Learning Spatio-Temporally Encoded Pattern Transformations in Structured Spiking Neural Networks

Brian Gardner

Submitted for the Degree of Doctor of Philosophy from the University of Surrey

Department of Computer Science
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

March 2016
Summary

Increasing evidence indicates that biological neurons process information conveyed by the precise timings of individual spikes. Such observations have prompted studies on artificial networks of spiking neurons, or Spiking Neural Networks (SNNs), that use temporal encodings to represent input features. Potentially, SNNs used in this way are capable of increased computational power in comparison with rate-based networks.

This thesis investigates general learning methods for SNNs which utilise the timings of single and multiple output spikes to encode information. To this end, three distinct contributions to SNN learning are made as follows.

The first contribution is a proposed reward-modulated synaptic plasticity method for training SNNs to learn sequences of precisely-timed output spikes in response to spatio-temporal input patterns. Results demonstrate the high temporal accuracy of this method, even when synaptic weights in the network are modified by a delayed feedback signal. This method is potentially of biological significance, since synaptic strength modifications have been observed to be modulated by a reward signal, such as dopamine, in the nervous system.

The second contribution proposes two new supervised learning rules for SNNs that perform input-output transformations of spatio-temporal spike patterns. Simulations demonstrate the rules are capable of encoding large numbers of input patterns as precisely timed output spikes, comparing favourably with existing work.

The final contribution is a new supervised learning rule, termed MultilayerSpiker, for training SNNs containing hidden layers of spiking neurons to temporally encode spatio-temporal spike patterns using single or multiple output spikes. Simulations show MultilayerSpiker supports a very large number of encodings, that is a substantial improvement over existing spike-based multilayer rules, and provides increased classification accuracy when using the timings of multiple rather than single output spikes to identify input patterns.
Contents

1 Introduction .......................... 1

1.1 Objectives and Contributions ................. 2
1.2 Thesis Outline ........................ 4

2 Spiking Neurons ......................... 6

2.1 Biological Background ..................... 6

2.1.1 Neuron Structure .................. 7
2.1.2 Electrical Signalling ................. 7
2.1.3 Synapses and Chemical Signalling .......... 10

2.2 Spiking Neuron Models .................... 12

2.2.1 Leaky Integrate-and-Fire (LIF) Model .... 14
2.2.2 Spike Response Model (SRM) .......... 16
2.2.3 Escape Noise Model .............. 16
2.2.4 Further Related Models ............. 19

2.3 Spiking Network Structures ................ 20

2.3.1 Feed-Forward Networks ............. 20
2.3.2 Recurrent Networks ................ 22

2.4 Neural Coding ......................... 22

2.4.1 Rate Coding ......................... 23
2.4.2 Temporal Coding ........................ 24

2.5 Chapter Summary ........................ 25
## 3 Learning in Spiking Neural Networks

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Synaptic Plasticity</td>
<td>27</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Spike-Timing-Dependent Plasticity (STDP)</td>
<td>28</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Homeostatic Plasticity</td>
<td>31</td>
</tr>
<tr>
<td>3.2</td>
<td>Unsupervised Learning</td>
<td>32</td>
</tr>
<tr>
<td>3.2.1</td>
<td>STDP-Based Spike Pattern Learning</td>
<td>32</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Discussion</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>Reinforcement Learning</td>
<td>35</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Background</td>
<td>35</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Eligibility Traces</td>
<td>36</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Reward-modulated STDP (R-STDTP) Rule</td>
<td>37</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Reward-maximisation (R-max) Rule</td>
<td>39</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Discussion: R-STDTP and R-max</td>
<td>40</td>
</tr>
<tr>
<td>3.4</td>
<td>Supervised Learning</td>
<td>41</td>
</tr>
<tr>
<td>3.4.1</td>
<td>SpikeProp (Spike-based Backpropagation)</td>
<td>42</td>
</tr>
<tr>
<td>3.4.2</td>
<td>ReSuMe (Remote Supervised Learning Method)</td>
<td>44</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Chronotron (Gradient Descent Learning)</td>
<td>46</td>
</tr>
<tr>
<td>3.4.4</td>
<td>SPAN (Spike Pattern Association Neuron)</td>
<td>49</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Optimal STDP for Precise Spiking</td>
<td>52</td>
</tr>
<tr>
<td>3.4.6</td>
<td>Discussion</td>
<td>60</td>
</tr>
<tr>
<td>3.5</td>
<td>Chapter Summary</td>
<td>61</td>
</tr>
</tbody>
</table>

## 4 Reward-Modulated Learning for Precise Spiking

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>Methods</td>
<td>66</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Single Neuron Model</td>
<td>66</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Reward-modulated Synaptic Plasticity Rule</td>
<td>69</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Learning a Target Spike Train</td>
<td>71</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Plasticity Rules</td>
<td>75</td>
</tr>
</tbody>
</table>
## Contents

4.3 Simulation Results ........................................... 75
  4.3.1 Network Setup and Learning Task ....................... 76
  4.3.2 Learning Temporally Precise Spiking Patterns ........... 77

4.4 Discussion .................................................. 80

4.5 Chapter Summary ............................................ 82

5 Supervised Learning for Precise Spiking ...................... 84
  5.1 Introduction ................................................. 84
  5.2 Learning Theory ............................................. 86
    5.2.1 Single Neuron Model .................................... 86
    5.2.2 INSTantaneous-error (INST) Synaptic Plasticity Rule .... 89
    5.2.3 FILTered-error (FILT) Synaptic Plasticity Rule .......... 91
  5.3 Analysis of the Learning Rules ............................ 93
    5.3.1 Order of Postsynaptic Spikes ......................... 95
    5.3.2 Relative Timing between Spikes ....................... 98
    5.3.3 Temporally Contiguous Postsynaptic Spikes .......... 99
    5.3.4 Synaptic Variance ................................... 102
    5.3.5 Summary ........................................... 103
  5.4 Simulation Results ......................................... 104
    5.4.1 Network Setup .......................................... 104
    5.4.2 General Learning Task ................................. 105
    5.4.3 Performing a Single Input-Output Mapping ............ 106
    5.4.4 Impact of the Learning Rate ......................... 109
    5.4.5 Classifying Spike Patterns ........................... 111
    5.4.6 Input Noise .......................................... 117
  5.5 Discussion ................................................ 119
    5.5.1 Optimal Learning ...................................... 120
    5.5.2 Related Work ......................................... 121
    5.5.3 Biological Plausibility .............................. 123
  5.6 Chapter Summary ........................................... 123
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Schematic of a typical neuron structure.</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Example firing patterns of two types of neocortical neuron.</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>The time course of an action potential.</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Schematic of a chemical synapse.</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>Illustration of the escape noise model.</td>
<td>18</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of a soft threshold escape rate function.</td>
<td>19</td>
</tr>
<tr>
<td>2.7</td>
<td>Illustration of spiking neural network structures.</td>
<td>21</td>
</tr>
<tr>
<td>3.1</td>
<td>Different forms of STDP rules identified in the nervous system.</td>
<td>30</td>
</tr>
<tr>
<td>3.2</td>
<td>Example of competitive-STDP learning a spike pattern.</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>Illustration of learning by reward-modulated STDP.</td>
<td>38</td>
</tr>
<tr>
<td>4.1</td>
<td>Example of a stochastic neuron learning to map between an input spatio-temporal spike pattern and a target output spike train.</td>
<td>78</td>
</tr>
<tr>
<td>4.2</td>
<td>Performance comparison between two different stochastic neuron models when learning an increasing number of target output spikes.</td>
<td>79</td>
</tr>
<tr>
<td>4.3</td>
<td>Output spike rasters of two different stochastic neuron models when learning a target spike train containing 20 spikes.</td>
<td>80</td>
</tr>
<tr>
<td>5.1</td>
<td>Illustration of the time course of postsynaptic kernels used, and the resulting membrane potential of an SRM(_0) neuron.</td>
<td>90</td>
</tr>
<tr>
<td>5.2</td>
<td>Illustration of INST and FILT synaptic plasticity rules.</td>
<td>97</td>
</tr>
<tr>
<td>5.3</td>
<td>Dependence of synaptic weight change on the relative timing between a target output spike and an input spike.</td>
<td>99</td>
</tr>
<tr>
<td>5.4</td>
<td>Dependence of reversal lag time (s_{\text{switch}}) on filter time constant (\tau_q), at which point the direction of a synaptic weight change is reversed.</td>
<td>102</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.5</td>
<td>Example of INST and FILT neurons learning to map between a single input-output spike pattern pair.</td>
<td>107</td>
</tr>
<tr>
<td>5.6</td>
<td>Synaptic weight distributions of INST and FILT neurons.</td>
<td>108</td>
</tr>
<tr>
<td>5.7</td>
<td>Network error as a function of the learning rate for INST, FILT and CHRON neurons.</td>
<td>110</td>
</tr>
<tr>
<td>5.8</td>
<td>Performance of INST, FILT and CHRON neurons on a classification task using precise temporal coding.</td>
<td>113</td>
</tr>
<tr>
<td>5.9</td>
<td>Comparative memory capacities of INST, FILT and CHRON neurons as a function of their temporal encoding precision.</td>
<td>115</td>
</tr>
<tr>
<td>5.10</td>
<td>Performance of INST, FILT and CHRON neurons when learning multiple target output spikes.</td>
<td>116</td>
</tr>
<tr>
<td>5.11</td>
<td>Illustration of learning paradigm based on Gaussian input noise.</td>
<td>118</td>
</tr>
<tr>
<td>5.12</td>
<td>Performance of INST, FILT and CHRON neurons when classifying noisy patterns.</td>
<td>119</td>
</tr>
<tr>
<td>6.1</td>
<td>Example of output layer weight update for MultilayerSpiker learning rule.</td>
<td>132</td>
</tr>
<tr>
<td>6.2</td>
<td>Example of hidden layer weight update for MultilayerSpiker.</td>
<td>137</td>
</tr>
<tr>
<td>6.3</td>
<td>Example of MultilayerSpiker learning to map between an input spike pattern and a target output spike train.</td>
<td>142</td>
</tr>
<tr>
<td>6.4</td>
<td>Evolution and final distribution of synaptic weights in a trained multilayer network.</td>
<td>143</td>
</tr>
<tr>
<td>6.5</td>
<td>Robustness of MultilayerSpiker to input noise.</td>
<td>145</td>
</tr>
<tr>
<td>6.6</td>
<td>Learning the linearly non-separable exclusive-or (XOR) function as a temporal code.</td>
<td>148</td>
</tr>
<tr>
<td>6.7</td>
<td>Performance of various network structures when learning mappings of multiple input-output pattern pairs.</td>
<td>149</td>
</tr>
<tr>
<td>6.8</td>
<td>Performance of MultilayerSpiker on a classification task with fully temporal coding.</td>
<td>152</td>
</tr>
<tr>
<td>6.9</td>
<td>Performance of MultilayerSpiker as applied to a noisy synthetic dataset.</td>
<td>155</td>
</tr>
<tr>
<td>6.10</td>
<td>Learning a mapping between a spatio-temporal input-output spike pattern pair.</td>
<td>157</td>
</tr>
</tbody>
</table>
6.11 Dependence of performance on the network structure when learning spatio-temporal spike pattern transformations. . . . . . . . . . 159
6.12 Illustration of the bio-backprop learning rule: a biologically inspired implementation of spike-based backpropagation. . . . . . . . . . 161
6.13 Performance of bio-backprop as a function of the number of input patterns learned, and network structure. . . . . . . . . . 163
Chapter 1

Introduction

Spiking Neural Networks (SNNs) represent a third generation of artificial neural network models (Maass, 1997), conferring a substantial improvement with regard to the biological realism of neural simulations in comparison with previous generation models. In particular, SNNs incorporate the time-domain as part of their operational model, such that spiking neurons constituting a network can communicate with each other by transmitting pulsed signals or ‘spikes’ over time. In this way, an SNN is capable of encoding information by the precise timings of individual spikes, theoretically enabling superior computational power in comparison with a previous generation model such as a sigmoidal neural network (Maass, 1997).

Precise spike timing as a means to convey information in neural networks is biologically supported (van Rullen et al., 2005) and is demonstrated to be advantageous over frequency-based codes by processing input features on a much shorter time-scale (Johansson & Birznieks, 2004; Gollisch & Meister, 2008). Moreover, the temporal precision by which individual spikes are reproduced can be very high: in some cases to within just one millisecond (Mainen & Sejnowski, 1995). For these reasons, much research attention is currently focused on the development of spike-based learning rules for SNNs, in order to more realistically model how learning and memory formation might take place in the nervous system. However, formulating effective learning rules that utilise a fully temporal code for SNNs is no small challenge owing to their inherent complexity.

Learning rules for SNNs have been formulated in supervised and reinforcement
settings, and work to train networks, through modifying the weights between neurons, to learn transformations between spatio-temporal input and desired output spike patterns (Kasinski & Ponulak, 2006); learning input-output transformations is considered a generic processing task of biological neurons, and in most cases can straightforwardly be adapted to more specific learning tasks of interest. Currently, the majority of learning rules are restricted to just single-layer SNN structures: few rules are applicable to more complex SNN structures, for example those containing hidden layers of spiking neurons (Güttig, 2014). Moreover, many spike-based learning rules lack analytical rigour in their formulation, meaning the optimality of their solutions cannot be guaranteed in general. Hence, in order to better realise the large potential in computational power offered by SNNs, it is desirable that a spike-based learning rule be proposed that is theoretically justifiable as well as versatile in its applicability.

This thesis aims to address the identified shortcomings of current learning rules for SNNs, by demonstrating that improved spike-based rules can not only be implemented in complex network structures but also provide optimal solutions when learning spike pattern transformations. Because of the significant role individual spikes play in neural processing this thesis also places a strong emphasis on temporal coding as a part of SNN learning.

1.1 Objectives and Contributions

At present, few learning rules have been established for SNNs that are technically efficient and yet versatile in their deployment. To take better advantage of the predicted high computational power of SNNs, it would be ideal to formulate general purpose learning rules that are applicable to various network structures and can convey information using the precise timings of spikes. Therefore, the overall aim of this thesis is to develop biologically-inspired learning rules for structured SNNs that utilise spike-based encoding for the purpose of high performance neural computation.

The specific objectives of this thesis are summarised as follows:

- Implement a fully temporal coding scheme in combination with spike-based
learning rules, to explore the effect of single- and multi-spike based encodings on the accuracy of pattern recognition in SNNs.

- Formulate theoretically justified, spike-based learning rules for SNNs to ensure the optimality of resulting solutions are generally guaranteed.

- Formulate a more generalised spike-based learning rule that is applicable to structured SNNs containing hidden layers of spiking neurons, similar in concept to the backpropagation method as applied to rate-coded networks.

- Respect biological constraints of real neural networks, to support the use of derived spike-based learning rules as an explanatory model of neurobiological processing.

These objectives have led to several novel contributions, with resulting publications listed at the end of this thesis. Each contribution is now briefly discussed in turn.

The first contribution is the application of an existing reward-modulated synaptic plasticity rule, specifically the Reward-maximisation (R-max) rule (Frémaux et al., 2010), to learning temporally precise sequences of output spikes by delayed reinforcement in an SNN. Our selection of the R-max rule is motivated by a desire for high biological realism in this initial work, and is inspired by emerging evidence suggesting that reward-modulated synaptic plasticity underlies behavioural learning in structures of the brain called the basal ganglia (Chakravarthy et al., 2010). This contribution is among the first to consider reinforcement-based learning of large numbers of desired output spike times in an SNN, and introduces a new, effective method for this purpose (Gardner & Grünig, 2013).

The second contribution is the formulation of two new supervised learning rules for training SNNs to transform between spatio-temporal input spike patterns and desired output spike trains. These learning rules are theoretically justified, and may be implemented online or offline depending on the level of biological realism that is desired. The performance of each rule is compared against that of the Chronotron: an existing supervised method which has previously been demonstrated to provide a very high network capacity in terms of the maximum number of input patterns that can be memorised (Florian, 2012).
Chapter 1. Introduction

The third contribution is the formulation of a new supervised multilayer learning rule, termed MultilayerSpiker (Gardner et al., 2015), for feed-forward SNNs contain hidden layer spiking neurons. The rule generalises the probabilistic learning method proposed by Pfister et al. (2006) from single-layer SNNs to multiple layers by combining a standard gradient ascent procedure with the method of backpropagation. MultilayerSpiker has strong theoretical justification, and works to modify synaptic weights in each layer of an SNN such that the likelihood of generating a desired spatio-temporal output spike pattern is maximised. The performance of the proposed learning rule is tested through several input-output spike pattern transformation tasks: both in terms of its final output accuracy and its convergence time. Finally, to better respect biological constraints of the actual nervous system, a biologically plausible implementation of the MultilayerSpiker rule is also proposed that mimics a reward-modulated learning paradigm.

1.2 Thesis Outline

The rest of this thesis is organised as follows.

Chapter 2 provides background information on the biological neuron, including its information processing functionality as carried out via electrical signalling. This biological background is then used to inform descriptive mathematical models of spiking neurons which constitute an SNN. Next, two main structural classes of SNNs are reviewed, including feed-forward and recurrent network structures. Finally, an overview is provided of two fundamental neural coding schemes used for neural computation: rate- and temporal-based coding.

Chapter 3 reviews two prominent synaptic plasticity mechanisms identified in the nervous system: Spike-Timing-Dependent Plasticity (STDP) and homeostatic plasticity. This is then followed by a review of biologically-inspired unsupervised, reinforcement and supervised learning methods that have been proposed for training synapses in an SNN.

Chapter 4 presents a new reinforcement-based learning method for training SNNs to learn temporally precise sequences of output spikes in response to a spatio-temporal input pattern. This method is biologically plausible by incor-
porating delayed reward signals to guide synaptic plasticity, and by realistically simulating background noise during network learning.

Chapter 5 proposes two new supervised learning rules for performing efficient input-output spike pattern transformations in an SNN. The rules are extensively tested on a generic spike pattern classification task and benchmarked against an existing high-performance learning rule.

Chapter 6 introduces the new MultilayerSpiker learning rule for feed-forward SNNs containing hidden layer spiking neurons. The performance of the rule is tested by applying it to a variety of classification tasks, including measuring a trained SNN’s memory capacity as defined by the maximum number of input patterns it can learn to memorise. This chapter also proposes a more biologically plausible implementation of spike-based backpropagation.

Chapter 7 provides the concluding remarks of this thesis, and discusses promising directions for future research.
Chapter 2

Spiking Neurons

The human central nervous system contains on the order of $10^{11}$ neurons which together form up to $10^{15}$ synaptic connections, and is responsible for directing behaviour in response to received sensory information. Understanding how such stimulus-response associations relate to the functions of constitutive neurons is a major aim of theoretical neuroscience, and has motivated the concept of a spiking neuron as an idealised model of a biological neuron.

This chapter begins by providing an overview of the biological neuron as found in the nervous system, with respect to its information processing capabilities via electrical signalling. This biological background is then used to form the basis of mathematical models for spiking neurons which transmit information via electrical pulses or ‘spikes’. Next, two main Spiking Neural Network (SNN) architectures are discussed: those containing layers of spiking neuron that are connected in a feed-forward manner, and those that contain a ‘reservoir’ of spiking neurons with recurrent connections. Finally, the topic of neural coding is examined which seeks to elucidate the relationship between stimulus and neuronal response.

2.1 Biological Background

The nervous system consists of two fundamental cell types: neurons and glial cells, with glial cells existing in a much greater abundance than neurons (Kandel et al., 2000). Currently, glial cells are understood to act mainly in a supportive
role, for example Schwann cells which aid neural signal transmission (Kandel et al., 2000), while neurons function as the primary information processing units in the nervous system (Trappenberg, 2010); despite this, it is noted that a certain subtype of glial cell, termed astrocytes, could play a more direct role in information processing than previously thought (Perea et al., 2009). For the purposes of this thesis, however, only information processing carried out by neuronal networks is considered.

### 2.1.1 Neuron Structure

The structure of a typical neuron consists of four well-defined regions: the soma, dendrites, axon and presynaptic terminals (schematic shown in Fig. 2.1). The soma is the cell body, and is the metabolic center of the neuron; it contains the nucleus, as well as the cellular apparatus necessary for the maintenance of its structure and the execution of its functions (Kandel et al., 2000). The dendrites are cellular extensions which branch out, analogously to a tree, to receive incoming signals from other neurons. The axon extends away from the soma, appearing as a long tube-like structure, and in contrast with dendrites transmits signals generated by the neuron to other neighbouring neurons. Towards its end, the axon divides into several smaller branches which come into contact with the dendrites (or sometimes soma) of other neurons. These points of contact are termed synapses and consist of a presynaptic cell which transmits a signal, and a postsynaptic cell which receives the signal; it is at these points where the terminates in what are called presynaptic terminals, allowing for the transmission of information from one neuron to the next. A postsynaptic neuron typically receives signals from a large number of presynaptic neurons, often having on the order of $10^4$ synaptic connections in the vertebrate cortex (Kandel et al., 2000).

### 2.1.2 Electrical Signalling

Neurons communicate information via electrical signals, appearing as short electrical pulses with an amplitude close to 100 mV and lasting 1–2 ms in duration (Gerstner & Kistler, 2002). These brief electrical pulses, also commonly referred to as action potentials or spikes, typically appear similar in form: implying that
information is encoded not in their precise shape, but rather in the frequency with which they are generated or their precise timings. In this way, the spike is considered the fundamental unit of signal transmission in the brain (Gerstner & Kistler, 2002). Shown in Fig. 2.2 are example sequences of spikes, or spike trains, found in typical neocortical neurons.

In order to support the generation of action potentials a neuron must be electrically polarised, necessitating the maintenance of a certain potential difference across its cellular membrane. This potential difference, or membrane potential, is maintained by controlling the concentration gradients of various ions across the neuron’s membrane, and is achieved by the actions of ion channels that are embedded in the cellular membrane. Examples of ionic elements in the nervous system include sodium (Na$^+$), potassium (K$^+$), calcium (Ca$^{2+}$) and chloride (Cl$^-$) (Trappenberg, 2010). Several types of ion channel exist, with varying levels of biophysical complexity, but in general they all share a common purpose: which is to regulate the passage of ions entering or exiting a neuron to control the membrane potential (Trappenberg, 2010). When a neuron is unstimulated its resting potential is typically measured at around $-70$ mV inside the cellular membrane, relative to its external environment (Dayan & Abbott, 2001).

When a neuron’s membrane potential increases above a certain threshold level, usually around 10 mV above its resting potential, a positive feedback process is
Figure 2.2: Example firing patterns of two types of neocortical neuron: regular-spiking and fast-spiking. (A) In response to a constant current, a regular-spiking neuron initially generates output spikes with high frequency which eventually become more widely spaced out. (B) A fast-spiking neuron has a much higher firing rate than that of (A) in response to the same current stimulus. Regular-spiking behaviour is most commonly found in electrophysiological studies. Regular-spiking neurons are excitatory, such that they work to induce postsynaptic spiking, while fast-spiking neurons are of an inhibitory nature, and suppress postsynaptic activity. Reproduced from Fig. 1 in Connors & Gutnick (1990).

initiated and an action potential results. The time course of such an action potential can be characterised by three distinct phases: the rising phase, the falling phase and the undershoot phase. In the rising phase, the membrane potential rapidly increases to become more positively valued (depolarization). Once the membrane potential has peaked the falling phase follows, that is a sharp decrease in the potential to a (usually) more negative value (hyperpolarization). During the undershoot phase, the membrane potential temporarily drops beneath its usual resting potential (hyperpolarized), from where it then slowly returns to its resting value. This final phase also coincides with the refractory period: a time period lasting around 10 ms in which a subsequent action potential is impossible / difficult to initiate (Trappenberg 2010). Fig. 2.3 illustrates the time course of a typical action potential.

Action potentials originate at a trigger site called the axon hillock, or initial segment of the axon (see Fig. 2.1); it is from here that they propagate away from the soma and along the axon at speeds ranging from between 1 and 100 m/s (Kandel et al. 2000). Moreover, the speed of action potential propagation is boosted by the presence of a myelin sheath, which envelopes most axons in the nervous system and works to prevent signal loss by acting as an insulating layer.
Chapter 2. Spiking Neurons

Figure 2.3: The time course of a typical action potential, illustrating the rapid change in form over just a few milliseconds. Adapted from Fig. 2.6 in Trappenberg (2010).

Action potentials propagating along myelinated axons essentially ‘hop’ between nodes of Ranvier, or brief unmyelinated segments of an axon for signal regeneration, in a process called ‘saltatory conduction’ (Kandel et al., 2000). In this way, the amplitude of a travelling action potential is maintained at a constant value, thereby allowing it to travel over large distances without attenuation (Dayan & Abbott, 2001). Myelination of neuron axons is supported by certain subtypes of glial cell: Schwann cells in the peripheral nervous system, and oligodendrocytes in the central nervous system (Kandel et al., 2000).

2.1.3 Synapses and Chemical Signalling

Synapses are the interfaces between neurons, enabling the transfer of information from one neuron to the next, and are located at the points of contact between presynaptic axon terminals and postsynaptic dendrites and soma. There are two main types of synapse: chemical and electrical, although in the vertebrate brain the chemical synapse is most common (Gerstner & Kistler, 2002). The chemical synapse allows for the unidirectional propagation of an action potential from the presynaptic terminal to the postsynaptic dendrite, whereas the electrical synapse is bidirectional and allows action potentials to be transmitted both ways (Kandel et al., 2000). Relatively little is known about the functional consequences of electrical synapses, although they might be involved in the synchronization of
neurons (Gerstner & Kistler, 2002). Therefore, the attention of this thesis is restricted to the dynamics of just chemical synapses.

Shown in Fig. 2.4 is a schematic of a chemical synapse, consisting of a presynaptic terminal and receiving sites on a postsynaptic dendrite. In the presynaptic terminal, chemical substances called neurotransmitters are synthesised and stored in synaptic vesicles. Upon the arrival of an action potential a cascade of biochemical events are triggered, culminating in the release of the stored neurotransmitters into the synaptic cleft, that is the gap between the presynaptic terminal and postsynaptic dendrite; the size of this gap is very small, measuring only a few micrometers in distance between the pre- and postsynaptic membranes. Once released into the synaptic cleft the neurotransmitters diffuse across to the other side, binding to receptor sites on specialised (ligand-gated) ion channels embedded within the membrane of the postsynaptic dendrite. As a result, these ion channels allow an influx of ions from the surrounding extracellular fluid, leading to a graded change in membrane potential of the postsynaptic neuron. The response of the postsynaptic membrane potential to a presynaptic action potential is commonly referred to as the Postsynaptic Potential (PSP).

A wide variety of neurotransmitters are present in the nervous system. Common examples include small biomolecules such as glutamate (Glu) or gamma-
aminobutyric acid (GABA) (Trappenberg, 2010). Beyond neurotransmitters, another class of chemical messenger is identifiable: neuromodulators, which act in a more global manner by being able to influence multiple synapses belonging to large groups of neurons. An important example of a neuromodulator is dopamine (DA), which is involved in neural circuits relating to motivation, attention and goal-directed behaviour (Schultz et al., 1997).

The response of a postsynaptic membrane potential to a presynaptic action potential differs depending on the species of neurotransmitter received at its synapse. In the case of ion channels gated by Glu, the postsynaptic membrane potential responds with a positive increase, hence this synapse is an excitatory one. The response itself is referred to as an Excitatory Postsynaptic Potential (EPSP). By contrast, a synapse with postsynaptic ion channels gated by GABA are of an inhibitory nature, and effectively counteracts responses in the postsynaptic membrane potential triggered via excitatory synapses; this type of response is termed an Inhibitory Postsynaptic Potential (IPSP) (Trappenberg, 2010). Interestingly, the neuromodulator DA has several receptor subtypes, allowing it to effect either excitatory or inhibitory responses depending on the synaptic site at which it becomes available (Frank & Claus, 2006). As mentioned previously, there exist many types of chemical transmitter present in the nervous system, capable of eliciting excitatory or inhibitory responses; the examples provided here are far from exhaustive, and just indicate at the enormous complexity of synaptic processing carried out by the nervous system. However, the fundamental mechanisms by which the biological synapse operates have been established, which shall be used in the next section to form the basis for mathematical modelling.

2.2 Spiking Neuron Models

Broadly speaking there exist three generations of artificial neuron model which constitute the computational units of a neural network, each aiming to replicate the essential functions of biological neurons in the nervous system (Maass, 1997). The first generation model is based on the McCulloch-Pitts neuron, which has since been developed into the more familiar perceptron (Minsky & Papert, 1987). Highly successful learning rules have been developed for perceptrons, and impor-
tantly have been extended to multilayer network structures, which are capable of generating any Boolean function subject to a sufficient number of hidden layer neurons (Bishop, 1995). Perceptrons mimic their biological counterparts in the sense that they take in a weighted input vector (synaptic scaling of input signals at dendrites) to then be equally summed over (as performed at the soma). The result of this summation is then thresholded to provide a binary output, using a step function as the neuron’s activation function (analogous to the functions of the axon hillock). Despite this, and unlike real neurons, perceptrons neglect the time domain, and are therefore restricted to processing inputs and producing binary outputs in an iterative manner.

The second generation neuron model instead uses a smoothed “activation function” to transform a weighted sum of inputs to a continuous set of possible output values, using for example a sigmoid function to transform between analog input and output signals (Maass, 1997). Importantly, sigmoidal networks containing hidden layer neurons are universal approximators, as applied to continuous functions (Hornik, 1991). Characteristic of second generation models are their support for gradient descent based procedures, such as backpropagation for multilayer learning. From a biological perspective (Maass, 1997), second generation models can be seen to emulate neuronal rate coding, and in this sense can be considered more biologically plausible than first generation models. However, such an implementation requires a relatively long time period in order to reliably sample the firing rate of a neuron; by comparison, the mammalian cortex is capable of rapidly processing visual stimuli in just under 100 ms (S. J. Thorpe & Imbert, 1989).

Spiking neurons are the third generation neuron model, improving largely on the biological realism of its predecessors through its incorporation of the precise timings of individual spikes. In this way, they offer the greatest potential for insight into the information processing capabilities of neural circuits in the nervous system. Furthermore, an SNN is theoretically predicted to be at least as computationally powerful as a previous generation neural network model when utilising a temporal code, and yet requiring less spiking neurons as its computational units (Maass, 1997); in particular, their usage of a temporal code is widely considered to allow for much faster processing time of briefly presented stimuli (van Rullen et al., 2005). Despite these important advantages, spiking neurons
are somewhat more complex in their mathematical description, to the extent that there is a current lack of learning rules for SNNs that are as general domain as backpropagation algorithms are for rate-coded networks. It is the intention of this thesis to address this identified shortcoming, by proposing new learning methods for SNNs that are increasingly versatile in terms of their application while still taking advantage of their temporal processing capability.

This section now turns to reviewing implementations of spiking neuron models which provide a suitable trade-off between high biological realism and analytical tractability. These models shall then form the basis of our theoretical analysis of spike-based learning rules in the contribution chapters of this thesis.

2.2.1 Leaky Integrate-and-Fire (LIF) Model

The Leaky Integrate-and-Fire (LIF) neuron is one of the most commonly used spiking neuron models in computational neuroscience, owing to its relative simplicity and ease of analytical treatment. In all cases, LIF neurons are stimulated by either an external current or synaptic input from presynaptic neurons, and a characteristic threshold is used to define their output responses.

If we consider a single neuron, indexed by $i$, with a membrane potential $u_i(t)$ at time $t$, then its subthreshold dynamics can be expressed by a differential equation:

$$\tau_m \frac{du_i(t)}{dt} = -u_i(t) + R I_i(t), \quad (2.1)$$

where the neuron’s resting membrane potential is zero, and $\tau_m$ and $R$ are model parameters. The parameter $\tau_m$ is the membrane time constant, relating to the ‘leakage’ of charge across the neuron’s membrane when it is not at rest, and $R$ is the effective membrane resistance. Eq. (2.1) is the standard form for a LIF neuron, which analogously describes an electrical circuit containing a resistor in parallel with a capacitor that is charged by an external current $I_i(t)$ (Gerstner & Kistler [2002]).

The LIF model avoids modelling the precise form of an action potential, and instead characterizes spikes based on just their firing time. Hence, if we use $t_i^f$ to refer to the $f^{th}$ spike timing of a neuron $i$, then according to the LIF model the
emission of a spike is determined by a threshold criterion:

\[ t^f_i : \quad u_i(t^f_i) = \vartheta , \]  

where \( \vartheta \) is the neuron’s firing threshold. Upon the emission of an output spike, the postsynaptic membrane potential is immediately reset to a new value:

\[ \lim_{t \to t^f_i, t > t^f_i} u_i(t) = u_r , \]  

where \( u_r < \vartheta \). If desired, this reset can be sustained over a brief period: \( u_i(t) = u_r \) for \( t^f_i < t \leq t^f_i + \Delta_{\text{abs}} \), where \( \Delta_{\text{abs}} \) is an absolute refractory period. For times \( t > t^f_i + \Delta_{\text{abs}} \) the dynamics of the neuron is again defined by Eq. (2.1) until the next incidence of a threshold crossing.

In the context of this thesis, the external current \( I_i \) injected into the postsynaptic neuron is appropriately defined as a summation over currents contributed by each synapse, elicited by presynaptic spiking. This sum depends on the strengths of individual synapses, parametrised by a real valued synaptic weight value \( w_{ij} \) between the \( j^{\text{th}} \) presynaptic neuron. The response of a postsynaptic current to a presynaptic spike is modelled by the so called ‘alpha-function’, denoted by \( \alpha \), that is typically taken as an exponential decay with a time constant on the order of a few milliseconds \([\text{Trappenberg, 2010}]\). Assuming synaptic responses are non-interacting, then the total current is simply a linear combination of the synaptic terms:

\[ I_i(t) = \sum_j \sum_f w_{ij} \alpha(t - t^f_j) , \]  

where \( t^f_j \) refers to the \( f^{\text{th}} \) spike timing of a presynaptic neuron \( j \).

Modelling synaptic responses by Eq. (2.4) is computationally efficient, and is well suited to the formulation of learning rules through parameter optimization techniques such as by gradient descent. An alternative, more biologically motivated approach might instead model the observed probabilistic release of neurotransmitters from synaptic vesicles, termed stochastic synaptic transmission \([\text{Dayan & Abbott, 2001}]\); however, such an increase in precision usually comes at the cost of computational speed, making large-scale network simulations less feasible.
2.2.2 Spike Response Model (SRM)

The Spike Response Model (SRM) is a generalisation of the LIF model, and notationally differs by instead expressing the neuron’s membrane potential at time $t$ in terms of an integral over the past. The generality of SRM comes from its inclusion of refractoriness, which can be modelled when the membrane potential explicitly depends on previous output spike times.

The SRM describes the subthreshold evolution of a postsynaptic membrane potential $u_i$ in response to presynaptic spikes. When a presynaptic spike is received, $u_i$ is perturbed from its resting value of zero, after which it gradually returns to rest. The PSP kernel $\epsilon$ describes the time course of $u_i$ in response to a presynaptic spike. If the presynaptic input is sufficient to drive $u_i$ to a firing threshold $\vartheta$, then an output spike is generated by the postsynaptic neuron. The reset kernel $\kappa$ influences the behaviour of $u_i$ in response to an output spike, effectively describing the afterpotential of the postsynaptic neuron. Hence, if we consider a postsynaptic neuron with a last output spike at time $\hat{t}_i$, then its membrane potential at time $t$ is defined by (Gerstner & Kistler, 2002):

$$u_i(t) := \sum_j w_{ij} \sum_f \epsilon(t - \hat{t}_i, t - t^f_j) + \kappa(t - \hat{t}_i) ,$$

(2.5)

where all functions have a dependence on the time since a last output spike, $t - \hat{t}_i$. The response functions are defined such that $\epsilon, \kappa \to 0$ for $(t - t^f_j), (t - \hat{t}_i) \to \infty$.

Unlike the LIF model discussed previously, the threshold is not necessarily fixed, and may be dynamically adjusted to alter the postsynaptic neuron’s spiking behaviour. For example, if an absolute refractory period is required, then $\vartheta$ can temporarily be set to a large positive value over selected time intervals $t^f_i < t \leq t^f_i + \Delta_{\text{abs}}$ (Gerstner & Kistler, 2002).

2.2.3 Escape Noise Model

The firing activity of a biological neuron is highly variable, and is attributable, at least in part, to the continual bombardment of spikes originating from the tens of thousands of presynaptic connections it typically receives. This source of
background noise, referred to as ‘stochastic spike arrival’, is unlikely to be purely ‘nuisance noise’ that hampers neural processing in the nervous system; instead, it is more plausible that such background activity actually conveys meaningful signals processed by different neural pathways (Faisal et al., 2008). This idea relates to the massively-parallel processing nature of the brain, that must deal with continuous, overlapping streams of information transmitted by a variety of different sources. However, with respect to individual neuron modelling, it is impractical to consider such large-scale dynamics for driving variable spiking activity. A more practical, phenomenological approach would instead introduce randomly generated noise as part of a model to mimic the observed variability of biological neurons.

There are a number of approaches to including noise in a spiking neuron model: a key example is the escape noise model defined in Gerstner & Kistler (2002). Escape noise assumes a ‘noisy threshold’ for a neuron, such that the neuron’s firing threshold effectively fluctuates about some reference value as a result of random background activity. In this way, an output spike may be generated by an escape noise neuron even when its membrane potential is below the formal firing threshold $\vartheta$. This idea is formalised by defining a probability density $\rho$ for distributing output spikes, that has a functional dependence on the momentary distance between the neuron’s (noiseless) membrane potential and threshold:

$$\rho(t) = g(u(t) - \vartheta), \quad (2.6)$$

where $u$ is defined by either the LIF model or SRM (see Eqs. (2.1) and (2.5), respectively). An illustration of this process is shown in Fig. 2.5. The arbitrary function $g$ is the ‘escape rate’, similar to that used to describe chemical reaction processes (van Kampen, 1992), and ideally is defined such that $g \to 0$ for $u \to -\infty$ (Gerstner & Kistler, 2002). The probability density $\rho$, also referred to as the stochastic intensity, is the likelihood of generating an output spike per unit time, and is interpreted as the neuron’s instantaneous firing density. The probability of generating an output spike at time $t$ is given by

$$\Pr \{ t' \in [t, t + \delta t]\} = 1 - \exp(-\delta t \rho(t)). \quad (2.7)$$

where $\delta t$ is a small time interval.
Chapter 2. Spiking Neurons

Figure 2.5: Illustration of the escape noise model for probabilistically generating neural spikes. A neuron with membrane potential $u$ can fire at time $t$ with probability density $\rho(t) = g(u(t) - \vartheta)$, even if its formal firing threshold $\vartheta$ hasn’t been reached. The neuron’s last spike time is denoted by $\hat{t}$. Reproduced from Fig. 5.5 in Gerstner & Kistler (2002).

A variety of choices are available to define the escape rate function $g$. A common selection is to take an exponential dependence:

$$g(u(t) - \vartheta) = \rho_0 \exp\left(\frac{u(t) - \vartheta}{\Delta u}\right), \tag{2.8}$$

where $\rho_0$ is the instantaneous firing density at threshold, and $\Delta u$ is a parameter which determines the ‘smoothness’ of the threshold. Interestingly, for suitable parameter choices of $\rho_0$ and $\Delta u$, Eq. (2.8) has been shown to well approximate the variable firing activity of neurons as recorded in vivo (Jolivet et al., 2006). An example of the exponential escape rate function defined by Eq. (2.8) is shown in Fig. 2.6.

Owing to its relative simplicity, the escape noise neuron represents an ideal choice for simulating background noise during simulations. Beyond its application as a stochastic spike generator, the escape noise model is also well suited to theoretical analysis by establishing a smooth functional dependence of output activity on internal network parameters; this shall become more apparent in subsequent chapters when escape noise is applied to formulating learning rules for SNNs.

Finally, it is worth mentioning that although background noise can clearly be detrimental to neural processing, such as through signal degradation or by jit-
Figure 2.6: Example of an escape rate with exponential dependence (see Eq. (2.8)). The instantaneous firing density \( \rho = g(u - \vartheta) \) rises exponentially with increasing \( u \). Different colour curves correspond to different choices of the threshold parameter \( \Delta u \); larger \( \Delta u \) results in a ‘smoother’ firing threshold, resulting in more variable output spiking. The rate at threshold is set to \( \rho_0 = 0.1 \text{ ms}^{-1} \).

2.2.4 Further Related Models

The LIF model and SRM represent just two possible choices for determining a postsynaptic neuron’s membrane potential. Other well-known models include the Izhikevich neuron \(^{[\text{Izhikevich 2003}]}\), the theta-neuron (or Ermentrout-Kopell canonical model) \(^{[\text{Ermentrout & Kopell 1986}]}\) and the Hodgkin-Huxley model \(^{[\text{Dayan & Abbott 2001}]}\). The Izhikevich neuron is defined by a two-dimensional system of differential equations, and is capable of reproducing the firing patterns of all known types of cortical neuron \(^{[\text{Izhikevich 2004}]}\). This model is computationally efficient, and as such is well suited for simulating the dynamics of large-scale networks of neurons with good biological plausibility \(^{[\text{Izhikevich 2006}]}\). By contrast, the theta-neuron is more simply defined by a one-dimensional differential equation, which depends on a single state variable. Also, given its non-linearity, a theta-neuron is still capable of exhibiting bursting behaviour unlike
the linear LIF model. Finally, the Hodgkin-Huxley model can exhibit most known behaviours of cortical neurons, and is also the most biophysically meaningful neuron model ([Dayan & Abbott, 2001]). The disadvantage of this model, however, is its computational complexity, requiring around 240 times more floating point operations than the LIF model ([Izhikevich, 2004]).

2.3 Spiking Network Structures

There exist a large number of different ways in which spiking neurons constituting a neural network can be organised in terms of their connectivity. Broadly speaking, however, the structure of an SNN can be categorised into one of two main types: that of a feed-forward nature, where neural signals propagate in one direction only, and the other recurrent, which allows signals to propagate in both directions for more dynamical behaviour. This section briefly reviews each type in turn.

2.3.1 Feed-Forward Networks

The simplest type of network has a feed-forward structure, where layers of neurons forwardly connect with those in subsequent layers. In the standard architecture of a feed-forward network there exists a single input and output layer, which presents input patterns to the network and determines its output responses, respectively ([Trappenberg, 2010]). Additional to this there can exist any number of intermediate layers between the input and output layers, which contain hidden neurons. As mentioned in the previous section, the technique of backpropagation has demonstrated large success in training multilayer networks containing layers of hidden, rate-based neurons; in particular, multilayer networks can approximate any continuous function to arbitrary precision for a suitable choice of neural activation function ([Hornik, 1991]). An illustration of a typical feed-forward multilayer SNN architecture is shown in Fig. 2.7A.
Chapter 2. Spiking Neurons

Figure 2.7: Illustration of example SNN structures and their operation. (A) Feed-forward structure: neurons in the input layer are fully connected to neurons in the hidden layer, which in turn project to readout neurons in the output layer. Input layer neurons perform the task of presenting spatio-temporal spike patterns to the network, to be processed by neurons in downstream layers. Typically, the task of the network is to learn to map between spatio-temporal input and output spike patterns by optimising synaptic weights between neuron layers. (B) Recurrent reservoir: input neurons transmit time series data (spike trains) to a random subset of recurrently-connected reservoir neurons. These reservoir neurons act as a liquid filter: mapping the input data to a higher dimensional space to increase the separation between input classes. The output activity of reservoir neurons is transmitted to a group of readout neurons, which determine the network’s response to the input data. Readout synapses are trained to produce desired network responses.
2.3.2 Recurrent Networks

Neural networks with recurrent connections exhibit more complex, dynamical behaviour than those with feed-forward connections, allowing for an internal, transient memory of input patterns that have previously been presented to the network. This behaviour emerges from the propagation of signals in both directions of the network, resulting from loops that are formed by the connections between neurons. An important example of a computing paradigm that takes advantage of this dynamical property is reservoir computing (Jaeger 2001; Maass et al. 2002; Chrol-Cannon & Jin 2014), which works by transforming sequential input data into a state of dynamic activity within a ‘reservoir’ of recurrently-connected neurons. This reservoir of neurons periodically transmits its output activity to a module of readout neurons which, by gradient descent, have their weights trained to associate the input data with a desired network response.

The spiking neuron variant for reservoir computing is referred to as a Liquid State Machine (LSM), which is capable of efficiently performing real-time computations on a time-varying input signal (see Fig. 2.7B) (Maass et al. 2002). An interesting observation by the authors of this study relates to the fading memory property of the system, or more specifically the ability of recurrently connected spiking neurons to remember the history of presented input data, as reflected in their output firing activity, that extends well beyond their short-term integration time constant $\tau_m$. This memory property is considered to be essential for the system’s success in making inferences about time series input data with regards to its long-term temporal evolution. However, despite their potential for increased computational power, the neural dynamics in an LSM is significantly more complex to analyse than that in a simpler feed-forward SNN structure, making the formulation of spike-based learning methods for them much less straightforward.

2.4 Neural Coding

Neurons are characterised by their ability to rapidly transmit electrical signals over large distances in the body, and it is by this mechanism that neurons are able to transform sensory inputs into appropriate motor actions. The relationship
between stimulus and individual or ensemble neuronal responses is referred to as the neural code, and one of the fundamental aims of neuroscience is to decipher this code.

At present, two distinct hypotheses exist to explain the underlying nature of neural processing: the first based on a rate code, the second a temporal code. With respect to a rate code, the firing rate of a neuron is considered to fully describe a stimulus, whereas a temporal code instead uses the timings of neuronal spikes. Temporal coding is somewhat broad in its definition, and can refer, for example, to a coding scheme that relies on the precise timings of individual output spikes, or to a code that relies on the order in which first-spikes are generated over ensembles of output neurons (commonly referred to as rank-order coding (S. Thorpe et al., 2001)). This section briefly reviews each of the aforementioned coding schemes in turn, as well as their supporting experimental evidence.

#### 2.4.1 Rate Coding

The idea that information pertaining to a stimulus is encoded in the firing rate of a neuron dates back to experiments performed by [Adrian (1926)] on the frog muscle, where the firing rate of stretch receptor neurons in the muscle depends on the force applied to them. Since then, rate coding has gained popularity as a mechanistic explanation for sensory processing in other areas of the nervous system, for example to describe the response of primary visual cortex neurons to a moving light stimulus ([Henry et al., 1974](#)).

In practice there exist several different definitions of a rate code, each depending on how the firing rate is calculated. The difference arises from the choice of averaging procedure used, such as by taking a temporal, repeated-trial or neuronal-ensemble average. A temporal average is defined as the neuronal spike count over some specified duration $T$ divided by $T$, whereas a repeated-trial average instead sums binned spike counts over several, identical experimental runs (trials) divided by the number of trials. A neuronal-ensemble average is defined similarly to a repeated trial average, but differs by instead averaging binned spike counts over large numbers of homogeneous neurons ([Gerstner & Kistler, 2002](#)). In most cases a simple temporal average is selected to define the firing rate, al-
though its usage can be restrictive: in order to reliably estimate the firing rate of a neuron a sufficiently long period of time must first elapse.

2.4.2 Temporal Coding

It is becoming increasingly clear that the relative timings of spikes transmitted by neurons, and not just their firing rates, are used to convey information regarding the features of input stimuli (van Rullen et al., 2005). Hence, the concept of a temporal code that is based on the timings of individual spikes becomes relevant. In many cases, a temporal code is identified by a high-frequency, rapidly fluctuating firing rate (Dayan & Abbott, 2001), or by the sensitivity of a postsynaptic neuron to the relative timings of presynaptic spikes: commonly referred to as coincidence detection (deCharms & Merzenich, 1996).

Spike timing as an encoding mechanism is advantageous over rate-based codes in the sense that it is capable of tracking rapidly changing input features, for example briefly presented images projected onto the retina (Gollisch & Meister, 2008), or tactile events signalled by the fingertip during object manipulations (Johansson & Birznieks, 2004). It is also apparent that spikes are generated with high temporal precision, typically on the order of a few milliseconds under variable conditions (Mainen & Sejnowski, 1995; Reich et al., 1997; Uzzell & Chichilnisky, 2004).

Precisely Timed Spikes. A possible application of a temporal coding scheme is the identification of input features using the precise timings of all output spikes, also referred to as a fully temporal code (Grüning & Bohte, 2014). This represents the most general usage of a temporal code, since in this case every individual spike timing is put to use, and has the potential to allow for a very large number of unique pattern encodings to be performed by just a single neuron operating over a limited time frame. Moreover, if there exist spike trains distributed over groups of neurons that are time-locked with respect to each other, then these patterns are referred to as polychronous groups (Izhikevich, 2006).

Rank-Order Coding. A further possible use of a temporal code relies on the order in which multiple output neurons emit their first spike in response to an
Chapter 2. Spiking Neurons

input stimulus \cite{Thorpe1998}. This coding scheme is ideally suited to time critical tasks, for example when neurons are subject to strong temporal constraints and only have time to emit a single output spike in response to brief input stimuli \cite{vanRullen2005}.

In terms of its implementation, rank-order coding represents a middle ground between a relatively simplistic rate code and a more complex fully temporal code: a rank-order code makes use of spike timings, allows for increased information storage in comparison with an equivalent rate code, and is simpler to decode than a code that uses multiple, precisely timed spikes \cite{Thorpe2001}. Despite this, rank-order coding has less potential than a fully temporal code in terms of the maximum number of pattern encodings it can perform, and additionally relies on large ensembles of output neurons for similar processing capability.

2.5 Chapter Summary

This chapter has outlined the fundamental principles of neuronal processing in the nervous system: starting with an overview of the biological neuron and its functions, spiking neurons as a descriptive mathematical model, and a final review of different neuronal coding mechanisms used to represent sensory information.

From a phenomenological perspective, the LIF model and SRM are sufficiently capable of recreating the essential functions of biological neurons, and are particularly well suited when the action potentials generated by a neuron are formalised as ‘all-or-none’ spike events. Importantly, their relative simplicity in comparison with alternative methods lends their utility to theoretical investigations of nervous system processing. For example, the SRM neuron, combined with escape noise, allows for a great deal of flexibility regarding the formulation of spike-based learning rules, as shall become apparent in the next chapter where existing learning rules for SNNs are reviewed.

From the overview in section 2.3 on the main structural classes of SNNs, it follows that there exist some disadvantages in implementing LSMs over feed-forward networks, despite their potential for strong computational power. In particular, an LSM can only provide a limited understanding of how neural computation
is actually carried out in the nervous system, which arises from its increased complexity in comparison with a simpler network structure: it is much more challenging to develop learning algorithms for recurrent network structures than it is for feed-forward structures. For this reason, the focus of this thesis shall be directed towards examining networks that are structured, or in other words, those that are organised as feed-forward single- and multilayer network structures, rather than those that have recurrent connections.

As reviewed in section 2.4, the advantages of temporal- over rate-based coding are clear; for example, by using a temporal code it is possible to track a rapidly changing stimulus based on the precise timings of individual spikes, whereas a rate code must first sample the stimulus over a relatively long duration before any representation can be formed. For the purposes of this thesis, the focus shall be on the use of a fully temporal code that uses multiple, precisely timed output spikes for pattern recognition, rather than, for example, a rank-order code; the reason for this is to better realise the maximum potential of derived learning methods for SNNs. Is it important to stress, however, that temporal coding is not a universal mechanism by which information is processed in the nervous system. Despite this, spikes themselves certainly underlie both rate- and temporal-based codes, and so it would seem most appropriate to formulate new learning methods based on individual spikes, for the simple reason that it is easier to transform a spike-based learning rule into a frequency-based rule than vice-versa.
Chapter 3

Learning in Spiking Neural Networks

This chapter reviews key advances in the field of theoretical neuroscience in relation to the learning of precisely timed spikes for neural information processing. The first section of this chapter provides an overview of synaptic plasticity processes identified in biological neural networks: with an emphasis on the process of Spike-Timing-Dependent Plasticity (STDP) which has been found through in-vivo and in-vitro experiments to essentially function as an unsupervised learning rule. In the next section a brief review of unsupervised learning is provided, including its relation to STDP as applied to training SNNs to learn spatio-temporal spike patterns. The following section then examines reinforcement-based learning as a biologically plausible scheme for training SNNs to form specific stimulus-response associations through reward-modulated synaptic weight changes. The final section then reviews prominent supervised learning rules for SNNs, many of which utilise the timings of multiple and precisely timed output spikes as a means to form representations of spatio-temporal input patterns.

3.1 Synaptic Plasticity

Learning in the brain is widely considered to take place through persistent modifications of synaptic strengths between neurons. A variety of synaptic processes
have been experimentally observed to drive such modifications, ranging from short
to longer term plasticity changes (Abbott & Nelson 2000; Morrison et al. 2008;
Caporale & Dan 2008; Roberts & Bell 2002). In this section two prominent
synaptic plasticity mechanisms in the nervous system are reviewed: STDP and
homoeostatic plasticity.

3.1.1 Spike-Timing-Dependent Plasticity (STDP)

The process of STDP has been identified in many areas of the nervous system,
and is believed to play a major part in neuronal organisation during development
(Caporale & Dan 2008). STDP describes a persistent change in the synaptic
strength between a pair of connected neurons based on their relative firing times,
which is typically effective for relative timing differences of less than a few tens of
milliseconds. In most cases the direction of synaptic strength change depends on
the order in which pre- and postsynaptic spikes occur, for example: an increase
in the synaptic strength (potentiation) for pre- before a postsynaptic spike, and
a decrease in the synaptic strength (depression) for pre- after a postsynaptic
spike. This process can be viewed as a spike-based formulation of Hebbian’s
postulate for learning in the nervous system: if a presynaptic neuron “repeatedly
or persistently takes part in firing” a postsynaptic neuron, then an increase in
the synaptic strength between will ensue (Hebb 1949). Experimentally, synaptic
plasticity in hippocampal and cortical neurons has been demonstrated to behave
in this way (Bi & Poo 1998; Sjöström et al. 2001).

STDP is mathematically formalised by first defining a presynaptic neuron’s spike
train as a sum of Dirac-delta functions:

\[ X_j(t) = \sum_f \delta(t - t^f_j) \quad (3.1) \]

that is a function of a sequence of firing times \( t^f_j \) for the \( j^{th} \) presynaptic neuron
(Gerstner & Kistler 2002). The \( i^{th} \) postsynaptic neuron’s spike train is similarly
defined by \( Y_i(t) \). Hence, the change in the synaptic strength (or weight) \( w_{ij} \)
Chapter 3. Learning in Spiking Neural Networks

according to a model of STDP \cite{Gerstner2002} is defined by

\[
\frac{dw_{ij}(t)}{dt} = F_+(w_{ij}) \mathcal{Y}_i(t) \int_0^\infty W_+(s) \mathcal{X}_j(t-s) \, ds \\
+ F_-(w_{ij}) \mathcal{X}_j(t) \int_0^\infty W_-(s) \mathcal{Y}_i(t-s) \, ds.
\]  (3.2)

The kernel \( W_{\pm}(s) = A_{\pm} \exp(-s/\tau_{\pm}) \) describes the ‘learning window’ of STDP: typically giving a positive weight change for a pre- before postsynaptic spike \((W_+)\), and a negative weight change for a pre- after postsynaptic spike \((W_-)\).

The parameters \( A_{\pm} \) and \( \tau_{\pm} \) control the amplitude and time scale of the learning window, respectively. Typical parameter selections are: \( A_+ = -A_- = 1, \tau_+ = 10\,\text{ms} \) and \( \tau_- = 20\,\text{ms} \) \cite{Gerstner2002}.

The function \( F_{\pm}(w_{ij}) \) signifies a dependence of STDP on the actual value of the synaptic weight, and its functional consequence can be to ensure that weights changed through STDP stay bounded. A common form is given by a power law:

\[
F_+(w_{ij}) = \eta (1 - w_{ij})^\mu \quad \text{and} \quad F_-(w_{ij}) = \eta \varphi w_{ij}^\mu,
\]  (3.3)

where \( \eta \) is the learning rate, \( \varphi \) an asymmetry parameter and \( \mu \) a non-negative exponent for determining the dependence of weight changes on the current value of \( w_{ij} \) \cite{Gutig2003}.

Two distinct STDP weight update rules emerge based on the choice of \( \mu \) in Eq. (3.3): so-called ‘additive STDP’ for \( \mu = 0 \), and ‘multiplicative STDP’ for \( \mu = 1 \) \cite{Gutig2003, Morrison2008}. Additive STDP updates have no dependence of the current weight value, therefore it is necessary to clip any weights that wander outside of a predefined boundary. By contrast, multiplicative STDP updates ensure weights remain bounded between \( 0 < w_{ij} < 1 \). The choice of \( \mu \) influences the equilibrium distribution of weights during unsupervised learning, such that additive STDP forms a bimodal distribution of weights, and multiplicative STDP a unimodal distribution of weights \cite{Gutig2003}.

Additive STDP has a tendency for run-away behaviour, by driving weights to extremal values, whereas multiplicative STDP is more homoeostatic in nature. It is noted that a choice of \( \mu = 0.4 \) gives a best fit to experimental data in Bi & Poo 29.
Figure 3.1: Examples of different Spike-Timing-Dependent Plasticity (STDP) rules identified in the nervous system, with synaptic change plotted as a function of the timing difference between a pre- and postsynaptic spike. Positive x-axis values indicate post- after presynaptic spikes. Panel A shows the classic, anti-symmetric Hebbian learning rule of STDP as found in hippocampal neurons [Bi & Poo 1998]. Panels B through F show several of the other forms STDP can take, including some that are non-localised in time. Figure adapted from Roberts & Bell (2002).

(1998), indicating a multiplicative dependence for STDP in biology (Morrison et al., 2007).

Although STDP is commonly assumed to behave in a Hebbian-like manner, there is no universal rule for explaining STDP throughout the entire nervous system. Experimental studies demonstrate STDP can take a variety of different forms, some even displaying anti-Hebbian learning (Roberts & Bell, 2002). Examples of STDP rules are shown in Fig. 3.1. The form of STDP described by Eq. (3.2) most closely approximates panel A.

The mechanisms of STDP discussed so far capture the essence of unsupervised learning in the brain, although it is by no means restricted to just the timings of coincident spikes; more recent studies have indicated at a further modulation of STDP by the postsynaptic, subthreshold membrane potential (Morrison et al., 2008). Such a level of detail is necessary when modelling experimental results obtained from multi-spike interactions in STDP (Clopath et al., 2010). Taken together, there can exist no ‘one-size fits all’ model for describing STDP, and the
selection of an appropriate model is highly conditional upon the problem under consideration.

### 3.1.2 Homeostatic Plasticity

Beyond STDP there exist further processes which contribute to synaptic plasticity in the nervous system. Homeostatic plasticity is an important example, and in contrast with STDP acts in a regulatory role to maintain an optimal level of firing activity in neural circuits. For instance, synaptic scaling is a homeostatic plasticity mechanism which controls the firing activity of a neuron by multiplicatively scaling (upwards or downwards) the synaptic strengths of all its afferent connections (Abbott & Nelson, 2000; Morrison et al., 2008). Interestingly, simulations of synaptic scaling in SNNs demonstrate an increase in competition between afferent synaptic connections; such competition is useful for influencing developmental processes, and to normalise output neuronal firing rates to minimise the impact of variable input activity on learning (van Rossum et al., 2000).

A simplified mathematical model of synaptic scaling, with respect to a postsynaptic neuron $i$, is given by (van Rossum et al., 2000):

$$
\Delta w_{ij} = \begin{cases} 
\psi |w_{ij}| (\nu_{\text{max}} - \nu_i) & \text{if } \nu_i > \nu_{\text{max}} \\
\psi |w_{ij}| (\nu_{\text{min}} - \nu_i) & \text{if } \nu_i < \nu_{\text{min}} 
\end{cases}
$$

where $\psi$ is the scaling strength, $\nu_i$ the actual output firing rate of the neuron, and $\nu_{\text{max}}$ and $\nu_{\text{min}}$ the maximum and minimum reference postsynaptic firing rates, respectively. The above drives the firing rate of the postsynaptic neuron to remain within the range $\nu_{\text{min}} \leq \nu_i \leq \nu_{\text{max}}$ by iteratively scaling weights, thereby making an SNN less sensitive to its initial state and preventing extremes in the firing activity of the neuron (Grüning & Sporea, 2012; Sporea & Grüning, 2013).

Another form of homeostatic plasticity is synaptic redistribution, a short term plasticity process which regulates postsynaptic neuronal firing activity based on the probabilistic release of presynaptic transmitters into the synaptic cleft (Abbott & Nelson, 2000). A high presynaptic firing rate leads to a more rapid depletion in the amount of presynaptic transmitters available for release, which in turn decreases the postsynaptic firing rate. Probabilistic approaches to modelling
this process and its relation to neuronal dynamics are reviewed in Morrison et al. (2008). Beyond its regulation of neuronal firing activity, synaptic redistribution has also been investigated as a stochastic process for driving reinforcement-based learning in networks of spiking neurons (Seung, 2003).

There are a multitude of further synaptic plasticity processes in the brain that contribute to learning, although discussing each in turn is beyond the scope of this thesis. The key mechanisms underlying synaptic plasticity in the nervous system have been outlined, and the aim shall now be to relate these plasticity processes to the learning of spatio-temporal spike patterns in SNNs.

### 3.2 Unsupervised Learning

The process of unsupervised learning in neural networks refers to the adaptation of synapses to the statistics of pre- and postsynaptic neuronal activity. In this learning paradigm no task is explicitly specified that must be solved by the network: the network essentially learns by itself to form internal representations of particular input patterns, based on their overall statistical structure (Morrison et al., 2008). By this process, a network can be trained to cluster or classify input data through self-organisation (Bohte, Poutré, & Kok, 2002), for instance by adjusting synaptic weights via the correlation-based STDP rule as described previously. In this section, a learning method which uses STDP to train SNNs to form representations of spatio-temporal spike patterns is examined.

#### 3.2.1 STDP-Based Spike Pattern Learning

An important example of unsupervised learning in SNNs has been proposed by Masquelier et al. (2008), where STDP (see Eq. (3.2)) was applied to training a single postsynaptic neuron to detect a single, repeatedly presented, input spatio-temporal spike pattern embedded in random background spiking activity. Specifically, the network was of a feed-forward structure, containing a large number of presynaptic neurons that provided input to the postsynaptic ‘detector’ neuron. Half of the presynaptic neurons continuously transmitted independent Poisson spike trains to the postsynaptic neuron in the form of background ‘distraction’
noise, while the other half alternated between transmitting similarly generated background noise and a fixed spatio-temporal spike pattern of 50 ms duration. From running these simulations, the authors found that the postsynaptic neuron learned to respond to the onset of the repeated pattern with a precisely timed output spike, whilst suppressing its activity when subject to just random background noise.

As an extension, the same authors applied this idea to a more general ‘one-winner-take-all’ competitive learning scheme, where the network contained additional postsynaptic neurons, with lateral inhibitory connections, that competed with each other for being the first to detect the onset of the repeated pattern (Masquelier et al., 2009). The key finding from this extended study was that multiple postsynaptic neurons learned to stack their single-spike responses to the repeated pattern, such that each neuron essentially identified a different segment of the pattern (see Fig. 3.2 for an illustration of this process).

More recently, the subsequent work of Humble et al. (2012) has further extended the studies of Masquelier et al. (2008, 2009) by instead considering chains of recurrently connected postsynaptic neurons, which collectively are capable of learning to recognise the temporal sequence of spatio-temporal spike patterns, based upon the order in which chained neurons fire. As in the previous studies, the authors considered a network consisting of multiple postsynaptic neurons receiving their input from a large number of presynaptic neurons in a feed-forward manner, and, similarly as in Masquelier et al. (2009), fixed lateral inhibitory connections between the neurons were used to drive competitive learning. However, as the authors’ unique contribution, lateral excitatory connections (recurrent connections) between the postsynaptic neurons were also used that could be trained in the network during learning. Hence, by using STDP to train both feed-forward and recurrent synapses in the network during pattern learning, a sequence (or chain) of neurons emerged that could encode for the repeated pattern in their responsive firing times, where each neuron in the chain responded to a different segment of the pattern. Crucially, the authors found that by also training recurrent synapses in the network, then a learned chain of firing neurons could also encode for the temporal order of the pattern. In other words, if the segments which constituted a pattern were to be placed in a different order then the same neurons would instead respond with a different sequence of firing times.
Figure 3.2: Example of three postsynaptic neurons learning a spatio-temporal spike pattern that is embedded in random background noise. Both panels show their membrane potentials, in arbitrary units (a.u.), plotted as a function of time. Vertical lines indicate their firing times, and the grey shaded regions correspond to the repeated presentation of the pattern. (Top) The beginning of a simulation run, where the neurons initially fire at random. (Bottom) The end of the simulation, by which point the neurons only fire in response to the pattern and have learned to stack their spike responses. Each neuron recognises a different segment of the pattern. Figure adapted from Masquelier et al. (2009).

3.2.2 Discussion

Taken together, the studies of Masquelier et al. (2008, 2009); Humble et al. (2012) highlight a learning method which is not only biologically plausible, but also capable of identifying spatio-temporal spike patterns on a very fast time-scale using a temporal code that is akin to rank-order coding. Furthermore, and more closely relevant to the direction of this thesis, it follows from these studies that if a postsynaptic neuron learns to respond to a pattern with just a single spike, then effectively it just recognises a small segment of that pattern preceding its spike with a time-window on the order of the neuron’s membrane time constant, $\tau_{m}$. 
This is an important observation, since it places a constraint on the maximum amount of information a single-spike based neural encoder is capable of conveying during pattern recognition. Aside from simply increasing the number of encoding neurons, a potential solution to this caveat might instead utilise the timings of multiple output spikes for identifying a pattern, such that each spike is selective to a different segment of the pattern; in this way, information pertaining to the entirety of the pattern could be efficiently communicated using just a single neural encoder.

3.3 Reinforcement Learning

The theory of reinforcement learning, as originally defined in the field of machine learning by Barto & Sutton (1998), has emerged as a biologically plausible candidate for modelling decision-making in the brain (Dayan & Abbott, 2001). This learning hypothesis is motivated by the strong indication that the firing activity of dopaminergic neurons in the nervous system encode a form of reward prediction error, comparable with the Temporal Difference (TD) error signal defined in reinforcement learning (Schultz, 2000). Dopaminergic neurons are a class of neuron responsible for producing the neuromodulator dopamine, and their axons project to brain structures involved in motivation and goal-directed behaviour (Schultz et al., 1997).

This section reviews reinforcement learning as a biologically inspired method for training SNNs to form stimuli-response associations. First, a brief background of reinforcement-based learning is provided in relation to reward-modulated synaptic plasticity. Next, the general theory behind reinforcement learning based on eligibility traces is outlined, followed by a review of two plasticity rules for reward-modulated learning in SNNs: the Reward-modulated Spike-Timing-Dependent Plasticity (R-STDP) rule, and the Reward-maximisation (R-max) rule.

3.3.1 Background

Synaptic plasticity modulated by dopamine is thought to drive reinforcement learning in a region of the brain called the basal ganglia (Chakravarthy et al.)
Chapter 3. Learning in Spiking Neural Networks

2010 (Cohen & Frank 2009), where synaptic connections between the cortex and striatum (corticostriatal synapses) are modified based on the extracellular concentration of dopamine present (Reynolds & Wickens, 2002). Hence, if the activity of dopaminergic neurons projecting to corticostriatal synapses encode a TD error signal, then it is presumed that the concentration of dopamine signals the novelty of stimuli. If the occurrence of reward in response to a stimulus is surprising or unpredicted then a phasic release of dopamine results, and learning, through corticostriatal synaptic strength modifications, takes place (Schultz, 2000; Reynolds & Wickens, 2002).

With respect to behavioural learning, corrective feedback pertaining to a presented stimulus or responding action, and delivered in the form of a global reward signal, is often delayed (Dayan & Abbott, 2001). In relation to neuronal firing activity in the brain, identifying and reinforcing those spike patterns which contribute to the delivery of reward poses a significant challenge, and is known as the ‘credit assignment problem’ in reinforcement learning (Barto & Sutton, 1998). To address this issue, the general concept of a synaptic eligibility trace has been proposed to form the basis of reinforcement learning rules formulated for SNNs (Izhikevich, 2007; Farries & Fairhall, 2007; Legenstein et al., 2008), which shall now be the main focus of this section.

3.3.2 Eligibility Traces

Eligibility traces are the basic building blocks of reinforcement learning, and are designed to handle credit assignment in the case of delayed reward signals. An eligibility trace provides a temporary record of the occurrence of events, such as the generation of spikes by a neuron, such that memory parameters associated with these events are then marked as eligible for undergoing subsequent learning changes. Upon the availability of a delayed reinforcement signal, i.e. a ‘success signal’, only events that have been marked for change by an eligibility trace are assigned credit for the returned signal (Barto & Sutton, 1998). In this way, the eligibility trace helps to provide a causal link between events on the neuronal and training level.

In the context of SNN training, the eligibility trace forms the basis of a generic class of reward-modulated synaptic learning rules, defined as follows (Frémaux et
Chapter 3. Learning in Spiking Neural Networks

\[ \tau_e \frac{dE_{ij}(t)}{dt} = -E_{ij}(t) + \eta e_{ij}(t) \]  

(3.5)

\[ \frac{dw_{ij}(t)}{dt} = S[R(t)] E_{ij}(t) , \]  

(3.6)

where \( E_{ij} \) is a low-pass filtered trace of the synaptic eligibility \( e_{ij} \) between a presynaptic neuron \( j \) and postsynaptic neuron \( i \), with time constant \( \tau_e \). By itself the eligibility \( e_{ij} \) acts as an unsupervised Hebbian learning rule, capturing correlations between pre- and postsynaptic spikes, hence it corresponds to the candidate change in a weight \( w_{ij} \) which is made persistent only in the presence of a time-dependent success signal \( S[R(t)] \). The success signal is globally available at every synapse, and is a monotonic function of released reward \( R \). The specific functional choice of \( S \) is defined based on the learning task under consideration. The reward \( R \) is released in response to output activity by the network, and signals the general ‘correctness’ of network activity with respect to its input stimulus. The parameter \( \eta \) is a positive-valued learning rate.

For clarification, two distinct learning rules specified by Eqs. (3.5) and (3.6) are now considered: an empirically formulated R-STDP rule, and a theoretically derived R-max rule based on a reward maximisation principle.

### 3.3.3 Reward-modulated STDP (R-STDP) Rule

The R-STDP rule (Farries & Fairhall, 2007; Izhikevich, 2007; Legenstein et al., 2008) hypothesises that candidate weight changes triggered by unsupervised STDP are made persistent by an external reward signal, and draws its inspiration from the observed neuromodulation of synaptic plasticity in experimental studies (Seol et al., 2007; Reynolds & Wickens, 2002; Pawlak & Kerr, 2008).

The synaptic eligibility for R-STDP is defined by an appropriate choice of STDP function, for example:

\[
e^{R-STDP}_{ij}(t) = F_+(w_{ij}) \mathcal{Y}_i(t) \int_0^\infty W_+(s) \mathcal{X}_j(t-s) \, ds \]

\[ + F_-(w_{ij}) \mathcal{X}_j(t) \int_0^\infty W_-(s) \mathcal{Y}_i(t-s) \, ds , \]  

(3.7)
Figure 3.3: Scheme of reward-modulated STDP, as defined by Eqs. (3.5), (3.6) and (3.7). Coincident pre- and postsynaptic spikes influence candidate weight changes according to $e^{R-STDP}(t)$, which are then low-pass filtered as the eligibility trace $E(t)$. For example, a coincident pre- then postsynaptic spike is indicated by a rectangle in this figure, which triggers a large instantaneous increase in the eligibility trace. The network then deems this input-response association to be desirable by responding with a delayed, positive reward signal, leading in turn to an increase in the global success signal $S(t)$. The synaptic weight between the two neurons $w(t)$ then increases as the product of the signal $S(t)$ and synaptic trace $E(t)$, thereby strengthening the association between the paired neurons. Figure adapted from Izhikevich (2007).

as originally presented in Eq. (3.2). An example of a plasticity change triggered by R-STDP is shown in Fig. 3.3. An advantage of the unsupervised learning component defined by Eq. (3.7) is that it doesn’t rely on a specific choice of neuron model in simulations; for example, it is compatible with the nonlinear Izhikevich neuron (Izhikevich, 2007) as well as the relatively simple LIF neuron (Legenstein et al., 2008). This allows for more diverse behaviours of simulated neurons for increased biological realism.

In order for learning to take place according to R-STDP, it is necessary that a persistent level of background noise is present to drive variable postsynaptic spiking. Variable output activity allows for the stochastic exploration of solution
space, allowing a neuron to discover those spike patterns which give rise to the most reward \cite{Dayan2001,Legenstein2008}. A straightforward approach to incorporating noise in a postsynaptic neuron is to introduce variable presynaptic activity, as implemented in \cite{Farries2007}. Alternatively, diffusive white noise might be injected into the neuron, generated by an Ornstein-Uhlenbeck stochastic process \cite{Legenstein2008}. The escape noise model, as discussed in section \ref{sec:escape_noise}, represents another alternative, and has been used to form the basis of the R-max rule as discussed in the next subsection.

### 3.3.4 Reward-maximisation (R-max) Rule

The R-max rule has a longer history than R-STDP, having its theoretical framework originally laid down by \cite{Xie2004}, and later extended to take into account neuronal refractoriness by \cite{Pfister2006,Florian2007}. This rule was derived from a reward maximisation principle, and its synaptic eligibility component is given by

\begin{equation}
\epsilon_{ij}^{R_{-}\text{max}}(t) = \frac{1}{\Delta u} \left[ \mathcal{Y}_i(t) - \rho_i(t) \right] \int_{0}^{\infty} \epsilon(s) \mathcal{X}_j(t - s) \, ds , \tag{3.8}
\end{equation}

where $\Delta u$ is defined in Eq. \eqref{eq:delta_u}, and $\rho_i(t)$ is the instantaneous firing density of the postsynaptic neuron. Chapter 4 shall go through the derivation of this rule in detail. For the purposes of this section however, it is sufficient to realise that this rule depends explicitly on the SRM combined with exponential escape noise (see Eqs. \eqref{eq:SRM} and \eqref{eq:escape_noise}). The integral term in Eq. \eqref{eq:R-max} corresponds to PSPs evoked by presynaptic spikes, and the left-hand side term corresponds to the error between a postsynaptic spike train and its underlying, instantaneous firing density.

It is important to note that $\epsilon_{ij}^{R_{-}\text{max}}$ by itself is useless for unsupervised learning, because its ensemble average vanishes when conditioned on the set of input spike
trains $\{X_j\}$ \cite{Fremaux2010}:

$$
\langle e_{ij}^{R_{\text{max}}} \rangle_{Y_i|\{X_j\}} = \left\langle \frac{1}{\Delta u} \left[ Y_i(t) - \rho_i(t) \right] \int_0^\infty \epsilon(s) X_j(t-s) \, ds \right\rangle_{Y_i|\{X_j\}} \\
= \frac{1}{\Delta u} \left\langle Y_i(t) - \rho_i(t) \right\rangle_{Y_i|\{X_j\}} \int_0^\infty \epsilon(s) X_j(t-s) \, ds \\
= 0 , \quad (3.9)
$$

having used the relation for a fixed stimulus: $\rho_i(t) = \left\langle Y_i(t) \right\rangle_{Y_i|\{X_j\}}$. Hence, learning only takes place through correlations between the postsynaptic spike train $Y_i(t)$ and the success signal $S[R(t)]$ (see Eq. (3.6)).

Similarly to R-STDP, background noise is required in order for R-max to search for desirable output spike patterns during learning. This condition is satisfied naturally by its inclusion of escape noise for driving variable postsynaptic spiking.

### 3.3.5 Discussion: R-STDP and R-max

Mechanistically, both the R-STDP and R-max rules rely on three factors for learning to proceed: 1. presynaptic activity to represent the stimulus, 2. postsynaptic activity as the network response, and 3. the feedback of a success signal to effect synaptic weight changes \cite{Vasilaki2009}. Both rules share certain similarities in their functional form \cite{Fremaux2010}; for example, assuming $F_\pm(w_{ij})$ and $\Delta u$ are equal to one, the pre-before-post part of R-STDP, $Y_i(t) \int_0^\infty W_+(s) X_j(t-s) \, ds$, closely resembles that of R-max, $Y_i(t) \int_0^\infty \epsilon(s) X_j(t-s) \, ds$. Their difference only arises from the shape of the coincidence kernels $W_+(s)$ and $\epsilon(s)$ for R-STDP and R-max, respectively. However, while the pre-after-post part of R-STDP, $X_j(t) \int_0^\infty W_-(s) Y_i(t-s) \, ds$, depends on postsynaptic spiking events, R-max instead depends on the instantaneous firing density $\rho_i$.

In terms of their learning dynamics, a significant difference between R-STDP and R-max relates to the lack of an unsupervised learning bias with R-max, as opposed to R-STDP (see Eq. (3.9)). In general, an unsupervised component hinders reward-modulated learning, as evidenced by the decreased performance of R-STDP in comparison with R-max when applied to learning multiple input-output spike pattern mappings in an SNN \cite{Fremaux2010}. 

40
Both R-STDP and R-max are considered biologically plausible rules for modelling learning in the brain, since they are minimally supervised and their dependence on a success signal is consistent with neuromodulation of STDP by extracellular dopamine concentration (Pawlak & Kerr, 2008). Significant progress has been made in applying these learning rules to solving typical reinforcement learning problems (Vasilaki et al., 2009; Frémaux et al., 2013; Friedrich et al., 2011), however, aside from work by Frémaux et al. (2010), there is still a lack of work that has aimed at learning multiple and precisely-timed target output spikes. The motivation for learning a fully temporal code is biologically relevant (see section 2.4.2), and has inspired our original contributions (Gardner & Grüning, 2013; Gardner et al., 2014) as presented in chapter 4.

3.4 Supervised Learning

The importance of precise neuronal spike-timing in neural and cognitive information processing has been indicated at in a variety of studies (Bohte, 2004). For example, in the olfactory system the precision of spike-timing has been associated with accurate odour-classifications (Laurent et al., 1996; Vickers et al., 2001), and populations of auditory neurons are known to signal input features by the relative timing of spikes (deCharms & Merzenich, 1996; Knudsen, 2002). However, an understanding of how the brain learns to reliably associate specific input patterns with desired spike responses through synaptic strength modifications remains a significant challenge.

To address this, a variety of supervised learning rules for SNNs have been proposed that allow transformations of spatio-temporal input spike patterns into desired sequences of temporally precise output spikes (Kasinski & Pomulak, 2006; Güttig, 2014). In contrast with reward-modulated learning, supervised learning relies explicitly on an instructive signal to guide synaptic strength modifications during learning; although this may come at the cost of decreased biological realism, it allows for vastly increased precision of temporal encoding. The efficient learning of multiple and precisely-timed output spikes is currently a major limitation for most reward-modulated approaches.

This section provides an overview of prominent supervised learning methods for
SNNs appropriate to this thesis, including: SpikeProp, ReSuMe, Chronotron, SPAN and optimal STDP for precise spiking. This is then followed by their discussion in relation to key criteria for effective learning, and finally the biological evidence of supervised-based learning in the nervous system.

3.4.1 SpikeProp (Spike-based Backpropagation)

Among the first supervised learning rules to be formulated for an SNN is SpikeProp (Bohte et al., 2000; Bohte, Kok, & Poutré, 2002), a technique which draws parallels with traditional error-backpropagation as applied to rate-based neural networks.

SpikeProp is applicable to feed-forward multilayer network structures, and its goal is to learn a set of arbitrary target spike times \( \{\hat{t}_i\} \) at postsynaptic neurons \( i \in I \) in response to an input spike pattern. This method is restricted to learning a single target spike at each neuron however, which is a consequence of neglecting a neuron’s membrane potential after it has fired. Such an approximation was necessary by the authors in order to deal with the discontinuity of a neuron’s membrane potential about its firing time.

The SpikeProp algorithm was derived starting with an SRM neuron, where the membrane potential of neuron \( i \) is described by

\[
u_i(t) = \sum_{j \in \Gamma_i} \sum_l w^l_{ij} \epsilon(t - t_j - d^l),
\]

which, in comparison with the previously presented SRM in Eq. (2.5), lacks an explicitly defined neuronal refractory term. The first summation runs over all the direct presynaptic neurons of neuron \( i \), represented by the set \( \Gamma_i \). The second summation runs over multiple subconnections, indexed by \( l \), between neurons \( j \) and \( i \), where each subconnection has its own synaptic weight \( w^l_{ij} \) and conduction delay \( d^l \). The parameter \( t_j \) is the timing of a spike contributed from presynaptic neuron \( j \). The PSP kernel is assumed as \( \epsilon(s) = s/\tau \exp(1 - s/\tau) \Theta(s) \), where \( \tau \) is some time constant and \( \Theta(s) \) the Heaviside step function (\( \Theta(s) = 1 \) if \( s > 0 \), \( \Theta(s) = 0 \) otherwise).

A gradient descent procedure is used to optimise weights in the network, where
the error function to minimise is defined by

$$E = \frac{1}{2} \sum_i (t_i - \bar{t}_i)^2,$$

(3.11)

where $t_i$ and $\bar{t}_i$ are the actual and target output spike timings of a postsynaptic neuron $i$, respectively. Weights are modified in the direction of negative gradient:

$$\Delta w^l_{ij} = -\eta \frac{\partial E}{\partial w^l_{ij}},$$

(3.12)

where $\eta > 0$ is the learning rate. As shown in Bohte, Kok, & Pouthé (2002), for a fully connected feed-forward network containing a single hidden layer, the gradient of the error can be expressed in terms of a backpropagated error signal $\delta_i$:

$$\frac{\partial E}{\partial w^l_{ij}} = \lambda^l_j(t_i) \delta_i,$$

(3.13)

where the first term on the right-hand side denotes the PSP due to a spike from a neuron $j$ with timing $t_j$ through subconnection $l$: $\lambda^l_j(t) = \epsilon(t - t_j - d^l)$. When treating postsynaptic neurons as the output layer neurons, the error signal equals

$$\delta_i = \frac{(\bar{t}_i - t_i)}{\sum_{j \in \Gamma_i} \sum_l w^l_{ij} \frac{\partial \lambda^l_j(t_i)}{\partial t}},$$

(3.14)

When instead treating postsynaptic neurons as the hidden layer neurons, the error signal becomes

$$\delta_i = \frac{\sum_{j \in \Gamma^i} \delta_j \sum_l w^l_{ij} \frac{\partial \lambda^l_j(t_i)}{\partial t}}{\sum_{j \in \Gamma_i} \sum_l w^l_{ij} \frac{\partial \lambda^l_j(t_i)}{\partial t}},$$

(3.15)

where the set $\Gamma^i$ represents all neurons directly succeeding neuron $i$, and the set $\Gamma_i$ again represents all the direct presynaptic neurons (i.e. input layer neurons) of neuron $i$.

Finally, using Eqs. (3.12) and (3.13) and selecting either Eq. (3.14) or (3.15) to define $\delta_i$, weights in a layer are modified according to

$$\Delta w^l_{ij} = -\eta \lambda^l_j(t_i) \delta_i.$$

(3.16)
ties of benchmark classification tasks, such as solving the classical exclusive-or (XOR) computation and classifying Iris dataset species; it is noted that these tasks require the presence of a hidden layer for their solution, hence motivating the application of a multilayer learning rule like SpikeProp. The authors also encoded input and output values by the latencies of spikes, thereby transforming analog values into temporal representations. On these tasks, SNNs trained by SpikeProp were comparable with traditional sigmoidal neural networks, returning a similar accuracy level and convergence time. However, SNNs seemed to be more reliable to train, whereas sigmoidal neural networks occasionally failed to converge (Bohte, Kok, & Pouthé, 2002).

There are certain limitations to the SpikeProp rule: the most critical being its failure to learn in the absence of postsynaptic spiking in response to input patterns, and its restriction to learning just a single output spike. Furthermore, SpikeProp is dependent on SRM neurons and cannot, for example, be applied to Izhikevich neurons which display more diverse firing behaviours, therefore making its biological plausibility questionable.

### 3.4.2 ReSuMe (Remote Supervised Learning Method)

A supervised learning method proposed by Ponulak & Kasinski (2010), Remote Supervised learning Method (ReSuMe), has been put forward as a more biologically plausible method for learning precisely timed output spikes. This approach takes its inspiration from Hebbian-like learning: adapting the process of unsupervised STDP as presented in Gerstner & Kistler (2002) to instead training SNNs by supervision.

Unlike the previous SpikeProp rule, ReSuMe is capable of learning target output spike trains containing *multiple* spikes. In its original form ReSuMe was derived to train single-layer SNNs (Ponulak & Kasinski, 2010), although it has since been generalised to multilayer SNNs as shown by Sporea & Grüning (2013); this section reviews the single-layer formulation as shown by Ponulak & Kasinski (2010).

ReSuMe aims to minimise the error between a target output signal $y_i^{\text{ref}}$ and an actual output signal $y_i$ at a postsynaptic neuron $i$. To this end, the Widrow-Hoff rule, as is commonly used to train traditional artificial neural networks, is taken
as the rule’s starting point:

$$
\Delta w_{ij} = \eta (y_i^{\text{ref}} - y_i) x_j ,
$$

(3.17)

where $\eta$ is a learning rate and $x_j$ an input signal contributed by the $j$th presynaptic neuron. Eq. (3.17) can be interpreted as a combination of two Hebbian-like processes: the first one relating to positive correlations between the desired output signal $y_i^{\text{ref}}$ and input $x_j$, and the second an anti-Hebbian process since correlations between the actual output signal $y_i$ and input $x_j$ are made negative.

In order to relate Eq. (3.17) to a spike-based synaptic plasticity rule, the authors substitute the signals $y_i^{\text{ref}}$, $y_i$ and $x_j$ with their respective spike trains $Y_i^{\text{ref}}$, $Y_i$ and $X_j$. A spike train for each signal is defined similarly to Eq. (3.1), that is a function of the respective neuronal firing times. Hence, Eq. (3.17) can be rewritten for spiking neurons operating in continuous time as

$$
\frac{d w_{ij}(t)}{dt} = C_{ij}^{\text{ref}}(t) + C_{ij}(t) ,
$$

(3.18)

where, for clarity, the learning rate has been set to $\eta = 1$. The first term $C_{ij}^{\text{ref}}$ corresponds to positive temporal correlations between $Y_i^{\text{ref}}$ and $X_j$, and the second term $C_{ij}$ corresponds to negative temporal correlations between $Y_i$ and $X_j$.

Finally, the authors map Eq. (3.18) to the STDP process proposed by Gerstner & Kistler (2002) (see Eq. (3.2)) to estimate the correlation terms $C_{ij}^{\text{ref}}$ and $C_{ij}$, which, after some simplifications, provides the ReSuMe rule for synaptic weight modifications:

$$
\frac{d w_{ij}(t)}{dt} = \left[ Y_i^{\text{ref}}(t) - Y_i(t) \right] \left[ a_0 + \int_{0}^{\infty} W(s)X_j(t-s)ds \right] .
$$

(3.19)

The learning window $W$ is defined similarly to $W_+$ in Eq. (3.2), and $a_0$ is an additional non-correlative factor. The $a_0$ term allows weight changes to take place contingent on just postsynaptic spiking, and has the effect of driving the mean actual firing rate of $Y_i$ to approach the mean target firing rate of $Y_i^{\text{ref}}$. Introducing this non-Hebbian term has been found to significantly speed up the convergence of the rule during learning. The full derivation of Eq. (3.19) can be found in Ponulak & Kasinski (2010).
Chapter 3. Learning in Spiking Neural Networks

Through simulations ReSuMe was shown to be highly efficacious in learning, such that trained SNNs were able to precisely match desired output spike patterns with rapid convergence. The rule has also displayed resilience against background noise, and has also successfully been applied to training the weights of a readout neuron within a recurrently-connected SNN. Furthermore, ReSuMe is not limited to any specific neuron model, and is applicable to LIF, Hodgkin-Huxley and Izhikevich neurons, among others (Ponulak & Kasinski, 2010).

ReSuMe has some limitations, the most obvious being the heuristic approach taken during its formulation, and the assumptions made regarding the precise nature of the STDP process itself. To address these issues, an alternative and more rigorous formulation of ReSuMe based on a gradient descent procedure has been proposed by Sporea & Grüning (2013). Despite this, the choice of a learning window for ReSuMe remains somewhat arbitrary, hence its optimality cannot be guaranteed analytically.

3.4.3 Chronotron (Gradient Descent Learning)

The Chronotron (CHRON) proposed by Florian (2012) has emerged as a supervised learning method for training a spiking neuron to fire temporally precise spike patterns. This method utilises a gradient descent procedure to minimise the temporal error of trained SNNs, thereby lending it increased analytical rigour in comparison with most other supervised methods.

In Florian (2012), CHRON is defined as a spiking neuron which, in response to a given input, learns to precisely match its actual output to a prescribed target spike train through changing its neural parameters. Additionally, CHRON must be capable of performing multiple such input-output spike pattern associations using a single set of neural parameters. To this end, the author proposed two supervised learning rules, termed E-learning and I-learning, both of which aim to optimise the synaptic weights of CHRON for precise temporal encoding of input patterns. The motivation behind E-learning is to provide a rule with an associated high network memory capacity, in terms of the maximum number of input patterns it can learn to reliably identify, and I-learning a rule with increased biological realism. This section focuses on the more commonly implemented E-
learning rule, which shall later be used as a benchmark for our original results in chapter 5.

The E-learning rule works to minimise the temporal error between an arbitrary target and actual postsynaptic spike train, \( Y_i^{\text{ref}} \) and \( Y_i \), respectively, in response to a set of presynaptic spike trains \( \{ X_j \} \), by changing synaptic weights \( \{ w_{ij} \} \) in an SNN. Hence, E-learning minimises a cost function of output spike trains: specifically the Victor & Purpura Distance (VPD) (Victor & Purpura, 1996). The VPD is a metric used to measure the temporal difference between two spike trains, and is determined from computing the minimum cost required to transform one spike train into the other by inserting, removing or shifting the timings of individual spikes.

To mathematically formalise the E-learning rule, we start by considering a list of actual output spikes \( \mathcal{F}_i = \{ t_{i1}, t_{i2}, ... \} \) and a list of target output spikes \( \tilde{\mathcal{F}}_i = \{ \tilde{t}_{i1}, \tilde{t}_{i2}, ... \} \), for a postsynaptic neuron \( i \). According to the VPD, spikes within \( \mathcal{F}_i \) that should be removed are denoted by the subset \( \mathcal{F}_i^* \), and spikes within \( \tilde{\mathcal{F}}_i \) that should be inserted into \( \mathcal{F}_i \) are denoted by the subset \( \tilde{\mathcal{F}}_i^* \). It is noted that the subsets \( \mathcal{F}_i - \mathcal{F}_i^* \) and \( \tilde{\mathcal{F}}_i - \tilde{\mathcal{F}}_i^* \) are in a one-one correspondence with each other; i.e. these are the spikes that should be shifted towards each other and not removed or inserted. The VPD has been adapted to depend on the dynamics of an SRM neuron (Florian, 2012) which is used to define an error function:

\[
E_{i}^{\text{VP}} = \sum_{t'_{i} \in \mathcal{F}_i} u(t'_{i}) + \sum_{\tilde{t}'_{i} \in \tilde{\mathcal{F}}_i} (\vartheta - u(\tilde{t}'_{i})) + \frac{\gamma_d}{2\tau_q} \sum_{(t'_{i}, \tilde{t}'_{i})} (t'_{i} - \tilde{t}'_{i})^2 + \gamma_d \sum_{t'_{i} \in \mathcal{F}_i - \mathcal{F}_i^*, \tilde{t}'_{i} \in \tilde{\mathcal{F}}_i - \tilde{\mathcal{F}}_i^*} (t'_{i} - \tilde{t}'_{i})^2 ,
\tag{3.20}
\]

where the first sum is the cost in removing actual spikes, the second sum the cost from inserting target spikes, and the final sum the cost from shifting actual spikes towards their target timings. In the final sum, the parameter \( \gamma_d > 0 \) is a constant proportionality term and \( \tau_q \) acts as a coincidence time constant.

The error function is minimised by piecewise gradient descent with respect to SNN synaptic weights; therefore, the change in the weight between pre- and postsynaptic neurons \( j \) and \( i \), respectively, is given by

\[
\Delta w_{ij} \sim -\frac{\partial E_{i}^{\text{VP}}}{\partial w_{ij}} .
\tag{3.21}
\]
Hence, using Eqs. (3.20) and (3.21), and the generic SRM neuron defined by Eq. (2.5), the resulting E-learning rule can be derived:

$$\Delta w_{ij} = \eta \left[ \sum_{\tilde{t}_f^i \in \tilde{F}_i^*} \lambda_{ij} (t_f^i, \tilde{t}_f^i) - \sum_{t_f^i \in F_i^*} \lambda_{ij} (t_f^i, \hat{t}_f^i) \right] + \gamma_r \tau_2 \sum_{(t_f^i, \tilde{t}_f^i) \in (F_i^* - \tilde{F}_i^*)} \frac{t_f^i - \tilde{t}_g^i}{\bar{\tau}^2} \lambda_{ij} (t_f^i, \hat{t}_f^i) \left( t_f^i - \tilde{t}_g^i \right),$$

where $\eta$ is the learning rate, and $\gamma_r > 0$ a newly defined constant term related to $\gamma_d$. The presynaptic factors are defined by $\lambda_{ij} (t, \hat{t}_i) = \sum_f \epsilon (t - \hat{t}_i, t - t_f^i)$, that is the effective PSP evoked at $i$ due to $j$. It is noted that $\lambda_{ij}$ additionally depends on a last postsynaptic spike $\hat{t}_i$, such that a reset in the membrane potential is enforced at an actual output spike timing. The full derivation of this rule can be found in Florian (2012).

The E-learning rule in Eq. (3.22) can be understood to operate in three distinct phases: first, each independent target spike (those which do not have a corresponding actual spike close to them) triggers an increase in the weight $w_{ij}$ for each synapse $j$, that is proportional to the PSP $\lambda_{ij}$ elicited at this moment in time. Second, each independent actual spike (those which do not have a corresponding target spike close to them) triggers a decrease in the weight for each synapse, that is proportional to the PSP elicited at this moment in time. Finally, each pair of target and actual spikes which are in one-one correspondence with each other triggers a change in the weight for each synapse, that is proportional to the relative timing of the actual spike with respect to its corresponding target spike, as well as the PSP elicited at the timing of the actual spike. By this mechanism, E-learning works to encourage postsynaptic spiking at target timings, suppress activity at erroneous actual timings and shift actual spikes towards their respective target timings based on their current temporal precision. When every target spike is precisely matched by the neuron, then weight changes become zero for each synapse and there is no further learning.

In Florian (2012), the performance of the E-learning rule was tested through exhaustive simulations studying its ability to transform arbitrary input spike patterns into temporally precise sequences of target output spikes. The studied rule was demonstrated to be capable of matching target postsynaptic spike patterns with very high temporal precision, and reduced the impact of input noise.
on output responses during learning. In particular, E-learning allowed for a relatively large number of input patterns to be memorised by an SNN for each of its synapses (that is its memory capacity), where classes patterns belonged to were identified using the latency of an output spike with single millisecond precision. As an indication of the rule’s high performance, the maximum number of patterns that could reliably be memorised by an SNN when trained under E-learning was around an order of magnitude larger than that when trained under the ReSuMe rule.

A significant limitation of the E-learning rule arises from its biological implausibility in comparison with most other supervised methods, owing to the rules restriction to offline learning; in order for weight changes to be effected, it is necessary to wait until the end of a trial to compute the sets: \( F_i \) and \( \tilde{F}_i \), and their complements \( F^*_i \) and \( \tilde{F}^*_i \), respectively. Finally, E-learning is restricted to SRM neurons, making it a model-dependent rule.

### 3.4.4 SPAN (Spike Pattern Association Neuron)

A further supervised learning method, termed Spike Pattern Association Neuron (SPAN), has been proposed by Mohemmed et al. (2012) to train spiking neurons to learn associations between arbitrary input-output spike patterns. In common with ReSuMe, the authors of SPAN take their inspiration from the traditional Widrow-Hoff learning rule, however the authors instead transform spike trains into analogue signals to derive a more general spike-based rule.

The goal of SPAN is to associate a spatio-temporal input signal, \( x_j \in \mathbf{x} \), with a target output signal \( y_{i}^{\text{ref}} \) through synaptic weight modifications in an SNN. If \( x_j \) is the input received from a presynaptic neuron \( j \), and \( y_{i}^{\text{ref}} \) and \( y_i \) refer to the target and actual output signals at a postsynaptic neuron \( i \), respectively, then the Widrow-Hoff rule is defined by

\[
\Delta w_{ij} = \eta (y_{i}^{\text{ref}} - y_i) x_j ,
\]

where \( \eta \) is the learning rate (see also Eq. (3.17)).

As for the ReSuMe rule, the input and output signals in Eq (3.23) are substituted with their corresponding spike trains: \( x_j \rightarrow X_j \), \( y_{i}^{\text{ref}} \rightarrow Y_i^{\text{ref}} \) and \( y_i \rightarrow Y_i \) (see
Eq. (3.1)). However, unlike with ReSuMe, the authors of SPAN next convolve these spike trains with an arbitrary kernel function $K(t)$ to provide the following signals:

$$\tilde{X}_j(t) = \int_{0}^{t} K(s) X_j(t-s) \, ds$$  \hspace{1cm} (3.24)
$$\tilde{Y}_{i}^{\text{ref}}(t) = \int_{0}^{t} K(s) Y_{i}^{\text{ref}}(t-s) \, ds$$  \hspace{1cm} (3.25)
$$\tilde{Y}_{i}(t) = \int_{0}^{t} K(s) Y_{i}(t-s) \, ds.$$  \hspace{1cm} (3.26)

Hence, substituting $x_j$, $y_{i}^{\text{ref}}$ and $y_{i}$ in Eq. (3.23) with $\tilde{X}_j(t)$, $\tilde{Y}_{i}^{\text{ref}}(t)$ and $\tilde{Y}_{i}(t)$, respectively, provides the following learning rule for a spiking neuron:

$$\Delta w_{ij} = \eta \int_{0}^{\infty} \left( \hat{Y}_{i}^{\text{ref}}(t) - \hat{Y}_{i}(t) \right) \hat{X}_j(t) \, dt,$$  \hspace{1cm} (3.27)

where the integration is performed to provide a batch version of the learning rule, as is considered in Mohemmed et al. (2012).

There are many different possible choices of kernel function for $K(t)$; in Mohemmed et al. (2012), the kernel is taken as $K(t) = t/\tau \exp(1 - t/\tau) \Theta(t)$, where $\tau$ is some time constant. From convolving each spike train with this kernel function, the integration in Eq. (3.27) can be performed to provide the following batch learning rule:

$$\Delta w_{ij} = \eta \left[ \sum_{g} \sum_{f} \left( |t_{g}^f - \tilde{t}_{i}^g| + \tau \right) \exp \left( - \frac{|t_{g}^f - \tilde{t}_{i}^g|}{\tau} \right) - \sum_{h} \sum_{f} \left( |t_{j}^f - \tilde{t}_{i}^h| + \tau \right) \exp \left( - \frac{|t_{j}^f - \tilde{t}_{i}^h|}{\tau} \right) \right],$$  \hspace{1cm} (3.28)

where input spike times are indexed by $\{t_{j}^f\}$ for a presynaptic neuron $j$, and target and actual output spike times are indexed by $\{\tilde{t}_{i}^g\}$ and $\{\tilde{t}_{i}^h\}$, respectively, both for a postsynaptic neuron $i$. According to the above equation, weights are modified at each synapse $j$ at the end of a learning trial, although this rule is also implementable as an online rule if continuous weight modifications are desired. Similarly to ReSuMe, the SPAN rule operates based on two Hebbian-
like processes: the first one corresponding to positive correlations between input and target output spikes, and the second corresponding to negative correlations between input and actual output spikes. The difference between this SPAN rule and ReSuMe comes from the shape of the kernel function $K(t)$ used to derive Eq. (3.28).

A closely related supervised method to SPAN is the Precise-Spike-Driven (PSD) synaptic plasticity rule proposed by Yu et al. (2013). The PSD rule can be considered a special case of the SPAN method, since only the input spike train is substituted with a kernel function $K(t)$. The authors of PSD choose to define the kernel by an alpha-function $\alpha(t)$, corresponding to the response of a postsynaptic current triggered by an input spike train (see Eq. (2.4)). Hence, the PSD is defined by

$$\frac{dw_{ij}^{PSD}(t)}{dt} = \eta \left[ Y_{ref}^i(t) - Y_j^i(t) \right] I_{PSC}^j(t),$$  \hspace{1cm} (3.29)$$

where $I_{PSC}^j(t) = \int_0^t \alpha(s) X_j(t-s) \, ds$ is the effective postsynaptic current elicited by presynaptic neuron $j$. Performing an integration over Eq. (3.29) provides a batch version of the PSD rule:

$$\Delta w_{ij}^{PSD} = \eta \left[ \sum_g \sum_f \alpha(\bar{t}_i^g - t_j^f) - \sum_h \sum_f \alpha(t_i^h - t_j^f) \right],$$  \hspace{1cm} (3.30)$$
such that synaptic weights are modified at the end of each learning trial, in an offline manner. When run as an online rule, PSD is more computationally efficient than SPAN since synaptic weight modifications are only triggered at the precise moments of a target or actual output spike (compare Eqs. (3.27) and (3.29)).

Experimental analysis of SPAN in Mohemmed et al. (2012) has indicated its capability in learning to map between multiple input spike patterns and target output spike trains with good temporal precision. SPAN has shown resilience against noisy stimuli, and was demonstrated to perform accurate classifications on a synthetic dataset as a measure of its generalisation ability. The PSD rule has been successfully tested through similar experiments to SPAN, and it would be of interest to directly compare them in future work. Both the SPAN and PSD rules are potentially biologically relevant: they are implementable as online-based learning rules, and are neuron model independent.
As for ReSuMe, the largest limitation of both the SPAN and PSD methods is the heuristic approach taken to derive them: for example, the Widrow-Hoff learning rule as applied to a rate- rather than spike-based neuron model was taken as their starting point, and an arbitrary kernel function was selected to drive synaptic weight modifications. Therefore, the efficiency of these methods cannot be guaranteed analytically. This issue shall be revisited in chapter 5, where we aim to more rigorously formulate learning rules inspired by SPAN and PSD.

### 3.4.5 Optimal STDP for Precise Spiking

A statistical, supervised method for learning desired sequences of postsynaptic spikes has been proposed by Pfister et al. (2006), which optimally relies on the relative timing between presynaptic spike arrival and desired postsynaptic firing. This method can be seen as a continuation of the work by Barber (2003), extending from discrete to continuous time for the purposes of spike-based modelling.

The aim of this statistical method is to optimise, by gradient ascent, synaptic weights \( \{w_{ij}\} \) in an SNN to maximise the likelihood of generating a set of desired postsynaptic spikes \( y_{i}^{\text{ref}} \) in response to a spatio-temporal spike pattern \( x \). To this end, it is necessary to assume spiking neurons constituting an SNN behave as stochastic units, such that a generative model of postsynaptic activity can be described. This model is then used to form the basis of a suitable objective function, i.e. one that depends smoothly on network parameters, in order that the likelihood of desired sequences of postsynaptic spikes can be maximised.

The probabilistic framework in which this spike-based learning rule is derived is general, and has successfully been applied in establishing rules in the areas of both supervised and reinforcement learning for diverse SNN structures (Pfister et al., 2006; Florian, 2007; Brea et al., 2013; Rezende & Gerstner, 2014). This observation has inspired the contributions of this thesis, for which the methodology in Pfister et al. (2006) has been adapted to suite our reward-modulated (see chapter 4) and supervised (see chapters 5 and 6) learning rules for single- and multilayer SNN structures. Therefore, it is appropriate that this section goes into detail concerning the formulation behind the learning rule proposed by Pfister et al. (2006), including its applications as indicated through simulations.
Neuron Model. As a starting point, the authors of Pfister et al. (2006) consider a single postsynaptic neuron \( i \) receiving its inputs from a total of \( n_j \) presynaptic neurons. Each presynaptic neuron, indexed by \( j \), contributes a list of spikes \( x_j = \{ t_{f_j}^1, t_{f_j}^2, \ldots \} \), where \( t_{f_j}^f \) denotes the \( f \)-th spike fired. The spatio-temporal input pattern due to all presynaptic spikes is denoted by boldface \( x = \{ x_1, x_2, \ldots, x_{n_j} \} \). Hence, if the postsynaptic neuron generates a list of output spikes \( y_i(t) = \{ t_{i}^1, t_{i}^2, \ldots \hat{t}_i < t \} \) in response to \( x \), where \( \hat{t}_i \) is always the last postsynaptic spike before time \( t \), then its membrane potential at time \( t \) is defined by the simplified Spike Response Model (SRM\(_0\)):

\[
u_i(t | x, y_i) := \sum_j w_{ij} \sum_{t_{f_j}^s \in x_j} \epsilon(t - t_{f_j}^s) + \sum_{t_{i}^s \in y_i} \kappa(t - t_{i}^s), \quad (3.31)\]

where \( w_{ij} \) is the synaptic weight between neurons \( i \) and \( j \), and \( \epsilon \) and \( \kappa \) are the PSP and reset kernels respectively (Gerstner & Kistler, 2002). Note that this variant of the SRM lacks a dependence of the PSP on historic postsynaptic spikes, and the reset kernel is summed over all postsynaptic spikes (compare with the more general SRM defined in Eq. (2.5)).

Fundamentally, the method of Pfister et al. (2006) assumes that the firing events of a postsynaptic neuron \( i \) are generated by a point process with stochastic intensity \( \rho_i(t) = g(u_i(t)) \), or in other words, postsynaptic spikes are distributed according to an underlying, instantaneous firing density \( \rho_i(t) \) that depends non-linearly on the neuron’s momentary membrane potential \( u_i(t) \) (see section 2.2.3 in the previous chapter). The function \( g(u_i(t)) \) is referred to as the ‘escape rate’ for which many different choices exist (Gerstner & Kistler, 2002). According to Eq. (3.31) the postsynaptic membrane potential depends on both its input and firing history, therefore the stochastic intensity is written in full as

\[
\rho_i(t | x, y_i) = g(u_i(t | x, y_i)). \quad (3.32)
\]

The above is distinct from a Poisson process since the stochastic intensity \( \rho_i(t | x, y_i) \) depends on previous postsynaptic spikes, hence this neuron model has some memory of its own history.
Renewal Statistics. The approach taken by Pfister et al. (2006) is to analyse the likelihood of generating an observed postsynaptic spike train $y_i$ over the duration of an observation period $T$. However, to fully appreciate this, it is necessary that we first understand some background theory of renewal statistics in relation to probabilistic neuronal spiking.

A renewal process is ideally suited to describing the occurrence of neuronal spike events, given that it takes into account the effect of refractoriness triggered by a previous spike; in this sense, a renewal process is considered a generalisation of a memoryless Poisson process. In a renewal process, spikes are generated with a stochastic intensity $\rho(t|\hat{t})$ (c.f. Eq. (3.32)) which depends on the time since a last spike at $\hat{t}$. Hence, the aim here is to predict the probability of a next spike occurring given some time interval $t - \hat{t}$, which is characterised by an interval distribution $P(t|\hat{t})$.

The interval distribution is a probability density, and its integration over time $\int_{\hat{t}}^{t} P(t'|\hat{t})dt'$ is the probability that a neuron emits a spike between $\hat{t}$ and $t$, given the neuron last fired at $\hat{t}$. Therefore, it follows that the probability of a neuron remaining quiescent between $\hat{t}$ and $t$ is given by

$$S(t|\hat{t}) = 1 - \int_{\hat{t}}^{t} P(t'|\hat{t})dt',$$  \hspace{1cm} (3.33)

where $S(t|\hat{t})$ is referred to as the ‘survivor function’ (Gerstner et al., 2014). The survivor function is defined such that it has an initial value $S(\hat{t}|\hat{t}) = 1$ (i.e. the neuron cannot immediately re-emit a spike) and decreases to $S(t|\hat{t}) \rightarrow 0$ for $t \rightarrow \infty$. The survivor function monotonically decreases with time, and its rate of decay is defined by the stochastic intensity (Gerstner et al., 2014):

$$\rho(t|\hat{t}) = -\frac{d}{dt} S(t|\hat{t}) \frac{S(t|\hat{t})}{S(t|\hat{t})}.$$  \hspace{1cm} (3.34)

By solving for $S(t|\hat{t})$ in Eq. (3.34), the survivor function can be expressed as follows:

$$S(t|\hat{t}) = \exp \left( -\int_{\hat{t}}^{t} \rho(t'|\hat{t})dt' \right),$$  \hspace{1cm} (3.35)

corresponding to the probability of having no spikes from $\hat{t}$ to $t$. Furthermore, using the definition of the survivor function in Eqs. (3.33) and (3.34), the interval
distribution is given by

\[ P(t|\hat{t}) = -\frac{d}{dt}S(t|\hat{t}) \]
\[ = \rho(t|\hat{t})S(t|\hat{t}). \]  

(3.36)

Finally, from inserting Eq. (3.35) into (3.36), the interval distribution is expressed explicitly in terms of the stochastic intensity:

\[ P(t|\hat{t}) = \rho(t|\hat{t}) \exp \left( - \int_{\hat{t}}^{t} \rho(t'|\hat{t}) dt' \right). \]

(3.37)

The above equation can be intuitively understood to mean that a neuron must first ‘survive’ the interval (\(\hat{t}, t\)) without firing, in order that its next spike can be emitted at time \(t\). Hence, the above is used to describe the statistical properties of a renewal system.

**Likelihood of Generating a Spike Train.** The interval distribution given by Eq. (3.37) is of use when determining the likelihood of generating an entire output spike train by a neuron over some observation period \(T\). Specifically, the joint probability density, or likelihood, \(P(t_1^i, t_2^i, ..., t_F^i)\) of a postsynaptic neuron \(i\) generating spikes at times \(y_i = \{t_1^i, t_2^i, ..., t_F^i\}\), where \(t_F^i\) is the final spike before \(T\), can be expressed as follows (Pfister et al., 2006):

\[ P(t_1^i, t_2^i, ..., t_F^i) = P(t_1^i) \prod_{j=2}^{F} P(t_j^i|t_{j-1}^i, ..., t_1^i) S(T|t_F^i), \]

(3.38)

where the final term \(S(T|t_F^i)\) is included to determine the likelihood of no spikes occurring between the last spike time \(t_F^i\) and final duration \(T\). The likelihood of \(y_i\) can be expressed explicitly in terms of the stochastic intensity \(\rho_i\) by inserting
Eqs. (3.35) and (3.37) into (3.38):

\[
P(y_i) = \rho_i(t_i^f|y_i) \exp \left( - \int_0^{t_i^f} \rho_i(t|y_i) dt \right) \\
\times \left\{ \prod_{j=2}^F \rho_i(t_i^f|y_i) \exp \left( - \int_{t_i^{f-1}}^{t_i^f} \rho_i(t|y_i) dt \right) \right\} \exp \left( - \int_{t_i^F}^T \rho_i(t|y_i) dt \right) \\
= \left( \prod_{t_i^f \in y_i} \rho_i(t_i^f|y_i) \right) \exp \left( - \int_0^T \rho_i(t|y_i) dt \right),
\]

(3.39)

where \(\rho_i(t|y_i)\) denotes the dependence of the postsynaptic neuron’s stochastic intensity on its previous spikes \(y_i\) up to \(t\). If the output spike train is generated in response to an input pattern \(x\), then we can write:

\[
P(y_i|x) = \left( \prod_{t_i^f \in y_i} \rho_i(t_i^f|x, y_i) \right) \exp \left( - \int_0^T \rho_i(t|x, y_i) dt \right) \\
= \exp \left( \int_0^T \log(\rho_i(t|x, y_i)) \mathcal{Y}_i(t) - \rho_i(t|x, y_i) dt \right),
\]

(3.40)

where \(\mathcal{Y}_i(t) = \sum_{t_i^f \in y_i} \delta(t - t_i^f)\) is the postsynaptic neuron’s output spike train, and the stochastic intensity \(\rho_i(t|x, y_i)\) is defined by Eq. (3.32). The above summarises the main theoretical result of Pfister et al. (2006), and provides us with a generative model to describe the probability density (likelihood) of observing an actual output spike train \(\mathcal{Y}_i\) in response to some input pattern \(x\) lasting \(T\). Additionally, instead of the likelihood, the logarithm of the likelihood can be taken:

\[
\log P(y_i|x) = \int_0^T \log(\rho_i(t|x, y_i)) \mathcal{Y}_i(t) - \rho_i(t|x, y_i) dt,
\]

(3.41)

which is termed the log-likelihood, and is generally more convenient to work with during statistical analysis. Importantly, if the neuron model is described by a linear SRM and the escape rate is exponential (see Eqs. (3.31) and (2.8), respectively), then the log-likelihood is a concave function of its parameters (Paninski, 2004). Log-concavity is ideal since it ensures no non-global local maxima exist in the likelihood, thereby allowing for computationally efficient parameter optimisation methods.
Learning a Desired Spike Train. Ultimately, the aim is to learn a desired postsynaptic spike train \( y_i^{ref} = \{t_1^i, t_2^i, \ldots \} \) in response to a given input pattern \( x \) by finding an optimal set of synaptic weights \( \{w_{ij}\} \) in an SNN. To this end, it is first necessary to propose a suitable objective function of \( y_i^{ref} \) that relates to Eq. (3.41) and can be maximised through synaptic weight modifications. An obvious choice is to impose \( y_i = y_i^{ref} \), providing the following function:

\[
\log P(y_i^{ref}|x) = \int_0^T \log(\rho_i(t|x, y_i)) \gamma_i^{ref}(t) - \rho_i(t|x, y_i) dt ,
\]

(3.42)

which describes the probability of generating \( y_i^{ref} \) given \( \rho_i(t|x, y_i) \) which in turn depends on the input \( x \). It is emphasised, however, that the sampled stochastic intensity \( \rho_i(t|x, y_i) \) is used, which depends on the actual output spike train \( y_i \), and not \( y_i^{ref} \); this is motivated by a desire for biological relevance, since it is unrealistic to presume we have prior knowledge of \( \rho_i(t|x, y_i^{ref}) \) during learning.

The standard technique of gradient ascent is used as an optimisation procedure:

\[
\Delta w_{ij} = \eta \frac{\partial \log P(y_i^{ref}|x)}{\partial w_{ij}} ,
\]

(3.43)

where \( \eta > 0 \) is the learning rate. Taking the derivative of Eq. (3.42) and using the definition of \( u_i \) in Eq. (3.31) provides the gradient of the log-likelihood (Pfister et al., 2006):

\[
\frac{\partial \log P(y_i^{ref}|x)}{\partial w_{ij}} = \int_0^T \frac{\rho'_i(t|x, y_i)}{\rho_i(t|x, y_i)} [\gamma_i^{ref}(t) - \rho_i(t|x, y_i)] \sum_{t' \in x_j} \epsilon(t - t'_j) dt ,
\]

(3.44)

where \( \rho'_i(t|x, y_i) = \frac{d\rho_i(t|x, y_i)}{du} \big|_{u=u_i(t|x, y_i)} \). Moreover, if the stochastic intensity is defined by an exponential escape rate function (see Eq. (2.8)) it follows that

\[
\frac{\rho'_i(t|x, y_i)}{\rho_i(t|x, y_i)} = \frac{1}{\Delta u} ,
\]

(3.45)

which when inserted into Eq. (3.44) and then combined with Eq. (3.43) gives the
final weight update rule:

$$\Delta w_{ij} = \eta \int_0^T \left[ Y^\text{ref}_i(t) - \rho_i(t|x, y_i) \right] \sum_{t^f_j \in x_j} \epsilon(t - t^f_j) dt,$$  \hspace{1cm} (3.46)

where the factor $\frac{1}{\Delta u}$ has been folded into the learning rate $\eta$. This rule describes a two-phase learning window similar to the process of STDP: the positive term triggers synaptic potentiation based on coincident pre- and then target postsynaptic spikes, and the negative term triggers synaptic depression through correlations between presynaptic spikes and postsynaptic activity. Although weight updates in this case do not rely explicitly on the timings of actual output spikes, it does rely on the instantaneous firing density $\rho_i$ which in turn reflects the underlying membrane potential $u_i$; encouragingly, experimental evidence of STDP suggests that plasticity changes also depend on the momentary membrane potential of a postsynaptic neuron, rather than just the actual timings of spikes themselves \cite{Sjostrom2001, Lisman2005}.

**Simulations.** In \cite{Pfister2006} a set of experiments were devised to explore synaptic weight modifications for a postsynaptic neuron learning various tasks and subject to different stimulation protocols. Depending on the task, a supervised objective function related to Eq. (3.40) was used for maximisation purposes. Stimulation to the neuron might consist of either a single presynaptic spike or an entire spatio-temporal input pattern, combined with an external supervisory signal to encourage output activity at desired time(s).

The authors considered three scenarios of learning: first, a single pre- and postsynaptic neuron constrained to learn a single, desired output spike timing. Second, a single pre- and postsynaptic neuron as before, but this time allowing for spontaneous activity in addition to learning a desired output spike. Third, a temporal coding scheme where a postsynaptic neuron must learn to respond to just one (out of several) presynaptic spike patterns with a desired output spike train while remaining quiescent for the remaining other patterns. Interestingly, from the first two learning scenarios, the authors demonstrated certain similarities between weight updates triggered by their optimisation method and that found experimentally from STDP studies \cite{Bi1998, Zhang1998}. From
the third learning scenario examining a temporal coding scheme, the authors indicated at a potential application of their supervised method for performing spike-based classifications of input patterns. However, their approach was restricted to learning at most two target output spikes for pattern encodings, thus making it difficult to predict its general suitability as a multi-spike based neural classifier.

**Related Work.** The optimality approach taken by Pfister et al. (2006) for deriving a class of supervised learning rules is flexible, and has led to its extension to training more complex recurrent SNN structures (Brea et al., 2011, 2013). As our main contribution we have combined this optimisation method with backpropagated learning as applied to multilayer SNNs (Gardner et al., 2015), which is the topic of chapter 6. We have also explored the performance of the supervised learning rule defined by Eq. (3.46) in classifying spike patterns based on the timings of multiple output spikes, using single-layer SNNs (Gardner & Grüning, 2014); our results obtained using this classification method indicated a high network memory capacity, comparable with that of the E-learning rule in Florian (2012). Moreover, the optimisation method of Pfister et al. (2006) equally applies to reinforcement learning paradigms (Urbanczik & Senn, 2009; Frémaux et al., 2010, 2013), and has been extended to learning by reinforcement in recurrent SNNs (Rezende & Gerstner, 2014). Our earlier contributions have investigated reward-modulated learning in single-layer SNNs based on this optimisation principle (Gardner & Grüning, 2013 Gardner et al., 2014). This shall be the topic of the next chapter, where it will be shown how the likelihood function of Eq. (3.41) relates to the R-max rule defined by Eq. (3.8) in section 3.3.

Although the parameter optimisation method of Pfister et al. (2006) is theoretically well-defined, it is also strongly model-dependent: relying on an SRM neuron with escape noise as part of its formulation. As discussed previously, it is of biological relevance to simulate more complex neuron models to study the effect of diverse firing behaviours on learning. A potential solution to this issue might instead consider model-independent learning rules that function *similarly* to that of Pfister et al. (2006), or in other words, the theory serves as the basis for heuristically formulating a simplified learning rule (Florian, 2007).
3.4.6 Discussion

This section has reviewed several supervised learning rules for training SNNs through synaptic weight modifications, such that associations are learned between spatio-temporal input patterns and sequences of temporally precise output spikes. Some supervised methods, such as ReSuMe and SPAN, rely on relating the Widrow-Hoff rule, as used for rate-based neurons, to spike-based learning, while others such as SpikeProp and CHRON work through minimising an error function by gradient descent. A more novel approach, optimal STDP for precise spiking, optimises weights in an SNN to maximise the likelihood of generating a desired output spike train. The selection of supervised rules considered in this section is by no means exhaustive; additional rules have been developed for SNNs (Güttig & Sompolinsky, 2006; Albers et al., 2013; Xu et al., 2013; Memmesheimer et al., 2014), many of which similarly start with an error function that is to be minimised through gradient descent, or by assuming a spike-based analogue of the Widrow-Hoff rule as used for networks of rate-based neurons. The rules reviewed in this section, however, are generally representative of most supervised learning methods for SNNs.

During the course of this review, several key criteria for establishing an effective supervised learning method have been identified. First, any biologically relevant method should be capable of learning multiple and precisely timed output spikes. Most methods considered here fulfil this requirement, although the SpikeProp rule and the Tempotron proposed by Güttig & Sompolinsky (2006) are only capable of learning single output spikes. A second criterion is the analytical rigour of the method, which should ideally have strong theoretical justification. The statistical method of Pfister et al. (2006) is a clear example of this, especially since it has been proven to have a unique global maximum that is obtainable through gradient ascent (Paninski, 2004). By contrast, the methods based on an interpreted Widrow-Hoff rule, as applied to a non-spiking neuron model, cannot be guaranteed to provide optimal solutions. A final key criterion is that a method be general, such that it is extendible to diverse SNN structures. Examples of this include the SpikeProp rule as derived for multilayer SNNs, ReSuMe as extended to multilayer learning (Sporea & Grüning, 2013) and the statistical method of Pfister et al. (2006) as extended to recurrent SNNs (Brea et al., 2013).
together, the supervised method of Pfister et al. (2006) satisfies many of these criteria, hence motivating our selection of this approach to form the basis of the contributions of this thesis.

These supervised methods have all assumed the presence of an instructive signal to inform weight updates during learning. Presently, it remains unclear where such a signal might originate from, representing the largest source of uncertainty regarding their biological plausibility. A possible explanation for supervised learning might come from so-called ‘referent activity templates’, that are spike patterns generated by neurons existing elsewhere in the brain, which are to be mimicked by networks of interest during learning (Knudsen, 1994; Miall & Wolpert, 1996). For example, such a mechanism has been offered as an explanation for functional plasticity changes in neurons encoding for auditory stimuli in the barn owl (Knudsen, 2002). A further possibility, and one that is gaining increasing interest, is that instructed signalling might actually represent an instantiation of reinforcement learning, but operating on a much smaller time scale. An example of this idea is provided in our previous work, where we have successfully demonstrated how reward-modulated synaptic plasticity can lead to the learning of multiple, and precisely timed, target output spikes (Gardner & Grüning, 2013).

Finally, it should be mentioned that although the reviewed methods of this section have been formulated using gradient-based procedures, there also exist methods that utilise alternative forms of optimisation for training SNNs. An example of this is evolutionary multi-objective optimisation, for which a genetic algorithm can be applied to optimising the connectivity (such as the strength of synaptic weights and connection delays) between spiking neurons in a network (Jin et al., 2007). Evolutionary-based optimisation is competitive with gradient-based learning, as indicated in Jin et al. (2007), however this approach falls outside of the scope of this thesis.

3.5 Chapter Summary

Learning and memory formation in the brain are postulated to rely on the modification of synaptic strengths in large networks of interconnected neurons, or neural networks. This chapter has begun by reviewing two prominent synaptic
plasticity rules for strengthening or weakening synapses over time: STDP and homoestatic plasticity, both of which have been identified as operating in the nervous system through in vivo and in vitro experiments. The process of STDP is of particular interest, since it relies on the precise, relative timings of pre- and postsynaptic spikes to elicit weight changes between spiking neurons.

As discussed in this chapter, STDP can be applied to training SNNs to learn spatio-temporal spike patterns in an unsupervised manner, even when a repeatedly presented pattern is embedded in background noise. This represents a biologically plausible learning method, since minimal instruction is provided to the network during training, and a temporal rank-order code emerges by which patterns can be reliably identified. However, by itself, STDP is only capable of unsupervised learning, and must be informed by external instruction if specific representations of input stimuli are desired. Furthermore, it has been noted that when a detector neuron fires a single spike in response to a pattern then only a small segment of that pattern, before the spike, is actually identified by the neuron, and the rest effectively ignored. This detail is important to take into account when constructing any spike-based neural classifier system, although the contributions of this thesis are primarily focused on a proof of concept of proposed learning rules rather than their real-world pattern recognition capability.

As an alternative to unsupervised learning in SNNs, hypotheses of reinforcement- and supervised-based learning in SNNs have been proposed, which rely on reward-modulated or instructed synaptic modifications, respectively, to form more specific stimulus-response associations. This chapter has presented a generic class of reward-modulated STDP rules, where candidate weight changes triggered by unsupervised STDP are subsequently reinforced by a delayed reward signal for goal-directed learning. Finally, and proportionate to the large body of previous studies regarding supervised learning in SNNs, the final section of this chapter has provided a thorough review of spike-based supervised learning methods that are relevant to the scope of this thesis. Supervised learning is advantageous over reinforcement learning by more rapidly training SNNs to encode input patterns using multiple and precisely-timed output spikes, although this comes at the cost of biological plausibility, considering it relies explicitly on instructed signalling for learning.
In the course of this review a statistical supervised method presented by Pfister et al. (2006) has been deemed to be a high performance, theoretically justified learning rule that is very flexible in terms of its application and yet retains a reasonable degree of biological realism. This method works to optimise synaptic weights in an SNN such that the likelihood of generating a desired output spike train is maximised, and is adaptable to reinforcement learning and diverse spiking network structures. For these reasons, the contributions of this thesis are developed within the statistical framework of this method.
Chapter 4

Reward-Modulated Learning for Precise Spiking

This chapter applies the Reward-maximisation (R-max) learning rule, as discussed in subsection 3.3.4, to training an SNN to form associations between spatio-temporal input patterns and sequences of temporally precise output spikes. Since this chapter is primarily concerned with biological realism, and to differentiate this approach from supervised learning, only a delayed reward signal is considered, such that general feedback signalling the correctness of a network response only becomes available upon the cessation of a presented input pattern. This chapter also compares the difference in performance between two escape rate functions defining output firing activity: the Arrhenius & Current (A&C) and Exponential (EXP) models. Parts of this contribution have been published in Gardner & Grüning (2013); Gardner et al. (2014).

4.1 Introduction

There is increasing interest in reward-modulated synaptic plasticity as a biologically plausible rule for modelling learning in the brain, such that candidate weight changes triggered by correlations between pre- and postsynaptic spikes further rely on the presence of an external reward signal to become persistent (Florian 2007; Izhikevich 2007; Legenstein et al. 2008). For a detailed sum-
Chapter 4. Reward-Modulated Learning for Precise Spiking

mary, see the section on reinforcement learning in chapter 3. However, while there has been significant progress in applying this rule to solving typical reinforcement learning problems such as the cart-pole balancing task (Frémaux et al., 2013), or various navigation tasks (Vasilaki et al., 2009; Friedrich et al., 2011), there is still currently a lack work that has aimed at learning sequences of multiple and precisely-timed output spikes. As discussed in the previous chapter, the learning of a temporal code by neurons is of biological significance.

Therefore, this contribution investigates the efficacy of an SNN using the Reward-maximisation (R-max) rule when applied to learning a target output spike train; the R-max rule is preferentially selected over the R-STDP rule given its increased general performance (Frémaux et al., 2010), as discussed in section 3.3. The novelty of this contribution comes from our implementation of the R-max rule in relation to the learning of a large number of target output spikes, and our comparison between the commonly used EXP and less used but biologically realistic A&C escape rate functions for driving output neuronal firing activity.

This contribution is organised as follows. The next section presents our methods, including: the neuron model used; the definition of the A&C escape rate function; the mathematical formulation of the R-max rule, starting from first principles; and the learning of a target spike train by delayed reinforcement. The following section 4.3 shows results from computer simulations, testing the performance of the R-max rule on learning up to 25 target output spikes by reinforcement. This section compares A&C against EXP in terms of their success rate when learning temporally precise sequences of output spikes, as well as their convergence speed. Finally, section 4.4 discusses the results of this contribution in relation to the choice of network parameters used, and the importance of intermediate background noise as a means to facilitate explorative learning through variable output spiking activity.
4.2 Methods

4.2.1 Single Neuron Model

As an initial step, we consider a fully-connected, single-layer feed-forward SNN, inspired by the setup of Pfister et al. (2006). A single postsynaptic neuron, with index $i$, receives its input from $1 \leq j \leq n_i$ presynaptic neurons. The sequence of spikes received from the $j^{th}$ neuron corresponds to a list of firing times $x_j = \{t^1_j, t^2_j, \ldots\}$, where $t^f_j$ is its $f^{th}$ spike time, and the spatio-temporal spike pattern over all $n_i$ presynaptic neurons is denoted by boldface $\mathbf{x} = \{x_1, \ldots, x_{n_i}\}$.

In the framework of the SRM, the evoked PSP due to a presynaptic spike is approximated well by a double exponential function (Gerstner & Kistler, 2002):

$$
\epsilon(s) = \epsilon_0 \left[ \exp \left( -\frac{s}{\tau_m} \right) - \exp \left( -\frac{s}{\tau_s} \right) \right] \Theta(s),
$$

(4.1)

where the argument $s$ is the lag time since a presynaptic spike, for example $s = t - t^f_j$. The membrane time constant is set to $\tau_m = 10$ ms and the synaptic time constant $\tau_s = 0.7$ ms, such that the PSP attains its peak value after a delay of 2 ms. The PSP coefficient is $\epsilon_0 = 1.3$ mV, resulting in a peak PSP value close to 1 mV. $\Theta(s)$ is the Heaviside step function, which has the value $\Theta(s) = 1$ for $s > 0$ and $\Theta(s) = 0$ otherwise. These parameters choices come from those of Pfister et al. (2006).

The list of spikes emitted by the postsynaptic neuron up to time $t$ is given by $y_i(t) = \{t^1_i, t^2_i, \ldots, \hat{t}_i < t\}$, where $\hat{t}_i$ refers to the most recent spike before $t$. A reset kernel models the postsynaptic neuron’s afterpotential in response to a last output spike, and is described by an exponential function:

$$
\kappa(s) = \kappa_0 \exp \left( -\frac{s}{\tau_m} \right) \Theta(s),
$$

(4.2)

where $s = t - \hat{t}_i$ is the lag time since a last postsynaptic spike in $y_i(t)$. Throughout our simulations we take $\kappa_0 = -5$ mV to ensure the neuron remains quiescent for at least 1 ms after spiking. Refractory effects do not cumulate, therefore only the most recent postsynaptic spike contributes to the neuron’s afterpotential. Spikes themselves aren’t modelled explicitly, and are reduced to formal firing times.
Taken together, Eqs. (4.1) and (4.2) define the SRM (Gerstner & Kistler, 2002); hence, the postsynaptic neuron’s membrane potential in response to an input pattern $x$ and its last firing time $\hat{t}_i$ is given by

$$u_i(t|\mathbf{x, \hat{t}_i}) := u_{\text{rest}} + \sum_j w_{ij} \int_{\hat{t}_i}^{t} \epsilon(t - t') \mathcal{X}_j(t') \, dt' + \kappa(t - \hat{t}_i),$$

(4.3)

where $u_{\text{rest}} = -70 \, \text{mV}$ is the neuron’s resting potential and $\mathcal{X}_j(t) = \sum_f \delta(t - t^f_j)$ is the formalised spike train of the $j^{th}$ presynaptic neuron, as originally defined by Eq. (3.1). Only PSP’s evoked by presynaptic spikes after $\hat{t}_i$ are summed over, ensuring a ‘hard-reset’ at $\hat{t}_i$ to prevent excessive postsynaptic spiking (Frémaux et al., 2010). This SRM model is closely related to the generalised model presented in Eq. (2.5).

To account for variations in output firing activity due to background noise we implement the escape noise model as a stochastic spike generator, as discussed in subsection 2.2.3. To summarise, postsynaptic firing events are driven by a stochastic intensity $\rho_i(t) = g(u_i(t))$, that is the postsynaptic neuron’s instantaneous firing density which in turn depends nonlinearly on its momentary membrane potential. In our simulations we set the internal time step to $\delta t = 1 \, \text{ms}$, hence the probability of the neuron firing at time $t$ can be approximated by $\rho_i(t)\delta t$, after expanding Eq. (2.7) for small $\delta t$.

We note that the time step $\delta t$ used for Euler integration in our experiments was somewhat larger than the synaptic rise time constant $\tau_s$ by 0.3 ms, which might potentially contribute to numerical instabilities during simulation runs and provide less accurate solutions. Our selection of a relatively large $\delta t$ value was motivated by a need for faster simulation run times, especially since we considered a very large number of learning iterations to ensure convergence in network training. Despite this single parameter discrepancy, final results showed no indication of numerical instabilities and network training remained stable in all cases.

Here we consider two different escape rate functions that define the postsynaptic neuron’s stochastic intensity, the first being the Arrhenius & Current (A&C)
model (Gerstner & Kistler 2002):

\[ g^{A&C}(u_i, \dot{u}_i) = 2 \left( \frac{c_1}{\tau_m} + \frac{c_2}{\sigma} [\dot{u}_i]_+ \right) \frac{\exp \left\{ -\frac{[u_i-\vartheta]^2}{\sigma^2} \right\}}{1 + \text{erf} \left\{ -\frac{u_i-\vartheta}{\sigma} \right\}}, \]  

(4.4)

where \( u_i \) is the time-dependent, noise-free trajectory of the neuron’s membrane potential and \( \dot{u}_i = \frac{du_i}{dt} \) its time derivative. The firing threshold is set to \( \vartheta = -55 \text{ mV} \). The parameter \( \sigma \) is the noise amplitude, corresponding to the magnitude of fluctuations in \( u_i \) due to background stochastic spike arrival. We set \( \sigma = 5 \text{ mV} \), mimicking that measured from in vivo experiments (Chance et al., 2002; Stern et al., 1997). The parameters \( c_1 \) and \( c_2 \) are set to 0.72 and \( \frac{1}{\sqrt{\pi}} \), respectively (Plesser & Gerstner, 2000). The term \([\dot{u}_i]_+\) indicates that only positive gradients in the membrane potential contribute to the stochastic intensity, defined by \([\dot{u}_i]_+ = (|\dot{u}_i| + \dot{u}_i)/2\). The error function erf is included to ensure a linear increase in the stochastic intensity for \( u_i > \vartheta \) (Herrmann & Gerstner 2001).

The second, simpler escape rate function, henceforth referred to as Exponential (EXP), is more commonly used in simulations (Gerstner & Kistler 2002):

\[ g^{\text{EXP}}(u_i) = \rho_0 \exp \left( \frac{u_i - \vartheta}{\Delta u} \right), \]  

(4.5)

where we set the stochasticity parameters \( \rho_0 = 0.156 \text{ ms}^{-1} \) and \( \Delta u = 3 \text{ mV} \). These parameter choices were made such that for \( u_i < \vartheta \): \( g^{\text{EXP}}(u_i) \approx g^{A&C}(u_i, \dot{u}_i = 0) \), giving comparable noise levels between both escape rate functions. Importantly, these parameters are consistent with those provided in Jolivet et al. (2006), which closely approximate the variable firing activity of somatosensory cortex neurons as measured from in-vitro experiments.

Our motivation for including the A&C model comes from its comparative realism over EXP, since it has previously been found to be highly accurate in recreating spike-timing distributions of simulated LIF neurons that are subject to Gaussian white-noise (Plesser & Gerstner, 2000). Such conditions are considered to be similar to those encountered by real neurons in the nervous system that are continuously bombarded by ‘random’ background spikes due to the many tens of thousands of synaptic connections they receive. The more common usage of
EXP over the A&C model in simulations of stochastic spiking neurons comes mainly from its comparative functional simplicity, making it more amiable to analytical treatment. In this contribution we seek to apply both escape rate models to postsynaptic neurons learning arbitrary target output spike trains, with the intention of comparing their relative performance.

4.2.2 Reward-modulated Synaptic Plasticity Rule

This subsection goes through the formulation behind the R-max learning rule in detail, as originally presented in subsection 3.3.4. The R-max rule works based on the optimality principle of Pfister et al. (2006), such that the expectation of reward received by an SNN is maximised through reward-modulated synaptic weight modifications.

As found by Pfister et al. (2006), using a stochastic spike generator allows us to determine the likelihood of observing a list of postsynaptic spikes $y_i$ in response to an input $x$. As shown in subsection 3.4.5, the logarithm of the probability density, or log-likelihood, $\log P(y_i|x)$ of generating $y_i$ in response to $x$ over an observation period $T$ is given by

$$\log P(y_i|x) = \int_0^T \log \rho_i(t) Y_i(t) - \rho_i(t) dt,$$

where $Y_i(t) = \sum_{t'_i \in y_i} \delta(t-t'_i)$ is the actual output spike train of the postsynaptic neuron.

We wish to relate the log-likelihood function, defined by Eq. (4.6), to reinforcement instead of supervised learning. To this end, a global reinforcement signal $\mathcal{R}(y_i|x)$ is defined that depends on the entire postsynaptic output $y_i$ conditioned on an input $x$. In reinforcement learning the optimised parameter is the expected reward $\langle \mathcal{R}(y_i|x) \rangle_{y_i|x}$ which is averaged over all lists of output spikes:

$$\langle \mathcal{R}(y_i|x) \rangle_{y_i|x} = \int \mathcal{R}(y_i|x) P(y_i|x)dy_i.$$

As shown by Pfister et al. (2006); Florian (2007), a learning rule can be defined by maximising the expected reward using a standard gradient ascent procedure
Chapter 4. Reward-Modulated Learning for Precise Spiking

with respect to synaptic weights in an SNN:

\[ \langle \Delta w_{ij} \rangle_{y_i|x} = \eta \left( R(y_i|x) \frac{\partial \log P(y_i|x)}{\partial w_{ij}} \right)_{y_i|x} , \]  

(4.8)

where \( \eta \) is the learning rate, and having used the relation \( \frac{1}{P} \frac{\partial P}{\partial w_{ij}} = \frac{\partial \log P}{\partial w_{ij}} \). Since our best estimate for the gradient during learning comes from a current observation of \( y_i \) given \( x \), the expectation operator is dropped to provide:

\[ \Delta w_{ij} = \eta R(y_i|x) \frac{\partial \log P(y_i|x)}{\partial w_{ij}} , \]

(4.9)

where weight updates currently take place in an iterative manner. By using the SRM defined by Eq. (4.3), the gradient of the log-likelihood is solved similarly to Eq. (3.44), giving

\[ \frac{\partial \log P(y_i|x)}{\partial w_{ij}} = \int_0^T \rho'_i(t) \rho_i(t) \left[ Y_i(t) - \rho_i(t) \right] \left( \int_{t_i}^t \epsilon(t-t') X_j(t') dt' \right) dt , \]

(4.10)

where \( \rho'_i(t) = \frac{dg}{du}|_{u=u_i(t)} \).

In a biological setting, it is unrealistic to assume that the neuron has prior knowledge of the duration \( T \) over which the stimulus is presented, after which the reward signal \( R \) is delivered. Therefore, by the same approach as Urbanczik & Senn (2009); Friedrich et al. (2010), an instantaneous synaptic eligibility term \( e_{ij}(t) \) is defined as the integrand of Eq. (4.10), which is low-pass filtered to provide a moving average called the synaptic eligibility trace \( E_{ij}(t) \). To summarise, the synaptic eligibility is defined by

\[ e_{ij}(t) = \frac{\rho'_i(t)}{\rho_i(t)} \left[ Y_i(t) - \rho_i(t) \right] \int_{t_i}^t \epsilon(t-t') X_j(t') dt' , \]

(4.11)

and synaptic weight modifications are now determined by the set of equations:

\[ \tau_e \frac{dE_{ij}(t)}{dt} = -E_{ij}(t) + \eta e_{ij}(t) \]

(4.12)

\[ \frac{dw_{ij}(t)}{dt} = R(t) E_{ij}(t) \]

(4.13)

where \( \tau_e \) is the eligibility time constant, which is typically matched to the stimulus
duration: $\tau_e \approx T$ [Urbanczik & Senn, 2009; Frémaux et al., 2010], and $\eta$ is the learning rate. According to Eqs. (4.11), (4.12) and (4.13) synaptic weights modifications now take place continuously using a time-dependent reward signal $R(t)$, defining the R-max learning rule as originally discussed in section 3.3. Currently, the reward $R$ acts in place of the global success signal $S[R]$ as defined previously in Eq. (3.6). As shall be shown later in this section, the learning of a target output spike train is more efficacious when $R$ is substituted with an appropriate functional choice for $S[R]$.

For each escape rate model, defined by Eqs. (4.4) and (4.5), the corresponding synaptic eligibility defined by Eq. (4.11) is provided. For the A&C model we determined the eligibility as

$$e_{ij}^{A&C}(t) = A(u_i) \left[ Y_i(t) - g^{A&C}(u_i, \dot{u}_i) \right] \int_{\hat{t}_i}^t \epsilon(t - t') X_j(t') \, dt'$$

with

$$A(u_i) = \frac{2}{\sigma} \left( \frac{1}{\sqrt{\pi}} \exp \left\{ -\frac{(u_i - \vartheta)^2}{\sigma^2} \right\} - \frac{u_i - \vartheta}{\sigma} \right) \cdot$$

For simplicity, we neglected an acceleration-dependent term containing $\ddot{u}$: although such a neglected term may not necessarily be numerically negligible, simulations showed that learning nevertheless performed well under this approximation. For the EXP model, the eligibility is simply given by

$$e_{ij}^{\text{EXP}}(t) = \frac{1}{\Delta u} \left[ Y_i(t) - g^{\text{EXP}}(u_i) \right] \int_{\hat{t}_i}^t \epsilon(t - t') X_j(t') \, dt'$$

which is similar to that defined by Eq. (3.8), except the above equation instead integrates over previous presynaptic spikes coming after $\hat{t}_i$.

### 4.2.3 Learning a Target Spike Train

We wish to train an SNN to respond to a spatio-temporal spike pattern $x$ with a sequence of output spikes $y_i = \{t_1^i, t_2^i, \ldots\}$ that matches as closely as possible a desired sequence of target output spikes $y_i^{\text{ref}} = \{\tilde{t}_1^i, \tilde{t}_2^i, \ldots\}$. The R-max learning rule is used for this purpose, where a delayed reward signal $R$ is globally released to all synapses in the network upon cessation of the input $x$. In this way, the
Chapter 4. Reward-Modulated Learning for Precise Spiking

reward signal provides feedback to the network regarding the overall correctness of its response. This entire process can be described as a form of episodic learning, where each learning episode corresponds to the presentation of an input pattern to the network lasting time $T$.

To formalise this, we denote the episode number by $n$ and redefine the reward signal as an impulse delivered to the network at time $t = T$ at the end of $n$th episode:

$$ R(t) := R(n) \delta(t - T) , \quad (4.16) $$

where $R(n)$ now denotes the reward more simply as a function of the episode number, instead of time. From substituting this redefined value for $R(t)$ into Eq. (4.13) and then performing the integration over time, the total weight change per episode becomes:

$$ \Delta w_{ij}(n) = \eta R(n) E_{ij}(T) . \quad (4.17) $$

Hence, we now proceed to defining an appropriate functional relationship between the accuracy of network responses and received reward $R$ on each episode.

**The Reward Signal.** Similar in approach to Farries & Fairhall (2007); El-Laithy & Bogdan (2011), we begin by using the van Rossum Distance (vRD) (van Rossum 2001) to measure the (dis)similarity between $y_i$ and $y^\text{ref}_i$. The vRD initially involves performing a convolution over each output spike train, $\mathcal{Y}_i$ and $\mathcal{Y}^\text{ref}_i$ for $y_i$ and $y^\text{ref}_i$, respectively, using an exponential kernel:

$$ \tilde{\mathcal{Y}}_i(t) = \sum_{t'_i \in y_i} \exp \left( -\frac{t - t'_i}{\tau_c} \right) \Theta(t - t'_i) , \quad (4.18) $$

where we set the coincidence time constant $\tau_c = 15$ ms, somewhat longer than the membrane time constant $\tau_m$. Eq. (4.18) is similarly used to obtain the filtered target output spike train $\tilde{\mathcal{Y}}^\text{ref}_i(t)$ from $\mathcal{Y}^\text{ref}_i$. The vRD between the two output spike trains can now be computed, and is defined by van Rossum 2001):

$$ \mathcal{D}(\mathcal{Y}_i, \mathcal{Y}^\text{ref}_i) := \frac{1}{\tau_c} \int_0^\infty \left[ \tilde{\mathcal{Y}}_i(t) - \tilde{\mathcal{Y}}^\text{ref}_i(t) \right]^2 \, dt , \quad (4.19) $$

72
where the value $D = 0$ corresponds to an exact match between $y_i$ and $y_i^{\text{ref}}$. In order to remove the dependence of the vRD on the number of target spikes, we normalise $D$ by instead taking $D^* = D/D_0$, where $D_0$ is the vRD from just $Y_i^{\text{ref}}$. This is expressed by

$$D^*(Y_i, Y_i^{\text{ref}}) = \frac{\int_0^\infty \left[ \tilde{Y}_i(t) - \tilde{Y}_i^{\text{ref}}(t) \right]^2 \, dt}{\int_0^\infty \left[ \tilde{Y}_i^{\text{ref}}(t) \right]^2 \, dt}. \quad (4.20)$$

Finally, this normalised distance measure $D^* \in [0, \infty)$ is mapped to a numerical reward value $R \in (0, 1]$ using a stretched exponential dependence:

$$R = \exp \left( -aD^* \right), \quad (4.21)$$

where we set the scaling factor $a = 4$, such that reward becomes negligible for distances $D^* > 1$ and maximum reward being attained when $D^* = 0$. Values for $a < 4$ are undesirable, given that preliminary simulations demonstrated that such values led to self-sustaining inaccurate spiking responses by an SNN. Finally, we also set $R = 0$ when no output spikes are generated, since a lack of firing activity would prevent learning from taking place.

**Temporal Difference Learning.** Rather than directly substituting the $n^{th}$ episodic reward value $R(n)$ into the weight update rule given by Eq. (4.17) we instead implement an adaptation of the Temporal Difference (TD) learning rule (Farries & Fairhall, 2007), originally defined in classical reinforcement learning (Barto & Sutton, 1998). The biological justification for this comes from the observed similarity of dopaminergic neuron firing activity with an encoded reward prediction error signal (Schultz, 2000), as discussed in section 3.3.

TD learning adjusts synaptic weights based on the error between successive estimates of future reward, where the expectation of future reward, given the current input $x(n)$, is defined by a value function:

$$V^\pi[x(n)] = \mathbb{E}_\pi \left[ \sum_{n' = 0}^{\infty} \gamma^{n'} R(n + n') \right]. \quad (4.22)$$

The operation $\mathbb{E}_\pi[R(n)]$ is the expected future reward when following the current
policy $\pi$, where more distant reward contributes less to $V^\pi[x(n)]$ for a discount factor $\gamma < 1$. The policy is determined by the postsynaptic neuron’s weight vector, which maps an input $x(n)$ to an actual output spike train $y_i(n)$ on the $n^{th}$ episode. Hence, the TD error is defined by (Barto & Sutton 1998):

$$S[R(n)] = R(n) + \gamma V^\pi[x(n+1)] - V^\pi[x(n)] , \quad (4.23)$$

which is used to improve the current policy $\pi$ by summing the actual reward with the updated future expected reward. Given that neuronal responses have no bearing on the selection of future states, Eq. (4.22) can be decomposed as $V^\pi[x(n)] = \langle R \rangle + \gamma \langle R \rangle + \gamma^2 \langle R \rangle + ...$, where the only reward prediction available is the average reward $\langle R \rangle$ under the current policy $\pi$ (Farries & Fairhall 2007). Therefore, equation Eq. (4.23) becomes:

$$S[R(n)] = R(n) + \gamma \langle R \rangle \sum_{n'=0}^{\infty} \gamma^{n'} - \langle R \rangle \sum_{n'=0}^{\infty} \gamma^{n'}$$

$$= R(n) - \langle R \rangle . \quad (4.24)$$

The expected reward is suitably taken as a moving average (Farries & Fairhall 2007), updated after $S$ is computed at the end of each episode, according to:

$$\langle R \rangle \leftarrow \frac{9}{10} \langle R \rangle + \frac{1}{10} R(n) , \quad (4.25)$$

where preliminary simulations indicated that a time constant on the order of 10 episodes gave optimal performance. Finally, after substituting $R$ with the TD or success signal $S[R]$ in Eq. (4.17), the total weight change per episode becomes:

$$\Delta w_{ij}(n) = \eta S[R(n)] E_{ij}(T) . \quad (4.26)$$

Using TD learning to drive weight modifications has the advantage of allowing for both Hebbian and anti-Hebbian plasticity processes to take place, given that $S$ can take either positive or negative values. This has the effect of allowing for smoother convergence towards the target spike train to be learned, by maintaining a recent history of the networks output accuracy. Interestingly, this definition of the success signal is close to optimal in minimising the variance of weight changes, which is ideal for multi-pattern learning (Frémaux et al. 2010).
4.2.4 Plasticity Rules

During network training an analogue of ‘additive’ STDP is implemented, such that a synaptic weight change $\Delta w_{ij}$ is simply clipped if its absolute value $|w_{ij}|$ moves outside of the range $[w_{\text{min}}, w_{\text{max}}]$ (see subsection 3.1.1). We set $w_{\text{min}} = 10^{-3}$ and $w_{\text{max}} = 10$ as the minimum and maximum attainable absolute synaptic weights, respectively. In all cases, plasticity takes place in both excitatory and inhibitory connections, where inhibitory connections have negative values for $w_{ij}$. In these experiments we are interested in simulating neurons with high biological plausibility, therefore we added the additional constraint that excitatory weights are prevented from becoming inhibitory or vice-versa; this satisfies Dale’s principle, which states that a neuron performs the same chemical action at all of its postsynaptic targets [Eccles 1976].

To maintain a homeostatic firing rate and to introduce competition between afferent connections, an adaptation of the synaptic scaling rule defined by Eq. (3.4) is used:

$$\Delta w_{ij}^{\text{scaling}} = \psi |w_{ij}| (\nu_{\text{ref}} - \nu_{i}) ,$$

(4.27)

where the previously defined scaling strength is set to $\psi = 5 \times 10^{-4}$ Hz$^{-1}$. In this case, $\nu_{\text{ref}}$ and $\nu_{i}$ are the target and actual output firing rates, respectively, as measured over a single episode (measured in Hz). The target output firing rate corresponds to the number of target spikes $n_s$ the network has to learn. Synaptic weights are scaled at the end of each learning episode, in addition to being modified by Eq. (4.26).

4.3 Simulation Results

Simulations were run to test the capability of the R-max rule in training an SNN to learn temporally precise sequences of target output spikes by delayed reinforcement. Furthermore, the performance of the considered A&C and EXP models for driving postsynaptic firing activity were compared with respect to their respective success rates and convergence times.
4.3.1 Network Setup and Learning Task

We implemented a fully-connected, feed-forward SNN: consisting of \( n_i = 500 \) presynaptic neurons and a single postsynaptic neuron. The presynaptic neurons provided an input pattern \( \mathbf{x} \) to the network, to be processed by the postsynaptic neuron. The input pattern consisted of an independent Poisson-distributed spike train for each presynaptic neuron, with a mean firing rate of 6 Hz. Our selection of a relatively large number of presynaptic neurons was to ensure that arbitrarily generated target output spike patterns could be learned by the postsynaptic neuron.

Either the A&C or EXP escape rate function determined the output activity of the postsynaptic neuron, defined by Eq. (4.4) or Eq. (4.5), respectively. Synaptic weights were initialized by independently selecting each value from a Gaussian distribution with a mean of 0.32 or 0.26 for an A&C or EXP postsynaptic neuron, respectively, and a standard deviation \( 1/3 \) the mean. These values were selected to drive the initial firing rate of the postsynaptic neuron to 6 Hz. The ratio of the number of excitatory to inhibitory weights was set as 4 : 1, as is typically found in the mammalian cortex [Braitenberg & Schüz 1991].

Learning took place on an episodic basis, where each episode corresponded to the presentation of the same input pattern \( \mathbf{x} \) to the network. Each episode had a duration of \( T = 500 \) ms, hence the eligibility time constant in Eq. (4.12) was similarly set to \( \tau_e = 500 \) ms. The input pattern was associated with an arbitrarily defined target output spike train, which the network’s postsynaptic neuron had to learn to reproduce. The target spike train contained between \( 1 \leq n_s \leq 25 \) spikes, where the timing of each spike was randomly selected from a uniform distribution over \([20, T - 20]\). For multi-spike target trains, a minimum inter-spike interval of 15 ms was enforced to avoid confliction.

In simulations we set the learning rate to \( \eta = 200 \) for both A&C and EXP postsynaptic neurons: our preliminary results indicated that such a value was optimal for both escape rates, where increases in \( \eta \) lead to deterioration in long-term performance.
4.3.2 Learning Temporally Precise Spiking Patterns

As an example of network training under the R-max learning rule, we first tasked the network with learning to form an association between a fixed input pattern $\mathbf{x}$ and a target postsynaptic spike train containing five spikes. As described in the previous section, the input and target output spike patterns were arbitrarily generated, and learning here proceeded over a total of 5000 episodes to ensure convergence. With our choice of $T = 500$ ms, this corresponds to almost 42 minutes of biological time. In this example, the response of the postsynaptic neuron was determined by the A&C escape rate.

Fig. 4.1 demonstrates the output spiking activity of an A&C postsynaptic neuron in response to a repeatedly-presented spatio-temporal spike pattern. The random input pattern is displayed in the left panel, and a spike raster of the output response is shown in the right panel. As shown in the right panel, highly variable output spike timings over the first 1000 episodes were observed, followed by a near spontaneous alignment with the target timings. Interestingly, the first two target timings were relatively close to each other, which the network gradually learned to discriminate by increasing the precision of output spiking.

By relating these observations back to reward-modulated learning, we can interpret the initial phase of variable output activity as a form of stochastic exploration by the network, during which time the network explored reward space to discover desirable sequences of output spikes. As increasingly desirable responses were discovered by the network, participating synapses were further reinforced to maximise the likelihood of generating the target output spike train. It is noted, however, that learning remained imperfect: there still persisted a low level of spontaneous output activity that was unrelated to learning. This is to be expected in a reinforcement learning scheme, for which there is usually a trade-off between exploration in the form of variable output spiking by the network, and exploitation of previously discovered accurate output spike responses.

We next considered the general task of training a network to map between an input pattern and a target output spike train containing a variable number of spikes. On this task, the response of the postsynaptic neuron was determined by either the A&C or EXP escape rate for comparison purposes. The aim was to establish the capability of an SNN trained under R-max in learning a large
Chapter 4. Reward-Modulated Learning for Precise Spiking

Figure 4.1: Mapping a single, fixed, input pattern to an arbitrary five-spike target train. (Left) The input pattern ($x_j \subset x$) repeatedly presented to the network over 5000 episodes, where each dot represents a spike. (Right) Spike raster of an A&C postsynaptic neuron in response to $x$. This spike raster is generally representative of the neuronal activity in response to other similarly generated input patterns.

number of target output spikes, and to determine the effect of an escape rate function on the performance of the network and its convergence speed.

To characterise the performance of the network, we defined a performance measure $P_s$ such that successful responses by the network gave $P_s = 100\%$ and $P_s = 0$ otherwise. We considered a successful response to occur on an episode where every target output spike could be paired to within $\Delta t = 10\text{ ms}$ of an actual output spike given that such values for $\Delta t$ between target and actual output spikes had the effect of reducing $D^*$. We additionally set the constraint that the actual output spike train must contain the same number of spikes as its target, thereby disallowing spurious spiking.

Since output spiking responses fluctuated between episodes, we took the performance as a moving average. The average performance was updated on each episode according to $\bar{P}_s(n) = (1 - \lambda)\bar{P}_s(n - 1) + \lambda P_s(n)$, with the timing parameter set to $\lambda = 0.004$. In this way, $\bar{P}_s(n)$ measured the frequency of successful network responses on the $n^{th}$ episode. To measure the convergence in learning, we took a similar approach to Florian (2007): convergence was considered to

\begin{footnote}
Our justification for the choice of $\Delta t$ comes from the relation $\Delta t = -\tau_c \ln(1 - D^*)$, indicating that time shifts $\Delta t < 10\text{ ms}$ for paired target and actual output spikes have the effect of minimising the overall distance with $D^* < 1$.
\end{footnote}
Figure 4.2: An SNN learning to reproduce an arbitrary target output spike train containing $n_s$ spikes in response to a fixed input pattern. Either the A&C or EXP escape rate determined the response of the postsynaptic neuron. (Left) Performance $\tilde{P}_s$ at convergent episode number $n_c$ for each network. (Right) Number of episodes to convergence $n_c$ in the performance $\tilde{P}_s$ for each network. Each value is an average over 10 independent simulation runs, where error bars show the standard error of the mean.

Fig. 4.2 shows the performance and convergence time for the A&C and EXP escape rates for an SNN learning to reproduce between 1 and 25 target spikes. From this figure it is clear that A&C consistently outperformed EXP, where the difference in $\tilde{P}_s$ between the two models exceeded 60% when learning the maximum number of target spikes. A&C maintained a good level of performance over the entire range of target spikes considered, with a minimum of $\tilde{P}_s = 88.0 \pm 0.9\%$ for 25 target spikes. By contrast, the performance of EXP deteriorated rapidly when learning more than 10 target spikes, with $\tilde{P}_s = 26\pm2\%$ for 25 target spikes. In terms of their convergence time, A&C converged marginally faster than EXP when learning between 1 and 10 target spikes, although the reverse was found when learning more than 10 target spikes, where the convergence time for EXP remained fixed. For A&C, there was an indication of the convergence time levelling off from 20 target spikes. We note that while EXP converged more
rapidly than A&C for a larger number of target spikes, at no point during learning did EXP outperform A&C: the decreased convergence time for EXP reflected the relatively fast attainment of a lower convergent performance value.

For illustration, a typical spike raster corresponding to each escape rate when reproducing a 20-spike target train, after convergence, is shown in Fig. 4.3. Clearly, it can be seen that A&C outperformed EXP, where in this case A&C attained a convergent performance of $92.0 \pm 0.8$% and EXP $40 \pm 2$%, with a difference in excess of 50%. According to Fig. 4.2, the number of episodes to convergence in $\bar{P}_s$ for 20 target spikes was $n_c = (3.2 \pm 0.3) \times 10^4$ and $n_c = (1.8 \pm 0.2) \times 10^4$ for A&C and EXP, respectively. Aside from the more rapid convergence time of EXP over A&C, we found from visual inspection that there existed an intermediate period of ‘fine-tuning’ for A&C, during which time spurious spikes were further eliminated by the network to allow for relatively small but significant gains in the performance.

### 4.4 Discussion

We have explored the utility of a stochastic neuron model in learning to reproduce temporally precise spiking patterns through a reward-modulated synaptic
plasticity rule, representative of a process that might underpin learning in the brain. Furthermore, we have investigated two different escape rate functions to drive neuronal spiking, and compared their performance over a wide range of target output spike trains.

We found using an escape noise neuron model to be ideally suited to the task of reproducing target spike trains by reinforcement, given that a degree of background noise was essential to driving explorative spiking during learning. With regard to the choice of an escape rate, A&C consistently performed better than EXP for an intermediate level of background noise, with the difference in performance between the two models being most apparent for a large number of target output spikes. As shall be elaborated on below, this difference in the performance between the two models was related to the inclusion of a gradient term $\dot{\mu}_i$ for the A&C model. In terms of their convergence speed, A&C converged faster than EXP when learning a smaller number of target spikes, while the reverse was found when learning a greater number of target spikes. Such an effect was attributed to the continuation in learning for large episode numbers with A&C, during which time spurious spiking was further eliminated, while EXP led to a relatively rapid convergence towards a persistently low performance level.

Our main motivation in implementing the A&C model, despite its functional complexity in comparison with the relatively simple EXP model, comes from its inclusion of a gradient term for the postsynaptic membrane potential, $\dot{\mu}_i$. As discussed in section 4.2, both models share similar subthreshold values for the escape rate function $g$ when the gradient term of the A&C model is held fixed at $\dot{\mu}_i = 0$; from this, and following the increased performance of A&C over EXP as measured through simulations, it is clear that the inclusion of a gradient term for a generative model of spikes affords an increase in network accuracy when learning multiple, and precisely timed, output spikes with background noise. Essentially, the gradient term $\dot{\mu}_i$ of the A&C model allows for increased temporal precision of output spikes by driving rapid increases in the neuron’s instantaneous firing density, but without sacrificing variable and spontaneous background spiking activity as is required in order to drive the network’s initial stochastic exploration of reward space. By contrast, the EXP model only depends on the neuron’s momentary membrane potential $u_i$, and not its gradient $\dot{\mu}_i$, so the model can only give rise to spikes with either high temporal precision or high variability, but not
both. For these reasons the A&C model was able to outperform the EXP model, despite both models being subject to comparable levels of background noise.

Our further motivation for applying the A&C model came from its reliance on just a single free parameter, the noise amplitude $\sigma$, which relates to spontaneous fluctuations in the postsynaptic membrane potential arising from background noise. This parameter is more directly measurable from in vivo experiments (Stern et al., 1997), whereas there exists somewhat more ambiguity in the selection of the EXP model noise parameters.

We note that there exist certain conditions under which the performance of EXP might be enhanced: specifically when synaptic weights are unbounded and free to take on any value. Such a scenario is less biologically relevant however, since preliminary simulations demonstrated EXP had to attain synaptic weight values many times larger than that of A&C to perform adequately. Unbounded weights were also found to lead to numerical instabilities, and removed the possibility for any form of robustness to input pattern noise during learning, such as when learning patterns with jittered spike timings.

In our simulations we used a relatively large degree of background noise, mimicking that typically found in vivo (Stern et al., 1997; Jolivet et al., 2006). Although additional results from preliminary simulations indicated similar levels of performance between A&C and EXP as noise was minimised, we also found a trend for an increasing failure rate, following from a reduced ability of the network to stochastically explore reward space when learning target spiking patterns. We therefore surmise that there must exist some optimum level of background noise to maximise the receipt of reward by the network, as has been commented on by Farries & Fairhall (2007). This relates to the exploration-exploitation dilemma as is commonly discussed in the context of reinforcement learning.

### 4.5 Chapter Summary

This chapter has demonstrated that it is possible to train an SNN, using the R-max rule with delayed reinforcement, to accurately learn a large number of target postsynaptic spikes. Furthermore, the A&C model has been indicated as a
strong contender to the more commonly used EXP model for driving postsynaptic firing activity. Finally, although this chapter has been restricted to learning single input-output pattern associations, we have successfully extended simulations to the more complex task of mapping between multiple pairs of input-output spike patterns with input noise, as published in Gardner et al. (2014). For the sake of brevity, we have just considered single pattern learning in this chapter. Taken together, reward-modulated synaptic plasticity has the potential to form input representations by the timings of multiple output spikes, which may well be of biological significance as discussed in section 2.4.
Chapter 5

Supervised Learning for Precise Spiking

This chapter examines supervised learning rules for single-layer SNNs and proposes two new theoretically justified rules: termed INST and FILT. There are two main objectives of this chapter; the first is to provide a more rigorous formulation of both the PSD and SPAN learning rules (see subsection 3.4.4) as the newly formed INST and FILT rules, respectively, that is based on maximising the likelihood of generating desired target output spike trains. The second aim is to draw a comparison between the INST, FILT and Chronotron rules in terms of their performance when learning generic input-output spike pattern transformations, and their associated memory capacities when identifying input patterns using precisely timed output spikes.

5.1 Introduction

The indicated importance of precise spiking as a means to process information has motivated a number of theoretical studies on learning rules for spiking neural networks (reviewed in Kasinski & Ponulak (2006); Güttig (2014)). See chapter 3 for an overview of reinforcement and supervised learning rules in SNNs. Despite this, there still lack supervised learning rules that combine high technical performance with biological plausibility, and yet have a solid theoretical foundation. For
example, the recently proposed Spike Pattern Association Neuron (SPAN) (Mohemmed et al., 2012) and Precise-Spike-Driven (PSD) (Yu et al., 2013) learning rules have both demonstrated success in learning temporally-coded representations of spatio-temporal spike patterns, and even for noisy inputs, but have lacked analytical rigour during their formulation; both of these rules were similarly derived by starting from a heuristic adaptation of the Widrow-Hoff learning rule as applied to rate- rather than spike-based neurons, thereby making it difficult to predict the efficiency of their solutions in general.

Here we present two supervised learning rules, termed INST and FILT, which are initially derived based on the statistical method of Pfister et al. (2006) but later adapted for compatibility with the deterministically spiking LIF neuron model; in this way, these rules are justifiable theoretically and also allow for the learning of precisely timed spikes. The two rules differ in their formulation with respect to the treatment of output spike trains: while INSTantaneous error (INST) simply relies on the instantaneous difference between a target and actual output spike train to inform weight changes, FILTered error (FILT) goes a step further, and convolves output spike trains with an exponential filter to effectively link together neighbouring target and actual output spikes. By this filtering mechanism, we find the FILT rule is able to match the high performance of the E-learning Chronotron (CHRON) rule (see subsection 3.4.3 for a review of CHRON).

This contribution is organised as follows. In section 5.2 the INST and FILT learning rules are formulated based on a LIF neuron model, and compared with the previously discussed SPAN and PSD rules. Next, section 5.3 analyses synaptic weight changes triggered according to the INST and FILT rules; in this section, weight changes are determined as a function of the order in which pre- and postsynaptic spikes are generated, their relative timing differences, and target and actual postsynaptic spikes which are close together in time. Section 5.4 then examines the performance of each learning rule when they are tasked with classifying arbitrarily generated spike patterns. Classifications of input patterns are made based on the precise timing of an output spike, emitted by a single postsynaptic neuron, such that all patterns belonging to the same class are identified by the same target spike. For comparison purposes, results are also obtained for the highly efficient E-learning CHRON rule. Finally, section 5.5 discusses the
presented rules in relation to alternative methods, and includes a commentary on their biological plausibility.

5.2 Learning Theory

This section proposes two supervised learning rules for SNNs, termed INST and FILT, that are initially formulated using the statistical approach of [Pfister et al., 2006] for analytical rigour, but later adapted for use with a deterministically spiking neuron model for the purpose of precise temporal encoding. Following this, the general task of spiking neurons trained to perform transformations between arbitrary input and output spike patterns is specified.

5.2.1 Single Neuron Model

We first consider a single postsynaptic neuron, indexed \(i\), that receives its inputs from a total of \(n_i\) presynaptic neurons indexed by \(j\). If the postsynaptic neuron generates a list of output spikes \(y_i(t)\) in response to a spatio-temporal input pattern \(x = \{x_1, x_2, ..., x_{n_i}\}\), then its membrane potential at time \(t\) is defined by the simplified Spike Response Model (SRM\(_0\)):

\[
u_i(t|x, y_i) := \sum_j w_{ij} \sum_{t_j \in x_j} \epsilon(t - t_j) + \sum_{t_i' \in y_i} \kappa(t - t_i'), \tag{5.1}
\]

where all symbols have their usual meaning (see Eq. (3.31)).

The PSP kernel is given by the difference between two exponential functions:

\[
\epsilon(s) = \epsilon_0 \left[ \exp \left( -\frac{s}{\tau_m} \right) - \exp \left( -\frac{s}{\tau_s} \right) \right] \Theta(s), \tag{5.2}
\]

where \(\epsilon_0 = 4 \text{ mV}\) is a scaling constant, \(\tau_m = 10 \text{ ms}\) the membrane time constant, \(\tau_s = 5 \text{ ms}\) the synaptic rise time and \(\Theta(s)\) the Heaviside step function; for these choice of parameters, a presynaptic spike evokes a PSP with a maximum value of 1 mV after a lag time close to 7 ms. Our choice of a large \(\tau_s\) value (compare with that used in the previous chapter) was motivated by a desire for a longer
rise time of the PSP kernel, the reasons for which shall become clear later on. The reset kernel is given by

\[ \kappa(s) = \kappa_0 \exp \left( -\frac{s}{\tau_m} \right) \Theta(s) , \]  

(5.3)

where the scaling constant is equal to the negative firing threshold \( \kappa_0 = -\vartheta \).

The SRM\(_0\) neuron model can be seen as a generalization of the LIF neuron, which mainly differs in terms of its formulation; specifically, the postsynaptic membrane potential of the SRM\(_0\) is defined more explicitly as a linear sum over historic pre- and postsynaptic spike times, whereas the LIF formulation is given in terms of differential equations. As we shall now show, and following Gerstner & Kistler (2002), the LIF model can be mapped to the SRM\(_0\), thereby motivating our selection of the PSP and reset kernels given above and also to support our subsequent analysis of supervised learning rules for SNNs.

If we consider a postsynaptic neuron with membrane resistance \( R \), driven by a total time-dependent current \( I_i(t) \), the standard form of the LIF model can be used to define the dynamics of its membrane potential:

\[ \tau_m \frac{du_i(t)}{dt} = -u_i(t) + RI_i(t) , \]  

(5.4)

where \( \tau_m \) is the previously stated membrane time constant (see Eq. (2.1)). The driving current term can be split into two components: \( I_i = I_i^\text{syn} + I_i^\text{out} \), where \( I_i^\text{syn} \) is the synaptic current flow into the postsynaptic neuron due to presynaptic spike arrival, and \( I_i^\text{out} \) is a ‘reset’ current pulse which discharges the postsynaptic neuron immediately after it fires. The synaptic current is a weighted summation over individual postsynaptic current pulses:

\[ I_i^\text{syn}(t) = \sum_j w_{ij} \sum_{t'_j \in x_j} \alpha(t - t'_j) , \]  

(5.5)

where the kernel \( \alpha(s) \) is the time course of a received postsynaptic current, and \( w_{ij} \) reflects the total charge transferred to the postsynaptic neuron due to a single presynaptic spike (see Eq. (2.4)). Following Gerstner & Kistler (2002), the reset
current pulse is defined by

\[ I_{\text{out}}^i(t) := -C(\vartheta - u_r) \sum_{t'_j \in y_i} \delta(t - t'_j), \]  

(5.6)

where \( C \) is the neuron’s membrane capacitance, and \( u_r \) is a new value to set the membrane potential to when the neuron fires. Hence, from combining Eqs. (5.4), (5.5) and (5.6) and solving for the membrane potential with the initial condition \( u_i(t_0) = 0 \), it can be shown that:

\[
u_i(t) = \frac{1}{C} \sum_j w_{ij} \sum_{t'_j \in x_j} \int_{t'=0}^t \alpha(t' - t'_j) \exp \left( -\frac{t - t'}{\tau_m} \right) \, dt' - (\vartheta - u_r) \sum_{t'_i \in y_i} \exp \left( -\frac{t - t'_i}{\tau_m} \right) \Theta(t - t'_i) ,
\]

(5.7)

where we have used the relation for the time constant: \( \tau_m = RC \). We approximate the time course of a postsynaptic current pulse by an exponential filter:

\[
\alpha(s) = \frac{q}{\tau_s} \exp \left( -\frac{s}{\tau_s} \right) \Theta(s) ,
\]

(5.8)

where \( q \) is the total charge transferred due to a single presynaptic spike and \( \tau_s \) is a synaptic time constant. Substituting Eq. (5.8) into Eq. (5.7) and then performing the integration yields:

\[
u_i(t) = \frac{q}{C} \frac{\tau_m}{\tau_m - \tau_s} \sum_j w_{ij} \sum_{t'_j \in x_j} \left[ \exp \left( -\frac{t - t'_j}{\tau_m} \right) - \exp \left( -\frac{t - t'_j}{\tau_s} \right) \right] \Theta(t - t'_j) - (\vartheta - u_r) \sum_{t'_i \in y_i} \exp \left( -\frac{t - t'_i}{\tau_m} \right) \Theta(t - t'_i) .
\]

(5.9)

We respectively define the PSP and reset kernels:

\[
\epsilon(s) = \frac{q}{C} \frac{\tau_m}{\tau_m - \tau_s} \left[ \exp \left( -\frac{s}{\tau_m} \right) - \exp \left( -\frac{s}{\tau_s} \right) \right] \Theta(s) ,
\]

(5.10)

\[
\kappa(s) = -(\vartheta - u_r) \exp \left( -\frac{s}{\tau_m} \right) \Theta(s) .
\]

(5.11)
The coefficient in Eq. (5.10) can be compacted into a new constant $\epsilon_0 = \frac{q}{C} \frac{\tau_m}{\tau_m - \tau_s}$ and similarly in Eq. (5.11) we can rewrite $\kappa_0 = -(\vartheta - u_r)$; hence, if we take $q = 2C$ and $u_r = 0$ then we recover the kernels of Eqs. (5.2) and (5.3). For this choice of parameters, Eq. (5.9) becomes an exact match of the originally introduced SRM$_0$ of Eq. (5.1), as desired. Shown in Fig. 5.1 are graphical illustrations of the postsynaptic current, PSP and reset kernels, and an example of a resulting postsynaptic membrane potential as defined by the SRM$_0$. It is apparent from this figure that the reset of the membrane potential is controlled exclusively by the reset kernel, and not by shunting the PSP kernel to zero, as is characteristic of the SRM$_0$. This ensures the PSP kernel is a continuous function, which is a necessary condition in order for the synaptic plasticity rules presented in subsections 5.2.2 and 5.2.3 to perform well.

We next consider the spike generation mechanism of the postsynaptic neuron. A stochastic variant of the LIF neuron model is implemented, such that postsynaptic firing events are generated by a point process with stochastic intensity $\rho_i(t) = g(u_i(t))$, that is the instantaneous firing density of the neuron (see subsection 2.2.3). For analytical tractability, we take an exponential dependence of the firing density on the membrane potential as defined by Eq. (2.8), where the formal firing threshold of the model is set to $\vartheta = 15$ mV.

### 5.2.2 INSTantaneous-error (INST) Synaptic Plasticity Rule

Implementing a stochastic model for generating postsynaptic spikes allows for the determination of the likelihood of generating a desired list of target output spikes $y_i^{\text{ref}} = \{\tilde{t}_i^1, \tilde{t}_i^2, ..., \tilde{t}_i^{n_s}\}$, containing a total of $n_s$ output spikes, for the $i$th postsynaptic neuron, in response to an input pattern $x$ presented over duration $T$. As shown originally by Pfister et al. (2006), and summarised in subsection 3.4.5, gradient ascent on the log-likelihood with respect to synaptic weights can be taken to derive a supervised learning rule, such as that given by Eq. (3.46).

The probabilistic weight update rule of Eq. (3.46) was derived by taking a maximum-likelihood approach using a stochastic spiking neuron model, but can be adapted to the case of a deterministically firing LIF neuron model as considered here. By taking the limit $\Delta u \to 0$ for the stochastic threshold parameter in Eq. (2.8), the
Chapter 5. Supervised Learning for Precise Spiking

Figure 5.1: The kernels used in our analysis, and an example of a postsynaptic membrane potential, $u_i$, defined by the simplified Spike Response Model (SRM$_0$). (A) The time course of the postsynaptic current kernel $\alpha$. (B) The PSP kernel $\epsilon$. (C) The reset kernel $\kappa$. (D) The resulting membrane potential $u_i$. In this example, a single presynaptic spike is received at $t_j = 0$ ms, and a postsynaptic spike is generated at $t_i = 4$ ms from selectively tuning both the synaptic weight $w_{ij}$ and firing threshold $\vartheta$ values. We take $C = 2.5 \text{nF}$ for the capacitance, such that the postsynaptic current attains a maximum value of 1 nA.

The instantaneous firing density can assume one of two values:

$$g(u_i(t)) = \begin{cases} 
\delta(t - t_l^i) & \text{for } u_i(t_l^i) > \vartheta \\
0 & \text{otherwise}
\end{cases}$$

(5.12)

where the term $\delta(t - t_l^i)$ is the Dirac delta distribution about a postsynaptic firing time $t_l^i \in y_i$, since immediately after a spike is emitted: $u(t_l^i+) < \vartheta$ as a result of the reset term in Eq. (5.1). Hence, the firing density can be substituted with the postsynaptic spike train $\rho_i(t) \to \mathcal{Y}_i(t)$ to provide a deterministic adaptation.
of Eq. (3.46):

$$\lim_{\Delta u \to 0} \Delta w_{ij} = \eta \int_0^T [Y_i^{\text{ref}}(t) - Y_i(t)] \sum_{t_f^j \in x_j} \epsilon(t - t_f^j) \, dt,$$

that is now a function of the difference between a target and actual postsynaptic spike train. Finally, the above can be integrated out to provide a batch weight update rule:

$$\Delta w_{ij}^{\text{INST}} = \eta \left[ \sum_{t_{g_i} \in y_i^{\text{ref}}} \sum_{t_f^j \in x_j} \epsilon(\tilde{t}_{g_i} - t_f^j) - \sum_{t_{h_i} \in y_i} \sum_{t_f^j \in x_j} \epsilon(t_{h_i} - t_f^j) \right],$$

which we term the INSTantaneous error (INST) synaptic plasticity rule, to reflect the discontinuous nature of the postsynaptic error signal that is the difference between two spike trains.

It is important to note that the above is closely related to the PSD plasticity rule proposed by Yu et al. (2013) (see Eq. (3.30)) and the I-learning variant of the Chronotron in Florian (2012); weight updates for both PSD and I-learning depend on a presynaptic term combined with an instantaneous, postsynaptic error signal, as for INST, but differ in terms of their functional dependence on presynaptic inputs. Specifically, both PSD and I-learning rely on a current term $\alpha$, defined similarly to that in Eq. (5.8), instead of the above $\epsilon$ term as defined in Eq. (5.10).

The INST rule is analytically more rigorous than both PSD and I-learning given that an optimality criterion was taken as the starting point in its formulation, with the determination that the $\epsilon$ term should act as a presynaptic factor. By contrast, the PSD rule was heuristically derived by Yu et al. (2013) when adapting the Widrow-Hoff learning rule for application in single-layer spiking networks (see subsection 3.4.4). The I-learning rule was initially derived from minimizing the VPD (Victor & Purpura, 1996) with respect to synaptic weights, but the author finally assumed a weighted synaptic current to act as a presynaptic factor.

### 5.2.3 FILTered-error (FILT) Synaptic Plasticity Rule

As it currently stands, the rate of synaptic weight change $\dot{w}_{ij}(t)$ resulting from Eq. (5.13) depends on the instantaneous difference between two spike trains $Y_i^{\text{ref}}$ and $Y_i(t)$.
and \( \mathcal{Y}_i \) during learning. In other words, weight updates are only effected at the precise moments in time when target or actual output spikes are present. Although this leads to the simplified batch update rule of Eq. (5.14), there are two distinct disadvantages to this approach. The first concerns the convergence of actual output spikes towards matching their desired target outputs: if the instantaneous error between two spike trains is communicated to every synapse during learning, then large fluctuations in the changes of synaptic weights will inevitably emerge as an undesired by-product. It then becomes problematic for the network to smoothly converge towards a stable, non-oscillating output spike train while counteracting this source of synaptic noise. Secondly, from a biological standpoint it is implausible that synaptic weights can be effected instantaneously at the precise timings of output spikes. More realistically, it can be supposed that output spikes would leave some form of synaptic trace on the order of the membrane time constant, which might act as a postsynaptic ‘linkage’ variable for coupling together temporally contiguous, or close together, target and actual output spikes.

To counteract this source of synaptic noise resulting from discontinuous postsynaptic signals we convolve the target and actual output spike trains in Eq. (5.13) with an exponential filter, thereby providing the following learning rule:

\[
\Delta w_{ij} = \eta \int_0^\infty \left[ \tilde{\mathcal{Y}}_{\text{ref}}^i(t) - \tilde{\mathcal{Y}}_i(t) \right] \sum_{t_{f} \in x_j} \epsilon(t - t_{f}') \, dt,
\]

(5.15)

where a convolved output spike train is equivalent to

\[
\tilde{\mathcal{Y}}_i(t) \equiv \frac{1}{\tau_q} \int_0^t \mathcal{Y}_i(t') \exp \left( -\frac{t - t'}{\tau_q} \right) \, dt',
\]

(5.16)

and a similar equivalence for \( \tilde{\mathcal{Y}}_{\text{ref}}^i(t) \), with \( \tau_q = 10 \text{ ms} \) a decay time constant. The choice of value for \( \tau_q \) is matched to the membrane time constant \( \tau_m \), and has been indicated to give increased performance from preliminary parameter sweeps. The upper limit of \( \infty \) in Eq. (5.15) is important in order to ensure the filtered target and actual output spike trains are fully integrated over. Performing the integration of Eq. (5.15) using the PSP kernel given by Eq. (5.2) yields the batch
weight update rule:

\[
\Delta w^{\text{FILT}}_{ij} = \epsilon_0 \eta \left[ \sum_{\tilde{t}^g_i \in y_{\text{ref}}} \sum_{t^f_j \in x_j} \exp \left( -\frac{\max\{t^f_j, \tilde{t}^g_i\} - t^f_j}{\tau_q} \right) \left( C_m \exp \left( -\frac{\max\{t^f_j, \tilde{t}^g_i\} - t^f_j}{\tau_m} \right) - C_s \exp \left( -\frac{\max\{t^f_j, t^h_i\} - t^h_i}{\tau_q} \right) \right) - \sum_{\tilde{t}^h_i \in y_i} \sum_{t^f_j \in x_j} \exp \left( -\frac{\max\{t^f_j, \tilde{t}^h_i\} - t^f_j}{\tau_q} \right) \right) \times \left( C_m \exp \left( -\frac{\max\{t^f_j, t^h_i\} - t^f_j}{\tau_m} \right) - C_s \exp \left( -\frac{\max\{t^f_j, t^h_i\} - t^f_j}{\tau_s} \right) \right) \right],
\]

where the membrane and synaptic coefficient terms are \( C_m = \frac{\tau_m}{\tau_m + \tau_q} \) and \( C_s = \frac{\tau_s}{\tau_s + \tau_q} \), respectively. We term the above the FILTered error (FILT) synaptic plasticity rule, that depends on the smoothed difference between filtered target and actual output spike trains.

Eq. (5.17) bears a similarity to the SPAN learning rule (see Eq. (3.28)), in the sense that weight updates depend on convolved input and output spike trains. However, as for the PSD learning rule, SPAN was formulated from adapting the Widrow-Hoff learning rule to single-layer spiking networks and allowed for any arbitrary choice of kernel function with which to convolve input and output spike trains (reviewed in subsection 3.4.4). According to our analysis, input spike trains are optimally convolved with the PSP kernel of Eq. (5.2), although the exponential filtering of postsynaptic spike trains is arbitrary. Selecting an exponential filter simplifies the resulting learning rule however, and coincidentally provides a resemblance of FILT to the van Rossum Distance (vRD) measure as previously introduced by van Rossum (2001). Furthermore, and biologically speaking, filtered postsynaptic traces might originate from backpropagated action potentials which travel towards the neuron’s afferent synapses as a result of postsynaptic spiking.

5.3 Analysis of the Learning Rules

In this section we analyse synaptic weight changes resulting from the INST and FILT learning rules. For ease of analysis we examine just the weight change
between a single pair of pre- and postsynaptic neurons: each emitting a single spike at times $t_j$ and $t_i$, respectively. A single target postsynaptic spike at time $\tilde{t}_i$ is also imposed, which must be matched by the postsynaptic neuron. Detailed below are the resulting weight updates for each rule.

**INST Rule.** According to the definition of the INST rule in Eq. (5.14), the synaptic weight change triggered by a single pre- and postsynaptic spike is given by

$$
\Delta w_{ij}^{\text{INST}} = \eta \left[ \epsilon (\tilde{t}_i - t_j) - \epsilon (t_i - t_j) \right],
$$

that is simply the difference between two PSP kernels. From the above, a weight change is zero when target and actual spikes are aligned, i.e. when $t_i = \tilde{t}_i$, and the left or right PSP terms are equal to zero if their respective arguments are negatively valued.

**FILT Rule.** The FILT batch weight update rule of Eq. (5.17) can be solved for single pre- and postsynaptic spikes:

$$
\Delta w_{ij}^{\text{FILT}} = \epsilon_0 \eta \left[ \exp \left( - \frac{\max\{t_j, \tilde{t}_i\} - \tilde{t}_i}{\tau_q} \right) \left( C_m \exp \left( - \frac{\max\{t_j, \tilde{t}_i\} - t_j}{\tau_m} \right) \right) 
- C_s \exp \left( - \frac{\max\{t_j, t_i\} - t_j}{\tau_s} \right) \right] 
\times \left( C_m \exp \left( - \frac{\max\{t_j, t_i\} - t_j}{\tau_m} \right) 
- C_s \exp \left( - \frac{\max\{t_j, t_i\} - t_j}{\tau_s} \right) \right) \right],
$$

which further simplifies to

$$
\Delta w_{ij}^{\text{FILT}} = \epsilon_0 \eta \left[ C_m \exp \left( - \frac{\tilde{t}_i - t_j}{\tau_m} \right) 
- C_s \exp \left( - \frac{\tilde{t}_i - t_j}{\tau_s} \right) \right] 
- \left( C_m \exp \left( - \frac{t_i - t_j}{\tau_m} \right) 
- C_s \exp \left( - \frac{t_i - t_j}{\tau_s} \right) \right),
$$

when assuming all postsynaptic spikes follow the presynaptic spike: $\tilde{t}_i, t_i > t_j$. From the above, it can be found that weight changes are zero when target and
actual postsynaptic spikes are aligned; however, unlike the INST rule, a negative timing of a postsynaptic spike relative to a presynaptic spike can still elicit a change in the synaptic weight.

Eq. (5.18) or (5.20) influences the placement of an actual postsynaptic spike by driving an increase or decrease in a postsynaptic neuron’s membrane potential close to its firing threshold, via synaptic weight modification. For our choice of PSP function, an increase in the synaptic weight works to shift an actual spike backwards in time, and a decrease in the synaptic weight shifts an actual spike forwards in time. Hence, by this process, the aim of a trained neuron is to find an optimal synaptic weight value which minimises the temporal difference of an actual output spike with respect to its target.

In the rest of this section, we start by simply examining the synaptic weight change as a function of the order in which postsynaptic spikes occur, as well its dependence on individual spikes. Next, we explore the synaptic weight change as a function of the relative timing difference between a target postsynaptic spike and input presynaptic spike, and either in the absence or presence of an actual postsynaptic spike, to establish the temporal window of each synaptic plasticity rule. Following this, we examine in detail the effect of temporally contiguous postsynaptic spikes on synaptic plasticity, and indicate the importance of the PSP in influencing the direction of synaptic weight changes. Finally, we analyse the variance of synaptic weight changes in response to fluctuating postsynaptic responses. For demonstrative purposes the learning rate of the INST and FILT rule is set to unity here, although there is no qualitative change in the results for different values.

5.3.1 Order of Postsynaptic Spikes

The panels in Fig. 5.2 illustrate the change in the synaptic weight under INST and FILT based on the presence or order of postsynaptic spiking for values $\tilde{t}_i, t_i > t_j$:

(A) only an existing target spike triggers potentiation, such that the future emission of a postsynaptic spike is encouraged,

(B) only an existing actual spike triggers depression, which acts to suppress future postsynaptic spiking,
an actual spike following its target by 5 ms, that is generated at 20 ms after stimulus onset, triggers potentiation; this acts to shift a future postsynaptic spike backwards in time towards its target.

an actual spike preceding its target by 5 ms, that is generated at 15 ms after stimulus onset, triggers depression; this acts to shift a future postsynaptic spike forwards in time towards its target.

Additionally, the third subplot in each panel shows the time course of the FILT rule’s error signal, that is equal to the difference between the filtered target and actual postsynaptic spike trains (the first integrand term in Eq. (5.15)).

From this figure, it is clear that the direction of synaptic weight changes are the same for both learning rules, but differ in terms of their magnitude: in all cases, weight changes triggered by the FILT rule are weaker. The reason for this becomes apparent when taking into account the shape of the FILT error signal, which is multiplicatively combined with the evoked PSP at each point in time to smoothly drive synaptic weight changes, rather than the rapid changes triggered by the INST rule. It is highlighted here that the FILT rule is implementable as an online learning method, based on the time derivative of Eq. (5.15), although the batch weight update rule of Eq. (5.19) turns out to be computationally more efficient.

At this point it is necessary to discuss the relationship between the timing of an actual output spike fired by a postsynaptic neuron and the shape of a PSP evoked by an input spike. By itself, the synapse of Fig. 5.2 would be incapable of allowing the postsynaptic neuron to precisely fire at its desired target timing since the target coincides with the falling segment of the PSP curve; effectively, the postsynaptic neuron can only fire a single output spike with a lag time up to the peak value of the PSP kernel, since this is the only region over which the neuron’s membrane potential can be adjusted to cross its firing threshold from below. Despite this, the synapse is well capable of acting in concert with other synapses that do coincide with the neuron’s target timing, which is ideal for distributing the synaptic load of a network during learning.
Chapter 5. Supervised Learning for Precise Spiking

Figure 5.2: Illustration of INST and FILT synaptic plasticity rules. For each subplot within a panel such as (A), from top to bottom: the first subplot shows the order of existing target and actual output spikes at the postsynaptic neuron, second is the PSP generated due to a single presynaptic spike at $t_j = 0$ ms, third is the filtered postsynaptic error signal for the FILT rule, and the final panel shows the time course of iterative weight updates for each rule. In each separate panel, either one or both of the target and actual output spikes at the postsynaptic neuron are considered. See the main text for a detailed description of each panel. The design of this figure is inspired from Florian (2012).
5.3.2 Relative Timing between Spikes

Shown in Fig. 5.3 is the synaptic weight change for each learning rule as a function of the relative timing between a target postsynaptic spike and a presynaptic spike, denoted $t_{\text{ref}} - t_{\text{pre}}$, including for negative relative timings. The top panels correspond to the absence of an actual postsynaptic spike, and the lower panels correspond to the presence of an actual postsynaptic spike; in this example, the actual spike is held at a fixed positive timing of 20 ms with respect to the presynaptic spike, and is denoted by $t_{\text{post}} - t_{\text{pre}}$.

From the top panel of Fig. 5.3A for the INST rule, it is observed that the plot of the synaptic change simply follows the form of a PSP kernel. In this case, the synaptic change is zero for negative values of the relative timing, demonstrating the causality of a presynaptic spike in eliciting a desired postsynaptic spike. Interestingly, the top panel of Fig. 5.3B for the FILT rule instead demonstrates a more symmetrical dependence of synaptic change on the relative timing difference, in the absence of an actual spike, which is centred just right of the origin. This contrasts with the INST rule, and can be explained by the FILT rule instead working to minimise the filtered difference between an actual and target spike train, rather than just their instantaneous difference; in other words, even if an actual postsynaptic spike cannot technically be aligned with its target, then a close match is deemed to be sufficient under FILT.

Each lower panel of Fig. 5.3 effectively shifts the plot in their respective upper panel downwards, and relates to the effect of synaptic depression triggered by the presence of an actual postsynaptic spike. In each lower panel it is worth noting the change in the synaptic strength about $t_{\text{post}} - t_{\text{pre}}$, where there is a cross-over point from positive to negative values: this region has the effect of shifting actual spikes generated on successive trials towards their target timings, as was discussed previously in relation to Fig. 5.2. It is also noted that the magnitude of synaptic change is reduced for FILT in comparison with INST, for the same reasons as discussed previously.
Figure 5.3: Dependence of synaptic weight change $\Delta w_{ij}$ on the relative timing between a target postsynaptic spike $t_{\text{ref}}$ and an input presynaptic spike $t_{\text{pre}}$. Columns (A) and (B) correspond to the INST and FILT learning rules, respectively. Both columns: The top panel shows the synaptic change as a function of the relative timing, but in the absence of an actual postsynaptic spike. The bottom panel is for an actual postsynaptic spike $t_{\text{post}}$ which follows the presynaptic spike by 20 ms. The downwards shift in the synaptic change reflects depression triggered by the presence of an actual postsynaptic spike.

5.3.3 Temporally Contiguous Postsynaptic Spikes

It is important to discuss weight changes resulting from target and actual postsynaptic spikes that are close together in time, and in particular with respect to the shape of an evoked PSP. To this end, we consider a postsynaptic neuron emitting an actual spike at a time $t_i = \tilde{t}_i + \Delta t_i$ in response to a single presynaptic spike at time $t_j$, where $\Delta t_i$ is a time shift relative to the neuron’s target timing of $\tilde{t}_i$. We also assume the conditions $\tilde{t}_i > t_j$ and $\tilde{t}_i + \Delta t_i > t_j$, such that a postsynaptic spike always occurs after a presynaptic spike. The resulting weight change for each learning rule is calculated and discussed below.
\[ \Delta w_{ij}^{\text{INST}} = \eta \left[ \epsilon \left( \hat{t}_i - t_j \right) - \epsilon \left( \hat{t}_i + \Delta t_i - t_j \right) \right]. \quad (5.21) \]

If the PSP kernel of Eq. (5.2) is substituted into the above, the INST rule can be more explicitly expressed:

\[
\Delta w_{ij}^{\text{INST}} = \epsilon_0 \eta \left\{ \exp \left( - \frac{\hat{t}_i - t_j}{\tau_m} \right) \left( 1 - \exp \left( - \frac{\Delta t_i}{\tau_m} \right) \right) - \exp \left( - \frac{\hat{t}_i - t_i}{\tau_s} \right) \left( 1 - \exp \left( - \frac{\Delta t_i}{\tau_s} \right) \right) \right\}. \quad (5.22)
\]

Next, by considering small time shifts \( \Delta t_i \ll \tau_m, \tau_s \) such that the actual output spike is close to its target, then the above can be simplified to give the final weight update rule:

\[
\Delta w_{ij}^{\text{INST}} = \epsilon_0 \eta \Delta t_i \left[ \frac{1}{\tau_m} \exp \left( - \frac{\hat{t}_i - t_j}{\tau_m} \right) - \frac{1}{\tau_s} \exp \left( - \frac{\hat{t}_i - t_j}{\tau_s} \right) \right]. \quad (5.23)
\]

Interestingly, we find that small lag times \( \hat{t}_i - t_j < s^{\text{peak}} \) depress a synapse for positive time shifts \( (\Delta t_i > 0) \) and potentiate for negative time shifts \( (\Delta t_i < 0) \), where \( s^{\text{peak}} = \frac{\tau_m \tau_s}{\tau_m - \tau_s} \log \left( \frac{\tau_m}{\tau_s} \right) \) is the lag time at which point the PSP kernel assumes its maximum value; in effect, a postsynaptic spike which initially follows its target will be driven to fire even later on successive trials, and a postsynaptic spike which initially precedes its target will be driven to fire even earlier on successive trials. Clearly this is undesirable when the objective is to train a postsynaptic neuron to precisely match a target timing, and is explained by the absence of a distinct treatment for coupling together temporally contiguous target and actual postsynaptic spikes. By contrast, increased lag times \( \hat{t}_i - t_j > s^{\text{peak}} \) potentiate a synapse for positive time shifts \( (\Delta t_i > 0) \) and depress a synapse for negative time shifts \( (\Delta t_i < 0) \), as is desired.

Taken together, this analysis of the INST rule demonstrates erroneous synaptic changes over the rising segment of the PSP curve, but correct synaptic changes over the falling segment of the PSP curve as was examined earlier (see Fig. 5.2).
The lag time at which point the PSP assumes its maximum value is $s_{\text{peak}} \approx 7 \text{ ms}$ when using our parameter choice given in subsection 5.2.1 and is visualised in Fig. 5.1B.

**FILT Rule.** With respect to the FILT rule, weight updates due to Eq. (5.20) can instead be rewritten in terms of shifted postsynaptic spikes:

$$
\Delta w_{ij}^{\text{FILT}} = \epsilon_0 \eta \left[ \frac{1}{\tau_m + \tau_q} \exp \left( - \frac{\tilde{t}_i - t_j}{\tau_m} \right) \left( 1 - \exp \left( - \frac{\Delta t_i}{\tau_m} \right) \right) 
- \frac{1}{\tau_s + \tau_q} \exp \left( - \frac{\tilde{t}_i - t_j}{\tau_s} \right) \left( 1 - \exp \left( - \frac{\Delta t_i}{\tau_s} \right) \right) \right],
$$

(5.24)

where we have written in full the $C_m$ and $C_s$ coefficient terms. It is confirmed that when $\Delta t_i = 0$, i.e. when an actual postsynaptic spike is aligned with its target, then the synaptic weight change is zero. Also, in taking the limit $\Delta t_i \to \infty$ only potentiation in the synaptic weight results, since effectively only a target postsynaptic spike is coincident with the presynaptic spike.

We next consider small time shifts $\Delta t_i < \tau_m, \tau_s$, such that the above can be expanded and then factorised to give the final weight update rule:

$$
\Delta w_{ij}^{\text{FILT}} = \epsilon_0 \eta \Delta t_i \left[ \frac{1}{\tau_m + \tau_q} \exp \left( - \frac{\tilde{t}_i - t_j}{\tau_m} \right) - \frac{1}{\tau_s + \tau_q} \exp \left( - \frac{\tilde{t}_i - t_j}{\tau_s} \right) \right].
$$

(5.25)

From the above, we find that small lag times $\tilde{t}_i - t_j < s_{\text{switch}}$ depress a synapse for positive time shifts ($\Delta t_i > 0$) and potentiate for negative time shifts ($\Delta t_i < 0$), where $s_{\text{switch}} = \frac{\tau_m \tau_s}{\tau_m - \tau_s} \log \left( \frac{\tau_m + \tau_q}{\tau_s + \tau_q} \right)$ is the lag time at which point the direction of a synaptic weight change is reversed; in this way, the behaviour of synaptic weight changes about $s_{\text{switch}}$ is similar to that of $s_{\text{peak}}$ for the INST rule. The lag time $s_{\text{switch}}$ has a functional dependence on the filter time constant $\tau_q$, such that $\tau_q \in [0, \infty)$ is mapped to a latency of $s_{\text{switch}} \in [s_{\text{peak}}, 0)$ as illustrated in Fig. 5.4 for $\tau_q \leq 40 \text{ ms}$. As discussed previously, it is desirable that synapses are potentiated for actual postsynaptic spikes following their targets, and are depressed otherwise; hence, decreasing $s_{\text{switch}}$ with respect to its parameter $\tau_q$ should predictably lead to increased temporal precision of the FILT rule.

101
Figure 5.4: The lag time $s^{\text{switch}}$, at which point the direction of synaptic weight change is reversed, plotted as a function of the filter time constant $\tau_q$. In this figure, the values of $\tau_m$ and $\tau_s$ used to determine $s^{\text{switch}}$ are 10 ms and 5 ms, respectively. At $\tau_q = 0$ ms the lag time $s^{\text{switch}}$ is equivalent to $s^{\text{peak}}$, that is the lag time corresponding to the maximum value of a PSP kernel. As a reference, the value $\tau_q = 10$ ms was selected for use in our computer simulations, which was indicated to give optimal performance on preliminary runs.

5.3.4 Synaptic Variance

Here we extend upon our analysis in the preceding subsection to considering the variance of synaptic weight changes in response to variable postsynaptic spiking. As before, we just focus on a single pair of pre- and postsynaptic neurons: each emitting a single spike at times $t_j$ and $\tilde{t}_i + \Delta t_i$, respectively, such that the actual output spike is shifted relative to its target timing $\tilde{t}_i$ by $\Delta t_i$. In all cases we assume the conditions $\tilde{t}_i > t_j$ and $\tilde{t}_i + \Delta t_i > t_j$, so that postsynaptic spikes always occur after presynaptic spikes. The objective in this case would be to minimise the variance of weight changes to increase the resilience of the network to background noise.

**INST Rule.** We analyse the dynamics of $\Delta w_{ij}$ due to fluctuating postsynaptic responses. To this end, we substitute the shift in the postsynaptic spike time with a random variable that is distributed according to a normal distribution $\Delta t_i \sim \mathcal{N}(0, \sigma^2_{\Delta t})$ with zero mean and variance $\sigma^2_{\Delta t}$. Hence, the variance of synaptic
weight change for the INST learning rule given by Eq. (5.23) is determined as

\[ \text{Var}(\Delta w_{ij}^{\text{INST}}) = \epsilon_0^2 \eta^2 \sigma^2_{\Delta t} \left[ \frac{1}{\tau_m} \exp \left( -\frac{\tilde{t}_i - t_j}{\tau_m} \right) - \frac{1}{\tau_s} \exp \left( -\frac{\tilde{t}_i - t_j}{\tau_s} \right) \right]^2. \]  

(5.26)

The noise parameter \( \sigma^2_{\Delta t} \) might arise from a large-valued learning rate, or from an extrinsic noise source such as background stochastic spike arrival that would effectively give rise to a noisy firing threshold. A strategy to minimise the variance of weight change might be to ‘smooth out’ the PSP kernel by increasing the time constants \( \tau_m \) and \( \tau_s \), although this has the drawback of weakening associations that are formed between coincident pre- and postsynaptic spikes during learning; in other words, a greater fraction of synapses would be recruited on any given learning trial, thereby impacting negatively on the networks ability to form unique input representations. For this reason, it would seem appropriate to find an alternative approach to reducing the variance of synaptic weight change.

**FILT Rule.** With respect to the dynamics of synaptic weight changes under the FILT rule, the variance is given by

\[ \text{Var}(\Delta w_{ij}^{\text{FILT}}) = \epsilon_0^2 \eta^2 \sigma^2_{\Delta t} \left[ \frac{1}{\tau_m + \tau_q} \exp \left( -\frac{\tilde{t}_i - t_j}{\tau_m + \tau_q} \right) - \frac{1}{\tau_s + \tau_q} \exp \left( -\frac{\tilde{t}_i - t_j}{\tau_s + \tau_q} \right) \right]^2, \]

(5.27)

where as for the INST rule the time shift is treated as a normally distributed random variable \( \Delta t_i \sim \mathcal{N}(0, \sigma^2_{\Delta t}) \) with zero mean and a variance of \( \sigma^2_{\Delta t} \). Interestingly, the synaptic variance has an inverse relationship with the filter time constant \( \tau_q \), suggesting that for increasing values of \( \tau_q \) the variance in response to fluctuating postsynaptic spiking is reduced. This comes as an added benefit of the FILT rule which, as was shown in the previous subsection, is also capable of precisely ‘linking together’ temporally contiguous target and actual output spikes.

### 5.3.5 Summary

To summarise, this section has analysed the dynamics of synaptic weight modifications determined by the INST and FILT rules, based on the order, relative
timing and temporal precision of single target and actual postsynaptic spikes. These rules are analytically rigorous and have been predicted to give rise to desirable synaptic changes under most conditions, and in particular for the FILT rule, while at the same time avoiding any distinct treatment concerning the dependence of the neuron’s state on its presynaptic input about its firing threshold. Previous examples highlighting the challenges faced in determining the change in a neuron’s state about its firing threshold are found in [Bohte, Kok, & Pouthé (2002); Florian (2012)] (both reviewed in section 3.4), where typically a linear functional dependence is assumed, but which restricts the learning rate to small values and adds to their computational complexity. By contrast, simply convolving postsynaptic spikes with an exponential filter turns out to be sufficient in ensuring convergence towards a stable solution.

5.4 Simulation Results

This section presents results from computer simulations testing the performance of the INST, FILT and CHRON learning rules. The E-learning variant of the CHRON rule, defined by Eq. (3.22), is used in our simulations, and represents an ideal benchmark against which our derived rules can be compared; CHRON is ideal since it incorporates a mechanism for linking together target and actual postsynaptic spikes, analogous to the proposed FILT rule, as well as allowing for a very high network capacity in terms of the maximum number of input patterns it can learn to memorise (Florian 2012).

5.4.1 Network Setup

In all simulations, the network consisted of a single postsynaptic neuron receiving input spikes from a variable number \( n_i \) of presynaptic neurons. The dynamics of the postsynaptic neuron’s membrane potential \( u_i \) was governed according to the SRM\(_0\) model of Eq. (5.1), and output spikes were instantly generated when the neuron’s membrane potential reached the formal firing threshold \( \vartheta \); hence, we implemented a deterministic adaptation of the stochastic neuron model presented in Eq. (2.8), as necessitated by our derived learning rules. The internal simulation time step was taken as \( \delta t = 0.1 \text{ ms} \) for temporal precision.
Chapter 5. *Supervised Learning for Precise Spiking*

The synaptic weight between each presynaptic neuron \( j \) and the postsynaptic neuron \( i \) was initialised randomly at the start of every simulation run, with \( w_{ij} \) values uniformly distributed between 0 and \( 200/n_i \) for a total number of presynaptic neurons \( n_i \); as a result, the initial firing rate of the postsynaptic neuron was driven to \( \sim 1 \) Hz.

Input patterns were conveyed to the network by the collective firing activity of presynaptic neurons, where a pattern consisted of a single, uniformly distributed spike at each neuron; the choice of single rather than multiple input spikes to form pattern representations proved to be more amenable to the subsequent analysis of gathered results. Unless otherwise stated, an arbitrary realisation of each pattern was used at the start of each simulation run, which was then held fixed thereafter. By this method, a total number \( p \) of unique patterns were generated. Patterns were generated with a duration \( T = 200 \) ms, that is approximately the time-scale of sensory processing in the nervous system.

### 5.4.2 General Learning Task

The postsynaptic neuron was trained to reproduce an arbitrary target output spike train in response to each of the \( p \) input patterns through synaptic weight modifications in the network, using either the INST, FILT or CHRON learning rules. In this way, the network learned to perform precise temporal encoding of input patterns. During training, all \( p \) input patterns were sequentially presented to the network in batches, where the completion of a batch corresponded to one epoch of learning. Resulting synaptic weight changes computed for each of the individually presented input patterns (or each trial) were accumulated, and applied at the end of an epoch.

The learning rate used for each of the rules was by default \( \eta = 600/(n_i \ n_s \ p) \), where \( n_s \) was the number of target output spikes; any exceptions to this are specified in the main text. As shall be shown in our simulation results, it was indicated that the learning rules shared a common, optimal value for the learning rate, thereby allowing less biased comparisons to be made between them in terms of their convergence speed.
5.4.3 Performing a Single Input-Output Mapping

For demonstrative purposes, we first applied the INST and FILT learning rules to training the network to perform a mapping between a single, fixed input spike pattern and a target output spike train containing four spikes. The network contained 200 presynaptic neurons, and the target output spikes were equally spaced out with timings: 40, 80, 120 and 160 ms. Simulations for the learning rule were run over 200 epochs, where each epoch corresponded to the repeated presentation of the pattern. Hence, a single simulation run represented a total 40 s of biological time.

Illustration of the Learning Task. Fig. 5.5 illustrates this learning task, where panel A is a spike raster of an arbitrarily generated input pattern. In this example, two postsynaptic neurons learned to transform the input pattern into the target output spike train through synaptic weight modifications as determined by either the INST or FILT rule. From the actual output spike rasters depicted in panel B, it can be seen that both postsynaptic neurons learned to rapidly match their target responses during learning. Despite this, persistent fluctuations in the timings of postsynaptic spikes were associated with just the INST rule, while the FILT displayed stability over the remaining epochs. This disparity can be attributed to the lack of a distinct mechanism for coupling together temporally contiguous target and actual output spikes for the the INST rule, as was discussed in section 5.3.3. Finally, panel C shows the accuracy for each learning rule, given as the average vRD plotted as a function of the number of learning epochs. With respect to the INST rule, it can be seen the vRD failed to reach zero and was subject to a high degree of variance, as reflected by the corresponding spike raster in panel B; its final, convergent vRD value was $0.2 \pm 0.2$, that is an output spike timing error of around 1 ms with respect to its target. By contrast, the FILT rule’s vRD value rapidly approached zero, and was subject to much less variation during the entire course of learning.

Synaptic Weight Distributions. Shown in Fig. 5.6 are the distributions of synaptic weights before and after network training for the INST and FILT learning rules, corresponding to the same simulation of Fig. 5.5. In plotting Fig. 5.6
Figure 5.5: Two postsynaptic neurons trained under the synaptic plasticity rules, that learned to map between a single, fixed input spike pattern and a four-spike target output train. (A) A spike raster of a typical input pattern, lasting 200 ms. (B) Actual output spike rasters corresponding to both the INST rule (left) and the FILT rule (right) in response to the repeated presentation of the input pattern. Target output spike times are indicated by crosses. (C) The evolution the vRD for each learning rule, taken as a moving average over 40 independent runs. The shaded regions show the standard deviation.
Figure 5.6: Averaged synaptic weight values before and after learning, for a network trained under the INST or FILT learning rule with the same setup as in Fig. 5.5. The input synaptic weight values are plotted in chronological order with respect to their associated time of firing. The top panel is the distribution of weights before learning, the middle panel corresponds to post training under the INST rule, and the bottom panel the FILT rule. The gold coloured vertical lines indicate the target postsynaptic firing times. Results were averaged based on 40 independent runs. The design of this figure is inspired from Mohemmed et al. (2012).
synaptic weights were sorted in chronological order with respect to their associated time of presynaptic firing: for example, the height of a bar at 40 ms reflects the average value of a synaptic weight from a presynaptic neuron which contributed its spike at 40 ms. The gold overlaid lines correspond to the previously defined target output spike timings: 40, 80, 120 and 160 ms.

From this figure, the upper panel illustrates the uniform distribution of synaptic weights used to initialise the network, before any learning took place, which had the effect of driving the initial postsynaptic firing rate to \( \sim 1 \text{ Hz} \). The middle and lower panels show the distribution of synaptic weights at the end of learning when the INST and FILT rules were respectively applied. From these two panels, a rapid increase in the synaptic weight values preceding target spikes can be seen, which then proceeded to fall off. Comparatively, the magnitude of weight change was largest for the INST rule: with peak values over three times that produced by FILT. Furthermore, only the INST rule resulted in negatively-valued weights, which is especially noticeable for weights associated with input spikes immediately following the target output spike timings. In effect, these sharp depressions offset the relatively strong input drive received just before the target output timings, which is indicative of the unstable nature of the INST learning rule. By contrast, the FILT rule led to a ‘smoother landscape’ of synaptic weight values, following a periodic pattern when plotted chronologically.

### 5.4.4 Impact of the Learning Rate

In this experiment we explored the dependence of each rule’s performance on the learning rate parameter \( \eta \) in terms of the spike-timing accuracy of a trained postsynaptic neuron, measured using the vRD. The primary objective was to establish the relative sensitivity of the rules to large values of \( \eta \), and secondly to establish a value of \( \eta \) which provided a suitable trade-off between learning speed and final convergent accuracy. Here we first include the E-learning variant of the CHRON rule proposed by [Florian (2012)](https://example.com), to provide a benchmark for the INST and FILT rules. With respect to the experimental setup, the network consisted of 200 presynaptic neurons and was tasked with learning to map a total of 10 different input patterns to the same, single target output spike with a timing of 100 ms. In this case learning took place over 500 epochs.
Figure 5.7: The vRD as a function of the learning rate $\eta$ for each learning rule. The E-learning variant of the Chronotron (CHRON) rule of Florian (2012) is included as a benchmark for the INST and FILT rules. In every instance, a network containing 200 presynaptic neurons was tasked with mapping 10 arbitrary input patterns to the same target output spike with a timing of 100 ms. Learning took place over 500 epochs, and results were averaged over 40 independent runs. In this case, error bars show the standard error of the mean rather than the standard deviation: the vRD was subject to very high variance for large $\eta$ values, therefore we considered just its average value and not its distribution.

As shown in Fig. 5.7, it is clear that the INST rule was most sensitive to changes in the learning rate, with an average vRD value $2.5 \times$ that of FILT for the largest learning rate value $\eta = 1$. The least sensitive rule turned out to be CHRON, which still managed to maintain an average vRD value close to zero when plotted up to the maximum value of $\eta$. Interestingly, all three distance plots displayed the same general trend over the entire range of learning rates considered: there was a rapid decrease for small $\eta$ values, followed by a plateau up to around $\eta = 0.5$, and then a noticeable increase towards the end. The large distance values for small $\eta$ related to a lack of convergence in learning by the trained neuron.

To summarise, these results support our choice of an identical learning rate for all three learning rules as used in the subsequent learning tasks of this subsection. Additional, more exhaustive parameter sweeps in preliminary simulations indicated that the learning rates for all three learning rules shared the same inverse proportionality with the number of presynaptic neurons, patterns and target output spikes. This corresponded to an optimal value of $\eta = 0.3 \pm 0.1$ in Fig. 5.7.
5.4.5 Classifying Spike Patterns

An important characteristic of a neural network is the maximum number of patterns it can reliably memorise, as well the time taken to train it. Therefore, we tested the performance of an SNN on a generic classification task, where input patterns belonging to different classes were identified by the precise timings of individual postsynaptic spikes. Here, we first determine the performance of a network when trained to identify separate classes of input patterns based on the precise timing of a single postsynaptic spike (a latency code), then later consider identifications based on multiple postsynaptic spike timings (a fully temporal code). In all cases, an SNN was trained under the INST, FILT or CHRON learning rule for comparison purposes.

An SNN was tasked with learning to classify arbitrarily generated input patterns into five separate classes through hetero-association; an equal number of input patterns were randomly assigned to each class, and all inputs belonging to the same class were identified by a shared, target postsynaptic spike time. Hence, an input pattern was considered to be correctly identified if the network responded with just a single output spike within $\Delta t$ of its desired target timing. The value of $\Delta t$ was varied depending on the level of temporal precision required, with values in the range of $\Delta t \in (0, 5]$ ms that correspond to the typical level of spike timing precision observed in the brain (Reich et al., 1997). The network performance was reported as the percentage of patterns correctly classified on an epoch of training, and was taken as an exponentially-weighted moving average with an averaging window of 20 epochs to take into account variable postsynaptic spiking during the learning process. For each input class a target postsynaptic spike time was randomly generated according to a uniform distribution, that ranged in value between 40 and 200 ms; the lower bound of 40 ms was enforced, given previous evidence indicating that smaller values are harder to reproduce by an SNN (Florian, 2012; Mohemmed et al., 2012). To ensure input classes were uniquely identified, target output timings were distanced from each other by a vRD of at least 0.5, corresponding to a minimum timing separation of 7 ms.

Performance of the Learning Rules. Shown in the left column of Fig. 5.8 is the performance of a network containing either 200, 400 or 600 presynaptic...
neurons, as a function of the number of input patterns to be classified. In this case, we took \( \Delta t = 1 \text{ ms} \) as the required timing precision of a postsynaptic spike with respect to its target, for each input class. As a means to determine the maximum number of patterns memorisable by a network, we took a performance level of at least 90\% as our cut-off point when deciding whether all of the input patterns were classified with sufficient reliability; this criterion was also used to determine the number of epochs taken by the network to learn all the inputs, and is plotted in the right column of this figure. Epoch values not plotted for increased numbers of input patterns reflected an inability of the network to learn every input within 500 epochs.

As expected, Fig. 5.8 demonstrates a decrease in the classification performance as the number of input patterns presented to the network was increased, with a clear dependence on the number of presynaptic neurons contained in the network. For example, a network trained under INST was able to classify 15, 30 and 40 patterns at a 90\% performance level when containing 200, 400 and 600 presynaptic neurons, respectively. The number of input patterns memorisable by a network can be characterised by defining a load factor \( \alpha = p/n_i \), where \( p \) is the number of patterns memorised by a network containing \( n_i \) presynaptic neurons (G"utig & Sompolinsky, 2006). Furthermore, the maximum number of patterns memorisable by a network is given by its memory capacity \( \alpha_m = p_m/n_i \), where \( p_m \) is the maximum number of patterns memorised using \( n_i \) synapses. Hence, by taking 90\% as the cut-off point for reliable pattern classifications, we found the INST rule had an associated memory capacity \( \alpha_m = 0.07 \pm 0.01 \). By comparison, the memory capacities for the FILT and CHRON rules were 0.14 \pm 0.01 and 0.15 \pm 0.01 respectively, being around twice the capacity of that determined for INST. Beyond these increased memory capacity values, networks trained under FILT or CHRON were capable of performance levels very close to 100\% when learning a relatively small number of input patterns; by contrast, the maximum performance level attainable under INST was just over 95\%, and was subject to a relatively large variance of around 5\%. Finally, it is evident from this figure that both FILT and CHRON shared roughly the same performance levels over the entire range of input patterns and network structures considered. In terms of the time taken to train the network, both FILT and CHRON were equally fast, while INST was typically slower than the other rules by a factor of between three
Figure 5.8: The performance of each learning rule as a function of the number of input patterns when learning to classify $p$ patterns into $c = 5$ separate classes. Each input class was identified by a single, unique target output spike, which the postsynaptic neuron had to learn to match to within 1 ms. Left: The classification performance for $n_i = 200, 400$ and 600 input neurons. Right: The corresponding number of epochs taken to reach a performance level of 90%. More than 500 epochs was considered a failure to learn all the inputs at the desired performance level. Results were averaged over 20 independent runs.
and four. This difference in the training time became more pronounced as both the number of input patterns and presynaptic neurons were increased.

**Memory Capacity.** We now explore in more detail the memory capacity $\alpha_m$ supported under each learning rule, specifically with respect to its dependence on the timing precision $\Delta t$ used to identify input patterns. In determining the memory capacity as a function of the timing precision, we used the same experimental setup as considered previously for $\Delta t = 1$ ms, but extended to also consider values of $\Delta t$ between 0.2 and 5 ms (equally spaced in increments of 0.2 ms). As before, we assumed the maximum number of patterns memorisable by the network as those that were classified with a performance level of at least 90% within 500 epochs.

From Fig. 5.9 it can be seen that the memory capacity associated with each learning rule increased with the value of the timing precision, which eventually levelled off for values $\Delta t > 3$ ms. It is also clear that the trend for the FILT rule is consistent with that of the CHRON rule over the entire range of timing precisions considered, while the INST rule gave rise to the lowest memory capacities. For values $\Delta t < 2$ ms the difference in memory capacity between INST and FILT was most pronounced, to the extent that INST was incapable of memorising any input patterns for $\Delta t < 0.8$ ms. By contrast, FILT still maintained a memory capacity close to 0.07 when classifying patterns based on ultra-precise spike timings of within 0.2 ms. As a validation of our method, we note that our measured memory capacity for CHRON at a timing precision of 1 ms is in close agreement with that determined originally in Fig. 9A of [Florian (2012)]: with a value close to 0.15 after 500 epochs of network training.

**Multiple Target Spikes.** This subsection finally examines the performance of the learning rules when input patterns are identified by the precise timings of multiple postsynaptic spikes. In this case, the network contained 200 presynaptic neurons and was trained to classify a total of 10 input patterns into five separate classes, with two patterns belonging to each class. Both patterns belonging to a class were identified by the same target output spike train; hence, a correct pattern classification was considered when the number of actual output spikes fired by the postsynaptic neuron matched the number of target output spikes, and
Figure 5.9: The memory capacity $\alpha_m = p_m / n_i$, that is the maximum number of memorised patterns $p_m$ over the number of synapses $n_i$ in a network trained under either the INST, FILT or CHRON rule, as a function of the required timing precision $\Delta t$ of a single postsynaptic spike. This experiment corresponds to a network learning to classify input patterns into five classes using a latency code, within a maximum of 500 epochs. Memory capacity values were determined based on networks containing $n_i = 200, 400$ and $600$ input neurons. Results were averaged over 20 independent runs.

every actual spike fell within $\Delta t$ of its respective target. For each input class, target output spikes were randomly generated according to a uniform distribution bound between 40 and 200 ms, as used previously. To ensure input classes were uniquely represented, generated target output spike trains were distanced from one another by a vRD of at least $n_s/2$, where $n_s$ was the number of spikes contained in a target train.

Shown in Fig. 5.10 is the performance of the network trained under each learning rule when classifying input patterns based on the precise timings of between one and five target output spikes, with a timing precision $\Delta t = 1$ ms. Because the learning rate was inversely proportional to the number of target spikes, we extended the maximum number of epochs to 1000 to ensure convergence of each rule. As seen in this figure, the performance dropped as the number of output spikes increased, and most noticeably for the INST rule which returned a minimum performance value approaching 0% when input patterns were identified by five output spikes. By comparison, the CHRON rule gave rise to the highest performance levels over the entire range of output spikes tested, closely followed
Figure 5.10: The classification performance of each learning rule as a function of the number of target output spikes, when classifying 10 input patterns into 5 separate classes. Correct classifications were considered when the number of actual output spikes matched that of its target, and each actual output spike fell within 1 ms of its corresponding target. In this case, a network containing 200 presynaptic neurons was trained over an extended 1000 epochs, and results were averaged over 20 independent runs.

by the FILT rule. If we count the maximum number of output spikes learnable by the network with a minimum 90% performance level, we obtain one, three and five output spikes for INST, FILT and CHRON, respectively, where the associated number of training epochs in each instance are plotted in the right panel of the figure. It is also observed that CHRON was fastest in training the network to learn multi-spike based pattern classifications, followed by FILT and then INST.

**Summary.** Taken together, the results of this subsection demonstrate a similarity in the performance of the FILT and CHRON rules under most circumstances when classifying input patterns, except when applied to learning multi-spike based classifications for which CHRON was best suited. The INST rule, however, performed worst in all cases, and in particular displayed difficulties when classifying input patterns with increasingly fine temporal precision. This disparity between INST and the other two rules can be explained by its lack of a distinct treatment for shifting together neighbouring target and actual postsynaptic spikes, which, as predicted in the previous section, can lead to oscillatory postsynaptic spiking (see Fig. 5.5). Hence, it is evident that incorporating a mechanism for linking together postsynaptic spikes in a learning rule confers a strong advantage when
temporally precise encoding of input patterns is desired.

From the experiment concerning pattern classifications based on multiple output spike timings, it was found for each of the learning rules that the performance decreased with the number of target output spikes. This is not surprising given that the network needed to match every one of its targets with the same level of temporal precision, effectively increasing the synaptic load of the network during learning. Qualitatively, these results are consistent with those found in Florian (2012) for the CHRON rule. It is important to note, however, that multi-spike based encoding can in some instances lead to increased performance; for example, if classes are distinguished by finding a closest match for an output response, rather than with absolute temporal precision (Gardner & Grüning, 2014; Gardner et al., 2015).

In these experiments a fixed number of classes was used when measuring the performance and memory capacity of the network. Our reason for not also measuring the dependence of the network memory capacity on the number of classes comes from the study by Florian (2012) where it was demonstrated, at least for the CHRON rule, that there is no such dependence when using the latency of an output spike to encode input information.

### 5.4.6 Input Noise

In this final experiment, we investigated the impact of input noise on the temporal precision of postsynaptic spiking when an SNN was trained under the INST, FILT or CHRON learning rule. Similarly as before, the network was tasked with classifying input patterns into different classes based on the precise timing of a postsynaptic spike; however, in this case input patterns were also jittered to simulate background noise. In this way, we can also interpret this experiment as testing the network’s ability to generalise to similar, jittered input patterns.

On this learning task, the network contained 400 presynaptic neurons and was tasked with learning to classify a fixed number, \( p = 50 \), of arbitrarily generated input patterns into five separate classes, based on the timing of a single postsynaptic spike. An equal number of input patterns was assigned to each class, and a pattern was correctly identified when an actual output spike fell within 5 ms of
its respective target timing. Target output spikes were distributed at the start of every simulation with equally spaced timings: 60, 90, 120, 150 and 180 ms for each of the five classes, respectively. In this way, patterns belonging to separate classes could more easily be discriminated by the network as the level of input noise was increased. Input noise was simulated by jittering the timings of input spikes at the start of each trial; specifically, each input spike was randomly displaced about its reference timing according to a Gaussian distribution with a given noise amplitude. An illustration of our implementation is shown in Fig. 5.11. Here we consider a maximum noise amplitude of 20 ms, with measurements taken in 2 ms increments. Additionally, for this experiment the learning rate was scaled down to \( \eta = 300/(n_i n_s p) \); this value was indicated through preliminary simulations to give optimal performance for INST, FILT and CHRON when applied in the presence of input noise.

Shown in Fig. 5.12 is the performance of the network as a function of the noise amplitude or input jitter. The network contained 400 presynaptic neurons, and was trained over 500 epochs to classify all the input patterns into their respective classes. From this figure, it can be seen all three learning rules followed the same general trend of a decrease in the performance with the input jitter. Clearly, the learning rules were effectively equivalent in terms of their performance when subject to input noise; this is to be expected, however, since it becomes significantly
Figure 5.12: The performance of each learning rule as a function of the input noise when classifying 50 input patterns into five separate classes. A single, unique, target spike time identified each class of input patterns, which the network’s postsynaptic neuron had to learn to match to within 5 ms. The network contained 400 presynaptic neurons, and was trained over 500 epochs. Results were averaged over 20 independent runs.

more challenging for the network to link together neighbouring target and actual output spikes if fluctuations are present due to background noise.

Essentially, the principal aim of a noisy network would simply be to respond with a postsynaptic spike with reasonable temporal precision, rather than attempting to shift an actual output spike towards its target timing for a precise match. If we neglect the linkage between neighbouring postsynaptic spikes, then INST, FILT and CHRON would effectively give rise to identical synaptic weight changes.

### 5.5 Discussion

We have presented two theoretically justified learning rules: INST and FILT, and tested their performance in terms of the maximum number of spike patterns memorisable per synapse. In this case, spike patterns were identified based on the precise timing of an output spike by a postsynaptic neuron, reflecting experimental observations of real biological networks. We also obtained results for the E-learning CHRON rule, against which the performance of INST and FILT was compared. Interestingly, we found FILT approached the high performance level
of CHRON: relating to its ability to smoothly converge towards an optimum solution by virtue of its postsynaptic spike-timing linkage mechanism. By contrast, INST returned the lowest performance in most cases, which was underpinned by its tendency to result in oscillations of emitted postsynaptic spikes around their target timings.

### 5.5.1 Optimal Learning

Essentially, weight changes driven by the INST and FILT rules depend on a combination of two activity variables: a postsynaptic error term to signal appropriate output responses, and a presynaptic eligibility term to capture the coincidence of input spikes with the output error. Also, INST and FILT differ with respect to their postsynaptic error term: while INST relies on the instantaneous difference between a target and actual output spike train, FILT instead relies on the smoothed difference between an exponentially convolved target and actual output spike train. Both rules, however, share the same presynaptic eligibility term, that is the PSP evoked due to an input spike. From our formulation, the PSP was determined as the presynaptic factor, whereas the PSD and SPAN rules instead rely on an arbitrarily defined presynaptic kernel that is typically related to the postsynaptic current (Mohemmed et al., 2012; Yu et al., 2013). Interestingly, in the authors’ analysis of the SPAN rule an alpha kernel was indicated as providing the best performance during simulation runs, which closely resembles the shape of a PSP curve as used here (see Fig. 5.13).

In our analysis, we determined the FILT rule as giving rise to desired weight changes under most circumstances, thereby explaining its high performance in simulations. Specifically, FILT operates in such a way as to remove erroneously-timed output spikes, insert output spikes at their desired target timings, and shift remaining output spikes towards their targets if they are close enough in time. These three distinct operations bear a close resemblance to those carried out by the E-learning CHRON rule (Florian, 2012), which also happens to be a highly efficient spike-based neural classifier. The FILT and CHRON rules differ, however, in terms of their implementation: while FILT is compatible with online-based learning (c.f. Eq. (5.15)), CHRON is only implementable offline, given that it depends on discrete summations over cost functions derived from the VPD measure.
Comparatively, the INST rule is prone to imperfect convergence during learning, which we attribute to its absence of a distinct mechanism for linking together neighbouring target and actual output spikes.

Computer simulations were run to test the performance of the INST and FILT rules in terms of their temporal encoding precision, including the CHRON rule for comparison purposes. In general, we found FILT and CHRON were consistent with each other performance-wise, and largely outperformed INST. It is worth pointing out, however, that FILT is more straightforward to implement than CHRON, since it avoids the added complexity of having to establish whether target and actual output spikes are independent of each other or not based on the VPD measure \cite{Florian2012}. By comparison, INST is the simplest rule to implement, but comes at the cost of significantly decreased spike timing precision.

In terms of the learning tasks considered, networks were trained to categorise input patterns by the precise timings of output spikes; an alternative and more practical method for classifying patterns might instead take the minimum distance between a target and actual output spike train to discriminate between different input classes, which would be more effective in counteracting misclassifications due to input noise \cite{Gardner2014,Gardner2015}. In this work, we adopted a method based on the precise timings of output spikes for the sake of consistency with more directly related past studies \cite{Florian2012, Mohemmed2012, Yu2013}, and to more thoroughly compare the relative performance of each learning rule based on their temporal precision.

\subsection{Related Work}

In our approach, we started by taking gradient ascent on an objective function for maximising the likelihood of generating desired output spike trains, based on the statistical method of \cite{Pfister2006}. Next, we substituted the original stochastic spiking neuron model with a deterministic LIF neuron, such that output spikes were instead immediately generated upon crossing a fixed firing threshold. In this way, the resulting INST and FILT learning rules are theoretically justified, and yet also allow for very high temporal precision of output spikes. By comparison, most previous approaches to formulating supervised learning rules have relied on heuristic approximations, such as adapting the Widrow-Hoff rule,
as applied to rate-based networks, for use in spiking networks (Mohemmed et al., 2012; Yu et al., 2013), or mapping from Perceptron learning to spike-based learning (Albers et al., 2013; Xu et al., 2013). Also, although the well known Remote Supervised learning Method (ReSuMe) (Ponulak & Kasinski, 2010) can more rigorously be reinterpreted as a gradient descent procedure (Sporea & Grüning, 2013), assumptions are still made regarding the functional dependence of weight changes on the relative timings of spikes; specifically, exponential kernels are assumed to define the shape of a learning window, mimicking a Hebbian-like STDP rule (Gerstner & Kistler, 2002). Although many of the aforementioned rules have demonstrated good performance when tested on various learning tasks, the heuristics used in their formulation makes it difficult to guarantee the efficiency of their solutions in general. The main intention of this work has been to address this shortcoming.

It is worth highlighting that the INST and FILT rules are capable of learning multiple target output spikes; this is an important characteristic of any spike based learning rule, and makes them more biologically relevant when considering that precise spike timings in the nervous system represent a more fundamental unit of computation than that afforded by lengthier firing rates (van Rullen et al., 2005). Multi-spike learning rules are a natural progression from single-spike rules, such as the original SpikeProp algorithm which is restricted to learning single-spike target timings (Bohte, Kok, & Poutré, 2002), and the Tempotron which is only capable of learning to fire or not-fire (Gütig & Sompolinsky, 2006).

Furthermore, an important point to raise here is the selectivity of a postsynaptic neuron to a particular segment of an input pattern (Humble et al., 2012), as described in detail in the discussion of section 3.2. In our experiments it might appear that a postsynaptic neuron just learns to respond to a small segment of an input pattern when firing a single output spike, although more accurately it is also actively learning to suppress its output activity in response to the pattern at all other moments in time. Despite this, it can still be argued that since the neuron only positively responds to a small segment of the pattern, then it follows that only limited information pertaining to its received presynaptic input can be communicated further downstream of the network. This is necessary to take into account when designing a spike-based neural classifier, although this falls outside of the scope of this thesis, which is focused instead on the proof of concept of a
derived learning method rather than its real-world application.

5.5.3 Biological Plausibility

Out of the rules studied here, we believe FILT matches most criteria to be considered of biological relevance. First, weight updates under FILT depend on pre- and postsynaptic activity variables that are locally available at each synapse. Second, its postsynaptic error term is communicated by a smoothly decaying signal that is based on the difference between filtered target and actual output spikes, which might conceivably arise from calcium related signalling influenced by backpropagated action potentials (Palmer & Stuart 2009; Bush & Jin 2012). Finally, it is implementable as an online learning method, which is important when considering this is how information is likely processed in the nervous system.

5.6 Chapter Summary

In this contribution we have addressed the scarcity of existing learning rules for spiking networks that are theoretically justified and which allow for the learning of multiple and precisely-timed target output spikes. In particular, we have demonstrated our proposed FILT rule, that is based on exponentially filtered output spike trains, to be a highly efficient learning method.

Classifications based on the timings of spikes are of interest, since they are theoretically more capable than rate-based representations in conveying information regarding rapidly changing input features. In this way, SNNs have strong potential for solving real-world problems in areas such as image recognition, or time series prediction tasks.
Chapter 6

Learning in Multilayer Spiking Neural Networks

This chapter investigates how multi-layered networks of spiking neurons can learn to form representations of input patterns using a fully temporal coding scheme. To this end, a new supervised learning rule is presented, termed MultilayerSpiker, that can train SNNs containing hidden layer neurons to perform transformations between spatio-temporal input and output spike patterns.

The performance of MultilayerSpiker is demonstrated in terms of the number of pattern mappings it can learn, the complexity of network structures it can be used on and its classification accuracy when utilising multi-spike based encodings of input patterns. Finally, MultilayerSpiker is shown to have a biologically plausible implementation, such that synaptic weight modifications in each layer depend on a linear combination of global error signals that mimics the function of a neuromodulator such as dopamine.

This work contributes both to a systematic understanding of how computations might take place in the nervous system, and a learning rule that displays strong technical capability. Most of this contribution has been published in Gardner et al. (2015).
6.1 Introduction

Drawing on the process of Hebbian learning as inspiration, a variety of supervised learning rules have been proposed that can train either single- or multilayer networks to generate desired output spike patterns in response to spatio-temporal input spike patterns; an overview of learning rules for single-layer SNNs is provided in Chapter 3. Comparatively, however, the majority of research has focused on training single- rather than multilayer networks.

To address this, we propose a new supervised learning rule for multilayer spiking neural networks, termed MultilayerSpiker. Our rule extends the single-layer learning rule of Pfister et al. (2006) to multiple layers by combining a maximum likelihood approach with error-backpropagation. We demonstrate the efficacy of the proposed learning rule on several spike pattern transformation tasks: both in terms of the accuracy of input pattern classifications based on multi-spike codes and the time taken to converge in learning. Finally, we propose a more biologically plausible implementation of backpropagation learning, and discuss the underlying neural mechanisms that might guide the learning of desired target output spike trains.

This contribution is organised as follows. In section 6.2 an overview of existing learning rules for multilayer spiking networks is provided, including a brief discussion of their formulation and limitations. Section 6.3 describes the neuron model used in our analysis, and introduces our multilayer learning rule as applied to a structured, feed-forward network containing a hidden layer of spiking neurons. This section also details the pattern recognition method used in our work, that utilises multi-spike based encodings of input patterns. Contained in section 6.4 are simulation results, benchmarking the performance of our MultilayerSpiker learning rule on a variety of tests, including: mapping between arbitrary input-output spike patterns, solving the linearly non-separable XOR computation, classifying large numbers of patterns, classifying a noisy dataset and learning fully spatio-temporal input-output spike pattern transformations. We also present an alternative and more biologically plausible formulation of backpropagation learning, and compare its performance against that of our derived rule for both single- and multiple-output multilayer networks. Finally, section 6.5 discusses MultilayerSpiker in relation to existing backpropagation learning rules.
6.2 Background

Existing work that has examined networks containing hidden spiking neurons include the gradient descent learning rules: SpikeProp (Bohte, Kok, & Poutré, 2002) and multilayer ReSuMe (Sporea & Grüning, 2013), an approach treating spiking neurons as tunable filters (Bohte, 2011), an online method combining supervised and unsupervised learning (Wang et al., 2014) and the recurrent network learning rules formulated by Brea et al. (2013); Rezende & Gerstner (2014).

Learning rules for multilayer spiking networks have proven to be a challenge to formulate, especially given the discontinuous nature of neuronal spike-timing. A typical solution has been to assume a linear functional dependence of a neuron’s spike timing on presynaptic inputs about its firing threshold, such that small changes in its input with respect to synaptic weights shifts the timing of a generated output spike. However, such an approach has the disadvantage of constraining the learning rate to a small value, as with SpikeProp (see subsection 3.4.1). A more recent study (McKennoch et al., 2009) has circumvented this issue by instead formulating a gradient descent learning rule for a multilayer network containing theta neurons, which avoids any reliance on the precise nature of presynaptic inputs. The theta neuron is a nonlinear phase model, and is advantageous in the sense that it more accurately approximates the dynamical behaviour of biological neurons than the standard LIF neuron model, while retaining analytical tractability. Despite this, the rule is restricted to producing single output spikes, and can only encode for inputs by the latency of an output spike. A further approach has instead treated a spiking neuron as a stochastically firing unit, such that spikes are distributed according to an underlying, instantaneous firing rate that in turn has a smooth functional dependence on network parameters; for example, in multilayer ReSuMe a linear Poisson neuron model was used as a substitute for deterministic spiking neurons in each layer during its derivation (Sporea & Grüning, 2013).

Multilayer learning rules have demonstrated success on several benchmark classification tasks, including the linearly non-separable XOR computation and Iris dataset (Bohte, Kok, & Poutré, 2002; McKennoch et al., 2009; Sporea & Grüning, 2013) that cannot otherwise be solved by single-layer networks. Aside from this, few attempts have been made in establishing the performance of a multilayer
spiking network when learning to perform a large number of input-output spike pattern mappings; it is likely that the presence of more than one layer can enhance the computational capacity of the network, by increasing the number of spiking neurons that can perform computations on network inputs. Progress in this area has been hindered by the complexity that arises from applying learning rules to multilayer rather than single-layer spiking networks.

Typically, the classification methods used for both single- and multilayer learning rules have considered simplified coding schemes. For example, both SpikeProp and the Chronotron have taken the latencies of single output spikes in order to perform input classifications, and the Tempotron a binary spike / no-spike output code to discriminate between two input classes. Ideally, a fully temporal coding scheme should be utilised such that input patterns are classified based on the precise timings of multiple rather than single output spikes. We have previously indicated the advantages of using a fully temporal code in Gardner & Grüning (2014), and in particular found that multiple, rather than single, output spikes increased the accuracy of classifications.

Here we propose a supervised learning rule, termed MultilayerSpiker, for training multilayer spiking neural networks to perform transformations between spatio-temporal input-output spike patterns. In the rule’s formulation, we first consider a suitable likelihood function for generating desired output spike patterns, upon which stochastic gradient ascent can be taken. The technique of backpropagation, as is traditionally used for rate-coded networks, is subsequently applied in finding hidden layer weight updates. In this way, our technique can be viewed as a generalisation of the single-layer learning rule by Pfister et al. (2006) to multiple layers. Our multilayer learning rule differs from those proposed by Brea et al. (2013) and Rezende & Gerstner (2014), which have instead taken gradient descent on the KL-divergence in a supervised and reinforcement setting respectively. The novelty of our approach comes from the application of backpropagation, and its indicated high performance when encoding for a large number of input spike patterns as multiple and precisely timed output spikes.
Chapter 6. Learning in Multilayer Spiking Neural Networks

6.3 Methods

6.3.1 Single Neuron Model

We start by considering a single postsynaptic neuron in the network, indexed $o$, that receives its input from $1 \leq h \leq n_h$ presynaptic neurons. The list of spikes due to a presynaptic neuron $h$ up to time $t$ is $y_h(t) = \{t^h_1, ..., t^h_h < t\}$, where $\hat{t}_h$ is always the last spike before $t$. If the postsynaptic neuron generates a list of output spikes $z_o(t) = \{t^o_1, ..., \hat{t}_o < t\}$ in response to the presynaptic spike pattern $y_h \in y$, then its membrane potential at time $t$ is defined by SRM:

$$u_o(t|y, z_o) := \sum_h w_{oh}(\mathcal{Y}_h * \epsilon)(t) + (\mathcal{Z}_o * \kappa)(t), \quad (6.1)$$

where $w_{oh}$ is the synaptic weight between neurons $h$ and $o$ \cite{Gerstner2002}. The term $(\mathcal{Y}_h * \epsilon)(t)$ denotes a convolution between a presynaptic spike train $\mathcal{Y}_h$ and a PSP kernel $\epsilon$, where $\mathcal{Y}_h$ is defined similarly to Eq. (3.1), and a convolution is defined by

$$(\mathcal{Y}_h * \epsilon)(t) \equiv \int_0^t \mathcal{Y}_h(t') \epsilon(t - t')dt'. \quad (6.2)$$

Similarly, the term $(\mathcal{Z}_o * \kappa)(t)$ denotes a convolution between a postsynaptic spike train $\mathcal{Z}_o$ and a reset kernel $\kappa$. The PSP and reset kernels are respectively given by:

$$\epsilon(s) = \epsilon_0 [e^{-s/\tau_m} - e^{-s/\tau_s}] \Theta(s) \quad \text{and} \quad (6.3)$$
$$\kappa(s) = \kappa_0 e^{-s/\tau_m} \Theta(s), \quad (6.4)$$

where $\epsilon_0 = 4 \text{ mV}$ is the PSP scaling factor, $\tau_m = 10 \text{ ms}$ the membrane time constant, $\tau_s = 5 \text{ ms}$ the synaptic rise time and $\Theta(s)$ the Heaviside step function; these choice of parameters resulted in a PSP with a maximum value of $1 \text{ mV}$ at a lag time of around $7 \text{ ms}$. The scaling factor for the reset kernel was set to $\kappa_0 = -\vartheta$, that is the negative of the formal firing threshold $\vartheta$. The SRM defined by Eq. (6.1) is equivalent that of Eq. (3.31), but expressed using a different notation; the reasons for this shall become apparent in the next subsection, where the multilayer learning rule is formulated.
Neuronal spike events are generated by a point process with stochastic intensity \( \rho_o(t) \), that is the instantaneous firing rate of a postsynaptic neuron, where the probability of generating a spike at time \( t \) over a small time interval \([t, t + \delta t]\) is given by \( \rho_o(t)\delta t \). The firing rate has a nonlinear dependence on the postsynaptic neuron’s membrane potential, that in turn depends on both its presynaptic input and the postsynaptic neuron’s firing history: \( \rho_o(t|y, z_o) = g(u_o(t|y, z_o)) \).

Here we take an exponential dependence of the firing rate on the membrane potential, defined by Eq. (2.8), with the instantaneous firing rate at threshold \( \rho_0 = 0.01 \text{ ms}^{-1} \) and the formal firing threshold \( \vartheta = 15 \text{ mV} \); these value choices ensure neurons remain quiescent when receiving low input drive. The parameter \( \Delta u \), as described in Eq. (2.8), controls the variability of spike timings, which has a prominent impact on network performance; specific value choices for \( \Delta u \) are presented in the next section.

### 6.3.2 Learning Rule

The learning rule is derived for a fully-connected, feed-forward SNN containing a single hidden layer. Input layer neurons just present spike patterns to the network, while both hidden and output neurons are free to perform computations on their respective inputs. In this chapter, input layer neurons are instead indexed as \( i \), hidden neurons \( h \) and output neurons \( o \).

We initially derive weight update rules for the connections between the hidden and output layers, as originally shown by Pfister et al. (2006). We then extend our analysis to include weight updates between the input and hidden layers using backpropagation, that is our novel contribution of a multilayer learning rule for networks of spiking neurons.

**Objective Function.** Both hidden and output layer neurons have their spikes distributed according to Eq. (2.8); the advantage of implementing a stochastic neuron model is that it allows for the determination of the likelihood of generating a desired output spike train. Specifically, the probability density of an output neuron \( o \) generating a list of desired output spikes \( z_o^{\text{ref}} \) = \( \{t_1^o, t_2^o, \ldots\} \) in response
to a hidden spike pattern \( y \) is found similarly to Eq. (3.40):

\[
P(z_{o}^{\text{ref}}|y) = \exp \left( \int_{0}^{T} \log (\rho_{o}(t|y, z_{o})) Z_{o}^{\text{ref}}(t) - \rho_{o}(t|y, z_{o}) dt \right),
\]

where \( Z_{o}^{\text{ref}}(t) = \sum_{f} \delta(t - \tilde{t}_{o}^{f}) \) and \( T \) is the duration over which pattern \( y \) is presented \( \)\( \text{(Pfister et al., 2006)} \). For more than one output neuron, the probability density of generating a desired spatio-temporal output spike pattern \( z_{o}^{\text{ref}} \in z^{\text{ref}} \) is given by

\[
P(z^{\text{ref}}|y) = \prod_{o} P(z_{o}^{\text{ref}}|y) = \exp \left( \sum_{o} \int_{0}^{T} \log (\rho_{o}(t|y, z_{o})) Z_{o}^{\text{ref}}(t) - \rho_{o}(t|y, z_{o}) dt \right).
\]

Taking the logarithm of Eq. (6.6) provides us with an objective function, that is a smooth function of the network parameters:

\[
\log P(z^{\text{ref}}|y) = \sum_{o} \int_{0}^{T} \log (\rho_{o}(t|y, z_{o})) Z_{o}^{\text{ref}}(t) - \rho_{o}(t|y, z_{o}) dt.
\]

Hence, we aim to maximise the log-likelihood of generating a desired output spike pattern through gradient ascent with respect to synaptic weights in the network. For clarity, we consider a network containing a single hidden layer, although our technique can straightforwardly be extended to include multiple hidden layers.

**Output Weight Updates.** Taking the positive gradient of the log-likelihood, Eq. (6.7), provides us with the direction of weight updates for output layer neurons, such that the expectation of generating a desired output spike pattern \( z^{\text{ref}} \) is increased, i.e.:

\[
\Delta w_{oh} = \eta_{o} \frac{\partial \log P(z^{\text{ref}}|y)}{\partial w_{oh}},
\]

where \( \eta_{o} \) is the output layer learning rate. By a similar approach as Eq. (3.44), the derivative of the log-likelihood can be found as

\[
\frac{\partial \log P(z^{\text{ref}}|y)}{\partial w_{oh}} = \int_{0}^{T} \frac{\rho_{o}(t|y, z_{o})}{\rho_{o}(t|y, z_{o})} \left[ Z_{o}^{\text{ref}}(t) - \rho_{o}(t|y, z_{o}) \right] (Y_{h} * \epsilon)(t) dt,
\]
where $\rho'_o(t|y, z_o) = \frac{d g(u)}{du}|_{u = u_o(t|y, z_o)}$, and $(Y_h \ast \epsilon)(t)$ is the convolution of the hidden spike train $Y_h(t')$ with the PSP kernel $\epsilon(t - t')$ as defined in Eq. (6.2). Given our choice of an exponential dependence for the firing rate on the membrane potential, defined by Eq. (2.8), it follows that

$$\frac{\rho'_o(t|y, z_o)}{\rho_o(t|y, z_o)} = \frac{1}{\Delta u_o},$$

(6.10)

where $\Delta u_o$ controls the variability of output neuron spiking. Hence, combining Eqs. (6.8), (6.9) and (6.10) provides the output layer weight update rule:

$$\Delta w_{oh} = \eta_o \int_0^T \delta_o(t|y, z_o) (Y_h \ast \epsilon)(t) dt .$$

(6.11)

We define the backpropagated error signal $\delta_o$ for the $o^{th}$ output neuron as

$$\delta_o(t|y, z_o) := \frac{1}{\Delta u_o} [Z_{o \text{ref}}(t) - \rho_o(t|y, z_o)],$$

(6.12)

that is substituted into Eq. (6.11) for compactness:

$$\Delta w_{oh} = \eta_o \int_0^T \delta_o(t|y, z_o) (Y_h \ast \epsilon)(t) dt .$$

(6.13)

From the above, we find positive values of $\delta_o$ signal the timings of desired output spikes, while negative values signal erroneous output activity. For output layer neurons, we set the ‘smoothness’ of the firing threshold to a small value $\Delta u_o = 0.2 \text{ mV}$ to increase the precision of output spike timings. This was originally derived by Pfister et al. (2006) for a single-layer SNN, and is equivalent but denoted differently to Eq. (3.46). An example of a weight update taking place in the output layer is shown in Fig. 6.1.

**Hidden Weight Updates.** Continuing through to the hidden layer, weights between input and hidden layer neurons are updated according to

$$\Delta w_{hi} = \eta_h \frac{\partial \log P(z_{\text{ref}}|y)}{\partial w_{hi}} .$$

(6.14)
Chapter 6. Learning in Multilayer Spiking Neural Networks

Figure 6.1: Example of the output layer weight update rule, in terms of hidden and output neuron activity. **Top row:** The left panel shows the fluctuations of a hidden neuron membrane potential relative to the firing threshold $\vartheta$, in response to an input pattern lasting duration $T$, where hidden spikes are indicated by vertical lines. The right panel shows the membrane potential of an output neuron, that responds to stimulation from hidden layer neurons. Target output spike times are indicated by the dotted lines. **Bottom row:** The left panel is the PSP evoked at the output neuron due to hidden spiking. The right panel is the candidate weight change as a function of time between the hidden and output neurons of this example, that depends on both the hidden-evoked PSP and output neuron activity, according to Eq. (6.13). Note the depressions in $\Delta w_{oh}$ coincide with the timings of actual output spikes which lead their target timings, while increases coincide with the timings of target output spikes. In this case, the effective weight update at time $T$ is positive, demonstrating the causal role of hidden spikes in eliciting accurate output responses.
where $\eta_h$ is the hidden layer learning rate. Using Eq. (6.7), and by making use of the chain rule, the gradient of the log-likelihood with respect to hidden layer weights can be expressed as

$$
\frac{\partial \log P(z^{\text{ref}}|y)}{\partial w_{hi}} = \sum_o \int_0^T \frac{\partial}{\partial w_{hi}} \left[ \log \left( \rho_o(t|y, z_o) \right) Z^\text{ref}_o(t) - \rho_o(t|y, z_o) \right] dt \\
= \sum_o \int_0^T \frac{\partial}{\partial w_{hi}} \left[ \rho'_o(t|y, z_o) \left[ Z^\text{ref}_o(t) - \rho_o(t|y, z_o) \right] \right] \frac{\partial u_o(t|y, z_o)}{\partial w_{hi}} dt.
$$

(6.15)

Using Eqs. (6.10) and (6.12), the above can be compacted:

$$
\frac{\partial \log P(z^{\text{ref}}|y)}{\partial w_{hi}} = \sum_o \int_0^T \delta_o(t|y, z_o) \frac{\partial u_o(t|y, z_o)}{\partial w_{hi}} dt.
$$

(6.16)

The membrane potential of an output layer neuron has a dependence on the firing activity of neurons in the hidden layer according to Eq. (6.1), hence the second term on the right-hand side of Eq. (6.16) can be rewritten as

$$
\frac{\partial u_o(t|y, z_o)}{\partial w_{hi}} = w_{oh} \frac{\partial}{\partial w_{hi}} (Y_{h} * \epsilon)(t).
$$

(6.17)

Weights changes take place on a time scale of $T \gg \tau_m$, therefore the gradient of the convolution $(Y_{h} * \epsilon)(t)$ can be well approximated by

$$
\frac{\partial}{\partial w_{hi}} (Y_{h} * \epsilon)(t) \approx \int_0^t \frac{\partial Y_{h}(t')}{\partial w_{hi}} \epsilon(t - t') dt'.
$$

(6.18)

The spike train $Y_{h}(t')$ is a discontinuous random variable with no smooth dependence on network parameters, leaving the gradient $\frac{\partial Y_{h}(t')}{\partial w_{hi}}$ difficult to solve analytically. Therefore, applying the technique used in [Frémaux et al. (2013)] we heuristically make the substitution $Y_{h}(t') \to \langle Y_{h}(t') \rangle_{y_h|x}$, that is the expectation of the hidden spike train $Y_{h}(t')$ conditioned on the input spike pattern $x$. The expectation of $Y_{h}(t')$ has a smooth dependence on network parameters, and its
gradient is given by:

\[
\frac{\partial \langle Y_h(t') \rangle_{y_h|x}}{\partial w_{hi}} = \frac{\partial}{\partial w_{hi}} \int Q(t') P(y_h = q|x) dq = \int Q(t') P(y_h = q|x) \frac{\partial \log P(y_h = q|x)}{\partial w_{hi}} dq ,
\]

(6.19)

where we have used the relation \( \frac{1}{P} \frac{\partial P}{\partial w_{hi}} = \frac{\partial \log P}{\partial w_{hi}} \), the integral runs over all possible lists of spikes \( q(t') = \{t'^1, t'^2, ..., t' < t'\} \) up to time \( t' \) and \( Q(t') = \sum_{t'^j \in q} \delta(t' - t'^j) \) is a spike train. \( P(y_h = q|x) \) is the probability density of the list of hidden spikes \( y_h \) being equal to \( q \), conditioned on \( x \). The probability density or likelihood of a hidden neuron generating a list of spikes \( q \) up to time \( t' \) in response to \( x \) is defined similarly to Eq. (6.5):

\[
P(y_h = q|x) = \exp \left( \int_0^{t'} \log (\rho_h(s|x, q)) Q(s) - \rho_h(s|x, q) ds \right),
\]

(6.20)

and the gradient of the log-likelihood:

\[
\frac{\partial \log P(y_h = q|x)}{\partial w_{hi}} = \frac{1}{\Delta u_h} \int_0^{t'} [Q(s) - \rho_h(s|x, q)] (X_i * \epsilon)(s) ds .
\]

(6.21)

Hence, Eq. (6.19) becomes:

\[
\frac{\partial \langle Y_h(t') \rangle_{y_h|x}}{\partial w_{hi}} = \frac{1}{\Delta u_h} \int Q(t') P(y_h = q|x) \left( \int_0^{t'} [Q(s) - \rho_h(s|x, q)] (X_i * \epsilon)(s) ds \right) dq ,
\]

(6.22)

such that a spike generated by the neuron at time \( t' \) depends not only on recent input spikes, but also on its own entire spiking history \( q \) through the integration between times 0 and \( t' \).

The above can be simplified if we choose to neglect the neuron’s firing history by taking the last hidden spike time \( \hat{t}'_h < t' \) as given, allowing the substitution \( Y_h(t') \rightarrow \langle Y_h(t') \rangle_{y_h|x, \hat{t}'_h} \) for the expectation of the hidden spike train \( Y_h(t') \) conditioned on both the input pattern \( x \) and last hidden spike \( \hat{t}'_h \) (Frémaux et al., 2013). In this case, neglecting the neuron’s firing history is not an unreasonable choice, given that the gradient of \( Y_h(t') \) is convolved by the exponential PSP kernel \( \epsilon \) in Eq. (6.18) that already captures the recent firing history of the neuron. Hence, the gradient of the hidden spike train in Eq. (6.18) can instead be expressed in
terms of the value of a spike train \( Q \) at each point in time:

\[
\frac{\partial \langle Y_h(t') \rangle_{y_h|x,\hat{t}_h}}{\partial w_{hi}} = \frac{\partial}{\partial w_{hi}} \sum_{Q \in \{0, \delta\}} Q(t') P(y_h = q|x, \hat{t}_h) \\
= \delta(t' - \hat{t}') \frac{\partial \rho_h(t'|x, \hat{t}_h)}{\partial w_{hi}},
\]

(6.23)

where we have used the identity \( P(y_h = q|x, \hat{t}_h) = \rho_h(t'|x, \hat{t}_h) \) and \( \delta(t' - \hat{t}') \) is the Dirac distribution as a function of a last spike \( \hat{t}' \). Using Eqs. (6.1) and (2.8) we find:

\[
\frac{\partial \langle Y_h(t') \rangle_{y_h|x,\hat{t}_h}}{\partial w_{hi}} = \frac{1}{\Delta u_h} \delta(t' - \hat{t}') \rho_h(t'|x, \hat{t}_h)(\mathcal{X}_i \ast \epsilon)(t') \\
= \frac{1}{\Delta u_h} \langle Y_h(\mathcal{X}_i \ast \epsilon) \rangle_{y_h|x,\hat{t}_h}.
\]

(6.24)

On each learning episode, our best estimate for the expected gradient comes from considering the current observation of \( y_h \) given \( x \); hence, the expectation can be dropped and the above combined with Eqs. (6.17) and (6.18) to give

\[
\frac{\partial u_o(t|y,z_o)}{\partial w_{hi}} = \frac{w_{oh}}{\Delta u_h} (\mathcal{Y}_h(\mathcal{X}_i \ast \epsilon)) \ast \epsilon(t),
\]

(6.25)

where we have defined a double convolution as:

\[
(\mathcal{Y}_h(\mathcal{X}_i \ast \epsilon)) \ast \epsilon(t) \equiv \int_t^0 \mathcal{Y}_h(t') \left[ \int_{t'}^{t''} \mathcal{X}_i(t'') \epsilon(t' - t'') dt'' \right] \epsilon(t - t') dt'.
\]

(6.26)

Finally, combining Eq. (6.25) with Eqs. (6.14) and (6.16) provides the hidden layer weight update rule:

\[
\Delta w_{hi} = \frac{\eta_h}{\Delta u_h} \sum_o w_{oh} \int_0^T \delta_o(t|y,z_o)(\mathcal{Y}_h(\mathcal{X}_i \ast \epsilon)) \ast \epsilon(t) dt.
\]

(6.27)

For hidden layer neurons, we set \( \Delta u_h = 2 \text{ mV} \); our choice of \( \Delta u_h > \Delta u_o \) was motivated by the need for increased variation in hidden neuron spiking for learning to succeed, as indicated by preliminary results. Furthermore, given the direct dependence of hidden weight updates on the availability of hidden neuron spikes, it is necessary that a degree of activity persists in the hidden layer: an absence of
hidden activity would otherwise prevent updates from taking place and result in stagnated learning. To this end, hidden weights are additively modified through synaptic scaling, as discussed in the next section. An example of a weight update taking place in the hidden layer is shown in Fig. 6.2. Taken together, Eqs. (6.13) and (6.27) define the MultilayerSpiker learning rule.

### 6.3.3 Synaptic Scaling

In order for hidden layer weight updates to take place, a degree of background hidden neuron spiking activity is necessary during learning. This condition can be satisfied if we apply synaptic scaling to hidden layer weights, that has previously been shown to maintain a homeostatic firing rate and introduce competition between afferent connections (van Rossum et al., 2000).

Therefore, in addition to Eq. (6.27), hidden weights $w_{hi}$ are modified by a scaling rule defined similarly to Eq. (3.4), where $\psi = 10^{-2}$ Hz$^{-1}$ is the scaling strength, $\nu_h$ the actual firing rate of the $h$th hidden neuron and $\nu_{\text{max}} = 40$ Hz and $\nu_{\text{min}} = 2$ Hz the maximum and minimum reference firing rates, respectively. This drives the firing rate of each hidden neuron to remain within the range: $2 \leq \nu_h \leq 40$ Hz, thereby making the network less sensitive to its initial state and preventing extremes in the firing activity of hidden neurons (Sporea & Grünig, 2013).

### 6.3.4 Pattern Statistics

Input patterns were presented to the network by $n_i = 100$ input layer neurons, where each input neuron contained an independent Poisson spike train with a mean firing rate of 6 Hz. A relative refractory period with a time constant of 10 ms was simulated when generating each spike train for increased biological realism. A random realisation of each input pattern was used, for a total of $p$ different patterns.

Learning took place on an episodic basis, where each episode corresponded to the presentation of an input pattern to the network lasting duration $T = 500$ ms which is on the same time scale as sensory processing in biological networks. The order in which input patterns were presented was random. Unless otherwise
Figure 6.2: Example of the hidden layer weight update rule, in terms of input, hidden and output neuron activity.  

Top row: The left panel shows an input spike train, that is generated from a Poisson process. The middle panel shows the membrane potential of an example hidden neuron, which is partly stimulated by input spikes from the preceding panel. The right panel shows the membrane potential of an output neuron, where target output spike times are indicated by dotted lines. 

Bottom row: Shown along this row are a series of synaptic traces, which have a functional dependence on their preceding panel. The left panel is the PSP evoked at the hidden neuron, resulting from the input spike train. The middle panel is a double convolution (Eq. (6.26)) that captures coincident input and hidden neuron spikes, and acts as a hidden synaptic eligibility trace. In this case, only input and hidden neuron spikes up to the first half of \( T \) are causally related. The right panel shows the progression of the candidate weight change \( \Delta w_{hi} \) between the input and hidden neurons of this example, that has a functional dependence on both the hidden synaptic eligibility trace and output neuron activity, according to Eq. (6.27). The final, effective weight update is positive, indicative of the causal contribution of the input spike train in generating a desired output spike train through the hidden layer.
stated, simulations were run over 1000 episodes to ensure a sufficient amount of time for the network to learn the desired number of inputs. Hence, on average, each input pattern was presented 1000 times.

Every input pattern was associated with a target output pattern, and multiple inputs belonging to the same class shared the same target output. A target output pattern consisted of a predetermined spike train at each output neuron, and target spike trains contained the same number of spikes $n_s \in \{1, 10\}$ at each output, depending on the learning task. Target spike trains were initialised by randomly selecting each target spike time $\tilde{t}_f$ from a uniform distribution over the interval $\tilde{t}_f \in [40, T]$ ms, with an interspike separation of at least 10 ms to avoid conflicted output responses during learning. A minimum target spike timing of 40 ms was taken given the evidence that values $\tilde{t}_f < 4\tau_m$ led to reduced network performance [Florian 2012].

At each output neuron, target spike trains differed from each other by a minimum vRD (see Eq. (6.28)) of $D_{\text{min}} > n_s/2$ to ensure each class of input patterns was assigned a unique target response, and to reduce crosstalk during learning. The minimum distance scaled with the number of target output spikes, thereby increasing the separation between classes. For our definition of $D_{\text{min}}$ and choice of $T$, a maximum of $c = 66$ classes identifiable by single target output spikes was supported, and more correspondingly for multiple target output spikes.

6.3.5 Pattern Recognition

Networks were trained to classify input patterns by the timings of output spikes, such that multiple inputs belonging to the same class shared the same target output. Target outputs were randomly set at the start of each simulation, and networks were trained to assign $p$ input patterns into $c$ classes. For each class, a target output contained between one and ten spikes, depending on the learning task.

Instead of relying on precisely matched target and actual output spike trains to classify inputs, we instead allowed for sufficiently accurate output spike trains that were closest to their desired targets in comparison with any other potential target; the reason for this relaxed classification criterion in comparison with the
previous chapter comes for our use of a stochastic neuron model, which is subject
to much higher variability of output spike timings. To discriminate between input
classes we used the van Rossum Distance (vRD) \(^{(\text{van Rossum, 2001}}\), that is a
metric for the temporal distance between two spike trains.

If \(Z_o\) and \(Z^{\text{ref}}_o\) are the actual and target output spike trains of an output neuron
\(o\), respectively, then the vRD between them is defined by \(^{(\text{van Rossum, 2001}}\):

\[
D(Z_o, Z^{\text{ref}}_o) := \frac{1}{\tau_c} \int_0^\infty \left[ \tilde{Z}_o(t) - \tilde{Z}_o^{\text{ref}}(t) \right]^2 dt ,
\]

where we set the coincidence time constant \(\tau_c = 10\) ms. The low-pass filtered
spike trains \(\tilde{Z}_o\) and \(\tilde{Z}_o^{\text{ref}}\) are determined using a similar operation to Eq. \((4.18)\).

Using Eq. \((6.28)\), the vRD between an actual output generated by the network
and each potential target output is computed, giving the set of distances \(D = \{D_1, D_2, ..., D_c\}\) for a total of \(c\) class labels. A correct classification of the input is
then made if the desired class label \(l\) matches the index of the minimum distance,
that is if \(l = \arg\ min_k D\) for \(D_k \in D\). In the case of no output spikes, an erroneous
classification was taken.

For a network containing more than one output neuron, responses consist of
spatio-temporal output patterns \(Z_o \in Z\) with corresponding target outputs \(Z^{\text{ref}}_o \in Z^{\text{ref}}\). To compute the distance between two spatio-temporal spike patterns, the
vRD is summed over every output neuron:

\[
D(Z, Z^{\text{ref}}) = \frac{1}{\tau_c} \sum_o \int_0^\infty \left[ \tilde{Z}_o(t) - \tilde{Z}_o^{\text{ref}}(t) \right]^2 dt ,
\]

and is determined with respect to each potential class. Similarly to a network
containing a single output neuron, a correct classification of an input is made if its
desired class label matches the index of the minimum spatio-temporal distance.

### 6.4 Simulation Results

The performance of the MultilayerSpiker learning rule was tested through simula-
tions of multilayer networks trained to perform temporally-precise spike pattern
transformations. In our analysis we considered networks containing either single or multiple output neurons.

Example tasks for single-output networks include measuring the resilience of the network to input noise during learning, the solution of the XOR computation, a comparison between specific network structures, the memory capacity of the network and its ability to classify patterns on a synthetic dataset. For multiple-output networks, the performance of the learning rule was tested for networks tasked with performing fully spatio-temporal spike pattern transformations.

6.4.1 Network Setup

In all simulations, input patterns were represented by the firing times of \( n_i = 100 \) input layer neurons, which consisted of a Poisson spike train at each input neuron (see section 6.3.4). Patterns were presented to the network episodically in a random order, and weights were updated at the end of each episode. Depending on the learning task, a variable number \( n_h \) of hidden neurons were implemented in the network to establish the dependence of the network performance on the hidden layer size. Here we first present results from simulations of multilayer networks containing a single output neuron as the readout, and then extend our analysis to include multilayer networks containing multiple output neurons. A more detailed description of the network setup used in each set of simulations can be found in Appendix A.

6.4.2 Performance of the Learning Rule

The performance of the proposed learning rule is demonstrated by training a multilayer network to perform generic input-output spike pattern mappings. We first focus on the relatively simple task of performing a single input-output mapping, and then consider more complex multiple input-output mappings that are subject to noise.

Single Input-Output Mapping. A multilayer network was trained to map between a fixed input pattern and a target output spike train. The network
contained 10 hidden neurons and a single output neuron, and was tasked with learning to match the timings of five target output spikes. An illustration of the network setup is shown in Fig. 6.3, along with example spike rasters depicting input, hidden and output neuron spiking activity over a typical simulation run.

In this example, we examine a selected hidden neuron that contributed strongly to the responses of the output neuron close to the target spike times: 166, 249 and 415 ms (Fig. 6.3B). From this hidden neuron spike raster, highly variable spike times were observed over the first 200 episodes, that subsequently fine-tuned themselves to the timings of target output spikes; this initial phase of variable activity demonstrated a form of internal stochastic exploration by the network, during which time desirable hidden spike patterns were searched for by the network which contributed to eliciting accurate output spikes. As learning progressed, hidden neurons generated bursts of spikes around the timings of target output spikes, such that the likelihood of the network generating accurate output responses increased. In this simulation, the majority of hidden layer neurons contributed to driving accurate output spike responses, such that the load imposed on the network in the form of hidden synaptic weight modifications was more evenly distributed amongst hidden layer neurons.

From the output spike raster (Fig. 6.3C) it is clear that every target output spike was learnt successfully, and within just 100 episodes. However, because a stochastic rather than a deterministic neuron model was implemented, a small degree of variation in the timings of output spikes about their respective targets was apparent. Despite this, the network still generated output responses to a sufficiently high level of accuracy, that is supported by the vRD (defined in Eq. (6.28)) with a final average value $\bar{\Delta} = 0.55 \pm 0.13$ (Fig. 6.3E). For an impression of this vRD value, a distance of 0.55 corresponds to a typical time shift of 1.17 ms between paired actual and target output spikes.

**Synaptic Weight Distributions.** Shown in Fig. 6.4 is an example of the evolution of both hidden and output synaptic weights with the number of learning episodes and their final distribution, that corresponds to the previous experimental setup. In the left panel (Fig. 6.4A), the weights on the hidden neuron can be seen to diverge continuously during learning, with almost twice as many positive as negative weights by the final episode. This contrasts with the evolution of the
Figure 6.3: Learning a target output spike train in response to a single, fixed input pattern. The network contained \( n_i = 100 \) input neurons, \( n_h = 10 \) hidden neurons and a single output neuron. The input pattern was repeatedly presented to the network over 1000 episodes, where each episode lasted duration \( T = 500 \) ms. The target output spike train contained five spikes at times: 83, 166, 249, 332 and 415 ms. (A) A spike raster of the input pattern that was presented to the network on each episode. (B) The activity of a hidden neuron with each episode, that contributed strongly to the firing times of the output neuron. (C) The activity of the output, where the five target output spike times are indicated by crosses. (D) An illustration of the multilayer network setup. (E) The evolution of the distance between the actual and target output spike trains of the network, given as a moving average of the van Rossum Distance \( \tilde{D} \) with each episode (Appendix [B]) and taken over 100 independent simulation runs. The shaded region shows the standard deviation.
weights on the output neuron (Fig. 6.4B, left panel), which attained rapid convergence during learning. We note that in our implementation output weights were confined to positive values, while hidden weights had no such restriction (see Appendix A); preliminary simulations indicated that negative output weight values for a single output neuron had little impact on its performance. This simulation was run for a maximum of 1000 learning episodes, yet it can be seen that only output weights, and not hidden weights, converged by the end of learning; despite this, only small decreases in the associated network accuracy was observed towards the end of learning (see Fig. 6.3E), hence it follows that synaptic weight
changes had little impact on the network performance by this stage of training.

At the end of learning, hidden weights closely followed a Gaussian distribution (Fig. 6.4A, right panel) and output weights a positively skewed distribution (Fig. 6.4B, right panel), with coefficients of variation $1.52 \pm 0.01$ and $0.375 \pm 0.009$ in the magnitude of hidden and output weight values respectively. Hence, in terms of the absolute value, hidden weights were more widely dispersed than output weights by a factor of just over four.

Multiple Input-Output Mappings with Noise. We next tested the performance of the multilayer network when learning to map between 10 arbitrary input-output spike pattern pairs, and the impact of input noise on learning. In this case, each input pattern was identified by a unique target output spike time. The network contained $n_h = 10$ hidden neurons and a single output neuron. In this experiment we introduced two new measures: the time shift $\Delta t$ and the performance $\hat{P}_c$. The time shift was taken as a moving average, with each learning episode, of the absolute temporal difference between target and actual output spikes, $\Delta t = |t_o - \hat{t}_o|$, which was computed only for instances where a single actual output spike was generated to provide a correct input classification. The measure $\hat{P}_c$ was taken as a moving average of the network’s classification performance, as defined in Appendix B. The time shift $\Delta t$ shared the same averaging window as $\hat{P}_c$, and its motivation came from providing a more physical perspective of the spike train dissimilarity measure $D$. The performance $\hat{P}_c$ measured the accuracy of network classifications based on a temporal code.

As shown in Fig. 6.5 learning took place over $10^4$ episodes to ensure sufficient training for the network. Noise was introduced to the network by jittering the timing of each input spike according to a Gaussian distribution at the start of every episode, with a standard deviation or amplitude that ranged in value from between 0 and 20 ms.

From the top row of panels (Fig. 6.5A) we found that noiseless input patterns resulted in the most accurate output spike times, providing a final distance of $0.11 \pm 0.02$ and a typical time shift of $0.8 \pm 0.1$ ms. By comparison, introducing 10 ms amplitude of input jitter (Fig. 6.5B) gave a final distance of $0.43 \pm 0.02$ and resulted in output spikes shifted by $4.0 \pm 0.2$ ms, thereby reducing the tem-
Figure 6.5: Learning to map between 10 input-output pattern pairs, with and without input noise. The network contained \( n_h = 10 \) hidden neurons and a single output neuron. Each input pattern was associated with a unique target output spike. **Left column:** The vRD between actual and target output spike trains. **Middle column:** The time shift between matching actual and target output spikes. **Right column:** The performance \( \tilde{P}_c \) of the network (Methods), when recognising input patterns by the timing of an output spike. (A) Learning in the absence of any input noise, and (B) learning with intermediate input noise. Input noise was simulated by adding jitter to the timings of input spikes on each episode, where jitter with an amplitude of 10 ms was used in (B). (C) Averaged values after \( 10^4 \) learning episodes, as a function of the input jitter amplitude. In all panels, each value was averaged over 20 independent runs, and error bars show the standard deviation.
poral precision of output spikes by a factor of five. In terms of the accuracy of input classifications, noiseless inputs resulted in a high performance level of 96 ± 2%, which dropped to 70 ± 4% with the addition of 10 ms amplitude of input jitter. Input noise increased the time taken to converge in learning, taking \((1.5 \pm 0.3) \times 10^3\) and \((2.0 \pm 0.2) \times 10^3\) episodes for noiseless and noisy (10 ms jitter) inputs respectively (see Appendix B for our choice of convergence measure).

The panels in Fig. 6.5C summarise results obtained for 0, 5, 10, 15 and 20 ms amplitude of input jitter, which show a smooth decrease in the network performance with the degree of input noise. However, even for up to 20 ms amplitude of input jitter output spikes still fell within 8 ms of their targets and inputs were classified correctly at least 40% of the time. This remains well above the chance performance level of 10%, thereby demonstrating the robustness of the multilayer network to strong input noise.

Multilayer Spiker has proven capable of training multilayer networks to perform generic input-output spike pattern mappings, and in particular when inputs were subject to high levels of noise. We have also indicated the necessity of both persistent and variable hidden neuronal spiking to ensure convergence of the learning rule, which was supported through synaptic scaling of hidden layer weights.

### 6.4.3 Dependence on Network Structure

In this section we compare the performance between single- and multilayer networks trained to perform spike pattern classifications. First we test the capability of each network structure in solving the classic, linearly non-separable XOR computation, and then apply each network to performing an increasing number of arbitrary spike pattern classifications to provide an indication of their relative capacity.

**The XOR Computation.** The learning rule was applied to solving the exclusive-or (XOR) computation, that is a non-trivial classification task. This is considered a standard benchmark for rate-coded neural network training, given that a hidden layer is necessary for its solution. The necessity of a hidden layer has also been specifically indicated for spiking networks in Grüning & Sporea (2012).
Chapter 6. Learning in Multilayer Spiking Neural Networks

An XOR computation maps two binary inputs to a single binary output as follows: 
\{0, 0\} \rightarrow 0, \{0, 1\} \rightarrow 1, \{1, 0\} \rightarrow 1 \text{ and } \{1, 1\} \rightarrow 0. \text{ To represent binary values as spike patterns, we used a similar setup to that used in Gru"ung & Sporea (2012); Seung (2003). Specifically, each binary input value was represented over the firing activity of 50 input neurons, with Poisson spike trains with a mean firing rate of 6 Hz, predetermined at the start of each simulation run; hence, paired binary input values were encoded as spike patterns over two groups of 50 input neurons. For the output a latency coding scheme was used, where the binary output values 0 and 1 corresponded to late / early output neuron spike timings of 334 ms and 167 ms, respectively. In our simulations we considered single- and multilayer networks: both networks contained 100 input neurons and a single output neuron, and the multilayer network contained 10 hidden neurons. For single-layer networks, Eq. (6.13) was applied in updating input-output weights. Paired binary inputs were presented to the network episodically in a random order. A correct classification of an input was made when an actual output spike train was closest to its target output as measured by the vRD.

From Fig. 6.6A it can be seen that the multilayer network successfully solved the XOR computation within 1000 episodes, with a final accuracy approaching 100%. The single-layer network, however, maintained an accuracy around 40% that is consistent with chance level. It is further apparent from Fig. 6.6B that the multilayer network was capable of separating the two classes, such that output spike responses for each input class matched their respective targets. In contrast, the single layer network generated erroneous output spikes in response to both input classes, which is indicative of its failure to discriminate between the two classes. Hence, these results support the necessity of including a hidden layer in a spiking network when solving the linearly non-separable XOR computation.

Multiple Input-Output Mappings. The performance as a function of the network setup was tested when tasked with mapping between an increasing number of spike pattern pairs. Specifically, the performance of three different network setups were examined: a ‘free’ multilayer network, a ‘fixed’ multilayer network and a single-layer network. Both free and fixed multilayer networks contained 10 hidden neurons and a single output neuron, but differed from each other by their restriction on hidden weight updates: a free multilayer network was allowed...
Chapter 6. Learning in Multilayer Spiking Neural Networks

Figure 6.6: Learning the XOR function of two binary variables as a temporal code. Either a single- or multilayer network structure was trained. Both networks contained $n_i = 100$ input neurons and a single output neuron. The multilayer network contained $n_h = 10$ hidden neurons. The two binary variables were encoded as predetermined spike patterns over two populations of input neurons, each of size 50. The latency of an output spike encoded for the binary value 1 (early spiking) or 0 (late spiking). (A) Evolution of the classification accuracy for single- and multilayer networks. (B) Output spike rasters for multilayer (left panel) and single-layer (right panel) networks, taken over the final 60 episodes on a typical run. Black dots correspond to responses from inputs $\{0, 1\}$ and $\{1, 0\}$, and grey dots correspond to responses from inputs $\{0, 0\}$ and $\{1, 1\}$. Target spike times are indicated by crosses for each class of input. Results were averaged over 20 independent runs.

changes in the hidden weights during learning by Eq. (6.27), while hidden weights were not allowed to change in the fixed multilayer network other than through synaptic scaling. The single-layer network lacked a hidden layer and contained a single output neuron. For a more direct comparison, both single- and multilayer networks contained 100 input neurons. The purpose of this experiment was to highlight the increase in computational capacity provided by a multilayer network with fully plastic weights in each layer.

Shown in Fig. 6.7 is the dependence of the network performance on the number of arbitrarily generated input patterns $p$, up to a maximum of 40, where each input pattern was associated with a unique target output spike (see section 6.3.4 for pattern statistics). From the left panel, it is clear that both the free multilayer and single-layer networks outperformed the fixed multilayer network over the entire range of input patterns considered; for example, after learning 40 inputs the performance values were $95.2 \pm 0.3\%$, $11.8 \pm 0.8\%$, and $0.8 \pm 0.2\%$ for free, single and fixed respectively. The performance of the fixed multilayer net-
Figure 6.7: The dependence of the performance on the number of input patterns and network setup. Each input pattern was associated with a unique target output spike. Left: The performance as a function of the number of input patterns. Right: The number of episodes to convergence in learning. The blue curve shows the performance of a multilayer network where hidden weights $w_{hi}$ are free to be updated according to Eq. (6.27), and the red curve is a multilayer network with fixed hidden weights. The green curve corresponds to a single-layer network with no hidden layer. Each network contained a single output neuron. Left panel error bars show the standard deviation, and right panel error bars show the standard error of the mean; the convergence measure was subject to high variance in most cases, therefore just the average number of episodes taken to converge in learning was considered, and not its distribution. Results were averaged over 20 independent runs.

work remained consistently low over the entire range of inputs considered, with a maximum value of $22 \pm 3\%$ for just two inputs. The fixed network essentially behaved as a single-layer network containing just 10 input layer neurons, since hidden spike patterns in response to a given input would only initially be perturbed through synaptic scaling during the course of learning. This was confirmed through further simulations of a single-layer network containing 10 input neurons, which also displayed a very low capacity.

From comparing the free multilayer and single-layer networks, it can be seen that the performance of the single-layer network was greatest for less than 12 inputs; however, for a greater number of inputs the performance of the free multilayer network dominated over the single-layer network. Over the entire range of inputs considered the performance level of the free multilayer network remained around $96\%$, and showed no indication of decreasing. Hence, the inclusion of a fully plastic hidden layer of neurons enabled all 40 patterns to be reliability encoded for
in this example. However, from close examination of the networks’ performance for a small number of inputs (less than 12 input patterns) it can be seen that the single-layer network consistently outperformed the free multilayer network; this was related to our selection of a stochastic neuron model for generating spikes in the hidden layer, which prevented the multilayer network from achieving a maximum performance of 100% due to variable spiking activity.

Shown in the right panel are the number of episodes taken for each network to converge in learning as a function of the number of input patterns. It can be seen that the convergence time for a multilayer network increased with the number of inputs, and was an order of magnitude larger for free in comparison with fixed networks when learning 40 inputs. The difference in convergence time between the free and fixed multilayer networks was attributed to the increased performance of the free multilayer network: a larger number of episodes was necessary to reach an increased performance level. The convergence time for a single-layer network decreased when learning more than 18 inputs, coinciding with a rapid drop in its performance level.

To summarise, the results of this section are supportive of multilayer over single-layer learning, and importantly when linearly non-separable classifications are performed for which the presence of a hidden layer is required. In order for single-layer networks to remain competitive with multilayer networks when mapping between a large number of spike patterns it would be necessary to scale up the input layer size, although clearly this would be disadvantageous when sparse input pattern representations are desired.

### 6.4.4 Capacity of the Multilayer Network

An important consideration when training any neural network is the maximum amount of information it can memorise. Therefore, we measured the dependence of the performance on the number of input patterns that were presented to a multilayer network, that extends the previous experiment in Fig. [6.7]. Given our implementation of a multilayer network, we also explored the dependence of the performance on the hidden layer size. Finally, the dependence of the performance on the number of target output spikes used to identify input classes was tested.
Chapter 6. Learning in Multilayer Spiking Neural Networks

The aim was to establish the relationship between the hidden layer size and the number of target output spikes supported, and how this impacted on the computational capacity of the network.

In this experiment, the network was tasked with classifying an increasing number of input patterns $p$ into $c = 10$ different classes. An equal number of input patterns were assigned to each class, and all inputs belonging to the same class were identified by a unique target output spike train containing between 1 and 10 spikes (see section 6.3.4 for a description of how target output spike trains were selected). In terms of the network setup, the network contained: 100 input neurons, either 10, 20 or 30 hidden neurons and a single output neuron as the readout.

Fig. 6.8 shows the multilayer performance as a function of the number of input patterns and the number of target output spikes $n_s$ identifying each class of input. From comparing results between the different hidden layer sizes, a larger number of hidden neurons was found to support more target output spikes at a given performance level. For example, 10 hidden neurons resulted in decreased performance when trained on more than a single output spike, for more than 60 input patterns (Fig. 6.8A), while 30 hidden neurons resulted in increased performance when trained on at least five output spikes, over the entire range of input patterns considered (Fig. 6.8C). Furthermore, from a closer inspection of Fig. 6.8 it can be seen that over a small region of input patterns $p < 50$ the network performance approached 100% when trained on multiple rather than single output spikes, which was more pronounced for a larger number of hidden neurons. To give an indicator of the network memory capacity, the maximum number of input patterns learnt at a performance level greater than 90% was around 100, 150 and 200 for 10, 20 and 30 hidden neurons respectively.

In terms of the time taken by the network to perform input classifications, the number of episodes increased with both the number of hidden neurons and number of target output spikes: taking up to 70% longer for 30 over 10 hidden neurons when trained on 200 input patterns and 10 target output spikes. A decrease in the number of episodes was generally indicative of the networks inability to learn all input patterns.

The above results correspond to a multilayer network with a fixed number of
Figure 6.8: The dependence of the network performance on the number of input patterns, the number of hidden neurons $n_h$, and the number of target output spikes $n_s$. In all cases, the network contained one output neuron. In this experiment, input patterns $p$ were equally assigned between $c = 10$ classes. Left: The performance as a function of the number of input patterns, for $n_h = 10$ (A), $n_h = 20$ (B) and $n_h = 30$ (C) hidden neurons. In each panel, different curves correspond to the $n_s$ target output spikes identifying each class of input. Right: The number of episodes to convergence in learning. Results were averaged over 20 independent runs.
Table 6.1: Scaling of the Capacity with the Network Size.

<table>
<thead>
<tr>
<th>Input neurons</th>
<th>$n_h = 10$</th>
<th>$n_h = 20$</th>
<th>$n_h = 30$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>150</td>
<td>200</td>
</tr>
<tr>
<td>200</td>
<td>220</td>
<td>320</td>
<td>360</td>
</tr>
<tr>
<td>300</td>
<td>300</td>
<td>440</td>
<td>480</td>
</tr>
<tr>
<td>500</td>
<td>460</td>
<td>680</td>
<td>740</td>
</tr>
</tbody>
</table>

Note: The maximum number of patterns is taken as the largest number of input patterns that can be accurately classified by the network at a 90% performance level. These measurements are accurate to within ±20 patterns.

input layer neurons: $n_i = 100$. Hence, we further explored the scaling of the network capacity with respect to the input layer size. As before, the multilayer network here contained either $n_h = 10, 20$ or 30 hidden layer neurons and a single output neuron as the readout. As an extension, the input layer size of the network was varied between 100 and 500 input neurons. Again, all input patterns were arbitrarily generated and equally assigned between 10 classes, where each class of input was identified by a single, unique target output spike.

Table 6.1 demonstrates an increase in the the maximum number of patterns memorised by the network as the input and hidden layer sizes are scaled up. Moreover, the maximum number of patterns has a sub-linear dependence on the number of hidden neurons; for example, a $200 \times 10$ (input $\times$ hidden) network structure outperforms a $100 \times 20$ structure. Despite this, it remains desirable that input features can be represented over an input layer of minimal size than one that is prohibitively large. Hence, hidden layer neurons can perform the task of transforming otherwise sparse input representations into a higher dimensional space for improved pattern discriminations.

From these experiments it is evident that an increase in the hidden layer size provides more capacity to the network, and is supportive of multi-spike target output trains for more reliable input classifications. These results are attributed to the internal representations of input patterns afforded by hidden layer neurons, such that class discriminations can be performed at an early stage before being
processed by the readout. Qualitatively, it was observed from spike rasters that individual hidden neurons selectively responded to certain input patterns, and only contributed to generating a fraction of the total number of target output spikes. From this, it is apparent that hidden layer neurons distribute the synaptic load between them, as was previously indicated for the experiment in Fig. 6.3 when performing single input-output mappings.

### 6.4.5 Robustness to Input Noise

The ability of a network trained under MultilayerSpiker to minimise the impact of noise on input pattern classifications was tested. In this case we considered a more realistic dataset than that used in Fig. 6.5, which consisted of several classes of input patterns subject to varying levels of noise. The network was tasked with identifying similar inputs belonging to the same class by the timings of output spikes.

We devised a synthetic data set based on that used by Mohemmed et al. (2012); specifically, the accuracy (or classification performance) of the network was tested on a generated dataset that consisted of both training and testing patterns, where the aim of the network was to learn to classify patterns into 10 classes. In generating the training patterns, a single reference spike pattern was randomly created for each class. Each of the 10 reference patterns was then duplicated 15 times, where input spikes for each duplicate were subsequently jittered according to a Gaussian distribution with a given noise amplitude that depended on the learning task. Hence, a total of 150 training patterns were generated. In the same way, 25 testing patterns were generated for each class, giving a total of 250 testing patterns. Both testing and training patterns were generated at the start of each simulation run with a selected noise amplitude, equally spaced between 2 and 20 ms, and held fixed thereafter. Patterns belonging to each class were identified by a unique target output spike train, containing between 1 and 5 spikes. On this task only training patterns were used to train the network, and testing patterns were used to test the network’s robustness to input noise. The network contained 20 hidden neurons and a single output neuron as the readout.

Shown in Fig. 6.9 is the network accuracy as a function of the noise amplitude used to initialise input patterns, and the number of target output spikes. As can
Figure 6.9: The classification accuracy on a synthetic dataset at varying noise levels. The network contained $n_h = 20$ hidden neurons and a single output neuron. The number of training patterns was 150 and the number of testing patterns 250. Both training and testing patterns were equally assigned between 10 classes. **Left:** The training and testing accuracy as a function of the amplitude of input jitter in the generated dataset, for 1 and 5 target output spikes per class. **Right:** The training and testing accuracy as a function of the number of target output spikes, at 10 ms amplitude input jitter. To ensure convergence in learning, the number of training episodes was 75000. Results were averaged over 20 independent runs.

be expected, a high degree of noise presented a greater challenge to the network, given that the network had to learn the underlying distribution of input spikes well in order to accurately classify previously unseen patterns during the testing phase. Despite this, the network still managed to classify testing patterns at least 40% of the time at 20 ms noise. Furthermore, it is clear that multiple target output spikes led to more accurate classifications in comparison with a single target output spike, giving an increase of almost 25% at 10 ms noise. From the right panel, a smooth increase in the accuracy with the number of target output spikes at 10 ms noise can be seen, along with a reduction in the standard deviation; the accuracy of one target output spike was $64 \pm 6\%$ compared with $88 \pm 2\%$ for five output spikes. However, the difference in the accuracy between single and multiple target output spikes became minimal as the noise amplitude approached 20 ms.

The multilayer network was successful in learning to classify patterns subject to noise, and especially for classifications based on multi-spike target output trains.
Two key reasons explain the increase in accuracy with the number of target output spikes. The first relates to the redundancy inherent in multi-spike based classifications: even if an output spike train cannot match its target in terms of the number of spikes generated, an accurate classification can still be made based on the precision of existing spikes with respect to their targets. The second reason comes from the larger separation between input classes as the number of target output spikes increases, such that class discriminations made by the network become less sensitive to error resulting from fluctuating output responses.

### 6.4.6 Learning Spatio-Temporal Output Patterns

MultilayerSpiker supports weight updates in a multilayer network containing more than one output neuron; therefore, we tested the performance of the network when learning to map between spatio-temporal input and output spike patterns. Here a spatio-temporal output pattern consisted of a unique target spike train at each output neuron, which taken together was used to identify each input class (see Pattern Recognition, section 6.3.5).

**Single Input-Output Mapping.** First, we considered a mapping between a single input-output spike pattern pair, where the network was tasked with learning a target spatio-temporal output pattern in response to a single, fixed input pattern. In this experiment, the network contained 20 hidden neurons and three output neurons, where each output was assigned a single, unique target spike time. For multiple output neurons, output weights were allowed to change sign during learning (see Appendix A).

Fig. 6.10 shows an example of a single simulation run, that depicts hidden (Fig. 6.10A) and output (Fig. 6.10B) neuron spike rasters towards the end of learning. Out of the 20 hidden neurons implemented in the network, three were selected for demonstrative purposes that contributed intensely to the target output timings. From Fig. 6.10A it can be seen that the selected hidden neurons generated stereotypical spike patterns, and particularly around the timings of target output spikes, where rapid firing was observed. In response to hidden layer activity, each output neuron demonstrated a successful learning of their respective target timing (Fig. 6.10B) and to a good degree of temporal accuracy, that is
Figure 6.10: Learning a mapping between a single input-output pattern pair, in a network containing multiple output neurons. The network contained $n_h = 20$ hidden neurons and $n_o = 3$ output neurons. Each output neuron was assigned a unique target spike for the input, at times: 125, 250 and 375 ms for the first, second and third output neurons, respectively. Learning took place over 1000 episodes. (A) Example hidden neuron spike rasters, and (B) output neuron spike rasters, shown over the final 60 learning episodes. Each panel in row (A) shows a hidden neuron that contributes strongly to the output neuron response shown in the panel below. In (B), each panel indicates the target spike time of an output with a cross. The left, middle and right panels show the activity of the first, second and third output neurons, respectively. (C) Heatmap of output layer weights $w_{oh}$ after 1000 learning episodes. The intensity corresponds to the strength of synaptic weights. For reference, the left, middle and right panels in (A) show the activity of hidden neuron numbers 15, 9 and 2, respectively. (D) The evolution of the vRD, averaged over 40 independent runs.
indicated by a final vRD of $0.4 \pm 0.1$ (Fig. 6.10D) with a corresponding time shift of $1.9 \pm 0.7$ ms at each output. Furthermore, the network learned to distribute the synaptic load between hidden layer neurons, such that hidden spiking activity became more diverse. This is supported by a heatmap of the output weight matrix shown in Fig. 6.10C, corresponding to the same simulation in panels A and B, which demonstrates a high degree of variance in the synaptic strength between hidden and output neurons.

**Dependence on the Hidden Layer Size.** We next explored the performance of the network when input patterns were classified by spatio-temporal output patterns. In this experiment, a total of 50 input patterns were equally assigned between 10 classes, such that all five patterns belonging to the same class were identified by a unique, target spatio-temporal output pattern. To increase the separation between classes, target output spike trains assigned to each output neuron differed from each other by a vRD of at least $n_s/2$ for $1 \leq n_s \leq 10$ output spikes, similarly as for a network containing a single output neuron. A correct input classification was made when the vRD between an actual and desired target output pattern assumed a minimum value (Methods). In measuring the relationship between the performance and network setup, an increasing fractional number $n_h/n_o$ of hidden to output neurons was implemented, for either $n_o = 10, 20$ or $30$ output layer neurons.

From Fig. 6.11A it is clear that an increase in the fractional number of hidden to output neurons increased the performance of the network, with the performance approaching 100 % for between $2 < n_h/n_o < 3$. Furthermore, there was a dependence of the performance on the number of output neurons; for example, at a fixed fractional number $n_h/n_o = 1$ the performance values were close to 48, 63 and 79 % for 10, 20 and 30 output neurons, respectively. Hence, it was apparent that a larger number of output neurons increased the separation between classes for more accurate classifications, while a sufficiently large number of hidden neurons provided capacity to the network during learning. There was a trend for lower $n_h/n_o$ needed to reach a performance level of 100 % as the number of output neurons increased.

Fig. 6.11B shows the minimum fractional number $n_h/n_o$ of hidden to output neurons needed by the network to attain 90 % performance for ten output neurons,
Figure 6.11: The dependence of the network performance on the ratio of hidden to output neurons, and the number of target output spikes. The network contained an increasing number $n_h$ of hidden neurons, and $n_o = 10, 20$ and $30$ output neurons. $p = 50$ input patterns were equally assigned between $c = 10$ classes, where all patterns belonging to the same class were identified by a unique target output spike pattern. (A) The performance as a function of the ratio of hidden to output neurons. (B) The minimum ratio of hidden to output neurons required to achieve 90% performance, as a function of the number of target spikes at each output neuron. Results were averaged over 10 independent runs.

as a function of the number of target output spikes. An increase in the minimum value of $n_h/n_o$ with the number of target output spikes was found, that showed an indication of levelling off between 8 and 10 output spikes.

An important consideration when designing any multilayer network is the hidden layer size, and whether it is sufficient to allow for reasonably accurate input classifications during learning. From the above experiments we have quantified the ratio of hidden to output neurons required by the network to allow for accurate classifications to be made, and in particular for the more general case of fully spatio-temporal, spike based pattern encodings.

### 6.4.7 Biologically Inspired Backpropagation

The technique of backpropagation is commonly associated with poor biological plausibility, an issue that has been challenged in Grünig (2007). To address this, we propose an alternative and more biologically plausible implementation of MultilayerSpiker.
Reformulation of Backpropagation. As the learning rule currently stands, output weight updates (defined by Eq. (6.13)) are considered biologically plausible, given that updates have a dependence on locally available pre- and postsynaptic activity variables at each synapse. It is, however, more realistic to effect output weight changes in an online manner, expressed by

$$
\dot{w}_{oh}(t) = \eta_o \delta_o(t) (Y_h \ast \epsilon)(t) .
$$

(6.30)

The supervisory error signals $\delta_o$ are specific to each output neuron, and it is reasonable to suppose that desired postsynaptic activity is provided by an ‘activity template’ external to the network, that is a reference target output spike pattern originating in another network (Knudsen, 1994). This idea is illustrated by the schematic shown in Fig. 6.12.

Hidden weight updates (defined by Eq. (6.27)), on the other hand, are more difficult to justify biologically. Although locality at each input-hidden synapse is satisfied via the causal, double convolution term $([Y_h(X_i \ast \epsilon)] \ast \epsilon)$, there is also a non-local dependence on a summation over hidden-specific error signals $\sum_o w_{oh} \delta_o$. It is unclear by what mechanism the strengths of output weights might be communicated back through the network, and further how these weights would then combine with specific error signals to inform synaptic weight updates. Retroaxonal signalling by the backward propagation of chemical messengers along axons has been hypothesised to contribute to learning in the nervous system (Harris, 2008), however this process would realistically take place on a time-scale many orders of magnitude slower than that required for backpropagation learning.

In order to provide a more biologically plausible implementation of backpropagation learning, it is therefore necessary to make a few heuristic assumptions regarding the network structure and synaptic plasticity. Specifically, we assume output weights are positively-valued at initialisation with the same magnitude, and are constrained to positive values during learning; hence, hidden layer weight updates all depend on output weight values with the same (positive) sign (c.f. Eq. (6.27)). Following this, we now make an approximation, and assume that all output weights will share (roughly) the same value over the entire course of learning. Therefore, the dependence of the hidden layer weight update rule on specific hidden-output weights can now be discarded, thereby providing a simpli-
Figure 6.12: Biologically inspired implementation of backpropagation in a fully-connected, feed-forward multilayer SNN. Input layer neurons are fully connected with hidden layer neurons which in turn are fully connected with two output neurons. Each output neuron must learn a prescribed target spike train in response to an input pattern presented to the network. It is posited that reference neurons external to the network are available to provide the required target output spike trains during learning. Same colour neurons have a one-one correspondence with each other, such that actual and target activity for each output neuron pair is fed into a comparator system to provide a separate error signal. These error signals are then backpropagated through the network in two stages. At the first stage, each error signal is used to inform the direction and magnitude of synaptic weight updates on its respective output neuron. At the second stage, all error signals are combined together to inform synaptic weight updates on hidden neurons. In this illustration, just two output neurons with corresponding reference outputs are used for clarity, but this setup is similarly extendable to any number of outputs.

fied hidden weight update rule,

\[
\dot{w}_{hi}(t) = \frac{\eta_h}{\Delta u_h} \left[ \sum_o \delta_o(t) \right] (\mathcal{Y}_h(\mathcal{X}_i * \epsilon) * \epsilon)(t),
\]

where the term \( \sum_o \delta_o \) is now simply a shared summation over individual output error signals. Fig. 6.12 illustrates this concept, and indicates the shared dependence of input-hidden synaptic weight updates on a combination of output error signals.
We consider the backpropagated error signals $\delta_o$ that are shared between all input-to-hidden synapses. Biologically, it is plausible that a neuromodulator might perform this function, and particularly given the evidence that neuromodulators can influence both the magnitude and direction of synaptic weight changes triggered by STDP (Seol et al., 2007). It is further known that the firing activity of dopaminergic neurons can encode a form of error signal (Schultz et al., 1997; Schultz, 2000), and for example influence corticostriatal plasticity by regulating the concentration of dopamine surrounding each synapse (Reynolds & Wickens, 2002). Based on these biological observations, previously instantaneous output error signals $\delta_o$ used for hidden and output weight updates are instead substituted with concentration-like variables $\tilde{\delta}_o$ which evolve according to:

$$\tau_D \dot{\tilde{\delta}_o}(t) = -\tilde{\delta}_o(t) + [Z_{o}^{\text{ref}}(t) - Z_o(t)],$$

(6.32)

with a decay time constant $\tau_D = 50\text{ ms}$. In the above, we have also substituted the instantaneous firing rate of an output neuron with its spike train (see Eq. (6.12)), corresponding to a concrete observation on a given run.

**Performance of Biologically Inspired Backpropagation.** The performance of the biologically inspired implementation of backpropagation (bio-backprop), defined by Eqs. (6.30) and (6.31), was compared against that of the Multilayer-Spiker rule (simply referred to as backprop here). For bio-backprop, the output error signals $\delta_o$ in Eqs. (6.30) and (6.31) were substituted with the filtered signal $\tilde{\delta}_o$ as defined by Eq. (6.32). Both the backpropagation learning rules were applied to either a multilayer network containing a single output neuron, or a multilayer network containing multiple output neurons. In both cases, networks were trained to classify input patterns by the timing of a single output spike. The single-output network contained 10 hidden neurons, and was tasked with classifying an increasing number of input patterns into 10 classes (c.f. experiment of Fig. 6.8). As shown in Fig. 6.13A, little difference was found in the performance between the bio-backprop and backprop learning rules for less than 80 input patterns. However, as the number of input patterns increased there was a small performance difference in favour of backprop, approaching 8% by 200 input patterns. In terms of its learning speed, bio-backprop was consistently
slower than backprop, taking at least 1.5 times the number of episodes needed by backprop to converge to a stable solution.

We next considered a multiple-output network containing $n_o = 10$ output neurons and an increasing number $n_h$ of hidden neurons. The network was tasked with classifying 50 input patterns into 10 classes (c.f. experiment of Fig. 6.11). From Fig. 6.13B, a marked difference in the performance favouring backprop over bio-backprop was apparent; for example, while it took just twice the fractional number of hidden-output neurons for backprop to reach a 90% performance level, it took bio-backprop at least ten times the fractional number of hidden-output neurons to reach the same performance level. In other words, bio-backprop needed five times the number of hidden neurons as backprop to accurately classify all 50 input patterns presented to the network. With respect to the convergence time, bio-backprop took almost 1.5 times as many episodes as backprop to reach a 90% performance level, using $n_h/n_o = 10$ and $n_h/n_o = 2$ for bio-backprop and backprop, respectively.

From these results, it is clear that the heuristic bio-backprop rule can maintain a
similar level of performance to the derived backprop rule for networks containing a single output neuron. However, for networks containing more than one output neuron the performance of bio-backprop lagged behind that of backprop; a reasonable performance level for bio-backprop could only be recovered by a large increase in the number of hidden neurons. Furthermore, weight distributions from associated, preliminary simulation runs indicated a reduced ability of networks trained with bio-backprop to effectively distribute synaptic weight changes between hidden layer neurons: individual hidden neurons either contributed intensely or weakly to the activity over all output neurons. Ideally, individual hidden neurons would instead be selected to contribute to specific output responses, which was evidenced for the backprop rule (c.f. Fig. 6.10C).

Despite some limitations, bio-backprop still proved to be a capable learning rule, and maintained a performance level well above the chance level of 10% in most cases. As such, bio-backprop represents an alternative to our derived backprop rule when increased biological plausibility is desired.

6.5 Discussion

This chapter has contributed a new and technically efficient learning rule for training multilayer SNNs, termed MultilayerSpiker, which has demonstrated a high performance level on several benchmark tests. The learning rule is capable of learning fully spatio-temporal input and output spike pattern transformations, and can perform input classifications to a high level of accuracy using multiple output spikes. Furthermore, an alternative and more biologically plausible implementation of MultilayerSpiker has been proposed, along with a discussion of the underlying neural mechanisms which might support such a form of backpropagation learning in the nervous system.

Here we took the approach of maximising the log-likelihood of generating a desired output spike pattern in a multilayer network through a combination of gradient ascent and backpropagation, that is an extension of the single-layer learning rule proposed by Pfister et al. (2006) to multilayer networks. In terms of the output layer, weight updates result from a product of locally available pre- and postsynaptic activity terms, that bears a resemblance to Hebbian-like learning: the
presynaptic term originates from filtered hidden neuron spike trains in the form of PSPs, and the postsynaptic term an output error signal that controls the direction and magnitude of weight changes (see Eqs. (6.12) and (6.13), respectively). Hidden layer weight updates, however, appear as a three-factor rule: PSPs due to input spikes are combined with hidden spike trains, to then be modulated by backpropagated error signals to allow hidden weight changes (see Eq. (6.27)).

In training multilayer networks to map between spike patterns, it proved necessary to represent input patterns with sufficiently rich spiking activity at each input neuron; preliminary simulations indicated that sparse representations otherwise led to decreased performance. This requirement is apparent from an examination of the hidden layer weight update rule, which has an explicit dependence on hidden neuron spike trains: a lack of input-driven hidden layer activity prevented weight updates from taking place, thereby resulting in diminished learning. Previous multilayer learning rules (Bohte, Kok, & Poutré, 2002; Sporea & Grünig, 2013) have faced a similar challenge in effectively presenting input patterns to the network, but instead took the approach of introducing multiple synaptic connections with varying conduction delays between neurons of neighbouring layers.

An important contribution is the large number of accurate pattern encodings that can be performed by MultilayerSpiker; in comparison with multilayer ReSuMe (Sporea & Grünig, 2013) for a network containing 100 input neurons, a variable number of hidden neurons and a single output neuron tasked with mapping between arbitrary input-output pattern pairs, MultilayerSpiker was capable of at least 10× as many pattern classifications at a 90% performance level but requiring less than a 1/10th the number of hidden neurons (see Table 6.1, and Table 7 in Sporea & Grünig (2013)). In addition to this, MultilayerSpiker scaled well with both the input and hidden layer sizes, and performed classifications with higher accuracy when using multi-spike based encodings. It is worth noting that most standard ANNs, for example those containing perceptron units, are fundamentally incapable of performing such temporally-based pattern mappings, and fall behind spiking networks in terms of their computational power according to Maass (1997).

We believe our classification method better takes advantage of spike-timing than many alternative methods (Gütig & Sompolinsky, 2006; Florian, 2012; Mohemmem...
for example, the Tempotron (Gütig & Sompolinsky, 2006) is limited to binary classifications using a spike / no-spike coding scheme, and the experiments run for the Chronotron (Florian, 2012) and SPAN (Mohemmed et al., 2012) required precisely matched actual and target output spikes, which would invariably be detrimental to the network performance when learning noisy input patterns given that actual output spikes would fluctuate about their respective target timings. An example of the detrimental impact of input noise on precisely-timed spike based classifications can be seen in subsection 5.4.6.

6.6 Chapter Summary

In principle, the formulation of our MultilayerSpiker rule follows from that of the learning rules previously proposed in Bohte, Kok, & Poutré (2002); Sporea & Grüning (2013), which have adapted backpropagation for use in multilayer spiking networks. Through several benchmark tests, this contribution has indicated the high performance of MultilayerSpiker, thereby lending support to its practical deployment as an efficient neural classifier. We have also highlighted the advantages of using a fully temporal code based on multiple output spike timings to reliably encode for input patterns, and characterised the dependence of the network performance on the output layer size. Finally, to address the biological shortcomings of backpropagation learning, we presented a heuristic reformulation of our learning rule which we argue can be considered more biologically plausible.
Chapter 7

Conclusions

The research aim of this thesis has been to investigate biologically-inspired learning rules for SNNs that are both general purpose in terms of their implementation and provide high performance by making full use of spike-based computations.

The motivation for developing new learning rules for SNNs is clear, given the potential increase in computational power offered by neural networks composed of spiking rather than rate-based neurons (Maass, 1997). Furthermore, it is becoming increasingly well known that biological neurons utilise a temporal code, based on precise spike timings, to more efficiently transmit information for processing in the nervous system (van Rullen et al., 2005). It is likely that such a temporal coding scheme benefits SNNs by enabling input features to be more rapidly processed in comparison with rate-based codes. Despite these advantages, few learning rules for SNNs currently exist which properly make use of a fully temporal coding scheme, and fewer still that are applicable to more complex network structures such as those containing hidden layers of spiking neurons.

This thesis has aimed to address these identified shortcomings by setting out several specific objectives, including: combining fully temporal coding with spike-based learning; ensuring derived rules are analytically rigorous for optimality purposes; formulating generalised spike-based learning rules applicable to multilayer network structures; and by making every effort to satisfy biological constraints of the actual nervous system in order to support a derived rule’s use as a model of neurobiological processing.
7.1 Summary

The work of this research is now summarised, ordered by chapter, and concluded in relation to the stated objectives of this thesis.

Chapter 2 provided background information on a typical biological neuron as found in the nervous system. Two key spiking neuron models for describing the functions of a biological neuron were then reviewed: the LIF model and SRM. These models formalise action potentials emitted by a neuron as identical stereotyped events, or ‘spikes’, for the purpose of computational efficiency. Despite some simplifications, these spiking neuron models still retain a sufficient level of biological plausibility and are much better suited to analytical treatment than, for example, the complex Hodgkin-Huxley model. This chapter also reviewed the escape noise neuron as an ideal model for incorporating background noise into neural simulations. The escape noise model has also been used extensively throughout the contribution chapters of this thesis to rigorously derive new spike-based learning rules. Furthermore, two main SNN architectures were discussed: feed-forward and recurrent network structures. SNNs that are feed-forward in their structure were selected for the purposes of mathematical modelling in this thesis, owing to their analytical tractability. Finally, two fundamental neural coding mechanisms identified in the nervous system have been reviewed: rate- and temporal-based coding. The encoding of information by the precise timings of individual spikes rather than their frequency has been indicated to be advantageous for rapid neural processing, therefore motivating our preference of temporal-over rate-based learning in SNNs.

Chapter 3 reviewed existing unsupervised, reinforcement and supervised learning rules for training SNNs to perform statistical associations or specific transformations between input and output spike patterns through synaptic weight modifications. In many cases these learning rules were inspired by the biological process of STDP identified in the nervous system, where correlations between pre- and postsynaptic spiking activity give rise to persistent change in the synaptic strength. By itself, STDP has been demonstrated to be capable of driving competitive unsupervised learning of spike patterns in SNNs, such that patterns can be detected using a temporal code; despite this, unsupervised STDP is incapable of learning specific responses to spike patterns. With respect to reinforcement-
based learning in SNNs, a generic class of reward-modulated synaptic plasticity rules was examined, capable of high biological realism by relying on minimal instruction during learning and by describing a form of modulated STDP as found experimentally in the corticostriatal junction (Pawlak & Kerr, 2008). However, a limitation of reward-modulated learning in SNNs comes from a current lack of methods combining it with a fully temporal coding scheme. With respect to supervised learning in SNNs, several relevant spike-based rules were reviewed, most of which worked to either minimise a predefined error function or rely on an adaptation of the Widrow-Hoff rule. Importantly, this chapter highlighted a statistical supervised method proposed by Pfister et al. (2006) as well suited to learning temporally precise sequences of output spikes, by virtue of its strong theoretical justification and extendibility to diverse SNNs structures. The statistical framework of this method has been used as the basis for developing the spike-based learning rules contributed by this thesis.

Chapter 4 presented a new reinforcement-based method for SNNs, trained by reward-modulated synaptic plasticity, to learn temporally precise sequences of output spikes in response to a spatio-temporal input pattern. For heightened biological realism, only delayed reward signals were used to provide general feedback to the network during learning. This chapter implemented the R-max rule to train the network, that is an analytically rigorous method based on a reward-maximisation principle, and applied two specific escape rate functions to drive variable output spiking activity: referred to as A&C and EXP. The A&C model is functionally more complex than EXP, but was shown through simulations to result in higher performance when learning large numbers of target output spikes. This contribution of a multi-spike, reinforcement-based learning method satisfies several of the thesis objectives: concerning temporal coding as a means to efficiently represent information in a biologically realistic setting.

Chapter 5 presented two new supervised learning rules for training SNNs to learn associations between spatio-temporal input patterns and temporally precise target output spike trains. The two rules are referred to as INST and FILT, which rely on the instantaneous or filtered error between a target and actual output spike train to inform synaptic weight updates, respectively. These rules were thoroughly analysed with respect to the change in a synaptic weight arising from the relative timing between an input and output spike, when a network was tasked
Chapter 7. Conclusions

with learning a single target output spike. This analysis predicted FILT as being close to optimal, given its ability to accurately shift actual output spikes towards their respective targets under most conditions, whereas INST was predicted to be suboptimal by potentially resulting in oscillatory output spiking with respect to target timings. The performance of the INST and FILT rules were subsequently tested through simulations, and benchmarked against that of the technically efficient E-learning CHRON rule (Florian, 2012). From this comparison, FILT was found to match CHRON with respect to its high temporal accuracy and rapid convergence speed. By contrast, INST fell short of both FILT and CHRON when increased temporal precision of output spikes was required. Taken together, FILT achieves most of the stated objectives of this thesis: it is predicted to be an optimal spike-based learning rule; it is fully compatible with temporal coding by learning spike timings with very high precision; and it still retains a reasonable degree of biological plausibility by its possible implementation as an online rule.

Chapter 6 presented a new supervised multilayer learning rule, termed MultilayerSpiker, applicable to feed-forward SNNs containing hidden layers of spiking neurons. The rule was formulated based on a maximum likelihood approach using SRM neurons with escape noise, and by combining the standard techniques of gradient ascent and error-backpropagation; in this way, MultilayerSpiker can be considered a generalisation of the learning method proposed by Pfister et al. (2006) from single- to multilayer SNNs structures. Extensive simulations were run to test the performance of MultilayerSpiker on several classification tasks, where it was demonstrated that the rule was capable of solving the linearly non-separable XOR task, and could learn to accurately classify spike patterns, based on the precise timings of output spikes, with or without input noise. Additionally, MultilayerSpiker allowed for a very high network capacity with respect to the maximum number of input pattern encodings it could learn to memorise, comparing favourably with existing work. Importantly, MultilayerSpiker resulted in increased network performance when a synthetic dataset was classified based on the timings of multiple, rather than single, output spikes. Finally, a more biologically plausible implementation of MultilayerSpiker was formulated, such that hidden layer weight updates instead relied on an identical, shared back-propagated output error signal that more closely approximates the function of a global neuromodulatory signal in the nervous system such as dopamine. In
Chapter 7. Conclusions

In conclusion, MultilayerSpiker fulfils the stated thesis objectives: the rule takes maximum advantage of a fully temporal coding scheme; the rule’s optimality has been guaranteed analytically, following its formulation based on a maximum likelihood approach; the rule is more general purpose than most existing methods given its application to complex SNN structures consisting of multiple layers; and the rule has an alternative implementation which better respects the biological constraints of the nervous system.

7.2 Thesis Contributions

This thesis has made several novel contributions to the area of theoretical neuroscience, regarding reinforcement- and supervised-based learning methods for SNNs. These research contributions and their significance are summarised as follows:

- Proposal of a reward-modulated synaptic plasticity method for learning temporally precise sequences of output spikes in an SNN. This method combines high biological realism with a fully temporal coding scheme to better realise the potential in computational power offered by SNNs. The R-max rule, combined with the A&C model as a stochastic neuronal spike generator, is capable of accurately learning large numbers of output spikes, outperforming existing methods in Farries & Fairhall (2007); Frémaux et al. (2010). Results based on this method have been published in Gardner & Grüning (2013) and Gardner et al. (2014).

- Presentation of two analytically derived supervised rules for SNNs, termed INST and FILT, that are applicable to learning spatio-temporal input-output spike pattern transformations. Simulations have combined these learning rules with a temporal code to form rapid, spike-based representations of input patterns. The FILT rule relies on a filtered output error signal to inform weight updates, thereby matching the high-performance E-learning CHRON rule proposed by Florian (2012). Preliminary results leading to this contribution have been published in Gardner & Grüning (2014). A manuscript of this contribution is to be submitted for publication as a journal article.
• Presentation of an analytically derived supervised learning rule for feed-forward SNNs containing hidden layers of spiking neurons. This rule is termed MultilayerSpiker, and generalises the single-layer learning method proposed by Pfister et al. (2006) to multilayer SNN structures. Simulations testing MultilayerSpiker on classification tasks have demonstrated its high performance when input patterns are encoded as multiple and precisely-timed output spikes. This work progresses existing approaches to applying error-backpropagation to multilayer SNNs (Bohte, Kok, & Poutré, 2002; McKennoch et al., 2009; Sporea & Grüning, 2013), resulting in a final rule which has strong theoretical justification, is capable of learning a very large number of input-output spike transformations and makes full use of a multi-spike based code to accurately classify input patterns. This contribution has been accepted for publication as a journal article in Gardner et al. (2015).

7.3 Future Work

Several spike-based learning methods have been proposed by this thesis, all suited to training SNNs to perform input-output spike pattern transformations. During the course of this research, however, potential extensions or applications of the methods for future work have been identified. These ideas for future investigation are discussed in the following subsections.

7.3.1 Implementation in Neuromorphic Hardware

In most of the analysis of this thesis, proposed learning methods were applied to relatively small network sizes in order to establish their performance. Although this is appropriate as a proof-of-concept for newly derived methods, it would be more biologically relevant to test the performance of a method as applied to much larger network sizes: containing on the order of $10^4$ synapses per neuron as is typical in the nervous system. Practically, this could well be achieved via implementation in neuromorphic hardware, such as the massively-parallel computing architecture of SpiNNaker (Furber et al., 2014). As a starting point, the simplistic INST rule presented in chapter 5 could be implemented in SpiNNaker, representing an achievable, and exciting, aim for future work.
7.3.2 Population-based Neural Processing

Aside from the work of chapter 6, this thesis has considered SNNs containing a single output neuron for simplicity reasons. Biologically, however, it is well known that populations of output neurons are receptive to shared patterns of input activity, such that the detrimental impact of synaptic noise on neural processing is minimised (Faisal et al., 2008). Previous studies have examined population-based learning in SNNs, with the key result that the speed of learning increases with the population size (Urbanczik & Senn, 2009; Friedrich et al., 2010, 2011). These studies were devised using a spike / no-spike or latency code to perform binary classifications of input patterns, therefore it would be interesting to extend this approach to using a fully temporal code for rapid multi-class classifications.

7.3.3 Biologically Inspired Backpropagation

The statistical framework in which the MultilayerSpiker rule was developed is general, and has successfully been applied in establishing learning rules in the areas of both supervised and reinforcement learning for diverse SNN structures (Friedrich et al., 2011; Brea et al., 2013; Rezende & Gerstner, 2014). It might therefore be assumed the multilayer learning rule has a reinforcement analogue that instead uses a delayed feedback signal to indicate the overall ‘correctness’ of network responses during learning. In Grüning (2007) it has been shown how backpropagation can be reimplemented as a cognitively more plausible reinforcement learning scheme, but for rate-coded neural networks; future work could attempt to relate such a technique to our own rule for SNNs, with the intent of supporting a more biologically plausible backpropagation rule.

7.3.4 Further Investigation

In formulating the spike-based learning methods contributed by this thesis it was necessary to use the SRM neuron combined with escape noise for analytical tractability; ideally, however, it would be preferential to propose optimal spike-based learning methods that are also neuron model independent. Therefore, an interesting direction for future work could investigate establishing an adaptation
of the MultilayerSpiker rule that is compatible with any choice of neuron model for the purposes of increased generality and biological plausibility.
Publications

Publications by the author during the period of this thesis are listed as follows.

Journal


Conferences


Conference Abstracts


Appendix A

Simulation Details

In all simulations of the multilayer network of chapter 6 and unless specified otherwise, we used a fixed number $n_i = 100$ of input layer neurons and a variable number $n_h$ of hidden layer neurons. Depending on the learning task, either a single neuron or multiple neurons in the output layer determined the response of the network to presented input patterns. The simulation time step was set to $\delta t = 1$ ms; in order to avoid numerical instability resulting from this relatively large $\delta t$ value, the backpropagated error signal in Eq. (6.12) can be approximated well by $\delta_o(t) \approx \frac{1}{\Delta u_o} \left[ Z^\text{ref}_o(t) - Z_o(t) \right]$, that is the difference between the target and actual output spike trains of an output neuron. This approximation holds from considering small values of $\Delta u_o$, such that the output neuron effectively only fires upon reaching its formal firing threshold (see Eqs. (2.8) and (5.12)).

Multilayer Networks. In all simulations of a multilayer network, hidden layer synaptic weights were initialised by independently selecting each value from a uniform distribution over the range: $w_{hi} \in [0, 3)$, that gave rise to an initial hidden neuron firing rate of $\sim 24$ Hz. Output weights were initialised depending on the number of output neurons. During learning, hidden weights were constrained to the range: $0 \leq |w_{hi}| \leq 100$, and were free to take either positive or negative values. To increase the number of eligible synapses available to the network, and to increase the diversity of hidden neuron spiking, axonal conduction delays were introduced between the input and hidden layers (Bohte, Kok, & Poutré 2002; Izhikevich, 2006). Conduction delays were selected from a uniform distribution.
over the range $d_{hi} \in (0, 40]$ ms and rounded to the nearest 1 ms, where $d_{hi}$ was the conduction delay between the $i^{th}$ and $h^{th}$ input and hidden neurons, respectively. Hence, a conduction delay $d_{hi}$ resulted in a PSP evoked at $h$ due to an input spike $t'_i$ with an effective time course of $e(t - t'_i - d_{hi})$ (c.f. Eq. (6.3)). Conduction delays were neglected between the hidden and output layers. The maximum delay value of 40 ms is consistent with that found in biological networks (Swadlow, 1985), and preliminary simulations using larger values gave negligible performance increases. Hidden and output layer learning rates were set to $\eta_h = 4/(n_i n_o n_s)$ and $\eta_o = 0.02/n_h$, respectively, where it was indicated through preliminary simulations that the dependence of $\eta_h$ on the number of output neurons $n_o$ and number of target output spikes $n_s$ dominated over the number of input patterns $p$. Both $\eta_h$ and $\eta_o$ had an inverse dependence on the number of their afferent synapses: $n_i$ (number of input neurons) and $n_h$ (number of hidden neurons), respectively.

**Single Outputs.** In simulations of a multilayer network with a single output neuron, initial values of output synaptic weights were all set to the same value $w_{oh} = 12/n_h$ that drove the output firing rate to $\sim 1$ Hz. Each initial value of $w_{oh}$ was identical to allow equal contributions from every hidden layer neuron at the start of learning. During learning, output weights were constrained to the range $0.01 \leq w_{oh} \leq 100$; the lower bound of 0.01 was enforced to enable hidden weight updates to keep taking place, given that updates depended on output weight values according to $\Delta w_{hi} \propto w_{oh}$ (see Eq. (6.27)). Values of $w_{oh}$ were positive and prevented from changing sign during learning; preliminary simulations indicated that constraining output weights to positive values for a single output neuron had no adverse impact on learning.

**Multiple Outputs.** In simulations of a multilayer network with multiple output neurons, output synaptic weights were initialised by independently selecting each value from a uniform distribution over the range $w_{oh} \in [0, 30/n_h)$, that drove the firing rate of each output neuron to $\sim 1$ Hz. Randomising output weights was necessary to increase the diversity between output responses, which improved learning in the initial stages of each simulation run. Output weights were constrained to the range $0 \leq |w_{oh}| \leq 100$, and were allowed to change sign during learning.
Single-Layer Networks. In simulations of a single-layer network, synaptic weights were initialised by independently selecting each value from a uniform distribution over the range $w \in [0, 1.7)$, that gave rise to an initial output firing rate of $\sim 1$ Hz. The learning rate was set to $\eta = 4/n_i$ and weights were constrained to the range $0 \leq |w_{oh}| \leq 100$, where the values of weights were allowed to change sign during learning. For a closer comparison, the model and parameter set used to generate output spikes in the single-layer network matched those used to generate output spikes in the multilayer network.
Appendix B

Performance and Convergence Measures

The classification performance of the network of chapter 6 was taken as an exponentially-weighted moving average \( \tilde{P}_c \) up to the \( n \)th episode, given by 
\[
\tilde{P}_c(n) = (1 - \lambda) \tilde{P}_c(n - 1) + \lambda P_c(n).
\]
On each episode, the performance either took a value of 
\( P_c = 100\% \) for a correct input classification or 
\( P_c = 0 \) otherwise (see Pattern Recognition, section 6.3.5). The timing parameter was taken as 
\( \lambda = \frac{2}{1 + 20p} \), which corresponds to an averaging window of \( 20p \) for a total of \( p \) input patterns.

The vRD was also taken as a moving average \( \tilde{D} \), with the same averaging window as used for \( \tilde{P}_c \). A moving average of each measure was necessary, given our choice of a stochastic neuron model that gave rise to fluctuating network responses between episodes.

In our simulations we measured the number of episodes taken for the network to converge in learning, that was defined in terms of its classification performance \( \tilde{P}_c \). Specifically, given a total of \( N \) learning episodes, we considered that learning had converged on the \( n \)th episode for the first value 
\( \tilde{P}_c(n) > 0.99 \tilde{P}_c(N) \), by which point the network performance fell within 1\% of its final value.
References


REFERENCES


REFERENCES


