Neural learning rules for generating flexible predictions and computing the successor representation

Ching Fang, Dmitriy Aronov, L.F. Abbott, Emily Mackevicius

Zuckerman Institute, Department of Neuroscience, Columbia University, New York, NY, USA.

Abstract

The predictive nature of the hippocampus is thought to be useful for memory-guided cognitive behaviors. Inspired by the reinforcement learning literature, this notion has been formalized as a predictive map called the successor representation (SR). The SR captures a number of observations about hippocampal activity. However, the algorithm does not provide a neural mechanism for how such representations arise. Here, we show the dynamics of a recurrent neural network naturally calculate the SR when the synaptic weights match the transition probability matrix. Interestingly, the predictive horizon can be flexibly modulated simply by changing the network gain. We derive simple, biologically plausible learning rules to learn the SR in a recurrent network. We test our model with realistic inputs and match hippocampal data recorded during random foraging. Taken together, our results suggest that the SR is more accessible in neural circuits than previously thought and can support a broad range of cognitive functions.

1. Introduction

To learn from the past, plan for the future, and form an understanding of our world, we require memories of personal experiences. These types of memories depend on the hippocampus for formation and recall [1, 2, 3], but an algorithmic and mechanistic understanding of memory formation and retrieval in this region remains elusive. The need to support planning and inference suggests that one of the key features of memory is the ability to predict possible outcomes [4, 5, 6, 7]. Consistent with this hypothesis, experimental work has shown that, across species and tasks, hippocampal activity is predictive of the future experience of an animal [8, 9, 10, 11, 12, 13, 14, 15]. Furthermore, theoretical work has

found that models endowed with predictive objectives tend to resemble hippocampal activity [16, 17, 18, 19, 20, 21, 6, 22]. Thus, it is clear that predictive representations are an important aspect of hippocampal memory.

Inspired by work in the reinforcement learning (RL) field, these observations have been formalized by describing hippocampal activity as a predictive map under the successor representation (SR) algorithm [23, 24, 18]. Under this framework, an animal's experience in the world is represented as a trajectory through some defined state space, and hippocampal activity predicts the future experience of an animal by integrating over the likely states that an animal will visit given its current state. This algorithm further explains how, in addition to episodic memory, the hippocampus may support relational reasoning and decision making [21, 25], consistent with differences in hippocampal representations in different tasks [26, 27]. The SR framework captures many experimental observations of neural activity, leading to a proposed computational function for the hippocampus [18].

While the SR algorithm convincingly argues for a computational function of the hippocampus, it is unclear what biological mechanisms might compute the SR in a neural circuit. Thus, several relevant questions remain that are difficult to probe with the current algorithm. What kind of neural architecture should one expect in a region that can support this computation? Are there distinct forms of plasticity and neuromodulation needed in this system? What is the structure of hippocampal inputs to be expected? A biologically plausible model can explore these questions and provide insight into both mechanism and function [28, 29, 30].

In other systems, it has been possible to derive biological mechanisms with the goal of achieving a particular network function or property [31, 32, 33, 34, 35, 36, 37, 38]. Key to many of these models is the constraint that learning rules at any given neuron can only use information local to that neuron. A promising direction towards such a neural model of the SR is to use the dynamics of a recurrent network to perform SR computations [39, 40]. However, this idea has not been tied to neural learning rules that support its operation and allow for testing of specific hypotheses.

Here, we show that an RNN with local learning rules and an adaptive learning rate exactly calculates the SR at steady state. We test our model with realistic inputs and make comparisons to neural data. In addition, we compare our results to the standard SR

algorithm with respect to the speed of learning and the learned representations in cases where multiple solutions exist. Our work provides a mechanistic account for an algorithm that has been frequently connected to the hippocampus, but could only be interpreted at an algorithmic level. This network-level perspective allows us to make specific predictions about hippocampal mechanisms and activity.

2. Results

2.1. The successor representation

The SR algorithm described in Stachenfeld et al. [18] first discretizes the environment explored by an animal (whether a physical or abstract space) into a set of n states that the animal transitions through over time (Figure 1a). The animal's behavior can then be thought of as a Markov chain with a corresponding transition probability matrix $T_{n\times n}$ (Figure 1b). T gives the probability that the animal transitions to a state s' from the state s in one time step: $T_{ji} = P(s' = i|s = j)$. The SR matrix is defined as

$$M = \sum_{t=0}^{\infty} \gamma^t T^t = (I - \gamma T)^{-1} \tag{1}$$

Here, $\gamma \in (0, 1)$ is a temporal discount factor. M_{ji} can be seen as a measure of the occcupancy of state i over time if the animal starts at state j, with γ controlling how much to discount time steps in the future (Figure 1c). The SR of state j is the jth row of M and represents the states that an animal is likely to transition to from state j. Stachenfeld et al. [18] demonstrate that, if one assumes each state drives a single neuron, the SR of j resembles the population activity of hippocampal neurons when the animal is at state j (Figure 1d). They also show that the ith column of M resembles the place field (activity as a function of state) of a hippocampal neuron representing state i (Figure 1e). In addition, the ith column of M shows which states are likely to lead to state i.

2.2. Recurrent neural network computes SR at steady state

We begin by drawing connections between the SR algorithm [18] and an analogous neural network architecture. The input to the network encodes the current state of the animal and

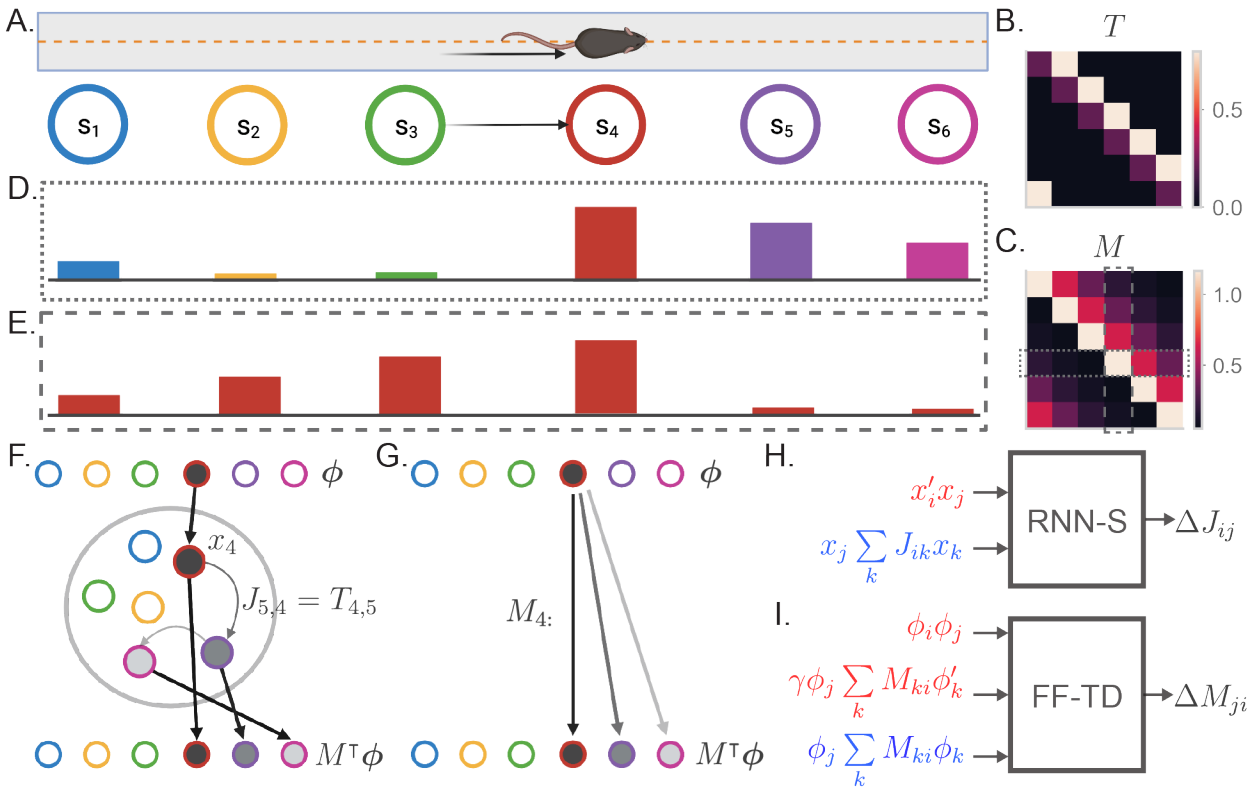

Figure 1: **The successor representation and an analogous recurrent network model. a.** The behavior of an animal running down a linear track can be described as a transition between discrete states where the states encode spatial location. **b.** By counting the transitions between different states, the behavior of an animal can be summarized in a transition probability matrix T. **c.** The successor representation matrix is defined as $M = \sum_{t=0}^{\infty} \gamma^t T^t$. Here, M is shown for $\gamma = 0.6$. Dashed boxes indicate the slices of M shown in (d) and (e). **d.** The fourth row of the M matrix describes the activity of each state-encoding neuron when the animal is at the fourth state. **e.** The fourth column of the M matrix describes the place field of the neuron encoding the fourth state. **f.** Recurrent network model of the SR (RNN-S). The current state of the animal is one-hot encoded by a layer of input neurons. Inputs connect one-to-one onto RNN neurons with synaptic connectivity matrix $J = T^\intercal$. The activity of the RNN neurons are represented by $\boldsymbol{x}$. SR activity is read out from one-to-one connections from the RNN neurons to the output neurons. The example here shows inputs and outputs when the animal is at state 4. **g.** Feedforward neural network model of the SR (FF-TD). The M matrix is encoded in the weights from the input neurons to the output layer neurons, where the SR activity is read out. **h.** Diagram of the terms used for the RNN-S learning rule. Terms in red are used for potentiation while terms in blue are used for normalization (equation 4). **i.** As in (h) but for the feedforward-TD model (equation 11). To reduce the notation indicating time steps, we use ' in place of (t) and no added notation for $(t-1)$.

is represented by a layer of input neurons (Figure 1fg). These neurons feed into the rest of the network that computes the SR (Figure 1fg). The SR is then read out by a layer of output neurons so that downstream systems receive a prediction of the upcoming states (Figure 1fg). We will first model the inputs $\boldsymbol{\phi}$ as one-hot encodings of the current state of the animal (Figure 1fg). That is, each input neuron represents a unique state and are one-to-one

connected to the hidden neurons.

We first consider an architecture in which a recurrent neural network (RNN) is used to compute the SR (Figure 1f). Let us assume that the T matrix is encoded in the synaptic weights of the RNN. In this case, the steady state activity of the network in response to input $\boldsymbol{\phi}$ retrieves a row of the SR matrix, $M^\intercal \boldsymbol{\phi}$ (Figure 1f, Supplementary Notes 1). Intuitively, this is because each recurrent iteration of the RNN progresses the prediction by one transition. In other words, the tth recurrent iteration raises T to the tth power as in equation 1. To formally derive this result, we first start by defining the dynamics of our RNN with classical rate network equations [41]. At time t, the firing rate $\boldsymbol{x}(t)$ of N neurons given each neurons' input $\boldsymbol{\phi}(t)$ follows the discrete-time dynamics (assuming a step size $\Delta t = 1$)

$$\Delta \boldsymbol{x} = -\boldsymbol{x}(t) + f(\gamma J \boldsymbol{x}(t)) + \boldsymbol{\phi}(t) \tag{2}$$

Here, γ scales the recurrent activity and is a constant factor for all neurons. The synaptic weight matrix $J \in \mathcal{R}_{N \times N}$ is defined such that J_{ij} is the synaptic weight from neuron j to neuron i. Notably, this notation is transposed from what is used in RL literature, where conventions have the first index as the starting state. Generally, f is some nonlinear function in equation 2. For now, we will consider f to be the identity function, rendering this equation linear. Under this assumption, we can solve for the steady state activity x_{SS} as

$$\boldsymbol{x}_{ss} = (I - \gamma J)^{-1} \boldsymbol{\phi} \tag{3}$$

Equivalence between equation 1 and equation 3 is clearly reached when $J = T^\intercal$ [40, 39]. Thus, if the network can learn T in its synaptic weight matrix, it will exactly compute the SR.

A benefit of this scheme is that γ is not encoded in the synaptic weights. Thus, γ can be a flexibly modulated gain factor (see, for example, Sompolinsky et al. [42]) allowing the system to retrieve successor representations of varying predictive strengths. We will refer to the γ used during learning of the SR as the baseline γ, or γ_B.

We next consider what is needed in a learning rule such that J approximates $T^\intercal$. In order to learn a transition probability matrix, a learning rule must associate states that occur

sequentially and normalize the synaptic weights into a valid probability distribution. We derive a learning rule that addresses both requirements (Figure 1h, Supplementary Notes 2),

$$\Delta J_{ij} = \eta x_i(t) x_j(t-1) - \eta x_j(t-1) \sum_k J_{ik} x_k(t-1), \tag{4}$$

where η is the learning rate. The first term in equation 4 is a temporally asymmetric potentiation term that counts states that occur in sequence. This is similar to spike-timing dependent plasticity, or STDP [43, 8, 44]. The second term in equation 4 normalizes the synapses into a valid transition probability matrix, such that each column of $J = T^\intercal$ sums to 1.

Crucially, this update rule (equation 4) uses information local to each neuron (Figure 1h). We show that, in the asymptotic limit, the update rule extracts information about the inputs $\boldsymbol{\phi}$ and learns T exactly despite having access only to neural activity $\boldsymbol{x}$ (Supplementary Notes 3). We will refer to an RNN using equation 4 as the RNN-Successor, or RNN-S. Combined with recurrent dynamics (equation 3), RNN-S computes the SR exactly (Figure 1h).

As an alternative to the RNN-S model, we consider the conditions necessary for a feedforward neural network to compute the SR. Under this architecture, the M matrix must be encoded in the weights from the input neurons to the hidden layer neurons (Figure 1g). This can be achieved by updating the synaptic weights with a temporal difference (TD) learning rule, the standard update used to learn the SR in the usual algorithm. Although the TD update learns the SR, it requires information about multiple input layer neurons to make updates for the synapse from input neuron j to output neuron i (Figure 1i). Thus, it is useful to explore other possible mechanisms that are simpler to compute locally. We refer to the model described in Figure 1ih as the feedforward-TD (FF-TD) model.

2.3. Evaluating SR learning by biologically plausible learning rules

To evaluate the effectiveness of the RNN-S learning rule, we tested its accuracy in learning the SR matrix for random walks. Specifically, we simulated random walks with different transition biases in a 1D circular track environment (Figure 2a). The RNN-S can learn the SR for these random walks (Figure 2b).

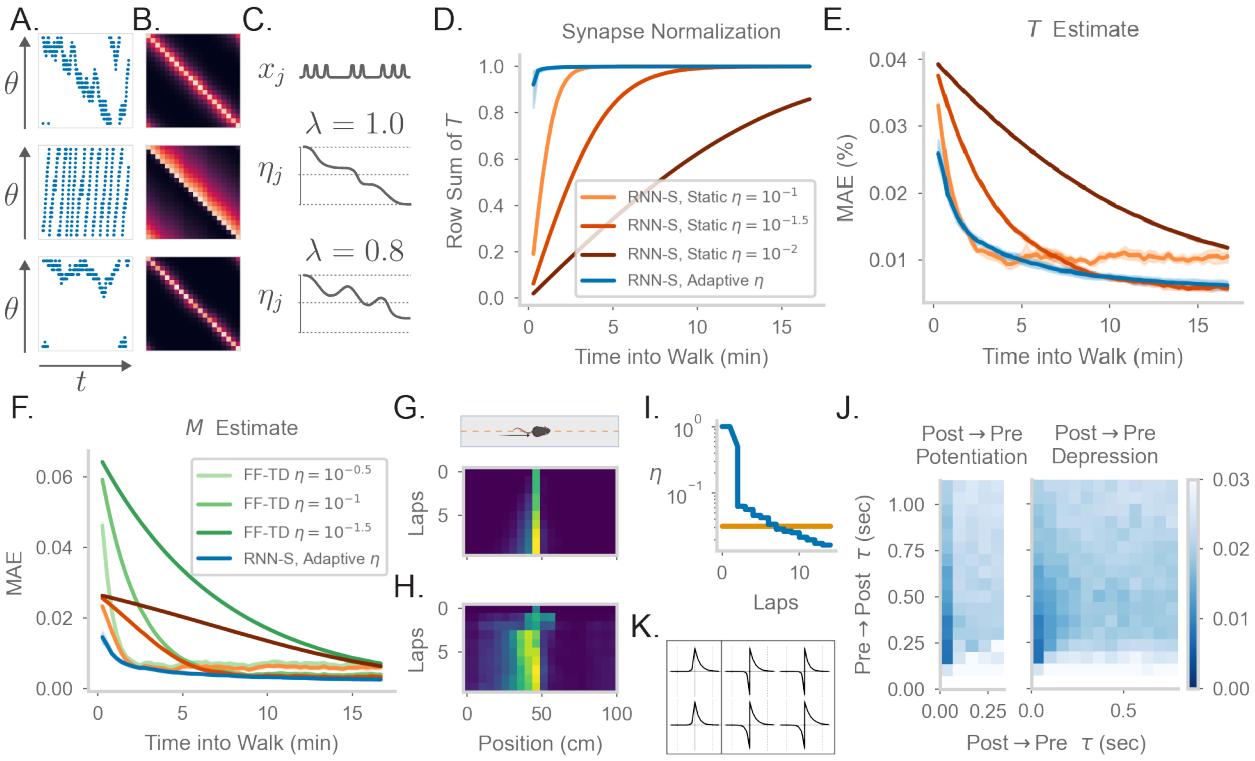

Figure 2: **Comparing the effects of an adaptive learning rate and plasticity kernels in RNN-S. a.** Sample one-minute segments from random walks on a 1 meter circular track. Possible actions in this 1D walk are to move forward, stay in one place, or move backward. Action probabilities are uniform (top), biased to move forward (middle), or biased to stay in one place (bottom). **b.** M matrices estimated by the RNN-S model in the full random walks from (a). **c.** The proposed learning rate normalization. The learning rate η_j for synapses out of neuron j changes as a function of its activity $\boldsymbol{x}_j$ and recency bias λ. Dotted lines are at $[0.0, 0.5, 1.0]$. **d.** The mean row sum of T over time computed by the RNN-S with an adaptive learning rate (blue) or the RNN-S with static learning rates (orange). Darker lines indicate larger static learning rates. Lines show the average over simulations from walks with a forward bias. A correctly normalized T matrix should have a row sum of 1.0. **e.** As in (d), but for the mean absolute error in estimating T. **f.** As in (e), but for mean absolute error in estimating the real M, and with performance of FF-TD included, with darker lines indicating slower learning rates for FF-TD. **g.** Lap-based activity map of a neuron from RNN-S with static learning rate $\eta = 10^{-1.5}$. The neuron encodes the state at 45 cm on a circular track. The simulated agent is moving according to forward-biased transition statistics. **h.** As in (g), but for RNN-S with adaptive learning rate. **i.** The learning rate over time for the neuron in (g) (orange) and the neuron in (h) (blue). **j.** Mean-squared error (MSE) at the end of meta-learning for different plasticity kernels. The pre→post (K_+) and post→pre (K_-) sides of each kernel were modeled by $Ae^{-\frac{1}{\tau}}$. Heatmap indices indicate the values τs were fixed to. Here, K_+ is always a positive function (i.e., A was positive), because performance was uniformly poor when K_+ was negative. K_- could be either positive (left, "Post → Pre Potentiation") or negative (right, "Post → Pre Depression"). Regions where the learned value for A was negligibly small were set to high errors. Errors are max-clipped at 0.03 for visualization purposes. **k.** Plasticity kernels chosen from the areas of lowest error in the grid search from (j). Left is post → pre potentiation. Right is post → pre depression. Kernels are normalized by the maximum, and dotted lines are at one second intervals.

Because equivalence is only reached in the asymptotic limit of learning (i.e., $\Delta J \to 0$), our RNN-S model learns the SR slowly. In contrast, animals are thought to be able to learn

the structure of an environment quickly [45], and neural representations in an environment can also develop quickly [46, 47, 48]. To remedy this, we introduce a dynamic learning rate that allows for faster normalization of the synaptic weight matrix, similar to the formula for calculating a moving average (Supplementary Notes 4). For each neuron, suppose that a trace n of its recent activity is maintained with some time constant $\lambda \in (0, 1]$,

$$\boldsymbol{n}(t) = \sum_{t'<t} \lambda^{(t-t')} \boldsymbol{x}(t') \tag{5}$$

If the learning rate of the outgoing synapses from each neuron j is inversely proportional to n_j ($\eta = \frac{1}{n_j(t)}$), the update equation quickly normalizes the synapses to maintain a valid transition probability matrix (Supplementary Notes 4). We refer to this as an adaptive learning rate and contrast it with the previous static learning rate. We consider the setting where $\lambda = 1$, so the learning rate monotonically decreases over time (Figure 2c). In general, however, the learning rate could increase or decrease over time if $\lambda < 1$ (Figure 2c), and n could be reset, allowing for rapid learning. Our learning rule with the adaptive learning rate is the same as in equation 4, with the exception that $\eta = \texttt{min}(\frac{1}{n_j(t)}, 1.0)$ for synapses J_{*j}. This learning rule still relies only on information local to the neuron as in Figure 1i.

The RNN-S with an adaptive learning rate normalizes the synapses more quickly than a network with a static learning rate (Figure 2d, Figure S2a) and learns T faster (Figure 2e, Figure S2b). The RNN-S with a static learning rate exhibits more of a tradeoff between normalizing synapses quickly (Figure 2d, Figure S2a) and learning M accurately (Figure 2e, Figure S2b). However, both versions of the RNN-S estimate M more quickly than the FF-TD model (Figure 2f, Figure S2c).

Place fields can form quickly, but over time the place fields may skew if transition statistics are consistently biased [18, 46, 47, 48]. The adaptive learning rate recapitulates both of these effects, which are thought to be caused by slow and fast learning processes, respectively. A low learning rate can capture the biasing of place fields, which develops over many repeated experiences. This is seen in the RNN-S with a static learning rate(Figure 2g). However, a high learning rate is needed for hippocampal place cells to develop sizeable place fields in one-shot. Both these effects of slow and fast learning can be seen in the neural activity of an

example RNN-S neuron with an adaptive learning rate (Figure 2h). After the first lap, a sizeable field is induced in a one-shot manner, centered at the cell's preferred location. In subsequent laps, the place field slowly distorts to reflect the bias of the transition statistics (Figure 2h). The model is able to capture these learning effects because the adaptive learning rate transitions between high and low learning rates, unlike the static version (Figure 2i).

Thus far, we have assumed that the RNN-S learning rule uses pre→post activity over two neighboring time steps (equation 4). A more realistic framing is that a convolution with a plasticity kernel determines the weight change at any synapse. We tested how this affects our model and what range of plasticity kernels best supports the estimation of the SR. We do this by replacing the pre→post potentiation term in equation 4 with a convolution:

$$\Delta J_{ij} = x_i(t) \sum_{t'=-\infty}^{t} K_+(t-t')x_j(t') + x_j(t) \sum_{t'=-\infty}^{t} K_-(t-t')x_i(t') - \eta x_j(t-1) \sum_k J_{ik}x_k(t-1) \quad (6)$$

In the above equation, the full kernel K is split into a pre→post kernel (K_+) and a post→pre kernel (K_-). K_+ and K_- are parameterized as independent exponential functions, $Ae^{-t/\tau}$.

To systematically explore the space of plasticity kernels that can be used to learn the SR, we performed a grid search over the sign and the time constants of the pre→post and post→pre sides of the plasticity kernels. Plasticity kernels that are STDP-like are more effective than others, although plasticity kernels with slight post→pre potentiation work as well (Figure 2j). The network is sensitive to the time constant and tends to find solutions for time constants around a few hundred milliseconds (Figure 2jk). Our robustness analysis indicates the timescale of a plasticity rule in such a circuit may be longer than expected by standard STDP, but within the timescale of changes in behavioral states. We note that this also contrasts with behavioral timescale plasticity [48], which integrates over a window that is several seconds long. Finally, we see that even plasticity kernels with slightly different time constants may give a result that is SR-like, even if they do not estimate the SR exactly (Figure 2j).

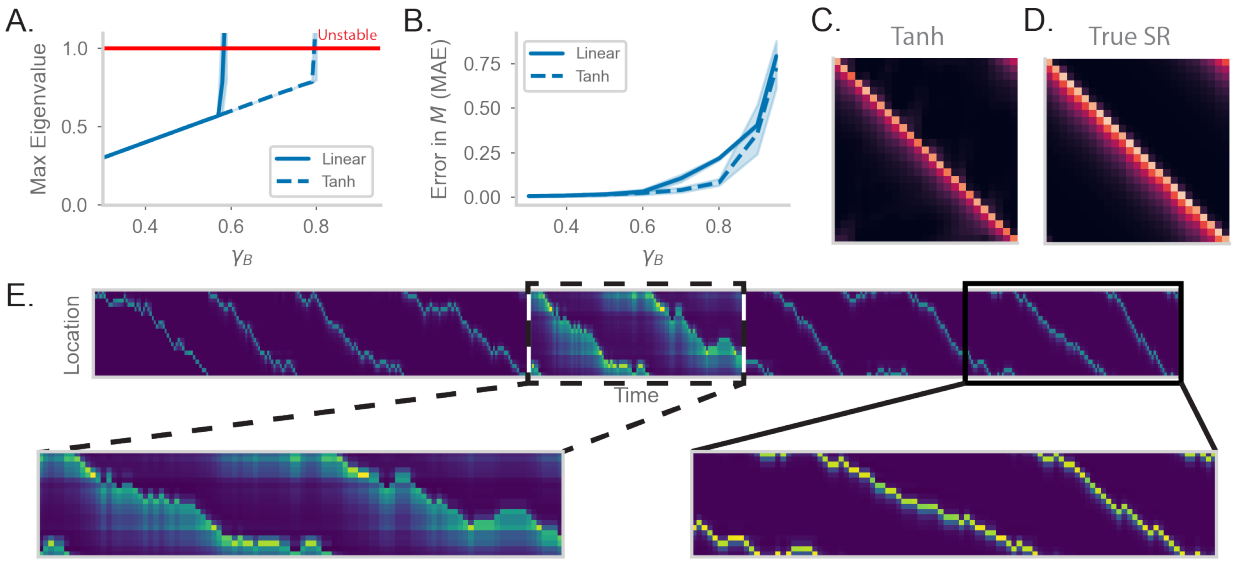

Figure 3: **RNN-S requires a stable choice of γ_B during learning, and can compute SR with any γ_R a.** Maximum real eigenvalue of the J matrix at the end of random walks under different γ_B. The network dynamics were either fully linear (solid) or had a tanh nonlinearity (dashed). Red line indicates the transition into an unstable regime. **b.** MAE of M matrices learned by RNN-S with different γ_B. RNN-S was simulated with linear dynamics (solid line) or with a tanh nonlinearity added to the recurrent dynamics (dashed line). Test datasets used various biases in action probability selection. **c.** M matrix learned by RNN-S with tanh nonlinearity added in the recurrent dynamics. A forward-biased walk on a circular track was simulated, and $\gamma_B = 0.8$. **d.** The true M matrix of the walk used to generate (c). **e.** Simulated population activity over the first ten laps in a circular track with $\gamma_B = 0.4$. Dashed box indicates the retrieval phase, where learning is turned off and $\gamma_R = 0.9$. Boxes are zoomed in on three minute windows.

2.4. RNN-S can compute the SR with arbitrary γ_R under a stable regime of γ_B

We next investigate how robust the RNN-S model is to the value of γ. Typically, for purposes of fitting neural data or for RL simulations, γ will take on values as high as 0.9 [18, 49]. However, previous work that used RNN models reported that recurrent dynamics become unstable if the gain γ exceeds a critical value [42, 45]. This could be problematic as we show analytically that the RNN-S update rule is effective only when the network dynamics are stable and do not have non-normal amplification (Supplementary Notes 2). If these conditions are not satisfied during learning, the update rule no longer optimizes for fitting the SR and the learned weight matrix will be incorrect.

We first test how the value of γ_B, the gain of the network during learning, affects the RNN-S dynamics. The dynamics become unstable when γ_B exceeds 0.6 (Figure S3a-e). Specifically, the eigenvalues of the synaptic weight matrix exceed the critical threshold for stability when $\gamma_B > 0.6$ (Figure 3a, "Linear"). As expected from our analytical results,

the stability of the network is tied to the network's ability to estimate M. RNN-S cannot estimate M well when $\gamma_B > 0.6$ (Figure 3b, "Linear"). We explored two strategies to enable RNN-S to learn at high γ.

One way to tame this instability is to add a saturating nonlinearity into the dynamics of the network. Instead of assuming the network dynamics are fully linear (f is the identity function in equation 2), we add a hyperbolic tangent into the dynamics equation. This extends the stable regime of the network– the eigenvalues do not exceed the critical threshold until $\gamma_B > 0.8$ (Figure 3a). Similar to the linear case, the network with nonlinear dynamics fits M well until the critical threshold for stability (Figure 3b). These differences are clear visually as well. While the linear network does not estimate M well for $\gamma_B = 0.8$ (Figure 3b), the estimate of the nonlinear network (Figure 3c) is a closer match to the true M (Figure 3d). However, there is a tradeoff between the stabilizing effect of the nonlinearity and the potential loss of accuracy in calculating M with a nonlinearity (Figure S3h).

We explore an alternative strategy for computing M with arbitrarily high γ in the range $0 \leq \gamma < 1$. We have thus far pushed the limits of the model in learning the SR for different γ_B. However, an advantage of our recurrent architecture is that γ is a global gain modulated independently of the synaptic weights. Thus, an alternative strategy for computing M with high γ is to consider two distinct modes that the network can operate under. First, there is a learning phase in which the plasticity mechanism actively learns the structure of the environment and the model is in a stable regime (i.e., γ_B is small). Separately, there is a retrieval phase during which the gain γ_R of the network can be flexibly modulated. By changing the gain, the network can compute the SR with arbitrary prediction horizons, without any changes to the synaptic weights. We show the effectiveness of separate network phases by simulating a 1D walk where the learning phase uses a small γ_B (Figure 3e). Halfway through the walk, the animal enters a retrieval mode and accurately computes the SR with higher γ_R (Figure 3e).

Under this scheme, the model can compute the SR for any $\gamma < 1$ (Figures S3f-h). The separation of learning and retrieval phases stabilizes neural dynamics and allows flexible tuning of predictive power depending on task context.

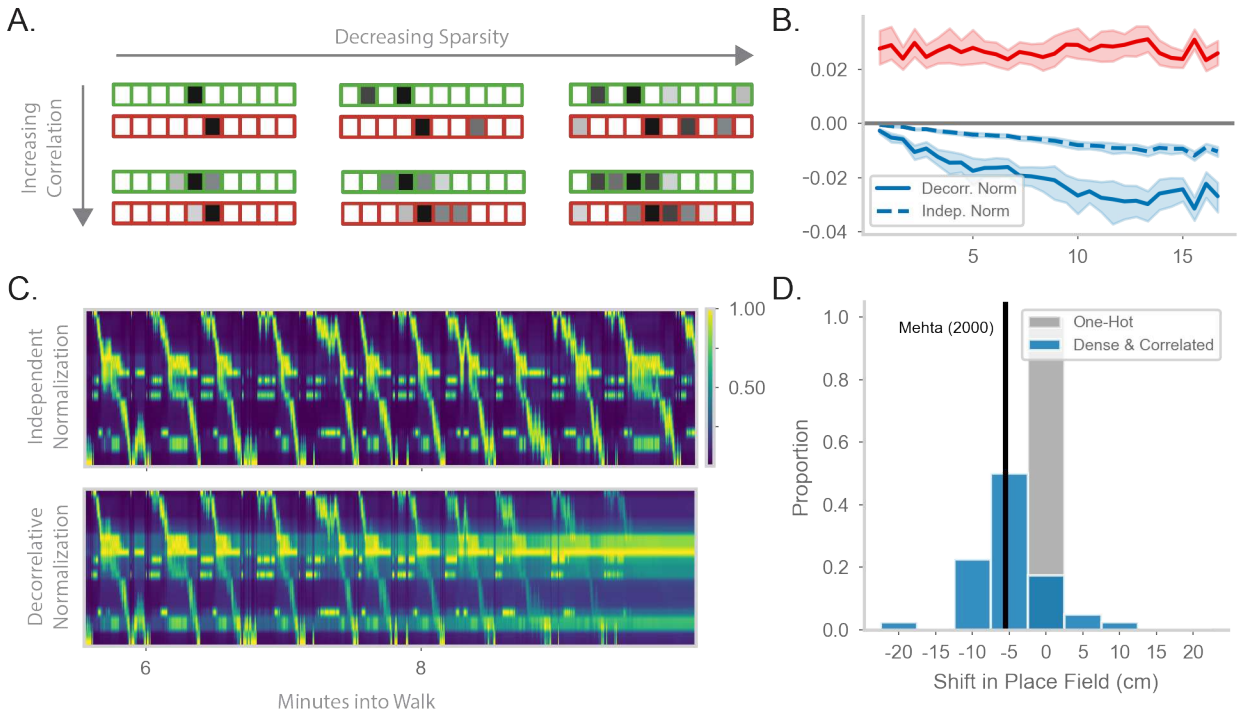

Figure 4: **Generalizing the model to more realistic inputs. a.** Illustration of possible feature encodings ϕ for two spatially adjacent states in green and red. Feature encodings may vary in sparsity level and spatial correlation. **b.** Average value of the STDP component (red) and the decorrelative normalization (solid blue) component of the gradient update over the course of a random walk. In dashed blue is a simpler Oja-like independent normalization update for comparison. Simulations are from forward-biased walks on a circular track. Input features are 3% sparse, with 10 cm spatial correlation. **c.** Top: Example population activity of neurons in the RNN-S using the full decorrelative normalization rule over a 2 minute window of a forward-biased random walk. Population activity is normalized by the maximum firing rate. Bottom: As above, but for RNN-S using the simplified normalization update. **d.** Shifts in place field peaks after a half hour simulation from the first two minutes of a 1D walk. Proportion of shifts in RNN-S with one-hot inputs shown in gray. Proportion of shifts in RNN-S with feature encodings (10% sparsity, 7.5 cm spatial correlation, $\gamma_R = 0.8$) shown in blue. Each data point is the average shift observed in one simulated walk, and each histogram is over 40 simulated walks. Solid line indicates the reported measure from Mehta & Wilson (2000).

2.5. RNN-S can be generalized to more complex inputs with successor features

We wondered how RNN-S performs given more biologically realistic inputs. We have so far assumed that an external process has discretized the environment into uncorrelated states so that each possible state is represented by a unique input neuron. In other words, the inputs $\boldsymbol{\phi}$ are one-hot vectors. However, inputs into the hippocampus are expected to be continuous and heterogeneous, with states encoded by overlapping sets of neurons [50]. When inputs are not one-hot, there is not always a canonical ground-truth T matrix to fit and the predictive representations are referred to as successor features [49, 51]. In this setting, the performance of a model estimating successor features is evaluated by the temporal difference

(TD) loss function.

Using the RNN-S model and update rule (equation 4), we explore more realistic inputs $\boldsymbol{\phi}$ and refer to $\boldsymbol{\phi}$ as "input features" for consistency with the successor feature literature. We vary the sparsity and spatial correlation of the input features (Figure 4a). As before (Figure 3h), the network will operate in separate learning and retrieval modes, where γ_B is below the critical value for stability. Under these conditions, the update rule will learn

$$J = R_{\boldsymbol{\phi}\boldsymbol{\phi}}(-1)R_{\boldsymbol{\phi}\boldsymbol{\phi}}(0)^{-1} \tag{7}$$

at steady state, where $R_{\boldsymbol{\phi}\boldsymbol{\phi}}(\tau)$ is the correlation matrix of $\boldsymbol{\phi}$ with time lag τ (Supplementary Notes 3). Thus, the RNN-S update rule has the effect of normalizing the input feature via a decorrelative factor ($R_{\boldsymbol{\phi}\boldsymbol{\phi}}(0)^{-1}$) and mapping the normalized input to the feature expected at the next time step in a STDP-like manner ($R_{\boldsymbol{\phi}\boldsymbol{\phi}}(-1)$). This interpretation generalizes the result that $J = T^\intercal$ in the one-hot encoding case (Supplementary Notes 3).

We wanted to further explore the function of the normalization term. In the one-hot case, it operates over each synapse independently and makes a probability distribution. With more realistic inputs, it operates over a set of synapses and has a decorrelative effect. We first ask how the decorrelative term changes over learning of realistic inputs. We compare the mean value of the STDP term of the update ($x_i(t)x_j(t-1)$) to the normalization term of the update ($x_j(t-1)\sum_k J_{ik}x_k(t-1)$) during a sample walk (Figure 4b). The RNN-S learning rule has stronger potentiating effects in the beginning of the walk. As the model learns more of the environment and converges on the correct transition structure, the strength of the normalization term balances out the potentiation term. It may be that the normalization term is particularly important in maintaining this balance as inputs become more densely encoded. We test this hypothesis by using a normalization term that operates on each synapse independently (similar to Oja's Rule, [52], Supplementary Notes 5). We see that the equilibrium between potentiating and depressing effects is not achieved by this type of independent normalization (Figure 4b, Supplementary Notes 6).

We wondered whether the decorrelative normalization term is necessary for the RNN-S to develop accurate representations. By replacing the decorrelative term with an independent

normalization, features from non-adjacent states begin to be associated together and the model activity becomes spatially non-specific over time (Figure 4c, top). In contrast, using the decorrelative term, the RNN-S population activity is more localized (Figure 4c, bottom).

Interestingly, we noticed an additional feature of place maps as we transitioned from one-hot feature encodings to more complex feature encodings. We compared the representations learned by the RNN-S in a circular track walk with one-hot features versus more densely encoded features. For both input distributions, the RNN-S displayed the same skewing in place fields seen in Figure 2 (Figure S4). However, the place field peaks of the RNN-S model additionally shifted backwards in space for the more complex feature encodings (Figure 4d). This was not seen for the one-hot encodings (Figure 4d). The shifting in the RNN-S model is consistent with the observations made in Mehta et al. [17] and demonstrates the utility of considering more complex input conditions. A similar observation was made in Stachenfeld et al. [18] with noisy state inputs. In both cases, field shifts could be caused by neurons receiving external inputs at more than one state, particularly at states leading up to its original field location.

2.6. RNN-S estimates successor features even with naturalistic trajectories.

We ask whether RNN-S can accurately estimate successor features, particularly under conditions of natural behavior. Specifically, we used the dataset from Payne et al. [11, 53], gathered from foraging Tufted Titmice in a 2D arena (Figure 5a). We discretize the arena into a set of states and encode each state as a randomly drawn feature ϕ. Using position-tracking data from Payne et al. [11, 53], we simulate the behavioral trajectory of the animal as transitions through the discrete state space. The inputs into the successor feature model are the features associated with the states in the behavioral trajectory.

We first wanted to test whether the RNN-S model was robust across a range of different types of input features. We calculate the TD loss of the model as a function of the spatial correlation across inputs $\boldsymbol{\phi}$ (Figure 5b). We find that the model performs well across a range of inputs but loss is higher when inputs are spatially uncorrelated. This is consistent with the observation that behavioral transitions are spatially local, such that correlations across spatially adjacent features aid in the predictive power of the model. We next examine the

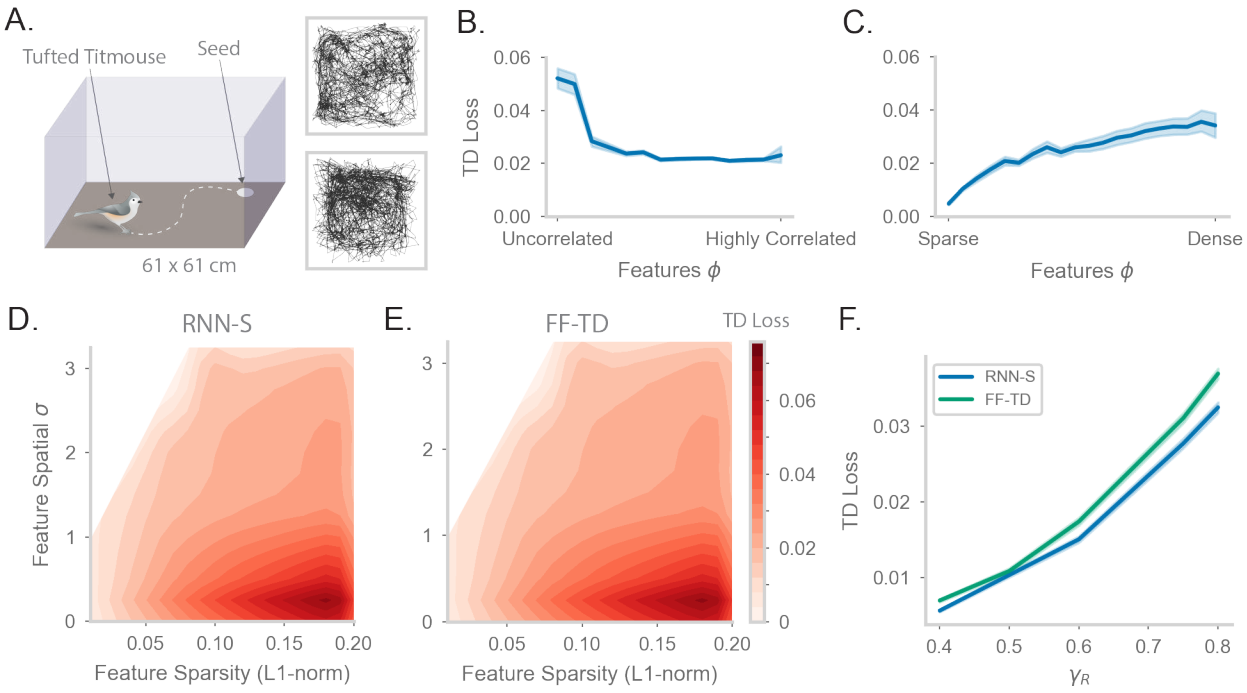

Figure 5: **Fitting successor features to data with RNN-S over a variety of feature encodings. a.** We use behavioral data from Payne et al, where a Tufted Titmouse randomly forages in a 2D environment while electrophysiological data is collected (replicated with permission from authors). Two example trajectories are shown on the right. **b.** Temporal difference (TD) loss versus the spatial correlation of the input dataset, aggregated over all sparsity levels. Here, $\gamma_R = 0.75$. **c.** As in (b), but measuring TD loss versus the sparsity level of the input dataset, aggregated over all spatial correlation levels. **d.** TD loss for RNN-S with datasets with different spatial correlations and sparsities. White areas were not represented in the input dataset due to the feature generation process. Here, $\gamma_R = 0.75$ **e.** As in (g), but for FF-TD. **f.** TD loss of each model as a function of γ_R, aggregated over all input encodings.

model performance as a function of the sparsity of inputs $\boldsymbol{\phi}$ (Figure 5c). We find the model also performs well across a range of feature sparsity, with lowest loss when features are sparse.

To understand the interacting effects of spatial correlation and feature sparsity in more detail, we performed a parameter sweep over both of these parameters (Figure 5d, Figure S5a-e). We generated random patterns according to the desired sparsity and smoothness with a spatial filter to generate correlations. This means that the entire parameter space is not covered in our sweep (e.g., the top-left area with high correlation and high sparsity is not explored). Note that since we generate $\boldsymbol{\phi}$ by randomly drawing patterns, the special case of one-hot encoding is also not included in the parameter sweep (one-hot encoding is already explored in Figure 2). The RNN-S seems to perform well across a wide range, with highest loss in regions of low spatial correlation and low sparsity.

We want to compare the TD loss of RNN-S to that of a non-biological model designed to minimized TD loss. We repeat the same parameter sweep over input features with the FF-TD model (Figure 5e, Figure S5f). The FF-TD model performs similarly to the RNN-S model, with lower TD loss in regions with low sparsity or higher correlation. We also tested how the performance of both models is affected by the strength of γ_R (Figure 5f). Both models show a similar increase in TD loss as γ_R increases, although the RNN-S has a slightly lower TD loss at high γ than the FF-TD model. Unlike in the one-hot case, there is no ground-truth T matrix for non-one-hot inputs, so representations generated by RNN-S and FF-TD may look different, even at the same TD loss. Therefore, to compare the two models, it is important to compare representations to neural data.

2.7. RNN-S fits neural data in a random foraging task.

Finally, we tested whether the neural representations learned by the models with behavioral trajectories from Figure 5 match hippocampal firing patterns. We performed new analysis on neural data from Payne et al. [11, 53] to establish a dataset for comparison. The neural data from Payne et al. [11] was collected from electrophysiological recordings in titmouse hippocampus during freely foraging behavior (Figure 6a). Payne et al. discovered the presence of place cells in this area. We analyzed statistics of place cells recorded in the anterior region of the hippocampus, where homology with rodent dorsal hippocampus is hypothesized [54]. We calculated the distribution of place field size measured relative to the arena size (Figure 6b), as well as the distribution of the number of place fields per place cell (Figure 6c). Interestingly, with similar analysis methods, Henriksen et al. [55] see similar statistics in the proximal region of dorsal CA1 in rats, indicating that our analyses could be applicable across organisms.

In order to test how spatial representations in the RNN-S are impacted by input features, we performed parameter sweeps over input statistics. As in [11], we define place cells in the model as cells with at least one statistically significant place field under permutation tests. Under most of the parameter range, all RNN-S neurons would be identified as a place cell (Figure 6d). However, under conditions of high spatial correlation and low sparsity, a portion of neurons (12%) do not have any fields in the environment. These cells are excluded

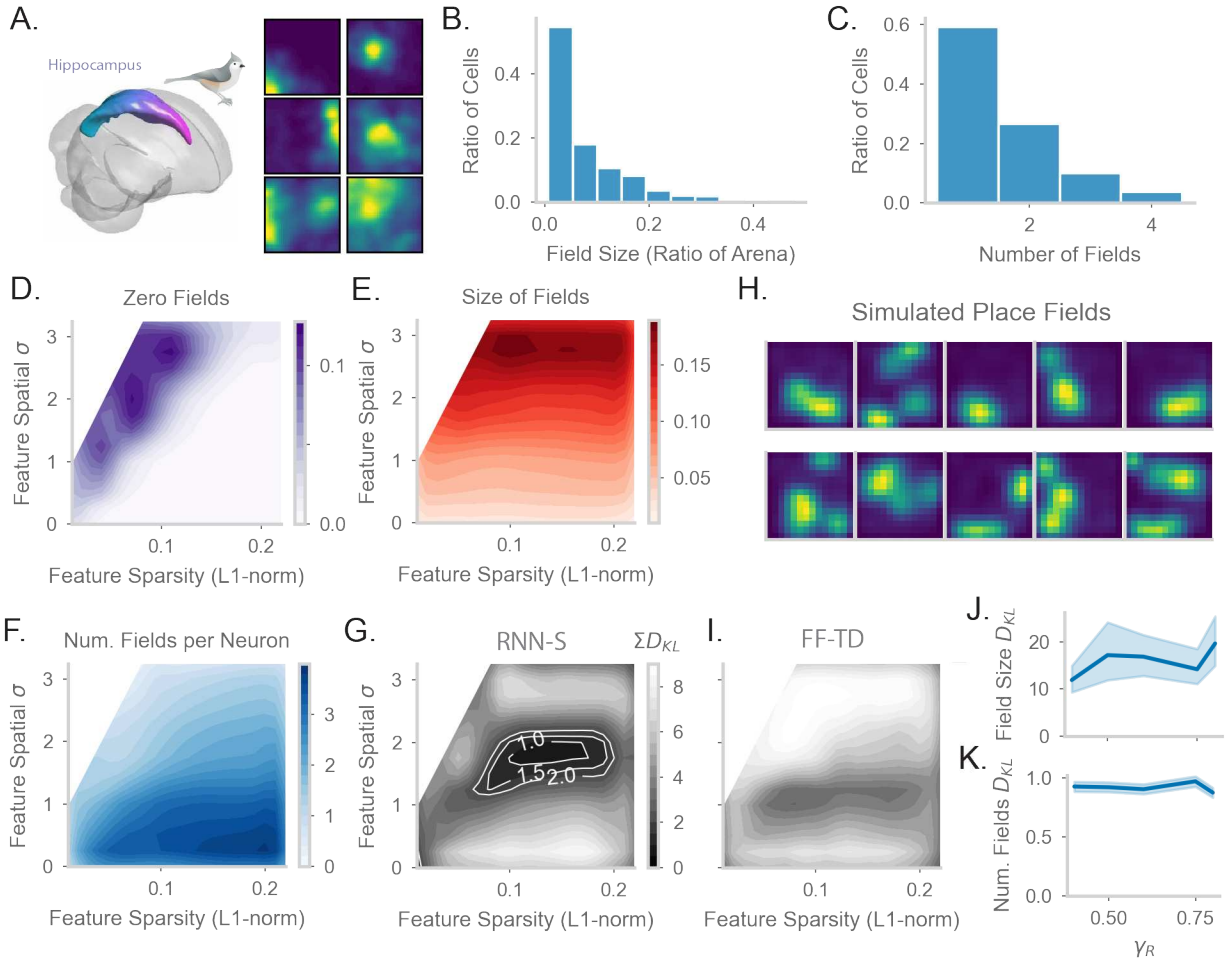

Figure 6: **Comparing place fields from RNN-S to data. a.** Dataset is from Payne et al, where a Tufted Titmouse randomly forages in a 2D environment while electrophysiological data is collected (replicated with permission from authors). **b.** Distribution of place cells with some number of fields, aggregated over all cells recorded in all birds. **c.** Distribution of place cells with some field size as a ratio of the size of the arena, aggregated over all cells recorded in all birds. **d.** Average proportion of non-place cells in RNN-S, aggregated over simulations of randomly drawn trajectories from Payne et al. Feature encodings are varied by spatial correlation and sparsity as in Figure 5. **e.** As in (d), but for average field size of place cells. **f.** As in (d), but for average number of fields per place cell. **g.** As in (d) and (e), but comparing place cell statistics using the KL divergence (D_{KL}) between RNN-S and data from Payne et al. At each combination of input spatial correlation and sparsity, the distribution of field sizes is compared to the neural data, as is the distribution of number of fields per neuron, then the two D_{KL} values are summed. Contour lines are drawn at D_{KL} values of 1, 1.5, and 2 bits. **h.** Place fields of cells chosen from the region of lowest KL divergence. **i.** As in (g) but for FF-TD. **j.** Change in KL divergence for field size (top) and number of fields (bottom) as function of γ.

from further analysis. We measured how the size of place fields varies across the parameter range (Figure 6e). The size of the fields increases as a function of the spatial correlation of the inputs, but is relatively insensitive to sparsity. This effect can be explained as the spatial correlation of the inputs introducing an additional spatial spread in the neural activity.

Similarly, we measured how the number of place fields per cell varies across the parameter range (Figure 6f). The number of fields is maximal for conditions in which input features are densely encoded and spatial correlation is low. These are conditions in which each neuron receives inputs from multiple, spatially distant states.

Finally, we wanted to identify regions of parameter space that were similar to the data of Payne et al. [11, 53]. We measured the KL divergence between our model's place field statistics (Figure 6de) and the statistics measured in Payne et al. [11] (Figure 6bc). We combined the KL divergence of both these distributions to find the parameter range in which the RNN-S best fits neural data (Figure 6g). This optimal parameter range occurs when inputs have a spatial correlation of $\sigma \approx 8.75$ cm and sparsity ≈ 0.15. We can visually confirm that the model fits the data well by plotting the place fields of RNN-S neurons (Figure 6h).

We wondered whether the predictive gain (γ_R) of the representations affects the ability of the RNN-S to fit data. The KL divergence changes only slightly as a function of γ_R. Mainly, the KL-divergence of the place field size increases as γ_R increases (Figure 6i), but little effect is seen in the distribution of the number of place fields per neuron (Figure 6j).

We next tested whether the neural data was better fit by representations generated by RNN-S or the FF-TD model. Across all parameters of the input features, despite having similar TD loss (Figure 5de), the FF-TD model has much higher divergence from neural data (Figure 6gi, Figure S6).

Overall, our RNN-S model seems to strike a balance between performance in estimating successor features, similarity to data, and biological plausibility. Furthermore, our analyses provide a prediction of the input structure into the hippocampus that is otherwise not evident in an algorithmic description or in a model that only considers one-hot feature encodings.

3. Discussion

Hippocampal memory is thought to support a wide range of cognitive processes, especially those that involve forming associations or making predictions. However, the neural mechanisms that underlie these computations in the hippocampus are not fully understood. A promising biological substrate is the recurrent architecture of the CA3 region of the hippocampus and the plasticity rules observed. Here, we showed how a recurrent network

with local learning rules can implement the successor representation, a predictive algorithm that captures many observations of hippocampal activity. We used our neural circuit model to make specific predictions of biological processes in this region.

A key component of our plasticity rule is a decorrelative term that depresses synapses based on coincident activity. Such anti-Hebbian or inhibitory effects are hypothesized to be broadly useful for learning, especially in unsupervised learning with overlapping input features [56, 57, 58]. Consistent with this hypothesis, anti-Hebbian learning has been implicated in circuits that perform a wide range of computations, from distinguishing patterns, [37], to familiarity detection [38], to learning birdsong syllables [59]. This inhibitory learning may be useful because it decorrelates redundant information, allowing for greater specificity and capacity in a network [57, 37]. Our results provide further support of these hypotheses and predict that anti-Hebbian learning is fundamental to a predictive neural circuit.

We derive an adaptive learning rate that allows our model to quickly learn a probability distribution, and generally adds flexibility to the learning process. The adaptive learning rate changes such that neurons that are more recently active have a slower learning rate. This is consistent with experimental findings of metaplasticity at synapses [60, 61, 62], and theoretical proposals that metaplasticity tracks the uncertainty of information [36]. In RNN-S, the adaptive learning rate improves the speed of learning and better recapitulates hippocampal data. Our adaptive learning rate also has interesting implications for flexible learning. Memory systems must be able to quickly learn new associations throughout their lifetime without catastrophe. Our learning rate is parameterized by a forgetting term λ that controls the timescale in which environmental statistics are expected to be stationary. Although we fixed $\lambda = 1$ in our simulations, there are computational benefits in considering cases where $\lambda < 1$. This parameter provides a natural way for a memory system to forget gradually over time and prioritize recent experiences, in line with other theoretical studies that have also suggested that learning and forgetting on multiple timescales allow for more flexible behavior [63, 64].

We tested the sensitivity of our network to various parameters and found a broad range of valid solutions. Prior work has sought to understand how an emergent property of a network could be generated by multiple unique solutions [65, 66, 67, 68]. It has been suggested that

redundancy in solution space makes systems more robust, accounting for margins of error in the natural world [69, 70]. In a similar vein, our parameter sweep over plasticity kernels revealed that a sizeable variety of kernels give solutions that resemble the SR. Although our model was initially sensitive to the value of γ, we found that adding biological components, such as nonlinear dynamics and separate network modes, broadened the solution space of the network.

Several useful features arise from the fact that RNN-S learns the transition matrix T directly, while separating out the prediction timescale, γ, as a global gain factor. It is important for animals to engage in different horizons of prediction depending on task or memory demands [71, 72]. In RNN-S, changing the prediction time horizon is as simple as increasing or decreasing the global gain of the network. Mechanistically, this could be accomplished by a neuromodulatory gain factor that boosts γ, perhaps by increasing the excitability of all neurons [73, 74]. In RNN-S, it was useful to have low network gain during learning (γ_B), while allowing higher gain during retrieval to make longer timescale predictions (γ_R). This could be accomplished by a neuromodulatory factor that switches the network into a learning regime [75, 76], for example Acetylcholine, which reduces the gain of recurrent connections and increases learning rates [77, 78]. The idea that the hippocampus might compute the SR with flexible γ could help reconcile recent results that hippocampal activity does not always match high-γ SR [79, 80]. Finally, estimating T directly provides RNN-S with a means to sample likely future trajectories, or distributions of trajectories, which is computationally useful for many memory-guided cognitive tasks beyond reinforcement learning, including reasoning and inference [81]. We also found that the recurrent network fit hippocampal data better than a feedforward network. An interesting direction for further work involves untangling which brain areas and cognitive functions can be explained by deep (feed forward) neural networks [82], and which rely on recurrent architectures, or even richer combinations of generative structures [83]. Recurrent networks, such as RNN-S, support generative sequential sampling, reminiscent of hippocampal replay, which has been proposed as a substrate for planning, imagination, and structural inference [84, 85, 86, 87, 88].

Other recent theoretical works have also sought to find biological mechanisms to learn successor representations, albeit with different approaches [89, 90, 91, 92, 93]. The model

from George et al. [93] focuses on a feedforward architecture, using STDP and theta phase precession to learn the SR. It is important to note that these mechanisms are not mutually exclusive with RNN-S. Taken together with our work, these models suggest that there are multiple ways to learn the SR in a biological circuit and that these representations may be more accessible to neural circuits than previously thought.

4. Methods

4.1. Code availability

Code is posted on Github: https://github.com/chingf/sr-project

4.2. Random walk simulations

We simulated random walks in 1D (circular track) and 2D (square) arenas. In 1D simulations, we varied the probability of staying in the current state and transitioning forwards or backwards to test different types of biases on top of a purely random walk. In 2D simulations, the probabilities of each possible action were equal. In our simulations, one timestep corresponds to $\frac{1}{3}$ second and spatial bins are assumed to be 5 cm apart. This speed of movement (15 cm/sec) was chosen to be consistent with previous experiments. In theory, one can imagine different choices of timestep size to access different time horizons of prediction– that is, the choice of timestep interacts with the choice of γ in determining the prediction horizon.

4.3. RNN-S model

This section provides details and pseudocode of the RNN-S simulation. Below are explanations of the most relevant variables:

J	$(N \times N)$ synaptic weight matrix
M	$(N \times N)$ SR matrix
$\boldsymbol{\phi}$	N-length input vector into network
b	binary variable indicating learning (0) or retrieval (1) mode
γ_B	Value of γ the network uses to calculate M in learning mode
γ_R	Value of γ the network uses to calculate M in retrieval mode
$\boldsymbol{n}$	Variable that tracks the activity of neurons integrated over time
λ	Discount value the network uses to calculate $\boldsymbol{n}$
$\boldsymbol{\eta}$	Learning rates of neurons

The RNN-S algorithm is as follows:

4.4. RNN-S with plasticity kernels

We introduce additional kernel-related variables to the RNN-S model above that are optimized by an evolutionary algorithm (see following methods subsection for more details):

A_+, τ_+	pre$\rightarrow$ post side of the kernel as $K_+(t) = A_+ E^{-t/\tau_+}$
A_-, τ_-	As above, but for the post$\rightarrow$ pre side
α_d	Scaling term to allow for different self-synapse updates
α_n	Scaling term to allow for different learning rate updates

We also define the variable $t_k = 20$, which is the length of the temporal support for the plasticity kernel. The value of t_k was chosen such that $e^{-t_k/\tau}$ was negligibly small for the range of τ we were interested in. The update algorithm is the same as in Algorithm 1, except lines 15-16 are replaced with the following:

Algorithm 1 RNN-S

Inputs:

  $\boldsymbol{\phi}(t)$ for $t \in 1, \ldots, T$

  $b(t)$ for $t \in 1, \ldots T$

Initialize:

  $J \leftarrow \mathbf{0}_{N \times N}$

  $\boldsymbol{n} \leftarrow \mathbf{0}_N$

  $\boldsymbol{x}(t) \leftarrow \mathbf{0}_N$ for $t \in 1, \ldots, T$

for $t \in 1, \ldots, T$ **do**

  if $b(t) == 1$ **then** ▷ Retrieval Mode

    $M^\intercal \leftarrow (1 - \gamma_R J)^{-1}$

    $\boldsymbol{x}(t) \leftarrow M^\intercal \boldsymbol{\phi}(t)$

  else ▷ Learning Mode

    $M^\intercal \leftarrow (1 - \gamma_B J)^{-1}$

    $\boldsymbol{x}(t) \leftarrow M^\intercal \boldsymbol{\phi}(t)$

    $\boldsymbol{n} \leftarrow \boldsymbol{x}(t) + \lambda \boldsymbol{n}$ ▷ Learning rate update

    $\Delta J \leftarrow \boldsymbol{x}(t)\boldsymbol{x}(t-1)^\intercal - (J\boldsymbol{x}(t-1))\boldsymbol{x}(t-1)^\intercal$ ▷ Calculate weight update

    $\boldsymbol{\eta} = \frac{1}{\boldsymbol{n}}$ ▷ Get learning rates (elementwise inversion)

    $\boldsymbol{\eta} = \texttt{min}(\boldsymbol{\eta}, 1.0)$ ▷ Learning rates can't exceed 1.0

    $J_{ij} \leftarrow J_{ij} + \eta_j \Delta J_{ij}$ ▷ Update synaptic weight matrix

  end if

end for

return $\boldsymbol{x}$

4.5. Metalearning of RNN parameters

To learn parameters of the RNN-S model, we use covariance matrix adaptation evolution strategy (CMA-ES) to learn the parameters of the plasticity rule. The training data provided are walks simulated from a random distribution of 1D walks. Walks varied in the number of states, the transition statistics, and the number of timesteps simulated. The loss function was the mean-squared error (MSE) loss between the RNN J matrix and the ideal estimated

Algorithm 2 Plasticity kernel update

1: $\boldsymbol{n} \leftarrow \alpha_n \boldsymbol{x} + \lambda \boldsymbol{n}$ ▷ Learning rate update

2: $\boldsymbol{k}_+ \leftarrow A_+ \sum_{t'=0}^{t_k} \boldsymbol{x}(t-t') e^{-t'/\tau_+}$ ▷ Convolution with plasticity kernel

3: $\boldsymbol{k}_- \leftarrow A_- \sum_{t'=0}^{t_k} \boldsymbol{x}(t-t') e^{-t'/\tau_-}$

4: $\Delta J_K \leftarrow \boldsymbol{x}(t)\boldsymbol{k}_+^\intercal + \boldsymbol{k}_- \boldsymbol{x}(t)^\intercal$ ▷ Calculate contribution to update from plasticity kernel

5: $\Delta J_K[ii] \leftarrow \alpha_d \boldsymbol{x}(t)\boldsymbol{k}_+^\intercal$ ▷ Updates to self-synapses use separate scaling

6: $\Delta J \leftarrow \Delta J_K - (J\boldsymbol{x})\boldsymbol{x}^\intercal$ ▷ Calculate weight update

T matrix at the end of the walk.

4.6. RNN-S with truncated recurrent steps and nonlinarity

For the RNN-S model with t_{max} recurrent steps, lines 10 and 13 in algorithm 1 is replaced with $M^\intercal \leftarrow \sum_{t=0}^{t_{max}} \gamma^t J^t$.

For RNN-S with nonlinear dynamics, there is no closed form solution. So, we select a value for t_{max} and replace lines 10 and 13 in algorithm 1 with an iterative update for t_{max} steps: $\Delta \boldsymbol{x} = -\boldsymbol{x} + \gamma \texttt{tanh}(J\boldsymbol{x}') + \boldsymbol{\phi}$. We choose t_{max} such that $\gamma_{max}^t < 10^{-4}$.

4.7. RNN-S with successor features

We use $\gamma_B = 0$ and a tanh nonlinearity as in Methods 4.6. For simplicity, we set $\gamma_B = 0$.

4.8. RNN-S with independent normalization

As in algorithm 1, but with the following in place of line 16

$$\Delta J_{ij} \leftarrow x_i(t)x_j(t-1) - J_{ij}x_j(t-1)^2 \tag{8}$$

4.9. FF-TD Model

In all simulations of the FF-TD model, we use the temporal difference update. We perform a small grid search over the learning rate η to minimize error (for SR, this is the MSE between the true M and estimated M; for successor features, this is the temporal difference error). In the one-hot SR case, the temporal difference update given an observed transition from state s to state s' is:

$$\Delta M_{ji} = \begin{cases} \gamma M_{s'i} - M_{si} & \text{if } s = j \neq i \\ 1 + \gamma M_{s'i} - M_{si} & \text{if } s = j = i \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

for all synapses $j \to i$. Given arbitrarily strucutred inputs (as in the successor feature case), the temporal difference update is:

$$\Delta M^{\intercal} = \eta \Big(\boldsymbol{\phi} + \gamma M \boldsymbol{\phi}' - M \boldsymbol{\phi} \Big) \boldsymbol{\phi}^{\intercal} \tag{10}$$

or, equivalently,

$$\Delta M_{ji} = \eta \Big(\phi_i + \gamma \sum_k M_{ki} \phi'_k - \sum_k M_{ki} \phi_k \Big) \phi_j \tag{11}$$

4.10. Generation of feature encodings for successor feature models

For a walk with n states, we created n-dimensional feature vectors for each state. We choose an initial sparsity probability p and create feature vectors as random binary vectors with probability p of being "on". The feature vectors were then blurred by a 2D Gaussian filter with variance σ with 1 standard deviation of support. The blurred features were then min-subtracted and max-normalized. The sparsity of each feature vector was calculated as the L1 norm divided by N. The sparsity s of the dataset then was the median of all the sparsity values computed from the feature vectors. To vary the spatial correlation of the dataset we need only vary σ. To vary the sparsity s of the dataset we need to vary p, then measure the final s after blurring with σ. Note that, at large σ, the lowest sparsity values in our parameter sweep were not possible to achieve.

4.11. Measuring TD loss for successor feature models.

We use the standard TD loss function (equation S7). To measure TD loss, at the end of the walk we take a random sample of observed transition pairs $(\boldsymbol{\phi}, \boldsymbol{\phi}')$. We use these transitions as the dataset to evaluate the loss function.

4.12. Analysis of place field statistics

We use the open source dataset from Payne et al. [11, 53]. We select for excitatory cells in the anterior tip of the hippocampus. We then select for place cells using standard measures (significantly place-modulated and stable over the course of the experiment).

We determined place field boundaries with a permutation test as in Payne et al. [11]. We then calculated the number of fields per neuron and the field size as in Henriksen et al. [55]. The same analyses were conducted for simulated neural data from the RNN-S and FF-TD models.

4.13. Behavioral simulation of Payne et al.

We use behavioral tracking data from Payne et al. [11]. For each simulation, we randomly select an experiment and randomly sample a 28 minute window from that experiment. If the arena coverage is less than 85% during the window, we redo the sampling until the coverage requirement is satisfied. We then downsample the behavioral data so that the frame rate is the same as our simulation (3 FPS). Then, we divide the arena into a 14×14 grid. We discretize the continuous X/Y location data into these states. This sequence of states makes up the behavioral transitions that the model simulates.

4.14. Place field plots

From the models, we get the activity of each model neuron over time. We make firing field plots with the same smoothing parameters as Payne et al. [11].

Acknowledgements

This work was supported through NSF NeuroNex Award DBI-1707398, the Gatsby Charitable Foundation, the New York Stem Cell Foundation (Robertson Neuroscience Investigator Award), National Institutes of Health (NIH Director's New Innovator Award (DP2-AG071918)), and the Arnold and Mabel Beckman Foundation (Beckman Young Investigator Award). CF received support from the NSF Graduate Research Fellowship Program. ELM received support from the Simons Society of Fellows. We thank Jack Lindsey and Tom George for comments on the manuscript, as well as Stefano Fusi, William de Cothi, Kimberly Stachenfeld, and Caswell Barry for helpful discussions.

Citation diversity statement

Systemic discriminatory practices have been identified in neuroscience citations, and a 'citation diversity statement' has been proposed as an intervention [94, 95]. There is evidence that quantifying discriminatory practices can lead to systemic improvements in academic settings [96]. Many forms of discrimination could lead to a paper being under-cited, for example authors being less widely known or less respected due to discrimination related to gender, race, sexuality, disability status, or socioeconomic background. We manually estimated the number of male and female first and last authors that we cited, acknowledging that this quantification ignores many known forms of discrimination, and fails to account for nonbinary/intersex/trans folks. In our citations, first-last author pairs were 64% male-male, 21% female-male, 6% male-female, and 9% female-female, somewhat similar to base rates in our field (biaswatchneuro.com). To familiarize ourselves with the literature, we used databases intended to counteract discrimination (blackinneuro.com, anneslist.net, connectedpapers.com). The process of making this statement improved our paper, and encouraged us to adopt less biased practices in selecting what papers to read and cite in the future. We were somewhat surprised and disappointed at how low the number of female authors were, despite being a female-female team ourselves. Citation practices alone are not enough to correct the power imbalances endemic in academic practice [97] — this requires corrections to how concrete power and resources are distributed.

References

[1] W. B. Scoville, B. Milner, Loss of recent memory after bilateral hippocampal lesions, Journal of neurology, neurosurgery, and psychiatry 20 (1) (1957) 11.

[2] W. Penfield, B. Milner, Memory deficit produced by bilateral lesions in the hippocampal zone, AMA Archives of Neurology & Psychiatry 79 (5) (1958) 475–497.

[3] S. Corkin, What's new with the amnesic patient hm?, Nature reviews neuroscience 3 (2) (2002) 153–160.

[4] A. Bubic, D. Y. Von Cramon, R. Schubotz, Prediction, cognition and the brain, Frontiers

in Human Neuroscience 4 (2010). doi:10.3389/fnhum.2010.00025.
URL https://www.frontiersin.org/article/10.3389/fnhum.2010.00025

[5] G. Wayne, C.-C. Hung, D. Amos, M. Mirza, A. Ahuja, A. Grabska-Barwinska, J. Rae, P. Mirowski, J. Z. Leibo, A. Santoro, et al., Unsupervised predictive memory in a goal-directed agent, arXiv preprint arXiv:1803.10760 (2018).

[6] J. C. Whittington, T. H. Muller, S. Mark, G. Chen, C. Barry, N. Burgess, T. E. Behrens, The tolman-eichenbaum machine: Unifying space and relational memory through generalization in the hippocampal formation, Cell 183 (5) (2020) 1249–1263.

[7] I. Momennejad, Learning structures: predictive representations, replay, and generalization, Current Opinion in Behavioral Sciences 32 (2020) 155–166.

[8] W. Skaggs, B. McNaughton, Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience, Science (1996). doi:10.1126/science.271.5257.1870.

[9] J. Lisman, A. D. Redish, Prediction, sequences and the hippocampus, Philosophical Transactions of the Royal Society B: Biological Sciences 364 (1521) (2009) 1193–1201.

[10] M. R. Mehta, C. A. Barnes, B. L. McNaughton, Experience-dependent, asymmetric expansion of hippocampal place fields, Proceedings of the National Academy of Sciences 94 (16) (1997) 8918–8921.

[11] H. Payne, G. Lynch, D. Aronov, Neural representations of space in the hippocampus of a food-caching bird, Science 373 (6552) (2021) 343–348.

[12] R. U. Muller, J. L. Kubie, The firing of hippocampal place cells predicts the future position of freely moving rats, in: The Journal of neuroscience : the official journal of the Society for Neuroscience, 1989.

[13] B. E. Pfeiffer, D. J. Foster, Hippocampal place-cell sequences depict future paths to remembered goals, Nature 497 (7447) (2013) 74–79.

[14] A. C. Schapiro, N. B. Turk-Browne, K. A. Norman, M. M. Botvinick, Statistical learning of temporal community structure in the hippocampus, Hippocampus 26 (1) (2016) 3–8.

[15] M. M. Garvert, R. J. Dolan, T. E. Behrens, A map of abstract relational knowledge in the human hippocampal–entorhinal cortex, Elife 6 (2017) e17086.

[16] K. I. Blum, L. F. Abbott, A model of spatial map formation in the hippocampus of the rat, Neural computation 8 (1) (1996) 85–93.

[17] M. R. Mehta, M. C. Quirk, M. A. Wilson, Experience-dependent asymmetric shape of hippocampal receptive fields, Neuron 25 (3) (2000) 707–715.

[18] K. Stachenfeld, M. Botvinick, S. Gershman, The hippocampus as a predictive map, Nature Neuroscience (2017). doi:10.1038/nn.4650.

[19] I. Momennejad, E. M. Russek, J. H. Cheong, M. M. Botvinick, N. D. Daw, S. J. Gershman, The successor representation in human reinforcement learning, Nature human behaviour 1 (9) (2017) 680–692.

[20] J. P. Geerts, F. Chersi, K. L. Stachenfeld, N. Burgess, A general model of hippocampal and dorsal striatal learning and decision making, Proceedings of the National Academy of Sciences 117 (49) (2020) 31427–31437.

[21] S. Recanatesi, M. Farrell, G. Lajoie, S. Deneve, M. Rigotti, E. Shea-Brown, Predictive learning as a network mechanism for extracting low-dimensional latent space representations, Nature communications 12 (1) (2021) 1–13.

[22] D. George, R. V. Rikhye, N. Gothoskar, J. S. Guntupalli, A. Dedieu, M. Lázaro-Gredilla, Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps, Nature communications 12 (1) (2021) 1–17.

[23] P. Dayan, Improving generalization for temporal difference learning: The successor representation, Neural Computation 5 (4) (1993) 613–624.

[24] S. J. Gershman, C. D. Moore, M. T. Todd, K. A. Norman, P. B. Sederberg, The successor representation and temporal context, Neural Computation 24 (6) (2012) 1553–1568.

[25] M. G. Mattar, N. D. Daw, Prioritized memory access explains planning and hippocampal replay, Nature neuroscience 21 (11) (2018) 1609–1617.

[26] E. J. Markus, Y.-L. Qin, B. Leonard, W. E. Skaggs, B. L. McNaughton, C. A. Barnes, Interactions between location and task affect the spatial and directional firing of hippocampal neurons, Journal of Neuroscience 15 (11) (1995) 7079–7094.

[27] K. J. Jeffery, How environmental movement constraints shape the neural code for space, Cognitive processing 22 (1) (2021) 97–104.

[28] D. Marr, T. Poggio, From understanding computation to understanding neural circuitry (1976).

[29] M. Frank, Linking across levels of computation in model-based cognitive neuroscience, in: An Introduction to Model-Based Cognitive Neuroscience, Springer, New York, NY, 2015. doi:https://doi.org/10.1007/978-1-4939-2236-9.

[30] B. C. Love, Levels of biological plausibility, Philosophical Transactions of the Royal Society B 376 (1815) (2021) 20190632.

[31] F. Zeldenrust, B. Gutkin, S. Denéve, Efficient and robust coding in heterogeneous recurrent networks, PLoS computational biology 17 (4) (2021) e1008673.

[32] P. Karimi, S. Golkar, J. Friedrich, D. Chklovskii, Learning a biologically plausible linear controller for nonlinear systems, Bulletin of the American Physical Society (2022).

[33] C. Pehlevan, S. Mohan, D. B. Chklovskii, Blind nonnegative source separation using biological neural networks, Neural computation 29 (11) (2017) 2925–2954.

[34] B. A. Olshausen, D. J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, Nature 381 (6583) (1996) 607–609.

[35] K. S. Burbank, Mirrored stdp implements autoencoder learning in a network of spiking neurons, PLoS computational biology 11 (12) (2015) e1004566.

[36] L. Aitchison, J. Jegminat, J. A. Menendez, J.-P. Pfister, A. Pouget, P. E. Latham, Synaptic plasticity as bayesian inference, Nature neuroscience 24 (4) (2021) 565–571.

[37] P. Földiak, Forming sparse representations by local anti-hebbian learning, Biological cybernetics 64 (2) (1990) 165–170.

[38] D. Tyulmankov, G. R. Yang, L. Abbott, Meta-learning synaptic plasticity and memory addressing for continual familiarity detection, Neuron 110 (3) (2022) 544–557.

[39] E. Vértes, M. Sahani, A neurally plausible model learns successor representations in partially observable environments, Advances in Neural Information Processing Systems 32 (2019).

[40] E. Russek, M. I, M. Botvinick, S. Gershman, N. Daw, Predictive representations can link model-based reinforcement learning to model-free mechanisms, PLoS Comput Biol (2017). doi:10.1371/journal.pcbi.1005768.

[41] S.-I. Amari, Characteristics of random nets of analog neuron-like elements, IEEE Transactions on Systems, Man, and Cybernetics SMC-2 (5) (1972) 643–657. doi:10.1109/TSMC.1972.4309193.

[42] H. Sompolinsky, A. Crisanti, H. J. Sommers, Chaos in random neural networks, Phys. Rev. Lett. 61 (1988) 259–262. doi:10.1103/PhysRevLett.61.259.
URL https://link.aps.org/doi/10.1103/PhysRevLett.61.259

[43] G.-q. Bi, M.-m. Poo, Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type, Journal of neuroscience 18 (24) (1998) 10464–10472.

[44] L. Abbott, K. Blum, Functional significance of long-term potentiation for sequence learning and prediction., Cereb Cortex (1996). doi:10.1093/cercor/6.3.406.

[45] T. Zhang, M. Rosenberg, P. Perona, M. Meister, Endotaxis: A universal algorithm for mapping, goal-learning, and navigation, bioRxiv (2021). arXiv:https://www.biorxiv.org/content/early/2021/09/25/2021.09.24.461751.full.pdf, doi:10.1101/2021.09.24.461751.
URL https://www.biorxiv.org/content/early/2021/09/25/2021.09.24.461751

[46] J. D. Monaco, G. Rao, E. D. Roth, J. J. Knierim, Attentive scanning behavior drives one-trial potentiation of hippocampal place fields, Nature neuroscience 17 (5) (2014) 725–731.

[47] M. E. Sheffield, D. A. Dombeck, Calcium transient prevalence across the dendritic arbour predicts place field properties, Nature 517 (7533) (2015) 200–204.

[48] K. C. Bittner, C. Grienberger, S. P. Vaidya, A. D. Milstein, J. J. Macklin, J. Suh, S. Tonegawa, J. C. Magee, Conjunctive input processing drives feature selectivity in hippocampal ca1 neurons, Nature neuroscience 18 (8) (2015) 1133–1142.

[49] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, D. Silver, Successor features for transfer in reinforcement learning, Advances in neural information processing systems 30 (2017).

[50] K. Hardcastle, N. Maheswaranathan, S. Ganguli, L. M. Giocomo, A multiplexed, heterogeneous, and adaptive code for navigation in medial entorhinal cortex, Neuron 94 (2) (2017) 375–387.

[51] T. D. Kulkarni, A. Saeedi, S. Gautam, S. J. Gershman, Deep successor reinforcement learning, arXiv preprint arXiv:1606.02396 (2016).

[52] E. Oja, Simplified neuron model as a principal component analyzer, Journal of mathematical biology 15 (3) (1982) 267–273.

[53] H. Payne, G. Lynch, D. Aronov, Neural representations of space in the hippocampus of a food-caching bird, Dataset (2021). doi:https://doi.org/10.5061/dryad.pg4f4qrp7.

[54] M. A. Tosches, T. M. Yamawaki, R. K. Naumann, A. A. Jacobi, G. Tushev, G. Laurent, Evolution of pallium, hippocampus, and cortical cell types revealed by single-cell transcriptomics in reptiles, Science 360 (6391) (2018) 881–888.

[55] E. J. Henriksen, L. L. Colgin, C. A. Barnes, M. P. Witter, M.-B. Moser, E. I. Moser, Spatial representation along the proximodistal axis of ca1, Neuron 68 (1) (2010) 127–137.

669 [56] A. Litwin-Kumar, B. Doiron, Formation and maintenance of neuronal assemblies
670 through synaptic plasticity, Nature communications 5 (1) (2014) 1–12.

671 [57] S. Sadeh, C. Clopath, Excitatory-inhibitory balance modulates the formation and
672 dynamics of neuronal assemblies in cortical networks, Science advances 7 (45) eabg8411.

673 [58] C. Pehlevan, A. M. Sengupta, D. B. Chklovskii, Why do similarity matching objectives
674 lead to hebbian/anti-hebbian networks?, Neural computation 30 (1) (2017) 84–124.

675 [59] E. L. Mackevicius, M. T. Happ, M. S. Fee, An avian cortical circuit for chunking tutor
676 song syllables into simple vocal-motor units, Nature communications 11 (1) (2020)
677 1–16.

678 [60] W. C. Abraham, M. F. Bear, Metaplasticity: the plasticity of synaptic plasticity, Trends
679 in neurosciences 19 (4) (1996) 126–130.

680 [61] W. C. Abraham, Metaplasticity: tuning synapses and networks for plasticity, Nature
681 Reviews Neuroscience 9 (5) (2008) 387–387.

682 [62] S. R. Hulme, O. D. Jones, C. R. Raymond, P. Sah, W. C. Abraham, Mechanisms
683 of heterosynaptic metaplasticity, Philosophical Transactions of the Royal Society B:
684 Biological Sciences 369 (1633) (2014) 20130148.

685 [63] C. Kaplanis, M. Shanahan, C. Clopath, Continual reinforcement learning with complex
686 synapses, in: International Conference on Machine Learning, PMLR, 2018, pp. 2497–
687 2506.

688 [64] S. Fusi, W. F. Asaad, E. K. Miller, X.-J. Wang, A neural circuit model of flexible
689 sensorimotor mapping: learning and forgetting on multiple timescales, Neuron 54 (2)
690 (2007) 319–333.

691 [65] M. S. Goldman, J. Golowasch, E. Marder, L. Abbott, Global structure, robustness, and
692 modulation of neuronal models, Journal of Neuroscience 21 (14) (2001) 5229–5238.

693 [66] A. A. Prinz, D. Bucher, E. Marder, Similar network activity from disparate circuit
694 parameters, Nature neuroscience 7 (12) (2004) 1345–1352.

[67] S. R. Bittner, A. Palmigiano, A. T. Piet, C. A. Duan, C. D. Brody, K. D. Miller, J. Cunningham, Interrogating theoretical models of neural computation with emergent property inference, Elife 10 (2021) e56265.

[68] L. Hertäg, C. Clopath, Prediction-error neurons in circuits with multiple neuron types: Formation, refinement, and functional implications, Proceedings of the National Academy of Sciences 119 (13) (2022) e2115699119.

[69] E. Marder, J.-M. Goaillard, Variability, compensation and homeostasis in neuron and network function, Nature Reviews Neuroscience 7 (7) (2006) 563–574.

[70] E. Marder, A. L. Taylor, Multiple models to capture the variability in biological neurons and networks, Nature neuroscience 14 (2) (2011) 133–138.

[71] M. G. Mattar, M. Lengyel, Planning in the brain, Neuron 110 (6) (2022) 914–934. doi:https://doi.org/10.1016/j.neuron.2021.12.018.
URL https://www.sciencedirect.com/science/article/pii/S0896627321010357

[72] J. L. S. Bellmund, I. Polti, C. F. Doeller, Sequence Memory in the Hippocampal–Entorhinal Region, Journal of Cognitive Neuroscience 32 (11) (2020) 2056–2070. arXiv:https://direct.mit.edu/jocn/article-pdf/32/11/2056/1862288/jocn_a_01592.pdf, doi:10.1162/jocn_a_01592.
URL https://doi.org/10.1162/jocn_a_01592

[73] C. Heckman, C. Mottram, K. Quinlan, R. Theiss, J. Schuster, Motoneuron excitability: the importance of neuromodulatory inputs, Clinical Neurophysiology 120 (12) (2009) 2040–2054.

[74] F. Nadim, D. Bucher, Neuromodulation of neurons and synapses, Current Opinion in Neurobiology 29 (2014) 48–56, sI: Neuromodulation. doi:https://doi.org/10.1016/j.conb.2014.05.003.
URL https://www.sciencedirect.com/science/article/pii/S0959438814001044

[75] V. Pawlak, J. R. Wickens, A. Kirkwood, J. N. Kerr, Timing is not everything: neuromodulation opens the stdp gate, Frontiers in synaptic neuroscience 2 (2010) 146.

[76] Z. Brzosko, S. B. Mierau, O. Paulsen, Neuromodulation of spike-timing-dependent plasticity: past, present, and future, Neuron 103 (4) (2019) 563–581.

[77] M. E. Hasselmo, Neuromodulation: acetylcholine and memory consolidation, Trends in cognitive sciences 3 (9) (1999) 351–359.

[78] M. E. Hasselmo, The role of acetylcholine in learning and memory, Current opinion in neurobiology 16 (6) (2006) 710–715.

[79] J. Widloski, D. J. Foster, Flexible rerouting of hippocampal replay sequences around changing barriers in the absence of global place field remapping, Neuron (2022).

[80] É. Duvelle, R. M. Grieves, A. Liu, S. Jedidi-Ayoub, J. Holeniewska, A. Harris, N. Nyberg, F. Donnarumma, J. M. Lefort, K. J. Jeffery, et al., Hippocampal place cells encode global location but not connectivity in a complex space, Current Biology 31 (6) (2021) 1221–1233.

[81] N. D. Goodman, J. B. Tenenbaum, T. P. Contributors, Probabilistic Models of Cognition, `http://probmods.org/v2`, accessed: 2022-5-3 (2016).

[82] T. Bonnen, D. L. Yamins, A. D. Wagner, When the ventral visual stream is not enough: A deep learning account of medial temporal lobe involvement in perception, Neuron 109 (17) (2021) 2755–2766.

[83] R. Das, J. B. Tenenbaum, A. Solar-Lezama, Z. Tavares, Autumnsynth: Synthesis of reactive programs with structured latent state, in: Advances in Programming Languages and Neurosymbolic Systems Workshop, 2021.

[84] D. J. Foster, M. A. Wilson, Reverse replay of behavioural sequences in hippocampal place cells during the awake state, Nature 440 (7084) (2006) 680–683.

[85] A. C. Singer, M. F. Carr, M. P. Karlsson, L. M. Frank, Hippocampal swr activity predicts correct decisions during the initial learning of an alternation task, Neuron 77 (6) (2013) 1163–1173.

[86] I. Momennejad, A. R. Otto, N. D. Daw, K. A. Norman, Offline replay supports planning in human reinforcement learning, Elife 7 (2018) e32548.

[87] T. Evans, N. Burgess, Replay as structural inference in the hippocampal-entorhinal system, BioRxiv (2020).

[88] K. Kay, J. E. Chung, M. Sosa, J. S. Schor, M. P. Karlsson, M. C. Larkin, D. F. Liu, L. M. Frank, Constant sub-second cycling between representations of possible futures in the hippocampus, Cell 180 (3) (2020) 552–567.

[89] J. Brea, A. T. Gaál, R. Urbanczik, W. Senn, Prospective coding by spiking neurons, PLoS computational biology 12 (6) (2016) e1005003.

[90] W. de Cothi, C. Barry, Neurobiological successor features for spatial navigation, Hippocampus 30 (12) (2020) 1347–1355.

[91] J. Bono, S. Zannone, V. Pedrosa, C. Clopath, Learning predictive cognitive maps with spiking neurons during behaviour and replays., bioRxiv (2021).

[92] H. Lee, Toward the biological model of the hippocampus as the successor representation agent, Biosystems 213 (2022) 104612.

[93] T. M. George, W. de Cothi, K. Stachenfeld, C. Barry, Rapid learning of predictive maps with stdp and theta phase precession, bioRxiv (2022). arXiv:https://www.biorxiv.org/content/early/2022/04/21/2022.04.20.488882.full.pdf, doi:10.1101/2022.04.20.488882.
URL `https://www.biorxiv.org/content/early/2022/04/21/2022.04.20.488882`

[94] J. D. Dworkin, K. A. Linn, E. G. Teich, P. Zurn, R. T. Shinohara, D. S. Bassett, The extent and drivers of gender imbalance in neuroscience reference lists, Nature neuroscience 23 (8) (2020) 918–926.

[95] P. Zurn, D. S. Bassett, N. C. Rust, The citation diversity statement: a practice of transparency, a way of life, Trends in Cognitive Sciences 24 (9) (2020) 669–672.

[96] N. Hopkins, A study on the status of women faculty in science at mit, in: AIP Conference Proceedings, Vol. 628, American Institute of Physics, 2002, pp. 103–106.

[97] E. National Academies of Sciences, Medicine, et al., Sexual harassment of women: climate, culture, and consequences in academic sciences, engineering, and medicine (2018).

[98] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, 2018.

[99] A. Kumar, K. Bouchard, Non-normality in neural networks, in: B. Jalali, K. ichi Kitayama (Eds.), AI and Optical Data Sciences III, Vol. 12019, International Society for Optics and Photonics, SPIE, 2022, pp. 70 – 76. doi:10.1117/12.2613472.
URL https://doi.org/10.1117/12.2613472

[100] B. K. Murphy, K. D. Miller, Balanced amplification: a new mechanism of selective amplification of neural activity patterns, Neuron 61 (4) (2009) 635–648.

[101] M. S. Goldman, Memory without feedback in a neural network, Neuron 61 (4) (2009) 621–634.

Supplementary Notes

The successor representation is defined as

$$M = (I - \gamma T)^{-1} \tag{S1}$$

where T is the transition probability matrix such that $T_{ji} = P(s' = i|s = j)$ for current state s and future state s'

Supplementary Notes 1. Finding the conditions to retrieve M from RNN steady-state activity

For an RNN with connectivity J, activity $\boldsymbol{x}$, input $\boldsymbol{\phi}$, and gain $\gamma \in [0, 1)$, the (linear) discrete-time dynamics equation is [41]

$$\Delta \boldsymbol{x} = -\boldsymbol{x}(t) + \gamma J \boldsymbol{x}(t) + \boldsymbol{\phi}(t). \tag{S2}$$

Furthermore, the steady state solution can be found by setting $\Delta \boldsymbol{x} = 0$:

$$\boldsymbol{x}_{SS} = (I - \gamma J)^{-1} \boldsymbol{\phi} \tag{S3}$$

Assume that $J = T^T$ as a result of the network using some STDP-like learning rule where pre-post connections are potentiated. The transposition is due to notational differences from the RL literature, where the ijth index typically concerns the direction from state i to state j. This is a result of differences in RL and RNN conventions in which inputs are left-multiplied and right-multiplied, respectively. Let γ be a neuromodulatory factor that is applied over the whole network (and, thus, does not need to be encoded in the synaptic weights). Then, the equivalence to equation S1 becomes clear and our steady state solution can be written as:

$$\boldsymbol{x}_{SS} = M^T \boldsymbol{\phi} \tag{S4}$$

This is consistent with the successor representation framework shown in Stachenfeld, et al. [18], where the columns of the M matrix represent the firing fields of a neuron, and the rows of the M matrix represent the network response to some input.

Supplementary Notes 2. Deriving the RNN-S learning rule from TD Error and showing the learning rule is valid under a stability condition

Transitions between states (s, s') are observed as features $(\phi(s), \phi(s'))$ where ϕ is some function. For notational simplicity, we will write these observed feature transitions as $(\boldsymbol{\phi}, \boldsymbol{\phi}')$. A dataset $\mathcal{D}$ is comprised of these observed feature transitions over a behavioral trajectory. Successor features are typically learned by some function approximator $\psi(\boldsymbol{\phi}; \theta)$ that is parameterized by θ and takes in the inputs $\boldsymbol{\phi}$. The SF approximator, ψ, is learned by minimizing the temporal difference (TD) loss function [98]:

$$L(\theta) = \mathbb{E}\left[\|\boldsymbol{\phi} + \gamma\psi^{\pi}(\boldsymbol{\phi}'; \theta) - \psi(\boldsymbol{\phi}; \theta)\|_2^2 \,|\mathcal{D}\right] \tag{S5}$$

for the current policy π. Here, the TD target is $\phi + \gamma\psi^{\pi}(\boldsymbol{\phi}'; \theta)$. Analogous to the model-free setting where the value function V is being learned, $\boldsymbol{\phi}$ is in place of the reward r. Following these definitions, we can view the RNN-S as the function approximator ψ:

$$\psi(\boldsymbol{\phi}; \theta = J) = (I - \gamma J)^{-1}\boldsymbol{\phi} \tag{S6}$$

For a single transition $(\boldsymbol{\phi}, \boldsymbol{\phi}')$ we can write out the loss as follows:

$$L(\theta) = \left\|\boldsymbol{\phi} + \gamma\psi^{\pi}(\boldsymbol{\phi}'; \theta) - (I - \gamma J)^{-1}\boldsymbol{\phi}\right\|_2^2 \tag{S7}$$

For each observed transition, we would like to update ψ such that the loss L is minimized. Thus, we take the gradient of this temporal difference loss function with respect to our parameter $\theta = J$:

$$\nabla_J L(\theta) = 2\Big(\boldsymbol{\phi} + \gamma\psi^{\pi}(\boldsymbol{\phi}'; \theta) - (I - \gamma J)^{-1}\boldsymbol{\phi}\Big)\;\; \nabla_J\Big(-(I - \gamma J)^{-1}\boldsymbol{\phi}\Big)^{\intercal} \tag{S8}$$

We can make the TD approximation $\psi^{\pi}(\boldsymbol{\phi}'; \theta) \approx \psi(\boldsymbol{\phi}'; \theta) = (I - \gamma J)^{-1}\boldsymbol{\phi}'$ [98]:

$$\nabla_J L(\theta) = 2\Big(\boldsymbol{\phi} + \gamma(I - \gamma J)^{-1}\boldsymbol{\phi}' - (I - \gamma J)^{-1}\boldsymbol{\phi}\Big)\;\; \nabla_J\Big(-(I - \gamma J)^{-1}\boldsymbol{\phi}\Big)^{\intercal} \tag{S9}$$

$$= 2\Big(\boldsymbol{\phi} + \gamma(I-\gamma J)^{-1}\boldsymbol{\phi}' - (I-\gamma J)^{-1}\boldsymbol{\phi}\Big)\Big(- (I-\gamma J)^{-1}(-\gamma)(I-\gamma J)^{-1}\boldsymbol{\phi}\Big)^{\intercal} \quad \text{(S10)}$$

$$= -2\Big((I-\gamma J)\boldsymbol{x} + \gamma\boldsymbol{x}' - \boldsymbol{x}\Big)\Big(\gamma(I-\gamma J)^{-1}\boldsymbol{x}\Big)^{\intercal} \quad \text{(S11)}$$

$$= -2\gamma^2\Big(\boldsymbol{x}' - J\boldsymbol{x}\Big)\Big((I-\gamma J)^{-1}\boldsymbol{x})\Big)^{\intercal} \quad \text{(S12)}$$

$$= -2\gamma^2(\boldsymbol{x}' - J\boldsymbol{x})\boldsymbol{x}^{\intercal}(I-\gamma J)^{-\intercal} \quad \text{(S13)}$$

While $-\nabla_J L(\theta)$ gives the direction of steepest descent in the loss, we will consider a linear transformation of the gradient that allows for a simpler update rule. This simpler update rule will be more amenable to a biologically plausible learning rule. We define this modified gradient as $D = \nabla_J L(\theta) M$ where $M = (I-\gamma J)^{\intercal}$. We must first understand the condition for D to be in a direction of descent:

$$\langle D, \nabla_J L\rangle > 0 \quad \text{(S14)}$$

$$\mathrm{Tr}(D^{\intercal}\nabla_J L) > 0 \quad \text{(S15)}$$

$$\mathrm{Tr}(\nabla_J L M \nabla_J L) > 0 \quad \text{(S16)}$$

$$\mathrm{Tr}(\nabla_J L(\frac{M+M^{\intercal}}{2} + \frac{M-M^{\intercal}}{2})\nabla_J L) > 0 \quad \text{(S17)}$$

$$\frac{1}{2}\mathrm{Tr}(\nabla_J L(M+M^{\intercal})\nabla_J L) > 0 \quad \text{(S18)}$$

This expression is satisfied if $M + M^{\intercal}$ is positive definite (its eigenvalues are positive). Thus, we find that our modified gradient points towards a descent direction if the eigenvalues of $M + M^{\intercal}$ are positive. Interestingly, this condition is equivalent to stating that the recurrent network dynamics are stable and do not exhibit non-normal amplification [99, 100, 101]. **In other words, as long as the network dynamics are in a stable regime and do not have non-normal amplification, our modified gradient reduces the temporal difference loss.** Otherwise, the gradient will not point towards a descent direction.

We will use the modified gradient $-D = (\boldsymbol{x}' - J\boldsymbol{x})\boldsymbol{x}^{\intercal}$ as our synaptic weight update rule. Our theoretical analysis explains much of the results seen in the main text. As the gain parameter γ_B is increased, the network is closer to the edge of stability (the eigenvalues of M are close to positive values, Figure 3a). Stability itself is not enough to guarantee that our update rule is valid. We need the additional constraint that non-normal amplification

should not be present (eigenvalues of $M + M^\intercal$ are positive). In practice, however, this does not seem to be a mode that affects our network. That is, the γ_B value for which the error in the network increases coincides with the γ_B value for which the network is no longer stable (Figure 3b). Our theoretical analysis also shows that the gain γ_B can always be decreased such that the eigenvalues of $M + M^\intercal$ are positive and our update rule is valid (Figure 3e). At the most extreme, one can set $\gamma_B = 0$ during learning to maintain stability (as we do in Figure 4 and onwards).

Supplementary Notes 3. Proving the RNN-S update rule calculated on firing rates ($\boldsymbol{x}$) depends only on feedforward inputs ($\boldsymbol{\phi}$) at steady state

We will show that our update rule, which uses $\boldsymbol{x}$ (neural activity), converges on a solution that depends only on $\boldsymbol{\phi}$ (the feedforward inputs). We will also show that in the one-hot case, we learn the SR exactly.

As a reminder, our learning rule for each $j \to i$ synapse is:

$$\Delta J = \eta(\boldsymbol{x}' - J\boldsymbol{x})\boldsymbol{x}^\intercal \tag{S19}$$

We can solve for the steady state solution of equation S19 (set $\Delta J = 0$). Let $A = (1 - \gamma J)^{-1}$ for notational convenience, and recall that in steady state $x = A\boldsymbol{\phi}$. Let $\langle \boldsymbol{x} \rangle$ denote the average of $\boldsymbol{x}$ over time.

$$J = \langle \boldsymbol{x}'\boldsymbol{x}^\intercal \rangle \langle \boldsymbol{x}\boldsymbol{x}^\intercal \rangle^{-1} \tag{S20}$$

$$J = \langle A\boldsymbol{\phi}'(A\boldsymbol{\phi})^\intercal \rangle \langle A\boldsymbol{\phi}(A\boldsymbol{\phi})^\intercal \rangle^{-1} \tag{S21}$$

$$J = \langle A\boldsymbol{\phi}'\boldsymbol{\phi}^\intercal A^\intercal \rangle \langle A\boldsymbol{\phi}\boldsymbol{\phi}^\intercal A^\intercal \rangle^{-1} \tag{S22}$$

$$J = A\langle \boldsymbol{\phi}'\boldsymbol{\phi}^\intercal \rangle A^\intercal (A\langle \boldsymbol{\phi}\boldsymbol{\phi}^\intercal \rangle A^\intercal)^{-1} \tag{S23}$$

Note that, since $A = (1 - \gamma J)^{-1}$, $J = \frac{1}{\gamma}(1 - A^{-1})$.

$$A\langle \boldsymbol{\phi}'\boldsymbol{\phi}^\intercal \rangle A^\intercal = \tfrac{1}{\gamma}(1 - A^{-1})A\langle \boldsymbol{\phi}\boldsymbol{\phi}^\intercal \rangle A^\intercal \tag{S24}$$

$$A\langle \boldsymbol{\phi}'\boldsymbol{\phi}^\intercal\rangle A^\intercal = \tfrac{1}{\gamma}(A\langle \boldsymbol{\phi}\boldsymbol{\phi}^\intercal\rangle A^\intercal - \langle \boldsymbol{\phi}\boldsymbol{\phi}^\intercal\rangle A^\intercal) \tag{S25}$$

$$A\langle \boldsymbol{\phi}'\boldsymbol{\phi}^\intercal\rangle = \tfrac{1}{\gamma}(A\langle \boldsymbol{\phi}\boldsymbol{\phi}^\intercal\rangle - \langle \boldsymbol{\phi}\boldsymbol{\phi}^\intercal\rangle) \tag{S26}$$

Thus,

$$\langle \boldsymbol{\phi}'\boldsymbol{\phi}^\intercal\rangle = \tfrac{1}{\gamma}(1 - A^{-1})\langle \boldsymbol{\phi}\boldsymbol{\phi}^\intercal\rangle \tag{S27}$$

Therefore,

$$J = \langle \boldsymbol{\phi}'\boldsymbol{\phi}^\intercal\rangle\langle \boldsymbol{\phi}\boldsymbol{\phi}^\intercal\rangle^{-1} \tag{S28}$$

$$J = R_{\boldsymbol{\phi}\boldsymbol{\phi}}(-1)R_{\boldsymbol{\phi}\boldsymbol{\phi}}(0)^{-1} \tag{S29}$$

where $R_{\boldsymbol{\phi}\boldsymbol{\phi}}(\tau)$ is the autocorrelation matrix for some time lag τ. Therefore, the RNN-S weight matrix J at steady state is only dependent on the inputs into the RNN over time.

In the case where $\boldsymbol{\phi}$ is one-hot, we compute the SR exactly. This is because the steady state solution at each $j \to i$ synapse simplifies into the following expression:

$$J_{ij} = \frac{\sum_{t'} \phi_j(t'-1)\phi_i(t')}{\sum_{t'} \phi_j(t')} \tag{S30}$$

This is the definition of the transition probability matrix and we see that $J = T^\intercal$. Note that the solution for J_{ij} in equation S30 is undefined if state j is never visited. We assume each relevant state is visited at least once here.

Supplementary Notes 4. Deriving the adaptive learning rate update rule

This section explains how the adaptive learning rate is derived. The logic will be similar to calculating a weighted running average. Let $d_{ij}(t)$ be a binary function that is 1 if the transition from timestep $t-1$ to timestep t is state j to state i. Otherwise, it is 0. Assume $\boldsymbol{\phi}$ is one-hot encoded. Notice that in the one-hot case, the RNN-S update rule (equation 4) simplifies to:

$$\Delta J_{ij} \approx \eta(d_{ij} - J_{ij}x_j) \tag{S31}$$

What η should be used so J approaches $T^\intercal$ as quickly as possible? During learning, the empirical transition matrix, $T(t)$, changes at each timestep t, based on transitions the animal has experienced. Define the total number of times that state ϕ_j happened prior to time t as $n_j(t) = \sum_{t'=1}^{\infty} \phi_j(t-t')$, and define the running count of transitions from state j to state i as $c_{ij}(t) = \sum_{t'=1}^{\infty} d_{ij}(t-t')$. We want $J(t) = T^\intercal(t)$, which necessitates

$$\Delta J_{ij}(t) = T_{ji}(t) - T_{ji}(t-1) = \frac{c_{ij}(t)}{n_j(t)} - \frac{c_{ij}(t-1)}{n_j(t-1)} \tag{S32}$$

$$= \frac{n_j(t-1)c_{ij}(t) - c_{ij}(t-1)n_j(t)}{n_j(t)n_j(t-1)} \tag{S33}$$

Note that $n_j(t) = n_j(t-1) + \phi_j(t-1)$, and $c_{ij}(t) = c_{ij}(t-1) + d_{ij}(t)$, which gives us

$$\Delta J_{ij}(t) = \frac{n_j(t-1)c_{ij}(t-1) + n_j(t-1)d_{ij}(t) - c_{ij}(t-1)n_j(t-1) - c_{ij}(t-1)\phi_j(t-1)}{n_j(t)n_j(t-1)} \tag{S34}$$

$$= \frac{n_j(t-1)d_{ij}(t) - c_{ij}(t-1)\phi_j(t-1)}{n_j(t)n_j(t-1)} \tag{S35}$$

$$= \frac{1}{n_j(t)}\left(d_{ij}(t) - \frac{c_{ij}(t-1)\phi_j(t-1)}{n_j(t-1)}\right) \tag{S36}$$

$$= \frac{1}{n_j(t)}\left(d_{ij}(t) - T_{ji}\phi_j(t-1)\right) \tag{S37}$$

Therefore, comparing with equation S31, we can see that a learning rate $\eta_j = \frac{1}{n_j(t)}$ will let $J = T^\intercal$ as quickly as possible. We have defined n in terms of the inputs $\boldsymbol{\phi}$ for this derivation, but in practice the adaptive learning rate as a function of $\boldsymbol{x}$ works well with the RNN-S update rule (which is also a function of $\boldsymbol{x}$). Thus, we use the adaptive learning rate defined over $\boldsymbol{x}$ in our combined learning rule for increased biological plausibility.

In its current form, the update equation assumes transitions across all history of inputs are integrated. In reality, there is likely some kind of memory decay. This can be implemented with a decay term $\lambda \in (0, 1]$:

$$n_j(t) = \sum_{t'=1}^{\infty} \lambda^{t'} x_j(t-t') \tag{S38}$$

λ determines the recency bias over the observed transitions that make up the T estimate.

The addition of λ has the added benefit that it naturally provides a mechanism for learning rates to modulate over time. If $\lambda = 1$, the learning rate can only monotonically decrease. If $\lambda < 1$, the learning rate can become strong again over time if a state has not been visited in a while. This provides a mechanism for fast learning of new associations, which is useful for a variety of effects, including remapping.

Supplementary Notes 5. Endotaxis model and the successor representation

The learning rule and architecture of our model is similar to a hypothesized "endotaxis" model [45]. In the endotaxis model, neurons fire most strongly near a reward, allowing the animal to navigate up a gradient of neural activity akin to navigating up an odor gradient. The endotaxis model discovers the structure of an environment and can solve many tasks such as spatial navigation and abstract puzzles. We were interested in similarities between RNN-S and the learning rules for endotaxis, in support of the idea that SR-like representations may be used by the brain for a broad range of intelligent behaviors. Here, we outline similarities and differences between the two model architectures.

The endotaxis paper [45] uses Oja's rule in an RNN with place-like inputs. The SR can also be learned with an Oja-like learning rule. Oja's rule is typically written as [52]:

$$\Delta J_{ij} = \eta x_j x_i - \eta J_{ij} x_i^2 \tag{S39}$$

If we assume that there is a temporal asymmetry to the potentiation term (e.g., potentiation is more STDP-like than Hebbian), then we have

$$\Delta J_{ij} = \eta x_j(t-1) x_i(t) - \eta J_{ij} x_i(t)^2 \tag{S40}$$

We then solve for the steady state solution of this equation, when $\Delta J_{ij} = 0$:

$$0 = \eta \langle x_j(t-1) x_i(t) \rangle - \eta J_{ij} \langle x_i(t)^2 \rangle \tag{S41}$$

$$J_{ij} = \frac{\langle x_j(t-1) x_i(t) \rangle}{\langle x_i(t)^2 \rangle} \tag{S42}$$

$$J_{ij} = \frac{\sum_{t'} x_j(t'-1)x_i(t')}{\sum_{t'} x_i(t')^2} \tag{S43}$$

where $\langle \cdot \rangle$ indicates the time-average of some term. Assume that the plasticity rule does not use $\boldsymbol{x}$ exactly, but instead uses $\boldsymbol{\phi}$ directly. Given that inputs are one-hot encodings of the animal's state at some time t, the expression becomes

$$J_{ij} = \frac{\sum_{t'} \phi_j(t'-1)\phi_i(t')}{\sum_{t'} \phi_i(t')} \tag{S44}$$

If we assume T is symmetric, $J = T^\intercal$. Alternatively, if we use pre-synaptic normalization as opposed to the standard post-synaptic normalization of Oja's rule (i.e., index j instead of i in the denominator), we also have $J = T^\intercal$. Thus, the steady state activity of a RNN with this learning rule retrieves the SR, as shown in Supplementary Notes 1.

Supplementary Notes 6. Independent normalization and successor features

If we assume the same Oja-like rule as in Supplementary Notes 5, we can also arrive at a similar interpretation in the successor feature case as in equation 7. By solving for the steady state solution without any assumptions about the inputs $\boldsymbol{\phi}$, we get the following equation:

$$J = R_{\phi\phi}(-1)\texttt{diag}(R_{\phi\phi}(0))^{-1} \tag{S45}$$

where `diag` is a function that retains only the diagonal of the matrix. This expression provides a useful way to contrast the learning rule used in RNN-S with an Oja-like alternative. While RNN-S normalizes by the full autocorrelation matrix, an Oja-like rule only normalizes by the diagonal of the matrix. This is the basis of our independent normalization model in Figure 4bc.

Supplementary Figures

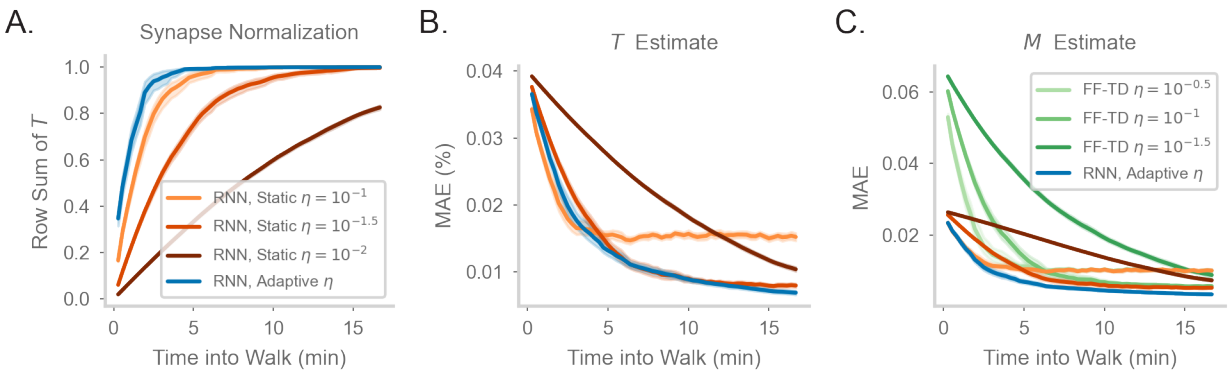

Figure S2: **Comparing model performance in different random walks. a-c.** As in Figure 2d-f of the main document, but for a walk with uniform action probabilities.

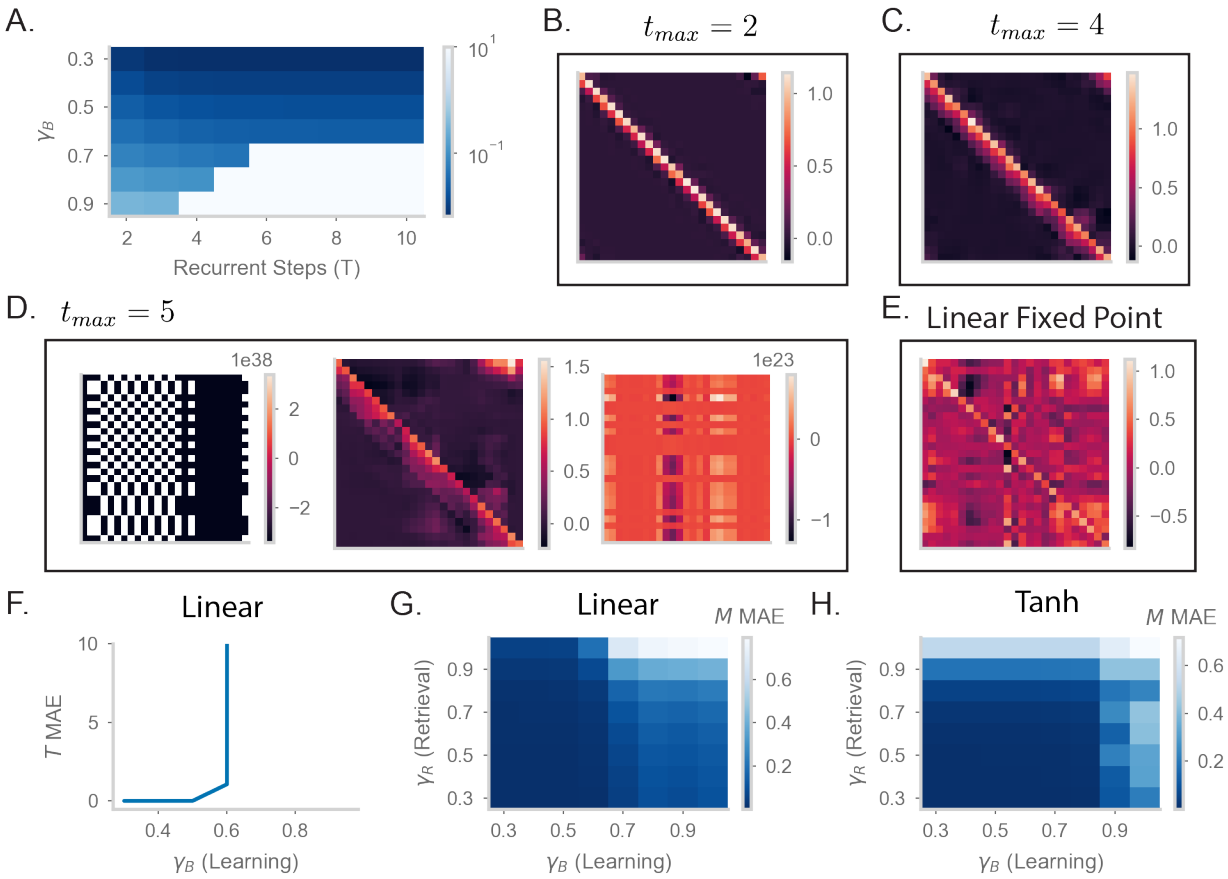

Figure S3: **Understanding the effects of recurrency on stability. a.** Mean absolute error (MAE) of M matrices learned by RNN-S with different baseline γ and different numbers of recurrent steps in dynamics. Test datasets used various biases in action probability selection. Errors are max-clipped at 10^1 for visualization purposes. **b.** M matrix learned by RNN-S with two recurrent steps in dynamics and baseline $\gamma = 0.8$. A forward-biased walk on a circular track was simulated. **c.** As in (b), but for four recurrent steps. **d.** As in (b), but for five recurrent steps. Three examples are shown from different sampled walks to highlight the runaway activity of the network. **e.** As in (b) but for the RNN-S activity calculated as $(I - \gamma J)^{-1}$. Note that this calculation amounts to an unstable fixed point in the dynamics that cannot be reached when the network is in an unstable regime. **f.** Mean absolute error (MAE) in T made by RNN-S with linear dynamics using γ_B during learning. **g.** MAE in M for γ_R made by RNN-S with linear dynamics using γ_B during learning. **h.** As in (g), but the dynamics now have a tanh nonlinearity.

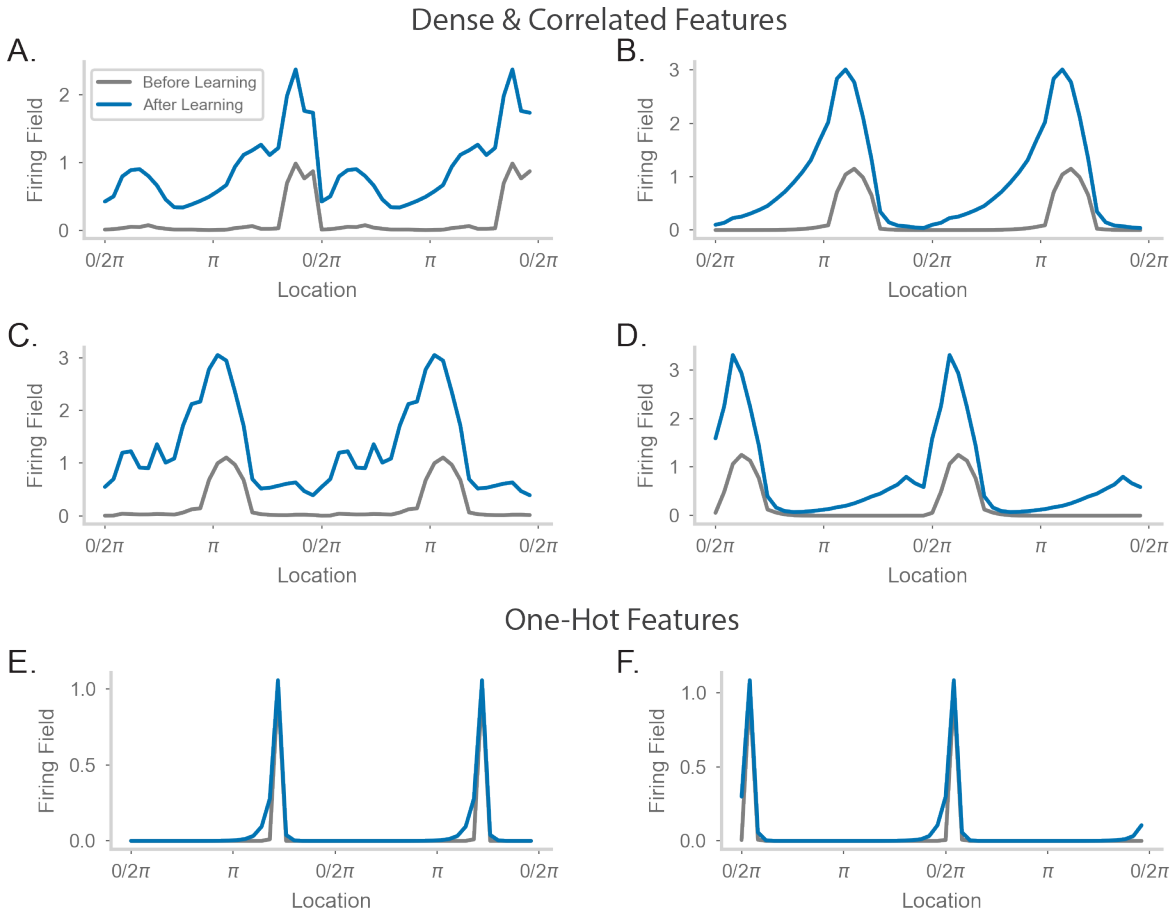

Figure S4: **Comparing place field shift and skew effects for different feature encodings. a-d.** Average firing rate as a function of position on a circular track for four example neurons. The walk and feature encodings were generated as in Figure 4d of the main text. Each neuron is sampled from a different walk. "Before Learning" refers to firing fields made from the first 2 minute window of the walk. "After Learning" refers to firing fields made from the entire walk. **e-f.** As in (a-d), but for two neurons from a walk where the features were one-hot encoded.

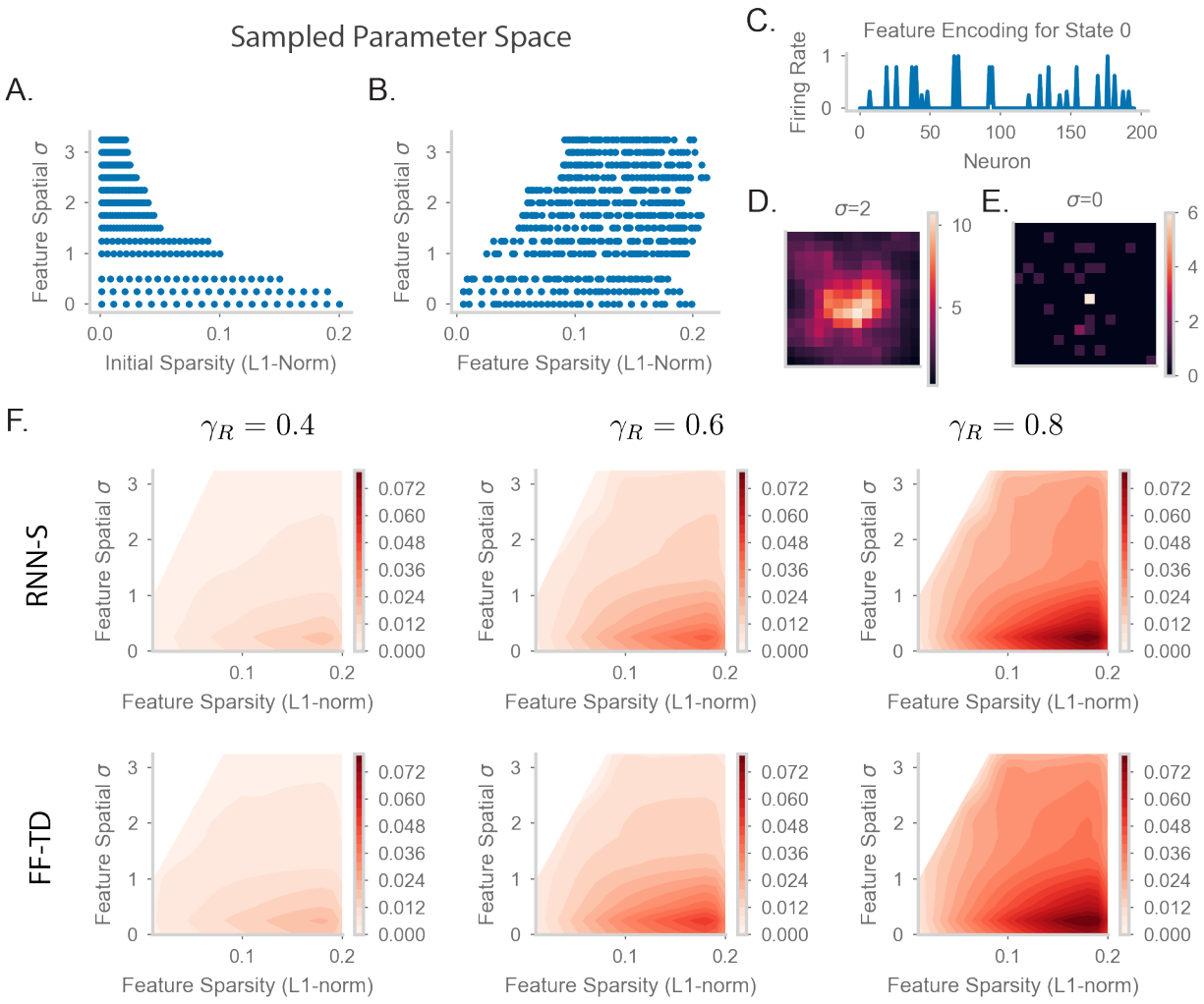

Figure S5: **Parameter sweep details and extended TD error plots. a.** The values of p (initial sparsity of random vectors before spatial smoothing) and σ sampled in our parameter sweep for Figures 5-6 in the main text. See methods 4.10 for more details of how feature encodings were generated. **b.** The values of s (final sparsity of features, measured after spatial smoothing) and σ sampled in our parameter sweep for Figures 5-6 in the main text. **c.** A sample state encoded by the firing rate of 200 input neurons. Here, $s = 0.11$ and $\sigma = 2.$. **d.** Spatial correlation of the feature encoding for an example state with the features of all other states. The 14×14 states are laid out in their position in the 2D arena. Here, the sample state is the state in the center of the 2D arena and $\sigma = 2.0$. **e.** As in (d), but for $\sigma = 0.0$. **f.** As in Figure 5d of the main text, but for RNN-S (first row) and FF-TD (second row) with $\gamma_R = 0.4$ (left column), $\gamma_R = 0.6$ (middle column), and $\gamma_R = 0.8$ (right column).

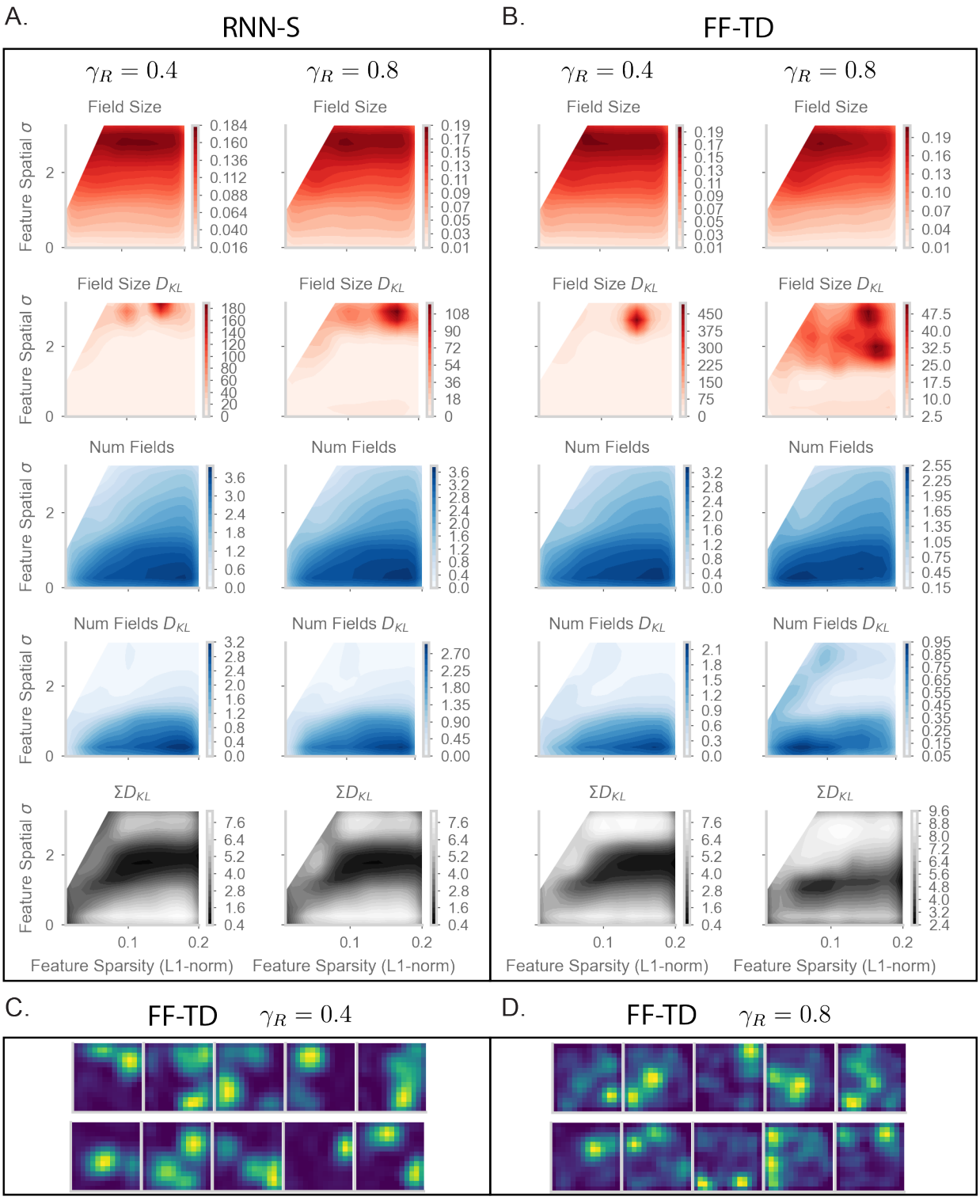

Figure S6: **Extended place field evaluation plots. a.** As in Figures 6e-g of the main text, but for $\gamma_R = 0.4$ (left column) and $\gamma_R = 0.8$ (right column). In addition, the plots showing KL divergence (in bits) for the distribution of field sizes and number of fields per cell are shown. **b.** As in (a) but for FF-TD. **c.** A in Figure 6h of the main text, but for FF-TD with $\gamma_R = 0.4$ and **d.** FF-TD with $\gamma_R = 0.8$