nature portfolio

https://doi.org/10.1038/s41593-021-00809-5

Supplementary information

Synaptic plasticity as Bayesian inference

In the format provided by the authors and unedited

Supplementary Math Note

Here we provide a complete derivation of the Bayesian and classical learning rules for linear, cerebellar and reinforcement learning, and derive the simplified learning rules used in Eq. (4) of the main text (Sec. S1), derive the relationship between variability and firing rate (Sec. S2), provide details of the simulations (Sec. S3, which includes tables containing the parameters used in the simulations), discuss how we chose the parameters of our model (Sec. S4), provide a normative explanation for the Synaptic Sampling hypothesis (Sec. S5), and explore the robustness of our model (Sec. S6).

S1 Derivation of the learning rules

The learning rules derived in Methods, Eq. (M.29), all depend on the likelihood of the data given the target weights. Here we compute the likelihood for our three single neuron feedback signals, and use those to write down the learning rules. The learning rules for the recurrent neural network were derived in Methods, Sec. M2.3.

S1.1 Single neuron learning rules in terms of the log likelihood

Our starting point is the likelihood given in Methods, Eq. (M.31); the corresponding log likelihood is

$$L(\lambda_{\mathrm{tar},i}) = \log \int df_{\mathrm{lin}}\, P(f|f_{\mathrm{lin}})\, P(f_{\mathrm{lin}}|x_i, w_i, \lambda_{\mathrm{tar},i}) \tag{S.1}$$

where f is either f_{lin}, f_{cb} or f_{rl}, all of which are deterministic functions of f_{lin} (see main text).

The second term inside the integral, $P(f_{\mathrm{lin}}|x_i, w_i, \lambda_{\mathrm{tar},i})$, is given in Methods, Eq. (M.37). Although that distribution is Gaussian in f_{lin}, its mean depends on $e^{\lambda_{\mathrm{tar},i}}$, so as a function of $\lambda_{\mathrm{tar},i}$ it is somewhat unwieldy. We thus use statistical linearisation [1] to simplify it: we approximate $\pm e^{\lambda_{\mathrm{tar},i}}$ with a straight line, $a(\lambda_{\mathrm{tar},i} - m_i) + b$, with a and b chosen to minimize the expected mean squared error between $\pm e^{\lambda_{\mathrm{tar},i}}$ and $a(\lambda_{\mathrm{tar},i} - m_i) + b$, where the expectation is with respect to the posterior on the previous time step, $P(\lambda_{\mathrm{tar},i}|\mathcal{D}(t-1))$. The solution is especially simple, $a = b = \mu_i$, which gives

$$\pm e^{\lambda_{\mathrm{tar},i}} \approx \mu_i \left(1 + \lambda_{\mathrm{tar},i} - m_i\right) \tag{S.2}$$

(recall that μ_i is the expected value of $w_{\mathrm{tar},i}$; see Methods, Eq. (M.13a)). Inserting this into Methods, Eq. (M.37), we have

$$\begin{aligned} L_{\mathrm{lin}}(\lambda_{\mathrm{tar},i}) &\equiv \log P(f_{\mathrm{lin}}|x_i, w_i, \lambda_{\mathrm{tar},i}) \\ &= -\frac{\left(f_{\mathrm{lin}} - (\mu_i - w_i)\, x_i - \mu_i x_i \left(\lambda_{\mathrm{tar},i} - m_i\right)\right)^2}{2\sigma_\delta^2} + \mathrm{const} \end{aligned} \tag{S.3}$$

where σ_δ^2 is given in Methods, Eq. (M.36). Note that equality in this expression (and many that follow) is shorthand for equality under the assumption that $P(f_{\rm lin}|x_i, w_i, \lambda_{{\rm tar},i})$ is given by Eq. (S.3). Inserting Eq. (S.3) into the expression for the log likelihood, Eq. (S.1), we arrive at

$$L(\lambda_{{\rm tar},i}) = \log \int df_{\rm lin}\, P(f|f_{\rm lin})\, e^{L_{\rm lin}(\lambda_{{\rm tar},i})} + \text{const.} \tag{S.4}$$

To write down an explicit expression for σ_δ^2 – which we will need below – we use the fact that σ_i^2 is the variance of $e^{\lambda_{{\rm tar},i}}$. Then, because the synapses are independent, σ_δ^2 can be written as combination of the uncertainty about the target weight, $w_{{\rm tar},i}$, and the variability in the actual weight due to noise,

$$\sigma_\delta^2 = \sum_j \left(\sigma_j^2 + \text{Var}\left[w_j\right]\right) x_j^2 + \sigma_0^2 \tag{S.5}$$

where, recall, σ_0^2 is the combined variance of the noise in the membrane potential and the feedback signal (Methods, Eq. (M.33)). What we use for the variance of w_j depends on whether we are sampling (for which $\text{Var}\left[w_i\right] = \sigma_i^2$), or using a variance proportional to the mean, (for which $\text{Var}\left[w_i\right] = k\mu_i$); see Methods, Eq. (M.39).

Derivatives of the log likelihood, $L(\lambda_{{\rm tar},i})$ – the main ingredients in the learning rule, Methods, Eq. (M.29) – can be expressed as derivatives of $L_{\rm lin}$; as is straightforward to show,

$$L'(\lambda_{{\rm tar},i}) = \mathbb{E}_{\rm lin}\left[L'_{\rm lin}(\lambda_{{\rm tar},i})\right] \tag{S.6a}$$

$$L''(\lambda_{{\rm tar},i}) = \mathbb{E}_{\rm lin}\left[L''_{\rm lin}(\lambda_{{\rm tar},i})\right] + \text{Var}_{\rm lin}\left[L'_{\rm lin}(\lambda_{{\rm tar},i})\right] \tag{S.6b}$$

where the expectation and variance are with respect to the distribution

$$P(f_{\rm lin}|f, x_i, w_i, \lambda_{{\rm tar},i}) = \frac{P(f|f_{\rm lin})\, e^{L_{\rm lin}(\lambda_{{\rm tar},i})}}{\int df'_{\rm lin} P(f|f'_{\rm lin})\, e^{L_{\rm lin}(\lambda_{{\rm tar},i})}}\,. \tag{S.7}$$

(Recall that $L_{\rm lin}(\lambda_{{\rm tar},i})$ depends on $f_{\rm lin}$; see Eq. (S.3).) Using Eq. (S.3) for $L_{\rm lin}(\lambda_{{\rm tar},i})$ the derivatives are straightforward; combining those with Methods, Eq. (M.29) (for which we need to evaluate the derivatives at $\lambda_{{\rm tar},i} = m_i$), we have, after a small amount of algebra,

$$\Delta m_i = \left(\frac{s_i^2 \mu_i}{\sigma_\delta^2}\right) x_i \big(\mathbb{E}_{\rm lin}\left[f_{\rm lin}\right] + x_i\left(w_i - \mu_i\right)\big) - \frac{m_i - m_{\rm prior}}{\tau} \tag{S.8a}$$

$$\Delta s_i^2 = -\left(\frac{s_i^4 \mu_i^2}{\sigma_\delta^2}\right) x_i^2 \left(\frac{\sigma_\delta^2 - \text{Var}_{\rm lin}\left[f_{\rm lin}\right]}{\sigma_\delta^2}\right) - \frac{2\left(s_i^2 - s_{\rm prior}^2\right)}{\tau}\,. \tag{S.8b}$$

In the next section, we use these expressions to derive the explicit learning rules for our three single neuron feedback signals, $f_{\rm lin}$, $f_{\rm cb}$ and $f_{\rm rl}$. Note that because $\lambda_{{\rm tar},i}$ is evaluated at $\lambda_{{\rm tar},i} = m_i$, the expression for $L_{\rm lin}(\lambda_{{\rm tar},i})$, Eq. (S.3), simplifies considerably.

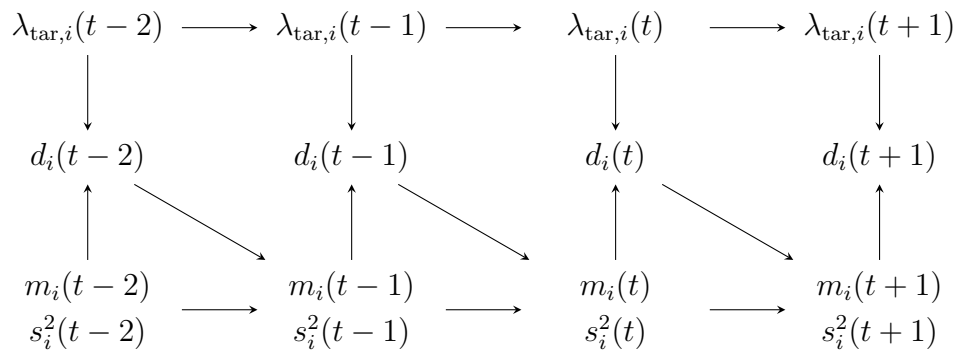

Figure S.1: The graphical model describing the dependencies in our simulations. The log of the target weight, $\lambda_{\text{tar},i}(t)$, evolves independently of all other variables, under the Ornstein-Uhlenbeck (OU) process described in Methods, Eq. (M.11). The data, $d_i(t)$, which consists of the presynaptic input, $x_i(t)$, the actual weight, $w_i(t)$, and a signal that provides information about the target weights (Methods, Eq. (M.18)), depends on both the target weight, $\lambda_{\text{tar},i}(t)$, and on past inference, $m_i(t)$ and $s_i^2(t)$. The mean and uncertainty at time t, $m_i(t)$ and $s_i^2(t)$, depend on the mean and uncertainty at the previous time step, $m_i(t-1)$ and $s_i^2(t-1)$, and also on past data, $d_i(t-1)$, through the learning rules (Methods, Eq. (M.29)).

In Fig. S.1 we show a dependency graph which describes how each variable is generated. This is a graphical model — a compact method for describing dependencies among random variables. This graphical model has the unusual feature that the results of inference at one time step influence the data at subsequent time steps.

S1.2 Single neuron learning rules for our three feedback signals

To derive explicit learning rules for our three feedback signals, we just need to compute the mean and variance of f_{lin}, and then insert those into Eq. (S.8). Those calculations, which are mainly straightforward, follow.

Linear feedback, $f = f_{\text{lin}}$

For linear feedback, $\mathbb{E}_{\text{lin}}[f_{\text{lin}}] = f$ and $\text{Var}_{\text{lin}}[f_{\text{lin}}] = 0$. Thus, the update rules for the mean and variance of the log weights, Eq. (S.8), become

$$\Delta m_i = \left(\frac{s_i^2 \mu_i}{\sigma_\delta^2}\right) x_i \left(f_{\text{lin}} + x_i\big(w_i - \mu_i\big)\right) - \frac{1}{\tau}\left(m_i - m_{\text{prior}}\right), \tag{S.9a}$$

$$\Delta s_i^2 = -\left(\frac{s_i^4 \mu_i^2}{\sigma_\delta^2}\right) x_i^2 - \frac{2}{\tau}\left(s_i^2 - s_{\text{prior}}^2\right). \tag{S.9b}$$

While these rules are easily updated on a digital computer, they are somewhat problematic for real synapses, since they are non-local: σ_δ^2 depends on all the synapses (see

Eq. (S.5)). To remedy this, we replace σ_δ^2 with a constant, $\sigma_{\delta 0}^2$, the the average value of σ_δ^2 under the prior (see Methods, Eq. (M.38)). In addition, as in Methods (see discussion following Eq. (M.56)), we drop the term $x_i(w_i - \mu_i)$ in Eq. (S.9a), as it is small compared to f_{lin}. The resulting learning rules, along with their classical counterparts, are given in Methods, Eqs. (M.41) and (M.42).

Cerebellar feedback, $f = f_{\text{cb}} = \text{sign}(f_{\text{lin}} - \theta)$

For cerebellar feedback, $P(f_{\text{cb}}|f_{\text{lin}})$ may be written

$$P(f_{\text{cb}}|f_{\text{lin}}) = f_{\text{cb}}\Theta(f_{\text{lin}} - \theta) + (1 - f_{\text{cb}})\Theta(\theta - f_{\text{lin}}) \tag{S.10}$$

where f_{cb} can take on only the values 0 and 1 and, as usual, Θ is the Heaviside step function. This expression tells us that if $f_{\text{cb}} = 1$ then $f_{\text{lin}} \geq \theta$, and if $f_{\text{cb}} = 0$ then $f_{\text{lin}} < \theta$. Consequently, $P(f_{\text{lin}}|f_{\text{cb}}, x_i, w_i, m_i)$, Eq. (S.7), is a truncated Gaussian whose mean and variance can be computed in terms of the cumulative normal function.

To derive the learning rules, it is easiest to compute $L(\lambda_{\text{tar},i})$ directly from Eq. (S.4) and then take derivatives with respect to $\lambda_{\text{tar},i}$, rather than using Eq. (S.8). To this end, we insert Eq. (S.10) into Eq. (S.4); then, using Eq. (S.3) for $L_{\text{lin}}(\lambda_{\text{tar},i})$, we have

$$\begin{aligned} L(\lambda_{\text{tar},i}) = \log \Bigg[\int df_{\text{lin}} \left(f_{\text{cb}}\Theta(f_{\text{lin}} - \theta) + (1 - f_{\text{cb}})\Theta(\theta - f_{\text{lin}}) \right) \\ \times \exp\left(-(f_{\text{lin}} - (\mu_i - w_i)x_i - \mu_i x_i(\lambda_{\text{tar},i} - m_i))^2 / 2\sigma_\delta^2 \right) \Bigg] + \text{const.} \end{aligned} \tag{S.11}$$

The integral is straightforward, and we arrive at

$$L(\lambda_{\text{tar},i}) = \log \left[\Phi \left((2f_{\text{cb}} - 1) \left[\frac{(\mu_i - w_i)x_i + \mu_i x_i(\lambda_{\text{tar},i} - m_i)}{\sigma_\delta^2} - \theta \right] \right) \right] + \text{const} \tag{S.12}$$

where Φ is the cumulative normal function, defined in Methods, Eq. (M.44a). Taking the first and second derivatives with respect to $\lambda_{\text{tar},i}$ and evaluating them at $\lambda_{\text{tar},i} = m_i$ gives us

$$L'(m_i) = \frac{\mu_i x_i}{\sigma_\delta} (2f_{\text{cb}} - 1) \frac{\mathcal{N}(\tilde{\theta}_{\text{cb}})}{\Phi(\tilde{\theta}_{\text{cb}})} \tag{S.13a}$$

$$L''(m_i) = -\frac{\mu_i^2 x_i^2}{\sigma_\delta^2} \frac{\mathcal{N}(\tilde{\theta}_{\text{cb}})}{\Phi(\tilde{\theta}_{\text{cb}})} \left[\tilde{\theta}_{\text{cb}} + \frac{\mathcal{N}(\tilde{\theta}_{\text{cb}})}{\Phi(\tilde{\theta}_{\text{cb}})} \right] \tag{S.13b}$$

where $\tilde{\theta}_{\text{cb}}$ is given by

$$\tilde{\theta}_{\text{cb}} \equiv (2f_{\text{cb}} - 1) \frac{(\mu_i - w_i)x_i - \theta}{\sigma_\delta} \tag{S.14}$$

and $\mathcal{N}$ is the standard normal function (Methods, Eq. (M.44b)). Inserting these expressions into Methods, Eq (M.29), leads to

$$\Delta m_i = \left(\frac{s_i^2\mu_i}{\sigma_\delta^2}\right) x_i\sigma_\delta(2f_{\rm cb}-1)\,\frac{\mathcal{N}(\tilde\theta_{\rm cb})}{\Phi\big(\tilde\theta_{\rm cb}\big)} \;-\frac{1}{\tau}(m_i-m_{\rm prior}) \tag{S.15a}$$

$$\Delta s_i^2 = -\left(\frac{s_i^4\mu_i^2}{\sigma_\delta^2}\right) x_i^2\,\frac{\mathcal{N}(\tilde\theta_{\rm cb})}{\Phi\big(\tilde\theta_{\rm cb}\big)}\left[\tilde\theta_{\rm cb}+\frac{\mathcal{N}\big(\tilde\theta_{\rm cb}\big)}{\Phi\big(\tilde\theta_{\rm cb}\big)}\right]-\frac{2}{\tau}\left(s_i^2-s_{\rm prior}^2\right). \tag{S.15b}$$

As with linear feedback, to ensure that the learning rule is local, we replace σ_δ^2 with $\sigma_{\delta0}^2$. In addition, we drop the contribution of $(\mu_i-w_i)x_i$ to $\tilde\theta_{\rm cb}$ (Eq. (S.14)). That follows from the same reasoning we used for the linear feedback: θ is on the order of σ_δ, which is much larger than $(\mu_i-w_i)x_i$. We thus define $\theta_{\rm cb}$ to be $\tilde\theta_{\rm cb}$ but without the term $(\mu_i-w_i)x_i$ and with σ_δ replaced with $\sigma_{\delta0}$, The resulting learning rules, along with their classical counterparts, are given in Methods, Eqs. (M.43) and (M.46).

Reinforcement learning, $f = f_{\rm rl} = -|f_{\rm lin}|$

For reinforcement learning,

$$P(f_{\rm rl}|f_{\rm lin}) = \delta\left(f_{\rm rl}+|f_{\rm lin}|\right). \tag{S.16}$$

Combining this with Eq. (S.3) for $L(\lambda_{{\rm tar},i})$ and using Eq. (S.7), we have

$$P(f_{\rm lin}|f_{\rm rl},x_i,w_i,m_i) \propto \delta(f_{\rm rl}+|f_{\rm lin}|)\,\exp\left(f_{\rm lin}(\mu_i-w_i)x_i/\sigma_\delta^2\right) \tag{S.17}$$

where we evaluated $\lambda_{{\rm tar},i}$ at $\lambda_{{\rm tar},i}=m_i$ and used the fact that, because of the delta-function, $f_{\rm lin}^2=f_{\rm rl}^2$. Consequently, the expectations needed for the update rule, Eq. (S.8), are

$$\mathbb{E}\left[f_{\rm lin}|f_{\rm rl},x_i,w_i,m_i\right] = f_{\rm rl}\tanh\left((\mu_i-w_i)x_if_{\rm rl}/\sigma_\delta^2\right) \tag{S.18a}$$

$$\mathrm{Var}\left[f_{\rm lin}|f_{\rm rl},x_i,w_i,m_i\right] = \frac{f_{\rm rl}^2}{\cosh^2\left((\mu_i-w_i)x_if_{\rm rl}/\sigma_\delta^2\right)}. \tag{S.18b}$$

Inserting these into Eq. (S.8) yields

$$\Delta m_i = \left(\frac{s_i^2\mu_i}{\sigma_\delta^2}\right) x_i\big(f_{\rm rl}\tanh\left((\mu_i-w_i)x_if_{\rm rl}/\sigma_\delta^2\right)-(\mu_i-w_i)x_i\big)-\frac{1}{\tau}(m_i-m_{\rm prior}) \tag{S.19a}$$

$$\Delta s_i^2 = -\left(\frac{s_i^4\mu_i^2}{\sigma_\delta^2}\right) x_i^2\left(1-\frac{f_{\rm rl}^2/\sigma_\delta^2}{\cosh^2\left((\mu_i-w_i)x_if_{\rm rl}/\sigma_\delta^2\right)}\right)-\frac{2}{\tau}\left(s_i^2-s_{\rm prior}^2\right). \tag{S.19b}$$

Not surprisingly, for reinforcement learning the synapse must explore: if it sets the weight, w_i, to μ_i (the mean value of $w_{{\rm tar},i}$), m_i will relax to the prior. Sampling in this case is critical.

Because $f_{\rm rl}$ and σ_δ are both large, we can make the approximations $\tanh z \approx z$ and $\cosh^2 z \approx 1$, yielding simplified learning rules. Those learning rules, along with their classical counterparts (for which do not make the approximation $\tanh z \approx z$, as it leads to an instability), are given in Methods, Eqs. (M.47) and (M.48).

S1.3 Simplifying the single neuron learning rules

In our simulations we use the update rules derived above (and given explicitly in Methods, Sec. M2.2). However, for illustrative purposes, in the main text, Eq. (4), we gave simplified learning rules relevant to linear feedback, $f = f_{\rm lin} = \delta + \xi_\delta$. Here we derive those simplified rules. The derivation involves rather severe approximations; we make them only to illustrate the essence of the learning rules in the simplest possible setting.

For this analysis we consider the small noise regime, $s_i^2 \ll 1$. In that regime, Methods, Eq. (M.13b) becomes

$$\sigma_i^2 \approx \mu_i^2 s_i^2. \tag{S.20}$$

Combining this with Methods, Eq. (M.13a), we have

$$\Delta\mu_i = \mu_i \left(\Delta m_i + \tfrac{1}{2}\Delta s_i^2\right) \tag{S.21a}$$

$$\Delta\sigma_i^2 = 2\mu_i^2 s_i^2 \left(\Delta m_i + \tfrac{1}{2}\Delta s_i^2\right) + \mu_i^2 \Delta s_i^2 \,. \tag{S.21b}$$

Because Δs_i^2 is a factor of s_i^2 smaller than Δm_i (see Eq. (S.9)), our small noise approximation lets us drop the term proportional to Δs_i^2 in the first expression and the term proportional to s_i^2 in the second expression. Consequently, the update rules for μ_i and σ_i^2 become

$$\Delta\mu_i \approx \mu_i \Delta m_i \tag{S.22a}$$

$$\Delta\sigma_i^2 \approx \mu_i^2 \Delta s_i^2 \,. \tag{S.22b}$$

Inserting these into Eq. (S.9) and using the approximation given in Eq. (S.20), we arrive at

$$\Delta\mu_i \approx \frac{\sigma_i^2}{\sigma_\delta^2}\, x_i f_{\rm lin} - \frac{\mu_i}{\tau}\left(m_i - m_{\rm prior}\right) \tag{S.23a}$$

$$\Delta\sigma_i^2 \approx -\frac{\sigma_i^4}{\sigma_\delta^2}\, x_i^2 - \frac{2\mu_i^2}{\tau}\left(s_i^2 - s_{\rm prior}^2\right) \,. \tag{S.23b}$$

As in Methods, we neglected the term $x_i(w_i - \mu_i)$ because it is small compared to $f_{\rm lin}$ (see comments following Eq. (M.56)).

To show that the contribution from the prior is approximately the form given in the main text, Eq. (4a), we use Methods, Eq. (M.13a) to write

$$\mu_i - \mu_{\rm prior} = \mu_i \left(1 - e^{-\left(m_i - m_{\rm prior}\right) - \left(s_i^2 - s_{\rm prior}^2\right)/2}\right). \tag{S.24}$$

Taylor expanding the exponent and neglecting both s_i^2 and $s_{\rm prior}^2$, we arrive at

$$\mu_i - \mu_{\rm prior} \approx \mu_i \left(m_i - m_{\rm prior}\right) \,. \tag{S.25}$$

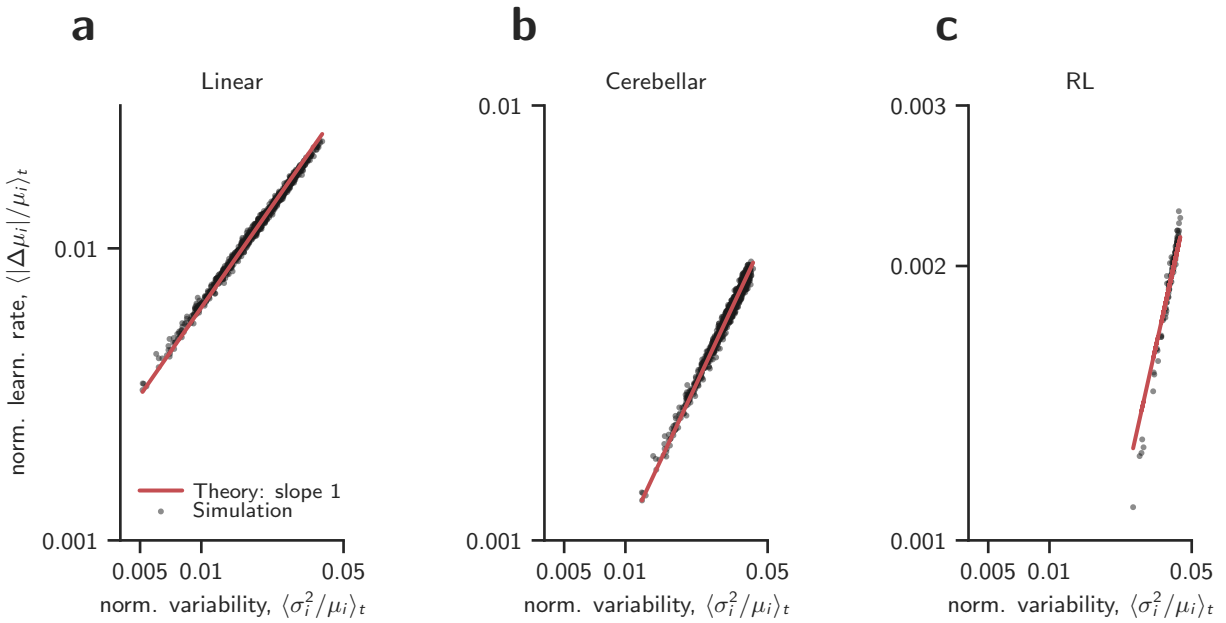

Figure S.2: Time average of normalized learning rate in our simulations, $\langle|\Delta\mu_i/\mu_i|\rangle_t$, versus normalized variability, $\langle\sigma_i^2/\mu_i\rangle_t$. As predicted in Sec. S1.3, $|\Delta\mu_i/\mu_i| \propto \sigma_i^2/\mu_i$, so the slope is 1 on a log-log plot. **a**. Linear feedback, $f_{\text{lin}} = \delta + \xi_\delta$. **b**. Cerebellar learning, $f_{\text{cb}} = \Theta(\delta + \xi_\delta - \theta)$. **c**. Reinforcement learning, $f_{\text{rl}} = -|\delta + \xi_\delta|$. Parameters from Table 1 (Sec. S3); w_i was sampled from the posterior.

Similarly, we use our standard approximation for the variance,

$$\sigma_i^2 - \sigma_{\text{prior}}^2 \approx s_i^2\mu_i^2 - s_{\text{prior}}^2\mu_{\text{prior}}^2 \approx \mu_i^2(s_i^2 - s_{\text{prior}}^2) \tag{S.26}$$

where the last expression follows, approximately, because $\mathbb{E}[\mu_i] = \mu_{\text{prior}}$.

The learning rules given in Eq. (S.23) imply that the change in the mean weight, $\Delta\mu_i$, is proportional to the variance, σ_i^2. Thus, the relative change in the mean weight, $\Delta\mu_i/\mu_i$, is proportional to the variance divided by the mean, which is the normalized uncertainty. Under sampling, the latter should be equal to the observed normalized uncertainty. This is a very general feature: under the approximations given in Eq. (S.22), for all of our learning rules (Eqs. (M.41), (M.43) and (M.47)), $\Delta\mu_i/\mu_i \propto \sigma_i^2/\mu_i$. We see this in our simulations; see Fig. S.2.

S2 The relationship between variability and firing rate

To find the relationship between the mean and uncertainty, μ_i and σ_i^2, and the firing rate, ν_i, we consider the steady state behavior of Eq. (S.8b), where $\langle \Delta s_i^2 \rangle = 0$,

$$0 \approx \frac{x_i s_i^4 \mu_i^2}{\sigma_\delta^2} \frac{\sigma_\delta^2 - \mathrm{Var}\left[f_{\mathrm{lin}}\right]}{\sigma_\delta^2} + \frac{2}{\tau}\left(s_i^2 - s_{\mathrm{prior}}^2\right). \tag{S.27}$$

Replacing x_i by its average, $\nu_i \Delta t$, making the definition

$$\chi_i \equiv \frac{\sigma_\delta^2 - \mathrm{Var}\left[f_{\mathrm{lin}}\right]}{\sigma_\delta^2}, \tag{S.28}$$

solving for s_i^2, and then replacing s_i^2 using the approximate expression $s_i^2 \approx \sigma_i^2/\mu_i^2$ (Eq. (S.20)), we arrive at

$$\frac{\sigma_i^2}{\mu_i} \approx \frac{\left(2\mu_i^2 \tau \nu_i \Delta t \chi_i s_{\mathrm{prior}}^2/\sigma_\delta^2 + 1\right)^{1/2} - 1}{\mu_i \tau \nu_i \Delta t \chi_i / \sigma_\delta^2}. \tag{S.29}$$

In the limit that the firing rate is sufficiently large,

$$\frac{\mu_i^2 \tau \nu_i \Delta t \chi_i s_{\mathrm{prior}}^2}{\sigma_\delta^2} \gg 1, \tag{S.30}$$

Eq. (S.29) simplifies considerably,

$$\frac{\sigma_i^2}{\mu_i} \approx \frac{\sigma_\delta}{\sqrt{\nu_i \Delta t}} \left(\frac{2 s_{\mathrm{prior}}^2}{\tau \chi_i}\right)^{1/2}. \tag{S.31}$$

This last expression tells us that for sufficiently large presynaptic firing rate, the normalized variability scales as $\nu^{-1/2}$. However, for small firing rate there are nontrivial corrections. Using Eq. (S.29), and noting that μ_i is independent of the presynaptic firing rate, ν_i, our analysis predicts that the normalized variability should depend on presynaptic firing rate as

$$\frac{\sigma_i^2}{\mu_i} = a \frac{\sqrt{\nu_i/\nu_0 + 1} - 1}{\nu_i}. \tag{S.32}$$

In Fig. S.3 (top row) we plot σ_i^2/μ_i versus firing rate on a log-log plot, along with the prediction given in Eq. (S.32) (with parameters a and ν_0 chosen to minimized the mean squared error between the data and the prediction). The fit is remarkably good, especially given the rather gross approximations that we made. Note that the correction to the $1/\sqrt{\nu}$ scaling is most pronounced for reinforcement learning. In hindsight, that's expected: χ is the difference between the variance of δ and the variance of f_{lin}; for reinforcement learning, the goal of the learning rule is to make these two quantities as close as possible (see Methods, Eq. (M.47), and note that $f_{\mathrm{rl}}^2 = f_{\mathrm{lin}}^2$), so χ is small; and small χ implies large ν_0, and thus a large correction.

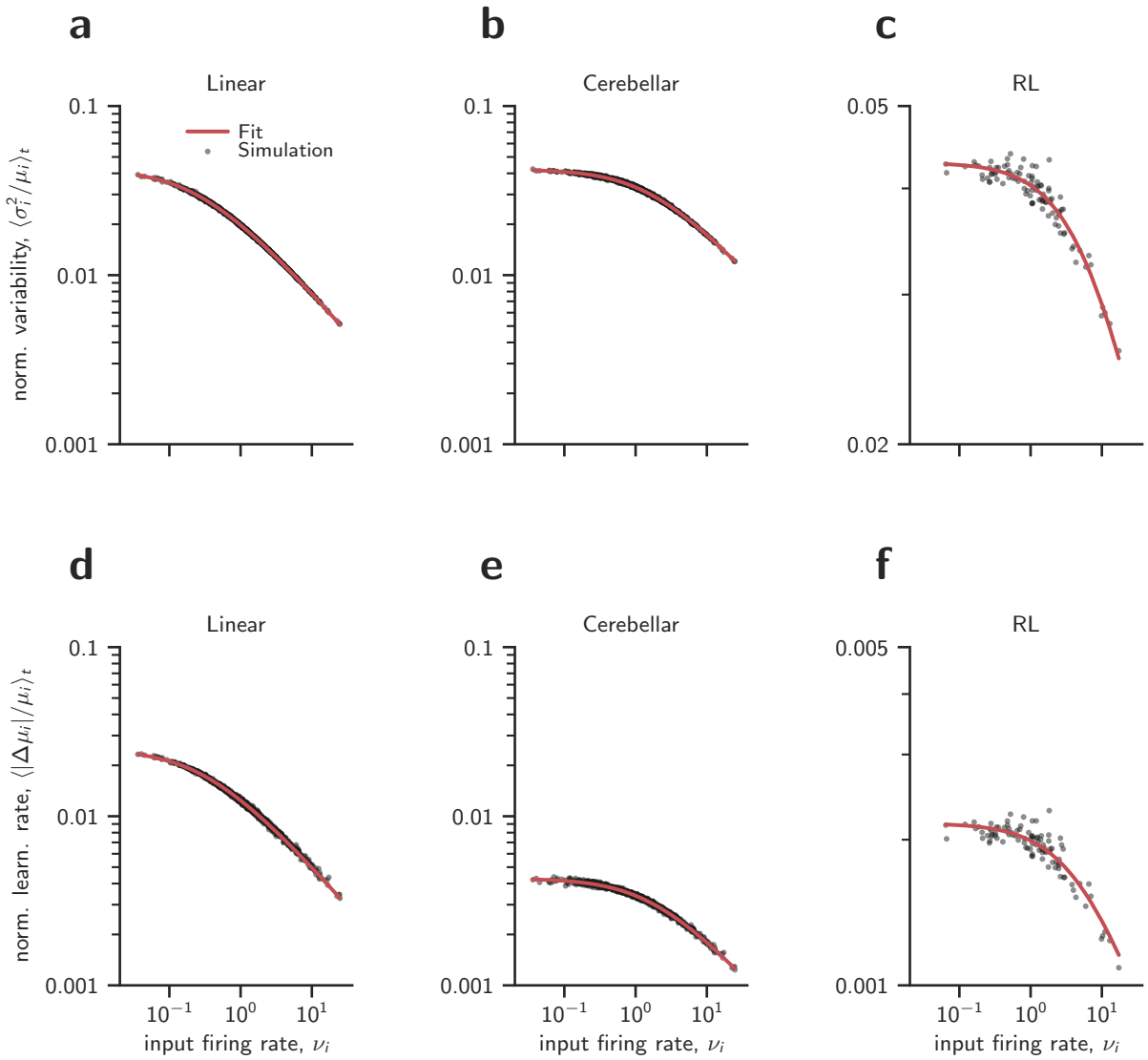

Figure S.3: Time average of normalized variability in our simulations, $\langle \sigma_i^2/\mu_i \rangle_t$ (top row), and normalized learning rate, $\langle |\Delta\mu_i/\mu_i| \rangle_t$ (bottom row), versus presynaptic firing rate, ν_i. Red lines are best fit (minimum mean squared error) to Eq. (S.32). As predicted in that equation, both have a slope of -1 for sufficiently high firing rate. **a** and **d**. Linear feedback, $f = \delta + \xi_\delta$. **b** and **e**. Cerebellar learning, $f = \Theta(\delta + \xi_\delta - \theta)$. **c** and **f**. Reinforcement learning, $f = -|\delta + \xi_\delta|$. Parameters from Table 1 (Sec. S3); w_i was sampled from the posterior.

Because the normalized learning rate, $|\Delta\mu_i/\mu_i|$, is proportional to σ_i^2/μ_i (see Fig. S.2), we expect $|\Delta\mu_i/\mu_i|$ to also be well fit by Eq. (S.32), with approximately the same values of a and ν_0 as in the top row of Fig. S.3. We do indeed see this in our simulations (Fig. S.3, bottom row).

Parameter	Value	Basis
m_{prior}	-0.669	Matched to data from [2] (Sec. S4.1)
s^2_{prior}	0.863	Matched to data from [2] (Sec. S4.1)
n (linear, cerebellar)	1000	Number of synapses; offers a good trade-off between biological realism [3] and computational tractability
n (reinforcement)	100	Number of synapses; uses a reduced number because of the difficulty of the learning problem
τ (linear, cerebellar)	10^5	corresponds to 1,000 s (Sec. S4.3)
τ (reinforcement)	5×10^5	corresponds to 5,000 s (Sec. S4.3)
Δt	10 ms	Time step; set to the typical membrane time constant [4]
σ_0	2 mV	Standard deviation of combined membrane potential and feedback signal noise.
k	0.0877	Normalized variability, matched to data [2] (Sec. S4.2)
θ	-4.2	Used for the cerebellar feedback; chosen so that f_{cb} is 1 about every 100 time steps, corresponding to a feedback signal of about 1 Hz.

Table 1: Parameters used in the single neuron simulations (Figs. 2 and 3, main text).

S3 Simulations details

Our simulations were relatively straightforward: either we iterated the single neuron update rules (Eqs. (M.41), (M.43) or (M.47) for the Bayesian rules and Eqs. (M.42), (M.46) or (M.48) for the classical ones) or solved the differential equation for the recurrent neural network (Methods, Eq. (M.60) for the Bayesian rules or the same equation but with η_i set to a constant, independent of time or synapse, for the classical ones).

For Figs. 2 and 3 (single neuron update rules; see Table 1), we did not sample for the linear and cerebellar rules, and we used sampling with variance proportional to the mean for the reinforcement learning rule, as described in Methods, Sec. M1.2. For both figures, we ran the simulations for 500 OU time constants. In Fig. 2, we plotted the last three OU time constants. For Fig. 3, we used the first two OU time constants for burn-in and then computed the mean squared error using the remaining 498 OU time constants.

The parameters of the recurrent neural network (Methods, Eq. (M.49)) are given in Table 2. Those parameters mainly describe the weights and target functions, which were chosen

as follows. The recurrent weights were sampled randomly as in [5],

$$J_{ij} = \xi_{ij} J^0_{ij} \tag{S.33}$$

where ξ_{ij} is a Bernoulli random variable with probability p, and J^0_{ij} is Gaussian and independent,

$$\xi_{ij} \sim \text{Bernoulli}(p) \tag{S.34a}$$

$$J^0_{ij} \sim \mathcal{N}\left(0, \frac{g^2}{pN}\right) \tag{S.34b}$$

(recall that N is the number of neurons in the network). The feedback weights, A_i, were also random, and chosen to be uniform between -1 and 1. The readout weights, w_i, were initialized to be Gaussian and independent,

$$w_i \sim \mathcal{N}\left(0, \frac{1}{N}\right) . \tag{S.35}$$

The inputs, I_i, were transient pulses of 10 ms duration with amplitude sampled from a uniform distribution between -1 and 1. The initial conditions, $x_i(0)$, were obtained by first running the network with no input ($I_i = 0$) for 2000 ms. This allowed the network to relax to a strange attractor determined by its intrinsic dynamics.

The target functions, $V_{\text{tar}}(t)$, were sampled randomly from a Gaussian process with periodic kernel

$$k(t, t') = \exp\left[-2\sin^2\left(\frac{\pi|t - t'|}{\tau_{\text{target}}}\right)\right] . \tag{S.36}$$

This kernel ensures that the sampled functions are periodic with period τ_{target}.

We sampled 20 random target functions and 20 random recurrent networks as indicated above. For each network/target pair, we continually trained for 6000 periods of the target function (we show only the first 1000 in Fig. 4 because the Bayesian learning rules converged by then). Every 100 periods we stopped training, ran the network for 50 periods under the current readout weights (without feedback), and computed the mean squared error (MSE) between the readout and the true target function.

All figures show median MSE over all 400 network/target pairs. Error bars are bootstrapped confidence intervals for the median, using the percentile bootstrap method. Median was used rather than mean because there were always a few pairs in which learning failed. In these cases, the MSE remained very high, thus distorting the mean MSE in a way that did not allow for an illustrative comparison.

Parameter	Value	Basis
N	500	Number of neurons in the network
g	1.5	Scale factor for weights; this put the network in a mildly chaotic regime
p	0.1	Connection probability
τ_m	10 ms	Time constant of the v_i
τ_{target}	500 ms	Period of sampled functions

Table 2: Parameters used in the recurrent neural network simulations (Fig. 4)

S4 Choosing parameters

Below we summarize how we chose parameters for the single neuron update rules.

S4.1 Estimate of priors from data

To determine the prior mean and variance of the log weights, m_{prior} and s^2_{prior}, we used connectivity data from Ref. [2] (http://plasticity.muhc.mcgill.ca/DataPage/DataPage.html). The data is the mean and variance (over trials) of synaptic strength from paired recordings in rat visual cortex *in vitro*. To translate these to the mean and variance of the log-normal, we inverted Methods, Eq. (M.13),

$$m_i = \log \mu_i - \frac{1}{2} \log \left(1 + \frac{\sigma_i^2}{\mu_i^2} \right) \tag{S.37}$$

where μ_i and σ_i^2 are the mean and variance of the i^{th} connection strength. The mean and variance of the prior are then given by the empirical mean and variance of the connections strengths,

$$m_{\text{prior}} = \frac{1}{M} \sum_{i=1}^{M} m_i \tag{S.38a}$$

$$s^2_{\text{prior}} = \frac{1}{M} \sum_{i=1}^{M} (m_i - m_{\text{prior}})^2 \tag{S.38b}$$

where M (=852) is the number of connection strengths in the dataset. We are assuming that μ_i and σ_i^2 are good estimates of the true mean and variance of the log weights, and that an average over neurons is a good proxy for an average over time.

S4.2 Variance versus mean: computing k

As discussed in Methods, Sec. M1.2, to make comparison with the classical reinforcement learning rule, we sample from the weights (see Eq. (M.14)). To do that, we assume that

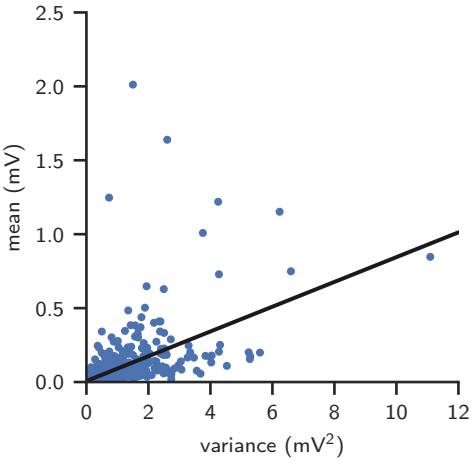

Figure S.4: Variance versus mean, using data (blue dots) taken from Ref. [2]. A least squares fit to the data gives $\sigma^2 = k\mu$ with $k = 0.0877$ (black line).

the variance is linear in the mean, and compute the slope, k, using, as above, data from Ref. [2] (http://plasticity.muhc.mcgill.ca/DataPage/DataPage.html). A plot of the variance, σ^2, versus the mean, μ, is given in Fig. S.4, along with the best fitting line, $\sigma^2 = 0.0877\,\mu$.

S4.3 Timescale for weight drift

As shown in Sec. S2, the normalized variability, σ_i^2/μ_i, is related to the timescale for weight drift, τ. To get a very approximate estimate for τ, we replace all quantities in Eq. (S.31) by an average over neurons, for which we use an overline. Doing this, and solving that for τ, we have

$$\tau \approx \frac{\sigma_\delta^2}{\overline{\nu}\Delta t}\,\frac{2s_{\text{prior}}^2}{\overline{\chi}}\,\frac{1}{\left(\overline{\sigma^2/\mu}\right)^2}\,. \tag{S.39}$$

To determine the size of σ_δ^2, we use Methods, Eq. (M.38), assume sampling (Eq. (M.39)) and small $\nu_i\Delta t$, and ignore σ_0^2; that gives us

$$\sigma_\delta^2 \approx 2n\sigma_{\text{prior}}^2\overline{\nu}\Delta t \tag{S.40}$$

where, recall, n is the number of presynaptic neurons. For the remaining parameters we use $\overline{\sigma^2/\mu} = k = 0.0877$ and $\sigma_{\text{prior}}^2 = m_{\text{prior}}^2[e^{s_{\text{prior}}^2} - 1]$ (Methods, Eq. (M.13)). The latter quantity is approximately equal to 0.61 (Table 1), so Eq. (S.39) becomes

$$\tau \approx \frac{270\,n}{\overline{\chi}}\,. \tag{S.41}$$

For linear and cerebellar learning, $n = 1000$ and $\overline{\chi}$ is $\mathcal{O}(1)$; consequently, τ should be on the order of 3×10^5; we used 10^5 in our simulations. (Recall that τ is measured in timesteps,

which are 10 ms, so $\tau = 10^5$ corresponds to 1000 s.) For reinforcement learning, we reduce n by a factor of 10, to 100 (because learning is hard). However, as discussed in Sec. S2, $\overline{\chi}$ is small; we therefore used $\tau = 5 \times 10^5$. While the approximations are somewhat crude, in simulations (Fig. S.3, top row) the normalized variability is indeed near its experimental value, 0.0877.

S4.4 Firing rate data used for Fig. 5

To obtain the p-value for Fig. 5, we performed standard linear regression: we regressed log(variance/mean) against log(firing rate) and log(mean); the former to test our prediction and the latter to eliminate the PSP amplitude as a possible confound. To estimate the firing rate, we took the mean of a FOOPSI-based firing rate estimate [6] supplied to us by the authors of [7]. This estimate is proportional to the true firing rate [8]; because our predicted relationship was linear on a log-log plot, the constant of proportionality plays no role. Using this approach, the best fit line was statistically significantly different from zero ($p < 0.003$, t-test, $n = 135$), and its slope, -0.62 was not significantly different from our prediction, $-1/2$ ($p = 0.57$, t-test).

S5 Synaptic Sampling

Here we provide an expanded normative argument for Synaptic Sampling. The argument starts with the observation that to select the correct action, knowing the uncertainty in task relevant quantities is critical [9]. For instance, to decide whether you can jump over a puddle without getting your feet wet, you need more than just an estimate of mean landing location; you also need an estimate of uncertainty (Fig. S.5). Uncertainty about the landing location comes from two sources: uncertainty about the current state of the world and uncertainty about the target weights (i.e., the weights that would give the best estimate of landing location). To see how the brain might compute uncertainty in landing location, consider a simplified scenario in which we use $\mathbf{x}_{\text{tar}}$ to denote the best possible spike-based representation of the true state of the external world. The neuron's estimate of landing location is a function of the neuron's output, V, so the optimal estimate of landing location is given by the target output,

$$V_{\text{tar}} = \mathbf{w}_{\text{tar}} \cdot \mathbf{x}_{\text{tar}}. \tag{S.42}$$

The assumption that the synapse combines $\mathbf{w}_{\text{tar}}$ and $\mathbf{x}_{\text{tar}}$ via a dot product is for simplicity only; the cell could use any nonlinear relationship and our arguments would hold.

Of course, the brain knows neither the target weights, $\mathbf{w}_{\text{tar}}$, nor the true state of the external world, $\mathbf{x}_{\text{tar}}$. The brain could compute a "best guess" of $\mathbf{x}_{\text{tar}}$, and the neuron could

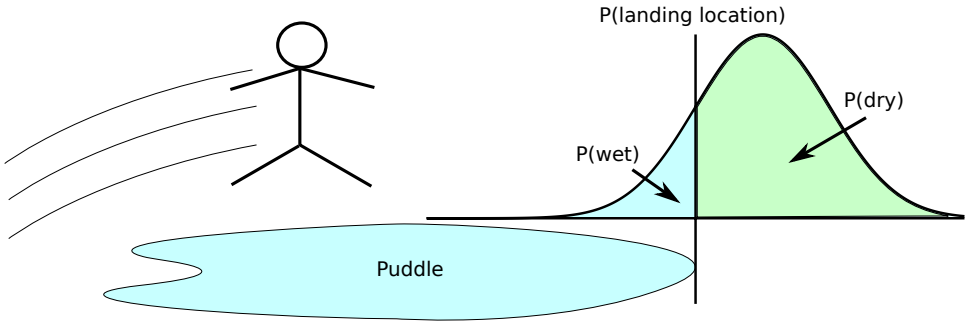

Figure S.5: A schematic diagram of a stick-person jumping over a puddle. The probability of landing in the puddle, $P(\text{wet})$, depends not only on the mean estimate, but also on the uncertainty.

use a "best guess" of $\mathbf{w}_{\text{tar}}$, resulting in

$$V_{\text{best guess}} = \mathbf{w}_{\text{best guess}} \cdot \mathbf{x}_{\text{best guess}} \,. \tag{S.43}$$

However, this scheme is unable to give an estimate of uncertainty — so offers little guidance as to whether or not to jump over the puddle.

To get an estimate of uncertainty, it is necessary to account for uncertainty both in the state of the world, $\mathbf{x}_{\text{tar}}$, and in the relationship between the state of the world and jump distance, parameterized by $\mathbf{w}_{\text{tar}}$. As information about $\mathbf{x}_{\text{tar}}$ comes from sensory data, and information about $\mathbf{w}_{\text{tar}}$ comes from training data (e.g., from past jumps), we can represent our (probabilistic) knowledge about these quantities as two distributions, $P(\mathbf{x}_{\text{tar}}|\text{Sensory Data})$ and $P(\mathbf{w}_{\text{tar}}|\text{Training Data})$. To combine these distributions into a distribution over V_{tar}, we need to integrate over all possible settings of $\mathbf{x}_{\text{tar}}$ and $\mathbf{w}_{\text{tar}}$,

$$\begin{aligned} &P(V_{\text{tar}}|\text{Sensory Data}, \text{Training Data}) = \\ &\quad \int d\mathbf{w}_{\text{tar}}\, d\mathbf{x}_{\text{tar}}\, P(V_{\text{tar}}|\mathbf{x}_{\text{tar}}, \mathbf{w}_{\text{tar}})\, P(\mathbf{x}_{\text{tar}}|\text{Sensory Data})\, P(\mathbf{w}_{\text{tar}}|\text{Training Data})\,. \end{aligned} \tag{S.44}$$

It is difficult for neurons to compute this integral, as it is high dimensional and rarely has a closed form expression. However, by combining neural and synaptic sampling, it is possible for neural circuits to evaluate the integral via sampling; that is, by drawing samples, V, from the distribution,

$$V \sim P(V_{\text{tar}}|\text{Sensory Data}, \text{Training Data})\,. \tag{S.45}$$

To do that, we simply need to draw neural activity, $\mathbf{x}$, from its distribution given sensory data,

$$\mathbf{x} \sim P(\mathbf{x}_{\text{tar}}|\text{Sensory Data}) \tag{S.46}$$

(this is known as the neural sampling hypothesis [10, 11, 12]), and draw synaptic weights, $\mathbf{w}$, from their distribution given training data,

$$\mathbf{w} \sim P(\mathbf{w}_{\text{tar}}|\text{Training Data}) \tag{S.47}$$

(this is our Synaptic Sampling hypothesis). A sample of landing location is given by combining the sampled neural activity with the sampled weights, which could be done by a single neuron,

$$V = \mathbf{w} \cdot \mathbf{x}\,. \tag{S.48}$$

Thus, simply by drawing repeated samples, a single neuron can estimate uncertainty about V, and thus about landing location.

Our argument appears to assume that the brain uses the output of a single neuron to make predictions. This is not too implausible — the cerebellum does contain a large number of Purkinje cells [13] that are believed to use supervised learning to, among other things, make predictions. However, it is certainly possible that such a computation is performed by a large multi-layer network. As long as the network is effectively feedforward, we can still, by the logic described above, estimate its uncertainty by combining synaptic sampling with neural sampling.

S6 Robustness

Our Bayesian update rules depend on a number of parameters, and in our simulations so far we set them to their theoretically optimal values. However, the brain can't do this; there will always be some model mismatch. Here we explore the robustness of Bayesian plasticity to this mismatch. For linear, cerebellar and reinforcement learning, there are three main parameters: σ_0^2, the combined noise in the feedback signal and membrane potential, and m_{prior} and s_{prior}^2, the mean and variance that govern the random drift in synaptic weights (see Eqs. (M.41), (M.43) and (M.47)). For these parameters, we examine performance when they change by a factor of 2 in either direction relative to their nominal values (with, of course, the update rules assuming they have not changed). The results are shown in Figs. S.6a-c. The linear and cerebellar rules were robust with respect to changes in all these parameters. Sensitivity to m_{prior} was highest, although $\pm 50\%$ changes in that parameter had little effect on the mean squared error. For the other two parameters, the mean squared error changed very little over the range tested. The reinforcement learning rule was more sensitive to parameters, especially σ_0^2, where small decreases led to an instability that greatly increased the mean squared error. However, increases had little effect. Overall, the Bayesian learning rule is relatively robust. It certainly does not require fine tuning.

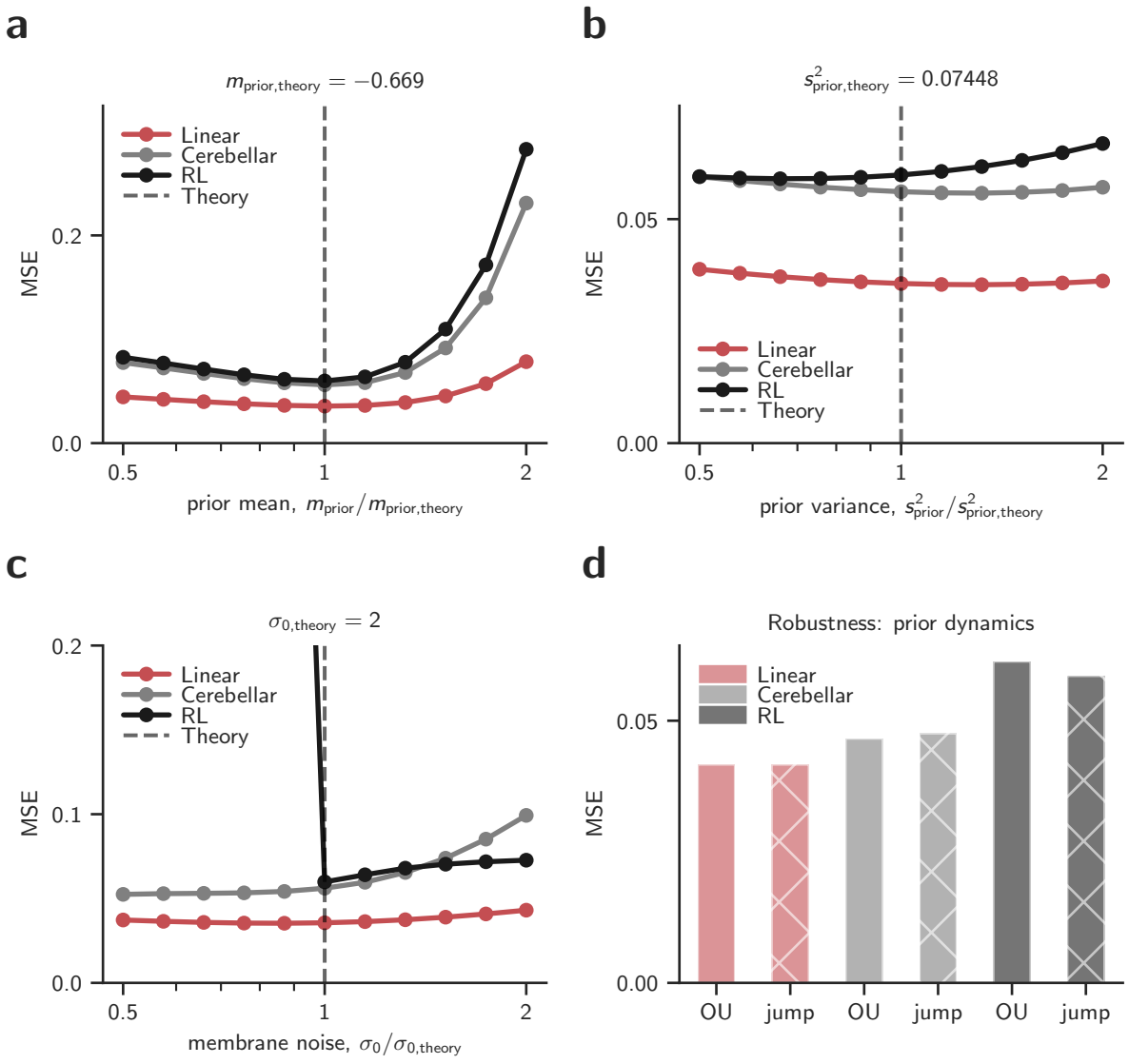

Figure S.6: Robustness with respect to parameters for the single neuron learning rules. In panels a-c, quantities with a subscript "theory" are the optimal values as predicted by our theoretical model. **a**. m_{prior}; mean of the Ornstein-Uhlenbeck (OU) process (Methods, Eq. (M.11)). **b**. s^2_{prior}; variance of the OU process (Eq. (M.11)). **c**. σ_0; standard deviation of the combined noise in the membrane potential and the feedback signal (Methods, Eq. (M.33)). **d**. Effect of using jump, rather than Gaussian, noise in the weight drift. For "OU", $\xi_{\text{tar},i} \sim \mathcal{N}(0,1)$; for "jump", $\xi_{\text{tar},i}$ is $+1$ or -1, with equal probability.

We also looked at the effect of binary, rather than Gaussian, drift. That is, in Methods, Eq. (M.11), rather than drawing $\xi_{\text{tar},i}$ from a Gaussian distribution, we used $\xi_{\text{tar},i} = \pm 1$, with $+1$ and -1 occurring with equal probability. As can be seen in Fig. S.6d, the results were virtually identical to the Gaussian case. Given the small time step relative to the OU time

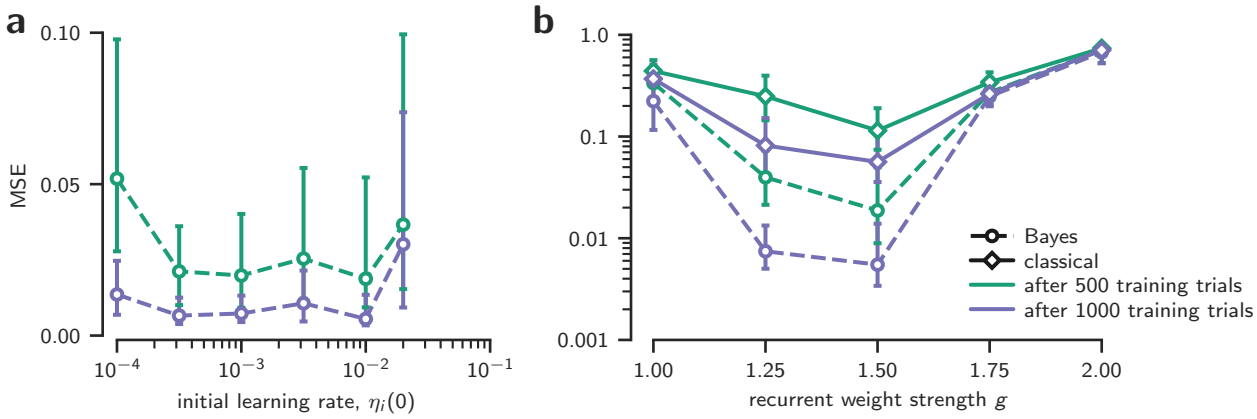

Figure S.7: Robustness with respect to parameters for the recurrent neural network. **a**. Mean squared error versus the initial learning rate, $\eta_i(0)$. The value used in our simulations was $\eta_i(0) = 10^{-2}$. **b**. Mean squared error of the Bayesian and classical learning rule as a function of g, the parameter that controls how chaotic the network is (see Eq. (S.34b); we used $g = 1.5$ in our simulations). In both panels we report median over $n = 400$ network/target pairs, and error bars are 95% confidence intervals computed using the percentile bootstrap.

constant, this is not surprising.

For the recurrent neural network, the main parameter in the update rule is the initial learning rate, $\eta_i(0)$ (Methods, see Eq. (M.60)). Performance is extremely insensitive to this parameter: it can change by a factor of 30 without affecting the mean squared error (Fig. S.7a). We also checked sensitivity to the parameter g, which determines how chaotic the network is (see Eq. (S.34b)). For values of g that yielded low mean squared errors (1.25 and 1.5; we used 1.5 in our simulations), the Bayesian learning rules remained about an order of magnitude better than the classical ones (Fig. S.7b). Thus, Bayesian learning in the recurrent neural network is extremely robust.

References

[1] A. Gelb. *Applied Optimal Estimation*. MIT Press, 1974.

[2] S. Song et al. "Highly nonrandom features of synaptic connectivity in local cortical circuits". In: *PLoS Biology* 3.3 (2005), e68.

[3] T. Binzegger, R. Douglas, and K. Martin. "A quantitative map of the circuit of cat primary visual cortex". In: *J. Neurosci.* 24 (2004), pp. 8441–8453.

[4] S. J. Tripathy et al. "Brain-wide analysis of electrophysiological diversity yields novel categorization of mammalian neuron types". In: *Journal of Neurophysiology* 113.10 (2015), pp. 3474–3489.

[5] B. DePasquale et al. "Full-FORCE: A Target-Based Method for Training Recurrent Networks". In: *PLoS ONE* 13.2 (2018), e0191527.

[6] J. T. Vogelstein et al. "Fast nonnegative deconvolution for spike train inference from population calcium imaging". In: *Journal of Neurophysiology* 104.6 (2010), pp. 3691–3704.

[7] H. Ko et al. "The emergence of functional microcircuits in visual cortex". In: *Nature* 496.7443 (2013), pp. 96–100.

[8] A. M. Packer et al. "Simultaneous all-optical manipulation and recording of neural circuit activity with cellular resolution in vivo". In: *Nature Methods* 12.2 (2015), pp. 140–146.

[9] M. O. Ernst and M. S. Banks. "Humans integrate visual and haptic information in a statistically optimal fashion". In: *Nature* 415.6870 (2002), pp. 429–433.

[10] P. O. Hoyer and A. Hyvarinen. "Interpreting neural response variability as Monte Carlo sampling of the posterior". In: *Advances in Neural Information Processing Systems*. 2003, pp. 293–300.

[11] J. Fiser et al. "Statistically optimal perception and learning: from behavior to neural representations". In: *Trends in Cognitive Sciences* 14.3 (2010), pp. 119–130.

[12] P. Berkes et al. "Spontaneous Cortical Activity Reveals Hallmarks of an Optimal Internal Model of the Environment". In: *Science* 331.6013 (2011), pp. 83–87.

[13] P. Dean et al. "The cerebellar microcircuit as an adaptive filter: experimental and computational evidence". In: *Nature Reviews Neuroscience* 11.1 (2010), pp. 30–43.