Self-healing neural codes: Hebbian and homeostatic mechanisms can track evolving neural representations

M. E. Rule[1,a] and T. O'Leary[2,a]

[1]ORCID iD 0000-0002-4196-774X; mer49@eng.cam.ac.uk [2]ORCID iD 0000-0002-1029-0158; tso24@cam.ac.uk

[a]University of Cambridge, Engineering Department, Trumpington Street, Cambridge CB2 1PZ, United Kingdom

September 9, 2021

Recent studies have found that the tuning of neurons to behavioral variables "drifts" over time in many parts of the brain, even in the absence of measurable changes in behavior. This drift retains task information at the population level while allowing individual cells to change their tuning substantially. At the same time, other circuits exhibit long-term stability in tuning at the single cell level. How can unstable and stable neuronal populations interoperate reliably, and how might single cells maintain stable readout from a drifting population code? We explore how known plasticity mechanisms allow single neurons to retain learned representations without external error feedback. We find that interactions between Hebbian learning and single-cell homeostasis can exploit redundancy in a distributed population code to compensate for gradual changes in tuning. Recurrent feedback of partially stabilized readouts allows a pool of readout cells to further correct inconsistencies introduced by representational drift without an external learning signal. This shows how relatively simple, known mechanisms can stabilize neural tuning in the short term, and provides a plausible circuit architecture for long term maintenance of neural tuning. Our findings suggest a novel role for Hebbian plasticity in retaining existing memories as opposed to learning new associations.

The cellular and molecular components of the brain change continually. In addition to synaptic turnover (1), ongoing reconfiguration of the tuning properties of single neurons has been seen in parietal (2), frontal (3), visual (4, 5), and olfactory (6) cortices, and the hippocampus (7, 8). Remarkably, the "representational drift" (9) observed in these studies occurs without any obvious change in behavior or task performance. Reconciling dynamic reorganization of neural activity with stable circuit-level properties remains a major open challenge (9, 10). Furthermore, not all circuits in the brain show such prolific reconfiguration, including populations in primary sensory and motor cortices (11–13). How might populations with stable and drifting neural tuning communicate reliably? Put another way, how can an internally consistent 'readout' of neural representations survive changes in the tuning of individual cells?

These recent, widespread observations suggest that neural circuits can preserve learned associations at the population level while allowing the functional role of individual neurons to change (14–16). Such preservation is made possible by redundancy in population codes, because a distributed readout allows changes in the tuning of individual neurons to be offset by changes in others. However, this kind of stability is not automatic: changes in tuning must either be constrained in specific ways (e.g. 17, 18), or corrective plasticity needs to adapt the readout (19). Thus, while there are proposals for what might be required to maintain population codes dynamically, there are few suggestions as to how this might be implemented with known cellular mechanisms and without recourse to external reference signals that re-calibrate population activity to behavioral events and stimuli.

In this paper we show that neural tuning in a readout population can be made resilient to drift using well known plasticity mechanisms: Hebbian learning and homeostatic plasticity. Homeostasis is a feature of all biological systems, and examples of homeostatic plasticity in the nervous system are pervasive (e.g. 20, 21 for reviews). Broadly, homeostatic plasticity is a negative feedback process that maintains physiological properties such as average firing rates (e.g. 22), neuronal variability (e.g. 23), distributions of synaptic strengths (e.g. 24, 25), and population-level statistics (e.g. 26). Hebbian plasticity complements homeostatic plasticity by strengthening connectivity between cells that undergo correlated firing, further reinforcing correlations (27, 28). Pairwise correlations in a population provide local bases for a so-called task manifold in which task-related neural activity resides (29). We show that these two

Significance The brain reconfigures connections continuously while maintaining stable long-term memories and learned skills. This work examines how stable neural population codes can be maintained internally using known cellular mechanisms, despite large ongoing changes in connectivity. We show how Hebbian plasticity interacts with homeostasis in single neurons to allow a circuit to continuously repair a learned representation. This provides a plausible mechanism for retaining neural representations without reference to an external learning signal, and suggests a novel role for Hebbian plasticity in memory maintenance as opposed to learning.

Author Contributions TO: Conceptualization, Funding acquisition, Project administration, Supervision, Writing - review & editing MR: Conceptualization, Formal analysis, Investigation, Methodology, Simulation, Visualization, Writing - original draft, Writing - review & editing

mechanisms can operate on a redundant neural representation to constrain drift and preserve the global structure of a population code.

We first illustrate how single-cell feedback regulation can stabilize the information-coding capacity of a population; We then show how population-level feedback allows learned representations to be read out from unstable neural populations. This implies that long-term stability of learned associations is consistent with substantial instability in the neural codes that drive these associations, through relatively simple, known mechanisms. This implies a previously unidentified role for Hebbian plasticity in maintaining associations, as opposed to learning new ones.

Background

We briefly review representational drift and the broader context of the ideas used in this manuscript. Representational drift refers to seemingly random changes in neural responses during a habitual task that are not associated with learning (9). For example, in Driscoll et al. (2) mice navigated to one of two endpoints in a T-shaped maze (Fig. 1a), based on a visual cue. Population activity in Posterior Parietal Cortex (PPC) was recorded over several weeks using fluorescence calcium imaging. Neurons in PPC were tuned to the animal's past, current, and planned behavior. Gradually, the tuning of individual cells changed: neurons could change the location in the maze in which they fired, or become disengaged from the task (Fig. 1b). The neural population code eventually reconfigured completely (Fig. 1c). However, neural tunings continued to tile the task, indicating stable task information at the population level. These features of drift have been observed throughout the brain (4, 5, 8).

Downstream readouts could track gradual drift using external error feedback to re-learn how to interpret an evolving neural code, e.g. during ongoing rehearsal (19). Indeed, simulations confirm that learning in the presence of noise can lead to a steady state, in which drift is balanced by error feedback (30–33). Previous studies have also shown that stable functional connectivity could be maintained despite synaptic turnover (30, 34, 35). Recent work has also found that discrete representations can be stabilized using neural assemblies that exhibit robust, all-or-nothing reactivation (36, 37).

Our work extends these results as follows. Rather than using external learning signals (19, 30–32), we show that drift can be tracked using internally generated signals. We allow the functional role of neurons in an encoding population to reconfigure completely, rather than just the synaptic connectivity (30, 34, 35). We extend ideas based on neuronal assemblies (36, 37) to address the low-dimensional manifold dynamics observed in sensorimotor tasks (38) and spatial navigation (2, 7). The geometry of these representations is consistent over time, although the way it is reflected in neuronal firing changes (39, 40). Engineers have applied online recalibration and transfer learning and to track drift in brain-machine interface decoders (e.g. 41; 42 for review). Could neural circuits in the brain do something similar? We argue that neuronal homeostasis and Hebbian plasticity driven by internally generated prediction errors allows neural networks to "self heal".

Results

We explore how neural networks could track drift in low-dimensional manifold representations. There are two important general principles to keep in mind throughout. First, distributed neural representations are redundant. To create ambiguity at the macroscopic level, many smaller disruptive changes must occur in a coordinated way. Neurons can exploit this to improve their robustness to drift. Second, learning creates recurrent connections that allow neural populations to model and predict the world. Even if learning has ceased, these connections continue to constrain activity. This could allow a downstream readout to correct errors caused by by drift, and use this corrected readout as a training signal.

We first describe a model of representational drift, in which homeostasis stabilizes the capacity of a "drifting" population to encode information despite instability in single-neuron tunings. We then explore how a single neuron could stabilize its own readout in the presence of this upstream drift. In the latter half of the manuscript, we show that these rules imply a form of Hebbian learning that achieves homeostasis. We extend these ideas to neural populations, and show that recurrent dynamics can stabilize a readout of an unstable neural code.

A model for representational drift

We have previously used the data from Driscoll et al. (43) to assess how much plasticity would be required to track drift in a linear readout (19). However, these data contain gaps of several days, and the number of high signal-to-noise units tracked for over a month is limited. To explore continual, long-term drift, we therefore construct a model inspired by the features of representational drift seen in spatial navigation tasks (2, 7).

We focus on key properties of drift seen experiments. In both (2) and (7), neural populations encode continuous, low-dimensional behavioral variables (e.g. location). Neurons exhibit localized, 'bump-like' tuning to these variables. Tuning curves overlap, creating a redundant code. Over time, neurons change their preferred tunings. Nevertheless, on any given day there is always a complete 'tiling' of a behavioral variable, indicating stability at the population level.

To model this, we consider a population of N neurons that encode a behavioral variable, θ. We assume θ lies on a low-dimensional manifold, and is encoded in the vector of firing rates in a neural population with tuning curves $\mathbf{x}_d(\theta)=\{x_{d,1}(\theta),..,x_{d,N}(\theta)\}^\top$. These tunings change over time (day d).

We abstract away some details seen the experimental data in Fig. 1c. We focus on the slow component of drift, and model excess day-to-day tuning variability via a configurable parameter. We assume uniform coverage of the encoded space, which can be ensured by an appropriate choice of coordinates. We consider populations of 100 units that encode θ, and whose tunings evolve independently. Biologically, noise correlations and fluctuating task engagement would limit redundancy, but this would be offset by the larger number of units available.

To model drift, we first have to model an encoding 'feature' population whose responses depend on θ, and from which it is possible to construct bump-like tuning with a weighted readout. To keep our assumptions general, we do not assume that the encoding population has sparse, bump-like activity, and simply define a set of K random features (tuning curves),

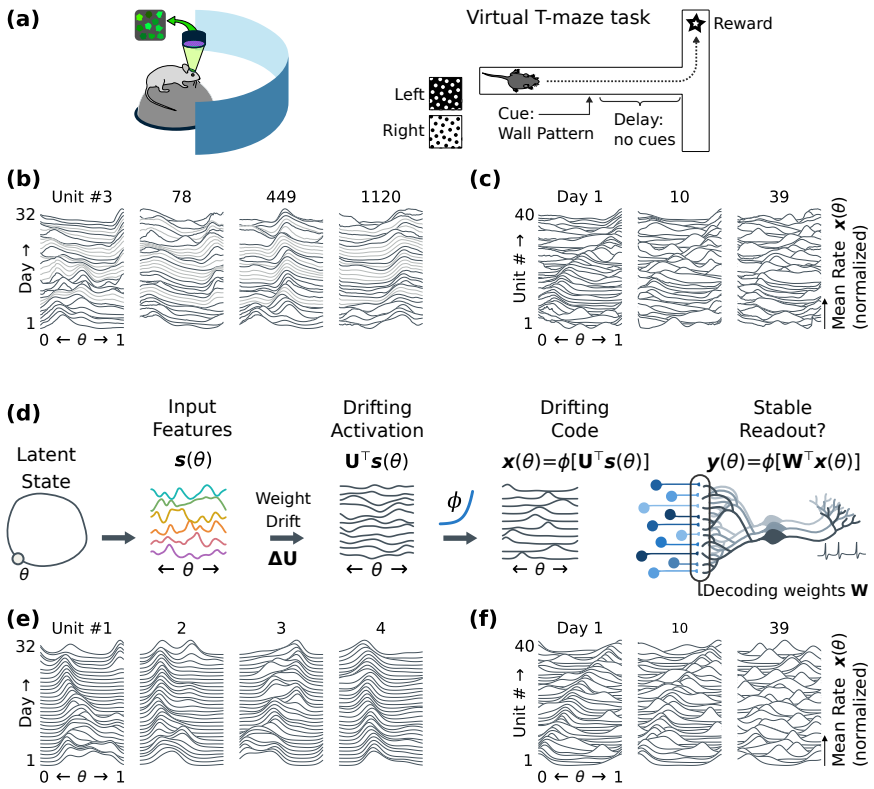

Figure 1: **A model for representational drift.** **(a)** Driscoll et al. (2) imaged population activity in PPC for several weeks, after mice had learned to navigate a virtual T-maze. Neuronal responses continued to change even without overt learning. **(b)** Tunings were often similar between days, but could change unexpectedly. Plots show average firing rates as a function of task pseudotime (0=beginning, 1=complete) for select cells from (2). Tuning curves from subsequent days are stacked vertically, from day 1 up to day 32. Missing days (light gray) are interpolated. Peaks indicate that a cell fired preferentially at a specific location (Methods: *Data and analysis*). **(c)** Neuronal tunings tiled the task. Within each day, the mouse's behavior could be decoded from population activity (2, 19). Plots show normalized tuning curves for 40 random cells, stacked vertically. Cells are sorted by their preferred location on day 1. By day 10, many cells have changed tuning. Day 39 shows little trace of the original code. **(d)** We model drift in a simulated rate network (Methods: *Simulated drift*). An encoding population $\mathbf{x}(\theta)$ receives input $\mathbf{s}(\theta)$ with low-dimensional structure, in this case a circular track with location θ. The encoding weights $\mathbf{U}$ driving the activations $\mathbf{U}^\top\mathbf{s}(\theta)$ of this population drift, leading to drifting activations. Homeostasis preserves bump-like tuning curves. **(e)** As in the data (a-c), this model shows stable tuning punctuated by large changes. **(f)** The neural code reorganizes, while continuing to tile the task. We will examine strategies that a downstream readout could use to update how it decodes $\mathbf{x}(\theta)$ to keep its own representation $\mathbf{y}(\theta)$ stable. This readout is also modeled as linear-nonlinear rate neurons, with decoding weights $\mathbf{W}$.

sampled independently from a random Gaussian process on θ. These features have an arbitrary but stable relationship to the external world, from which it is possible to reconstruct θ by choosing sufficiently large K:

$$\begin{aligned} \mathbf{s}(\theta)^\top &= \{s_1(\theta), .., s_K(\theta)\} \\ s_i(\theta) &\sim \mathcal{GP}[0, \Sigma(\theta, \theta')]. \end{aligned} \tag{1}$$

In the above equations, $\Sigma(\theta, \theta')$ denotes the covariance between the values of $s(\theta)$ at two states θ and θ'.

We next define an encoding of θ driven by these features with a drifting weight matrix $\mathbf{U}_d = \{\mathbf{u}_{d,1}, .., \mathbf{u}_{d,N}\}$, where $\mathbf{u}_{d,i}^\top = \{u_{d,i,1}, .., u_{d,i,K}\}$ reflects the encoding weights for unit $x_{d,i}(\theta)$ on day d. Each weight $u_{d,i,j}$ evolves as a discrete-time Ornstein-Uhlenbeck (OU) process, taking a new value on each day (Methods: *Simulated drift*). The firing rate of each encoding unit is given as a nonlinear function of the synaptic activation $a_{d,i}(\theta) = \mathbf{u}_{d,i}^\top \mathbf{s}(\theta)$:

$$x_{d,i}(\theta) = \phi[\gamma_i \, a_{d,i}(\theta) + \beta_i], \tag{2}$$

where γ_i and β_i are vectors that set the sensitivity and threshold of each unit. To model the nonlinear response of the readout and prevent negative firing rates, we use an exponential nonlinearity $\phi(\cdot) = \exp(\cdot)$.

In this model, the mean firing-rate and population sparsity of the readout can be tuned by varying the sensitivity γ and threshold β in Eq. (2). *In vivo*, these single cell properties are regulated by homeostasis (23). Stabilizing mean rates $\langle x_{d,i}(\theta)\rangle_\theta \approx \mu_0$ ensures that neurons remain active. Stabilizing rate variability $\mathrm{var}_\theta[x_{d,i}(\theta)] \approx \sigma_0^2$ controls population code sparsity, ensuring that $\mathbf{x}_d(\theta)$ carries information about θ (44). This is achieved by adapting the bias β_i and gain γ_i of each unit $x_{d,i}(\theta)$ based on the errors $\epsilon_\mu, \epsilon_\sigma$ between the statistics of neural activity and the homeostatic targets μ_0, σ_0:

$$\begin{aligned} \Delta\gamma &\propto \varepsilon_\sigma = \sigma_0 - \sigma_x \\ \Delta\beta &\propto \varepsilon_\mu = \mu_0 - \mu_x \end{aligned} \tag{3}$$

Fig. 1 shows that this model qualitatively matches the drift seen *in vivo* (2). Tuning is typically stable, with intermittent changes (Fig. 1e). This occurs because the homeostatic regulation in Eq. (3) adjusts neuronal sensitivity and threshold to achieve a localized, bump-like tuning curve at the location of peak synaptic activation, θ_0. Changes in tuning arise when the drifting weight matrix causes the encoding neuron to be driven more strongly at a new value of θ. The simulated population code reconfigures gradually and completely over a period of time equivalent to several weeks in the experimental data (Fig. 1f).

Hebbian homeostasis improves readout stability without external error feedback

Neural population codes are often redundant, with multiple units responding to similar task features. Distributed readouts of redundant codes can therefore be robust to small changes in the tuning of individual cells. We explored the consequences of using such a readout as an internal error signal to retrain synaptic weights in a readout population, thereby compensating for gradual changes in a representation without external feedback. This re-encodes a learned readout function $\mathbf{y}(\theta)$ in terms of the new neural code $\mathbf{x}_d(\theta)$ on each "day" d. Such "self-healing" plasticity improves the tuning stability of neurons that are driven by unstable population codes, even in single neurons. We first sketch an example of this plasticity, and then explore why this works.

Using our drifting population code as input, we model a readout population of M neurons with tuning curves $\mathbf{y}_d(\theta) =$

$\{y_{d,1}(\theta), .., y_{d,M}(\theta)\}^\top$ (Fig. 1d). We model this decoder as a linear-nonlinear function, using decoding weights $\mathbf{W}$ and biases (thresholds) $\mathbf{b}$ (leaving dependence on the day d implicit):

$$\mathbf{y}(\theta) = \phi[\mathbf{W}^\top \mathbf{x}(\theta) + \mathbf{b}]. \tag{4}$$

On each simulated "day", we re-train the decoding weights using an unsupervised Hebbian learning rule (c.f. 45). This potentiates weights $w_{i,j}$ whose input $x_j(\theta)$ correlates with the postsynaptic firing rate $y_i(\theta)$. We modulate the learning rate by an estimate of the homeostatic error in firing-rate variability ($\tilde{\delta}$). Thresholds are similarly adapted based on the homeostatic error in mean-rate ($\tilde{\beta}$). We include a small baseline amount of weight decay (ρ) and a larger amount of weight decay (c) that is modulated by $\tilde{\delta}$. For a single readout neuron $y(\theta)$, the weights and biases evolve as:

$$\begin{aligned} \Delta\mathbf{w} &\propto \tilde{\delta}\left[\langle \mathbf{x}(\theta) y(\theta)^\top \rangle_\theta - c\mathbf{w}\right] - \rho\mathbf{w} \\ \Delta b &\propto \tilde{\beta}\,\langle \mathbf{x}(\theta)\rangle_\theta \end{aligned} \tag{5}$$

We apply Eq. (5) for 100 iterations on each simulated "day", sampling over all θ on each iteration. We assume that the timescale of Hebbian and homeostatic plasticity is no faster than the timescale of representational drift. The error terms $\tilde{\delta}$, $\tilde{\beta}$ are leaky integrators of instantaneous errors (Eq. (3)) for each cell, ε_σ, ε_μ, respectively: $\tilde{\delta}_{t+1} = 0.5\,\tilde{\delta}_t + \varepsilon_\sigma$ (analogously for $\tilde{\beta}$, ε_μ). For the readout $\mathbf{y}(\theta)$, the homeostatic targets (μ_0, σ_0) are set to the firing-rate statistics in the initial, trained state (before drift has occurred). Eq. (5) therefore acts homeostatically. Rather than scale weights uniformly, it adjusts the component of the weights most correlated with the postsynaptic output, $\mathbf{y}(\theta)$. Plasticity occurs only when homeostatic constraints are violated. Further discussion of this learning rule is given in Methods: *Synaptic learning rules*.

Fig. 2 simulates a single readout neuron driven by a drifting population code. In each subpanel (a-c) we introduce additional plasticity mechanisms and illustrate how drift affects the tuning curve of the neuron.

To test whether the readout can tolerate complete reconfiguration in the encoding population, we change encoding features one at a time. For each change, we select a new, random set of encoding weights $\mathbf{u}_i$ and apply homeostatic compensation to stabilize the mean and variability of $x_i(\theta)$. Eq. (5) is then applied to update the decoding weights of the readout cell. This procedure is applied 200 times, corresponding to two complete reconfigurations of the encoding population of N=100 cells (Methods: *Single-neuron readout*).

With fixed weights, drift reduces the readout's firing rate without changing its tuning (Fig. 2a),. This is because the initial tuning of the readout requires coincident activation of specific inputs to fire for its preferred θ_0. Drift gradually destroys this correlated drive, and is unlikely to spontaneously create a similar conjunction of features for some other θ. For small amounts of drift, homeostasis Eq. (3) can stabilize the readout by compensating for the reduction in drive (Fig. 2b). Eventually, however, no trace of the original encoding remains. At this point, a new (random) θ will begin to drive the readout more strongly. Homeostasis adjusts the sensitivity of the readout to form a new, bump-like tuning curve at this location.

Fig. 2c shows the consequences of Hebbian homeostasis. Drift in the encoding $\mathbf{x}(\theta)$ decreases the excitatory drive to the readout, activating Hebbian learning. Because small amounts of drift have minimal effect on tuning, the readout's own output provides a self-supervised teaching signal. It re-learns the decoding weights for inputs that have changed due to drift. Applying Hebbian homeostasis periodically improves stability, despite multiple complete reconfigurations of the encoding population. In effect, the readout's initial tuning curve is transported to a new set of weights that estimate the same function from an entirely different input (for further discussion see Supplement: *Weight filtering*). In the long term the representation degrades, for reasons we dissect in the next section.

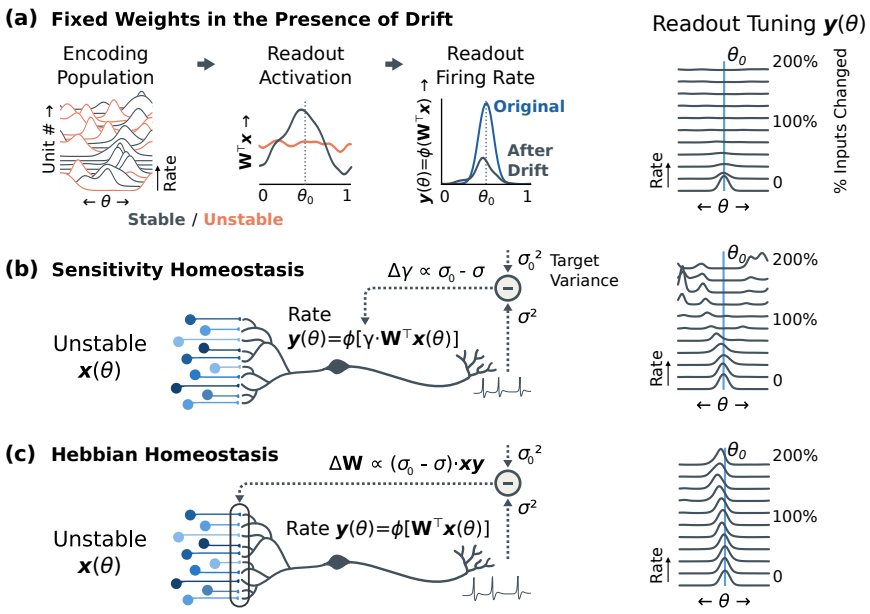

Figure 2: **Homeostatic Hebbian plasticity enables stable readout from unstable populations. (a)** Simulated drift in a redundant population causes a loss of excitability, but little change in tuning, to a downstream linear-nonlinear readout neuron. Since the cell is selective to a conjunction of features, it loses excitatory drive when some of its inputs change. Since most drift is orthogonal to this readout, however, the preferred tuning θ_0 does not change. The right-most plot shows that the excitability diminishes as a larger fraction of inputs change. Two complete reconfigurations of an encoding population of 100 cells is shown. **(b)** Homeostatic adjustments to increase the readout's sensitivity can compensate for small amounts of drift. As more inputs reconfigure, the cell compensates for loss of excitatory drive by increasing sensitivity ("gain", γ). However, the readout changes to a new, random location once a substantial fraction of inputs have reconfigured (right). This phenomenon is the same as the model for tuning curve drift in the encoding population (c.f. Fig. 1e). **(c)** Hebbian homeostasis increases neuronal variability by potentiating synaptic inputs that are correlated with post-synaptic activity, or depressing those same synapses when neuronal variability is too high. This results in the neuron re-learning how to decode its own tuning curve from the shifting population code, improving the stability of the readout despite complete reconfiguration (right). (Methods: *Single-neuron readout*)

Hebbian homeostasis with network interactions

In the remainder of the manuscript, we show how Hebbian homeostatic principles combine with population-level interactions to make readouts more robust to drift. Generally, a mechanism for tracking drift in a neural population should exhibit three features:

I The readout should use redundancy to mitigate error caused by drift.

II The readout should use its own activity as a training signal to update its decoding weights.

III The correlations in input-driven activity in the readout neurons should be homeostatically preserved.

We explore three types of recurrent population dynamics that could support this: (1) Population firing-rate normalization; (2) Recurrent dynamics in the form of predictive feedback; (3) Recurrent dynamics in the form of a linear-nonlinear map. Fig. 3 summarizes the impact of each of these scenarios on a nonlinear population readout, and we discuss each in depth in the following subsections.

Population competition in unsupervised Hebbian learning

In Fig. 2c, we saw that Hebbian homeostasis improved stability in the short term. Eq. (5) acts as an unsupervised learning rule, and pulls the readout $y(\theta)$ towards a family of bump-like tuning curves that tile θ (33). Under these dynamics, only drift $\Delta\mathbf{x}(\theta)$ that changes the peak of $y(\theta)$ to some new, nearby θ_0' can persist. All other modes of drift are rejected. If the encoding population is much larger than the dimension of θ, there is large null space in which drift does not change the preferred tuning. However, in the long run Hebbian homeostasis drives the neural population toward a steady-state which forgets the initial tuning (Fig. 3c). This is because Hebbian learning is biased towards a few salient θ_0 that capture directions in $\mathbf{x}(\theta)$ with the greatest variability (28, 46, 47).

Models of unsupervised Hebbian learning address this by introducing competition among a population of readout neurons (46, 47). Such rules can track the full covariance structure of the encoding population, and lead to a readout population of bump-like tuning curves that tile the space θ (48–51). In line with this, we incorporate response normalization into a readout population (52). This serves as a fast-acting form of firing-rate homeostasis in Eq. (3), causing neurons to compete to remain active and encouraging diverse tunings (50, 53).

Because it is implemented via inhibitory circuit dynamics, we assume that this normalization acts quickly relative to plasticity, and model it by dividing the rates by the average firing rate across the population. If $\mathbf{y}_f(\theta)$ is the forward (unnormalized) readout from Eq. (4), we define the normalized readout $\mathbf{y}_n(\theta)$ by dividing out the average population rate, $\langle \mathbf{y}_f(\theta)\rangle_{\mathrm{M}}$, and multiplying by a target mean rate μ_p:

$$\mathbf{y}_n(\theta) = \mu_p \cdot \mathbf{y}_f(\theta)/\langle \mathbf{y}_f(\theta)\rangle_{\mathrm{M}}. \tag{6}$$

Response normalization improves stability (Fig. 3d). However, it does not constrain individual readout neurons to any specific preferred θ_0. The readout remains sensitive to noise and perturbations, which can cause neurons to swap preferred tunings (Fig. 3d; Methods: *Population simulations*).

Error-correcting recurrent dynamics

The error-correction mechanisms explored so far use redundancy and feedback to reduce errors caused by incremental drift. However, there is no communication between different readouts $y_i(\theta)$ to ensure that the correlation structure of the readout population is preserved. Could stability be improved by using population dynamics to maintain these correlations, and actively correct decoding mistakes caused by drift?

Neural populations can compute prediction errors based on learned internal models (54), and experiments find that neural population activity recapitulates (55) and predicts (56) input statistics. Theories of distributed learning predict that local populations should coordinate by exchanging prediction errors (57), and models of efficient coding posit that neural populations cancel the predictable components of their input via negative feedback (58, 59). We propose that these same error signals could provide error-correction to improve the stability of neural population codes in the presence of drift.

To explore this, we consider two kinds of recurrent dynamics. We keep these models abstract. In particular, we do not specify a physiological means for how the structure of the fixed recurrent connectivity emerges. Our results thus serve as hypotheses for how recurrent dynamics could interact synergistically with the simpler mechanisms we have examined so far. We first consider a network that uses inhibitory feedback to cancel the predictable aspects of its input. This is in line with models of predictive coding (58–60). We then consider a linear-nonlinear mapping that provides a prediction of $\mathbf{y}(\theta)$ from a partially corrupted readout, using this signal to retrain readout weights.

Recurrent feedback of prediction errors

Some theories propose that neural populations retain a latent state that is used to predict future inputs (58–60). Inhibitory interneurons compare this prediction to incoming information to generate a prediction error, which is fed back through recurrent interactions to update the latent state. Here, we assume that the network contains a latent state "$\mathbf{z}$" and predicts error-corrected estimates of the readout's activity, $\hat{\mathbf{y}} = \phi(\mathbf{z})$, with ϕ as defined previously. Inputs provide a feed-forward estimate $\mathbf{y}_f$, which is corrupted by drift. The prediction error is the difference between $\mathbf{y}_f$ and $\hat{\mathbf{y}}$. The dynamics of $\mathbf{z}$ are chosen as:

$$\tau_z \dot{\mathbf{z}} = -\mathbf{z} + \mathbf{A}_p[\mathbf{y}_f - \hat{\mathbf{y}}]. \tag{7}$$

We set the weight matrix $\mathbf{A}_p$ to the covariance of the activations $\mathbf{z} = \mathbf{W}^\top \mathbf{x}$ during initial training (motivation for this choice is in Supplement: *Predictive coding as inference*). In making this choice, we assume that part of the circuit can learn and retain the covariance of $\mathbf{z}$. This could in principle be achieved via Hebbian learning (45, 46, 61).

Assuming that a circuit can realise the dynamics in Eq. (7), the readout $\hat{\mathbf{y}}$ will be driven to match the forward predictions $\mathbf{y}_f$. We assume that this converges rapidly relative to the timescale at which $\mathbf{y}_f(\theta)$ varies. This improves the tracking of a drifting population code when combined with Hebbian homeostasis and response normalization (Fig. 3e). The readout continuously re-aligns its fixed internal model with the activity in the encoding population. We briefly discuss intuition behind why one should generally expect this to work.

The recurrent weights, $\mathbf{A}_p$, determine which directions in population-activity space receive stronger feedback. Feedback

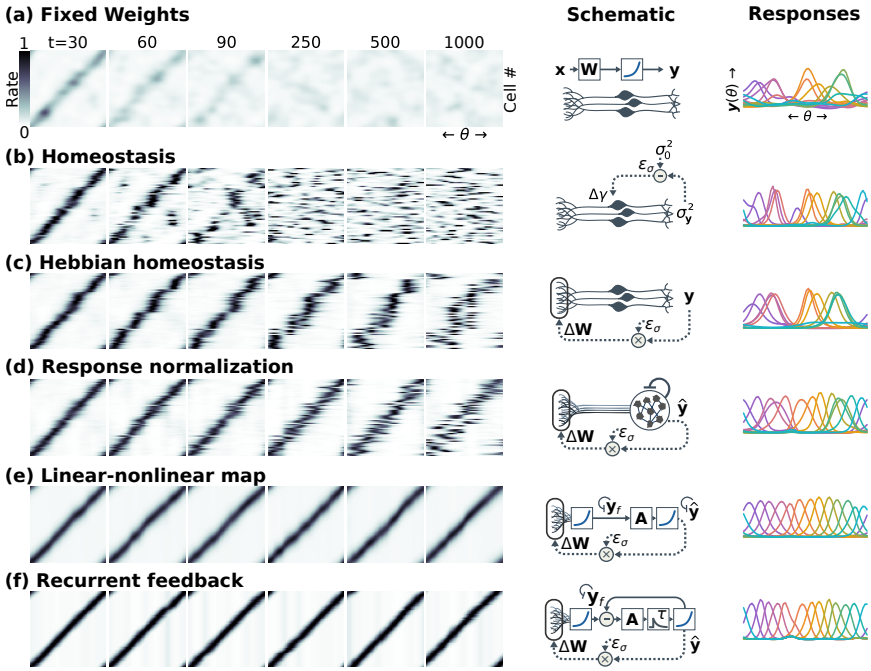

Figure 3: ***Self-healing in a nonlinear rate network.*** Each plot shows **(left)** a population readout $\mathbf{y}(\theta)$ from a drifting code $\mathbf{x}(\theta)$ of N=100 cells; **(middle)** a schematic of the readout dynamics; and **(right)** a plot of readout tuning after applying each learning rule if 60% of the encoding cells were to change to a new tuning (Methods: *Population simulations*). **(a)** For fixed weights, drift destroys the correlated activity readout degrades with increasing drift. Drift is gradual, with τ=100. The simulated time frame corresponds to ≈10 complete reconfigurations. **(b)** Homeostasis increases the sensitivity to compensate for loss of drive, but cannot stabilize tuning (σ^2: firing-rate variance, σ_0^2: target variance, ϵ_σ: homeostatic error, $\Delta\gamma$: gain adjustment). **(c)** Hebbian homeostasis (Eq. 5) restores drive using the readout's output as a learning signal. Error correction is imperfect and Hebbian learning biases tuning toward θ that drive more variability in the encoding population. ($\Delta\mathbf{W}$: weight updates) **(d)** Response normalization (Eq. 7) stabilizes the population statistics, but readout neurons can swap preferred tunings. ($\hat{\mathbf{y}}$: normalized response). **(e)** An additional linear-nonlinear map (Eq. 8) pools information over the population, providing better error correction ($\mathbf{y}_f$: feed-forward estimates, $\mathbf{A}$: recurrent weights). **(f)** Predictive coding (Eq. 6) uses a learned internal model of $\mathbf{y}(\theta)$ to correct errors caused by drift (τ indicates dynamics in time).

through larger eigenmodes of $\mathbf{A}_p$ is amplified, and these modes are more rapidly driven to track $\mathbf{y}_f$. Due to the choice of $\mathbf{A}_p$ as the covariance of $\mathbf{z}$, the dominant modes reflect directions in population activity that encode θ. Conversely, minor eigenmodes are weakly influenced by $\mathbf{y}_f$, and ignored by the readout. This removes directions in population activity that are unrelated to θ, thereby correcting errors in the readout activity caused by drift.

In summary, Eq. (7) captures qualitative dynamics implied by some theories of predictive coding. If neural populations update internal states based on prediction errors, then only errors related to tracking variations in θ should be tracked aggressively. This causes the readout to ignore "off manifold" activity in $\hat{\mathbf{y}}(\theta)$ caused by drift. However, other models of recurrent dynamics also work, as we explore next.

Low-dimensional manifold dynamics

Recurrent dynamics with a manifold of stable (or nearly stable) solutions distributed over θ could also support error correction. We model this by training the readout to make a prediction $\hat{\mathbf{y}}$ of its own activity based on the feed-forward activity $\mathbf{y}_f$, via a linear-nonlinear map, (c.f. 62):

$$\hat{\mathbf{y}}_{t+1} \leftarrow \phi[\mathbf{A}_r^\top \mathbf{y}_{f,t} + \mathbf{v}], \tag{8}$$

with timestep, t, and recurrent weights and biases $\mathbf{A}_r$ and $\mathbf{v}$ (Methods: *Learning recurrent weights*). We chose this discrete mapping for computational expediency, and Eq. (8) was applied once for each input $y_f(\theta)$ alongside response normalization. In simulations, the recurrent mapping is also effective at correcting errors caused by drift, improving readout stability (Fig. 3f).

We briefly address some caveats that apply to both models of recurrent dynamics. The combination of recurrent dynamics and Hebbian learning is potentially destabilizing, because leaning can transfer biased predictions into the decoding weights. Empirically, we find that homeostasis (Eq. 3) prevents this, but must strong enough to counteract all destabilizing influences. Additionally, when the underlying θ has continuous symmetries, drift can occur along these symmetries. This is evidenced by a gradual, diffusive rotation of the code for e.g. a circular environment. Other manifolds, like the T-shaped maze in (2), have no continuous symmetries and are not susceptible to this effect (Supplemental Figure S4). Overall, these simulations illustrate that internal models can constrain network activity. This provides ongoing error correction, preserves neuronal correlations, and allows neural populations to tolerate substantial reconfiguration of the inputs that drive them.

Discussion

In this work, we outlined homeostatic principles that could allow stable and plastic representations to coexist in the brain. We argue that self-healing codes should have three components: (I) Neuronal responses should be robust to small amounts of drift; (II) Neurons should use their own output as a training signal to update their decoding weights, and (III) Stable codes should homeostatically preserve internal models, which are reflected in stable population statistics.

Hebbian plasticity is synonymous with learning novel associations in much of contemporary neuroscience. Our findings offer the complementary hypothesis that Hebbian mechanisms can also reinforce learned associations in the face of ongoing change. This view is compatible with the observation that Hebbian plasticity is a positive feedback process, where existing correlations become strengthened, in turn promoting correlated activity. At an abstract level, positive feedback is a key ingredient of any memory retention mechanism because it rejects external disturbances by reinforcing existing states.

Homeostasis, by contrast, is typically seen as antidote to possible runaway Hebbian plasticity (63). However, this idea is problematic due to the relatively slow timescale at which homeostasis acts (25). Our findings posit a richer role for homeostatic (negative) feedback in maintaining and distributing responsiveness in a population. This is achieved by regulating the mean and the variance of neural activity (23).

We considered two populations, a drifting population that encodes a variable, and another that extracts a drift-resilient

readout. This could reflect communication between stable and plastic components of the brain, or the interaction between stable and plastic neurons within the same circuit. This is consistent with experiments that find consolidated stable representations (12, 16), or with the view that neural populations contain a mixture of stable and unstable cells (64).

By itself, Hebbian homeostasis preserves population codes in the face of drift over a much longer timescale than the lifetime of a code with fixed readout (Fig. 2). Even though this mechanism ultimately corrupts a learned tuning, the time horizon over which the code is preserved may be adequate in a biological setting, particularly in situations where there are intermittent opportunities to reinforce associations behaviourally. In the absence of external feedback, extending the lifetime of this code still further required additional assumptions about circuit structures that remain to be tested experimentally.

We found that a readout population can use an internal model to maintain a consistent interpretation of an unstable encoding population. Such internal models are widely hypothesized to exist in various guises (54, 59, 60). We therefore did not address how these internal models are learned initially, or how they might be updated. We conjecture that the error signals used in the internal models are the same ones that would be used for initial (or ongoing) learning. Since changes in weights are driven by correlations between presynaptic and postsynaptic activity, modulating the relative influence of internal dynamics and external inputs could regulate the balance between homeostatic repair and new learning. By setting fixed recurrent weights, we are also assuming that some connectivity in the circuit is unaffected by drift. This may be reasonable, given that functional connectivity in some circuits is found to be stable (13).

The recurrent architectures we studied here are reminiscent of mechanisms that attenuate forgetting via replay (e.g. 65, 66). The internal models must be occasionally re-activated through rehearsal or replay to detect and correct inconsistencies caused by drift. If this process occurs infrequently, drift becomes large, and the error correction will fail.

The brain supports both stable and volatile representations, typically associated with memory retention and learning, respectively. Artificial neural networks have so far failed to imitate this, and suffer from catastrophic forgetting wherein new learning erases previously learned representation (67). Broadly, proposed strategies to mitigate this segregate stable and unstable representations into distinct subspaces of the possible synaptic weight changes (c.f. 18). These learning rules therefore prevent disruptive drift in the first place. The mechanisms explored here do not restrict changes in weights or activity: the encoding population is free to reconfigure arbitrarily. However, any change in a neural code leads to an equal and opposite change in how that code is interpreted—The brain must publish new translations of its changing internal language. This preserves the functional relationships between neurons.

To integrate stable and plastic representations, changes anywhere in the brain should be accompanied by compensatory changes throughout the brain. The learning rules explored here emphasized Hebbian homeostasis and the role of predictive models in generating robust representations. In the long term, these processes could support widespread reallocation or reconsolidation of neuronal function. Further exploration of these principles may clarify how the brain can be simultaneously plastic and stable, and provide clues to how to build artificial networks that share these properties.

Materials and Methods

Data and analysis Data shown in Fig. 1b,c were taken from Driscoll et al. (2), and are available online at at Dryad (43). Examples of tuning curve drift were taken from mouse four, which tracked a sub-population of cells for over a month using calcium fluorescence imaging. Normalized log-fluorescence signals ($\ln[x/\langle x\rangle]$) were filtered between 0.3 and 3 Hz (4th Butterworth, forward-backward filtering), and individual trial runs through the T maze were extracted. Traces from select cells were aligned based on task pseudotime (0: start, 1: reward). On each day, log-fluorescence was averaged over all trials and exponentiated to generate the average tuning curves shown in Fig. 1b. For Fig. 1c, a random sub-population of forty cells was sorted based on their peak firing location on the first day. For further details, see (2, 19).

Simulated drift We modeled drift as a discrete-time Ornstein-Uhlenbeck (OU) random walk on encoding weights $\mathbf{U}$, with time constant τ (in days) and per-day noise variance α. We set the noise variance to α=$2/\tau$ to achieve unit steady-state variance. Encoding weights for each day are sampled as:

$$u_{d+1,i,j} = u_{d,i,j}\sqrt{1-\alpha} + \xi\sqrt{\alpha}, \qquad \xi \sim \mathcal{N}(0,1). \tag{9}$$

These drifting weights propagate the information about θ available in the features $\mathbf{s}(\theta)$ (Eq. 1) to the encoding units $\mathbf{x}(\theta)$, in a way that changes randomly over time.

This random walk in encoding-weight space preserves the population code statistics on average: It preserves the geometry of θ in the correlations of $\mathbf{a}_t(\theta)$, and the average amount of information about θ encoded in the population activations (Supplement: *Stability of encoded information*). This implies that the difficulty of reading out a given tuning curve $y(\theta)$ (in terms of the L2 norm of the decoding weights, $\|\mathbf{w}_j\|^2$) should remain roughly constant over time. This assumption, that $\mathbf{x}(\theta)$ encodes a stable representation for θ in an unstable way, underlies much of the robustness we observe. We discuss this further in Methods: *Synaptic learning rules*.

Because the marginal distribution of the encoding weights on each day is Gaussian, $\mathbf{U}_d \sim \mathcal{N}(0, I_N)$, the synaptic activations $\mathbf{a}_d(\theta) = \mathbf{U}_d^\top \mathbf{s}(\theta)$ are samples from a Gaussian process on θ, with covariance inherited from $\mathbf{s}(\theta)$ (Supplement: *Gaussian-process tuning curves*). In numerical experiments, we sampled the synaptic activation functions $\mathbf{a}_d(\theta)$ from this Gaussian process directly. We simulated $\theta \in [0,1)$ over a discrete grid with 60 bins, sampling synaptic activations from a zero-mean Gaussian process on θ with a spatially-low-pass squared-exponential kernel ($\sigma = 0.1$). The gain and threshold (Eq. 2) for each encoding unit was homeostatically adjusted for a target mean-rate of $\mu_0 = 5$ and rate variance of $\sigma_0^2 = 25$ (in arbitrary units). This was achieved by running Eq. (3) for 50 iterations with rates $\eta_\gamma = 0.1$, $\eta_\beta = 0.2$ for the gain and bias homeostasis, respectively.

To show that the readout can track drift despite complete reconfiguration of the neural code, we replace gradual drift in all features with abrupt changes in single features in Fig. 2. For this, we re-sampled the weights for single encoding units one-at-a-time from a standard normal distribution. Self-healing plasticity rules were run each time 5 out of the 100 encoding features changed. Supplemental Fig. S1 confirms that abrupt drift in a few units is equivalent to gradual drift in all units. Unless otherwise stated, all other results are based on an OU model of encoding drift.

We modeled excess variability in the encoding population that was unrelated to cumulative drift. This scenario resembles to the drift observed in vivo (9; Supplemental Fig. S3). We sampled a unique "per-day" synaptic activation $\tilde{a}_{d,i}(\theta)$ for each of the encoding units, from the same Gaussian process on θ used to generate the drifting activation functions $a_{d,i}(\theta)$. We mixed these two functions with a parameter

$r = 0.05$ such that the encoding variability was preserved (i.e. 5% of the variance in synaptic activation is related to random variability):

$$a'_{d,i}(\theta) = a_{d,i}(\theta)\sqrt{1-r} + \tilde{a}_{d,i}(\theta)\sqrt{r}. \tag{10}$$

Supplemental Fig. S2a shows that the readout can tolerate up to 30% excess variability with modest loss of stability. Supplemental Fig. S3 shows that neuronal recordings from Driscoll et al. (43) are consistent with 30% excess variability, and that the qualitative conclusions of this paper hold for this larger amount of day-to-day variability (Supplement: *Calibrating the model to data*).

We also applied drift on the decoding synapses $\mathbf{W}$. This is modeled similarly to Eq. (10), with the parameter n controlling the percentage of variance in synapse weight that changes randomly at the start of each each day:

$$w'_{d,i,j} = w_{d,i,j}\sqrt{1-n} + \sigma_d \cdot \xi\sqrt{n} \qquad \xi \sim \mathcal{N}(0,1). \tag{11}$$

where σ_d is the empirical standard-deviation of the the decoding weights on day d. Unless otherwise stated, we use $n = 1\%$. Larger values of drift on the decoding weights is destabilizing for Hebbian homeostasis (with or without response normalization), but readouts with stable internal recurrent dynamics can tolerate larger ($\sim 8\%$) amounts of readout-weight drift (Supplemental Fig. S2b).

Synaptic learning rules The learning rule in Eq. (5) is classical unsupervised Hebbian learning, which is broadly believed to be biologically plausible (45, 46, 61). However, it has one idiosyncrasy that should be justified: The rates of learning and weight decay are modulated by a homeostatic error in firing-rate variability. The simplest interpretation of Eq. (5) is a premise or ansatz: learning rates should be modulated by homeostatic errors. This is a prediction that will need to be experimentally confirmed. Such a learning might be generically useful, since it pauses learning when firing-rate statistics achieve a useful dynamic range for encoding information. The fact that weight decay is proportional to learning rate is also biologically plausible, since each cells has finite resources to maintain synapses.

Eq. (5) may also emerge naturally from the interaction between homeostasis and learning rules in certain scenarios. When Hebbian learning is interpreted as a supervised learning rule, it is assumed that other inputs bias the spiking activity y of a neuron toward a target y^*. This alters the correlations between presynaptic inputs $\mathbf{x}$ and postsynaptic spiking. Hebbian learning rules, especially temporally asymmetric ones based on spike timing (68), adjust readout weights $\mathbf{w}$ to potentiate inputs that correlate with this target. In the absence of external learning signals, homeostatic regulation implies a surrogate training signal $\tilde{y}^*$. This $\tilde{y}^*$ is biased toward a target mean-rate an selectivity. For example, recurrent inhibition could regulate both population firing rate and population-code sparsity. This could restrict postsynaptic spiking, causing Hebbian learning to adjust readout weights to achieve the desired statistics. Cells may also adjust their sensitivity and threshold homeostatically. Hebbian learning could then act to adjust incoming synaptic weights to achieve the target firing-rate statistics, but in a way that is more strongly correlated with synaptic inputs.

In Supplement: *Hebbian homeostasis as an emergent property*, we verify the intuition that Eq. (5) should arise through emergent interactions between homeostasis and Hebbian learning in a simplified, linear model. In the remainder of this section, we us a linear readout to illustrate why one should expect Eq. (5) to be stabilizing.

The decoder's job is to generate a stable readout from the drifting code $\mathbf{x}(\theta)$. This is a regression problem: the decoding weights $\mathbf{W}$ should map $\mathbf{x}(\theta)$ to a target $\mathbf{y}(\theta)$. Since small amounts of drift are akin to noise, $\mathbf{W}$ should be regularized to improve robustness. The L2-regularized linear least-squares solution for $\mathbf{W}$ is:

$$\mathbf{W} = [\Sigma_x + \rho_2\mathbf{I}]^{-1}\Sigma_{xy}. \tag{12}$$

The regularization $\rho_2\mathbf{I}$ corresponds to the assumption that drift will corrupt the activity of $\mathbf{x}(\theta)$ by an amount $\Delta\mathbf{x} \sim \mathcal{N}(0, \rho_2\mathbf{I})$.

Can drift be tracked by re-inferring $\mathbf{W}$ on each day? We lack the ground-truth covariance Σ^d_{xy} to re-train $\mathbf{W}$, but could estimate it from decoded activity $\mathbf{y}_d(\theta)$:

$$\hat{\Sigma}^d_{xy} = \langle \mathbf{x}_d(\theta)\mathbf{y}_d(\theta)^\top\rangle_\theta \tag{13}$$

Since $\mathbf{y}_d(\theta)$ is decoded from $\mathbf{x}_d(\theta)$ through weights $\mathbf{W}_d$, the estimated covariance is $\hat{\Sigma}^d_{xy} = \Sigma^d_x\mathbf{W}_d$, where $\Sigma^d_x = \langle \mathbf{x}_d(\theta)\mathbf{x}_d(\theta)^\top\rangle_\theta$ is the covariance of the inputs $\mathbf{x}_d(\theta)$. The regularized lead-squares weight update is therefore:

$$\mathbf{W}_{d+1} = [\Sigma^d_x + \rho_2\mathbf{I}]^{-1}\Sigma^d_x\mathbf{W}_d. \tag{14}$$

This update can be interpreted as recursive Bayesian filtering of the weights (Supplement: *Weight filtering*).

Because $\mathbf{x}(\theta)$ continues to encode information about θ, we know that variability in the decoded $\mathbf{y}(\theta)$ should be conserved. Each readout $y_i(\theta)$ homeostatically adjusts its sensitivity to maintain a target variability σ_0^2. As introduced earlier, this multiplies the firing rate by a factor $\gamma = \sigma_0/\sigma_1 = 1 + \delta$, where δ is a small parameter. and σ_1 is the standard deviation of the readout's firing rate after drift but before normalization. Accounting for this in Eq. (14) and considering the weight vector $\mathbf{w}$ for a single readout neuron yields:

$$\mathbf{w}_{d+1} = [\Sigma^d_x + \rho_2\mathbf{I}]^{-1}\Sigma^d_x\mathbf{w} \cdot (1+\delta). \tag{15}$$

To translate this into a plausible learning rule, the solution Eq. (15) can be obtained via gradient descent. Recall the loss function $\mathcal{L}(\mathbf{w})$ for optimizing regularized linear least-squares:

$$\mathcal{L}(\mathbf{w}) = \tfrac{1}{2}\langle\|\mathbf{w}^\top\mathbf{x} - \mathbf{y}\|^2\rangle + \tfrac{1}{2}\rho_2\|\mathbf{w}\|^2. \tag{16}$$

Gradient descent $-\nabla_\mathbf{w}\mathcal{L}(\mathbf{w})$ on Eq. (16) implies the weight update

$$\Delta\mathbf{w} \propto \langle \mathbf{x}(\mathbf{y} - \mathbf{w}^\top\mathbf{x})^\top\rangle - \rho_2\mathbf{w}. \tag{17}$$

After matching terms between Eqs. (13-15) and Eq. (17) and simplifying, one finds the following Hebbian learning rule:

$$\Delta\mathbf{w} \propto \delta\langle \mathbf{x}_d(\theta)y_d(\theta)\rangle_\theta - \rho_2\mathbf{w}. \tag{18}$$

Eq. (18) is equivalent to Eq. (5) for a certain regularization strength ρ_2 (now taking the form of weight decay). The optimal value of ρ_2 depends on the rate of drift. Since drift drives homeostatic errors, it follows that $\rho_2 \propto \delta$ for small δ. Here, we set $\rho_2 = \delta$, corresponding to $c = 1$ in Eq. (18).

Single-neuron readout In Fig. 2, we simulated a population of 100 encoding neurons $\mathbf{x}_d(\theta)$ that changed one at a time (Methods: *Simulated drift*). We initialized a single readout $y(\theta) = \phi[\mathbf{w}^\top\mathbf{x}(\theta)]$ to decode a Gaussian bump $y_0(\theta)$ ($\sigma = 5\%$ of the track length) from the activations $\mathbf{x}_0(\theta)$ on the first day. We optimized this via gradient descent using a linear-nonlinear Poisson loss function.

$$\Delta\mathbf{w} \propto \left\langle \mathbf{x}_0(\theta)[\mathbf{y}_0(\theta) - y(\theta)]^\top\right\rangle_\theta - \rho_r\mathbf{w}, \tag{19}$$

with regularizing weight decay $\rho_r = 10^{-4}$. In this deterministic firing-rate model, the Poisson error allows the squared-norm of the residuals to be proportional to the rate. We simulated 200 time points of drift, corresponding to two complete reconfigurations of the encoding population. After each encoding-unit change, we applied 100 iterations of either naïve homeostasis (Fig. 2b; Eq. 3) or Hebbian homeostasis (Fig. 2c; Eq. 5). For naïve homeostasis, the rates for gain and threshold homeostasis were $\eta_\beta = 10^{-3}$ and $\eta_\gamma = 10^{-5}$, respectively. For Hebbian homeostasis, the rates were $\eta_\beta = 10^{-1}$ and $\eta_\gamma = 10^{-3}$.

Homeostatic regulation requires averaging the statistics over time (44). To model this, we calculated the parameter updates for the gain and bias after replaying all θ and computing the mean and variance of the activity for each neuron. Since the processes underlying cumulative changes in synaptic strength are also slower than the timescale of neural activity, weight updates were averaged over all θ on each iteration. We applied additional weight decay with a rate $\rho = 1 \times 10^{-4}$ for regularization and stability, and set c=1 in Eq. (5) such that the rate of weight decay was also modulated by the online variability error $\tilde{\delta}$.

Learning recurrent weights For recurrent dynamics modeled as feedback in Eq. (7), supervised, linear Hebbian learning implies that the recurrent weights should be proportional to the covariance of the state variables $\mathbf{z}$. To see this, consider a linear Hebbian learning rule, where $\mathbf{z}$ has been entrained by an external signal, and serves as both the presynaptic input and postsynaptic output:

$$\frac{d}{dt}\mathbf{A}_p = \langle \mathbf{z}\mathbf{z}^\top \rangle_\theta - \alpha \mathbf{A}_p, \tag{20}$$

where α is a weight decay term. This has a fixed point at $\mathbf{A}_p = \langle \mathbf{z}\mathbf{z}^\top \rangle / \alpha$. In our simulations, we ensure that $\mathbf{z}$ is zero-mean such that the second moment, $\langle \mathbf{z}\mathbf{z}^\top \rangle$, is equal to the covariance.

For the linear-nonlinear map model of recurrent dynamics Eq. (8), neurons could learn $\mathbf{A}_r$ by comparing a target $\mathbf{y}_0$ to the predicted $\mathbf{y}_r$ at the same time that the initial decoding weights $\mathbf{W}_0$ are learned. For example, $\mathbf{y}_0$ could be an external (supervised) signal or the forward predictions in Eq. (4) before drift occurs, and $\mathbf{y}_r$ could arise though recurrent activity in response to $\mathbf{y}_0$. A temporally-asymmetric plasticity rule could correlate the error between these signals with the recurrent synaptic inputs to learn $\mathbf{A}_r$ (68). This plasticity rule should update weights in proportion to the correlations between inputs $\mathbf{y}_f$ and a prediction error $\mathbf{y}_0 - \mathbf{y}_r$:

$$\Delta \mathbf{A}_r \propto \left\langle \mathbf{y}_f (\mathbf{y}_0 - \mathbf{y}_r)^\top \right\rangle_\theta - \rho_r \mathbf{A}_r, \tag{21}$$

where $\rho_r = 10^{-4}$ sets the amount of regularizing weight decay.

Eq. (8) is abstract, but captures the two core features of error correction through recurrent dynamics. It describes a population of readout neurons that predict each-other's activity through recurrent weights. Eq. (21) states that these weights are adapted during initial learning to minimize the error in this prediction. We assume $\mathbf{A}_r$ is fixed once learned.

Population simulations In Fig. 3, we simulated an encoding population of 100 units. Drift was simulated as described in Methods: *Simulated drift*, with $\tau = 100$. In all scenarios, we simulated $M = 60$ readout cells tiling a circular θ divided into $L = 60$ discrete bins. Learning and/or homeostasis was applied every 5 iterations of simulated drift. The readout weights and tuning curves were initialized similarly to the single-neuron case, but with tuning curves tiling θ.

For the predictive coding simulations (Eq. 7), we simulated a second inner loop to allow the network activity $\mathbf{z}$ to reach a steady state for each input $\mathbf{x}(\theta)$. This loop ran for 100 iterations, with time constant of $\tau_z = 100$. The recurrent weights $\mathbf{A}_p$ were initialized as the covariance of the synaptic activations on the first day (Σ_z where $\mathbf{z}(\theta) = \mathbf{W}^\top \mathbf{x}(\theta)$) and held fixed over time. The final value $\hat{\mathbf{z}}$ was used to generate a training signal, $\hat{\mathbf{y}} = \phi(\hat{\mathbf{z}})$, to update the readout weights. For the recurrent map, recurrent weights were learned initially using Eq. (21) and held fixed through the simulations.

For both the linear-nonlinear map and the recurrent feedback models, weights were updated as in Eq. (5), where the output of the recurrent dynamics was used to compute homeostatic errors and as the signal $\hat{\mathbf{y}}$ in Hebbian learning. For naïve homeostasis (Fig. 3b) and Hebbian homeostasis (with and without response normalization; Fig. 3cd), learning rates were the same as in the single-neuron simulations (Fig. 2; Methods; *Single-neuron readout*). For the linear-nonlinear map (Fig 3e), learning rates were set to $\eta_\gamma = 10^{-4}$ and $\eta_\beta = 10^{-1}$. For recurrent feedback (Fig 3f), the learning rates were $\eta_\gamma = 5 \times 10^{-3}$ and $\eta_\beta = 5$. Learning rates for all scenarios were optimized via grid search.

Response normalization was added on top of Hebbian homeostasis for Fig. 3d, and was also included in Fig. 3ef to ensure stability. The population rate target μ_p for response normalization was set to the average population activity in the initially trained state.

Different parameters were used to generate the right-hand column of Fig. 3, to show the effect of a larger amount of drift. After training the initial readout, 60% of the encoding features were changed to a new, random tuning. Rates were increased by 50× for naïve homeostasis to handle the larger transient adaptation needed for this larger change. The other methods did not require any adjustments in parameters. Each homeostatic or plasticity rule was then run to steady-state (1000 iterations).

Code availability Source code for all simulations is available online at github.com/michaelerule/selfhealingcodes.

Acknowledgements This work was supported by the European Research Council (Starting Grant FLEXNEURO 716643), the Human Frontier Science Program (RGY0069), the Leverhulme Trust (ECF-2020-352), and the Issac Newton Trust.

References

[1] AR Chambers, S Rumpel, A stable brain from unstable components: emerging concepts and implications for neural computation. *Neuroscience* **357**, 172–184 (2017).

[2] LN Driscoll, NL Pettit, M Minderer, SN Chettih, CD Harvey, Dynamic reorganization of neuronal activity patterns in parietal cortex. *Cell* **170**, 986–999 (2017).

[3] A Singh, A Peyrache, MD Humphries, Medial prefrontal cortex population activity is plastic irrespective of learning. *J. Neurosci.* **39**, 3470–3483 (2019).

[4] TD Marks, MJ Goard, Stimulus-dependent representational drift in primary visual cortex. *bioRxiv* (2020).

[5] D Deitch, A Rubin, Y Ziv, Representational drift in the mouse visual cortex. *Curr. Biol.* (2021).

[6] CE Schoonover, SN Ohashi, R Axel, AJ Fink, Representational drift in primary olfactory cortex. *bioRxiv* (2020).

[7] Y Ziv, et al., Long-term dynamics of ca1 hippocampal place codes. *Nat. neuroscience* **16**, 264 (2013).

[8] SJ Levy, NR Kinsky, W Mau, DW Sullivan, ME Hasselmo, Hippocampal spatial memory representations in mice are heterogeneously stable. *Hippocampus* **31**, 244–260 (2021).

[9] ME Rule, T O'Leary, CD Harvey, Causes and consequences of representational drift. *Curr. opinion neurobiology* **58**, 141–147 (2019).

[10] C Clopath, T Bonhoeffer, M Hübener, T Rose, Variance and invariance of neuronal long-term representations. *Philos. Transactions Royal Soc. B: Biol. Sci.* **372**, 20160161 (2017).

[11] RD Flint, MR Scheid, ZA Wright, SA Solla, MW Slutzky, Long-term stability of motor cortical activity: implications for brain machine interfaces and optimal feedback control. *J. neuroscience* **36**, 3623–3632 (2016).

[12] KA Katlowitz, MA Picardo, MA Long, Stable sequential activity underlying the maintenance of a precisely executed skilled behavior. *Neuron* **98**, 1133–1140 (2018).

[13] T Rose, J Jaepel, M Hübener, T Bonhoeffer, Cell-specific restoration of stimulus preference after monocular deprivation in the visual cortex. *Science* **352**, 1319–1322 (2016).

[14] M Fauth, F Wörgötter, C Tetzlaff, Formation and maintenance of robust long-term information storage in the presence of synaptic turnover. *PLoS computational biology* **11**, e1004684 (2015).

[15] JA Hennig, et al., Constraints on neural redundancy. *Elife* **7**, e36774 (2018).

[16] J Pérez-Ortega, T Alejandre-García, R Yuste, Long-term stability of cortical ensembles. *eLife* **10**, e64449 (2021).

[17] S Druckmann, DB Chklovskii, Neuronal circuits underlying persistent representations despite time varying activity. *Curr. Biol.* **22**, 2095–2103 (2012).

[18] L Duncker, L Driscoll, KV Shenoy, M Sahani, D Sussillo, Organizing recurrent network dynamics by task-computation to enable continual learning. *Adv. Neural Inf. Process. Syst.* **33** (2020).

[19] ME Rule, et al., Stable task information from an unstable neural population. *Elife* **9**, e51121 (2020).

[20] E Marder, JM Goaillard, Variability, compensation and homeostasis in neuron and network function. *Nat. Rev. Neurosci.* **7**, 563–574 (2006).

[21] T O'Leary, DJ Wyllie, Neuronal homeostasis: time for a change? *The J. physiology* **589**, 4811–4826 (2011).

[22] T O'Leary, MC van Rossum, DJ Wyllie, Homeostasis of intrinsic excitability in hippocampal neurones: dynamics and mechanism of the response to chronic depolarization. *The J. physiology* **588**, 157–170 (2010).

[23] J Cannon, P Miller, Stable control of firing rate mean and variance by dual homeostatic mechanisms. *The J. Math. Neurosci.* **7**, 1–38 (2017).

[24] GG Turrigiano, The self-tuning neuron: synaptic scaling of excitatory synapses. *Cell* **135**, 422–435 (2008).

[25] F Zenke, W Gerstner, Hebbian plasticity requires compensatory processes on multiple timescales. *Philos. Transactions Royal Soc. B: Biol. Sci.* **372**, 20160259 (2017).

[26] YK Wu, KB Hengen, GG Turrigiano, J Gjorgjieva, Homeostatic mechanisms regulate distinct aspects of cortical circuit dynamics. *Proc. Natl. Acad. Sci.* **117**, 24514–24525 (2020).

[27] LF Abbott, SB Nelson, Synaptic plasticity: taming the beast. *Nat. neuroscience* **3**, 1178–1183 (2000).

[28] KD Miller, DJ MacKay, The role of constraints in hebbian learning. *Neural computation* **6**, 100–126 (1994).

[29] JA Gallego, et al., Cortical population activity within a preserved neural manifold underlies multiple motor behaviors. *Nat. Commun.* **9**, 4233 (2018).

[30] D Kappel, R Legenstein, S Habenschuss, M Hsieh, W Maass, A dynamic connectome supports the emergence of stable computational function of neural circuits through reward-based learning. *eNeuro* **5**, ENEURO–0301 (2018).

[31] M Gillett, U Pereira, N Brunel, Characteristics of sequential activity in networks with temporally asymmetric hebbian learning. *Proc. Natl. Acad. Sci.* **117**, 29948–29958 (2020).

[32] DV Raman, T O'leary, Optimal synaptic dynamics for memory maintenance in the presence of noise. *BioRxiv* (2020).

[33] S Qin, et al., Coordinated drift of receptive fields during noisy representation learning. *bioRxiv* (2021).

[34] D Acker, S Paradis, P Miller, Stable memory and computation in randomly rewiring neural networks. *J. neurophysiology* (2019).

[35] L Susman, N Brenner, O Barak, Stable memory with unstable synapses. *Nat. communications* **10**, 1–9 (2019).

[36] MJ Fauth, MC van Rossum, Self-organized reactivation maintains and reinforces memories despite synaptic turnover. *ELife* **8**, e43717 (2019).

[37] FYK Kossio, S Goedeke, C Klos, RM Memmesheimer, Drifting assemblies for persistent memory. *bioRxiv* (2020).

[38] JA Gallego, MG Perich, LE Miller, SA Solla, Neural manifolds for the control of movement. *Neuron* **94**, 978–984 (2017).

[39] A Rubin, et al., Revealing neural correlates of behavior without behavioral measurements. *Nat. Commun.* **10**, 4745 (2019).

[40] JA Gallego, MG Perich, RH Chowdhury, SA Solla, LE Miller, Long-term stability of cortical population dynamics underlying consistent behavior. *Nat. neuroscience* **23**, 260–270 (2020).

[41] A Farshchian, et al., Adversarial domain adaptation for stable brain-machine interfaces. *arXiv preprint arXiv:1810.00045* (2018).

[42] E Sorrell, ME Rule, T O'Leary, Brain–machine interfaces: Closed-loop control in an adaptive system. *Annu. Rev. Control. Robotics, Auton. Syst.* **4** (2021).

[43] LN Driscoll, NL Pettit, M Minderer, SN Chettih, CD Harvey, Data from: Dynamic reorganization of neuronal activity patterns in parietal cortex dataset. *Dryad* **(Dataset) https://doi.org/10.5061/dryad.gqnk98sjq** (2020).

[44] P Miller, J Cannon, Combined mechanisms of neural firing rate homeostasis. *Biol. cybernetics* **113**, 47–59 (2019).

[45] E Oja, Simplified neuron model as a principal component analyzer. *J. mathematical biology* **15**, 267–273 (1982).

[46] P Földiák, P Fdilr, Adaptive network for optimal linear feature extraction in *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks*. (IEEE Press, Washington D.C.), Vol. 1, pp. 401–405 (1989).

[47] P Földiák, Forming sparse representations by local anti-hebbian learning. *Biol. cybernetics* **64**, 165–170 (1990).

[48] C Pehlevan, S Mohan, DB Chklovskii, Blind nonnegative source separation using biological neural networks. *Neural computation* **29**, 2925–2954 (2017).

[49] C Pehlevan, AM Sengupta, DB Chklovskii, Why do similarity matching objectives lead to hebbian/anti-hebbian networks? *Neural computation* **30**, 84–124 (2018).

[50] A Sengupta, C Pehlevan, M Tepper, A Genkin, D Chklovskii, Manifold-tiling localized receptive fields are optimal in similarity-preserving neural networks in *Advances in Neural Information Processing Systems*. pp. 7080–7090 (2018).

[51] A Giovannucci, V Minden, C Pehlevan, DB Chklovskii, Efficient principal subspace projection of streaming data through fast similarity matching in *2018 IEEE International Conference on Big Data (Big Data)*. (IEEE), pp. 1015–1022 (2018).

[52] M Carandini, DJ Heeger, Normalization as a canonical neural computation. *Nat. Rev. Neurosci.* **13**, 51–62 (2012).

[53] D Krotov, JJ Hopfield, Unsupervised learning by competing hidden units. *Proc. Natl. Acad. Sci.* **116**, 7723–7731 (2019).

[54] GB Keller, TD Mrsic-Flogel, Predictive processing: a canonical cortical computation. *Neuron* **100**, 424–435 (2018).

[55] M Wilf, et al., Spontaneously emerging patterns in human visual cortex reflect responses to naturalistic sensory stimuli. *Cereb. cortex* **27**, 750–763 (2017).

[56] SE Palmer, O Marre, MJ Berry, W Bialek, Predictive information in a sensory population. *Proc. Natl. Acad. Sci.* **112**, 6908–6913 (2015).

[57] JC Whittington, R Bogacz, An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation* **29**, 1229–1262 (2017).

[58] S Denève, CK Machens, Efficient codes and balanced networks. *Nat. neuroscience* **19**, 375–382 (2016).

[59] M Boerlin, CK Machens, S Denève, Predictive coding of dynamical variables in balanced spiking networks. *PLoS computational biology* **9**, e1003258 (2013).

[60] S Recanatesi, et al., Predictive learning as a network mechanism for extracting low-dimensional latent space representations. *Nat. communications* **12**, 1–13 (2021).

[61] TD Sanger, Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks* **2**, 459–473 (1989).

[62] R Darshan, A Rivkind, Learning to represent continuous variables in heterogeneous neural networks. *bioRxiv* (2021).

[63] T Keck, et al., Integrating hebbian and homeostatic plasticity: the current state of the field and future research directions. *Philos. Transactions Royal Soc. B: Biol. Sci.* **372**, 20160158 (2017).

[64] T Hainmueller, M Bartos, Parallel emergence of stable and dynamic memory engrams in the hippocampus. *Nature* **558**, 292–296 (2018).

[65] S Káli, P Dayan, Off-line replay maintains declarative memories in a model of hippocampal-neocortical interactions. *Nat. neuroscience* **7**, 286 (2004).

[66] OC González, Y Sokolov, GP Krishnan, JE Delanois, M Bazhenov, Can sleep protect memories from catastrophic forgetting? *Elife* **9**, e51005 (2020).

[67] RM French, Catastrophic forgetting in connectionist networks. *Trends cognitive sciences* **3**, 128–135 (1999).

[68] J Brea, W Senn, JP Pfister, Matching recall and storage in sequence learning with spiking neural networks. *J. neuroscience* **33**, 9565–9575 (2013).

[69] J Cannon, P Miller, Synaptic and intrinsic homeostasis cooperate to optimize single neuron response properties and tune integrator circuits. *J. neurophysiology* **116**, 2004–2022 (2016).
[70] F Pedregosa, et al., Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

Supplemental Information

Gaussian-process tuning curves Equation (10) in the main text (Methods: *Simulated drift*) defines an Ornstein-Uhlenbeck (OU) random walk on the encoding weights $\mathbf{U}$. This is equivalent to assuming that the synaptic activations of the encoding population undergo a random walk, and are samples from a stationary distribution of functions on θ.

Assume that individual encoding weights u vary randomly over time, sampling from a stationary distribution that can be approximated as Gaussian. Without loss of generality, choose units such that this distribution is a standard normal distribution $u \sim \mathcal{N}(0,1)$. Denote the time-varying vector of synaptic weights for a single encoding neuron as $\mathbf{u}_d$. Now, consider the synaptic activation for an encoding neuron driven by features $\mathbf{s}$: $a_d(\theta) = \mathbf{u}_d^\top \mathbf{s}(\theta)$. If $\mathbf{u}_d \sim \mathcal{N}(0,\mathbf{I})$, then the second moment $\Sigma_a(\theta,\theta') = \langle a_d(\theta)\, a_d(\theta')\rangle_d$ is:

$$\begin{aligned}\langle a_d(\theta)\, a_d(\theta')\rangle_d &= \langle [\mathbf{s}(\theta)^\top \mathbf{u}_d][\mathbf{u}_d^\top \mathbf{s}(\theta')]\rangle_d \\ &= \mathbf{s}(\theta)^\top \langle \mathbf{u}_d \mathbf{u}_d^\top\rangle_d\, \mathbf{s}(\theta') \\ &= \mathbf{s}(\theta)^\top \mathbf{s}(\theta'),\end{aligned} \tag{22}$$

which is constant since $\mathbf{s}(\theta')$ do not change over time. A similar logic holds for the other moments, confirming that the OU drift on the encoding weights samples from a stationary distribution of activation functions.

If the encoding features $\mathbf{s}(\theta)$ are sampled from a Gaussian process on θ, then OU drift on the encoding weights amounts to OU drift over a Gaussian-process distribution of activation functions. Let the encoding weights change as $\mathbf{u}_d\sqrt{1-\alpha} + \xi\sqrt{\alpha}$, where α sets the drift rate and ξ are are the Gaussian noise sources as in Eq. (9) in the main text. Define the drift in activation at each time-point as $\Delta a(\theta) = \xi^\top \mathbf{s}(\theta)\sqrt{\alpha}$. The updated synaptic activations $a_{d+1}(\theta)$ are:

$$\begin{aligned}a_{d+1}(\theta) &= \mathbf{u}_{d+1}^\top \mathbf{s}(\theta) \\ &= (\mathbf{u}_d\sqrt{1-\alpha} + \xi\sqrt{\alpha})^\top \mathbf{s}(\theta) \\ &= \mathbf{u}_d^\top \mathbf{s}(\theta)\sqrt{1-\alpha} + \xi^\top \mathbf{s}(\theta)\sqrt{\alpha} \\ &= a_d(\theta)\sqrt{1-\alpha} + \Delta a(\theta).\end{aligned} \tag{23}$$

The increments $\Delta a(\theta)$ are samples from a Gaussian Process $\Delta a(\theta) \sim \mathcal{GP}(0, \Sigma_{\Delta a}(\theta,\theta'))$, with second moment:

$$\begin{aligned}\Sigma_{\Delta a}(\theta,\theta') &= \langle \Delta a_{i,t}(\theta)\Delta a_{i,t}(\theta')^\top\rangle_t \\ &= \alpha\, \langle \mathbf{s}(\theta)^\top \xi\xi^\top \mathbf{s}(\theta)\rangle_t \\ &= \alpha\, \mathbf{s}(\theta)^\top \langle \xi\xi^\top\rangle_t\, \mathbf{s}(\theta) \\ &= \alpha\, \mathbf{s}(\theta)^\top \mathbf{s}(\theta)\end{aligned} \tag{24}$$

If units are chosen such that the encoding features $\mathbf{s}(\theta)$ are zero-mean, then the second moments in Eqs. (22-24) can be interpreted as covariances. The randomly-drifting encoding weights $\mathbf{u}$ are therefore equivalent to an OU random walk through the space of possible activation functions on θ. We use this to simplify computations.

If the synaptic activations $a(\theta)$ are samples from a stationary distribution of functions over θ, then the tuning curves $x(\theta) = \phi[a(\theta)]$ are also samples from a stationary distribution over possible tuning curves. Adding homeostasis scales and shifts the activation to achieve the target firing rate statistics. This regulates the amount of information about θ encoded in the firing-rate variations of the population $\mathbf{x}(\theta)$ (Supplement: *Stability of encoded information*).

Stability of encoded information The model of drift described in Methods: *Simulated drift* and Supplement: *Gaussian-process tuning curves* conserves the information-coding capacity of $\mathbf{x}(\theta)$. Because individual encoding neurons evolve independently in this model, the population of N encoding cells represents N independent samples from the distribution of tuning curves on θ. If we consider the large ($N\to\infty$) population limit, the average population variability related to θ is given by the expected variability caused by θ for the typical tuning curve:

$$\lim_{N\to\infty} \tfrac{1}{N}\textstyle\sum_{i=1}^N \|\nabla_\theta \mathbf{a}_t(\theta)\|^2 = \langle \|\nabla_\theta a(\theta)\|^2\rangle_d, \tag{25}$$

where $\mathbf{a}_t(\theta)=\mathbf{U}_t^\top \mathbf{s}(\theta)$ is the vector of synaptic activations for all encoding units at location θ. The expected amount of population variability driven by θ is conserved, and is a function of the covariance of the activation functions $\Sigma_a(\theta,\theta')$:

$$\langle \|\nabla_\theta a(\theta)\|^2\rangle_d = \mathrm{tr}\left[\nabla_\theta \langle a(\theta)\, a(\theta)^\top\rangle_d \nabla_\theta^\top\right] = \nabla_\theta \Sigma_a(\theta,\theta)\nabla_\theta^\top. \tag{26}$$

The second step in Eq. (26) follows from the linearity of the trace and the identity $\|\mathbf{q}\|^2 = \mathrm{tr}[\mathbf{q}\mathbf{q}^\top]$. The operator $\nabla_\theta^\top$ refers to differentiating $\Sigma_a(\cdot,\cdot)$ in its second argument. This is in turn relates to the amount of variability in the features $\mathbf{s}(\theta)$ that is driven by θ:

$$\nabla_\theta \Sigma_a(\theta,\theta)\nabla_\theta^\top = \nabla_\theta \mathbf{s}(\theta)^\top \mathbf{s}(\theta)\nabla_\theta^\top = \|\nabla_\theta \mathbf{s}(\theta)^\top\|^2. \tag{27}$$

This shows that $\mathbf{a}(\theta)$ inherits the correlation structure of $\mathbf{s}(\theta)$, and that, in the large population limit, the variation in $\mathbf{a}(\theta)$ driven by θ is approximately conserved.

Additional assumptions about the nonlinearity $\phi[\cdot]$ are needed to show that stable information in $\mathbf{a}(\theta)$ implies stable information in $\mathbf{x}(\theta) = \phi[\mathbf{a}(\theta)]$. In the special case of an exponential nonlinearity ϕ= exp, the trace of Fisher information $\mathcal{I}(\theta)$ of $\mathbf{x}_{i,t}(\theta) = \exp[\mathbf{u}_{i,t}^\top \mathbf{s}(\theta)]$ is proportional to the average variation in $\mathbf{a}(\theta)$ driven by θ:

$$\mathrm{tr}[\mathcal{I}(\theta)] \propto \left\langle \|\nabla_\theta \ln[\mathbf{x}(\theta,t)]\|^2\right\rangle = \left\langle \|\nabla_\theta \mathbf{a}(\theta,t)\|^2\right\rangle \tag{28}$$

(Formally, the Fisher information is infinite when the noise in $\mathbf{x}$ is zero, but Eq. (28) can be viewed as the zero-variance limit of homogeneous and IID Gaussian noise with suitable normalization.) With a threshold nonlinearity, the dynamic range of each $a(\theta)$ must remain in a certain range to ensure that information is not lost due to the saturation in the firing-rate response. This can be ensured by homeostasis (23, 44, 69).

Hebbian homeostasis as an emergent property Here we explore a simplified linear model to make concrete the intuition that Eq. (5) in the main text should emerge through interactions between homeostasis and Hebbian learning. Consider a linear readout with inputs $\mathbf{x}$, weights $\mathbf{w}$, and output firing rate y:

$$y = \mathbf{w}^\top \mathbf{x} \tag{29}$$

Let $\mathbf{X}$ and $\mathbf{Y}$ be a training dataset of presynaptic inputs $\mathbf{x} \in \mathbf{X}$ and postsynaptic outputs $y \in \mathbf{Y}$. Assume that y and $\mathbf{x}$ are both zero-mean over this dataset. Consider a learning rule with a Hebbian term and other unknown contributions $g(\vartheta)$ (e.g. homeostatic terms). Since weight changes occur slowly, we consider the average weight update over the training data:

$$\Delta\mathbf{w} \propto \langle \mathbf{x}y\rangle + g(\vartheta). \tag{30}$$

Above, $g(\vartheta)$ reflects unknown contributions to the weight changes. We assume that $g(\vartheta)$ can be treated as constant over all $(\mathbf{x}, y)$, for a given weights and training dataset $\vartheta = (\mathbf{w}, \mathbf{X}, \mathbf{Y})$. If our neuron is in an initial, trained state $\vartheta_0 = (\mathbf{w}_0, \mathbf{X}_0, \mathbf{Y}_0)$, then learning has reached an equilibrium and $\langle\Delta\mathbf{w}\rangle$=0. This implies that the term $g(\cdot)$ must balance Hebbian contribution to synaptic plasticity:

$$g(\vartheta_0) = -\langle \mathbf{x}_0 y_0\rangle. \tag{31}$$

Define the homeostatic target σ_0^2 to be the variance of the firing-rate in this initial state y_0 (if y is zero mean, this is the second moment):

$$\sigma_0^2 = \langle y_0^2\rangle \tag{32}$$

Now, assume that a small amount of drift in the input encoding has occurred, $\mathbf{X}_1 = \mathbf{X}_0 + \Delta_{\mathbf{X}}$. This changes the readout's firing to $y_1 = y_0 + \Delta_y$, where $\Delta_y = \mathbf{w}_0^\top \Delta_{\mathbf{x}}$ is small. This alters the activity statistics of y_1, changing the rate variability $\sigma_1^2 \neq \sigma_0^2$. Assume that the variance of y_1 has been restored by homeostatic processes that multiply the firing rate by a factor $\gamma = \sigma_0/\sigma_1 = 1 + \delta$, where δ is a small parameter:

$$\tilde{y}_1 = \gamma y_1 = y_1 + \delta y_1 \tag{33}$$

Let's examine the impact of this adjusted activity on the Hebbain rule in Eq. (30).

$$\Delta\mathbf{w} \propto \langle \mathbf{x}_1 \tilde{y}_1 \rangle + \langle g(\vartheta_1) \rangle. \tag{34}$$

The unknown terms $\langle g(\vartheta_1) \rangle$ could also change their values due to drift. We can approximate this change to first order as:

$$\langle g(\vartheta_1) \rangle = \langle g(\vartheta_0) \rangle + \langle \Delta_{\mathbf{x}}^\top \nabla_{\mathbf{x}} g(\vartheta_0) \rangle + \langle \Delta_y \cdot \tfrac{d}{dy} g(\vartheta_0) \rangle + O(.^2), \tag{35}$$

where $O(.^2)$ denotes all terms at second order and higher. Since the learned state is at equilibrium, we can substitute Eq. (31) for $\langle g(\vartheta_0) \rangle$:

$$\langle g(\tilde{\vartheta}_1) \rangle = -\langle \mathbf{x}_0 y_0 \rangle + \langle \Delta_{\mathbf{x}}^\top \nabla_{\mathbf{x}} g(\vartheta_0) \rangle + \langle \Delta_y \cdot \tfrac{d}{dy} g(\vartheta_0) \rangle + O(.^2) \tag{36}$$

If $g(\vartheta)$ is constant in $(\mathbf{x}, y)$ for a given $\vartheta = (\mathbf{w}, \mathbf{X}, \mathbf{Y})$, the second and third terms reduce to $\langle \Delta_{\mathbf{x}}^\top \rangle \nabla_{\mathbf{x}} g(\vartheta_0)$ and $\langle \Delta_y \rangle \frac{d}{dy} g(\vartheta_0)$, respectively. Since $\mathbf{x}$ and y are constrained to be zero-mean, $\langle \Delta_{\mathbf{x}} \rangle$ and $\langle \Delta_y \rangle$ must also be zero, and these terms vanish. With these assumptions, Eq. (35) reduces to $\langle g_w(\tilde{\vartheta}_1) \rangle = -\langle \mathbf{x}_0 y_0 \rangle + O(.^2)$. This can be expressed as a function of $(\mathbf{x}_1, y_1)$ at first order:

$$\begin{aligned} \langle g_w(\tilde{\vartheta}_1) \rangle &= -\langle \mathbf{x}_0 y_0 \rangle + O(.^2) \\ &= -\langle (\mathbf{x}_1 - \Delta_{\mathbf{x}})(y_1 - \Delta_y) \rangle + O(.^2) \\ &= -\langle \mathbf{x}_1 y_1 \rangle - \langle \mathbf{x}_1 \Delta_y \rangle - \langle \Delta_{\mathbf{x}} y_1 \rangle + O(.^2) \end{aligned} \tag{37}$$

We can use Eq. (33) and Eq. (37) to write Eq. (34) as:

$$\Delta\mathbf{w} \propto \langle \mathbf{x}_1 (y_1 + \delta y_1) \rangle - \langle \mathbf{x}_1 y_1 \rangle - \langle \mathbf{x}_1 \Delta_y \rangle - \langle \Delta_{\mathbf{x}} y_1 \rangle + O(.^2). \tag{38}$$

Expanding and canceling, and neglecting second-order terms, yields:

$$\Delta\mathbf{w} \propto \delta \langle \mathbf{x}_1 y_1 \rangle - \langle \mathbf{x}_1 \Delta_y \rangle - \langle \Delta_{\mathbf{x}} y_1 \rangle + O(.^2) \tag{39}$$

Substituting $\Delta_y = \Delta_{\mathbf{x}}^\top \mathbf{w}$ and $y_1 = \gamma \mathbf{x}_1^\top \mathbf{w}$ gives:

$$\Delta\mathbf{w} \propto \delta \langle \mathbf{x}_1 y_1 \rangle - \left[\langle \mathbf{x}_1 \Delta_{\mathbf{x}}^\top \rangle + \gamma \langle \Delta_{\mathbf{x}} \mathbf{x}_1^\top \rangle \right] \mathbf{w} + O(.^2). \tag{40}$$

The terms $\langle \mathbf{x}_1 \Delta_{\mathbf{x}}^\top \rangle$ and $\langle \Delta_{\mathbf{x}} \mathbf{x}_1^\top \rangle$ are zero if the drift $\Delta_{\mathbf{x}}$ is uncorrelated with the encoding $\mathbf{x}_0$. This reduces Eq. (40) to:

$$\Delta\mathbf{w} \propto \delta \langle \mathbf{x}_1 y_1 \rangle + O(.^2). \tag{41}$$

This is similar to the Hebbian term in Eq. (5), if $\delta \propto \varepsilon_\sigma$ at first order. This is easily verified:

$$\frac{\varepsilon_\sigma}{\sigma_0} = \frac{\sigma_0 - \sigma_1}{\sigma_0} = 1 - \frac{\sigma_1}{\sigma_0} = 1 - \frac{1}{\gamma} = 1 - \frac{1}{1+\delta} = \delta + O(\delta^2). \tag{42}$$

Eq. (39) is therefore equivalent to the Hebbian contribution to the Hebbian homeostatic rule in Eq. (5) in the main text, with $\delta = \varepsilon_\sigma/\sigma_0$. What about the weight decay terms?

We assume that the norm of the weight vector, $\|\mathbf{w}\|^2$, is conserved. Hebbian learning will generally disrupt this. If we assume that the norm of the weight vector is restored by weight decay $-\rho\mathbf{w}$, what value of ρ would keep the norm of the weight vector constant? Consider a weight update as in Eq. (39), with an unknown weight decay term $-\rho\mathbf{w}$:

$$\mathbf{w}_1 = \mathbf{w}_0 + \delta \langle \mathbf{x}_1 y_1 \rangle - \rho \mathbf{w} \tag{43}$$

Assume that $\rho \sim O(\delta)$. What value would ρ need to take to ensure that $\|\mathbf{w}_1\|^2 = \|\mathbf{w}_0\|^2$? The norm of the updated weight vector is:

$$\|\mathbf{w}_1\|^2 = \|\mathbf{w}_0\|^2 + \delta \mathbf{w}_0^\top \langle \mathbf{x}_1 y_1 \rangle - \rho \|\mathbf{w}_0\|^2 + O(.^2). \tag{44}$$

We see that $\|\mathbf{w}_1\|^2 = \|\mathbf{w}_0\|^2$ if $\delta \mathbf{w}_0^\top \langle \mathbf{x}_1 y_1 \rangle = \rho \|\mathbf{w}_0\|^2 + O(.^2)$, implying that

$$\rho = \delta \frac{\mathbf{w}_0^\top \langle \mathbf{x}_1 y_1 \rangle}{\|\mathbf{w}_0\|^2} + O(.^2). \tag{45}$$

Since $\mathbf{w}_0^\top \mathbf{x}_1 = y_1$, the term $\mathbf{w}_0^\top \langle \mathbf{x}_1 y_1 \rangle = \langle \mathbf{w}_0^\top \mathbf{x}_1 y_1 \rangle = \langle y_1^2 \rangle = \sigma_1^2$ is equal to the firing-rate variability after perturbation by drift. This implies that

$$\Delta\mathbf{w} \propto \delta \left[\langle \mathbf{x}_1 y_1 \rangle - \frac{\sigma_1^2}{\|\mathbf{w}_0\|^2} \mathbf{w} \right]. \tag{46}$$

This suggests that the optimal value of weight decay is $\rho = \sigma_1^2 / \|\mathbf{w}_0\|^2$. In practice this value is not critical (we found that and setting $\rho = 1$ still led to good stability in simulations).

This derivation is not intended to prove that Hebbian homeostasis should arise in any specific physiological model, but rather to illustrate that a learning rule of this form could emerge from the interplay of Hebbian and homeostatic plasticity.

Weight filtering

The action of the Hebbian homeostatic rule (Eq. 5 in the main text) can be interpreted as a form of filtering. Consider a linear readout trained initially on day $d = 0$ with weights $\mathbf{W}_0$ (leave dependence of $\mathbf{x}$ and $\mathbf{y}$ on θ implicit to simplify notation):

$$\mathbf{y}_0 = \mathbf{W}_0^\top \mathbf{x}_0. \tag{47}$$

The encoding $\mathbf{x}_d$ changes on each day d. Tracking these changes entails translating the population code-words $\mathbf{x}_d$ into the code originally used on day 0. Using this translation $\hat{\mathbf{x}}_{0|d}$, one might achieve an approximately stable readout.

$$\hat{\mathbf{y}} = \mathbf{W}_0^\top \hat{\mathbf{x}}_{0|d}. \tag{48}$$

How might one estimate $\hat{\mathbf{x}}_{0|d}$? Consider estimating the code-words on day d from those on day $d+1$. Let drift $\Delta\mathbf{x}_d$ be sampled from a known distribution $\Delta\mathbf{x}_d \sim \mathcal{N}(0, \Delta_d)$. An estimate of $\hat{\mathbf{x}}_{d|d+1}$ can be obtained via linear least-squares:

$$\hat{\mathbf{x}}_{d|d+1} = \Sigma_d [\Sigma_d + \Delta_d]^{-1} \mathbf{x}_{d+1}, \tag{49}$$

where Σ_d is the covariance of the code $\mathbf{x}_d$ on the previous day.

Eq. (49) provides the minimum squared error estimate of $\hat{\mathbf{x}}_d$. In the linear, Gaussian case this is also the Bayesian *maximum a posteriori* estimate. Applying Eq. (49) iteratively yields an estimate of the original code $\hat{\mathbf{x}}_0$, thereby translating the current representation $\mathbf{x}_d$ back through time to when the readout was first learned:

$$\hat{\mathbf{x}}_{0|d} = \left\{ \textstyle\prod_{d' \in 0..d-1} \Sigma_{d'} [\Sigma_{d'} + s\Delta_{d'}]^{-1} \right\} \mathbf{x}_d. \tag{50}$$

Now, consider the effect of Eq. (49) on a decoded $\hat{\mathbf{y}}$ by substituting Eq. (49) into Eq. (48):

$$\begin{aligned} \hat{\mathbf{y}}_{d+1} &= \mathbf{W}_d^\top \Sigma_d [\Sigma_d + \Delta_d]^{-1} \mathbf{x}_{d+1} \\ &= [(\Sigma_d + \Delta_d)^{-1} \Sigma_d \mathbf{W}_d]^\top \mathbf{x}_{d+1}. \end{aligned} \tag{51}$$

The expression $[(\Sigma_d + \Delta_d)^{-1} \Sigma_d \mathbf{W}_d]$ in Eq. (51) is the same one used to re-train the readout from its own output (Eq. (14) in the main text, Methods: *Synaptic learning rules*, with Δ_d estimated as $\rho\mathbf{I}$):

$$\hat{\mathbf{y}}_{d+1} = [(\Sigma_d + \Delta_d)^{-1} \Sigma_d \mathbf{W}_d]^\top \mathbf{x}_{d+1} = \mathbf{W}_{d+1}^\top \mathbf{x}_{d+1}. \tag{52}$$

This illustrates that Hebbian homeostasis can be viewed (loosely) as filtering the current code-words $\mathbf{x}_d$ to recover the original code $\mathbf{x}_0$ against which the readout was first trained.

Predictive coding as inference In the main text (Results: *Predictive error feedback*), we argued that negative feedback of prediction errors removes variations in $\mathbf{y}(\theta)$ that are inconsistent with a learned internal model. In particular, if recurrent weights $\mathbf{A}_p$ are learned through symmetric Hebbian learning, then this feedback selectively tracks only the modes of $\mathbf{y}(\theta)$ known to encode information about θ. This leads the readout to infer a de-noised estimate $\hat{\mathbf{y}(\theta)}$ that can be used to update decoding weights. Here, we show that this process is structurally similar to Bayesian inference under certain circumstances.

Assume that the readout has internal states $\mathbf{z}$, and has learned a prior for what values these states should take for encoding θ, $\Pr(\mathbf{z})$. This prior reflects the ground-truth distribution of $\mathbf{z}$ experienced when the inputs $\mathbf{x}$ are driven by true external inputs $\mathbf{s}(\theta)$ during behavior. Assume that the error model for the feed-forwarded decoding $\mathbf{y}_f$ (Eq. 4 in the main text) in known, $\Pr(\mathbf{y}_f|\mathbf{z})$. For a given $\mathbf{y}_f$, the Baysian posterior estimate for $\hat{\mathbf{z}}$ is

$$\Pr(\hat{\mathbf{z}}|\mathbf{y}_f) \propto \Pr(\mathbf{y}_f|\mathbf{z}) \Pr(\mathbf{z}) \tag{53}$$

The estimate $\hat{\mathbf{z}}$ can be optimized by finding the posterior mode of Eq. (53). This can be done by maximizing the log-posterior:

$$\mathcal{L}(\mathbf{z}) = \ln \Pr(\mathbf{y}_f|\mathbf{z}) + \ln \Pr(\mathbf{z}) + \text{const.} \tag{54}$$

Now, consider the case where the prior on $\mathbf{z}$ is multivariate Gaussian. Let this prior be zero mean for convenience, without loss of generality, $\mathbf{z} \sim \mathcal{N}(0, \mathbf{A}_p)$. Let the observation model $\Pr(\mathbf{y}_f|\mathbf{z})$ be of the natural exponential family, where $\mathbf{z}$ are the natural parameters, and $\mathbf{y}_f$ are the natural statistics. Let $f(\cdot)$ be an element-wise function of $\mathbf{z}$ such that its derivative matches the firing-rate nonlinearity, $\phi(\mathbf{z}) = f'(\mathbf{z})$. The log-prior and log-likelihood then take the forms:

$$\begin{aligned} \ln \Pr(\mathbf{z}) &= -\tfrac{1}{2}\mathbf{z}^\top \mathbf{A}_p^{-1}\mathbf{z} + \text{const.} \\ \ln \Pr(\mathbf{y}_f|\mathbf{z}) &= \mathbf{z}^\top \mathbf{y}_f - f(\mathbf{z}) + \text{const.} \end{aligned} \tag{55}$$

The states $\mathbf{z}$ can be optimized via gradient ascent of $\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z})$ on Eq. (54), implying the following dynamics:

$$\dot{\mathbf{z}} = -\mathbf{A}_p^{-1}\mathbf{z} + \mathbf{y}_f^\top - \phi(\mathbf{z}) \tag{56}$$

Multiplying through by the prior covariance $\mathbf{A}_p$ does not change the fixed points, and so Eq. (56) can be written as:

$$\dot{\mathbf{z}} = -\mathbf{z} + \mathbf{A}_p[\mathbf{y}_f^\top - \phi(\mathbf{z})]. \tag{57}$$

Eq. (57) is the same as Eq. (7) in the main text (Results: *Predictive error feedback*), with the exception of a time constant τ_z which does not change the fixed points.

As an aside, modulating the feedback gain by a factor κ in Eq. (57) allows a neural population to dynamically adjust the influence of the external inputs and its internal model:

$$\dot{\mathbf{z}} = -\mathbf{z} + \tfrac{1}{\kappa}\mathbf{A}_p[\mathbf{y}_f^\top - \phi(\mathbf{z})]. \tag{58}$$

One might set κ to be small when one is confident in $\mathbf{y}_f$. This would cause learning to overwrite a learned $\mathbf{y}(\theta)$ with external input. Larger κ can be used to rely on priors more heavily when $\mathbf{y}_f$ are uncertain. We conjecture that modulating the feedback gain might control whether internal models vs. external input dominate activity, and, as a result, synaptic plasticity.

Overall, the correspondence between predictive coding and Bayesian inference is exact when

1. Latent states $\mathbf{z}$ receive prediction-error feedback, and can be interpreted as the natural parameters of an exponential-family distribution for $\hat{\mathbf{y}} = \phi(\mathbf{z})$.
2. Feedback weights $\mathbf{A}_p$ are proportional to the covariance of a Gaussian prior on $\mathbf{z}$.
3. The nonlinearity implies a natural exponential family that can reasonably capture error and uncertainty in $\mathbf{y}_f$.

The natural exponential family includes most common firing-rate models of neural dynamics, including linear-Gaussian, linear-nonlinear-Poisson, and linear-nonlinear-Bernoulli.

It is unlikely that this correspondence is exact *in vivo*. Regardless, Eq. (57) generically restricts the activity in $\mathbf{y}$ to the *subspace* associated with encoding θ (Results: *Recurrent feedback of prediction errors.*). Eq. (57) can provide useful error correction via subspace projection, even if it does not exactly correspond to a prior covariance on $\mathbf{z}$. The recurrent feedback model should therefore be interpreted as a qualitative hypothesis for how corrective feedback might aid in tracking drift.

Learning as inference Eq. (19) in the main text describes how the forwrad weights evolve based on the correlation between a prediction error $\mathbf{y}_0 - \mathbf{y}$ and the inputs $\mathbf{x}$ (Methods: *Single-neuron readout*). Analogously to how negative feedback optimizes a trade-off between input and an internal estimate of $\hat{\mathbf{y}}$, this optimization resembles statistical inference under certain assumptions (68).

Consider training the readout weight vector $\mathbf{w}$ for a single decoding unit, using L training examples, each consisting of an input $\mathbf{x}$ and a desired output y. The readout makes predictions $y_f = \phi(z)$, where $z = \mathbf{w}^\top\mathbf{x}$. Now, define a zero-mean Gaussian prior for these weights, $\mathbf{w} \sim \mathcal{N}(0, \frac{1}{\rho}\mathbf{I})$. Let the firing-rate nonlinearity $\phi(z) = f'(z)$ correspond to the derivative of a known function $f(\cdot)$, and assume that our observation model is taken from a natural exponential family with natural parameter z and natural statistics y. Up to constants, the log-posterior for $\mathbf{w}$ can then be written as

$$\begin{aligned} \langle \ln \Pr(y|\mathbf{w},\mathbf{x})\rangle &= \langle zy - f(z)\rangle + O(1) \\ \ln \Pr(\mathbf{w}) &= -\tfrac{1}{2}\mathbf{w}^\top[\tfrac{1}{\rho}\mathbf{I}]^{-1}\mathbf{w} + O(1) \\ \Rightarrow \langle \ln \Pr(\mathbf{w}|\mathbf{x},y)\rangle &\propto \langle zy - f(z)\rangle - \tfrac{1}{2}\rho\|\mathbf{w}\|^2 + O(1), \end{aligned} \tag{59}$$

where the expectation $\langle\cdot\rangle$ averages over the training data. This objective can be optimized by ascending the gradient $\nabla_{\mathbf{w}}$ of the log-likelihood, implying to the following weight dynamics:

$$\begin{aligned} \Delta\mathbf{w} &\propto \langle \mathbf{x}y - \nabla_{\mathbf{w}} f(z)\rangle - \rho\mathbf{w} \\ &= \langle \mathbf{x}(y - \phi(\mathbf{w}^\top\mathbf{x}))\rangle - \rho\mathbf{w} \\ &= \langle \mathbf{x}(y - y_f)\rangle - \rho\mathbf{w} \end{aligned} \tag{60}$$

Eq. (60) is equivalent to the error-based learning rule used in the main text (Eq. 25), if one applies this update in discrete steps with learning rate η to a population readout. As was the case for prediction-error feedback, this statistical interpretation does not correspond exactly to processes *in vivo*. Nevertheless, learning rules resembling Eq. (60) emerge in some timing-dependent plasticity rules (68).

Calibrating the model to data The results in the main text are abstract, because drift statistics vary across brain areas, species, and experimental conditions. Stability depends on many unknown parameters, include drift rate, noise levels, population-code redundancy, and the frequency of reconsolidation. Nevertheless, in Supplemental Figure 3, we present a best-effort calibration of our simulations to the Driscoll et al. (2017) data recorded from mouse posterior parietal cortex (2, 43).

We calibrated a model of simulated drift using three subjects from Driscoll et al. (43) (mice M1, M3, and M4). For each subject, we selected a subset of fifteen recording sessions sharing common neurons (N=10, 60, and 83 neurons from M1, M3, and M4 respectively). Each subpopulation was tracked for over a month (58, 35, and 38 days respectively), with gaps no longer than 13, 8, and 10 consecutive days, respectively. For each day, we filtered log-Calcium fluorescence traces between 0.03 and 0.3 Hz and normalized them by z-scoring. We aligned filtered traces from successful trials to task pseudotime ("θ") based on progress through the maze. These traces were averaged to estimate neuronal tuning curves.

The tuning curves can be interpreted as vectors in a high-dimensional space, with a different component for each location θ. We used the cosine of the angle ψ between two tuning curves as an "alignment" measure to quantify drift. This can be computed as the average product $\cos(\psi) = \langle \tilde{x}_1(\theta)\tilde{x}_2(\theta)\rangle_\theta$ between normalized (z-scored) tuning curves $\tilde{x}_i(\theta)$, and is 1 if the tuning curves are identical and 0 (on average) if they are unrelated.

This estimator is biased by measurement noise and trial-to-trial variability. We compensated for this by bootstrap-resampling the average alignment between two tuning curves from the same cell and day, where each tuning curve is estimated as an average over a different random subsets of trials. This baseline was calculated separately for each neuron and day, averaged over ten random samples. Bias was removed by dividing the tuning-curve alignment measure by this baseline. This retains excess per-day variability that cannot be explained by drift, noise, or trial-to-trial fluctuations.

Alignment decayed exponentially over time (Figure S3-a). The decay time constant (τ) indicates the drift rate. The extrapolated value at $\Delta = 0$ days (a_0) indicates the excess per-day variability. We estimated these parameters using least-squares exponential regression. Results were similar across subjects (τ = 57, 23, 54 and a_0 = 0.63, 0.69, 0.72; for M1, M3, M4 respectively). We used the average values of these parameters across subjects ($\bar{\tau}$ = 45 and $\bar{a}_0$ = 0.67) to calibrate a model of drift (Figure S9-b).

Comparing simulation and experiment also requires assumptions about the frequency of "self-healing" reconsolidation, relative to the drift rate. In our simulations we apply self-healing every $\Delta = 5$ time points. If we assume that "self-healing" occurs once per day *in vivo*, then an excess per-timepoint variability of $r = 30\%$ and a time constant of $\tau = 45$ days matches the simulated drift to the Driscoll et al. (2017) (2) data. This implies approximate stability out to ~10 days using redundancy alone (fixed readout weights), a few months using Hebbian homeostasis, and over a year if the readout contains a stable internal model (Figure S9-d). However, a rigorous test of how these ideas apply *in vivo* would require new experimental studies.

Normalized Root Mean Squared Error (NRMSE) In Supplemental Figures 1, 2, and 5 we summarized the stability of the readout population code by measuring the normalized distance between the initial, trained readout firing-rates $\mathbf{y}(\theta)$, and the firing rates on a given time-step $\mathbf{y}_d(\theta)$.

$$\mathrm{NRMSE}(\mathbf{y}(\theta), \mathbf{y}_d(\theta)) = \sqrt{\tfrac{1}{2}\,\langle \tilde{\mathbf{y}}(\theta) - \tilde{\mathbf{y}}_d(\theta)\rangle_{\theta,M}}. \tag{61}$$

The values $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{y}}_d$ reflect normalized tuning curves , in which the firing-rate function $y(\theta)$ for each readout neuron has been z-scored The average $\langle\cdot\rangle_{\theta,M}$ is taken over all M decoding units and all L values of θ. The normalization by 1/2 ensures NRMSE ranges from 0 (identical codes) to 1 (chance level).

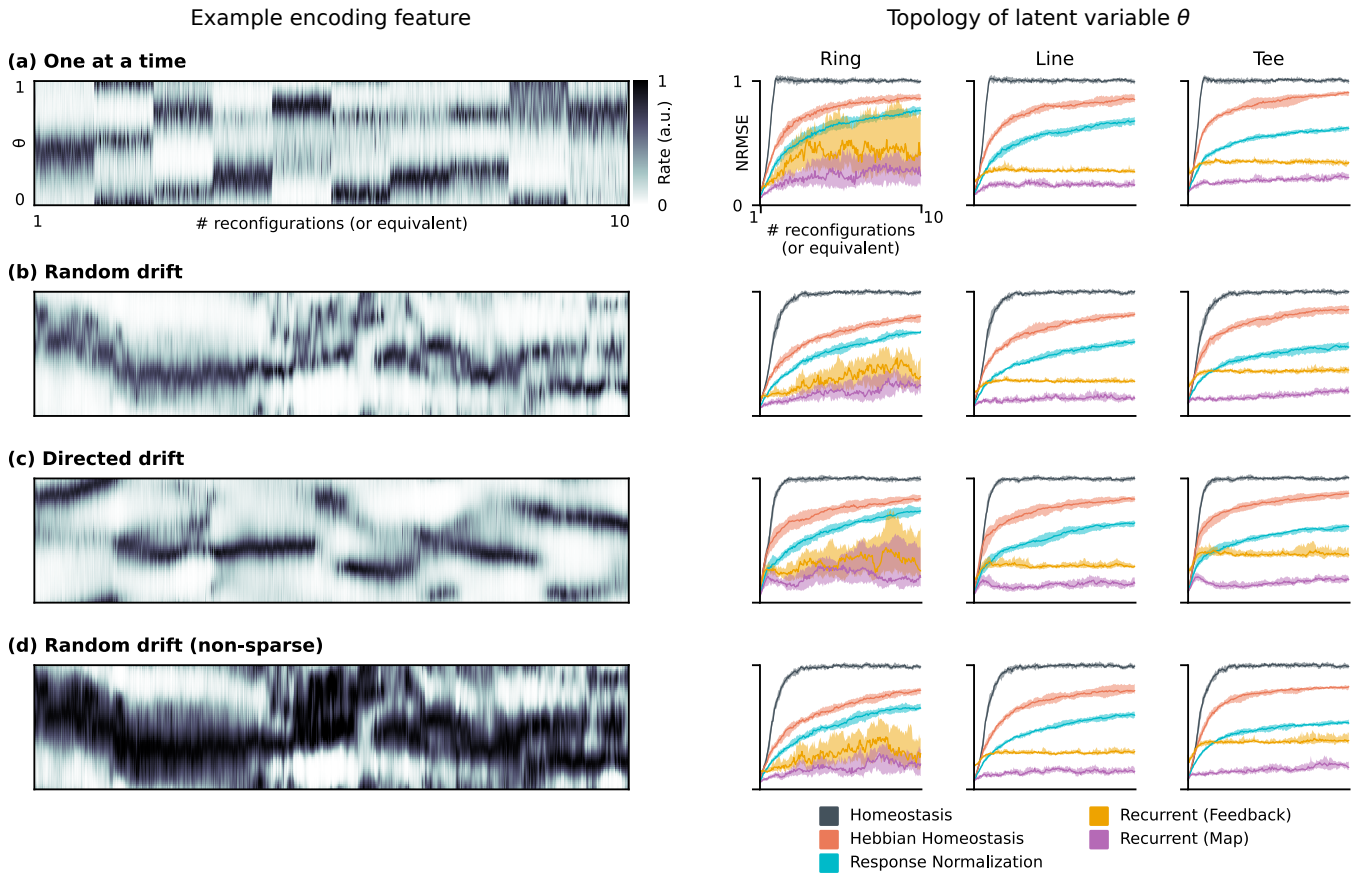

Figure S1: ***Self-healing stabilizes readout population codes for diverse types of representational drift.*** **Left:** An example of a single drifting encoding feature $x(\theta)$ sampled for a circular θ for different hypothetical drift scenarios. The horizontal axes for all plots are expressed in terms of the number of complete reconfigurations of the encoding population-code (or equivalent, for scenarios b-d). All simulations were run for T=1000 time-steps, corresponding to 10 complete reconfigurations in the encoding population. For continuous drift (b-d), time-constants were set to match the rate of population drift corresponding to changing encoding features one-at-a-time for a population of $N = 100$ encoding neurons. **Right:** Normalized Root-Mean-Squared-Error (NRMSE; 0=perfect match, 1=chance) of the readout population code over time. Lines indicate the median over 10 random seeds, and shaded regions the inter-quartile range. Simulation parameters are the same as for scenarios b-f in Fig. 3 in the main text. We ran "self-healing" reconsolidation every $\Delta = 5$ iterations. We explored three topologies for θ: circular, linear, and T-maze (compare to Supplemental Figure S4). The rate of decay of the readout population code does not depend on the style of drift in the encoding population. **(a)** "One-at-a-time" drift changes one out of $N = 100$ encoding neurons on each iteration of the simulation. 100 simulated time-steps corresponds to one complete reconfiguration of the encoding population. **(b)** "Random drift" applies Ornstein-Uhlenbeck (OU) drift with a time constant $\tau = 100$ (Methods: *Simulated drift*). **(c)** "Non-sparse drift" samples the encoding curves directly from a linear, Gaussian process, and does not apply the firing-rate nonlinearity $\phi(\cdot)$. These features lack the sparse, bump-like tuning curves present in the other scenarios. The variance has been scaled to match that of the other drift scenarios. The correlation time is $\tau = 100$. **(d)** "Directed" drift simulates a second-order OU process evolving as two stages of Eq. (9) in the main text chained in series, such that consecutive changes are correlated in time. Each stage has a time constant $\tau = 50$.

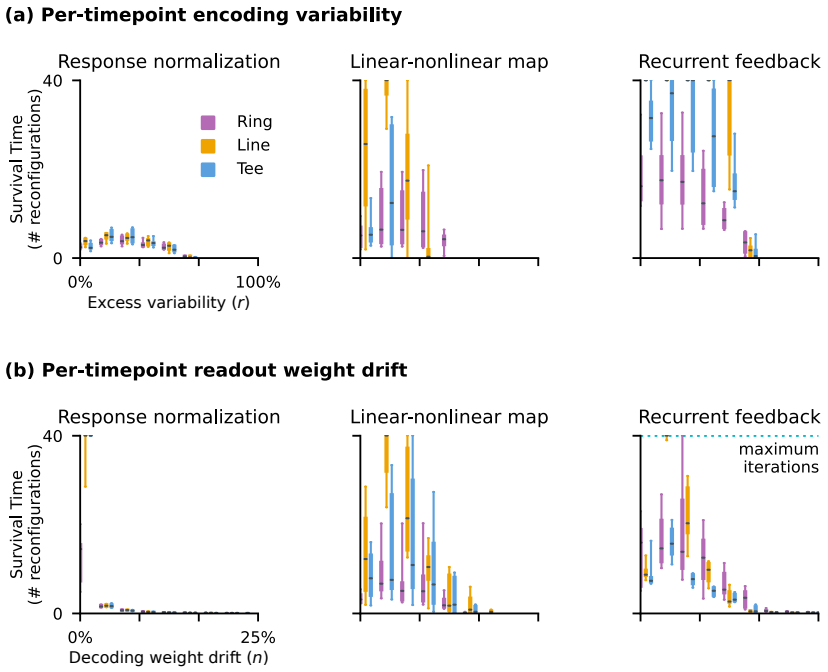

Figure S2: ***Encoding variability and readout-weight drift affect stability.*** Each plot shows the survival time for a self-healing readout as in Fig. 3 of the main text, measured as the time until the normalized root-mean-squared error of the readout exceeded 0.5. $T = 4000$ time-points are simulated for an encoding population of N=100 cells with a drift time constant of τ=100. Time (vertical axis) is expressed in multiples of the drift time constant. Boxes show the median (black) and inter-quartile range. Whiskers indicate the 5th-95th percentiles over 10 random seeds. We explored three topologies for θ: circular, linear, and T-maze (compare to Supplemental Figure S4). All simulations used the same parameters as in Fig. 3 in the main text, with the exception of the noise parameter (r or n) which is varied along the horizontal axis. Drift is gradual as described in Methods: *Simulated drift*. "Self-healing" reconsolidation is applied every $\Delta = 5$ time-steps. **(a)** We varied the amount of daily variability in the code $\mathbf{x}(\theta)$ that is unrelated to cumulative drift. This is expressed as the percentage of the variance in synaptic activation for the encoding neurons that is unique to each day (r, horizontal axis; Eq. (10) in the main text). The "response normalization", "recurrent feedback", and "linear-nonlinear map" scenarios all show good stability up until $n \approx 40\%$ (c.f. Fig. 3 d-f in the main text). **(b)** We varied the amount of drift applied to the readout's decoding weights $\mathbf{W}$ on each day (n, horizontal axis; Eq. (11) in the main text). Recurrent dynamics can tolerate small amounts of readout-weight drift, but stability degrades if drift exceeds $\approx$8% per time-step.

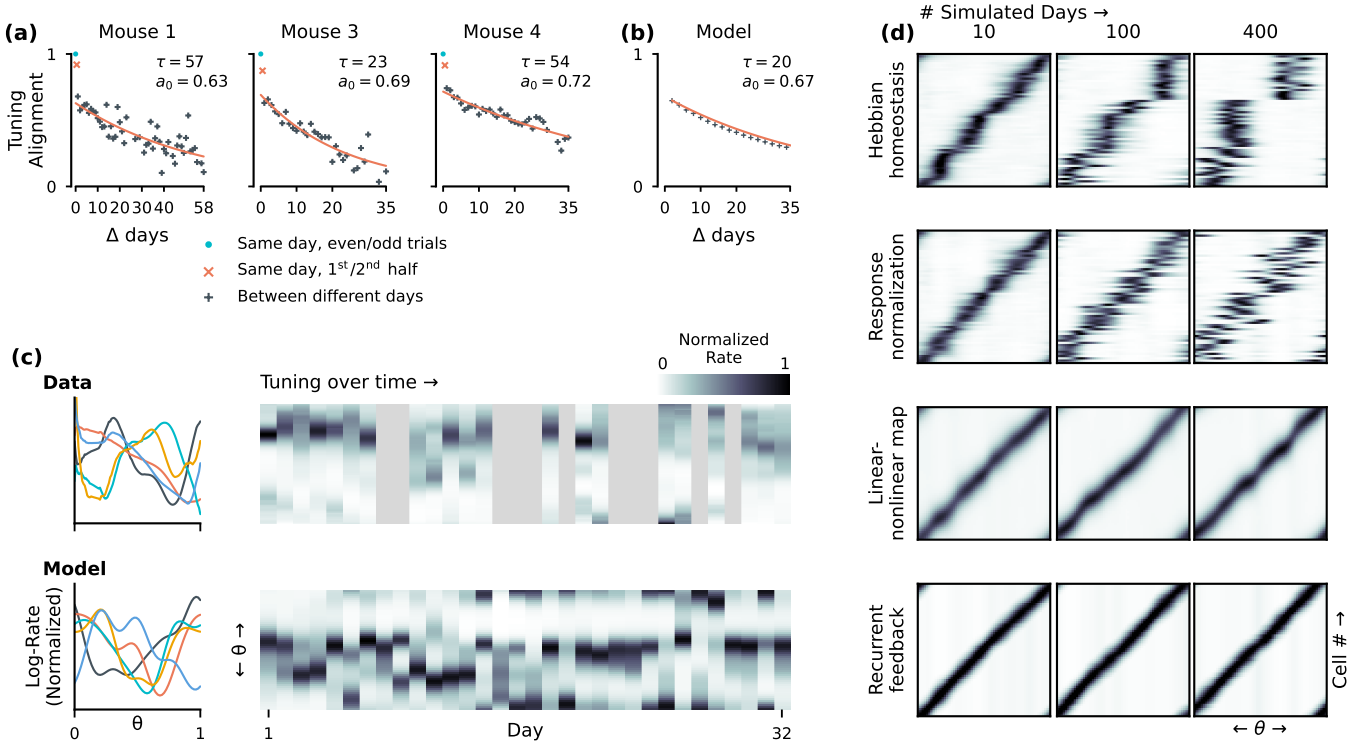

Figure S3: ***The statistics of drift observed in vivo are compatible with long-term stability.*** **(a)** The rate of drift can be measured using the cosine of the angle between two tuning curves ("alignment"), estimated on different days for the same cell (1: identical; 0: unrelated). Plots show tuning-curve alignment for a population tracked for over a month, taken from three subjects in Driscoll et al. (43). Each point reflects the average alignment across the population for a pair of recordings separated by Δ days. Measurement noise exaggerates apparent tuning differences, so alignment was normalized via a bootstrap estimator such that tuning curves estimated from different trials on the same day were fully aligned (teal '∘'). Alignment decays exponentially, with a similar timescales across subjects ("τ"). Extrapolating the day-to-day alignment (black '+') to a separation of zero days (a_0) does not yield perfect alignment, indicating that not all day-to-day tuning variability is explained by drift. This excess variability also cannot be attributed to systematic drift during the recording session (measured as the alignment between the first and second half of the recording session; red '×'). **(b)** We modeled tuning curves as samples from a log-Gaussian process over the latent space θ, with drift modeled as an Ornstein-Uhlenbeck random walk in tuning over time (Methods: *Simulated Drift*). We used a time-constant of $\tau = 45$ days and applied $r = 30\%$ excess per-day variability to match the model to experimental data. **(c)** Tuning curves sampled from the model (bottom left) qualitatively resemble those in the experimental data. Tuning curves *in vivo*, however, exhibited nonuniform statistics in θ (top left). The statistics of drift (right) are also similar, with both model and data exhibiting day-to-day variability superimposed over long-term drift. **(d)** Evolution of readout population tuning curves under simulated drift. The drift timescale and day-to-day variability were calibrated as in (b). All other parameters were the same as in Fig. 3. Readouts with a stable internal model ("linear-nonlinear map" and "recurrent feedback") exhibit long-term stability.

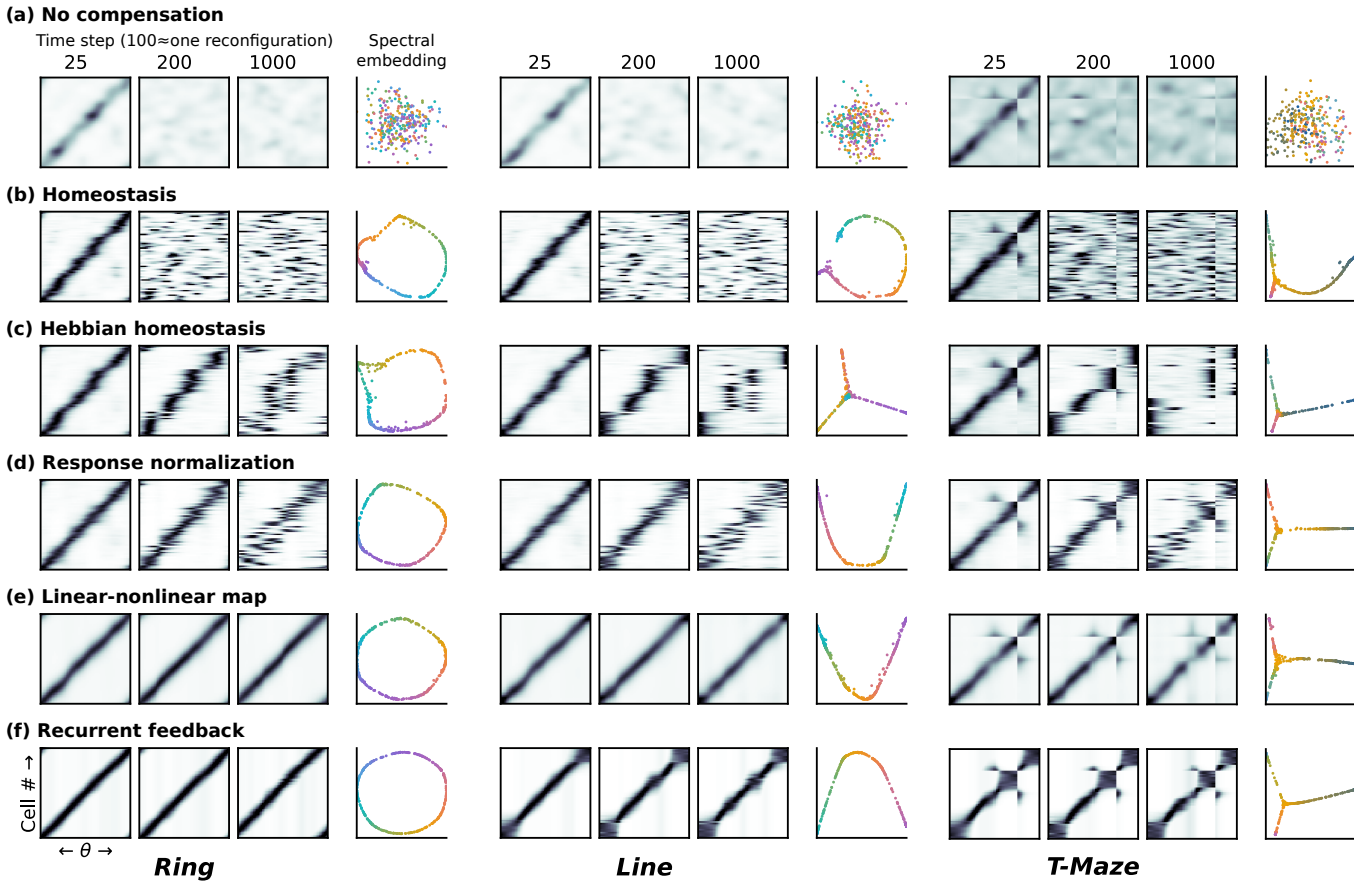

Figure S4: ***Self-healing plasticity stabilizes various geometries.*** We simulated representational drift under various self-healing plasticity scenarios as in Fig. 3 of the main text, applied to different geometries (**left**: ring, **middle**: line, **right**: T-maze). The covariance kernel of the Gaussian-process synaptic activations for $\mathbf{x}(\theta)$ was adapted to each geometry, keeping the correlation as a function of distance the same as in Fig. 3 in the main text (Methods: *Simulated drift*). We simulated 1000 iterations of drift with time-constant $\tau = 100$. Results are similar across all three topologies. Black-and-white plots show the configuration of the readout population code at various times. Colored plots show the result of applying unsupervised dimensionality reduction to the final readout population tuning curves (Python sklearn SpectralEmbedding (70); c.f. (39)). We applied this embedding to points sampled from five random 'walkthroughs' of θ with additive Gaussian noise $\sigma_\xi = 1.2 \times 10^{-2}$ to emphasize the degradation in the signal-to-noise ratio. **(a)** Without compensation, the amount of variability in $\mathbf{y}(\theta)$ that is related to θ decays, lowering the signal-to-noise ratio. Both the original tunings, and the capacity to encode θ, is lost. **(b)** With homeostasis, the original readout tuning curves are lost. However, homeostasis stabilizes the information-coding capacity of the readout. Nonlinear dimensionality reduction recovers the underlying topology of θ. **(c)** Hebbian homeostasis provides some stability, but causes the readout population code to collapse around a few salient preferred θ_0. **(d)** Response normalization compensates for the destabilizing impact of Hebbian homeostasis. However, noise causes readout neurons to swap their preferred tunings. **(e, f)** Long-term stability is possible in readouts with a stable internal model. Sharing of information among the readout population, modeled here as either a linear-nonlinear map or recurrent feedback, allows for more robust error correction.

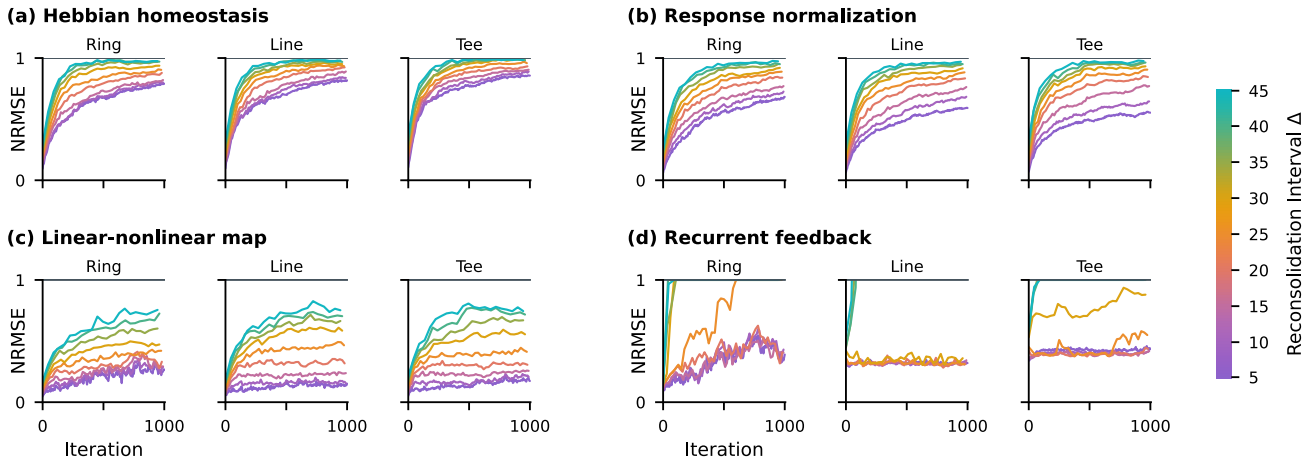

Figure S5: ***Larger amounts of drift between reconsolidation sessions reduces stability.*** Changing the rate of drift relative to the frequency of reconsolidation affects the stability of the readout population code. In these simulations, all parameters are the same as in Fig. 3 in the main text, with the exception of the frequency of reconsolidation Δ. In the main text, $\Delta = 5$. We explore up to $\Delta = 45$, equivalent to nine times faster drift. The rate of degradation for Hebbian homeostasis scales with the rate of drift, with **(a)** and without **(b)** response normalization. Error correction via linear-nonlinear map **(c)** also degrades with increasing Δ, but less so. The error levels off to a steady-state, suggesting long-term stability. With recurrent feedback **(d)**, the population readout is stable for modest rates of drift, but loses stability above a certain rate ($\Delta \approx 25$).