Reinforcement Learning, Spike Time Dependent Plasticity and the BCM Rule

Dorit Baras and Ron Meir
Department of Electrical Engineering
Technion, Haifa 32000, Israel
doritb@il.ibm.com, rmeir@ee.technion.ac.il

September 8, 2006

Abstract

Learning agents, whether natural or artificial, must update their internal parameters in order to improve their behavior over time. In reinforcement learning, this plasticity is influenced by an environmental signal, termed a reward, which directs the changes in appropriate directions. We apply a recently introduced policy learning algorithm from Machine Learning to networks of spiking neurons, and derive a spike time dependent plasticity rule which ensures convergence to a local optimum of the expected average reward. The approach is applicable to a broad class of neuronal models, including the Hodgkin-Huxley model. We demonstrate the effectiveness of the derived rule in several toy problems. Finally, through statistical analysis we show that the synaptic plasticity rule established is closely related to the widely used BCM rule, for which good biological evidence exists.

1 Policy Learning and Neuronal Dynamics

Reinforcement Learning (RL) is a general term used for a class of learning problems in which an agent attempts to improve its performance over time at a given task (e.g., Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998). Formally, it is the problem of mapping situations to actions in order to maximize a given reward signal. The interaction between the agent and the environment is modelled mathematically as a Partially Observable Markov Decision Process (POMDP, see Appendix C.3.2). At each (discrete) time

step t, the agent and the environment are in a particular state $x(t)$. The state determines a, possibly noisy, observation vector $y(t)$ that is seen by the agent. Upon observing $y(t)$ the agent performs an action $a(t)$, and receives a reward $r(t)$, based on the quality of its action. The mapping from states to actions is referred to as a policy. The objective of RL is to learn an optimal policy through exploring and interacting with the environment. An optimal policy is a policy that maximizes the long term average reward. Appendix C.3 provides definitions of MDPs and POMDPs. We refer the reader to (Bertsekas and Tsitsiklis, 1996; Baxter and Bartlett, 2001) for the precise mathematical definition of policy learning in general.

We begin with the simple Leaky Integrate and Fire (LIF) neuronal model (e.g., Gerstner and Kistler, 2002; Koch, 1999). A more general model neuron will be introduced in Section 3. Consider a population of N LIF neurons, each of which is characterized by a potential $v_i(t), i = 1, 2, \ldots, N$. Each neuron receives inputs from both external sources and from other neurons in the network. We assume that each LIF neuron is characterized by a reset value v_r, a leakage voltage v_L, and threshold v_θ. The dynamics of neurons *between spikes* is given by

$$\tau_m \frac{dv_i(t)}{dt} = -\big(v_i(t) - v_L\big) + RI_i(t) + RI_i^{\text{ext}}(t) \qquad (i = 1, 2, \ldots, N). \tag{1}$$

Upon reaching a threshold v_θ a spike is emitted and the neuron is reset to v_r (we neglect the refractory period in the present analysis).

The current $I_i^{\text{ext}}(t)$ denotes the external current entering neuron i, and the synaptic current $I_i(t)$ is given by

$$I_i(t) = \sum_j w_{ij} \sum_f \alpha\left(t - t_j^{(f)}\right), \tag{2}$$

where w_{ij} denotes the synaptic efficacy. The first summation in (2) is over all neurons which connect to neuron i, while the sum over f runs over the firing times of the presynaptic neurons, where $\{t_j^{(f)}\}$ denote the firing times of neuron j. Although the derivation below applies to general α functions, we specify two special and analytically tractable cases in Section 2.1.

An advantage of the simple linear dynamics (1) and (2) is that the potential value between spikes can be precisely computed, yielding

$$v_i(t) = v_r \exp\left\{-\frac{t - \hat{t}_i^{(0)}}{\tau_m}\right\} + \frac{1}{c} \int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\frac{t-s}{\tau_m}\right\} \left(I_i(s) + I_i^{\text{ext}}(t)\right) ds \ , \tag{3}$$

where $c = \tau_m / R$ and $\hat{t}_i^{(0)}$ is the last spiking time of neuron i preceding t.

The mapping of our problem to a POMDP is defined as follows (see also Bartlett and Baxter, 1999):

State The membrane potential value. We use discrete values within a fixed resolution (the voltage range was $[-60, 0]$ mV, divided into 120 discrete bins of equal size), along with a binary variable denoting a spiking event in the neuron.[1]

Observation A noisy measurement of the state. In this work the observation is taken to be the state itself, although the formalism is more general.

Actions Given the membrane potential, each neuron may fire or remain quiescent. The action, to fire or not to fire, depends on the state stochastically.

Reward The reward is provided as a real valued number, based on the underlying task. The reward process is assumed to be global, namely available to all the neurons in the network.

The dynamics described by (1) and (2), including the spiking process, is purely deterministic. In order to allow for stochasticity in action selection, required within the POMDP framework (Baxter and Bartlett, 2001), we assume that the spiking process is random. For each neuron we define a spiking point process $\zeta_i(t) \in \{0, 1\}$ through

$$\begin{aligned} \mathbf{P}\left[\zeta_i(t) = 1 | v_i(t)\right] &= \left[1 + \exp\left(-\left(\lambda v_i(t) - \varphi_i\right)\right)\right]^{-1} \\ &\triangleq \sigma\left(\lambda v_i(t) - \varphi_i\right) \end{aligned} \tag{4}$$

where $\zeta_i(t) = 1$ denotes a spike at time t, $\zeta_i(t) = 0$ denotes the absence of a spike, and λ determines the steepness of the sigmoidal function. In the sequel we use the same threshold φ for all neurons, but for generality we allow different φ_i.

When applying the direct RL approach presented in Section 2 to networks of spiking neurons, the set of synaptic weights parameterize the stochastic policy. In previous work, as well as in ours, the neurons are modelled as MDPs or POMDPs (see Appendix C.3) with the possible actions of generating or not generating a spike. The approach presented in (Bartlett and Baxter, 1999) is similar to ours, but neurons are modelled as simple

[1] This state space suffices for deriving the algorithm. In simulations presented later in this paper, the state space is larger and contains the following additional ingredients: exact spiking times of each presynaptic neuron (required for exact computation of the weight update) and a variable denoting the number of spike events that occurred in the particular neuron (required for the computation of the reward signal). We emphasize that this extension of the state space does not affect the derivation of the algorithm as appears in Section 2, and is only required for implementation of the simulations as described in Section 4

McCulloch-Pitts binary elements. In (Xie and Seung, 2004; Seung, 2003) a more realistic neural model is used, but learning is based on spiking rates rather than on the membrane potential itself, and the derivation of the algorithm requires additional statistical assumptions about the spiking events. In addition, we are able to relate the plasticity rule derived to the well-known BCM rule.

The remainder of the paper is organized as follows. Section 2 is devoted to the derivation of the basic synaptic plasticity rule in the context of LIF neurons. The framework is then extended to general neural models in Section 3. Some results demonstrating the applicability of the approach to several toy problems is presented in Section 4. A detailed statistical analysis of the derived rule is presented in Section 5, showing the close relationship to the BCM rule. We conclude with a Discussion and a presentation of several open problems.

2 Derivation of the Weight Update

In order to limit the state space and efficiently implement the algorithm, a discretization procedure is required. Discretizing the differential equation (1) we have

$$v(t+\Delta) = v(t) + \frac{1}{\tau_m}\Delta\left(v_L - v(t) + RI(t)\right), \tag{5}$$

where $I(t) = I_i(t) + I_i^{\text{ext}}(t)$. The notation $t+\Delta,\ t-\Delta$ refer to the next and previous time steps, respectively. This procedure is analogous to solving the original differential equation using Euler's method. A similar approach is taken with respect to $\zeta_i(t)$, i.e. spiking can occur with a certain probability in each discrete time step bin. Note that, as in (1), the voltage is reset after every spiking event.

We use the update rule for the parameters as suggested in the GPOMDP algorithm (Baxter and Bartlett, 2001). In general, let $\boldsymbol{\theta}$ denote the vector of parameters describing a system, and let $\mu_a(y(t), \boldsymbol{\theta})$ denote a probabilistic policy based on the observation $y(t)$ and the parameter value $\boldsymbol{\theta}$. More precisely,

$$\mu_a(y(t), \boldsymbol{\theta}) = \Pr\left\{\text{choosing action } a | y(t), \boldsymbol{\theta}\right\}.$$

The GPOMDP parameter update rule (Baxter and Bartlett, 2001) is given by

$$\boldsymbol{\theta}(t) = \boldsymbol{\theta}(t-\Delta) + \gamma r(t)\mathbf{z}(t), \tag{6}$$

where $r(t)$ is the reward signal, and the *eligibility trace*, $\mathbf{z}(t)$, is given by

$$\mathbf{z}(t+\Delta) = \beta\mathbf{z}(t) + \frac{\nabla\mu_{a(t)}\big(y(t), \boldsymbol{\theta}\big)}{\mu_{a(t)}\big(y(t), \boldsymbol{\theta}\big)}\ . \tag{7}$$

Here $\gamma > 0$ is a small step size parameter and $\beta \in [0,1)$ is a bias-variance tradeoff parameter. The convergence of the (temporal) average reward to its expected value, based on the parameter update rules (6) and (7), is established in (Baxter and Bartlett, 2001) under the appropriate technical conditions.

In the present context the parameters denote the synaptic weights $\mathbf{W} = \{w_{ij}\}_{i,j=1}^{N}$, and the actions are denoted by the firing pattern $\{\zeta_i(t)\}$. From (4)

$$\mu_{\zeta_i(t)}\big(y(t), \mathbf{W}\big) = \sigma\big((2\zeta_i(t) - 1)(\lambda v_i(t) - \varphi_i)\big),$$

where we have used $\sigma(x) = 1 - \sigma(-x)$.

We compute the gradient required in (7). Assume initially that $\zeta_i(t) = 1$. Setting $x_i(t) = \lambda(v_i(t) - \varphi_i)$, we have

$$\begin{aligned}
\frac{\partial \mu_1\big(y(t), \mathbf{W}\big)}{\partial w_{ij}} &= \sigma'(x_i(t)) \frac{\partial x_i(t)}{\partial w_{ij}} \\
&= \lambda \sigma(x_i(t))(1 - \sigma(x_i(t))) \frac{\partial v_i(t)}{\partial w_{ij}} \\
&= \sigma(x_i(t))\big(1 - \sigma(x_i(t))\big) \frac{\lambda}{c} \int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\frac{t-s}{\tau_m}\right\} \sum_f \alpha\left(s - t_j^{(f)}\right) ds \ ,
\end{aligned}$$

where we have used $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ and the explicit solution (3) in the derivation. A similar result can be established for the case $\zeta_i(t) = 0$, leading to the relation

$$\frac{\partial \mu_{\zeta_i(t)}\big(y(t), \mathbf{W}\big)/\partial w_{ij}}{\mu_{\zeta_i(t)}\big(y(t), \mathbf{W}\big)} = \big(\zeta_i(t) - \sigma(x_i(t))\big) \frac{\lambda}{c} \int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\frac{t-s}{\tau_m}\right\} \sum_f \alpha\left(s - t_j^{(f)}\right) ds.$$

Summarizing the derivation, we obtain the following update rule for the synaptic weights,

$$\begin{aligned}
w_{ij}(t) &= w_{ij}(t - \Delta) + \gamma r(t) z_{ij}(t) \ , \\
z_{ij}(t + \Delta) &= \beta z_{ij}(t) + \big(\zeta_i(t) - \sigma_i(x_i(t))\big) \frac{\lambda}{c} \int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\frac{t-s}{\tau_m}\right\} \sum_f \alpha\left(s - t_j^{(f)}\right) ds \ . \qquad (8)
\end{aligned}$$

Note that the weight update depends on the correlation between the reward and the eligibility trace z_{ij}. The eligibility trace of a weight is updated based on the spiking activity of the presynaptic and postsynaptic neurons only. The presynaptic spikes contribute only if they occurred after the last postsynaptic spike (the reset effect), and the sign of the

change depends on the event of postsynaptic firing, through the term $(\zeta_i(t) - \sigma_i(x_i(t)))$ in (8). For example, the largest positive contribution to the eligibility trace occurs in the event that both the presynaptic and the postsynaptic cells fire vigorously while a positive reward is delivered.

2.1 Two Explicit Choices for α

In order to better understand the meaning of the update rule established, we consider two special choices for the function α.

Pulse function Setting $\alpha(t) = q\delta(t)$, the integral in (8) can be computed exactly leading to the update

$$z_{ij}(t+\Delta) = \beta z_{ij}(t) + \big(\zeta_i(t) - \sigma_i(t)\big)\frac{\lambda q}{c}\sum_f{}' \exp\left\{-\frac{t - t_j^{(f)}}{\tau_m}\right\}, \tag{9}$$

where $\sum_f'$ denotes a summation over all presynaptic firing times $t_j^{(f)}$ such that $\hat{t}_i^{(0)} < t_j^{(f)} \leq t$.

Exponentially decaying window A common choice for α is

$$\alpha(t) = \frac{q}{\tau_s} e^{-s/\tau_s} I\big\{t \geq 0\big\} \ .$$

An explicit (but tedious) calculation (see Appendix C.1 for details), leads to the update

$$z_{ij}(t+\Delta) = \beta z_{ij}(t) + \big(\zeta_i(t) - \sigma_i(t)\big) F\left(\left\{t_j^{(f)}\right\}, \hat{t}_i^{(0)}\right)$$

where

$$F\left(\left\{t_j^{(f)}\right\}, \hat{t}_i^{(0)}\right) = \frac{\lambda q \tau_m}{c\,(\tau_m - \tau_s)} \sum_f{}' e^{-\left(\frac{t - t_j^{(f)}}{\tau_s}\right)} \left[\exp\left\{\left(\frac{\tau_m - \tau_s}{\tau_m \tau_s}\right)\left(\min\left(t - t_j^{(f)}, t - \hat{t}_i^{(0)}\right)\right)\right\} - 1\right].$$

3 Extensions to General Neuronal Models

The synaptic plasticity rules derived in Section 2 are based on a simple integrate and fire model, which neglects both nonlinearities and adaptation mechanisms known to take place within a neuron. We consider now a network of generalized neural elements. Let $v_i(t)$ denote the membrane potential of neuron i, and set $\mathbf{u}_i(t) = \big(u_{i,1}(t), ..., u_{i,P}(t)\big)^\top$ to be a vector of additional neural variables, describing, for example, adaptation mechanisms

neglected in the simple LIF model. The network model is given by

$$\begin{aligned}\frac{dv_i}{dt} &= F\big(v_i(t),\mathbf{u}_i(t)\big) + G\big(v_i(t),\mathbf{u}_i(t)\big)I_i(t),\\ \frac{du_{i,k}}{dt} &= H_k\big(v_i(t),\mathbf{u}_i(t)\big) \qquad (i=1,2,\ldots,N,\ k=1,2,\ldots,P)\ ,\end{aligned}$$

with appropriate reset definitions, if needed (e.g., reset $v_i(t)$ after spiking in integrate and fire type models). For example, such a model may describe many biophysically motivated models, such as the Hodgkin-Huxley model and simpler two-dimensional approximations. In general, one cannot solve the differential equation and find an explicit solution for $v_i(t)$ as was done in (3), and one has to resort to a different approach in order to determine the synaptic update rules.

Proceeding similarly to the derivation in Section 2, we find that

$$\begin{aligned}w_{ij}(t) &= w_{ij}(t-\Delta) + \gamma r(t) z_{ij}(t)\ ,\\ z_{ij}(t+\Delta) &= \beta z_{ij}(t) + \lambda\big(\zeta_i(t)-\sigma_i(t)\big)\frac{\partial v_i(t)}{\partial w_{ij}}\ .\end{aligned}$$

In order to compute $\partial v_i(t)/\partial w_{ij}$ we use an idea similar to the Real Time Recurrent Learning algorithm described in (Haykin, 1999). Discretizing time, as in Section 2 (and neglecting I^{ext} for simplicity), the neural dynamics is given by

$$\begin{aligned}v_i(t+\Delta) &= v_i(t) + \Delta\left[F\big(v_i(t),\mathbf{u}_i(t)\big) + G\big(v_i(t),\mathbf{u}_i(t)\big)\sum_j w_{ij}\sum_f \alpha_i(t-t_j^{(f)})\right]\\ u_{i,k}(t+\Delta) &= u_{i,k}(t) + \Delta H_k\big(v_i(t),\mathbf{u}_i(t)\big)\ .\end{aligned} \tag{10}$$

Note that $t_j^{(f)} = \ell\Delta$ for some integer ℓ.

In order to obtain an algorithm that is both computationally efficient and requires low memory, we demand that (and demonstrate in 3.1)

$$\alpha_i(t+\Delta) = \mathcal{A}\big(\alpha_i(t), n, \boldsymbol{\xi}^i(t+\Delta)\big), \tag{11}$$

for some function $\mathcal{A}$, where $\boldsymbol{\xi}^i(t)$ is a binary vector which indicates a spike at a presynaptic neuron, i.e. $\xi_j^i(t) = 1$ if a presynaptic spike occurred at neuron j at time t, and zero otherwise. In other words, the value of $\alpha_i(t+\Delta)$ depends only on the values of $\alpha_i(t)$ at time t and not on previous times.

Denote $\mathbf{w}_i = \big(w_{i1}, w_{i2}, ..., w_{iN}\big)^\top$, the vector of weights afferent to neuron i, and set

$$\boldsymbol{\alpha}_i(t) = \left(\sum_f \alpha_i(t-t_1^{(f)}), \sum_f \alpha_i(t-t_2^{(f)}), \cdots, \sum_f \alpha_i(t-t_N^{(f)})\right)^\top.$$

We express (10) as

$$v_i(t+\Delta) = v_i(t) + \Delta\left[F\big(v_i(t),\mathbf{u}_i(t)\big) + G\big(v_i(t),\mathbf{u}_i(t)\big)\mathbf{w}_i^\top \boldsymbol{\alpha}_i(t)\right]. \tag{12}$$

Differentiating (12) with respect to $\mathbf{w}_i$, and using the chain rule we obtain

$$\begin{aligned}\nabla_w v_i(t+\Delta) = \nabla_w v_i(t) &+ \Delta F'\big(v_i(t),\mathbf{u}_i(t)\big)\nabla_w v_i(t)\\ &+ \Delta G'\big(v_i(t),\mathbf{u}_i(t)\big)\nabla_w v_i(t)\mathbf{w}_i^\top \boldsymbol{\alpha}_i(t)\\ &+ \Delta G\big(v_i(t),\mathbf{u}_i(t)\big)\boldsymbol{\alpha}_i(t)\ ,\end{aligned}$$

or equivalently,

$$\begin{aligned}\nabla_w v_i(t+\Delta) = \nabla_w v_i(t)&\left[1 + \Delta F'\big(v_i(t),\mathbf{u}_i(t)\big) + \Delta G'\big(v_i(t),\mathbf{u}_i(t)\big)\mathbf{w}_i^\top \boldsymbol{\alpha}_i(t)\right]\\ &+ \Delta G\big(v_i(t),\mathbf{u}_i(t)\big)\boldsymbol{\alpha}_i(t)\ .\end{aligned} \tag{13}$$

This is a recursive update rule for $\nabla_w v_i(t)$, with an initial condition set at every postsynaptic spike, namely

$$\nabla_w v_i(t)\big|_{t=\hat{t}_i} = 0\ . \tag{14}$$

Summarizing the learning algorithm along with the neuronal dynamics we obtain algorithm 1.

3.1 Explicit Calculation of the Update Rules for Different $\boldsymbol{\alpha}$ Functions

As mentioned above, we assume that $\alpha(t)$ obeys (11). Here we demonstrate the update rule of the α function.

3.1.1 Demonstration for $\alpha(s) = q\delta(s)$

In the present discrete time setting the Dirac delta function is replaced by a Kronecker delta function, namely $\delta(t) \mapsto \delta_{0,t}$ where $t = \ell\Delta$ for some integer ℓ. Thus

$$\sum_f \alpha_i(t+\Delta-t_j^{(f)}) = \sum_f \delta_{t+\Delta,t_j^{(f)}} = q\xi_j^i(t+\Delta)$$

since the only contribution to the last sum arises when a presynaptic spike occurs at $t+\Delta$.

-

Initialization

Initialize weights $w_{ij}(0)$, eligibility traces $z_{ij}(0) = 0$ and $\alpha_i(0) = 0$

Initialize $v_i(0), u_{i,k}(0)$.

Step 1: Weight Update

$$w_{ij}(t) = w_{ij}(t-\Delta) + \gamma r(t) z_{ij}(t) \ ,$$
$$z_{ij}(t+\Delta) = \beta z_{ij}(t) + \lambda\big(\zeta_i(t) - \sigma\big)\frac{\partial v_i(t)}{\partial w_{ij}} \ ,$$

Step 2: Dynamic Update

$$v_i(t+\Delta) = v_i(t) + \Delta\left[F\big(v_i(t), \mathbf{u}_i(t)\big) + G\big(v_i(t), \mathbf{u}_i(t)\big)\sum_j w_{ij}\sum_f \alpha_i(t - t_j^{(f)})\right]$$
$$u_{i,k}(t+\Delta) = u_{i,k}(t) + \Delta H_k\big(v_i(t), \mathbf{u}_i(t)\big) \ .$$

Step 3: Parameter Update

$$\frac{\partial v_i(t+\Delta)}{\partial w_{ij}} = \frac{\partial v_i(t)}{\partial w_{ij}}\Big[1 + \Delta F'\big(v_i(t), \mathbf{u}_i(t)\big) + \Delta G'\big(v_i(t), \mathbf{u}_i(t)\big)\mathbf{w}_i^\top \boldsymbol{\alpha}_i(t)\Big]$$
$$+ \Delta G\big(v_i(t), \mathbf{u}_i(t)\big)\sum_f \alpha_i(t - t_j^{(f)}) \ ,$$
$$\sum_f \alpha_i(t + \Delta - t_j^{(f)}) = \sum_f \mathcal{A}\big(\alpha_i(t - t_j^{(f)}), n, \boldsymbol{\xi}^i(t+\Delta)\big) \ .$$

At every postsynaptic spike at neuron i

Reset $v_i(t),\ \mathbf{u}_i(t)$ and $\partial v_i(t)/\partial w_{ij}$.

Algorithm 1: Synaptic update rule for a generalized neuronal model

3.1.2 Demonstration for $\alpha(s) = \frac{q}{\tau_s} e^{-\frac{s}{\tau_s}}$

In this case,

$$\begin{aligned}\sum_f \alpha_i(t+\Delta-t_j^{(f)}) &= \frac{q}{\tau_s}\sum_f e^{-\left(\frac{t+\Delta-t_j^{(f)}}{\tau_s}\right)} \\ &= \frac{q}{\tau_s} e^{-\frac{\Delta}{\tau_s}} \sum_f e^{-\left(\frac{t-t_j^{(f)}}{\tau_s}\right)} + \frac{q}{\tau_s}\xi_j^i(t+\Delta) \\ &= e^{-\frac{\Delta}{\tau_s}} \sum_f \alpha_i(t-t_j^{(f)}) + \frac{q}{\tau_s}\xi_j^i(t+\Delta)\ ,\end{aligned}$$

with the appropriate initial conditions $\alpha_i(0) = 0$.

3.2 Depressing Synapses

As a simple application of the general procedure presented we consider the case of LIF neurons possessing a synaptic depression mechanism. While several formulations currently exist for depression, we consider the simple formulation proposed in (Richardson et al., 2005), leading to the dynamic equations (between spikes)

$$\begin{aligned}\tau_m \frac{dv_i}{dt} &= -\big(v_i(t) - v_r\big) + R\sum_j w_{ij} \sum_f \alpha\left(t - t_j^{(f)}\right) D_j(t) \\ \frac{dD_j(t)}{dt} &= \frac{1 - D_j(t)}{\tau_d} \qquad (t_j^{(f-1)} \le t \le t_j^{(f)}),\end{aligned}$$

where $t_j^{(f)}$ are the presynaptic spiking times. Whenever a presynaptic spike occurs, the depression variable, $D_j(t)$, is updated according to the rule

$$D_j(t_j^{(f)}) \leftarrow D_j(t_j^{(f)}) - v_D D_j(t_j^{(f)})\ ,$$

where v_D is related to the level of synaptic resource depletion (Richardson et al., 2005).

These equations describe a depletion of the synaptic resources whenever a presynaptic spike arrives, followed by a recovery process at a temporal scale of τ_d. Note that setting $D_j(t) = 1$, we recover the standard LIF model.

An explicit expression for $D_j(t)$ between two presynaptic spikes can be easily derived. The initial conditions are

$$D_f \stackrel{\triangle}{=} D_j(t)\Big|_{t=t_j^{(f)+}} = \Big[D_j(t) - v_D D_j(t)\Big]\Big|_{t=t_j^{(f)-}}\ ,$$

and for the first presynaptic spike, $D_j(0) = 1$.

Integrating the differential equation for $D_j(t)$, we obtain

$$D_j(t) = 1 - e^{-\left(t - t_j^{(f)}\right)/\tau_D}\Big[1 - D_f\Big] \qquad \left(t_j^{(f-1)} \leq t \leq t_j^{(f)}\right) \ .$$

The exact solutions for $v_i(t)$ and $D_j(t)$ enable an explicit calculation of the update rule for the case of depressing synapses. The general solution is rather cumbersome, and we present only the result for the case where $\alpha(t) = q\delta(t)$. The full derivation and general solution can be found in Appendix C.2.

$$\begin{aligned} w_{ij}(t) &= w_{ij}(t-\Delta) + \gamma r(t) z_{ij}(t) \ , \\ z_{ij}(t+\Delta) &= \beta z_{ij}(t) + \frac{\lambda q}{c}\big(\zeta_i(t) - \sigma_i(t)\big) \sum_{f: t_j^{(f)} > \hat{t}_i^{(0)}} \exp\left\{-\frac{t - t_j^{(f)}}{\tau_m}\right\} D_j(t_j^{(f)}) \ , \end{aligned}$$

which is very similar to (9), except that the presynaptic contribution is multiplied by the depression variable $D_j(t)$.

4 Simulation Results

In order to study the effectiveness of the update rules derived above, we present two learning experiments. The experiments are performed with the simple LIF neuron model augmented with the stochastic spiking procedure described in Section 1. We note that these experiments are aimed at demonstrating the derived synaptic rules, rather than proving their efficiency. Moreover, the results presented were obtained using episodic learning only (see definition below). We refer the reader to (Baras, 2006) for further numerical experiments and the application to online learning.

The learning rule presented in Section 2 is derived for a single neuron. However, based on Theorem 1 in (Bartlett and Baxter, 1999) the same learning rule applies at the network level. What these authors show is that each neuron essentially treats other neurons as though they were part of the environment; the only communication between neurons is through the global reward signal and the influence of other neurons on the state observations. The claim can be formally demonstrated by noting that the derivative of the average reward with respect to the (concatenated) vector of parameters corresponding to all neurons, can be broken down into derivatives with respect to each of the neurons' parameters, since the values of the weights afferent to each neuron are independent of

those relating to other neurons. This argument is similar to the one used to derive the real time recurrent learning algorithm (e.g., Haykin, 1999).

The derivation in Section 2 assumed an online learning setup, where both the eligibility trace and the weights are updated continually in time. In many applications, especially where the inputs and outputs are encoded using rates averaged over a temporal window, it is more meaningful to consider episodic learning. Episodic learning over a temporal window T is obtained when the eligibility traces are updated at each Euler step, but the weights are only updated at the end of the period T. More precisely,

$$\textbf{Episodic learning:} \qquad w_{ij}(T) = w_{ij}(0) + \gamma r(T)\bar{z}_{ij}(T), \tag{15}$$

$$\bar{z}_{ij}(T) = \langle z_{ij}(t)\rangle_T \stackrel{\triangle}{=} \frac{1}{N_{ep}}\sum_{t=0}^{T-\Delta} z_{ij}(t), \tag{16}$$

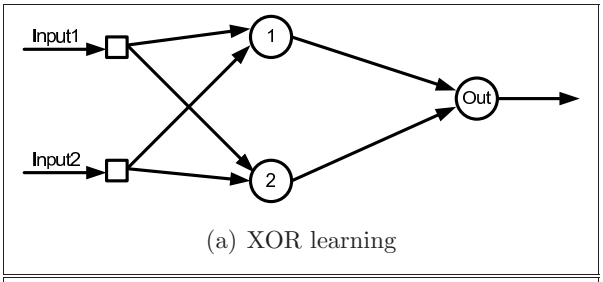
where T is the episode length, $N_{ep} = \frac{T}{\Delta} + 1$ and $z_{ij}(t)$ is given by (8), i.e. the eligibility traces are updated at every (discrete) time step (we assume for simplicity that T/Δ is an integer).

We comment briefly on the rate coding used in the experiments discussed below.

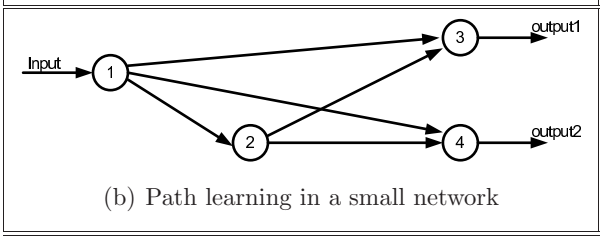
From initial experimentation we found that a high rate input stimulus is required in order to generate a network response. Therefore input coding consists of high firing rates (the exact values are provided in Section 4). Since the network weights are restricted, limiting the possible firing rates of output neurons, we used a lower threshold for output coding; specifically an output value of 1 was coded by a firing rate above 80 Hz. Low rates are encoded as 0 Hz for both input and output regardless of the problem.

We consider the following learning tasks.

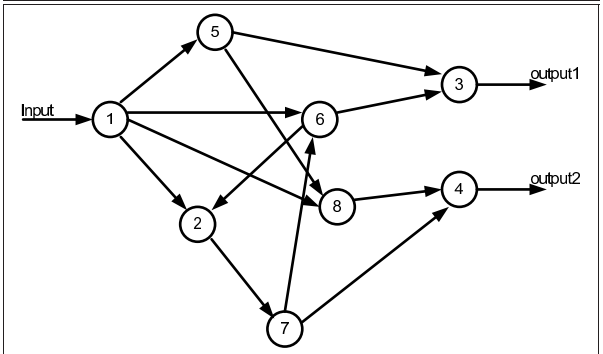
XOR The network shown in Figure 1(a), is a fully connected feedforward network with two input neurons, two hidden neurons and a single output neuron. We denote the weights entering hidden neuron i by w_{ij}^{h}, $j = 1, 2$ and the output weights by w_i^{out}, $i = 1, 2$. The task consists of solving the XOR problem, where inputs '0' and '1' are encoded as spike trains with low and high firing rates, respectively. The high input/output rates used in this experiment were 200/120 Hz. During the learning process, time is split into different learning epochs (or episodes). In each episode the network is presented with a randomly chosen pattern (the probability of all 4 patterns is equal). At the end of each episode, the output is determined as correct, incorrect or undetermined. The decision is obtained by comparing the output spiking rate to a predetermined threshold. An undetermined output relates to the case of a spiking rate very close to the threshold.



(a) XOR learning



(b) Path learning in a small network

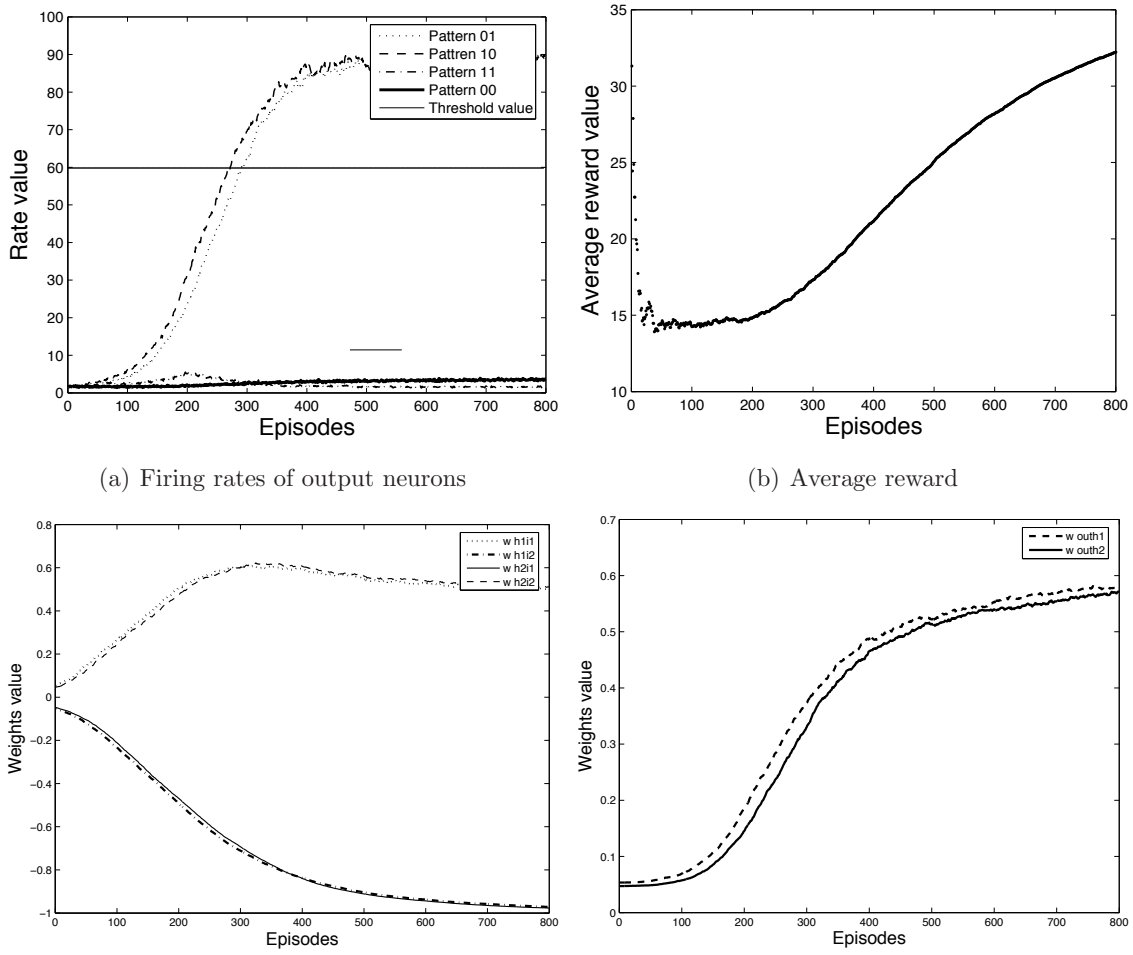


(c) Path learning in a large network

Figure 1: Networks architectures used in the simulations.

Reward is assigned depending on the output: positive reward for correct output and negative reward otherwise. Weights are randomly initialized but the following constraints were used: w^{h}_{11} and w^{h}_{22} are initialized with positive values, and w^{h}_{12} and w^{h}_{21} are initialized with negative values. This is the only problem (among those solved in Baras, 2006) in which a solution forces some of the weights to be negative (inhibitory synapses). All technical details and further explanations (exact reward values, parameter values etc.) are given in Appendix B. Figure 2 displays learning curves averaged over all successful learning simulations (99% successful convergence out of 100 experiments). We observe that hill climbing occurs for the average reward. Since the curves are averaged, it is impossible to see that the learning speed differs from simulation to simulation, although this phenomenon does indeed occur (Baras, 2006).

Path learning The networks shown in Figures 1(b) and 1(c) consist of a single input neuron, two output neurons and a varying number of hidden neurons. The aim here is to



(a) Firing rates of output neurons

(b) Average reward

(c) Evolution of first layer weights

(d) Evolution of second layer weights

Figure 2: XOR Learning (color online).

cause one of the output neurons to fire vigorously upon stimulation of the input neuron, while demanding that the other output neuron remains silent. There are no limitations on the behavior of other neurons in the network ("hidden" neurons). We consider both feedforward and recurrent networks here.

The reward scheme used in this case is more informative than the one used in the previous task, and depends on the exact spiking rate of each output separately. The idea is that the reward should be proportional to the performance of the network. Every output neuron contributes a positive term when it is correct, and a negative term otherwise. The exact details are given in Appendix B. The high input/output rates used in this experiment were 200/160 Hz. Figure 3 displays learning curves averaged over all successful learning simulations for the small network (94% successful convergence).

We also studied sequential switching between the two tasks for the smaller network of Figure 1(b). When considering switching tasks, we allow the network to learn a certain path for 2000 episodes. We then reverse the learned task. This process is repeated 3 times (6 tasks overall). Figure 4 displays learning curves averaged over all successful learning simulations in the small network (99% successful convergence). Note especially the behavior of weights during the learning process. Several observations are in order

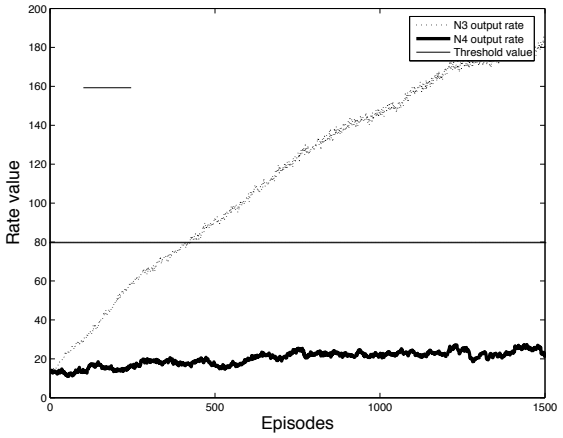
- From Figure 4(c), weights reaching the same output neuron are high together or low together. Namely w_{31} and w_{32} increase together when the first output should be high, while w_{41} and w_{42} decrease together and vice versa, depending on the task.
- The speed of learning in different weights is different. Observe that w_{31} and w_{41} reach stable values (either low or high) faster than w_{32} and w_{42}. This occurs since the input neuron stimulus possesses a high rate, but weights are constrained. No matter how high w_{21} is, neuron 2 will always have a lower rate than the input neuron. Therefore, the input neuron has more influence on both output neurons, and hence on the reward. Thus, the correlation between the reward and the eligibility traces of w_{31} and w_{41} is higher than the correlation between the reward and the eligibility traces of w_{32} and w_{42}. This causes the slower learning process of these "indirect" weights.

Figure 5 displays learning curves averaged over all successful learning simulations in the large network of Figure 1(c) (82% successful convergence). The "hidden" units rate curve (Baras, 2006) show that the recurrence (closed loop in the network) did not cause saturation in the rates, and they all remain within some reasonable area. The convergence rate is smaller than the previous network due to the following reasons:

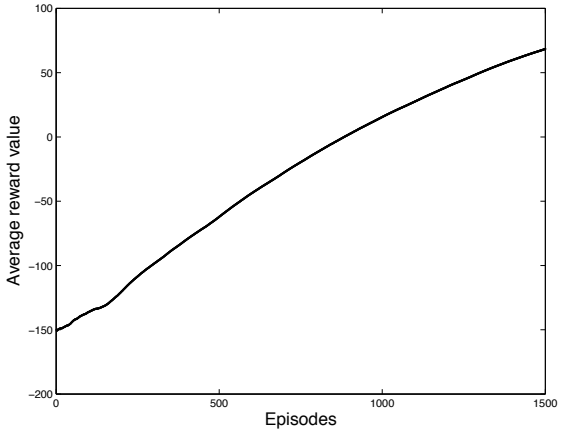
1. A large network implies a large number of parameters. When the number of parameters increases, the correlation between the reward and each eligibility trace is reduced, sometimes leading to unsuccessful learning.
2. A large number of parameters increases the probability of converging to a local maximum due to a larger number of extrema.
3. The state space is larger (proportional to the number of neurons in the network). Therefore, more exploration might be required.

All learning experiments described perform hill-climbing in the sense that the average reward increases with time. However, in some cases the system reaches a local maximum

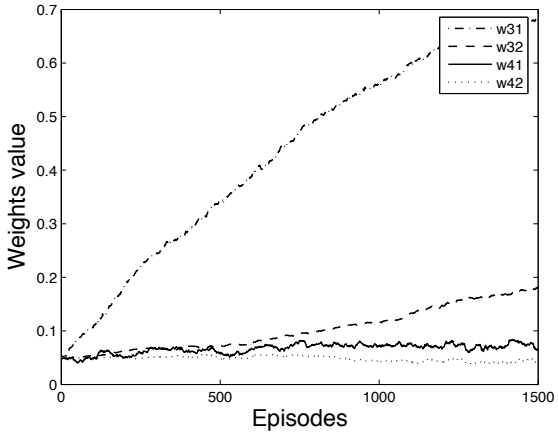
of the average reward. By unsuccessful learning we refer to situations where the reward did not reach a prescribed reward threshold value within the allotted time. This may result from either slow convergence or a poor local maximum.



(a) Firing rates of output neurons



(b) Average reward

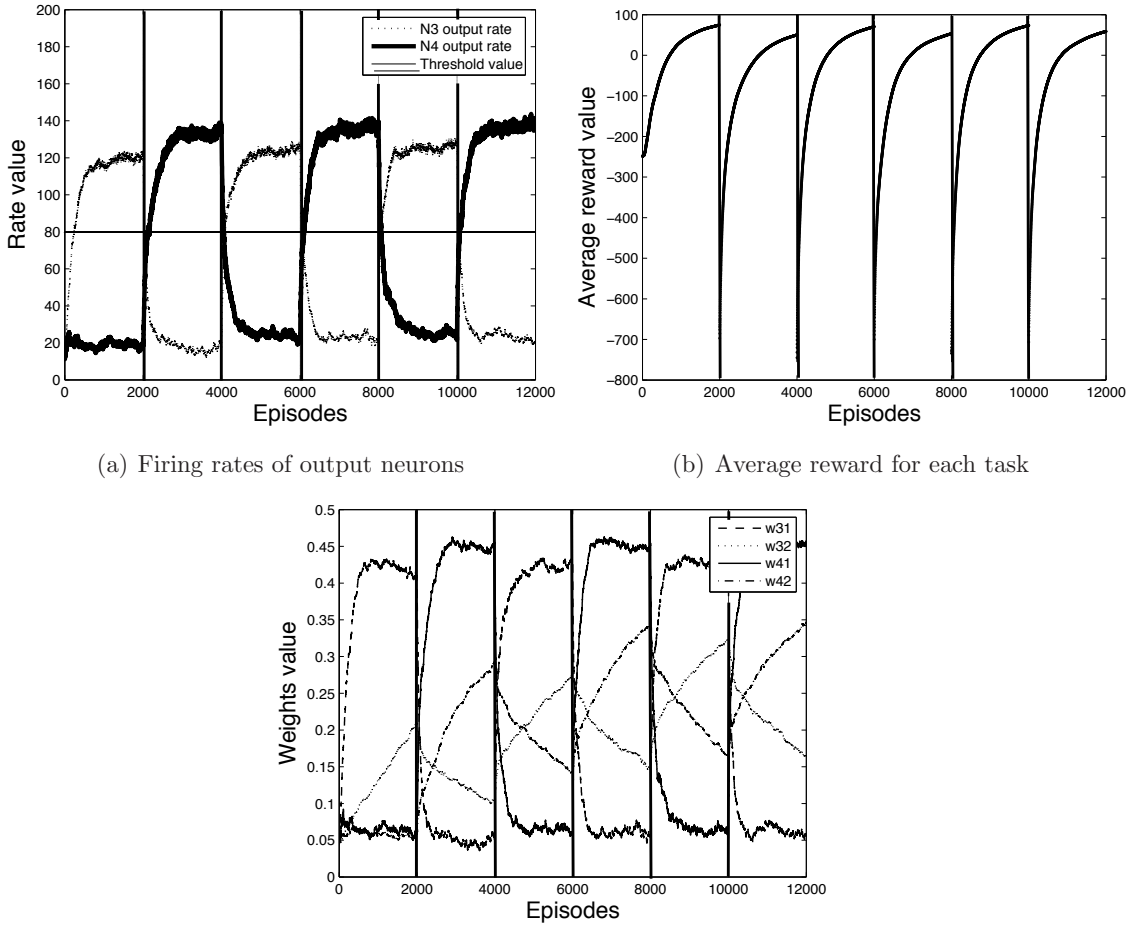


(c) Weight evolution

Figure 3: Path learning in a small network (color online).

5 Relation to the BCM Rule

The synaptic plasticity rules derived in Section 2 belong to the class of spike time dependent plasticity rules. In this section we consider the average behavior of the derived rule, where the expectation is taken over a temporal window allowing us to transform individual spikes to firing rates. We will show that the derived average rule behaves very similarly to the well known BCM rule (Bienenstock et al., 1982), for which good exper-



(a) Firing rates of output neurons

(b) Average reward for each task

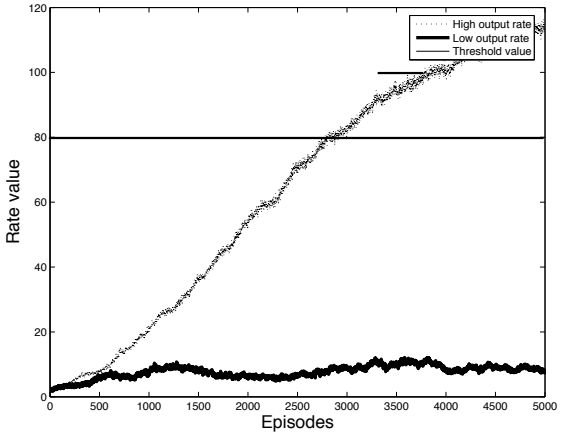
(c) Weight evolution

Figure 4: Path learning - switching tasks (color online).

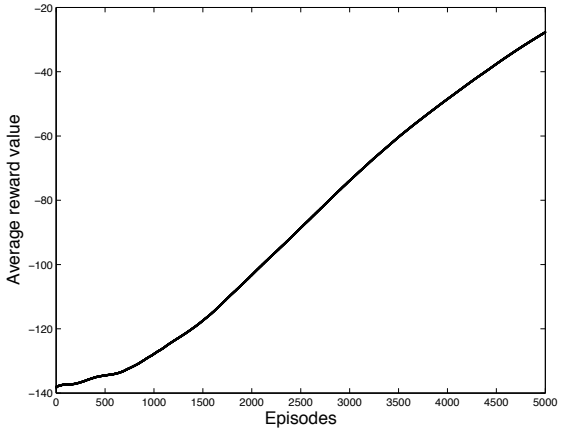
imental evidence exists (e.g., L. Cooper and Shouval, 2004). A related result for several forms of STDP was presented in (Izhikevich and Desai, 2003). We comment on this work at the end of the present section.

For the analysis presented in this section we assume $\alpha(t) = q\delta(t)$, and episodic learning. In order to average the synaptic rule one needs to make appropriate statistical assumptions. Here we assume that the *presynaptic* firing pattern is described by a stationary Poisson process, while the output firing is given by a Poisson process with a rate function which depends on the presynaptic neurons. More precisely

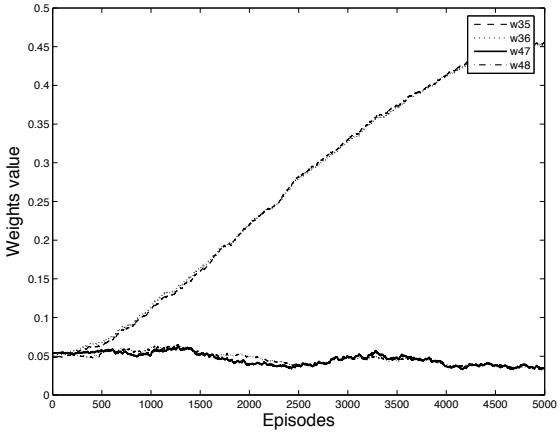
- The input spike train is a homogeneous Poisson process with rate x_j.



(a) Firing rates of output neurons



(b) Average reward



(c) First layer weight evolution

Figure 5: Path learning in a large network (color online).

- The output spike train is an inhomogeneous Poisson process with rate $\lambda_i(t)$, where

$$\lambda_i(t) = x_i + \eta w_{ij} {\sum_f}' e^{-\left(t - t_j^{(f)}\right)/\tau_m}, \tag{17}$$

where ${\sum_f}'$ denotes a summation over all presynaptic firing times following the last postsynaptic spike, and where x_i is the output rate when no presynaptic activity at neuron j is present. This relation captures the effect of increased postsynaptic firing rate due to both enhanced presynaptic activity and large synaptic weight (a similar and slightly more general approach is presented in Gerstner and Kistler, 2002, p. 408). In the sequel we will allow η to be a function of the weight w_{ij} and the presynaptic firing rate x_j. This allows for extra flexibility in determining postsynaptic firing rate based on (17).

Based on (15) and (16), the expected weight change over an episode is given by

$$\begin{aligned}\mathbf{E}\Delta w_{ij}(T) &= \mathbf{E}_{\text{post,pre}} \sum_{t=0}^{T-\Delta} z_{ij}(t+\Delta) \\ &= \mathbf{E}_{\text{post}|\text{pre}}\mathbf{E}_{\text{pre}} \sum_{t=0}^{T-\Delta} z_{ij}(t+\Delta), \end{aligned} \tag{18}$$

where we have assumed for simplicity that $w_{ij}(0) = 0$ and have neglected the constant term N_{ep}, which does not affect the form of the result. The expectation in (18) is with respect the presynaptic and postsynaptic spike trains.

The computation of the expectations in (18) is rather tedious, and can be found in the appendix. Based on this derivation we obtain an expression which holds in the low firing rate limit, where $(\Delta x_j)^2 \ll 1$.

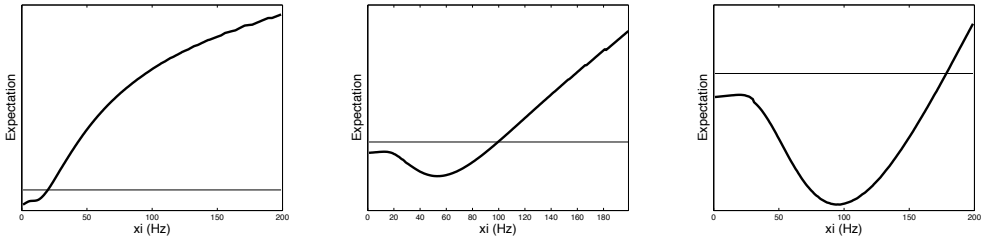
$$\mathbf{E}\Delta w_{ij}(T) \approx r(T)(x_j\Delta) \sum_{t=0}^{T-\Delta} (1-x_j\Delta)^{\frac{h}{\Delta}-1} \Big\{ x_i H_1(x_i,t) - H_2(x_i,w_{ij},t) \Big\} , \tag{19}$$

where

$$\begin{aligned} H_1(x_i,t) &= \Delta \sum_{k=1}^{\frac{h}{\Delta}} (e^{-\frac{t-t_k}{\tau_m}}) , \\ H_2(x_i,w_{ij},t) &= \sum_{k=1}^{\frac{h}{\Delta}} \left\{ \eta w_{ij}\Delta \left(e^{-\frac{t-t_i}{\tau_m}}\right)^2 - e^{-\frac{t-t_k}{\tau_m}} f(e^{-\frac{t-t_k}{\tau_m}}) \right\} , \end{aligned}$$

and where $h = \min(t,m)$, $f(x) = \sigma(v_r + w_{ij}x)$, $t_k = t - h + k\Delta$, and m is defined through (20) in the appendix. Note that both H_1 and H_2 depend on x_i through h which depends on x_i through (20) in Appendix A.

While the expressions are rather cumbersome, it is easy to compute them numerically and compare them graphically with the BCM plasticity rule. Figure 6 presents examples of the synaptic modification curve. The curves were created with η such that $\eta w_{ij} = 0.1x_j$, meaning that the contribution of a presynaptic spike to the weight update is no more than 10% of the presynaptic rate. This type of normalization guarantees that the firing of a single presynaptic cell does not dominate the postsynaptic firing rate. The specific choice of 0.1 is arbitrary, and different values lead to very similar behaviors. For example, we have tried corrections of the form $\eta w_{ij} = \text{const}x_j$. This relation is not mandatory, but some restriction on the value of η should be enforced, in order to prevent unrestrained growth of the postsynaptic firing rate. Note, however, that the effect of η is in any event small as can be seen in (31), where it is multiplied by an additional (small) term Δ. Other



(a) $x_j = 20$ (b) $x_j = 100$ Hz (c) $x_j = 180$ Hz

Figure 6: Average weight change as a function of the firing rate of the postsynaptic neuron for three presynaptic firing rates.

terms in (31), contain products of the form $x_i\Delta$ or $x_j\Delta$, where x_i and x_j, the firing rates, are much larger.

Examining (19) and Figure 6 we observe the following properties:

- The qualitative form of the expected weight change is similar to that of the BCM rule.
- Both potentiation and depression are observed, based on an activity dependent threshold.
- Zero crossing depends on both pre *and* postsynaptic activities. Higher presynaptic activity leads to increasing the threshold, which prevents uncontrolled weight increase (self-regulation). In fact, we find that the threshold is essentially linear in the presynaptic firing rate.

The curves were obtained using the same parameter values used in the simulations of the toy problems in Section 4. Using other parameter values (such as τ_m, T etc.) led to qualitatively similar results.

Izhikevich and Desai, 2003 have also recently presented some theoretical work related to the relationship between STDP and the BCM rule. Essentially, this work showed that BCM followed directly from STDP when the pre- and postsynaptic neurons fire weakly correlated Poisson spike trains, and only near-neighbor spike interactions are taken into account (in contrast to some implementations of STDP which use all spikes to update the weight). It is interesting to observe that our formulation leads to a similar result, except that instead of considering only nearest neighbors, we take into account all spikes between two consecutive postsynaptic spikes. This follows from the resetting mechanism taking place upon reaching threshold. Additionally, the STDP rule used here was derived from

the gradient POMDP framework rather than being postulated. Note that in the statistical analysis of the learning rule, we take into account all spikes over a fixed temporal window.

6 Discussion

Learning is a behavioral process, taking place at the level of an organism, based on interacting with the environment and receiving feedback. Synaptic plasticity is an internal process, that depends on neural activity. The aim of this work was to suggest an answer to how these two processes are related in the context of learning a stochastic policy in a network of spiking neurons. By formalizing the learning process as a Reinforcement Learning problem, and using the direct RL approach (Baxter and Bartlett, 2001), we derived an algorithm that updates the synaptic weights, based on the reward obtained as feedback. While previous work has addressed these issues (Bartlett and Baxter, 1999; Seung, 2003; Xie and Seung, 2004), we have been able to extend these approaches to general neural models, which can be made as biologically realistic as is required. Additionally, a detailed analysis has shown that the derived rule is related to the BCM rule. We are not aware of any previous work relating the BCM rule to reinforcement learning.

The synaptic update rule derived is computationally efficient, and yields good results in several learning tasks. Moreover, the basic rule derived in Section 2 depends on local activity and data available at the synaptic site, and requires mechanisms that are available at the cellular level. The update rule derived in Section 3 for more realistic neural models is indeed rather complex. A question left open at this point relates to the possible biological implementation of such rules.

Finally, we comment on the relation of our work to other recent contributions (we have already related our work to Bartlett and Baxter, 1999 and Xie and Seung, 2004). The work of Rao and Sejnowski, 2001 begins with a biophysical model of a neuron and shows how the temporal difference learning rule (e.g., Sutton and Barto, 1998) can reproduce the temporally asymmetric window observed in STDP plasticity rules. Wörgötter and Porr, 2004 present a general overview of temporal difference learning and its implementation in networks of spiking neurons. Both these approaches relate to ours in that they connect a well known computational approach, namely temporal difference learning, to synaptic plasticity rules. Our work differs form these in at least three ways. First, we *derive* (rather than relate) a synaptic modification rule for learning an optimal *policy* (a control problem), rather than attempting to estimate an optimal *value function* (a prediction task), as is done in temporal difference methods. The latter approach always requires an additional step of deciding on an appropriate action, given the value estimate. Second,

the relationship to the BCM rule, and its direct relation to policy improvement is novel. Third, the approach taken here enables the extension of the approach to a broad range of novel policy improvement algorithms proposed in the recent machine learning literature (see below).

As a final note, we comment that Toyoizumi et al., 2005 have recently derived a BCM-like learning rule for spiking neurons based on information theoretic arguments within an unsupervised learning framework. Based on an integrate and fire neuronal model and Poisson statistics assumptions, they show that maximizing the mutual information between the input and output, subject to constraints on the output neuron's firing rate, leads to a BCM-like plasticity rule. It would be interesting to see whether this type of result, may shed an information theoretic light on the reinforcement learning setup we have considered in this paper.

The framework developed here can be extended in many directions. For example, more realistic neural models may be easily incorporated within the setting presented in Section 3. Moreover, general synaptic window functions can be considered, based on more detailed physiological information. Finally, recent extensions of gradient POMDP approaches, based on actor-critic architectures (Konda and Tsitsiklis, 2003), are based on a temporal difference signal, rather than on the reward itself as in this work. Given the flurry of recent interest in the relationship between the temporal difference signal in RL, and the behavior of Dopaminergic neurons in the brainstem (e.g. Schultz, 2002), it would be particularly interesting to extend the present framework accordingly.

Acknowledgments This work was partially supported by EU Project PASCAL, and by the Technion VPR fund for promotion of research and by the Ollendorf foundation. The authors are grateful to the anonymous reviewers for helpful remarks, which greatly improved the clarity of the presentation.

A Appendix

Our objective here is to compute the expectation

$$\mathbf{E}_{\text{post,pre}} \sum_{t=0}^{T-\Delta} z_{ij}(t+\Delta) = \mathbf{E}_{\text{post,pre}} \sum_{t=0}^{T-\Delta} \big(\zeta_i(t) - \sigma_i(t)\big) \sum_{f: t_j^{(f)} > \hat{t}_i^{(0)}} \exp\left\{-\frac{t - t_j^{(f)}}{\tau_m}\right\},$$

where the expectation is with respect to the presynaptic and postsynaptic spike processes. Calculating this expression directly is very difficult.

A possible solution to this difficulty is to sum over a *fixed* time window of length m. A

reasonable choice for the window length is given by the average inter-spike interval of a Poisson process with rate x_i. Since these intervals are exponentially distributed with the same parameter as the original process, we set[2]

$$m' \triangleq \mathbf{E}(\text{Interspike interval}) = \frac{1}{x_i} \qquad \text{and} \qquad m = \Delta \left\lceil \frac{m'}{\Delta} \right\rceil \ , \tag{20}$$

implying the approximation

$$\sum_{f:t_j^{(f)}>\hat{t}_i^{(0)}} \exp\left\{-\frac{t-t_j^{(f)}}{\tau_m}\right\} \approx \sum_{f:t_j^{(f)}>t-m} \exp\left\{-\frac{t-t_j^{(f)}}{\tau_m}\right\} .$$

This approximate equality can also be expressed as

$$\sum_{f:t_j^{(f)}>\hat{t}_i^{(0)}} \exp\left\{-\frac{t-t_j^{(f)}}{\tau_m}\right\} \approx \sum_{t'=t-h}^{\top} \left[I(\text{Spike at time } t') \exp\left\{-\frac{t-t'}{\tau_m}\right\}\right] ,$$

where we have introduced an indicator function at the times of the presynaptic spikes, and where $h = \min(t, m)$ and all times are multiples of Δ.

Defining

$$y(t) \triangleq \sum_{t'=t-h}^{T} \left[I(\text{Spike at time } t') \exp\left\{-\frac{t-t'}{\tau_m}\right\}\right] , \tag{21}$$

and

$$f(\theta) = \sigma_i\big\{v_r + w_{ij}\theta\big\}, \tag{22}$$

we are left with

$$\mathbf{E}_{\text{post,pre}} \sum_t z_{ij}(t+\Delta) \approx \mathbf{E}_{\text{pre}} \mathbf{E}_{\text{post|pre}} \sum_t \left[\Big(\zeta_i(t) - f\big(y(t)\big) \Big) y(t) \right] . \tag{23}$$

Expectation with respect to the postsynaptic spike train

The expectation $\mathbf{E}_{\text{post|pre}}[\cdot]$ can be computed easily, based on the relation

$$\zeta_i(t) = \begin{cases} 1 & \text{with probability } \lambda_i(t)\Delta \\ 0 & \text{with probability } 1-\lambda_i(t)\Delta \end{cases} ,$$

where $\lambda_i(t) = w_{ij} + \eta w_{ij} y(t)$.

[2]Note that the postsynaptic rate is not strictly x_i (see (17)).

We obtain

$$\begin{aligned}\mathbf{E}_{\text{post}|\text{pre}}\sum_t z_{ij}(t+\Delta) &= \mathbf{E}_{\text{post}|\text{pre}}\sum_t\Big[\Big(\zeta_i(t)-f\big(y(t)\big)\Big)y(t)\Big]\\ &= \sum_t\Big[\lambda_i(t)\Delta\Big(1-f\big(y(t)\big)\Big)y(t)+(1-\lambda_i(t)\Delta)\Big(0-f\big(y(t)\big)\Big)y(t)\Big]\\ &= \sum_t\Big[y(t)\Big(\lambda_i(t)\Delta-f\big(y(t)\big)\Big)\Big]\\ &= \sum_t\Big[y(t)\Big(x_i\Delta+\eta w_{ij}y(t)\Delta-f\big(y(t)\big)\Big)\Big]\ . \end{aligned}\tag{24}$$

Expectation with respect to the presynaptic spike train

Based on (24), we are the left with the task of to computing

$$\begin{aligned}&\mathbf{E}_{\text{pre}}\textstyle\sum_t\Big[y(t)\Big(x_i\Delta+\eta w_{ij}\Delta y(t)-f\big(y(t)\big)\Big)\Big]\\ &=\textstyle\sum_t\Big[x_i\Delta\mathbf{E}_{\text{pre}}y(t)+\eta w_{ij}\Delta\mathbf{E}_{\text{pre}}y^2(t)-\mathbf{E}_{\text{pre}}\Big(y(t)f\big(y(t)\big)\Big)\Big]\ ,\end{aligned}\tag{25}$$

which involves expectations of functions of the form $\mathbf{E}_{\text{pre}}G_i(y(t))$ with $G_1(y)=y$, $G_2(y)=y^2$ and $G_3(y)=yf(y))$. To do so, we present the computation of $\mathbf{E}_{\text{pre}}G\big(y(t)\big)$ for any function G. We first prove a simple lemma.

Lemma A.1 *Let y be a random variable with characteristic function $\Psi(\omega)\stackrel{\triangle}{=}\mathbf{E}e^{-j\omega y}$. Then*

$$\mathbf{E}G\big(y\big)=\int_{-\infty}^{\infty}dsG(s)\mathcal{F}^{-1}\big(\Psi(\omega)\big)(s),$$

where $\mathcal{F}^{-1}\big(\Psi(w)\big)$ is the inverse Fourier transform of $\Psi(w)$, given by

$$\mathcal{F}^{-1}\big(\Psi(w)\big)=\frac{1}{2\pi}\int_{-\infty}^{\infty}\Psi(\omega)e^{j\omega y}d\omega\ .$$

Proof Using the equality $\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} d\omega e^{j\omega t}$, we have

$$\begin{aligned}
\mathbf{E}G\big(y\big) &= \mathbf{E}\int\limits_{-\infty}^{\infty} dsG(s)\delta\big(s-y\big) \\
&= \frac{1}{2\pi}\mathbf{E}\int\limits_{-\infty}^{\infty} dsG(s)\int\limits_{-\infty}^{\infty} d\omega e^{j\omega\big(s-y\big)} \\
&= \frac{1}{2\pi}\int\limits_{-\infty}^{\infty} dsG(s)\int\limits_{-\infty}^{\infty} d\omega e^{j\omega s}\mathbf{E}e^{-j\omega y} \\
&= \int\limits_{-\infty}^{\infty} dsG(s)\mathcal{F}^{-1}\big(\mathbf{E}e^{-j\omega y}\big)(s)\ . \qquad\qquad \square
\end{aligned}$$

Note that Lemma A.1 follows easily from the equality $\mathbf{E}G(y) = \int G(s)f(s)ds$, where f is the pdf of the random variable y, and recalling that the pdf is the inverse Fourier transform of the characteristic function.

Using Lemma A.1 we have

$$\mathbf{E}_{\text{pre}}G\big(y(t)\big) = \int\limits_{-\infty}^{\infty} G(s)\mathcal{F}^{-1}\big(\mathbf{E}_{\text{pre}}e^{-j\omega y(t)}\big)(s)ds. \tag{26}$$

Defining the Fourier pair

$$\Phi_t(\omega) \stackrel{\triangle}{=} \mathbf{E}_{\text{pre}}e^{-j\omega y(t)} \qquad ; \qquad \phi_t(s) \stackrel{\triangle}{=} \mathcal{F}^{-1}\big(\Phi_t(\omega)\big)(s),$$

we have

$$\begin{aligned}
\Phi_t(\omega) &= \mathbf{E}_{\text{pre}}e^{-j\omega\sum_{t'=t-h}^{t}\left[I(\text{Spike at time } t')\exp\left\{-\frac{t-t'}{\tau_m}\right\}\right]} \\
&\stackrel{(a)}{=} \prod_{t'=t-h}^{t}\mathbf{E}e^{-j\omega\left[I(\text{Spike at time } t')\exp\left\{-\frac{t-t'}{\tau_m}\right\}\right]} \\
&= \prod_{t'=t-h}^{t}\left\{x_j\Delta e^{-j\omega\left[\exp\left\{-\frac{t-t'}{\tau_m}\right\}\right]} + \big(1-x_j\Delta\big)\right\}\ ,
\end{aligned}$$

where (a) used independence.

Inverse Fourier transforming $\Phi_t(\omega)$ we have

$$\phi_t(s) = \mathcal{F}^{-1}\left(\prod_{t'=t-h}^{t}\left\{x_j\Delta e^{-j\omega\left[\exp\left\{-\frac{t-t'}{\tau_m}\right\}\right]} + \big(1-x_j\Delta\big)\right\}\right)(s)\ .$$

Based on the properties of the Fourier transform, we convert a product over functions in the time domain to a convolution of transforms in the frequency domain. Setting $K = h/\Delta$, and defining

$$\Phi_t^k(\omega) \triangleq x_j \Delta e^{-j\omega\left[\exp\left\{-\frac{t-t_k}{\tau_m}\right\}\right]} + \big(1 - x_j\Delta\big), \qquad (k = 0, 1, \ldots, K),$$

where $t_k = t - h + k\Delta$, and

$$\phi_t^k(s) \triangleq \mathcal{F}^{-1}\big(\Phi_t^k(\omega)\big)(s),$$

we have that

$$\Phi_t(\omega) = \prod_{k=0}^{K} \Phi_t^k(\omega),$$

and denoting a convolution by $\star$ we have that

$$\phi_t(s) = \phi_t^1 \star \phi_t^2 \star \cdots \star \phi_t^K(s). \tag{27}$$

Now,

$$\begin{aligned}\phi_t^k(s) &= \mathcal{F}^{-1}\bigg(x_j\Delta e^{-j\omega\left[\exp\left\{-\frac{t-t_k}{\tau_m}\right\}\right]} + \big(1 - x_j\Delta\big)\bigg)(s) \\ &= x_j\Delta\delta\big(s - e^{-\frac{t-t_k}{\tau_m}}\big) + (1 - x_j\Delta)\delta\big(s\big) \qquad (t_k = t - h + k\Delta).\end{aligned}$$

In computing (27)we face the following task. Given m numbers, $\{x_1, x_2, ..., x_m\}$, and a constant a, compute

$$A \stackrel{\triangle}{=} \big(a\delta(x - x_1) + (1 - a)\delta(x)\big) \star \cdots \star \big(a\delta(x - x_m) + (1 - a)\delta(x)\big).$$

Using the properties of the δ-function, it is not hard to show that

$$A = a^m\delta\big(x - \sum_{k=1}^{m} x_k)\big) + a^{m-1}(1 - a)\sum_{i=1}^{m}\delta\big(x - \sum_{k:k\neq i} x_k\big) + \cdots + (1 - a)^m\delta(x).$$

Thus, we find

$$\begin{aligned}\phi_t(s) = \Big[&(x_j\Delta)^K\delta(s - \sum_{k=1}^{K} e^{-\frac{t-t_k}{\tau_m}}) + (x_j\Delta)^{K-1}(1 - x_j\Delta)\sum_{i=1}^{K}\delta\big(s - \sum_{k:k\neq i} e^{-\frac{t-t_k}{\tau_m}}\big) \\ &+ \cdots + (1 - x_j\Delta)^K\delta(s)\Big]\end{aligned} \tag{28}$$

where $K = h/\Delta$.

Recalling that $\phi_t(s) = \mathcal{F}^{-1}(\Phi(\omega))$, and substituting (28) into (26) yields

$$\begin{aligned}\mathbf{E}_{\text{pre}}G\big(y(t)\big) = \Big[&(x_j\Delta)^K G(\sum_{i=1}^{K} e^{-\frac{t-t_i}{\tau_m}}) + (x_j\Delta)^{K-1}(1 - x_j\Delta)\sum_{i=1}^{K} G\big(\sum_{k:k\neq i} e^{-\frac{t-t_k}{\tau_m}}\big) \\ &+ \cdots + (1 - x_j\Delta)^K G(0)\Big].\end{aligned} \tag{29}$$

Thus, the expectation of a function of $y(t)$ equals a weighted sample of this function at particular points in time.

At this stage, we assume a single presynaptic neuron. The extension to several presynaptic neurons is straightforward.

Combining (24), (25) and (29) yields the final result. While the computation can be performed exactly, it leads to the following very cumbersome expressions:

$$
\begin{aligned}
&\mathbf{E}_{\text{pre}} \sum_t \Big[y(t) \Big(x_i \Delta t + \eta w_{ij} \Delta t y(t) - f\big(y(t)\big) \Big) \Big] \\
&= \sum_t \Big[x_i \Delta t \mathbf{E}_{\text{pre}} y(t) + \eta w_{ij} \Delta t \mathbf{E}_{\text{pre}} y^2(t) - \mathbf{E}_{\text{pre}} \Big(y(t) f\big(y(t)\big) \Big) \Big] \\
&= \sum_t \Big\{ x_i \Delta t \Big[(x_j \Delta t)^{\frac{h}{\Delta t}} \sum_{i=1}^{\frac{h}{\Delta t}} e^{-\frac{t-t_i}{\tau_m}} + (x_j \Delta t)^{\frac{h}{\Delta t}-1} (1 - x_j \Delta t) \sum_{i=1}^{\frac{h}{\Delta t}} \sum_{k:k\neq i} e^{-\frac{t-t_k}{\tau_m}} + \cdots \Big] \\
&\quad + \eta w_{ij} \Delta t \Big[(x_j \Delta t)^{\frac{h}{\Delta t}} \big(\sum_{i=1}^{\frac{h}{\Delta t}} e^{-\frac{t-t_i}{\tau_m}} \big)^2 + (x_j \Delta t)^{\frac{h}{\Delta t}-1} (1 - x_j \Delta t) \sum_{i=1}^{\frac{h}{\Delta t}} \big(\sum_{k:k\neq i} e^{-\frac{t-t_k}{\tau_m}} \big)^2 + \cdots \Big] \\
&\quad - \Big[(x_j \Delta t)^{\frac{h}{\Delta t}} f\big(\sum_{i=1}^{\frac{h}{\Delta t}} e^{-\frac{t-t_i}{\tau_m}} \big) \big(\sum_{i=1}^{\frac{h}{\Delta t}} e^{-\frac{t-t_i}{\tau_m}} \big) \\
&\quad + (x_j \Delta t)^{\frac{h}{\Delta t}-1} (1 - x_j \Delta t) \sum_{i=1}^{\frac{h}{\Delta t}} f\big(\sum_{k:k\neq i} e^{-\frac{t-t_k}{\tau_m}} \big) \big(\sum_{k:k\neq i} e^{-\frac{t-t_k}{\tau_m}} \big) + \cdots \Big] \Big\} .
\end{aligned}
$$

Note that here, $t_1 = t - h, t_K = t$, and $G_i(0) = 0,\ i = 1, 2, 3$.

The expressions become easier to handle under the approximation of low firing rates, where $(\Delta x_j)^2 \ll 1$, which allows us to neglect higher order terms. Specifically, neglecting

terms of order $(x_j\Delta t)^k,\ k \geq 2$, we obtain

$$\begin{aligned}
&\mathbf{E}_{\text{pre}} \sum_t \Big[y(t) \Big(x_i \Delta t + \eta w_{ij} \Delta t y(t) - f\big(y(t)\big) \Big) \Big] \\
&\quad = \sum_t \Big[x_i \Delta t \mathbf{E}_{\text{pre}} y(t) + \eta w_{ij} \Delta t \mathbf{E}_{\text{pre}} y^2(t) - \mathbf{E}_{\text{pre}} \Big(y(t) f\big(y(t)\big) \Big) \Big] \\
&\quad = \sum_t \Big[x_i \Delta t (x_j \Delta t)(1 - x_j \Delta t)^{\frac{h}{\Delta t} - 1} \sum_{i=1}^{\frac{h}{\Delta t}} (e^{-\frac{t-t_i}{\tau_m}}) \\
&\quad + \eta w_{ij} \Delta t (x_j \Delta t)(1 - x_j \Delta t)^{\frac{h}{\Delta t} - 1} \sum_{i=1}^{\frac{h}{\Delta t}} (e^{-\frac{t-t_i}{\tau_m}})^2 \\
&\quad - (x_j \Delta t)(1 - x_j \Delta t)^{\frac{h}{\Delta t} - 1} \sum_{i=1}^{\frac{h}{\Delta t}} \Big\{ e^{-\frac{t-t_i}{\tau_m}} f(e^{-\frac{t-t_i}{\tau_m}}) \Big\} \Big] \ .
\end{aligned} \tag{30}$$

Which can be rewritten as

$$\begin{aligned}
\mathbf{E}_{\text{post,pre}} \sum_{t=0}^{T-\Delta} z_{ij}(t+\Delta) \approx (x_j \Delta) \sum_{t=0}^{T-\Delta} (1 - x_j \Delta)^{\frac{h}{\Delta} - 1} \Big[x_i \Delta \sum_{k=1}^{\frac{h}{\Delta}} e^{-\frac{t-t_k}{\tau_m}} + \eta w_{ij} \Delta \sum_{k=1}^{\frac{h}{\Delta}} (e^{-\frac{t-t_k}{\tau_m}})^2 \\
- \sum_{k=1}^{\frac{h}{\Delta}} \Big\{ e^{-\frac{t-t_k}{\tau_m}} f(e^{-\frac{t-t_i}{\tau_m}}) \Big\} \Big] \ ,
\end{aligned} \tag{31}$$

where $t_k = t - h + k\Delta$.

B Simulation Details

The values used for the main simulation parameters are shown in Table 1. The reward values for the XOR problem are summarized in Table 2. For the path learning problem, we first define the following variables:

- n_{th} - the threshold value for determining "high" and "low" output.
- S_{out1}, S_{out2} - the number of spikes that occurred in the first and second outputs, respectively.
- F_{1high}, F_{2low} - Boolean flags denoting whether output 1 is high and output 2 is low, respectively.

Parameter	Value	Comments
Δ	$5 \cdot 10^{-4}$	Euler step size
λ	120	Squashing function parameter
β	0	Bias/variance tradeoff variable
γ	0.001	Learning rate for the policy gradient method
T	250 msec	Episode length
$v_r = v_L$	-60 mV	Neuron resting potential
R_m	$10^6\Omega$	Membrane resistance
C_m	$3 \cdot 10^{-8}$ F	Membrane capacitance
τ_m	30 msec	$\tau_m = R_m \cdot C_m$
τ_s	3 msec	Time constant of α function
q	$1.8 \cdot 10^{-9}$ C	

Table 1: Simulation parameter values

Description	**Value**
Correct output	$R_+ = 96$
Incorrect output	$R_- = -66$
Undetermined output	$R_u = -69$

Table 2: XOR reward values

The reward is determined based on the formula

$$R = 3(F_{2low}(n_{th} - S_{out2}) - 2(1 - F_{2low})S_{out2} + F_{1high}S_{out1} - 2(1 - F_{1high})(n_{th} - S_{out1}) - 50).$$

We also comment about the reward signals used to train the networks. In the case of the XOR learning network, the reward values are constant based on the network's performance. The specific values were chosen by trial and error. The fact that the reward scheme affects the quality of the solution is a well known phenomenon in direct policy search methods. As far as we are aware, there do not appear at present to be systematic approaches to selecting optimal reward schemes. In the case of the path learning task, in both networks, we tried a more realistic and informative reward signal. The general approach that led us to the reward function form has already been explained in Section 4, and the exact constants and coefficients were also determined by trial and error.

Problem	**Episode length (msec)**	**Number of episodes**	**Successful convergence percentage**
XOR	250	800	99%
Path learning, small network	500	1500	94%
Path learning, small network, switching tasks	500	2000 per task	99%
Path learning, large network	500	5000	82%

Table 3: Technical details and convergence rates

Problem	**Weight Constrains**	**Weight Initialization**
XOR	range $[-1, 1]$	randomly initialized from a uniform distribution over $[-0.1, 0], [0, 0.1]$, sign predetermined (see section 4 for details)
Path learning, small network	range $[0, 0.5]$	randomly initialized from a uniform distribution over $[0, 0.1]$
Path learning, small network, switching tasks	range $[0, 0.5]$	randomly initialized from a uniform distribution over $[0, 0.1]$
Path learning, large network	range $[0, 0.5]$	randomly initialized from a uniform distribution over $[0, 0.1]$

Table 4: Weight constraints and initialization used

C Technical Derivations

The objective of this appendix is to provide some explicit technical derivations that are not essential to understanding the main results.

C.1 Decaying Exponential α function

We present the calculation leading to the expression for the case of a decaying exponential function presented in 2.1. The α function is $\alpha(s) = \frac{q}{\tau_s} \exp\left(-\frac{s}{\tau_s}\right) I\{s \geq 0\}$.

Substituting this choice of $\alpha(s)$ on the right hand side of (8) we obtain

$$
\begin{aligned}
&\int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\tfrac{t-s}{\tau_m}\right\} \textstyle\sum_f \alpha\left(s - t_j^{(f)}\right) ds \\
&= \int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\tfrac{t-s}{\tau_m}\right\} \textstyle\sum_f \frac{q}{\tau_s} \exp\left(-\frac{s-t_j^{(f)}}{\tau_s}\right) I\{s - t_j^{(f)} \geq 0\} ds \\
&= \tfrac{q}{\tau_s} \textstyle\sum_f \int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\tfrac{t-s}{\tau_m}\right\} \exp\left(-\frac{s-t_j^{(f)}}{\tau_s}\right) I\{s - t_j^{(f)} \geq 0\} ds \\
&= \tfrac{q}{\tau_s} \textstyle\sum_f \exp\left\{\tfrac{-t}{\tau_m}\right\} \exp\left\{\tfrac{t_j^{(f)}}{\tau_s}\right\} \int_{\hat{t}_i^{(0)}}^{t} \exp\left\{s\left(\tfrac{1}{\tau_m} - \tfrac{1}{\tau_s}\right)\right\} I\{s - t_j^{(f)} \geq 0\} ds \\
&= \tfrac{q}{\tau_s} \textstyle\sum_f \exp\left\{\tfrac{-t}{\tau_m}\right\} \exp\left\{\tfrac{t_j^{(f)}}{\tau_s}\right\} \int_{\max\left(t_j^{(f)}, \hat{t}_i^{(0)}\right)}^{t} \exp\left\{s\left(\tfrac{1}{\tau_m} - \tfrac{1}{\tau_s}\right)\right\} ds \\
&= \tfrac{q}{\tau_s} \textstyle\sum_f \exp\left\{\tfrac{-t}{\tau_m}\right\} \exp\left\{\tfrac{t_j^{(f)}}{\tau_s}\right\} \frac{1}{\left(\frac{1}{\tau_m} - \frac{1}{\tau_s}\right)} \left\{ \exp\left\{t\left(\tfrac{1}{\tau_m} - \tfrac{1}{\tau_s}\right)\right\} - \exp\left\{\max\left(t_j^{(f)}, \hat{t}_i^{(0)}\right)\left(\tfrac{1}{\tau_m} - \tfrac{1}{\tau_s}\right)\right\} \right\} \\
&= \frac{1}{\left(\frac{1}{\tau_m} - \frac{1}{\tau_s}\right)} \tfrac{q}{\tau_s} \textstyle\sum_f \left\{ \exp\left\{-\left(\frac{t-t_j^{(f)}}{\tau_s}\right)\right\} - \exp\left\{\tfrac{-t}{\tau_m}\right\} \exp\left\{\tfrac{t_j^{(f)}}{\tau_s}\right\} \exp\left\{\max\left(t_j^{(f)}, \hat{t}_i^{(0)}\right)\right\} \right\} \\
&= \tfrac{q\tau_m}{(\tau_s - \tau_m)} \textstyle\sum_f \exp\left\{-\left(\frac{t-t_j^{(f)}}{\tau_s}\right)\right\} \left[1 - \exp\left\{\left(t - \max\left(t_j^{(f)}, \hat{t}_i^{(0)}\right)\right)\left(\tfrac{1}{\tau_m} - \tfrac{1}{\tau_s}\right)\right\}\right] \\
&= \tfrac{q\tau_m}{(\tau_m - \tau_s)} \textstyle\sum_f \exp\left\{-\left(\frac{t-t_j^{(f)}}{\tau_s}\right)\right\} \left[\exp\left\{\left(\tfrac{\tau_m - \tau_s}{\tau_m \tau_s}\right)\left(\min\left(t - t_j^{(f)}, t - \hat{t}_i^{(0)}\right)\right)\right\} - 1\right] ,
\end{aligned}
$$

hence

$$
\begin{aligned}
z_{ij}(t + \Delta t) = \beta z_{ij}(t) \\
&+ \frac{\left(\zeta_i(t) - \sigma_i(t)\right) \lambda q \tau_m}{c\left(\tau_m - \tau_s\right)} \sum_f e^{-\left(\frac{t-t_j^{(f)}}{\tau_s}\right)} \left[\exp\left\{\left(\frac{\tau_m - \tau_s}{\tau_m \tau_s}\right)\left(\min\left(t - t_j^{(f)}, t - \hat{t}_i^{(0)}\right)\right)\right\} - 1\right] .
\end{aligned}
$$

C.2 Depressing Synapses

We present the exact details and derivation in the case of the depressing synapses. We will not repeat details presented in Section 3.2.

POMDP model

We use the same POMDP model as in previous case (see Section 1), except that the state of each neuron also includes the current value of the depression variable.

Parameter update

The same update rule for the parameters is used. Let us now calculate an explicit expression for the weight update for this neural model.

As presented in Section 3.2, the update rule is given by,

$$w_{ij}(t) = w_{ij}(t-\Delta t) + \gamma r(t) z_{ij}(t) \ , \tag{32}$$

$$z_{ij}(t+\Delta t) = \beta z_{ij}(t) + \big(\zeta_i(t) - \sigma\big)\frac{\lambda}{c}\int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\frac{t-s}{\tau_m}\right\} \sum_f \alpha\left(s - t_j^{(f)}\right) D_j(s) ds \ . \tag{33}$$

We distinguish between two cases: no presynaptic spike occurred since the last postsynaptic spike, or at least one presynaptic spike occurred since then.

Case 1: No presynaptic spike occurred since the last postsynaptic spike

Denote $t_j^{(f),last}$ as the time of the last presynaptic spike, and $D_{f,last}$ as its initial condition. In the first case, the algorithm has the form

$$\begin{aligned} w_{ij}(t) &= w_{ij}(t-\Delta t) + \gamma r(t) z_{ij}(t) \ , \\ z_{ij}(t+\Delta t) &= \beta z_{ij}(t) + \big(\zeta_i(t) - \sigma\big)\frac{\lambda}{c}\int_{\hat{t}_i^{(0)}}^{t} e^{-\left\{\frac{t-s}{\tau_m}\right\}}\left[1 - e^{-\left\{\frac{s-t_j^{(f),last}}{\tau_D}\right\}}\big(1 - D_{f,last}\big)\right] \\ &\qquad \times \sum_f \alpha\left(s - t_j^{(f)}\right) ds \ . \end{aligned} \tag{34}$$

Case 2: At least one presynaptic spike occurred since the last postsynaptic spike

In this case, assume that k spikes occurred since $\hat{t}_i^{(0)}$. Denote $t_j^{(0)}$ as the time of last presynaptic spike before $\hat{t}_i^{(0)}$ and denote the appropriate initial condition for $D_j(t)$ for this

spike as D_{s0}. Let $t_j^{(s1)}, ..., t_j^{(sk)}$ denote the presynaptic spike times that occurred in the time interval $[\hat{t}_i^{(0)}, t]$, and denote the appropriate initial conditions for $D_j(t)$ at those times as $D_{s1}, ..., D_{sk}$ respectively. Then the integral in (33) is given by

$$\begin{aligned}
&\int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\frac{t-s}{\tau_m}\right\} \sum_f \alpha\left(s - t_j^{(f)}\right) D_j(s) ds = \int_{\hat{t}_i^{(0)}}^{t_j^{(s1)}} e^{-\left\{\frac{t-s}{\tau_m}\right\}} \sum_f \alpha\left(s - t_j^{(f)}\right) D_j(s) ds \\
&+ \int_{t_j^{(s1)}}^{t_j^{(s2)}} e^{-\left\{\frac{t-s}{\tau_m}\right\}} \sum_f \alpha\left(s - t_j^{(f)}\right) D_j(s) ds + ... + \int_{t_j^{(sk)}}^{t} e^{-\left\{\frac{t-s}{\tau_m}\right\}} \sum_f \alpha\left(s - t_j^{(f)}\right) D_j(s) ds \\
&= \int_{\hat{t}_i^{(0)}}^{t_j^{(s1)}} e^{-\left\{\frac{t-s}{\tau_m}\right\}} \left[1 - e^{-\left\{\frac{s-t_j^{(f),last}}{\tau_D}\right\}} \left(1 - D_{f,last}\right)\right] \sum_f \alpha\left(s - t_j^{(f)}\right) ds \qquad (35) \\
&+ \int_{t_j^{(s1)}}^{t_j^{(s2)}} e^{-\left\{\frac{t-s}{\tau_m}\right\}} \left[1 - e^{-\left\{\frac{s-t_j^{(s1)}}{\tau_D}\right\}} \left(1 - D_{s1}\right)\right] \sum_f \alpha\left(s - t_j^{(f)}\right) ds + ... \\
&+ \int_{t_j^{(sk)}}^{t} e^{-\left\{\frac{t-s}{\tau_m}\right\}} \left[1 - e^{-\left\{\frac{s-t_j^{(sk)}}{\tau_D}\right\}} \left(1 - D_{sk}\right)\right] \sum_f \alpha\left(s - t_j^{(f)}\right) ds \ ,
\end{aligned}$$

and the algorithm can be written using the expressions above.

Explicit expression for $\alpha(t) = q\delta(t)$

In this case,

$$\begin{aligned}
\int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\frac{t-s}{\tau_m}\right\} \sum_f \alpha\left(s - t_j^{(f)}\right) D_j(s) ds &= \int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\frac{t-s}{\tau_m}\right\} \sum_f q\delta\left(s - t_j^{(f)}\right) D_j(s) ds \\
&= \sum_f q \int_{\hat{t}_i^{(0)}}^{t} \exp\left\{-\frac{t-s}{\tau_m}\right\} \delta\left(s - t_j^{(f)}\right) D_j(s) ds \\
&= q \sum_{f: t_j^{(f)} > \hat{t}_i^{(0)}} \exp\left\{-\frac{t - t_j^{(f)}}{\tau_m}\right\} D_j(t_j^{(f)}) ds \ ,
\end{aligned}$$

hence

$$z_{ij}(t + \Delta t) = \beta z_{ij}(t) + \left(\zeta_i(t) - \sigma\right) \cdot \frac{\lambda q}{c} \sum_{f: t_j^{(f)} > \hat{t}_i^{(0)}} \exp\left\{-\frac{t - t_j^{(f)}}{\tau_m}\right\} D_j(t_j^{(f)}) \ .$$

Two remarks:

- The contribution of the depression variable is considered only at presynaptic spiking times (for this α-function).
- An explicit expression for $D_j(t_j^{(f)})$ is available. The expression depends on $t_j^{(f-1)}$.

C.3 MDPs, POMDPs

C.3.1 MDPs

An MDP (Markov Decision Process) is a quartet $\{\mathcal{S}, \mathcal{A}, \mathbf{P}, r\}$ where

- $\mathcal{S}$ is the state space.
- $\mathcal{A}$ is the action space.
- $\mathbf{P}(a) = p_{ij}(a) = \mathbf{P}(x_{t+1} = j | x_t = i, a_t = a)$ is the transition probability when action a is taken.
- $\mathbf{P}_r(r) = \mathbf{P}(r|x, a)$ is the reward distribution when choosing action a in state x.

Additionally, we set

- $\pi : \mathcal{S} \to \mathcal{A}$ (policy) is a mapping from state space to action space.
- $\mathcal{A}(x)$ is the set of available actions from $\mathcal{A}$ in state x.

C.3.2 POMDPs

A POMDP (partially Observed Markov Decision Process) is a sextet $\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathbf{P}_s, \mathbf{P}_r, \mathbf{P}_o$ where

- $\mathcal{S}$ is the state space.
- $\mathcal{A}$ is the action space.
- $\mathcal{O}$ is the observation space.
- $\mathbf{P}_s = p_{ij}^s(a) = \mathbf{P}(x_{t+1} = j | x_t = i, a_t = a)$ is the transition probability when action a is taken.

- $\mathbf{P}_r(r) = \mathbf{P}(r|x, a)$ is the reward distribution.
- $\mathbf{P}_o(y) = \mathbf{P}(y|x)$ is the observation process distribution.

Additionally, set

- $\pi : \mathcal{S} \to \mathcal{A}$ (policy) is a mapping from state space to action space.
- $\mathcal{A}(x)$ is the set of available actions from $\mathcal{A}$ in state x.

Note that the states (hidden variables) form a standard MDP.

References

Baras, D. (2006). Direct policy search in reinforcement learning and synaptic plasticity in biological neural networks. Msc, Technion - IIT. http://www.ee.technion.ac.il/rmeir/BarasThesis06.pdf.

Bartlett, P. L. and Baxter, J. (1999). Hebbian synaptic modifications in spiking neurons that learn. Technical report, Reasearch School of Information Sciences and Engineering, Australian National University.

Baxter, J. and Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Mass.

Bienenstock, E. L., Cooper, L. N., and Munro, P. W. (1982). Theory for the development of neuron selectivity: Oreientation specificity and binocular interaction in visual cortex. *The Journal of Neuroscience*, 2(1):32–48.

Gerstner, W. and Kistler, W. (2002). *Spiking Neuron Models*. Cambridge University Press.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition.

Izhikevich, E. M. and Desai, N. S. (2003). Relating STDP to BCM. *Neural Comp.*, 15(7):1511–1523.

Koch, C. (1999). *Biophysics of Computation*. Oxford University Press.

Konda, V. R. and Tsitsiklis, J. N. (2003). On actor-critic algorithms. *SIAM journal on Control and Optimization*, 42(4):1143–1166.

L. Cooper, N. Intrator, B. B. and Shouval, H. Z. (2004). *Theory of Cortical Plasticity*. World Scientific, Singapore.

Rao, R. and Sejnowski, T. (2001). Spike-timing-dependent Hebbian plasticity as temporal difference learning. *Neural Comput*, 13(10):2221–2237.

Richardson, M. J. E., Melamed, O., Silberberg, G., Gerstner, W., and Markram, H. (2005). Short-term synaptic plasticity orchestrates the response of pyramidal cells and interneurons to population bursts. *Journal of Computational Neuroscience*, 18(3):323–331.

Schultz, W. (2002). Getting formal with dopamine and reward. *Neuron*, 36(2):241–63. 290.

Seung, H. S. (2003). Learning in spiking neural networks by reinforcement of stocahstic synaptic transmission. *Neuron*, 40:1063–1073.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Inoriduction*. MIT Press.

Toyoizumi, T., Pfister, J., Aihara, K., and Gerstner, W. (2005). Generalized Bienenstock-Cooper-Munro rule for spiking neurons that maximizes information transmission. *Proc Natl Acad Sci U S A*, 102(14):5239–5244. 401.

Wörgötter, F. and Porr, B. (2004). Temporal sequence learning, prediction, and control - a review. *Neural Computation*, 17:1–75.

Xie, X. and Seung, H. S. (2004). Learning in neural networks by reinforcement of irregular spiking. *Physical Review E*, 69:041909.