LETTER Communicated by Bruno Averbeck

Learning Spike-Based Population Codes by Reward and Population Feedback

Johannes Friedrich
friedrich@pyl.unibe.ch
Robert Urbanczik
urbanczik@pyl.unibe.ch
Walter Senn
senn@pyl.unibe.ch
Department of Physiology, University of Bern, CH-3012 Bern, Switzerland

We investigate a recently proposed model for decision learning in a population of spiking neurons where synaptic plasticity is modulated by a population signal in addition to reward feedback. For the basic model, binary population decision making based on spike/no-spike coding, a detailed computational analysis is given about how learning performance depends on population size and task complexity. Next, we extend the basic model to *n*-ary decision making and show that it can also be used in conjunction with other population codes such as rate or even latency coding.

1 Introduction

Behavioral decision making in the brain results from processes occurring at widely differing spatial scales. In a rough account, one might consider the release of single vesicles as the most microscopic level, with the actions of many vesicles being aggregated into a total synaptic release. Disregarding dendritic processing, the next level is a neuron aggregating many synaptic releases into a postsynaptic response. Next, in population coding (Pouget, Dayan, & Zemel, 2000; Averbeck, Latham, & Pouget, 2006), such postsynaptic neuronal responses are again aggregated by a readout that may be a decision-making circuitry (Wang, 2002).

In contrast to this, in mathematical theories of reinforcement learning the learning system is just a collection of elementary entities called agents (Williams, 1992). Each agent responds to a stimulus with an action, and the set of these actions determines whether the collective behavior is rewarded. Reward generates a feedback signal driving the adaptation of the agents. Due to this reinforcement procedure, the system can evolve toward a state-maximizing reward, even if the agents are oblivious of each other's actions and of how the collective behavior determines reward.

There is obviously considerable flexibility in mapping these mathematical concepts onto the structures in the brain. While considering the vesicles as corresponding to the agents seems far-fetched, initial work on reinforcement learning in spiking neurons treated the synapses as agents (Seung, 2003), whereas in later work, the agents were assumed to be the neurons (Fiete & Seung, 2006; Pfister, Toyoizumi, Barber, & Gerstner, 2006; Florian, 2007; Izhikevich, 2007). Most recently, for decision learning, we have suggested treating an entire feedforward population of neurons as an agent (Urbanczik & Senn, 2009). In all of these mappings, the agents adapt by modification of synaptic strengths, but the different approaches result in plasticity rules that differ greatly in biophysical mechanism and in learning performance. In particular, the synapse-agent approach leads to a learning rule where synaptic plasticity is determined only by the presynaptic firing times (besides reward). In neuron-agent approaches (e.g., Pfister et al., 2006), plasticity is also modulated by postsynaptic firing. Simulation results for single-neuron learning indicate that such pre-post rules yield a dramatic performance increase compared to pre-only rules (Gütig & Sompolinsky, 2006). Analytically, similar results have been obtained for linear neurons by Werfel, Xie, and Seung (2005). On the next level, synaptic plasticity in the population-agent approach is also modulated by a feedback signal encoding the population response. Again, learning in the feedforward population is faster with the population-agent approach, compared to treating population neurons as single agents. In particular for the population-agent procedure, learning speeds up with increasing population size, whereas it becomes slower and slower with the neural-agent approach (Urbanczik & Senn, 2009).

In neurons, the backpropagating action potential provides a near-instantaneous record of postsynaptic firings that can be read out by a synapse to modulate plasticity. As a consequence, neuron-agent learning can in principle be used even when the delivery of reward is contingent on the production of precisely timed postsynaptic spikes. In contrast, putative delivery mechanisms for a population response, such as changes in neurotransmitter concentrations (Matsuda, Marzo, & Otani, 2006; Seol et al., 2007) or axonal backpropagation (Harris, 2008), operate on much longer timescales, placing restrictions on the postsynaptic code that can be used in conjunction with population-agent learning.

The purpose of this letter is to provide a detailed computational analysis of population-agent learning. We first revisit the framework in which the approach was introduced: a single population with a binary decision-making readout assuming a postsynaptic spike/no-spike code. For this case, we provide a detailed account of the scaling properties with respect to both the population size and the dimensionality of the spike patterns representing the stimuli. We next demonstrate that n-way decision problems can be learned by using more than one population. Finally, we investigate the use of different postsynaptic coding strategies for the population neurons

and show that the approach can be used in conjunction with a standard rate code but also with a latency code. However, for the latency code, a modification of the synaptic plasticity rule is needed. So in contrast to the case of lower-level agents, in population learning there is a link between how a neuron codes and its synaptic plasticity rule.

2 The Population Model

2.1 Single Neuron Model. Our model population consists of the escape noise neurons introduced by Pfister et al. (2006): leaky integrate-and-fire neurons with an instantaneous firing rate that is a function of the somatic potential. In response to an input spike pattern X, the escape neuron produces an output spike train Y with a probability $P_w(Y \mid X)$ modulated by its synaptic weight vector w.

In more detail, an input spike train X_i (represented as a set of spike times) for the ith afferent of the neuron leads, in spike response parlance, to a postsynaptic potential given at time t by

$$\mathrm{PSP}_i(t; X) = \sum_{s \in X_i} \epsilon(t - s).$$

The output spike train Y (also a set of spike times) generates a reset potential given by

$$\mathrm{RP}(t; Y) = \sum_{s \in Y} \kappa(t - s),$$

and, in combination, the resulting somatic potential is

$$u(t) = U_{\mathrm{rest}} - \mathrm{RP}(t; Y) + \sum_i w_i \, \mathrm{PSP}_i(t; X).$$

For the postsynaptic response kernel, we assume $\epsilon(t) = (e^{-t/\tau_{\mathrm{m}}} - e^{-t/\tau_{\mathrm{s}}})/(\tau_{\mathrm{m}} - \tau_{\mathrm{s}})$ if t is positive; otherwise, $\epsilon(t) = 0$. We use $\tau_{\mathrm{m}} = 10$ for the membrane time constant and $\tau_{\mathrm{s}} = 1.4$ (here and in the sequel, time is in milliseconds). The reset kernel is $\kappa(t) = e^{-t/\tau_{\mathrm{m}}}$ for $t > 0$; otherwise, $\kappa(t) = 0$. The value of the resting potential is $U_{\mathrm{rest}} = -1$ (arbitrary units).

The likelihood of producing the postsynaptic spike train Y is governed by a stochastic intensity $\phi(u(t))$, so the probability that the neuron fires at time t, that is, $t \in Y$, is given by $\phi(u(t))\,\delta t$, where δt represents an infinitesimal time window. (In the simulation, we use $\delta t = 0.2$ and $\phi(u) = e^{5u}/100$.) As Pfister et al. (2006) showed, we now have

$$\log P_w(Y \mid X) = \sum_{t \in Y} \log \phi(u(t)) - \int_0^T \mathrm{d}t \, \phi(u(t)) \tag{2.1}$$

for an observation period running from $t = 0$ to $t = T$. To keep this letter self-contained, a derivation of this is given in appendix B. In reinforcement learning, we need the gradient of $\log P_w(Y \mid X)$ with respect to the synaptic strength (Williams, 1992). For the escape noise neuron,

$$\frac{\partial}{\partial w_i} \log P_w(Y \mid X) = \int_0^T \mathrm{d}t\, \Psi_i(t) \qquad \text{with}$$

$$\Psi_i(t) = \left(\sum_{s \in Y_t} \frac{\phi'(u(t))}{\phi(u(t))}\, \delta(t-s) \; - \phi'(u(t)) \right) \mathrm{PSP}_i(t; X). \quad (2.2)$$

2.2 Population Model. In our population model, N escape noise neurons receive highly correlated inputs coming from an input layer with M sites, where each site projects with probability $p = 0.8$ onto each of the neurons. So a given stimulus, corresponding to a fixed spike pattern X presented in the input layer is seen by the νth neuron as a spike pattern X^ν consisting (on average) of $0.8M$ of the M spike trains in X. Of course, full connectivity ($p = 1$) would likely enhance learning, but using $p = 0.8$ takes into account that it seems unrealistic that all neurons in a biological population receive exactly the same input.

The postsynaptic spike trains Y^ν ($\nu = 1, \ldots, N$) elicited by a stimulus will differ from one neuron to the next, and a coding procedure is needed to combine the diverse responses into a single population decision. For this, we assume that each Y^ν is read out as a numerical value given by a scoring function $c(Y^\nu)$. In a rate code, for instance, $c(Y^\nu)$ might just count the number of spikes in Y^ν. Technically, however, it is convenient if the values of $c(Y^\nu)$ are balanced around zero. So in the sequel, in a rate code, $c(Y^\nu)$ will be the difference between the number of spikes in Y^ν and an appropriate threshold value. The population readout can now aggregate the postsynaptic responses based on simply summing the scores. In particular, we assume that the population decision D, determining the behavioral decision, is obtained as

$$D = \mathrm{sign}\left(\sum_{\nu=1}^{N} c(Y^\nu) \right). \quad (2.3)$$

When one and the same stimulus X is presented in different trials, the population decision may fluctuate due to the noisy neural processing. The likelihood of this happening can be assessed based on the population activity

$$A = \frac{1}{\sqrt{N}} \sum_{\nu=1}^{N} c(Y^\nu). \quad (2.4)$$

The neurons are conditionally independent given a stimulus, so the conditional distribution $P_w(A \mid X)$ becomes gaussian in a large population, and its variance stays on the order of 1. Consequently, since $D = \text{sign}(A)$, the population decision is unlikely to fluctuate if the conditional mean of A, $E(A \mid X)$, has a large absolute value. In this sense, the magnitude of A provides a measure for the reliability of the decision D.

2.3 Population Learning. The population decision elicits reinforcement feedback encoded in a binary variable R, with $R = \pm 1$ signaling that the decision was correct or incorrect. In the neuron-agent approach, this global feedback leads to the following learning update:

$$\Delta w_i^\nu = \eta\,(R - b)\frac{\partial}{\partial w_i^\nu} \log P_{w^\nu}(Y^\nu \mid X). \tag{2.5}$$

Here w^ν is the vector of the synaptic strengths of neuron ν, and the derivative with respect to w_i^ν is obtained by applying equation 2.2 to each of the N neurons. The learning rate η and the reinforcement baseline b are parameters that can be tuned to optimize performance.

As mentioned in section 1, learning slows down with increasing population size in this approach. The reason becomes apparent when one considers binary codes where $c(Y^\nu) = \pm 1$. The overall decision D can now be regarded as representing the majority decision of the population neurons. Consequently a dissenting neuron should not get the same reward as a majority neuron; indeed, if $c(Y^\nu) = -D$, the appropriate reinforcement is $-R$, not R. In general terms, the appropriate reinforcement for neuron ν is $RDc(Y^\nu)$. This differs from neuron to neuron, but the update, equation 2.5, uses the same reinforcement R for all neurons. Further, for generic choices of the synaptic strengths, the population vote will initially tend to be split almost evenly. Then the global reward signal R provides the wrong reinforcement for nearly half of the population neurons.

In Urbanczik and Senn (2009, Supplementary Information), this has led us to mathematically derive a new gradient learning rule for the task. For this (stochastic) gradient rule, feedback about reward R and population activity A modulates synaptic plasticity just when the stimulus ends. Such instantaneous feedback is biologically unrealistic, so to allow for delays in feedback delivery, we also considered the modified, approximate gradient rule described below.

A key finding in the new approach is that plasticity should be modulated by the neuron-specific reward signal $RDc(Y^\nu)$, so the neuron-agent learning problem could be overcome by assuming a second feedback signal encoding the population decision D. It turns out, however, that it is even better if graded information about the population activity A, and not just $D = \text{sign}(A)$, is fed back. The reason is that in a large population, each neuron does not have to respond correctly to each stimulus, and, in fact, it is desirable to have a division of labor between the neurons. If plasticity is

based on the graded feedback A, in case of a correct decision, learning can be attenuated (or even stop completely) once $|A|$ becomes sufficiently large to ensure a reliable population decision.

The synaptic quantity in the proposed rule is not $\frac{\partial}{\partial w_i^\nu} \log P_{w^\nu}(Y^\nu \mid X)$ because the computation requires an integration over the duration of the stimulus (see equation 2.2). Synapses are unlikely to know when stimuli start and end, so, for the sake of biological realism, the model instead uses the low-pass-filter approximation

$$\tau_{\mathrm{M}} \dot{E}_i^\nu = -E_i^\nu + \Psi_i^\nu(t). \tag{2.6}$$

Here $\Psi_i^\nu(t)$ is the integrand introduced in equation 2.2, and τ_{M} is a time constant that should be at least roughly matched to stimulus duration. We assume 500 ms stimuli and, accordingly, $\tau_{\mathrm{M}} = 500$. In the model, each neuron has a memory mechanism encoding some information about its response Y^ν to the stimulus. For this, a calcium-like variable $\tilde{Y}^\nu$ is introduced and is set to 1 each time neuron ν fires; at other times, it decays exponentially. Formally,

$$\tau_{\mathrm{M}} \dot{\tilde{Y}}^\nu = -\tilde{Y}^\nu(t) + \tau_{\mathrm{M}} \sum_{s \in Y^\nu} (1 - \tilde{Y}^\nu(s))\, \delta(t - s). \tag{2.7}$$

Note that the time constant is the same as for E_i^ν, since for both, the relevant timescale is typical stimulus duration.

Information about the reinforcement R and the population activity A is delivered to the synapses via continuously changing signals $\tilde{R}(t)$ and $\tilde{A}(t)$. The signals will be described in detail below, together with the following population learning rule that changes synaptic strength in continuous time according to

$$\dot{w}_i^\nu = \eta\, |\tilde{R}|\, (\mathrm{sign}(\tilde{R}\tilde{A}\,\mathrm{c}(\tilde{Y}^\nu)) - 1) a(\tilde{R}, \tilde{A})\, E_i^\nu. \tag{2.8}$$

Here, to reduce clutter, we do not make the time dependence of the quantities on the right-hand side explicit in the notation. For a trial ending at time T, $\tilde{R}(t)$ will be negligibly small except during a time window of some 100 ms after time T. So the modulation with $|\tilde{R}|$ in equation 2.8 in effect confines synaptic plasticity to a short time window after trial ending. The feedback signals $\tilde{R}$ and $\tilde{A}$ are such that in this time window, (1) $R = \mathrm{sign}(\tilde{R}(t))$ and (2) $D = \mathrm{sign}(\tilde{A}(t))$. Ideally we also have a readout function $\mathrm{c}(\tilde{Y}^\nu)$ for the neuronal memory trace, satisfying (3) $c(Y^\nu) = \mathrm{c}(\tilde{Y}^\nu(t))$. If the three conditions hold during the poststimulus time window, the $\mathrm{sign}(\tilde{R}\tilde{A}\,\mathrm{c}(\tilde{Y}^\nu))$ term in the update rule is equal to $RDc(Y^\nu)$; that is, it computes the correct reward signal for neuron ν. Subtracting 1 from this term amounts to choosing the reinforcement baseline such that synaptic changes occur only to correct a

neuronal response deemed wrong. Finally, $a(\tilde{R}, \tilde{A})$ provides for attenuating the learning. This is achieved by choosing

$$a(\tilde{R}, \tilde{A}) = \begin{cases} 1 & \text{for } \tilde{R}(t) \leq 0 \\ |\tilde{A}(t)| & \text{for } \tilde{R}(t) > 0 \end{cases}.$$

Due to the first clause, no attenuation of learning occurs if the population decision was wrong. In the model, the magnitude of the population feedback $|\tilde{A}(t)|$ is proportional to e^{-A^2}, so the second clause leads to a strong reduction in the effective learning rate in case of a correct and reliable population decision.

In the detailed modeling for $\tilde{R}$ and $\tilde{A}$, the signals are taken to be delivered by changes in ambient concentration levels of neurotransmitters. Reinforcement is assumed to change the release rate of a neurotransmitter (e.g., dopamine) for a duration of 50 ms. Due to a linear degradation process, this has only a transient effect on the concentration level, which in due course returns to baseline. $\tilde{R}(t)$ is the difference between the current and the baseline concentration of the neurotransmitter and is given by

$$\tau_{\text{rew}} \dot{\tilde{R}} = -\tilde{R} + R\, \mathbb{1}_{50}(t - T). \tag{2.9}$$

Here $\mathbb{1}_{50}(s)$ is the indicator function on the interval $[0, 50]$ = zero unless $0 \leq s \leq 50$, in which case $\mathbb{1}_{50}(s) = 1$. Population feedback is delivered similarly by a second neurotransmitter. Since this signal should depend on the magnitude of the population activity, we use

$$\tau_{\text{pop}} \dot{\tilde{A}} = -\tilde{A} + \gamma D e^{-A^2} \mathbb{1}_{50}(t - T), \tag{2.10}$$

where γ is a positive parameter. In case of reward, this parameter modulates the value of the attenuation factor $a(\tilde{R}, \tilde{A})$ in equation 2.8. The choice of γ thus tunes the balance between the learning from correct and incorrect population decisions (see appendix A).

2.4 Learning Task and Simulation Protocol. The description above of the plasticity rule is couched in terms of learning a single stimulus-response relationship. Of course, ultimately, the task is to learn multiple (m) such relationships, where each stimulus, represented as a particular input spike pattern, has a specific target value for the population decision. For this, the m spike patterns are presented in random order to the network (with no delay period between the successive trials). The neuronal and synaptic learning equations 2.6 to 2.8 are simply used verbatim, that is, the population neurons and their synapses are oblivious of stimulus boundaries. Only the interpretation of the equations describing feedback delivery, equations 2.9 and 2.10,

changes slightly. These are now valid only up to the time T' when the presentation of a subsequent stimulus X' ends. From then on, in equations 2.9 and 2.10, the values for reinforcement and population response pertaining to X' are used (and T is replaced by T').

The m input spike patterns used in each task are statistically independent with an equal number of patterns for each target decision. Each spike pattern is made up of M independent 5 Hz Poisson spike trains with 500 ms duration.

3 Simulation Results

3.1 Scaling Properties of Population Learning. Here we quantify the speed-up of learning with increasing population size N in dependence on the dimensionality M of the spike patterns and of the number m of the patterns to be learned. We assume that the population is read out based on a spike/no-spike code, that is, $c(Y^\nu) = -1$ if neuron ν did not fire at all in response to the stimulus, and otherwise $c(Y^\nu) = 1$. For the neuronal readout of the memory trace, we use

$$\mathrm{c}(\tilde{Y}^\nu(t)) = \mathrm{sign}(\tilde{Y}^\nu(t) - \vartheta). \tag{3.1}$$

For an appropriate choice of the threshold ϑ, namely, $\vartheta = e^{-1}$, we have $c(Y^\nu) = \mathrm{c}(\tilde{Y}^\nu(T))$ at time T when the stimulus ends; we then have a perfect match between the population readout and the neuronal readout. This is one reason for adopting such a reduced, binarized version of a rate code. A related reason is that for the binary code, the update rule, equation 2.8, can be mathematically understood as a gradient rule in the limit that feedback delivery is instantaneous after stimulus presentation (Urbanczik & Senn, 2009, Supplementary Information). In the simulations, of course, feedback delivery is not instantaneous. Hence, a slightly smaller value of the threshold ϑ seems appropriate; we use $\vartheta = e^{-1.1}$ to account for the fact that the bulk of the synaptic changes induced by equation 2.8 takes place some 50 ms after a trial has ended.

To assess performance, we counted the total number $\mathcal{E}$ of erroneous population decisions during the repeated trials needed to learn a task. As can be seen in Figure 1a, the error count $\mathcal{E}$ does not diverge as the number of trials is increased, but due to learning, it converges toward an asymptotic value. For a perfect one-shot learner, the asymptotic value would on average be $\mathcal{E} = \frac{1}{2}m$, since for half of the m patterns, the initial guess happens to be the right one and since the wrong responses on the remaining patterns are instantly corrected (without introducing new mistakes). While in our simulations, $\mathcal{E}$ decreases markedly with increasing population size, the error counts we were able to obtain with population learning remain higher than the one-shot value.

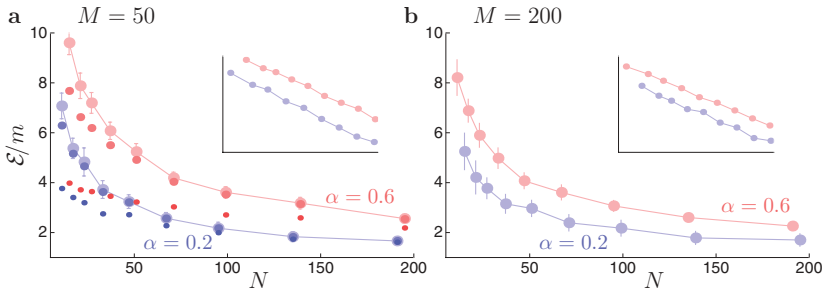

Figure 1: Speed-up of learning with population size N for different input dimensions M and different numbers $m = \alpha M$ of patterns to be learned in each task. Note that for a given N, the parameter α determines the synaptic load. The total number of errors $\mathcal{E}$ was counted based on $40m$ trials (large circles). (a) To demonstrate convergence, the $\mathcal{E}$-values after $25m$ trials (medium circles) and $10m$ trials (small circles) are also shown. In the insets, the total error counts in the $40m$ trials are replotted as $\log(\frac{\mathcal{E}}{m} - \frac{1}{2})$ versus $\log N$. Data points are averages over 20 runs with different initial conditions and different task instances. Error bars show 1 SEM of the mean in a, but in b, they show the standard deviations for the fluctuations from run to run. The simulation setup is detailed in section 2.4.

When learning $m = \alpha M$ patterns with a given number N of population neurons, the parameter α determines the synaptic load. Hence, for fixed N and α, one expects total learning time to scale linearly with the dimensionality M of the input. Figure 1b shows the result obtained when rerunning the simulations for Figure 1a with $M = 200$ instead of $M = 50$. The error counts per pattern in the two panels are very similar, confirming the linear scaling.

It has been previously shown that on average, single-neuron performance does not increase with population size and may in fact decrease slowly (Urbanczik & Senn, 2009). So the speed-up in learning occurs because in a larger population, errors of single neurons are less likely to corrupt the population decision. An additional benefit of this, besides improved average population performance, can be read of from Figure 1b. There, in contrast to the 1 SEM values of Figure 1a, the error bars show the fluctuations in performance from run to run (with different sets of stimulus-response pairs and different initial synaptic strengths). One reason for the difference in presentation between Figures 1a and 1b is that for $M = 200$, the 1 SEM values would hardly exceed the size of the symbols. More important, the error bars shown in Figure 1b highlight that the fluctuations in learning performance decrease with population size; that is, it becomes less likely that learning happens to become unexpectedly slow for a certain set of patterns. This is a definite advantage when failing to learn a specific task incurs a severe penalty (e.g., getting eaten).

When replotted on logarithmic scales, the data points for different population sizes fall nearly on a straight line (insets). This may be indicative of an algebraic decay of $\mathcal{E}$ with increasing N toward the $\mathcal{E} = \frac{1}{2}m$ value for one-shot learning. But convergence is rather slow; extrapolation from the simulation data suggests that some 10,000 neurons are needed to achieve $\mathcal{E} = 0.6m$ (for $\alpha = 0.2$, $M = 50$).

The analysis presented is based on the cumulative error count $\mathcal{E}$ and not on estimating population performance by a running average similar to equation 3.2 below. Since the latter approach is more customary, a few words of explanation are in order. In choosing the time window (or time constant) for running average estimation, there is a bias-variance trade-off because for a large time window, the estimator on average lags behind the true performance, whereas the variance is large with a short time window. The trade-off becomes troublesome when learning is so fast that it approaches one-shot learning, even if the main goal is just to determine performance after a given number of trials. But for comparing the learning speed in different settings as we do, the statistical problem in fact is to estimate the number of trials needed to achieve a given performance level; for this, the bias-variance trade-off is even more pernicious. Say, we run the simulations up to the time need to achieve a 99% probability of correct decisions, as indicated by our running mean estimator and then record this stopping time. Now, stopping will likely be triggered by run-to-run fluctuations for which the running mean is unexpectedly large. This creates a stopping bias toward underestimating the required learning time. But due to the lag introduced by the time window, we have a bias toward overestimating the learning time. So the net bias in the stopping time depends in a complicated manner on the length of the time window, the number m of patterns, and the learning speed of the system. Its direction is hence different in the different settings, and this makes it difficult to compare learning times. In contrast, the bias in the cumulative error count $\mathcal{E}$ is relatively easy to control for by checking for convergence, as shown above. Further, the bias is uniform, in that it will tend to underestimate the number of errors committed by slow learners. This means that with regard to the speed-up of learning with population size, the above findings are conservative because we are, if anything, underestimating $\mathcal{E}$ in the case of small populations.

3.2 Flexibility of the Online Scheme. Behavioral adaptation takes many forms, and there is certainly more to reinforcement learning than just binary decisions based on a spike/no-spike code. Hence, we provide examples showing that the above scheme can be applied as well to some other learning scenarios with little or no modification.

We first consider the case that more than two responses to a stimulus are possible. This can be addressed by assuming several (n) populations of neurons, each responding with a binary decision to a stimulus. The behavioral response is then determined by the combined output of the populations and

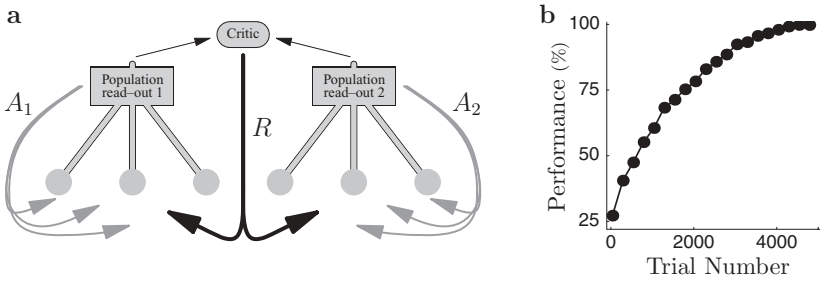

Figure 2: Learning of four-way decisions by two populations. (a) Sketch of the feedback structure used for the task. (b) Learning curve for two populations with $N = 33$ neurons each. The number of patterns to be learned was $m = 24$, with six patterns allocated to each of the four output classes. The reported values are for $M = 50$, averaged over 40 runs.

can thus have one of 2^n values. The global reinforcement signal R encodes whether this combined output is correct. Now if each of the n populations has its individual population feedback, the above online procedure can simply be used for each of the populations. So as sketched in Figure 2a, learning at the level of the single neuron is based on its own response to the stimulus, on feedback about the output of the population it belongs to, and on the global reinforcement assessing the behavioral response. Simulation results for two populations learning a four-way decision task are shown in Figure 2b. The performance percentages are computed as a running mean $\bar{p}$ that is updated after each pattern presentation as

$$\bar{p} \leftarrow (1 - \lambda)\bar{p} + \lambda p. \tag{3.2}$$

Here $p = 100\%$ if the presented stimulus was classified correctly; otherwise, $p = 0$. The timing parameter was set to $\lambda = 0.01$.

Next we investigate the use of different coding strategies at the level of the single neuron. Until now, we have assumed that in decoding postsynaptic spike trains, the population readout considers only whether the neuron does or does not fire. While this suggests itself for its theoretical simplicity, the readout is somewhat involved, since for this spike/no-spike code, a neuron firing more than once in response to the stimulus should have the same impact on the population decision as a neuron that emits just a single spike. In contrast, for a rate code, only the total number of spike emitted by the population needs to be considered. To obtain a proper firing rate code, we redefine the scoring function $c(Y^\nu)$ used by the population readout to be

$$c(Y^\nu) = \mathrm{card}(Y^\nu) - \frac{2}{3},$$

where $\mathrm{card}(Y^\nu)$ denotes the number of spikes emitted by neuron ν in response to the stimulus. To balance $c(Y^\nu)$ around zero, we have subtracted $\frac{2}{3}$,

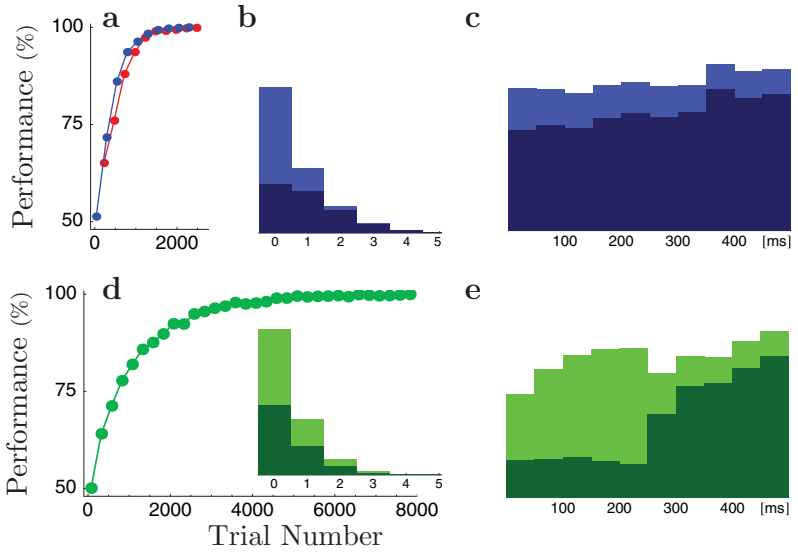

Figure 3: Different coding strategies for a binary decision task. (a) Learning curve (blue) when the population readout assumes a firing rate code ($N = 33$ and 30 patterns). For comparison, the corresponding curve for the spike/no-spike code is shown in red. For both curves, the performance values are obtained with equation 3.2. After learning with the firing rate code: (b) distribution of the number of spikes in response to a stimulus; (c) distribution of the spike times within stimulus duration. In panels b and c, dark (light) blue gives the contribution from the patterns with target output 1 (-1). (d) Learning curve for the spike-early/spike-late code ($N = 67$, 30 patterns) and the corresponding spike number histogram (d, inset) and spike timing histogram (e). Dark (light) green is for target 1 (-1) patterns. The values reported are averages over 30 runs with $M = 50$.

not $\frac{1}{2}$. This choice takes into account that a neuron may occasionally spike more than once in response to a stimulus. As the learning curve in Figure 3a shows, using a firing rate code instead of the spike/no-spike code has only a minor effect on performance. For simplicity, we are still using the same plasticity rule as for the spike/no-spike code, although the exact number of postsynaptic spikes cannot be determined from $\tilde{Y}^\nu$. But given the low activity level (see Figure 3b), the limited information about the postsynaptic rate readout from $\tilde{Y}^\nu$ by equation 3.1 is sufficient for learning to succeed.

We are assuming throughout that inputs are fixed low-activity spike patterns, so the neuronal outputs are highly dependent on relative input spike times. But the two output codes considered up to now do not take postsynaptic spike timing into account, leading to a code switch between inputs and outputs. While this could be avoided by assuming mean firing

rate inputs, it is of interest to ask if population learning itself can be based on a spike-timing-dependent output code. For this, we now study a spike-early/spike-late code. In particular, we use as a scoring function $c(Y^\nu) = 1$ if there are more spikes in Y^ν during the second half of stimulus duration than during the first; otherwise, $c(Y^\nu) = -1$. (For an equal number of spikes and in the case of no spike, $c(Y^\nu) = 0$.)

As shown in Figure 3d, population learning can be based on such a spike-timing-dependent output, even if it is slower than for the rate codes. The spike-early/spike-late coding is easily seen as the difference between target 1 and target -1 patterns in the timing histogram (see Figure 3e) whereas firing rates do not distinguish between the two target classes (see Figure 3d, inset).

For learning with the timing-dependent code, the readout of the neuronal memory traces had to be modified. The first change is a larger value for the threshold in equation 3.1, since the comparison of the calcium-like variable $\tilde{Y}^\nu$ to ϑ must now reflect the spike-early/spike-late decision instead of the spike/no-spike decision. In addition, the case that the spike-early/spike-late scoring function yields zero because the stimulus elicited no postsynaptic spike has to be addressed. For this, we set $\mathrm{c}(\tilde{Y}^\nu)$ to zero when $\tilde{Y}^\nu$ is small by replacing equation 3.1 with

$$\mathrm{c}(\tilde{Y}^\nu(t)) = \mathrm{sign}(\tilde{Y}^\nu(t) - \vartheta)\,\Theta\,(\tilde{Y}^\nu(t) - \hat{\vartheta}), \tag{3.3}$$

where $\vartheta = e^{-0.55}$, $\hat{\vartheta} = e^{-1.1}$, and Θ is the Heaviside step function.

Even using equation 3.3, the value of $\mathrm{c}(\tilde{Y}^\nu)$ will often differ from the correct value $c(Y^\nu)$ if two or more postsynaptic spikes are produced in response to a stimulus. The spike-early/spike-late code is simply too complicated to be exactly represented by a single concentration variable $\tilde{Y}^\nu$. Nevertheless, this is addressed rather simply by introducing noise to make the errors less systematic. In particular, instead of updating $\tilde{Y}^\nu$ to 1 whenever neuron ν produces a postsynaptic spike at time t, this update is skipped with a probability $\tilde{Y}^\nu(t)$, that is, $\tilde{Y}^\nu$ simply ignores some postsynaptic spikes. This stochastic dithering helps because it decorrelates the deviations between $\mathrm{c}(\tilde{Y}^\nu)$ and $c(Y^\nu)$ from the underlying spike train Y^ν without destroying the correlation of $\mathrm{c}(\tilde{Y}^\nu)$ with $c(Y^\nu)$.

The memory trace with stochastic dithering is more flexible than the deterministic one and can also be used in conjunction with the other coding strategies. The flexibility, however, comes at a price. For the spike/no-spike code, we observed a twofold increase in learning times when using the stochastic instead of the deterministic memory trace (data not shown).

4 Discussion

We have presented a computational analysis of reinforcement learning in populations of spiking neurons when synaptic plasticity is modulated not

just by global reinforcement but also by feedback about the population response, as well as by a memory trace encoding a neuron's past firing behavior. Learning now speeds up with increasing population size, in contrast to the case where only global reinforcement is available. For the simulations, we have assumed as specific neuronal model the escape noise neuron (Pfister et al., 2006). This suggests itself because the synaptic variable in the plasticity rule depends on the relative timing of pre- and postsynaptic spikes, as well as on the somatic potential. But our population approach is not confined to this model and could readily be adapted to use other reinforcement learning procedures at the single neuron level (Seung, 2003; Fiete & Seung, 2006; Florian, 2007). Obviously, in absolute terms, population performance will depend on the specifics of the neuronal model and associated plasticity rule. But considering the scaling of the performance, we expect our findings to be generic: with just global reinforcement, the performance degrades with increasing population size, but it improves when plasticity is properly modulated by the population response and the neuronal memory trace.

In the scaling analysis, we have focused on learning time as function of population size N. One might, of course, also ask how the maximal number of stimulus-response association that can be memorized depends on N. Results from statistical physics for related architectures may suggest that this storage capacity increases with population size. There (Monasson & Zecchina, 1996; Urbanczik, 1997), a slightly supralinear increase with N was found for the case that the population neurons are perceptrons and not spiking neurons. But at the capacity limit, almost by definition, there is no redundancy in the system. So the capacity will depend strongly on minute details such as synaptic accuracy. Also, population decisions become finely balanced as the limit is approached and thus rely critically on the performance of each single neuron. The main biological interest in population coding, however, is that it provides a means to aggregate fluctuating but redundant neuronal responses into reliable decisions. This is why we have considered only values of the load parameter α, which are likely to be far below the technical capacity limit.

The model assumes a fixed population readout with plasticity confined to the population neurons that learn to identify salient features in the input. Pouget et al. (2000) have pointed out that learning is much simpler if one assumes an architecture that in essence is dual to ours: plasticity confined to the readout with the population neurons themselves just serving as fixed feature detectors. This seems adequate for basic sensory-motor integration tasks when stimuli can be described by a few features such as spatial location or angle. Further, in this case, the topographically organized lateral connectivity observed in sensory areas can provide an effective way of damping neuronal response variability. But for learning complex stimuli, a prohibitively large population size is necessary if just the readout is plastic. In particular, for feature neurons with gaussian tuning curves, the flexibility

of readout learning is severely compromised unless the population size increases exponentially with the stimulus dimension (Pouget et al., 2000). It thus seems unlikely that plasticity in higher cortical areas is confined to a population readout.

Further, the population-agent approach described here addresses head-on a problem implicit in many mean firing rate models of learning. A rate in such models is often taken to represent an ensemble average over a population of spiking neurons, since interpreting the rate as a temporal average over a single neuron would lead to unrealistically slow information processing. But for an ensemble average, even in cases where the rate description of the spiking population is itself carefully established (Fusi, Asaad, Miller, & Wang, 2007), it is often unclear what the mean firing rate plasticity rule actually means at the level of the single-spiking neuron. In particular, an interpretational conundrum arises when plasticity is modulated by the mean postsynaptic rate, as in Hebbian learning. Since this rate is really a population average, it is not immediately available at the synaptic level and may differ from the true postsynaptic behavior of any single neuron. Our model resolves this conundrum by explicitly describing a biophysically reasonable delivery mechanism for the population response and showing how differences between the population-averaged rate and the actual neuronal rates can be resolved when each neuron keeps a memory trace of its recent spiking behavior.

We have not considered here how the postsynaptic code is read out. To monitor the population for the duration of the stimulus, a neural integrator is needed. This is likely to involve a combination of cellular mechanisms (e.g., plateau potentials) and recurrent network connectivity (Major & Tank, 2004). In addition, for a spike-timing-dependent output code, the time elapsed since stimulus onset would have to be measured. While this can be seen as a special case of a neural integrator, dedicated implementations of such neural clocks are also possible (Durstewitz, 2004; Reutimann, Yakovlev, Fusi, & Senn, 2004). A detailed neuronal model based on two competing integrators has been presented in Wang (2002) for reading out a binary decision encoded in the accumulated activity difference between two input populations. This decision circuitry readily specializes to the single population case considered here, when the neurons code by firing rate. Just assume that the population projects to one of the integrators, whereas to provide the threshold, the other integrator receives input at a fixed rate.

Our population learning model offers a functional interpretation for the abundant experimental observations that synaptic plasticity is regulated by various neuromodulators (see Foehring & Lorenzon, 1999; Centonze, Gubellini, Pisani, Bernardi, & Calabresi, 2003, for reviews); in different combinations, some neuromodulators can even switch the polarity of the long-term synaptic change (Matsuda et al., 2006; Seol et al., 2007). But the model also makes the general prediction that synaptic plasticity should depend on postsynaptic coding. In its basic form, the learning rule for

the escape noise neuron just changes the probability that a postsynaptic spike train generated in response to a stimulus is produced again on a further presentation of the same stimulus. So the neuron can in principle be reinforced to learn any output code—in contrast to, for example, the tempotron (Gütig & Sompolinsky, 2006), where the spike/no-spike code is hardwired into the plasticity rule. However, in population-agent learning, the universality of the escape noise rule is compromised when plasticity is modulated by a comparison between the population response and the neuronal memory trace. The outcome of this comparison depends on the code used in reading out the population, so the local modulatory factor in the plasticity rule must be matched to the postsynaptic code. One of our spike-early/spike-late neurons, for instance, would not be able to learn with the mismatched feedback signals arising from a firing rate readout. Given the high performance of population-agent learning, the need for such matching may be indicative of unavoidable trade-offs between efficiency and universality in plasticity mechanisms. So in dependence on the neuronal code, plasticity mechanisms may differ from one brain area to the next. In fact, our more general finding—that synaptic plasticity should be tailored to the network architecture—suggests further reasons for expecting such differences. Obviously the brain has many subnetworks that do not resemble our population model and will require other plasticity rules. Hence, an interesting question for future theoretical research is to identify other specific network architectures where reinforcement learning speeds up when plasticity is modulated, in addition to the external reward, by a limited number of internally generated feedback signals.

Appendix A: Simulation Details

Here we provide the remaining simulation details. We first discuss the choice of the parameter γ determining the magnitude of the population signal in equation 2.10. In a binary decision task, consider a situation where for a given stimulus, the expected magnitude of A is small. The population decision is then essentially random, and on any single trial, the reward $R = 1$ is as likely as $R = -1$. Hence the magnitude of the synaptic update should not depend on the value of R. In view of equation 2.8, this magnitude is determined by the time integral of $|\tilde{R}|a(\tilde{R}, \tilde{A})$. When choosing $\gamma = 2.5$, the value of this integral is approximately the same for $R = 1$ as for $R = -1$ when A is small. Hence, we used this value for γ in all binary decision tasks. For a four-way decision task, the situation is slightly different because success ($R = 1$) is more informative than failure ($R = -1$), at least initially during learning. To get a larger synaptic update for $R = 1$ in this task, we used $\gamma = 5$.

In the simulations for binary decision making with spike/no-spike and with the rate code, the learning rate was $\eta = \frac{640}{(\alpha+0.2)M}$. For the four-way

decision task, $\eta = 8$ was used, and for the latency code simulations, $\eta = 2$. In all simulations, initial synaptic strengths were chosen from a gaussian distribution with zero mean and standard deviation $40/\sqrt{M}$.

Appendix B: Probability of a Postsynaptic Spike Train

Here we give a derivation of the expression 2.1 for $\log P_w(Y \mid X)$.

Let $Y = \{t_1, t_2, \ldots, t_n\}$ be a spike train with n spikes between T_1 and T_2. To have a unique representation, we assume that the t_i are in ascending order and that, in particular, $T_1 < t_1 < t_2 < \cdots < t_n \leq T_2$. Let $\mathcal{Y}_n$ be the set of all possible spike trains with n spikes between T_1 and T_2. Let ϕ be a (rate) function that takes on nonnegative values.

Now consider the function

$$h(Y) = h(t_1, t_2, \ldots, t_n) = e^{-\int_{T_1}^{T_2} \mathrm{d}t\, \phi(t)} \prod_{i=1}^{n} \phi(t_i). \tag{B.1}$$

Since $h(Y)$ is nonnegative, we can use it as a density to define a measure μ on $\mathcal{Y}_n$ in the standard way, $\mu(\Omega) = \int_\Omega \mathrm{d}t_1 \mathrm{d}t_2 \ldots \mathrm{d}t_n\, h(t_1, t_2, \ldots, t_n)$, for any measurable subset $\Omega \subset \mathcal{Y}_n$. By applying this for different n, we obtain a measure, which we also denote by μ, on the set $\mathcal{Y} = \cup_{n=0}^{\infty} \mathcal{Y}_n$ of all spike trains between T_1 and T_2.

We now determine the measure $\mu(\mathcal{Y}_n)$ of observing exactly n spikes. First,

$$\begin{aligned}\mu(\mathcal{Y}_n) &= \int_{T_1}^{T2_1} \mathrm{d}t_1 \int_{t_1}^{T_2} \mathrm{d}t_2 \int_{t_2}^{T_2} \mathrm{d}t_3 \ldots \int_{t_{n-1}}^{T_2} \mathrm{d}t_n\, h(t_1, t_2, \ldots, t_n) \\ &= \frac{1}{n!} \int_{T_1}^{T_2} \mathrm{d}t_1 \int_{T_1}^{T_2} \mathrm{d}t_2 \ldots \int_{T_1}^{T_2} \mathrm{d}t_n\, h(t_1, t_2, \ldots, t_n),\end{aligned}$$

where the second equality holds because $h(t_1, t_2, \ldots, t_n)$ does not change when permuting the spike times and because there are $n!$ permutations. Now the integral factorizes since the contribution from the different spike times to $h(t_1, t_2, \ldots, t_n)$ is given by a simple product. Hence,

$$\begin{aligned}\mu(\mathcal{Y}_n) &= \frac{1}{n!} e^{-\int_{T_1}^{T_2} \mathrm{d}t\, \phi(t)} \prod_{i=1}^{n} \int_{T_1}^{T_2} \mathrm{d}t_i\, \phi(t_i) \\ &= \frac{1}{n!} e^{-\int_{T_1}^{T_2} \mathrm{d}t\, \phi(t)} \left(\int_{T_1}^{T_2} \mathrm{d}t\, \phi(t) \right)^n . \end{aligned} \tag{B.2}$$

The last line is the textbook definition for the probability that an inhomogeneous Poisson process with rate function ϕ generates n events in the time interval $[T_1, T_2]$, so μ is not just measure on $\mathcal{Y}$ but in fact a probability measure. Further, since equation B.2 holds for all time intervals $[T_1, T_2]$, the

function $h(Y)$ in equation B.1 is the probability density that the spike train Y is generated by the inhomogeneous Poisson process.

In passing, we note that $\mu(\mathcal{Y}_n)$ can be written as $\mu(\mathcal{Y}_n) = e^{-\kappa}\kappa^n/n!$ with $\kappa = \int_{T_1}^{T_2} \mathrm{d}t\, \phi(t)$. This motivates the name *Poisson process*. Further, if ϕ is differentiable, setting $\delta = T_2 - T_1$, we have $\mu(\mathcal{Y}_0) = 1 - \mathcal{O}(\delta)$, that is, we are most likely to observe no event in a small time interval. Also

$$\mu(\mathcal{Y}_1) = \phi(T_1)\delta + \mathcal{O}(\delta^2) \quad \text{and} \quad 1 - \mu(\mathcal{Y}_0) - \mu(\mathcal{Y}_1) = \mathcal{O}(\delta^2).$$

The last equation means that the probability of observing two or more events in a small time interval is negligible, and hence the first equation, giving the probability of a single event, motivates calling ϕ the rate function.

Now, equation 2.1 is obtained from equation B.1 by taking the logarithm and using as a rate function the stochastic intensity computed from the membrane potential. So the above demonstrates equation 2.1 for the case that postsynaptic spikes do not reset the membrane potential. For the case with reset, we write $h(t_1, t_2, \ldots, t_n)$ as $h(t_1, t_2, \ldots, t_n; \phi, T_1, T_2)$ to make all dependencies explicit and denote by ϕ_Y the stochastic intensity in the case that the observed spike train is $Y = \{t_1, t_2, \ldots, t_n\}$. We note that

$$h(t_1, t_2, \ldots, t_n; \phi_Y, T_1, T_2) = h(t_1; \phi_Y, T_1, t_1)\, h(t_2, \ldots, t_n; \phi_Y, t_1, T_2). \tag{B.3}$$

When this decomposition is used, the correctness of equation 2.1 is readily shown for the general case by induction over the number of spikes n. The equation is correct for $n = 0$, (if the neuron does not fire) because then the reset plays no role. It is correct, for the same reason, even for $n = 1$ if the single spike occurs at the end of the observation period; the probability density is then given by $h(t_1; \phi_Y, T_1, t_1)$, the first term on the right-hand side of equation B.3. Now we may by induction assume correctness for the $n - 1$ spikes occurring in the time window between t_1 and T_2, corresponding to the second term on the right-hand side of equation B.3, and thus equation B.3 implies the correctness of equation 2.1 also for the n spikes in the whole observation period T_1 to T_2

Acknowledgments

This work was supported by the Swiss National Science Foundation (SNSF), grant K-32K0-1180, and a grant from the Swiss SystemsX.ch initiative (evaluated by the SNSF).

References

Averbeck, B., Latham, P., & Pouget, A. (2006). Neural correlations, population coding and computation. *Nature Rev. Neurosci., 7*, 358–366.

Centonze, D., Gubellini, P., Pisani, A., Bernardi, G., & Calabresi, P. (2003). Dopamine, acetylcholine and nitric oxide systems interact to induce corticostriatal synaptic plasticity. *Rev. Neurosci., 14*, 207–216.

Durstewitz, D. (2004). Neural representation of interval time. *Neuroreport, 15*, 745–749.

Fiete, I., & Seung, H. (2006). Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Phys. Rev. Letts., 97*, 048104.

Florian, R. (2007). Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation, 19*, 1468–1502.

Foehring, R., & Lorenzon, N. (1999). Neuromodulation, development and synaptic plasticity. *Can. J. Exp. Psychol., 53*, 45–61.

Fusi, S., Asaad, W., Miller, E., & Wang, X.-J. (2007). A neural circuit model of flexible sensori-motor mapping, learning and forgetting on multiple timescales. *Neuron, 54*, 319–333.

Gütig, R., & Sompolinsky, H. (2006). The tempotron, a neuron that learns spike timing–based decision. *Nature Neurosci., 9*, 420–428.

Harris, K. (2008). Stability of the fittest, organizing learning through retroaxonal signals. *Trends in Neurosci., 31*, 130–136.

Izhikevich, E. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex, 17*, 2443–2452.

Major, G., & Tank, D. (2004). Persistent neural activity, prevalence and mechanisms. *Curr. Opin. Neurobiol., 14*, 675–684.

Matsuda, Y., Marzo, A., & Otani, S. (2006). The presence of background dopamine signal converts long-term synaptic depression to potentiation in rat prefrontal cortex. *J. Neurosci., 26*, 4803–4810.

Monasson, R., & Zecchina, R. (1996). Learning and generalization theories of large committee machines. *Mod. Phys. Lett. B, 9*, 1887–1897.

Pfister, J., Toyoizumi, T., Barber, D., & Gerstner, W. (2006). Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Computation, 18*, 1318–1348.

Pouget, A., Dayan, P., & Zemel, R. (2000). Information processing with population codes. *Nature Rev. Neurosci., 1*, 125–132.

Reutimann, J., Yakovlev, V., Fusi, S., & Senn, W. (2004). Climbing neuronal activity as an event-based cortical representation of time. *J. Neurosci., 24*(13), 3295–330.

Seol, G., Ziburkus, J., Huang, S., Song, L., Kim, I., Takamiya, K., et al. (2007). Neuromodulators control the polarity of spike-timing-dependent synaptic plasticity. *Neuron, 55*, 919–929. Erratum in, *Neuron 56*, 754.

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

Urbanczik, R. (1997). Storage capacity of the fully connected committee machine. *J. Phys. A, 30*, L387–L391.

Urbanczik, R., & Senn, W. (2009). Reinforcement learning in populations of spiking neurons. *Nature Neurosci., 12*, 250–252.

Wang, X. (2002). Probabilistic decision making by slow reverberation in cortical circuits. *Neuron, 36*, 955–968.

Werfel, J., Xie, X., & Seung, H. S. (2005). Learning curves for stochastic gradient descent in linear feedforward networks. *Neural Computation, 17*, 2699–2718.

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*, 229–256.

Received May 4, 2009; accepted October 28, 2009.

This article has been cited by: