neuron j,

- $\Delta\omega_{ij}$ is the change in synaptic strength ω_{ij},
- κ is the learning rate,
- i indexes a particular presynaptic neuron (which is usually part of a population),
- j indexes a particular postsynaptic neuron,
- a_i is the amount of activity in the presynaptic neuron i, and
- a_j is the amount of activity in the postsynaptic neuron j.

"Activity" can be interpreted many different ways. The most common measure of activity is the firing rate of a neuron. However, membrane voltage, spike times, the amount of current flowing into the cell, filtered spike trains, and other measures can be interpreted as activity. Unless otherwise specified, activity is taken to be the firing rate of a neuron.

This rule forms the basis of most of what is studied in biologically plausible plasticity research to this day; most learning rules in some way use the presynaptic and postsynaptic activity along with other local variables, making them *Hebbian* (though non-Hebbian learning rules also exist; see section 2.2).

An important observation about this rule is that it is unstable. Stability, when discussing learning rules, means that the learning rule will result in a constant distribution of synaptic strengths for synapses attached to a cell. If the synapses are only *potentiated* – strengthened – then all of the synapses will reach some maximum strength (or in mathematical models, go to infinity). If the synapses are only *depressed* – weakened – then all of the synapses will reach some minimum strength or get pruned (or in some mathematical models, go to negative infinity). The original Hebbian learning rule, equation (2.1), can only result in potentiation, so it is unstable.

One way to address the instability of learning rule (2.1) is to use the derivatives of the pre- and postsynaptic activities (see [84, 197, 111, 108]). This is referred to as differential Hebbian learning, and can be expressed as

$$\Delta\omega_{ij} = \kappa \dot{a}_i \dot{a}_j. \tag{2.2}$$

This rule, while clearly different from rule (2.1), still holds to Hebb's original idea. Essentially, the rule makes the additional assumption that there is some kind of normal activity between these two neurons – some average state that is uninteresting from a learning perspective. Any deviations from that average state are reinforced in a Hebbian way, potentiating synapses between neurons that become more excited, and depressing synapses between neurons that become less excited. However, Kosko's mathematical analysis of the rule showed that, despite this rule allowing for negative weight changes (unlike typical Hebbian learning), weights will generally increase over time, so this form of differential Hebbian learning is also unstable [111].

A final permutation of this idea is a type of differential Hebbian learning that considers the product of presynaptic activity and the derivative of postsynaptic activity [109].

$$\Delta\omega_{ij} = \kappa a_i \dot{a}_j \tag{2.3}$$

This expression of Hebb's original postulate maintains the original idea, but has been shown to be stable, and has analogies to spike-timing dependent plasticity (see [132, 170, 173, 174] and section 2.1.3) and temporal-difference learning (see [109] and section 5.1.4).

An important observation that recurs throughout this thesis is that in Hebbian learning (both differential and non-differential), the only variables affecting the modulation of synaptic strength are those that are locally available to the synapse. While this seems straightforward from a biological perspective, it presents many difficulties for learning at higher levels. This constraint will be referred to as the *locality* constraint.

2.1.1 Long-term potentiation (LTP)

Despite strong arguments for Hebbian learning and successful applications to artificial neural networks, experimental evidence for this relationship did not come until 1966 when long-term potentiation (LTP) was discovered in the rabbit hippocampus by Terje Lømo and published by Bliss and Lømo in 1973 [125, 2, 26]. In their experiment, they strongly stimulated the perforant pathway of the hippocampus for several seconds and examined how a population of downstream neurons responded before and after the strong stimulation. Downstream neuron response was measured by the amplitude of the excitatory postsynaptic potential (EPSP) of the population, which correlates with the amount of current that is imparted into postsynaptic neurons by presynaptic neurons. An increase in EPSP means that each presynaptic spike is imparting more current to the postsynaptic neuron, or in other words, has more influence on whether the postsynaptic neuron fires.

In their experiments, they found that short bursts of very strong stimulation (referred to as *tetanic* stimulation) resulted in an increase in EPSP that persisted for long periods of time. The increase was measured over the period of several hours, though it is possible the changes persisted longer than that. Figure 2.3 from [26] summarizes their results.

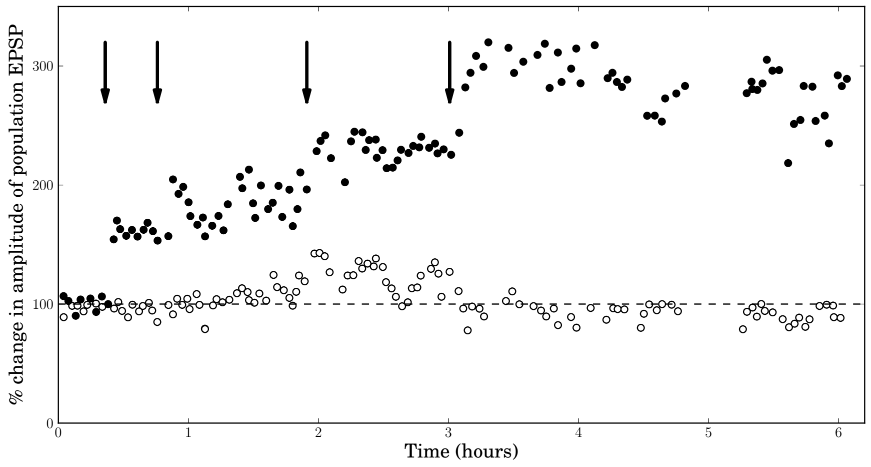


Figure 2.3: First evidence of LTP, from [26]. Black dots represent the EPSP of neurons in the stimulated pathway, white dots represent the EPSP in the unstimulated pathway. Tetanic stimulation was delivered at each arrow.

This initial result has been replicated many times in many different animals and in different areas of those animals' brains (for reviews see [27] and [131]). Improvements in neural recording devices have illuminated many features of LTP. One of the most important observations, seen shortly after the Bliss and Lømo's initial paper describing LTP, is that the amount of potentiation increases if more presynaptic cells are taking part in stimulating the postsynaptic cell, suggesting that these inputs cooperate [117, 120, 139]. This is known as *associative LTP* (see figure 2.4).

Many predicted that associative LTP could be explained by additional postsynaptic activity as a result of increased input. This was confirmed, and it was shown that the high-frequency tetanic stimulation that was used in early LTP experiments was not necessary, and instead LTP could be induced by pairing presynaptic spikes with postsynaptic spikes [76, 77, 101].

These findings contrasted earlier experiments that claimed that postsynaptic activity was not necessary for the induction of LTP [117]; further experiments revealed that it was not the postsynaptic spikes that were necessary, but some consequence of the postsynaptic spike, specifically an influx of calcium [78, 101]. Another known consequence of an action potential in some neuron types is propagation of the action potential back into the dendritic tree, a phenomenon known as a *back-propagating action potential* [61, 83, 164, 193, 202].

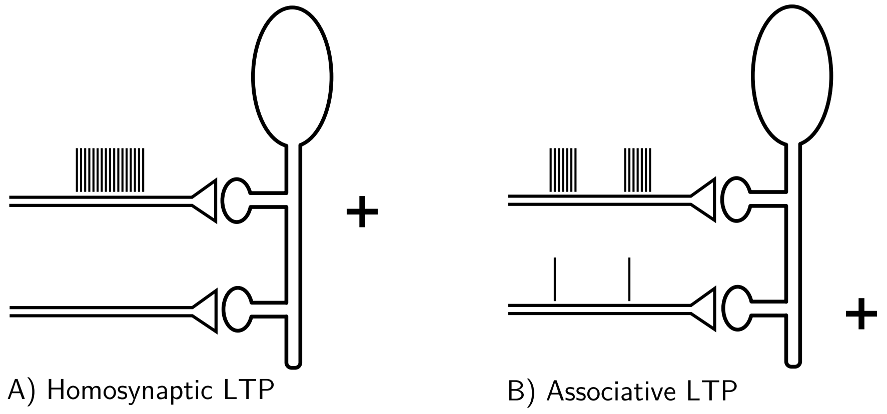


Figure 2.4: Cartoon depiction of the two classical methods of inducing LTP, recreated from [121]. Vertical lines do *not* depict spikes, but pulses applied to the presynaptic cell. + marks show which synapse is potentiated. (A) Homosynaptic (synapse-specific) LTP is induced by high-frequency tetanic stimulus (usually 100Hz for 1 second) of the presynaptic cell. (B) Associative LTP is induced by pairing a tetanic stimulus in one or more presynaptic cells with a low-frequency (usually 5Hz) stimulus in the presynaptic cell whose synapse is to be potentiated. Note that typically the synapses stimulated with the tetanic stimulation will also be potentiated.

A back-propagating action potential allows calcium to enter the cell and the dendrites through NMDA receptors and other sources [145], a process which is now widely believed to be necessary for LTP [45, 113, 129, 217], though recent experiments have suggested that local synaptic depolarization and dendritic spikes may be much more influential on LTP than back-propagating action potentials [33, 82].

Despite decades of LTP research and thousands of publications, there are still many open questions in the field. The varying and sometimes contradictory findings suggest that there are a number of mechanisms that result in the long-term potentiation of excitatory synapses [131], making it difficult to build general models that can be applied to the various parts of the brain where LTP has been shown to take place.

Regardless, many of the mechanisms that are termed LTP provide experimental credence behind the main idea of Hebb's postulate, that a neuron that fires at the same time as a neuron it projects to will have its connection to that neuron strengthened. However, it faces the same problem as the original Hebbian learning rule (2.1): it can only potentiate. Natural decay of synapses over several days can theoretically enable networks to arrive at any synaptic configuration through LTP alone [41]; however, the existence of a depression mechanism would make learning much faster and easier [121].

2.1.2 Long-term depression (LTD)

Long-term depression (LTD) is the depression mechanism that works opposite LTP. While LTP causes increases in synaptic strength that persist over days and months, LTD causes decreases in synaptic strength that persist on the same time scale. Strangely, the mechanisms that give rise to certain forms of LTD appear to be very similar to those of LTP.

The first evidence of LTD was discovered almost a decade after LTP, and published by Lynch et al. in 1977 [57, 126]. They found that after giving a tetanic stimulation to some neurons – which induced LTP – other afferents to the same neurons had weaker EPSPs, suggesting that their synaptic strength had decreased. This type of LTD, termed *heterosynaptic LTD* (see figure 2.5), is interesting because it does not require presynaptic activity in the synapse that is being depressed; it, therefore, is not predicted by the classic Hebbian formulations (see section 2.1), and is sometimes referred to as 'anti-Hebbian' [136].

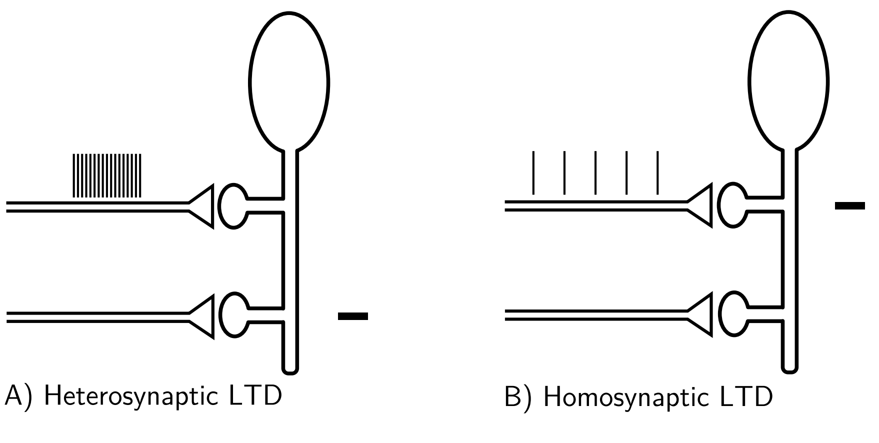


Figure 2.5: Cartoon depiction of the two classical methods of inducing LTD, recreated from [121]. Vertical lines do *not* depict spikes, but pulses applied to the presynaptic cell. - marks show which synapse is depressed. (A) Heterosynaptic LTD is induced with tetanic stimulation in some presynaptic cells; those that are not stimulated may become depressed. Note that the stimulated cells often have their synapses potentiated. (B) Homosynaptic LTD is induced with long period of low-frequency stimulation (typically 1 Hz for 10 minutes) of the presynaptic cell.

Evidence for depression that follows the Hebbian formulation – that is, depends on presynaptic activity – was first seen by Dunwiddie and Lynch in 1978 [57], but it was not well understood until two convincing experiments in 1992 [56, 144]. It was found that LTD could be induced by periods of low-frequency (0.5 to 5Hz) stimulation of a presynaptic cell. This type of LTD is known as *homosynaptic LTD* (i.e. synapse-specific LTD; see figure 2.5). Despite the fact that it does the opposite of LTP, the induction mechanism is the same: paired presynaptic and postsynaptic firing, only with lower frequency.

Hypothesizing that the mechanisms underlying both types of plasticity may also be the same, experiments were performed that blocked calcium influx through NMDA receptors [56, 144], and it was shown that LTD was blocked, meaning that calcium influx is necessary for the induction of LTD. Not only is LTD thought to be a calcium dependent process, it has also been shown that back-propagating action potentials and dendritic spiking have a strong influence on the induction of LTD, like LTP [44].

Other forms of LTD exist in many different areas of the brain (the cerebellum, for example [90, 121]; see [136] for a review) but we focus on the well characterized calcium-dependent forms of LTP and LTD in this thesis. While we do not model these processes at a biophysical level (i.e., with ion flux equations), the mechanisms that produce these types of synaptic plasticity are useful to keep in mind when creating models that reproduce, at a phenomenological level, synaptic plasticity in the biological brain.

2.1.3 Spike-timing dependent plasticity (STDP)

Recall that Hebb's original postulate predicts potentiation when a cell "repeatedly or persistently takes part in firing" another cell. This implies that there should be high correlation between the spike times of pre- and postsynaptic neurons, and raises the question of how highly correlated the spike times need to be. More generally, the experimental question to explore is how much influence the time difference between a pre- and postsynaptic spike has on the amount of synaptic potentiation or depression.

The first experiment examining the temporal requirements of synaptic strength changes was done by Levy and Steward in 1983 in the hippocampus [119]. Despite technological limitations, they were able to show that LTP was induced most strongly when pre- and postsynaptic stimulation occur at the same time, LTP was induced weakly when presynaptic stimulation precedes postsynaptic, and LTD can be induced if postsynaptic stimulation precedes presynaptic stimulation (see figure 2.6). This finding was surprising, and not predicted by Hebb's original postulate.

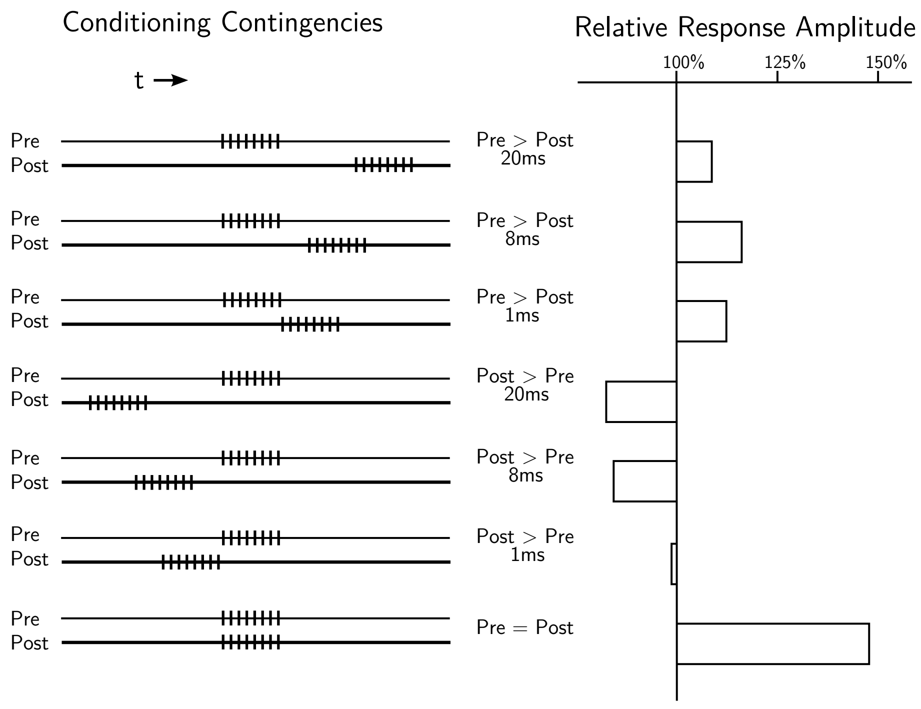


Figure 2.6: Evidence that the temporal order of pre- and postsynaptic stimulation affects the induction of LTP/LTD, recreated from [119]. (Left) The stimulation protocol. Each vertical line represents a pulse of current. (Right) The ratio of the amplitude of the EPSP before the stimulation protocol and 20 minutes after the stimulation protocol. Note that depression happens when postsynaptic neurons are stimulated before presynaptic neurons, potentiation when presynaptic neurons are stimulated before postsynaptic neurons, and strong potentiation occurs when they are simultaneously stimulated.

Markram et al. were the first to describe the temporal relationship that is now referred to as spike-timing dependent plasticty (STDP) [132, 183]: when a presynaptic action potential is followed by a postsynaptic action potential there is potentiation in the synapse (*pre-post* pairing), and when a postsynaptic action potential is followed by a presynaptic action potential there is depression in the synapse (*post-pre* pairing). Markram found this effect in neocortex when the action potentials were offset by 10 ms in either direction. Bell et al. showed the same effect in the cerebellum, and showed that a shorter offset between the spikes produced stronger potentiation and depression [17]. Bi and Poo further explored the relationship between the length of the offset and the amount of synaptic strength change, producing figure 2.7, the now stereotypical “STDP curve” [21], which demonstrates that the critical window for large strength changes is around 20 ms, and the sign of the strength change depends on which spike is first.

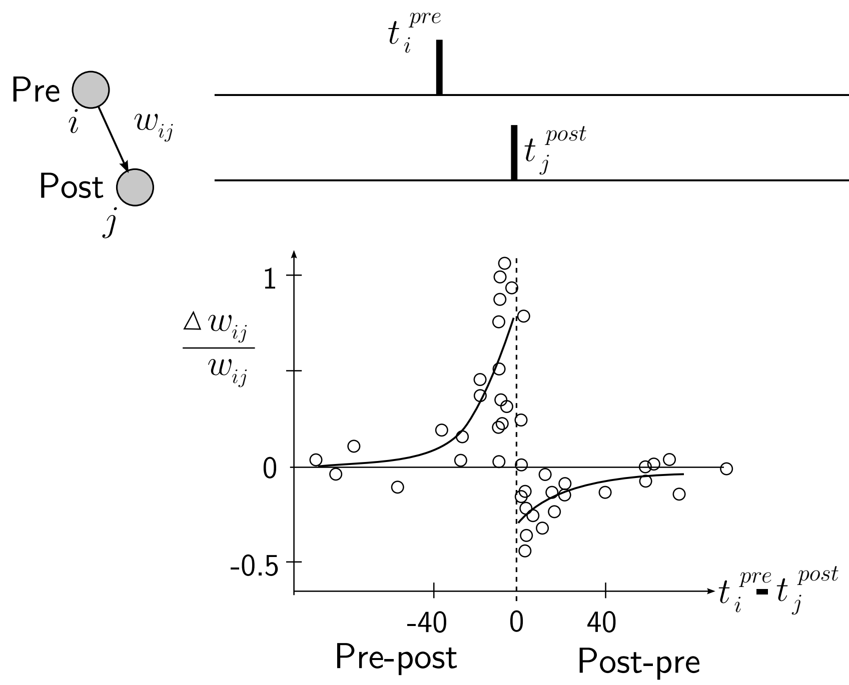


Figure 2.7: The STDP curve, discovered and first plotted in [21]. Each dot represents the relative change in synaptic strength after 60 pre-post or post-pre spike pairings. Recreated from [183].

Many experiments followed Bi and Poo's, and found that a similar STDP curve could be found in the synapses of several other systems (e.g. frog retino-tectal synapses [222]). However, other experiments discovered STDP curves with wildly different shapes, including some that only saw depression regardless of temporal order (see [1, 22, 38, 185] for reviews, and figure 2.8). This highlights the heterogeneity of synapses, and lends credence to the idea that timing matters; different STDP curves may come as a result of the biophysical properties of the cells or synapses, which may be a reflection of the functions performed by those cells. To complicate matters further, recent experiments have shown that the synapse's spatial location on a dendritic tree can have a profound effect on the STDP curve, to the point that "typical" STDP is seen in synapses close (proximal) to the soma of the cell, and a flipped STDP curve is seen in synapses far (distal) from the soma of the cell in layer-5 neurons in the neocortex of rats [118].

It should be unsurprising that STDP, like LTP and LTD, is critically dependent on

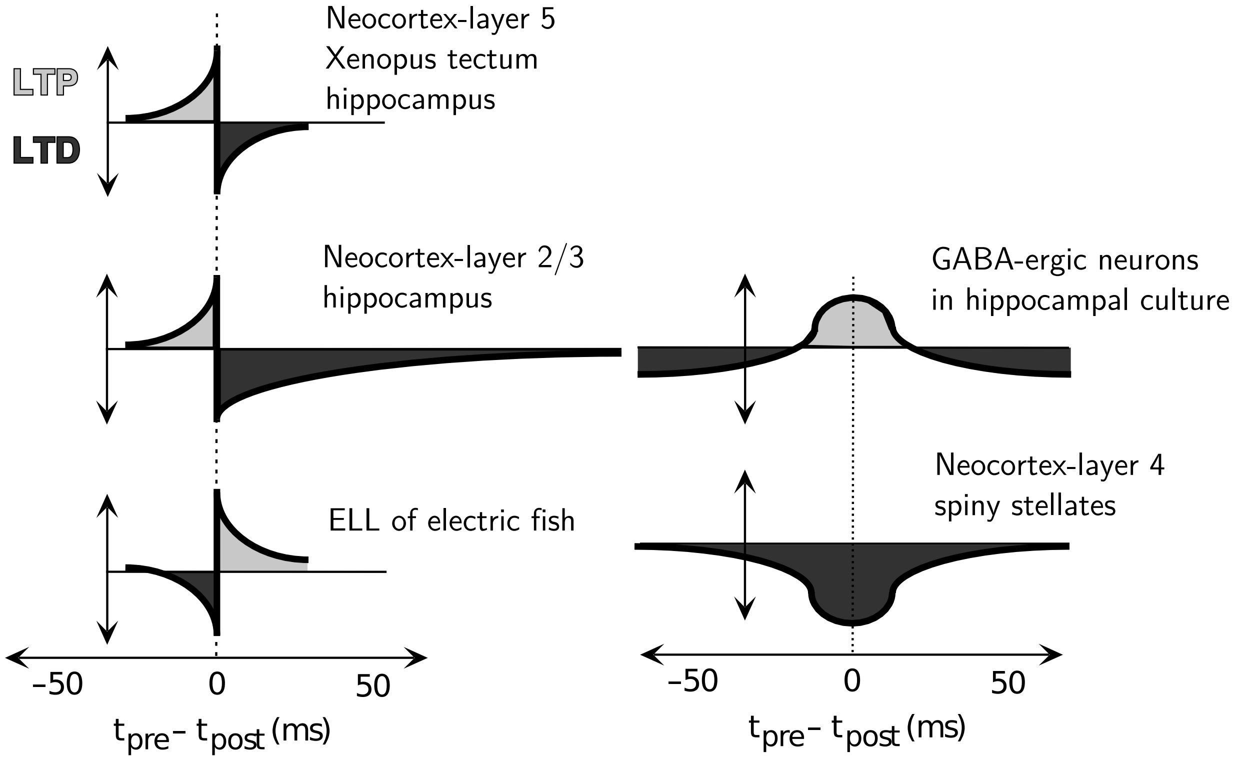


Figure 2.8: Five different STDP curves, showing the diversity of STDP in different synapses. Recreated from [1].

calcium levels in the synapse [21, 123, 147, 148, 184]; the stimulus protocols that produce LTP, discussed previously, could be explained with STDP, as potentiation is strongest when presynaptic and postsynaptic spiking occurs at the same time. Blocking calcium influx through NMDA receptors blocks this kind of LTP [78], and the introduction of a calcium chelator around the synapse blocks both LTP and LTD when using STDP protocols [67, 184, 185]. As discussed in section 2.1.1, backpropagating action potentials affect the calcium levels in the dendritic tree, and as such can contribute to LTP and LTD [87], though more recent evidence suggests that it is local dendritic spikes and other local dendritic activity that is critical for inducing synaptic strength changes when using STDP protocols [71, 122].

This begs the question of whether STDP is a separate plasticity mechanism in the brain, or just an experimental protocol that can be used to induce LTP and LTD. There is currently no definitive experiment that can answer that question. The similarity in the biophysical mechanisms suggests that they are two methods of explaining the same phenomenon; further, the cooperativity requirement of LTP, which is not predicted by the classical STDP curve, is still present when using STDP protocols: many presynaptic cells must be coactive before LTP is induced [186].

One feature in favour of further study into STDP as a plasticity mechanism is that it is more biologically plausible than traditional LTP/LTD mechanisms, in the sense that it does not require the unrealistic stimulus protocols that earlier LTP/LTD studies utilized [94, 140]. However, one feature of STDP that is sometimes overlooked in the discussion of biological plausibility is that STDP exhibits frequency dependence. Typical STDP protocols induce spiking at around 10Hz, which can induce either LTP or LTD; however, Sjöström et al. found that spike trains lower than 10Hz could not induce potentiation, and spike trains higher than 40Hz could not induce depression [122, 186] (see figure 2.9). Further, Wang et al. showed in 2005 that looking at triplets of spikes (i.e. pre-post-pre and post-pre-post relationships instead of just pre-post and post-pre) and quadruplets produced synaptic strength changes that were not be predicted by pair-based STDP models [216]. Incorporating these findings into a triplet-based rule, however, not only recreated Wang et al.'s findings (for triplets and quadruplets), but also the frequency dependence shown by Sjöström et al. (see [158] and section 6.2.3).

The general consensus in the experimental community is that STDP is a convenient experimental protocol that robustly induces LTP and LTD and is not a separate plasticity mechanism [183]. We explore this question theoretically in section 6.2.4.

2.2 Non-Hebbian plasticity

While Hebbian plasticity began as the original postulate expressed by the simple learning rules (2.1), (2.2) and (2.3), in contemporary literature, it generally refers to any change in synaptic strength that is synapse-specific and depends on correlated presynaptic and postsynaptic activity. By that definition, heterosynaptic LTD is non-Hebbian (see section 2.1.2 and figure 2.5), though there are synapse-specific models that can exhibit heterosynaptic LTD (see section 6.2.1). There are also experimental protocols that can induce LTP in non-Hebbian ways: high-frequency presynaptic stimulation without postsynaptic spiking causes LTP in the mossy fiber - hippocampus area CA3 synapse [210] and cortico-striatal synapses [62]; high-frequency postsynaptic stimulation of hippocampus area CA1 without presynaptic spiking causes LTP cell-wide (i.e. in all afferent synapses) [100].

Another non-Hebbian form of plasticity is synaptic scaling, in which the strength of a synapse changes relative to other synapses on the postsynaptic cell such that the postsynaptic cell's firing rate remains at some target value. While hypothesized to exist in

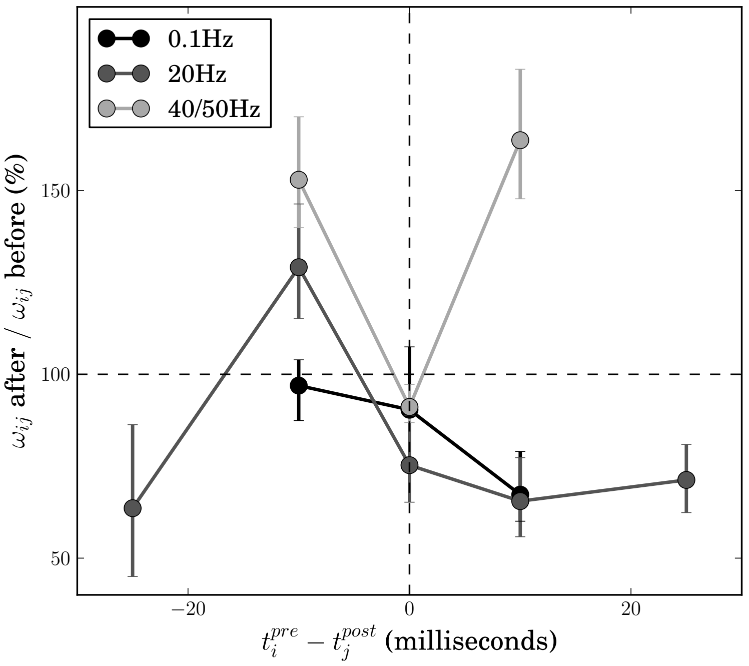


Figure 2.9: Frequency dependence of the STDP protocol, from [186]. The three STDP curves are all from layer-5 pyramidal neurons in visual cortex. The axes are the same as previous STDP curves.

theoretical models due to its inherent stability, synaptic scaling was experimentally confirmed in 1998 in cultured networks of neocortical [209], hippocampal [124] and spinal-cord neurons [150]. Early theories of the biophysical mechanisms for synaptic scaling suggested a change in the number of glutamate (both AMPA and NMDA) receptors in inverse proportion to postsynaptic activity [150, 218]. More recent reports have implicated a wide variety of factors, such as activity-dependent release of brain-derived neurotrophic factor (BDNF) and somatic calcium levels (see [208] for a review).

2.3 Dopamine modulated plasticity

The previously discussed synaptic plasticity mechanisms are typically studied in controlled conditions, in which the spiking behaviour of the presynaptic and postsynaptic cells are the only variable. The synapses being investigated are usually excitatory synapses that use glutamate as their primary neurotransmitter, though inhibitory synapses that use GABA as their primary neurotransmitter can also be modified by similar protocols [40, 79]. However, the neurotransmitter dopamine has been found to modulate synaptic plasticity dramatically. Dopamine has been shown to be important for reinforcement learning, which is discussed in chapter 5. In this section we review dopamine's effect on synaptic plasticity.

The effect of dopamine is not consistent across brain areas. Each dopamine receptor type (there are five, D1 through D5) can contribute to synaptic plasticity in different ways. The striatum has been widely studied, as it receives strong projections from the midbrain dopamine system. Calabresi et al. found that mice lacking D2 receptors in the striatum experienced LTP when the same stimulus produced LTD in mice with D2 receptors [37]. In a later study, they found that blocking D1 and D5 in the mouse striatum prevented the induction of both LTP and LTD ([35, 36], results replicated by [102, 199]). Pawlak and Kerr confirmed that D1 and D5 are required for LTP/LTD in rat striatum, and showed that D2 receptor activation delays LTD and speeds induction of LTP [156]. In the CA1 region of the hippocampus of mice, D1 and D5 receptors appear to be necessary for the maintenance of LTP but not necessarily the induction, while D2 agonists had no effect [95, 154]. In the dentate gyrus of the hippocampus, a D1/D5 receptor antagonist had no effect on either the induction or maintenance of LTP [200]. In the prefrontal cortex of rats, D1 receptor activation modulates LTP proportionally, while D2 receptor activation has no effect on LTP [73] (for a review of dopamine-dependent synaptic plasticity, see [95]).

Despite the varying effects of dopamine, its role in synaptic plasticity has led to an extension of traditional Hebbian plasticity rules: *three-factor* plasticity rules [149, 168, 219]. These are learning rules that use presynaptic activity, postsynaptic activity, and phasic dopamine levels (D_p) to determine the amount and direction of synaptic strength change.

$$\Delta\omega_{ij} = \kappa a_i a_j D_p \tag{2.4}$$

The multiplicative dopamine level term has the effect of gating a Hebbian learning rule, allowing plasticity to occur only when there is a phasic change in dopamine levels, which is consistent with some of the experiments described in this section.

2.4 Explaining behaviour with synaptic strengths

While the biological literature examined above identifies some situations in which synaptic strength changes occur in the brain, it does not directly address the important questions about how animals learn and remember. Theoretical discourse on the role of synaptic plasticity in learning and memory usually focuses on whether memories can be encoded and retrieved using synaptic plasticity. Stevens posed this “million dollar question” to neuroscientists in 1998 and found consensus neither for nor against [190]. Martin et al. examined this more formally in 2000 [133]. Their review was motivated by a succinct formulation of the synaptic plasticity and memory hypothesis.

> “Activity-dependent synaptic plasticity is induced at appropriate synapses during memory formation, and is both necessary and sufficient for the information storage underlying the type of memory mediated by the brain area in which that plasticity is observed.”

Their conclusion was that synaptic plasticity is necessary for memory, but little evidence exists to argue for sufficiency. We aim in this thesis to give support for the sufficiency of synaptic plasticity to learning and memory by creating biologically plausible spiking neural network models that learn to perform certain functions through synaptic weight changes. This work, it should be noted, says little about necessity, as these models are not the only ones that could be used to learn these behaviours.

Chapter 3

Large-scale neural modelling

At the beginning of chapter 2, we provided a sketch of how biological neurons operate. This chapter fills in the details of that sketch and examines computational models that emulate the biological neuron. As well, the Neural Engineering Framework [59] will be proposed as a method of determining how to connect those single-neuron models together in a way that can explain high-level behaviour while maintaining biological plausibility.

3.1 Single-neuron models

Single-neuron modelling, like the modelling of synaptic plasticity, can be split into two broad categories: detailed biophysical models that aim to emulate nature with fine granularity, and simple phenomenological models that aim to emulate the *results* of such fine-grained processes using minimal computational resources.

On one end of the scale, the most detailed biophysical models are based on the Hodgkin-Huxley model [86]. These types of models define differential equations for a number of biophysical parameters, such as the current flow through each type of ion channel. They model biological neurons very accurately, but at the cost of computational efficiency; simulating one Hodgin-Huxley neuron for 1 ms involves over 1000 computations. Given that we wish to scale up to large-scale networks of thousands of neurons, these biophysical models are too computationally intensive.

On the other end of the scale is the leaky integrate-and-fire model, which captures many important features of biological neurons [97] in a simple mathematical model that takes less than 10 computations for 1 ms of simulation. We use this model exclusively for the simulations discussed in the remainder of this thesis.

Between these two representative models there is a continuum of single-neuron models, each of which offers a different balance between biological realism and computational efficiency. Izhikevich provides a review of these models [91].

3.1.1 Leaky integrate-and-fire model

In chapter 2 we introduced the neuron as an input/output device, noting that it receives input from its dendrites, performs some non-linear function on that input, and outputs action potentials through its axon. It should be noted that there are many different neuron types that all respond differently to input. A well understood and common neuron type is a neocortical pyramidal cell, seen widely in mammalian prefrontal cortex. The leaky integrate-and-fire (LIF) model has been shown to model well the neocortical pyramidal cell [167].

Input, as discussed before, comes from dendrites. The input is measured as current; when neurotransmitter is released from a presynaptic cell and binds to a postsynaptic cell, some amount of current is imparted. The weighted sum of this dendritic input plus some background bias current, at some point in time, t, is denoted as $J(t)$.

Output is delivered in the form of action potentials. Action potentials are dramatic changes in the membrane potential (i.e. the voltage) of the cell. The behaviour of interest in a single-neuron model, especially a phenomenological model like the leaky integrate-and-fire (LIF) neuron, is at what times action potentials occur; to find this, we explicitly model the voltage changes over time with the equation

$$\frac{dV}{dt} = -\frac{1}{RC}\left(V(t) - J(t)R\right),$$

where R is the resistance of the membrane, C is the membrane capacitance, and $V(t)$ is the voltage at time t.

This equation may be recognized as the linear differential equation for RC circuits; indeed, a neuron can be idealized as an RC circuit, with the bilipid membrane of the cell acting as a capacitor accumulating electric charge from the dendrites, and the ion channels acting as resistors (see figure 3.1). Because the bilipid membrane is not a perfect insulator, current "leaks" out of the cell at a speed defined by the membrane resistance and voltage, hence the "leaky" moniker. See figure 3.2 for a circuit diagram depicting how the LIF neuron works.

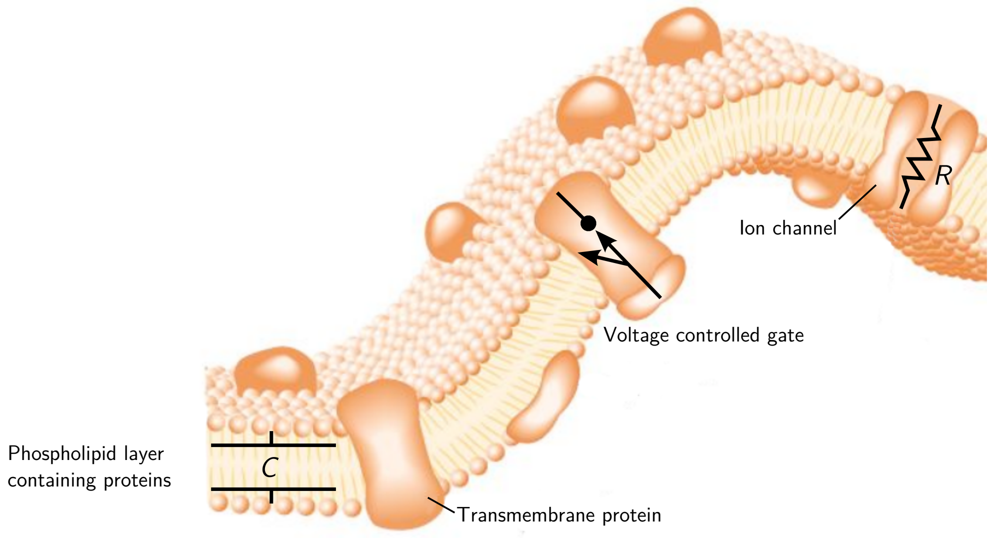


Figure 3.1: Illustration of the bilipid membrane of a neuron. Labelled items refer to elements of the circuit diagram, figure 3.2. Recreated from [58].

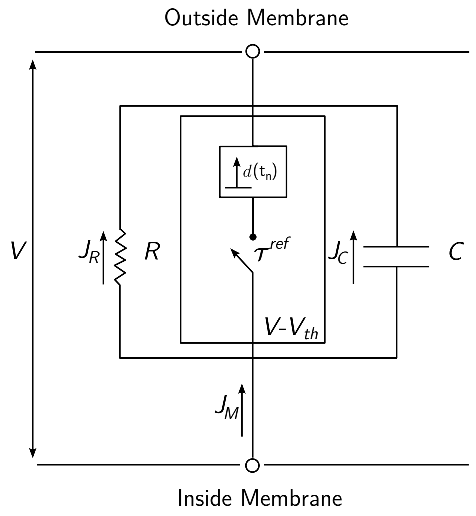


Figure 3.2: Circuit diagram that corresponds to the leaky integrate-and-fire (LIF) neuron. Recreated from [58].

Unlike a typical RC circuit there are two non-linearities in the LIF model. First, once $V(t)$ reaches a certain threshold, V^{th}, the model "spikes." The time of the spike, t_n, is recorded, and modelled as a Dirac delta function, $\delta(t - t_n)$. The membrane voltage is then reset to 0. Note that this means that the time of a spike matters, but the super-threshold behaviour does not; in reality, action potentials can vary in duration, amplitude and shape, but it is generally accepted in the neural coding field that little information is encoded in these action potential features [204].

The second non-linearity, which is not present in all LIF models, is a refractory period. When a biological neuron spikes, there is a small amount of time, usually around 1 ms [171], during which no amount of injected current will raise the membrane voltage. We call the length of the refractory period τ_{ref} and force $V(t) = 0$ during the refractory period, after which the neuron resumes accumulating charge (see figure 3.3).

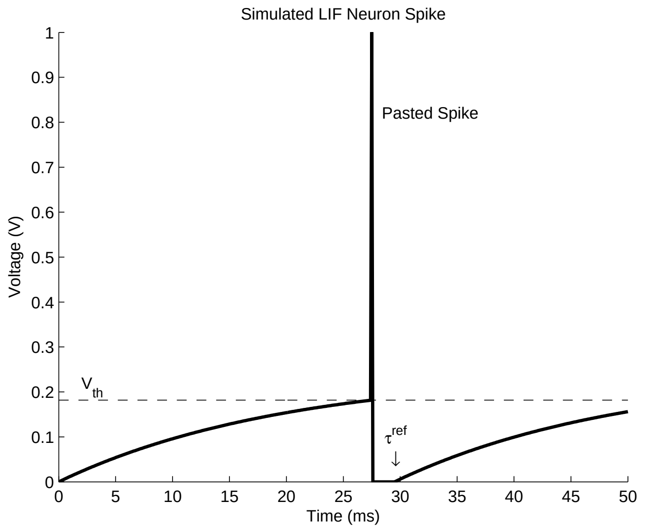


Figure 3.3: Membrane voltage of a LIF neuron with constant input J, from [42].

LIF neurons can be connected naïvely, or to match the statistics of biological neural networks (e.g. 10% all-to-all connectivity). However, connecting millions of these models together tells us little about what the brain is doing – what information is being encoded, and how is that information processed by the various neural circuits in the brain? We consider these questions fundamental, and use Eliasmith and Anderson's Neural Engineering Framework as a means of tackling them [59].

3.2 The Neural Engineering Framework

The Neural Engineering Framework (NEF) is a set of principles that can be used to build large-scale networks of single-neuron models. It enforces biological constraints, such as the considerable amount of noise in the brain, and the heterogeneity of neurons and synapses. Within those constraints, it defines an encoding / decoding scheme that can be used to represent n-dimensional vectors in neural populations, and an analytical method of creating connection weight matrices (i.e. synaptic strength matrices) such that encoded vectors can be transformed by arbitrary nonlinear functions.

The three principles of the NEF are as follows.

1. "Neural representations are defined by the combination of nonlinear encoding and weighted linear decoding."
2. "Transformations of neural representations are functions of variables that are represented by neural populations. Transformations are determined by using an alternately weighted linear decoding."
3. "Neural dynamics are characterized by considering neural representations as control theoretic state variables. Thus, the dynamics of neurobiological systems can be analyzed using control theory."

Principle 1, representation, and principle 2, transformation, are especially relevant to the discussion of learning in large-scale networks. Principle 3 will not be discussed in this thesis; details on it can be found in [59].

3.2.1 Representation

Representation in biological neural networks is a rich area of research in its own right. The NEF's notion of representation has the most in common with the idea of *population coding*, first proposed by Georgopoulos et al. based on experiments in monkey cortex [69].

The NEF makes the assumption that information is encoded in the brain through the spiking patterns of a population of neurons. In other words, representation is a *process*. This process allows the NEF to be employed at multiple levels: the **vector** level, where we are concerned with a set of numerical values, and the **spiking** level, where we are concerned with the firing patterns of a population of neurons.

Representation is important to the study of learning in large-scale spiking networks by giving us a method of relating the spiking behaviour of neurons to more traditional statistical and mathematical analysis methods. To understand representation, the NEF defines a method of **encoding** information at the vector level using spikes, and a method of **decoding** spikes into information at the vector level.

Encoding

From our discussion of single-neuron models, we know that neurons receive input in the form of current, J. Neurons output spikes based on some non-linear function, which we will denote $G[\cdot]$. The activity of a neuron, then, is completely determined by the non-linearity and the input current.

$$a(J) = G[J] \tag{3.1}$$

In our case, the $G[\cdot]$ nonlinearity is defined by the LIF neuron model. For a sensory neuron, the NEF assumes that the input current J is proportional to whatever external signal that the neuron is sensitive to; for example, in the visual system, some neuron may be sensitive to brightness. The brighter the visual field, the more current is injected into the neuron. Other brightness-sensitive neurons will respond differently to the level of brightness; the NEF models these differences with the equation

$$J(x) = \alpha x + J^{bias}, \tag{3.2}$$

where x is the external signal, e.g. brightness, α is a positive scaling factor representing how sensitive the neuron is to x, and J^{bias} is some amount of current that is injected in the cell regardless of any external stimulus; J^{bias} explains the background firing rate of neurons receiving no stimuli.

Both α and J^{bias} vary from neuron to neuron. In an ideal world, we would be able to measure α and J^{bias} for each neuron when modelling a particular system of neurons. This is not feasible, so instead we generate random α and J^{bias} values within a range that is plausible for the system being modelled.

With the input current, we can then use any neuron model to determine the membrane voltage, and record when spikes happen. In our brightness example, it's clear that a population of neurons will not spike very much in the dark, spike a little bit with moderate light, and spike a lot when it's very bright. As it turns out, there are also neurons that are sensitive to the opposite stimulus (darkness); this turns out to produce a representation of brightness that is easier to decode. We can then say that some neurons encode the brightness of the visual field, and some neurons encode the darkness of the visual field. We represent this differential preference with a new variable called an *encoder*, e, which is 1 for brightness and -1 for darkness. The current can then be determined by the equation

$$J(x) = \alpha e x + J^{bias}. \tag{3.3}$$

Again, we randomly choose the encoder to be either 1 or -1, unless we have some reason to believe that the system being modelled preferentially encodes for one direction over another. Note that it is not necessary to choose e values between 1 and -1, because α defines the scale of the sensitivity, while e is simply direction.

This suffices for the simple 1-dimensional example of brightness, but brains operate in an extremely high-dimensional world. As a slightly more complicated example, let us consider a population of neurons that represents the animal's eye position, which can be encoded in two dimensions, the horizontal and vertical position of the eye. Both the external signal and the encoder are now vectors; we determine the amount of current to inject into the cell by the dot product $(\cdot)$ of the external signal and the encoder; i.e.,

$$J(\mathbf{x}) = \alpha(\mathbf{e} \cdot \mathbf{x}) + J^{bias}. \tag{3.4}$$

We still want $\mathbf{e}$ to represent a direction only, so for each neuron, we select a random 2-dimensional unit vector for $\mathbf{e}$, unless we have some reason to believe that the neurons would be more likely to be sensitive to some directions than others.

As an aside, there is evidence that neurons in populations sensitive to high-dimensional spaces can be sensitive to only one dimension (e.g. horizontal eye position). The existence of these types of neurons (for example the now-famous "Jennifer Aniston neuron" [163]) have led to many debates about whether representation is local or distributed across many neurons; a recent paper by Stewart et al. argues that an encoding vector approach is more useful for understanding representation than local vs. distributed arguments [191].

Moving away from concrete examples, in general, we can use equation (3.4) to determine how much current we should inject into a neuron model for a vector of any length. Further, we can also represent a function by approximating that function with some finite-length vector, wherein each element of the vector is $f(x)$ for x discretized over some finite range.

Note that because we always choose $\mathbf{e}$ to be a unit vector, the dot product $\mathbf{e} \cdot \mathbf{x}$ is the scalar projection of $\mathbf{x}$ in the $\mathbf{e}$ direction, or mathematically

$$\mathbf{e} \cdot \mathbf{x} = |\mathbf{x}| \cos \theta$$

where θ is the angle between $\mathbf{e}$ and $\mathbf{x}$. This can be interpreted as meaning that the amount of current injected in the cell is proportional to the similarity between a neuron's encoding vector and the stimulus ($\cos\theta$).

A final amendment to (3.4) is to incorporate time-dependence. The input signal vector, $\mathbf{x}$, varies over time, and the amount of current injected will also vary. Our final equation is then

$$J(t) = \alpha(\mathbf{e} \cdot \mathbf{x}(t)) + J^{bias} \tag{3.5}$$

To summarize, the encoding procedure – the process that takes an n-dimensional vector and encodes it in the spiking activity of a population of neuron models – involves the usually-random selection of three properties for each model neuron:

1. α, a positive scalar representing the scaling factor,
2. J^{bias}, a scalar amount of background current, and
3. $\mathbf{e}$, a unit vector the same length as $\mathbf{x}$, representing the part of the input space the neuron is sensitive to.

Equation (3.5) then uses these three properties to determine the amount of current to inject into each cell, which causes neuron activity that depends on the neuron model used (i.e. equation (3.1)).

Decoding

Decoding is the opposing process to encoding: we map the spiking profile of a population of neurons to the space of n-dimensional vectors. We start this section with an analogous decoding process, converting from a binary number to a decimal number, and show how the same idea can be applied to neural populations.

Computers represent data in binary format, i.e. in base-2, with 0s and 1s as the only digits. An integer like 39 is encoded as 100111 in binary. To convert from the binary representation back to base-10 integers, we consider each digit as a power of two, increasing from right to left; i.e.

$$\begin{aligned} 100111 =& 1 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 + 0 \times 2^3 + 0 \times 2^4 + 1 \times 2^5 \\ =& 1 + 2 + 4 + 32 \\ =& 39 \end{aligned}$$

In general, if we have a series of binary digits, $b_{n\ldots0}$, representing a number encoded in binary, we can convert it to decimal with the formula

$$x_{10} = \sum_{i=0}^{n} 2^i b_i,$$

where x_{10} is the number in decimal. This equation is known, generally, as a weighted sum. Note that this a linear operation.

The process of decoding the spiking profile of a population of neurons can also be expressed as a weighted sum.

$$\hat{\mathbf{x}} = \sum_{i=0}^{n} \mathbf{d}_i a_i, \tag{3.6}$$

where $\hat{\mathbf{x}}$ is the decoded estimate of the encoded vector $\mathbf{x}$, $\mathbf{d}_i$ is a *decoding vector*, and a_i is the activity of neuron i.

To see how this simple equation can decode a value from the spiking activity of neurons, let us start again from the simple scalar case without considering time, and build up from there.

We can visualize the activity of a neuron, a_i, by plotting its *tuning curve*. A tuning curve is a plot of the firing rate of a neuron as a function of input current. See figure 3.4.

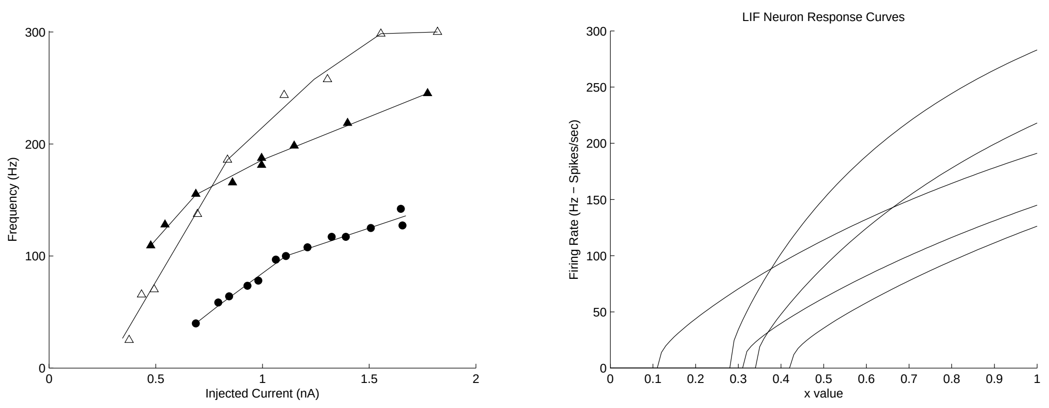


Figure 3.4: Example tuning curves. (Left) Experimentally determined tuning curves, from [137]. (Right) Similar tuning curves for LIF neurons, from [42].

Experimental tuning curves are found by injecting different levels of current into a cell and measuring the resulting firing rate of the neuron. We can do the same with our single-neuron models to plot tuning curves; however, the LIF neuron is simple enough that we solve for the firing rate of a LIF neuron given some amount of current.

$$a(J) = \begin{cases} \frac{1}{\tau^{ref} - \tau^{RC} ln\left(1 - \frac{J^{th}}{J}\right)}, & \text{if } J > J^{th} \\ 0 & \text{otherwise,} \end{cases} \tag{3.7}$$

where J^{th} is the current threshold above which the neuron will spike. The derivation of this formula can be found in [59].

It is usually more convenient to solve for the activity based on some arbitrary scalar stimulus x rather than the current; this requires a simple substitution of equation (3.2), giving as a final equation

$$a(x) = \begin{cases} \frac{1}{\tau^{ref} - \tau^{RC} ln\left(1 - \frac{J^{th}}{\alpha x + J^{bias}}\right)}, & \text{if } \alpha x + J^{bias} > J^{th} \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

It should be noted that the NEF neither requires nor favours the use of any particular single-neuron model; we are using the LIF neuron in this section because there is a simple formula for generating its tuning curve, but we could find the tuning curve for any neuron either by solving for $a(x)$ or by direct simulation using a sufficient discretization of a range of possible x values.

With the tuning curves of several neurons, we can visualize how, for any encoded value x, we can decode an estimate of x, $\hat{x}$, with the weighted sum in equation (3.6). Figure 3.5 shows this; it is important to note that the solid black line, representing $\hat{x}$, is calculated using the same decoding weights, $\mathbf{d}$, for all values of x.

Solving for these decoding weights is done through a least-squares minimization of the decoding error, $x - \hat{x}$, resulting in the following equations.

$$\begin{aligned} \mathbf{X} &= [-x, -x + dx, ..., -dx, 0, dx, ..., x - dx, x] \\ \mathbf{A} &= \begin{bmatrix} a_0(\mathbf{X}) \\ a_1(\mathbf{X}) \\ \vdots \\ a_n(\mathbf{X}) \end{bmatrix} \\ \mathbf{d} &= \mathbf{\Gamma^{-1}\Upsilon}, \quad \text{where } \mathbf{\Gamma} = \mathbf{AA^T} \quad \text{and } \mathbf{\Upsilon} = \mathbf{AX^T} \end{aligned} \tag{3.9}$$

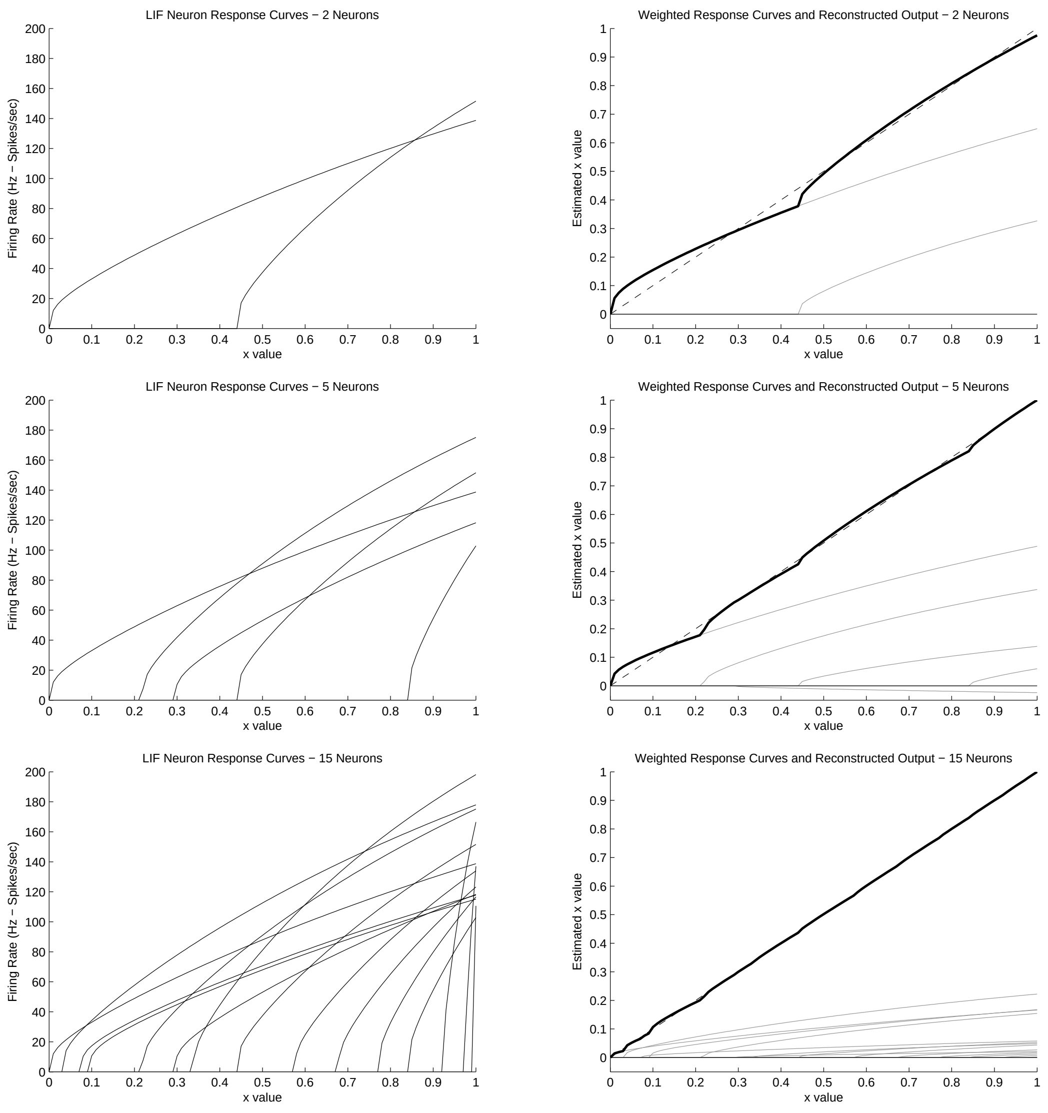


Figure 3.5: Illustration showing how the tuning curves of a population of LIF neurons can be linearly combined to estimate an input signal, x. From [42].

Note that the definition of $\mathbf{X}$ requires making explicit the range of values that the neural population is expected to encode. We then discretize that range into $\frac{2x}{dx}+1$ values; smaller dx values result in more accurate decoders, while larger dx values make the decoder calculation faster. We call $\mathbf{A}$ the *activity matrix*, and it is essentially a matrix containing the tuning curves of all of the neurons in the population. In this case, where we are considering only a scalar x, $\mathbf{d}$ is a vector of length n; each neuron's activity is weighted by a scalar decoder to calculate the decoded estimate. The full derivation of these equations can be found in [59].

Considering the vector case instead of the scalar case does not change equation (3.9), except that $\mathbf{X}$ changes from a vector of possible x values to a matrix of possible $\mathbf{x}$ vectors. This changes the dimensionality of some of the computations, with the end result being that $\mathbf{d}$ changes from a vector of decoding weights to a matrix of decoding vectors. This makes sense because our decoded value also needs to be a vector; note in equation (3.6) that $\hat{\mathbf{x}}$ and $\mathbf{d}_i$ are vectors of the same length.

Finally, we must also take into account time. Our notion of the activity of a neuron model changes. $a(\mathbf{x})$ is measured by the firing rate of the neuron; $a(t)$ is measured by a filtered spike train. Recall from section 3.1.1 that a spike is often modelled as a Dirac δ-function. Therefore,

$$a(t) = \sum_s \delta(t - t_s) * h(t) = \sum_s h(t - t_s)$$

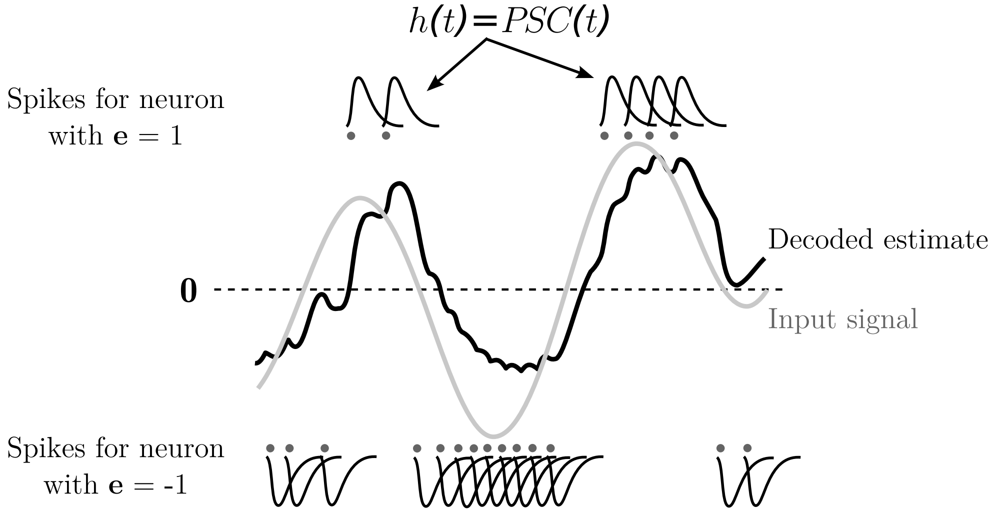
where s is the time of each spike, and $*$ is the convolution operator. Basically, we inject current into the neuron models using equation (3.5), and when a spike occurs (according to whatever single-neuron model we're using) we paste in a filter, $h(t)$.

We can then find the decoded estimate as a function of time; i.e.,

$$\begin{aligned}\hat{\mathbf{x}}(t) &= \sum_{i=0}^{n} \mathbf{d}_i a_i(t) \\ &= \sum_{i=0}^{n} \mathbf{d}_i \sum_{s_i} h(t - t_{s_i}). \end{aligned} \tag{3.10}$$

The looming question is what to use for the filter $h(t)$. We can find an optimal filter by once again minimizing the squared error between the actual value and the decoded estimate over time (see [51, 59]). However, these optimal filters are acausal; that is, they contain components in negative time, which would mean that a downstream neuron's activity at time t depends on a spike that will not occur until some time in the future. This could be solved by simply eliminating the negative-time portion of the optimal filter. However, we have already discussed in chapter 2 what happens to a downstream neuron's activity when a spike occurs: delivered neurotransmitter causes a transient rise in the current of the cell, which exponentially decays. A curve describing the rise and fall in intracellular current is called the post synaptic current curve (PSC), and we use it as our filter $h(t)$ (see figure 3.6).

$$h(t) = PSC(t) = \frac{1}{\tau^{PSC}} e^{-t/\tau^{PSC}} \tag{3.11}$$



h(t)=PSC(t)
Spikes for neuron with **e** = 1
Decoded estimate
0
Input signal
Spikes for neuron with **e** = -1

Figure 3.6: Example of decoding a time-varying scalar signal using a filtered spike train. For each spike, the filter $h(t)$ is pasted in, weighted by the decoding weight (1 or -1 in this case). Recreated from [58].

To summarize, the decoding procedure – the process that takes the spiking activity of a population of neuron models and decodes an n-dimensional vector – involves solving for decoding weights with equation (3.9), filtering spikes with the post-synaptic current curve, equation (3.11), and then summing the filtered spike trains of the population weighted by the decoding vectors, as in equation (3.10).

3.2.2 Transformation

In section 3.2.1 we discussed representation from the perspective of a sensory neuron; that is, if we had some input vector $\mathbf{x}$, which represents some environment variable, how would we encode it in neurons, and decode the value for higher-level analysis. This is a good starting point, but the vast majority of neurons in the brain do not receive their input from the environment, they receive it from the projections of other neurons, as originally discussed in chapter 2.

Fortunately, this does not affect the decoding procedure at all, it is simply a different – albeit more common – method of determining the input current being delivered to a neuron. The encoding procedure is still necessary for any model, for receiving the initial input.

From chapter 2, we know that neurons are not connected together with equal weight. We will represent the strength of the connection with ω_{ij} (following the conventions introduced in section 2.1); with this, we can determine the input current to neuron j in the postsynaptic population with

$$J_j = \sum_{i=0}^{n} \omega_{ij} a_i + J_j^{bias}, \tag{3.12}$$

where n is the number of neurons in the presynaptic population.

The remainder of this section details how to calculate ω matrices that will perform linear and non-linear transformations of encoded vectors.

Linear transformations

Suppose we have a population of neurons encoding some input vector $\mathbf{x}$, and we want a second population to encode the same value; this is the simple transformation $\mathbf{y} = \mathbf{x}$, a communication channel. We can solve for the amount of current that we want to be injected into a neuron j in the output population by substituting our decoded estimate $\hat{\mathbf{x}}$ into equation (3.4).

$$\begin{aligned} J_j =& \alpha_j(\mathbf{e}_j \cdot \hat{\mathbf{x}}) + J_j^{bias} \\ =& \alpha_j(\mathbf{e}_j \cdot \sum_{i=0}^{n} \mathbf{d}_i a_i) + J_j^{bias} \\ =& \sum_{i=0}^{n} \alpha_j \mathbf{e}_j \cdot \mathbf{d}_i a_i + J_j^{bias} \end{aligned} \tag{3.13}$$

Setting (3.12) equal to (3.13), we can easily determine the connection weight matrix.

$$\begin{aligned} \sum_{i=0}^{n} \omega_{ij} a_i + J_j^{bias} =& \sum_{i=0}^{n} \alpha_j \mathbf{e}_j \cdot \mathbf{d}_i a_i + J_j^{bias} \\ \omega_{ij} =& \alpha_j \mathbf{e}_j \cdot \mathbf{d}_i \end{aligned} \tag{3.14}$$

This procedure can be extended to any linear combination. If we want to scale an input vector, we construct a matrix $\mathbf{C}_{ji}$ with the first dimension having the same length as the output vector, and the second dimension the same length as the input vector. This is multiplied by the output population's encoding vector and the decoded estimate of the input population, giving us

$$\begin{aligned} J_j =& \alpha_j(\mathbf{e}_j \mathbf{C}_{ji} \hat{\mathbf{x}}) + J_j^{bias} \\ =& \alpha_j(\mathbf{e}_j \mathbf{C}_{ji} \sum_{i=0}^{n} \mathbf{d}_i a_i) + J_j^{bias} \\ \omega_{ij} =& \alpha_j \mathbf{e}_j \mathbf{C}_{ji} \mathbf{d}_i. \end{aligned}$$

Summing together vectors essentially happens automatically, assuming that the vectors are represented in separate populations. If vector $\mathbf{x}$ is being represented by one population, and vector $\mathbf{y}$ by another population, each connects to the output population through a separate connection weight matrix, and each imparts a certain amount of current that will be summed.

$$\begin{aligned} J_j =& J_j^{\mathbf{x}} + J_j^{\mathbf{y}} + J_j^{bias} \\ =& \sum_i \omega_{ij}^{\mathbf{x}} a_i^{\mathbf{x}} + \sum_i \omega_{ij}^{\mathbf{y}} a_i^{\mathbf{y}} + J_j^{bias} \\ \omega_{ij}^{\mathbf{x}} =& \alpha_j \mathbf{e}_j \mathbf{C}_{ji}^{\mathbf{x}} \mathbf{d}_i^{\mathbf{x}} \text{ and } \omega_{ij}^{\mathbf{y}} = \alpha_j \mathbf{e}_j \mathbf{C}_{ji}^{\mathbf{y}} \mathbf{d}_i^{\mathbf{y}} \end{aligned} \tag{3.15}$$

This can be extended to any number of input populations, allowing us to calculate any linear function ($\mathbf{C_1 x} + \mathbf{C_2 y} + \mathbf{C_3 z} + ...$).

Non-linear transformations

Non-linear transformations cannot be implemented in the same way. They require that

- the vector of arguments to the function, $\mathbf{x}$, is encoded by the population computing the function, and
- a new set of decoding weights, $\mathbf{d}^{f(\mathbf{x})}$, is computed.

This means that if you wish to do a non-linear computation involving the vectors encoded by two separate populations, you will first need to project each of those populations into an intermediate population through communication channels (see figure 3.7).

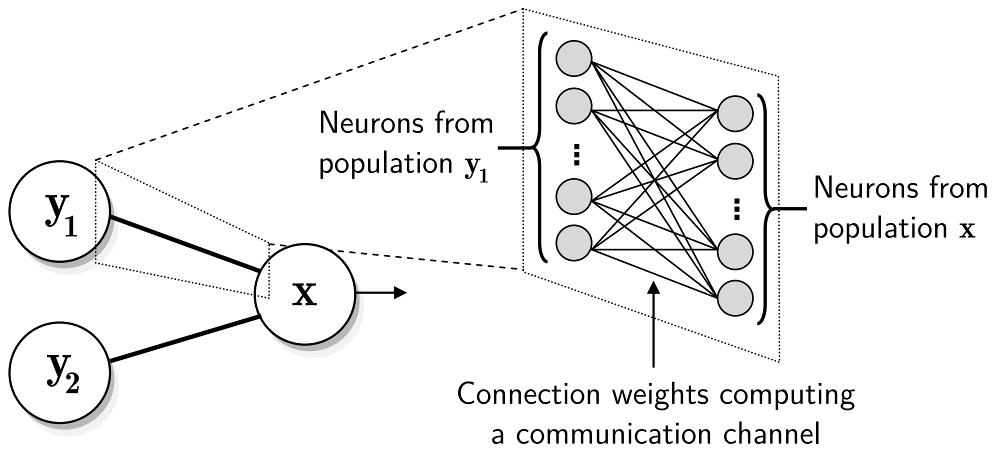

Neurons from population $\mathbf{y_1}$
$\mathbf{y_1}$
$\mathbf{y_2}$
$\mathbf{x}$
Neurons from population $\mathbf{x}$
Connection weights computing a communication channel

Figure 3.7: The network structure that is required for a non-linear transformation of values encoded in two separate populations. Note that the connection between two populations is all-to-all; that is, every neuron in the input population is connected to every neuron in the output population (though that connection can have weight 0). Adapted from [42].

As an example, say that you want to compute the element-wise product of two 3-dimensional vectors, $\mathbf{y_1}$ and $\mathbf{y_2}$.

$$f(\mathbf{y_1}, \mathbf{y_2}) = [y_{11}y_{21}, y_{12}y_{22}, y_{13}y_{23}]$$

We require an intermediate population that will represent $\mathbf{x} = [\mathbf{y_1}||\mathbf{y_2}]$, where $||$ is vector concatenation, so $\mathbf{x}$ is a 6-dimensional vector. If we compute decoders as usual, using equation (3.9), then we end up with a matrix of 6-dimensional decoders that can be used for linear transformations. Instead, we modify the calculation of the decoding weights such that instead of optimizing in order to calculate $\mathbf{x}$, we optimize to calculate $f(\mathbf{x})$.

$$\mathbf{d}^{f(\mathbf{x})} = \mathbf{\Gamma^{-1}\Upsilon}, \quad \text{where } \mathbf{\Gamma} = \mathbf{AA^T} \quad \text{and } \mathbf{\Upsilon} = \mathbf{A}f(\mathbf{X})^T \tag{3.16}$$

A visualization of how this linear decoding process can result in a nonlinear function can be seen in figure 3.8. With these decoders, we can use the result of this nonlinear function in the same way as a linear function and determine the connection weight matrix.

$$\begin{aligned} J_j =& J_j^{f(\mathbf{x})} + ... + J_j^{bias} \\ =& \sum_i \omega_{ij}^{f(\mathbf{x})} a_i^{\mathbf{x}} + ... + J_j^{bias} \\ \omega_{ij}^{f(\mathbf{x})} =& \alpha_j \mathbf{e}_j \mathbf{C}_{ji} \mathbf{d}_i^{f(\mathbf{x})} \end{aligned}$$

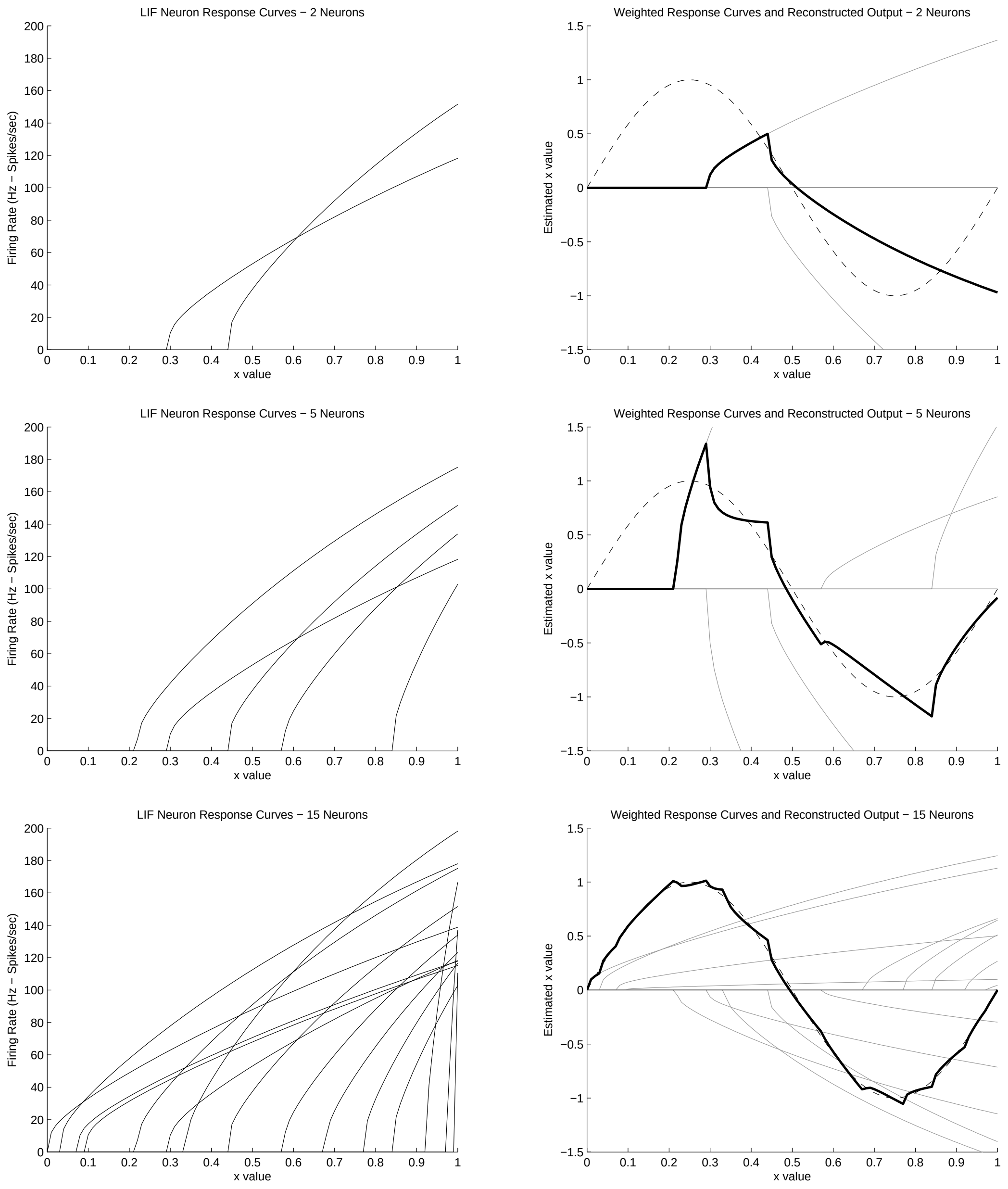


Figure 3.8: Illustration showing how the tuning curves of a population of LIF neurons can be linearly combined to estimate a nonlinear function, in this case $\sin(2\pi x)$. From [42].

3.3 Plasticity in the NEF

Chapter 2 discussed the experimentally observed phenomenon of synaptic plasticity, the idea that synaptic strengths (i.e. connection weights) change over time. The principles of the NEF from [59] say nothing about the modification of connection weights after their initial calculation through the method described in section 3.2.2; principle 2, transformation, analytically solves the same problem that learning algorithms attempt to solve.

Despite that, there are some practical benefits of modifying connection weights during the course of a simulation. Learning can help fine-tune models generated with the NEF – despite being determined 'optimally,' they can still be improved upon due to approximations made in the computation of decoding vectors [128]. Comparing learned networks to analytically determined ones can also highlight the strengths and weaknesses of the NEF approach.

However, from a more theoretical standpoint, investigating plasticity in the NEF is important as a verification of some of the assumptions of the theory. In determining these connection weight matrices that perform certain transformations, there is an implicit assumption that some learning process could arrive at a connection weight matrix that performs that function. One important contribution of this thesis is to show that there is a learning method that will arrive at connection weight matrices that can perform the functions that NEF models assume can be performed in the brain.

Implementing plasticity in simulations created according to the NEF is done through activity-dependent changes in ω matrices over the course of a simulation. This matches exactly how learning rules are written in plasticity literature (e.g. equation (2.1)); we can easily apply these learning rules to networks created in the NEF and analyze them both at the level of spiking patterns, and decoded n-dimensional vectors.

3.3.1 Error minimization rule

MacNeil and Eliasmith [128] derived an error-modulated learning rule that they used to fine-tune a neural integrator. That derivation is replicated here as it will be used in the simulations discussed in this thesis, and will be built upon in later chapters.

Once again, we do a least-squares minimization on the representation error.

$$SE = -\frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^2$$

Substituting in equation (3.6) gives

$$SE = -\frac{1}{2}\left(\mathbf{x} - \sum_{i=0}^{n} \mathbf{d}_i a_i\right)^2 .$$

In order to minimize this error, we differentiate with respect to the decoding weights.

$$\frac{dSE}{d\mathbf{d}_i} = \left(\sum_j \mathbf{d}_j a_j - \mathbf{x}\right) a_i,$$

where the subscript j indexes over all neurons in the population, while i indexes the neuron currently being optimized.

Note that the bracketed term represents the difference between the decoded estimate, $\hat{\mathbf{x}}$, and the actual value, $\mathbf{x}$. We will call this quantity $\mathbf{E}$.

$$\mathbf{E} = \hat{\mathbf{x}} - \mathbf{x}$$

Our equation is now

$$\frac{dSE}{d\mathbf{d}_i} = \mathbf{E} a_i, \tag{3.17}$$

which indicates that in order to minimize total error, we need to modify each decoding vector $\mathbf{d}_i$ by the signed error, scaled by the activity coming from the presynaptic cell. One interpretation of (3.17) is that, since we are using essentially the negative of the representation error, we are punishing (decreasing the weight of) each decoder by a value proportional to its activity; if we are doing a poor job of representing $\mathbf{x}$, then the decoding weight of a neuron that is highly active will move closer to zero, essentially decreasing the influence of the neuron that is contributing poorly to the encoding.

If we include a learning rate parameter, κ, this can be expressed succinctly as

$$\Delta \mathbf{d}_i = \kappa \mathbf{E} a_i. \tag{3.18}$$

While this is interesting from a computational efficiency standpoint (see section 7.4.1), it is not biologically plausible because we are not changing synaptic connection weights. Recall from equation (3.14) that $\omega_{ij} = \alpha_j \mathbf{e}_j \cdot \mathbf{d}_i$. We can multiply both sides by the encoding vector $\mathbf{e}_j$ and the neuron gain α_j to express the rule in terms of ω_{ij}.

$$\begin{aligned} \Delta \mathbf{d}_i \cdot \mathbf{e}_j \alpha_j =& \kappa \alpha_j \mathbf{e}_j \cdot \mathbf{E} a_i \\ \Delta \omega_{ij} =& \kappa \alpha_j \mathbf{e}_j \cdot \mathbf{E} a_i \end{aligned} \tag{3.19}$$

Despite the rule being derived without considering biological plausibility, it is plausible in the sense that it meets the locality constraint; all of the terms in this rule can be argued for being available locally to the synapse, and it is synapse-specific.

Chapter 4

Supervised learning

This thesis will discuss three machine learning problems: supervised learning, reinforcement learning, and unsupervised learning. While there are varied methods of solving these problems in machine learning, we will only discuss methods in artificial neural networks, as they are the most directly applicable to our spiking neural networks.

In machine learning, supervised learning is the problem of learning how to perform an unknown function in a neural network given a set of input and output examples. Having access to input-output pairs is an important distinction between supervised learning and other types of learning.

Mathematically, the supervised learning problem is inferring a function, $G(\mathbf{x})$, given a set of input vectors, $\mathbf{X}$, and their corresponding outputs, $\mathbf{Y}$. Usually, most of the input-output pairs are used for training, and some input-output pairs are designated for use in testing to see how well the network has learned to approximate $G(\mathbf{x})$. This is done to show that the network has generalized the function, rather than simply learning the mapping $\mathbf{X} \rightarrow \mathbf{Y}$ (see figure 4.1).

The general strategy for solving the supervised learning problem is to feed the input into the neural network and record its output. Since we have access to the target (desired) output, we take the difference between the network's current output and the desired output, and call that the error. The error can then be used to update the connection weights in the neural network, such that the next time that the network is supplied with the same input, it arrives at a result closer to or exactly the desired output.

Implementing this general strategy poses a few problems.

Training

Given input $\mathbf{X}$ and output $\mathbf{Y}$,

$$\text{Minimize } \left(\mathbf{Y}_{train} - G(\mathbf{X}_{train})\right)^2$$

Testing

$$\text{Evaluate } \left(\mathbf{Y}_{test} - G(\mathbf{X}_{test})\right)^2$$

Figure 4.1: The supervised learning problem.

1. Calculating the error signal. The difference between the network's current and ideal output may be non-trivial to compute.
2. Determining how the error signal should translate to changes in connection weight strengths.

4.1 Supervised learning in traditional artificial neural networks

In machine learning, the first problem is trivial, because the output of the network can be made to match the type of the ideal output. That is, if the output of the function is a scalar value in a certain range, the activation function of the output layer can also create a scalar value in that range. It is thus easy to calculate the general error signal.

However, how that error signal translates into connection weight strength changes is non-trivial. This problem was first addressed by Bryson and Ho in 1969 [34] with what is now called the backpropagation algorithm, though the method did not reach widespread use until its rediscovery in the 1980s [172].

4.1.1 Backpropagation

Backpropagation is usually used in a type of artificial neural network known as a multi-layer perceptron. This type of network is made up of at least three layers: one input layer, one output later, and one or more hidden layers (see figure 4.2). Each non-input node (neuron) has a nonlinear activation function (also called the transfer function), which is meant to emulate the *tuning curve* of a biological neuron. Typically, the activation function is a sigmoid curve (see equation (4.1) and figure 4.3 – note that this sigmoid curve has some qualitative similarities with the LIF tuning curve, figure 3.4).

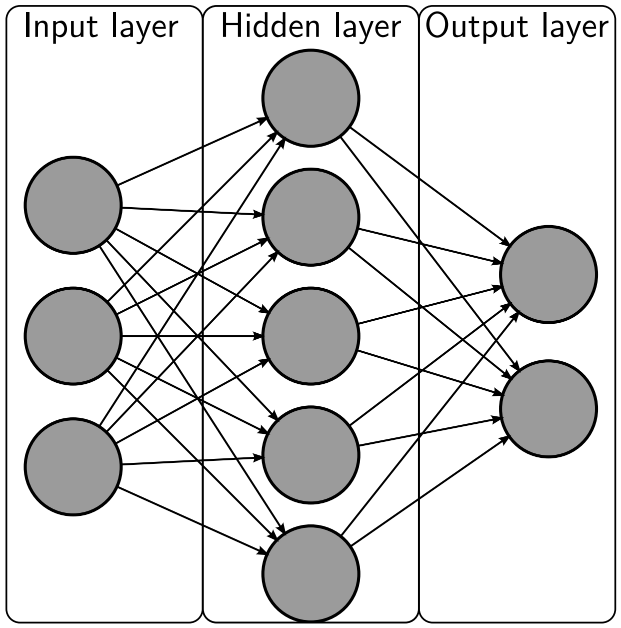


Figure 4.2: The architecture of a multi-layer perceptron. There can be many hidden layers if desired, but for the purposes of supervised learning, the Universal Approximation Theorem states that any non-linear function can be computed with only one hidden layer, so this 3-layer architecture is the most common.

$$a(x) = \tanh(x) \quad \text{or} \quad a(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

In order for backpropagation to work, the nonlinear activation function must be differentiable. The main idea of the algorithm is to calculate the global error signal, and use that to calculate the error for each neuron in the output layer. That error is used to calculate the "local" error for each neuron in the previous hidden layer, which is used to calculate the error in the previous hidden layer if one exists. These local error calculations depend on the derivative of the activation function of each neuron. In other words, backpropagation determines which nodes most contribute to the error, and modifies connection weights such that those nodes receive less input from afferent neurons in the future. A

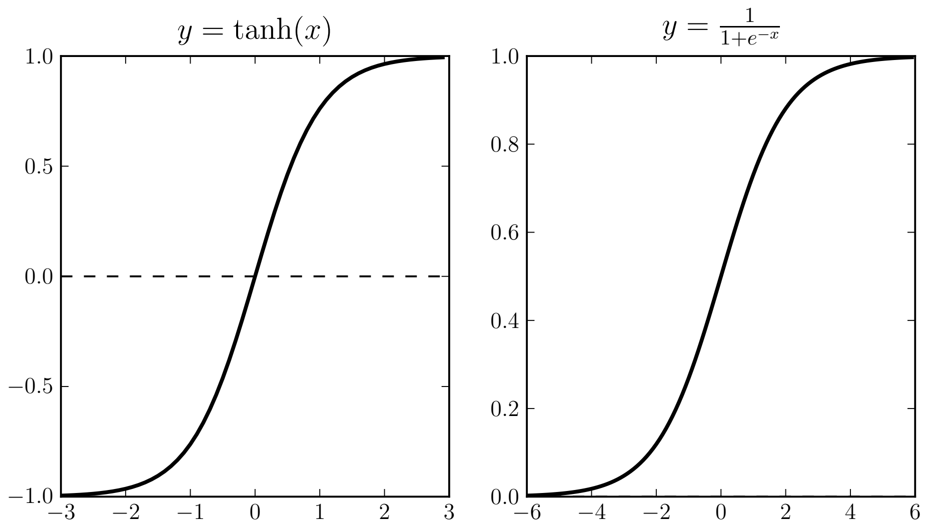


Figure 4.3: The two sigmoid curves from equation (4.1). Note that $\tanh(x)$ gives values from -1 to 1, and the other values from 0 to 1.

simplified version of the complete learning rule is

$$\Delta\omega_{ij} = -\kappa\delta_j a_i, \tag{4.2}$$

where δ_j is the local error at neuron j.

The Universal Approximation Theorem states that the multilayer perceptron architecture, as depicted in figure 4.2, can approximate any nonlinear function with only one hidden layer [46, 88]. Additionally, it has been shown to be able to simulate a Turing Machine, meaning that is able to do any arbitrary computation [182]. Unfortunately, even if it is true that a neural network exists to compute a given function, no amount of input-output pairs will guarantee that backpropagation will converge to this network. It is sensitive to many factors, such as the number of nodes in the hidden layer, the choice of input-output pairs, the learning rate, and the initial network weights. As a gradient descent algorithm, it can fall into local minima. Further, even if the network converges, it can be very slow to do so [221].

From the perspective of applying backpropagation ideas to our spiking neural networks, we run into a number of additional issues. The most obvious is that the activation function expresses the output of a neuron as a scalar value, usually between 0 and 1 or -1 and 1. In a spiking neural network, the activation function produces spikes, which are necessarily discontinuous and therefore non-differentiable.

The most important issue for backpropagation is that is not possible to frame the algorithm in a biologically plausible manner. As discussed in chapter 2, backpropagating action potentials exist, making the idea of propagating error signals backwards attractive; however, in the traditional backpropagation algorithm, local errors are calculated using connection weights between neurons one or more layers downstream. It is difficult to see how this information could be communicated to a synapse, even with feedback connections.

There are many other methods of doing supervised learning in neural networks, including extensions of traditional backpropagation, such as QuickProp and RProp [221]. One can also treat the neural network as a nonlinear function with each connection weight a free parameter, and use nonlinear optimization techniques like simulated annealing and genetic algorithms to find $G(\mathbf{x})$. However, we discuss backpropagation in this section because it is the most well known learning algorithm in artificial neural networks, and it is often used in conjunction with the Universal Approximation Theorem to say that a neural network can learn to approximate an arbitrary nonlinear function. As we have already argued, however, backpropagation is not a valid algorithm for the type of large-scale neural networks that we are interested in this thesis, and so the question of whether supervised learning can be accomplished in this framework remains unanswered.

4.2 Supervised learning in spiking neural networks

Experimental neuroscience literature has little to say about supervised learning and how it might be implemented in the brain. One of the few areas in the literature that mention supervised learning is cerebellar research. There is evidence that the climbing fibers represent a fine-grained error signal, the same kind that one would expect to see if a supervised learning scheme was implemented neurally. In fact, the neural architecture is such that it seems uniquely poised to offer neural evidence of supervised learning (see figure 4.4). However, leading models of the cerebellum have taken this error signal to mean that it is an adaptive filter [52, 68]. While these models are computationally powerful, it is not clear that they offer a solution to the supervised learning problem.

The theoretical neuroscience community as a whole has offered many solutions to a type of supervised learning that we will refer to as supervised spike-time learning (see figure 4.5).

The problem is similar. In both cases, we want the neural network to approximate a function $G(\cdot)$; in machine learning, this function accepts a vector ($G(\mathbf{x})$), and in theoretical neuroscience this function accepts an input spike train ($G(S(t_i))$).

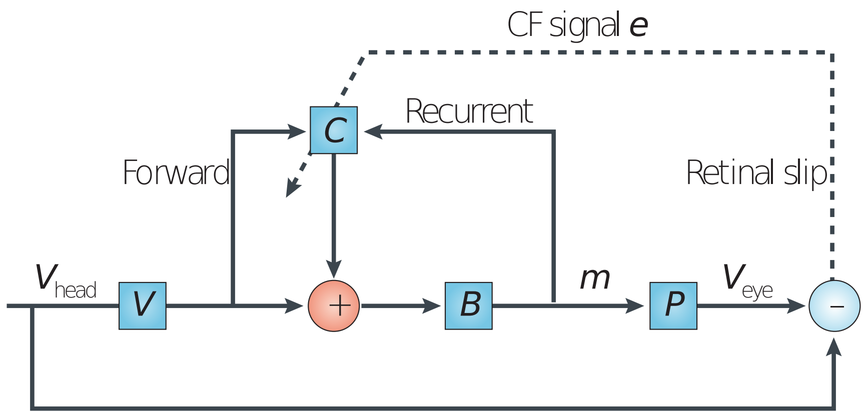


Figure 4.4: Architecture of a model of the cerebellum that posits that the cerebellum is an adaptive filter. From [52].

Construct a network with arbitrary connection weights, ω. Given

- $S(t_j^d)$, the desired spike train of output neuron j, and
- $S(t_i)$, the spike train of an input neuron i,

modify ω such that

$$D\left(S(t_i), S(t_j)\right)$$

is minimized, where $D(S_1, S_2)$ is a measure of the dissimilarity between two spike trains [47, 214].

Figure 4.5: The supervised spike-time learning problem.

Accepting only spike-times as input and outputting only spike-times is typical for spiking neural networks that employ temporal coding schemes. In these coding schemes, information in the network is encoded in spike-times and inter-spike intervals [204]. Evidence that the brain uses precisely timed spikes has been found in many sensory systems [213], and it has been argued that networks using solely temporal information can simulate a Turing Machine [127].

While we have not found biologically plausible solutions to the general supervised learning problem, temporal-coding based solutions to the supervised spike-time learning problem exist. In the next section, we summarize a representative sample of algorithms that provide a solution to the supervised spike-time learning problem.

4.2.1 Temporal-coding based models

The most straightforward approach to implementing supervised learning in spiking neural networks is to follow the same approach as Bryson and Ho: use a gradient descent approach on the error in the network. This is more complicated than traditional backpropagation because a spiking neuron's activation function is not differentiable, but Bohte et al. derived a backpropagation-like algorithm for spiking networks with some additional assumptions [31, 32]. Their algorithm, called *SpikeProp*, operates much like traditional backpropagation in that it calculates the global error – the time difference between the spike train created by the network and the desired spike train – and assigns local error for each node, which is used to modify connection weights proportionally to the node's activity. Like backpropagation, however, the local error for each node depends on the connection weights of downstream neurons, making this algorithm biologically implausible.

Ignoring the biological implausibility, the only simulation result from Bohte et al. was a solution to the XOR problem, a result which has seen some difficulty being replicated [143]. While this shows that it can solve a nonlinear problem (with only 10 neurons, to its credit), it is not clear how this would scale up to problems with complicated representations. Additionally, only the time of the first spike for each neuron in the network is considered; all subsequent spikes are ignored, and the network must be reset to do another learning step.

Since its derivation in 2002, SpikeProp has been extended by Schrauwen and Campenhout by learning synaptic time constants and neuron thresholds in addition to synaptic connection weights [176] and by McKennoch et al. to create spiking versions of the previously mentioned QuickProp and RProp [138]. Neither of these extensions addresses the issues with SpikeProp, however.

A more recent algorithm that does supervised spike-time learning is the Remote Supervised Method (*ReSuMe*) [161]. The learning rule used by ReSuMe is inspired by biology; specifically,

$$\begin{aligned}\Delta\omega_{ij} &= -\Delta\omega_{ij}^{STDP} + \Delta\omega_{ik}^{STDP} \text{ for excitatory synapses,}\\ \Delta\omega_{ij} &= \Delta\omega_{ij}^{STDP} - \Delta\omega_{ik}^{STDP} \text{ for inhibitory synapses,}\end{aligned} \tag{4.3}$$

where i, j, and k index presynaptic, postsynaptic, and "teaching" neurons respectively, and ω_{ij}^{STPD} is the result obtained from a spike-timing dependent plasticity rule. The STDP curve used in ReSuMe is not the stereotypical asymmetric curve (see figure 4.6, in comparison to figure 2.7).

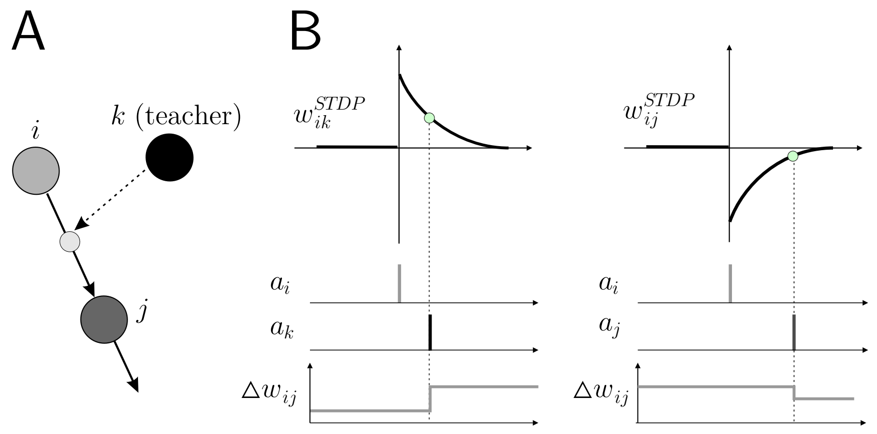


Figure 4.6: A summary of the ReSuMe algorithm, from [161]. (A) The structure of the network: a neuron is either an input, output, or teaching neuron. (B) The STDP curves of the two learning rules. Note that one rule only implements potentiation and one depression, though since they are summed in the final learning rule, bidirectional synaptic strength changes occur.

ReSuMe also uses a non-Hebbian term to speed up learning, but even without the term, Ponulak proves that ReSuMe can learn to modify synaptic weights such that an output neuron spikes at the desired time given the same input spike trains [161].

ReSuMe's learning rule, (4.3), is inspired by biology, but its biological plausibility is questionable based on current evidence. Notice in equation (4.3) that the change in the synaptic strength between neurons i and j requires information about the synapse between neurons i and k. Heterosynaptic plasticity has certainly been observed, especially in the context of modulatory neurotransmitters, as discussed in section 2.3, but typically in those cases it is not the precise timing of the modulatory synapse that matters, but the phasic level of the neurotransmitter available to the plastic synapse (as in the three-factor rule (2.4)). It is, however, still possible that this type of heterosynaptic plasticity happens in the brain, particularly in the cerebellum, in which the error signal is thought to be transmitted through typical excitatory and inhibitory synapses [52] rather than through dopaminergic synapses, as in the striatum [180].

However, ReSuMe, like many temporal coding models, does not account for the large amounts of noise in the brain. One temporal coding study that accounted for noise was Pfister et al.'s [157, 159], in which the spiking behaviour of neurons depended on both synaptic input and a stochastic term intended to account for noise. They derived an STDP-like learning rule that was able to solve the temporal-coding supervised learning problem in the face of noise; however, simulations were only done with target spike trains of at most two spikes. It is not clear if the method would scale up to complex spike trains with many spikes.

Other methods based on temporal coding have been proposed ([10, 16, 39, 155, 220]) but because we are primarily interested in solving the general supervised learning problem, 4.1, we will leave further investigation of these methods to the interested reader.

4.2.2 Biologically plausible backpropagation

Another set of approaches to supervised learning in theoretical neuroscience attempt to address the biological plausibility of the original backpropagation algorithm rather than create a completely new approach.

Bogacz et al. made a model of perirhinal cortex that learns with a backpropagation-like synaptic learning rule that they call *FreqProp* [29, 30]. In their model, they fix the weights from the hidden layer to the output layer, according to the observation that in perirhinal cortex, neurons that spike depending on the novelty of a stimulus compute a trivial vote-summing scheme, and thus the connection weights from the hidden to output layer are fixed and can be ignored in the calculation of the learning rule modifying the input-hidden layer synapses. That learning rule is

$$\Delta\omega_{ij} = \frac{1}{N}\delta a_i a_j, \tag{4.4}$$

where δ is the activity of the output layer nodes, which is projected back to the ω_{ij} synapse. The name FreqProp comes from the notion that the frequency of firing of the output layer neurons modulates the amount of synaptic change in the earlier layers. The authors note that the dependence on the weights of hidden-output synapses being equal is necessary for the biological plausibility of the rule, and results from observations of perirhinal cortex, which is unlikely to be true of other cortical areas.

Hinton and McClelland proposed an alternative to backpropagation that introduces recurrent connections between each layer to allow computation of local contributions to the global error in a biologically plausible manner [85]. The output layer, then, projects back to the hidden layer, enabling local computation of the hidden node's contribution to error, and similarly for the input and hidden layer projections. Their algorithm, termed the recirculation algorithm, was built on by O'Reilly to create a learning model called the $\underline{\text{l}}$ocal $\underline{\text{e}}$rror-driven $\underline{\text{a}}$ssociative $\underline{\text{b}}$iologically $\underline{\text{r}}$ealistic $\underline{\text{a}}$lgorithm (LEABRA) [152, 153]. Their method was able to learn a number of difficult nonlinear tasks, but the neuron model used did not produce spikes, so the issue of the non-differentiability of a spiking transfer function was not addressed.

Finally, a study by Körding and König argues for biological plausibility of backpropagation with a model that uses two different sites of synaptic integration, based on the experimental observation that activity in basal dendrites and apical dendrites differs significantly, and in ways that strongly modulate learning [110]. In their model, the learning rule is the product of presynaptic and postsynaptic action potentials and "dendritic bursts." Action potentials are generated by applying a transfer function to basal dendritic activity, while dendritic bursts are generated by applying a different transfer function to apical dendritic activity. The authors propose a mapping from their model to backpropagation, though they do not assert that the brain is actually doing this, only that the two sites of synaptic integration make this type of learning biologically plausible, with some additional assumptions.

4.3 Supervised learning with the NEF

Our goal in examining supervised learning in the NEF is to solve the general supervised learning problem summarized in figure 4.1. In the remainder of this section, we argue that this problem is more general than the supervised spike-time supervised learning problem (figure 4.5), and we show that a combination of the encoding/decoding scheme of the NEF and learning rule (3.19) can solve this more general problem.

In the previous section, we discussed temporal-coding in spiking neural networks. In it, information is transmitted by the time of spikes in a spike train. It should be noted that models using the NEF's principles can be created such that they are sensitive to very precise spike-times, but in general, the NEF's coding scheme considers a small time window of spiking behaviour, making it somewhere in-between traditional "rate coding" and "temporal coding" schemes.

4.3.1 Theoretical argument

Recall from section 3.2.1 that the NEF defines a process for encoding vectors of real-valued numbers in the spiking profiles of neural populations, and a process for decoding spiking profiles back into vectors of numbers. Also recall the two versions of the supervised learning problem introduced in this chapter (figures 4.1 and 4.5).

One approach to using the NEF's encoding/decoding scheme to solve the machine learning problem would be to take an input vector, $\mathbf{x}$, encode it in a population, and record the spike trains of the population for some length of time. We could then take an output vector, $\mathbf{y}$, encode it in a different neural population, record its spike trains, and call those our desired spike times. If we connect the input and output populations in a manner described by the algorithms presented in the previous section, with random connection weights, we should be able to use their algorithms to learn a set of connection weights that approximates $G(S(t_i)) \approx G(S(t_j^d))$.

However, recall that $\mathbf{x}$ and $\mathbf{y}$ can be encoded in an infinite number of different spiking profiles. Intuitively, it would seem that an algorithm that could learn the $G(\mathbf{x}) \approx \mathbf{y}$ transformation in a biologically plausible manner would be more general than one that learns the spike-time equivalent version of this. Additionally, functions that operate in vector spaces are ones that can be well understood by humans, whereas functions that operate in the space of spikes require additional steps to understand.

In order to more rigorously show that the supervised spike-time learning problem is less general than the supervised learning problem, it would be ideal to show that a solution to the general problem can also solve the supervised spike-time learning problem. This work has not yet been done. However, a solution to the general supervised learning problem follows.

Recall from the beginning of this chapter the general strategy to solving the supervised learning problem. We need to first compute an error signal, and then determine how that error signal translates into changes in connection weights. In backpropagation, the second part is difficult, but in our case we have already derived a learning rule that accomplishes this, rule (4.5) (copied below for convenience).

$$\Delta\omega_{ij} = \kappa\alpha_j\mathbf{e}_j \cdot \mathbf{E}a_i \tag{4.5}$$

The more difficult part in this case is calculating the error signal, because we want the network to compute this quantity in a biologically plausible manner. This requires using the transformation principle of the NEF (see section 3.2.2). Ideally, all of the connection weight matrices in the network would be learned, but even though we use some analytically determined connection weight matrices, they are simple linear transformations, and so it does not affect the biological plausibility of the overall network, pictured in figure 4.7.

The connection matrix being learned is the one between the `Input` and `Output` populations. Looking at rule (4.5), when we create the output population, we know α_j, the gain, and $\mathbf{e}_j$, the encoding vector. We choose some appropriate learning rate κ. The activity of the `Input` population, a_i, is the spiking activity that represents the encoded vector, $\mathbf{x}$.

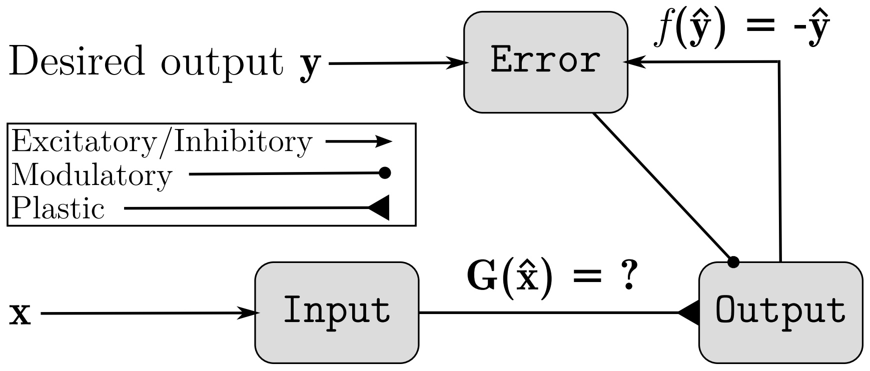

Desired output $\mathbf{y}$
Error
$f(\hat{\mathbf{y}})$ = -$\hat{\mathbf{y}}$
Excitatory/Inhibitory
Modulatory
Plastic
$\mathbf{x}$
Input
$\mathbf{G}(\hat{\mathbf{x}})$ = ?
Output

Figure 4.7: The network topology used to solve the supervised learning with a model created according to the principles of the NEF. Note the similarity with the biologically inspired network in figure 4.4.

All of these are available locally to the synapse. The error signal, $\mathbf{E}$, is the global error; that is, it is the difference between our actual output and desired output. Note that this quantity is multidimensional. $\mathbf{E}$ is computed by the `Error` population; it sums the input from the `Output` population and a population that represents the desired output according to equation (3.15). Since the desired output is also encoded in a neural population, $\mathbf{E} = \hat{\mathbf{x_d}} - \hat{\mathbf{x}}$. With this, we are able to evaluate the learning rule continuously online.

It is interesting to note that, although we did not model our learning rule after backpropagation, because it works on the idea of gradient descent, the two have a very clear mapping.

$$\begin{aligned}
&\text{Backpropagation} \\
\Delta\omega_{ij} &= -\kappa\delta_j a_i \\
&\text{NEF error-minimization} \\
\Delta\omega_{ij} &= \kappa\alpha_j \mathbf{e}_j \cdot \mathbf{E} a_i \\
&= -\kappa\delta_j a_i, \qquad \text{where } \delta_j = -\alpha_j \mathbf{e}_j \cdot \mathbf{E}
\end{aligned}$$

In backpropagation, we noted that δ_j is the local error at neuron j. If we rearrange the terms as above, the NEF error-minimization rule's δ_j is also a measurement of the local error at neuron j; $\mathbf{e}_j$ tells us the direction that neuron j is sensitive to, and α_j tells us how sensitive it is to that direction. Since the error $\mathbf{E}$ is also a vector, we are projecting the error onto the neuron's encoding vector, and scaling it by α_j. This is essentially determining how much neuron j is responsible for the global error; if the error is high, but in a direction that j is not sensitive to, then δ_j is low, but high error in the direction that j is sensitive to results in a high δ_j. It does the same thing as backpropagation without requiring backpropagation because the encoders prescribe what part of the error space it should be sensitive to, rather than determining it during the simulation based on downstream neurons. This means that neurons in the hidden layer are less flexible in the NEF, and thus an NEF network may require more neurons than would be required for learning the same function in a multilayer perceptron, but this is a small cost for the benefit of biological plausibility.

It should be noted that, while the network shown in figure 4.7 appears to be a two layer network with just the `Input` and `Output` populations (and thus solving supervised learning in this framework would violate the proof of the Universal Approximation Theorem [88, 46]), this should be thought of as a three-layer network because there are at least two nonlinearities applied to the input signal. Encoding the input vector $\mathbf{x}$ in the input population is a nonlinear process, making the `Input` population of our network more analogous to the hidden layer of a traditional 3-layer perceptron. The encoding process does not use synaptic weights (it is, like the learning rule, prescribed by the encoding vectors); it is only the `Input-Output` (i.e. hidden-output layer) connections that are learned. Recall from the representation principle (see section 3.2) that the decoding process is linear, so this is still analogous to the 3-layer perceptron architecture that is most often used in artificial neural networks.

4.3.2 Biological plausibility

In this section we argue for the biological plausibility of our neural solution to the supervised learning problem.

The network structure from figure 4.7 is simple, and there is some evidence that similar networks exist in parts of the brain like the cerebellum (see figure 4.4). All of the elements of the learning rule are local and available at the synapse, which is the primary determinant of biological plausibility for a learning rule.

- κ, the learning rate, is a convenience term that ensures that the amount of synaptic modification is appropriate for the network. It collects a number of scalar factors that would likely combine simply in a very detailed biophysical model.
- α_j, the gain factor, is an intrinsic property of a neuron.
- a_i, the activity of the presynaptic cell, is communicated by the axon terminal of the presynaptic side of the synapse.

More thought must be given to the final two terms, $\mathbf{E}$ and $\mathbf{e}_j$. If we examine each of these individually, $\mathbf{E}$ is some modulatory signal projected from the `Error` population. This global error signal could be signalled by a phasic change in dopamine levels, as in three-factor rules, and indeed much of the literature treats dopamine quantity as a single global variable. Alternatively, $\mathbf{E}$ could be the activity trace of some other neural activity, as is more likely in the cerebellum. The encoding vector, $\mathbf{e}_j$, should not be thought of as a property intrinsic to a neuron, though an argument could be made for it in purely sensory neurons. Instead, a better physical correlate for the encoding vector is that it reflects a neuron's spatial position in a physical network of neurons. As mentioned several times in this thesis, learning appears to be critically modulated by a synapse's location on the dendritic tree of a neuron. The neuron's spatial location in a network certainly affects how other neurons will connect to its dendritic tree, and the effects of that spatial position may be well summarized by the encoding vector $\mathbf{e}_j$. This would suggest a possible role for random local connections that are made during development, and for connections that are made based on genetics.

We also submit as an alternative argument for biological plausibility that we can consider the dot product $\mathbf{e}_j \cdot \mathbf{E}$ as a single quantity with a single physical correlate. As mentioned, this is essentially the local error for neuron j; this (scalar) value is more easily analogized to phasic dopamine levels available to a synapse, as it does not require the same amount of dopamine to be delivered to each synapse.

Experimental explorations of this type of heterosynaptic plasticity were discussed in section 2.3. We make the additional hypothesis that dopamine levels in and around each synapse are different across a neural population. Aragona et al. showed that phasic dopamine levels are different in subregions of the nucleus accumbens, lending credence to our hypothesis [3]. More precise experiments measuring phasic dopamine levels at individual neurons would differentiate between the two arguments for the biological plausibility of $\mathbf{e}_j \cdot \mathbf{E}$. Specifically, if it were shown that phasic dopamine levels were different between two neurons in close spatial proximity, it would support the explanation that identifies a single neural correlate of the quantity $\mathbf{e}_j \cdot \mathbf{E}$, because spatial location in a population is not a factor in this explanation. If, however, it were shown that phasic dopamine levels are similar for neurons in close spatial proximity, but different for those farther apart, these findings would support the explanation that identifies separate neural correlates of $\mathbf{e}_j$ and $\mathbf{E}$, because in this explanation, spatial location is hypothesized to be an important determinant of the area of the error space a neuron is sensitive to.

4.3.3 Simulation results

A number of simulations were run to confirm the theoretical results, and determine how fast certain functions could be learned.

For each of the mathematical functions tested, a network like figure 4.7 was constructed; the only differences between the networks were the number of neurons used in the various populations and the function used to produce input-output pairs. All of the models use only leaky integrate-and-fire neurons. The number of neurons used varies, and are listed in table 4.1.

The `Input` population receives as input a randomly varying vector. Each dimension of the input vector changes by a small amount on each timestep (determined randomly from a Gaussian distribution with mean 0 and variance 0.05). The ideal output is computed directly using the function being tested.

The connection weight matrix between the `Input` and `Output` populations is initialized with random values chosen from a uniform distribution in the range $[-0.0001, 0.0001]$. We apply learning rule (4.5) to every connection weight in the `Input-Output` matrix on each timestep of the simulation. The timestep used was $dt = 1\text{ms}$.

Evaluation method

Recall that the `Error` population encodes the $\mathbf{E}$ value that we use to modify the connection weights. We use this measure in evaluating how well the network is performing the target function. A simulation is broken up into training and testing phases. In the testing phase, learning is disabled. Until the end of the testing phase, the absolute value of the value represented by the `Error` population is accumulated. When the training phase begins, the accumulated error is recorded, and learning resumes.

The accumulated error can be calculated with the formula

$$\begin{aligned}\mathbf{E}_{acc} &= \int_{t^s}^{t^e} |\mathbf{E}| \\ &= \int_{t^s}^{t^e} |\hat{\mathbf{x}}_{\mathbf{d}} - \hat{\mathbf{x}}|,\end{aligned}$$

where t^s is the start of a training phase and t^e is the end of a training phase.

The accumulated error of a network during learning is not solely due to the network having not yet learned the transformation. Some amount of the error in the learning network is due to factors other than the connection weight matrix; noise, randomly chosen neuron parameters, the nonlinearities present in spiking neural networks, and the theoretical limitations of linear decoding in populations of neurons all contribute to error when examining the simulation on a timestep-by-timestep basis. For this reason, the accumulated error values from the networks over the course of learning are compared to the average accumulated error values of an analogous control network. The control network has connection weights calculated using the NEF's transformation principle, as discussed in section 3.2.2.

The metric by which we will evaluate network performance over time is the ratio of the accumulated error over the testing phase of the learning network to the average accumulated error over the same length of time in the control network. The average accumulated error over the same length of time in the control network can be thought of as the expected value of the accumulated error, giving us as a final network performance metric

$$\frac{\mathbf{E}_{acc}}{E[\mathbf{E}_{acc}]}.$$

Test functions

Five networks to learn nonlinear functions were created. The functions learned in these networks differed in the dimensionality of the input and output, among other factors. In this section, we make explicit the five functions and the properties that make them interesting targets for testing the performance of the learning networks.

In addition to the practical benefits of the functions chosen, we note that previous NEF models have used these nonlinear functions [42, 166, 192]. Since the implicit assumption in their use is that such functions could be learned in the brain, these experiments lend credence to those models by showing that they indeed can be learned with sufficient time and an appropriate network setup.

Multiplying two scalars

$$f(x_1, x_2) = x_1 x_2$$

Multiplying two numbers is the simplest function that was attempted, in terms of the dimensionality of the input and output. The input is two-dimensional, and the output is one-dimensional. It is essential a minimal example, similar to the XOR problems that many of the supervised spike-timing learning algorithms solve; though, because the output is scalar and not binary, it is a more difficult problem.

Combining two products

$$f(x_1, x_2, x_3, x_4) = x_1x_2 + x_3x_4$$

This function combines two separate, unrelated products. The input is four-dimensional, and the output is one-dimensional. An important feature of this function is that it is a linear combination of non-linear functions. Despite that, the error signal is scalar, matching the dimensionality of the output of this function; so, while the network is learning two non-linearities and the linear combination of them, the only reinforcement it receives is how incorrect it is at the current timestep.

This function was chosen because it increases the dimensionality of the input space, providing some indication of how increasing input dimension affects the number of neurons needed in the input population, how long the network takes to learn, and so on.

Three separate products

$$f(x_1, x_2, x_3) = [x_1x_2, x_1x_3, x_2x_3]$$

This function computes three separate, but related, products. The input is three-dimensional, and the output is three-dimensional. This is the first demonstration of learning a function that produces multidimensional output. While it is not a minimal example of this, it is a function that can be separated into three functions that have previously been examined. So, a learning rule that can learn to multiply two numbers can solve this problem with three subnetworks. Using multidimensional input and output values tests the learning rule's ability to learn in these higher-dimensional spaces.

Supervised learning in high-dimensional spaces is a difficult problem because the size of the input and output spaces grows exponentially with dimensionality. Because of this, previous studies have examined different ways to work around this problem (e.g. [70]). By using a multidimensional error signal, however, we show that it is possible to learn in the high-dimensional space directly.

Two-dimensional circular convolution

$$f(x_1, x_2, x_3, x_4) = [x_1, x_2] \otimes [x_3, x_4]$$

The circular convolution operation is used in many NEF models because it can be used to bind and unbind vectors that represent symbols. Previous models using circular convolution have used a 3-layer network to improve accuracy. In the first layer, a linear transformation projects the original vector into the frequency domain (i.e., computes the discrete Fourier transform). In the second layer, the vectors in the frequency domain are computed. In the third layer, the vectors are projected back into the signal domain using the inverse discrete Fourier transform.

This 3-layer approach differs from the network structure shown in figure 4.7, and requires more neurons. It is not clear how an analogous 3-layer network would be learned, so we attempt to learn the circular convolution operation with the typical 2-layer structure. The learned network, in this case, is compared to both the 3-layer approach typically used in networks without learning, and a 2-layer approach in which decoders to approximate circular convolution are computed.

Three-dimensional circular convolution

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = [x_1, x_2, x_3] \otimes [x_4, x_5, x_6]$$

Adding a third dimension to the circular convolution network described above gives insight into how additional dimensions affects the nature of the learning in the network. Like in the 2D case, we compare the learned network with a 3-layer and 2-layer version of the control network.

While it would be ideal to continue exploring convolution in even higher-dimensional spaces, three dimensions starts to reach the ceiling of how complex the network can be tested in a reasonable amount of computer time.

A summary of these five functions and some of specifics of the networks is provided in table 4.1.

Results

The five learning networks were simulated for an appropriate length of time based on initial tests of how long the network takes to converge to a good solution. Each non-convolution network was simulated 40 times. The convolution networks were simulated 10 times each, due to the length of time necessary to run these simulations. The average relative accumulated error values over time are plotted for each learning network in figure 4.8.

As a whole, these graphs provide strong evidence that our method can learn to compute arbitrary nonlinear functions. The nonlinear functions are in an arbitrary vector space, and

Function	**Input dimensions**	**Output dimensions**	**Neurons in learning network**	**Neurons in control network**
$f(\mathbf{x}) = x_1x_2$	2	1	420	420
$f(\mathbf{x}) = x_1x_2 + x_3x_4$	4	1	756	756
$f(\mathbf{x}) = [x_1x_2, x_1x_3, x_2x_3]$	3	3	756	756
$f(\mathbf{x}) = [x_1, x_2] \otimes [x_3, x_4]$	4	2	700	704 (2-layer)
				1500 (3-layer)
$f(\mathbf{x}) = [x_1, x_2, x_3] \otimes [x_4, x_5, x_6]$	6	3	800	804 (2-layer)
				1700 (3-layer)

Table 4.1: A summary of the functions tested. The network structure for all but the 3-layer convolution networks is depicted in figure 4.7. Note that the number of neurons listed includes the neurons for the input, output, and error populations only; the desired output and some non-essential populations used for convenience are not included.

not in the space of spike-times, meaning that it solves the more general supervised learning problem. Additionally, unlike approaches in artificial neural networks, these networks are robust; almost all parameters are randomly generated (the tuning properties of neurons, encoding vectors, initial connection weights). The only free parameter is the learning rate, κ, which in these simulations was always 1×10^{-7}; it was not optimized for each network.

The amount of time needed to learn each function is difficult to characterize. In artificial neural network literature, learning curves are plotted as a function of the number of learning episodes. In the case of our networks, learning happens continuously; one could say that each timestep is a learning episode, and for that reason we plot the learning curves as a function of time, but because the input-output pairs are also continuously generated and randomly varying, it's possible that the networks could learn faster with better choices for the input-output pairs. Similarly, it is likely that the network would learn slower if the input and output pairs varied at a slower rate or discretely.

Nevertheless, the networks appear to learn quickly. More importantly, increasing the number of dimensions does not have a markedly negative effect on the speed of learning. Figure 4.9 quantifies this relationship. From those plots, it seems that neither input nor output dimensionality is significantly more important than the other. The total dimensionality seems to show the best relationship in the data; the best-fit line appears to fit quite well, suggesting that this method of supervised learning scales linearly in the number of dimensions, meaning that from a simulation-time point of view, it does not suffer from

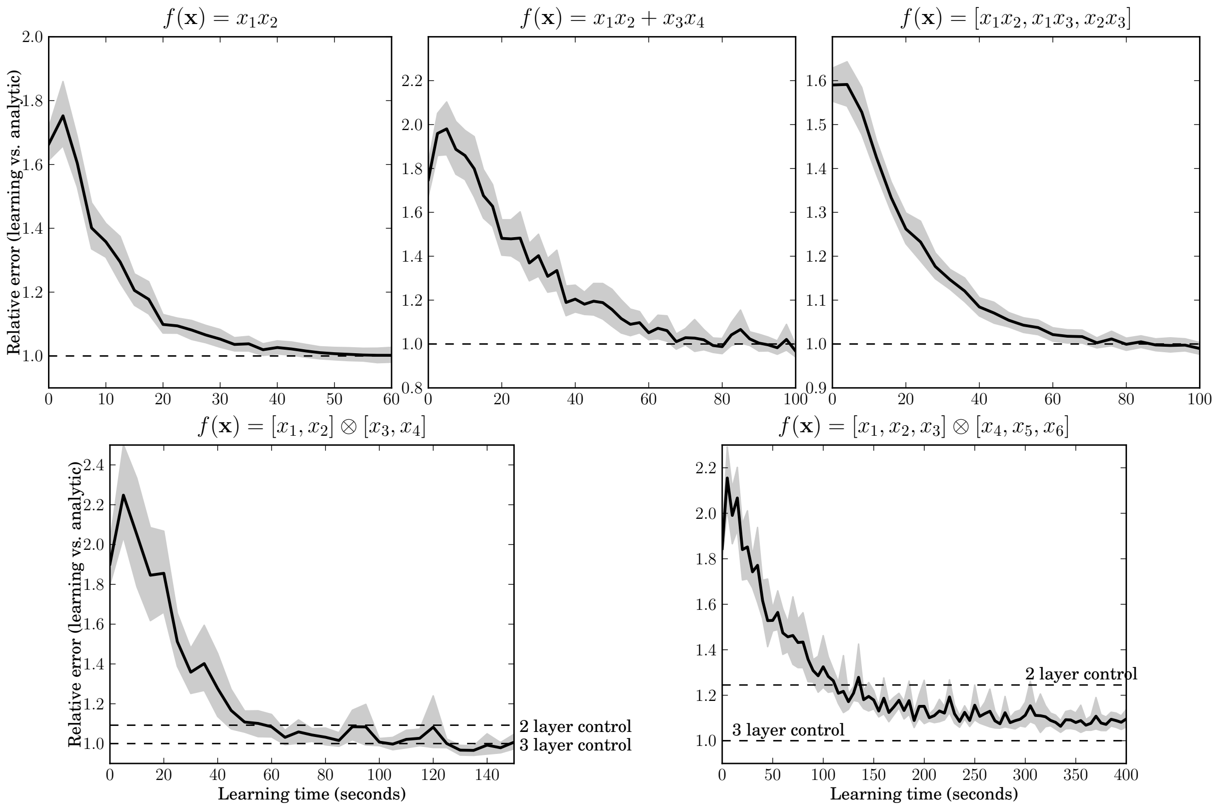


Figure 4.8: Results of the learning error's accumulated error compared to the control network's, whose connection weight matrices are determined analytically. In the case of circular convolution (bottom two plots), the results of the 2-layer control network is also compared to the 3-layer control network. Black lines represent the average relative accumulated error values. Filled grey areas represent bootstrapped 95% confidence intervals. These confidence intervals are determined by randomly choosing 40 (or 10) samples from the 40 (or 10) trails, computing the mean of those samples, and then repeating the procedure 1000 times. The bootstrapped 95% confidence interval is the interval between elements 25 and 975 of the sorted list of 1000 random samples of the data. The time indicated on the x-axis is the amount of time that the network has been allowed to learn. At 0 seconds, the `Input` and `Output` populations are connected with a completely random connection weight matrix. At 10 seconds, the network has learned for 10 seconds, but has also gone through one or more testing phases that are not included in the time represented by the x-axis.

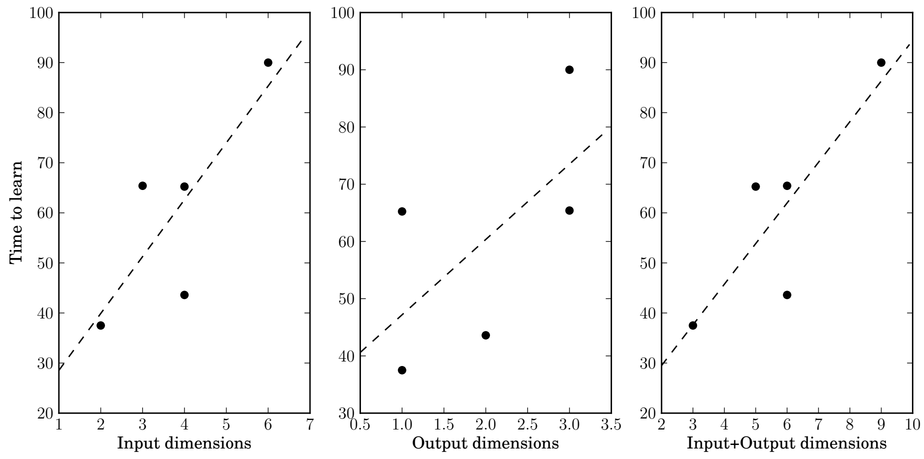


Figure 4.9: Amount of time taken to "learn" a function, as a function of input dimensionality (left), output dimensionality (middle), and total dimensionality (right). A function is considered "learned" when the 95% confidence interval reaches 1.0 relative error (see figure 4.8). The dashed line is fit to the data points.

the curse of dimensionality. The amount of computing time taken to do each simulation is not plotted, but because of the increase in the number of neurons, computationally the algorithm slows superlinearly as the dimensions increase.

We can extrapolate from the best-fit line for the plot with the total number of dimensions, and estimate how long it would take to learn a function on 500 dimensional vectors, which are the size of vectors hypothesized to be represented in the brain [59]. For a function with 500-dimensional input and output, the best-fit line predicts that it would take 8130 seconds, or just over 2.25 hours, to learn any nonlinear function, with a continuous stream of input-output examples.

An interesting result shown in figure 4.8 is that the learned 2-layer convolution networks perform significantly better than the analytically determined 2-layer control networks; in the 2-dimensional case, the learned 2-layer network performs as well as the 3-layer control. The 3-layer architecture was created because a 2-layer solution does not compute the convolution accurately enough; however, our results contradict that and suggest that connection weight matrices to do circular convolution in 2-layers exist.

4.3.4 Conclusion

In this section, we proposed a solution for the machine learning supervised learning problem, based on the principles of the NEF and learning rule (4.5). We have showed that this learning rule is analogous to backpropagation, but maintains biological plausibility as all of the elements used in the learning rule are available locally to the synapse.

Simulation results showed that our method can learn a number of complicated nonlinear functions. Additionally, in terms of simulated time, our method does not suffer from the curse of dimensionality. Our results also suggested that the NEF is capable of performing more complicated transformations in 2-layers than are currently implemented. It may be the case that solving for the weights with many more evaluation points can reach the same performance of the learned network, though it may also point to a need for improvements in the process of solving for decoding weights.

Chapter 5

Reinforcement learning

While supervised learning is a problem initially posed and solved in computer science, reinforcement learning is inspired by animal behaviour. Animals are born with a certain amount of their capabilities available, but it is clear that animals learn from their environment by taking actions in different situations and adapting their future actions based on reinforcement, either internally or from the environment. The simplest example of this is when a child first learns "the hard way" not to touch the stove when it's hot: the child sees the red-hot element, takes the action to touch it, and receives negative reinforcement in the form of a painful burn. In the future, the normally functioning child will associate touching a red-hot element with this negative reinforcement, and not perform that action in the future. Because this type of learning is dependent on some kind of negative or positive reinforcement, it is called reinforcement learning.

Early behavioural and psychological explorations of this can be best summarized by Thorndike's *law of effect* [203], which, simply stated, says that animals in a certain situation are more likely to perform actions that have produced a satisfying effect, and are less likely to perform actions that have produced detrimental effects. This idea led to many models of reinforcement learning, though the term was not used and the field generally disorganized until the pioneering work of Barto and Sutton in the early 1980s [13, 196], inspired by Klopf's work in the 1970s [106, 107]. Since then, reinforcement learning has become a large part of machine learning research.

One of the hallmarks of this research is that modelling efforts have informed neuroscience experiments, revealing that mathematical models of reinforcement learning may not be far removed from the neural implementation of learning from sparse reinforcement by trial and error [177, 179].

In the remainder of this chapter, we present the mathematical framework that has been developed by the machine learning community, discuss some simple models, and then summarize the evidence that these simple models are analogous to reinforcement learning in the brain. Finally, we present a model that shows that we can use the NEF learning rule from chapters 3 and 4 to create a biologically plausible model that bridges the gap between purely mathematical models and experimental evidence.

5.1 Reinforcement learning in traditional artificial neural networks

At the core of all work done in reinforcement learning are two ideas: the **agent-environment** interface and **Markov decision processes**.

5.1.1 The agent-environment interface

Unlike supervised learning in which everything involved in the problem is *observable* and *controllable* by the learning algorithm, in reinforcement learning a distinction is made between the elements of a problem that are controllable by the learning algorithm and which are only observable. The controllable aspects are said to be modified by an *agent*, and the observable aspects are said to be sensed through the *environment* (see figure 5.1).

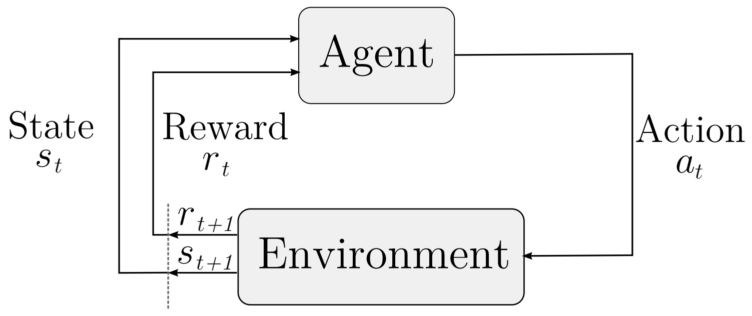


Figure 5.1: The basic agent-environment interface. Recreated from [198].

The agent and environment interact over time; this notion of time could represent the realistic passage of time, with $t = [0, dt, 2dt, ...]$, or it could be an abstraction of time, with each timestep representing the time at which an "important" aspect of the environment has changed.

In either case, at each point in time, the environment is in some state, s_t, which is one of a finite set of states, represented by $\mathcal{S}$. The agent has access to that state, and based on it takes some action, a_t, which is one of a finite set of actions available at a given state, represented by $\mathcal{A}(s_t)$. That action has some effect on the environment, which pushes it into its next state, s_{t+1}. The environment also emits a scalar reward value, r_{t+1}. The interpretation of the reward value is task-specific; in some tasks, the reward is very sparse (e.g. $r_t = 0$, except when in some goal state).

The goal of a reinforcement learning algorithm is to choose the best actions for the agent. The best action in any given state is usually defined as the one that results in the maximum future reward; that is, the agent is attempting to maximize the quantity

$$R = \sum_{t=0}^{\infty} r_{t+1}.$$

It is, however, impossible to reason about rewards delivered for all of time, so rewards are weighted such that those delivered sooner are weighted higher, and rewards delivered very far in the future are ignored entirely. To do this, we add a discounting factor $\gamma \leq 1$ to the above equation.

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1} \tag{5.1}$$

5.1.2 Markov decision processes

Given the above description, a reinforcement learning task would be intractable if the optimal strategy depended on knowing the current state and all previous states that led to the current state; this would require knowledge of an infinite combinations of state sequences.

Instead, reinforcement learning problems enforce the *Markov property*. Any process that exhibits the Markov property does not require memory. In reinforcement learning tasks, this means that a state contains all of the information necessary to make a rational decision; having memory of the states that led up to the current state gives no additional information. Stated in a different way, the next state in a reinforcement learning problem exhibiting the Markov property is completely determined by the current state, the action taken in that state, and possibly one or more random variables.

In tasks where some kind of memory of previous states affects the decision one would make in the current state, the general strategy is to include any necessary information in the state representation. Creating state representations with sufficient information but not extraneous information can be difficult, and often requires domain knowledge. For example, to learn how to act in a board game like checkers, all that is necessary is the current position of all the pieces on the board – the history of how those pieces got there is usually (though not always) irrelevant. In a more dynamic game like soccer, the position of the players and the ball is clearly not sufficient; velocity information, and possibly acceleration would be necessary to act rationally in this environment.

If the environment exhibits the Markov property, or if it can be made so with augmented states, then we can phrase the reinforcement learning problem using *Markov decision processes* (MDPs). An MDP is made up of four elements.

1. $\mathcal{S}$, the set of possible states.
2. $\mathcal{A}_s$, the set of possible actions from each state.
3. $\mathcal{P}_a(s, s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$
   The transition probability matrix. This matrix defines the probability of arriving in each possible successor state, s', when action a is taken in state s.
4. $\mathcal{R}_a(s, s') = E(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s')$
   The expected reward matrix. This matrix defines the expected value of the reward given when taking action a in state s results in successor state s'.

In reinforcement learning tasks, the agent knows $\mathcal{S}$ and $\mathcal{A}_s$, but usually does not know one or both of $\mathcal{P}_a(s, s')$ and $\mathcal{R}_a(s, s')$.

As previously stated, the reinforcement learning problem is to choose the best actions for the agent; because of the Markov property, this decision is only dependent on the current state and the agent's accumulated experience from taking actions in that state previously. We can express this as a function called the *policy*, $\pi(s)$, which outputs an action given a state. The optimal policy, $\pi^*(s)$, is one that maximizes the discounted sum of future rewards (equation (5.1)).

This mathematical framework forms the basis of most reinforcement learning research in machine learning. The reinforcement learning problem is summarized in figure 5.2.

5.1.3 Value functions

Our goal is to learn the optimal policy, or some policy very close to optimal.

Given some information about a Markov Decision Process,

$$MDP = (\mathcal{S}, \mathcal{A}_s, \mathcal{P}_a(s, s'), \mathcal{R}_a(s, s')),$$

find a policy $\pi(s) = a$ such that following that policy maximizes

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1}.$$

Figure 5.2: The reinforcement learning problem.

If we know the whole MDP, then we can find an optimal policy by repeatedly "simulating" the MDP and keeping track of which states can lead to high reward values [18]. This algorithm, called *value iteration*, involves keeping track of the value for each state, which we will represent by $V(s)$. If we have complete information about the MDP, then we can evaluate

$$V(s) = \max_a \left[\sum_{s'} \mathcal{P}_a(s, s')(\mathcal{R}_a(s, s') + \gamma V(s')) \right], \tag{5.2}$$

which is known as the *Bellman equation* for MDPs. Note the recursive nature of this equation; $V(s)$ depends on the value of the successor state, $V(s')$. Because of this, we use *dynamic programming* to solve this equation. This approach, which follows, is conceptually simple, but computationally costly.

1. Create a table that will store the value of each state in the set $\mathcal{S}$. Randomly assign initial values to each state.

2. For each state, evaluate equation (5.2); use the value table instead of evaluating $V(s')$.

3. Repeat step 2 until all $V(s)$ values remain unchanged after evaluating the equation for each state.

In practice, it is usually better to stop repeating step 2 when the overall change in $V(s)$ is some small amount. It has been proven that, even if the value function does not converge, value iteration can be used to find the optimal policy in some finite number of steps [20].

Note in equation (5.2) that we use the transition probabilities and expected reward values to determine the optimal policy. This dependence makes value iteration a *model-*

based approach to reinforcement learning. In model-based approaches, we either must know the full MDP at the outset, or learn it through trial and error.

5.1.4 Temporal-difference learning

An alternative to model-based approaches are *model-free* approaches, which aim to find an optimal policy without explicitly learning the underlying MDP. One of the first model-free approaches is called *temporal-difference learning* (TD-learning). In TD-learning, we also keep track of the value for each state, but instead of using the model to update our estimate of the value, we instead take the action that our policy recommends in the current state, and then update our estimate using the information from the environment on the next state.

Specifically, our value update function is

$$V(s_t) = V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right], \tag{5.3}$$

where α is a learning rate, which determines how closely our new estimate of the value matches what the current timestep's prediction dictates, and γ is the discount factor from equation (5.1), included because the value function represents the discounted sum of future rewards, which at time $t+1$ is discounted by γ.

Note again that this equation is recursive, in that the value of one state depends on the value of the next state. This means that we must explicitly store the value for each state, usually in a lookup table.

The value of the next state reflects our prediction of how much reward we are likely to accumulate after time $t+1$; the value of the current state is that amount plus what we expect the reward to be at time $t+1$. If we predict correctly, then we will not change the value we have assigned to the current state.

$$\begin{aligned}
&r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \\
&\quad= r_{t+1} + \sum_{i=t+1}^{\infty} \gamma^{i-t} r_{t+2} - \sum_{i=t}^{\infty} \gamma^{i-t} r_{t+1} \\
&\quad= r_{t+1} - \gamma^0 r_{t+1} = 0
\end{aligned}$$

If we have predicted incorrectly, then this term will be some non-zero value. For this reason, we call this value the prediction error, δ.

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{5.4}$$

If we have underestimated the value of the next state, then δ is positive, and we raise our estimate of the current state. If we overestimated the value of the next state, then δ is negative, and we lower our estimate of the current state.

TD-learning converges to the optimal value function given a fixed policy. But, without knowing the correct transition probabilities, we cannot use the value function alone to determine the optimal policy. To do that, we define a new value function that determines the value of state-action pairs. This function is called the Q-function.

We can adapt equation (5.3) to use state-action pairs instead of just states.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{5.5}$$

This equation is called Sarsa because it depends on the quintuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$. With this, we update Q-values with experience, and given a state, choose a rational action. Sarsa is called an "on-policy" method because it learns solely through the actions it takes while interacting with the environment.

An "off-policy" temporal-difference learning algorithm is known as Q-learning. Its value update equation is

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]. \tag{5.6}$$

Note that while we still follow the policy given by the current Q value, the value update equation uses whichever action produces the largest value in the next state, even if the policy does not choose that action (hence the "off-policy" label). That means that we can take actions that explore the space without slowing down learning.

5.1.5 Using artificial neural networks

TD-learning alone converges to the optimal state-value function [49, 196], and Q-learning converges to the optimal state-action-value function [207]; however, most early convergence proofs depended on a table-based implementation of the value functions (i.e., $V(s)$ and $Q(s, a)$ values are stored in a table). We are interested in artificial neural networks, and since $V(s)$ and $Q(s, a)$ are just specific nonlinear functions, it should be straightforward to apply the previously discussed ideas to an agent implemented with a neural network;

however, efforts to this end have met with varied success. Baird created a method called *residual algorithms* that implemented Q-learning using linear function approximators with guaranteed convergence, but it is not clear that the proof would hold for nonlinear function approximators like a multilayer perceptron [8, 198]. Doya devised a version of TD-learning that operates in continuous time and space [53, 54], and Baddeley showed that it could be applied to a multilayer perceptron, though learning was extremely slow unless a specific rehearsal strategy was used [7].

5.1.6 Comparison to supervised learning

The distinction that is commonly made between supervised learning and reinforcement learning is that supervised learning offers a constant fine-grained error signal – that is, at any point in time, the neural network knows that exactly how "wrong" it is. In reinforcement learning, the reinforcement (which could be interpreted as an error signal) is temporally sparse and of varying granularity. The reward could be as simple as a binary value – the neural network performs some function, and after a certain amount of time performing that function, it is told if it is right or wrong.

However, the MDP-based formulation of the reinforcement learning problem, summarized in figure 5.2, is very flexible. Although we typically expect to get sparse rewards with little detail, we could instead make the reward have the same detail as the error signal in a supervised learning problem. With this, it is easy to pose the supervised learning problem as a reinforcement learning problem.

It is generally accepted that the converse of that statement is not true: a reinforcement learning problem cannot be phrased as a supervised learning problem [12]. MDP model random processes, and an approach to solving them must model this stochasticity, which cannot be done in a single nonlinear function. This may explain why implementing reinforcement learning techniques using multilayer perceptrons has proved difficult, and suggests that a neural implementation of reinforcement learning will require a more complicated architecture than the one in figure 4.7. We use the complexity of reinforcement learning over supervised learning when discussing previous neural implementations of reinforcement learning.

5.2 Reinforcement learning in spiking neural networks

Despite being a more difficult problem than supervised learning, the literature for reinforcement learning in experimental and theoretical neuroscience is vast and rapidly developing.

This section presents experimental evidence that several areas of the brain encode a signal that is closely related to temporal-difference reward prediction error (see equation (5.4)). We will examine previous neural models of reinforcement learning motivated by these experiments, and then present our own model that replicates the behaviour and single-cell recordings of rats in a simple reinforcement learning task.

5.2.1 Dopamine may encode TD-like reward prediction error

The origins of reinforcement learning are in the classical conditioning experiments of Pavlov, who showed that animals can be trained to respond in certain ways to arbitrary stimuli [175]. The classic experiment performed by Pavlov paired an auditory tone with food powder delivery to hungry dogs. After repeated pairings of the tone with food, the dogs started to salivate after the tone, but before the food was delivered. We call the auditory tone the *conditioned stimulus* (CS), and the delivery of food the *unconditioned stimulus* (US). The US usually has some intrinsic positive or negative value to the animal, while the CS is arbitrary; classical conditioning can be used to transfer some aspect of the US to the CS. The dogs salivating after the tone but before the food was delivered indicates that the dogs responded to the tone by predicting the delivery of food, which produced a autonomic response. In classical conditioning, the animal does not have to perform an action to ensure that the US is delivered, so it is essentially learning to associate two stimuli; this is sometimes referred to as learning stimulus-stimulus (S-S) associations.

Operant conditioning (sometimes called instrumental conditioning) is a similar paradigm for teaching animal behaviour, except that after the CS, or instead of the CS, the animal takes some action in order to receive the positive US, or avoid the negative US [189]. This is more related to Thorndike's law of effect; instead of learning S-S associations, the animal learns stimulus-response (S-R) associations. Operant conditioning can be further subdivided into *habits*, which are simple S-R associations, and *goal-directed* behaviour, in which the animal also learns the outcome of the response (S-R-O associations), and modifies its response to the stimulus based on whether the outcome is desirable. The difference between learning habits and goal-directed behaviours maps well onto the reinforcement learning distinction between model-free and model-based learning, respectively.

While the behavioural mapping seems clear, a neurological mapping was not elucidated until Wolfram Schultz et al. showed evidence in the mid-90s that the activity of dopaminergic neurons in the midbrain appeared to encode something very similar to TD reward prediction error (δ from equation (5.4)) [141, 177, 179, 180]. Figure 5.3 summarizes the main result, which is that the dopaminergic neurons initially respond to a juice reward, indicating that they are sensitive to appetitive events. With repeated associations of the CS and juice reward, the dopamine neurons shift from responding to the actual appetitive event to the conditioned stimulus. Further, when the CS does not result in the predicted reward, there is a dip in dopaminergic neuronal activity when the reward would normally be delivered.

Temporal-difference reinforcement learning can explain this activity. We can think of the animal as the agent in a reinforcement learning situation. The value of an arbitrary state is some baseline value, with small changes due to internal states and environmental changes, as indicated by the baseline activity of the neuron in figure 5.3. The time of the CS is t^{CS} and the time of the reward is t^{US}. During initial pairing, the delivery of reward at t^{US} increases the value of the state associated with that time; i.e., $V(s_{t^{US}})$ increases. As the two stimuli are repeatedly paired, the value of $s_{t^{CS}}$ also increases if we use a temporal difference learning rule like equation (5.3), as the difference $V(s_{t^{US}}) - V(s_{t^{CS}})$ is positive. Over time, the value of $V(s_{t^{CS}})$ increases, and because the CS occurs at an arbitrary point in time, it is not predicted by a previous state, and so its prediction error is positive, and dopamine neurons activate. When the US is not delivered, then the prediction error is negative, and decreased activity in dopamine neurons is observed.

One important thing to note is that dopamine neurons do not gradually shift their activity back in time to different points between t^{CS} and t^{US}; activity shifts directly from t^{US} to t^{CS}. Further, if the US is delayed by even a small amount of time, dopamine neurons show increased activity after the stereotypical dip in activity where the reward was predicted [130]. This indicates that some part of the brain keeps track of time, and possibly associates it with states. That is, when the CS occurs, or the appropriate action is taken, the animal does not transition to the state associated with the US until the appropriate amount of time has passed. Encoding this temporal information presents many difficulties to neural models of reinforcement learning in the brain.

It is also important to note that the prediction error theory of dopamine is only one of a handful of theories concerning what information the dopamine system encodes. Other roles attributed to the dopamine system include incentive salience [19], uncertainty [63], and motivation [160]. However, in our domain, which is in changing behaviour in a simple task based on conditioning with reward, Tsai et al. showed that phasic dopamine firing is sufficient for behavioural conditioning, suggesting that while dopamine may play many

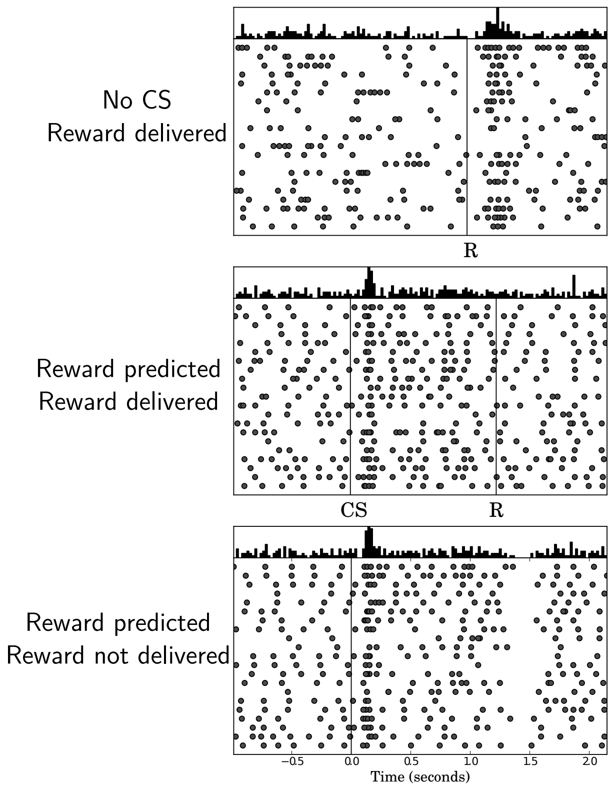


Figure 5.3: The activity of dopamine neurons appears to code for reward prediction error. Each plot is the response of one neuron over multiple trials, with the peri-stimulus time histogram plotted above. (Top) With no conditioned stimulus (CS) delivery of juice reward causes firing of the dopamine neuron. (Middle) After repeated pairings of a CS with juice reward, the dopamine neuron now spikes in response to the CS, rather than the reward. Note that as the animal learns, the dopamine response does not travel back in time from the time of reward to the CS; over time, the neuron responds less at the time of reward and more at the time of the CS. (Bottom) In the absence of reward, the neuron does not fire when reward would normally be delivered, indicating that the reward is predicted at a specific time. Recreated from [179].

roles, one such role is in this type of learning [206].

Dopaminergic neurons exist in the ventral tegmental area (VTA) and substania nigra pars compacta (SNc). Neurons from these areas project through dopaminergic pathways primarily to the frontal cortex, hippocampus, and the basal ganglia, particularly the striatum (see [28] and figure 5.4). There have been many attempts to map the elements of the reinforcement learning problem (figure 5.2) to these areas and area with projections to them (see [130] for a review). Of particular importance to the current study are theories positing that the basal ganglia does action selection, and that dopamine projections to the striatum inform action selection by changing the values assigned to states and actions [96, 103].

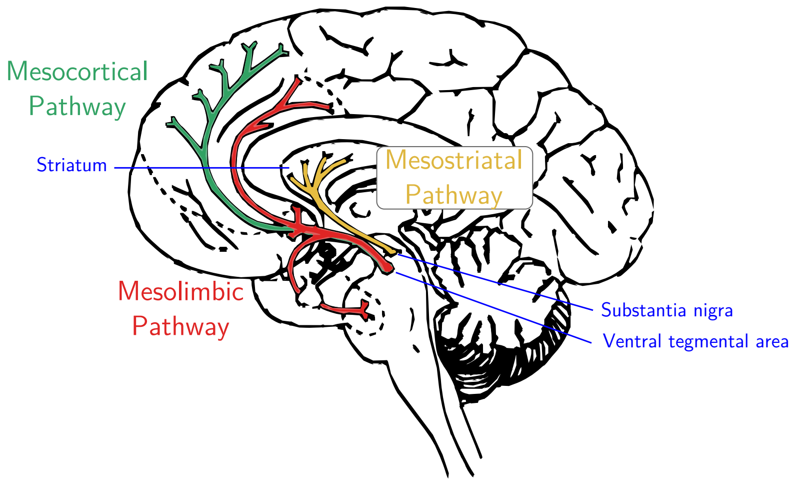


Figure 5.4: The three primary dopaminergic pathways in the brain. Note the strong connections to striatum and frontal cortex. Recreated from [28].

5.2.2 Previous neural models

Many models have been proposed that map reinforcement learning ideas onto brain areas, or use neuroscientific evidence to inform machine learning models (e.g. the actor-critic architecture [11, 81, 96, 103, 194, 195], Neural-fitted Q-iteration [169], and others [48, 66, 98]). However, as we are concerned in this thesis with maintaining biological plausibility, in this section we review only those models that use spiking neural networks.

RL through reward-modulated STDP

The majority of spiking neural network models of reinforcement learning use a reward-modulated spike-timing dependent plasticity rule to reinforce more strongly those neurons active in a certain desirable state.

Florian developed a reinforcement learning technique based on a previous non-spiking approach that did online optimization of average reward in partially observable MDPs (POMDPs), which are like MDPs except that the agent cannot directly observe the state s_t and instead must infer the state given some observations [64, 65]. By applying the ideas of the previous non-spiking approach, they arrived at a learning rule that is essentially the product of a simple STDP curve (see figure 2.7) and a global reward signal. While the algorithm is biologically plausible, including a stochastic firing threshold, simulation results are unconvincing as a proposed method of reinforcement learning in the brain. In an attempt to learn a network to solve the XOR problem, one of the simplest nonlinear functions, and a supervised learning problem (which, as discussed in section 5.1.6, is an easier problem than reinforcement learning), the algorithm took many learning episodes to convincingly solve the task. The temporal coding scheme used represented each binary value as a 500ms long spike train, which calls into question whether the algorithm could operate in a situation with sophisticated representations. Baras and Meir derive a very similar rule, also inspired by reinforcement learning solutions to POMDPs, and showed similar results on the XOR task [9]. They also showed results from a path learning task, but this problem can also be reduced to a supervised learning problem, and thus does not warrant further investigation as a solution to the reinforcement learning problem.

Farries and Fairhall used an error modulated STDP rule to train a network to take an arbitrary input spike train and produce a target output spike train [60]. While they pose their algorithm in the context of reinforcement learning, this problem is clearly equivalent to the supervised spike-time learning problem (figure 4.5). Further, their algorithm, like other solutions to supervised spike-time learning, is not scalable; it can only map from an input spike train to a target output spike train with five spikes or less.

Izhikevich used a reward-modulated STDP rule to solve a temporal credit assignment problem [92]. He used a straightforward learning rule that was the product of an eligibility trace that accumulates the effects of STDP, and the amount of extracellular dopamine. The eligibility trace determines what kind of weight changes one would expect from typical STDP experiments, but the weight is not changed until reward is delivered in the form of extracellular dopamine; the dopamine acts as a gate to learning, which is consistent with activity in some areas of the brain (see section 2.3). In one experiment, 100 different subsets of neurons are stimulated at arbitrary times and reward delivered only after one of the 100 stimuli; this stimulus causes much more spiking activity, indicating that the stimulus now predicts the reward (see figure 5.5).

However, we note that this experiment is dependent on the speed of decay of the eligibility trace (see fig 5.6); in these experiments, the reward must be delivered within 3 seconds of the CS or synaptic strengths will not change. Izhikevich also showed that the

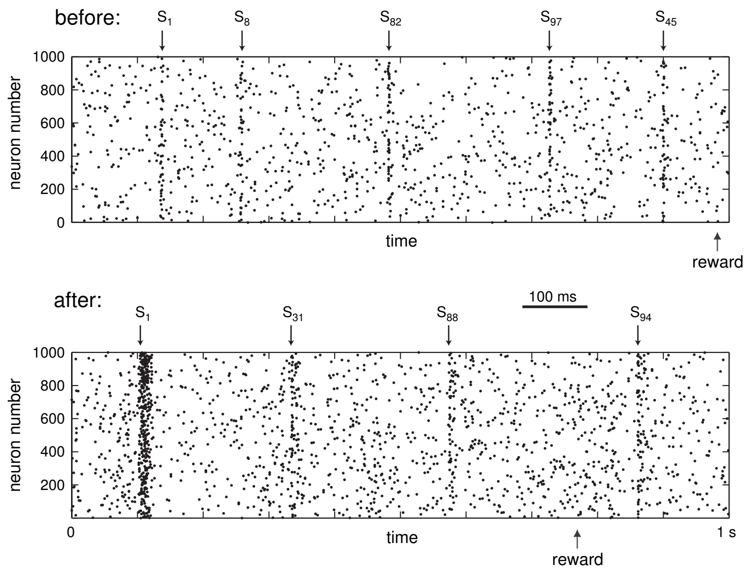


Figure 5.5: Conditioning the network to respond preferentially to stimlus S_1, but not any of the other 99 stimuli. (Top) Before conditioning, the response to S_1 is the same as any of the other stimuli. (Bottom) After conditioning, there is a larger response to S_1 over any of the other stimuli. From [92].

network shifted its strongest response from the US to the CS using the same learning rule. The only element missing from from Izhikevich's model is the decrease in activity when the CS predicts reward that is not delivered; however, Chorley and Seth extended this model and were able to simulate the decreased activity from non-delivered predicted reward [43]. These models propose a good mechanism for classical conditioning, but it is not clear that it could be used for models with highly structured representations of states, nor does an extension for model-based reinforcement learning seem possible. It is also, as mentioned before, limited by the rate of decay of the eligibility trace.

A final perspective on STDP-based implementations of reinforcement learning is the view that STDP itself implements temporal difference learning. Roa and Sejnowski showed that modelling dendritic backpropagating action potentials in a detailed biophysical model and changing synaptic weights according to the temporal difference in the postsynaptic membrane potential yields STDP [165]. Kolodziejski et al. provided a mathematical proof that differential Hebbian learning (i.e., equation (2.3)) modulated multiplicatively by a third term is equivalent to TD-learning; however, the approach has not yet been applied in a convincing simulation that shows that differential Hebbian learning can solve a reinforcement learning task [109].

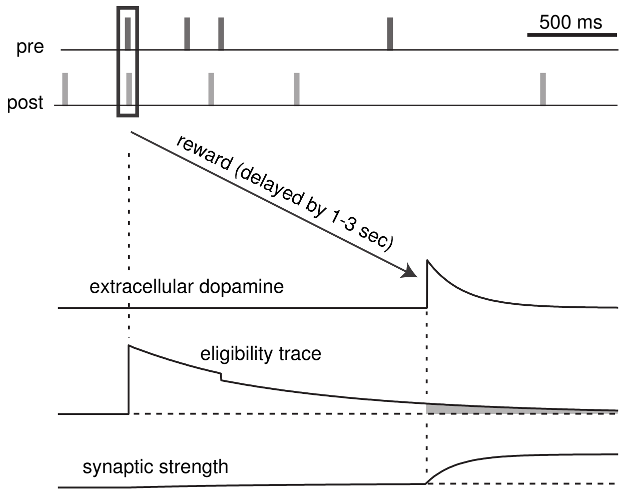


Figure 5.6: A demonstration of how Izhikevich's learning rule works. The eligibility trace accumulates the effects that would occur with normal STDP (see, for example, the dip after a post-pre pairing). The delivered reward allows the plasticity to occur, raising the synaptic strength [92].

As a general commentary on the biological plausibility of these error-modulated STDP models, all of the models summarized in this section used simple pair-based STDP, which does not capture all of the observed STDP effects in the brain (see section 2.1.3). None of the studies mention the possibility of extending the model with a triplet-based rule, which would assuage these concerns.

Further, the models that did simulate reinforcement learning tasks utilized a global reward signal. Schultz provides for the justification for using global reward; in [178] he argues that there are between 300-400 times more striatal neurons than dopamine neurons, and the population response of those dopamine neurons is relatively homogeneous. However, as we have previously stated, phasic dopamine levels have been shown to be different in subregions of the nucleus accumbens, suggesting that phasic dopamine levels are at some level local (see [3] and section sec:sup-nef-bioplaus).

One approach that showed a clear improvement in performance using a local formulation of reward was described by Urbanczik and Senn [211]. They used population coding to encode the response of a neural network to input rather than a single neuron or a small assembly of neurons. Their approach was similar to Florian's in that input was delivered as 500ms long spike trains, and like their formulation of the XOR task, there were two possible outputs. They showed that using a population of neurons to code the output rather than a single neuron improved the speed and accuracy of learning if the learning rule incorporated a local error term that represented an individual neuron's contribution to the global error. Further, performance improved with more neurons in the output population. While a good motivator of using population coding (as the NEF does), this classification problem is reducible to a nonlinear function, meaning it can be solved with supervised learning, and therefore is not a convincing demonstration of RL in a spiking neural network.

Spiking actor-critic

Potjans et al. developed a spiking network model that can solve a nontrivial reinforcement learning task, and at the moment is the most convincing account of reinforcement learning in a spiking neural network [162]. Their network is based on the actor-critic architecture, in which action selection is done by an "actor" module, and the result of the action is evaluated by a "critic" module, modifying the future behaviour of the actor. This architecture is based on a mapping to biology in which the dorsal striatum is the "actor" module, and the ventral striatum is the "critic" [130]. This mapping has been widely explored, but there remain unanswered questions concerning what the ventral striatum encodes, and what function it performs [212].

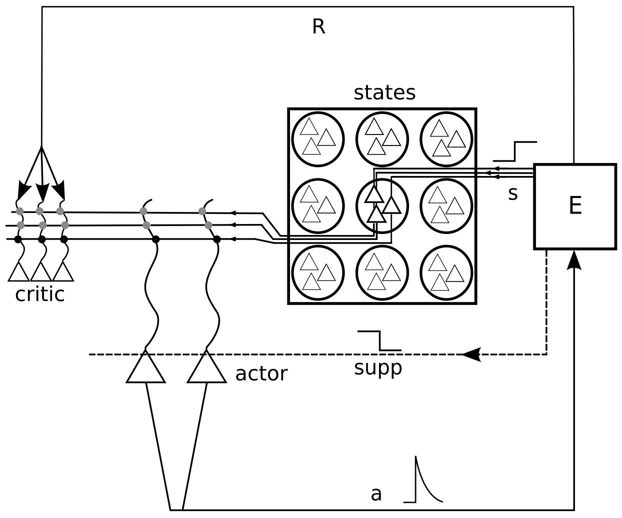


Figure 5.7: Circuit diagram summarizing Potjans et al.'s spiking actor-critic model [162].

The network structure in figure 5.7 summarizes the approach. States are represented by populations of 40 LIF neurons, and the critic by a population of 20 LIF neurons. Each action has an associated neuron, and when the environment signals a change in state by stimulating one of the "state" populations, the first action neuron to spike signals what action will be taken (this technique is seen in many other models, and is termed *first-spike coding*). The environment emits a reward that is globally signalled to critic neurons after an action is taken and the state changed.

The connections from state to critic are plastic, and represent the state-value function. Because the prediction error cannot be computed until the agent leaves the state whose value is modified, a critical period after the state transition is identified, and the state-critic projection is only plastic during that critical period (see figure 5.8). Activity traces at different timescales are used to simultaneously represent the values of the previous and current states. These elements are combined to calculate a TD prediction error, which is used directly to modify connection weights (i.e., the strength of synapses from state A to the critic population is $V(A)$).

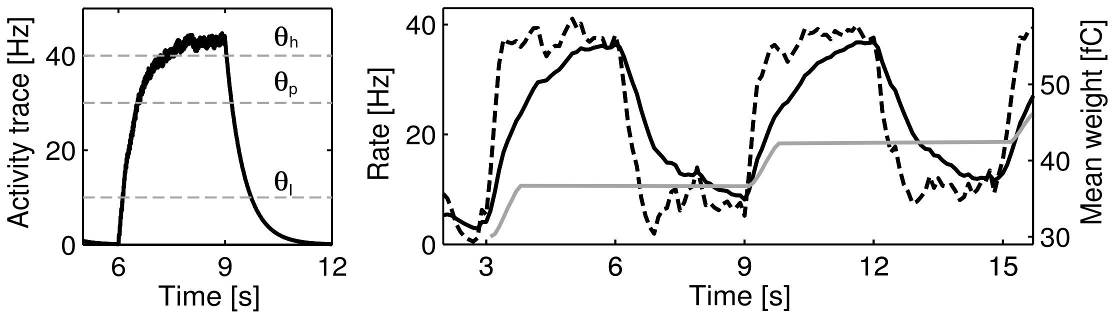


Figure 5.8: From [162]. (Left) The activity of a presynaptic neuron. At 6 seconds, the cell is stimulated; when the trace passes θ_h, it enters the "high-activity" state. At 9 seconds the stimulation is removed. When it decreases past θ_p, it enters the "plastic" state, at which time any efferent connections from this neuron can be modified. Once the activity decreases past θ_l, the neuron is in the "low-activity" state and is no longer plastic. These state transitions are unidirectional. (Right) Example of a change in a state-critic connection weight. The dashed and solid black lines represent two postsynaptic activity traces, one fast and one slow, respectively. The grey line is the synaptic strength, which increases at 3, 9, and 15 seconds in response to a state change; the synapse is only plastic for a short time, due to the effect described in the left plot.

The connections from state to actor are also plastic, and represent the policy. The changes in synaptic strength are proportional to the state-critic strength changes; when the actor neuron fires, a postsynaptic activity trace begins that defines the period in which the state-actor connections are plastic. The amount of weight change is proportional to the state-critic changes, the biological plausibility of which is argued with cited experimental findings of axonal spread of LTP/LTD of nearby synapses, though we point out that this makes the unstated assumption that actor neurons would be spatially close to critic neurons.

The algorithm reliably and quickly solves a gridworld task. As a neural implementation of the classic actor-critic algorithm, it is the most sophisticated that we found. However, one of the strengths of neural networks is the ability to generalize, and the algorithm as it is now cannot generalize due to discrete populations representing each state. Further, these states do not represent any information, only signal that the agent is in a particular discrete state; as such, this architecture could not scale up to the infinitely large space of states that an organism can find itself in. The action selection mechanism is not robust, as losing one neuron would result in no longer being able to perform an action, though the authors outline a number of more robust action selection mechanisms that could be used in the network. There are many concerns about the biological plausibility of the approach, though plausibility was not necessarily the authors' main focus.

5.3 Reinforcement learning in the NEF

In the previous section, we discussed a spiking implementation of the actor-critic architecture. A previous model implemented using the principles of the NEF implements something similar to the "actor" module. We augment this model with a "critic" module, and show that this model can replicate the behavioural and single-cell recording results of a rat in a simple reinforcement learning task.

5.3.1 Action selection

In reinforcement learning terminology, "action selection" is the problem of determining a policy given the state value function, or more commonly, the state-action value function. If the optimal state-action value function is known, then the greedy policy is optimal. When determining state-action values, following the greedy policy can result in local minima; this is often referred to as the *exploration-exploitation* tradeoff.

The basal ganglia have been hypothesized as the part of the brain responsible for action selection. A model based on experimental evidence of this hypothesis was implemented by Gurney, Prescott and Redgrave in 2001 [74, 75]. A spiking version of their model, implemented using the NEF by Stewart et al., matches behavioural and neurobiological experiments [192].

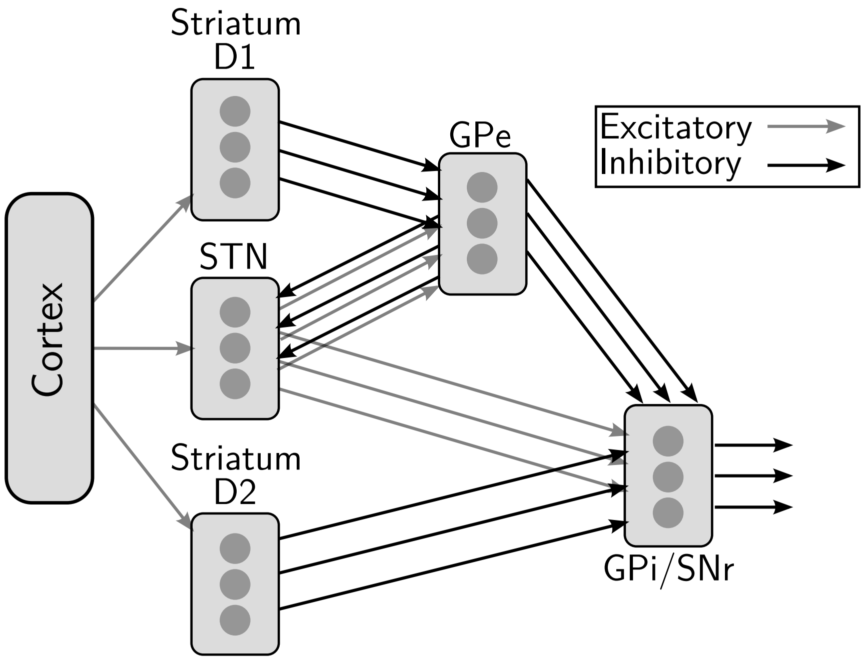


Figure 5.9: Network model of the basal ganglia, proposed by Gurney et al. [74, 75] and implemented in the NEF by Stewart et al. [192]. Performs action selection via a winner-take-all approach.

Figure 5.9 describes the structure of the network. The population of neurons labelled `Cortex` represents the state of the animal, which is hypothesized to be stored in various areas of cortex. The state-action value function, $Q(s, a)$, is computed in the `Cortex-Striatum` connections and the `Cortex-STN` connection. This computation is identical across the three different projections, though the connection weight matrices implementing the computation differ. The `Striatum` and `STN` populations encode each action's Q-value in a separate dimension. The rest of the network can be thought of as a winner-take-all circuit, doing a series of transformations on those Q-values such that the output of the `GPi/SNr` is non-zero except for the dimension that corresponds to the chosen action. The `GPi/SNr` is meant to connect through inhibitory connections to the thalamus, inhibiting all but the chosen action.

Action selection using this model can be summarized with the simple equation

$$\pi(s_t) = \arg\max_a Q(s_t, a) \tag{5.7}$$

5.3.2 Critiquing the actor

With an "actor" module as simple as the one described above, the "critic" model only needs to ensure that Q-values are updated appropriately. This is accomplished by calculating the temporal difference prediction error, δ, and using that value to drive the error-modulated learning rule previously applied to the NEF model of supervised learning, copied below for convenience.

$$\Delta\omega_{ij} = \kappa\alpha_j \mathbf{e}_j \cdot \mathbf{E} a_i \tag{5.8}$$

Despite the problem being more complicated, the use of the rule does not change; in supervised learning the error term $\mathbf{E}$ was a continuous fine-grained signal. In the reinforcement learning case, the error term is likely to be temporally sparse, but that does not affect the learning rule; this demonstrates the generality and flexibility of this rule.

To implement a critic, then, we only need to create a population of neurons that will compute the TD prediction error. Recall from section 5.1.4 that there are two different possible prediction errors, defined by the Sarsa and Q-learning algorithms, but in this case, because $Q(s_{t+1}, a_{t+1}) = \max_a Q(s_{t+1}, a)$, these are equivalent.

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{5.9}$$

Figure 5.10 shows the modified basal ganglia model. Populations representing the ventral striatum and the dopaminergic substantia nigra and ventral tegmental area were added. The ventral striatum implements the critic by calculating the temporal difference error through the projection to the dopaminergic neurons, whose output modulates plasticity on `Cortex-Striatum` synapses. This model differs from Gurney et al.'s in that they do not differentiate between the dorsal and ventral striatum; however, there is significant experimental evidence that suggests that the two have different functional roles and therefore should be separated in a functional model [130].

5.3.3 Simulation results

To test the model, we attempted to recreate the results of Kim et al.'s 2009 study of rats performing a simple reinforcement learning task [104]. This study was chosen because the task is simple but stochastic (i.e., cannot be solved through supervised learning), and because the study includes both behavioural results and single-cell recordings, both of which we aim to recreate with the proposed model.

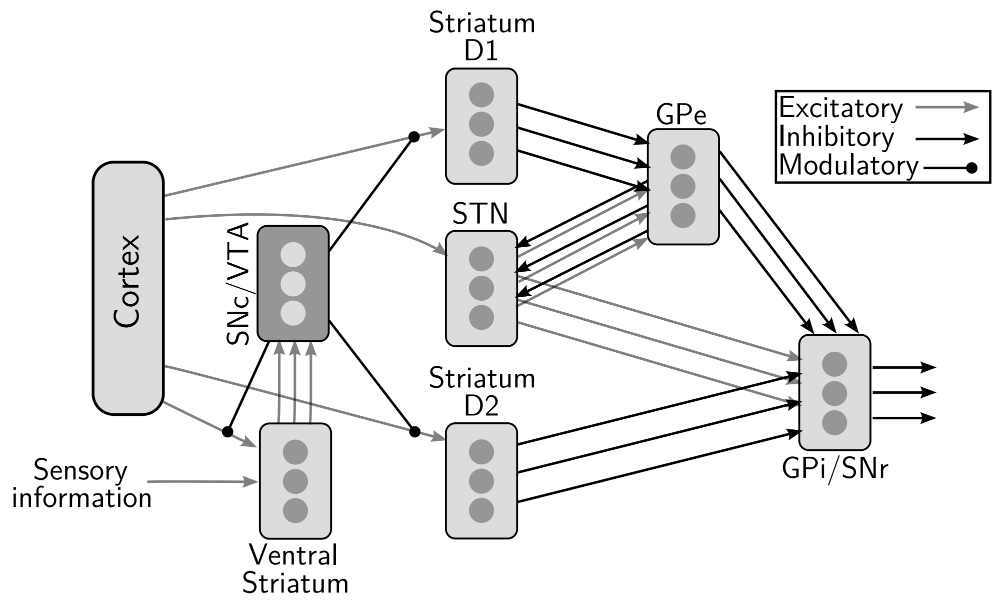


Figure 5.10: Modified basal ganglia model. The ventral striatum implements the "critic" module, getting state-action-value information from the cortex and sensory information (i.e. reward) from the environment. The temporal difference error is calculated through the projection to the dopaminergic SNc/VTA neurons, whose output modulates plasticity of cortico-striatal connections.

The specific task is a 2-armed bandit task. In this task, rats are given one decision to make, whether to turn left or right, and are given reward with a certain probability, depending on which decision was made. The task can be broken up into five stages, as described in the caption of figure 5.11.

The experiment was implemented in Nengo, software that enables quick creation and simulation of neural network models created with the principles of the NEF. The environment, described by figure 5.11, was simulated with a finite state machine. The states and transitions of the finite state machine match the MDP on the right side of figure 5.11.

The model's behaviour, compared to animal behaviour, is summarized in figure 5.12. Model behaviour matches closely the animal behaviour in the single experiment plotted in the paper. The average of all 200 trials shows that this result is not atypical; the model quickly learns to switch to the reward site with higher probability of reward in each block of trials.

Because the model is implemented in spiking neurons, we can also compare the spiking

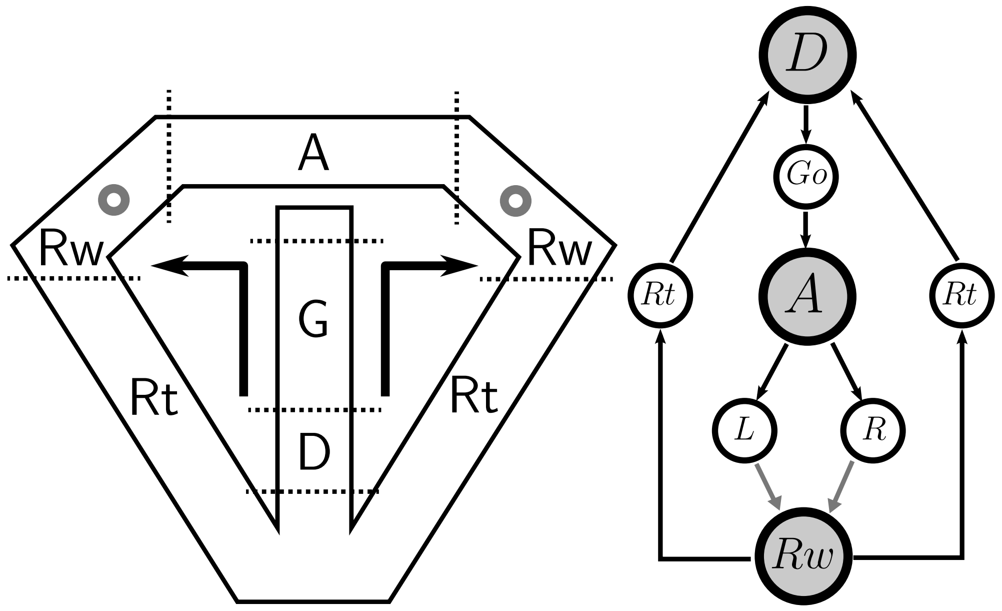


Figure 5.11: (Left) The 2-armed bandit task. On each trial of the task, the animal goes through the following five stages.

1. Animal waits (**D**elay phase).
2. Bridge lowers, allowing animal to move (**G**o phase).
3. Animal reaches decision point, either turns left or right (**A**pproach phase).
4. Reward is stochastically delivered at reward site (**R**e**w**ard phase).
5. Animal returns to delay area (**R**e**t**urn phase).

(Right) A Markov Decision Process modelling the 2-armed bandit task. Grey nodes represent states, and white nodes represent actions. Black arrows are normal state transitions, and grey arrows indicate transitions that may result in reward delivery.

activity of ventral striatum neurons in the model and in the rat. The model presented here predicts that the ventral striatum is encoding the variables necessary to compute prediction error: the current and next state-values, and the reward. The population's activity also depends on the presence or absence of reward: when no reward is delivered, the population's activity is suppressed through inhibitory gating, even if there is large disparity between the value of the current and next state. The neural results presented

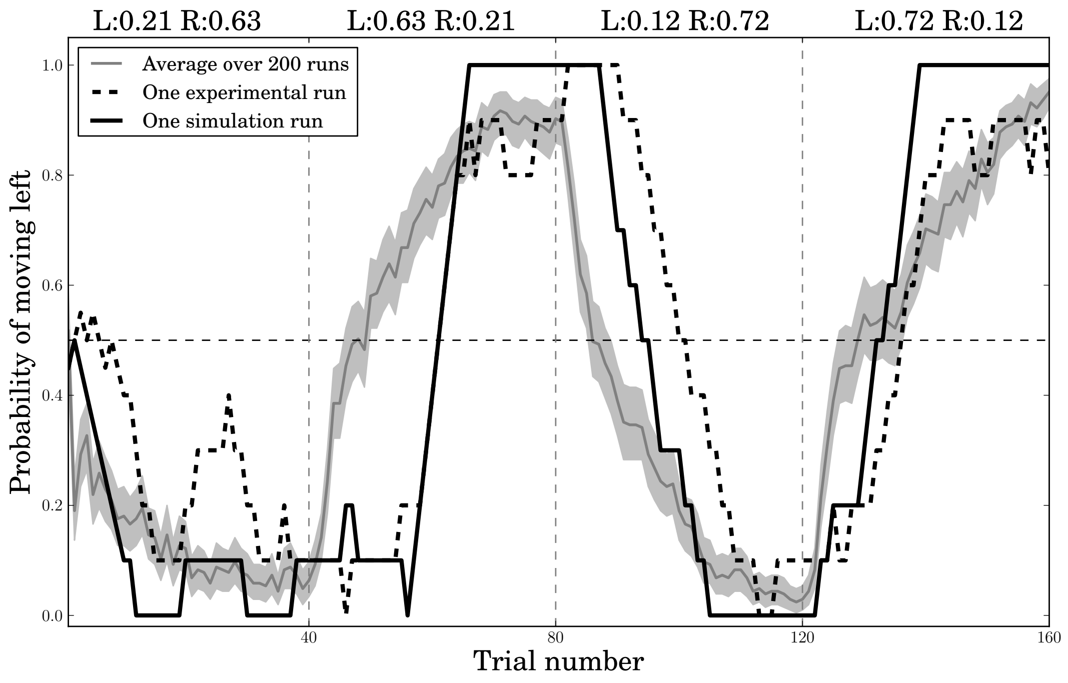


Figure 5.12: Behavioral results from real rats in a 2-armed bandit task from Kim et al., and simulated rats from the model described in this chapter. Each experiment is broken up into 40 trials, and each block of trials has a certain probability of reward for the left and right reward sites. The block reward probabilities are given in the figure title. 200 simulations were run, and the averaged results are shown in the grey line. The solid area represents bootstrapped 95% confidence intervals. The experimental run data is from [104]. The matching simulation run is the run whose mean squared error is lowest compared to the experimental data; for the run plotted, MSE = 0.0273125.

in figure 5.13 and 5.14 show that, in general, activity increases sharply during the reward phase and is low otherwise in both the model and the rat.

5.3.4 Challenges for the current model

While these results are promising, they are best described as a proof of concept that the existing basal ganglia model can be adapted to do dynamic decision making. The challenges central to reinforcement learning remain. In this section, we discuss those challenges, and propose modifications to our network that may solve them.

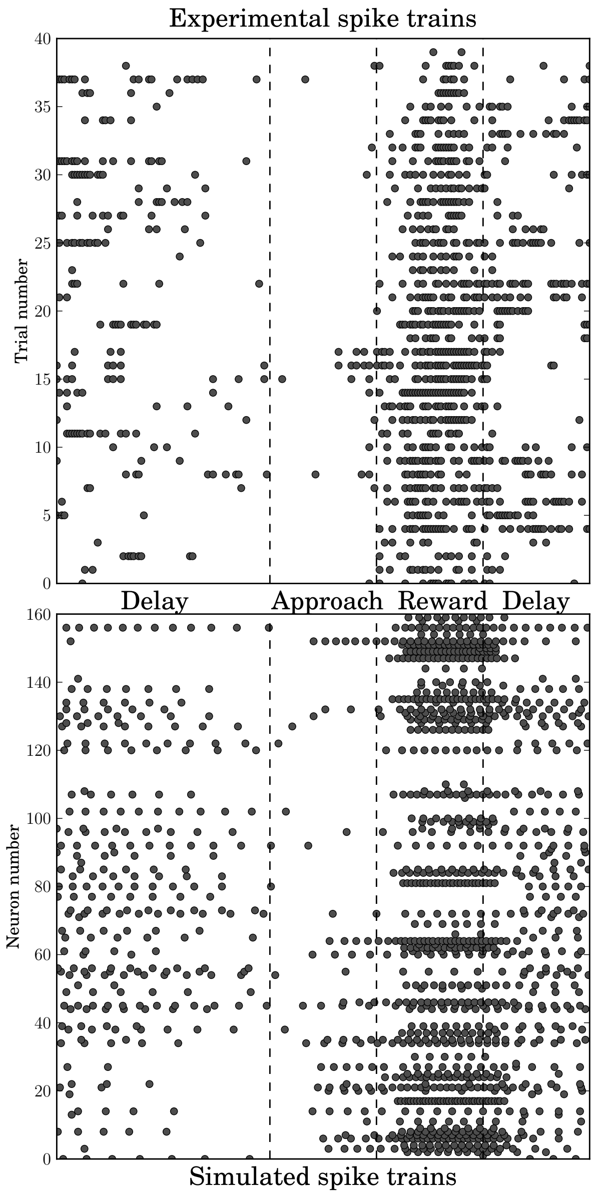


Figure 5.13: (Top) Spike data from a single cell in the ventral striatum of real rats in a 2-armed bandit task, over 40 trials, from Kim et al. (Bottom) Spike data from a population of cells in the ventral striatum of the basal ganglia model described in this chapter.

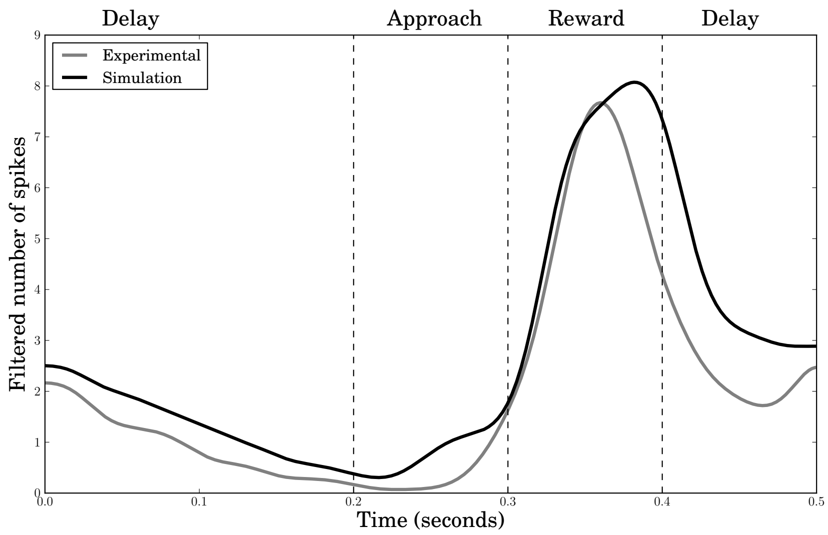


Figure 5.14: Spike trains from figure 5.13 filtered with Gaussians with mean 0 and standard deviation 0.015 seconds (15 milliseconds).

Spatial credit assignment

Generally stated, the spatial credit assignment problem is determining how to route reinforcement signals to the synapses responsible for the delivery of the reinforcement. In the case of spiking neural networks operating in continuous time, this is difficult because by the time the network has enough information to change the estimated value of a state, the agent is no longer in that state. In table-based algorithms this does not matter, but in neural networks, learning rules are activity dependent. To appropriately update connection weights, the activity must reflect the state whose value is to be changed.

Of the previous models, only Potjans et al.'s spiking actor-critic had a solution to this problem, which was to use two activity traces to track postsynaptic activity of the critic neurons. A slow activity trace remembers the value of the last state, while a faster activity trace quickly reflects the value of the current state. At a critical point in time, the values of the current and last states are represented, which can be used to compute the temporal difference reward prediction error. This solution could be implemented in our network.

However, the biological plausibility of this approach is questionable, and would not work if the reward was not delivered at the exact same time as the state transition. This is assumed in most reinforcement learning systems, but in a real-time system, this may not necessarily be the case.

We have begun an implementation of an alternative solution to this problem. The solution is inspired by other models implemented in the NEF that use recurrent connections to temporarily store values. These populations can be best described as integrators, which when driven to arbitrary points in a vector space, will attempt to remain there until disturbed by further input. We can use integrators to remember the current state, as well as the value of the current state. Once the agent transitions to a new state, the previous state and its value remains available to be used in a temporal-difference update equation. This architecture would provide the flexibility to wait until reward is delivered before updating the value of the previous state.

Temporal credit assignment

The spatial credit assignment problem discussed above is often presented as the temporal credit assignment problem. However, there is a more important temporal issue: in a real-time simulation like those discussed in this thesis, time does not move in idealized discrete steps like an MDP. While the MDP moves from s_t to s_{t+1} and so on, in real time the state represented by s_t may transition to s_{t+1} after 1 second, s_{t+2} 1 minute later, and so on. While this should not matter in general, it is clear from previously discussed experimental evidence that the brain tracks precisely the time of a predicted reward.

Izhikevich solved this problem using activity traces, but the predictive ability of the network is limited by the decay of the activity traces. In his paper, he also stated that activity occurring between the CS and US is part of the temporal credit assignment problem, though this does not appear to be an issue with our model.

Representations of time in spiking neural networks is itself a significant and growing area of research, and mechanisms for representing time, such as an integrator with constant input, could be added to the network.

However, it is our opinion that the timing issues of these spiking neural networks operating in continuous time require more formal investigation before being implemented in spiking neural networks. While the MDP is good for table-based reinforcement learning, it may be the case that spiking neural networks require more complex representations of reinforcement learning problems. Continuous-time MDPs were recently formalized [72], but have yet to be widely examined in reinforcement learning. Many solutions to exact timing problems have also been proposed in the field of real-time programming in computer science; these ideas may be well-suited to reinforcement learning in spiking neural networks.

The exploration-exploitation tradeoff

The action selection mechanism used in this network is simple and greedy, making it likely to get stuck in local minima. There are a few ways in which exploration could be implemented in the network.

Noise could be injected into the system, directly in the striatal populations. This method is biologically plausible, in that the brain is a very noisy system, though further investigation will have to be done to determine the amount of noise in various parts of the basal ganglia. This method has the additional benefit of more exploration when two actions are close in value, as large differences in value would not frequently be overcome even with large amounts of noise.

It would also be possible to explicitly drive exploration through a new population of neurons. The population would either stochastically or deterministically increase the value of some random, or somehow salient, action. This would be easy to implement, as it only requires a population providing extra input to the striatum.

5.3.5 Conclusions

In this section, we have proposed a modification to Stewart et al.'s basal ganglia model, adding a population of neurons that acts as a "critic." The critic, which we map onto the ventral striatum, calculates the TD prediction error through the projection to dopaminergic neurons, whose activity modulates a plasticity rule applied to cortico-striatal connections. This model is able to recreate the behavioural effects and neural data recorded in rats performing a two-arm bandit task.

While this model has not solved the spatial and temporal credit assignment problems, it has made rational dynamic decisions in a stochastic reinforcement learning task using a biologically plausible neural architecture. We suggest that this model is more scalable than previous models, and provides the first step toward a general architecture for solving continuous time reinforcement learning problems in high-dimensional state and action spaces.

Chapter 6

Unsupervised learning

Unsupervised learning is easily distinguishable from supervised and reinforcement learning by a lack of an error signal. Typically, it is said that unsupervised learning is the problem of finding patterns in a set of data; this leads to unsupervised learning solutions to problems like clustering, feature extraction, dimension reduction, and others.

In previous chapters discussing other machine learning problems, we attempted to build biologically plausible models to solve the same problems solved by machine learning techniques. However, in this chapter we take a different approach; instead of determining suitable learning rules to solve a problem, as machine learning does, we instead investigate the problems solved by learning rules that describe plasticity in the brain.

As will be discussed in the next section, unsupervised learning in machine learning can also be thought of as "self-supervised learning." Self-supervised techniques exist for spiking neural networks as well, but in section 6.2 we instead focus on neural models in which synaptic plasticity rules are applied to a network with no error signals. Finally, we test some of the theoretical ideas in unsupervised learning through an NEF model, and propose a novel learning rule that ties together the learning rule previously used for supervised and reinforcement learning and the unsupervised learning rules introduced in this chapter.

6.1 Unsupervised learning in traditional artificial neural networks

The machine learning unsupervised learning problem is summarized in figure 6.1. An assumption in most unsupervised learning techniques is that the input data represent samples from some underlying probability distribution, and the network attempts to infer information about that distribution [50]. For this reason, a common choice for the cost function is Kullback-Leibler divergence [114], which measures the difference between two probability distributions. A cost function like KL-divergence can be incorporated in an artificial neural network's learning rule; multi-layer perceptrons with learning rules derived from KL-divergence-like cost functions have been used as generative models [115], for clustering [142], and many other supervised learning tasks that require statistical modelling of large amounts of data (see [151] for a review).

$$\boxed{\begin{array}{c}\text{Given input } \mathbf{X} \text{ and cost function } C(\mathbf{x},\mathbf{y}),\\ \text{generate output } \mathbf{Y} \text{ and minimize } C(\mathbf{x},\mathbf{y}).\end{array}}$$

Figure 6.1: The unsupervised learning problem.

In machine learning, this problem is considered unsupervised because there is no explicit error signal; however, the cost function provides a way for the network to implicitly determine an error signal. The main difference between supervised learning and unsupervised learning in machine learning is that in unsupervised learning the computation of the error signal is embedded in the learning rule, while in the supervised case the learning rule is flexible and accepts an arbitrary signal as the error. We can determine a learning rule for any arbitrary cost function; in the brain, however, the learning rules are already determined, and the question is what cost function they minimize. Because the cost function is available by definition for this problem in machine learning, we leave the discussion of unsupervised learning in artificial neural networks at this very general level.

6.2 Unsupervised learning in spiking neural networks

Unsupervised learning in machine learning is essentially the process of finding a learning rule that will minimize a cost function. We consider the neural unsupervised learning problem to be the opposite: experimental evidence has suggested several learning rules, so we attempt to determine the cost function that the learning rule is minimizing.

Previous models in which a biologically inspired learning rule is applied to a network in an unsupervised fashion have suggested a number of possible cost functions, or in many cases, behavioural consequences, which we relate to possible cost functions. Additionally, this section will introduce learning rules that describe the plasticity mechanisms that were discussed in chapter 2.

6.2.1 Artola, Bröcher, Singer (ABS) rule

The Artola, Bröcher, Singer (ABS) rule, proposed by Hancock et al. [80], models the results found by Artola, Bröcher and Singer in rat visual cortex [4]. They found that direction of synaptic weight change was dependent on the rate of postsynaptic spiking. The simplest learning rule that captured their qualitative results is

$$\Delta\omega_{ij} = a_i\sigma(a_j), \text{ where } \sigma(a_j) = \begin{cases} \Delta^+ & \text{if } a_j \geq \theta^+ \\ \Delta^- & \text{if } \theta^- < a_j < \theta^+ \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

$\sigma(a_j)$ is a simple filter of the postsynaptic activity. The filter from equation (6.1) is plotted in figure 6.2. In Hancock et al.'s original paper, this version of the ABS rule is applied to a feedforward network learning to associate random patterns, and it is argued that the rule is useful for correcting errors.

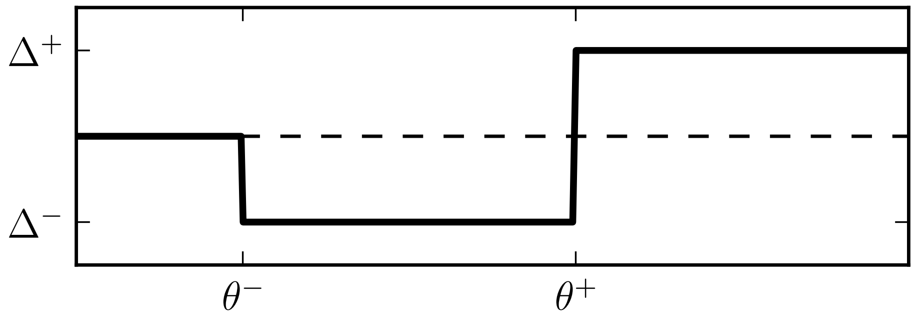


Figure 6.2: The simple version of the ABS rule.

Artola and Singer provided a version of the ABS rule in which the direction and magnitude of weight change is dependent on postsynaptic activity – in their case, they consider the postsynaptic neuron's membrane potential as its activity, a_j [5]. Their version of the rule can be seen in figure 6.3. Of particular importance is the ability for the rule's two thresholds to change as a function of the presynaptic activity of *other* cells connected to the presynaptic cell. This modification is meant to model heterosynaptic LTD (see figure 2.5), which few other learning rules do.

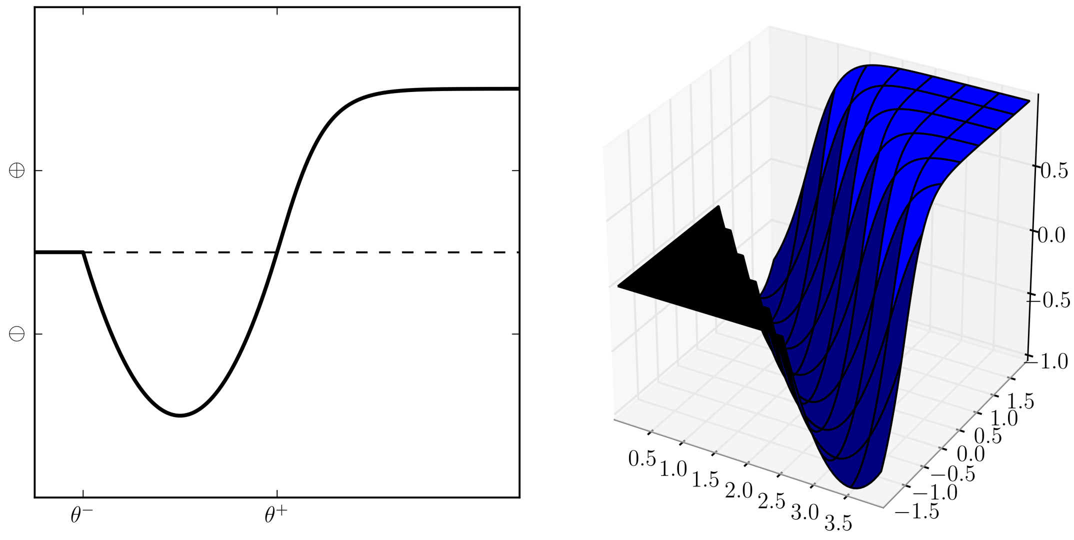


Figure 6.3: A more complicated version of the ABS rule in which the amount of potentiation or depression differs based on the exact time since a postsynaptic spike. (Right) The thresholds θ^- and θ^+ are modified by presynaptic activity. As the presynaptic activity increases (right axis) the ABS curve (Left) is shifted to the left.

The ABS rule, however, has not seen wide use in the modelling community, and we can find no studies that investigate its functional significance in a spiking neural model. This may be due to it having few functional benefits, or because it is very similar to the BCM rule that was proposed earlier, only without guaranteed stability. The BCM rule is examined in the next section.

6.2.2 Bienenstock, Cooper, Munro (BCM) rule

Bienenstock, Cooper and Munro proposed a learning rule in 1982 that models biological LTP/LTD findings [23]. It is similar to the ABS rule in that it filters postsynaptic activity with a curve that has an initial negative component followed by a positive component; unlike ABS, however, there is only one threshold, which we call θ. Only one threshold is needed because with any postsynaptic activity, there will be some weight change, either positive or negative (unless $a_j = \theta$); this contradicts the findings of Artola et al. [4], but matches other experimental findings and simplifies the model. Like the more complicated version of the ABS rule, the threshold changes, in this case as a function of the expected value of the postsynaptic activity. In other words, the threshold carries some information about the history of the cell's activity, and deviations from its historical firing patterns result in synaptic weight changes.

The original mathematical description of the BCM rule is below. Its postsynaptic filter is plotted in figure 6.4.[1]

$$\begin{aligned} \Delta\omega_{ij} &= a_i a_j (a_j - \theta) \\ \theta &= E[a_j/c], \end{aligned} \tag{6.2}$$

where c is some constant scaling factor, and $E[\cdot]$ is the expected value. The expected value is meant to be the average response over all possible input patterns, but in practice this is estimated well enough by a temporal average over a sufficiently long timescale. Other variants of the BCM rule exist (e.g. Intrator and Cooper's version [89]), with slightly different BCM curve shapes and rules for modifying the threshold, but the general idea in all BCM learning rules is the same.

Looking at the curves, we can see that the qualitative aspects of LTD/LTP induction could be satisfied in certain conditions, mainly if the neuron's current threshold is at an appropriate level. If the neuron is typically moderately active, then an experimental setup that stimulates the cell with low frequency stimulation would result in LTD, and high frequency stimulation would result in LTP, as would be expected. Some counterexamples to these basic forms of LTD/LTP could be explained with arguments about the neuron's typical activity compared to the experimental procedure.

This, however, raises the question of whether the threshold θ is a biologically plausible element intrinsic to a cell. It could be hypothesized that each cell has some fixed threshold, but the adaptive threshold posited by BCM requires experimental verification. Kirkwood et al. provided that verification by showing that light-deprived rats showed more LTP and less LTD in visual cortex compared to rats raised in normal light environment (see [105] and figure 6.5). This finding supports the idea that the threshold at which LTD switches to LTP is not fixed. Wang and Wagner showed a similar result in rat hippocampus [24, 215]. Bear proposed that a possible neurobiological mechanism for the sliding threshold is the calcium dependent protein kinase CaMKII [14].

[1]The original formulation of equation (6.2) included a decay term, $\epsilon\omega_{ij}$. We do not include it as we have not considered decay in any learning rules to this point, nor do we from this point forward.

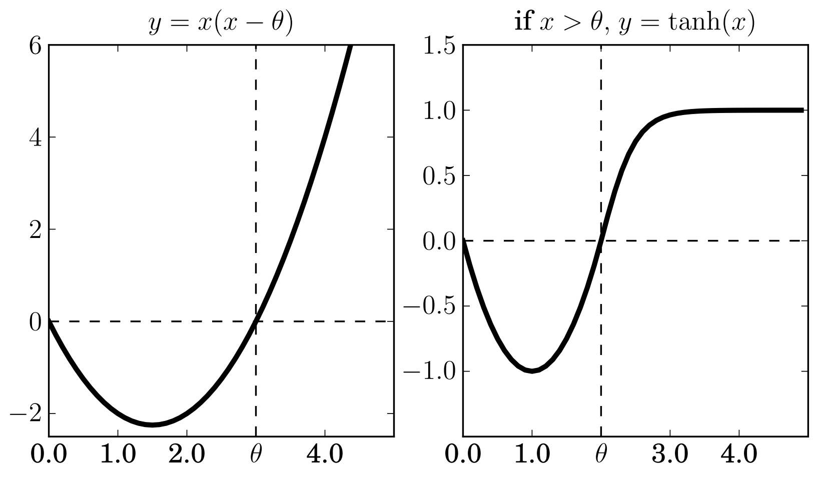


Figure 6.4: The postsynaptic activity filters for two different BCM rules. (Left) The original BCM rule proposed in [23]. Note that in the original BCM formulation, the positive portion of the curve will continue off to infinity. (Right) A different version of the BCM rule in which the positive portion has a definite maximum due to the use of the tanh function.

Early models using the BCM rule focused on the unsupervised development of orientation selectivity and ocular dominance in visual cortex [116, 181]. In those models, a representation of the visual field is projected through two populations of input neurons, one from each eye, to an output population. The connections between the input and output populations begin with small random values, and evolve according to a BCM rule like equation (6.2). After a sufficient amount of training, the output of the output population looks similar to what is observed in visual cortex: neurons are sensitive to small areas of the visual field, known as their receptive fields, and information in that visual field is filtered with Gabor filters with certain orientations. Ocular dominance is also observed, meaning that the second population is, in general, more sensitive to input from one eye than the other.

A non-neural unsupervised learning technique applied to the same network is Independent Component Analysis (ICA) [201]. ICA attempts to identify components of a vector such that output components are statistically independent. One implementation of ICA for nonlinear data models used a multi-layer perceptron and a learning rule using KL-divergence as the cost function [99]. Other implementations of ICA applied to the development of visual cortex have used functions measuring kurtosis (a statistical measure that is high when a probability distribution has a high peak and heavy tails) and skewness

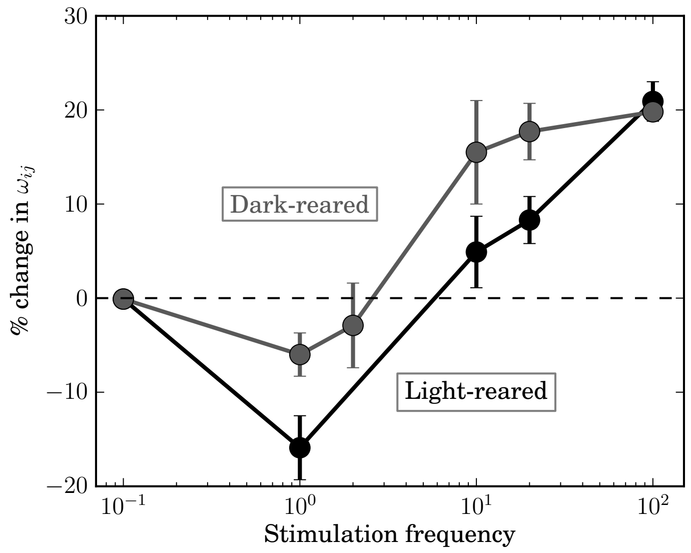


Figure 6.5: Experimental evidence supporting evidence of a modifiable threshold that can change the direction and magnitude of synaptic strength changes. In this experiment by Kirkwood et al. [105], rats deprived of light (i.e., with lower average postsynaptic activity) were more readily potentiated than those raised in normal light conditions.

(a measure of the asymmetry in a probability distribution) as cost functions [25]. This suggests that there are a number of different possible cost functions that the BCM rule may be implicitly minimizing.

Intrator and Cooper applied a BCM rule to the more general unsupervised learning problem of projection pursuit, in which the goal is to find the most statistically "interesting" projection in a multidimensional set of data, where "interesting" usually means non-Gaussian [89]. In their study, they derived a rule that follows the principles of BCM theory using an explicit cost function that can be thought of as a measure of sparsity of the output distribution. This suggests yet another cost function for the BCM rule, and another possible functional use.

All of the models discussed to this point have used non-spiking neurons, though many of the neuron models are biologically inspired. Our interest is in unsupervised learning in spiking neural networks; fortunately, most of the BCM rules (like equation (6.2)) can be adapted for use in a spiking neural network by changing the activities of cells from firing rates or idealized scalar values to filtered spike trains, with the exception of Intrator and Cooper's rule, which uses the derivative of the neuron's activation function in the synaptic update rule [89].

One study that used a BCM rule with spiking neurons was Toyoizumi et al.'s, which proposed a learning rule that minimized KL-divergence under the constraint of attempting to maintain a target postsynaptic firing rate [205]. That a BCM rule was found under this additional constraint suggests that homeostasis could be another functional effect of a BCM rule. The study confirms that the rule can be applied to spiking neural networks.

6.2.3 Spike-timing dependent plasticity rules

The history and experimental exploration of STDP was previously discussed in section 2.1.3. Below we present learning rules that model STDP effects phenomenologically. Early models using pair rules include Song et al.'s [187], and triplet rules were first mathematically characterized by Pfister and Gerstner [158].

The first STDP rules were based on the interactions between pairs of spikes: a pre-post pairing causes potentiation and a post-pre pairing causes depression. This can be expressed with the following two equations

$$\begin{aligned}\Delta\omega_{ij}(t^{pre}) &= A^- \exp\left(\frac{t^{post_l} - t^{pre}}{\tau^-}\right)\\ \Delta\omega_{ij}(t^{post}) &= A^+ \exp\left(\frac{t^{pre_l} - t^{post}}{\tau^+}\right),\end{aligned} \tag{6.3}$$

where t^{pre} is the time of a presynaptic spike, t^{pre_l} is the time of the *last* presynaptic spike, and similarly for t^{post} and t^{post_l} for postsynaptic spikes. A^- is a (negative) constant representing the maximum amplitude of post-pre depression, and τ^- is the time constant controlling its exponential decay; similarly, A^+ and τ^+ define the positive pre-post part of the STDP curve. Note that this learning rule, unlike all others that we have examined in this thesis, is applied only at the times of pre- and postsynaptic spikes, rather than on every timestep.

This learning rule only takes into account the most recent presynaptic and postsynaptic spike; however, we know that different frequencies of activity affect plasticity. A first step toward modelling this is to incorporate more than one spike by using activity traces for the pre- and postsynaptic activities. This results in the following set of differential equations.

$$
\begin{aligned}
\frac{do(t)}{dt} &= -\frac{o(t)}{\tau^-}, \text{ if } t = t^{post} \text{ then } o(t) = o(t) + 1 \\
\frac{dr(t)}{dt} &= -\frac{r(t)}{\tau^+}, \text{ if } t = t^{pre} \text{ then } r(t) = r(t) + 1 \\
\Delta\omega_{ij}(t^{pre}) &= A^- o(t) \qquad \Delta\omega_{ij}(t^{post}) = A^+ r(t)
\end{aligned} \tag{6.4}
$$

However, just adding activity traces does not capture the effects of experiments like [122, 186] that varied the frequency of pre-post pairings. As discussed previously, incorporating a third spike and adding rules for pre-post-pre and post-pre-post interactions is sufficient to replicate the experimental results. Using only nearest spike interactions, this results in

$$
\begin{aligned}
\Delta\omega_{ij}(t^{pre}) &= -\exp\left(\frac{t^{post_l} - t^{pre}}{\tau^-}\right)\left[A_2^- + A_3^- \exp\left(\frac{t^{pre_l} - t^{pre}}{\tau^x}\right)\right] \\
\Delta\omega_{ij}(t^{post}) &= \exp\left(\frac{t^{pre_l} - t^{post}}{\tau^+}\right)\left[A_2^+ + A_3^+ \exp\left(\frac{t^{post_l} - t^{post}}{\tau^y}\right)\right].
\end{aligned} \tag{6.5}
$$

Note that in the presynaptic rule, we now put the negative sign in front of the equation and let the amplitude constants A_2^- and A_3^- be positive values. Also note that this rule is a generalization of the pair-based rules, as setting the triplet amplitude constants $A_3^- = A_3^+ = 0$ results in equation (6.3).

To do all-to-all interactions instead of nearest spikes, we introduce two additional activity traces to equation (6.4).

$$
\begin{aligned}
\frac{do_1(t)}{dt} &= -\frac{o_1(t)}{\tau^-}, \text{ if } t = t^{post} \text{ then } o_1(t) = o_1(t) + 1 \\
\frac{do_2(t)}{dt} &= -\frac{o_2(t)}{\tau^y}, \text{ if } t = t^{post} \text{ then } o_2(t) = o_2(t) + 1 \\
\frac{dr_1(t)}{dt} &= -\frac{r_1(t)}{\tau^+}, \text{ if } t = t^{pre} \text{ then } r_1(t) = r_1(t) + 1 \\
\frac{dr_2(t)}{dt} &= -\frac{r_2(t)}{\tau^x}, \text{ if } t = t^{pre} \text{ then } r_2(t) = r_2(t) + 1 \\
\Delta\omega_{ij}(t^{pre}) &= -o_1(t)\left[A_2^- + A_3^- r_2(t - dt)\right], \ \Delta\omega_{ij}(t^{post}) = r_1(t)\left[A_2^+ + A_3^+ o_2(t - dt)\right]
\end{aligned} \tag{6.6}
$$

In the final equations, we use $r_2(t - dt)$ and $o_2(t - dt)$ so that the current spike does not affect the activity trace that is supposed to represent the activity based on previous spikes.

STDP has been applied to many different models, and has been attributed many different functional consequences, indicating a wide range of possible cost functions that it may be minimizing.

Masquelier and Thorpe connected two population of neurons representing two layers of the visual hierarchy together and showed that an STDP learning rule caused the network to do a kind of feature selection in a first-to-spike coding scheme [135]. Masquelier also collaborated on a later paper that suggested that STDP is key to the development of temporal coding schemes in the brain, meaning that the encoding of high-dimensional information may require STDP [134].

Sprekeler et al. showed that the STDP rule could be derived by assuming that the cost function was one that measures slowness, which enables a network using STDP to do slow feature analysis [188]. Slowness can be thought of as temporal stability; if we think about using the rule in visual cortex, we would expect objects in the visual field to stay generally the same from one point in time to the next. Slow feature analysis aims to find those time-invariant features.

Nessler et al. showed that STDP is able to do expectation-maximization, which is essentially the process of minimizing KL-divergence, which suggests that it may be a possible cost function for STDP as well [146]. Krieg and Triesch derived an STDP rule by maximizing kurtosis (i.e., sparseness) of the membrane potential. This rule was able to recover both precise spike-time and frequency dependence, making it a good candidate for a possible cost function for STDP [112].

Finally, STDP has been implicated in two different forms of normalization. In one form, it has been shown that STDP can be used to maintain a relatively constant firing rate in postsynaptic cells without an explicit normalization step [183]. In another, introducing a small amount of random jitter in the STDP window results in intrinsic stability of synaptic connection weights, which in other studies, have often polarized (i.e., half of the synapses go to some very small strength, half go to some very large strength) [6].

All of these simulation results together indicate that the space of possible cost functions that STDP may be minimizing is rather large, though there are promising overlaps between the functional consequences of the BCM rule and STDP.

6.2.4 Relationship between BCM and STDP rules

Similarities in the functional consequences and cost functions of the BCM and STDP learning rules raises a question about the similarity between the two rules. BCM was originally devised with rate-based neuron models, and STDP necessarily uses spiking neurons; however, both are modelling essentially the same thing: induction of LTP and LTD in the brain.

Izhikevich showed that BCM and STDP can be mapped onto each other directly, reflecting their similarities in both experimental motivations and modelling studies [93]. However, this equivalence is contingent on two conditions: that the firing patterns of pre- and postsynaptic neurons are uncorrelated or weakly correlated Poisson spike trains, and that only nearest-spike interactions are take into account (i.e., only equation (6.3) can be mapped to a BCM rule).

Izhikevich's argument preceded Pfister et al.'s triplet rule proposal. When a third spike is considered, it was shown that the all-to-all triplet rule (i.e., equation (6.6)) maps onto a BCM rule [158]; in this case, the conditions are that A_3^- must be 0, meaning that only the post-pre rule is applied and not the pre-post-pre rule, and that the input and output spike trains have Poisson statistics.

6.2.5 General unsupervised learning

The variety of problems solved by the same or similar unsupervised learning techniques, and the fact that rules similar to BCM rules and STDP rules can be obtained by minimizing many different but related cost functions, suggests that these rules do general unsupervised learning, extracting salient features from input. All unsupervised learning systems do this abstractly, the main difference is in how each system defines salience.

There are two possible ways in which different saliency measures could be identified by the same network with a similar learning rule.

1. Saliency may be solely dependent on input.
2. STDP/BCM may each refer to a class of unsupervised learning rules, each specific rule designed to pick out a particular salient feature.

The variety of STDP curves found in the brain and the different mathematical descriptions of BCM theory suggest the latter explanation is more plausible.

6.3 Unsupervised learning in the NEF

We wanted to experimentally confirm the theoretical arguments put forth by Izhikevich, Pfister and Gerstner that the BCM rule is equivalent to STDP. To do this, we created a network with only two LIF neurons connected unidirectionally. We manually chose the neuron parameters such that the neurons could be stimulated deterministically from 0 to 100 Hz. The encoding vector was 1 for both neurons. A node in the network was programmed to compute the BCM postsynaptic filter, which was used as the error term $\mathbf{E}$ in our normal learning rule. Recall that that learning rule is

$$\Delta\omega_{ij} = \kappa\alpha_j \mathbf{e}_j \cdot \mathbf{E} a_i. \tag{6.7}$$

Substituting the BCM filter $a_j(a_j - \theta)$ in for $\mathbf{E}$, and setting $\mathbf{e}_j$ to 1 gives

$$\Delta\omega_{ij} = \kappa\alpha_j a_i a_j (a_j - \theta). \tag{6.8}$$

This is essentially the original BCM learning rule, equation (6.2). Note that the neuron gain is a scalar, and can easily be absorbed into the learning rate constant κ.

This mapping from the synapse-specific but arguably non-Hebbian error-minimization rule, equation (6.7), to a Hebbian BCM rule, equation (6.8), lends more credibility to the assertion that the error-minimization rule is biologically plausible.

6.3.1 Simulation results

Using the network of two neurons with the BCM rule, we aimed to recreate the effects of the STDP protocol, as seen in dozens of experiments. In doing so, we would add simulated experimental credence to the theoretical findings of Izhikevich, and Pfister and Gerstner. Additionally, it suggests that the learning rule that has been used in the previous chapters can be modified slightly such that it is sensitive to exact spike-times, as seen in STDP experiments, despite the fact that it does not explicitly track the times of pre- and postsynaptic spikes.

To recreate the STDP protocol, the two neurons are given a series of current pulses lasting 3ms, chosen because it elicited a single spike robustly. There are three free parameters that were varied in these experiments, and one that was randomly selected.

Pulse delay The pulse of current to the presynaptic and postsynaptic cells is delayed by a certain amount. Using appropriate delays ensures pre-post or post-pre spikes, which will be used to generate STDP curves.

Pulse rate The number of current pulses delivered per second can vary, in order to recreate experiments that show plasticity's frequency dependence.

Threshold value We consider the threshold to be a temporal average of recent postsynaptic activity. While the learning rule is implemented such that the threshold will change over time due to postsynaptic activity, the temporal average is done on the scale of hours, meaning that in these short simulations, θ does not change significantly. The threshold, θ, starts at a certain value in each simulation, which we vary. Note that Pfister and Gerstner also assume a long timescale temporal average in order to show that STDP can be mapped onto BCM [158].

Initial connection weight Unlike other NEF simulations, in this network we impose a minimum and maximum connection weight, 5×10^{-4} and 2×10^{-3} respectively. The initial connection weight is randomly selected uniformly between the minimum and maximum weight. This makes analysis of the results easier because we are always dealing with positive connection weights.

It should be noted that there are other parameters that can affect these results, such as the shape and amplitude of spike filters. In these simulations, the default value of 10ms was used for the time constants of the spike filters.

In the results presented below, each data point represents the relative change in the connection weight between the two neurons after 60 spike pairings (pre-post, post-pre, or simultaneous spikes), averaged over 20 trials.

The STDP curve

The simple STDP curve, first discussed in this thesis in section 2.1.3, is the most iconic graph in spike-timing dependent plasticity research. It describes the relationship between the amount of synaptic weight change as a function of the difference in time between the presynaptic and postsynaptic spikes.

In order to recreate an STDP curve, we varied the pulse delay in a number of situations. The results are plotted in figure 6.6.

Despite not explicitly remembering spike times, the postsynaptic BCM filter enables the learning rule to be critically dependent on the relative timings of presynaptic and postsynaptic activity. In some cases, most notably when both neurons are spiking at 20Hz, there is a sharp discontinuity when changing from pre-post to post-pre spike pairs.

None of these curves match exactly the STDP curve originally discovered by Bi and Poo (see figure 2.7); however, several of the experiments, most notably those done at 20Hz, match the qualitative effects: pre-post pairings that are sufficiently close cause potentiation,

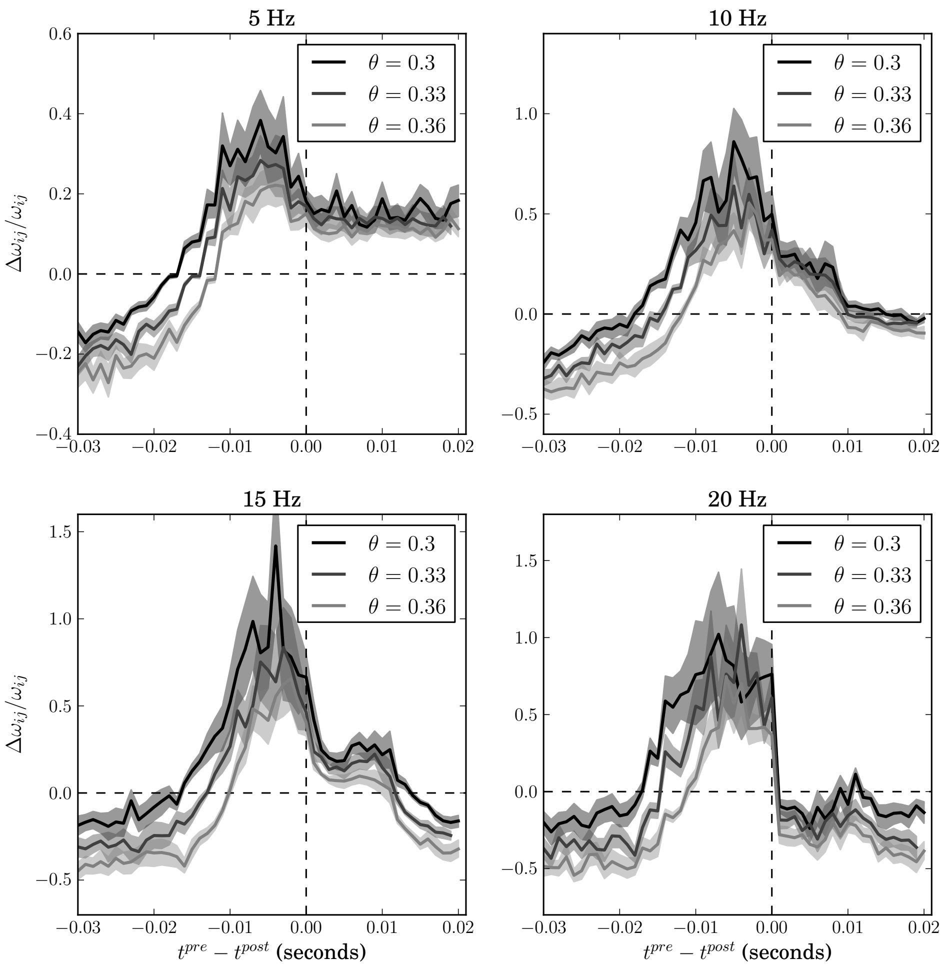


Figure 6.6: Simulated STDP curves using the two-neuron network with a BCM learning rule. Surrounding areas represent bootstrapped 95% confidence intervals.

which gets stronger as the spikes get closer, until post precedes pre, and then depression occurs. Some of the other STDP curves, such as the one found at 10Hz, look qualitatively similar to others that have been seen experimentally (see figure 2.8). This suggests that BCM may be a more general rule than STDP, able to replicate several experimentally observed STDP curves, as opposed to the traditional STDP models presented earlier in this chapter, which only model one type of STDP.

The effect of changing θ is consistent throughout all of the experiments; a decrease in θ causes an upward shift of the STDP curve, regardless of its shape. Decreasing θ further may enable the network to achieve a more quantitative similarity to the traditional STDP curve; decreasing it could, for example, get rid of the pre-post depression in these curves.

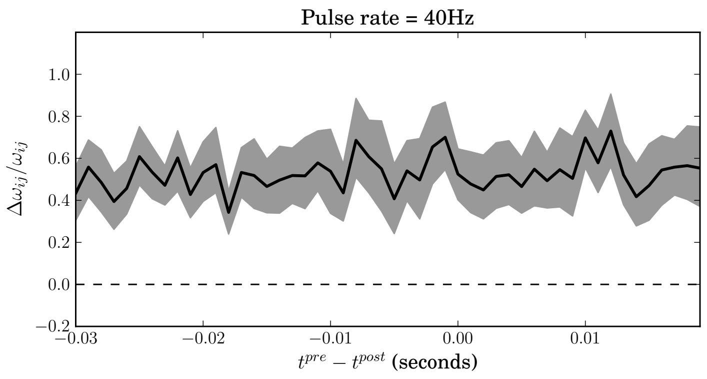


Figure 6.7: With high frequencies, the network is insensitive to precise timing effects.

We also tested the network with a relatively high pulse rate, as shown in figure 6.7. With very high frequencies, potentiation is seen regardless of temporal order. A more reasoned exploration of frequency dependence follows.

Frequency dependence curves

While frequency dependence is an aspect of STDP that is often overlooked, the experimental result that LTP/LTD induction with STDP protocols critically depends on the frequency of spikes calls into question the validity of pair-based STDP models. Pfister and Gerstner's triplet model can reproduce frequency effects; ideally, the BCM rule would be able to reproduce these effects as well, making it conceptually more powerful than pair-based STDP models.

To test this, we varied the pulse rate in the two-neuron network. The results, shown in figure 6.8, confirm that our network is sensitive to the frequency of pre- and postsynaptic spiking, in addition to the strict temporal effects previously discussed. This implies that the BCM rule has more in common with the more biologically plausible triplet model than with pair-based models.

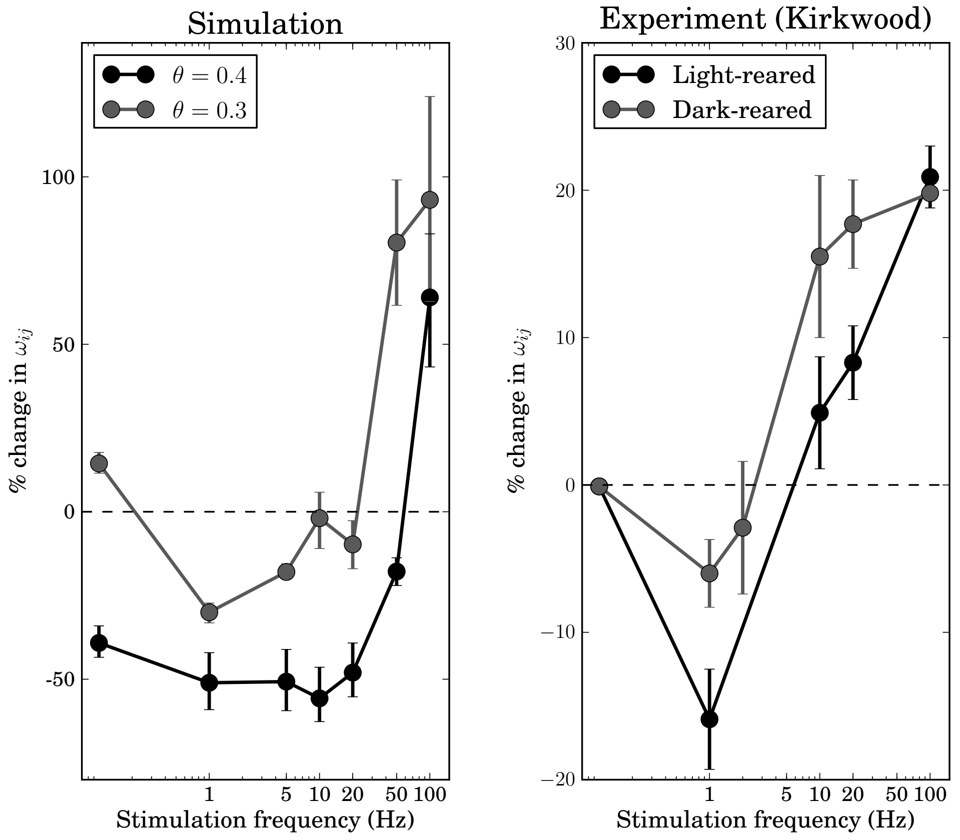


Figure 6.8: Plots showing that the two-neuron network with the BCM rule is sensitive to the frequency of spike pairings. (Left) Change in synaptic strength as a function of pulse rate for the simulated network. (Right) The same axes, but for Kirkwood et al.'s experimental data. From [105].

By varying θ, we have also replicated the effects of Kirkwood et al.'s experiments in rat visual cortex. A lower θ value represents less postsynaptic activity, which would be seen in the dark-reared rats; LTP is more readily induced in those rats, and also in the model with lower θ values.

Discussion

These results support Izhikevich, Pfister and Gerstner's theoretical arguments that STDP can be mapped onto BCM theory. We have shown that a spiking two-neuron network with a BCM rule can exhibit the essential feature of STDP: critical dependence on the time difference between pre- and postsynaptic spikes. Additionally, by showing that the model also exhibits frequency dependence, we assert that it is more similar to the biologically plausible triplet model than pair-based models. Testing in networks with more than two neurons is the logical next step for this line of research.

Additionally, we have shown that BCM theory can be applied directly to a spiking neural network easily. Where the original BCM rules considered "activity" to be the firing rates of neurons, we instead interpret it to be filtered spike trains. The sliding threshold θ is updated online by keeping track of the average postsynaptic activity over a long timescale. The network appears to perform as well as the rate-based BCM models, though more large-scale experiments should be done to explore this further.

6.3.2 A unifying learning rule

While we showed in section 6.3 that the error-minimization rule used in chapters 4 and 5 can be manipulated to essentially be the BCM rule, we propose that a general rule that can exhibit either supervised or unsupervised learning can be implemented, and hypothesize that it can be used in any network to do unsupervised, supervised, or reinforcement learning by changing parameters.

The main idea is to additively combine rules (6.7) and (6.8). There are several ways to do this.

It may be the case that unsupervised learning effects are applied independent of the encoding vector of the postsynaptic neuron. In this case, the rule is

$$\Delta\omega_{ij} = \alpha_j a_i \left[\kappa_u a_j (a_j - \theta) + \kappa_s \mathbf{e}_j \cdot \mathbf{E}\right], \tag{6.9}$$

where κ_u is the learning rate corresponding to the unsupervised learning term, and κ_s is the learning rate for the supervised learning term. It is hypothesized that these learning rates may be inversely proportion to each other, meaning that the more supervision, the less unsupervised learning occurs, and vice versa.

It may instead be the case that the postsynaptic spike filter should also be interpreted using the encoding vector, meaning that unsupervised learning would also depend on what part of the vector space the neuron is sensitive to. In this case, we treat the unsupervised portion of the rule in the same way we treat $\mathbf{E}$, resulting in the rule

$$\Delta\omega_{ij} = \alpha_j a_i \left[\kappa_u \mathbf{e}_j \cdot \mathbf{e}_j a_j (a_j - \theta) + \kappa_s \mathbf{e}_j \cdot \mathbf{E}\right]. \tag{6.10}$$

However, since $\mathbf{e}_j$ is a unit vector, $\mathbf{e}_j \cdot \mathbf{e}_j = |\mathbf{e}_j|^2 = 1^2 = 1$. Therefore, these rules are equivalent. The unsupervised term does not rely on the encoding vector unless that vector is not a unit vector.

At the moment, the simpler rule, (6.9), can be used in any of the simulations presented in this thesis; for the supervised and reinforcement learning sections, we would set $\kappa_u = 0$, and for the unsupervised learning section, we would set $\kappa_s = 0$. Further testing with both terms used together in supervised and reinforcement learning should give more insight into the functional role of unsupervised learning in the brain.

Chapter 7

Discussion and conclusions

This thesis aimed to give a comprehensive overview of learning in spiking neural networks. We have used the taxonomy of learning problems currently used by machine learning, and showed how those problems have been solved by traditional artificial neural networks, other spiking neural network models, and spiking neural network models created with the principles of the Neural Engineering Framework.

Novel contributions of this thesis include all of the NEF network models discussed; i.e., sections 4.3, 5.3, and 6.3. In addition, plasticity rules were implemented in Nengo, a piece of software for building NEF network models; a technical report detailing this process has been previously published [15].

While the error-minimization learning rule used in many of the NEF networks in this thesis was previously derived and published [128], we proposed a novel learning rule that integrates both unsupervised and supervised learning effects (equation (6.9)). This learning rule will be the focus of future research.

In the remainder of this thesis, we discuss other promising areas of future research.

7.1 Large-scale unsupervised learning

In chapter 6, we provided simulation results that supported the idea that STDP can be mapped onto BCM theory. However, we did not create a large-scale model that investigated the possible functional effects of BCM theory.

Chapter 9.5 of [59] discusses a technique for determining the function being computed given two populations and their connection weight matrix. Application of this, or a similar, technique to a large scale network learning without supervision would give insight into the cost function being minimized by these rules.

7.2 Model-based reinforcement learning

At the end of chapter 5, we discussed outstanding issues of the current model, including more attention being paid to the exploration-exploitation problem. One issue that we did not touch on in that chapter is the distinction between model-based and model-free reinforcement learning. We took a model-free approach for simplicity, but also because implementing a model-based approach would require a different mechanism for action selection; it would necessitate a network that actively makes predictions about future states.

Implementing such a system requires more thought than do the other challenges we discussed in that chapter; however, it is essential to examine model-based reinforcement learning because it is clear that humans and other animals can make active predictions and plan ahead, suggesting that they construct models of the environment and exploit them during reinforcement learning tasks. Doya et al. have even proposed a reinforcement learning algorithm that learns multiple models simultaneously, using information from a combination of them to select an appropriate action [55]. Neural implementations of model-based reinforcement learning will be an important step forward in learning in large-scale spiking neural networks.

7.3 Supervised error signals

Reinforcement learning has been widely explored in theoretical neuroscience in large part due to Schultz and other's work showing that dopaminergic neurons signal something akin to temporal-difference reward prediction error. Supervised learning is not motivated in the same way; it is a theoretical problem posed by machine learning, and largely unexplored in neuroscience.

The NEF and other theoretical neuroscience frameworks make the assumption that the brain is a computational device; groups of neurons do some kind of mathematical transformation on the information encoded by the spiking patterns of afferent neurons. If this assumption is true, then it should be possible to set up an experiment that would allow one to observe experimentally a supervised learning process. The creation of such an experiment would be highly informative to theoretical models of learning in large-scale spiking neural networks.

However, such an experiment is difficult to devise without knowing how fine-grained error signals would be generated. Work on the cerebellum has suggested that fine-grained error signals exist there, but the exact use of those error signals remains an open question [52].

From a theoretical perspective, further research into error signals in the brain, and the neural circuits that produce them, is the logical next step. It may be possible to identify and model a system that does not require an idealized error signal provided by the modeller, but instead computes it from a plausible neural model. Ideally, this model would be well-defined enough to be tested experimentally, giving more insight into supervised learning in the brain.

7.3.1 Solving the supervised spike-time learning problem

While we have shown that we can solve the general supervised learning problem, and argued that the supervised spike-time learning problem is less general and thus should be able to be solved by our solution, we have not yet shown this, either through a theoretical argument or simulation results. This result would be significant to the field of supervised learning in spiking neural networks.

7.4 Computational complexity

While previous discussions of complexity and time-to-learn have focused on simulation time, issues of computational complexity are a large issue for scaling up the models presented in this thesis. Already, the supervised learning network struggles to learn 3-dimensional convolution in a reasonable amount of computer time. Learning networks, as opposed to other models created with the principles of the NEF, must track each connection weight separately, and compute an update for each connection weight on each timestep of the simulation. As neuronal populations get larger, the number of connections grows multiplicatively. Specifically, the amount of computations done on each timestep is a function of the number of input neurons, output neurons, and the dimensionality of the output population; or, in big-O notation, the computational complexity of the error-minimization rule used in this thesis is $O(N_i N_j D_j)$.

The most promising avenue for speeding up these learning simulations without sacrificing biological plausibility is to use powerful graphical processing units (GPUs), which have previously been used to speed up NEF networks without learning. Because all of the connection weight update rules are independent of each other, they are well suited for being sped up by the highly parallel GPUs.

7.4.1 Decoder-level learning

Another promising avenue for speeding up learning simulations is to move to a higher-level abstraction of these spiking neural networks. While we have said that learning requires examination at the low level of connection weights, it may be the case that we can learn in the space of decoding weights instead, and still argue for biological plausibility.

In the derivation of the error-minimization rule, an intermediate stage was one such learning rule on decoding weights, equation (3.18). This rule should be implemented in the NEF and compared to the analogous rule operating in connection weights to determine if they have any functional differences. If they do not, it would mean that we can create networks to learn functions in much higher-dimensional spaces with more neurons; it may even be feasible to use learning in networks that would typically employ faster non-learning methods for changing behaviour over time.

References

[1] Larry F. Abbott and Sacha B. Nelson. Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3:1178–83, November 2000. ix, 14, 15

[2] Per Andersen. A prelude to long-term potentiation. *Philosophical Transactions of the Royal Society of London*, 358(1432):613–5, April 2003. 8

[3] Brandon J. Aragona, Jeremy J. Day, Mitchell F. Roitman, Nathan A. Cleaveland, R. Mark Wightman, and Regina M. Carelli. Regional specificity in the real-time development of phasic dopamine transmission patterns during acquisition of a cue-cocaine association in rats. *The European Journal of Neuroscience*, 30(10):1889–99, November 2009. 53, 76

[4] Alain Artola, S. Bröcher, and Wolf Singer. Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature*, 347(6288):69–72, 1990. 91, 93

[5] Alain Artola and Wolf Singer. Long-term depression of excitatory synaptic transmission and its relationship to long-term potentiation. *Trends in Neurosciences*, 16(11):480–487, November 1993. 91

[6] Baktash Babadi and Larry F. Abbott. Intrinsic stability of temporally shifted spike-timing dependent plasticity. *PLoS Computational Biology*, 6(11):e1000961, January 2010. 98

[7] Bart Baddeley. Reinforcement learning in continuous time and space: interference and not ill conditioning is the main problem when using distributed function approximators. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(4):950–956, June 2008. 69

[8] Leemon C. Baird. Residual algorithms: reinforcement learning with function approximation. In *Machine Learning*, pages 30–37. Citeseer, 1995. 69

[9] Dorit Baras and Ron Meir. Reinforcement learning, spike-time-dependent plasticity, and the BCM rule. *Neural Computation*, 19(8):2245–79, August 2007. 74

[10] David Barber. Learning in spiking neural assemblies. In *Advances in Neural Information Processing Systems*, volume 15, page 165. The MIT Press, 2002. 48

[11] Andrew G. Barto. Adaptive critics and the basal ganglia. *Models of Information Processing in the Basal Ganglia*, (1994):215–232, 1995. 73

[12] Andrew G. Barto and Thomas G. Dietterich. Reinforcement learning and its relationship to supervised learning. Technical report, 2004. 69

[13] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):834–846, 1983. 62

[14] Mark F. Bear. Mechanism for a sliding synaptic modification threshold. *Neuron*, 15(1):1–4, July 1995. 93

[15] Trevor Bekolay. Using and extending plasticity rules in Nengo Plasticity rules in Nengo. Technical report, Centre for Theoretical Neuroscience, 2010. 107

[16] Ammar Belatreche, Liam Maguire, Martin McGinnity, and Qing Xiang Wu. A method for supervised training of spiking neural networks. *Cybernetic Intelligence, Challenges and Advances*, page 11, 2003. 48

[17] Curtis C. Bell, Victor Z. Han, Yoshiko Sugawara, and Kirsty Grant. Synaptic plasticity in a cerebellum-like structure depends on temporal order. *Nature*, 246(5429):170–170, November 1997. 13

[18] Richard Bellman. A Markovian decision process. *Mathematics and Mechanics*, 6(5):679–684, 1957. 66

[19] Kent C. Berridge. The debate over dopamine's role in reward: the case for incentive salience. *Psychopharmacology*, 191(3):391–431, April 2007. 71

[20] Dimitri P. Bertsekas. *Dynamic programming: deterministic and stochastic models*. Simon and Schuster, 1978. 66

[21] Guo-Qiang Bi and Mu-Ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, 18(24):10464–72, December 1998. 13, 14, 15

[22] Guo-Qiang Bi and Mu-Ming Poo. Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual Review of Neuroscience*, 24(1):139–66, January 2001. 14

[23] Elie L. Bienenstock, Leon N. Cooper, and Paul Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2(1):32, 1982. 92, 94

[24] Brian S. Blais and Leon N. Cooper. BCM theory. *Scholarpedia*, 3(3):1570, 2008. 93

[25] Brian S. Blais, Nathan Intrator, Harel Z. Shouval, and Leon N. Cooper. Receptive field formation in natural scene environments: comparison of single-cell learning rules. *Neural Computation*, 10(7):1797–1813, September 1998. 95

[26] T. V. P. Bliss and Terje Lø mo. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of Physiology*, 232(2):331, 1973. 8, 9

[27] Tim V. P. Bliss, Graham L. Collingridge, and Richard G. M. Morris. *Long-term potentiation: enhancing neuroscience for 30 years*. Oxford University Press, USA, 2004. 9

[28] Hal Blumenfeld. *Neuroanatomy through clinical cases*, volume 14. Sinauer Associates, Sunderland, MA, 2002. x, 72, 73

[29] Rafal Bogacz, Malcolm W. Brown, and Christophe Giraud-Carrier. Frequency-based error back-propagation in a cortical network. In *International Joint Conference on Neural Networks*, pages 211–216 vol.2. Ieee, 2000. 48

[30] Rafal Bogacz, Malcolm W. Brown, and Christophe Giraud-Carrier. Model of familiarity discrimination in the perirhinal cortex. *Journal of Computational Neuroscience*, 10(1):5–23, 2001. 48

[31] Sander M. Bohte, Joost N. Kok, and Han La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002. 46

[32] Sander M. Bohte, Joost N. Kok, and Han La Poutre. SpikeProp: Backpropagation for networks of spiking neurons. *Neurocomputing*, 48(1-4):17, 2002. 46

[33] Tiago Branco and Michael Häusser. The single dendritic branch as a fundamental functional unit in the nervous system. *Current Opinion in Neurobiology*, 20(4):494–502, August 2010. 10

[34] Arthur Earl Bryson and Yu-Chi Ho. *Applied optimal control: optimization, estimation, and control*. Blaisdell Publishing Company, 1969. 41

[35] Paolo Calabresi, Paolo Gubellini, Diego Centonze, Barbara Picconi, Giorgio Bernardi, Karima Chergui, Per Svenningsson, Allen A. Fienberg, and Paul Greengard. Dopamine and cAMP-regulated phosphoprotein 32 kDa controls both striatal long-term depression and long-term potentiation, opposing forms of synaptic plasticity. *The Journal of Neuroscience*, 20(22):8443–51, November 2000. 18

[36] Paolo Calabresi, Barbara Picconi, Alessandro Tozzi, and Massimiliano Di Filippo. Dopamine-mediated regulation of corticostriatal synaptic plasticity. *Trends in Neurosciences*, 30(5):211–9, May 2007. 18

[37] Paolo Calabresi, Adolfo Saiardi, Aantonio Pisani, Ja-Hyun Baik, Diego Centonze, Nicola B. Mercuri, Giorgio Bernardi, and Emiliana Borrelli. Abnormal synaptic plasticity in the striatum of mice lacking dopamine D2 receptors. *Journal of Neuroscience*, 17(12):4536–44, June 1997. 18

[38] Natalia Caporale and Yang Dan. Spike timing-dependent plasticity: a Hebbian learning rule. *Annual Review of Neuroscience*, 31(1):25–46, 2008. 14

[39] Andrew Carnell and Daniel Richardson. Linear algebra for time series of spikes. In *European Symposium on Artificial Neural Networks*, number April, pages 363–368. Citeseer, 2005. 48

[40] Pablo E. Castillo, Chiayu Q. Chiu, and Reed C. Carroll. Long-term plasticity at inhibitory synapses. *Current Opinion in Neurobiology*, 21:328–338, February 2011. 18

[41] C. A. Castro, L. H. Silbert, Bruce L. McNaughton, and C. A. Barnes. Recover of spatial learning deficits after decay of electrically induced synaptic enhancement in the hippocampus. *Nature*, 342:545–548, 1989. 11

[42] Feng-Xuan Choo. *The ordinal serial encoding model: serial memory in spiking neurons*. PhD thesis, University of Waterloo, 2010. ix, x, 23, 28, 30, 35, 36, 55

[43] Paul Chorley and Anil K. Seth. Dopamine-signaled reward predictions generated by competitive excitation and inhibition in a spiking neural network model. *Frontiers in Computational Neuroscience*, 5(May):21, January 2011. 75

[44] Brian R. Christie, Jeffrey C. Magee, and Daniel Johnston. The role of dendritic action potentials and Ca2+ influx in the induction of homosynaptic long-term depression in hippocampal CA1 pyramidal neurons. *Learning & Memory*, 3(2-3):160–9, 1996. 12

[45] Leon N. Cooper. A possible organization of animal memory and learning. In *Nobel Symposium on Collective Properties of Physical Systems*, volume 252, page 264. Academic Press, New York, 1973. 10

[46] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989. 43, 52

[47] Justin Dauwels, Francois Vialatte, Theophane Weber, and Andrzej Cichocki. On similarity measures for spike trains. *Advances in Neuro-Information Processing*, 2009. 45

[48] Nathaniel D. Daw. *Reinforcement learning models of the dopamine system and their behavioral implications.* PhD thesis, 2003. 73

[49] Peter Dayan. The convergence of TD(λ) for general λ. *Machine Learning*, 8(3-4):341–362, May 1992. 68

[50] Peter Dayan. Unsupervised Learning. In *The MIT Encyclopedia of the Cognitive Sciences*, volume 47. October 1999. 90

[51] Peter Dayan and Larry F. Abbott. *Theoretical neuroscience.* MIT Press, 2001. 31

[52] Paul Dean, John Porrill, Carl-Fredrik Ekerot, and Henrik Jörntell. The cerebellar microcircuit as an adaptive filter: experimental and computational evidence. *Nature Reviews Neuroscience*, 11(1):30–43, January 2010. x, 44, 45, 47, 109

[53] Kenji Doya. Temporal difference in learning in continuous time and space. *Advances in Neural Information Processing Systems*, pages 1073–1079, 1996. 69

[54] Kenji Doya. Reinforcement learning in continuous time and space. *Neural Computation*, 12(1):219–45, January 2000. 69

[55] Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural Computation*, 12:1347–1369, August 2002. 108

[56] Serena M. Dudek and Mark F. Bear. Homosynaptic long-term depression in area CA1 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(10):4363–7, May 1992. 12

[57] Thomas Dunwiddie and Gary Lynch. Long-term potentiation and depression of synaptic responses in the rat hippocampus: localization and frequency dependency. *The Journal of Physiology*, (1978):353–367, 1978. 11, 12

[58] Chris Eliasmith. Notes from SYDE750 course, 2011. ix, 22, 32

[59] Chris Eliasmith and Charles H. Anderson. *Neural engineering: computation, representation, and dynamics in neurobiological systems*. Computational neuroscience. MIT Press, 2003. 20, 23, 24, 29, 31, 37, 60, 107

[60] Michael A. Farries and Adrienne L. Fairhall. Reinforcement learning with modulated spike timing dependent synaptic plasticity. *Journal of Neurophysiology*, 98(6):3648–65, December 2007. 74

[61] Paul Fatt. Electric potentials occurring around a neurone during its antidromic activation. *Journal of Neurophysiology*, 20(1):27, 1957. 9

[62] Elodie Fino, Jacques Glowinski, and Laurent Venance. Bidirectional activity-dependent plasticity at corticostriatal synapses. *The Journal of Neuroscience*, 25(49):11279–87, December 2005. 16

[63] Christopher D. Fiorillo, Philippe N. Tobler, and Wolfram Schultz. Discrete coding of reward probability and uncertainty by dopamine neurons. *Science*, 299(5614):1898–902, March 2003. 71

[64] Rzvan V. Florian. A reinforcement learning algorithm for spiking neural networks. In *Symbolic and Numeric Algorithms for Scientific Computing*, number Section 3, page 8 pp. Ieee, 2005. 74

[65] Rzvan V. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007. 74

[66] D. J. Foster, R. G. Morris, and Peter Dayan. A model of hippocampally dependent navigation, using the temporal difference learning rule. *Hippocampus*, 10(1):1–16, January 2000. 73

[67] Robert C. Froemke, Mu-ming Poo, and Yang Dan. Spike-timing-dependent synaptic plasticity depends on dendritic location. *Nature*, 434(7030):221–225, 2005. 15

[68] M. Fujita. Adaptive filter model of the cerebellum. *Biological Cybernetics*, 45(3):195–206, 1982. 44

[69] Apostolos P. Georgopoulos, Andrew B. Schwartz, and Ronald E. Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, September 1986. 24

[70] Samuel J. Gershman, Jonathan D. Cohen, and Yael Niv. Learning to selectively attend. In *Cognitive Science*, pages 1270–1275, 2010. 56

[71] Nace L. Golding, Nathan P. Staff, and Nelson Spruston. Dendritic spikes as a mechanism for cooperative long-term potentiation. *Nature*, 418(6895):326–331, 2002. 15

[72] Xianping Guo and Onesimo Hernandez-Lerma. *Continuous-Time Markov decision processes: theory and applications*, volume 26. Springer, December 2009. 87

[73] Hirac Gurden, Masatoshi Takita, and Therese M. Jay. Essential role of D1 but not D2 receptors in the NMDA receptor-dependent long-term potentiation at hippocampal-prefrontal cortex synapses in vivo. *The Journal of Neuroscience*, 20(22):RC106, November 2000. 18

[74] Kevin Gurney, Tony J. Prescott, and Peter Redgrave. A computational model of action selection in the basal ganglia. I. A new functional anatomy. *Biological Cybernetics*, 84(6):401–10, June 2001. 79, 80

[75] Kevin Gurney, Tony J. Prescott, and Peter Redgrave. A computational model of action selection in the basal ganglia. II. Analysis and simulation of behaviour. *Biological Cybernetics*, 84(6):411–23, June 2001. 79, 80

[76] B. Gustafsson and H. Wigstrom. Hippocampal long-lasting potentiation produced by pairing single volleys and brief conditioning tetani evoked in separate afferents. *The Journal of Neuroscience*, 6(6):1575, 1986. 9

[77] B. Gustafsson, H. Wigström, W. C. Abraham, and Y. Y. Huang. Long-term potentiation in the hippocampus using depolarizing current pulses as the conditioning stimulus to single volley synaptic potentials. *The Journal of Neuroscience*, 7(3):774–80, March 1987. 9

[78] Bengt Gustafsson and Holger Wigstrom. Physiological mechanisms underlying long-term potentiation. *Trends in Neurosciences*, 11(4):156–162, 1988. 9, 15

[79] Julie S. Haas, Thomas Nowotny, and H. D. I. Abarbanel. Spike-timing-dependent plasticity of inhibitory synapses in the entorhinal cortex. *Journal of Neurophysiology*, 96(6):3305–13, December 2006. 18

[80] Peter J. B. Hancock, Leslie S. Smith, and William A. Phillips. A biologically supported error-correcting learning rule. *Neural Computation*, 3(2):201–212, June 1991. 91

[81] Thomas Hanselmann, Lyle Noakes, and Anthony Zaknich. Continuous-time adaptive critics. *IEEE Transactions on Neural Networks*, 18(3):631–47, May 2007. 73

[82] Jason Hardie and Nelson Spruston. Synaptic depolarization is more effective than back-propagating action potentials during induction of associative long-term potentiation in hippocampal pyramidal neurons. *Journal of Neuroscience*, 29(10):3233–41, March 2009. 10

[83] Michael Hausser, Nelson Spruston, and Greg J. Stuart. Diversity and dynamics of dendritic signaling. *Science*, 290(5492):739–744, October 2000. 9

[84] Donald O. Hebb. *The organization of behavior*. Wiley & Sons, New York, 1949. 6, 7

[85] Geoffrey E. Hinton and James L. McClelland. Learning representations by recirculation. In *Neural Information Processing Systems*, pages 358–366. American Institute of Physics, New York, 1988. 48

[86] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500, 1952. 20

[87] Carl D. Holmgren and Yuri Zilberter. Coincident spiking activity induces long-term changes in inhibition of neocortical pyramidal cells. *The Journal of Neuroscience*, 21(20):8270–7, October 2001. 15

[88] K Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. 43, 52

[89] 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. 93, 95, 96

[90] Masao Ito. Long-term depression. *Annual Review of Neuroscience*, 12(1):85–102, 1989. 12

[91] Eugene M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15(5):1063–70, September 2004. 21

[92] Eugene M. Izhikevich. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 17(10):2443–2452, 2007. 74, 75, 76

[93] Eugene M. Izhikevich and Niraj S. Desai. Relating STDP to BCM. *Neural Computation*, 15(7):1511–1523, 2003. 99

[94] Vincent Jacob, Daniel J. Brasier, Irina Erchova, Dan Feldman, and Daniel E. Shulz. Spike timing-dependent synaptic depression in the in vivo barrel cortex of the rat. *The Journal of Neuroscience*, 27(6):1271–84, February 2007. 16

[95] Therese M. Jay. Dopamine: a potential substrate for synaptic plasticity and memory mechanisms. *Progress in Neurobiology*, 69(6):375–390, April 2003. 18

[96] Daphna Joel, Yael Niv, and Eytan Ruppin. Actor-critic models of the basal ganglia: new anatomical and computational perspectives. *Neural Networks*, 15(4-6):535–47, 2002. 73

[97] Renaud Jolivet, Felix Schürmann, Thomas K. Berger, Richard Naud, Wulfram Gerstner, and Arnd Roth. The quantitative single-neuron modeling competition. *Biological Cybernetics*, 99(4-5):417–26, November 2008. 20

[98] Matt Jones and Fabián Cañas. Integrating reinforcement learning with models of representation learning. In *Conference of the Cognitive Science Society*, pages 1258–1263, 2010. 73

[99] J. Karhunen. Nonlinear independent component analysis. In Stephen Roberts and Richard Everson, editors, *Independent Component Analysis: Principles and Practice*, pages 113–134. Cambridge University Press, Cambridge, 2001. 94

[100] Hiroyuki K. Kato, Ayako M. Watabe, and Toshiya Manabe. Non-Hebbian synaptic plasticity induced by repetitive postsynaptic action potentials. *The Journal of Neuroscience*, 29(36):11153–60, September 2009. 16

[101] S. R. Kelso, A. H. Ganong, and T. H. Brown. Hebbian synapses in hippocampus. *Proceedings of the National Academy of Sciences of the United States of America*, 83(14):5326–30, July 1986. 9

[102] Jason N. D. Kerr and J. R. Wickens. Dopamine D-1/D-5 receptor activation is required for long-term potentiation in the rat neostriatum in vitro. *Journal of Neurophysiology*, 85(1):117, 2001. 18

[103] Mehdi Khamassi, Loïc Lachèze, Benoît Girard, Alain Berthoz, and Agnès Guillot. Actor-critic models of reinforcement learning in the basal ganglia: from natural to artificial rats. *Adaptive Behavior*, 13(2):131–148, June 2005. 73

[104] Hoseok Kim, Jung Hoon Sul, Namjung Huh, Daeyeol Lee, and Min Whan Jung. Role of striatum in updating values of chosen actions. *Journal of Neuroscience*, 29(47):14701–12, November 2009. 81, 84

[105] Alfredo Kirkwood, Marc G. Rioult, and Mark F. Bear. Experience-dependent modification of synaptic plasticity in visual cortex. *Nature*, 381(6582):526–528, 1996. 93, 95, 104

[106] A. Harry Klopf. Brain function and adaptive systems: a heterostatic theory. Technical Report 133, DTIC Document, 1972. 62

[107] A. Harry Klopf. A comparison of natural and artificial intelligence. *ACM SIGART Bulletin*, (52):11–13, 1975. 62

[108] A. Harry Klopf. A neuronal model of classical conditioning. *Psychobiology*, 1988. 7

[109] Christoph Kolodziejski, Bernd Porr, and Florentin Worgotter. On the asymptotic equivalence between differential Hebbian and temporal difference learning. *Neural Computation*, 21(4):1173–1202, 2009. 8, 75

[110] Konrad P. Körding and Peter König. Supervised and unsupervised learning with two sites of synaptic integration. *Journal of Computational Neuroscience*, 11(3):207–15, 2002. 49

[111] Bart Kosko. Differential hebbian learning. In *American Institute of Physics: Neural Networks for Computing*, pages 277–282. Citeseer, 1986. 7, 8

[112] Daniel Krieg and Jochen Triesch. STDP explained? Connecting function and biophysics. In *Computational and Systems Neuroscience*, 2011. 98

[113] Nicola Kuczewski, Christophe Porcher, Nadine Ferrand, Hervé Fiorentino, Christophe Pellegrino, Richard Kolarow, Volkmar Lessmann, Igor Medina, and Jean-Luc Gaiarsa. Backpropagating action potentials trigger dendritic release of BDNF during spontaneous network activity. *Journal of Neuroscience*, 28(27):7013–23, July 2008. 10

[114] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. 90

[115] Harri Lappalainen and Xavier Giannakopoulos. Multi-layer perceptrons as nonlinear generative models for unsupervised learning: a Bayesian treatment. In *International Conference on Artificial Neural Networks*, volume 1999, pages 19–24. Iee, 1999. 90

[116] C. Charles Law and Leon 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):7797–801, August 1994. 94

[117] Kevin S. Lee. Cooperativity among afferents for the induction of long-term potentiation in the CA1 region of the hippocampus. *The Journal of Neuroscience*, 3(7):1369, 1983. 9

[118] Johannes J. Letzkus, Björn M. Kampa, and Greg J. Stuart. Learning rules for spike timing-dependent plasticity depend on dendritic synapse location. *The Journal of Neuroscience*, 26(41):10420–9, October 2006. 14

[119] W. B. Levy and O. Steward. Temporal contiguity requirements for long-term associative potentiation/depression in the hippocampus. *Neuroscience*, 8(4):791–797, 1983. 12, 13

[120] William B. Levy and Oswald Steward. Synapses as associative memory elements in the hippocampal formation. *Brain Research*, 175(2):233–245, 1979. 9

[121] David J. Linden. Long-term synaptic depression in the mammalian brain. *Neuron*, 12(3):457–72, March 1994. 10, 11, 12

[122] John Lisman and Nelson Spruston. Postsynaptic depolarization requirements for LTP and LTD: a critique of spike timing-dependent plasticity. *Nature Neuroscience*, 8(7):839–841, 2005. 15, 16, 97

[123] John E. Lisman, Howard Schulman, and Hollis Cline. The molecular basis of CaMKII function in synaptic and behavioural memory. *Nature Reviews Neuroscience*, 3(3):175–90, March 2002. 15

[124] Dmitri V. Lissin, Stephen N. Gomperts, Reed C. Carroll, Chadwick W. Christine, Daniel Kalman, Marina Kitamura, Stephen Hardy, Roger A. Nicoll, Robert C. Malenka, and Mark von Zastrow. Activity differentially regulates the surface expression of synaptic AMPA and NMDA glutamate receptors. *Proceedings of the National Academy of Sciences of the United States of America*, 95(12):7097–102, June 1998. 17

[125] Terje Lø mo. The discovery of long-term potentiation. *Philosophical Transactions of the Royal Society of London*, 358(1432):617–20, April 2003. 8

[126] Gary Lynch, Thomas Dunwiddie, and V. Gribkoff. Heterosynaptic depression: a postsynaptic correlate of long-term potentiation. *Nature*, 266:737–9, 1977. 11

[127] Wolfgang Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8(1):1–40, January 1996. 45

[128] David Macneil and Chris Eliasmith. Fine-tuning and stability of recurrent neural networks. *PloS One*, 2011. 37, 107

[129] Jeffrey C. Magee and Daniel Johnston. A synaptically controlled, associative signal for hebbian plasticity in hippocampal neurons. *Science*, 275(5297):209–213, January 1997. 10

[130] Tiago V. Maia. Reinforcement learning, conditioning, and the brain: successes and challenges. *Cognitive Affective & Behavioral Neuroscience*, 9(4):343–64, December 2009. 71, 73, 77, 81

[131] Robert C. Malenka and Mark F. Bear. LTP and LTD: an embarrassment of riches. *Neuron*, 44(1):5–21, September 2004. 9, 10

[132] Henry Markram, Joachim Lubke, Michael Frotscher, and Bert Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275(5297):213–215, January 1997. 8, 13

[133] S. J. Martin, P. D. Grimwood, and R. G. Morris. Synaptic plasticity and memory: an evaluation of the hypothesis. *Annual Review of Neuroscience*, 23:649–711, January 2000. 19

[134] Timothée Masquelier, Rudy Guyonneau, and Simon J. Thorpe. Competitive STDP-based spike pattern learning. *Neural Computation*, 21(5):1259–1276, 2009. 98

[135] Timothée Masquelier and Simon J. Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Computational Biology*, 3(2):e31, February 2007. 98

[136] Peter V. Massey and Zafar I. Bashir. Long-term depression: multiple forms and implications for brain function. *Trends in Neurosciences*, 30(4):176–84, April 2007. 11, 12

[137] David A. McCormick, Barry W. Connors, James W. Lighthall, and David A. Prince. Comparative electrophysiology of pyramidal and sparsely spiny stellate neurons of the neocortex. *Journal of Neurophysiology*, 54(4):782–806, October 1985. ix, 28

[138] Sam McKennoch, Dingding Liu, and Linda G. Bushnell. Fast modifications of the SpikeProp algorithm. In *IEEE International Joint Conference on Neural Networks*, pages 3970–3977. Ieee, 2006. 46

[139] B. L. McNaughton, R. M. Douglas, and G. V. Goddard. Synaptic enhancement in fascia dentata: cooperativity among coactive afferents. *Brain Research*, 157(2):277–293, 1978. 9

[140] C. Daniel Meliza and Yang Dan. Receptive-field modification in rat visual cortex induced by paired visual stimulation and single-cell spiking. *Neuron*, 49(2):183–9, January 2006. 16

[141] Jacques Mirenowicz and Wolfram Schultz. Importance of unpredictability for reward responses in primate dopamine neurons. *Journal of Neurophysiology*, 72(2), 1994. 71

[142] Jugurta R. Montalvho Filho, Murilo A. Bezerra, and Levi P. Oliveira. Clustering with multilayer perceptrons and hebbian learning based on kullback-leibler divergence. In *Machine Learning for Signal Processing*, pages 243–252. Ieee, 2004. 90

[143] S.C. Moore. *Back-propagation in spiking neural networks*. PhD thesis, University of Bath, 2002. 46

[144] R. M. Mulkey and Robert C. Malenka. Mechanisms underlying induction of homosynaptic long-term depression in area CA1 of the hippocampus. *Neuron*, 9(5):967–75, November 1992. 12

[145] Takeshi Nakamura, Jean-Gael Barbara, Kyoko Nakamura, and William N. Ross. Synergistic release of Ca2+ from IP3-sensitive stores evoked by synaptic activation of mGluRs paired with backpropagating action potentials. *Neuron*, 24(3):727–37, November 1999. 10

[146] Bernhard Nessler, Michael Pfeiffer, and Wolfgang Maass. STDP enables spiking neurons to detect hidden causes of their inputs. In *Advances in Neural Information Processing Systems*, volume 22, pages 1–9. MIT Press, 2009. 98

[147] Thomas Nevian and Bert Sakmann. Spine Ca2+ signaling in spike-timing-dependent plasticity. *The Journal of Neuroscience*, 26(43):11001–13, October 2006. 15

[148] Makoto Nishiyama, Kyonsoo Hong, Katsuhiko Mikoshiba, Mu-Ming Poo, and Kunio Kato. Calcium stores regulate the polarity and input specificity of synaptic modification. *Nature*, 408(6812):584–8, November 2000. 15

[149] Yael Niv. *Reinforcement learning and the basal ganglia.* PhD thesis, Tel Aviv University, 2001. 18

[150] Richard J. O'Brien, Sunjeev Kamboj, Michael D. Ehlers, Kenneth R. Rosen, Gerald D. Fischbach, and Richard L. Huganir. Activity-dependent modulation of synaptic AMPA receptor accumulation. *Neuron*, 21(5):1067–78, November 1998. 17

[151] Erkki Oja. Unsupervised learning in neural computation. *Theoretical Computer Science*, 287(1):187–207, September 2002. 90

[152] Randall C. O'Reilly. Biologically plausible error-driven learning using local activation differences: the generalized recirculation algorithm. *Neural Computation*, 8(5):895–938, July 1996. 48

[153] Randall C. O'Reilly. *The LEABRA model of neural interactions and learning in the neocortex.* PhD thesis, Carnegie Mellon University, 1996. 48

[154] Nonna A. Otmakhova and John E. Lisman. D1/D5 dopamine receptor activation increases the magnitude of early long-term potentiation at CA1 hippocampal synapses. *The Journal of Neuroscience*, 16(23):7478–86, December 1996. 18

[155] N. G. Pavlidis, D. K. Tasoulis, V. P. Plagianakos, G. Nikiforidis, and M. N. Vrahatis. Spiking neural network training using evolutionary algorithms. In *International Joint Conference on Neural Networks*, volume 4, pages 2190–2194. IEEE, 2005. 48

[156] Verena Pawlak and Jason N. D. Kerr. Dopamine receptor activation is required for corticostriatal spike-timing-dependent plasticity. *Journal of Neuroscience*, 28(10):2435–46, March 2008. 18

[157] Jean-Pascal Pfister, David Barber, and Wulfram Gerstner. Optimal hebbian learning: a probabilistic point of view. In *International Conference on Artificial Neural Networks and Neural Information Processing*, pages 92–98. Springer-Verlag, 2003. 47

[158] Jean-Pascal Pfister and Wulfram Gerstner. Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of Neuroscience*, 26(38):9673–82, September 2006. 16, 96, 99, 101

[159] Jean-Pascal Pfister, Taro Toyoizumi, David Barber, and Wulfram Gerstner. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Computation*, 18(6):1318–48, June 2006. 47

[160] Anthony G. Phillips, Giada Vacca, and Soyon Ahn. A top-down perspective on dopamine, motivation and memory. *Pharmacology, Biochemistry, and Behavior*, 90(2):236–49, August 2008. 71

[161] Filip Ponulak. *Supervised learning in spiking neural networks with ReSuMe method.* Phd, Poznan University of Technology, 2006. 46, 47

[162] Wiebke Potjans, Abigail Morrison, and Markus Diesmann. A spiking neural network model of an actor-critic learning agent. *Neural Computation*, 339:301–339, 2009. x, 77, 78

[163] R. Quian Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–7, June 2005. 26

[164] Wilfrid Rall and Gordon M. Shepherd. Theoretical reconstruction of field potentials and dendrodendritic synaptic interactions in olfactory bulb. *Journal of Neurophysiology*, 31(6):884, 1968. 9

[165] Rajesh P. N. Rao and Terrence J. Sejnowski. Spike-timing-dependent Hebbian plasticity as temporal difference learning. *Neural Computation*, 13(10):2221–37, October 2001. 75

[166] Daniel Rasmussen and Chris Eliasmith. A neural model of rule generation in inductive reasoning. *Topics in Cognitive Science*, 3(1):140–153, January 2011. 55

[167] Alexander Rauch, Giancarlo La Camera, Hans-Rudolf Luscher, Walter Senn, and Stefano Fusi. Neocortical pyramidal cells respond as integrate-and-fire neurons to in vivo-like input currents. *Journal of Neurophysiology*, 90(3):1598–612, September 2003. 21

[168] John N. J. Reynolds and Jeffery R. Wickens. Dopamine-dependent plasticity of corticostriatal synapses. *Neural Networks*, 15(4-6):507–21, 2002. 18

[169] Martin Riedmiller. Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. *Learning*, pages 317–328, 2005. 73

[170] Patrick D. Roberts. Computational consequences of temporally asymmetric learning rules: I. Differential Hebbian learning. *Journal of Computational Neuroscience*, 7(3):235–46, 1999. 8

[171] Edmund T. Rolls. Absolute refractory period of neurons involved in MFB self-stimulation. *Physiology & Behavior*, 7(3):311–5, September 1971. 23

[172] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 41

[173] Ausra Saudargiene, Bernd Porr, and Florentin Wörgötter. How the shape of pre- and postsynaptic signals can influence STDP: a biophysical model. *Neural Computation*, 16(3):595–625, 2004. 8

[174] Ausra Saudargiene, Bernd Porr, and Florentin Wörgötter. Synaptic modifications depend on synapse location and activity: a biophysical model of STDP. *Bio Systems*, 79(1-3):3–10, 2005. 8

[175] Nestor A. Schmajuk. Classical conditioning. *Scholarpedia*, 3(3):2316, 2008. 70

[176] Benjamin Schrauwen and Jan Van Campenhout. Improving SpikeProp: enhancements to an error-backpropagation rule for spiking neural networks. In *Prorisc Workshop*, volume 11, pages 301–305, 2004. 46

[177] Wolfram Schultz. Dopamine neurons and their role in reward mechanisms. *Current Opinion in Neurobiology*, 7(2):191–7, April 1997. 62, 71

[178] Wolfram Schultz. Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80(1):1–27, July 1998. 76

[179] Wolfram Schultz, Peter Dayan, and P. Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, March 1997. 62, 71, 72

[180] Wolfram Schultz, Leon Tremblay, and Jeffrey R. Hollerman. Reward prediction in primate basal ganglia and frontal cortex. *Neuropharmacology*, 37(4-5):421–9, 1998. 47, 71

[181] Harel Shouval, Nathan Intrator, and Leon N. Cooper. BCM network develops orientation selectivity and ocular dominance in natural scene environment. *Vision Research*, 37(23):3339–42, December 1997. 94

[182] Hava T. Siegelmann and Eduardo D. Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991. 43

[183] Jesper Sjöström and Wulfram Gerstner. Spike-timing dependent plasticity. *Scholarpedia*, 5(2):1362, 2010. 13, 14, 16, 98

[184] Per Jesper Sjöström and Sacha B. Nelson. Spike timing, calcium signals and synaptic plasticity. *Current Opinion in Neurobiology*, 12(3):305–314, June 2002. 15

[185] Per Jesper Sjöström, Ede A. Rancz, Arnd Roth, and Michael Hausser. Dendritic excitability and synaptic plasticity. *Physiological Reviews*, 88(2):769, 2008. 14, 15

[186] Per Jesper Sjöström, Gina G. Turrigiano, and Sacha B. Nelson. Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron*, 32(6):1149–64, December 2001. 16, 17, 97

[187] Sen Song, Kenneth D. Miller, and Larry F. Abbott. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–26, September 2000. 96

[188] Henning Sprekeler, Christian Michaelis, and Laurenz Wiskott. Slowness: an objective for spike-timing-dependent plasticity? *PLoS Computational Biology*, 3(6):e112, June 2007. 98

[189] John E. R. Staddon and Yael Niv. Operant conditioning. *Scholarpedia*, 3(9):2318, 2008. 70

[190] Charles F. Stevens. A million dollar question: does LTP = memory? *Neuron*, 20(1):1–2, January 1998. 19

[191] Terrence C. Stewart, Trevor Bekolay, and Chris Eliasmith. Neural representations of compositional structures: representing and manipulating vector spaces with spiking neurons. *Connection Science*, 23(2):145–153, June 2011. 26

[192] Terrence C. Stewart, Xuan Choo, and Chris Eliasmith. Dynamic behaviour of a spiking model of action selection in the basal ganglia. In *ICCM*, 2010. 55, 79, 80

[193] Greg J. Stuart, Nelson Spruston, Bert Sakmann, and Michael Häusser. Action potential initiation and backpropagation in neurons of the mammalian CNS. *Trends in Neurosciences*, 20(3):125–31, March 1997. 9

[194] Roland E. Suri and Wolfram Schultz. Learning of sequential movements by neural network model with dopamine-like reinforcement signal. *Experimental Brain Research*, 121(3):350–4, August 1998. 73

[195] Roland E. Suri and Wolfram Schultz. A neural network model with dopamine-like reinforcement signal that learns a spatial delayed response task. *Neuroscience*, 91(3):871–90, January 1999. 73

[196] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988. 62, 68

[197] Richard S. Sutton and Andrew G. Barto. Toward a modern theory of adaptive networks: expectation and prediction. *Psychological Review*, 88(2):135–70, March 1981. 7

[198] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*, volume 3. Cambridge University Press, September 1999. x, 63, 69

[199] Takeo Suzuki, Masami Miura, Kin-ya Nishimura, and Toshihiko Aosaki. Dopamine-dependent synaptic plasticity in the striatal cholinergic interneurons. *The Journal of Neuroscience*, 21(17):6492–501, September 2001. 18

[200] J. L. Swanson-Park, C. M. Coussens, S. E. Mason-Parker, C. R. Raymond, E. L. Hargreaves, M. Dragunow, A. S. Cohen, and W. C. Abraham. A double dissociation within the hippocampus of dopamine D1/D5 receptor and beta-adrenergic receptor contributions to the persistence of long-term potentiation. *Neuroscience*, 92(2):485–97, January 1999. 18

[201] Dharmesh R. Tailor, Leif H. Finkel, and Gershon Buchsbaum. Color-opponent receptive fields derived from independent component analysis of natural images. *Vision Research*, 40(19):2671–6, January 2000. 94

[202] Carlo A. Terzuolo and T. Araki. An analysis of intra- versus extracellular potential changes associated with activity of single spinal motoneurons. *Annals of the New York Academy of Sciences*, 94(2):547–558, 1961. 9

[203] Edward Lee Thorndike. Animal intelligence: an experimental study of the associate processes in animals. *Psychological Review*, 2(4):1–8, 1898. 62

[204] Simon J. Thorpe. Spike arrival times: a highly efficient coding scheme for neural networks. *Parallel Processing in Neural Systems*, pages 91–94, 1990. 23, 45

[205] Taro Toyoizumi, Jean-Pascal Pfister, Kazuyuki Aihara, and Wulfram Gerstner. Generalized Bienenstock-Cooper-Munro rule for spiking neurons that maximizes information transmission. *Proceedings of the National Academy of Sciences of the United States of America*, 102(14):5239–44, April 2005. 96

[206] Hsing-Chen Tsai, Feng Zhang, Antoine Adamantidis, Garret D. Stuber, Antonello Bonci, Luis de Lecea, and Karl Deisseroth. Phasic firing in dopaminergic neurons is sufficient for behavioral conditioning. *Science*, 324(5930):1080–4, May 2009. 72

[207] John N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16(3):185–202, 1994. 68

[208] Gina G. Turrigiano. The self-tuning neuron: synaptic scaling of excitatory synapses. *Cell*, 135(3):422–35, October 2008. 17

[209] Gina G. Turrigiano, Kenneth R. Leslie, Niraj S. Desai, Lana C. Rutherford, and Sacha B. Nelson. Activity-dependent scaling of quantal amplitude in neocortical neurons. *Nature*, 391(6670):892–6, February 1998. 17

[210] Nathaniel N. Urban and German Barrionuevo. Induction of Hebbian and non-hebbian mossy fiber long-term potentiation by distinct patterns of high-frequency stimulation. *The Journal of Neuroscience*, 16(13):4293–9, July 1996. 16

[211] Robert Urbanczik and Walter Senn. Reinforcement learning in populations of spiking neurons. *Nature Neuroscience*, 12(3):250–2, March 2009. 76

[212] Matthijs A. A. van Der Meer and A. David Redish. Ventral striatum: a critical look at models of learning and evaluation. *Current Opinion in neurobiology*, 21(3):387–392, March 2011. 77

[213] Rufin Van Rullen, Rudy Guyonneau, and Simon J. Thorpe. Spike times make sense. *Trends in Neurosciences*, 28(1):1–4, January 2005. 45

[214] Jonathan D. Victor. Spike train metrics. *Current Opinion in Neurobiology*, 15(5):585–92, October 2005. 45

[215] Hongyan Wang and John J. Wagner. Priming-induced shift in synaptic plasticity in the rat hippocampus. *Journal of Neurophysiology*, 82:2024–2028, 1999. 93

[216] Huai-Xing Wang, Richard C. Gerkin, David W. Nauen, and Guo-Qiang Bi. Coactivation and timing-dependent integration of synaptic potentiation and depression. *Nature Neuroscience*, 8(2):187–93, February 2005. 16

[217] Jack Waters, Andreas Schaefer, and Bert Sakmann. Backpropagating action potentials in neurones: measurement, mechanisms and potential functions. *Progress in Biophysics and Molecular Biology*, 87(1):145–70, January 2005. 10

[218] Alanna J. Watt, Mark C. W. van Rossum, Katrina M. MacLeod, Sacha B. Nelson, and Gina G. Turrigiano. Activity coregulates quantal AMPA and NMDA currents at neocortical synapses. *Neuron*, 26(3):659–70, June 2000. 17

[219] J. Wickens and R. Kötter. Cellular models of reinforcement. In *Models of information processing in the basal ganglia*, pages 187–214. The MIT Press, 1995. 18

[220] Jianguo Xin and Mark J. Embrechts. Supervised learning with spiking neural networks. In *International Joint Conference on Neural Networks*, volume 3, pages 1772–1777. IEEE, February 2001. 48

[221] Z. Zainuddin, N. Mahat, and Y. Abu Hassan. Improving the convergence of the backpropagation algorithm using local adaptive techniques. *World Academy of Science, Engineering and Technology*, (4):1200–1204, 2005. 43, 44

[222] Li I. Zhang, Huizhong W. Tao, Christine E. Holt, William A. Harris, and Mu-ming Poo. A critical window for cooperation and competition among developing retinotectal synapses. *Nature*, 395(6697):37–44, 1998. 14