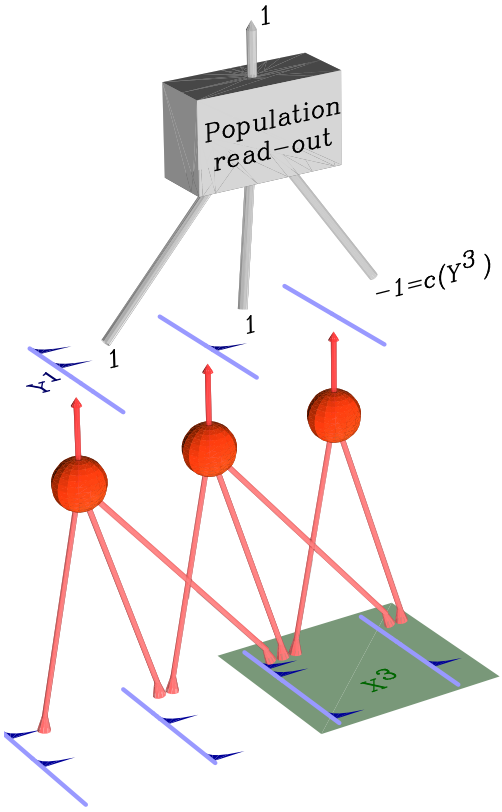
Supplementary Figure
Reinforcement learning in populations of spiking neurons

Robert Urbanczik, Walter Senn

Department of Physiology, University of Bern, Bühlplatz 5, CH-3012 Bern, Switzerland

---



Sketch of a population of spiking neurons and the corresponding population read-out. Each neuron (red) is connected to only a part of the input layer. Neuron 3, for instance, just responds to the spike pattern X^3 underlayed in green. The population read-out, assuming a spike/no-spike code, just takes into account whether or not a neuron has fired in response to the input pattern. So the value of the scoring function is $c(Y^3) = -1$ for the third neuron, but since the first two neurons did fire $c(Y^1) = c(Y^2) = 1$. The decision produced by the population read-out is the sign of the sum of the neuronal scores. This means that the decision is determined by whether or not the majority of the neurons fired and equals 1 in the example.

11 December 2008

Nature Neuroscience: doi:10.1038/nn.2264

Supplementary Methods
Reinforcement learning in populations of spiking neurons

Robert Urbanczik, Walter Senn

Department of Physiology, University of Bern, Bühlplatz 5, CH-3012 Bern, Switzerland

1 Model and Simulation details

1.1 Single neuron model

The input to the model neuron is a spike pattern X consisting of M spike trains X_i $(i = 1, \ldots, M)$ where each X_i is a list of the spike times in afferent i. The resulting postsynaptic spike train Y is also a list of spike times. If the neuron, with synaptic vector w, produces the output Y in response to X, its membrane potential at a time t is:

$$u(t) = U_{rest} + \sum_{i=1}^{M} w_i \sum_{s \in X_i} \epsilon(t-s) - \sum_{s \in Y} \kappa(t-s)\,.$$

Here $U_{rest} = -1$ (arbitrary units) is the resting potential, $\epsilon(t)$ is the postsynaptic and $\kappa(t)$ the reset kernel. For $t \leq 0$ the kernels vanish and for $t > 0$ they are given by

$$\epsilon(t) = \frac{1}{\tau_{\rm m} - \tau_{\rm s}} \left(e^{-t/\tau_{\rm m}} - e^{-t/\tau_{\rm s}} \right) \quad \text{and} \quad \kappa(t) = \frac{1}{\tau_{\rm m}} e^{-t/\tau_{\rm m}}\,,$$

where $\tau_{\rm m} = 10\,\text{ms}$ is used for the membrane time constant and $\tau_{\rm s} = 1.4\,\text{ms}$ for the synaptic time constant.

The emission of postsynaptic spikes is controlled by a stochastic firing intensity $\phi(u)$ which increases with the membrane potential: At each point t in time

Email addresses: urbanczik@pyl.unibe.ch (Robert Urbanczik), senn@pyl.unibe.ch (Walter Senn).

11 December 2008

the firing probability is $\phi(u(t))\,\Delta t$ where Δt represents an infinitesimal time window (we use $\Delta t = 0.2\,\mathrm{ms}$ in the simulations). Our stochastic intensity is

$$\phi(u) = ke^{\beta u},$$

with $k = 0.01$ and $\beta = 5$; in the limit of $\beta \to \infty$ one would recover the deterministic model with a spiking threshold $\theta = 0$. As shown in [1] the log-likelihood of actually producing upto some time t the output spike train Y_t is given by

$$\mathcal{L}_w(Y_t|X) = \sum_{s\in Y_t} \log \phi(u(s)) - \int_0^t \phi(u(s))\mathrm{d}s\,.$$

From this the basic quantity for computing the learning updates is obtained as

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial}{\partial w_i}\mathcal{L}_w(Y_t|X) = \beta\,\mathrm{PSP}_i(t)\left(\sum_{s\in Y_t}\delta(t-s) - \phi(u(t))\right)$$

where $\mathrm{PSP}_i(t) = \sum_{s\in X_i}\epsilon(t-s)$ is the contribution of the postsynaptic potential of synapse i and $\delta(t)$ is Dirac's delta function. Integrating the temporal derivative over the stimulus duration would require the synapses to know when a stimulus starts and ends. To avoid this, we use a low pass filter

$$\tau_{\mathrm{M}}\dot{E}_i^\nu = -E_i^\nu + \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial}{\partial w_i^\nu}\mathcal{L}_{w^\nu}(Y_t^\nu|X^\nu)$$

to obtain the eligibility trace E_i^ν of the i-th synapse of neuron ν.

In all the simulations initial values for the synaptic strengths were picked from a Gaussian distribution with mean and standard deviation equal to 1.7, independently for each afferent and each neuron.

1.2 Pattern statistics

Input patterns are made up of 50 independent Poisson spike trains with a mean firing rate of 6 Hz, independent realizations are used for each pattern. An input layer with 50 nodes presents patterns to the neuronal population. The connectivity from the input layer to the neurons is random, with each neuron receiving a connection from an input node with a probability of 0.8. So the input X^ν effectively seen by the ν-th neuron consists of roughly 40 parallel spike trains, and inputs to different neurons are different but highly correlated. Except for the simulations with stimuli of variable length, the duration of stimuli is 500 ms. The presentation order of the stimuli-response pairs was random.

1.3 Episodic learning

The performance values in Fig. 1 (main text) are averages over 100 ($N = 1$) to 20 ($N = 33$) learning tasks, with a different set of patterns and different initial synaptic strengths. 1 SEM values for the means are smaller than the symbols in the plot.

With just global reinforcement (Eq. 1, main text) the learning rate was $\eta = 1250/N$. Decreasing the learning rate with population size is essential to compensate for the increasingly loose relationship between single neuron performance and reward. We also considered schedules resulting in either a slower or a quicker decrease of the learning rate. But, assuming $\eta = 1250/N^k$, we found that using $k = \frac{1}{2}$ or $k = \frac{3}{2}$ results in a significant deterioration in population performance compared to the choice $k = 1$ used for the main text. As an example, for $N = 5$, the population performance percentages are: 53.7 ± 0.9 ($k = \frac{1}{2}$), 69.5 ± 0.9 ($k = 1$), 64.0 ± 0.7 ($k = \frac{3}{2}$).

For learning with individual reward (Eq. 2, main text) the learning rate was $\eta = 625$, and $\eta = 2500$ was used for attenuated learning (Eq. 3, main text).

1.4 On-line learning

Modeling of the feedback signals. For the reward feedback, we assume a temporary increase in the release rate of a neurotransmitter (e.g. dopamine) in case of success, whereas failure results in a decrease. Assuming that the stimulus X ends at time T, a simple model for the temporal evolution is given by

$$\tau_{\rm rew}\dot{\tilde{R}} = -\tilde{R} + R\,\Theta(t; T + \Delta_{\rm rew}, L_{\rm rew})\,.$$

The value of $\tilde{R}$ correspond to deviation from baseline of the concentration and can thus be negative. The actual concentration would be, say, $R^* = \tilde{R} + 1$ (obeying a differential equation obtained from the one above by simply adding 1 to the right hand side). The function $\Theta(t; T + \Delta_{\rm rew}, L_{\rm rew})$ is equal to 1 for t between $T + \Delta_{\rm rew}$ and $T + \Delta_{\rm rew} + L_{\rm rew}$ and is zero otherwise. So $L_{\rm rew}$ gives the time during which the release rate is changed, and the parameter $\Delta_{\rm rew}$ allows to model delayed onset of the reward. A typical time course of $\tilde{R}$ is shown in Fig. (2a, main text). The above equation describes the time course of $\tilde{R}$ upto the time $T' > T$ when the stimulus X' immediately following X ends. Then the reward variable R is set to the value appropriate for the response to stimulus X'. In units of a millisecond, parameter values in the simulations are $\tau_R = 10$, $L_R = 50$ and $\Delta_{\rm rew} = 0$, except for the blue curve in Fig. (2c, main text), where $\Delta_{\rm rew} = 100$.

Feedback about the population activity P is modeled similarly and, for the time interval from T to T', is given by

$$\tau_{\rm pop}\dot{\tilde{P}} = -\tilde{P} + {\rm sign}(P)\,\alpha e^{-P^2/N}\Theta(t;T,L_{\rm pop})\,. \qquad (*)$$

Again $\tilde{P}$ is the deviation from baseline in the concentration of the neurotransmitter encoding the population response. Note that we assume that the magnitude of the change in release rate is modulated by the population activity via the $\alpha e^{-P^2/N}$ term. In the simulations $\tau_{\rm pop} = L_{\rm pop} = 50\,{\rm ms}$.

Choice of α in the population feedback. For episodic attenuated learning the magnitude of the weight vector change given by Eq. (3, main text) is essentially the same for $R = 1$ and $R = -1$ if $|P|$ is small (compared to $\sqrt{N}$). For binary decision tasks this is reasonable since success and error yield the same amount of information about the desired output, and a small value of $|P|$ means that population performance is unreliable. To achieve the same effect in the on-line procedure, consider the magnitude of $\tilde{\eta}\tilde{a}$ in Eq. (6, main text). For $R = 1$ this depends on the magnitude of the population feedback which is controlled by the parameter α in the above equation (*). For the timing parameters of the feedback signals we use, the value of the integral of $\tilde{\eta}(t)\tilde{a}(t)$ over the update time window is approximately the same for $R = 1$ as for $R = -1$ if $\alpha = 2.5$ (and $|P|$ is small). Hence, we always use $\alpha = 2.5$.

Read-out of the neuronal memory traces. The threshold value in Eq. (5, main text) is $\theta = e^{-1.1}$. The rationale behind this choice is the following. If the time constant $\tau_{\rm M}$ is equal to stimulus duration, then a spike occurring in response to a stimulus yields a memory trace satisfying $s^\nu \geq 1/e$ at stimulus ending. Hence, choosing $\theta = 1/e$ gives ${\rm sign}(s^\nu - \theta) = c(Y^\nu)$ at the time T when the stimulus ends. So, at this point in time, the read-out of the memory trace accurately reflects what the neuron was doing. But, due to the latency of $\tilde{R}$, the strongest synaptic changes occur around $T+50$ (Fig. 2a , main text) even if reward onset is instantaneous and so we use a slightly smaller θ.

Learning rate and performance measure. A learning rate of $\eta = 8$ was used in all of the simulations of the on-line procedure.

The performance percentages shown in the learning curves are computed as a running mean $\bar{p}$ which is updated after each pattern presentation by $\bar{p} \leftarrow (1-\lambda)\bar{p} + \lambda p$. Here $p = 100\%$ if the presented stimulus was classified correctly, otherwise $p = 0$. The timing parameter was set to $\lambda = 0.2/M$, where $M = 30$ is the total number of patterns to be learned. The values shown in Fig. 2c (main text) are averages over 20 tasks.

Nature Neuroscience: doi:10.1038/nn.2264

2 Gradient property

We here give the supplementary information relating the updates (2) and (3) in the main text to gradients of appropriate cost functions. The basic idea is the following: Instead of optimizing expected reward (R), we minimize the expected value of $-g(R\,|P|/\sqrt{N})$. For (2) the function g is increasing and linear. For (3) it is increasing but saturates for large positive values, leading to learning attenuation. Using these cost functions in conjunction with the standard gradient estimator [2], does not yield an update with individualized reward. To achieve this, we analytically average the standard estimator over the two outcomes $c(Y^\nu) = \pm 1$, instead of leaving this to the sampling procedure. Hence compared to the standard estimator our update has reduced variance and this speeds up the learning.

2.1 The general gradient formula

Let $P_{\mathbf{w}}(\mathbf{Y}|\mathbf{X})$ be the probability that a population with weight vectors $\mathbf{w} = (w^1, \ldots, w^N)$ produces, upto time T, the output spike trains $\mathbf{Y} = (Y^1, \ldots, Y^N)$ in response to the stimulus $\mathbf{X}$. By $\mathbf{X}$, we refer to the total spike pattern applied to the input layer. Conditioned on the stimulus, the neurons are independent, so for the joint distribution $P_{\mathbf{w}}(\mathbf{X}, \mathbf{Y})$ of stimuli and responses we have

$$P_{\mathbf{w}}(\mathbf{X}, \mathbf{Y}) = P(\mathbf{X}) P_{\mathbf{w}}(\mathbf{Y}|\mathbf{X}) = P(\mathbf{X}) \prod_{\nu=1}^{N} P_{w^\nu}(Y^\nu|X^\nu).$$

To avoid confusion with the probabilities we no longer denote population activity by P but use the normalized form $S(\mathbf{Y}) = N^{-\frac{1}{2}} \sum_\nu c(Y^\nu)$ to measure the activity. Further, let $z_{\mathbf{X}} = \pm 1$ be the target label for the pattern. So $R = z_{\mathbf{X}} \operatorname{sign} S(\mathbf{Y})$. For a monotonically increasing function g, the value of $g(z_{\mathbf{X}} S(\mathbf{Y})) = g(R\,|S(\mathbf{Y})|)$ provides a measure for the correctness and reliability of the population signal. We now consider cost functions of the form

$$c_w = -\int \mathrm{d}\mathbf{X} \mathrm{d}\mathbf{Y}\, P_{\mathbf{w}}(\mathbf{X}, \mathbf{Y})\, g(z_{\mathbf{X}} S(\mathbf{Y})) \tag{1}$$

The following then holds for the gradient $\nabla_\nu c_w$ of the cost with respect to the weight vector w^ν of neuron ν:

$$\begin{aligned} -\nabla_\nu c_w = &\int \mathrm{d}\mathbf{X} \mathrm{d}\mathbf{Y}\, P_{\mathbf{w}}(\mathbf{X}, \mathbf{Y}) \times \\ &\tfrac{1}{2} A\left(z_{\mathbf{X}} S(\mathbf{Y}) - z_{\mathbf{X}} N^{-\frac{1}{2}} c(Y^\nu)\right) (z_{\mathbf{X}} c(Y^\nu) - 1) \mathcal{E}^\nu \end{aligned} \tag{2}$$

where

$$A(\chi) = g(\chi + N^{-\frac{1}{2}}) - g(\chi - N^{-\frac{1}{2}}) \tag{3}$$

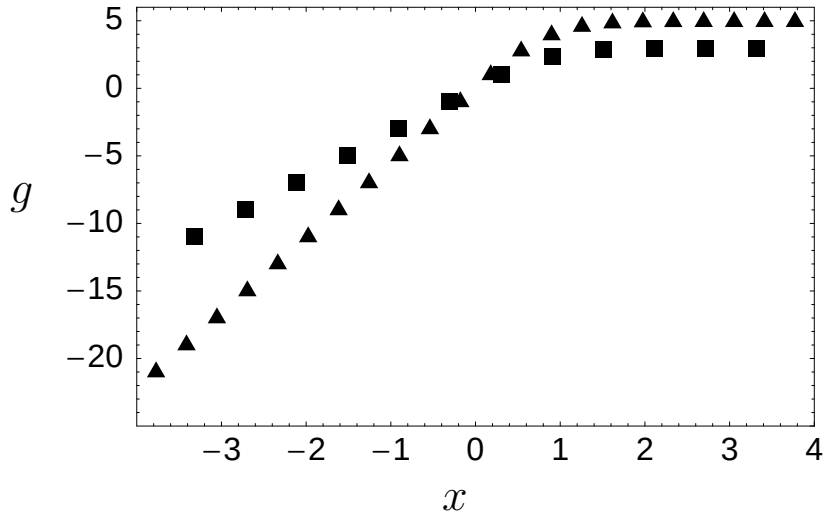


Fig. 1. The function g given by Eq. (5) for $N = 11$ (boxes) and for $N = 31$ (triangles).

and

$$\mathcal{E}^\nu = \nabla_\nu \ln P_{w^\nu}(Y^\nu|X^\nu) = \nabla_\nu \mathcal{L}_{w^\nu}(Y^\nu|X^\nu)\,. \tag{4}$$

2.2 Examples for specific cost functions

Before demonstrating the equality, let us show how it relates to the updates considered in the main text. First consider the choice $g(x) = N^{\frac{1}{2}}x$, so reward and punishment increase linearly with the magnitude of the population majority. For this choice, we have $A(x) = 2$ and the expression (2) for the cost-gradient becomes

$$-\nabla_\nu c_w = \int \mathrm{d}\mathbf{X}\mathrm{d}\mathbf{Y} P_{\mathbf{w}}(\mathbf{X},\mathbf{Y})\,(z_{\mathbf{X}}c(Y^\nu) - 1)\mathcal{E}^\nu\,.$$

Here $z_{\mathbf{X}}c(Y^\nu)$ is just the single neuron reward r^ν used for the update (2) in the main text. So this update implements a stochastic gradient descent in the cost (1) with $g(x) = N^{\frac{1}{2}}x$, except for the fact that we are using a running time window approximation to the probability gradient $\mathcal{E}^\nu$.

For the attenuated learning update (Eq. 3, main text) we shall for brevity assume that N is odd and consider the following choice for g:

$$g(x) = \begin{cases} N^{\frac{1}{2}}x & \text{if } x \le 0 \\ -1 + \sum_{l=0}^{(\sqrt{N}x-1)/2} 2e^{-(2l)^2/N} & \text{otherwise}\,. \end{cases} \tag{5}$$

This assigns the same cost as before to an error, but in case of success g

saturates with increasing population majority (see Fig. 1). Since $N^{\frac{1}{2}}S(\mathbf{Y})$ is an odd integer we have

$$\tfrac{1}{2}A\left(z_{\mathbf{X}}S(\mathbf{Y})-z_{\mathbf{X}}N^{-\frac{1}{2}}c(Y^{\nu})\right)=\begin{cases}1 & \text{if } z_{\mathbf{X}}S(\mathbf{Y})<0\\ e^{-\left(S(\mathbf{Y})-N^{-\frac{1}{2}}c(Y^{\nu})\right)^{2}} & \text{otherwise}\,.\end{cases}$$

Upto the $N^{-\frac{1}{2}}c(Y^{\nu})$ term, which becomes negligible for a large population size N, this is just the attenuation factor a used in Eq. (3) of the main text. This also motivates the somewhat complicated algebraic form of Eq. (5). Since the attenuation factor is a finite difference of g values, the sum of exponentials in (5) yields an attenuation factor which is a simple exponential. If, for $x>0$, g were based on a more common saturating function, similar learning performance could be achieved. But the attenuation factor would have a more complicated algebraic form.

2.3 Proof of the general gradient formula

To demonstrate Eq. (2), first note that it immediately follows from the two statements

$$\begin{aligned}0&=\int\mathrm{d}\mathbf{X}\mathrm{d}\mathbf{Y}\,P_{\mathbf{w}}(\mathbf{X},\mathbf{Y})\tfrac{1}{2}A\left(z_{\mathbf{X}}S(\mathbf{Y})-z_{\mathbf{X}}N^{-\frac{1}{2}}c(Y^{\nu})\right)\mathcal{E}^{\nu}\\ -\nabla_{\nu}\,c_{w}&=\int\mathrm{d}\mathbf{X}\mathrm{d}\mathbf{Y}\,P_{\mathbf{w}}(\mathbf{X},\mathbf{Y})\tfrac{1}{2}A\left(z_{\mathbf{X}}S(\mathbf{Y})-z_{\mathbf{X}}N^{-\frac{1}{2}}c(Y^{\nu})\right)z_{\mathbf{X}}c(Y^{\nu})\mathcal{E}^{\nu}\,.\end{aligned}\quad(6)$$

For the first identity note that the argument of A does not depend on Y^{ν}. So the identity is equivalent to the standard observation showing that one may introduce a reinforcement baseline: Averaged over all outcomes, the gradient of the log-likelihood vanishes.

To demonstrate the second identity in (6), we first rewrite the cost (1) as:

$$c_{w}=-\int\mathrm{d}\mathbf{X}\mathrm{d}\mathbf{Y}\,\left[P(\mathbf{X})\tfrac{P_{\mathbf{w}}(\mathbf{Y}|\mathbf{X})}{P_{w^{\nu}}(Y^{\nu}|X^{\nu})}\right]g(z_{\mathbf{X}}S(\mathbf{Y}))P_{w^{\nu}}(Y^{\nu}|X^{\nu}).$$

The term in square brackets does not depend on Y^{ν} so

$$-\nabla_{\nu}\,c_{w}=\int\mathrm{d}\mathbf{X}\mathrm{d}\mathbf{Y}^{/\nu}\,P(\mathbf{X})\tfrac{P_{\mathbf{w}}(\mathbf{Y}|\mathbf{X})}{P_{w^{\nu}}(Y^{\nu}|X^{\nu})}D(\mathbf{Y},X^{\nu},w^{\nu})\quad(7)$$

where $\mathrm{d}\mathbf{Y}^{/\nu}$ denotes the integration over all output spike trains but the ν-th.

Nature Neuroscience: doi:10.1038/nn.2264

Further

$$D(\mathbf{Y}, X^\nu, w^\nu) = \nabla_\nu \int \mathrm{d}Y^\nu g(z_{\mathbf{X}} S(\mathbf{Y})) P_{w^\nu}(Y^\nu|X^\nu)\,.$$

We now rewrite

$$\begin{aligned} D(\mathbf{Y}, X^\nu, w^\nu) &= \sum_{k=\pm 1} \int \mathrm{d}Y^\nu \delta_{k,c(Y^\nu)} g(z_{\mathbf{X}} S(\mathbf{Y})) \nabla_\nu P_{w^\nu}(Y^\nu|X^\nu) \\ &= \sum_{k=\pm 1} \int \mathrm{d}Y^\nu \delta_{k,c(Y^\nu)} g\left(z_{\mathbf{X}} S(\mathbf{Y}) + z_{\mathbf{X}} N^{-\frac{1}{2}}(k - c(Y^\nu))\right) \nabla_\nu P_{w^\nu}(Y^\nu|X^\nu) \\ &= \sum_{k=\pm 1} g\left(z_{\mathbf{X}} S(\mathbf{Y}) + z_{\mathbf{X}} N^{-\frac{1}{2}}(k - c(Y^\nu))\right) \nabla_\nu \int \mathrm{d}Y^\nu \delta_{k,c(Y^\nu)} P_{w^\nu}(Y^\nu|X^\nu) \end{aligned}$$

where the last equality holds since the argument of g does not depend on Y^ν. The integral in the last line is simply the probability of the outcome $c(Y^\nu) = k$, hence

$$\nabla_\nu \int \mathrm{d}Y^\nu \delta_{1,c(Y^\nu)} P_{w^\nu}(Y^\nu|X^\nu) = -\nabla_\nu \int \mathrm{d}Y^\nu \delta_{-1,c(Y^\nu)} P_{w^\nu}(Y^\nu|X^\nu) \qquad (8)$$

Using this we write $D(\mathbf{Y}, X^\nu, w^\nu)$ as

$$\begin{aligned} D(\mathbf{Y}, X^\nu, w^\nu) &= \sum_{k=\pm 1} g\left(z_{\mathbf{X}} S(\mathbf{Y}) + z_{\mathbf{X}} N^{-\frac{1}{2}}(k - c(Y^\nu))\right) k \times \\ &\quad \nabla_\nu \int \mathrm{d}Y^\nu \delta_{1,c(Y^\nu)} P_{w^\nu}(Y^\nu|X^\nu) \\ &= A\left(z_{\mathbf{X}} S(\mathbf{Y}) - z_{\mathbf{X}} N^{-\frac{1}{2}} c(Y^\nu)\right) z_{\mathbf{X}} \nabla_\nu \int \mathrm{d}Y^\nu \delta_{1,c(Y^\nu)} P_{w^\nu}(Y^\nu|X^\nu) \end{aligned}$$

with A given in (3). We have now averaged over the two outcomes $c(Y^\nu) = \pm 1$ but the result is in an asymmetric form since it contains just the probability gradient for the case that $c(Y^\nu) = 1$. To be used in a sampling procedure we need a symmetric form of the identity and for this we first note that (using Eq. 8 once again) we also have

$$D(\mathbf{Y}, X^\nu, w^\nu) = A\left(z_{\mathbf{X}} S(\mathbf{Y}) - z_{\mathbf{X}} N^{-\frac{1}{2}} c(Y^\nu)\right) z_{\mathbf{X}} k \nabla_\nu \int \mathrm{d}Y^\nu \delta_{k,c(Y^\nu)} P_{w^\nu}(Y^\nu|X^\nu)$$

for $k = \pm 1$. Further

$$\begin{aligned} D(\mathbf{Y}, X^\nu, w^\nu) &= \frac{1}{2} \sum_{k=\pm 1} D(\mathbf{Y}, X^\nu, w^\nu) \\ &= \frac{1}{2} \int \mathrm{d}Y^\nu A\left(z_{\mathbf{X}} S(\mathbf{Y}) - z_{\mathbf{X}} N^{-\frac{1}{2}} c(Y^\nu)\right) z_{\mathbf{X}} \nabla_\nu P_{w^\nu}(Y^\nu|X^\nu) \sum_{k=\pm 1} k \delta_{k,c(Y^\nu)} \\ &= \int \mathrm{d}Y^\nu \frac{1}{2} A\left(z_{\mathbf{X}} S(\mathbf{Y}) - z_{\mathbf{X}} N^{-\frac{1}{2}} c(Y^\nu)\right) z_{\mathbf{X}} \nabla_\nu P_{w^\nu}(Y^\nu|X^\nu) c(Y^\nu) \end{aligned}$$

Plugging the last line into Eq. (7) we obtain for the gradient of the cost:

Nature Neuroscience: doi:10.1038/nn.2264

$$-\nabla_\nu c_w = \int \mathrm{d}\mathbf{X}\mathrm{d}\mathbf{Y}\, P(\mathbf{X}) P_{\mathbf{w}}(\mathbf{Y}|\mathbf{X}) \times \frac{1}{2} A \left(z_{\mathbf{X}} S(\mathbf{Y}) - z_{\mathbf{X}} N^{-\frac{1}{2}} c(Y^\nu) \right) z_{\mathbf{X}} c(Y^\nu) \frac{1}{P_{w^\nu}(Y^\nu|X^\nu)} \nabla_\nu P_{w^\nu}(Y^\nu|X^\nu) \,,$$

which is equivalent to (6).

Nature Neuroscience: doi:10.1038/nn.2264

3 Standard reinforcement with graded reward

Our rule differs from standard reinforcement in that we (a) use a graded objective function, see Fig. 1, and (b) an improved estimator for the gradient which is based on a neuron-specific feedback signal. Here we report simulation results for a graded objective function in combination with the standard reinforcement update. These indicate that (b) is in fact necessary for learning to speed up with population size.

We first consider a simple sigmoidal grading of the reward signal with the poulation activity in the update rule

$$\Delta w_i^\nu = \eta \, \tanh(\, R \, S(\mathbf{Y}) \,) \, E_i^\nu(T) \,. \tag{9}$$

Here $S(\mathbf{Y}) = N^{-\frac{1}{2}} \sum_\nu c(Y^\nu)$ is the normalized population activity introduced in the previous section.

When learning the same task as in the main text with $N = 9$ neurons, the best performance achieved was 58.0 ± 0.8 percent, see Fig. 2. This is in fact a modest improvement on learning with a binary reward signal (Eq. 1, main text) where the corresponding performance percentage is 54.5 ± 0.8. But it is worse than the result for $N = 1$ where performance reached 74.0 ± 0.9 percent.

Rule (9) makes large updates when $|S(\mathbf{Y})|$ is large, i.e. when the response is reliable, even when the response is correct ($R = 1$). But no further learning seems necessary in this case. As a simple remedy, we also tried

$$\Delta w_i^\nu = \eta (R - 1) \, h \, (S(Y)) \, E_i^\nu(T) \,, \tag{10}$$

implementing graded error correction learning, with the function h tuning the grading. Arguably, the simplest choice for h is in fact $h(x) = x$ and not some-

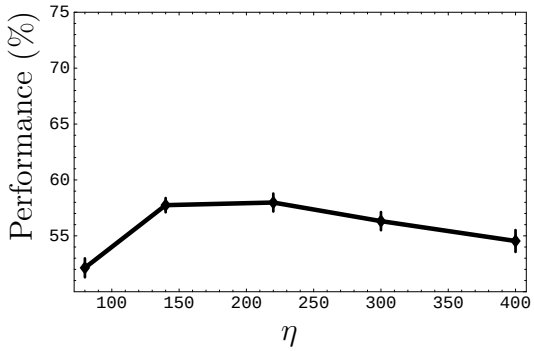


Fig. 2. Population performance for rule (1) as function of the learning rate for a population size N=9. (Mean values averaged over 30 tasks are shown.) All other simulation parameters are as for Figs. 1b and 1c in the manuscript. The error bars show the 1 SEM values for the mean.

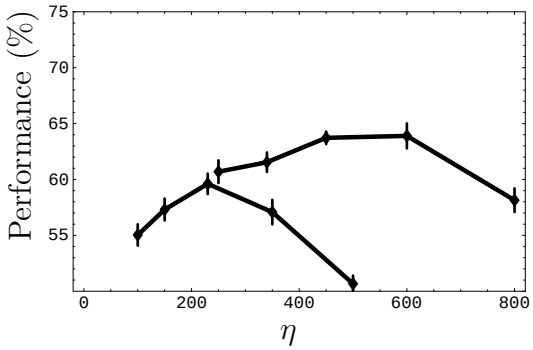


Fig. 3. Performance of rule (10) with $h(x) = x$ as function of the learning rate forn population sizes $N = 9$ (top curve) and $N = 33$ (bottom curve). All other parameters as in Fig. 2 above.

thing like $h(x) = \tanh(x)$, because it is not obvious why punishment should saturate when many of the neurons make a mistake.

For the $N = 9$ population with a good choice of the learning rate a performance percentage of 63.9 ± 1.1 is achieved (Fig. 3). This is a more noticeable improvement on the binary reward result. But it is still worse than the result for just a single neuron, indicating that learning does slow down with increasing population for graded error correction, albeit less dramatically than for binary reward.

To confirm this, we simulated (10) for a population of 33 neurons (Fig. 3). Even for a good choice of the learning rate, performance now decreases to 59.6 ± 0.9 percent. We also ran the $N = 33$ simulations with $h(x) = \tanh(x)$. This gave results similar (slightly worse, but within error bars) to the $h(x) = x$ choice. To us, comparing the $N = 9$ and $N = 33$ findings suggests that performance will eventually decay to chance level when further increasing N. So using a graded instead of a binary signal in the standard reinforcement update seems to just slow down the deterioration. From a biological perspective, it is worthwhile mentioning that rules (9,10) assume two feedback signals. Since we are considering binary decision tasks, the external reward feedback must be binary. So any grading has to result from remodulating the external binary signal with a second feedback about the population activity. In this, graded reward is not simpler than our attenuated learning rule. Given the performance difference, we do not think that using graded reward in the standard reinforcement update is an attractive option.

References

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

[2] R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Nature Neuroscience: doi:10.1038/nn.2264