Learning neural dynamics through instructive signals

Rich Pang[1,2,*], Juncal Arbelaiz[1,3], and Jonathan W. Pillow[1,**]

[1]Princeton Neuroscience Institute, Princeton University, Princeton, NJ, USA.
[2]Lewis-Sigler Institute for Integrative Genomics, Princeton University, Princeton, NJ, USA.
[3]Center for Statistics and Machine Learning, Princeton University, Princeton, NJ, USA.
[*]Correspondence: rkp.science@gmail.com
[**]Correspondence: pillow@princeton.edu

SUMMARY

Rapid learning is essential for flexible behavior, but its basis in the brain remains unknown. Here we introduce the PRISM plasticity rule, a unifying mechanistic model of three well-established, fast-acting synaptic plasticity rules—in hippocampus, cerebellum and mushroom body—which relies exclusively on pre-synaptic activity and an "instructive signal" from another brain area. Using a multi-region network model we show that guiding PRISM plasticity with instructive signals enables the network to quickly learn extremely flexible nonlinear dynamics underlying behaviorally relevant computations, as well as to emulate unknown external system dynamics from real-time error signals, which we demonstrate with comprehensive simulations supported by exact mathematical theory. Thus, PRISM plasticity guided by instructive signals is well-suited to rapidly learn general-purpose neural computations—in contrast to canonical Hebbian rules. Finally, we show how including this plasticity rule in artificial learning algorithms can solve long-range temporal credit assignment, a long-standing challenge in machine learning.

KEYWORDS

rapid learning, neural dynamics, plasticity, instructive signals, hippocampus, cerebellum, mushroom body, recurrent neural network

Highlights

- PRISM (PResynaptic and Instructive Signal-Mediated) plasticity—a unifying mechanistic model of three fast-acting plasticity rules in hippocampus, cerebellum, and mushroom body.
- Mathematical theory inspired by Support Vector Machine exactly predicts the dynamics learned in a network model through PRISM plasticity.
- Examples of learning nonlinear dynamics in a single shot via sparse instructive signals or from real-time error signals.
- Machine-learning application of instructive signals and PRISM plasticity to solve the long-range temporal credit assignment problem.

INTRODUCTION

Rapid learning is essential for flexible behavior, allowing animals to quickly adapt to changing environments and task conditions. Mechanistically, learning is thought to be mediated by synaptic

plasticity in the brain. To date, most models of biologically plausible learning focus on Hebbian plasticity, in which synaptic weight changes are driven by coincident pre- and post-synaptic activity[1-10]. Such rules have wide support from *in vitro* experiments[3,4,11,12] and in network models are well suited to form networks capable of performing pattern completion supporting memory recall, or the production of stereotyped activity sequences supporting motor outputs[2,5,6,9]. It has proved difficult, however, to coerce Hebbian rules to produce more general-purpose computations needed for flexible behavior[13], challenging their overall power and expressivity. Moreover, recent experiments suggest that some of the strongest plasticity rules *in vivo*, with timescales best matched to rapid learning ($\sim$ seconds), are in fact markedly non-Hebbian in character[14-18]. Thus, it sensible to consider what types of computations fast, non-Hebbian rules can produce.

In mammalian hippocampus and cerebellum, and the fly mushroom body—three highly plastic circuits involved in rapid learning—some of the strongest, most well-established synaptic plasticity rules depend not on pre- and post-synaptic activity, but on pre-synaptic activity and an "instructive signal" from another brain area[15-25] (Fig 1a-b). We term this mechanism PRISM ("PResynaptic and Instructive Signal-Mediated") plasticity. While the precise temporal dependencies vary[15,16,23], PRISM plasticity in these three areas is highly stereotyped in several key aspects. First, it can act very quickly, producing large, lasting weight changes in just seconds. Second, it is largely independent of the post-synaptic neuron's activity, in contrast to Hebbian plasticity. Third, it occurs at feed-forward output synapses from neural populations thought to exhibit *pattern-separated representations* (PSRs)—high-dimensional non-overlapping neural codes for upstream inputs[26-31]. PRISM plasticity at PSR outputs may thus reflect a common mechanism of rapid learning in the brain, yet its ability to produce flexible neural computations has received far less investigation than Hebbian rules.

Computation in the brain is thought to be mediated by the activity dynamics of populations of neurons, which are often studied via recurrent neural network (RNN) models[2,6,32-38]. An RNN is a flexible, nonlinear model that consists of a network of artificial neurons whose activities change over time under the influence of external and recurrent/feedback inputs. By adjusting synaptic weights, RNNs can be trained to perform diverse tasks and their internal activity subsequently examined to probe the learned dynamical mechanisms. Typically, RNN synaptic weights are learned through either back-propagation through time (BPTT)[37-39], a flexible machine-learning algorithm yet usually considered non-biological; or Hebbian plasticity[2,6,8], which has limited expressive power and may act too gradually in the brain to support rapid learning[4,11,40,41]. While emerging work suggests that PRISM rules can enable RNNs to learn certain bespoke tasks[42,43], whether they can produce dynamics supporting general-purpose nonlinear computation needed for flexible behavior is unknown.

Here we investigate a multi-region RNN consisting of a "base" RNN loosely analogous to neocortex and a feedback circuit analogous to hippocampus, cerebellum, or mushroom body. Synaptic outputs of PSR neurons within the feedback circuit are subject to PRISM plasticity guided by instructive signals external to the RNN. We show that given suitable instructive signals, PRISM plasticity enables the RNN to learn highly flexible, general-purpose nonlinear dynamics supporting behaviorally relevant computations, which we demonstrate with comprehensive simulations supported by exact mathematical theory. Our theory is precisely analogous to the well-understood theory of Support Vector Machines[44-46], except set in a dynamical context, concisely linking the learned network connectivity to the resulting neural dynamics[34,35,47]. This in turn reveals the ability of PRISM rules to produce arbitrarily complex dynamics. Finally, we show how incorporating PRISM rules into artificial learning algorithms based on gradient descent can provide a solution for linking network outputs to long-vanished causal inputs, a long-standing challenge in machine learning[48].

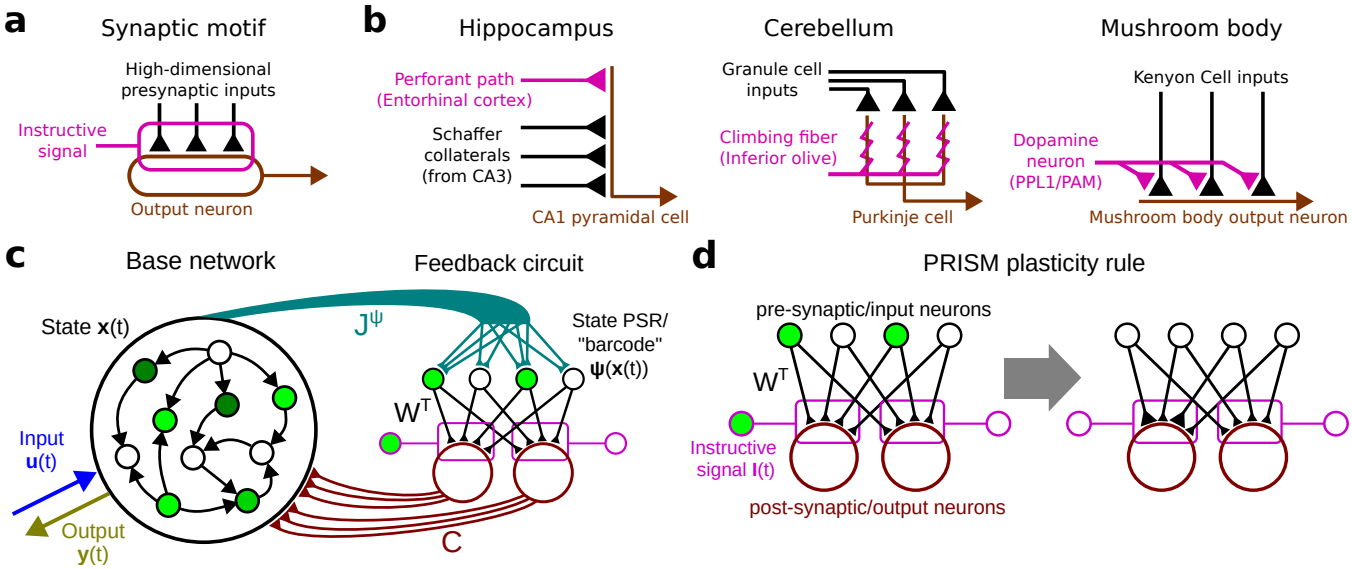

Figure 1: Network model and PRISM rule.
a. Stereotyped synaptic motif in which plasticity of pre-synaptic weights onto post-synaptic output neuron depends on pre-synaptic activity and an instructive signal. **b**. Schematics of physical instantiations of synaptic motif in **a** in three different brain areas[14–18,20–23]. **c**. Network model including base network and plastic feedback circuit, with the latter shaped by instructive signals. **d**. Schematic of PRISM plasticity rule in Eq. (3).

RESULTS

We first introduce our network model and a mathematical theory for the rapid learning of neural dynamics. We then present a series of simulations characterizing several distinct ways the network can be trained through instructive signals, revealing the wide-ranging capabilities of rapid biological plasticity rules observed in the brain.

Model

Architecture

Consider a "base" RNN reciprocally connected to a feedback circuit analogous to hippocampus (specifically, the CA3-to-CA1 layer), cerebellum, or mushroom body (Fig 1c). Let there be D neurons in the base RNN, and within the feedback circuit let there be N pre-synaptic or input neurons and Q post-synaptic or output neurons (Fig 1d). The pre-synaptic or input neurons correspond to CA3 pyramidal cells (hippocampus), granule cells (cerebellum), or Kenyon cells (mushroom body); and the post-synaptic or output neurons to CA1 pyramidal cells (hippocampus), Purkinje cells (cerebellum), or mushroom body output neurons. As in cerebellum and mushroom body, we will generally assume $N \gg Q$, i.e. there is strong convergence from the pre-synaptic/input neurons to the post-synaptic/output neurons within the feedback circuit.

Dynamics

The activity dynamics of the neurons in the base RNN consist of four terms: a leak term reflecting the intrinsic decay of activity within each neuron, a term corresponding to recurrent inputs arising from within the base RNN, a term corresponding to recurrent inputs from the feedback circuit,

and a term corresponding to external inputs. Specifically, let the dynamics of the D neurons in the base RNN follow:

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + J^{base}\phi(\mathbf{x}) + CW^T\boldsymbol{\psi}(\mathbf{x}) + B\mathbf{u}(t), \tag{1}$$

where $\mathbf{x}(t)$ are the activity levels of the D base neurons, ϕ is an element-wise nonlinearity such as tanh, $\boldsymbol{\psi}(\mathbf{x})$ are the PSR-like activities of the N input neurons within the feedback circuit (described in detail below), $\mathbf{u}(t)$ are external inputs, and τ is a time constant determining the leak rate when all other inputs are zero. The recurrent synaptic weights within the base network are given by J^{base}, the synaptic weights within the feedback circuit by W, the weights from the feedback circuit output neurons back to the base RNN by C, and the weights on the external inputs by B. We optionally include a linear readout $\mathbf{y}(t)$. To focus on the effects of the feedback circuit we will usually consider $J^{base} = 0$, i.e. no base dynamics besides the linear decay toward the origin.

The input neuron activations $\boldsymbol{\psi}(\mathbf{x})$ within the feedback circuit—corresponding to CA3, granule, or Kenyon cell activities—exhibit a PSR of the base RNN state $\mathbf{x}$, created via a random projection of $\mathbf{x}$ by matrix J^{ψ} into an N-dimensional space followed by a nonlinearity. Unless otherwise noted, the main example we will use throughout this work, which we term the "Rand-Sig" $\boldsymbol{\psi}$ (random projection followed by a sigmoid), is:

$$\boldsymbol{\psi}(\mathbf{x}) = \frac{\sqrt{2/N}}{1 + \exp(-2J^{\psi}\mathbf{x})} \tag{2}$$

where the weights J^{ψ}_{iq} are sampled from a normal distribution with standard deviation, or "gain" g. The activities of the output neurons of the feedback circuit are given by $\mathbf{o}(t) = W^T\boldsymbol{\psi}(\mathbf{x}(t))$ and are implicit in Eq. (1). Intuitively, the PSR $\boldsymbol{\psi}(\mathbf{x})$ acts as a "barcode" for the system state $\mathbf{x}$—each $\mathbf{x}$ maps to a unique (with high probability) high-dimensional vector $\boldsymbol{\psi}(\mathbf{x})$, with largely uncorrelated barcodes for distant states but similar barcodes for nearby states, in line with the well-studied pattern-separation action of biological random networks[29,49]. As we will show, the form of $\boldsymbol{\psi}$ influences the geometric structure of the learned dynamics. For instance, surrounding Eq. (2) with a K-winner-take-all nonlinearity that silences all but the K most active neurons, as proposed in mushroom body and CA1 models[50,51], can decrease interference and promote further pattern separation. However, we will usually consider the Rand-Sig $\boldsymbol{\psi}$.

PRISM Plasticity Rule

To focus on rapid plasticity mediated by instructive signals, we take inspiration from biological PRISM plasticity at PSR outputs and let all plasticity be localized exclusively to the weights W within the feedback circuit (Fig 1c). These weights are analogous to the feed-forward, convergent CA3 to CA1 projections (hippocampus), granule cell to Purkinje cell projections (cerebellum), and Kenyon cell to mushroom body output neuron projections. Our goal is to understand whether and how plasticity of these weights alone can produce flexible, general-purpose nonlinear dynamics, which would confer substantial power and expressivity to a single layer of feed-forward network weights subjected to a biological plasticity rule.

We formalize the PRISM rule as:

$$\frac{dW^T}{dt} = \mathbf{I}(t)\boldsymbol{\psi}(\mathbf{x}(t))^T, \qquad \text{or component-wise,} \qquad \frac{dW_{iq}}{dt} = I_q(t)\psi_i(\mathbf{x}(t)) \tag{3}$$

where $\mathbf{I}(t)$ is an external *instructive signal* with Q components, i.e. one instructive signal per output neuron. As in biology, PRISM plasticity is induced by pre-synaptic and instructive signals but

expressed at synapses from pre-synaptic to post-synaptic neurons (Fig 1d). Note that PRISM, while non-Hebbian, is still a "two-factor" rule, in contrast to "three-factor" rules in which Hebbian plasticity is gated by an instructive or neuromodulatory signal[52-54]. For simplicity, Eq. (3) relies on simultaneous activation of the pre-synaptic neuron and the instructive signal, but see Extended Methods for a generalization to temporally extended plasticity windows[15-17,23]. The simultaneous case provides crucial mathematical insights for addressing the temporally extended case.

An illustration of the PRISM rule is shown in Fig 1d. A synapse W_{iq} from pre-synaptic neuron i to post-synaptic neuron q is modified when: (1) its pre-synaptic activity $\psi_i(\mathbf{x}(t))$ is greater than 0 (non-negativity is enforced by Eq (2)); and (2) there is a simultaneous instructive signal $I_q(t) \neq 0$. If the instructive signal is positive (negative) the weight increases (decreases). The effect of the learning rule can also be understood as follows. At time t^*, the barcode of system state $\mathbf{x}(t^*)$ is given by $\boldsymbol{\psi}(\mathbf{x}(t^*))$. If an instructive signal impulse $I_q(t) = \alpha^* \delta(t - t^*)$ occurs at this time, then column q of the weight matrix W, corresponding to the q-th output neuron, is incremented by $\alpha^* \boldsymbol{\psi}(\mathbf{x}(t^*))$. Intuitively, $\boldsymbol{\psi}$ specifies "what" to learn, and $\mathbf{I}$ specifies "when" to learn it.

As we will show, different patterns of instructive signals $\mathbf{I}(t)$, along with external inputs $\mathbf{u}(t)$, can be used to train the network in different ways. In specific cases it will be natural to allow $\mathbf{I}(t)$ to follow an error signal between a target and the RNN trajectory, thus providing a "closed-loop" learning rule. Most generally, however, we remain agnostic as to the meaning or information content of $\mathbf{I}(t)$, and focus on its effects on connectivity and dynamics produced via PRISM plasticity.

Theory of the learned flow fields produced through instructive signals

Flow fields are visual representations of a dynamical system depicting the magnitude and direction of forces $d\mathbf{x}/dt$ at different states, and which can be used to understand the system geometrically[55,56]. How do the instructive signals $\mathbf{I}(t)$, which update the weights W through the PRISM rule of Eq. (3), in turn update the flow field described by Eq. (1)? Linking RNN connectivity to dynamics and flow fields is a longstanding challenge in neuroscience[35,57-59], and at first glance it may not seem that considering only connectivity and dynamics produced through PRISM plasticity would simplify this problem. Here, however, we show that the flow fields created through the PRISM rule can be expressed in a concise mathematical form precisely analogous to the "dual form" of Support Vector Machines—a widely studied and well-understood nonlinear machine-learning model—revealing a clear geometric interpretation of the learned dynamics.

Mathematical form of the learned dynamics

For simplicity, let the initial weights in the feedback circuit be zero ($W = 0$), and suppose the system is driven by inputs $\mathbf{u}(t)$, e.g. sensory stimuli, and subject to instructive signals $\mathbf{I}(t)$. We first consider point-process instructive signals expressed as a train of weighted impulses or delta functions,

$$\mathbf{I}(t) = \sum_l \boldsymbol{\alpha}_l^* \delta(t - t_l^*), \tag{4}$$

where t_l^* is the time of the l-th instructive signal impulse, and $\boldsymbol{\alpha}_l^* \in \mathbb{R}^Q$ is a vector quantity giving the strength of each instructive signal component at the time of the impulse. We assume that all instructive signals are presented to the network during a bounded *instruction period*, after which $\mathbf{I}(t)$ is set to 0. Time-integrating the PRISM rule (Eq. (3)) and substituting the resulting weight

matrix W into Eq. (1) reveals (see Extended Methods for derivation) that the learned dynamics following the instruction period can be written as

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + J^{base}\phi(\mathbf{x}) + \sum_l C\boldsymbol{\alpha}_l^* k(\mathbf{x}, \mathbf{x}_l^*) + B\mathbf{u}(t), \tag{5}$$

where the weight matrix W no longer appears and has been replaced by a summation over the instructive signals. The quantity $k(\mathbf{x}, \mathbf{x}_l^*)$ is a *kernel* function[46] specifying the effective similarity between state $\mathbf{x}$ and state $\mathbf{x}_l^*$ via the overlap of their barcodes, $\boldsymbol{\psi}(\mathbf{x})^T\boldsymbol{\psi}(\mathbf{x}^*)$. Note that the final learned dynamics depend only implicitly, and not explicitly, on the timecourse of $\mathbf{u}(t)$ during the instruction period, which we discuss shortly. Eq. (5) is directly analogous to the "dual form" of SVMs, in which test outputs or predictions are expressed not in terms of weights, but in terms of how similar or near a test input is to a set of training inputs called *support vectors*[46]. In our model, the flow field $d\mathbf{x}/dt$ at an arbitrary state $\mathbf{x}$ is expressed in terms of how similar or near $\mathbf{x}$ is to a set of *support states* $\{\mathbf{x}_l^*\}$. These support states are exactly the states of the base RNN when the instructive signals occurred: $\mathbf{x}_l^* \equiv \mathbf{x}(t_l^*)$. In analogy to SVMs, we term Eq. (5) the *dual form* of the learned dynamics. This dual form precisely links the learned high-dimensional network connectivity W to the resulting D-dimensional dynamics, which can be understood geometrically.

Geometric consequence of adding support states

The effect of an instructive signal impulse adding a support state can be understood geometrically by visualizing the corresponding change in the flow field. If we assume $J^{base} = 0$ and $W = 0$ initially, then prior to the instruction period the flow of the dynamics linearly draws all states back to the origin, due to the leak term $-\mathbf{x}$ in Eq. (5) (Fig 2a). Now drive the system with a time-varying input $\mathbf{u}(t)$ during the instruction period, which will move around the base RNN state. Applying a single instructive signal impulse at time t_1^* during this period adds a single support state $\mathbf{x}_1^* = \mathbf{x}(t_1^*)$ (Fig 2b). In accordance with Eq. (5) the resulting flow field becomes the sum of the base flow field $-\mathbf{x}/\tau$ and the patch of flow field, $C\boldsymbol{\alpha}_1^* k(\mathbf{x}, \mathbf{x}_1^*)/\tau$, contributed by $\mathbf{x}_1^*$. In the example of Fig 2c-d, the support state creates a second fixed point, revealing the capacity of PRISM plasticity to produce nonlinear dynamics, even when the base dynamics are fully linear. Thus, instructive signals add support states through the PRISM rule (Fig 2a-d), which concisely, yet precisely, describe the influence of the updated feedback circuit on the system dynamics.

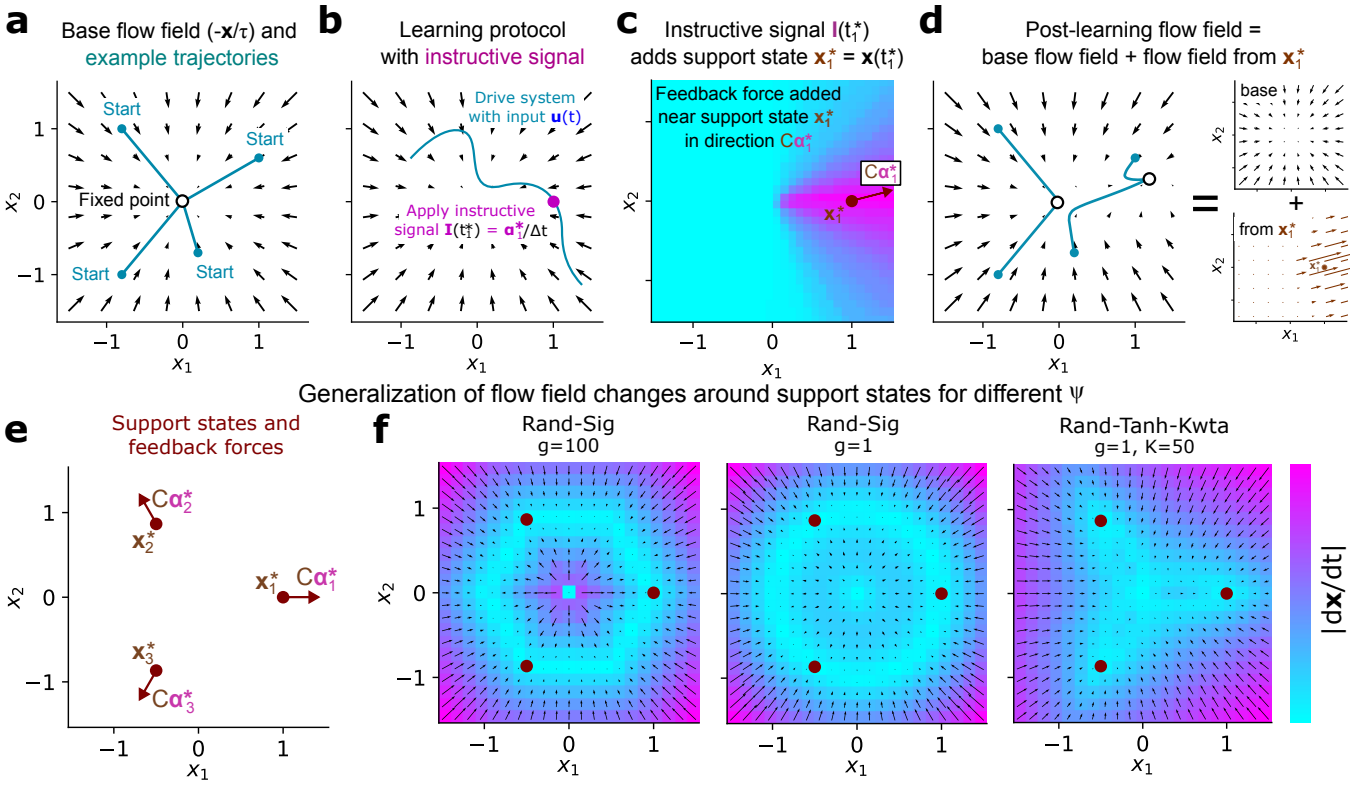

Figure 2: Flow fields produced by instructive signals.
a. Example base flow field in which activity decays through uncoupled linear dynamics to a stable fixed point at the origin. **b**. Schematic of applying a single instructive signal impulse while the system is being driven by external input $\mathbf{u}(t)$. **c**. Schematic of the effect of the instructive signal in **b**, which is to add a support state at $\mathbf{x}_1^* \equiv \mathbf{x}(t_1^*)$, whose influence on the learned dynamics spreads according to $k(\mathbf{x}, \mathbf{x}_1^*)$. **d**. Schematic of final flow field after instruction period, which is a sum of the base flow field and the flow field contributed by the support state $\mathbf{x}_1^*$. **e**. Schematic of three support states added to a 2-D state space with the same base dynamics as **a**. **f**. Flow field produced by the three support states in **e** for three different choices of ψ.

Note that to compute the final flow field it does not explicitly matter *when* the instructive signal was applied or what inputs were presented during the instruction period—the only information retained explicitly in the learned flow field equation is *where* the system was in state space at t_1^* and the (vector) value $\boldsymbol{\alpha}_1^*$ of the instructive signal. Multiple instructive signal impulses add multiple support states, each contributing its own patch of flow field around the support state, which is added to the sum in Eq. (5). (This statement holds for the PRISM rule in Eq. (3), but see Extended Methods for the generalization to temporally extended plasticity windows.) For continuous, rather than point-process instructive signals, the sum in (5) is replaced by an integral and the *set* of support states by a support state *density* (Extended Methods).

Generalization of the learned dynamics beyond the support states

Crucially, the manner in which the learned dynamics generalize to states not visited during the instruction period, or in which no instructive signal was applied, is determined by the kernel k, which defines the effective similarity between two states and in turn how a support state influences the dynamics at an arbitrary state. For instance, three support states optimized to produce three specific fixed points create different patterns of surrounding fixed and slow points when different $\psi(\mathbf{x})$ are used, since these yield different kernels (Fig 2e-f). Thus, how the learned flow field generalizes across the total state space can be controlled or predicted in a precise,

straightforward manner through the choice of $\boldsymbol{\psi}$. Certain kernels, e.g. for the $N, g \to \infty$ limit of the Rand-Sig nonlinearity (Eq. (2)), additionally admit a closed form that does not depend on J^{ψ} (Extended Methods), such that the dual form (Eq. (5)) depends on neither W nor J^{ψ}. In this case, Eq. (5) reveals a concise, exact mathematical relationship between network connectivity of the complete RNN and the corresponding D-dimensional flow fields.

Our theory enables to precisely predict how sequences of instructive signals will update the dynamics flow field through the PRISM rule, even before we run the simulation forward, as long as we know the system state at the time of each instructive signal. Using this theory, we now address which flow fields can be produced through instructive signals $\mathbf{I}(t)$, and how the latter should be chosen to learn specific target dynamics.

One-shot emulation of known target dynamics

Which dynamical systems can be expressed in terms of support states, and what patterns of instructive signals $\mathbf{I}(t)$ and external inputs $\mathbf{u}(t)$ can teach the RNN to produce these dynamics? We hypothesized that, unlike Hebbian plasticity—which produces dynamics largely limited to specific fixed-point attractors and sequences[2,6,9,60]—the flexibility of instructive signal timing combined with the observation that each instructive signal impulse adds a new patch of flow field would allow our model to produce a wide variety of nonlinear flow fields. In this section, we first show that a large family of flow fields can be expressed in terms of support states, and then that these support states can be added online via suitable inputs and instructive signals.

To test our hypothesis, we considered the $N, g \to \infty$ limit of the Rand-Sig nonlinearity (Eq. (2)), which we term the Rand-Sig-∞ nonlinearity, and whose associated kernel function can be concisely expressed (Extended Methods) as

$$k(\mathbf{x}, \mathbf{x}^*) = 1 - \theta(\mathbf{x}, \mathbf{x}^*)/\pi, \tag{6}$$

where θ is the *angle* between $\mathbf{x}$ and $\mathbf{x}^*$. Combining Eq. (5) and (6), the corresponding dual form for the dynamics is:

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + J^{base}\phi(\mathbf{x}) + \sum_l C\boldsymbol{\alpha}_l^* \left(1 - \theta(\mathbf{x}, \mathbf{x}^*)/\pi\right) + B\mathbf{u}(t). \tag{7}$$

According to this equation, for the Rand-Sig-∞ nonlinearity, the learned component of the flow field does not depend on the magnitudes of $\mathbf{x}$ and $\mathbf{x}^*$, but only on θ. Hence, this learning rule and nonlinearity cannot produce changes to the flow field that vary with distance from the origin (however, the *total* flow field can still vary with distance to the origin due to the base dynamics). We verified this using ridge regression to identify loadings $\{\boldsymbol{\alpha}_l^*\}$ to produce target flow fields that did and did not satisfy this criterion (Fig S1). However, this restriction can be lifted by adding random biases in the nonlinearity $\boldsymbol{\psi}$ (Fig S1).

Even with this restriction, the flow fields that can be constructed in terms of support states with the Rand-Sig-∞ $\boldsymbol{\psi}$ are highly flexible. This is because each term in the sum in Eq. (7) acts like a basis function over the unit sphere in D dimensions, which can be combined arbitrarily combined. When C is the identity, for instance, then for $D = 2, 3$, any function on the sphere that can be expressed as the sum of a constant and an odd function can be constructed from support states, with a similar result likely holding for $D > 3$ (Extended Methods). To illustrate this point, we used ridge regression to identify support states and associated $\{\boldsymbol{\alpha}_l^*\}$ that can produce a variety of flow fields, including multiple fixed points, limit cycles, and an approximate line attractor, corroborating our mathematical result (Fig S2). Thus, support states can be combined to produce a wide variety of nonlinear dynamics thought to subserve neural computation[61-63].

While the preceding ridge regression analyses show how flow fields can be decomposed in principle into support states, they do not show how these support states could be added online. We therefore asked whether such flow fields could be produced through appropriately patterned inputs $\mathbf{u}(t)$ and instructive signals $\mathbf{I}(t)$ via Eq. (3). If inputs and instructive signals exist that can add generic support states to the system, this would suggest that an animal could learn to "emulate" a target dynamical system in a brain area through sufficient $\mathbf{u}(t)$ and $\mathbf{I}(t)$. On the other hand, one might imagine that certain combinations of support states are unreachable through inputs and instructive signals.

We found that for identity B and C, and leveraging the radial invariance of the Rand-Sig-∞ kernel (Eq. (6)), we could construct a protocol for adding arbitrary support states through PRISM plasticity (Fig 3a). First, in an "offline" step (which we do not take to be biological) identify support states $\{\mathbf{x}_l^*\}$ and loadings $\{\boldsymbol{\alpha}_l^*\}$ needed to produce a target flow field, e.g. using ridge regression. Second, perform the following: for each support state $\mathbf{x}_l^*$ and loading $\boldsymbol{\alpha}_l^*$, drive the system with a large input in the same direction as $\mathbf{x}_l^*$, forcing the system into a state angularly close to $\mathbf{x}_l^*$; then apply an instructive signal impulse to trigger the PRISM rule and add the support state. Because support states at the same angular position are all equivalent (Eq. (7)), this provides a recipe for adding effectively arbitrary support states. Thus, any flow field expressible in terms of support states can emerge through appropriately timed inputs and instructive signals (Fig 3a).

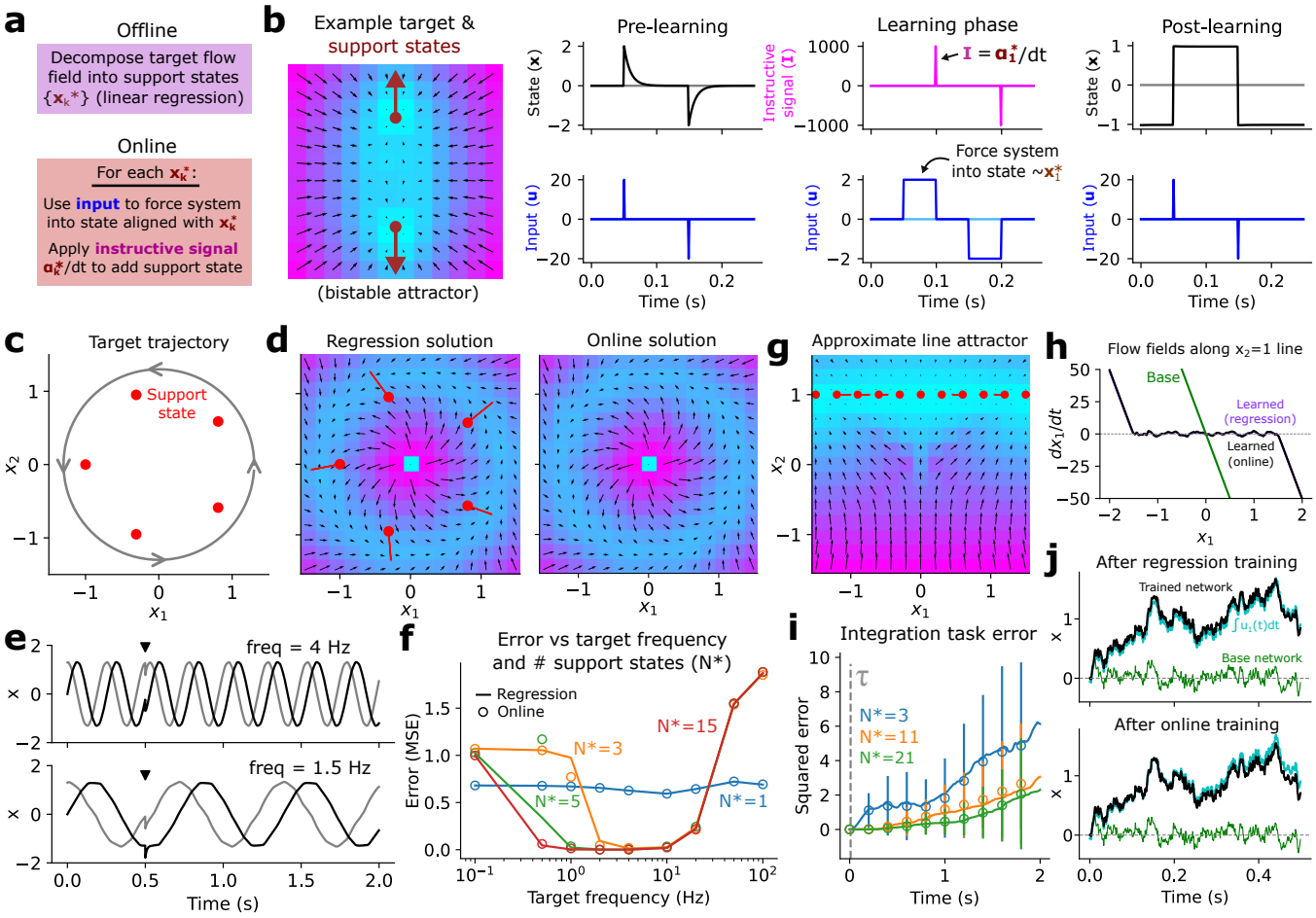

Figure 3: One-shot learning of known target dynamics

a. Offline-online algorithm for designing inputs $\mathbf{u}(t)$ and instructive signals $\mathbf{I}(t)$ to emulate a known target flow field. **b**. Example application of the algorithm in **a** to create a flow field with two fixed-point attractors. **c**. Schematic of flows/velocities to be learned along a specific trajectory by combining 5 support states. **d**. Left: regression solution (identifying $\boldsymbol{\alpha}_1^*, \ldots, \boldsymbol{\alpha}_5^*$) to approximate target flow in **c**. Right: solution produced by applying the online portion of the

algorithm. **e**. Activity time-courses for $\mathbf{x}(t)$ when the system is trained to produce either of two example limit cycles. The black triangle shows the time of a small perturbation, indicating that the dynamics are stable with respect to this perturbation. **f**. Error between target and learned activity time-courses $\mathbf{x}(t)$ as a function of the target limit cycle frequency and the number of support states. **g**. Flow field and support states reproducing an approximate line attractor. **h**. Dynamics along the $x_2 = 1$ line in **g** of the system trained to produce a line attractor that is used to integrate $u_1(t)$. The system is driven with an additional constant input $u_2(t) = 1$ during this task. **i**. Error in using the trained system to perform integration along the line attractor, vs elapsed time and the number of support states afforded. **j**. Example of the trained network integrating its inputs.

We tested this procedure by training our network model to emulate three different example systems with known dynamics. First, we considered a target system with two stable fixed points. Fig 3b demonstrates an application of the offline-online algorithm (Fig 3a) to construct the inputs and instructive signals needed to produce this system given no base dynamics beyond linear decay to the origin. As predicted, the system learned the target dynamics after only two instructive signal impulses, allowing the network to subsequently act as a 1-bit memory.

Second, we asked whether the network could learn to produce a limit cycle at a specific target frequency. Of note, during the offline step, we can identify support states needed to produce only flows/velocities at specific states, rather than the entire flow field—for instance the flows along a circular trajectory (Fig 3c). We were curious, however, whether the learned dynamics around such a trajectory would be stable or unstable to perturbations, even if we taught the RNN to only reproduce velocities on the trajectory itself. We tested this by using the above procedure to teach the RNN to emulate the flow velocities along a circular trajectory with a target frequency ν (Fig 3c). As expected, we found that the trajectory velocities could be learned via inputs and instructive signals (Fig 3d). We also found, however, that the resulting dynamics were stable around the target trajectory, producing a limit cycle at the target frequency (Fig 3e). The range of frequencies of stable limit cycles increased with the number of support states or instructive signal impulses (Fig 3f). Thus, presenting inputs and instructive signals to produce a circular trajectory automatically stabilized the dynamics into a limit cycle across a range of frequencies.

As a third example, we asked whether our protocol could produce a line attractor—a canonical model of a neural dynamical motif supporting integration[33,61,64]. Indeed, our protocol could produce an approximate line attractor capable of integrating inputs over a timescale much longer than the leak timescale τ (Fig 3g-j). Note, however, that for a finite number of support states the line attractor is not perfect. First, it is localized to a specific section of phase space, hence cannot integrate beyond its bounds (Fig 3h); second, it is composed of a microscopically rough landscape of slow and fixed points (Fig 3h). As a result, the inputs to be integrated need to be strong enough to drive the system around this landscape, but not so large as to escape the bounds of the attractor (Fig S3). Similar to the limit cycle example (Fig 3c-f), the stability of the line attractor against perturbations orthogonal to its longitudinal axis emerges automatically, even though we did not explicitly enforce this. Intuitively, this stability as well as that of the limit cycle arise from competition between the patches of flow field added by the instructive signals and the base flow field $-\mathbf{x}/\tau$.

Thus, a rich variety of flow fields can be produced given proper inputs $\mathbf{u}(t)$ and instructive signals $\mathbf{I}(t)$. Moreover, this rule is highly efficient, requiring only sparse instructive signals—one impulse per support state to be added—yet which can qualitatively change the behavior of the RNN. Such a system is therefore well-posed for rapid or one-shot learning to emulate flexible dynamical systems, and may be highly resource-efficient if instructive signals are costly.

Online learning of unknown dynamics through real-time error signals

We next asked whether our RNN could learn effectively through PRISM plasticity without having access to the target dynamics *a priori*. To address this we considered an online learning scheme in which a separate, reference system exists whose full dynamics are unknown but whose state $\mathbf{z}(t)$ can be observed during the instruction period. We asked whether an instructive signal could be derived from observations of the reference system state that would teach a learner RNN governed by Eq. (1)–(3) to eventually approximate the activity dynamics of the reference system without further instruction.

Leveraging tools from adaptive control theory (Extended Methods), we found that this was indeed the case. Suppose the true reference system follows dynamics

$$\tau \frac{d\mathbf{z}}{dt} = \mathbf{f}_{ref}(\mathbf{z}) + B\mathbf{u}, \tag{8}$$

for some function $\mathbf{f}_{ref}$ unknown to the learner RNN. Let the learner RNN follow the dynamics

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + \mathbf{f}_{base}(\mathbf{x}) + CW^{\top}\boldsymbol{\psi}(\mathbf{x}) + B\mathbf{u} + \mathbf{c}, \tag{9}$$

where $\mathbf{f}_{base}$ is e.g. $J^{base}\phi(\mathbf{x})$, and which is equivalent to Eq. (1), except modified to explicitly include an additional input $\mathbf{c}$, which we call a "corrective signal". Specifically, let $\mathbf{c} \equiv \kappa\mathbf{e}$, where $\mathbf{e}(t) \equiv \mathbf{z}(t) - \mathbf{x}(t)$ is the tracking error between the reference and learner system state and κ is a positive "control gain". Let the weight updates and instructive signal follow

$$\frac{dW^{\top}}{dt} = \mathbf{I}(t)\boldsymbol{\psi}(\mathbf{x}(t))^{\top}, \text{ where } \mathbf{I}(t) \equiv C^{\top}\mathbf{e}(t). \tag{10}$$

Note that the instructive signals here are continuous, rather than a point process. A schematic of the complete architecture is shown in Fig 4a.

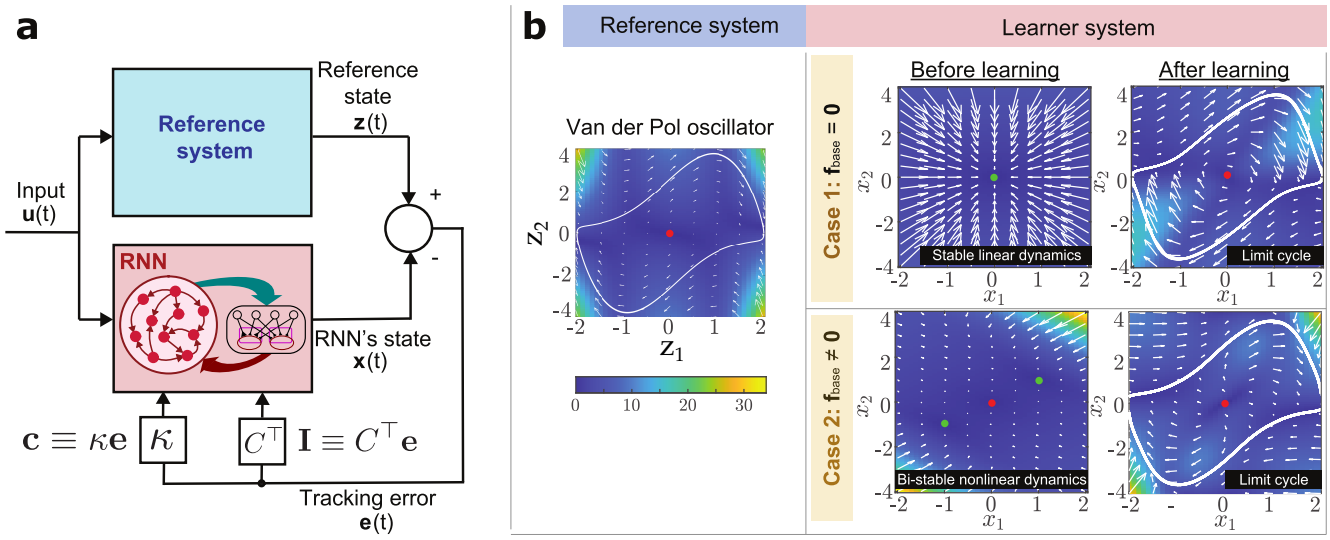

Figure 4: Learning to emulate an unknown system through real-time error signals
a. Architecture of learner RNN guided by error signals from a reference system. A more detailed block diagram is provided in Fig S4. The reference system and learner RNN both receive the same input $B\mathbf{u}$. The difference between the reference's and RNN's states defines the tracking error $\mathbf{e}$, which is used to generate the *corrective signal* $\mathbf{c} = \kappa\mathbf{e}$ and the *instructive signal* $\mathbf{I} = C^{\top}\mathbf{e}$ that are fed back into the RNN. **b.** Numerical case studies. The reference system is a Van der Pol oscillator, which has a stable limit cycle and an unstable fixed point at

the origin. Case 1 has $\mathbf{f}_{base} \equiv \mathbf{0}$ and exhibits the dynamics of a strictly stable linear system (due to the leakage term) before learning. In case 2, the base RNN dynamics support a bi-stable attractor. In both cases, the RNN successfully changes its local flow into stable limit cycle dynamics after the instruction period. Stable (unstable) fixed points are indicated by green (red) dots. Color-code of the flow fields is preserved across panels and indicated by the colorbar.

Our goal is for the learner RNN to learn to eventually reproduce the reference dynamics $\mathbf{f}_{ref}$ without requiring access to $\mathbf{z}$ or any further instruction. Thus, after the instruction period, we set $\mathbf{I}$ and κ to zero, ceasing learning and letting the RNN behave independently of the reference system. The learning framework presented in this section is grounded on the theory of adaptive control[65,66], specifically Model Reference Adaptive Control (MRAC). Under certain model matching assumptions and appropriate selection of the value of the gain κ, convergence of the RNN's state to that of the reference system can be asymptotically guaranteed using Barbalat's lemma[67] or through contraction theory[68-70]—methods used for the analysis of non-autonomous nonlinear dynamical systems. Further, selection of the inputs $\mathbf{u}$ to guarantee "persistency of excitation" conditions (intuitively, guaranteeing sufficient exploration of the state space) ensures learning of appropriate weight values[71,72]. Note that while the learner observes the reference system state, the reference receives no information from the learner. Such additional information exchange could make learning active and hasten learning rates; however, for simplicity we do not include it here, and it is not necessary for learning.

To validate the ability of the instructive signals (Eq. (10)) to teach the RNN to emulate the reference system, we numerically explored two illustrative case studies. In both cases, the reference dynamics to be emulated were described by a Van der Pol oscillator (Fig 4b, left), a 2-D nonlinear system that generates relaxation oscillations (Extended Methods). First, we analyzed the setting in which $\mathbf{f}_{base} \equiv \mathbf{0}$ (i.e. linear uncoupled base dynamics decaying to the origin), while in the second case we consider pre-existing base dynamics supporting a bi-stable attractor, reflecting a scenario in which an existing learned flow field must be overwritten in order to emulate the limit cycle of the reference Van der Pol oscillator. We found that, regardless of the base dynamics considered, the RNN successfully learned a Van-der-Pol-like limit cycle resembling the reference system after the instruction period (Fig 4b, right). Thus, the corrective and instructive signals were sufficient to teach the network to emulate the reference dynamics through real-time feedback.

As here the instructive signals are continuous rather than a point process, the dual form of the learned dynamics after the instruction period requires a slight modification from Eq. (5). In particular, here the dual form of the dynamics is represented not in terms of individual support states, but via a support state *density* (Fig S7; Extended Methods). In other words, the sum in Eq. (5) is replaced by integral of the product of k with the support state density, once again revealing the flexibility of dynamics and flow fields that can be produced in this manner. Thus, through PRISM plasticity, instructive signals support both the online, one-shot implementation of known target dynamics (Fig 3), as well as the ability to learn online to emulate a reference system through real-time error signals (Fig 4). In both cases, the set of flow fields that can be produced by the learning rule is extremely flexible, due to the basis-function-like nature of the kernel function—the central limitation is the set of states that can be accessed during the instruction period, which determines where support states can be added.

We note the related idea of online internal model adaptation through error feedback studied in motor control. For example, it has been proposed that the cerebellum contains internal inverse and forward models of the motor apparatus[73]. The inverse models provide the neural commands necessary to achieve a desired motor trajectory or state and are interpreted as feedforward controllers. Forward internal models predict the behavior of the motor system in response to motor commands and can be leveraged, for instance, to compensate for time delays

in feedback control[74]. Further, these models can in principle be learned within the cerebellum via different training signals: inverse models through motor command errors[73,75,76] and forward models through sensory prediction errors[73]. Thus, our results on instructive signals teaching the RNN to emulate reference dynamics are consistent with existing ideas in cerebellar motor control, while providing additional theoretical support and generality.

Optimizing the instructive signals for long-range credit assignment

Finally, we asked whether instructive signals and PRISM plasticity could be leveraged to train RNNs on tasks involving long-range temporal credit assignment, which challenges traditional RNN training methods. Canonical RNN training in machine learning, typically based on back-propagation through time (BPTT), updates the RNN weights to minimize a loss function by descending the loss gradient[39,77]. However, while BPTT is both flexible and admits biological approximations[78,79], training in the face of delayed loss evaluation is sensitive to "vanishing (or exploding) gradients"—these arise due to the repeated multiplication of the weight matrices across timesteps when calculating loss gradients[48,80]. In practice, this hinders temporal credit assignment between outputs that are separated from their causal inputs by long delays.

Here we give a proof of concept demonstrating how instructive signals can be used to address this problem by avoiding vanishing/exploding gradients. In our proposed method, which we term "instructor training", we do not optimize the weights directly but instead optimize the instructive signal timecourse. For simplicity, here we consider the discretized version of our RNN, in which $\mathbf{x}(t)$, $\mathbf{I}(t)$ are replaced by $\mathbf{x}^t, \mathbf{I}^t$, etc. Given a fixed input sequence $\mathbf{u}^t$, each candidate instructive signal timecourse $\mathbf{I}^t$ produces a trajectory of the weight matrix W^t according to the PRISM rule, and which after the instruction period yields a final, fixed W. We then optimize $\mathbf{I}^t$ using the same inputs and delayed loss function as one would in BPTT. Our final goal is to be able to run the initial plastic RNN forward (Eq. (1)–(3)) while applying the optimized instructive signals, such that the final weights W at the end of the instruction period allow the RNN to solve the task with no further instruction. Identifying task-optimized instructive signals could also provide a means to predict instructive signal dynamics in animals throughout task learning[81].

Crucially, optimizing the instructive signal timecourse $\mathbf{I}^t$ can converge to at least two possible outcomes. First, the optimal instructive signals could remain strong throughout the course of learning, corresponding to a case in which the network learns to rely on the instructive signals to solve the task, thus never learning to perform the task independently. More favorably, the optimal instructive signals could decrease over learning, i.e. with the network initially relying on the instructive signals but by the end of the instruction period only relying on the weights W—in this case the network learns to solve the task independently without further instruction. We hypothesized that it was possible for the second case to emerge automatically simply by optimizing the instructive signal timecourse directly, even without imposing an explicit decay of their magnitude.

We tested our hypothesis on a proof-of-concept task with loss evaluated at a variable delay. That is, the loss depends only on the network activity occurring after the input has ceased (Fig 5a). To illustrate the essential challenge of long-range temporal credit assignment, we considered a parsimonious "hold" task with a delayed loss evaluation (Fig 5b). Here, in response to an input pulse in the $[-1, 0]$ direction, the network should respond by holding the first neuron's activity x_1 at -1, and vice versa for an input pulse in the $[+1, 0]$ direction. Loss evaluation (the error between the true x_1 and the target), however, only occurs after a variable delay following each input. As shown in Fig 5c-d, both BPTT[39,77] and e-prop (a biological version of BPTT)[79] succeed for short delays (on the order of the network time constant τ) but fail for longer delays, as expected from the vanishing gradient problem[48].

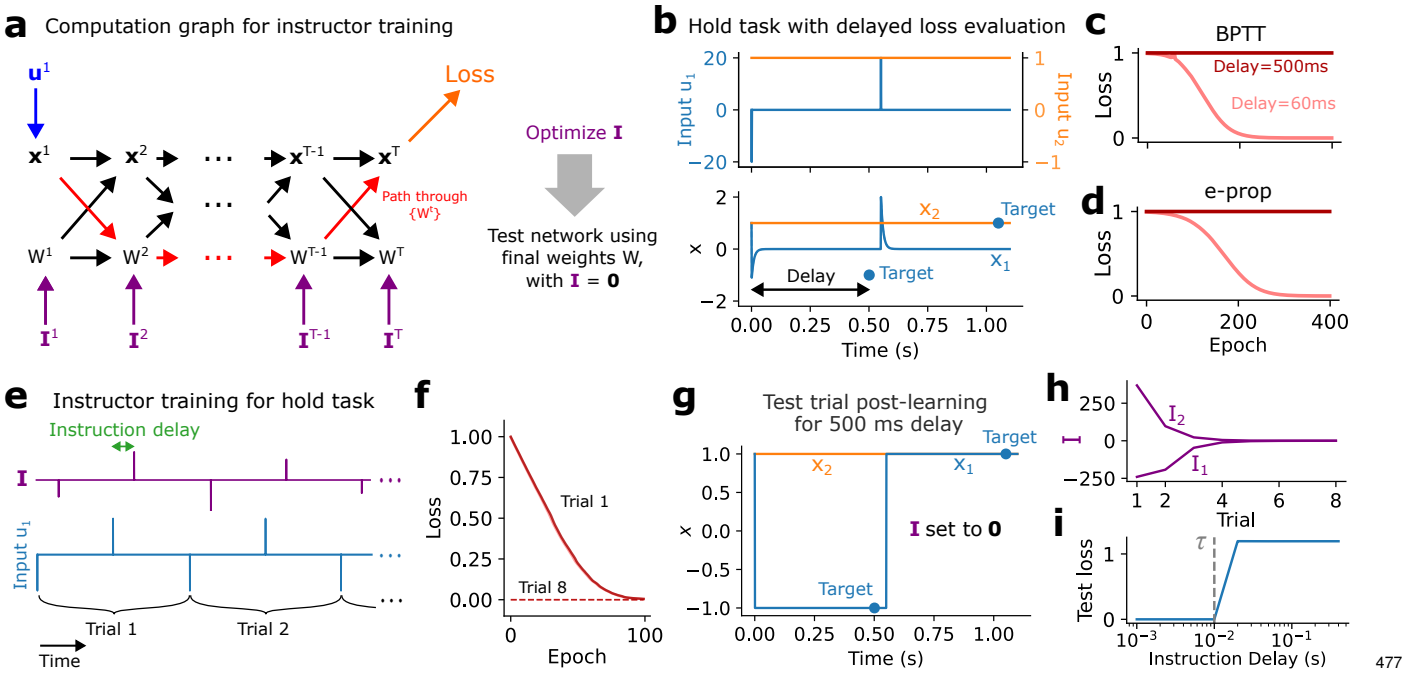

Figure 5: Instructor training proof-of-concept
a. Computation graph for instructor training. **b**. Example task in which loss is evaluated only after a delay, which is expected to trigger the vanishing gradient problem for BPTT when the delay is too long. The goal is for x_1 (blue) to pass through each of the targets. **c, d**. Loss vs training epoch for the hold task in **b** for BPTT and e-prop for two different delays. **e**. Schematic of instructor training paradigm used for the hold task. Here the magnitudes of the instructive signal impulses are optimized. **f**. Loss vs epoch for each trial in the instructor training paradigm of **e**, given a 500 ms delay. **g**. Network performing hold task after instructor training, and with the instructive signal silenced. **h**. Magnitude of each of the two instructive signals vs trial number. **i**. Test loss (corresponding to the test trial in **g**) as a function of the instruction delay.

We found, however, that instructor training could solve this task, even with long delays, and moreover without learning to rely on the instructive signal by the end of training. Specifically, we considered 8 sequential "trials" of the hold task, each containing two inputs, with one instructive signal impulse allowed after each input (Fig 5e). This represents an approximate version of optimizing the complete instructive signal timecourse. We then optimized the amplitudes of the instructive signals for each trial, using the resulting weights W at the end of each trial as the initial weights for the subsequent trial. Indeed, for a long delay (500 ms, which neither BPTT nor e-prop can handle) the loss decreased over both training epochs within a trial and over trials. (Note also that grouping BPTT or e-prop training epochs into trial groups does not help them solve this task.) Importantly, in a test trial in which no instructive signals were included, the network still solved the task (Fig 5g), indicating that it had learned to rely on the weights rather than the instructive signals. Indeed, the magnitude of the optimized instructive signals decreased over trials (Fig 5h). Finally, we found that in order for the network to learn to solve the task without relying on the instructive signals it was necessary for the instructive signals to occur shortly after the inputs (Fig 5i).

Instructor training succeeds for the following reason. Formally, loss gradients needed for optimization via gradient descent are computed by summing over all paths through the computation graph from the network parameters to the loss (Fig 5a, S5), with each path corresponding to a product of Jacobians (the matrix of partial derivatives of the output with respect to the input variables) along that path[82]. In BPTT, however, the paths linking inputs to loss evaluation through the network states $\{\mathbf{x}^t\}$ often "vanish" or "explode" if there is a long delay between the

input and loss evaluation[48,80], due to the repeated multiplication of the Jacobians in the gradient calculation. In instructor training, however, the weights are themselves dynamic variables, opening additional paths through the computation graph—i.e., through $\{W^t\}$—for information from past inputs to reach the loss. Because the PRISM rule (Eq. (3)) has no decay time constant, this allows information to persist in the weights over long time periods, which in turn allows gradient information to propagate through the computation graph without vanishing or exploding. Intuitively, the optimization process learns that an instructive signal can be invoked to store information about the inputs in W until it is needed again; at the same time, however, it learns over sequential trials to transfer this information storage ability into the dynamics encoded in W.

Thus, instructor training represents an alternative to canonical gradient methods for end-to-end training of RNNs. In both BPTT and instructor training, the final result is an RNN with fixed weights W. Instructor training, however, leverages instructive signals to learn long-range dependencies during the training process, even though at test time no instructive signals or plasticity are required. Note that we do not treat the process of optimizing the instructive signals as biologically plausible; however, the optimal instructive signals reflect a biological solution to updating the weights if those instructive signals could be generated in the brain. These results thus suggest a biologically inspired solution to a long-standing problem in machine learning.

DISCUSSION

We have shown how biologically inspired PRISM plasticity rules mediated by pre-synaptic activity and instructive signals can rapidly produce highly flexible nonlinear network dynamics. This work represents a complement to the theory of Hebbian plasticity's ability to produce fixed-point attractors or sequences that can be used for pattern completion[2,5,6,9]. Instructive signals, when applied to a plastic feedback circuit coupled to a base RNN, add support states to the dynamics flow field that can be combined to yield arbitrary nonlinear dynamics. A notable feature of the learning system we have proposed is that addition in the space of synaptic weights (via Eq. (3)) corresponds directly to addition in the flow field (Eq. (5))—thus, PRISM plasticity applied to feedback circuits reveals a simple, interpretable link between synaptic weights and flow fields, addressing the long-standing challenge of linking RNN connectivity and dynamics[34,35,47,57]. As PRISM plasticity also accords more strongly with experimental data on rapid learning than Hebbian plasticity[15-17,23,42], our model may additionally be more applicable to explaining empirical neural dynamics.

Our work extends a growing body of work on training RNNs with biological learning rules. One avenue of past research, encompassing the "RFLO"[78], "e-prop"[79], and MDGL[83] algorithms, focused on how loss gradients can be approximated by local cellular processes in a single RNN. A second avenue of research has focused on dynamics and learning in RNNs coupled to a plastic feedback circuit analogous to a cerebellum[43] or mushroom body[42], or more generically an adaptive feedback controller[66,84], similar to our model architecture (Fig 1c); however, while such RNNs can solve interesting tasks in a more biologically plausible manner than traditional training algorithms like BPTT, the mathematical relationship between the learning rule, the networks weights, and the resulting flow fields has remained elusive. Although some theoretical work has established important links between network connectivity and flow fields[35,85], these studies have largely not considered connectivity produced by rapid biological plasticity rules. Our work synthesizes the above by providing a unifying framework and concise mathematical theory that links network connectivity produced through rapid biological learning rules to interpretable nonlinear dynamics and flow fields. Our study also suggests an RNN algorithm—instructor training—that can in certain cases solve long-range temporal credit assignment problems by overcoming the vanishing/exploding gradient problem, which hinders learning in canonical algorithms[48] unless

additional slow variables are introduced that can be used at test time[79,86].

Our work represents a simplified model of how biological plasticity rules shape neural dynamics to support rapid learning and remains subject to important limitations. We have largely considered a plasticity rule in which weight changes are triggered by precise coincidence between presynaptic activity and the instructive signal, whereas empirical plasticity rules in hippocampus, cerebellum, and mushroom body exhibit various extended temporal windows of coincidence detection[15–17,23]. The dual form of the learned dynamics (Eq. (5)) can be generalized to accommodate such temporally extended windows (Fig S6, Extended Methods), but in future work it will be important to establish in detail how such modifications influence learning. Additionally, in all of our examples we have considered the case in which the (columns of the) input and feedback connections, B and C, respectively, span the entire space $\mathbb{R}^D$ of the base dynamics. In general, however, the columns of B and C might span only subspaces of $\mathbb{R}^D$, and it will be important to characterize which dynamics and tasks can be learned under these constraints. Finally, it will be useful to explore how different base dynamics $\mathbf{f}_{base}$ impair or accelerate learning, and to characterize in more detail the tasks that can be solved via instructor training.

Our theory makes testable general predictions, while also opening an avenue for fitting to data to make predictions in specific experimental contexts. The first general prediction is that activating an instructive signal will alter the neural dynamics in other brain areas predominantly around the system state these dynamics were in at the time of the instructive signal—the support state. This could be tested, for instance, by analyzing neural activity in cortical areas while stimulating climbing fibers inputs to cerebellum or perforant pathway inputs to hippocampus. Second, many different sequences of instructive signals can ultimately produce the same or similar flow fields—thus, for animals that all learn a task successfully, we predict there to be substantial heterogeneity in the instructive signals evoked throughout learning, even if the learned neural dynamics are similar across animals[87]. Finally, our model can in principle be fit to empirical neural population activity trajectories by identifying support states that would produce dynamics consistent with those trajectories. Such an approach could be used to make predictions about unseen neural responses in regions of state space not visited by the training data, benefiting from the inductive bias encoded by the kernel function.

METHODS

Methods details and mathematical derivations are available in the Extended Methods section accompanying this manuscript.

RESOURCE AVAILABILITY

Lead contact

Requests for further information and resources should be directed to and will be fulfilled by the lead contact, Rich Pang (rkp.science@gmail.com).

Data and code availability

- This project generated no new datasets.
- All code used to generate the figures in this manuscript is available at github.com/rkp8000/prism-rnn.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

This work was funded by The Princeton Neuroscience Institute Innovation Award (to RP and JWP), The Princeton Neuroscience Institute C.V. Starr Award (to JA), the NIH BRAIN initiative (R01EB026946; to JWP), the NIMH (R01EY033064; to JWP), a U19 NIH-NINDS BRAIN Initiative Award (5U19NS104648; to JWP), and the Simons Collaboration on the Global Brain (SCGB AWD543027; to JWP). We thank Hakan Tureci for insightful discusions during the development of this project, as well as Anushri Arora, Victor Geadah, and other members of the Pillow lab for many helpful thoughts and conversations.

AUTHOR CONTRIBUTIONS

Conceptualization, R.P., J.A., and J.P.; methodology, R.P., J.A., and J.P.; investigation, R.P. and J.A.; writing-–original draft, R.P. and J.A.; writing-–review & editing, R.P., J.A., and J.P.; funding acquisition, R.P., J.A., and J.P.; resources, R.P., J.A., and J.P.; supervision, J.P.

DECLARATION OF INTERESTS

The authors declare no competing interests.

SUPPLEMENTAL INFORMATION INDEX

Figures S1-S7 and their legends in a PDF

References

1. Hebb, D.O. (1949). The organization of behavior: A neuropsychological theory. Wiley.

2. Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences *79*, 2554–2558.

3. Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. Science *275*, 213–215.

4. Bi, G.q., and Poo, M.m. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. Journal of Neuroscience *18*, 10464–10472.

5. Fiete, I.R., Senn, W., Wang, C.Z., and Hahnloser, R.H. (2010). Spike-time-dependent plasticity and heterosynaptic competition organize networks to produce long scale-free sequences of neural activity. Neuron *65*, 563–576.

6. Brunel, N. (2016). Is cortical connectivity optimized for storing information? Nature neuroscience *19*, 749–755.

7. Whittington, J.C., Muller, T.H., Mark, S., Chen, G., Barry, C., Burgess, N., and Behrens, T.E. (2020). The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. Cell *183*, 1249–1263.

8. Eckmann, S., Young, E.J., and Gjorgjieva, J. (2024). Synapse-type-specific competitive hebbian learning forms functional recurrent networks. Proceedings of the National Academy of Sciences *121*, e2305326121.

9. Farrell, M., and Pehlevan, C. (2024). Recall tempo of hebbian sequences depends on the interplay of hebbian kernel with tutor signal timing. Proceedings of the National Academy of Sciences *121*, e2309876121.

10. Confavreux, B., Harrington, Z.P., Kania, M., Ramesh, P., Krouglova, A.N., Bozelos, P.A., Macke, J.H., Saxe, A.M., Gonçalves, P.J., and Vogels, T.P. (2025). Memory by a thousand rules: Automated discovery of multi-type plasticity rules reveals variety & degeneracy at the heart of learning. bioRxiv pp. 2025–05.

11. Feldman, D.E. (2012). The spike-timing dependence of plasticity. Neuron *75*, 556–571.

12. Mishra, R.K., Kim, S., Guzman, S.J., and Jonas, P. (2016). Symmetric spike timing-dependent plasticity at ca3–ca3 synapses optimizes storage and recall in autoassociative networks. Nature communications *7*, 11552.

13. Richards, B.A., Lillicrap, T.P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R.P., de Berker, A., Ganguli, S. et al. (2019). A deep learning framework for neuroscience. Nature neuroscience *22*, 1761–1770.

14. Cohn, R., Morantte, I., and Ruta, V. (2015). Coordinated and compartmentalized neuromodulation shapes sensory processing in drosophila. Cell *163*, 1742–1755.

15. Bittner, K.C., Milstein, A.D., Grienberger, C., Romani, S., and Magee, J.C. (2017). Behavioral time scale synaptic plasticity underlies ca1 place fields. Science *357*, 1033–1036.

16. Handler, A., Graham, T.G., Cohn, R., Morantte, I., Siliciano, A.F., Zeng, J., Li, Y., and Ruta, V. (2019). Distinct dopamine receptor pathways underlie the temporal sensitivity of associative learning. Cell *178*, 60–75.

17. Milstein, A.D., Li, Y., Bittner, K.C., Grienberger, C., Soltesz, I., Magee, J.C., and Romani, S. (2021). Bidirectional synaptic plasticity rapidly modifies hippocampal representations. eLife *10*, e73046.

18. Silva, N.T., Ramírez-Buriticá, J., Pritchett, D.L., and Carey, M.R. (2024). Climbing fibers provide essential instructive signals for associative learning. Nature Neuroscience *27*, 940–951.

19. Grienberger, C., and Magee, J.C. (2022). Entorhinal cortex directs learning-related changes in ca1 representations. Nature *611*, 554–562.

20. Marr, D., and Thach, W.T. (1991). A theory of cerebellar cortex. From the retina to the neocortex: selected papers of David Marr pp. 11–50.

21. Albus, J.S. (1971). A theory of cerebellar function. Mathematical biosciences *10*, 25–61.

22. Medina, J.F., and Lisberger, S.G. (2008). Links from complex spikes to local plasticity and motor learning in the cerebellum of awake-behaving monkeys. Nature neuroscience *11*, 1185–1192.

23. Suvrathan, A., Payne, H.L., and Raymond, J.L. (2016). Timing rules for synaptic plasticity matched to behavioral function. Neuron *92*, 959–967.

24. Cohen, T.S., Geiger, M., Köhler, J., and Welling, M. (2018). Spherical cnns. arXiv preprint arXiv:1801.10130.

25. Zolin, A., Cohn, R., Pang, R., Siliciano, A.F., Fairhall, A.L., and Ruta, V. (2021). Context-dependent representations of movement in drosophila dopaminergic reinforcement pathways. Nature Neuroscience *24*, 1555–1566.

26. Bakker, A., Kirwan, C.B., Miller, M., and Stark, C.E. (2008). Pattern separation in the human hippocampal ca3 and dentate gyrus. science *319*, 1640–1642.

27. Caron, S.J., Ruta, V., Abbott, L.F., and Axel, R. (2013). Random convergence of olfactory inputs in the drosophila mushroom body. Nature *497*, 113–117.

28. Rolls, E.T. (2013). The mechanisms for pattern completion and pattern separation in the hippocampus. Frontiers in Systems Neuroscience *7*, 74.

29. Leutgeb, J.K., Leutgeb, S., Moser, M.B., and Moser, E.I. (2007). Pattern separation in the dentate gyrus and ca3 of the hippocampus. science *315*, 961–966.

30. Litwin-Kumar, A., Harris, K.D., Axel, R., Sompolinsky, H., and Abbott, L. (2017). Optimal degrees of synaptic connectivity. Neuron *93*, 1153–1164.

31. Vandael, D., and Jonas, P. (2024). Structure, biophysics, and circuit function of a “giant” cortical presynaptic terminal. Science *383*, eadg6757.

32. Sussillo, D., and Abbott, L.F. (2009). Generating coherent patterns of activity from chaotic neural networks. Neuron *63*, 544–557.

33. Mante, V., Sussillo, D., Shenoy, K.V., and Newsome, W.T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. nature *503*, 78–84.

34. Barak, O. (2017). Recurrent neural networks as versatile tools of neuroscience research. Current opinion in neurobiology *46*, 1–6.

35. Mastrogiuseppe, F., and Ostojic, S. (2018). Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. Neuron *99*, 609–623.

36. Remington, E.D., Narain, D., Hosseini, E.A., and Jazayeri, M. (2018). Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. Neuron *98*, 1005–1019.

37. Yang, G.R., Joglekar, M.R., Song, H.F., Newsome, W.T., and Wang, X.J. (2019). Task representations in neural networks trained to perform many cognitive tasks. Nature neuroscience *22*, 297–306.

38. Driscoll, L.N., Shenoy, K., and Sussillo, D. (2024). Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. Nature Neuroscience *27*, 1349–1363.

39. Werbos, P.J. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE *78*, 1550–1560.

40. Pang, R., and Fairhall, A.L. (2019). Fast and flexible sequence induction in spiking neural networks via rapid excitability changes. eLife *8*, e44324.

41. Magee, J.C., and Grienberger, C. (2020). Synaptic plasticity forms and functions. Annual Review of Neuroscience *43*, 95–117.

42. Jiang, L., and Litwin-Kumar, A. (2021). Models of heterogeneous dopamine signaling in an insect learning and memory center. PloS computational biology *17*, e1009205.

43. Pemberton, J., Chadderton, P., and Costa, R.P. (2024). Cerebellar-driven cortical dynamics can enable task acquisition, switching and consolidation. Nature Communications *15*, 10913.

44. Boser, B.E., Guyon, I.M., and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory. pp. 144–152.

45. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. IEEE Intelligent Systems and their applications *13*, 18–28.

46. Bishop, C.M., and Nasrabadi, N.M. (2006). Pattern recognition and machine learning vol. 4. Springer.

47. Schuessler, F., Mastrogiuseppe, F., Ostojic, S., and Barak, O. (2024). Aligned and oblique dynamics in recurrent neural networks. eLife *13*, RP93060.

48. Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In International conference on machine learning. Pmlr pp. 1310–1318.

49. Chettih, S.N., Mackevicius, E.L., Hale, S., and Aronov, D. (2024). Barcoding of episodic memories in the hippocampus of a food-caching bird. Cell *187*, 1922–1935.

50. Dasgupta, S., Stevens, C.F., and Navlakha, S. (2017). A neural algorithm for a fundamental computing problem. Science *358*, 793–796.

51. Plitt, M.H., and Giocomo, L.M. (2021). Experience-dependent contextual codes in the hippocampus. Nature Neuroscience *24*, 705–714.

52. Farries, M.A., and Fairhall, A.L. (2007). Reinforcement learning with modulated spike timing–dependent synaptic plasticity. Journal of neurophysiology *98*, 3648–3665.

53. Frémaux, N., and Gerstner, W. (2016). Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. Frontiers in neural circuits *9*, 85.

54. Kuśmierz, Ł., Isomura, T., and Toyoizumi, T. (2017). Learning with three factors: modulating hebbian plasticity with errors. Current opinion in neurobiology *46*, 170–177.

55. Strogatz, S.H.S.H. (2018). Nonlinear Dynamics and Chaos: with applications to Physics, Biology, Chemistry, and Engineering. . ISBN 9780813349107.

56. Luo, T.Z., Kim, T.D., Gupta, D., Bondy, A.G., Kopec, C.D., Elliot, V.A., DePasquale, B., and Brody, C.D. (2023). Transitions in dynamical regime and neural mode underlie perceptual decision-making. bioRxiv pp. 2023–10.

57. Sompolinsky, H., Crisanti, A., and Sommers, H.J. (1988). Chaos in random neural networks. Physical review letters *61*, 259.

58. Pezon, L., Schmutz, V., and Gerstner, W. (2024). Linking neural manifolds to circuit structure in recurrent networks. bioRxiv pp. 2024–02.

59. Claudi, F., Chandra, S., and Fiete, I.R. (2025). A theory and recipe to construct general and biologically plausible integrating continuous attractor neural networks. bioRxiv pp. 2025–05.

60. Amit, D.J., and Amit, D.J. (1989). Modeling brain function: The world of attractor neural networks. Cambridge university press.

61. Seung, H.S. (1996). How the brain keeps the eyes still. Proceedings of the National Academy of Sciences *93*, 13339–13344.

62. Buzsaki, G., and Draguhn, A. (2004). Neuronal oscillations in cortical networks. science *304*, 1926–1929.

63. Wong, K.F., and Wang, X.J. (2006). A recurrent network mechanism of time integration in perceptual decisions. Journal of Neuroscience *26*, 1314–1328.

64. Khona, M., and Fiete, I.R. (2022). Attractor and integrator networks in the brain. Nature Reviews Neuroscience *23*, 744–766.

65. Astrom, K.J., and Wittenmark, B. (2008). Adaptive Control (2nd ed.). Second ed.. Dover Books on Engineering.

66. Alemi, A., Machens, C., Deneve, S., and Slotine, J.J. (2018). Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules. In Proceedings of the AAAI conference on artificial intelligence vol. 32.

67. Slotine, J.J.E., and Li, W. (1991). Applied Nonlinear Control. Prentice Hall. ISBN 0130408905. doi: 10.1088/1741-2552/8/6/060201.

68. Lohmiller, W., and Slotine, J.J.E. (1998). On contraction analysis for non-linear systems. Automatica *34*, 683–696. doi: 10.1016/S0005-1098(98)00019-3.

69. Bullo, F. (2022). Contraction theory for dynamical systems. . ISBN 9798836646806.

70. Tsukamoto, H., Chung, S.J., and Slotine, J.J.E. (2021). Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview. Annual Reviews in Control *52*, 135–169. doi: 10.1016/j.arcontrol.2021.10.001. arXiv:2110.00675.

71. Narendra, K.S., and Annaswamy, A.M. (1987). Persistent excitation in adaptive systems. International Journal of Control *45*, 127–160. URL: https://doi.org/10.1080/00207178708933715. doi: 10.1080/00207178708933715.

72. Panteley, E., Loría, A., and Teel, A. (2001). Relaxed persistency of excitation for uniform asymptotic stability. IEEE Transactions on Automatic Control *46*, 1874–1886. doi: 10.1109/9.975471.

73. Wolpert, D.M., Miall, R.C., and Kawato, M. (1998). Internal models in the cerebellum. Trends in cognitive sciences *2*, 338–347.

74. Wolpert, D.M., and Miall, R.C. (1996). Forward Models for Physiological Motor Control. Neural Networks *9*, 1265–1279.

75. Kawato, M., Furukawa, K., and Suzuki, R. (1987). A hierarchical neural-network model for control and learning of voluntary movement. Biological Cybernetics *57*, 169–185. doi: 10.1007/BF00364149.

76. Kawato, M., and Gomi, H. (1992). The cerebellum and VOR/OKR learning models. Trends in Neurosciences *15*, 445–453.

77. Lillicrap, T.P., and Santoro, A. (2019). Backpropagation through time and the brain. Current opinion in neurobiology *55*, 82–89.

78. Murray, J.M. (2019). Local online learning in recurrent networks with random feedback. Elife *8*, e43299.

79. Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. Nature communications *11*, 3625.

80. Engelken, R. (2023). Gradient flossing: Improving gradient descent through dynamic control of jacobians. Advances in Neural Information Processing Systems *36*, 10412–10439.

81. Pan-Vazquez, A., Araujo, Y.S., McMannon, B., Louka, M., Bandi, A., Haetzel, L., Faulkner, M., Pillow, J.W., Daw, N.D., and Witten, I.B. (2024). Pre-existing visual responses in a projection-defined dopamine population explain individual learning trajectories. Current Biology *34*, 5349–5358.

82. Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning representations by back-propagating errors. nature *323*, 533–536.

83. Liu, Y.H., Smith, S., Mihalas, S., Shea-Brown, E., and Sümbül, U. (2021). Cell-type–specific neuromodulation guides synaptic credit assignment in a spiking neural network. Proceedings of the National Academy of Sciences *118*, e2111821118.

84. Kaleb, K., Feulner, B., Gallego, J., and Clopath, C. (2024). Feedback control guides credit assignment in recurrent neural networks. Advances in Neural Information Processing Systems *37*, 5122–5144.

85. Eliasmith, C., and Anderson, C.H. (2003). Neural engineering: Computation, representation, and dynamics in neurobiological systems. MIT press.

86. Costacurta, J., Bhandarkar, S., Zoltowski, D., and Linderman, S. (2024). Structured flexibility in recurrent neural networks via neuromodulation. Advances in Neural Information Processing Systems *37*, 1954–1972.

87. Nieh, E.H., Schottdorf, M., Freeman, N.W., Low, R.J., Lewallen, S., Koay, S.A., Pinto, L., Gauthier, J.L., Brody, C.D., and Tank, D.W. (2021). Geometry of abstract learned knowledge in the hippocampus. Nature *595*, 80–84.

88. Green, G. (1889). An essay on the application of mathematical analysis to the theories of electricity and magnetism vol. 3. Mayer & Müller.

89. Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. Advances in neural information processing systems *31*.

90. Ramamoorthi, R., and Hanrahan, P. (2001). A signal-processing framework for inverse rendering. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 117–128.