Learning Spike Train to Spike Train Transformations in Multilayer Feedforward Neuronal Networks

Arunava Banerjee
Computer and Information Science and Engineering Department
University of Florida
Gainesville, FL 32611-6120
arunava@cise.ufl.edu

Abstract

We address the problem of learning temporally precise spike train to spike train transformations in multilayer feedforward networks of spiking neurons. We pursue a framework based strictly on spike timing, that is, one that avoids invoking concepts pertaining to spike rates. The proposed error functional compares the spike train emitted by the output neuron of the network to the desired spike train by way of their putative impact on a virtual postsynaptic neuron. This formulation sidesteps the need for spike alignment, which in turn leads to closed form solutions for all quantities of interest. Next, through a perturbation analysis of individual spike times and synaptic weights of the output as well as the intermediate neurons in the network, we derive the gradients of the error functional with respect to the said entities. Learning proceeds via a gradient descent mechanism that leverages these quantities. Simulation experiments demonstrate the efficacy of the proposed
learning framework. The experiments also highlight the effects of sparse input and asymmetries between synapses on excitatory and inhibitory neurons.

1 Introduction

A central question in Computational Neuroscience relates to how networks of neurons in animal brains are configured so as to generate specific spike train outputs in response to input stimuli. In such cases where the neuronal networks are not genetically hard-wired to implement the noted transformations, the networks have to, through some means, learn to generate the appropriate output spike trains. In this article, we consider this problem restricted to the domain of feedforward networks of spiking neurons in an online setting. Our overarching objective is to develop a learning rule built on a framework that is strictly spike timing based, that is, one that does not invoke concepts pertaining to spike rates.

We assume that the stimulus has been mapped (via a fixed mapping) to an input spike train. This input spike train is to be transformed into a desired output spike train using a spiking neuron network. Our goal is to derive a synaptic weight update rule that when applied to the neurons in the network, brings the output spike train of the network into alignment with the desired spike train. While our objective does not explicitly call for biological plausibility, we do refrain from appealing to computations that we believe would be difficult to implement in neuronal hardware. In addition, we do not address the issue of what the desired output spike train in response to an input spike train is, and how it is generated. We simply assume that such a spike train exists, and that the network learning the transformation has access to it. Finally, we do not address the question of whether the network has the intrinsic capacity to implement the input/output mapping; we undertake to learn the mapping without regard to whether or not the network, for some settings of its synaptic weights, can instantiate
the input/output transformation. There is, at the current time, little understanding of what transformations feedforward networks of a given depth/size and of a given spiking neuron model can implement, although some initial progress has been made in [1].

2 Background

The spike train to spike train transformation learning problem, as described above, has been a question of active interest for some time. Variants of the problem have been analyzed and significant progress has been achieved over the years.

One of the early results was that of the SpikeProp supervised learning rule [2]. Here a feedforward network of spiking neurons was trained to generate a desired pattern of spikes in the output neurons, in response to an input spike pattern of bounded length. The caveat was that each output neuron was constrained to spike exactly once in the prescribed time window during which the network received the input. The network was trained using gradient descent on an error function that measured the difference between the actual and the desired firing time of each output neuron. Although the rule was subsequently generalized in [3] to accommodate multiple spikes emitted by the output neurons, the error function remained a measure of the difference between the desired and the first emitted spike of each output neuron.

A subsequent advancement was achieved in the Tempotron [4]. Here, the problem was posed in a supervised learning framework where a spiking neuron was tasked to discriminate between two sets of bounded length input spike trains, by generating an output spike in the first case and remaining quiescent in the second. The tempotron learning rule implemented a gradient descent on an error function that measured the

---

1 Our goal is to demonstrate convergence for those mappings that can be learned. We evaluate this by recording the input/output spike train of a witness network and presenting that pair of spike trains to the network learning the transformation. For transformations that, in principle, lie beyond the capacity of the network to represent, the synaptic updates are, by construction, designed not to converge.
amount by which the maximum postsynaptic potential generated in the neuron, during the time the neuron received the input spike train, deviated from its firing threshold.

More recently, the ReSuMe learning rule for a single neuron was proposed in [5] based on a probabilistic model of the spiking neuron. Here, the neuron was modeled as a linear-Poisson unit, where the instantaneous output firing rate was set as a linear combination of the synaptically weighted instantaneous input firing rates. The output spike train was then modeled as a sample draw from a non-homogeneous Poisson process with intensity equal to the variable output rate. The authors implemented the Widrow-Hoff learning rule for linear units and then skillfully replaced the rates with spike trains. Although the rule was subsequently generalized to multilayered networks in [6], the linearity of the neuron model is at odds with the proposed generalization.

A bird’s eye view brings into focus the common thread that runs through all of the above approaches. In all cases there are three quantities at play: the error function $E(\cdot)$, the output $O$ of the neuron, and the weight $W$ assigned to a synapse. In each case, the authors have found a scalar quantity $\tilde{O}$ that stands-in for the real output spike train $O$: the timing of the only/first spike in [2, 3], the maximum postsynaptic potential in the prescribed window in [4], and the current instantaneous firing rate in [5, 6]. This has allowed each proposed solution to compute $\partial E/\partial \tilde{O}$ and $\partial \tilde{O}/\partial W$, quantities that are essential to implementing a gradient descent on $E$ with respect to $W$.

Viewed from this perspective, the immediate question becomes why not address $O$ directly instead of its surrogate $\tilde{O}$? After all, $O$ is merely a vector of output spike times. Upon reflection two major impediments emerge. Firstly, $O$, although a vector, can be potentially unbounded in length. Secondly, and this is the more difficult problem to

---

2When the constituent units are linear, any multilayered network can be reduced to a single layer network. Computational capacity-wise multilayered networks of linear units are therefore no more powerful than single layer networks. This also emerges in the model in [6] where the synaptic weights of the intermediate layer neurons act merely as multiplicative factors on the synaptic weights of the output neuron.
overcome, letting $O$ be a vector requires that $E(\cdot)$ compare the vector $O$ to the desired vector of spike times, and return a measure of disparity. This can potentially involve aligning the output to the desired spike train which not only makes differentiating $E(\cdot)$ difficult, but also strains biological plausibility.

We overcome these issues in turn. We first turn to the neuron model and resolve the first problem. We then propose a closed form differentiable error functional $E(\cdot)$ that circumvents the need to align spikes. Finally, we conduct a perturbation analysis of individual spike times and synaptic weights of the output as well as all intermediate neurons in the network. We derive the gradients of the error functional with respect to all output and intermediate layer neuron spike times and synaptic weights, and learning proceeds via a gradient descent mechanism that leverages these quantities. The perturbation analysis is of independent interest, in the sense that it can be paired with other suitable differentiable error functionals to generate new learning rules. The overall focus on individual spike times, both in the error functional as well as in the perturbation analysis, has the added benefit that it sidesteps any assumptions of linearity in the neuron model or rate in the spike trains, thereby affording us a learning rule for multilayered networks that is theoretically concordant with the nonlinear dynamics of the spiking neuron.

## 3 Model of the Neuron

The approach presented in this article applies to a general setup where the membrane potential function of a neuron can be expressed as a sum of multiple weighted $n$-ary functions of spike times, for varying values of $n$ (modeling the interactive effects of spikes), where gradients of the said functions can be computed. However, since the solution to the general setup involves the same set of conceptual underpinnings, for the sake of clarity we use a model of the neuron whose membrane potential function is additively separable (i.e., $n = 1$). The Spike Response Model (SRM), introduced in
is one such model. Although simple, the SRM has been shown to be fairly versatile and accurate at modeling real biological neurons [9]. The membrane potential, $P$, of the neuron, at the present time is given by

$$P = \sum_{i \in \Gamma} w_i \sum_{j \in F_i} \xi_i(t_{i,j}^I - d_i) + \sum_{k \in F} \eta(t_k^O)$$

where $\Gamma$ is the set of synapses, $w_i$ is the weight of synapse $i$, $\xi_i$ is the prototypical postsynaptic potential (PSP) elicited by a spike at synapse $i$, $d_i$ is the synaptic delay, $t_{i,j}^I - d_i$ is the time elapsed since the arrival of the $j^{th}$ most recent afferent (incoming) spike at synapse $i$, and $F_i$ is the potentially infinite set of past spikes at synapse $i$. Likewise, $\eta$ is the prototypical after-hyperpolarizing potential (AHP) elicited by an efferent (outgoing) spike of the neuron, $t_k^O$ is the time elapsed since the departure of the $k^{th}$ most recent efferent spike, and $F$ is the potentially infinite set of past efferent spikes of the neuron. The neuron generated a spike whenever $P$ crosses the threshold $\Theta$ from below.

We make two additional assumptions: (i) the neuron has an absolute refractory period that prohibits it from generating consecutive spikes closer than a given bound $r$, and (ii) all input and output spikes that have aged past a given bound $\Upsilon$ have no impact on the present membrane potential of the neuron.

The biological underpinnings of assumption (i) are well known. Assumption (ii) is motivated by the following observations. It is generally accepted that all PSPs and AHPs after an initial rise or fall, decay exponentially fast to the resting potential. This, in conjunction with the existence of an absolute refractory period, implies that for any given $\epsilon$ however small, there exists an $\Upsilon$ such that the sum total effect of all spikes that have aged past $\Upsilon$ can be bounded above by $\epsilon$ (see [10]). Finally, observing that the biological neuron is a finite precision device, we arrive at assumption (ii). The import of the assumptions is that the size of $F_i$ and $F$ can now be bounded above by $\lceil \Upsilon / r \rceil$. In essence, one has to merely look at a bounded past to compute the present membrane
potential of the neuron, and moreover, there are only finitely many efferent and afferent spikes in this bounded past. It helps to conceptualize the state of a network of neurons as depicted in Figure 1. The future spike trains generated by the neurons in the network depend only on the future input spikes and the spikes of all neurons in the bounded window $[0, \Upsilon]$.

We make one final observation. Since our objective is to update the synaptic weights in an online fashion, successive spikes on the same synapse can have potentially different weights (assigned to the spike at the time of its arrival at the synapse). We account for this by assigning weights to spikes rather than synapses; we replace $w_i$ by $w_{i,j}$ to get

$$P = \sum_{i \in \Gamma} \sum_{j \in F_i} w_{i,j} \xi_i(t^f_{i,j} - d_i) + \sum_{k \in \mathcal{F}} \eta(t^o_k)$$

(2)
4 The Error Functional

Having appropriately truncated the output spike train to a finite length vector of spike times, we now turn our attention to the error functional. The problem, stated formally, is as follows: given two vectors of spike times, the output spike train \( \langle t_1^O, t_2^O, \ldots, t_N^O \rangle \) and the desired spike train \( \langle t_1^D, t_2^D, \ldots, t_M^D \rangle \) of potentially differing lengths, assign the pair a measure of disparity.

There have been several such measures proposed in the literature (see [12, 13, 14] for details). However, for reasons that we delineate here, these measures do not fit our particular needs well. First and foremost comes the issue of temporal asymmetry. As described earlier, the effect of a spike on the potential of a neuron diminishes with age in the long run, until it ceases altogether at \( \Upsilon \). We prefer a measure of disparity that focuses its attention more on the recent than the distant past. If the output and desired spike trains align well in the recent past, this is indicative of the synaptic weights being in the vicinity of their respective desired values. A measure that does not suppress disparity in the distant past will lead weight updates to overshoot. Second comes the issue of the complex relationship between a spike train and its impact on the potential of a neuron, which is the quantity of real interest. We prefer a measure that makes this relationship explicit. Finally comes the issue of the ease with which the measure can be manipulated. We prefer a measure that one can take the gradient of, in closed form. We present here a measure that possesses these qualities.

We begin with a parametrized class of non-negative valued functions with shape resembling PSPs.

\[
f_{\beta, \tau}(t) = \frac{1}{\tau} e^{-\frac{\beta}{\tau}t} e^{-\frac{t}{\tau}} \quad \text{for} \quad \beta, \tau \geq 0 \quad \text{and} \quad t > \epsilon > 0 \quad (3)
\]

The functions are simplified versions of those in [11]. Figure 2 displays these functions for various values of \( \beta \) and \( \tau \).

We set the putative impact of the vector of output spike times \( t^O = \langle t_1^O, t_2^O, \ldots, t_N^O \rangle \)
on a virtual postsynaptic neuron to be $\sum_{i=1}^{N} f_{\beta,\tau}(t_i^O)$, and likewise for the vector of desired spike times $t^D = \langle t_1^D , t_2^D , \ldots , t_M^D \rangle$. Our goal is to assess the quantity

$$E(t^D , t^O) = \int_{0}^{T} \int_{0}^{\infty} \left( \sum_{i=1}^{M} f_{\beta,\tau}(t_i^D) - \sum_{i=1}^{N} f_{\beta,\tau}(t_i^O) \right)^2 d\beta d\tau$$

(5)

There are two paths we can pursue to eliminate the dependence on the parameters $\beta, \tau$. The first is to set them to particular values. However, reasoning that it is unlikely for a presynaptic neuron to be aware of the shape of the PSPs of its postsynaptic neurons, of which there may be several with differing values of $\beta, \tau$, we follow the second path; we integrate over $\beta$ and $\tau$. Although $\beta$ can be integrated over the range $[0, \infty)$, integrating $\tau$ over the same range results in spikes at $\Upsilon$ having a fixed and finite impact on the membrane potential of the neuron. To regain control over the impact of a spike at $\Upsilon$, we integrate $\tau$ over the range $[0, T]$, for a reasonably large $T$. By setting $\Upsilon$ to be substantially larger than $T$, we can make the impact of a spike at $\Upsilon$ be arbitrarily small. We therefore have:

$$E(t^D , t^O) = \int_{0}^{T} \int_{0}^{\infty} \left( \sum_{i=1}^{M} f_{\beta,\tau}(t_i^D) - \sum_{i=1}^{N} f_{\beta,\tau}(t_i^O) \right)^2 d\beta d\tau$$

(4)
Following a series of algebraic manipulations and noting that
\[
\int_0^\infty \int_0^{\infty} \frac{1}{\tau} e^{-\frac{\beta}{\tau}} e^{-\frac{t_1}{\tau}} \times \frac{1}{\tau} e^{-\frac{\beta}{\tau}} e^{-\frac{t_2}{\tau}} d\beta d\tau = \frac{t_1 \times t_2}{(t_1 + t_2)^2} e^{-\frac{t_1+t_2}{\tau}}
\]  
we get:

\[
E(t^D, t^O) = \sum_{i,j=1}^{M,N} \frac{t^D_i \times t^D_j}{(t^D_i + t^D_j)^2} e^{-\frac{t^D_i + t^D_j}{\tau}} + \sum_{i,j=1}^{N} \frac{t^O_i \times t^O_j}{(t^O_i + t^O_j)^2} e^{-\frac{t^O_i + t^O_j}{\tau}} - 2 \sum_{i,j=1}^{M,N} \frac{t^D_i \times t^O_j}{(t^D_i + t^O_j)^2} e^{-\frac{t^D_i + t^O_j}{\tau}}
\]  
(7)

Our immediate objective is to compute the gradient of \(E(\cdot)\) with respect to the spike times in \(t^O\). In the next section we consider the other side of the issue: how synaptic weight and input spike time perturbations affect the output spike times. The two can then be combined to relate weight perturbations to \(E(\cdot)\). Computing the gradient of \(E(\cdot)\) in Eq[7], we get:

\[
\frac{\partial E}{\partial t^O_i} = \frac{2}{\tau} \left( \sum_{j=1}^{N} \frac{t^O_j ((t^O_j - t^O_i) - \frac{t^O_j}{\tau} (t^O_j + t^O_i)) e^{-\frac{t^O_i + t^O_j}{\tau}}}{(t^O_j + t^O_i)^3} - \sum_{j=1}^{M} \frac{t^D_j ((t^D_j - t^O_i) - \frac{t^D_j}{\tau} (t^D_j + t^O_i)) e^{-\frac{t^D_j + t^O_i}{\tau}}}{(t^D_j + t^O_i)^3} \right)
\]  
(8)

5 Perturbation Analysis

We now turn our attention to how perturbations in the weights and times of the input spikes of a neuron translate to perturbations in the times of its output spikes.\(^3\) The following analysis applies to any neuron in the network, be it an output or an intermediate

\(^3\)We remind the reader that weights are assigned to spikes and not just to synapses to account for the online nature of synaptic weight updates.
layer neuron. However, we continue to refer to the input and output spike times as $t_{i,j}^I$ and $t_{k}^O$ to keep the nomenclature simple. We make one further change in nomenclature; we shift the function $\xi_i$ to the right so as to include the synaptic delay. By so doing, we are relieved of making repeated reference to the delay, thereby making the derivation easier to follow. To be more precise, what was previously $\xi_i(t_{i,j}^I - d_i)$ will now be $\xi_i(t_{i,j}^I)$, with the new shifted $\xi_i$ satisfying $\xi_i(t) = 0$ for $t < d_i$. The AHP $\eta$ will remain as before satisfying $\eta(t) = 0$ for $t < 0$.

Consider now the state of the neuron at the time of the generation of output spike $t_{l}^O$. Based on the present spike configuration, we can write

\[
\tilde{\Theta} = \sum_{i \in \Gamma} \sum_{j \in \mathcal{F}_i} w_{i,j} \xi_i(t_{i,j}^I - t_{l}^O) + \sum_{k \in \mathcal{F}} \eta(t_{k}^O - t_{l}^O) \tag{9}
\]

Note that following the definitions above, $\xi_i$ returns the value 0 for all $t_{i,j}^I < t_{l}^O + d_i$. Likewise $\eta$ returns the value 0 for all $t_{k}^O < t_{l}^O$. In other words, we do not have to explicitly exclude input/output spikes that were generated after $t_{l}^O$. Note also that we have replaced $\Theta$ with $\tilde{\Theta}$. This reflects the fact that we are missing the effects of all spikes that at the time of the generation of $t_{l}^O$ had values less that $\Upsilon$ but are currently aged beyond that bound. Since these are not quantities that we propose to perturb, their effect on the potential can be considered a constant.

Had the various quantities in Eq 9 been perturbed in the past, we would have

\[
\tilde{\Theta} = \sum_{i \in \Gamma} \sum_{j \in \mathcal{F}_i} (w_{i,j} + \Delta w_{i,j}) \xi_i(t_{i,j}^I + \Delta t_{i,j}^I - t_{l}^O - \Delta t_{l}^O) + \sum_{k \in \mathcal{F}} \eta(t_{k}^O + \Delta t_{k}^O - t_{l}^O - \Delta t_{l}^O) \tag{10}
\]

Combining Eq 9 and Eq 10 and using a first order Taylor approximation, we get:

\[
\Delta t_{l}^O = \frac{\sum_{i \in \Gamma} \sum_{j \in \mathcal{F}_i} \Delta w_{i,j} \xi_i(t_{i,j}^I - t_{l}^O) + \sum_{i \in \Gamma} \sum_{j \in \mathcal{F}_i} w_{i,j} \frac{\partial \xi_i}{\partial t_i}(t_{i,j}^I - t_{l}^O) \Delta t_{i,j}^I + \sum_{k \in \mathcal{F}} \frac{\partial \eta}{\partial t_i}(t_{k}^O - t_{l}^O) \Delta t_{k}^O}{\sum_{i \in \Gamma} \sum_{j \in \mathcal{F}_i} w_{i,j} \frac{\partial \xi_i}{\partial t_i}(t_{i,j}^I - t_{l}^O) + \sum_{k \in \mathcal{F}} \frac{\partial \eta}{\partial t_i}(t_{k}^O - t_{l}^O)}
\]
We can now derive the final set of quantities of interest from Eq 11:

\[ \frac{\partial t^O_i}{\partial w_{i,j}} = \frac{\xi_i(t^I_{i,j} - t^O_i) + \sum_{k \in F} \frac{\partial n_i}{\partial t}(t^O_k - t^O_i) \frac{\partial t^O_k}{\partial w_{i,j}}}{\sum_{i \in \Gamma} \sum_{j \in F} i w_{i,j} \frac{\partial \xi_i}{\partial t}(t^I_{i,j} - t^O_i) + \sum_{k \in F} \frac{\partial n_i}{\partial t}(t^O_k - t^O_i)} \]  

(12)

and

\[ \frac{\partial t^O_i}{\partial t^I_{i,j}} = \frac{w_{i,j} \frac{\partial \xi_i}{\partial t}(t^I_{i,j} - t^O_i) + \sum_{k \in F} \frac{\partial n_i}{\partial t}(t^O_k - t^O_i) \frac{\partial t^O_k}{\partial t^I_{i,j}}}{\sum_{i \in \Gamma} \sum_{j \in F} w_{i,j} \frac{\partial \xi_i}{\partial t}(t^I_{i,j} - t^O_i) + \sum_{k \in F} \frac{\partial n_i}{\partial t}(t^O_k - t^O_i)} \]  

(13)

The first term in the numerator of Eq 12 and Eq 13 corresponds to the direct effect of a perturbation. The second term corresponds to the indirect effect through perturbations in earlier output spikes. The equations are a natural fit for an online framework since the effects on earlier output spikes have previously been computed.

6 Learning via Gradient Descent

We now have all the ingredients necessary to propose a gradient descent based learning mechanism. Stated informally, neurons in all layers update their weights proportional to the negative of the gradient of the error functional. In what follows, we specify the update for an output layer neuron and an intermediate layer neuron that lies one level below the output layer. The generalization to deeper intermediate layer neurons follows along similar lines.

6.1 Synaptic weight update for an output layer neuron

In this case we would like to institute the gradient descent update \( w_{i,j} \leftarrow w_{i,j} - \mu \frac{\partial E}{\partial w_{i,j}} \), where \( \mu \) is the learning rate. However, since the \( w_{i,j} \)'s belong to input spikes in the past,
this would require us to reach back into the past to make the necessary change. Instead, we institute a delayed update where the present weight at synapse $i$ is updated to reflect the combined contributions from the finitely many past input spikes in $F_i$. Formally,

$$w_i \leftarrow w_i - \sum_{j \in F_i} \mu \frac{\partial E}{\partial w_{i,j}}$$  \hspace{1cm} (14)

The updated weight is assigned to the subsequent spike at the time of its arrival at the synapse. $\frac{\partial E}{\partial w_{i,j}}$ is computed using the chain rule, with the constituent parts drawn from Eq 8 and Eq 12 summed over the finitely many output spikes in $F$:

$$\frac{\partial E}{\partial w_{i,j}} = \sum_{k \in F} \frac{\partial E}{\partial t^O_k} \frac{\partial t^O_k}{\partial w_{i,j}}$$  \hspace{1cm} (15)

### 6.2 Synaptic weight update for an intermediate layer neuron

The update to a synaptic weight on an intermediate layer neuron follows along identical lines to Eq 14 and Eq 15 with indices $\langle i, j \rangle$ replaced by $\langle g, h \rangle$. The computation of $\frac{\partial t^O_k}{\partial w_{g,h}}$, the partial derivative of the $k^{th}$ output spike time of the output layer neuron with respect to the weight on the $h^{th}$ input spike on synapse $g$ of the intermediate layer neuron, is as follows. To keep the nomenclature simple, we assume that the $j^{th}$ output spike of the intermediate layer neuron, $t^H_{i,j}$, is the same as the $i^{th}$ input spike at the $i^{th}$ synapse of the output layer neuron. Then, applying the chain rule we have:

$$\frac{\partial t^O_k}{\partial w_{g,h}} = \sum_{j \in F_i} \frac{\partial t^O_k}{\partial t^H_{i,j}} \frac{\partial t^H_{i,j}}{\partial w_{g,h}}$$  \hspace{1cm} (16)

with the constituent parts drawn from Eq 15 applied to the output layer neuron and Eq 12 applied to the intermediate layer neuron, summed over the finitely many output spikes of the intermediate layer neuron which are identically the input spikes in $F_i$ of the output layer neuron.
6.3 A caveat concerning finite step size

The earlier perturbation analysis is based on the assumption that infinitesimal changes in the synaptic weights or the timing of the afferent spikes of a neuron lead to infinitesimal changes in the timing of its efferent spikes. However, since the gradient descent mechanism described above takes finite, albeit small, steps, caution is warranted for situations where the step taken is inconsistent with the underlying assumption of the infinitesimal nature of the perturbations. There are two potential scenarios of concern. The first is when a spike is generated somewhere in the network due to the membrane potential just reaching threshold and then retreating. A finite perturbation in the synaptic weight or the timing of an afferent spike can lead to the disappearance of that efferent spike altogether. The perturbation analysis does account for this by causing the denominators in Eq 12 and Eq 13 to tend to zero (hence, causing the gradients to tend to infinity). The second scenario is one where a finite perturbation leads to the appearance of an efferent spike. Since there exists, in principle, an infinitesimal perturbation that does not lead to such an appearance, the perturbation analysis is unaware of this possibility. Overall, these scenarios can cause $E(\cdot)$ to rise slightly at that timestep. However, since these scenarios are only encountered infrequently, the net scheme decreases $E(\cdot)$ in the long run.

7 Experimental Validation

We begin with a brief description of the PSP and AHP functions that were used in the simulation experiments. We chose the PSP $\xi$ and the AHP $\eta$ to have the following forms (see [11] for details):

\[ \xi(t) = \frac{1}{\alpha \sqrt{t}} e^{-\beta t} e^{\frac{t}{\tau_1}} \times \mathcal{H}(t) \quad \text{and} \quad (17) \]
\[ \eta(t) = -A e^\frac{-t}{\tau_2} \times \mathcal{H}(t) \] (18)

For the PSP function, \( \alpha \) models the distance of the synapse from the soma, \( \beta \) determines the rate of rise of the PSP, and \( \tau_1 \) determines how quickly it decays. \( \alpha \) and \( \beta \) are in dimensionless units. For the AHP function, \( A \) models the maximum drop in potential after a spike, and \( \tau_2 \) controls the rate at which the AHP decays. \( \mathcal{H}(t) \) denotes the Heaviside step function: \( \mathcal{H}(t) = 1 \) for \( t > 0 \) and 0 otherwise. All model parameters other than the synaptic weights were held fixed through the experiments. In the vast majority of our experiments, we set \( \alpha = 1.5 \) for an excitatory synapse and 1.2 for an inhibitory synapse, \( \beta = 1 \), \( \tau_1 = 20 \text{ msec} \) for an excitatory synapse and 10 \text{ msec} \) for an inhibitory synapse. In all experiments, we set \( A = 1000 \) and \( \tau_2 = 1.2 \text{ msec} \). A synaptic delay \( d \) was randomly assigned to each synapse in the range \([0.4, 0.9]\) \text{ msec} \). The absolute refractory period \( r \) was set to 1 \text{ msec} and \( T \) was set to 150 \text{ msec} \). \( \Upsilon \) was set to 500 \text{ msec} which made the impact of a spike at \( \Upsilon \) on the energy functional negligible.

### 7.1 Framework for testing and evaluation

Validating the learning rule would ideally involve presentations of pairs of input/desired output spike trains with the objective being that of learning the transformation in an unspecified feedforward network of spiking neurons. Unfortunately, as observed earlier, the state of our current knowledge regarding what spike train to spike train transformations feedforward networks of particular architectures and neuron models can implement, is decidedly limited. To eliminate this confounding factor, we chose a witness based evaluation framework. Specifically, we first generated a network, with synaptic weights chosen randomly and then fixed, from the class of architecture that we wished to investigate (henceforth called the witness network). We drove the witness network with spike trains generated from a Poisson process (with different rates for different experiments) and recorded both the precise input spike train and the network’s output spike train. We then asked whether a network of the same architecture, initialized with
random synaptic weights, could learn this input/output spike train transformation using the proposed synaptic weight update rule.

We chose a conservative criterion to evaluate the performance of the learning process; we compared the evolving synaptic weights on the neurons of the learning network to the synaptic weights on the corresponding neurons of the witness network. There are several reasons why this measure is conservative. Firstly, due to the finiteness of the length of the recorded input/output spike train of the witness network, it is conceivable that there exist other witness networks that map the input to the corresponding output. If the learning network were to tend toward one of these competing witness networks, one would erroneously deduce failure in the learning process. Secondly, changing the problem of learning a spike train to spike train transformation into one of learning the synaptic weights of a network adds a degree of complexity to the problem; the quality of the learning process now depends additionally on the characteristics of the input. It is conceivable that learning is slow or fails altogether for one input spike train while it succeeds for another. In spite of these concerns, we found this the most objective and persuasive criterion.

7.2 Time of update

The synaptic weight update rule presented in the previous section does not specify a time of update. In fact, the synaptic weights of the neurons in the network can be updated at any arbitrary sequence of time points or at regular intervals. However, as demonstrated here, the specific nature of one of the constituent parts of the rule makes the update insignificantly small outside a particular window of time.

Note that \( \frac{\partial E}{\partial t_k} \), the partial derivative of the error with respect to the timing of the \( k \)th efferent spike of the output neuron, appears in the update formulas of all synapses, they on the output neuron or the intermediate neurons. We generated 10,000 random samples of pairs of vectors \( t^O = (t^O_1, t^O_2, \ldots, t^O_N) \) and \( t^D = (t^D_1, t^D_2, \ldots, t^D_M) \) with \( N \) and \( M \) chosen independently and randomly from the range \([1, 10]\) and the individual
spike times chosen randomly from the range \([0, \Upsilon]\). As noted earlier, \(\Upsilon\) and \(\Upsilon\) were set to 500 and 150 \text{msec}, respectively. We computed \(\frac{\partial E}{\partial t_k}\) for the individual spikes in each \(t^O\) according to Eq \(8\). Figure 3 presents a scatter plot in log-scale of the absolute value of \(\frac{\partial E}{\partial t_k}\) plotted against \(t^O_k\), for the entire dataset. As is clear from the plot, \(|\frac{\partial E}{\partial t_k}|\) drops sharply with \(t^O_k\). Hence, the only time period during which the gradient update formulas can have significant values is when at least one \(t^O_k\) is small, that is, immediately after the generation of a spike by the output neuron. The symmetric nature of Eq \(8\) would indicate that this is also true for the timing of the desired spikes. We therefore chose to make synaptic updates to the entire network soon after the generation of a spike by the output neuron or the stipulation of a desired spike at the output neuron.

7.3 Learning results for various architectures

The synaptic weight update rule was applied to numerous examples from single layer and two layer networks. In each case, we tracked the evolving synaptic weights on each neuron in the learning network. The disparity between the synaptic weights on a neuron in the learning network and its corresponding neuron in the witness network
was quantified using the mean absolute percentage error (MAPE): the absolute value of the difference between a synaptic weight and the “correct” weight specified by the witness network, normalized by the “correct” weight, averaged over all synapses on that neuron. A MAPE of 1.0 in the plots corresponds to 100%.

### 7.3.1 Single layer network: base case

In our first set of experiments, we addressed the simplest scenario of learning in a single layer network: an output neuron receiving input on a single synapse. We randomly generated witness networks with synaptic weights in a range that caused the neuron to spike at approximately 10 Hz when driven by a 10 Hz Poisson spike train. The precise input and output spike trains were then recorded and utilized to learn the synaptic weight using the update rule.

We discovered that the error functional is very sensitive to an almost aligned pair of output and desired spikes early in the time window, when the remaining spikes are far from aligned. Whenever this scenario was encountered, gradient descent made large
updates and occasionally overshot, regardless of how low the learning rate was set. We applied a simple fix that resolved this issue. In addition to the learning rate, we set a second parameter that capped the length of the update vector. This eliminated the noted overshoots, as shown in the contrasting graphs in Figure 4(a). We instituted this fix for all subsequent experiments on all architectures.

In the present scenario, in all experiments where the neuron generated spikes (thereby triggering synaptic updates), it converged to the “correct” synaptic weight. Next, we conducted experiments where we reduced the rate of the input spike train to 1 Hz. The objective was to evaluate the update rule under conditions where the notion of a spike rate was absent for the most part (note that with $\Upsilon = 500 \text{ msec}$, most windows have none or one spike). Here too, we found that the neuron converged to the “correct” synaptic weight, albeit at a slower rate. Figure 4(b) displays one such example.

7.3.2 Single layer network: general case

Next, we considered the general case of an output neuron with multiple synapses. Experiments were conducted on a neuron with 10 synapses under a variety of setups: (a) with all synapses set to excitatory, (b) with a mix of excitatory and inhibitory
synapses, and (c) with widely differing PSPs, where half of the excitatory synapses were set to \( \tau_1 = 80 \text{msec}, \beta = 5 \), and half of the inhibitory synapses were set to \( \tau_1 = 100 \text{msec}, \beta = 50 \) (modeling slower NMDA and GABA\(_B\) synapses, respectively). The input to the synapses were drawn from independent 10 Hz Poisson spike trains. Once again, we found that the synaptic weights converged to the “correct” values regardless of how distant the initial weights were (our tests spanned a reasonably large initial MAPE of 1.0). Figure 5(a) shows two such results for the most general case, (c). Finally, we ran additional experiments with inputs drawn from 1 Hz Poisson spike trains. Here too, learning was successful. Figure 5(b) shows two such results for the most general case, (c).

### 7.3.3 Two layer network: base case

In the case of a two layer network, the simplest architecture is one with an intermediate layer neuron driven by input on a single synapse with the output of the intermediate layer neuron feeding into a single synapse on an output layer neuron. As in the previous experiments, we generated several witness networks with either slow or fast synapses. For each such witness network we randomly initialized learning networks at various MAPE disparities and trained them using the update rule. The most significant insight yielded by the experiments was that the domain of convergence for the weights on the synapse pair, although fairly large, was smaller than that in single layer networks. This is akin to what is observed in learning in multilayer networks of sigmoidal neurons. Figure 6(a) shows a prototypical example of a network with fast synapses that converged to the “correct” synaptic weights. As in the previous set of experiments, we ran additional trials with sparse inputs drawn from 1 Hz Poisson spike trains. Here too, learning was successful, although the domain of convergence was found to be smaller. Figure 6(b) shows one such result for a network with slow synapses.

The simplicity of the network architecture allowed us to explore, in finer detail, the nature of the local minima present in the energy landscape. We initialized several net-
Figure 6: Two layer with 2 synapses, 1 on the intermediate neuron and 1 on the output neuron. Each curve corresponds to a single neuron. See text for more details regarding each panel.

works with synaptic weight disparities located in the even quadrants (i.e., initial weight of the synapse on the intermediate neuron lower than the “correct” weight and that of the synapse on the output neuron higher than the “correct” weight, and vice versa). For a linear-Poisson model there exists a 1-dimensional subspace of intermediate and output synaptic weight pairs that instantiate identical input-output transformations. We found this not to be the case when the precise spike train input and output are considered. Instead there exist isolated local minimas to which the networks converge. Figures (c) and (d) show two nearby initializations in the second quadrant. The first
Figure 7: Two layer with 30 synapses (5 on each of 5 intermediate neurons and 5 on the output neuron). Each curve corresponds to a single neuron. See text for more details regarding each panel.

converges to the “correct” weight pair, whereas the second with a higher initial MAPE disparity converges to a distinct local minima. Interestingly, our experiments indicated that some local minimas are fairly insensitive to the precise input spike train; we conducted experiments with different instantiations of the Poisson spike train input and found that in some cases the synapses lingered at the same weight pair local minima.
7.3.4 Two layer network: general case

In our final set of experiments, we explored a network architecture with five inputs that drove each of five intermediate neurons which in turn drove an output neuron. There were, accordingly, a sum total of 30 synapses to train, 25 on the intermediate neurons and 5 on the output neuron. As in the previous cases, we generated several witness networks with (a) all synapses set to excitatory and (b) a mix of excitatory and inhibitory synapses. In the case of (b) two of the inputs were set to inhibitory and two of the intermediate neurons were set to inhibitory.

For the networks under category (a), the results of the experiments were similar to those of the base case. Networks converged to the “correct” weights on all 30 synapses over a sizable domain of convergence for spike train inputs drawn from independent 10 Hz Poisson processes. Figure 7(a) presents a prototypical example of this scenario. The domain of convergence for sparse 2 Hz input was substantially smaller. Furthermore, we observed cases similar to that shown in Figure 7(b). In this example, the synapses on the output neuron as well as all but one intermediate neuron converged to their respective “correct” weights. The conundrum was how the synapses on the output neuron converged with a potentially different spike train arriving from the recalcitrant intermediate neuron. Upon closer inspection we discovered that two of the synapses on the intermediate neuron in question did not converge, and the sparse input on those particular synapses were seldom involved in the generation of a spike at that intermediate neuron. In essence, the output of the intermediate neuron was well aligned with the “correct” output, which is what induced its synapse on the output neuron of the network to converge. This was therefore an example of a competing witness network that modeled the same input output transformation for that sparse and finite length input spike train.

For networks under category (b), the results of the experiments exhibited a recurring feature: the synapses on the inhibitory intermediate neurons, be they excitatory or inhibitory, converged substantially slower than the other synapses in the network.
Figure 7(c) displays an example of a network that converged to the “correct” weights. Note, in particular, that the two inhibitory intermediate neurons were initialized at a lower MAPE disparity as compared to the other intermediate neurons, and that their convergence was slow. The slow convergence is clearer in the close-up in Figure 7(d). The formal reason behind this asymmetric behavior has to do with the range of values $\frac{\partial t_k^O}{\partial t_j^H}$ takes for an inhibitory intermediate neuron as opposed to an excitatory intermediate neuron, and its consequent impact on Eq 16. Observe that $\frac{\partial t_k^O}{\partial t_j^H}$, following the appropriately modified Eq 13, depends on the gradient of the PSP elicited by spike $t_j^H$ at the instant of the generation of spike $t_k^O$ at the output neuron. The larger the gradient, the greater is the value of $\frac{\partial t_k^O}{\partial t_j^H}$. Typical excitatory (inhibitory) PSPs have a short and steep rising (falling) phase followed by a prolonged and gradual falling (rising) phase. Since spikes are generated on the rising phase of inhibitory PSPs, the magnitude of $\frac{\partial t_k^O}{\partial t_j^H}$ for an inhibitory intermediate neuron is smaller than that of an excitatory intermediate neuron. A remedy to speed up convergence would be to compensate by scaling inhibitory PSPs to be large and excitatory PSPs to be small, which, incidentally, is consistent with what is found in nature.

8 Conclusions

A synaptic update mechanism that learns spike train to spike train transformations is not only of significant importance to testing forward models in theoretical neurobiology, it can also one day play a crucial role in the construction of brain machine interfaces. We have, in this article, presented such a mechanism that was formulated with a singular focus on the timing of spikes. The rule is composed of two constituent parts, a differentiable error functional that computes the spike time disparity between the output spike train of a network and the desired spike train, and a suite of perturbation rules that directs the network to make incremental changes to the synaptic weights aimed at reducing this disparity. The perturbation analysis applies to a fairly large class of
neuron models and can therefore be combined with other suitable differentiable error functionals to devise novel synaptic update rules.

The synaptic update rule was experimentally validated over a variety of network architectures. The experiments highlighted the nature of local minima in multilayer networks, the effect of sparse inputs, and asymmetries between excitatory and inhibitory neurons that are a product of the shape of typical PSPs and the conditions under which spikes are generated in neurons. An immediate question would be to consider how the rule can be specialized to (deep) convolutional networks of spiking neurons.

References

[1] Ramaswamy, V. & Banerjee, A. (2014) Connectomic Constraints on Computation in Feedforward Networks of Spiking Neurons. *Journal of Computational Neuroscience*, 37(2):209-228.

[2] Bohte, S.M., Kok, J.N., & La Poutre, H. (2002) Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48(1):17-37.

[3] Booij, O. & Nguyen, H.t. (2005) A gradient descent rule for spiking neurons emitting multiple spikes. *Information Processing Letters* 95(6):552-558.

[4] Güttig, R. & Sompolinsky, H. (2006) The tempotron: a neuron that learns spike timing-based decisions. *Nature Neuroscience* 9(3):420-428.

[5] Ponulak, F. & Kasinski, A. (2010) Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural Computation* 22(2):467-510.

[6] Sporea, I. & Grüning, A. (2013) Supervised learning in multilayer spiking neural networks. *Neural Computation* 25(2):473-509.
[7] Florian, R. V. (2012) The Chronotron: A Neuron That Learns to Fire Temporally Precise Spike Patterns. *PLoS ONE* 7(8): e40233. doi:10.1371/journal.pone.0040233

[8] Gerstner, W. (2002) *Spiking neuron models: Single neurons, populations, plasticity.* Cambridge Univ Press.

[9] Jolivet, R., Lewis, T.J. & Gerstner, W. (2004) Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy. *Journal of Neurophysiology* 92(2):959.

[10] Banerjee, A. (2001) On the Phase-Space Dynamics of Systems of Spiking Neurons. I: Model and Experiments. *Neural Computation* 13(1):161-193.

[11] MacGregor, R.J. & Lewis, E.R. (1977) *Neural modeling.* Plenum New York.

[12] Victor, J.D. & Purpura, K.P. (1996) Nature and precision of temporal coding in visual cortex: a metric-space analysis. *J Neurophysiol* 76:13101326.

[13] van Rossum M.C.W. (2001) A novel spike distance. *Neural Computation* 13:751763.

[14] Schreiber, S., Fellous, J.M., Whitmer, J.H., Tiesinga, P.H.E. & Sejnowski, T.J. (2003). A new correlation based measure of spike timing reliability. *Neurocomputing* 52:925931.